AN INVESTIGATION OF MACHINE LEARNING TECHNIQUES FOR AN IMPROVED INTRUSION DETECTION SYSTEM FOR THE INTERNET OF THINGS

By
Hassan Adegbola Afolabi
219048801

Supervised by:
Dr. Abdurazzag Ali Aburas

A Thesis Submitted in Fulfillment of the Academic Requirements for the Degree of Doctor of Philosophy
in the

School of Engineering
College of Agriculture, Engineering, and Sciences
University of Kwazulu-Natal

Durban South Africa

December, 2022

Examiner's Copy

DECLARATION

The research studies described in this thesis were performed at the University of KwaZulu-Natal under Dr. Abdurazzag Aburas. I hereby declare that I am the sole author of this thesis, which is a presentation of my original work. It has never before been submitted in part or in whole to a university for a degree, whether it be this one or another. All sources are cited as references.

Hassan Adegbola Afolabi
Full Name

Signature

Date: 11/21/2022

Approved for final submission

Supervisor:

Signature

Dr. Abdurazzag Ali Aburas
Full Name

Date: 31/01/2023

DECLARATION 2 - PUBLICATIONS FROM THIS THESIS

I, Hassan Afolabi, declare that the following are contributions from this thesis that have already been published and/or are being considered for publication elsewhere:

1. H.A Afolabi and A.A Aburas, Comparison of Single and Ensemble Intrusion Detection Techniques using Multiple Datasets; *International Journal of Advanced Trends in Computer Science and Engineering, 10(4), July – August 2021, 2752 – 2761. https://doi.org/10.30534/ijatcse/2021/161042021* **[PUBLISHED]**

2. H.A Afolabi and A.A Aburas, RTL-DL: A Hybrid Deep Learning Framework For DDoS Attack Detection In Big Data Environment; *International Journal of Computer Networks & Communications (IJCNC) Vol.14, No.6, November 2022, pp 51-66.* DOI: 10.5121/ijcnc.2022.14604 **[PUBLISHED]**

3. H.A Afolabi and A.A Aburas., Statistical Performance Assessment Of Supervised ML Algorithms For The Design Of Anomaly Intrusion Detection System In Internet Of Things; *International Journal of Artificial Intelligence. IJ-AI.* **[ACCEPTED FOR PUBLICATION] Article ID #: 22511**

4. H.A Afolabi and A.A Aburas, Deep Stacking of Boosted Machines: An Intrusion Detection Technique for Internet of Things**;** *International Journal of Computer Network and Information Security IJCNIS.* **[ACCEPTED FOR PUBLICATION] Article ID #: PAPER007761**

5. H.A Afolabi and A.A Aburas, "A Review of Machine Learning Based Intrusion Detection System For The Internet of Things," *in the proceedings of IEEE Africon Conference*, 2023. **[ACCEPTED FOR PUBLICATION] Paper ID #: 47**

Note: Some similarities in this thesis are due to published work of the articles being incorporated into the thesis.

DEDICATIONS

This thesis is dedicated to my wife Teslimat and my daughter Nadia Afolabi

ACKNOWLEDGEMENTS

I would like to begin by expressing my heartfelt appreciation to my academic advisor, Dr. Abdurazzag Ali Aburas, for all of the support and guidance over the course of his supervision.

I owe a great deal of gratitude to the staff at the University of Kwazulu-Natal's Research Office for their numerous efforts on my behalf in aiding my Ph.D. studies. I'm also grateful to the open-source software community and my friends for their support and encouragement.

In closing, I'd like to thank my family for their unwavering support, encouragement, and motivation, all of which helped me get through any challenges I faced while conducting my research.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

AdaBoost: Adaptive Boosting

ADAM: Adaptive Moment Estimation Method

ANN: Artificial Neural Network

AUC: Area under Curve

BP: Back propagation

CICIDS2017: Canadian Institute of Cybersecurity Intrusion detection dataset

CNN: convolutional neural network

DDoS: Distributed Denial of Service Attacks

DoS: Denial of Service

DNN: Deep Neural Network

DNS: Domain Name System

DL: Deep Learning

DT: Decision Tree

DSBM: Deep Stacking of Boosted Machines

ETC: Extra Tree Classifiers

FCNN: Fully Connected Neural Network

FPR: False Positive Rate

FTP: File Transfer Protocol

GB: Gigabytes

GBM: Gradient Boosting Machine

HTTP: Hypertext Transfer Protocol

IDS: Intrusion Detection System

IoT: Internet of Things

IoMT: Internet of Medical Things

IoTID20: Internet of Things Intrusion Detection Dataset 2020

IG: Information Gain

KNN: k-nearest neighbors' algorithm

KDDCUP99: Knowledge Discovery and Data Mining

LSTM: Long short-term memory

LDA: Linear Discriminant Analysis

LGBM: Light Gradient Boosting Machines

MCC: Mathews Correlation Co-efficient

ML: Machine Learning

N-baIoT: Network-Based Anomaly Internet of Things

NB: Naive Bayes

NSL-KDD: Knowledge Discovery and Data Mining

PCAP: Packet Capture

ReLu: Rectified Linear Unit

RF: Random Forest

RNN: Recurrent neural network

ROS: Random Oversampling

ROC: Receiver Operating Characteristic

RTL: Random Oversampling + Tomek-Link

R2L: Root to Local

SGD: Stochastic Gradient Descent

SSH: Secure Shell

TCP: Transmission Control Protocol

TL: Tomek-Link

TPR: True Positive Rate

UDP: User Datagram Protocol

U2R: User to Root

WAN: Wide Area Network

XGBoost: Extreme Gradient Boosting (A scalable tree boosting system)

SMOTE: Synthetic Minority Oversampling Technique

ZB: Zettabyte

ABSTRACT

Internet of things (IoT) threats are difficult to detect because of the enormous variety of devices utilized in the internet of things environment. It is also challenging to design an effective security model for each type of device in the IoT. An intrusion detection system (IDS) is a tool for detecting network threats. Although IDSs have been well studied, it is challenging to realistically estimate their performance when deployed in real life due to the issues inherent to the datasets used to train them. This thesis examined machine learning (ML) techniques for an improved IDS for the IoT. During the investigation, a performance evaluation of some extensively used supervised ML methods was conducted using various benchmark imbalance datasets. Furthermore, a novel framework named random oversampling and tomek-links (RTL) was presented to minimize the effect of data imbalance in IDS datasets. Friedman and Dunn's statistical tests were also conducted to examine the significant differences between classifiers with the primary goal of proposing an appropriate method for selecting diverse base classifiers for a stacking-type ensemble IDS. An intrusion detection model based on stacking ensemble named deep stacking of boosted machines (DSBM) was presented using extreme gradient boosting, light gradient boosting machine and gradient boosted machines as the base classifiers, and a deep neural network model as meta classifier. The proposed model was evaluated using records from all the nine devices in N-baIoT dataset. Several performance parameters such as accuracy, precision, recall, Fscore, the area under receiver operating characteristic curve, precision-recall curve, confusion matrix, matthew's correlation coefficient, and kappa statistics were used to evaluate the proposed model. The transferability abilities of the proposed model were also studied by performing a cross-dataset evaluation using data samples from different devices in the N-BaIoT dataset as train and test sets to ensure that the proposed model can adjust to the inevitable changes in the network traffic generated by IoT. According to the results obtained, the proposed DSBM model is capable of outperforming other ML algorithms in terms of several metrics.

CHAPTER 1: INTRODUCTION

1.1 Background

Internet of things is an emerging technology field that has made big changes to how we work and live [1]. The Internet of Things (IoT) is expected to influence a wide range of aspects of our life in the 21st century due to the integration of billions of smart devices and IoT control systems. Power grids, houses, universities, businesses and transportation systems are only a few examples [2, 3]. Regardless of the multiple benefits offered by IoT-connected devices, implementing the numerous devices utilized in IoT technologies may result in a variety of security concerns. If an unauthorized malicious agent gains access to it, it has the potential to cause considerable damage to our personal security, reputation and wealth [4]. There have been a lot of examples of attackers being able to easily inject malicious code into wearable devices by using programming interfaces to obtain users' private information [5]. Implantable medical equipment has also been targeted, posing a major risk to the patient's life. Recent trends show that industrial and urban infrastructure attacks are also on the rise. However, despite the fact that IoT security remains a major issue, this may be greatly addressed by putting appropriate security procedures in place during the implementation of IoT systems. To ensure that users utilize the full capabilities of IoT and to deal with any potential security vulnerabilities that may occur, efforts should be made to investigate all anticipated security flaws and possible solutions such as intrusion detection systems that can be effectively applied in IoT security.

1.2 IoT Characteristics

Internet of Things is applied in several domains like smart homes, cities, health and telemedicine etc. They differ from typical systems in several ways, which makes them extremely difficult to secure. The characteristics of the IoT are illustrated in the figure 1-1, and a detailed discussion is given in [6-11].

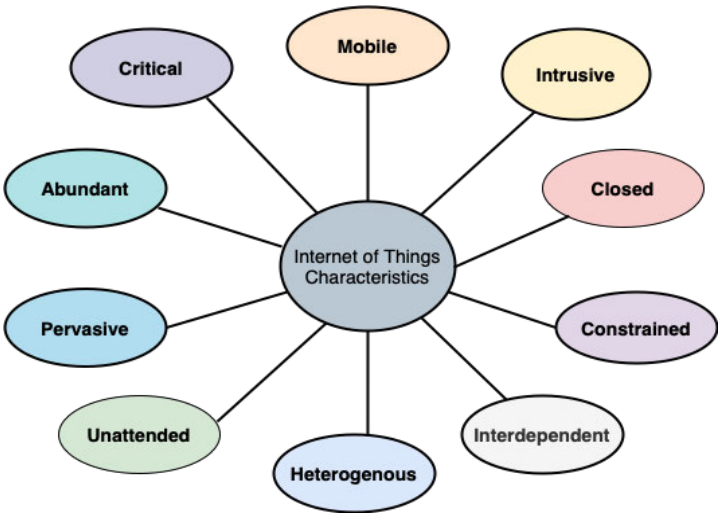


Figure 1-1: Characteristics of IoT

Each of the characteristics are described as follows: As a result of these characteristics, IoT systems are extremely difficult to secure and are a major target for attackers.

i. **Abundant:** IoT devices are increasing at an unprecedented rate, resulting in an enormous volume of data being generated. More than 41 billion IoT connected devices are expected to generate 79.4 ZB of data by 2025 [12]. IoT generated data can rapidly become infected by malware. The malware-infected IoT networks' huge data generating abilities can be exploited to launch various other attacks, such as distributed denial-of-service (DDoS) attacks.

ii. **Closed:** Many IoT devices are closed because they don't let you add additional software after they've been built [10]. Thus, users can't easily add any more security software to the device.

iii. **Constrained:** Several IoT devices, like sensors and medical implants, have extremely limited computing power and memory due to their lightweights. As a result, they are unable to perform complex resource-intensive cryptographic operations. Therefore, they cannot be protected using complex encryption or authentication algorithms. Additionally, the deployment of an antivirus is not possible due to the fact that it requires a significant amount of computational resources.

iv. **Critical:** IoT applications such as medical applications, self-driving cars, and industrial control infrastructures utilized in nuclear power plants are all very critical. Any vulnerability of the devices utilized in any of these applications has the potential to cause serious physical and human harm. For instance, in the future, the likelihood of hackers seizing control of automobiles in order to cause intentional crashes would become a significant security concern [13].

v. **Diversity and Heterogeneous:** The IoT application fields are infinite and diverse in many aspects; they include a wide range of application fields like smart cities, smart grids, smart health, smart homes, smart industries, smart transportation systems, wearables, and so on [2, 3], [9]. In various application fields, the Internet of Things (IoT) is employed in diverse scenarios. For instance, in a smart city context, it encompasses functions like traffic management, air quality control, parking, and garbage collection. The IoT also exhibits diversity in communication protocols (Bluetooth, Wi-Fi, ZigBee, Z-Wave) and hardware platforms (X86, ARM). Due to this diversity, creating a universally applicable security solution is challenging, as security requirements vary depending on application fields and device types.

vi. **Interdependent:** This is a scenario in which a device in an IoT application domain is in charge of the condition of another device. The air conditioner, for example, is turned off when the temperature value on the thermometer exceeds a predefined threshold. This is a security risk because hackers can circumvent a highly protected device's defensive mechanism by compromising a badly setup device that is interdependent with a highly protected device.

vii. **Intrusive:** Sensitive and personal data is often collected by IoT devices. Medical devices, for example, monitor data like the blood pressure, physical activities, heartbeat rate, and patient’s location. Smart-phones application makes the data acquired by these devices available to the user. Therefore, If the smart phone application, the IoT device and the channel of communication between the application and the IoT device are not adequately secure, sensitive personal data can be exposed.

viii. **Mobile:** This describes the movement of IoT devices from network A to network B. Because IoT devices can connect from one network to another and may need to communicate with other devices on that network, they are potential target for malware. This characteristic makes them more likely to be connected to a vulnerable network and may aid in malware spread due to the movement of devices from network A to network B and so on.

ix. **Pervasive:** Internet of Things devices are becoming integrated into every facet of our daily life. An increasing number of our daily activities will be impacted by the Internet of Things (IoT). IoT security risks are often overlooked despite their pervasiveness.

x. **Unattended/Autonomous**: Referred to as an IoT device's long-term unattended status by authors in [6]. Some devices, such as medical implants and smart meters, must operate for extended periods of time without physical access. Due to their physical inaccessibility, monitoring the states of these devices and detecting any compromise is extremely difficult.

1.3 IoT Applications

The IoT is beneficial in a variety of domains such as industrial, urban infrastructure, environmental, healthcare, transportation, and so on. The figure 1-2 shows a few of the IoT application domains.

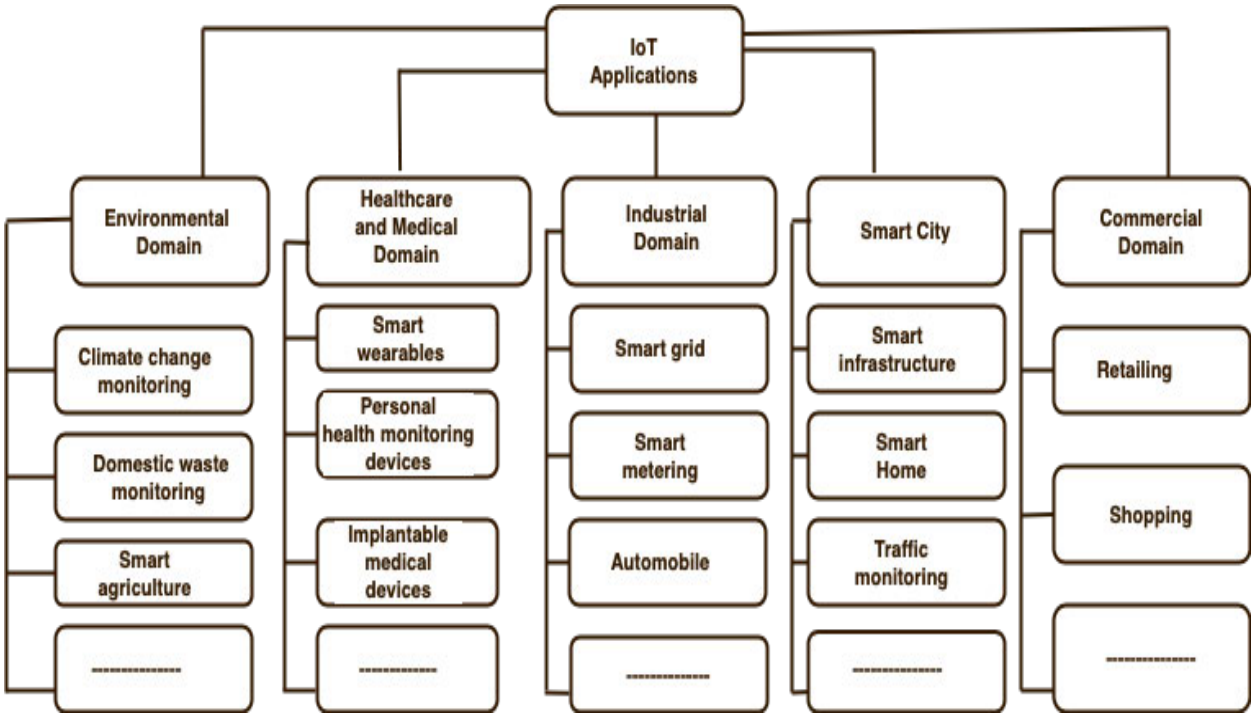


Figure 1-2: Taxonomy of IoT Applications

An urban area that uses technology and data-driven solutions to enhance the quality of life for its residents, improve efficiency, and address various challenges faced by modern cities is a "Smart city". It is among the hottest domains of IoT, covers areas like "smart home," "smart cars," "smart infrastructure," and "smart traffic monitoring." The idea of a "smart city" became more popular in the last decade, and it inspired several research

activities [14]. An example is a "smart home," which is made up of IoT-enabled appliances and electronics that communicate to each other to provide the best comfort, security, and energy efficiency. The global smart home market size was valued at $80.21 billion in 2022 and is projected to grow at a compound annual growth rate (CAGR) of 27.8% from 2023 to 2030 [15]. Healthcare and medical applications are another important area for IoT. This is also known as the Internet of Medical Things (IoMT). IoMT domain connects healthcare services with IT systems to obtain, process, store, transmit, and analyze data more efficiently [16, 17]. Hospitals can monitor and collect vital signs on patients in real time through wearable and implantable devices equipped with Wi-Fi and RFID technologies. An implantable cardioverter defibrillator is an example (ICD). Unfortunately, these devices are susceptible to attack that could endanger the patient's life due to inadequacy of security measures [18, 19]. The Internet of Things has also been applied to industrial sectors like power for smart grid and smart metering, as well as automobiles. The Internet of Things also has applicability in the environmental and commercial sectors.

1.4 Benefits of IoT

IoT makes humans to interact, contribute, and collaborate together with the things around us. This makes IoT beneficial in a number of ways such as:

i. Efficient use of resources
ii. Human effort minimization
iii. Time-saving
iv. Development of AI through IoT
v. Enhanced Security

1.5 Security Vulnerabilities of IoT

Privacy and security are among the IoT's most significant and difficult problems because of a wide range of cyber-attacks, risks, malicious threats, and vulnerabilities [20]. In addition to the above-mentioned characteristics contributing to IoT's security and privacy concerns, this section discusses the most prevalent issues that result in IoT system security vulnerabilities.

i. **Backdoors:** Backdoors are open ports that are not necessary used for services like FTP and SSH [21]. Vendors purposefully insert them for management and testing purposes [22]. Devices in the Internet of Things (IoT) are prone to this vulnerability, which can be found by running port scanning on them. These backdoors can be used by an attacker to take over the device. Attacks such as buffer overflow and DDoS attacks can be launched against other IoT devices from the compromised device.
ii. **Insecure Interfaces** [22, 23]**:** Mobile application and web interfaces that are used to set up IoT devices are prone to cyberattacks like SQL injection which can cause leakage of classified information. This is because most of them are poorly designed and not very secure. Authors in [24] presented a report that some web portals for controlling IoT devices had 10 security vulnerabilities that could allow an intruder to get into the backend systems.

iii. **Inadequate mechanisms for authentication and authorization:** Due to constrained resources, most IoT devices lack sufficient authentication and authorization mechanisms. [9, 10, 21, 23-25]. The authors in [24] stated in their report that the Belkin WeMo connected switch does not ask users for verification before connection. As a result, an intruder on the same network can send any command to the device in order to carry out attacks. Man-in-the-Middle (MitM) attacks are one example.

iv. **Inadequate encryption:** The majority of IoT devices lack an adequate encryption technique to secure data transmitted over a local network [9, 21, 24, 25]. This is a major privacy concern because data transmitted during communication can be eavesdropped on.

v. **Inadequate Physical Security:** Due to the unattended or autonomous characteristics of several IoT devices, attackers can easily get physical access to them. Most IoT devices have ports such as USB, which can be utilised to obtain sensitive information or change the configuration settings of the device [10, 21, 23, 25]. An experienced attacker may be able to read a device's firmware to gain a better understanding of how the device works and search for vulnerabilities. Attackers can also sell used IoT devices that have been intentionally compromised [22].

vi. **Inadequate Software Update Policies:** The majority of discovered vulnerabilities are not thoroughly patched due to insufficient software update policies [21, 26, 27]. While most malware quickly spread in the first few days after it is discovered, researchers in [26] reported a significant time lag between the discovery of the first samples of the malware and the official release of a security solution. Many IoT devices are vulnerable to malicious modifications because their manufacturers lack automated patch-update mechanisms [21]. Other vulnerabilities include software vulnerabilities [7, 21],[27, 28],[29] and weak passwords [26] [30].

1.6 IoT Security Threats

Security vulnerabilities in IoT devices and infrastructure have been highlighted in previous sections; attackers can exploit these flaws to launch coordinated and sophisticated attacks. These attacks can be classified as either external or internal. External attacks occur when devices or nodes from outside the network connect to network nodes in order to launch a malicious attack. On the other hand, internal threats involve compromised devices within the network attempting to launch attacks on other nodes or devices on the network. The following are some of the threats that an IoT infrastructure is susceptible to:

i. ***IoT Botnets:*** Botnets are a set of Internet of Things (IoT) network devices that have been compromised by malware known as bots or zombies in order to perform large-scale and coordinated attacks [31] Researchers in [26, 27, 32-36] conducted extensive research on IoT botnets. The most prevalent botnet kinds are mirai and bashlite and several researches concentrated on these two dues to their similarities to others. Other notable IoT botnets that functions in a different manner include brickerbot [32, 37], blackIoT [38], hajime [32, 39], hidenseek [40] [41], persirai [32, 42]and reaper [43]. The figure 1-3 depicts a diagram of a botnet attack:

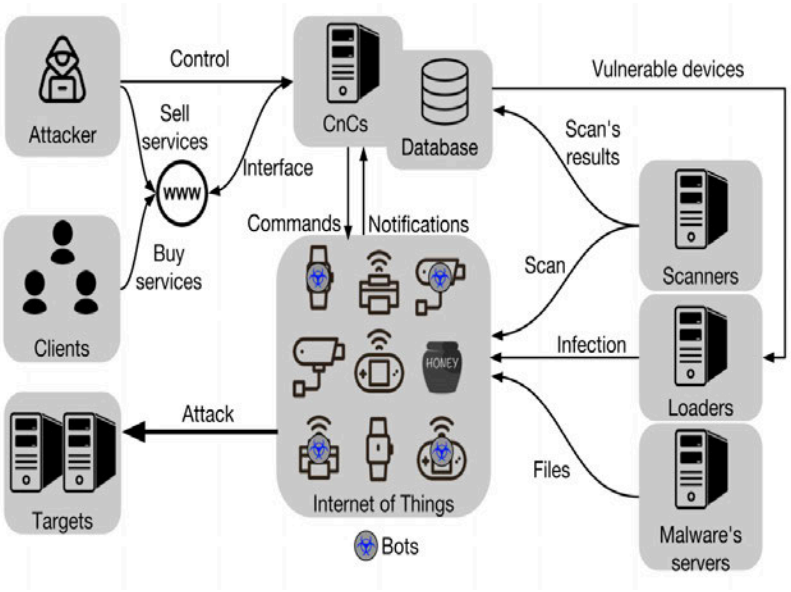


Figure 1-3: A diagrammatic representation of a botnet [31]

ii. ***DDoS/DoS Attacks:*** In this kind of attack, a large number of unwanted queries are used to overwhelm target servers. This disables the target server, causing genuine users' services to be disrupted. If the attacker uses multiple nodes to flood the target server, the attack is known as a DDoS or distributed denial of service attack. Because of the heterogeneity and complexity of IoT networks, the network layer is vulnerable to these kinds of attacks. The Mirai botnet attack, which was discussed in the previous Section, exploited this vulnerability by constantly propagating requests to the poorly configured IoT devices [32].

iii. ***Advanced persistent threat (APT):*** In this kinds of attack, unauthorized adversaries get access to the Internet of Things network. The main objective of this type of attack is to steal valuable data and information rather than to cause network damage. As a result, the intruder can stay undetected on the network for a long period of time. IoT applications constantly exchange valuable data, making them extremely vulnerable to such attacks [44].

1.7 IoT Security Solutions

According to the discussion of IoT threat categories in the previous section, a brief discussion of techniques which may be employed to secure IoT systems and networks are given in this section.

i. ***Encryption*** [45] [46] [47-49].

ii. ***Password Authentication Strategies:*** This simply entails assigning appropriate credentials to access resources. [22] [50-52] [53]

iii. ***Intrusion Detection System (IDS):*** IDS are tools that have been in use since 1970 for monitoring network traffic or systems so as to detect unauthorized or malicious activity [54, 55]. An illustration is shown in the figure 1-4.

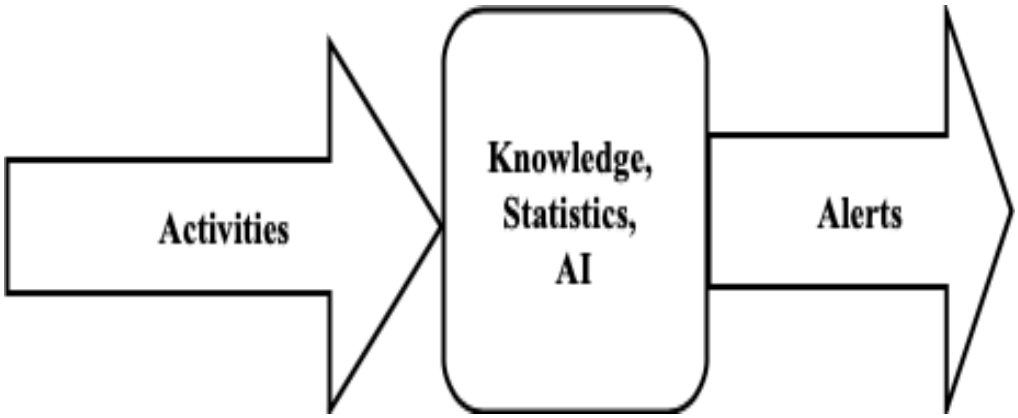


Figure 1-4: Function of an IDS

1.8 Research Motivation and Problem Statement

By 2030, It is anticipated that the total number of Internet of Things (IoT) devices will have reached 75 billion [56]. The Internet of Things will permeate all aspects of our lives, with applications ranging from home automation to smart agriculture, transportation, wearable devices, and e-health. IoT devices are major targets for botnet developers because of their vulnerability and the increasing deployment of IoT systems. The rise of IoT malware presents network administrators with new security challenges. Security of the internet of things is the major focus of our research study because the inherent characteristics of IoT systems make them different from traditional systems and extremely difficult to secure. The traditional mechanisms presented in Section 1.7 such as encryption and password authentication strategies are inadequate and insufficient [57] for protecting IoT systems. A crucial element of security of network systems is the intrusion detection system (IDS), and they are better suited to the IoT. When other defense tools are bypassed, they can be considered the last line of defense. Any IDS's validity is evaluated at the highest level by its capacity to raise an alarm upon the discovery of any malicious node that displays intrusive behavior [58]. IDS can be enhanced with Artificial Intelligence (AI) methods like the Machine Learning (ML), Ensemble learning and Deep Learning (DL), in addition to other advanced technologies. Furthermore, more time and effort are being put into research of AI-based intrusion detection systems, because of their capability to detect unknown attacks and their low false alarm rate. The majority of intrusion detection systems use either an anomaly or a misuse detection mechanism. The later mechanisms are widely used in industry to design commercial intrusion detection systems, whereas anomaly detection is limited to academia for research and development purposes [59]. Many IDS frameworks have been developed by researchers, with claims of 98%+ accuracy and a false alarm rate of <1%. However, only a few of these frameworks are adopted by industries to design real-world IDS. In general, an IDS requires existing data to identify potential attacks. Only a few studies used real IoT datasets to train and evaluate IDS model and it has also been found that most benchmark datasets have flaws that are sufficient to sway any standard IDS's detection engine. For example, the computational complexity of datasets with a large number of features will increase, use a large amount of resources, and consume too much time. Therefore, the study and development of an Intrusion Detection Technique based on Deep Learning method (DL) and Stacking Ensemble Learning

(EL) method for the IoT environment to detect and respond to potential threats as soon as they occur is the focus of this thesis. Such an IDS must consider the inherent characteristics of the IoT context especially heterogeneity, mobility and resource constraints. The proposed model was motivated by previous work with hybrid ensembles. Panda and Patra [60] proposed early hybrid classifier combinations for intrusion detection. Their research demonstrated that their hybrid ensembles outperformed any single classifier. Lastly, with the constant growth of IoT device numbers and deployment across several industries and sectors, the need to secure the IoT ecosystem by using real IoT dataset to evaluate the proposed model is more necessary than ever.

1.8.1 Research Objectives

The main objectives of the research studies conducted in thesis are highlighted as follows:

1. To explore the potentials of utilizing ML classifiers to secure the IoT from malicious threats and zero-day attacks.
2. To propose and design hybrid IDS solutions to address security challenges in today's IoT systems and to contribute the clear descriptions of the daunting task of designing an efficient IDS to knowledge.
3. To encourage researchers in IoT security to develop IDSs using ensemble learning.

1.8.2 Contributions of the Research

The main contributions of the research studies conducted in this thesis are highlighted below:

1. We carried out a survey on the state-of-the-art methods for intrusion detection on benchmark network and IoT datasets. We studied over 80 articles on this area from 2009 to 2022 to know the most efficient models and to identify gaps where improvement can be made.
2. We Identified and implemented effective solutions to the shortcoming of four benchmark datasets. It also investigates a feature selection method, an over-sampling and under-sampling technique for the design of a memory-efficient IDS and improves its performance respectively.
3. We performed statistical evaluation of ML performance results with the Friedman test for significance of the classifiers and Dunn's post-hoc test for pairwise comparison between the classifiers.
4. We proposed to leverage these ML classifiers to develop an efficient architecture for intrusion detection that overcomes the limitations of existing works. This architecture is specifically based on data augmentation, deep learning and ensemble learning algorithms. It combines the benefits of various machine learning techniques. to recognize intrusive behaviour in IoT ecosystem.
5. We investigated the proposed model abilities using transfer learning to address the issue of the lack of real-world network datasets that are labelled. We demonstrated the importance of transfer learning in certain kinds of threats. This approach was used to address the problem of a lack of datasets on real-world networks.

Some of these contributions have been published online, and others accepted for publications in Scopus indexed international journals and conferences.

1.8.3 Limitations of the Research

The publicly available datasets used to train the individual models are the main limitation of this study. The traffic characteristics observed in each of these datasets were limited. To efficiently implement the models presented, they must be trained on real-world internet traffic for a specific implementation, that varies from one establishment to the next. However, the proposed methods' generalizability, as demonstrated by evaluation on several datasets, provided some evidences that the techniques will be effective in real world implementations.

1.8.4 Overview of the Thesis

This section is composed of the organization of the rest of the chapters. The thesis is divided into 5 chapters. The chapters of this thesis reflect the different aspects of the research cycle which is illustrated in figure 1:5.

In Chapter 2, we outlined IDS architecture and provide a taxonomy for IDS. We go through the different researches done by various researchers in the previous years to get review about various machine learning, deep learning classifiers and ensemble techniques used to develop Intrusion detection System. It also introduces concepts and theories essential to IDS. The most commonly used metrics, which include not just accuracy but also a number of other performance measures used in the field of study, are defined. Finally, a summary of the chapter and an overview of the main challenges in the field are discussed.

In Chapter 3, the materials and methods used in this thesis are discussed in this chapter. Firstly, the learning techniques used are described. It outlines different machine learning techniques that could be applied for this classification experiment. It also describes ensemble learning and explains the ensemble learning paradigm. Secondly, it also introduces concepts and theories regarding intrusion detection datasets. the characteristics of the datasets utilized for the study are described. It also identifies issues inherent to each of these datasets and how these issues can be eliminated. Finally, a summary and discussion of the chapter is given.

In Chapter 4, several studies to investigate ML techniques for an improved IDS were conducted in this chapter. It concluded by conducting a detailed feature analysis to analyze the relevance of dataset’s features to be utilized to increase the accuracy of traffic anomaly detection, reduce amount of resource usage, computational complexity and its execution time.

In Chapter 5, A performance analysis of learning techniques was also conducted. These performances were compared and statistically analyzed for possible tradeoffs in the selection of base learning algorithms to generate classifiers which are members of the final stacking ensemble model. A stacking ensemble IDS model for IoT is presented. The proposed model is further tested for transfer learning capabilities. Lastly, conclusions and summary of the chapter are drawn.

In Chapter 6, we conclude the thesis based on the summary of previously mentioned chapters. A summary of the research's findings will be given in this chapter. References to future work is discussed.

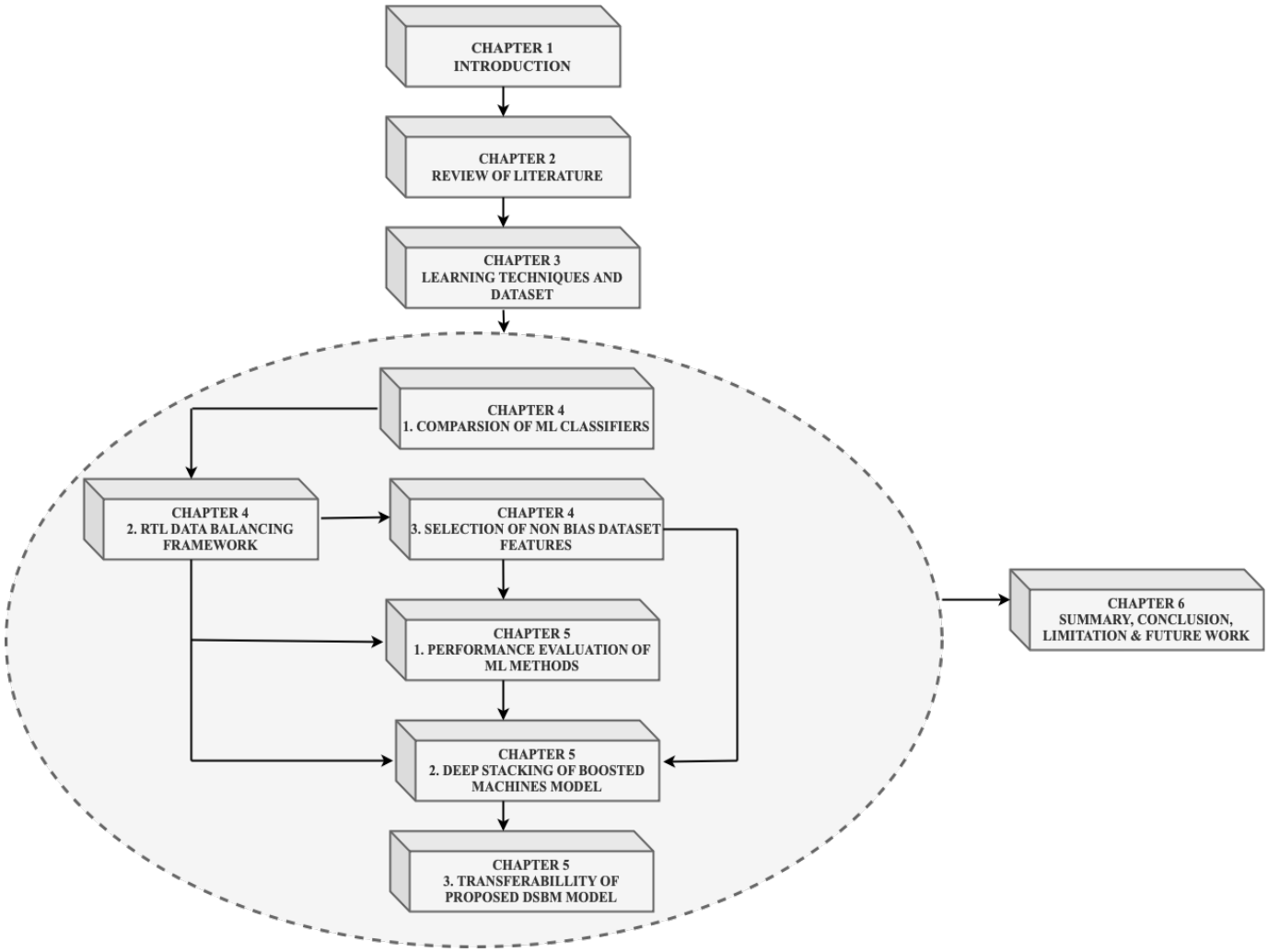


Figure 1-5: Thematic Organization of the Thesis

Figure 1.5 depicts the thesis chapters' connection and flow. The motivation, problem statement, and the objectives of the research are all discussed in Chapter 1. In Chapter 2, a literature review is conducted based on the research objectives described in Chapter 1. Chapter 3 discussed the materials and methods used in this study which includes the datasets and the learning techniques that yielded promising results in the studies reviewed in Chapter 2. The materials described in Chapter 3 were utilized in Chapter 4. The RTL technique given in section 4.2 was utilized to solve the problems of the study presented in section 4.1 and to improve the performance of the ML models presented in the remaining sections. Furthermore, the relevant dataset features presented in section 4.3 are used in the studies conducted in section 5.1 and to develop memory-efficient botnet attack detection in section 5.2. The learning algorithms that demonstrated the best possible trade-off in the performance evaluation conducted in section 5.1 were the ones used for designing anomaly-based IDS against

botnets in section 5.2. The model presented in section 5.2 was further investigated for generality in section 5.3. Chapter 6 summarizes the main findings from Chapters 3 and all the section of Chapters 4 and 5.

CHAPTER 2: LITERATURE REVIEW: TAXONOMY, RELATED WORKS AND PERFORMANCE METRICS

2.1 Introduction

Internet of Things encounters various security threats because of their fundamental characteristics and their vulnerability to a variety of security threats. Intrusion detection system for IoT security was discussed in this chapter. It also explains the intrusion detection system architecture and taxonomy. Then it provides an analysis and a review of relevant studies in this field. Finally, it summarizes and outlines the major concerns in intrusion detection domain.

2.2 Intrusion Detection System

2.2.1 Taxonomy of Intrusion Detection System

An IDS can be categorized into several kinds. Figure 2-1 depicts an intrusion detection system classified by detection strategy, deployment, architecture, and detection behavior or responses.

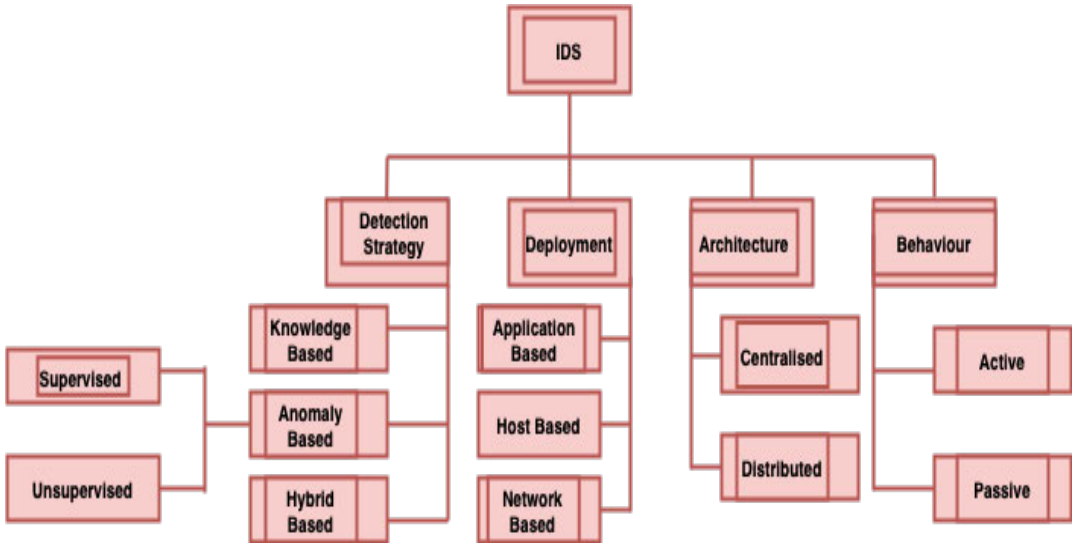


Figure 2-1: Taxonomy of IDS

- **Deployment:** These can include network-based intrusion detection systems, host-based intrusion detection systems, and application-based intrusion detection systems. Host-based intrusion detection systems (HIDSs) are installed on a host machine to monitor and analyse its activity, files, processes, system logs, and so on. Network-based intrusion detection systems (NIDSs) use sensors strategically placed throughout the network to capture and analyse packet flow. An application-based intrusion detection system (IDS) is a type of HIDS that analyses user-application interactions to monitor a specific application. Our research focuses on NIDS because our goal is to secure IoT networks.
- **Detection Strategy:** Detection system strategies are categorized into three main types namely anomaly-based detection, hybrid detection and knowledge-based detection. Knowledge-based compares

monitored data to a database of known attack signatures. If a suspicious event is discovered, it notifies an administrator. Only known attacks can be detected by this type of IDS. Unfortunately, due to a lack of signatures, it is unable to detect unknown or zero-day attacks [61, 62]. Anomaly detection attempts to learn the system's normal behaviour and any discrepancy from this pattern is regarded as a possible attack and it will react accordingly. Anomaly-based intrusion detection systems detect unknown or zero-day attacks using machine learning methods, statistical data analysis and mining**.** To address the shortcomings of each category, hybrid detection combines both knowledge-based and anomaly-based IDS.

- **Behavior:** This is also referred to as the IDS's detection response, which might be active or passive. While a passive IDS detects malicious activity and notifies the administrator or logs the activity, an active IDS is programmed to automatically block malicious attacks without the administrator's interaction. An active IDS provides the advantages of responding to an attack in real time.
- **Architecture:** The deployment architecture is classified into two types namely centralized and distributed IDS.

2.3 Artificial Intelligence

Researchers set out in the 1950s to see if computers can 'think' like human beings, giving rise to the concept of artificial intelligence (AI). It can be defined as the science of programming machines to conduct tasks that would normally necessarily involve human intelligence [63]. Similarly, the authors in [64] defined AI as the effort to automate intellectual tasks that humans normally perform. AI includes the sub – fields of machine learning, deep learning, and other methods used to achieve the purpose of automating intellectual day to day tasks that humans perform normally. Figure 2-2 depicts AI subfields such as machine learning ML and deep learning DL.

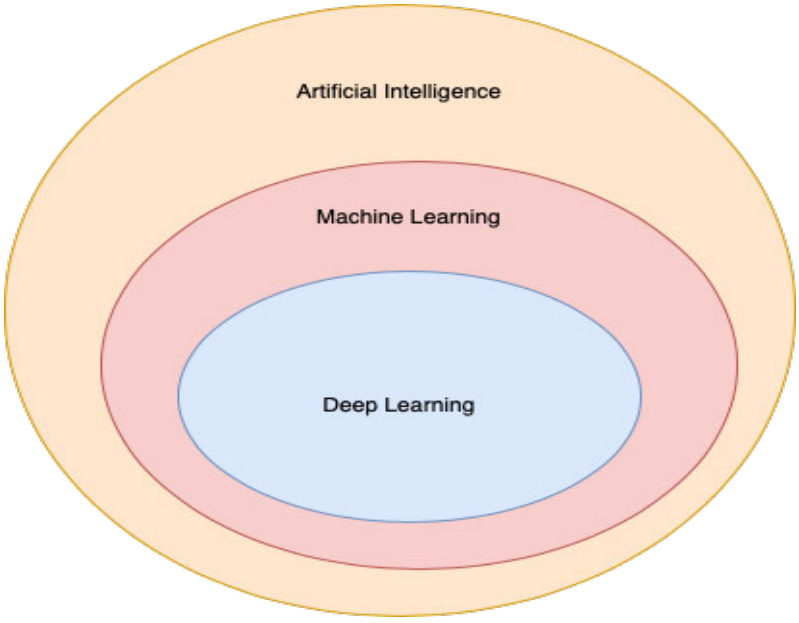


Figure 2-2: AI, ML, and DL Relationship [64]

Machine Learning is an area of study which enables computers to acquire knowledge without it being explicitly programmed. [65]. It is made up of several algorithms used to extract predictive analysis from data. High-level abstract representations are the main focus of deep learning, a subset of machine learning (ML) [66].

2.3.1 IDS Based on Machine Learning Techniques for IoT

During our review of the literature, we identified several studies that were introduced to boost IDS performance. This section highlights some noteworthy recent works that make use of both machine learning and ensemble learning methods to address IoT security challenges.

