Towards Autonomous IoT IDSs Using Ensemble Learning and Feature Selection Methods

by

© *Alaa Zaid Mohammad Alhowaide*

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computer Science

Memorial University of Newfoundland

April 2021

St. John's Newfoundland

ABSTRACT

Intrusion Detection Systems (IDSs) are an efficient and effective solution against polymorphic and zero-day cyberattacks in IoT networks. Many IDSs have failed in practice due to a considerable number of false alarms, high False Positive Rates (FPR), and low Detection Rates (DR). Furthermore, with the rapidly growing number of connected devices in IoT networks and the wide variety of traffic types, it becomes challenging to develop a fast, light, and accurate IDS.

This research provides substantial contributions to cybersecurity research on developing a scalable, adaptive, and lightweight IDS framework for IoT networks. It considers two main aspects, a novel ensemble feature selection method and a new ensemble detection model approach to achieve a reliable IDS architecture.

The first contribution is developing a novel ensemble evaluation method for Feature Selection Methods (FSMs) to automatically construct an Ensemble Feature Selection Method (ENFSM). The proposed methodology combined five evaluation measurements. One of them is a new evaluation measurement that integrated the reduction rate with method speed and two new measurements that scored the whole feature set quality. Also, a novel cutoff mechanism for filter-based FSMs is proposed.

The second contribution is developing a novel ensemble Model Selection Method (MSM) to automatically construct an ensemble detection model. The proposed method used three new integrated efficiency measurements and combined the recommendations in a novel way to increase the method's reliability.

Notably, the proposed ENFSM achieved a reduction percentage ranging from 51% to 79% over the four datasets without compromising the accuracy of the detection models. Furthermore, the proposed cutoff mechanism showed a noticeable improvement in the feature selection methods' efficiency. The proposed ENFSM F and ROC-AUC scores ranged from 0.9 to 1 using most detection models. Furthermore, the generated feature set suited a vast range of models.

The proposed ensemble models showed 0.99, 0.95, 1, and 0.99 F scores and 1, 0.98, 1, and 1 ROC-AUC scores on NSL-KDD, UNSW-NB15, BotNetIoT, and BoTIoT dataset, respectively. The proposed models overcame most models in terms of efficiency and showed a stable performance using a vast range of feature sets.

LIST OF PUBLICATIONS

Journal Articles

- **A. Alhowaide**, I. Alsmadi, J. Tang. "Towards the design of real-time autonomous IoT NIDS", Cluster Computing (2021), pages 1-14, Jan 2021.
- **A. Alhowaide**, I. Alsmadi, J. Tang. "Ensemble Feature Selection Method for IoT IDS", Submitted for publication, 2020.
- **A. Alhowaide**, I. Alsmadi, J. Tang. "Ensemble Detection Model for IoT IDS", Submitted for publication, 2020.

Conference Papers

- **Alhowaide**, I. Alsmadi, J. Tang. "PCA, Random-Forest and Pearson Correlation for Dimensionality Reduction in IoT IDS", *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages. 1-6. Vancouver, BC, Canada, Sept. 2020.
- **Alhowaide**, I. Alsmadi, J. Tang. "An Ensemble Feature Selection Method for IoT IDS", *2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys)*, Fiji, Dec. 2020.
- **Alhowaide**, I. Alsmadi, J. Tang, "Features Quality Impact on Cyber Physical Security Systems", *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Oct. 2019.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
DL	Deep Learning
DM	Data Mining
DR	Detection Rate
EL	Ensemble Learning
FN	False Negative
FP	False Positive
FPR	False Positive Rates
HIDS	Host-based IDS
IDS	Intrusion Detection System
IoT	Internet of Things
ML	Machine Learning
NIDS	Network-based IDS
SGS	Signature Generation System
TN	True Negative
TP	True Positive

1 Chapter One: Introduction

In recent years, there has been tremendous growth in the number of connected devices to the Internet. A vast increase in Internet and network applications has led to the appearance of function-based networks called the Internet of Things (IoT). IoT has penetrated every aspect of life, including the human body, home, and the living environment. Furthermore, it has been adapted to a higher level of usage by various private and public sectors, such as military and nuclear facilities, civilian institutions/organizations, and governmental bodies [1]. The aim is to improve their environmental management, decision making, intelligence, among other goals. Such vast and vital dependence comes with severe risks and infinite security threats, increasing exponentially every day [2].

An IoT network provides powerful computations as well as valuable and sensitive data collected from interconnected devices within the corresponding IoT network [1]. However, each of these devices is developed based on a specific type of functionality. Such functionality is based on low cost and limited resources in computational capabilities, power, and storage. As a result, this makes such networks highly vulnerable to many security threats [3]. Securing IoT networks is vital due to the importance and sensitivity of the data collected by them. Such valuable data threat the success of the organizations or the sectors depending on that.

Furthermore, it may threaten the life and safety of those individuals who are using or depending on IoT networks. For example, it may menace countries' national security, where many have started to depend on IoT to monitor their borders, airports, streets, and more critical

systems [1] [4]. This dependence raises the threat of cyber-attacks' harm and losses as a nuclear explosion or even more [5].

The nature of IoT networks is that it connects a large number of devices with limited resources, and heterogeneity between various IoT networks raises many security challenges. Thus, many traditional security methods, such as cryptography, become useless against IoT cyberattacks. Accordingly, there is a mandatory and urgent need for real-time detection of cyber-attacks to secure IoT networks. Such a necessity is met by having a fast and efficient IDS that meets the high scalability and dynamicity of IoT networks' environment.

1.1 Problem Statement

Many proposed IDSs use data mining (DM), machine learning (ML), and deep learning (DL) claimed to reach an accuracy close to 100% of correct detections, as shown in section 3.2. However, Such systems did not pay attention to detection speed, and many are not practical in real-life applications because of the highly complicated methods used. Furthermore, most of the current IDSs use one classification/detection process to classify network traffic, which raises questions regarding the confidence of these IDSs' decisions and their reliability.

1.2 Aims and Scope

This research aims to develop an IDS that detects all types of known or unknown security threats. The previous research aim will be achieved by developing a reliable ensemble detection model that can predict with high confidence whether traffic is malicious or not. Additionally, such systems should be able to deal with significantly large data efficiently. Also, it aims to develop an ensemble feature selection method to speed up the detection phase and reduce the total system response time by reducing the dataset dimensionality. This, in turn, will minimize

intrusion impacts on vulnerable devices at early stages and reduce new threat propagation ability.

1.3 Research Questions and Problem Formulations

This research aims to solve the problem of developing an efficient and reliable IDS to detect existing and zero-day attacks in IoT networks. These networks are characterized by high speed and voluminous, dynamic traffic in which the pattern of normal traffic could change over time. The dimensionality of the network flows is high, which negatively affects the IDS performance. The developed IDS techniques should be able to function effectively in the presence of polymorphic as well as mimic attacks, which are intelligent adversaries tailoring their intrusive activity to avoid the detection mechanism. This problem is divided into the following three major sub-problems based on IDSs components.

Sub-problem 1: finding a quality network dataset to train and test proposed IDS is one of the biggest challenges for evaluating the credibility of IDS' theories. It should be captured from an IoT network with the norm of current network environments running at high speeds with large network flows. It should equally have a wide variety of authentic contemporary legitimate and malicious patterns to ensure the quality of new IDS approaches for deployment in real networks.

Sub-problem 2: selecting the relevant/significant network features is also a prominent challenge to the norm of current IoT network environments. The aim is to reliably eliminate useless or redundant features to improve the efficiency of detection methods and establish a lightweight IDS. Thus, a careful study is required to select features that can be used to distinguish between normal and abnormal observations automatically.

Sub-problem 3: developing an adaptive, lightweight, scalable, and reliable/trustworthy detection method that can distinguish between normal and malicious observations in real IoT

network environments. It relies on its capability to automatically adapt and efficiently identify anomalous patterns in large high-speed networks with confidence.

Based on the above discussion and background presented in Chapter two, to address the above sub-problems, the following research questions are elaborated:

1. How does ensemble feature selection methodology help develop a lightweight IDS, and to what extent is it effective?
2. How can ensemble detection models be used to establish an adaptive, lightweight, scalable, and trustworthy IDS? And to what extent?
3. How can an ensemble IDS be applied in the industry/practice efficiently to detect known and unknown attacks?

1.4 Thesis Contributions

This thesis contributes to the field of cybersecurity by adapting ensemble learning in feature selection and detection models as a promising solution for the above IoT IDS sub-problems. This research evaluated the impact of several filters, wrappers, and an ensemble Feature Selection Method (FSM) in four different datasets. It aimed to find the optimal feature set that improves ML models by reducing data dimensionality, noise, and computational requirements. Furthermore, this research sought to fill the literature gap by not considering feature selection or Ensemble Feature Selection Methods (ENFSM) for IoT IDS.

The contributions of this research to cybersecurity in terms of feature selection consists of the following:

(i) Dynamic cutoff to select the best feature set recommended by a filter-based FSM.

(ii) Ensemble FSM evaluation/selection method based on new and novel feature set evaluation measurements.

(iii) Ensemble feature selection method for IDS for IoT.

(iv) Huge dimensionality reduction of the used datasets using the proposed ENFSM.

(v) Extensive experimental evaluation of the impact of various FSMs on different classifiers.

In the same vein, the contributions of this research related to adapting ensemble learning for detection models consist in the following:

(i) Fully automatic MSM based on combining several evaluation measurements.

(ii) Novel integrated model evaluation measurements

(iii) Ensemble detection models for deployment at two different levels of an IoT network infrastructure.

(iv) Extensive experimental evaluation of a vast range of classifiers conducted on four datasets by applying feature selection.

1.5 Thesis Structure

The thesis organization is presented as follows. Chapter two introduces the relevant background knowledge needed to develop this research project and to answer the research questions. Chapter three is devoted to the literature review and the discussion of the relevant studies. Subsequently, chapter four proposes the methodology endorsed in the effort of ensemble feature selection and

detection model. Experiments and results are presented in the fifth chapter[12]. Finally, chapter six provides the study conclusions and some future research recommendations.

[1] The first two sections' results were published in the 2019 IEEE 10th Annual Information Technology, Electronics, and Mobile Communication Conference (IEMCON) and a manuscript in Cluster Computing journal.
[2] Parts of the third section's results were published in the 2020 IEEE International IoT, Electronics, and Mechatronics Conference (IEMTRONICS) and the 2020 IEEE 6th International Conference on Dependability in Sensor, Cloud, and Big Data Systems and Application (DependSys).

2 Chapter Two: Background

This work embraces EL in IDSs. The used EL methods are presented in this chapter to provide a thorough understanding of the proposed solutions. Furthermore, other relevant and integral background knowledge about IDS, IoT, and feature selection problems are discussed as well.

2.1 IoT Networks

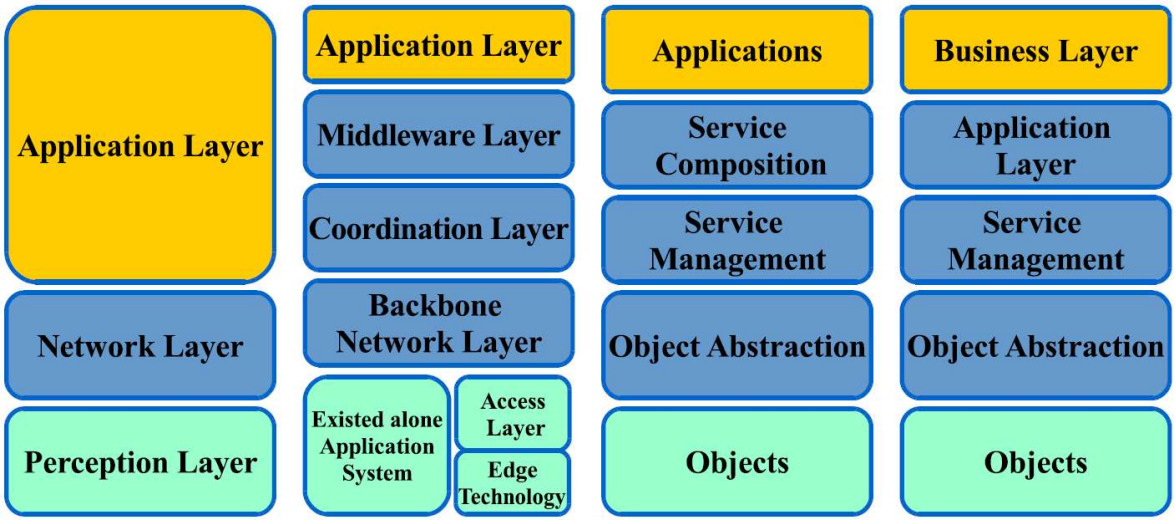
With the appearance of the first computer and the invention of the Internet, a dramatic change in human life took place. Computers are evolving in an accelerated trend bringing various new forms of technology. IoT is one of these new technological forms. IoT is defined as interconnected heterogeneous things, such as sensors, devices, computers, etc; through the Internet to provide a "smarter life" [3], [6], [7] that enables us to achieve our daily tasks more efficiently and effectively.

IoT applications are evolving rapidly, trying to connect everything from thermostats, toaster, fridge to complete systems to the Internet [8]. Some of the most common IoT applications include personal healthcare [9], smart transportations [1] [9], smart grid [10], smart industrial automation [11], and intelligent emergency response systems [9] ranging from monitoring to decision making. All these systems aim to provide a more comfortable lifestyle and improve our capabilities to experience a considerably better life [1], which looks "smart.". The number of connected devices reached 9.5 billion devices at the end of 2019, according to IoT Analytics [12]. IoT Analytics expects the previous figure to reach 22 billion by 2025 [13]. This rapid growth came aligned with weak security measures and defenses, which resulted in a significant increase in cyberattacks [3]. According to research performed by Checkpoint [14], since the appearance of COVID-19, 71% of security professionals have noticed an increase in security threats or attacks. Most of the attacks targeting IoT networks have tried to delay the devices'

service or compromise and convert them into botnet members under the attacker's control to amplify their attacks' speed. As a result, network security becomes essential. Thus, intrusion detection becomes vital for network security for handling malicious behaviors and abnormal acts and detect diverse intrusion [15].