2.3.1.1 IDS Based on Machine Learning Techniques

On a daily basis, Attackers keep themselves and the techniques they use in developing new cyberattacks updated. As a result, intrusion detection techniques are rapidly evolving to ensure that newly developed malware is effectively protected from network systems. Several studies were performed with that objective, and new studies are performed on a regular basis to improve the effectiveness of intrusion detection systems.

For instance, authors in [67], evaluated ML classifiers such as Gaussian Naive Bayes, logistic regression (LR), Random Forest (RF), and Support Vector Machine (SVM) on the NSL-KDD dataset. According to the results obtained, RF classifier outperforms the other three methods.

In 2017, Almseidin et al. [68] investigated the machine learning algorithms such as Bayes Network (BN), DT, J48, MLP, Nave Bayes (NB), RF, and Random Tree (RT). DT had the lowest false negative (FN) value of 0.002 on the KDD dataset, but RF performed better in terms of accuracy.

Similarly, Zaman et al [69] compared classifiers such as Fuzzy C-Means (FCM), k-Nearest Neighbors, k-Means, NB, Radial Basis Function (RBF), SVM and an ensemble method integrating all the six classifiers on the Kyoto+ dataset. Radial Basis Function was found to outperform the others.

Pahl and Aubet [70] presented a machine learning-based approach for predicting IoT service behavior simply by observing service-to-service communications in a multi-dimensional IoT, distributed site. The overall accuracy of this technique for anomaly detection is 96.5%, with a false-positive rate of 0.2%.

Authors in [71] presented a deep learning (DL) technique for detecting distributed attacks in a social IoT network using the NSL-KDD dataset. They compared the performance of the deep and shallow models for binary and multiclass categories. Their model achieved 99.20% and 98.270% accuracy for binary-class and multi-class identification, respectively, and also 95.220% and 96.750% for deep and shallow models, respectively.

In [72], Kozik et al. presented an attack detection technique that made use of the extreme learning machine (ELM) method and the Apache Spark cloud architecture. The ELM architecture and properties enable effective computation and analysis of fog computing data in Netflow format. Their study focused on three major scenarios in IoT systems: command and control, scanning, and infected host. They achieved 99.00%, 76.00%, and 95.00% accuracy, respectively.

Hasan et al. [73] presented a data analysis-based method for detecting attacks on an IoT infrastructure that avoids the data processing overhead of signature-based techniques. Their proposed method can detect and prevent attacks on systems that exhibit unusual behavior. They conducted their study using a publicly available IoT dataset (DS2OS traffic) in [70].

A lightweight intrusion detection system (IDS) based on SVM was presented in [74] for detecting malicious activities in an IoT network. They focus on the detection of DDoS attacks on various functions like the linear, polynomial, and radical basis. Complexity and the time of processing were reduced as a result of using selected features as input. The technique's inability to identify intrusions that had no impact on traffic flow was its major drawback.

The authors in [75]. Proposed an anomaly detection technique for cloud computing using SVM. The information gain ratio was used to extract the important features from the NSL-KDD dataset. According to the study, SVM provides significant advantages for the evaluation of IDS in the cloud.

The authors of [76] used a deep learning model to detect cyberattacks on the CICIDS-2017, NSL-KDD, UNSW-NB15, and WSN-DS [77] datasets. They concluded that deep learning outperforms other machine learning techniques.

Authors in [78] introduced a two-dimensional classification and reduction approach for detecting anomalies in the NSL-KDD dataset. Linear discriminate analysis (LDA) and principal component analysis (PCA) feature extraction techniques were employed to reduce the number of attributes in the NSL-KDD dataset. Naïve bayes and K-nearest Neighbor (KNN) were used to identify anomalies, achieving a detection rate of 84.82% and a false positive rate of 4.86%.

Authors in [79] presented a joint trust light probe-based defense (TLPD) mechanism to detect malicious network nodes in an industrial IoT site while it was in an on and off state. This method was designed to detect anomalies by utilizing a light probe routing mechanism with confidence estimation for each neighbor node.

Authors in [80] presented an anomaly detection model based on a random forest algorithm to identify compromised IoT devices in distributed fog nodes. They considered only 12 features from the UNSW-NB15 dataset extracted with the Extra Tree Classifier. They achieved 99.34 percent accuracy with a 0.02 percent false positive rate.

Doshi et al. [81] proposed to use classification algorithms to detect DDoS attacks in consumer IoT networks. Legitimate traffic for the study was generated using three IoT devices. HTTP GET flood attacks, TCP SYN flood and UDP flood attacks were used to generate DDoS traffic. Decision Tree, FCNN, KNN, Random Forest and SVM with linear kernel were tested. Depending on the algorithms, the findings showed an F1 score ranging from 0.927 to 0.999.

Yang, K., et al.[82] proposed an active learning method for detecting wireless IoT network intrusions. The authors used the extreme gradient boosting (XGBoost) classifier during the experiments. The performance is evaluated using two datasets: NSL-KDD and AWID. Experiments show that active learning achieves the desired performance much faster than the random-select method. Furthermore, nearly one-third fewer labelled instances are required to achieve the same performance.

The performance of a Fast KNN Classifier (FkNN) for Network Intrusion Detection System (NIDS) on a Cloud Environment was investigated by Krishna et al [83]. When compared to traditional kNN, the gain in computational time is more than 88.0%.

Several experiments to assess the effectiveness of seven ML classifiers was conducted by Alrowaily, M., et al [84], the results obtained shows that the KNN classifier outperforms others in terms of several metrics other than training and prediction time.

Asad et al. [85] presented a DNN architecture called DeepDetect. Their architecture was tested against the CICIDS2017 dataset. DeepDetect outperformed RF and DeepGFL with a F-score of 99.0% and an AUC value close to one when compared to the other approaches. This method was only tested for DDoS attacks in the application layer.

Muraleedharan and Janet [86] proposed a FC feed-forward deep neural network framework based on flow data for detecting slow DoS attacks on HTTP. The model was evaluated using only the DoS samples from the CICIDS2017 dataset, and it achieved an accuracy of 99.61%.

Sbai and El Boukhari [87] proposed a DNN-based DL model to detect data flooding attacks in MANETs. Their model was tested using the CICDDoS2019 dataset and trained using only the data flooding attack traffic samples. The model proposed produced promising results of 0.990 for precision, F1-score, and accuracy, as well as a recall of 1.0.

Amaizu et al [88] proposed a DL method to detect distributed denial of service attacks in 5th generation and B5G domains. Their framework was designed by integrating two different DNN techniques with a feature extraction technique. On the CICDDoS2019 dataset, their model achieved 99.660% accuracy and a loss function value of 0.011. Additionally, they compared their method to existing CNN ensemble, DeepDefense, KNN, and SVM methods. In terms of precision and recall, their approach outperformed all but the CNN ensemble. The proposed model's complex structure is a drawback of their research because it may prolong detection time and affect the performance of the model in real time.

Authors in [89] presented a deep learning classification technique with a feature extraction module. To evaluate their model, they used the CICDoS2019 dataset, which was divided into Dataset1 and Dataset2 for binary classification and multiclassification, respectively. On datasets 1 and 2, their model detected DDoS attacks with an accuracy rate of about 100% and 95%, respectively. Khoshgoftaar [90] used unbalanced classroom data to examine deep learning techniques.

Sabeel et al. [91] presented DNN and LSTM models for detecting unknown denial of service and distributed denial of service attacks. Their techniques were trained on the pre-processed denial of service and distributed denial of service attack samples from the CICIDS2017 dataset before being tested for accuracy on the synthetic ANTS2019 dataset. Then their frameworks are retrained using the merged synthetic dataset and the CICIDS2017 dataset, and their detection performance against newly synthesized unknown attacks was assessed. In the second stage of the experiment, the accuracy of the LSTM and DNN models was 98.72% and 96.16%, respectively.

Virupakshar et al [92] concentrated on bandwidth and connection flooding DDoS attacks. To identify DDoS attacks in an OpenStack-based cloud, the authors used DT, DNN, KNN, and NB algorithms. On a dynamically generated dataset, the authors compared these classifiers and chose the deep neural network technique because it has a greater value of accuracy and precision. In terms of precision and accuracy, the DNN classifier significantly outperforms the DT, KNN, and NB classifiers on cloud datasets. The DNN technique, however, has a lower precision value for the KDDCUP99 dataset when compared to the other algorithms. The study's major limitation is that it employs an old KDDCUP99 dataset, in addition to the insufficient knowledge of LAN and cloud datasets.

2.3.1.2 IDS Based on Ensemble Techniques

Ensemble Learning was chosen as a strategy for complex problem solving for various reasons. The use of an ensemble model decreases the total risk of learning a simple model. Furthermore, some problems are just too complex to solve with a single classifier. To learn the complex boundaries smoothly, Ensemble Learning uses a combination of different of simple models. Additionally, EL allows you to assign a confidence probability to the generated decision, which reduces the number of false alarms generated by the IDS.

To improve accuracy over base and/or single classifiers, several ensembles methods-based IDSs have been proposed. This subsection provides an overview of some notable ensemble approaches. For example, in [93], Several ML classification techniques, including DT, J48, and SVM, were used to design an ensemble-based technique for intrusion detection. Relevant features in the KDD'99 dataset was selected using particle swam optimization (PSO). Their model achieved results with a high precision of 90% and a low FAR of 0.9%.

Authors in [94] proposed an ensemble IDS for detecting botnets in protocols used in IoT network such as the DNS, HTTP, and MQTT. Four categories of features namely DNS features, flow features, HTTP features and MQTT features were used. The flow features contain IP addresses and ports the for source and destination, type of protocol, and the time of the last connection. MQTT features include the MQTT data length as well as the number of connections going to the same destination from the same source. DNS features include attributes like the domain name, type of query, length of query, and the length of response. The HTTP features include the method of HTTP request, the URL request, the content size, and the length of URL. The model was trained in the following manner: first, a correlation coefficient is employed to select the most relevant features; then, the classification is done using the AdaBoost ensemble learning technique. The proposed framework was evaluated on the UNSW-NB15 and NIMS botnet datasets, as well as synthetic IoT sensor data. Internet of Things sensor network was simulated using Raspberry Pi. They achieved detection rates ranging from 97% to 99%, with FPRs ranging from 1% to 3% depending on the data type.

The authors in [95] presented a hybrid IDS model based on NB and SVM. A real-time historical log dataset was normalised and pre-processed for their work. Their method achieved 95% accuracy and precision after optimization. It was observed that the addition of session-based features improved classifier performance. Khonde and Ulagamuthalvi [96] introduced a method for IDS using an ensemble approach. In their study,

several supervised and unsupervised ML classifiers were evaluated on the CICIDS17 dataset. The ensemble technique produced better performance according to the results obtained.

Aljawarneh et al. [97] proposed a hybrid IDS model in 2018 utilizing a voting strategy that comprised of AdaBoostM1, Decision Stump, Decision Tree, J48, Meta Bagging, Naive Bayes and Random Tree. Consequently, 99.81% detection accuracy was attained.

Authors in [98] presented an ensemble method-based IDS model that combined correlation-based feature selection (CFS) and the Bat algorithm for the selection of optimal feature, followed by an ensemble technique which composed of the DT, Forest by Penalizing Attributes (Forest PA), and RF algorithms. Experiments were conducted on the AWID, CIC-IDS2017, and NSL-KDD datasets, with 99.5% and 99.8% each for the AWID, CIC-IDS2017, and NSL-KDD datasets, respectively.

A hybrid intrusion detection system comprised of a C5 classifier and a single class support vector machine was introduced by authors in [99], the primary objective of their experiment is to detect common instructions and unknown attacks by analysing IoT network traffic containing various forms of threats. Performance evaluation revealed that the proposed hybrid model outperformed both the Signature and the Anomaly-based IDS in terms of intrusion detection accuracy.

Bagging and boosting ensemble methods were proposed by authors in [100], employing the use of DT and RF as base classifiers. The NSL-KDD dataset was used for their study, and it was observed that bagging with decision trees produced better results.

Authors in [101] investigated ML algorithms such as DT and KNN to develop a model to detect DDOS attacks. The performance was assessed on two datasets, namely NSLKDD and KDD-Cup99. Based on the correlation approach, eight features are selected in their study. KNN performed better than the Decision Tree with a detection accuracy of 98.51% and an error rate value of 1.5%.

Authors in [102]. Tested how well different classic Machine Learning algorithms like the SVM, KNN, and DT to look for attack traffic on several ID-based datasets. DT performed better than other algorithms by achieving detection accuracy rates ranging from 99 to 100% across all datasets.

Researchers in [103] propose an ensemble-based AIDS model that uses DT, LR, and gradient boosting as inputs to an ensemble learning stacking classifier. The Chi-squared correlation technique is used to extract 23 important features from the CICIDS2018 dataset. The proposed model outperforms seven different classifiers with 98.8% detection accuracy and a 97.9% F-measure score.

2.3.3 Previous Works on Comparison of ML and Ensemble Classifiers

Table 2-1 highlights previous significant work on IoT IDS utilizing machine learning and ensemble techniques.

Table 2-1: Related work on IoT IDS utilizing machine learning and ensemble techniques

Ref.	Year	Model	Datasets	Performance Metrics
Pahl and Aubet [70]	2018	K-means	Own	96.3%
Diro & Chilamkurti [71]	2018	DL	NSL-KDD	98.27%
In [72], Kozik et al.	2018	Extreme Learning Machine (ELM)	Netflow formatted data	99%

In [73], Hasan et al.	2019	ANN, LR, RF, SVM,	DS2OS traffic	99.4%
[76], Vinayakumar, R., et al	2020	Classical ML Learning, Deep Learning	CIC-IDS2017, NSL-KDD, UNSW-NB15, WSN-DS	96.0%, 93.0%, 63.0%, 98.0%
Pajouh et al [78]	2018	Naive Bayes K-Nearest Neighbor	NSL-KDD	IR = 84.82%
Liu, X., et al. [79]	2018	Light Probe Routing	Synthetic	IR = 0.80(>)
Alrashdi, I., et al. [80]	2019	RF	UNSW-NB15	99.34
R. Doshi et al. [81]	2018	Decision Tree, FCNN, KNN, Random Forest and LSVM	NA	99.9% for DT, FCNN, KNN and RF. 99.1% for LVSM
K. Yang et al.[82]	2018	XGBoost, Active Learning	NSL-KDD	-
Krishna et al[83]	2020	FkNN	CIC-IDS2017	Av. Acc. of 99.8% for k-values of 3,5 and 7.
M. Alrowaily et al[84]	2019	AdaBoost, Decision Tree, KNN, MLP, Naive Bayes, Random Forest, and QDA	CIC-IDS2017	91.70%, 90.40%, 99.60%, 97.0%, 84.90%, 95.70%, 84.90% respectively.
Asad, M., et al. [85]	2020	DNN	CICIDS2017	98.0%
Muraleedharan, N. and B. Janet [86]	2020	Feed Forward DNN	CICIDS2017	99.61%
Sbai, O. and M. El boukhari [87]	2020	DNN	CICDDoS2019	99.99%
Amaizu, G.C., et al [88]	2021	Composite DNN	CICDDoS2019	99.66%
Cil et al. [89]	2021	DNN	CICDDoS2019	99.99%: Binary, 94.57%: Multiclass
Johnson & Khoshgoftaar [90]	2020	AB, DT, GB, KNN, LDA and RF) + SMOTE	CSE-CIC-IDS2018	99.32%,98.56%, 99.38%,95.30%, 83.62% and 99.19%,
N. Moustafa et al. [94]	2018	AdaBoost Ensemble Learning	UNSW-NB15 & NIMS botnets dataset.	UNSW-NB15: 99.54% & 98.97% for DNS and HTTP data source respectively. NIMS: 98.29% and 98.36% for DNS and HTTP data source respectively
Zhou et al [98].	2020	Ensemble	NSL-KDD, AWID, CIC-IDS2017	99.8%,99.5%,99.9%
Khraisat, A., et al [99]	2019	C5 classifier One class SVM	Bot-IoT	99.97%
Pham, N.T., et al [100]	2018	Ensemble	NSL-KDD	84.25.

2.4 Performance Parameters for Assessment of Anomaly Detection Model

This section discusses the common metrics for assessing a machine learning model's efficiency. The meaning of the popular metrics for evaluating the performance of a classification model that are described in this section, how to calculate them and which ones to use for reporting are adopted in [104].

There are two types of classes namely positive and negative classes in a binary classification problem. In a cyberattack detection task, for instance, the positive class is the "malicious" samples (which simply refers to what we want to detect), whereas the negative class is the "legitimate", "benign", or "non-attack" samples. It should be noted that all of the performance parameters used to assess the binary classification task may also be

utilized to evaluate the anomaly detection models. Evidently, the performance of an anomaly detection model is evaluated on the test set of the experimental data. These test sets typically include both positive and negative class instances. The data utilized in the training phase is the primary distinction between a binary classification and anomaly detection models. An anomaly detection model is trained using only negative instances, whereas training is done using both the negative and positive instances in a binary classification model.

2.4.1 The Confusion Matrix

A confusion matrix, illustrated in figure 2-3, is employed to summarize an anomaly detection model's performance on a test dataset.

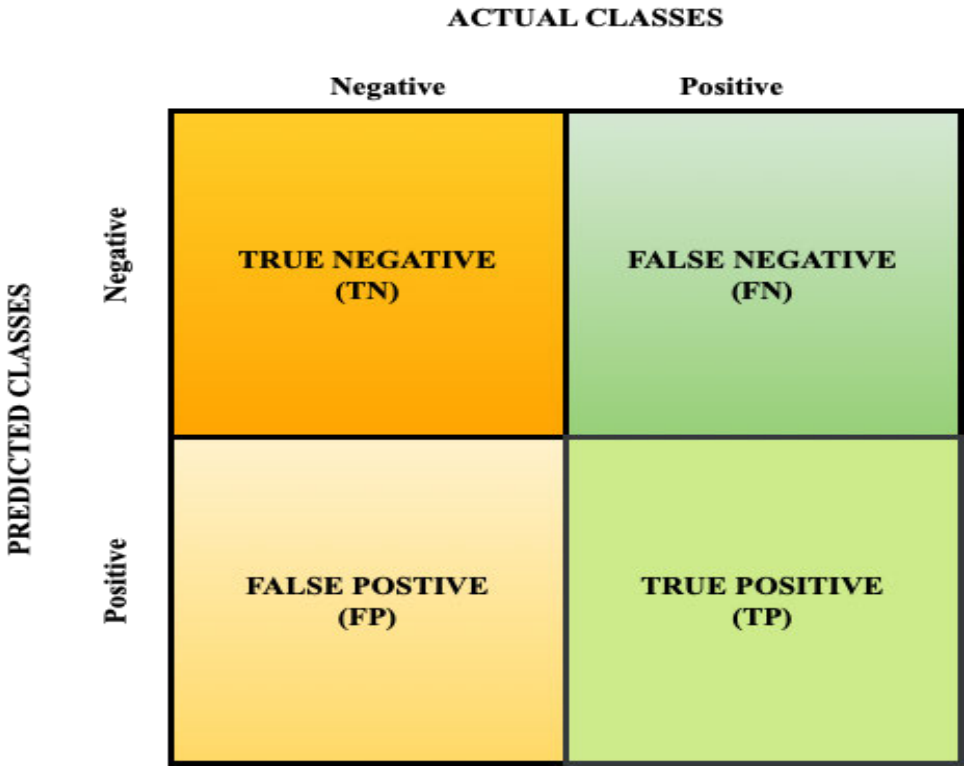


Figure 2-3: Confusion Matrix

A confusion matrix is also known as the error matrix. It is a matrix created on the basis of the result obtained by the machine learning classifiers and it is used to describe the overall performance of the classifier in terms of classification [104]. It represents the actual state of the classifier while making the predictions and it shows the number of True positives, True Negative, False Positive and False Negative. They are briefly described below:

True positives (TP): A situation in which the trained model predicted ‘malicious data’ and the source data are indeed ‘malicious’.

True Negative (TN): A situation in which the trained model predicted ‘benign data’ and the source data are indeed ‘benign’.

False Positive (FP): A situation in which the trained model predicted ‘attacked data’ but the source data are ‘benign data’.
False Negative (FN): A situation in which the trained model predicted ‘benign data’ but the source data are ‘malicious data’.

The values of true positive, true negative, false positive, and false negative are used to calculate various other performance metrics which are described in equations 2.1 to 2.5 [104].
Accuracy can be expressed as a ratio of classes that are correctly identified and it can be calculated with the equation shown in 2.1

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.1}$$

Precision is defined as the number of positive instances predicted that are actually positive, and it is calculated as shown in equation 2.2

$$precision = \frac{TP}{TP+FP} \tag{2.2}$$

The recall, also referred to as the sensitivity or attack detection rate, is the number of actual positive instances that are accurately predicted as positive. It is calculated as shown in equation 2.3

$$recall = TPR = DR = \frac{TP}{TP+FN} \tag{2.3}$$

The false positive rate (FPR) is defined as the number of actual negative instances that are incorrectly predicted as positive, and it is calculated as shown in equation 2.4

$$FPR = \frac{FP}{FP+TN} \tag{2.4}$$

The $F_{measure}$ is the harmonic mean of the Precision and the Recall. Mathematical expression is shown in equation 2.5

$$F_{measure} = 2 \times \frac{precision \times recall}{precision+recall} \tag{2.5}$$

2.4.2 The Area Under the Curve and The Receiver Operating Characteristic

A plot of the Receiver Operating Characteristic (ROC) curve and a measure of the Area Under the Curve (AUC) is another method to evaluate an anomaly detection model's performance [104]. A score is computed by the model to predict the class of an instance. If the score exceeds a particular threshold, the occurrence is classified

as positive. Or else, it is classified as negative. The ROC curve, as shown in the figure 2-4, which plots the true positive rate (TPR) versus the false positive rate (FPR) for several values of detection threshold.

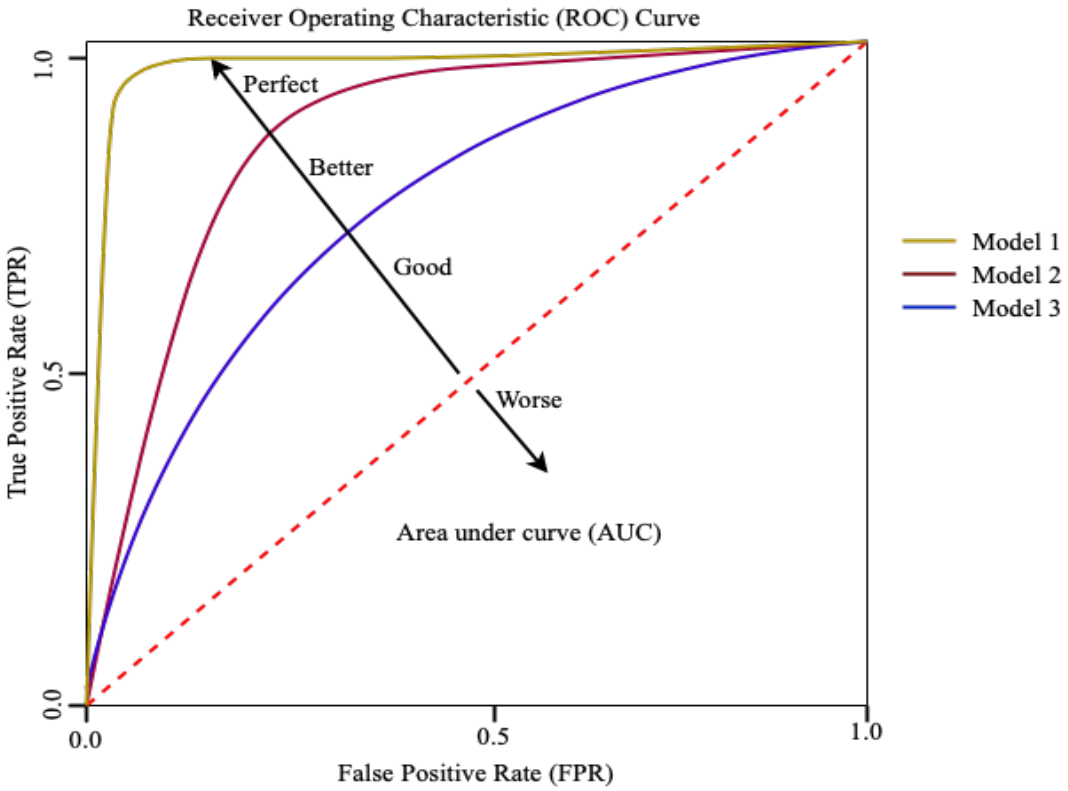


Figure 2-4: Detailed example of ROC curve

The AUC is a way of measuring of how well the positive and the negative classes can be separated. AUC value close to 1, shows the effectiveness of the model in separating the two classes. It is mathematically expressed as

$$AUC = \int_0^1 \frac{TP}{TP+FN} d \frac{FP}{FP+TN} \tag{2.6}$$

If the area under curve value is closer to 0.5, it suggests that the model performs not any better than randomly guessed values. Similarly, to the receiver operating characteristic curve is the Precision-Recall (PR) curve which plots a graph of precision against recall. The average precision (AP) is the area under the PR curve.

2.4.3 Classifier Train and Test Time

The classifier training and testing time is the time taken to train a particular classifier and the time it takes to detect attack instances. It is measured during the training and testing phase (it is the amount of time computed from when the process of classification begins until the process of classification is completed). Using the following command:

start_time = time.time()

y_pred = clf.predict(X_test)

The mean time is calculated by dividing the amount of time it took the model to classify all the samples in test instances by the total number of test instances available. The mathematical expression is given in the equation 2.7

$$\mathrm{MRT} = \frac{\sum_{i=1}^{n_{test}} t_i}{n_{test}} \tag{2.7}$$

where *i* is the instance number, t_i is the time taken to classify i_{th} test instance into a malicious or benign class, and n_{test} is the total number of test instances.

2.4.4 Macro, Micro and Weighted Average

One of the simplest and straightforward methods for averaging is macro averaging. It is calculated using the arithmetic mean of all per-class scores. This approach gives equal treatment to all classes, irrespective of their support values [104]. Micro averaging calculates the global mean score by adding the True Positives (TP), False Negatives (FN), and False Positives (FP). This definition is used to compute overall accuracy. This is due to the fact that it measures the proportion of observations that are classified correctly among all available observations [104]. The weighted-average score is computed by selecting the average of all scores per-class, while taking each class's support into consideration. The 'weight' simply refers to the proportion of the support of each class support in relation to the total value of support [104].

2.4.5 Matthews Correlation Coefficient

The Matthews correlation coefficient, abbreviated MCC, is used to evaluate the quality of binary and multiclass classifications in machine learning [104]. It considers true and false positives and negatives and is widely considered as a balanced measure that can be used even when the classes are very different in size. The MCC is a correlation coefficient value that ranges between -1 and +1. A coefficient of +1 indicates that the prediction was perfect, a coefficient of 0 represents an average random prediction, and a coefficient of -1 an inverse prediction. The statistic is also known as the phi coefficient.

2.4.6 Kappa Statistics

Also known as Cohen's kappa, it is a score that expresses the degree of agreement between two annotators on a classification problem [104]. It is mathematically expressed as follows:

$$\mathcal{K} = {(P_o - P_e)}\Big/{(1 - P_e)} \tag{2.8}$$

Where P_o is the observed agreement ratio, and P_e is the expected agreement when both annotators randomly assign labels.

2.5 Statistical Test of Significance

Comparing multiple algorithms across several datasets is important in ML studies [105]. A classifier may perform better on a dataset while failing to achieve a similar result on another. This could be due to the presence of outliers, feature distribution, or algorithmic characteristics. Therefore, comparing different classifiers may become extremely difficult. As a result, it becomes difficult to determine which algorithm is superior to others. In order to address this problem, statistical analysis is required to evaluate the performance results statistically.

2.5.1 Freidman and Dunn's Test

Two widely used tests of significance [106] are employed in this thesis to carry out the classifier comparative evaluation appropriately. The Friedman [107] and Dunn [108,109] tests are chosen for this purpose. The Friedman test seeks a significant difference between the classifiers under consideration, whereas the Dunn's test identifies the location of that difference. The major reason for using Friedman is the fact it is a very effective test when comparing more than five entities [106], [110]. The Friedman test can be used to determine whether at least one classifier outperforms the others across all datasets. If such a classification model is identified, Dunn's post-hoc tests for the pairwise multiple comparison is used. As stated in [105], post-hoc tests are necessary to identify performance differences between classifiers. For example, for Friedman test, let's assume that *d* and *k* are the numbers of datasets and classifiers respectively, the performance results of classifiers are given as (X_{ij}) and ranked as $R(X_{ij})$. Then, the addition of the $R(X_{ij})$ is calculated for each individual classifier to get R_j as shown in (Eq. 2.9), where $j = 1, 2, ..., k$. Friedman statistic is calculated with Eq. (2.10), where Q is calculated as expressed in Eq. (2.11)

$$R_j = \sum_{i=1}^{d} R(X_{ij}) \tag{2.9}$$

$$F\text{-}Statistic = \frac{(d-1)Q}{d(k-1)-Q} \tag{2.10}$$

$$Q = \frac{12}{dk(k+1)} \sum_{j=1}^{k} \left(R_j - \frac{d(k+1)}{2}\right)^2 \tag{2.11}$$

The Friedman statistic is evaluated against the F-quantiles for a given α with degree of freedom $f_1 = k - 1$ and $f_2 = (d - 1)(k - 1)$, where α is the significance level under consideration. Dunn's post-hoc tests are performed by computing test statistic γ_{xy} for all pairs of classifiers as illustrated in eq 2.12, where R_x and R_y are mean ranks of classifiers *x* and *y* respectively on all datasets, and computed as Eq. (2.13). After all the γ_{xy} are

computed and those that exceed a certain threshold are said to show a significant difference between classifiers x and y at α significance level.

$$\gamma_{xy} = \frac{\overline{R_x} - \overline{R_y}}{\sqrt{\frac{k(k+1)}{6d}}} \tag{2.12}$$

$$\overline{R_j} = \frac{1}{d}\sum_{i=1}^{d} R(X_{ij}) \tag{2.13}$$

2.6 Comparison and Discussion

As described in the preceding sections, IoT-powered NIDS by ML algorithms are of interest to many researchers. However, because some results cannot be verified, it is reasonable to be skeptical. Furthermore, researchers often don't conduct tests in the same manner. For example, they can employ a subset of the train dataset in place of the original testing dataset provided by their datasets or they may utilize only a portion of attack samples in the testing data traffic, or they may remove specific subcategories of attack samples. The disparities between the datasets is also a source of concern. Some datasets, as stated in their description, are known to have numerous shortcomings that have been extensively discussed over the years. The outcomes of these models on these datasets are uncorrelated to the outcomes of these models on real-world data. For example, high accuracy and detection rate demonstrates that an IDS has the ability to accurately detect malicious data. Unfortunately, these are misleading if the solution is tested on an unbalanced dataset. Table 2-1 compares the previously detailed proposals, with a focus on the ML algorithms used, utilized datasets, ensemble selection problems, decision combination/combination problem, metrics used, features and imbalanced handling, and threated threats.

Many intrusion detection solutions discussed in the literature are only concerned with traditional networking paradigms, and there has been little research into developing Machine Learning based IDS for IoT applications. In terms of datasets, latest information is required to train and evaluate NIDS for IoT. We discovered that proposals such as [69], [71], [76], [78], [81], [100],[102] did not provide the datasets used for their study, some are focused on non IoT datasets and did not consider IoT architecture and principles such as mobility and heterogeneity [70], [77]. Only the authors of [97] made use of a BoT-IoT dataset.

Another critical point to address is the use of ML algorithms. For example, they were used separately and in combination to improve outcomes in general systems [69]. K-means, DL, and ELM were all used independently in [70] Likewise, authors in [74], [76], [77], [78], [81], [83], [101] use ML algorithms independently. The rest, on the other hand, used hybrid methods. Although, single algorithms such as DL, Adaboost, RF and Xgboost used in [74], [78], [81], [83] produced good results, ensemble learning has emerged as a promising approach for developing efficient and reliable intrusion detection systems [111, 112]. These methods can improve the

rate of detection and false alarm of IDS over single classifiers. This is because the learning algorithms react differently to different types of attacks (for example, DoS, Probe, R2L and U2R,) [106].

A report, however, shows that the use of the heterogeneous type ensemble techniques in resolving issues in intrusion detection systems are still quite sparse [112]. Despite this, several theoretical studies have shown that the accuracy and diversity of the members of any ensemble of classifiers are related to the success of the ensemble [113]. Obtaining base learners, their parameter settings, and their number as described in [114], that meet both requirements simultaneously is a difficult task. One of the biggest challenges of stacking implementation is getting the right combination of base learners [115]. With regard to the solutions using DL, the positive outcomes could be justified by the following DL benefits: DL's capability of achieving a high rate of accuracy if given sufficient data and time, as well as training stability and generalization [116]. Because DL is a self-learning algorithm, no manual feature engineering is required [117]. Additionally, DL obtains complex and nonlinear hierarchical features from high-dimensional training data [71].

With regard to attacks, several studies frequently look into a very small number of attacks or are designed to secure internet of things systems from specific kinds of threats, primarily DoS, Probe, R2L, and U2R because they are NSL-KDD inspired. The ability to recognize attacks that it has never seen before is a requirement for an effective IDS. Zero-day attacks are quickly becoming one of the major threats to any system [118].

On performance metrics, the authors of [87] and [89] conducted model assessments employing fundamental evaluation metrics, including Accuracy, Precision, Recall, and F1-Score. In contrast, [86] and [88] extended their evaluations to encompass the Confusion Matrix metrics, while [85] performed assessments utilizing a combination of the Confusion Matrix, F1-Score, and ROC Curve. However, one of the flaws of using basic metrics such as accuracy is that if a class is imbalanced then making predictions at random could give a high accuracy score. Metrics such as Mathews correlation coefficient, Kappa statistics, macro, micro and weighted average should be considered in order to accurately evaluate the performance of a classification model. Furthermore, only a few studies in the literature examined the statistical significance of classifier performance, especially in IoT-based IDS.

2.7 Summary and Conclusion

Several services are delivered through the internet, and it has become an important factor in many domains as explained in the previous chapters. Unfortunately, these services are highly susceptible and are regularly targeted. The security of these services is a major challenge for their providers, and it is highly relevant to system users' daily lives. Furthermore, new threats emerge on a daily basis, and current mechanisms are insufficient. As a result of ML's successes in many other domains, researchers have used a variety of machine learning techniques to close these security gaps. We reviewed relevant studies related to the development of ML-based IDS for IoT in this chapter. The novelty, contributions and drawbacks of these studies are discussed

and argued. According to the literature, many factors can influence the performance of IDSs and there are still challenges that have been mostly disregarded or require additional research. The information gained from this chapter's analysis of the state-of-the-art is used in the thesis to design our own IDS. Firstly, we highlighted the most important issues to consider when designing an IDS while adhering to the constraints of internet of things, and filling several gaps found in the state-of-the-art internet of things NIDS. We will attempt to capitalise on the benefits of the works discussed in this chapter, combine the various benefits into a single model, and avoiding their flaws. The ML model parameters optimization is a hugely relevant subject where detection quality can be improved. Another area for improvement is the quality and method of pre-processing datasets. Modifications to the training dataset can improve performance as well. The experiments conducted, each proposed paradigm's framework, as well as the results of evaluation, are presented in the following chapters.

CHAPTER 3: LEARNING TECHNIQUES AND DATASETS WITH THEIR CHARACTERISTICS

3.1 Introduction

This chapter describes the materials and methods used in this study. Firstly, we discussed about the learning techniques and their characteristics. We also discussed about the datasets used.

3.2 The Machine Learning Techniques

This section describes the machine learning methods used in this thesis. To start with, we provided a general introduction to machine learning and deep learning. We define DL concepts and how IoT networks can be secured with these concepts. Furthermore, it introduces Ensemble learning approach and explains and demonstrate how to combine multiple learning methods to boost classification performance. Lastly, conclusions are drawn.

3.2.1 Introduction to Machine Learning

3.2.1.1 Logistic Regression

Logistic Regression (LR) is a supervised machine learning (ML) method for classifying data. It is applicable to categorical dependent variables. This technique has grown in popularity most recently, and its application has increased considerably. The goal of logistic regression is to assign data to the appropriate classes based on their correlation. Consider the following simple linear regression equation for a mathematical expression of logistic regression:

$$y = \beta_0 + \beta_1 * x \tag{3.1}$$

Applying the sigmoid function on the equation 3.1 yields:

$$p = \frac{1}{1+e^{-y}} \tag{3.2}$$

The LR equation is obtained by substituting eq. 3.2 into eq. 3.3.

$$\ln(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \cdots \beta_n x_n \tag{3.3}$$

Its value ranges from 0 to 1, and $\beta_0, \beta_1 \ldots . \beta_n$ are the regression parameters, $x_1, x_2 \ldots . x_n$ are the values of the predictor.

3.2.1.2 The Stochastic Gradient Descent Algorithm

Popularly known as SGD, it is a ML optimization technique which is commonly utilized to determine the metrics of the model which best fit the actual and expected outputs. SGD Algorithms are a kind of gradient descent method that address the challenges of computational time. SGD computes the gradient of a subset of observations chosen at random rather than all of them. Notably, SGD is an optimization technique, whereas Logistic Regression (LR) is a machine learning algorithm. A machine learning model describes a loss function,

which is minimized or maximized by the optimization technique. Some machine learning libraries may create confusion between the two notions. In scikit-learn, for instance, there is a model named SGDClassifier that could mislead a user into believing that SGD is a classifier. However, that is an SGD-optimized linear classifier.

3.2.2 Introduction to Deep Learning

Deep learning is a subfield of ML that mainly focus on learning successive layers of increasingly meaningful representations of all the input data. In addition, it enables the scalable training of nonlinear models on big datasets and excels at generalising to new instances when the input data is highly dimensional and complex [119]. Artificial neural network (ANN) with multiple layers are employed to perform the learning. A layer consists of neurons. Figure 3-1 shows how the output of a neuron is computed with an input vector of 2 dimensions. Let's assume that the input vector is (x_1, x_2), The neuron's output is given by the equation 3.4.

$$y = \sigma(x_1 w_1 + x_2 w_2 + b) \tag{3.4}$$

Where y = output, σ = activation function 'f', w_1 and w_2 are connection weights and the bias is b. The introduction of nonlinearity that allow the model to learn complex nonlinear functions is the role of an activation function. A matrix can be used to represent all connecting weights of all the units. Therefore, a layer is made up of an activation function and a matrix multiplication.

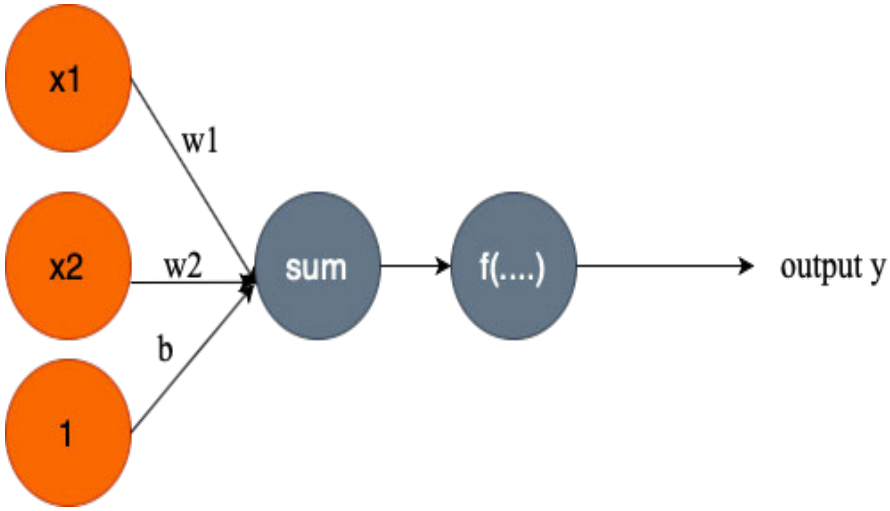


Figure 3-1: A detailed representation of how a neuron's output is computed

A model's depth is determined by how many layers it has, as illustrated in the figure 3-2, which is an ANN with one input and output layer each and two hidden layers. Deep neural networks no longer require feature engineering because successive layers allow them to learn very complex dependencies between the input features on their own. In fact, if the input data is large enough, a neural network can approximate any function of it.

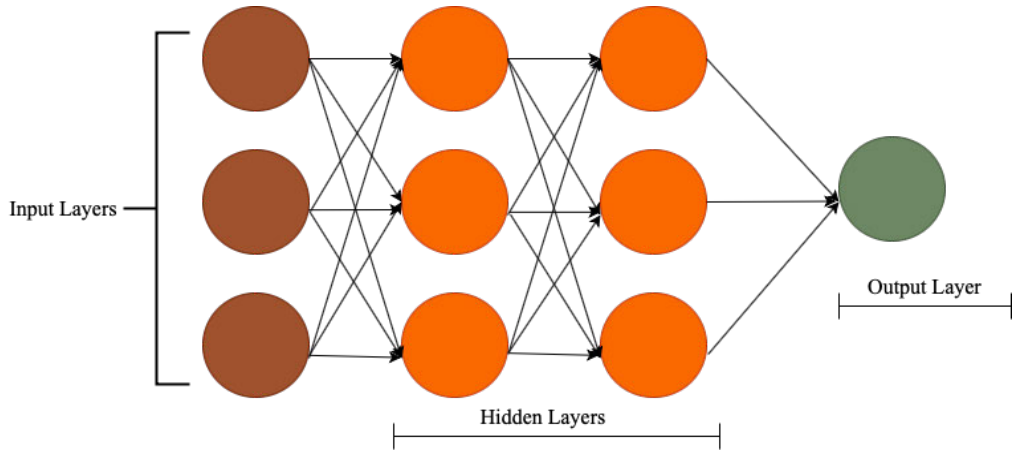


Figure 3-2: Artificial Neural Network architecture with two hidden layers

DL is relevant in the field of IDS because it deals with a huge number of data and the techniques generated by DL are capable of generalizing zero-day/new forms of malicious threats that are not explicitly represented in the presently available labelled data. DL models can be efficient for a new, unseen network environment and could also be leveraged in a ML pipeline as part of a transfer learning stage when utilized with dataset from a different network environment.

3.2.3 Ensemble Learning

Ensemble is a term that describes the idea of combining the outputs of multiple classifiers into a single output [120]. To obtain more reliable and accurate predictions, an ensemble may consist of a single or multiple (homogeneous or heterogeneous) learners. To generate and integrate learners, a variety of strategies can be employed; for example, different set of data could be employed for training the same learning methods or/and the same set of data could be employed for training different methods. [121]. The major issue with ensemble learning is deciding which algorithms to use to generate the ensemble and which decision function to use to combine the outputs of these algorithms. The more classifiers, of course, the stronger; though, the computational cost of integrating a new algorithm must be considered. Dieterich summarises three major reasons for using ensemble-based methods in his review article [120]. These reasons include statistical reasons due to inadequacy of sufficient data, representational reasons to overcome the issue of many ML algorithms' inability, and computational reasons. The ensemble model's design is divided into two major steps: generation and integration [122]. A set of base classifiers are produced in the generation phase. During the integration phase, the decision is made on how to merge the outputs of the base classifiers into a single one. The ensemble concept is now being used to develop many well-known modern machine learning techniques. The most commonly used ensemble methods are bagging, boosting, and stacking [112]. These methods combine multiple learning models into a single model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking). The diagram 3-3 depicts the general workflow of ensemble methods.