2.1.1 IoT Architecture

There are many proposed architecture models for IoT networks, and Figure 2-1 shows the common IoT architectures. The basic model consists of three layers: application, network, and perception [16], [17], [18]. Additional layers have been added to the basic model in recently proposed architectures [9], [16],[17]-[19]. A description of the five-layer model is shown in Figure 2-1 (d), which is considered the most practical architecture among all the proposed architectures.



(a) (b) (c) (d)

Figure 2-1 The common IoT architectures. (a) Three-layer. (b) Middle-ware based. (c) SOA based. (d) Five-layer. [1]

The Objects Layer represents the IoT physical devices which intends to collect data from the surrounding environment. Sometimes these physical devices may be able to handle information processing. Furthermore, these devices are functionality-based sensors and actuators, which are designed to perform single to several tasks such as querying location, temperature, weight,

vibration, acceleration, humidity, etc. The objects layer requires a standard plug-and-play mechanism to configure heterogeneous objects [17]. Furthermore, it digitizes and transfers the collected data to the next layer [18].

The Object Abstraction Layer aims to transfer the received data from the objects layer to the next upper layer (Service Management Layer) through secure channels. This layer uses several technologies to transfer data, such as RFID, 3G, GSM, UMTS, WiFi, Bluetooth Low Energy, infrared, ZigBee, etc. Moreover, it may handle functions, such as data management and cloud computing [17].

The Service Management/Middleware Layer uses addresses and names to pair service with its requester. It isolates the hardware platform details from IoT application programmers and enables them to work with heterogeneous objects. Furthermore, it processes the received data, makes decisions, and delivers the required services over the wired network's protocols [16], [18], [19].

The Application Layer provides the customers with the requested services, such as delivering humidity and temperature measurements. It is responsible for providing high-quality smart services to customers such as smart cities, smart transportation, industrial automation, and smart healthcare [16]–[18], [20]. In the five-layer model, this layer represents the interface for the end-users to interact with objects as well as to the Business Layer. Due to its complex and enormous computational needs, this layer is hosted on powerful computing systems.

The Business/Management Layer is responsible for managing the whole IoT system service and activities. It uses received data from the Application Layer to generate business models, charts, graphs, etc. Additionally, it designs, analyzes, implements, evaluates, monitors, and develops IoT system-related elements. It supports decision-making processes based on Big Data

analysis besides monitoring and managing the underlying four layers. Furthermore, it enhances services and maintains users' privacy by comparing the output of each layer with the expected output [16], [18].

The three-layer architecture displayed in Figure 2-1 (a) does not comply with practical IoT networks/systems. For instance, the Network Layer cannot handle all used data transferring technologies used in an IoT platform. Moreover, this architecture was designed for specific communication types such as (Wireless Sensor Networks) WSNs. Some of the other architectures may have unrealistic layers that do not consider the objects limited resources, such as Service Composition Layer in (c). Such a layer will take a considerable fraction of the objects' energy and time to communicate with other objects and integrate the required services. The five-layer architecture in (d) is the most applicable model for IoT systems after considering the previously mentioned points [1].

2.1.2 IoT Challenges

IoT networks face crucial challenges, including scalability, reliability, privacy, and security [8]. These challenges are a direct result of IoT networks' nature, where they are constructed based on low-cost and function-based computing devices, trading off security, with advanced communication capabilities. Operating in a completely isolated environment, its open development as well as deployment, and limited resources, such as energy, and computation, make the IoT vulnerable to malicious attacks [21]. Moreover, it is difficult to apply a standard security mechanism due to its heterogeneous and distributed character make [3] as well as the lack of a single standard IoT architecture [7].

As the number of smart objects is rapidly increasing in every aspect of life to collect, process, and communicate sensitive/vital data, attacks' impact on IoT may cause a catastrophic disaster and loss of lives [6].

2.2 Intrusion Detection Systems (IDS)

IDS represents the first defense line against cyber-attacks. It refers to a strategically allocated device or software at a strategic point on a network to monitor all traffic passing through it [3], [22]. Based on the three computer security principles: Confidentiality, Integrity, and Availability (CIA) [23], [24], an IDS is defined as a technique for monitoring and inspecting activities occurring in a computer or a network system. The main goal is to detect possible threats and malicious activities by measuring their violation of the CIA's computer security principles [25]. When detection occurs, the IDS generates and sends an alert to the network administrator or a defense response system.

Machine Learning (ML) plays a crucial role in building IDSs to detect such threats [26]. ML techniques are categorized into being supervised, unsupervised, and semi-supervised. IoT networks use diverse standards and protocols, forming heterogeneous networks, it becomes difficult for a single model to learn all the different traffic patterns. Thus, an ensemble model that combines different learning models gains the composing models' advantages in order to improve classification accuracy. Moreover, ensembling includes combining multiple models of the same type trained on different training dataset partitions.

2.2.1 Detection Methods

IDSs are classified based on the detection methods to two main types: signature-based and anomaly-based IDS. Signature-based IDS, also known as misuse detection, analyzes the network traffic looking for a specific pattern called the signature, representing or describing specific malicious traffic. This type of IDS has a fast detection speed, but they cannot detect new attacks, known as zero-day attacks, because the signature-based IDS does not have the zero-

day attacks' signatures yet. With the increasing number of new attacks every day, these IDS types' recommendation decreases [15].

On the other hand, anomaly-based IDS, also known as behavioral detection, does the opposite where it analyzes the normal traffic in a process called profiling. It compares the network traffic to the normal traffic profile. When a mismatch occurs, the IDS then sends an alert and considers the corresponding traffic as suspicious. This type of IDS has the advantage of discovering new attacks, but at the same time, it suffers from a high rate of false alarms [15]. A hybrid IDS combines both signature-based and anomaly-based techniques to overcome each IDS's drawbacks, but such an IDS consumes more energy and computational resources [3].

IDSs can be further classified into stateful and stateless IDSs based on the amount of network traffic required for detection [27]. A stateful IDS needs to build network sessions to make a detection, while stateless IDS does not. Stateless IDSs make detections based on the per-packet analysis. Thus, stateful IDSs are more accurate than stateless IDSs. On the other hand, stateful IDSs are slower and require more storage to construct network sessions and computational power.

2.2.2 IDS Anatomy

A classical IDS architecture is composed of four main components [15], [25], as indicated in Figure 2-2. The main components are packet decoder, feature extractor, feature selector, and detector. Unlike the suggested classical architecture by [25], the defense responder is not considered as a part of an IDS architecture in [15]. This component is responsible for deciding what response actions to take when detection occurs. This is not within the IDS main tasks based on the previously mentioned definitions.

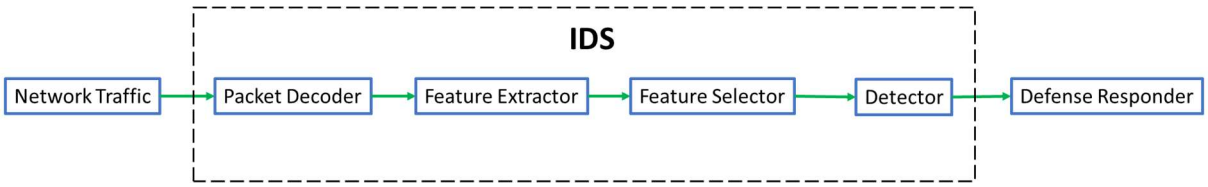


Figure 2-2: Architecture of classical IDS.