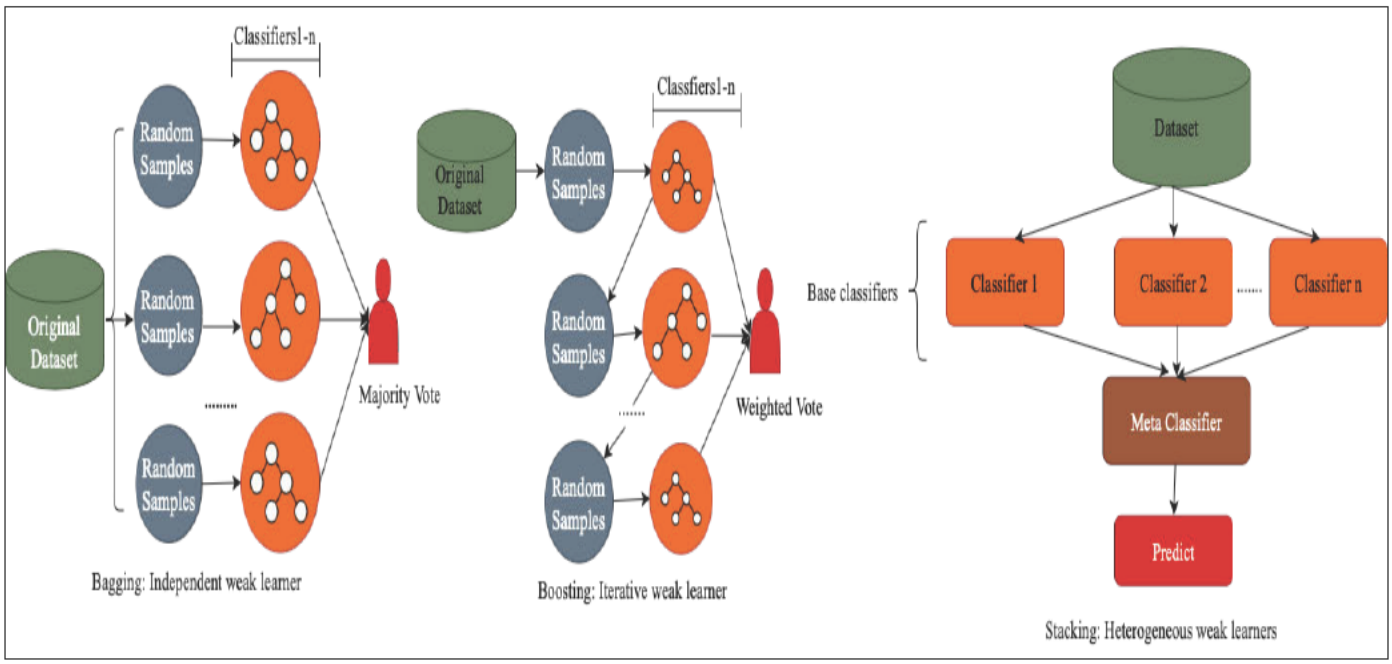


Figure 3-3: Workflow for Ensemble Methods

Bagging techniques are among the oldest and the most intuitive methods of assembly implemented in parallel. Several bootstrapped copies of the training dataset are inputted into the models, resulting in models that are only slightly different from one another. The copies are the subsets of the whole training data, with each copy removed at random with replacement. Lastly, the outputs of each learner are combined to produce the ensemble's final output. Boosting is another ensemble method that employs algorithms sequentially. The first algorithm analyses each instance in the dataset and assigns weights to them. The instances with a higher weight value are those that were classified incorrectly by the algorithm. The next classifier takes as input the weights for all instances in the dataset. The weights enable the algorithm to concentrate on the most difficult instances to categorize. The second algorithm updates the weight of the instances based on its results and sends the data to the third algorithm. This process is repeated until the data has been processed by the ensemble's final algorithm. The final decision for both of these methods can be obtained through majority voting. As a result, for any instance input, the ensemble's decision will be represented by the class that the learner model predicted the most. Nonetheless, there are other approaches to analysing and obtaining an optimal combination. Ensemble learning has emerged as an effective method for providing reliable and intelligent intrusion detection systems [111, 112]. These methods can improve the detection rate and false alarm rate of IDS (e.g, efficiency) over a single algorithm. This is because the learning algorithms react differently to different types of attacks (for example, denial of service, Probe, user to roots (U2R), and root to local (R2L)) [123]. However, according to a recent survey, the use of the heterogeneous ensemble in addressing IDS challenges is still quite limited [112].

3.2.3.1 XGBoost

Extreme gradient boosting (XGBoost) is a boosting method that is part of the ensemble-based technique that uses decision trees for building a model. Chen and Guestrin first presented the XGBoost algorithm at the SIGKDD Conference in 2016, and it has since had a significant impact on the ML community worldwide, from winning Kaggle competitions to addressing significant issues [124]. XGBoost algorithm performs best when the dataset size is medium to small. It is approximately 10 times quicker than the existing techniques on a single platform, removing time consumption issues, particularly during network data pre-processing. The portability of XGBoost makes it accessible and simpler to implement on a wide range of platforms. Multiple programming languages, including Java, Python, R, and C++, can handle XGBoost. XGBoost's ability to transform a weak algorithm into a stronger algorithm through the optimization phase for each attached new tree allows the classification model to produce fewer false alarms, easier labelling of data, and accurate data classification. Initially, an objective function that describes the model performance is defined in xgboost. The objective function [125] is divided into two stages called a training loss and regularization for stage one and two respectively.

$$obj(\theta) = TL(\theta) + R(\theta) \tag{3.5}$$

θ represents the parameters, which could be numerous depending on the dataset. *TL* represents the training loss and *R* represents the regularization term. TL is simply a measurement of the model's accuracy. Regularization helps to keep the complexity of the model within desired limits, avoiding issues like data overfitting that can lead to a model that is less accurate. Extreme gradient booting algorithm works by adding all of the dataset's tree predictions and optimizing the result. The new objective function (at step *t*) normally includes up to the second order and takes the shape of an expansion as stated by Taylor's theorem

$$obj^{(t)} = \sum_{i=1}^{n}[m_i f_t(p_i) + \frac{1}{2} c_i f_t^2(p_i)] + R(f_t) \tag{3.6}$$

where $m_i \; and \; c_i$ are used as the inputs. The outcome represents the ideal optimization for the new tree that wishes to join the model. XGboost handles loss functions in this manner. Moving on, regularisation is crucial in the definition of the tree complexity *R(f)*. Tree *f(p)* can be defined more precisely as stated in [125].

$$f_t(p) = w_{q(p)}, w \in R^L, q : R^d \rightarrow \{1,2,3,4 \dots \dots, L\} \tag{3.7}$$

In equation 3.7, function *q* is responsible for assigning leaves to the data points based on a vector of leaf scores represented by *w*. The number of leaves is represented by *L*. XGBoost's complexity is given in equation 3.8 [125].

$$R(f) = \alpha L + \frac{1}{2}\beta \sum_{j=1}^{L} w_j^2 \quad (3.8)$$

The resulting model includes the tree model that is newly reformed and measures the quality of the tree structure q(p). Because it is not possible to calculate all tree combinations simultaneously, the structure of the tree is established by computing the regularisation, leaf scores, and function of the objective at each level. The gain is computed at every level as a leaf is split into a left and a right leaf, and the gain is computed at the current leaf with the regularisation obtained at any additional leaves. The branch is terminated if the gain is less than the additional regularisation value (this concept is known as "pruning"). XGBoost runs so deep into trees and classifies data in this way, calculating accuracy and other parameters.

3.2.3.2 Light Gradient Boosted Machine

LGBM is a decision tree-based high-performance gradient boosting system. LGBM utilizes gradient-based One-Sided Sampling (GOSS) to filter out the samples for finding divided values [126]. It is employed for classification, ranking, and a variety of other ML tasks. Gradient reflects the tangent slope of the loss function. If the data gradient is large so they have more error. The points are so crucial in finding the ideal point for split. GOSS performs random sampling of the instances with tiny gradients and retains all the instances with large gradients. GOSS attains a good balance in maintaining the accuracy for learned decision trees and decreasing the amount of data instances. LGBM divides the tree leaf-wise according to the best fit. Thus, in LGBM, growing on the same leaf with the leaf wise approach reduces loss greatly compared to other existing boosting methods. High speed, the capacity to handle large data, increased prediction accuracy, support for GPUs, and distributed learning are the key characteristics of LGBM. Light GBM is faster than XG-Boost and is a better method for training on large data sets. Because of its high speed, it is prefixed as 'Light'. Its advantages are: takes lower memory to run, can handle the large size of data, focuses more on the accuracy of results.

3.2.3.3 The Gradient Boosted Machine

GBM belongs to the ensemble family, that is designed to enhance the decision trees (DT) performances [127]. Just like any other boosting technique, they combine weak learners such as the DT sequentially and enables them to optimize an arbitrary differential loss function to produce a stronger model for classification. Every learner (current tree) depends on the prediction of previous learners to improve prediction errors. Considering a set of random input variables *x* shown in equation 3.9 and random output variables denoted by *z*.

$$x = \{x_1, x_2 \dots \dots \dots \dots, x_N\} \quad (3.9)$$

Using training data $\{z, x_i\}^N_1$, we hope to obtain an approximate value A which maps *x* to *z*. The expression of *A* is given in Eq. (3.12). Let us assume that a dataset *(S)* containing *p* samples and *q* features, as denoted by Eq (3.10). The ensemble then employs M additive function for prediction of the final output Eq (3.11).

$$S = \{(x_i, z_i)\}(|D| = p, x_i \in \mathbb{R}^q, z_i \in \mathbb{R}) \quad (3.10)$$

$$\hat{z}_i = \phi(x_i) = \sum_{m=1}^{M} f_m\ (x_i), f_k \in A \quad (3.11)$$

$$A = \{f(x) = w_{r(x)}(r : \mathbb{R}^m \to K, w \in \mathbb{R}^K)\} \quad (3.12)$$

The set of DT instances is given as *A*, the structure of the tree that connects an instance to the corresponding leaf index is represented as *r*, *K* represents the total number of the trees, and f_m is a single tree with a structure of *r* and leaf weight of *w.* The ensemble tree determines the final output by adding the scores of the leaves *(w)*, identified by classifying the test sample provided. Assuming, the first classifier (i.e., tree) predicts $h_1(x)$ over a sample $\{(x_i, y_i)\}^{N}_{1}$. Then, the next classifier uses $h_1(x)$ as input to modify the weights of initially misclassified instances. As a result, the next classifier predicts $h_2(x)$ over $\{(x_i, y_i - h_1(x_i))\}^{N}_{1}$. A final prediction $h(x)$ for a given data sample *S* is the sum of the trees' predictions while reducing the error in prediction. GBM's hyper-tuned configuration parameters are five hundred estimators, maximum depth of the construction the tree is 3, a learning rate of 0.1 and 100 samples is the minimum required for split.

3.2.3.4 Extra Randomized Tree

Extra randomized tree is an induction technique used for supervised classification and regression [128]. ETC, specifically constructs an ensemble of unpruned decision trees. ETC's main procedure is to choose both features and cut-points at random, regardless of the target variable. This procedure is followed at each tree node by completely or partially selecting a subset of a set of features from which the best one is chosen. Worst case scenario, the algorithm chooses only one feature and cut-point at each node. Totally randomized trees that are not dependent on the values of target attribute in the training sample are built in this manner. When constructing the ensemble, the traditional top-down methodology is used. In contrast to other tree-based ensemble algorithms, ETC grows the trees while reducing biases and variances by using the entire train sample set, rather than replicas of the bootstrap. Three critical parameters influence the ETC splitting process for numeric features. *K* is the first parameter and it represents the number of features chosen at every node and it is also responsible for the feature selection procedure's strength. The second parameter is n_{min}, it represents the lowest training set size for splitting a node and regulates the output noise averaging. The third parameter is T_{count}, which specifies the reduction in variance and is the number of the trees in the ensemble. These parameters are important in the ETC architecture. ETC performs similarly to RF, but with optimal selection of features, and it is computationally faster than RF. The hyper tuned parameters for ETC gotten from randomized search are highlighted as follows:

- No of estimators: 1788
- Max. value of tree depth: 10
- Min. split for sample size: 5

- Number of features for best split: $log10_2$
- Criterion Considered: Gini with no bootstrapping.

3.2.3.5 Random Forest

Random Forest is a set of predictors of trees $\{t(x_{in}, \theta_n), n = 1, ...\}$ each of which makes a prediction on a given input x_{in} [129]. Each predictor is determined by a random set of variables given by $\{\theta_n\}$, which are independently sampled using distribution of the same type. The key concept of Random Forest is that by combining a large number of predictors, you can improve the accuracy of prediction and avoid the problems of over-fitting. In RF, each predictor grows to its maximum size without being pruned. As soon as huge number of trees is generated, they predict the input data by choosing its most popular class at the input x_{in}. The number of trees (estimators) for the performance assessment is set to 500, and 26 is assigned as the maximum depth for tree construction as recommended by [110, 129]. The remaining parameters are obtained through the use of randomized search.

3.2.3.6 AdaBoost

AB is a meta-estimator that learns the weights of the initial training from the original dataset [130]. Based on incorrectly classified instances, these weights serve as input to the additional copies of the classifiers. The weights of the classified instances, are then adjusted by subsequent classifiers. In this manner, AB enhances learning algorithm performance by boosting weak learners so that the final model converges to a strong learner. The equation (3.13) depicts a boosted classifier, in which c_p represents a weak learner and x represents an input object.

$$C_P(x) = \sum_{p=1}^{P} c_p\,(x) \tag{3.13}$$

$c_p(x)$ returns the predicted class value. For every instance in the training dataset, every c_p generates a hypothesis for the output ($h(x_i)$). At every iteration point p, a c_p is selected and a coefficient β_p is assigned so that the sum of training error E_t of the resulting p-stage $C_P(x)$ is minimized. This is denoted in equation 3.14. $C_{p-1}(x_i)$ is the built boosted classifier during the last training stage, $E(C)$ is the error function that needs to be minimized, and $c_p(x) = \beta_p h(x_i)$ is the weak learner which is added to the final model. AB's optimal parameters are 50 estimators and a learning rate of 0.1.

$$E_t = \sum_i E\,[C_{p-1}(x_i) + \beta_p h(x_i)] \tag{3.14}$$

3.2.3.7 Stacking Technique

Stacking approach is a supervised ML concept which seeks the most ideal combination of algorithms. The stacking algorithm differs from bagging and boosting in that it attacks a problem from multiple perspectives. Different kinds of models are trained independently to overcome various weaknesses. Then, a level two classifier, which is also called a 'meta-classifier' is trained to determine the best combination of the independently trained base classifiers to give the final results. The concept of stacking is also known as stacking generalization [131], a method of incorporating strong, diverse groups of classifiers. Not all combination strategies must be learned [112]. Several methods with good general motivation could be used, for example, "Majority Voting" could be used to ascribe a class to a specific instance if nearly half of the total models demonstrate that a specific class is greater than half of the entire number. Stacking is a technique used to create heterogeneous ensembles, as compared to bagging and boosting, which typically generates homogeneous ensembles.

3.3 Datasets for Developing Proposed Model

Dataset availability is important in the domain of intrusion detection systems. A massive quantity of data is required to train these machine-learning classifiers. Furthermore, because machine learning problems are heavily reliant on data, the quality of the datasets available is important. Therefore, the higher the data quality, the better the algorithms. However, one of the current major challenges in the domain of network intrusion detection inadequacy of publicly available, labelled data that could be utilized for efficient training, testing, evaluation, and comparison of ML approaches [132, 133]. There are numerous datasets available for intrusion detection, as evidenced by the survey conducted by the authors of [134]. While these datasets have proven useful, most of them are not created specifically for IoT networks. Furthermore, there are some debates about the usefulness of using them in modern research, as most of them may only be more suitable to provide additional validation and cross-checking of novel techniques [135]. Often, the most effective datasets for network intrusion detection contain captures of real-world network environments. After conducting an extensive survey, four datasets which are cicids2017, IoTID20, N-baIoT and Nsl-Kdd, datasets were selected. These are recent datasets with numerous real - world attacks and are designed specifically for IoT networks. This chapter describes these experimental datasets in detail, a description of the environments in which these datasets were generated, and a detailed characteristic of these datasets and attacks existing in them. Furthermore, this chapter identifies issues inherent to each of these datasets and how these issues were eliminated in order to improve the detection and classification of any future intrusion detection engine. Lastly, conclusions are drawn.

3.3.1 Dataset Gathering

A simulation environment, honeypots, access logs from web-servers, payload captured from a deployed high-performance network monitoring system and alerts from firewalls are all methods for gathering data. A honeypot is regarded as an effective and a useful tool for gathering and observing attack vectors and their characteristics [136]. Using a honeypot to act as a router offers researchers a malicious traffic dataset and will

aid in the capture of routing traffic in particular [137]. Another source of data is firewall logs and/or IDS alerts, which could be utilized for passive measurement of live networks [138], and the collection of various characteristics of the header files of tree which contains GET and POST request data which passes value to numerous web applications [139]. Access logs from web servers are datasets which stores comprehensive information about several user access and user session and they are thought to be very important for monitoring various web behaviours [140]. [141] provides an example of payload captured from a deployed high-performance network monitoring system on a website for different time periods. The authors of [142] utilised a network simulated platform as a source of data for training and testing their intrusion detection system. Data gathered can be enriched by including several pieces of information, such as the addresses of packet source, passive OS finger-printing on tcp-dump logs, and well-known internet protocol addresses that have been blacklisted. A dataset that is enriched may also supplement an entry with relevant working days, hours, holidays, and so forth. Description of the honeypot environments, port analyses of the observed sources, and the attacked honeypot machines concurrently or sequentially could also be employed to supplement the datasets [143]. Datasets could also include instances of various attacks simulations, penetration tests, and/or tools used for vulnerability discovery so as to make sure that the dataset consist of a variety of threats [144].

3.4 Pre-Processing Datasets

Real-world data is frequently insufficient, noisy, and not consistent. This is a result in poor-quality of data collected and, as a result, poor-quality of models built on such data [145]. Data Pre-processing is used to address these issues. It is a Data Mining step that deals with data preparation and data set transformation [146]. It provides methods that can assist us in understanding and making data knowledge discovery more efficient. As explained by [147] data pre-processing is in four steps namely cleansing, integration, transformation and reduction. Each of these methods consists of various techniques, some of which are used in this thesis and are explained in the next sub-sections.

3.4.1 Data Cleansing

Real-life datasets are mostly noisy and incomplete, having incorrect attribute values and inconsistent. This may be due to human and computer errors that may have occurred during the data entry stage. Errors during the transmission of data may also occur or the devices for data collection may be faulty. Data cleaning procedures aim to fill in missing values, reduce noise while detecting outliers, and fix data discrepancies. Therefore, a useful Data Pre-processing step is to put the data through a few Data Cleaning routines.

3.4.2 Data Integration

Data Integration is a process or method of combining data from various sources into a comprehensible data store. These sources could include multiple files, databases, and so on. For example, each dataset used in this thesis' experiment is scattered across several files. Individual file processing is a time-consuming task. As a result, we merged those files to create a single file. Additionally, in entity identification problem, how are real world entities

from various data sources matched? For example, a data analysis professional needs to ensure that 'Destination Port' in a database and 'Dst_Port' in another database refers to the same entity.

3.4.3 Data Normalization

Input data typically differs in dimensions and orders of magnitude. Data normalization is an important step when using machine learning techniques. It is a form of data transformation where the attributes of data are scaled to fall within a small specific range, like -1.0 to 1.0, or 0 to 1.0. The features in the datasets used in our experiment have different ranges and varying scales, therefore, data normalization techniques [148] was used to reduce the differences for improved performance. The min–max scaler technique was adopted in this thesis for the normalization of data and the reduction of differences in different dimensions. This technique scales the data to the interval of [0, 1] through a linear transformation. It can be done with the formula given as follows:

$$\widetilde{X_{jk}} = \frac{X_{jk} - \min(X_j)}{\max(X_j) - \min(X_j)} \tag{3.15}$$

where $\max(X_j)$ and $\min(X_j)$ represents the maximum and minimum value of the *jth* feature respectively, and X_j and $\widetilde{X_{jk}}$ represents the normalized feature value between [0, 1].

3.4.4 Feature Selection

Feature selection is a process of determining whether a feature is relevant or not relevant for a specific classification task. It entails identifying a subset of data from the entire dataset in order to reduce the amount of data while preserving the authenticity of the original feature set by identifying important features. Additionally, feature selection technique a method of data reduction which [149] can be employed as a pre-processing phase in machine learning algorithms to reduce computational complexity. Redundancy and relevance are two factors that influence the importance of features and should be considered when selecting them. When multiple features have similar characteristics, this is referred to as "redundancy." The importance of a feature to the overall model is defined as its relevance. The feature is said to be relevant if removing it from the feature set reduces model accuracy. The goal of feature selection should be to identify the relevant features and delete those that are redundant. These unnecessary and redundant features will reduce classification accuracy. To simplify data and reduce feature set size without sacrificing predictive accuracy, feature selection is frequently combined with machine learning modelling [150, 151]. Wrappers, filters, and embedded methods are three main feature selection methods [152]. The filter and wrapper methods are the most commonly employed [152]. Filter techniques are simple for scaling and assessing the relevance of features using intrinsic properties of the data analysis and evaluation criteria. Typically, a feature score is generated, and features with very low scores are eliminated [150]. Filter techniques do not rely on classifier feedback [153]. The disadvantage is that ignoring a classifier during pre-processing may lead to a poor classification performance. Several parameters, like the correlation, information gain, and entropy, could be used to measure feature importance in filter techniques. Authors in [154] presented a framework for eliminating features that are not necessary so as to produce a more

robust and efficient classifier by employing various feature selection approaches. The researchers discovered that when only 17 characteristics from the NSL-KDD dataset were used, their model achieved the highest accuracy. Researchers in [155], also proposed an IDS approach based on the extraction of features in the NSL-KDD dataset. This framework is based on subdividing the input dataset into several sub-sets and merging them using an Information Gain (IG) filter. The selection of an optimal number of features is a significant concern in filter-based feature selection. Cross validation can be used to overcome this, but it is computationally very expensive and defeats the intent of using appropriate filter-based methods [156].

3.4.5 Data Partitioning

In order to train and evaluate an ML model, the dataset must be divided into three parts, which are the training set, validation set, and test set [157, 158]. The model parameters are learned using the training set, and the validation set is used to tune the hyperparameters of the model. The numbers of layers, units per layer, and the total number of training iterations, are all hyperparameters of a neural network. Several models, with each having a different hyperparameter configurations, are trained and evaluated on a validation set to determine the best hyperparameter configuration yielding the best results. The final model's performance on previously unseen data is evaluated using the test set.

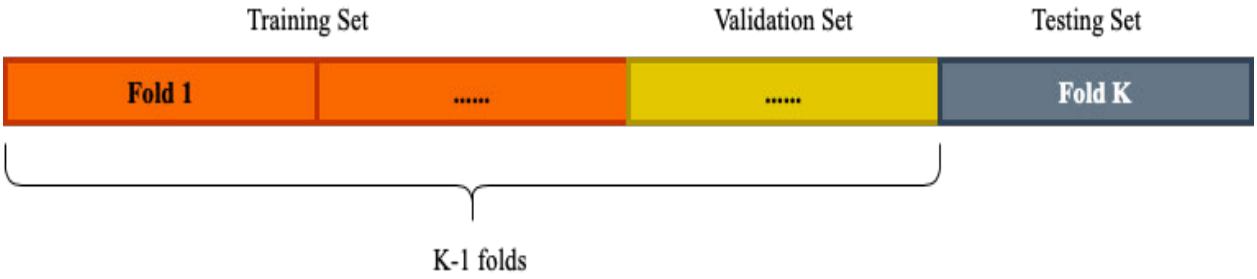


Figure 3-4: Cross Validation

Splitting a small dataset on the other hand, may significantly reduce the number of samples that would be available for training and testing. For various reasons listed below. Different researchers have used different methods and/or percentages to split these datasets such as;

- Selecting only a few types of attacks.
- Gathering a subset of the data.
- Creating a new data by merging the train and test sets of an existing dataset.

These discrepancies are problematic because it is possible to contend that varying the dataset proportions may produce totally different results. Also, the cost of simulating the settings each dataset's so as to be able to compare them is significant. Although, K-fold cross validation will be used in this thesis, we briefly described two most common data splitting techniques below:

3.4.5.1 Cross Validation

Cross-validation, also known as 'k-fold cross-validation,' occurs when a dataset is randomly splitted into 'k' number of groups. It is an efficient splitting technique for evaluating the performance of a model. As shown in figure 3-4, the concept is to split the dataset into k-folds. The model is trained with k-1 folds (train set and the validation set) and tested with the unused folds. The procedure is carried out k times, with each k-fold serving as the test set only once. This technique has been extensively employed in IDS studies, including [159-161]. The model's overall performance can be calculated by averaging the results obtained for each fold that is employed as the testing set. It should be noted that the nested k-fold cross validation could be used to further divide the k-1 folds utilized to learn the model into a new set of training and validation sets.

3.4.5.2 Hold Out Validation

Hold-out typically happens when you divide your dataset into 'train' and 'test' sets. The training set is used to train the algorithm, and the testing set is utillised to check how well the model performs on a new dataset. A common way to split the data when using the hold-out method is to use 80.0% of the data for training while the remaining 20.0% for testing.

3.4.6 Class Imbalance Nature of Datasets

An imbalanced training dataset is one in which the proportion of the benign traffic is much more than the malicious traffic. A model that trained on an imbalanced dataset will be biased towards the majority samples during the testing and validation phase [162-165]. The datasets used in this thesis are susceptible to a high level of class imbalance. The ratio of class imbalance is shown in the tables in section 3.5. This is a major challenge that we discovered in the experimental datasets. There are numerous approaches to dealing with a class imbalance problem of a dataset [162, 166-168]. Relabelling of class samples is one of the major ways to address the issue of class imbalance. Class relabelling entails splitting the samples of majority classes to form more classes or combining a few minority classes to form a single class, thereby improving the ratio of prevalence and reducing the challenge of class imbalance nature of the dataset. Cost-sensitive techniques, Classifier specific solutions, and Resampling technology are some other solutions to the problem of class imbalance in datasets. Because our experiments were conducted using different dataset with different characteristics, it is a tedious task to relabel and merge each of them. Therefore, a less tedious solution described in [169] is adopted in this thesis to solve the problem of class imbalance in our datasets.

3.5 DATASET DESCRIPTION

This study focuses on the use of newer benchmark datasets which are readily collected data that are generated using a simulated environment. We had two objectives in mind in the selection of our dataset highlighted below:

1. CICIDS2017 and NSL-KDD datasets were selected in order to provide additional validation, cross-checking, and comparison of our techniques with other techniques studied in the literature.

2. N-BaIoT and IoTID20 datasets were selected because they are created specifically for IoT networks and our model is specifically designed to secure IoT

   In addition to the above-named objectives, we considered datasets that consists of Denial of Service (DoS) and botnet attack types.

3.5.1 CICIDS2017 Dataset

CICIDS2017 is a recent dataset generated by the Canadian Institute for Cybersecurity [170]. CICIDS2017 not only contains current network attacks, but it also meets all criteria for real-world attacks. Since its inception, the CICIDS2017 dataset has attracted researchers to analyse and design new algorithms and models [171]. This dataset contains labelled network flows, as well as CSV files for ML and DL (MachineLearningCSV.zip), the corresponding profiles, full packet payloads in PCAP format, and the labelled flows (GeneratedLabelledFlows.zip), which are publicly available for research studies [172]. The CICIDS2017 dataset's ML file (MachineLearningCSV.zip) consists of eight CSV files representing the internet traffic profile for five days, including benign and malicious traffic on each day. The dataset consists of attack information as five days of traffic data; Afternoon dataset on Thursday and Friday working hours are ideal for binary classification, and morning dataset on Tuesday, Wednesday, and Thursday are well suited for developing a multiclassification detection model [173]. However, it should be noted that the best detection model should be capable of detecting all types of attacks. To design a typical IDS, the traffic data from the entire day should be merged into a single dataset to be used by the IDS. The dataset contains 2830743 instances as well as 79 features (78 features + 1 for attack type labels) and fifteen class labels (1 benign and 14 malicious). As described by the founders of the dataset, it is collected from the Canadian Institute of Cybersecurity in 8 separate files comprising 5 days of benign and malicious traffic data. [171, 174]. The CICIDS2017 dataset, which is publicly available, consists of a variety of malware attacks that is based on the 2016 McAfee report, such as the bot, brute force, DOS, DDOS, heart-bleed, infiltration, scan, and Web-based attacks. Interestingly, no duplicate instances were discovered. The following tables show a detailed description of files containing the CICIDS2017, overall characteristics, and class occurrences:

Table 3-1: CICIDS2017 dataset file description

File Name (csv)	Attacks	Flow count
Monday-WorkingHours.pcap_ISCX	No Attack	529918
Tuesday-WorkingHours.pcap_ISCX	Normal, FTP-Patator, Secure Shell (SSH)-Patator	445909
Wednesday-workingHours.pcap_ISCX	Normal, DoS Golden eye, DoS Slowhttptest, DoS slowloris, Heartbleed	692703
Thursday-WorkingHours-Morning-WebAttacks.pcap_ ISCX	Normal, Web Attack – Brute Force, Web Attack – SQL Injection, Web Attack – Cross site scripting (XSS)	170366
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ ISCX	Normal, Infiltration	288602
Friday-WorkingHours-Morning.pcap_ISCX	Normal, Bot	191033

Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX	Normal, PortScan	225745
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX	Normal, Distributed denial of service (DDoS).	286467

Table 3-2: Class instance occurrences in CICIDS2017 dataset

Class Label	Flow Count	Class Label	Flow Count
BENIGN	2273097	DoS Slowhttptest	5499
DoS Hulk	231073	Bot	1966
PortScan	158930	Web Attack – Brute Force	1507
DDoS	128027	Web Attack – XSS	652
DoS GoldenEye	10293	Infiltration	36
FTP-Patator	7938	Web Attack – Sql Injection	21
SSH-Patator	5897	Heartbleed	11
DoS slowloris	5796		
Total		**2830743**	

Table 3-3: Overall characteristics of CICIDS2017

Type of Dataset	Multi-class
Release Year	2017
Number of distinct instances	2830743
Number of Features	79
Number of distinct classes	15

3.5.2 NSL-KDD Dataset

The NSL-KDD is a publicly available dataset derived from the previous KDD-99 dataset [175]. A statistical analysis of the KDD99 dataset revealed significant issues that have a significant impact on the accuracy of intrusion detection and can lead to an inaccurate evaluation of AIDS [176]. The training samples contain 125,973 instances, while the testing samples contain 22,544 instances. The size of the NSL-KDD dataset is sufficient enough to allow for the use of the entire data without the neccesity for random sampling. Table 3-4 displays NSL-KDD train and test data instances along with their class [177]. The 42 features consist of information about the various 5 types of network connections, and each instance is classified as either benign or one of 4 malicious attacks. The training dataset contains 23 classes, while the testing dataset contains 38 classes, which include 21 attacks from the training dataset, 16 novel attacks, and 1 normal class. The class label of instances in the dataset is classified into five major categories (Benign, Denial of service, Probe, user to roots (U2R), and roots to local (R2L)). This dataset has a significant number of features which can be employed to classify various types of attacks. The features in the NSL-KDD dataset are classified into four types (basic, traffic, host, and content features). The kinds of features present in the NSL-KDD dataset are as follows: 4 are binary, 3 are nominal, and thirty-four are continuous [178]. Table 3-5 shows the different types of features in the NSL-KDD dataset. However, this new version of the kdd99 dataset has its own set of issues, as it typically does not represent current real-world network. Despite a scarcity of publicly available datasets for network-

based intrusion detection, it could still be used as an efficient dataset to assist in research studies of evaluating several intrusion detection techniques [179].

Table 3-4: Number of instances in the NSL-KDD dataset

Train Set		**Test Set**	
Classes	Instances	Classes	Instances
Benign	67343	Benign	9711
Denial of Service	45927	Denial of Service	7458
Probe	11656	Probe	2421
Root to Local	995	Root to Local	2754
User to Root	52	User to Root	200
Total	125973	Total	22544

Table 3-5: Feature Types in NSL-KDD dataset

Types	**Features**
Nominal	Service, Flag, protocol_type.
Binary	is_guest_login, is_host_login, land, logged_in.
Numeric	su_attempted, num_root, num_file_creations,num_shells, num_access_files, num_outbound_cmds, count, srv_count, serror_rate,srv_serror_rate,rerror_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate, dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, duration, src_bytes, dst_bytes, wrong_fragment, urgent, hot, num_failed_logins, num_compromised, root shell.

3.5.3 N-BaIoT Dataset

The N-BaIoT dataset was generated in the year 2018 to address the shortcomings of freely accessible botnet datasets, especially for IoT, and was generated utilizing real-time internet traffic data from 9 commercial IoT devices listed in table 3-6.

Table 3-6: N-BaIoT dataset device types and model names

Type of Device	**Model Name of Device**
Baby Monitor	Philips B120N/10
Doorbell	Danmini and Ennio
Security Camera	Provision (PT-737E and PT-838), and SimpleHome XCS7(1002 and 1003) WHT
Thermostat	Ecobee
Web-cam	Samsung SNH 1011 N

The IoT sensor traffic was obtained on the local network utilising Wireshark in the central switch and infected by real botnets namely Mirai and BASHLITE which are two common IoT-based botnets with demonstrated malicious abilities [180–181]. Each file in the N-BaIoT dataset contains 115 statistically designed independent features derived from the raw network packet (pcap) files, in addition to the label of the class derived from the name of the (e.g., "normal" or "Transmission Control Protocol TCP attack"). Seven statistical measures namely:

mean, variance, count, magnitude, radius, covariance, and correlation coefficient were computed over five separate time windows. A time window is a predetermined timeframe or a present period during which information or data is extracted from the traffic. This dataset is suited for the design of state-of-the-art Intrusion Detection Systems (IDS) since it includes a time window. In summary, four features were extracted from raw network packet (pcap) files, which are the packet count, the jitter, the size of outbound packets alone, and the aggregated size of outbound and inbound packets. For each of these four features, three or more statistical measures were computed, yielding a total of 23 features. These 23 distinct features were computed over five different time windows to obtain the 115 characteristics in this dataset. All 115 features of the N-BaIoT dataset are self-explanatory, and a feature is generalised as follows: 'Header_ time-windows_ statistical variables.' For example, if L0.01, L0.1, L1, L3, L5 refers to time-windows 1 to 5 respectively, then feature number two 'F2' given as 'MI_dir_L5_mean' will depict the following information:

- Header description: MI_dir
- Computed Time window: L5
- Statistical measure: Mean

The 115 features are presented in the next section, However, the header of all the features are the abbreviated terms described in table 3-7.

Table 3-7: NbaIoT Description of Header Names

Name	Brief Description
H	Statistics that summarizes the recent traffic from the packet's host (IP)
HH	Statistics that summarizes the recent traffic going from the packet's host (IP) to the packet's destination host.
HH_jit	Statistics that summarizes the jitter of the traffic going from the packet's host (IP) to the packet's destination host.
HpHp	Statistics that summarizes the recent traffic going from the packet's host+port (IP) to the packet's destination host+port. Example: 192.168.4.2:1242 ->192.168.4.12:80
MI	Statistics that summarizes the recent traffic from the packet's Source MAC-IP

A total of 229,829 samples were used for N-BaIoT dataset. The normal and attack class samples are 13,113 and 216,716 respectively. The goal was to employ anomaly detection techniques to distinguish normal from attack traffic samples. The dataset could be useful for multiclassification models, since the attack sample data is categorized into ten attack classes performed by two botnets and one "normal" class to make a total of 11 classes. Tables 3-8 to 3-10 shows the Class instance occurrence in the N-BaIoT dataset.

Table 3-8: NBaIoT Binary Label Distribution

Type	No of Samples
Benign	13,113
Malicious	216,716

Table 3-9: NBaIoT Category Label Distribution

Type	No of Samples
Benign	13,113
Mirai	101,409
Gafgyt	115,307

Table 3-10: NBaIoT Subcategory Label Distribution

Type	No of Samples
Benign	13,113
Mirai.ack	27,188
Mirai.scan	9502
Mirai.syn	23,361
Mirai.udp	15,148
Mirai.udpplain	26,210
Gafgyt.combo	21,205
Gafgyt.junk	24,250
Gafgyt.scan	21,995
Gafgyt.tcp	23,755
Gafgyt.udp	24,102

3.5.4 IoTID20 Dataset

The IoTID20 dataset is a recent dataset that was proposed by the authors of [182], and can be assessed in [183]. IoTID20 dataset is originally created by implementing two smart home devices namely SKT NUGU (NU100) and EZVIZ Wi- Fi Camera (C2C Mini O Plus 1080P) which are connected to a home wireless router. All the other devices involved such as laptops, tablets and smartphones are connected to the same wireless network. The SKT NGU and EZVIZ Wi-Fi camera are Internet of things victim devices and all the other devices in the testbed are the attacking devices. The testbed environment for IoTID20 dataset is illustrated in the figure 3-5. The newly developed dataset, which includes more extensive network and flow-based features, was derived from 42 raw network packet files (pcap) collected at various time points using the adapter's monitor mode of the wireless network. These raw Pcap file is available at [182]. A CIC flowmeter [184] application was used to extract features from packet capture files to generate a CSV format of the IoTID20 dataset. The flow-based characteristics can be employed to analyse and evaluate a flow-based IDS. Every instance of the IoTID20 dataset is labelled and it contains eighty network features and 3 labelled features. The labelled features are binary, categories, and sub-categories. Tables 3-11 to 3-13 shows the number of normal and attack instances in IoTID20Dataset. The description of the packet files in the dataset includes 'normal' as benign traffic samples, and 'malicious traffic' for attack class samples such as ARP spoofing, DoS, MITM, Mirai and Scan. Figure 3-6 shows the taxonomy of the labelled features in IoTID20 our proposed dataset.

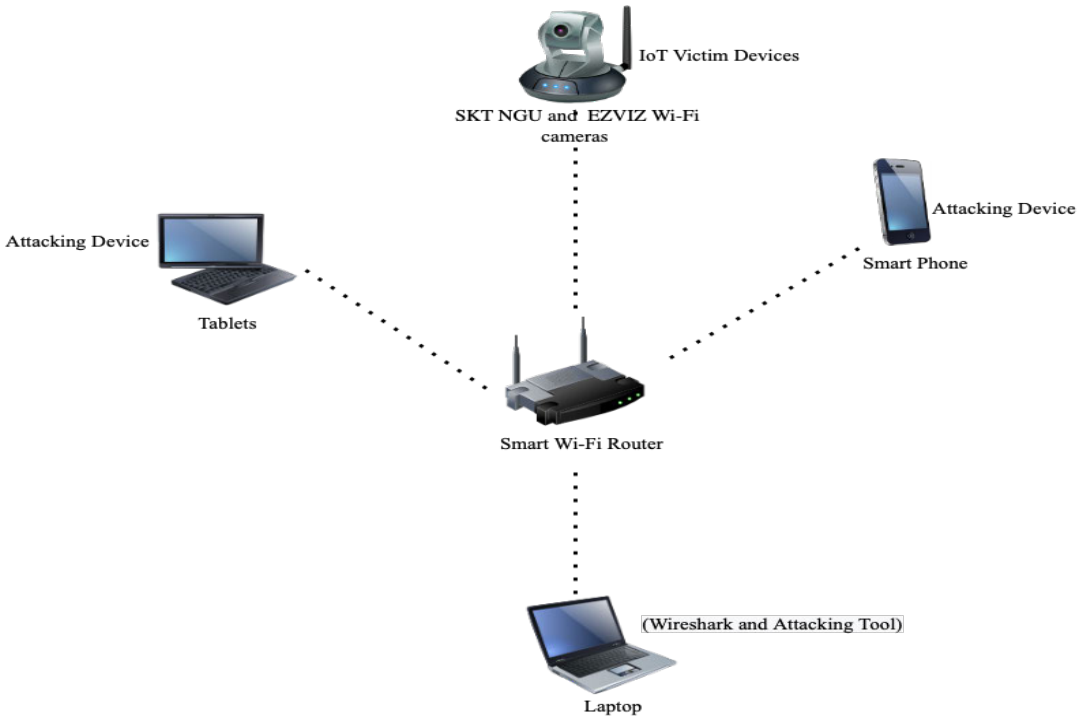


Figure 3-5: IoTID20 dataset testbed environment Adopted from [180]

Table 3-11: IoTID20 Binary Label Distribution

Class Label	**No. of Instances**
Benign	40,073
Malicious	585,710

Table 3-12: IoTID20 Category Label Distribution

Class Label	**No of Instances**
Benign	40,073
Denial of Service attack	59,391
Mirai	415,677
Man in the Middle attack	35,377
Scan	75,265

Table 3-13: IoTID20 Subcategory Label Distribution

Class Label	**No of Instances**
Benign	40073
Denial of Service	59391
Mirai Ack Flooding	55124
Mirai Brute force	121181
Mirai HTTP Flooding	55818
Mirai UDP Flooding	183554
MITM	35377
Scan Host Port	22192
Scan Port OS	53073

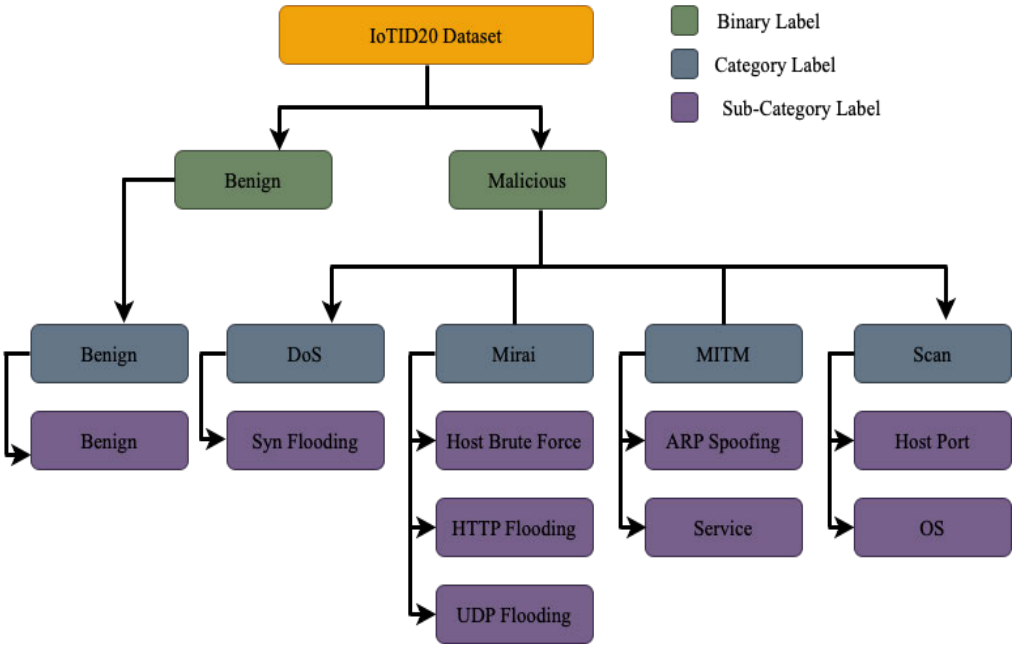


Figure 3-6: Taxonomy of Labelled Features in IoTID20 Dataset

The most significant advantages of the IoTID20 dataset is that It emulates a modern IoT network trend. communication. It was released recently in the year 2020 and It is one of the few publicly available datasets for IoT intrusion detection.

3.6. Handling Class Imbalance

To handle class imbalance problem of the datasets, we focused more on the Re-sampling technology which is categorized into over-sampling technique and under-sampling technique, or a combination of both. In oversampling methods, minority samples of the dataset are generated to achieve an almost balanced number of data samples in each category while in under sampling techniques, majority samples are discarded to achieve a rough balance in each category. Oversampling methods are mostly employed in the domain of Intrusion detection system because they have shown to produce better results on data imbalance problem [185]. There are several variations of over sampling techniques namely Random Over Sampling (ROS), Synthetic Minority Oversampling Technique (SMOTE), Borderline Smote, KMeans Smote, SVM Smote, Smote-NC (Nominal and Continuous), Adaptive Synthetic Sampling (ADASYN) etc. In this thesis, we focused on Random oversampling (ROS) and Tomek-Link under-sampling to solve the problem of data imbalance. The most basic form of oversampling to balance the imbalanced nature of datasets is random oversampling (ROS). It evens out the data by replicating minority-class samples. Consequently, there is no data loss due to this.

3.7 Selection of Non-Bias Features from Dataset

In this thesis, information gain technique was used for feature selection. It is a single-attribute evaluator that scores all the attributes based on their information gain when combined with ranker search technique. This is utilized to calculate the information gain in relation to the class label so as to determine the value of each attribute. The amount of information gained about the classes when that feature is used determines the score. IG was selected because it is a filtered-based method that, due to its robustness against overfitting, it offers more reliable sets of selected features. Also, because the computational complexity of filter-based technique is lower than that of embedded and wrapper-based methods. [186]. The complexity of wrapper-based methods raises the possibility of over-fitting. Therefore, employing a feature selection method that extracts a significant, relevant, and a smaller number of features with less computational complexities will decrease the time of execution of the classification methods utilized in the process of detecting attacks. The formula for Information Gain is expressed in Eq. 3.16,

$$IG(X) = H(Y) - H(Y|X) \tag{3.16}$$

where $H(Y)$ and $H(Y|X)$ represents the entropy of Y and the conditional entropy of Y for given value of X, respectively [187].