The packet decoder receives the raw network traffic using data collection tools, such as Tcpdump, and transfers each packet to the feature extractor. The feature extractor extracts a predefined/desired set of features from the packet header or its content/payload and can apply feature engineering techniques to generate more features. Those features extracted directly from the packet header or payload are referred to as basic features. In contrast, those features extracted by applying feature engineering methods are referred to as engineered features. This component uses diverse feature engineering techniques, such as merging multiple features, statistical, time-based, and content-based measurements [28]–[31].

The feature selector receives the extracted features and tries to reduce the feature set to its minimum. This process is also known as dimensionality reduction. The objective is to keep those features highly correlated with the class label and remove redundant features that are highly correlated with each other. This component is only used during training IDS and not during testing or running IDS in practice. The feature selector has a significant impact on the whole IDS speed and the total detection time.

The detector in the training phase of an IDS builds a model that can distinguish between normal and malicious traffic. It applies one or more detection techniques (misuse-based, anomaly-based, stateful analysis, or hybrid). The testing/running phase uses the built model to classify the incoming network traffic into normal and suspicious. The IDS sends an alert to an administrator or a security response system to take the required action(s) once it detects a suspicious traffic/packet.

2.2.3 IDS Deployment

An IDS can be used to monitor both host- and network-based systems [15]. A Host-based IDS (HIDS) gathers information about events that occur in a computer system. It must be installed on each host to monitor and log its operating system's behavior to provide audit trails for recognizing anomalous and suspicious observations. In contrast, a Network-based IDS (NIDS) monitors a network's traffic to detect remote attacks occurring in its environment. It is a powerful security solution that provides a solid defense against attacks before they access the system resources. The HIDS is an unpractical solution in IoT case where devices have minimal computing, storage, and power resources. Thus, this research aims to design a NIDS solution for large, high-speed network environments, known as large-scale networks. Its architecture considers the current paradigms of communications and computing to performing detection in wireless and wired sensing nodes [15].

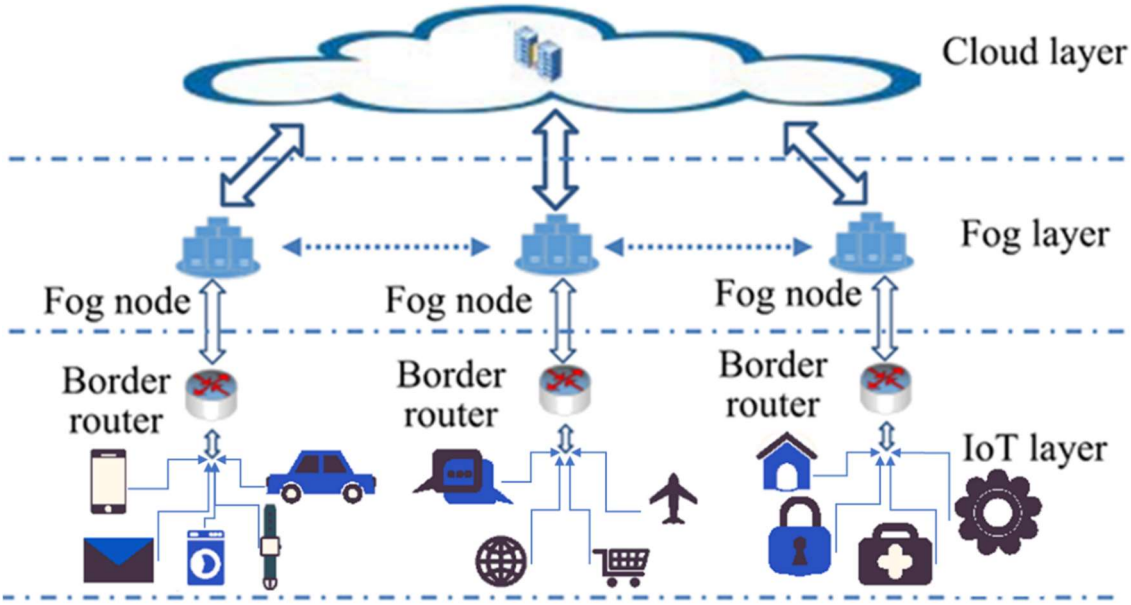


Figure 2-3 IoT network infrastructure [11].

The deployment architecture of an IDS is either centralized or distributed. The latter is composed of multiple intrusion detection subsystems mounted at different locations. These

subsystems are connected to exchange information and updates. Such IDS detect malicious events that can identify corresponding attacks from multiple locations at a particular time. Moreover, such a system has a communication overhead on the hosting machine. Conversely, a centralized IDS refers to a non-compound system mounted on only one site. The whole detection process occurs in one site; thus, this type of architecture has a high computational overhead at the hosting machine in tradeoff communication.

Regarding the IoT networks and considering its infrastructure, represented in Figure 2-3, there are three possible deployments for an IDS; the router, fog, and cloud-based. The edge nodes can perform basic computations and serve as the gateway for the IoT devices [32]. Its low computation power and storage cannot handle the huge processing overhead required by an IDS. Furthermore, installing an IDS at a router causes communication overhead between IoT devices and the router [11]. On the other hand, when deploying IDS at a cloud, it can lead to the disability of detecting an IoT network's internal malicious activities, even though this is a very powerful node of centralized networks providing the IDS as a service. A fog node has a medium computation power and storage resources when compared with router and cloud nodes. Furthermore, it can be traced to provide location awareness for IoT network devices. In specific terms, it can be geographically distributed to provide higher scalability and availability for mobile IoT devices [6].

A single fog node does not work efficiently when running a centralized IDS. CISCO developed a fog computing paradigm that transfers data and services between the edge and the cloud. This distributed paradigm aims to handle the voluminous data coming from the rapidly growing number of IoT devices [33]. Even having several fog nodes working together to run a distributed IDS suffers from high communication overhead between them, affecting the IoT communication latency [25].

Given the details above on IDS architecture, functioning, and drawbacks, this research aims at addressing some of these IDS gaps. In more specific terms, this research focuses on ensemble learning (EL) approaches and how to use them to design a fast, accurate, and confident ensemble NIDS for fog/edge and cloud deployments.

2.2.4 Threats

There are many known threats and other ones still unknown. Knowing the threats is integral to protecting the systems, which helps design more robust IDS against any potential threats. As there is a wide range of threat types, researchers proposed different ways to categorize them to facilitate their understanding, detection, and response. In [34], seven types of wireless network threats were proposed based on working techniques, which are Traffic Analysis, Passive Eavesdropping, Active Eavesdropping, Unauthorized Access, Man-in-the-middle, Session High-Jacking, and Replay. IoT threats were classified from a different perspective in [35] into five categories based on IoT requirements, which are Identification, Communication, Physical threat, Embedded Security, and Storage Management. However, the first threats classification suggested in [36] categorized them into four types, Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), and Probing. Recent research tends to classify IoT threats based on the Open Systems Interconnection (OSI) model layers, such as in [23]. The former classification facilitates for researchers to gain a deep understanding of the threats' sources, tools, and behaviors, which in turn helps to build a more accurate IDS with fewer false alarms, and better datasets. The OSI has seven layers; Application, Presentation, Session, Transport, Network, Datalink, and Physical. Significantly, a threat may affect one or more OSI layers, even if the threat is targeting a single layer. In Figure 2-4, Figure 2-5, and Figure 2-6, threats are classified based on three main criteria: threat target (Network, Host, Software, Physical, and Human), OSI affected layer, and threat mode (active and passive). An active threat, such as DoS,

impersonation, and virus, targets the performance, information, or other aspects of a targeted device. If the threat, such as adware, spyware, and information gathering, aims to collect information or monitor the network, it is therefore considered a passive threat [37]. Some types of threats cannot be categorized into active or passive until they are executed. For instance, a SQL-injection attack used to change or delete data from a database server is considered an active threat. However, when it is used to query data, it is thus classified as a passive threat. An identification number is shown next to a threat name in Figure 2-4, Figure 2-5, and Figure 2-6 stands for the type/subtype threat classification.

2.2.4.1 Network threats:

As illustrated in Figure 2-4, most of its attacks target the network layer. This type of threat is recognized based on a flow of packets over the network. Below are descriptions of the most common attacks:

- Denial of Service (DoS) and Distributed Denial of Service (DDoS) (1.1): These are the most common forms of network threats. These attacks are based on flooding the network with requests, rendering a service to slow its response to its authorized users. There are four main types of DoS and DDoS attacks [38]:
    - Flood attacks (1.1.1) occur when an attack generates/uses more memory resources than expected, such as Smurf attacks (1.1.1.1), which produce a large number of ping requests. Overflows (1.1.1.2), which write more bytes than allowed, happens when an attacker sends packets larger than 65536 bytes (allowed in the IP protocol), and the stack does not have appropriate input sanitation in place.
    - Amplification attacks (1.1.2) occur when packets are too large for the routers, and splitting is required, such as Ping of Death (1.1.4.1).

- Protocol exploit (1.1.3) attacks occur when an incorrect offset is set by the attacker, such as Teardrop (1.1.3.1).
- Malformed packets (1.1.4) attacks happen when the host allocates memory for a huge number of TCP SYN packets, such as SYN flood (1.1.1.3) attack.

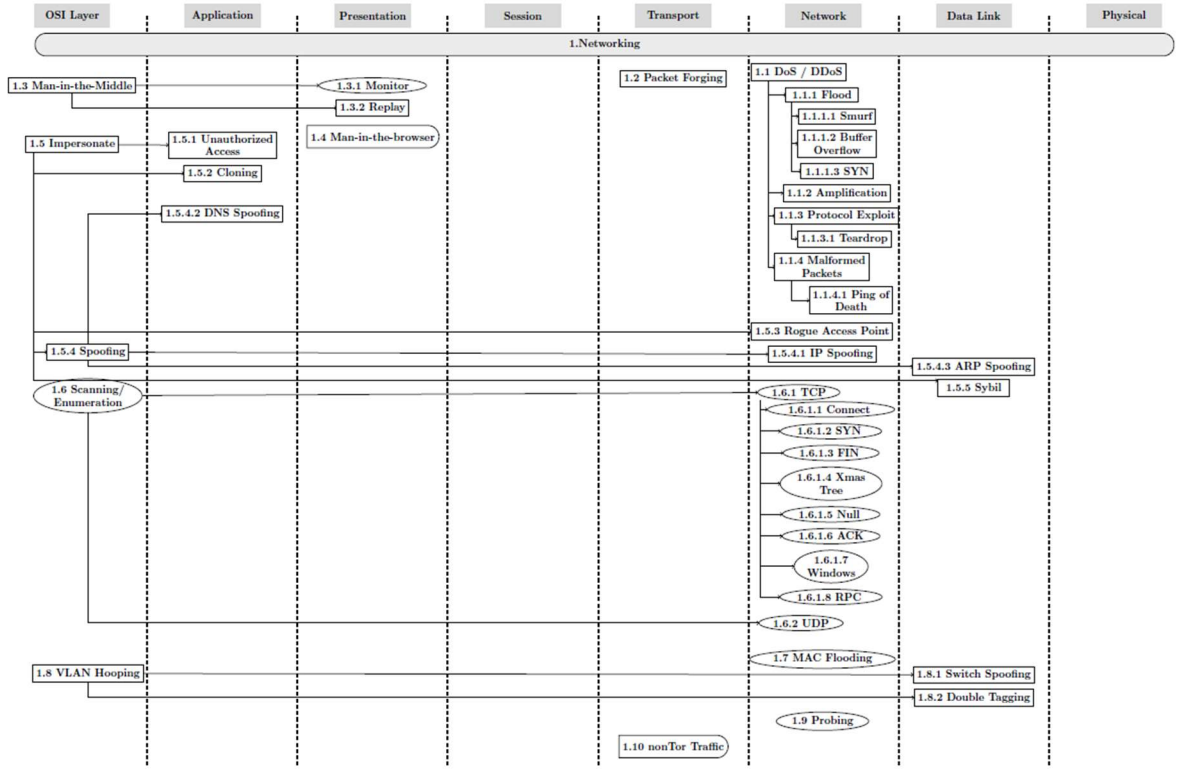


Figure 2-4 Taxonomy of Network threats, where rectangles stand for active threats, and oval stands for passive threats. [37]

- Packet forging (1.2), in which the attacker generates packets, looks similar to normal network packets and may be used later on to execute a specific action, such as stealing information.
- "Man in the Middle" attack (1.3) is when an attacker intercepts communications between two or more entities and starts to either control, alter, or spy on the communication between them.

- Unlike "Man in the Middle", “Man In The Browser” attack (1.4) intercepts the browser to alter or add fields to a web page to ask the user to enter confidential data [38].
- Impersonation (1.5) is pretending to be another user and can take a different number of forms:
    - The attacker may act as a user to gain a higher security level to access unauthorized data (1.5.1)
    - Cloning (1.5.2) is common in social networks that use cloned accounts.
    - Rogue access points (1.5.3) is common in wireless networks.
    - IP spoofing (1.5.4.1) an IP address is spoofed by an attacker to enable him to send packets impersonating a legitimate host. Another type of spoofing is DNS spoofing, known as DNS cache poisoning (1.5.4.2), where the attacker poisons the DNS to redirects packets. Finally, ARP spoofing (1.5.4.3) separates the legitimate victims’ IP and MAC addresses in the ARP tables, intending to perform attacks like Man In The Middle.
- Scanning/enumeration attack is considered an essential step for starting an attack. It refers to searching the network for information, such as active nodes, the running operating system, software versions, etc., by an attacker. It has many forms based on TCP's used protocols (1.6.1) or UDP (1.6.2).
- Media Access Control (MAC) address flooding (1.7) attack in which an attacker aims to redirect packets to the wrong physical ports by targeting the network switches.
- VLAN hopping attack (1.8) that targets the networked resources on a virtual LAN (VLAN). An attacking host on a VLAN tries to gain access to traffic on other VLANs

that would normally not be accessible. This attack has two primary forms, either switch spoofing (1.8.1) or double tagging (1.8.2).