3.8 Summary and Conclusion

This chapter gives an overview of the classification algorithm and concepts that will be used extensively throughout this thesis. We define ML and DL concepts and then covered the concept of ensemble and its kinds. This chapter also introduces a stacking ensemble learning method for IDS. It is worth mentioning that this chapter does not provide an exhaustive coverage of all deep learning concepts. Instead, its focus lies on the essential concepts and methodologies required to comprehend the subsequent chapters of this thesis. In addition, a detailed discussion of the datasets utilized in this thesis was provided. These datasets were generated using different kinds of environment, labelled and used currently by the research community. They consist of at least two classes with a number of current attacks and their variations. These datasets are known to have flaws which were highlighted in this chapter.

CHAPTER 4: INVESTIGATION OF ML MODELS FOR INTRUSION DETECTION SYSTEM

4.1 Introduction

The previous chapters provided a comprehensive review of the approaches used in Intrusion Detection Systems. This review highlighted the advancements made so far, as well as identified certain limitations. The chapters also covered learning techniques and datasets in detail. This chapter presents our contribution to the development of IDS for Internet of things technology. All contributions have been published, accepted for publication or under review for possible publications in various international journals in the domain of network security.

4.2 Comparing Single and Ensemble IDS Using Several Datasets

4.2.1 Introduction

In this section an evaluation and comparison of some ML models was conducted. These models comprise of a feature extraction and classification. Comparison was based on four important performance metrics. Furthermore, the evaluated models were merged in an ensemble classifier to assess how well they performed. The research in this section was motivated by the shortcomings of previous works due to the presence of redundant and irrelevant features in evaluation dataset. The findings are thought to be relevant in the construction of IDS systems that combine strong classification models. The article was published in [193] while the datasets can be downloaded at [172], [176], [180], [182]

4.2.2 Simulation Settings and Methods

This research study was conducted using the Google Collaboratory with Python 3, TensorFlow, and a GPU that is installed on a Mac running OS X 2.8 GHz Intel Core i7 CPU, 16.00 GB RAM 2133MHz LPDDR3. The experimental methods include data pre-processing to select relevant attributes with Information gain technique. 20 attributes from each dataset were chosen based on threshold values of 0.290, 0.400, 0.730, and 0.420 for the datasets NSL-KDD, CICIDS2017, N-BaIoT, and IoTID20, respectively. Features with IG values are less than the threshold is excluded from the dataset. LR, SGD, LGBM, XGboost, and DNN algorithms were used for the comparison, these algorithms were individually analyzed with the data attributes that are pre-processed and results were obtained to check their individual performance. Then, we proposed an intrusion detection system based on the stacking ensemble learning technique that involves training the base algorithms with each input dataset and training the meta-learner with the outputs of each individual base classifier.

4.2.3 Results and Analysis

The figures and tables 4-1 and 4-2, shows how each technique performed overall in relation to the corresponding datasets. To begin evaluating the performance of the stacking ensemble technique proposed, we compared the

accuracy and F-score metrics of all the algorithms employed in this study, as shown in figures 4-1 and 4-2. In terms of accuracy and F-score, LGBM and XGBoost are the most effective learners.

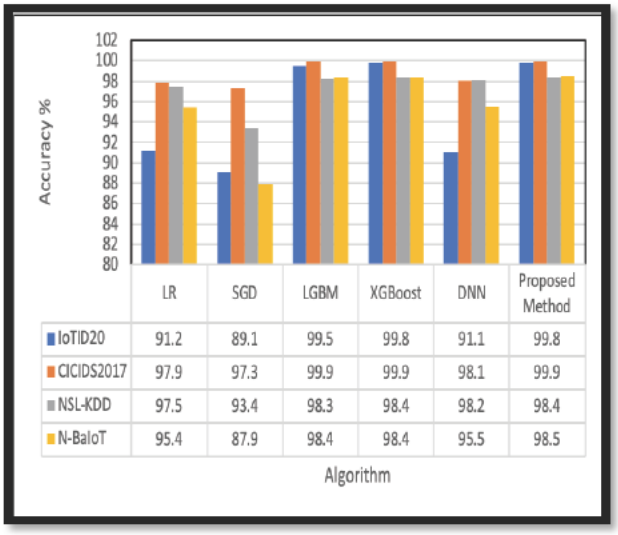


Figure 4-1: Performance comparison based on accuracy

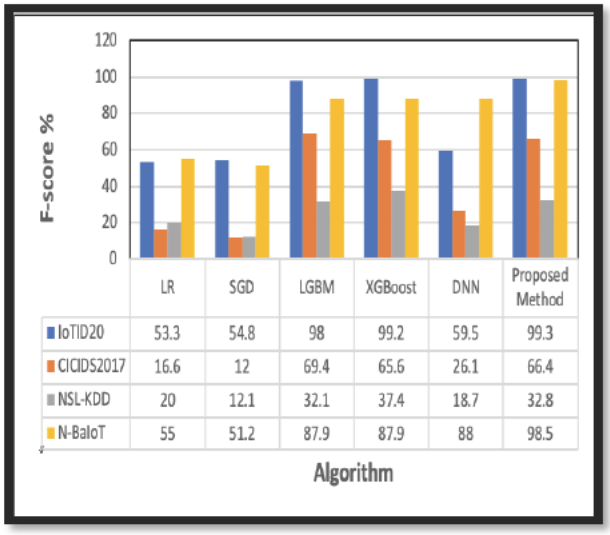


Figure 4-2: Performance comparison based on F-measure

It is shown in figures 4-1 and 4-2 that the logistic regression classifier outperforms the other algorithms in terms of accuracy and F-score when used as a meta classifier. It was discovered that the performance of these ML algorithms is dependent on the type of datasets used, as algorithms such as LGBM, XGBoost, and DNN performed better with N-BaIoT and IoTID20 datasets. Regardless, as shown in the table 4-1, our proposed method heterogenous Stacking ensemble outperformed all other algorithms.

Table 4-1: Performance results

Algorithm	**Performance on IoTID20 dataset**				**Performance on CICIDS 2017 dataset**			
	Mean Accuracy	**Mean Recall**	**Mean Precision**	**Mean F-measure**	**Mean Accuracy**	**Mean Recall**	**Mean Precision**	**Mean F-measure**
LR	91.2	52.2	60.4	53.3	97.9	15.7	19.2	16.6
SGD	89.1	55.8	55.8	54.8	97.3	11.2	15.1	12.0
LGBM	99.5	97.6	98.5	98.0	**99.9**	**68.8**	**70.8**	**69.4**
XGboost	**99.8**	**99.0**	**99.5**	**99.2**	**99.9**	66.2	65.2	65.6
DNN	91.1	49.1	91.1	59.5	98.1	22.1	41.4	26.1
Algorithm	**Performance on NSL-KDD dataset**				**Performance on N-BaIoT dataset**			
	Mean Accuracy	**Mean Recall**	**Mean Precision**	**Mean F-measure**	**Mean Accuracy**	**Mean Recall**	**Mean Precision**	**Mean F-measure**
LR	97.5	21.5	20.9	20.0	95.4	54.1	55.9	55.0
SGD	93.4	18.9	10.7	12.1	87.9	57.5	50.9	51.2
LGBM	98.3	34.3	31.7	32.1	**98.4**	**90.6**	86.6	**87.9**
XGboost	**98.4**	**39.3**	**37.2**	**37.4**	**98.4**	**90.6**	**86.7**	**87.9**
DNN	98.2	20.1	22.5	18.7	95.5	64.4	62.8	62.8

The obtained values of accuracy, precision, recall, and F-score when all the classifiers are tested with all datasets are shown in table 4-1. LGBM and XGBoost outperformed all other algorithms on IoTID20 datasets. SGD, on

the other hand, performed relatively poorly in terms of accuracy and precision, with 89.1% and 55.8%, respectively. Deep neural network classifier had the lowest recall performance of 49.1%, and LR had the lowest F-score of 53.3%. For CICIDS2017 dataset, Stochastic Gradient descent yielded the lowest performance. Whereas, LGBM outperformed all the other algorithms. For NSL-KDD and N-BaIoT, we also observed that the Stochastic Gradient descent had the lowest performance in all metrics except recall where LR had the lowest score for N-BaIoT dataset.

Table 4-2: Performance of Stacking Technique

Proposed Stacking Method	Mean Accuracy	Mean Recall	Mean Precision	Mean F-measure
IoTID20	99.8	**99.5**	**99.0**	**99.3**
N-BaIoT	98.5	86.6	90.6	88.0
CICIDS 2017	**99.9**	66.6	66.2	66.4
NSL-KDD	98.5	33.0	34.0	32.8

The table 4-2 shows the performance of the proposed stacking when evaluated with all experimental dataset. It had the highest F-measure of 99.30% and similar accuracy, recall, and precision to extreme gradient booting classifier. The model was successful in producing the ideal results comparable to the best algorithms for the IoTID20 dataset. However, when tested on the N-BaIoT dataset, the proposed stacking ensemble model outperformed the other algorithms considered for analysis in this work. As a result, it demonstrates how the new technique can be used to improve the performance of an IoT intrusion detection system.

4.3 RTL-DL: A Hybrid Deep Learning Framework for DDOS Attack Detection in Big Data Environment

4.3.1 Introduction

Among the most prevalent cybersecurity threats to the Internet of Things (IoT) is a distributed denial of service (DDoS) attack. To prevent DDoS attacks, numerous deep learning (DL) approaches have been used in intrusion detection systems. Unfortunately, their performance is hampered by the training dataset's high-class imbalance in addition to the existence of irrelevant and redundant features in them. The research conducted in this section is an improvement on the work done in Section 4.1. Consequent to the low precision and recall values obtained for all models, which was assumed to be caused by the high-class imbalance nature of the datasets used for the experiments. We propose a new framework named RTL-DL for an effective intrusion detection model. To reduce the effect of data imbalance observed in 4.1, the proposed framework employed the random oversampling and the Tomek-Links sampling methods (RTL). In addition, DDoS attacks were detected from the CICIDS2017 dataset which is a highly unbalanced IoT network traffic dataset using a deep learning (DL) model based on a deep neural network (DNN). The study demonstrated significant improvement in detecting network threats from an imbalanced dataset and justified the assumptions made in 4.1. The work done in this section has been published in [169], while the CICIDS2017 dataset can be downloaded at [172].

4.3.2 Research Gaps of Related Works in literature

Several related studies described in the previous chapters concentrated on detecting distributed denial of service attacks with deep learning methods, while others concentrated on the class imbalance challenges and the extraction of appropriate features from the benchmark datasets. A small number of studies addressed issues such as effective classification, data instability, and the presence of redundant and irrelevant features in the dataset. Further to a brief review of related works summarized in the preceding subsection, the following research challenges need to be addressed, as shown in table 4-3.

Table 4-3: Research Gaps Summary

Research Gaps in Related Works	References
Lack of comprehensive dataset	[91-92], [86], [88-89]
The presence of skewed samples of DDoS attack instances in comparison to normal instances in existing datasets.	[91-92], [86].
Evaluating with offline datasets	[86-89]
Real-time defensive system deployment was not automated.	

The lack of a huge dataset, challenges of class imbalance, and the implementations of models in real-world networks are all issues that can be challenging. This research work was carried out in a big data scenario to address the challenges of unavailability of big data.

4.3.3 Experimental Methods

We describe the materials and methodology that employed for this research in this section. The RTL-DL model was evaluated using the CICIDS2017 dataset, which was discussed in Chapter 3. Relevant theories underlying deep learning techniques are also discussed. Normal and malicious samples were extracted from the CICIDS2017 dataset. Pre-processing and normalization of the extracted samples fall within the range of [0, 1]. The RTL module receives the normalized data and generates synthesized data samples of the minority classes. Then the most relevant attributes of the dataset are chosen with a feature extraction method known as IG [187]. A deep learning classifier was trained with samples selected by the RTL. K-fold cross-validation was employed in simulation experiments. The proposed model differentiates between malicious and benign attacks by using the sigmoid function and binary cross-entropy. The normalization was done with min-max scaler, and the studies were carried out on the Jupyter notebook version, the libraries and the hardware specification described in section 4.1.2.

4.3.4 Dataset Preprocessing

Given that distributed denial of service constitutes a form of denial of service attack, the focus of this section centers on detecting distributed denial of service attacks. For this purpose, samples of malicious DoS and DDoS were chosen from the CICIDS2017 dataset. Subsequently, modifications were made to these samples, involving adjustments to both malicious and benign traffic components, as illustrated in figure 4-3.

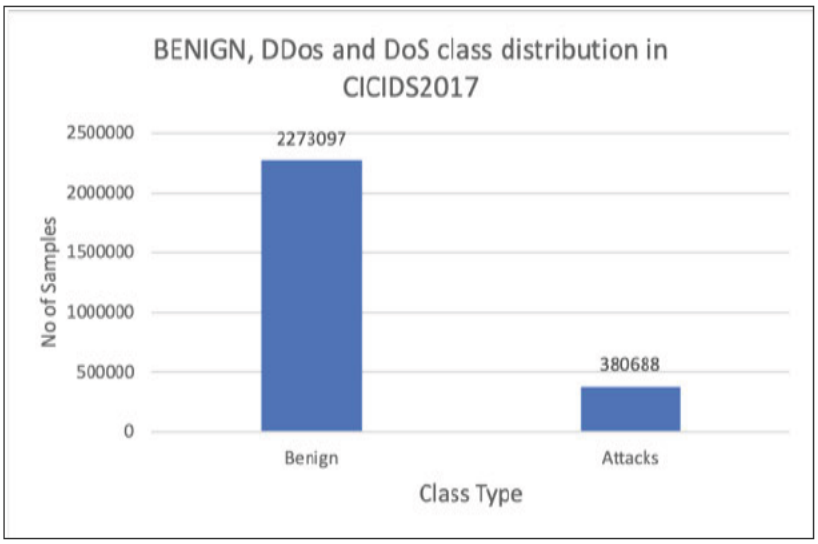


Figure 4-3: Extracted Samples in Dataset

4.3.4.1 Handling Class Imbalance

As described in the previous chapters, any trained model by a dataset that is not balanced may be skewed in favour of the majority samples [162-165]. As shown in Figure 4-3, CICIDS2017 dataset is imbalanced. To produce more samples of the minority (attack) class, the RTL (ROS + Tomek-Link) technique was employed in this study.

4.3.4.2 Random Oversampling

The simplest type of oversampling method is random oversampling (ROS), which is used to resolve the imbalanced characteristic of the CICIDS2017 dataset. ROS balances the dataset by recreating samples of the minority class. There is no loss of data as a consequence of this

4.3.4.3 The Tomek-Links

Data in the dataset that are identical but belongs to different classes are regarded to as sample pairs. Tomek links are the names given to these data pairs. The basic idea is to distinguish between the majority and the minority class samples. Let's assume a belong to one class and b belong to another, and a and b being the neighbours that are the closest, d (a, b); Assuming a certain distance separates a and b; Figures 4-4 and 4-5 shown depicts a pie chart of the actual samples and the generated samples by the RTL.

$$T\ (a, b) \text{ is a TomekLink if for any instance i;}$$

$$d\ (a, b) < d\ (a, i)\ or\ d\ (a, b) < d\ (b, i). \tag{4-1}$$

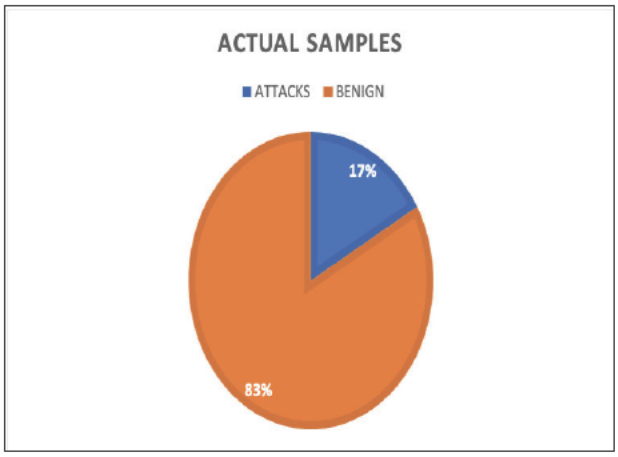


Figure 4-4: Actual Data Samples.

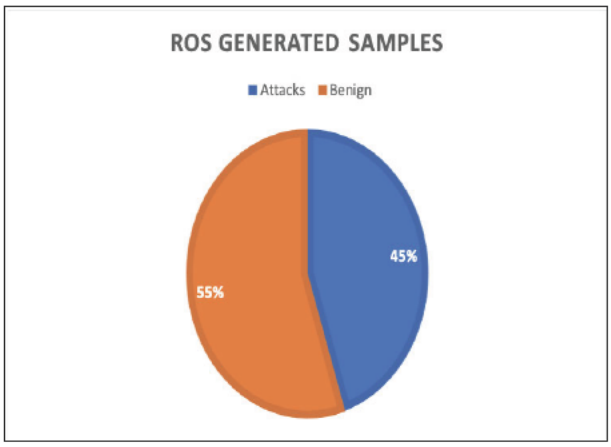


Figure 4-5: Generated Sythentic Samples

4.3.4.4 Data Partitioning

We used K-fold cross-validation because the CICIDS2017 dataset used for this study was not separated into train set and test set. 'K' represents 5-fold cross-validation. Figure 4-6 shows the number of malicious and normal classes in the training set and the testing set.

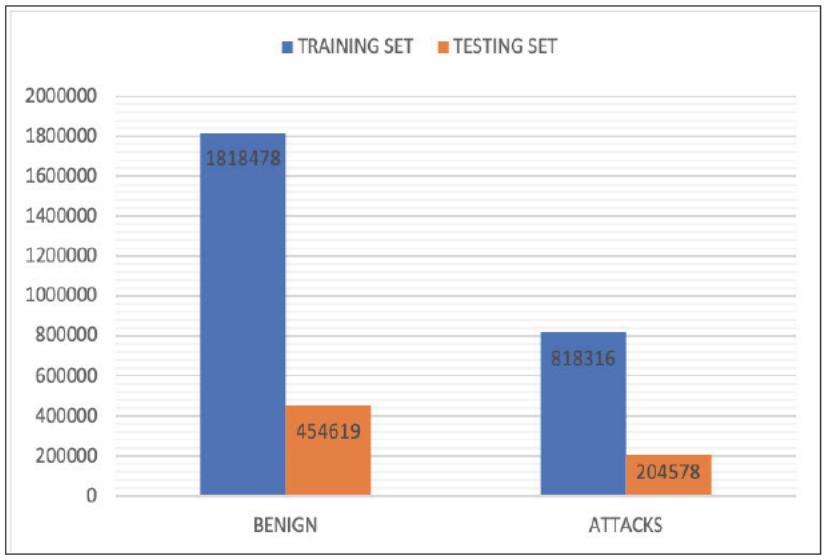


Figure 4-6: Training and Test Set.

4.3.4.5 Feature Selection

In this research, we selected significant features from the experimental dataset using an information gain feature selection approach. If a feature has a very low information gain score, it can be disregarded without affecting the model's accuracy [187]. The best feature subset is determined and chosen using the following algorithm:

Algorithm 4-3: Calculate Information Gain and extract best feature subset

Input	:	Train data with Z number of features	
Output	:	Select top n features as the new set of features	
1	:	Calculate IG: $IG(Y;X) = H(Y) - H(Y\|X)$	(4-2)
		Where: $H(Y) = -\sum_{i-1}^{n} P(Y = y_i) \log_2 P(Y = y_i)$	(4-3)
		$H(Y\|X) = -\sum_{i-1}^{m} P(X - x_i) H(Y\|X = x_i)$.	(4-4)
2	:	List the features according to their Information Gain score	
3	:	Compute Total IG = $\sum_{i=1}^{Z} IG(Y; x_i)$.	(4-5)
4	:	For every feature x, assign weights W (according to its IG with respect to the sum of IG of all features)	
		$W(x_i) = \frac{IG(Y;x_i)}{Total\ IG}$	(4-6)
5	:	Assign a value for threshold T and choose the top n features (Sum of weights of selected features ≥ the threshold value)	
		Start from $i = 1 (feature\ with\ the\ highest\ IG)$	
		$W_n = (W_n + W_{x_i})$	(4-7)
		If $W_n \geq T$	
		Proceed to next step 6	
		Else Increase and repeat the previous step 5	
6	:	Output top n features as the new feature set, n=i	

As shown in algorithm 4-3, X ($X_1 \ldots\ldots X_n$) is the parameter that indicates specific input features or attributes, and Z is the total number of features in the training dataset before Information Gain. Y is the variable that denotes the characteristics of the class (DDoS/benign: $Y_1 \ldots\ldots Y_n$). $P\ (Y = y_1)$ is the probability that the class attribute y occurs, the IG of attribute X is IG (Y; X), the entropy of Y is H(Y), and the mean conditional entropy of Y is H(Y|X). In this study, only 20 attributes were chosen when taking threshold values of 1.37 into consideration (where n=20, T=1.37). The threshold value T depends on the train set employed for the experiments. The deep learning model is then evaluated on the newly chosen features as shown in table 4-4.

Table 4-4: Selected Features

Description	**Information Gain**	**Description**	**Information Gain**
Average	2.010	Init_win_bytes_forward	1.370
Packet	2.000	Total	1.800
Packet	1.860	Max	1.850
Packet	1.860	Init_win_bytes_backward	1.630
Subflow	1.590	Fwd	1.760
Total	1.590	Flow	2.150
Bwd	1.380	Destination	1.380
Bwd	1.510	Flow	2.310
Average	1.500	Flow	1.940
Subflow	1.810	Fwd	2.260

4.3.5 Framework of the Proposed RTL-DL Model

A fully connected DL technique was developed from the scratch in this study. The back-propagation algorithm is the foundation of the proposed DNN model. Backpropagation starts at the top layer and moves back one layer at a time, calculating the error for each visited layer. In addition to one input layer and one output layer, the proposed model has four hidden layers. The number of input neurons in the RTL-DL model is the same as the number of the extracted features from the CICIDS2017 dataset in the overall framework's second component. As a result, the n = 20 input training sample is expressed as follows:

$$I = [i_1, i_2 \ldots\ldots i_n] \quad (4\text{-}8)$$

The next layer is the hidden layer h, that maps the input I from the input layer using random initialized weights w_a and a bias b_j. Therefore, hidden layer inputs can be written as follows:

$$h_j = \sum_a w_a i_a + b_j \quad (4\text{-}9)$$

In which $j = [1, 2, \ldots n_{th}]$ is the number of nodes that are hidden nodes in the DL model. Figure 4-7 depicts the flow diagram for the proposed deep learning model.

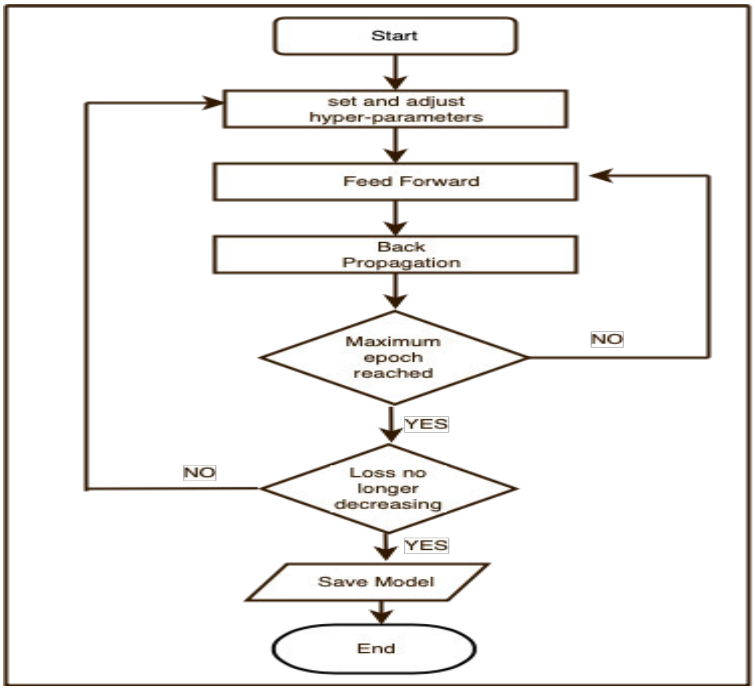


Figure 4-7: Flowchart of the model

The proposed model is trained in four stages, which are: initialization of the weights, feed-forward and back propagation phases, and weights and bias updates. The weights are initialized using the Bernoulli normal distribution, and the data samples of the training set is transmitted through all the 4 hidden layers

during the feed-forward phase to calculate the weighted inputs of each layer. At each hidden unit, the activation function σ is then applied to the sum of weighted input signals and passed to the next layer until it reaches the output unit. The output unit compares its calculated activation Y_o to its target value T during back propagation to determine the error at that unit. The error δo is calculated and returned to all previous layer units. Likewise, at each hidden unit h_j, error δ_j is calculated and sent to the previous layer. Utilizing the error function δ and the activation function σ, the weight and biases are updated until the stop conditions are reached. Table 4-5 shows the hyper-parameters that were employed when training the model:

Table 4-5: Description of the Hyper-parameters

Description	Value
Objectives	Binary classification
Input vector	20
Hidden layers	4
Neurons at hidden layer1-4 respectively	64,32,16,5
Output	1
Learning rate	0.001
Activation functions	ReLu, Sigmoid
Optimization	Adam optimizer
Loss Function	Binary cross entropy
Momentum	0.9
Weight initializer	Bernoulli normal distribution
Iterations	150

Framework of the Proposed Model

The RTL-DL model depicted in Figure 4-8 is composed of 3 main components: the RTL (ROS + Tomek-Link) unit, the IG unit, and the DNN classification algorithm.

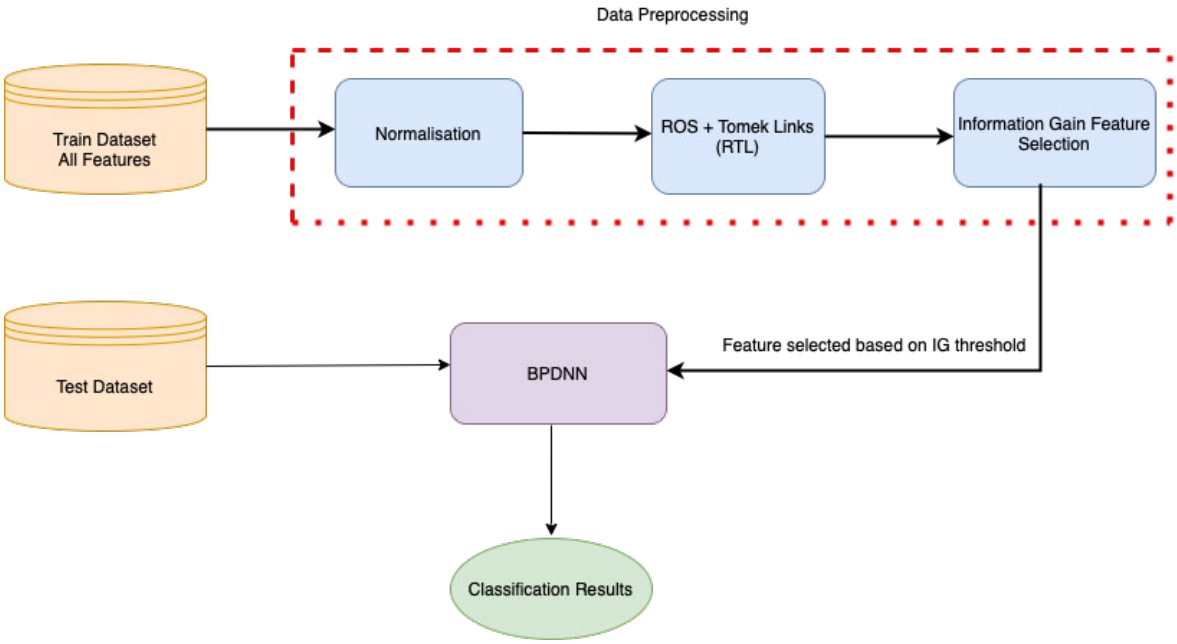


Figure 4-8: The RTL-DL Model Framework

The components described in figure 4-8 works in a sequential order. The RTL component generates acceptable minority class samples of the training data. The relatively balanced data generated by RTL is fed into the feature selection component. The feature extraction unit computes the information gain of all the features in the RTL generated data with the equation 4-2 which is illustrated in algorithm 1. Prior to applying the BPDNN algorithm, these features are ranked based on their information gain values, and the appropriate feature subset is chosen.

4.3.6 Results and Discussion

The table 4-6 displays the results of the RTL-DL model evaluation on the testing set.

Table 4-6: Obtained Results after Testing

Metrics	**Values (%)**
Accuracy	98.30
F-measure	97.80
Hamming Loss	4.60
Precision	98.80
Recall	98.30

RTL was applied to the samples extracted from the minority class in the original dataset during the research to verify the effectiveness of the RTL method for resolving class imbalance issues in the CICIDS 2017 datasets. This raises the number of instances in both the training and testing sets from 360,688 to 1,022,894. The use of 79 features in the generated samples increases accuracy by 19.2% and fscore by 66.3%, which is among the most crucial performance parameters to employ in a class imbalance challenge. In addition, to validate the performance of feature extraction in our study, information gain was applied to the samples generated by RTL component, and we obtained increases of (0.50, 8.30, 8.20, and 5.40) % for accuracy, precision, recall, and fscore, respectively, when the top 30 features were used. Additionally, when the top 20 features are utilised, there is a 0.1% increase, as shown in the table 4–7.

Table 4-7: Performance Results of The Study Conducted

	EXPERIMENTAL MODEL		**PROPOSED MODEL**	
Metrics	**DNN**	**RTL+DNN**	**RTL+IG+DNN**	**RTL+IG+DNN**
No. of Features	79	79	30	20
Accuracy	78.30%	97.50%	98.20%	98.30%
Precision	41.10%	90.50%	98.80%	98.80%
Recall	22.10%	90.10%	98.30%	98.30%
Fscore	**26.1**0%	**92.4**0%	**97.8**0%	**97.8**0%
Processing Time in secs	580s	406s	116s	103s

Furthermore, the time of processing was reduced significantly by using the RTL technique and IG technique, as illustrated in the figure 4-9.

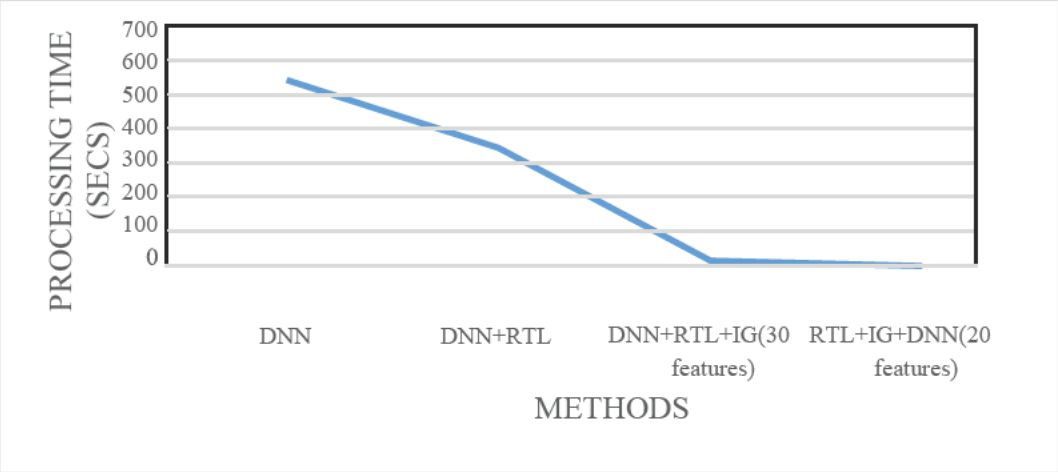


Figure 4-9: Processing times

4.3.6.1 Comparative Analysis

Table 4-8 compares the performance of the proposed RTL-DL to that of some state-of-the-art techniques discussed in the literature. In terms of accuracy, the proposed technique performed better than the DNN and RF techniques mentioned in [85] and [183], respectively. In addition, the RTL-DL model performed better than the KNN and LDA approaches in [90] as well as all of the techniques in [28]. Techniques discussed in [86–89] outperformed the proposed method. But even so, the model presented consists of data balancing and feature extraction methods, that were not taken into consideration in [86-89] and may have an impact on performance due to the imbalance nature of the classes in the datasets used for their study and the large number of irrelevant features contained in it.

Table 4-8: Comparative Analysis

Ref.	Dataset	Method	Accuracy (%)
[85]	CICIDS2017	DNN	98.0
[86]	CICIDS2017	Feed Forward DNN	99.610
[87]	CICDDoS2019	DNN	99.990
[88]	CICDDoS2019	Composite DNN	99.660
[182]	KDDCUP'99	SMOTE + Random Forest	92.570
[90]	CSE-CIC-IDS2018	SMOTE + (KNN, RF, GB, AB, DT,and LDA)	95.300, 99.190, 99.380, 99.320, 98.560 and 83.620
ProposedMethod	CICIDS2017	DNN + RTL	98.30

4.4 A Detailed Analysis and Feature Reduction of Big Data for Anomaly Detection in Internet of Things

4.4.1 Background

A dataset with a significant quantity of features will impact the resource usage, computational complexity and time required for analyzing data. Information Gain is a technique for selecting features employed in Intrusion Detection System (IDS) studies to tackle the data dimensionality problem. Despite the fact that many studies used IG as a technique for extracting features, there are very few explanations on how to avoid selection bias

and underestimating of any relevant feature. A detailed feature analysis of the NbaIoT and IoTID20 datasets to improve the detection accuracy of anomaly IDS, reduce amount of resources that are utillized, computational complexity and its execution time was conducted in this section. The major objective is to offer an overview of the dataset and to assess the impact of IG on a model's performance by comparing the model's performance while considering different feature groups selected based on their IG score. A trade-off was demonstrated between the number of features selected and the $F_{measure}$ metric, in such a way that the primary goal was to select a reduced number of features, keeping the $F_{measure}$ metric unchanged or at most insignificantly different at a reduced processing time. The study conducted in this section is an improvement on the study in Section 4.1 and 4.2. In both studies, relevant features were selected based on a given threshold value with no discussion on how to avoid selection bias and underestimating any relevant feature. The experiment findings reveal that the numbers of relevant features have a significant impact on detection accuracy and time of execution of IDS models. The study conducted in this section has been submitted for possible publication and it is currently under review in the International Journal of Applied Engineering and Technology.

4.4.2 Introduction

Though several anomaly detection methods have been designed, there are limitations and challenges because the datasets used to assess them influence the performance of these detection models. For example, some do have a high data dimensionality [189], which impacts computational complexity [190, 191], computational time [192]. Feature selection technique reduces computational complexity and improve predictive performance of machine learning models [149]. The most commonly used technique for selecting features in Intrusion Detection System (IDS) studies is information gain. In several studies, a given IG threshold value is considered and features whose IG values fall below the considered threshold value or has a very low IG score is removed from the dataset without affecting the accuracy of a model. For example, Khraisat et al. [99] identified 13 network traffic features. Likewise, in [193] the authors selected 20 features each from NSL-KDD, CICIDS2017, N-BaIoT, and IoTID20 datasets based on IG threshold values of 0.29, 0.40, 0.73, and 0.42 respectively. Authors in [97] and [194] used a feature score greater than 0.40 and greater than 0.0010, respectively. Meanwhile, studies in [195] considered a minimum score of 0.80. In contrast, the researchers in [154] extract features one at a time and use the classification model to determine the best accuracy. Following a review of the previously described studies, the authors suggested that a major limitation to the use of information gain is the removal of features based on a considered IG threshold value, this may lead to feature selection bias and underestimation of relevant features under some conditions which is a serious and tricky issue. For example, the inclusion or exclusion of a feature that falls below a given threshold because it doesn't affect the accuracy of a model is completely meaningless when dealing with highly imbalanced datasets. In such cases, an $F_{measure}$ metric would be more appropriate. In addition, the elimination of features one at a time, as in [154], is a procedure that is time consuming, especially when a significant number of features are present in the dataset. Finally, several researchers mainly use benchmark datasets like NSL-KDD, KDDCUP99, CICIDS2017 datasets, the use of huge IoT-specific datasets is still rare. As a result, the reliability

of the proposed approaches presented in this section has not been evaluated on larger IoT datasets such as IoTID20 and N-BaIoT.

4.4.3 Experimental Methods

This section discusses the classification method, data preparation and the details of experimenting with feature selection. A deep learning model was used for the classification. The DL model used in this section has two hidden layers: layer one has 64 neurons and layer two has 32 neurons. Rectified linear activation function (ReLu) is the activation function utilized at the hidden layers, while sigmoid was employed the output layer. The optimization algorithm used is adaptive moment optimizer (Adam) [196]. Parameters of the network were adjusted by passing a new training set into the network. The two network parameters that were tuned the learning rate and the number of iterations. As candidate parameters for the model, a range of learning rate which are [0.10, 0.010, 0.0010, and 0.00010] was selected. Measurement standard is the model's $F_{measure}$ on the verification set. The appropriate epoch numbers are obtained during the training of the model as the loss function value changes. It is concluded that training should end when peak performance is achieved. Otherwise, the performance value could deteriorate. 0.001 was selected as the learning rate for the model because it reaches the maximum $F_{measure}$ at that point. This is illustrated in figure. 4-10. Similarly, the number of iterations was set to 50 because the network stabilizes at 50^{th} iteration. Although parameter tuning was done using only NbaIoT dataset, we used same network parameters for both datasets. The model configurations are given in the table 4-9:

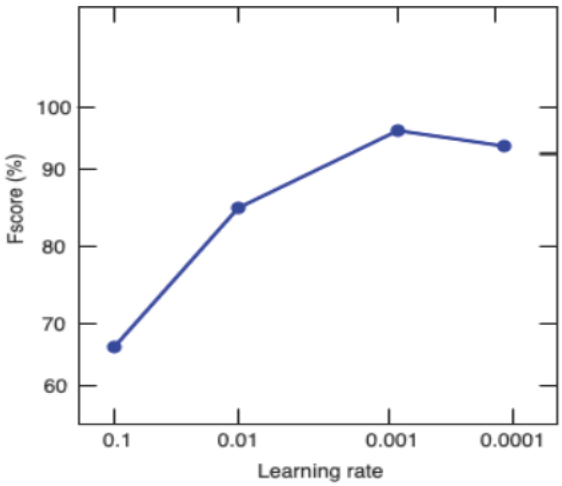


Figure 4-10: Parameter Tuning Performance

Table 4-9: Hyperparameter Description

Descriptions	Values
Objective	Binary classification
Input vector	27
Hidden layers	2
No of Neurons at hidden layer 1 and 2 respectively	64,32
Output	1
Learning rate	0.001
Activation functions	ReLu, Sigmoid
Optimization	Adam optimizer
Loss Function	Binary cross entropy
Momentum	0.9
Weight initializer	Bernoulli normal distribution
Epochs	50

All the studies were performed on the pre-processed NbaIoT and IoTID20 datasets. Each folder in the NbaIoT dataset contains benign and malicious samples for a specific device. We merged all the nine folders into a single csv file. Only 20% of the records in both databases were used in the experiment. There are no redundant features in the datasets, and we used the techniques described in [169] to minimize low detection accuracy and high false alarms due to the dataset's high-class imbalance. Using 10-fold cross-validation, we divided the

datasets into sections. Cross-validation was employed because it decreases processing time while maintaining the efficiency of the classification algorithm. As a result, the input dataset is divided into ten equal-sized folds at random. For each of the ten data sets, 9-fold is used in the training and 1-fold is utilized in the testing. We repeat the procedure ten times till every fold was used as a testing fold.

4.4.4 Selection of Features Using Information Gain

The Information Gain calculates the entropies of the features to evaluate them. Algorithm 4-4.1 shows the computation of feature rank and Tables 4-10 and 4-11 shows the feature description and the respective information gain values.