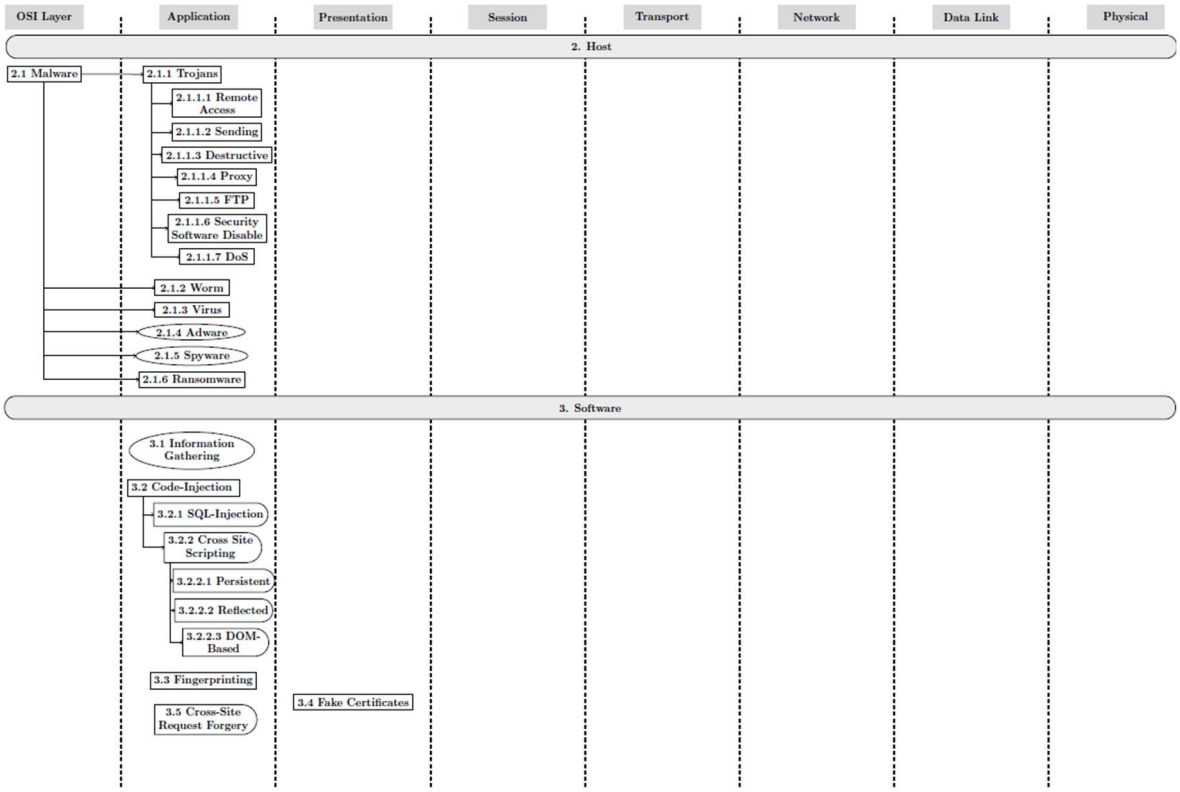


Figure 2-5: Taxonomy of Host and Software threats, where rectangles stand for active threats, and oval stands for passive threats. [37]

2.2.4.2 Host threats:

Specific hosts or systems are targeted through executing malicious software to corrupt or compromise the system functionalities. The most widely spread host threats category is the malware (2.1) category, as show in Figure 2-5. This category includes:

- Viruses, which affect programs and files when shared with other users on the network
- Worms are self-replicate, affecting multiple systems [39].
- Adwares display advertisements to users when surfing the Internet or installing software. It may compromise the performance of a system, but this is unlikely to happen.

- Spywares gather information such as documents, user cookies, browsing history, emails, etc., or monitor and track user actions.
- Trojans are often similar to trusted applications but allow the attacker to control the device.
- Ransomware is a relatively new type of malware where the system is kept under the control of the attacker by encrypting the files until the user/organization pays a ransom.

2.2.4.3 Software threats:

The attacks belong to this category, Figure 2-5, try to add malicious functionalities to software to steal information or to corrupt it. For example, Code injection (3.2) obtains or deletes confidential data by dropping columns, rows, or tables by SQL Injection (3.2.1). While Cross-site scripting (XSS) (3.2.2) executes malicious code to steal cookies or credentials, three main types of Cross-site scripting (XSS) are persistent/stored XSS (3.2.2.1), which save the script in the database. It executes every time the page is loaded, reflected XSS (3.2.2.2) in which the HTTP requests are sent to the server. The latter contains the script and DOM-based XSS (3.2.2.3) in which the attacker alters values in the Document Object Model (DOM), for example, document URL, document location, etc. DOM-based XSS is difficult to detect because the script is never transferred to the server. Fingerprinting (3.3) and misconfiguration (3.4) are also forms of software threats. Fake server certificates (3.5) should be considered during web application development or analyzing communications.

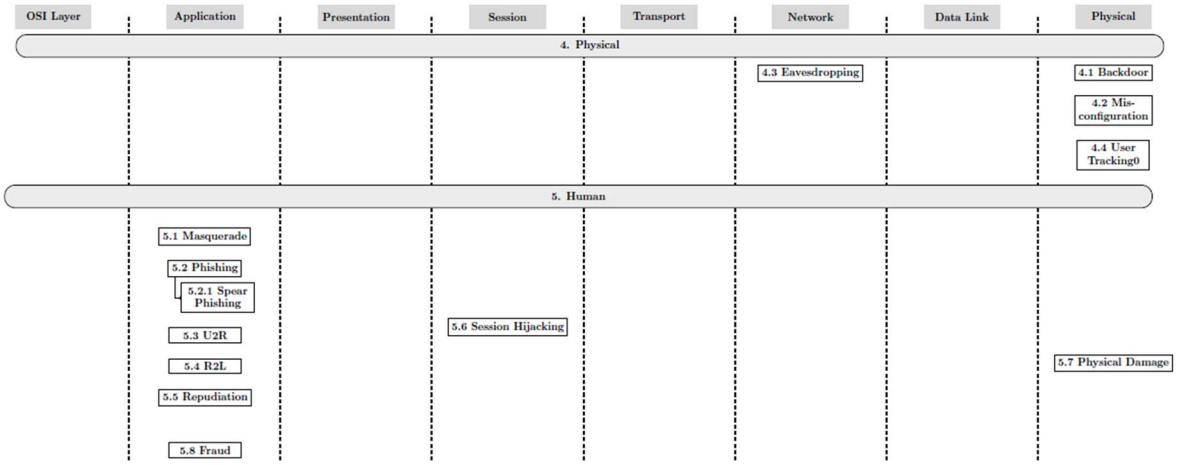


Figure 2-6: Taxonomy of Physical and Human threats, where rectangles stand for active threats, and oval stands for passive threats. [37]

2.2.4.4 Physical threats:

As shown in Figure 2-6, this category of attacks targets the network hardware, such as an edge, or other devices, or its configurations. It aims to alter the settings (4.2) and to find backdoors (i.e., The Evil Maid).