Algorithm 4-4.1: Feature Rank Computation

Procedure	:	Feature_Rank ()
Input F_N	:	A path to the main folder of training (N-BaIoT and IoTID20) datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively.
Output	:	Feature ID, $\mathbb{F}$; Feature Description, $\mathbb{D}$; Rank, $\mathbb{R}$; IG score $\mathcal{S}$;
1	:	Process $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{F}_3$, $\mathcal{F}_4$…… $\mathcal{F}_{nth}$ features of the training datasets $\mathcal{D}_1$ and $\mathcal{D}_2$
2	:	Compute IG value, $\mathcal{S}$ of each feature $\mathcal{F}_n$,
3	:	Rank each feature depending on their score.
4	:	Store $\mathbb{F}$, $\mathbb{D}$, $\mathbb{R}$ *and* $\mathcal{S}$
5	:	**End;**

Table 4-10: NbaIoT Dataset Feature Description and their respective information gain values

No	Feature Description	IG	No	Feature Description	IG	No	Feature Description	IG
1	MI_dir_L5_weight	1.387	41	HH_L3_magnitude	1.377	81	HpHp_L5_weight	0.244
2	MI_dir_L5_mean	1.740	42	HH_L3_radius	0.277	82	HpHp_L5_mean	1.0451
3	MI_dir_L5_variance	1.491	43	HH_L3_covariance	0.019	83	HpHp_L5_std	0.217
4	MI_dir_L3_weight	1.562	44	HH_L3_pcc	0.050	84	HpHp_L5_magnitude	1.183
5	MI_dir_L3_mean	1.806	45	HH_L1_weight	1.133	85	HpHp_L5_radius	0.028
6	MI_dir_L3_variance	1.566	46	HH_L1_mean	1.280	86	HpHp_L5_covariance	0.002
7	MI_dir_L1_weight	1.822	47	HH_L1_std	0.870	87	HpHp_L5_pcc	0.0
8	MI_dir_L1_mean	1.939	48	HH_L1_magnitude	1.430	88	HpHp_L3_weight	0.304
9	MI_dir_L1_variance	1.714	49	HH_L1_radius	0.605	89	HpHp_L3_mean	1.0454
10	MI_dir_L0.1_weight	1.957	50	HH_L1_covariance	0.064	90	HpHp_L3_std	0.218
11	MI_dir_L0.1_mean	2.155	51	HH_L1_pcc	0.151	91	HpHp_L3_magnitude	1.180
12	MI_dir_L0.1_variance	2.069	52	HH_L0.1_weight	0.135	92	HpHp_L3_radius	0.030
13	MI_dir_L0.01_weight	1.714	53	HH_L0.1_mean	1.401	93	HpHp_L3_covariance	0.0
14	MI_dir_L0.01_mean	2.229	54	HH_L0.1_std	1.117	94	HpHp_L3_pcc	0.001
15	MI_dir_L0.01_variance	2.162	55	HH_L0.1_magnitude	1.522	95	HpHp_L1_weight	0.326
16	H_L5_weight	1.387	56	HH_L0.1_radius	0.898	96	HpHp_L1_mean	1.046
17	H_L5_mean	1.739	57	HH_L0.1_covariance	0.094	97	HpHp_L1_std	0.376
18	H_L5_variance	1.489	58	HH_L0.1_pcc	0.312	98	HpHp_L1_magnitude	1.182
19	H_L3_weight	1.562	59	HH_L0.01_weight	1.159	99	HpHp_L1_radius	0.080
20	H_L3_mean	1.808	60	HH_L0.01_mean	1.465	100	HpHp_L1_covariance	0.0001
21	H_L3_variance	1.564	61	HH_L0.01_std	1.223	101	HpHp_L1_pcc	0.001
22	H_L1_weight	1.824	62	HH_L0.01_magnitude	1.575	102	HpHp_L0.1_weight	0.714
23	H_L1_mean	1.941	63	HH_L0.01_radius	0.975	103	HpHp_L0.1_mean	1.046
24	H_L1_variance	1.714	64	HH_L0.01_covariance	0.099	104	HpHp_L0.1_std	0.421
25	H_L0.1_weight	1.958	65	HH_L0.01_pcc	0.432	105	HpHp_L0.1_magnitude	1.185
26	H_L0.1_mean	2.155	66	HH_jit_L5_weight	0.975	106	HpHp_L0.1_radius	0.004
27	H_L0.1_variance	2.068	67	HH_jit_L5_mean	1.668	107	HpHp_L0.1_covariance	0.0
28	H_L0.01_weight	1.713	68	HH_jit_L5_variance	0.185	108	HpHp_L0.1_pcc	0.003
29	H_L0.01_mean	2.229	69	HH_jit_L3_weight	1.074	109	HpHp_L0.01_weight	0.709
30	H_L0.01_variance	2.163	70	HH_jit_L3_mean	1.675	110	HpHp_L0.01_mean	1.045
31	HH_L5_weight	0.972	71	HH_jit_L3_variance	0.214	111	HpHp_L0.01_std	0.481
32	HH_L5_mean	1.129	72	HH_jit_L1_weight	1.132	112	HpHp_L0.01_magnitude	1.178

33	HH_L5_std	0.748	73	HH_jit_L1_mean	1.712	113	HpHp_L0.01_radius	0.013
34	HH_L5_magnitude	1.365	74	HH_jit_L1_variance	0.317	114	HpHp_L0.01_covariance	0.006
35	HH_L5_radius	0.194	75	HH_jit_L0.1_weight	1.347	115	HpHp_L0.01_pcc	0.012
36	HH_L5_covariance	0.0	76	HH_jit_L0.1_mean	1.944			
37	HH_L5_pcc	0.035	77	HH_jit_L0.1_variance	0.841			
38	HH_L3_weight	1.074	78	HH_jit_L0.01_weight	1.158			
39	HH_L3_mean	1.156	79	HH_jit_L0.01_mean	2.028			
40	HH_L3_std	0.761	80	HH_jit_L0.01_variance	0.948			

Table 4-11: IoTID20 Description and their respective information gain values

S/no	**Feature Description**	**IG Values**	**S/no**	**Feature Description**	**IG Values**	**S/no**	**Feature Description**	**IG Values**
1	Src_Port	1.309	27	Fwd_IAT_Min	0.198	53	ECE_Flag_Cnt	0.0
2	Dst_Port	1.196	28	Bwd_IAT_Tot	0.684	54	Down/Up_Ratio	0.099
3	Protocol	0.244	29	Bwd_IAT_Mean	0.657	55	Pkt_Size_Avg	0.595
4	Flow_Duration	0.909	30	Bwd_IAT_Std	0.085	56	Fwd_Seg_Size_Avg	0.438
5	Tot_Fwd_Pkts	0.179	31	Bwd_IAT_Max	0.657	57	Bwd_Seg_Size_Avg	0.563
6	Tot_Bwd_Pkts	0.224	32	Bwd_IAT_Min	0.627	58	Fwd_Byts/b_Avg	0.0
7	TotLen_Fwd_Pkts	0.434	33	Fwd_PSH_Flags	0.0	59	Fwd_Pkts/b_Avg	0.0
8	TotLen_Bwd_Pkts	0.662	34	Bwd_PSH_Flags	0.005	60	Fwd_Blk_Rate_Avg	0.0
9	Fwd_Pkt_Len_Max	0.440	35	Fwd_URG_Flags	0.0004	61	Bwd_Byts/b_Avg	0.0
10	Fwd_Pkt_Len_Min	0.430	36	Bwd_URG_Flags	8.96e-05	62	Bwd_Pkts/b_Avg	0.0
11	Fwd_Pkt_Len_Mean	0.437	37	Fwd_Header_Len	0.335	63	Bwd_Blk_Rate_Avg	0.0
12	Fwd_Pkt_Len_Std	0.086	38	Bwd_Header_Len	0.723	64	Subflow_Fwd_Pkts	0.179
13	Bwd_Pkt_Len_Max	0.520	39	Fwd_Pkts/s	0.460	65	Subflow_Fwd_Byts	0.422
14	Bwd_Pkt_Len_Min	0.547	40	Bwd_Pkts/s	0.906	66	Subflow_Bwd_Pkts	0.225
15	Bwd_Pkt_Len_Mean	0.565	41	Pkt_Len_Min	0.511	67	Subflow_Bwd_Byts	0.654
16	Bwd_Pkt_Len_Std	0.092	42	Pkt_Len_Max	0.529	68	Init_Fwd_Win_Byts	0.012
17	Flow_Byts/s	0.632	43	Pkt_Len_Mean	0.571	69	Init_Bwd_Win_Byts	0.885
18	Flow_Pkts/s	0.922	44	Pkt_Len_Std	0.181	70	Fwd_Act_Data_Pkts	0.249
19	Flow_IAT_Mean	0.894	45	Pkt_Len_Var	0.182	71	Fwd_Seg_Size_Min	0.0005
20	Flow_IAT_Std	0.299	46	FIN_Flag_Cnt	0.0	72	Active_Mean	0.034
21	Flow_IAT_Max	0.900	47	SYN_Flag_Cnt	0.472	73	Active_Std	0.006
22	Flow_IAT_Min	0.788	48	RST_Flag_Cnt	0.0	74	Active_Max	0.034
23	Fwd_IAT_Tot	0.253	49	PSH_Flag_Cnt	0.0014	75	Active_Min	0.035
24	Fwd_IAT_Mean	0.235	50	ACK_Flag_Cnt	0.383	76	Idle_Mean	0.885
25	Fwd_IAT_Std	0.100	51	URG_Flag_Cnt	0.0019	77	Idle_Std	0.290
26	Fwd_IAT_Max	0.244	52	CWE_Flag_Count	0.0	78	Idle_Max	0.890
						79	Idle_Min	0.810

The features are then ranked and categorized according to their IG values. As a result, feature groups are formed, with each feature group containing a varied number of features, as indicated in the tables 4-12 and 4-13. We classified seven categories groups of features as new subset of features both NbaIoT and IoTID20 datasets.

Table 4-12: NbaIoT Dataset Feature Description and their respective information gain values

Feature Weights	No of Selected Features	New Feature Subset
≥ 2.0	9	11, 12, 14, 15, 26, 27, 29, 30, 79
≥ 1.8	18	5, 7, 8, 10, 11, 12, 14, 15, 20, 22, 23, 25, 26, 27, 29, 30, 76, 79
≥ 1.6	27	2, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 20, 22, 23, 24, 25, 26, 27, 28 29, 30, 67, 70, 73, 76, 79
≥ 1.4	38	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 29, 30, 48, 53, 55, 60, 62, 67, 70, 73, 76, 79
≥ 1.2	44	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 29, 30, 34, 41, 48, 53, 55, 60, 61, 62, 67, 70, 73, 75, 76, 79
≥ 1.0	63	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 29, 30, 32, 34, 38, 39, 41, 45, 48, 53, 54, 55, 59, 60, 61, 62, 67, 69, 70, 72, 73, 75, 76, 78, 79, 82, 84, 89, 91, 96, 98, 103, 105, 110, 112
All	115	All features

Table 4-13: Selected Features of IoTID20 Dataset by information Gain

Feature Weights	No of Selected Features	New Feature Subset
≥ 1.0	2	1, 2
≥ 0.8	11	1, 2, 4, 18, 19, 21, 40, 69, 76, 78, 79
≥ 0.6	20	1, 2, 4, 8, 17, 18, 19, 21, 22, 28, 29, 31, 32, 38, 40, 67, 69, 76, 78, 79
≥ 0.4	36	1, 2, 4, 7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 19, 21, 22, 28, 29, 31, 32, 38,39, 41, 42, 43, 40, 47, 55, 56, 57, 65, 67, 69, 76, 78, 79
≥ 0.2	47	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 19, 21, 22, 23, 24, 26 28, 29, 31, 32, 37, 38, 39, 41, 42, 43, 40, 47, 50, 55, 56, 57, 65, 66, 67, 69, 70, 76, 77, 78, 79
≥ 0.1	53	1, 2, 3, 4, 5 6, 7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 37, 38, 39, 41, 42, 43, 44, 45, 40, 47, 50, 55, 56, 57, 64 65, 66, 67, 69, 70, 76, 77, 78, 79
All	79	All features

4.4.5 Results and Analysis

In this study, each feature subset is classified by the DNN classifier and the performance is evaluated using two (2) performance metrics: $F_{measure}$ and processing times. The Overall Procedure is shown in Algorithm 4-4.2;

Algorithm 4-4.2: Overall Procedure of Subset Selection

Procedure	**:**	Experimental Process ()
Input	**:**	F_r= Feature_Ranked data
Output	**:**	Subsets of Features, $\mathcal{F}_s$, $\mathcal{F}_{measure}$ and execution time, $\mathcal{T}$
1	**:**	Reduce no. of features of datasets $\mathcal{D}_1$ *and* $\mathcal{D}_2$ to n features based on IG score $\mathcal{S}$ of each feature F_r
2	**:**	For each F_r feature in Feature_Ranked data, choose feature with score, $\mathcal{S}$ and save on feature groups, $\mathcal{G}$
3	**:**	For $\mathcal{D}_1$; Groups $\mathcal{G}_1$ to $\mathcal{G}_7$ consists of features with scores $\mathcal{S} \geq 2.0, 1.8, 1.6, 1.4, 1.2, 1.0$ *and all features* respectively For $\mathcal{D}_2$; Groups $\mathcal{G}_1$ to $\mathcal{G}_7$ consists of features with scores $\mathcal{S} \geq 1.0, 0.8, 0.6, 0.4, 0.2, 0.1,$ *and all features* respectively
4	**:**	For each feature groups in $\mathcal{D}_1$ *and* $\mathcal{D}_2$; input the selected features into the DL Classifier, $\mathbb{C}$
5	**:**	Execute $\mathbb{C}$ on the input features.
6	**:**	Compute $\mathcal{F}_{measure}$, and execution time, $\mathcal{T}$
7	**:**	Compare the $\mathcal{F}_{measure}$, execution time, $\mathcal{T}$ of all feature groups, $\mathcal{G}_1, \mathcal{G}_2$ ……$\mathcal{G}_n$
8	**:**	End;

Tables 4-14 and 4-15 shows the performance the classification algorithm using the feature subset selected by IG over NbaIoT and IoTID20 dataset respectively.

Table 4-14: N-BaIoT dataset evaluation result with all feature groups

S/no	Feature Subsets	F-measure (%)	Exec. Time (secs)
1	9	94.59	68
2	18	97.30	193
3	**27**	**98.54**	**314**
4	38	98.45	392
5	44	98.32	452
6	63	98.05	584
7	All	98.03	1132

Table 4-15: IoTID20 dataset evaluation result with all feature groups

S/no	Feature Subsets	F-measure (%)	Exec. Time (secs)
1	2	92.25	19
2	11	93.75	112
3	**20**	**98.37**	**209**
4	36	98.56	375
5	47	98.60	409
6	53	98.60	457
7	All	98.41	673

Although, the lowest performances are obtained when only 9 features were used from the NbaIoT dataset. It was observed that the time of execution is proportional to the numbers of features used for all the categories of selected features. Furthermore, there is a significant increase in the $F_{measure}$ when 27 features are selected. Additionally, it is seen that when features in groups 4-7 were selected, $F_{measure}$ reduces and there is a huge increase in execution time as shown in table 4-14. Based on the study conducted in this section, the optimal set of features are shown in tables 4-16 and 4-17, and the figures 4-11 and 4-12 shows the information gain graph for all the datasets considered.

Table 4-16: Selected Features from the NbaIoT dataset

Feat. No	Feature Description	Feat. No	Feature Description
F2	MI_dir_L5_mean	F23	H_L1_mean
F5	MI_dir_L3_mean	F24	H_L1_variance
F7	MI_dir_L1_weight	F25	H_L0.1_weight
F8	MI_dir_L1_mean	F26	H_L0.1_mean
F9	MI_dir_L1_variance	F27	H_L0.1_variance
F10	MI_dir_L0.1_weight	F28	H_L0.01_weight
F11	MI_dir_L0.1_mean	F29	H_L0.01_mean
F12	MI_dir_L0.1_variance	F30	H_L0.01_variance
F13	MI_dir_L0.01_weight	F67	HH_jit_L5_mean
F14	MI_dir_L0.01_mean	F70	HH_jit_L3_mean
F15	MI_dir_L0.01_variance	F73	HH_jit_L1_mean
F17	H_L5_mean	F76	HH_jit_L0.1_mean
F20	H_L3_mean	F79	HH_jit_L0.01_mean
F22	H_L1_weight		

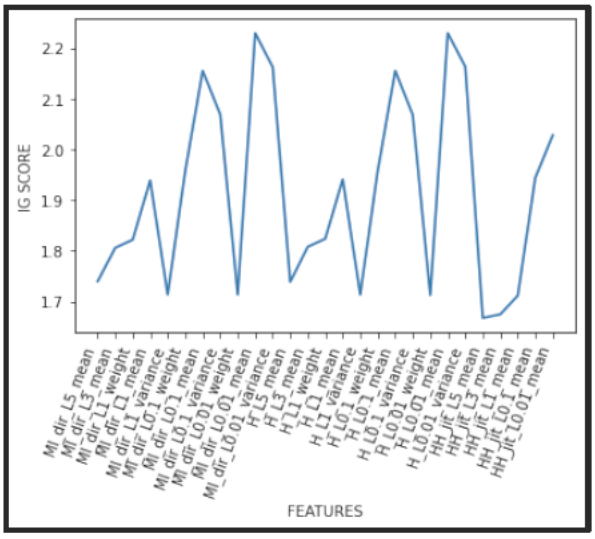


Figure 4-11: IG graph for N-baIoT dataset

Table 4-17: Selected Features from IoTID20 dataset

Feat. No	Feature Desc.	Feat. No	Feature Desc.
F1	Src_Port	F29	Bwd_IAT_Mean
F2	Dst_Port	F31	Bwd_IAT_Max
F4	Flow_Duration	F32	Bwd_IAT_Min
F8	TotLen_Bwd_Pkts	F38	Bwd_Header_Len
F17	Flow_Byts/s	F40	Bwd_Pkts/
F18	Flow_Pkts/s	F67	Subflow_Bwd_Byts
F19	Flow_IAT_Mean	F69	Init_Bwd_Win_Byts
F21	Flow_IAT_Max	F76	Idle_Mean
F22	Flow_IAT_Min	F78	Idle_Max
F28	Bwd_IAT_Tot	F79	Idle_Min

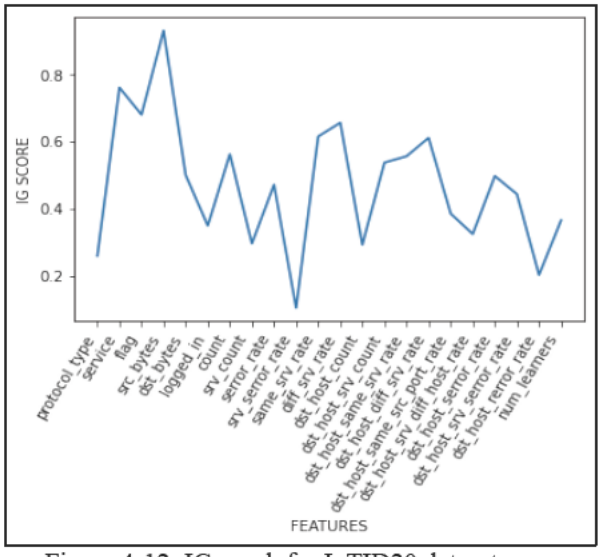


Figure 4-12: IG graph for IoTID20 dataset

4.5 Summary and Conclusion

Several studies to the development of IDS for IoT technology was conducted in this chapter. The first section starts by investigating the performance of some selected single, ensemble and deep learning techniques over traditional network datasets namely CICIDS2017 and NSL-KDD and IoT specific datasets namely IoTID20 and N-BaIoT'18 datasets. It also introduced an ensemble learning method based on stacking scheme for intrusion detection classification. The information gain feature extraction method was used to extract 20 relevant features from each dataset, and the techniques were compared using four common metrics. According to the results achieved in section 4.1, it was concluded that an efficient IDS can be designed with better feature selection, but identifying the most relevant features in an intrusion dataset that have a substantial impact on the outcome of an IDS is rather challenging. Also, as demonstrated in the same section, the ensemble technique outperformed other single classifiers. The improved performance was due to the combination of outputs from the single classifiers. Although, the results achieved in section 4.1 are promising and the model generalized well on all datasets, it recorded low precision and recall values especially on CICIDS2017 and NSL-KDD dataset, forming the basis for the next framework in section 4.2. These low values results were due to the high-class imbalance nature of the experimental datasets. The research study presented in section 4.1 has demonstrated to be useful in gaining a better insight of how to design security solutions for Internet of Things application, and the proposed method could be employed for practical intrusion detection system application. In section 4.2, an efficient IDS for detecting DDoS attack traffic on an IoT platform with the use of a back propagation-based DNN was presented. The IDS framework was based on the RTL and IG techniques to address class imbalance challenges and obtain the most important features from the CICIDS2017 dataset. The comparison of the models presented demonstrated the significance of data balancing and feature extraction

methods in the development of cutting-edge models. This research obtained accuracy of 98.30%, precision was 98.80%, recall was 98.30%, f-score was 97.80%, and hamming loss was 4.60%. When it was compared to the existing methods, the proposed model showed results that are promising in detecting malicious threats from an imbalanced network traffic dataset. The study conducted in section 4.2 is only concerned with the detection of denial of service and the distributed denial of service attacks. Other forms of malicious threats would be researched as a future study in the next chapters. Furthermore, tuning of hyper-parameters will be considered in order to reduce the number of iterations and the training time that is required. Lastly, the integration of the proposed model with other ML approaches will be considered in the next chapters in order to detect more new types of malicious threats and assess these models on some other recent benchmark IDS datasets. The objective of the study conducted in section 4.3 is to identify and provide effective solutions to the shortcomings of recent IoT IDS dataset, analyze its features, identify and select the most relevant subset of features to the research community for the design a state-of-the-art anomaly detection system with low computational resources and execution time. The motivation of that study was due to the several studies on IDS evaluations that have been conducted utilizing a limited number of extremely old datasets with several shortcomings. This is still a challenging problem that needs to be researched extensively. Information gain feature selection technique and a DNN classifier was used to address the issues of huge number of irrelevant features, and the challenges of selection bias and underestimation relevant features in the datasets. Features of the dataset were categorized based on their information gain scores into several groups. Then, these groups are fed into the DNN classifier to detect anomalies. Performances were compared amongst each group in the dataset with the aim of validating the relevance and the significance of the feature groups selected to be able to design a lightweight, more intelligent and accurate detection model with a low computational time for internet of things. Experimental results confirm that deep learning models have a good ability to detect anomalies in a considerable amount of execution time when the best features are selected. The features selected by our approach in this section was used in the remaining chapters of this thesis.

CHAPTER 5: ENSEMBLE METHOD FOR INTRUSION DETECTION SYSTEM

5.1 Introduction

This chapter focuses on the selection of suitable classifiers to develop an efficient solution for preventing attacks in IoT networks. Our solution was further investigated for generalization to study its ability to adapt to the inevitable changes in IoT network traffic. All contributions have been published, accepted for publication or under review for possible publications in various international journals in the domain of network security.

5.2 Statistical Performance Assessment of Supervised Machine Learning Classifiers for Intrusion Detection System

5.2.1 Background

Several studies have shown that an ensemble classifier's effectiveness is directly correlated with the diversity of its members. However, the algorithms used to build the base learners are one the issues encountered when using a stacking ensemble. Given the number of options, choosing the best ones might be challenging. In the study conducted in this section, we selected some of the most extensively applied supervised machine learning algorithms discussed in chapter 3, and performed an evaluation in terms of well-known metrics and validation methods using two Internet of Things (IoT) intrusion detection datasets, namely N-BaIoT and IoTID20. Their performance was further examined by conducting Friedman and Dunn's statistical tests to study the significant differences between these classifiers. The primary goal of this research study is to encourage security researchers in the field of IoT to develop IDS using ensemble learning and to propose an appropriate method for selecting diverse base classifiers for a stacking-type ensemble. The study conducted in this section has been accepted for possible publication in the International Journal of Artificial Intelligence (IJ-AI), while the datasets can be downloaded at [180], [182]

5.2.2 Introduction

The combination of several classifiers has helped machine learning research develop over the past decade. However, selecting the appropriate ones to combine can be challenging, especially when employing an ensemble of the stacking type. Therefore, it is important to compare the performances of the best ML classifiers in ML studies. Due to the existence of outliers, feature distribution, or/and algorithm properties, an algorithm could perform well on one dataset while failing to get the same results on another when compared across various datasets [197]. As a result, comparing several algorithms to one another becomes rather challenging. Several studies on the performance evaluation of ML classifiers, particularly in network attack detection, have been conducted. The authors in [67] employed four supervised algorithms to develop an intrusion detection system. The authors achieved 75% accuracy and 79% recall using SVM. The model achieved 99% accuracy and 99% recall with the Random Forest Classifier. Almseidin et al. [68] used Random

Forest and Decision Tree to create IDS, with accuracies of 93.77 and 92.44 percent on the KDD dataset, respectively. Authors in [198] used Ensemble methods, Fuzzy C-Means, Nave Bayes, Radial Basis Function and SVM to build an IDS using the Kyoto 2006+ dataset. They obtained results that are promising in terms of accuracy with ensemble methods reaching 96.72%. Gradient boosted machine was suggested for an anomaly-based IDS by Tama et al. [110] using various datasets which are GPRS, NSL-KDD and UNSW-NB15, the proposed IDS's performance is assessed with hold-out and cross-fold techniques, and the optimal GBM parameters are obtained using grid search. The proposed method outperformed fuzzy classifiers and tree-based ensembles in terms of all the metrics considered. Kilincer et al. [102] conducted a thorough literature review in 2021 to compare the performance of SVM, KNN, and DT. The CSE-CIC-IDS2018, UNSW-NB15, ISCX-2012, NSL-KDD, and CIDDS-001 datasets were all used for the comparative analysis. Except for the UNSW-NB15 dataset, the study found that the accuracy of the models varied from 95% to 100%. DT consistently outperformed all other implemented models, irrespective of dataset. The ability to detect unknown threats is also a concern when evaluating the performance of the IDS. Additionally, making a choice about which algorithm is superior to others is also difficult. In the study conducted in this section, statistical validation of the performance outcomes was used to address the problems of superiority amongst classifiers. The Classifiers that exhibit better trade-off between the metrics considered and evaluation time are recommended for the design of IoT-specific anomaly-based IDS.

5.2.3 Experimental Setup

The algorithms employed to create the pool of classification models for this experiment are implemented in the Anaconda platform's Jupyter notebook version 6.0.3. Using the Automatic Machine Learning (AutoML) framework. with Python 3, Keras, and TensorFlow. To develop the most effective models, it utilizes the random grid search and model parameter tuning during the classifier generation stage. The random grid search was implemented in the scikit-learn package using Python 3. We utilized the technique proposed in [169] to handle class imbalance problems in the datasets. To determine how well these classifiers performed overall, we ran repeated experiments using hold-out and k-fold cross-validation methods. To generate a training and testing set for hold-out, the dataset sample was divided into a 67:33 ratio (67 percent training instances and 33 percent testing instances). Similarly, the value of k for k-fold cross-validation is considered to be 10. All evaluation results given in this section are the weighted average of outputs from ten iterations of every validation approach repeated to avoid bias. The classifier evaluation time is measured during the testing phase (it is the calculated time from the beginning of classification process till the end of classification process). Using the following command

```
start_time = time.time()
y_pred = clf.predict(X_test)
```

The performance of ML classifiers for the design of an IDS is evaluated in this work. The accuracy, precision, recall, and f-score of classifiers such as LR, SGDC, and DNN, RF, AB, XGB, GBM, and ETC, are all

measured. All classifiers, excluding DNN, have their hyper-parameters tuned via random search [199]. Using well-known statistical tests, the major differences between classifiers are statistically examined.

5.2.4 Results and Discussion

Firstly, the study examined the hold-out validation performance results. Figures 5-1 and 5-2 illustrates the results for the two experiments conducted for hold-out validation on N-BaIoT and IoTID20 datasets respectively.

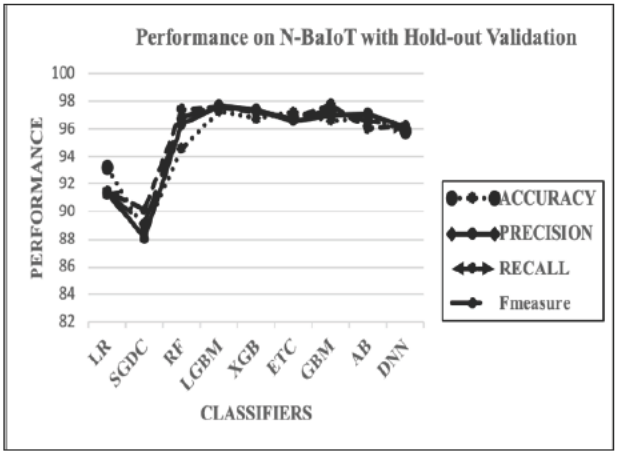


Figure 5-1: Results on N-BaIoT dataset

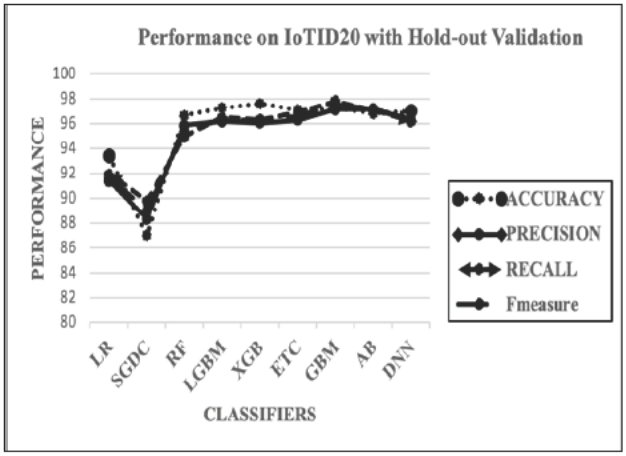


Figure 5-2: Results on IoTID20 dataset

The mean value of the performance achieved with hold-out validation across both datasets is shown in figure 5-3 which illustrates that classifiers outperform each other in terms of various metrics. For the ensemble algorithms, LGBM performs better than other ensembles when accuracy is considered with (97.31%), while adaboost performed better in terms of precision (97.18 %), GBM outperforms the others in terms of recall (97.82%), and F-measure outperforms the others (97.49%). Furthermore, when only single classifiers are considered, DNN outperforms LR and SGDC in terms of all metrics considered (96.42%, 96.23%, 96.25%, and 96.24%) for accuracy, precision, recall, and F-measure, respectively. SGDC, on the other hand, achieves the lowest values in terms of accuracy, precision, recall, and F-measure, with 87.84%, 88.21%, 89.99%, and 89.09%, respectively, among all classifiers considered.

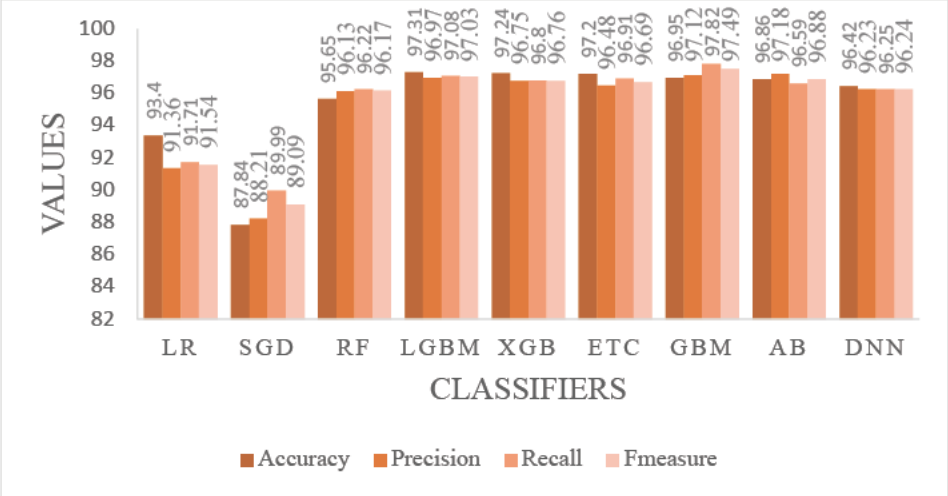


Figure 5-3: The average of each classifier's metrics with hold-out validation

Figures 5-4 and 5-5 illustrates the results for the experiments conducted for 10f validation on N-BaIoT and IoTID20 datasets respectively. The average values of all notable metrics obtained with 10f validation across both datasets are shown in Figure 5-6. Comparing the performance of the classifier with the hold-out validation, all utilized classifiers perform better with 10f validation. This is a result of sampling's effect, which selects random occurrences and results in unsatisfactory classification. The adoption of 10f validation rather than hold-out validation is encouraged by this phenomenon. The 10f validation results indicate that each classifier has a promising level of performance. DNN, on the other hand, outperforms all other methods considering accuracy with (98.76 %). Adaboost has the highest mean precision measure (98.30 %). With recall and F-measure scores of 98.83% and 98.53%, respectively, GBM performs best. Like the hold-out validation, SGDC still yields the lowest results when all metrics are taken into consideration.

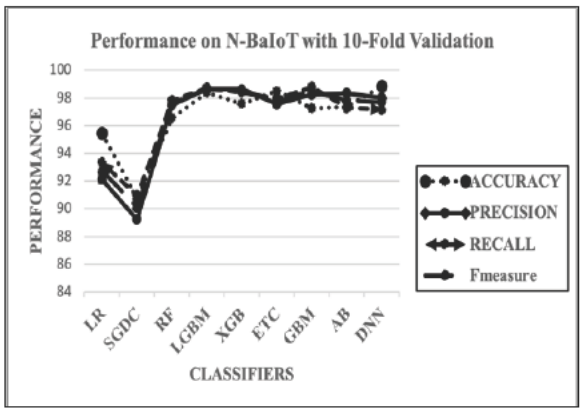


Figure 5-4: 10f Results on N-BaIoT dataset

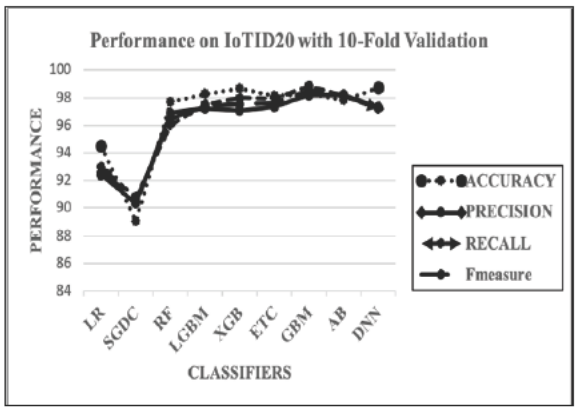


Figure 5-5: 10f Results on IoTID20 dataset

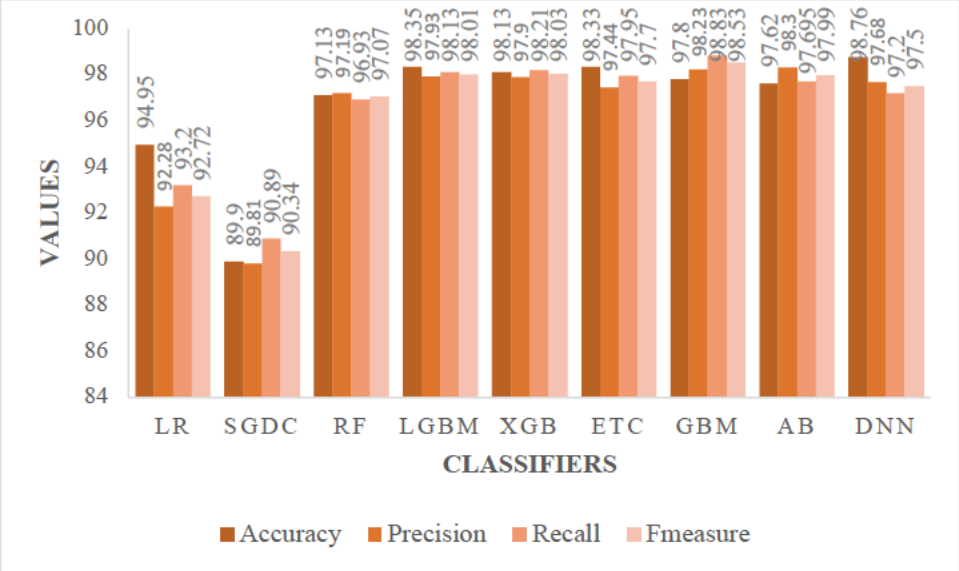


Figure 5-6: The average of each classifier's metrics with 10-fold validation

Classifier test times are listed in Table 5-1. Given that resource utilization is a key criterion for devices with limited resources, it is crucial to take a classifier's evaluation time into consideration because it assists in creating a suitable trade-off between a classifier's classification performance and resource utilization. On the N-BaIoT and IoTID20 datasets respectively, LGBM training takes about 11 and 3 seconds. For both datasets, DNN requires the most time for model evaluation. The test time of each classifier is generated solely for 10-fold validation.

Table 5-1: Classifier Test Time (seconds) across Both Datasets

Dataset	LR	SGDC	RF	LGBM	XGB	ETC	GBM	AB	DNN
N-BaIoT	50	43	14	11	27	15	20	29	106
IoTID20	37	28	8	3	9	7	13	17	97

5.2.4.1 STATISTICAL ASSESSMENT

The Friedman and Dunn post-hoc test is used to statistically analyze the performance results. The Friedman test can help determine whether one classifier performs better than the others by a significant margin across all datasets. If one of these classifiers is discovered, the pairwise multiple comparisons of Dunn's post-hoc test are performed to identify where that difference lies. Friedman test was chosen primarily because it is the most potent statistical test when more than five entities are being compared [200]. It is essential to undertake post-hoc tests, as proposed in [197] to identify differences in performance between classifiers. Due to the application of IoT in delicate industries like healthcare and financial institutions, a very low significance level set at (α=0.05) was chosen for this experiment. It is crucial to consider even the smallest difference in classifier performance. The null hypothesis (H_0) and the alternative hypothesis (H_A) are described in table 2. The values of d and k for the numbers of datasets and classifiers considered in this experimental study is 2 and 9 respectively. For $\alpha = \{0.05\}$ the values of degree of freedom $f1$ and $f2$ is 8, obtained from the formula, $f1 = k$

-1 & $f2 = (d-1)(k-1)$. The summary of the test hypothesis and the friedmann's test statistics for hold-out validation are given in tables 5-2 and 5-3 respectively.

Table 5-2: Hypothesis Test Summary

H_0	H_A	Test	sig. level	Asymp. Sig	Decisions
The classifier distributions are the same.	At least one Classifier different significantly from the rest.	Friedman's Two-Way Variance Analysis by Ranks for Related-Samples	0.05	Displayed	The null hypothesis is rejected.

Table 5-3: Friedmann's Test Statistics for holdout validation

N	4
Chi-square	28.533
df	8
Asymp. Sig	.000

As seen in the tables 5-2 and 5-3, the results show that the classifiers' performance is significantly different across all of the evaluated assessment metrics. As a result, it may be said that at least one classifier outperforms the others significantly. Therefore, the alternative hypothesis H_A is accepted, whereas the null hypothesis H_0 is rejected. The mean rank of each classifier used for hold-out validation is shown in the table 5-4.

Table 5-4: Mean ranks of Friedman test for hold-out validation

Classifier	Ranks
LR	2.00
SGDC	1.00
RF	3.00
LGBM	8.00
XGB	6.50
ETC	6.00
GBM	8.00
AB	6.50
DNN	4.00

Dunn's post-hoc test is used to determine which classifier pairs perform significantly differently. All pairwise comparisons' *p values* are checked against the considered significance level of 0.05 for this purpose. Assuming that C_1 and C_2 are the test classifiers, the results of Dunn's test (pairwise comparison) for hold-out validation are shown in table 5-5.

Table 5-5: Dunn's pairwise comparison for hold-out validation

$C_1 - C_2$	P-value	DECISION	$C_1 - C_2$	P-value	DECISION
SGDC-LR	0.606	A	LR-DNN	0.302	A
SGDC-RF	0.302	A	**LR-ETC**	**0.039**	**R**
SGDC-DNN	0.121	A	**LR-XGB**	**0.020**	**R**
SGDC-ETC	**0.010**	**R**	**LR-AB**	**0.020**	**R**
SGDC-XGB	**0.005**	**R**	**LR-LGBM**	**0.002**	**R**
SGDC-AB	**0.005**	**R**	**LR-GBM**	**0.002**	**R**
SGDC-LGBM	**0.000**	**R**	RF-DNN	0.606	A
SGDC-GBM	**0.000**	**R**	RF-ETC	0.121	A
LR-RF	0.606	A	RF-XGB	0.071	A
$C_1 - C_2$	P-value	DECISION	$C_1 - C_2$	P-value	DECISION
RF-AB	0.071	A	ETC-AB	0.796	A
RF-LGBM	**0.010**	**R**	ETC-LGBM	0.302	A
RF-GBM	**0.010**	**R**	ETC-GBM	0.302	A
DNN-ETC	0.302	A	XGB-LGBM	0.439	A

DNN-XGB	0.197	A	AB-LGBM	0.439	A
DNN-AB	0.197	A	XGB-AB	1.000	A
DNN-LGBM	**0.039**	**R**	XGB-GBM	0.439	A
DNN-GBM	**0.039**	**R**	AB-GBM	0.439	A
ETC-XGB	0.796	A	LGBM-GBM	1.000	A

According to the table 5-5, the classifiers for SGDC and LR vs (ETC, XGB, AB, LGBM, and GBM) are statistically and significantly different ($p < 0.05$), whereas the classifiers for RF-LGBM, RF-GBM, DNN-LGBM, and DNN-GBM pairs are less significant. Please take note that the pairs of classifiers that are red-shaded are not significantly different. Tables 5-6 and 5-7 shows the friedman test statistics and the mean ranks of each classifier respectively for 10-fold validation results.

Table 5-6: Friedmann's Test Statistics for 10-fold validation

N	4
Chi-square	25.133
df	8
Asymp. Sig	.001

Table 5-7: Mean ranks of Friedman test for 10-fold validation

Classifier	**Ranks**
LR	2.00
SGDC	1.00
RF	3.00
LGBM	7.25
XGB	7.00
ETC	5.50
GBM	7.75
AB	6.00
DNN	5.50

To determine which classifier performs statistically differently for 10-fold validation, Dunn's post-hoc test is used. The outcomes are displayed in table 5-8.

Table 5-8: Dunn's pairwise comparison for 10-fold validation

$C_1 - C_2$	**P-value**	**DECISION**	$C_1 - C_2$	**P-value**	**DECISION**
SGDC-LR	0.606	A	LR-DNN	**0.071**	A
SGDC-RF	0.302	A	**LR-ETC**	**0.071**	A
SGDC-DNN	**0.020**	**R**	**LR-XGB**	**0.010**	**R**
SGDC-ETC	**0.020**	**R**	**LR-AB**	**0.039**	**R**
SGDC-XGB	**0.002**	**R**	**LR-LGBM**	**0.007**	**R**
SGDC-AB	**0.010**	**R**	**LR-GBM**	**0.003**	**R**
SGDC-LGBM	**0.001**	**R**	RF-DNN	0.197	A
SGDC-GBM	**0.000**	**R**	RF-ETC	0.197	A
LR-RF	0.606	A	**RF-XGB**	**0.039**	**R**
$C_1 - C_2$	**P-value**	**DECISION**	$C_1 - C_2$	**P-value**	**DECISION**
RF-AB	0.121	A	ETC-AB	0.796	A
RF-LGBM	**0.028**	**R**	ETC-LGBM	0.366	A
RF-GBM	**0.014**	**R**	ETC-GBM	0.245	A
DNN-ETC	1.000	A	XGB-LGBM	0.897	A
DNN-XGB	0.439	A	AB-LGBM	0.519	A
DNN-AB	0.796	A	XGB-AB	1.000	A
DNN-LGBM	0.366	A	XGB-GBM	0.699	A
DNN-GBM	0.245	A	AB-GBM	0.366	A
ETC-XGB	0.439	A	LGBM-GBM	0.796	A

The classifiers are statistically and significantly different ($p < 0.05$) when comparing the classifiers SGDC to DNN, ETC, XGB, AB, LGBM and GBM, LR to XGB, AB, LGBM and GBM, and RF-LGBM, RF-GBM, and RF-XGB pairs. Please note the red-shaded to be the non-significantly different pair of classifiers for 10-fold validation.

5.3 Deep Stacking of Boosted Machines: An Intrusion Detection Technique for Internet of Things

5.3.1 Background

As discussed in chapters 1, 2 and 4, several machine learning algorithms have been successfully employed in intrusion detection and have been demonstrated to outperform traditional methods in detecting new attack trends, but there is still room for improvement in terms of the system's accuracy and efficiency. A single classifier, for example, may be incapable of detecting all forms of attacks. Therefore, the combination of various machine learning techniques to build more accurate classifiers has been the focus of many research studies recently. In this section, a stacking ensemble scheme for an intrusion detection model using three classifiers, namely extreme gradient boosting, light gradient boosting machine, gradient boosted machine as the base classifiers and a deep neural network model as meta classifier was presented. This technique was motivated by the promising results obtained from the selected models in Section 5.1. The proposed model was evaluated using records from all the nine devices in NbaIoT dataset. The results of the study conducted in this section has been accepted for publication in the International Journal of Computer Network and Information Security (IJCNIS)

5.3.2 Introduction

Numerous machine learning algorithms [201- 203] have been investigated and designed to increase the detection rate and performance of IDSs, these approaches have a significant disadvantage in that a single classifier may not be powerful enough to design an effective IDS. As a result, researchers developed the concept of ensemble classifiers for IDSs [204, 205]. Ensemble learning's primary objective is to merge a selection of independent models and then develop a more accurate classification decision about the input dataset [206]. For example, while training a single model on different subsets of an intrusion detection dataset may result in varying classification performances, an ensemble will average the output of numerous classifiers to become a better solution. As demonstrated by the authors in [193], it is an effective approach for rapidly improving the model's performance. However, using ensemble learning methods uncritically is not recommended. Different algorithms are appropriate for various classification scenarios. To significantly increase the overall effect of the model, the correct algorithm must be selected as the base classifier of an ensemble. Drawbacks of previous studies some of which have been stated in chapter 2 includes the detection of only a few malicious threats, insensitivity to unknown attacks, tuning a single algorithm to attain the highest performance on testing dataset, the use of limited performance parameters and the use of only one intrusion

detection dataset in the classification technologies of the methods. This reduces the reliability of those methods because it is unclear how some of these models performed with test data. Furthermore, utilizing only one dataset does not guarantee the model's effectiveness because attacks are very dynamic and varied, therefore, there is a possibility for a model to perform well on dataset A but not well on dataset B with additional or completely different attributes. This section proposes a framework using numerous high-performance models selected from the study conducted in 5.1 to develop a stronger stacking ensemble model named DSBM to overcome the aforementioned challenges by learning the benefits of each classifier. This study employs pre-processing technologies to deal with the dataset and opens up a new avenue of intrusion detection research.

5.3.3 Experimental Procedure

This section demonstrates the experimental set up of the study. The study conducted in this section are performed on the pre-processed NbaIoT dataset. The hidden layers consist of 64 and 32 number of neurons at hidden layers one and two respectively. Network parameters are adjusted by imputing a new training set, rectified linear activation function (ReLu) is the activation function used at the hidden layers whereas, sigmoid is the activation function used at the output layer. The optimization algorithm used for the DSBM network is adaptive moment optimizer (Adam) [196]. These base classifiers are trained independently in order to overcome different weaknesses and their predictions are given to a meta-classifier which finds the best combination of independently trained base classifiers to give final results. Through data pre-processing and hyperparameter tuning of the deep neural network model used as a meta-classifier, the best results are finally obtained. Parameter tuning methods presented in section 4.3 was adopted. The figure 5-7 is an illustration of the proposed model.

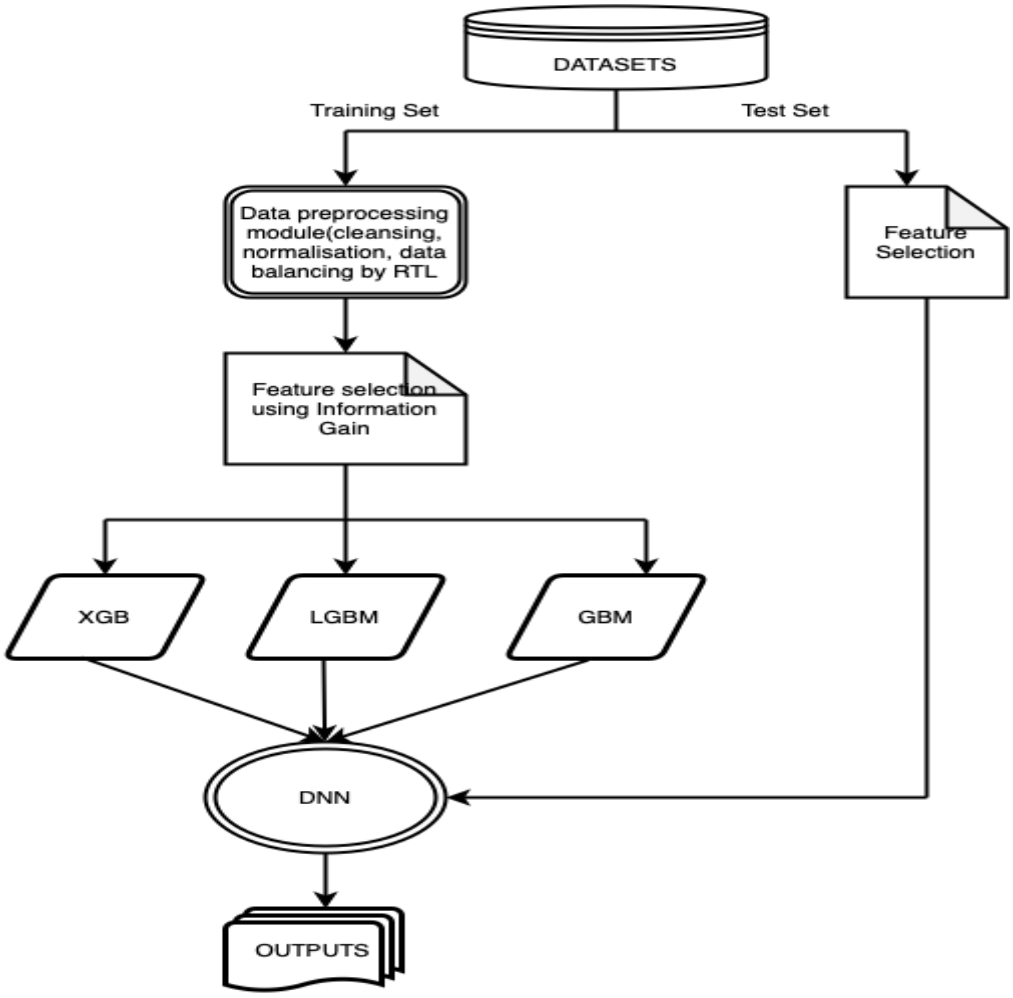


Figure 5-7: Proposed Model's Architecture

The training dataset is used to train the base classifiers, and a new dataset is generated for the meta-classifier. This new dataset is then used to train the meta-classifier, and the trained meta-classifier is then employed to predict the test dataset. The final proposed model will eliminate the individual shortcomings of all the base classifier, resulting in a robust model with enhanced model performance. The proposed model involves 5 steps described below:

1. Input the training data set for preprocessing (feature selection, imbalance data handling, data normalization)
2. Split the pre-processed data employing the 10-fold cross-validation and input into the base classifiers for training.
3. Build the proposed model by initializing the weights and training the network parameters of the DNN using a new training set which consists of the classification results of the train set and the original category.
4. Pre-process test data for prediction and discretization of results by trained base learners

5. The initial predictions of the base learners are inputted into the meta-learner to get the final classification results of the proposed model.

The proposed method involves two steps. The first step is the classification module, in which each classifier in the module has ten distinct model configurations based on 10-fold cross-validation which represents the whole train set. Each model predicts the results of 10 validation sets, which were superimposed to obtain the prediction results of a complete training set. These prediction results are used as the new training set for the next step and to assess the performance of the classifiers. Simultaneously, the data that needs to be predicted is inputted into each of the base models, and the new testing set to be inputted by the next step is the average of the ten models' predicted values. This process is clearly illustrated in figure 5-8.

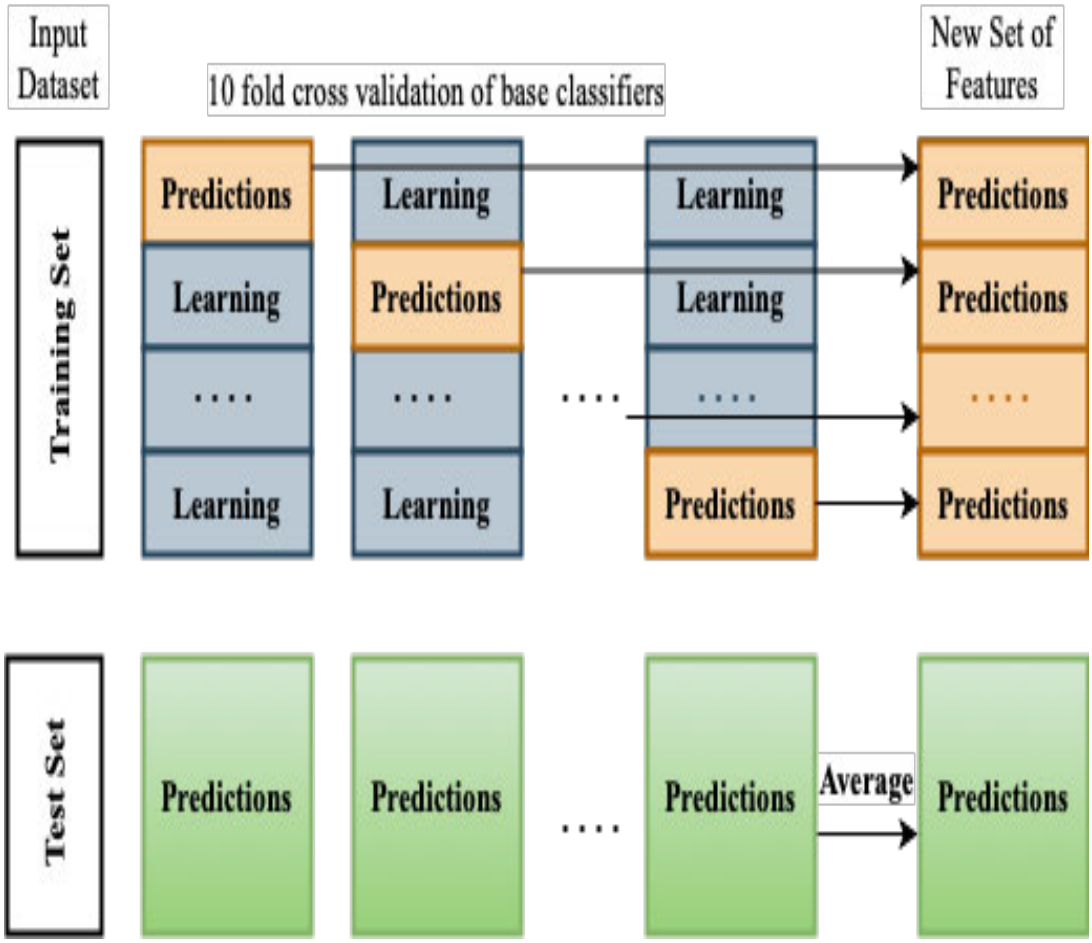


Figure 5-8: 10-fold cross-validation for new feature subset generation

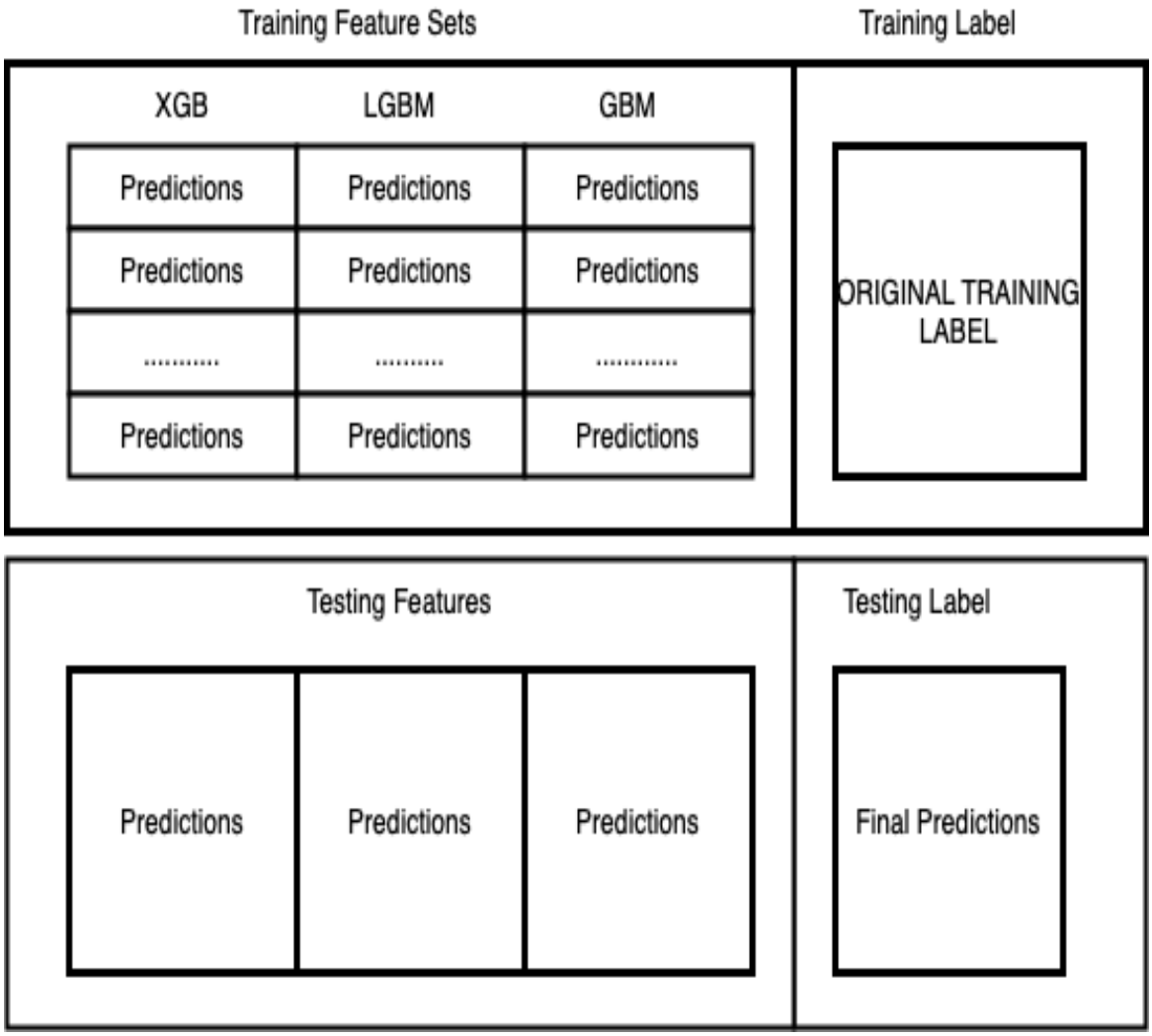


Figure 5-9: The new training and test sets' composition

Each base learner represents a new training and testing set. Prediction results are converted to discrete variables and 3-dimensional features from character variables and 1-dimensional characteristics, respectively, with one-hot encoding. Therefore, the predicted result for Mirai will become (0,1,0) to obtain training and test set features of 9 dimensions each. The combination of the training set's original classification label and the features of the new training set is the new training set's input, as illustrated in figure 5-9. The meta-learner layer is the second step in which a deep learning algorithm is used to integrate the base learners' decisions and generate the final stacked model. The deep learning model was trained using two hidden layers to analyze the deep relationships between the merged original classification label and the new training set's features. Unlike a stacking scheme that linearly combines the output of learners by majority voting or weighting strategies, which is not an ideal ensemble design [207], the use of deep learning as a meta-learner leads to a possible non-linear combination method to exploit the outputs of the base learners since neural networks are non-linear models.

5.3.4 Results and Discussion

Binary and multiclass classification were performed in this section. The N-baIoT dataset was categorized into normal and malicious samples in binary classification, while the malicious samples were further categorized during multiclassification. RTL sampling method presented in [169] was adopted to handle class imbalance problems in the dataset. A total of 59219 new samples were generated by the RTL module to balance the ratio of samples in the minority class to the majority class. Therefore, a total of 289,048 samples from all the 9 of the N-BaIoT dataset were used.

5.3.4.1 Results for Binary Classification

For binary classification, models were trained using records collected from all the 9 devices in the NbaIoT. Table 5-9 presents the precision, recall, and F1-score results for each ML model.

Table 5-9: Performance parameters on N-BaIoT dataset

MODELS	METRICS	CLASSIFICATION RESULTS (%)	
		Normal	Malicious
XGB	ACCURACY	98.11	98.01
	PRECISION	98.04	97.42
	RECALL	97.33	97.17
	F-measure	97.82	97.38
LGBM	ACCURACY	99.40	99.30
	PRECISION	99.32	98.62
	RECALL	99.39	99.23
	F-measure	99.33	98.89
GBM	ACCURACY	99.35	99.25
	PRECISION	99.13	98.43
	RECALL	99.30	99.14
	F-measure	99.02	98.56
PROPOSED MODEL	ACCURACY	**99.96**	**99.86**
	PRECISION	**99.86**	**99.12**
	RECALL	**99.76**	**99.60**
	F-measure	**99.82**	**99.36**

All the models in the table 5-9 are capable of classifying normal and malicious samples in the NbaIoT dataset with high accuracy. It could be observed that the DSBM model presented outperforms all other classifiers as illustrated by the bolded values in the table.

5.3.4.2 Results for Multiclass Classification

The normal samples and individual malicious samples injected into each device in the NbaIoT were classified into 11 categories for the proposed model's multiclass classification as presented in table 5-10:

Table 5-10: Multiclassification Performance of The Proposed Model

CATEGORIES	Results of Classification			
	ACCURACY (%)	PRECISION (%)	RECALL (%)	FMEASURE (%)
Benign	99.96	99.86	99.76	99.81
Gafgyt.combo	99.87	99.94	98.67	99.30
Gafgyt.junk	99.87	97.40	99.80	98.59
Gafgyt.scan	**99.98**	99.82	99.82	99.82
Gafgyt.tcp	99.89	99.83	99.85	99.84
Gafgyt.udp	99.97	99.91	**99.97**	**99.94**
Mirai.ack	99.56	96.38	99.65	98.00
Mirai.scan	99.97	99.87	99.95	99.91
Mirai.syn	99.97	**99.97**	99.89	99.93
Mirai.udp	99.72	99.65	99.00	99.32
Mirai.udpplain	99.72	99.78	97.05	98.40

As indicated in table 5-10, all 11 classes in the network traffic samples were accurately classified. 99.76% benign instances were accurately detected and 0.24% were wrongly classified as *Gafgyt.udp attack.* For the *gafgyt attack* category, 98.67% of the *combo* attacks were accurately classified while 1.33% were wrongly classified as *junk.* 99.80% of the *junk* attacks were accurately classified, while 0.20% were wrongly classified as *combo*. Furthermore, the model classified 99.8% of the *scan* attack samples correctly and wrongly classified 0.2% as *normal* network traffic. Almost all *udp* class samples in the *gafgyt* category were accurately classified. For the Mirai attack category, 99.6% of the ack attacks were accurately classified, while 0.4% were wrongly classified as udpp and udp attacks each. 99.9% of scan attacks were accurately classified while 0.1% were wrongly classified as udp attacks. Also, 99.9% of syn attacks were accurately classified while 0.1% were wrongly classified as scan attacks. 1% of the udp class samples were wrongly classified as *ack* attacks and 99% were accurately classified. Lastly, 97.1% of the udpp samples were accurately classified while there was a wrong classification of 2.3% and 0.6% as ack and udp attacks, respectively. We also recorded low execution times with 37.17s for training 260180 samples and 1.03s to detect malicious attacks in 28,909 traffic samples.

5.3.4.3 Device Type Binary Classification Results

In order to address the problems of using a single dataset for evaluation of IDS models as illustrated in the review conducted in section 2, we evaluated the proposed DSBM using data samples from 5 devices from the 9 devices in the NbaIoT which are Danmini Doorbell, Ecobee Thermostat, Ennio Doorbell, Philips B120N10 Baby Monitor, and Provision PT 737E Security Camera and attributed them as IoT 1-5 respectively. The results are presented in tables 5-11 to 5-13 and figures 5-10 to 5-25:

Table 5-11: Binary Classification Performance Results for IoT 1-4

Categories	Danmini Doorbell				Ecobee Thermostat			
	PRE	**REC**	**Fscore**	**Support**	**PRE**	**REC**	**Fscore**	**Support**
Benign	1.00	1.00	1.00	4000	1.00	1.00	1.00	3113
Malicious	1.00	1.00	1.00	3200	1.00	1.00	1.00	3200
Accuracy			1.00	7200			1.00	6313
Macro avg	1.00	1.00	1.00	7200	1.00	1.00	1.00	6313
Weighted Avg	1.00	1.00	1.00	7200	1.00	1.00	1.00	6313
Categories	Ennio Doorbell				Philips B120N10 Baby Monitor			
	PRE	**REC**	**Fscore**	**Support**	**PRE**	**REC**	**Fscore**	**Support**
Benign	0.83	1.00	0.91	4000	0.99	1.00	0.99	4000
Malicious	1.00	0.34	0.51	1200	1.00	0.99	0.99	3200
Accuracy			0.85	5200			0.99	7200
Macro avg	0.92	0.67	0.71	5200	0.99	0.99	0.99	7200
Weighted Avg	0.87	0.85	0.82	5200	0.99	0.99	0.99	7200

Table 5-12: Binary Classification Performance Results for IoT-5 (Provision PT 737E Security Camera)

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	**RECALL**	**FMEASURE**	
Benign	0.94	1.00	0.97	4000
Malicious	1.00	0.92	0.96	3200
Accuracy			0.96	7200
Macro-avg	0.97	0.96	0.96	7200
Weighted-Avg	0.97	0.96	0.96	7200

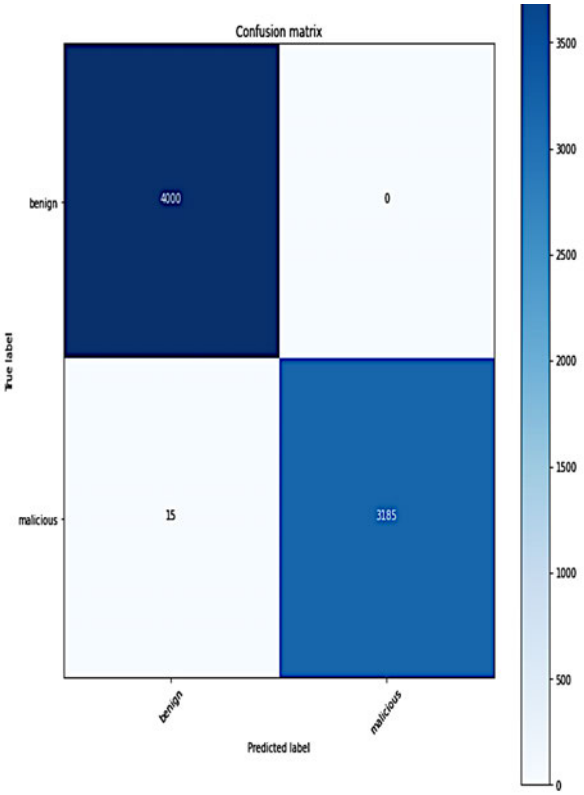


Figure 5-10: Binary Confusion matrix IoT-1

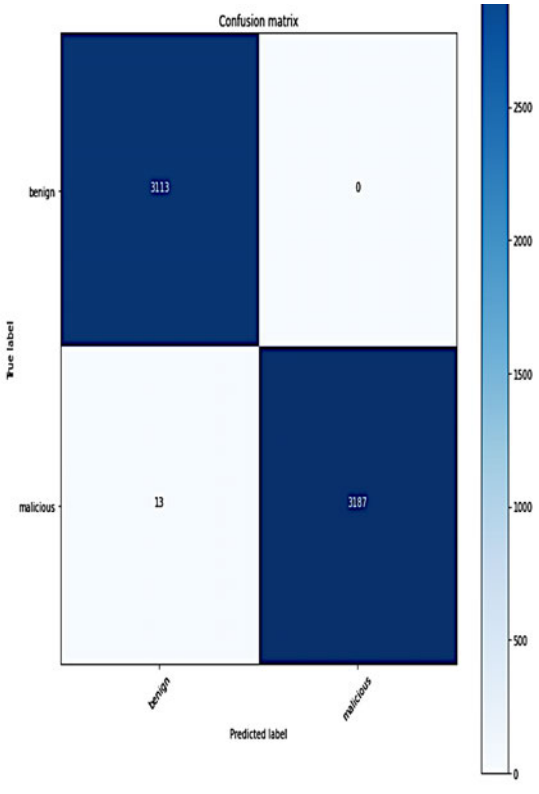


Figure 5-11: Binary Confusion matrix IoT-2

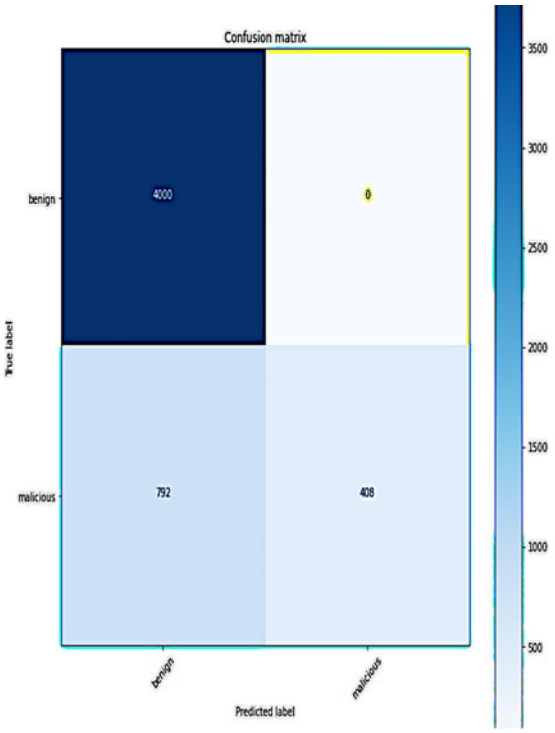


Figure 5-12: Binary Confusion matrix IoT-3.

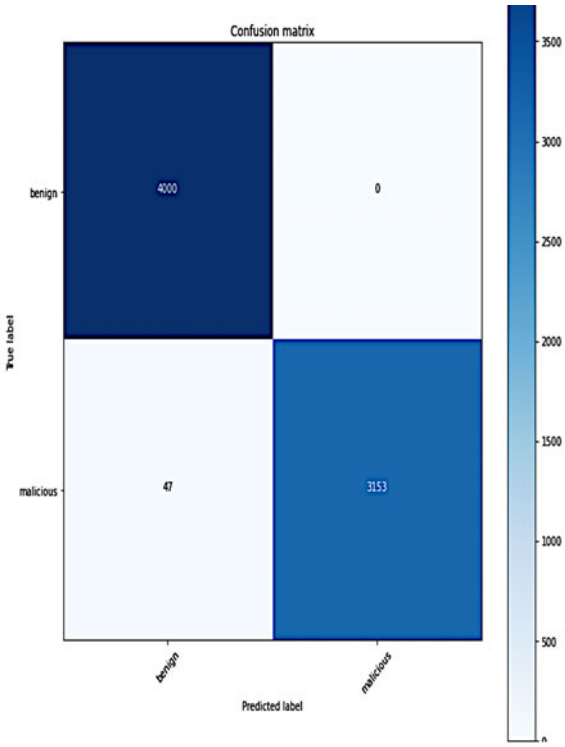


Figure 5-13: Binary Confusion matrix IoT-4

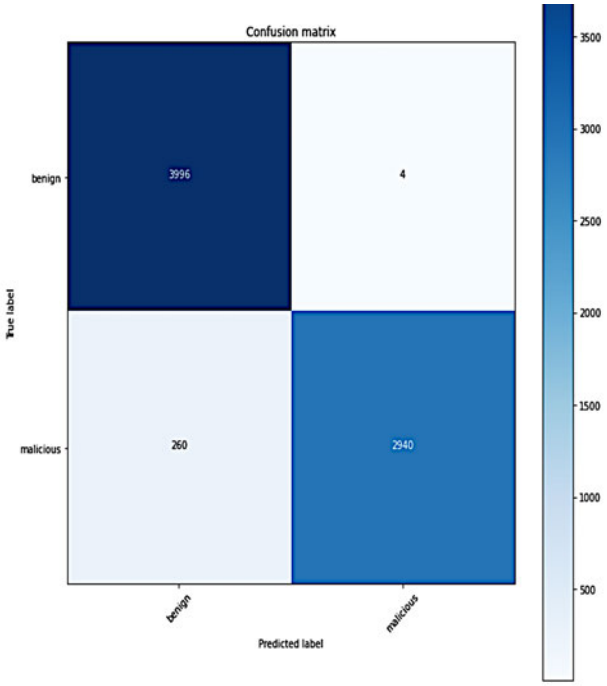


Figure 5-14: Binary Confusion matrix IoT-5

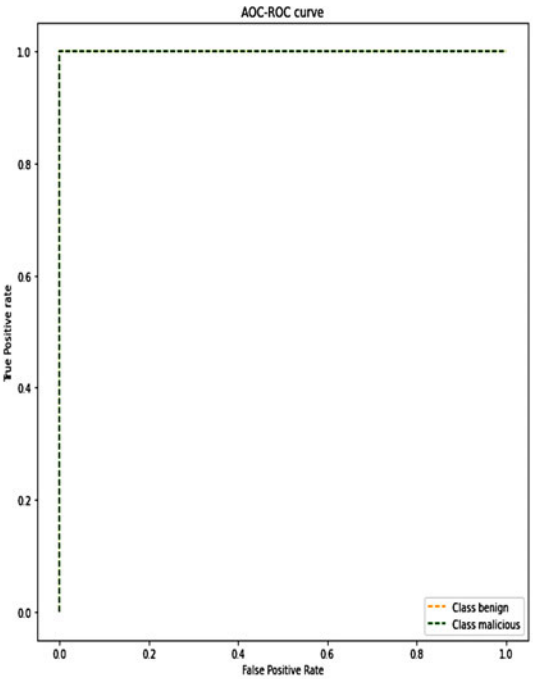


Figure 5-15: ROC-AUC curve IoT-1

Table 5-13: Binary Classification for IoT 1-5

Device	metrics			
	TRAINING (secs)	TESTING (secs)	KAPPA	MCC
Danmini Doorbell	139	0.22	0.9957	0.9957
Ecobee Thermostat	106	0.26	0.995	0.995
Ennio Doorbell	175	0.28	0.44	0.53
Philips B120N10 Baby Monitor	230	0.30	0.98	0.98
Provision PT 737E Security Camera	197	0.28	0.92	0.92

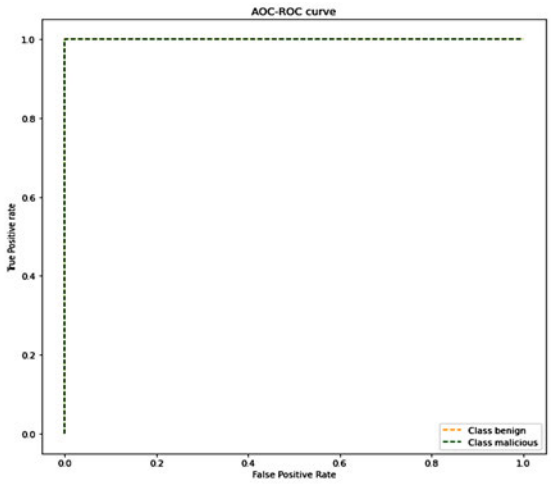


Figure 5-16: ROC-AUC curve IoT-2

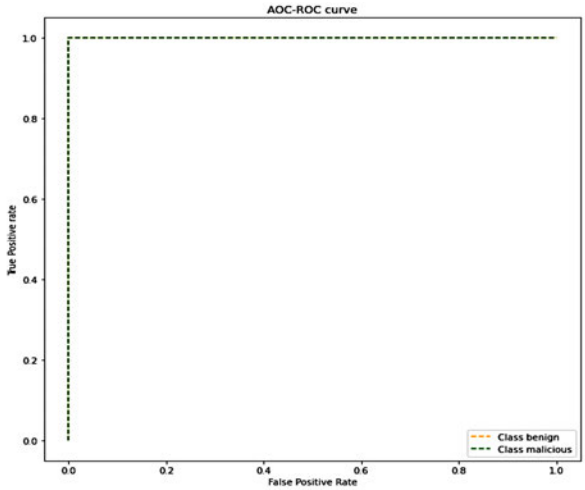


Figure 5-17: ROC-AUC curve IoT-3

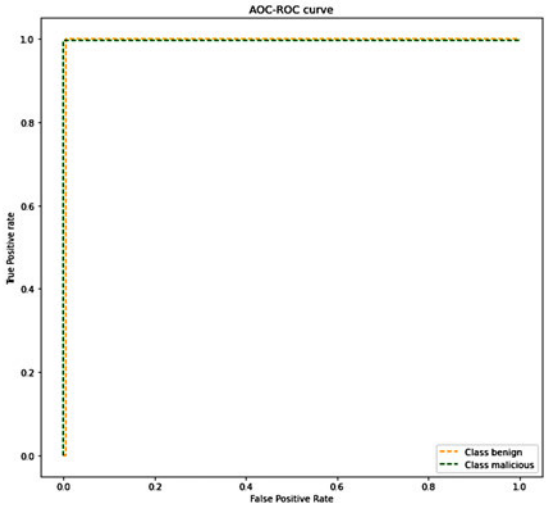


Figure 5-18: ROC-AUC curve IoT-4

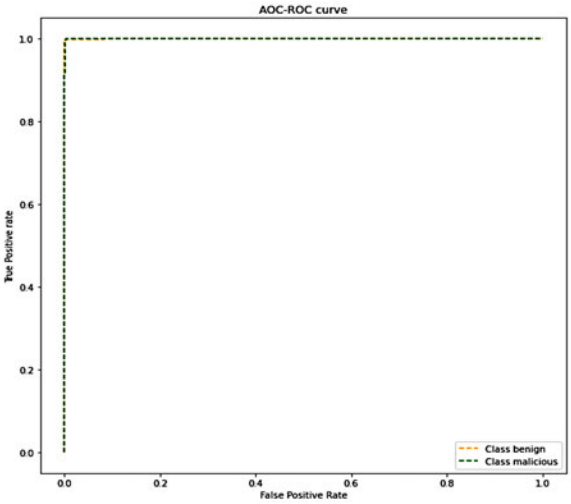


Figure 5-19: ROC-AUC curve IoT-5

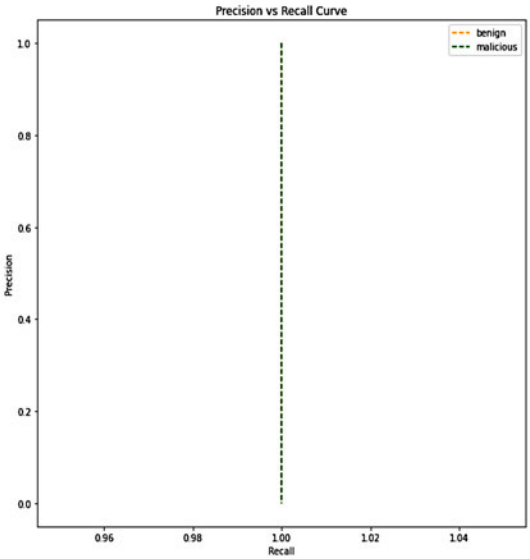


Figure 5-20: Precision-recall curve IoT-1

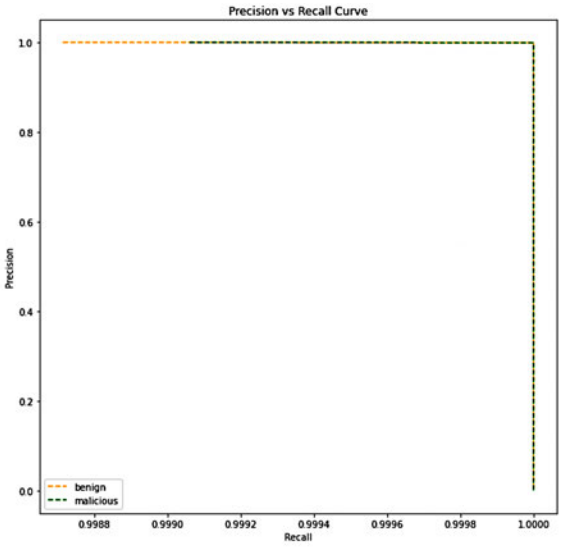


Figure 5-21: Precision-recall curve IoT-2

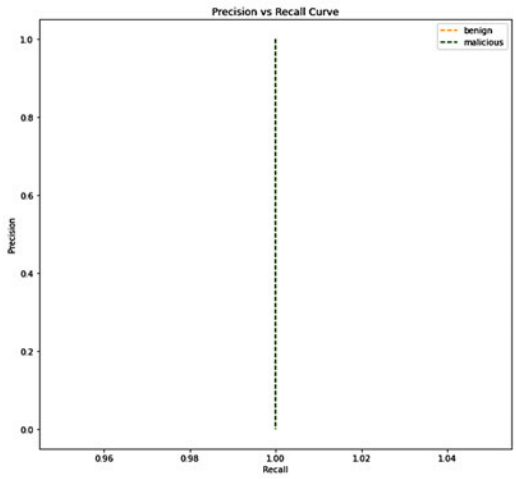


Figure 5-22: Precision-recall curve IoT-3

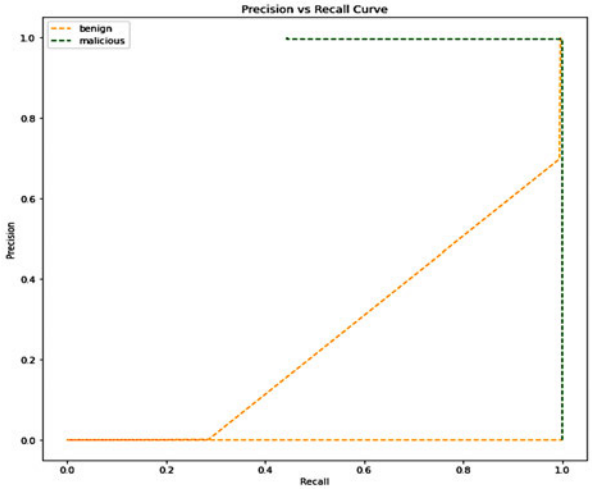


Figure 5-23: Precision-recall curve IoT-4

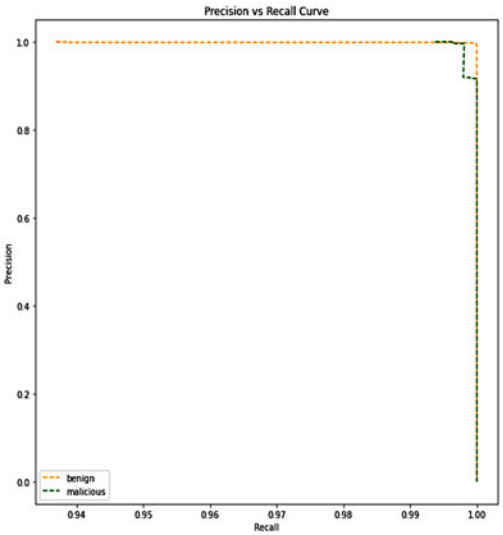


Figure 5-24: Precision-recall curve IoT-5

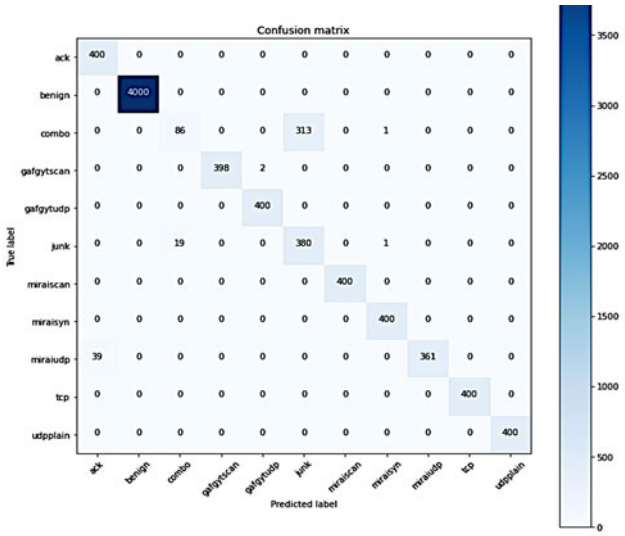


Figure 5-25: Multiclass Confusion matrix IoT-1

5.3.4.4 Device Type Multiclass Classification Results

A further study performed in this section is to investigate how responsive the proposed model is to unknown attacks and its efficiency on different types of attacks. Only gafgyt attacks were considered for Ennio doorbell because the device was not infected by Mirai attack classes. The results are presented in tables 5-14 to 5-19 and illustrated in figures 5-26 to 5-39.

Table 5-14: Multiclassification Performance Results for Danmini Doorbell

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	1.00	1.00	1.00	4000
Gafgyt.combo	0.82	0.21	0.34	400
Gafgyt.junk	0.55	0.95	0.70	400
Gafgyt.scan	1.00	0.99	1.00	400
Gafgyt.tcp	1.00	1.00	1.00	400
Gafgyt.udp	1.00	1.00	1.00	400
Mirai.ack	0.91	1.00	0.95	400
Mirai.scan	1.00	1.00	1.00	400
Mirai.syn	1.00	1.00	1.00	400
Mirai.udp	1.00	0.90	0.95	400
Mirai.udpplain	1.00	1.00	1.00	400

Accuracy			0.95	8000
Macro-avg	0.93	0.91	0.90	8000
Weighted-Avg	0.96	0.95	0.95	8000

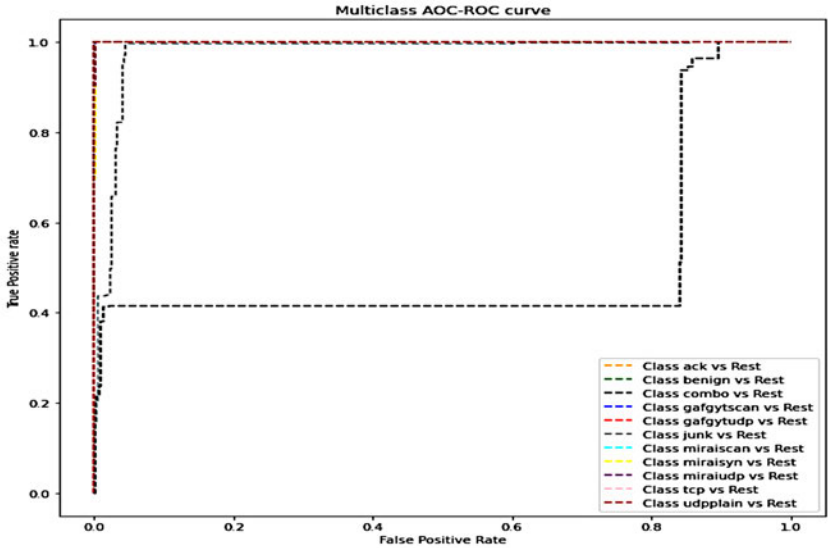


Figure 5-26: ROC-AUC curve IoT-1

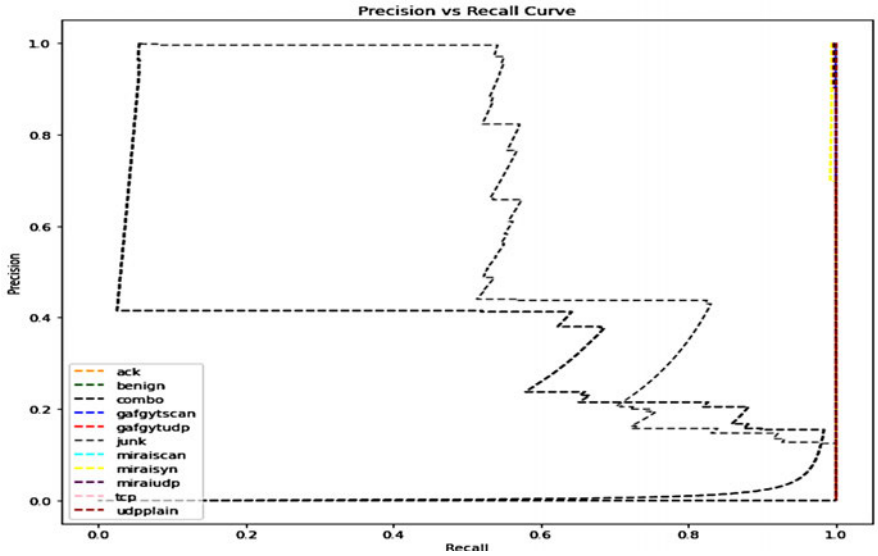


Figure 5-27: Precision-recall curve IoT-1

Table 5-15: Multiclassification Performance Results for Ecobee Thermostat

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	1.00	1.00	1.00	3113
Gafgyt.combo	0.50	1.00	0.67	400
Gafgyt.junk	1.00	1.00	1.00	400
Gafgyt.scan	1.00	1.00	1.00	400
Gafgyt.tcp	0.00	0.00	0.00	400
Gafgyt.udp	1.00	1.00	1.00	400
Mirai.ack	1.00	1.00	1.00	400
Mirai.scan	1.00	1.00	1.00	400
Mirai.syn	1.00	1.00	1.00	400
Mirai.udp	1.00	1.00	1.00	400
Mirai.udpplain	1.00	1.00	1.00	400
Accuracy			0.94	7113
Macro-avg	0.86	0.91	0.88	7113
Weighted-Avg	0.92	0.94	0.92	7113

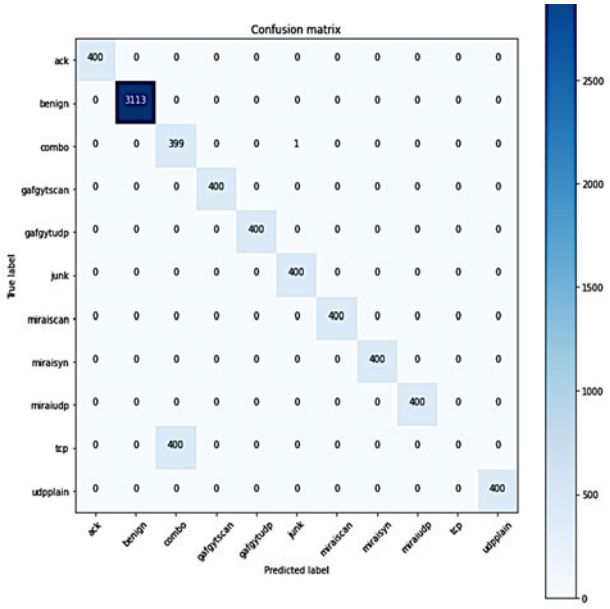


Figure 5-28: Confusion matrix IoT-2

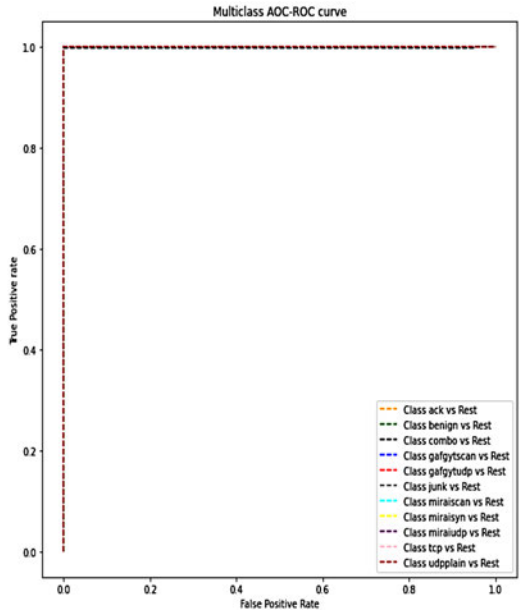


Figure 5-29: ROC-AUC curve IoT-2

Table 5-16: Multiclassification Performance Results for Ennio Doorbell

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	0.99	1.00	0.99	4000
Gafgyt.combo	0.81	1.00	0.90	400
Gafgyt.junk	1.00	0.66	0.80	400
Gafgyt.scan	1.00	1.00	1.00	400
Gafgyt.tcp	1.00	1.00	1.00	400
Gafgyt.udp	1.00	1.00	1.00	400
Accuracy			0.98	6000
Macro-avg	0.97	0.94	0.95	6000
Weighted-avg	0.98	0.98	0.98	6000