2.2.4.5 Human threats:

This category depends on human actions, Figure 2-6. It includes:

- User masquerade (5.1) is an attack that uses a fake identity, such as a network identity, to gain unauthorized access to personal computer information through legitimate access identification.

- Phishing (5.2) in which the attacker tries to obtain credentials or confidential data by using emails or other electronic messaging services.

- User to Root (5.3) and Remote to Local (R2L) (5.4) attacks are when a user attempts to take higher privileges.

- Repudiation (5.5) can be used to change the authoring information of actions executed by a malicious user in order to log the wrong data to log files. Any application or system that

does not adopt controls to properly track and log users' actions is vulnerable to this type of attack.

- Session hijacking or sniffing is gaining access over an active session to gain access to cookies and tokens.

2.3 Ensemble Learning (EL)

In everyday life, the opinions of several experts are always sought before finally making some decisions. For instance, before agreeing on a medical procedure, several physicians' opinions are taken into account. Prior to buying a product, multiple users' reviews are consulted. The main objective of doing these acts is to minimize the regrettable selection of an unnecessary medical procedure or a terrible product. The selection is based on a strategic way to reach a final decision, which is a process known in data science as "ensembled learning" (EL).

EL is the process of strategically combining and generating multiple models or methods, such as classifiers, to find a solution for a specific problem. The main objective is to improve the performance of a model/method or increase the probability of a fortunate selection of a good one. Moreover, it aims in some applications to assign confidence to the decision made by the used model, data fusion, selecting optimal features, etc. [40]. In this section, the main concepts of EL are discussed, along with its related applications to this research. This research will use EL for feature selection and malware detection, which represent the two main components of the proposed IDS.

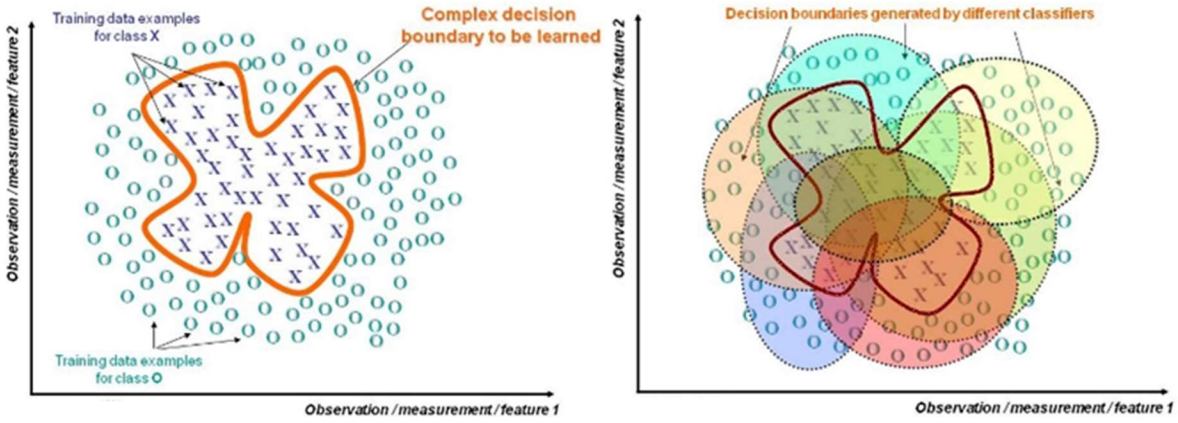
2.3.1 Why EL

Various are the reasons that lead to choosing EL as a strategy to solve complex problems occurring in practice. The list below includes the main motives of choosing EL:

- Model selection: This is a computational reason. Model selection is a critical issue before solving any problem. Why a researcher selected a specific method/model, such as a classifier, to solve a particular problem? This was the most commonly used one in literature, showed the best performance in many different problems, or showed the best performance on the training data compared with several classifiers. All the previous justifications are flawed, even if the last justification sounds a realistic one. Furthermore, even when the model performance was found using the cross-validation approach, doubts still exist regarding the performance on previously unseen data. Another matter, what if several classifiers on the same training dataset have the same performance, which one of them should be selected? A random model can be chosen; for example, a random choice comes with the risk of selecting a weak model. While combining such models' outputs will decrease the risk of an unfortunate selection of a weakly performing classifier. Figure 2-8, which will be discussed in further detail later, shows how three classifiers with different decision boundaries are combined to generate a better decision boundary than any of the three individual decision boundaries. It is important to clarify that an ensemble classifier's performance is not guaranteed to be always better than the best individual classifier. Not even an improvement is guaranteed, but for sure, an ensemble classifier will reduce the total risk of making a weak selection [41].

- Data size: EL is useful when dealing with large volumes of data or lack of required data, which is known as the statistical reason. When the data is too large and difficult to train a single classifier, the data can be partitioned into smaller subsets. Then for each partition, one can train a classifier that can be combined using a suitable combination rule, described in the incoming section. In the other case, when having

a lack of data, then bagging/bootstrapping can be used. A different classifier can be trained using different bootstraps samples of the data. Each bootstrap consists of randomly and independently drawn data samples with replacement from the training dataset [41].



(a) (b)

Figure 2-7 a) A complex decision boundary. b) A combination of multiple circular boundaries can solve the complex problem [40].

- Complex/non-linear problems: This is a representational reason. Some problems are too complicated for a single classifier to solve. The decision boundary that separates data from different classes may be too complicated or lie outside the chosen model's function space. Figure 2-7 (a) shows a two-class problem with non-linear boundaries in a two-dimensional space. A single classifier may not learn this complex non-linear boundary. In contrast, a proper combination of an ensemble classifiers can learn any non-linear boundary. For example, consider a classifier that generates circular decision boundaries. Such a classifier cannot solve the problem alone, but when considering a combination of a collection of circular classifiers generated, as shown in Figure 2-7 (b) can solve this non-linear problem. Each classifier labels the data as class O or X, and a decision is made using majority voting. Thus, such linear classifiers can smoothly learn this complex non-circular boundary [40].

- Data fusion: It is usual to receive data from various sources in automated decision-making. Data or information fusion is defined as adequate to combine such information. A decision made based on fused data can be more accurate compared to a decision made based on a single data source. Furthermore, when dealing with heterogeneous features, they may not be used all together to train a single classifier. EL can be used to solve these problems, where an independent classifier is trained on each set of features and then combine all classifiers' decisions to make a final decision [40].

- Confidence estimation: The structure of an ensemble model allows us to assign a confidence probability of the generated decision. If most of the classifiers agree on a decision, then the ensemble model has high confidence in its decision. But if half of the models decide differently from the other half of models, then the ensembled model has low confidence in its decision. It is significant to note that a high confidence score does not mean that the decision is correct, and it is not incorrect when the confidence is low. However, it has been shown that decision is usually right when its confidence is high and typically wrong when its confidence is low [40].