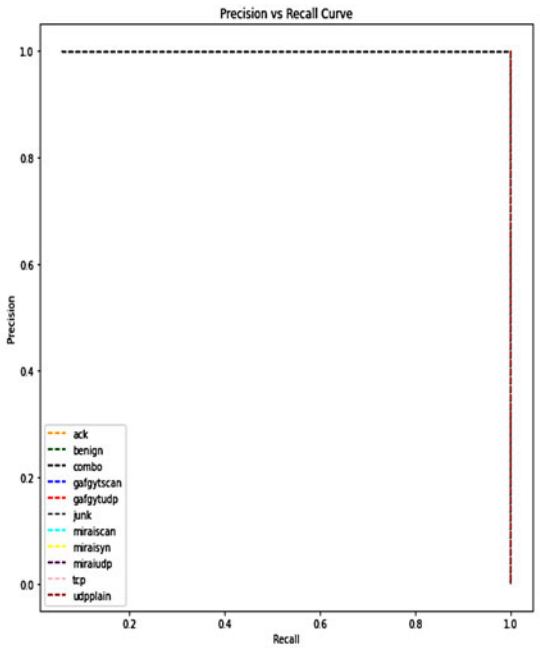


Figure 5-30: Precision-recall curve IoT-2

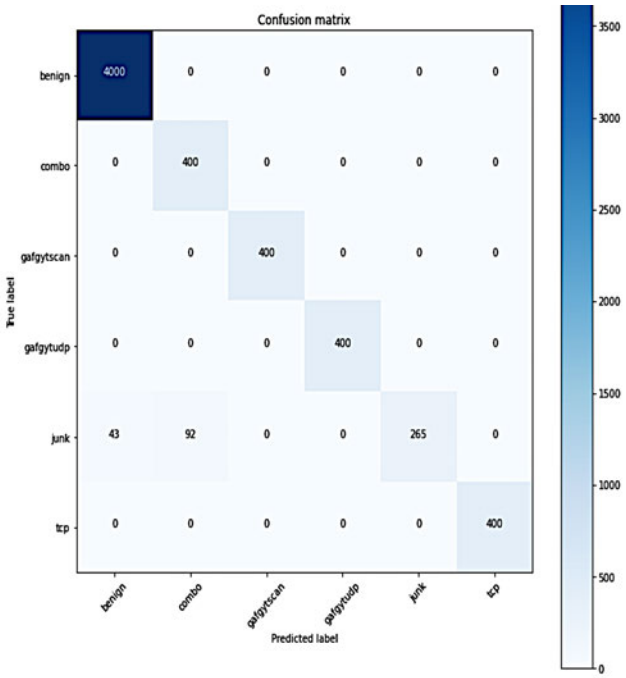


Figure 5-31: Confusion matrix IoT-3

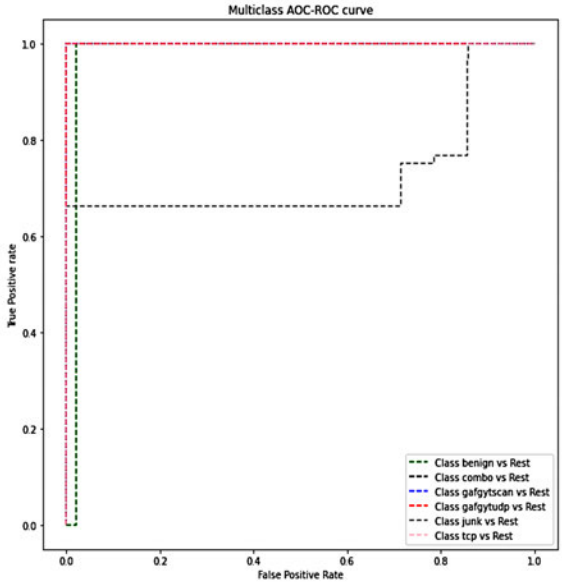


Figure 5-32: ROC-AUC curve IoT-3

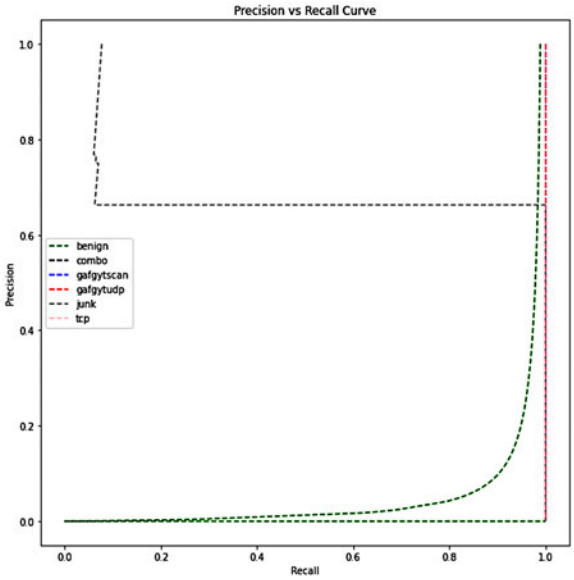


Figure 5-33: Precision-recall curve IoT-3

Table 5-17: Multiclassification Performance Results for Philips B120N10 Baby Monitor

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	1.00	1.00	1.00	4000
Gafgyt.combo	1.00	1.00	1.00	400
Gafgyt.junk				400
Gafgyt.scan	1.00	1.00	1.00	400
Gafgyt.tcp	0.50	1.00	0.67	400
Gafgyt.udp	0.00	**0.00**	**0.00**	400
Mirai.ack	1.00	1.00	1.00	400
Mirai.scan	1.00	1.00	1.00	400
Mirai.syn	1.00	1.00	1.00	400
Mirai.udp	1.00	1.00	1.00	400
Mirai.udpplain	1.00	1.00	1.00	400
Accuracy			0.95	8000
Macro-avg	0.86	0.91	0.88	8000
Weighted-avg	0.92	0.95	0.93	8000

Table 5-18: Multiclassification Results for IoT (1-5)

Device	metrics			
	TRAINING	TESTING	KAPPA	MCC
Danmini Doorbell	**2560**	**0.69**	0.935	0.937
Ecobee Thermostat	2231	0.69	0.927	0.93
Ennio Doorbell	1301	0.40	0.96	0.96
Philips B120N10 Baby Monitor	4001	0.81	0.93	0.93
Provision PT 737E Security Camera	3323	0.73	0.97	0.97

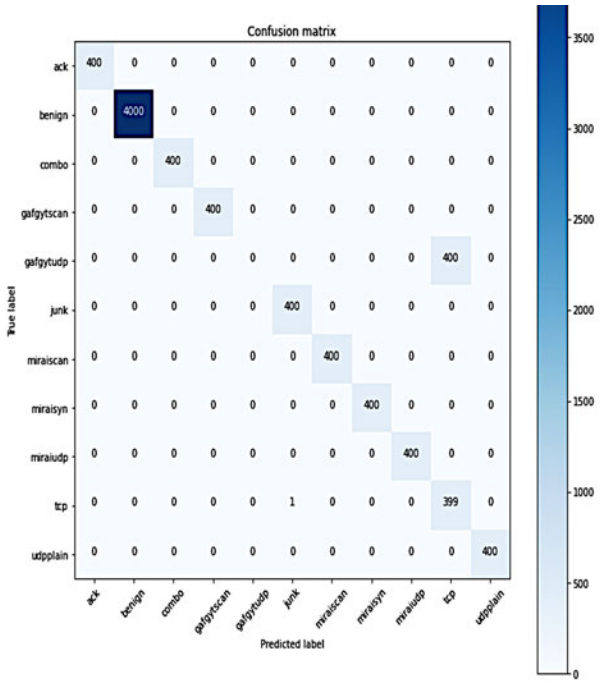


Figure 5-34: Confusion matrix IoT-4

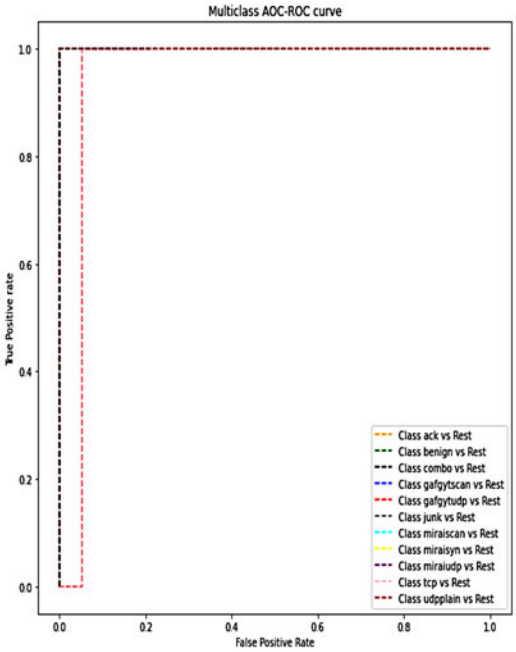


Figure 5-35: ROC-AUC curve IoT-4

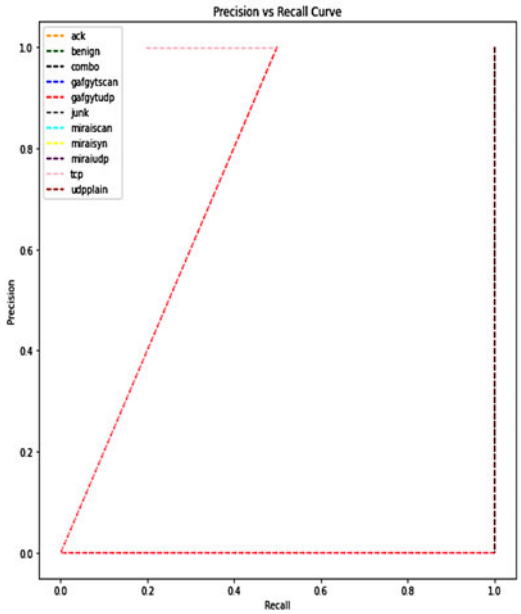


Figure 5-36: Precision-recall curve IoT-4

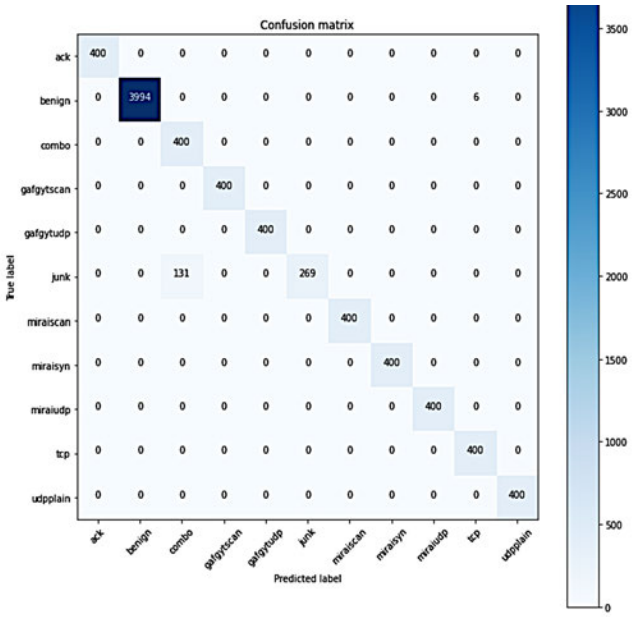


Figure 5-37: Confusion matrix for IoT 5

Table 5-19: Multiclassification Performance Results for Provision PT 737E Security Camera

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	1.00	1.00	1.00	4000
Gafgyt.combo	0.75	1.00	0.86	400
Gafgyt.junk	1.00	0.67	0.80	400
Gafgyt.scan	1.00	1.00	1.00	400
Gafgyt.tcp	0.99	1.00	0.99	400
Gafgyt.udp	1.00	1.00	1.00	400
Mirai.ack	1.00	1.00	1.00	400
Mirai.scan	1.00	1.00	1.00	400
Mirai.syn	1.00	1.00	1.00	400
Mirai.udp	1.00	1.00	1.00	400

Mirai.udpplain	1.00	1.00	1.00	400
Accuracy			0.98	8000
Macro-avg	0.98	0.97	0.97	8000
Weighted-avg	0.99	0.98	0.98	8000

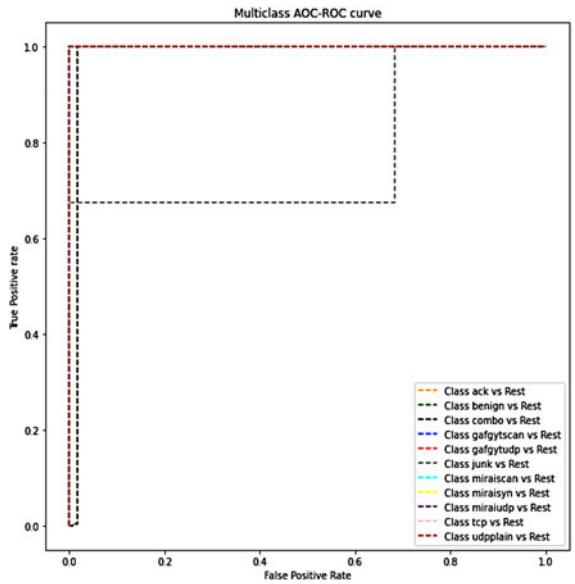


Figure 5-38: ROC-AUC curve IoT-5

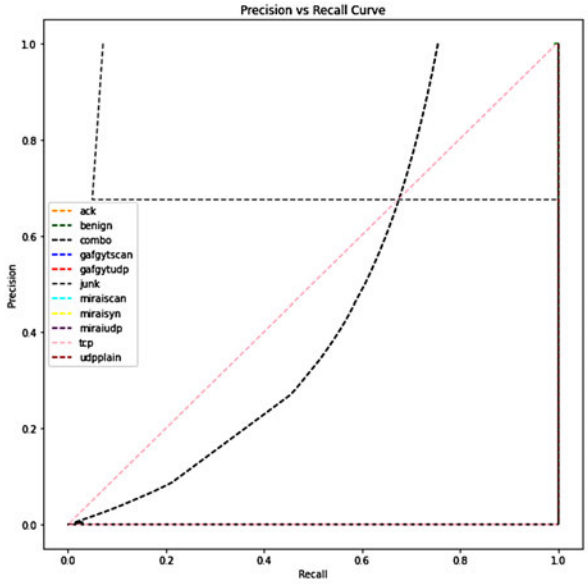


Figure 5-39: Precision-recall curve IoT-5

5.4 Generality of the Proposed DSBM Model

5.4.1 Introduction

Transfer learning relaxes the assumption that training datasets must be distributed independently and identically (i.i.d.) with testing datasets, encouraging us to use transfer learning to address the issue of insufficient training data. The main limitation of supervised learning algorithms is that they require a dataset of a very specific type which should be labelled, consist of both benign and malicious traffic, and must be located on the network being protected by the IDS. It is a challenging task to create such a dataset. It necessitates extensive planning on numerous fronts, including attack and schedule choice, automatic labelling, traffic representativeness, and so on. According to the Institute for Cyber-security in Canada, obtaining that kind of dataset is extremely difficult [208]. Transfer learning is a type of ML approach that is focused on reusing a technique developed for one purpose as a start point for another. A machine-learning method designed to detect dogs, for example, can be re-used to detect cats. Tasks A and B in our example correspond to two distinct devices on an IoT network. The primary benefit of transfer learning is that it expedites the training process and produces better results than the model designed solely for task B [209]. Transfer learning enables training to be done with data which is not an exact representation of the monitored traffic. The model's abilities to adapt to new contexts are useful for more than just deployment. These characteristics are still relevant, allowing it to adapt to the inevitable changes in the network traffic (addition and/or removal of devices, new protocols, etc.). Generability is thus beneficial during training and throughout the model's lifetime. The samples from

the devices in the NbaIoT datasets are suitable for testing the transfer learning abilities of our proposed DSBM model. Indeed, both datasets are IoT specific generated datasets and they share common attacks. However, the monitored network's generating devices are very different, illustrating a typical situation in which an Intrusion Detection System is trained with a dataset that doesn't conform to the network where it is being deployed. IDSs that are network-agnostic are useful because they can be deployed on any kind of network, but their performance is less effective than IDSs that have been actually trained to properly understand the behaviour of a specific network. As a result, this approach trades off ease of deployment for a possible loss of detection performance.

5.4.2 Methodology

The aim of this section is to train the proposed model on one dataset before testing it on another. Good results on the testing set would demonstrate that the proposed model is capable of ignoring network topology for attack detection. Firstly, the model was trained with dataset from IoT 6 and then tested with IoT 7 datasets. Since IoT 6 has a greater number of classes than IoT 7, we selected it as the train set. This is in line with the fact that the training set is considerably huge when compared to the test set. However, the results will contain only 6 attack classes. A repeat experiment was also conducted using IoT 8 and IoT 9 dataset as train and test respectively. Hold out validation is used in this section with typical ratio of 64/33. 33% of IoT 6 and IoT 8 is employed as the validation set to evaluate the effectiveness of proposed DSBM on their original network generating devices.

5.4.3 Dataset Preprocessing

In this subsection, the methods used for pre-processing the datasets in order to perform the transfer learning experiments are mentioned. Although the datasets used simulated attacks on an IoT network from commercial IoT devices, there are some differences that must be addressed before they can be used together. For example, the files in the IoT-6 dataset cannot be used directly for transfer learning evaluation because their features are out of sync with those in the IoT-7 dataset. This is because the category label distribution of IoT-7 dataset contains only two classes which are benign and gafgyt attacks with the absence of mirai attack classes. Therefore, mirai attack classes are not considered for transfer learning capabilities between IoT 6-7 and only considered for IoT 8-9. Performing some data cleansing routines is a useful data pre-processing step. In this case, all the IoT device dataset files were inspected for duplicates., incorrect attribute values and inconsistencies. The instances that were duplicated were removed, and inconsistencies in the datasets were corrected. Class imbalance problems was handled using the techniques discussed in chapter 4 section 2. The features suggested by authors in section chapter 4 section 3 will be used to train and test our proposed DSBM. These optimal number of features were obtained from conducting multiple experiments using information gain. Additionally, to make sure that the DSBM model performs well, the flow identifiers of the network packet, like the

IP addresses of the source and destination, numbers of the port, and timestamp, were ignored. The model could fail to generalise well when it is deployed if it is trained using those features because attackers can launch attacks on the network using different IP addresses at different times.

5.4.4 Analysis of the Results

In this section, the proposed model is evaluated to test its performance in transfer learning: 5,000 records from each of the classes in device (6 and device 7) and (8 and 9) was used for training and testing purposes. However, due to the absence of mirai attack classes in device 7, only 30,000 records each from device 6 and 7 containing gafgyt attack classes was considered for (6-7). The classification of the proposed model on the test set is promising but lower than its performance on the training set as shown in the 5-20.

5.4.4.1 BINARY

Table 5-20: Results for Transfer Learning Devices 6-7

DATASET	ACCURACY	PRECISION	RECALL	Fmeasure	Time	MCC	KAPPA
IoT 6 (training)	99.9	99.9	99.9	99.9	1409.9	-	-
IoT 7(testing)	99.4	99.5	99.5	99.7	2.6	99.313	99.309

While the model performed well on majority of the attack classes, the precision for for gafgyt.udp and recall for gafgyt.tcp is 0.97 which is lower than the results obtained during training. The results for the attack categories are stated in Table 5-21.

5.4.4.2 Multiclass 6-7

Table 5-21: Transfer Learning Multiclassification Results 6-7

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	1.00	1.00	1.00	5000
Gafgyt.combo	1.00	1.00	1.00	5000
Gafgyt.junk	1.00	1.00	1.00	10000
Gafgyt.scan	1.00	1.00	1.00	10000
Gafgyt.tcp	1.00	0.97	0.99	10000
Gafgyt.udp	0.97	1.00	0.99	10000
Accuracy			0.98	50000
Macro avg	1.00	1.00	1.00	50000
Weighted Avg	0.99	0.99	0.99	50000

Figures 5:40 to 5:42 shows the confusion matrix, roc-auc curve and precision-recall curve for IoT device dataset 7 after proposed model has been trained on device 6 dataset.

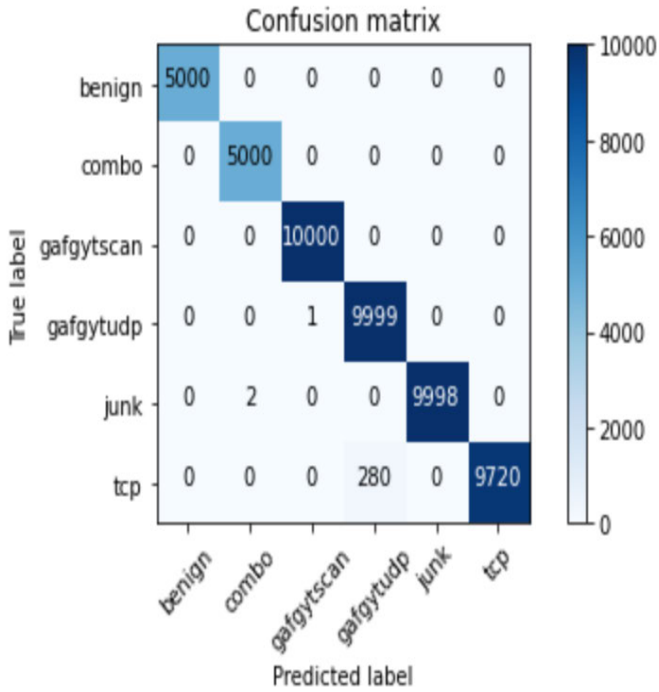


Figure 5-40: Confusion matrix for IoT devices 6-7

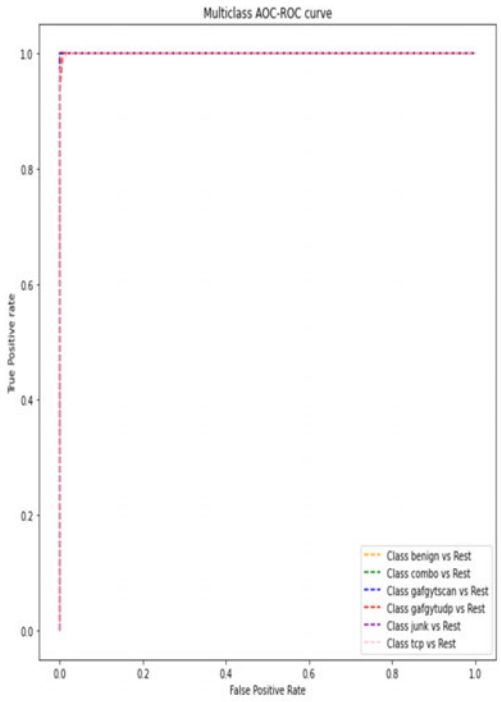


Figure 5-41: ROC-AUC curve IoT Devices 6-7

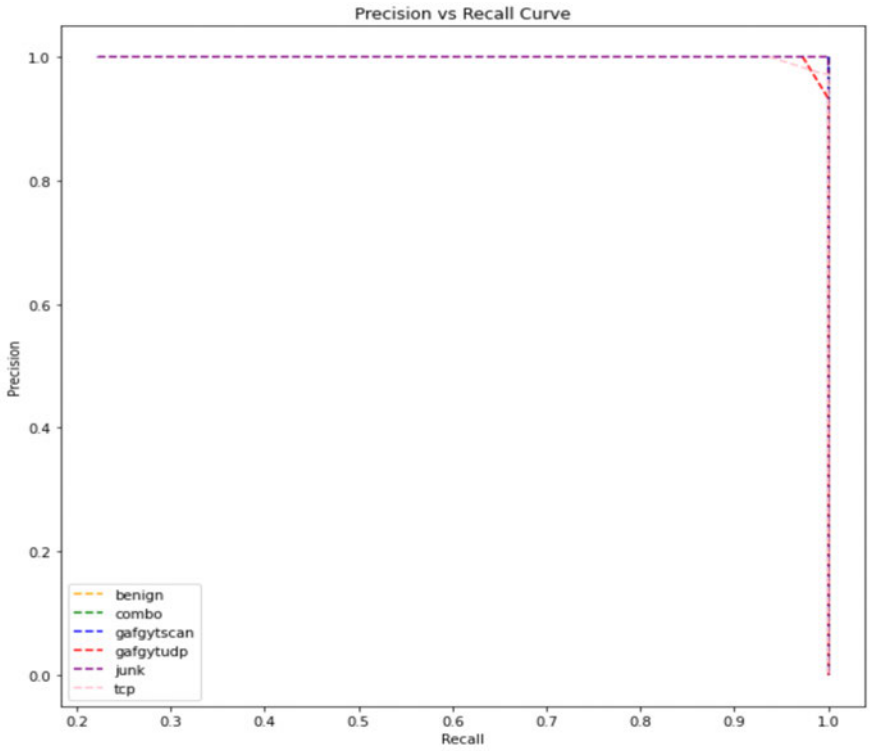


Figure 5-42: Precision-recall curve IoT devices 6-7

The model's overall performance is excellent as shown in the ROC CURVE in figure 5-41. However, this transfer learning approach doesn't improve the classification of all attack categories, but it performs well especially for gafgyt.combo, gafgyt.junk and gafgyt.scan attacks in IoT-6-7 datasets

(AUCROC=1.00). The transfer learning results for IoT 8-9 is illustrated in the tables 5-22 and 5-23, and figures 5-43 to 5-45.

5.4.4.3 BINARY FOR TL 8-9

Table 5-22 Transfer Learning Binary Classification Results 8-9

DATASET	ACCURACY	PRECISION	RECALL	Fmeasure	Time	MCC	KAPPA
IoT 8 (training)	99.7	99.0	98.0	98.0	3668	-	-
IoT 9(testing)	99.1	98.0	97.0	97.0	3.8	92.3	92.0

5.4.4.4 MULTICLASS 8-9

Table 5-23: Transfer Learning Multiclassification Results 8-9

CATEGORIES	Classification performance (%)			SUPPORT
	PRECISION	RECALL	FMEASURE	
Benign	0.99	1.00	1.00	5000
Gafgyt.combo	1.00	0.45	0.62	5000
Gafgyt.junk	0.62	1.00	0.77	5000
Gafgyt.scan	1.00	0.99	1.00	5000
Gafgyt.tcp	0.00	0.00	0.00	5000
Gafgyt.udp	1.00	1.00	1.00	5000
Mirai.ack	0.99	0.83	0.90	5000
Mirai.scan	1.00	1.00	1.00	5000
Mirai.syn	1.00	1.00	1.00	5000
Mirai.udp	0.85	0.99	0.92	5000
Mirai.udpplain	1.00	0.94	0.97	5000
Accuracy			0.93	55,000
Macro avg	0.95	0.93	0.92	55,000
Weighted Avg	0.95	0.93	0.92	55,000

Figures 5-43 to 5-45 shows the confusion matrix, roc-auc curve and precision-recall curve for IoT device dataset 9 after proposed model has been trained on device 8 dataset.

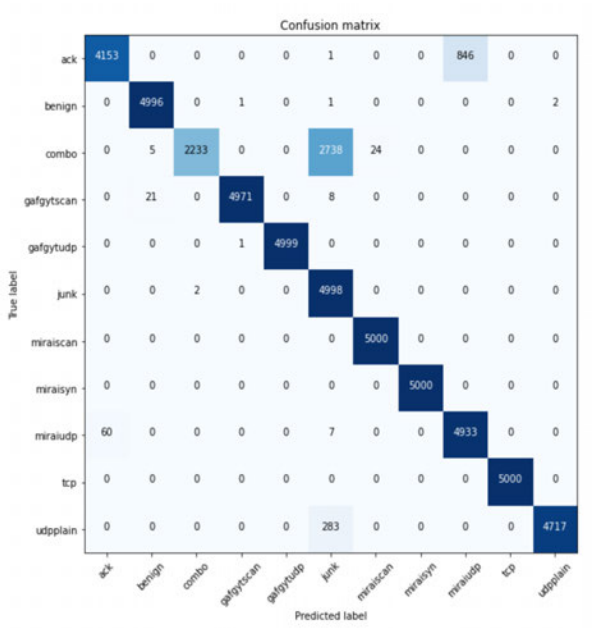


Figure 5-43: Confusion matrix for IoT devices 8-9

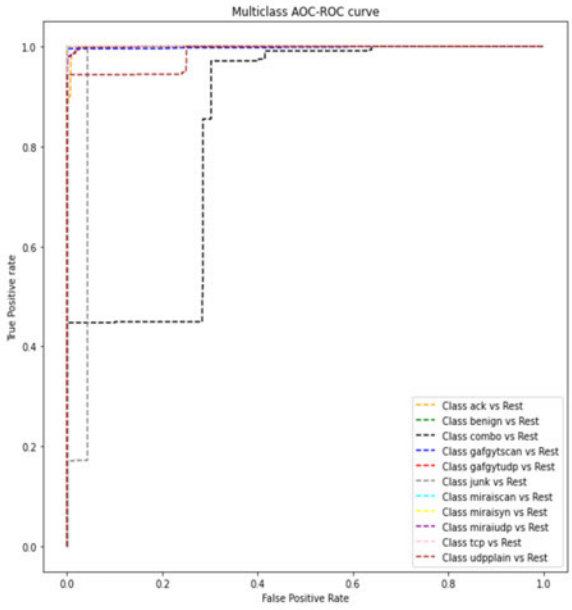


Figure 5-44: ROC-AUC curve IoT Devices 8-9

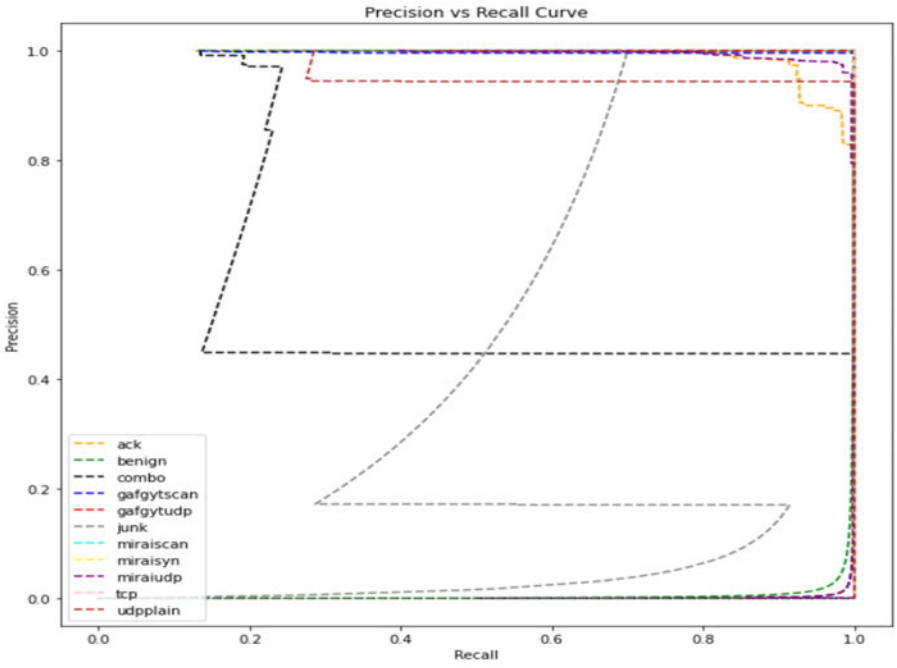


Figure 5-45: Precision-recall curve IoT devices 8-9

5.5 Conclusion, Summary and Discussion

In this first section in this chapter, an investigation into anomaly-based IDS techniques that can be used to secure IoT was conducted. The evaluation of the effectiveness of supervised machine learning algorithms such as RF, AB, LGBM, GBM, ETC, XGB, SGDC, LR, and DNN are given special attention. The best parameters for the classifiers are found using a random search approach. Benchmark datasets like NbaIoT and IoTID20 are used to evaluate classifier performance. Recall, precision, accuracy, and Fmeasure are used to assess the effectiveness of each classifier. The training time of all

classifiers is measured during the training phase. Additionally, Friedman and Dunn's post-hoc tests are used in the statistical analysis of performance measurements to detect significant differences amongst classifiers. The performance results and statistical analysis conducted in this research study indicates that DNN, Adaptive Boosting, and Gradient Boosting Machines (GBM, LGBM, XGB) classifiers exhibit the most appropriate trade-off between performance metrics and evaluation time making them the ideal choices for developing anomaly-based IDS specifically for IoT. The suitable classifiers selected in section 5.1 was utilized to develop the framework for preventing attacks in IoT networks in section 5.2. The stacking ensemble IDS model in proposed in section 5.2 used the strengths of three boosting algorithms, and a DNN as a meta-learner. The choice of DNN as meta-learner determines the success of the proposed model because it offered an intelligent decision combination method and proved to be effective in achieving state-of-the-art performance compared to single classifiers and addresses the challenges related to determining an appropriate meta-learner. According to the results obtained in section 5.2, the proposed model can effectively detect both botnet attacks and the types of botnet attacks. Therefore, the design of IDS based on boosting algorithms and an ensemble architecture would be an effective approach for IDS for several IoT devices. Therefore, we conclude that the performance of an IDS in the IoT is largely dependent on the type of IDS model rather than the type of IoT device. The proposed model has promising prospects of detecting relevant IoT botnets and it is worthy of further investigation; therefore, future work conducted in section 5.3 focused on generality of the proposed model to study the model's ability to adapt to a new scenario so that it keeps adapting to the inevitable changes in IoT network traffic. Due to the similarity in the data distributions of the train and test sets in the NbaIoT, the proposed model performed remarkably well when trained and tested within the dataset of the same device. The results obtained could be misleading because when the model is deployed, it will inevitably encounter different data. In section 5.3, we evaluated the proposed model using a dataset containing similar attacks simulated using different devices to help in generalization. The motivation of the study conducted in section 5.3 is because of the importance to correctly evaluate the performance of a machine learning model used to design an intrusion detection system. When models are trained with one dataset and tested with another from a different device, the model's performance slightly suffers. As a result, we draw the conclusion that it is essential to train IDS models with a variety of data and simulations in order to develop a robust IDS that can generalise well when used with real-world data.

CHAPTER 6: Discussion, Summary and Conclusion

6.1 Discussion and Summary

The global adoption of IoT devices has accelerated in recent years. Connected devices are now being used in several fields, which includes healthcare, smart cities, and education. While there has been little focus on the security and safety of IoT devices and networks, this presents a risk to IoT users and threatens the entire internet-connected environment. Furthermore, in terms of complexity and diversity, security attack vectors have advanced in both ways. As a result, more emphasis must be placed on the analysis and detection of these attacks. This thesis investigates several machine learning methods employed in intrusion detection and proposed an efficient technique based on the hybridization of deep learning and boosted machines in a stacking scheme to deploy anomaly-based intrusion detection systems (IDSs) on real networks in order to be able to detect and respond to possible threats as they occur. The thesis was divided into six chapters. We started by giving a brief overview of the IoT ecosystem's background, characteristics, applications, benefits, security flaws, security threats, and security solutions in chapter 1. we discuss our research motivation, objectives, contributions made during the research, and limitations of our study. A thorough review of state-of-the-art models presented by various researchers in previous years was discussed in Chapter 2 to get a review of various machine learning, deep learning classifiers, and ensemble techniques used to develop intrusion detection systems. We compared their strategies, architecture, and outcomes. The challenges in intrusion detection are discussed. The findings of the review prompted the experiments described in chapters 4, and 5 of this thesis. Chapter 3 contains a detailed description of the experimental datasets as well as the machine learning models employed in this study. In chapter 4, a comprehensive comparison of single and ensemble classifiers based on four key important evaluation parameters was conducted to see how well the algorithms performed, and they were also combined into an ensemble learner. The findings are thought to be relevant in the development of IDS systems that combine strong classification algorithms. It also includes an extensive research to highlight and troubleshoot the issues inherent in benchmark datasets. An investigation into an approach known as RTL-DL was also carried out. To reduce the effects of data imbalance and improve classification performance, the RTL-DL employs a deep neural network, random over-sampling, and Tomek-Links sampling technique (RTL). When compared to the current approaches discussed in Chapter 2, the RTL-DL model showed promising results in detecting network threats from an imbalanced dataset. Despite the fact that information gain was used in experiments conducted in chapter 4 sections 1 and 2, there is still a problem of selection bias and underestimation of relevant features. The authors of [169, 193] used a predefined threshold to select features; this limitation was addressed in the experiment conducted in chapter 4 section 3. The main goal is to provide an overview of some benchmark datasets and to assess the impact of IG on model performance by comparing model performance while considering different feature groups chosen based on their IG score. The significance and importance of the chosen feature

groups are validated by comparing the F-measure and processing time across each group in the dataset. According to the study, a DNN model has a good ability to detect anomalies in a significant amount of execution time when the best features are chosen. Finally, recommendations on how to eliminate problems in the IDS dataset were made, and unique features from the benchmark datasets were presented. The features presented are used for the remainder of the research in this thesis. A study conducted in chapter 5 focused on an investigation into anomaly-based IDS that can be used to secure IoT. The study was motivated by the difficulties in the selection of appropriate supervised ML technique for developing an efficient IDS. In that study, two groups of some of the most extensively applied and popular supervised machine learning algorithms were chosen. To detect significant differences in the performances of these classifiers, two statistical tests which are Friedman and Dunn's post-hoc test were conducted. The results proved that DNN, Adaptive Boosting, and Gradient Boosting Machines (GBM, LGBM, XGB) classifiers exhibit the very best possible trade-off between the performance parameters and time of evaluation, making them the ideal choices for developing anomaly-based IDS specifically for IoT. Additionally, our proposed model was implemented in chapter 5, it is based on a stacking ensemble technique which employed the classifiers that shows the best trade-offs in the study conducted in chapter 5 section 1. The focus of our model is on the IoT ecosystem and to the best of our knowledge, we are the first to employ a deep learning method as a meta learner for boosted machines to build an IDS for the IoT ecosystem. The proposed model is a consistent architecture that includes a security workflow that is complete, from collection of data to threat detection. It is also characterized by its adherence to resource constraints, real-time response, scalability and transfer learning abilities.

6.2 Limitations and Future Work

Despite the fact that the model presented in this thesis produced promising results, enhancing the efficiency of detection results in machine learning research is still remains an open topic. While respecting the constraints of Internet of Things, security researchers should always strive for excellent performance in evaluation metrics. The model developed in this thesis provides motivation for future research. This section outlines some contributions of this study, its limitations and suggest future directions to address these limitations. Among them are the following:

1. This thesis proposes a variety of learning techniques based on the type of learning employed. Each one of these methods can be improved by presenting solutions to address their shortcomings. The major limitations of the study conducted in this thesis are the adoption of supervised learning algorithms, which appears to be limited for intrusion detection due to the lack of labelled data. Unsupervised learning algorithms are anticipated to be implemented in future work.
2. The use of limited data samples from the dataset employed is another limitation of the study. The IoT datasets used were produced by only 9 and 2 IoT devices for the NbaIoT and IoTID20 datasets, respectively. In the future, we propose testing the model proposed in this thesis on a

larger number of samples from internet of things network traffic datasets with several devices. Consideration should also be given to the adaptability of the models to general networks comprising of not only internet of things devices but also workstations, smartphones and pcs. It will help the model to generalize well due to the training of a single model using a variety of datasets to account for all potential attack simulations.

3. Future work may also utilize ML techniques to create host-based intrusion detection systems (HIDS), anti-malware and other anti-spywares/viruses for Internet of Things. The trained models should be executable on internet of things devices with limited resources. To this end, various DL model compression techniques, such as pruning, quantization, and knowledge distillation, may be investigated**.** Obviously, the major issue would be to look for the ideal trade-off between the model compression and maintaining reasonable performances.
4. Currently, every study employs its own metrics to evaluate the performance of its model, making comparison between works very difficult. Defining a comprehensive set of metrics that allow for the assessment of the quality of presented models is critical to the success of machine learning applications. Networking researchers should create a set of metrics to assess the performance of machine learning models.

REFERENCES

1. Kumar, S., P. Tiwari, and M. Zymbler, *Internet of Things is a revolutionary approach for future technology enhancement: a review*. Journal of Big data, 2019. 6(1): p. 1-21.
2. Terroso-Saenz, F., Lopes, B.; Ramos, A.; Ribeiro, L. "*An open IoT platform for the management and analysis of energy data*". Future generation computer systems, 2019. 92: p. 1066-1079.
3. Kim, T.-h., C. Ramos, and S. Mohammed, "*Smart city and IoT*". 2017, Elsevier.
4. Chen, K., Zhang, Z.; Wang, J.; Lin, Z.; Li, L. *Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice.* Journal of Hardware and Systems Security, 2018. 2(2): p. 97-110.
5. Arias, O., K. Ly, and Y. Jin, *Security and privacy in IoT era*, in *Smart Sensors at the IoT Frontier*. 2017, Springer. p. 351-378.
6. Zhou, W.; Ding, X.; Wang, Y.; Wu, X.; Song, H. *The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved.* IEEE Internet of Things Journal, 2018. 6(2): p. 1606-1616.
7. Zhang, Z.-K.; Chen, K.; Lee, H.-B.; He, D.; Wang, L. *IoT security: ongoing challenges and research opportunities*. in *2014 IEEE 7th international conference on service-oriented computing and applications*. 2014. IEEE.
8. Chaabouni, N.; Khoukhi, L.; Trabelsi, R. B.; Laurent, M. *"Network intrusion detection for IoT security based on learning techniques."* IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2671-2701, 2019.
9. Alaba, F.A.; Awe, O.O.; Adewumi, A.O.; Ayo, C.K. *"Internet of Things security: A survey."* Journal of Network and Computer Applications, vol. 88, pp. 10-28, 2017.
10. Frustaci, M.; Nigro, L.; Cortese, G.; Monteleone, S.; Kotsokalis, C.; Pietrosanto, A. *"Evaluating critical security issues of the IoT world: Present and future challenges."* IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2483-2495, 2017.
11. Mahmoud, R.; Al-Kasassbeh, M.; Aslam, N.; Baker, T. *"Internet of things (IoT) security: Current status, challenges and prospective measures."* in 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST). 2015. IEEE.
12. *International Data Corporation (IDC). The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast. 2019 url: https: //www.idc.com/getdoc.jsp?containerId=prUS45213219.*
13. *The Independent. Hackers Now Able to Take Control of Cars to Cause Deliberate Accidents, Scientists Warn. 2017. https://www.independent.co.uk/life-style/gadgets- and - tech / news / computer - hackers - control - car - deliberate - accidents-national-security-issue-a8066466.html.*
14. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. *Internet of things for smart cities.* IEEE Internet of Things journal, 2014. 1(1): p. 22-32.
15. Fortune Business Insights., *Market Research Report*, Jul. 2023 *url* https://www.fortunebusinessinsights.com/industry-reports/smart-home-market-101900
16. Kumar, J.S. and D.R. Patel, *A survey on internet of things: Security and privacy issues.* International Journal of Computer Applications, 2014. 90(11).
17. Atzori, L., A. Iera, and G. Morabito, *The internet of things: A survey.* Computer networks, 2010. 54(15): p. 2787-2805.
18. Hanna, S.; Chin, E.; Bahl, V.; Kaiser, G. *"Take Two Software Updates and See Me in the Morning: The Case for Software Security Evaluations of Medical Devices."* in HealthSec. 2011. Citeseer.
19. Halperin, D.; Heydt-Benjamin, T.; Fu, K.; Kohno, T.; Maisel, W.H. *"Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses."* in 2008 IEEE Symposium on Security and Privacy (sp 2008). 2008. IEEE.
20. Babovic, Z.B., J. Protic, and V. Milutinovic, *Web performance evaluation for internet of things applications.* IEEE Access, 2016. 4: p. 6974-6992.

21. Neshenko, N.; Singh, R.; Anuar, N.B.; Kiah, M.L.M.; Younas, M. *Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations.* IEEE Communications Surveys & Tutorials, 2019. 21(3): p. 2702-2733.
22. Shahid, M.R., *Deep learning for Internet of Things (IoT) network security*. 2021, Institut Polytechnique de Paris.
23. Khan, M.A. and K. Salah, *IoT security: Review, blockchain solutions, and open challenges.* Future generation computer systems, 2018. 82: p. 395-411.
24. Barcena, M.B. and C. Wueest, *Insecurity in the Internet of Things*. Security response, symantec, 2015.
25. Bertino, E. and N. Islam, *Botnets and internet of things security.* Computer, 2017. 50(2): p. 76-79.
26. Costin, A. and J. Zaddach, *Iot malware: Comprehensive survey, analysis framework and case studies.* BlackHat USA, 2018 (accessed december 7, 2020).
27. Kambourakis, G., C. Kolias, and A. Stavrou. *The mirai botnet and the iot zombie armies*. in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*. 2017. IEEE.
28. Cozzi, E.; Pellegrino, G.; Zanero, S. *Understanding linux malware*. in *2018 IEEE symposium on security and privacy (SP)*. 2018. IEEE.
29. Alto, P., *unit 42 iot threat report.* Palo Alto, 2020 (accessed october,2021) https://start.paloaltonetworks.com/unit-42-iot-threat-report.
30. OWASP, *Owasp Internet of Things Top 10 2018. 2018 (accessed october 2020). https://owasp.org/www-pdf-archive/OWASP-IoT- Top-10-2018-final.pdf.*
31. Marzano, A.; Conti, M.; Dainotti, A.; Pescape, A. *The evolution of bashlite and mirai iot botnets*. in *2018 IEEE Symposium on Computers and Communications (ISCC)*. 2018. IEEE.
32. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. *DDoS in the IoT: Mirai and other botnets.* Computer, 2017. 50(7): p. 80-84.
33. Angrishi, K., *Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets.* arXiv preprint arXiv:1702.03681, 2017.
34. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; Lever, C.; Ma, M.; Mason, J.; Menscher, D.; Seaman, C.; Sullivan, N.; Thomas, K.; Zhou, Y. *Understanding the mirai botnet*. in *26th {USENIX} security symposium ({USENIX} Security 17)*. 2017.
35. Wang, A.; Chen, C.; Irwin, D.; Gao, J. *An inside look at IoT malware*. in *International Conference on Industrial IoT Technologies and Applications*. 2017. Springer.
36. Pa, Y.M.P.; Cheung, R.C.C.; Chow, K.P.; Tse, T.H.; Cao, J. *IoTPOT: Analysing the rise of IoT compromises*. in *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*. 2015.
37. *radware. "BrickerBot" Results In PDoS Attack. Apr. 2017. URL: https://security.radware.com/ddos-threats- attacks/brickerbot-pdos-permanent-denial-of-service/.*
38. Soltan, S., P. Mittal, and H.V. Poor. *BlackIoT: IoT botnet of high wattage devices can disrupt the power grid.* in *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 2018.
39. Edwards, S. and I. Profetis, *Hajime: Analysis of a decentralized internet worm for IoT devices.* Rapidity Networks, 2016. 16: p. 1-18.
40. *Bitdefender. New Hide 'N Seek IoT Botnet using custom-built Peer-to- Peer communication spotted in the wild. 2018 (accessed october, 2021). url: https://labs.bitdefender.com/2018/01/new-hide-n-seek-iot-botnet-using-custom-built peer-to-peer-communication- spotted-in-the-wild/.*
41. Haria, S., *The growth of the hide and seek botnet.* Network Security, 2019. 2019(3): p. 14-17.
42. Micro, T., *Persirai: New internet of things (IoT) botnet targets ip cameras.* URL: https://blog. trendmicro. com/trendlabs-security-intelligence/persirai-new-internetthings-iot-botnet-targets-ip-cameras, 2017.
43. *Radware. Reaper Botnet. 2017 (accessed october 2021) url: https:// security.radware.com/ddos-threats-attacks/threat-advisories- attack-reports/reaper-botnet/.*
44. Li, C. and C. Chen, *A multi-stage control method application in the fight against phishing attacks.* Proceeding of the 26th computer security academic communication across the country, 2011: p. 145.

45. Ferrag, M.A.; Maglaras, L.; Janicke, H.; Derhab, A. *Security for 4G and 5G cellular networks: A survey of existing authentication and privacy-preserving schemes.* Journal of Network and Computer Applications, 2018. 101: p. 55-82.
46. Al-Turjman, F. and S. Alturjman, *Confidential smart-sensing framework in the IoT era.* The Journal of Supercomputing, 2018. 74(10): p. 5187-5198.
47. Singh, S.; Kumar, N.; Pasricha, N.; Tyagi, S.K. *Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions.* Journal of Ambient Intelligence and Humanized Computing, 2017: p. 1-18.
48. Surendran, S., A. Nassef, and B.D. Beheshti. *A survey of cryptographic algorithms for IoT devices*. in *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. 2018. IEEE.
49. Choo, K.-K.R., S. Gritzalis, and J.H. Park, *Cryptographic solutions for industrial Internet-of-Things: Research challenges and opportunities.* IEEE Transactions on Industrial Informatics, 2018. 14(8): p. 3567-3569.
50. Jaballah, W.B., et al., *Lightweight secure group communications for resource constrained devices.* International Journal of Space-Based and Situated Computing, 2015. 5(4): p. 187-200.
51. Al-Turjman, F.; Bahi, J.M.; Alqerm, I. *Seamless key agreement framework for mobile-sink in IoT based cloud-centric secured public safety sensor networks.* IEEE Access, 2017. 5: p. 24617-24631.
52. Al-Turjman, F. and S. Alturjman, *Context-sensitive access in industrial internet of things (IIoT) healthcare applications.* IEEE Transactions on Industrial Informatics, 2018. 14(6): p. 2736-2744.
53. Chaabouni, N., *Intrusion detection and prevention for IoT systems using Machine Learning*. 2020, Université de Bordeaux.
54. *M. Ahmed, A. Naser Mahmood, and J. Hu. A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60:19–31, January 2016.*
55. Haider, W.; Sezer, S.; Shah, S.A.; Asif, M. *"Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling."* Journal of Network and Computer Applications, vol. 87, pp. 185-192, 2017.
56. Columbus., L., *Roundup of Internet of Things Forecasts and Market Estimates.* 2016. https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup- of-internet-of-things-forecasts-and-market-estimates-2016/. .
57. Raza, S., L. Wallgren, and T. Voigt, *SVELTE: Real-time intrusion detection in the Internet of Things.* Ad hoc networks, 2013. 11(8): p. 2661-2674.
58. Otoum, S., B. Kantarci, and H.T. Mouftah. *Mitigating False Negative intruder decisions in WSN-based Smart Grid monitoring*. in *2017 13th International wireless communications and mobile computing conference (IWCMC)*. 2017. IEEE.
59. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. *"A detailed analysis of the KDD CUP 99 data set."* in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009. IEEE.
60. Panda, M., A. Abraham, and M.R. Patra, *A hybrid intelligent approach for network intrusion detection.* Procedia Engineering, 2012. 30: p. 1-9.
61. Bou-Harb, E., M. Debbabi, and C. Assi, *Cyber scanning: a comprehensive survey.* Ieee communications surveys & tutorials, 2013. 16(3): p. 1496-1519.
62. Casas, P., J. Mazel, and P. Owezarski, *Unsupervised network intrusion detection systems: Detecting the unknown without knowledge.* Computer Communications, 2012. 35(7): p. 772-783.
63. *M.A. Boden. Artificial Intelligence and Natural Man. Computer science/psychology. Basic Books, 1977.*
64. Chollet, F., *Deep learning with Python*. 2021: Simon and Schuster.
65. *J. F. Puget. What Is Machine Learning? (IT Best Kept Secret Is Optimization), May 2016.*
66. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; Hassabis, D.

67. Belavagi, M.C. and B. Muniyal, *Performance evaluation of supervised machine learning algorithms for intrusion detection.* Procedia Computer Science, 2016. 89: p. 117-123.
68. Almseidin, M.; Faisal, R. M.; Kamaludin, N. H.; Al-kasasbeh, A. S. *Evaluation of machine learning algorithms for intrusion detection system*. in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*. 2017. IEEE.
69. Zaman, M. and C.-H. Lung. *Evaluation of machine learning techniques for network intrusion detection*. in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. 2018. IEEE.
70. Pahl, M.-O. and F.-X. Aubet. *All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection*. in *2018 14th International Conference on Network and Service Management (CNSM)*. 2018. IEEE.
71. Diro, A.A. and N. Chilamkurti, *Distributed attack detection scheme using deep learning approach for Internet of Things.* Future Generation Computer Systems, 2018. 82: p. 761-768.
72. Kozik, R.; Vapen, A.; Wilk, S. *A scalable distributed machine learning approach for attack detection in edge computing environments.* Journal of Parallel and Distributed Computing, 2018. 119: p. 18-26.
73. Hasan, M.; Alam, M. M.; Soni, A.; Alamri, A. *Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches.* Internet of Things, 2019. 7: p. 100059.
74. Jan, S. U.; Abbasi, A. M.; Yang, L.; Seet, B. C.; Jeon, G. *Toward a lightweight intrusion detection system for the internet of things.* IEEE Access, 2019. 7: p. 42450-42471.
75. Krishnaveni, S.; Akila, G.; Mala, C.; Ramaiah, M. *Anomaly-based intrusion detection system using support vector machine*, in *Artificial intelligence and evolutionary computations in engineering systems*. 2020, Springer. p. 723-731.
76. Vinayakumar, R.; Jacob, L.; Joy, J.; Anand, R. S.; Anjali, R. *Deep learning approach for intelligent intrusion detection system.* IEEE Access, 2019. 7: p. 41525-41550.
77. Almomani, I., B. Al-Kasasbeh, and M. Al-Akhras, *WSN-DS: A dataset for intrusion detection systems in wireless sensor networks.* Journal of Sensors, 2016. 2016.
78. Pajouh, H.H.; Dehghantanha, A.; Khayami, R.; Choo, K. K. R.; Dobbie, G. *A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks.* IEEE Transactions on Emerging Topics in Computing, 2016. 7(2): p. 314-323.
79. Liu, X.; Wang, W.; Wu, J.; Zhang, J.; Li, Y.; Li, T. *Defending ON–OFF attacks using light probing messages in smart sensors for industrial communication systems.* IEEE Transactions on Industrial Informatics, 2018. 14(9): p. 3801-3811.
80. Alrashdi, I.; Al-Absi, A.; Choo, K. K. R.; Dobbie, G. *Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning.* in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. 2019. IEEE.
81. Doshi, R., N. Apthorpe, and N. Feamster. *Machine learning ddos detection for consumer internet of things devices*. in *2018 IEEE Security and Privacy Workshops (SPW)*. 2018. IEEE.
82. Yang, K., et al., *Active learning for wireless IoT intrusion detection.* IEEE Wireless Communications, 2018. 25(6): p. 19-25.
83. Krishna, K.V., K. Swathi, and B.B. Rao, *A Novel Framework for NIDS through Fast kNN Classifier on CICIDS2017 Dataset.*
84. Alrowaily, M., F. Alenezi, and Z. Lu. *Effectiveness of machine learning based intrusion detection systems*. in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. 2019. Springer.
85. Asad, M.; Hussain, S. A.; Anwar, M.; Iqbal, W. *Deepdetect: detection of distributed denial of service attacks using deep learning.* The Computer Journal, 2020. 63(7): p. 983-994.
86. Muraleedharan, N. and B. Janet, *A deep learning based HTTP slow DoS classification approach using flow data.* ICT Express, 2021. 7(2): p. 210-214.
87. Sbai, O. and M. El boukhari. *Data flooding intrusion detection system for manets using deep learning approach*. in *Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications*. 2020.

88. Amaizu, G. C.; Mu, H.; Chen, H.; Seneviratne, A.; Seneviratne, S. *Composite and efficient DDoS attack detection framework for B5G networks.* Computer Networks, 2021. 188: p. 107871.
89. Cil, A.E., K. Yildiz, and A. Buldu, *Detection of DDoS attacks with feed forward based deep neural network model.* Expert Systems with Applications, 2021. 169: p. 114520.
90. Johnson, J.M. and T.M. Khoshgoftaar, *Survey on deep learning with class imbalance. J. Big Data 6 (1), 1–54 (2019)*.
91. Sabeel U, Heydari SS, Mohanka H, Bendhaou Y, Elgazzar K, El- Khatib K (2019) *'Evaluation of deep learning in detecting unknown network attacks. In: 2019 international conference on Smart Applications, Communications and Networking, SmartNets'* 2019. Institute of Electrical and Electronics Engineers Inc
92. Virupakshar KB, Asundi M, Channal K, Shettar P, Patil S, Narayan DG (2020) *Distributed Denial of Service (DDoS) attacks detection system for OpenStack-based Private Cloud. Procedia Comput Sci* 167:2297–2307.
93. Kumari, A. and A.K. Mehta. *A hybrid intrusion detection system based on decision tree and support vector machine*. in *2020 IEEE 5th International conference on computing communication and automation (ICCCA)*. 2020. IEEE.
94. Moustafa, N., B. Turnbull, and K.-K.R. Choo, *An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things.* IEEE Internet of Things Journal, 2018. 6(3): p. 4815-4830.
95. Pokharel, P., R. Pokhrel, and S. Sigdel. *Intrusion detection system based on hybrid classifier and user profile enhancement techniques*. in *2020 International Workshop on Big Data and Information Security (IWBIS)*. 2020. IEEE.
96. Khonde, S. and V. Ulagamuthalvi, *Hybrid framework for intrusion detection system using ensemble approach.* Int. J. Adv. Trends Comput. Sci. Eng, 2020. 9(4): p. 4881-4890.
97. Aljawarneh, S., M. Aldwairi, and M.B. Yassein, *Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model.* Journal of Computational Science, 2018. 25: p. 152-160.
98. Zhou, Y.; Feng, Y.; Xu, Z.; Wang, C.; Zheng, Q. *Building an efficient intrusion detection system based on feature selection and ensemble classifier.* Computer networks, 2020. 174: p. 107247.
99. Khraisat, A.; Almomani, H.; Rawashdeh, A.; Jararweh, Y.; Benkhelifa, E. *A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks.* Electronics, 2019. 8(11): p. 1210.
100. Pham, N. T.; Tran, D. H.; Huynh, V. H.; Dang, H. N.; Nguyen, T. M.; Suh, S. *Improving performance of intrusion detection system using ensemble methods and feature selection*. in *Proceedings of the Australasian Computer Science Week Multiconference*. 2018.
101. Kachavimath, A.V., S.V. Nazare, and S.S. Akki. *Distributed denial of service attack detection using naïve bayes and k-nearest neighbor for network forensics*. in *2020 2nd International conference on innovative mechanisms for industry applications (ICIMIA)*. 2020. IEEE.
102. Kilincer, I.F., F. Ertam, and A. Sengur, *Machine learning methods for cyber security intrusion detection: Datasets and comparative study.* Computer Networks, 2021. 188: p. 107840.
103. Fitni, Q.R.S. and K. Ramli. *Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems*. in *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*. 2020. IEEE.
104. Rebecca, V. *8 Metrics to Measure Model Performance.* https://towardsdatascience.com/8-metrics-to-measure-classification-performance-984d9d7fd7aa
105. Demšar, J., *Statistical comparisons of classifiers over multiple data sets.* The Journal of Machine Learning Research, 2006. 7: p. 1-30.
106. Conover, W.J., *Practical nonparametric statistics*. Vol. 350. 1999: john wiley & sons.
107. Friedman, M., *The use of ranks to avoid the assumption of normality implicit in the analysis of variance.* Journal of the american statistical association, 1937. 32(200): p. 675-701.
108. Dunn, O.J., *Multiple comparisons among means.* Journal of the American statistical association, 1961. 56(293): p. 52-64.

109. Dunn, O.J., *Multiple comparisons using rank sums.* . Technometrics 1964. 6 p. 241–252.
110. Tama, B.A. and K.-H. Rhee, *An in-depth experimental study of anomaly detection using gradient boosted machine.* Neural Computing and Applications, 2019. 31(4): p. 955-965.
111. Kumar, G. and K. Kumar, *The use of artificial-intelligence-based ensembles for intrusion detection: a review.* Applied Computational Intelligence and Soft Computing, 2012. 2012.
112. Aburomman, A.A. and M.B.I. Reaz, *A survey of intrusion detection systems based on ensemble and hybrid classifiers.* Computers & security, 2017. 65: p. 135-152.
113. Dietterich, T.G., *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization.* Machine learning, 2000. 40(2): p. 139-157.
114. Sesmero, M., L. P., A. , and S. I., A. (2015). , *Generating ensembles of heterogeneous classifiers using Stacked Generalization.* . Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(1), doi:10.1002/widm.1143 p. 21–34.
115. Ordóñez, F.J., A. Ledezma, and A. Sanchis. *Genetic Approach for Optimizing Ensembles of Classifiers*. in *FLAIRS conference*. 2008.
116. Mahdavinejad, M.S., et al., *Machine learning for Internet of Things data analysis: A survey.* Digital Communications and Networks, 2018. 4(3): p. 161-175.
117. Rathore, S. and J.H. Park, *Semi-supervised learning based distributed attack detection framework for IoT.* Applied Soft Computing, 2018. 72: p. 79-89.
118. https://www.cisco.com/c/en_uk/products/security/security-reports.html., C.a.c.r.
119. Goodfellow, I., Y. Bengio, and A. Courville, *Deep learning*. 2016: MIT press.
120. Dietterich, T.G. *Ensemble methods in machine learning*. in *International workshop on multiple classifier systems*. 2000. Springer.
121. Folino, G. and F.S. Pisani. *Combining ensemble of classifiers by using genetic programming for cyber security applications*. in *European Conference on the Applications of Evolutionary Computation*. 2015. Springer.
122. Sesmero, M.P., A.I. Ledezma, and A. Sanchis, *Generating ensembles of heterogeneous classifiers using stacked generalization.* Wiley interdisciplinary reviews: data mining and knowledge discovery, 2015. 5(1): p. 21-34.
123. Kumar, G. and K. Kumar. *AI based supervised classifiers: an analysis for intrusion detection*. in *Proceedings of the International Conference on Advances in Computing and Artificial Intelligence*. 2011.
124. Chen, T. and C. Guestrin. *Xgboost: A scalable tree boosting system*. in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.
125. *Introduction to Boosted Trees—Xgboost 0.7 Documentation. Available online: http://xgboost.readthedocs.io/en/latest/model.html*
126. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.
127. Friedman, J.H., *Greedy function approximation: a gradient boosting machine.* Annals of statistics, 2001: p. 1189-1232.
128. Geurts, P., D. Ernst, and L. Wehenkel, *Extremely randomized trees.* Machine learning, 2006. 63(1): p. 3-42.
129. Breiman, L., *Random forests.* Machine learning, 2001. 45(1): p. 5-32.
130. Freund, Y. and R.E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of computer and system sciences, 1997. 55(1): p. 119-139.
131. *https://www.geeksforgeeks.org/stacking-in-machine-learning/.*
132. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M.; Alrajeh, N. *A deep learning approach for network intrusion detection system, in: proceedings of the 9th EAI International Conference on bio-inspired information and communications technologies (formerly BIONETICS).* formerly BIONETICS), 2016.
133. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A. A. *Toward developing a systematic approach to generate benchmark datasets for intrusion detection.* computers & security, 2012. 31(3): p. 357-374.

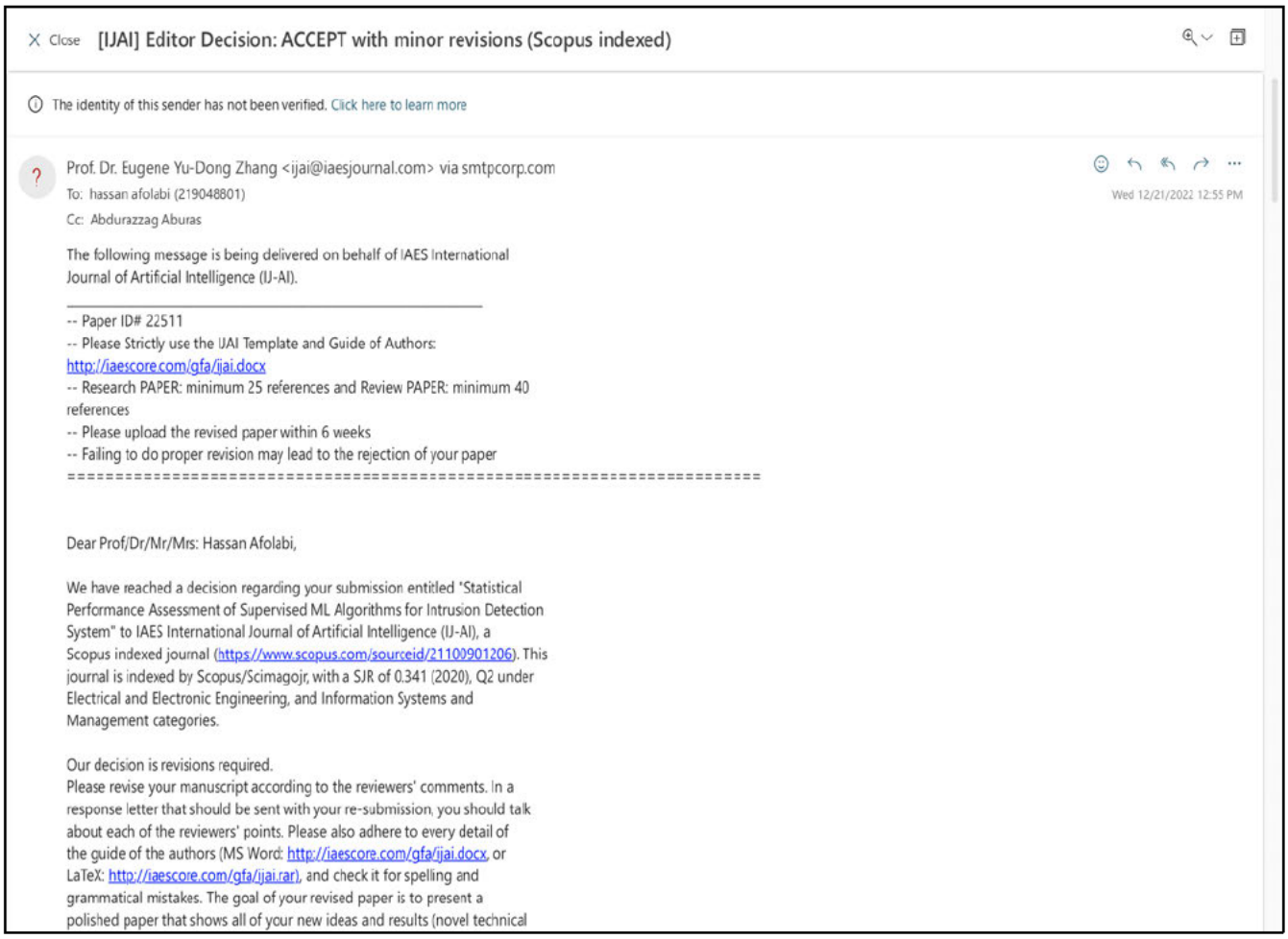
134. Ring, M.; Wunder, M.; Landes, D. *A survey of network-based intrusion detection data sets*. Computers & Security, 2019. 86: p. 147-167.
135. Sommer, R. and V. Paxson. *Outside the closed world: On using machine learning for network intrusion detection*. in *2010 IEEE symposium on security and privacy*. 2010. IEEE.
136. Ghourabi, A., T. Abbes, and A. Bouhoula, *Characterization of attacks collected from the deployment of Web service honeypot.* Security and Communication Networks, 2014. 7(2): p. 338-351.
137. Ghourabi, A., T. Abbes, and A. Bouhoula. *Data analyzer based on data mining for honeypot router*. in *ACS/IEEE International Conference on Computer Systems and Applications-AICCSA 2010*. 2010. IEEE.
138. *S. I. S. Center, Sans isc: Demonstrating the value of your intrusion detection program and analysts. [Online]. Available: https://isc.sans.edu/diary. html?date=2012-09-02 (Accessed on 11/08/2021).*
139. Park, Y. and J. Park. *Web application intrusion detection system for input validation attack*. in *2008 Third International Conference on Convergence and Hybrid Information Technology*. 2008. IEEE.
140. Stevanovic, D., A. An, and N. Vlajic, *Feature evaluation for web crawler detection with data mining techniques.* Expert Systems with Applications, 2012. 39(10): p. 8707-8717.
141. Auld, T., A.W. Moore, and S.F. Gull, *Bayesian neural networks for internet traffic classification.* IEEE Transactions on neural networks, 2007. 18(1): p. 223-239.
142. Sen, S. and J.A. Clark, *Evolutionary computation techniques for intrusion detection in mobile ad hoc networks*. Computer Networks, 2011. 55(15): p. 3441-3457.
143. Alata, E.; Couturier, R.; Debar, H. *Collection and analysis of attack data based on honeypots deployed on the Internet*, in *Quality of Protection*. 2006, Springer. p. 79-91.
144. Ghourabi, A., T. Abbes, and A. Bouhoula. *Behavior analysis of web service attacks*. in *IFIP International Information Security Conference*. 2014. Springer.
145. Danubianu, M. *Step by step data preprocessing for data mining. A case study*. in *Proceedings of the International Conference on Information Technologies (InfoTech-2015)*. 2015.
146. Alasadi, S.A. and W.S. Bhaya, *Review of data preprocessing techniques in data mining.* Journal of Engineering and Applied Sciences, 2017. 12(16): p. 4102-4107.
147. Sivakumar, A. and R. Gunasundari, *A Survey on Data Preprocessing Techniques for Bioinformatics and Web Usage Mining.* International Journal of Pure and Applied Mathematics, 2017. 117(20): p. 785-794.
148. Yu, L., Y. Pan, and Y. Wu. *Research on data normalization methods in multi-attribute evaluation*. in *2009 International Conference on Computational Intelligence and Software Engineering*. 2009. IEEE.
149. Maza, S. and M. Touahria, *Feature selection algorithms in intrusion detection system: A survey.* KSII Transactions on Internet and Information Systems (TIIS), 2018. 12(10): p. 5079-5099.
150. Zhao, X.; Liu, H.; Han, C.; Liu, H. *A two-stage feature selection method with its application.* Computers & Electrical Engineering, 2015. 47: p. 114-125.
151. Pirscoveanu, R.S., et al. *Analysis of malware behavior: Type classification using machine learning*. in *2015 International conference on cyber situational awareness, data analytics and assessment (CyberSA)*. 2015. IEEE.
152. Sheena, K.K. and G. Kumar. *Analysis of feature selection techniques: A data mining approach*. in *IJCA Proceedings on International Conference on Advances in Emerging Technology*. 2016.
153. Nag, K. and N.R. Pal, *A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification.* IEEE transactions on cybernetics, 2015. 46(2): p. 499-510.
154. Hota, H. and A.K. Shrivas, *Decision tree techniques applied on NSL-KDD data and its comparison with various feature selection techniques*, in *Advanced Computing, Networking and Informatics-Volume 1*. 2014, Springer. p. 205-211.
155. Abdullah, M.; Khan, S.U.; Anwar, M.W.; Nadir, S. *Enhanced intrusion detection system using feature selection method and ensemble learning algorithms.* International Journal of Computer Science and Information Security (IJCSIS), 2018. 16(2): p. 48-55.

156. Ferreira, A.J. and M.A. Figueiredo, *Efficient feature selection filters for high-dimensional data.* Pattern Recognition Letters, 2012. 33(13): p. 1794-1804.
157. Géron, A., *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. 2019: " O'Reilly Media, Inc.".
158. Chollet, F., *Deep learning with python, vol. 1.* Greenwich, CT: Manning Publications CO, 2017.
159. *V. Vijayakumar and V. Neelanarayanan, "Intrusion Detection model using Chi Square feature selection and modified Naïve Bayes classifier," Smart Innovation, Systems and Technologies, vol. 49. Springer, Switzerland, pp. v–vii, 2016.*
160. Kirubavathi, G. and R. Anitha, *Botnet detection via mining of traffic flow characteristics.* Computers & Electrical Engineering, 2016. 50: p. 91-101.
161. Serpen, G. and E. Aghaei, *Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms.* Intelligent Data Analysis, 2018. 22(5): p. 1101-1114.
162. Mera, C. and J.W. Branch, *A survey on class imbalance learning on automatic visual inspection.* IEEE Latin America Transactions, 2014. 12(4): p. 657-667.
163. Wang, S., L.L. Minku, and X. Yao, *A systematic study of online class imbalance learning with concept drift.* IEEE transactions on neural networks and learning systems, 2018. 29(10): p. 4802-4821.
164. Song, Q., Y. Guo, and M. Shepperd, *A comprehensive investigation of the role of imbalanced learning for software defect prediction.* IEEE Transactions on Software Engineering, 2018. 45(12): p. 1253-1269.
165. Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; Herrera, F. *A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2011. 42(4): p. 463-484.
166. Wang, S. and X. Yao, *Multiclass imbalance problems: Analysis and potential solutions.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012. 42(4): p. 1119-1130.
167. *R. Longadge,S.S. Dongre,L. Malik, "Class Imbalance Problem in Data Mining: Review", International Journal of Computer Science and Network (IJCSN), Volume 2, Issue 1, February 2013, ISSN 2277-5420.*
168. Abd Elrahman, S.M. and A. Abraham, *A review of class imbalance problem.* Journal of Network and Innovative Computing, 2013. 1(2013): p. 332-340.
169. Afolabi, H.A and Aburas, A.A. RTL-DL: A Hybrid Deep Learning Framework For DDoS Attack Detection In Big Data Environment; *International Journal of Computer Networks & Communications (IJCNC) Vol.14, No.6, November 2022, pp 51-66.* DOI: 10.5121/ijcnc.2022.14604
170. Singh Panwar S, Raiwani Y, Panwar LS*, Evaluation of network intrusion detection with features selection and machine learning algorithms on CICIDS-2017 dataset. in International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India; 2019.*
171. Nicholas L, et al*. Study of long short-term memory in flow-based network intrusion detection system. Journal of Intelligent & Fuzzy Systems. 2018;35(6):5947-5957.*
172. CICIDS2017; 2017. *Available:*http://www.unb.ca/cic/datasets/ID *S2017.html.*
173. Panigrahi R, Borah S. *A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. International Journal of Engineering & Technology. 2018;7(3.24):479-482.*
174. Panwar, S.S.; Singh, A.; Yadav, D.; Tanwar, S. *Implementation of machine learning algorithms on CICIDS- 2017 dataset for intrusion detection using WEKA; 2019.*
175. Chandolikar, N. and V. Nandavadekar. *Efficient algorithm for intrusion attack classification by analyzing KDD Cup 99*. in *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*. 2012. IEEE.
176. *NSL-KDD; 2009. Available:*https://www.unb.ca/cic/datasets/ *nsl.html.*

177. Verma, P.; Kaushik, A.; Kumar, N.; Maurya, R.K. *Network intrusion detection using clustering and gradient boosting*. in *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*. 2018. IEEE.
178. Dhanabal, L. and S. Shantharajah, *A study on NSL-KDD dataset for intrusion detection system based on classification algorithms*. International journal of advanced research in computer and communication engineering, 2015. 4(6): p. 446-452.
179. Revathi, S. and A. Malathi, *A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection.* International Journal of Engineering Research & Technology (IJERT), 2013. 2(12): p. 1848-1853.
180. Meidan, Y.; Mathov, Y.; Bohadana, M.; Shabtai, A.; Ochoa, M.; Streich, P.; Dankner, A.; Hershcovich, M.; Munitz, H.; Mirsky, Y.; Elovici, Y. *N-baiot—network-based detection of iot botnet attacks using deep autoencoders*. IEEE Pervasive Computing, 2018. 17(3): p. 12-22.
181. Mirsky, Y.; Shabtai, A.; Ochoa, M.; Elovici, Y. *Kitsune: an ensemble of autoencoders for online network intrusion detection.* arXiv preprint arXiv:1802.09089, 2018.
182. Kang, H.; Kim, M.; Kang, S.; Lee, J.; Choi, J. *IoT network intrusion dataset.* IEEE Dataport, 2019.
183. Ullah, I. and Q.H. Mahmoud. *A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks*. in *Canadian Conference on AI*. 2020.
184. Lashkari, A. H.; Abdul-Mageed, M.; Al-Rakhami, M. S.; Mohaisen, A. *Characterization of tor traffic using time based features*. in *ICISSp*. 2017.
185. Fernández, A.; García, S.; Herrera, F. *SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary.* Journal of artificial intelligence research, 2018. 61: p. 863-905.
186. *T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: data mining, inference, and prediction, 2nd Ed., 12th Printing, Springer Science & Business Media, New York, 2017.*
187. Karimi, Z., M.M.R. Kashani, and A. Harounabadi, *Feature ranking in intrusion detection dataset using combination of filtering methods.* International Journal of Computer Applications, 2013. 78(4).
188. Ivan, T. (1976). Two modifications of CNN. *IEEE transactions on Systems, Man and Communications, SMC*, *6*, 769-772.
189. J. Zhang, H. Li, Q. Gao, H. Wang, and Y. Luo, "Detecting anomalies from big network traffic data using an adaptive detection approach," Inf. Sci. (Ny)., vol. 318, no. August, pp. 91– 110, 2015.
190. V. Jyothsna and V. V. Rama Prasad, "FCAAIS: Anomaly based network intrusion detection through feature correlation analysis and association impact scale," ICT Express, vol. 2, no. 3, pp. 103–116, 2016.
191. A. Satoh, Y. Nakamura, and T. Ikenaga, "A flow-based detection method for stealthy dictionary attacks against Secure Shell," J. Inf. Secur. Appl., vol. 21, pp. 31–41, 2015.
192. A. Juvonen and T. Hamalainen, "An efficient network log anomaly detection system using Random Projection Dimensionality Reduction," 2014 6th Int. Conf. New Technol. Mobil. Secur., pp. 1–5, 2014.
193. Afolabi, H.A., Aburas, A.A, *Comparison of Single and Ensemble Intrusion Detection Techniques using Multiple Datasets.* International Journal of Advanced Trends in Computer Science and Engineering, 2021. 10(4): p. 2752 – 2761. *https://doi.org/10.30534/ijatcse/2021/161042021*
194. El Boujnouni, M. and M. Jedra, *New Intrusion Detection System Based on Support Vector Domain Description with Information Gain Metric.* Int. J. Netw. Secur., 2018. 20(1): p. 25-34.
195. Alhaj, T. A.; Alrawashdeh, M. M.; Al-Badarneh, N. M.; Algarni, A. D.; Tubaishat, M. S.; Chaczko, Z. *Feature selection using information gain for improved structural-based alert correlation.* PloS one, 2016. 11(11): p. e0166017.
196. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.
197. J. Demšar, "Statistical comparisons of classifiers over multiple data sets.," *The Journal of Machine Learning Research, 2006.,* vol. 7, pp. 1–30, 2006.

198. M. Zaman and C.-H. Lung, “Evaluation of machine learning techniques for network intrusion detection,” in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2018, pp. 1–5. doi: 10.1109/NOMS.2018.8406212.
199. J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, pp. 281–305, Feb. 2012.
200. W. J. Conover, *Practical nonparametric statistics.*, vol. 350. John wiley & sons, 1999.
201. M. Du, K. Wang, Y. Chen, X. Wang, Y. Sun, Big data privacy preserving in multi-access edge computing for heterogeneous internet of things, IEEE Commun. Mag. 56 (8) (2018) 62–67, doi:10.1109/MCOM.2018.1701148
202. M. Du, K. Wang, Z. Xia, Y. Zhang, Differential privacy preserving of training model in wireless big data with edge computing, IEEE Trans. Big Data (2018), doi:10.1109/TBDATA.2018.2829886.
203. P. Mishra, V. Varadharajan, U. Tupakula, E.S. Pilli, A detailed investigation and analysis of using machine learning techniques for intrusion detection, IEEE Commun. Surv. Tutor. (2018), doi:10.1109/COMST.2018.2847722.
204. Feng, X.; Xiao, Z.; Zhong, B.; Qiu, J.; Dong, Y. Dynamic ensemble classification for credit scoring using soft probability, Appl. Soft Comput. 65 (2018) 139–151, doi:10.1016/j.asoc.2018.01.021.
205. F. Salo, A.B. Nassif, A. Essex, Dimensionality reduction with ig-pca and ensemble classifier for network intrusion detection, Comput. Netw. 148 (2019) 164–175, doi:10.1016/j.comnet.2018.11.010.
206. N.T. Pham, E. Foo, S. Suriadi, H. Jeffrey, H.F.M. Lahza, improving performance of intrusion detection system using ensemble methods and feature selection, in: Proceedings of the Australasian Computer Science Week Multiconference, ACM, 2018, p. 2, doi:10.1145/3167918.3167951.
207. Acosta-Mendoza, N., et al., *Learning to assemble classifiers via genetic programming.* International Journal of Pattern Recognition and Artificial Intelligence, 2014. 28(07): p. 1460005.
208. Panigrahi, R. and S. Borah, *A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems.* International Journal of Engineering & Technology, 2018. 7(3.24): p. 479-482.
209. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. *A survey on deep transfer learning*. in *International conference on artificial neural networks*. 2018. Springer.

Appendix A: Proof of acceptances research papers based on this thesis for publications



Close [IJAI] Editor Decision: ACCEPT with minor revisions (Scopus indexed)

The identity of this sender has not been verified. Click here to learn more

Prof. Dr. Eugene Yu-Dong Zhang <ijai@iaesjournal.com> via smtpcorp.com
To: hassan afolabi (219048801)
Cc: Abdurazzag Aburas
Wed 12/21/2022 12:55 PM

The following message is being delivered on behalf of IAES International Journal of Artificial Intelligence (IJ-AI).

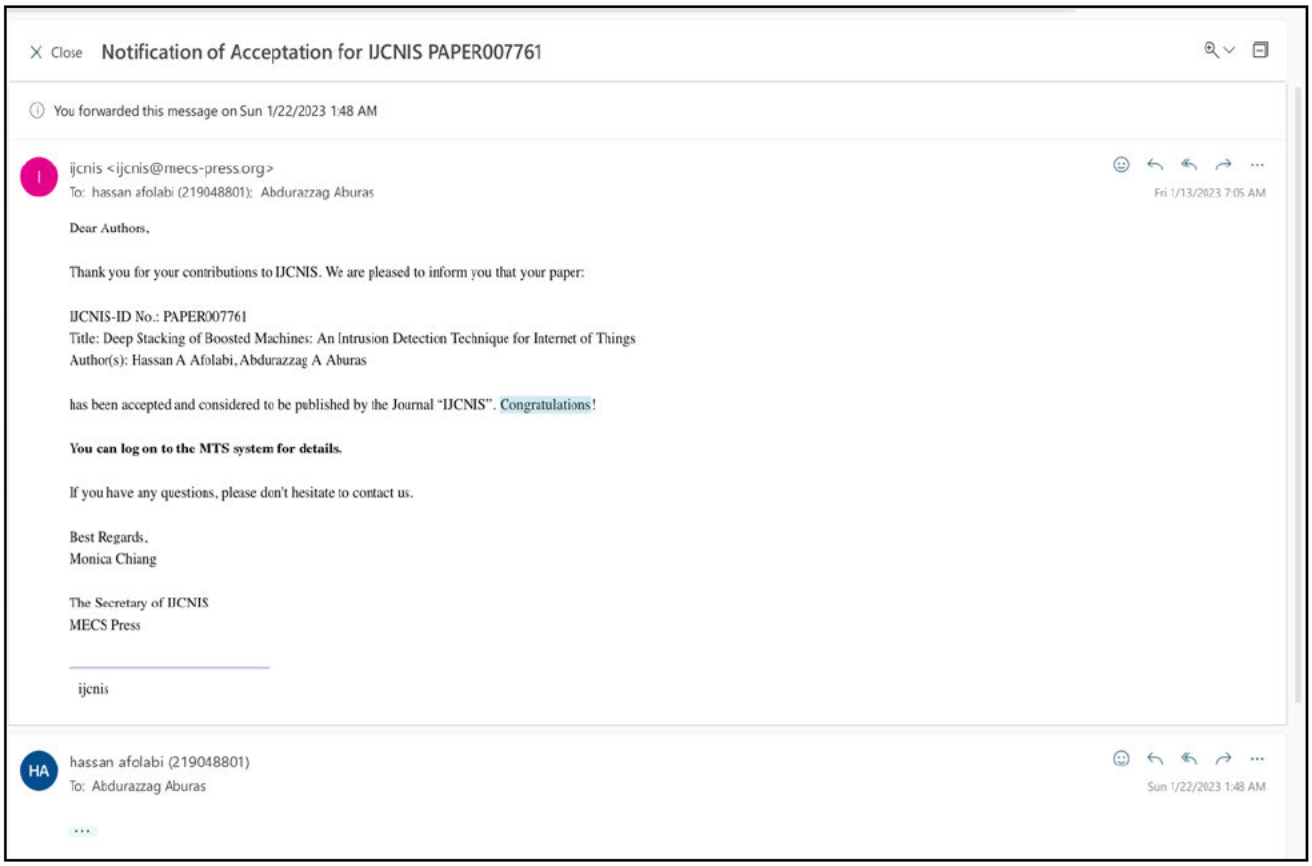
-- Paper ID# 22511
-- Please Strictly use the IJAI Template and Guide of Authors: http://iaescore.com/gfa/ijai.docx
-- Research PAPER: minimum 25 references and Review PAPER: minimum 40 references
-- Please upload the revised paper within 6 weeks
-- Failing to do proper revision may lead to the rejection of your paper

==========

Dear Prof/Dr/Mr/Mrs: Hassan Afolabi,

We have reached a decision regarding your submission entitled "Statistical Performance Assessment of Supervised ML Algorithms for Intrusion Detection System" to IAES International Journal of Artificial Intelligence (IJ-AI), a Scopus indexed journal (https://www.scopus.com/sourceid/21100901206). This journal is indexed by Scopus/Scimagojr, with a SJR of 0.341 (2020), Q2 under Electrical and Electronic Engineering, and Information Systems and Management categories.

Our decision is revisions required.
Please revise your manuscript according to the reviewers' comments. In a response letter that should be sent with your re-submission, you should talk about each of the reviewers' points. Please also adhere to every detail of the guide of the authors (MS Word: http://iaescore.com/gfa/ijai.docx, or LaTeX: http://iaescore.com/gfa/ijai.rar), and check it for spelling and grammatical mistakes. The goal of your revised paper is to present a polished paper that shows all of your new ideas and results (novel technical



Close **Notification of Acceptation for IJCNIS PAPER007761**

You forwarded this message on Sun 1/22/2023 1:48 AM

ijcnis <ijcnis@mecs-press.org>
To: hassan afolabi (219048801); Abdurazzag Aburas
Fri 1/13/2023 7:05 AM

Dear Authors,

Thank you for your contributions to IJCNIS. We are pleased to inform you that your paper:

IJCNIS-ID No.: PAPER007761
Title: Deep Stacking of Boosted Machines: An Intrusion Detection Technique for Internet of Things
Author(s): Hassan A Afolabi, Abdurazzag A Aburas

has been accepted and considered to be published by the Journal "IJCNIS". Congratulations!

You can log on to the MTS system for details.

If you have any questions, please don't hesitate to contact us.

Best Regards,
Monica Chiang

The Secretary of IJCNIS
MECS Press

ijcnis

HA hassan afolabi (219048801)
To: Abdurazzag Aburas
Sun 1/22/2023 1:48 AM

...

AFRICON 2023 notification for paper 47

From: **AFRICON 2023** <africon2023@easychair.org>
Date: Mon, 15 May 2023 at 22:57
Subject: AFRICON 2023 notification for paper 47
To: Abdurazzag Aburas

Dear Authors,

Congratulations!

We are pleased to inform you that your paper A REVIEW OF MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM FOR THE INTERNET OF THINGS (IoT) has been accepted for publications for IEEE. Consider all the following below.

Please kindly consider the recommendations of the reviewers in your final submission.

Pay for the paper at: https://2023.ieee-africon.org/registration-fees/#

Download the copyright form from the website at https://2023.ieee-africon.org/

Submit copyright form and camera ready paper to:
https://easychair.org/conferences/?conf=africon2023

For Visa Letter contact: africon-secretariat@ieee.org

All accepted papers will be placed on the final program. Accepted, registered and presented papers will be published in IEEE Xplore.

If you have any inquiry or difficulty, contact africon-tpc-chairs@ieee.org

Thank you.

Mary Ahuna, PhD, MIEEE
Technical Program Committee Chair, IEEE Africon 2023

SUBMISSION: 47
TITLE: A REVIEW OF MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM FOR THE INTERNET OF THINGS (IoT)