2.3.2 Common EL Algorithms

An ensemble system succeeds when it can correct some of its members' errors, which can be achieved when individual classifiers make different errors. The idea is that when each classifier makes different errors, then the strategic combination of these classifiers can reduce the total error. Thus, an ensemble system requires a set of classifiers said to be diverse, given that their decision boundaries are adequately different from each other. There are four common ways to achieve such diversity. The first method, being the most popular one, is using a different training datasets to train individual classifiers. These datasets can be obtained through re-sampling

methods such as bagging and bootstrapping that draw the data subsets randomly and usually with replacement. For instance, considering Figure 2-8 below, it graphically explains this method, where several classifiers trained in different data subsets of the training data. Such a fact leads to different errors, but with a strategic combination of these different classifiers, that will result in the best decision boundary.

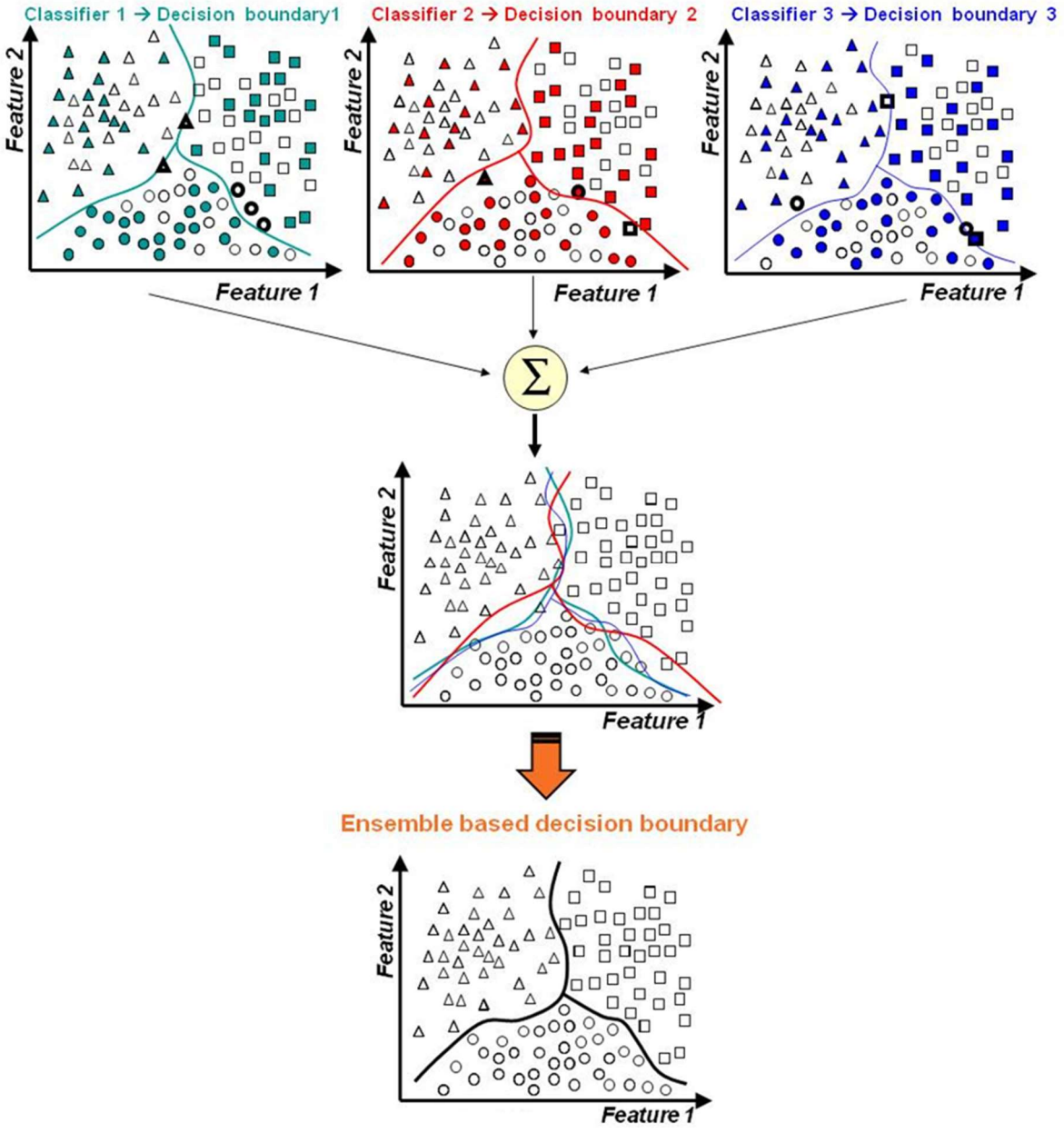


Figure 2-8 Combining ensemble classifiers for reducing classification error [40].

The second method is about using a different hyperparameter for a classifier. For example, in a neural network (NN), a classifier can be trained using different weight initializations, number of layers/nodes, error goals, etc. This method enables generating different individual classifiers, hence achieving diversity. The third method is based on using completely different types of classifiers, for instance, combining NN, support vector machine, decision trees, and k-nearest neighbor. The fourth method uses different features or different features subset to train the individual classifier, as it is done in random forests. Below are the commonly used EL algorithms/methods related to this research:

- Bagging is the simplest, commonly-used ensembled algorithm, which is also known as bootstrap aggregation. It achieves diversity by training individual classifiers on different training data subsets (bootstrapped replicas), which are randomly drawn with replacement. The decisions of individual classifiers are combined using vast majority voting. In other words, the chosen class label is selected by most of the individual classifiers. Usually, weak classifiers are chosen to compose the ensembled model to increase diversity, where the training data may overlap substantially [41].
- Boosting trains individual classifiers on resampled data and combines decisions using majority voting, like bagging. The difference from bagging is that resampling is strategically controlled to provide the most informative training data for each consecutive classifier. In Schapire boosting algorithm, each iteration creates three classifiers. The first classifiers *C1* is trained with a random subset from the training data. While the second classifier *C2* is trained on a data subset, half of it is correctly classified by *C1*, and *C1* misclassifies the other half. This subset is the most informative subset giving *C1*. The third classifier *C3* is trained on a data subset contains those instances on which *C1* and *C2* disagree. This ensemble algorithm has

an error upper bound, where an algorithm *A* used to create the classifiers *C1*, *C2*, and *C3* has an error ϵ, then the ensembled upper error bounded is $f(\epsilon)=3\epsilon^2-2\epsilon^3$. Note that $f(\epsilon)\leq\epsilon$ for $\epsilon<0.5$, which holds only if *A* performs better than random guessing. Thus, the boosting ensemble, which combines the three classifiers *C1*, *C2*, and *C3* created by *A,* always outperforms *A*. Hence, out of weak classifiers, a stronger classifier is generated. Boosting only works for binary classification problems, which is considered a limitation [41].

- AdaBoost, standing for Adaptive Boosting. It extends boosting to regression and multi-class problems. There are several variations of AdaBoost, and the most popular one is AdaBoost.M1. AdaBoost.M1 requires the individual classifiers to attain a weighted error of less than 0.5; otherwise, the algorithm aborts. Its bootstraps samples are drawn as uniform distribution in the beginning but then misclassified samples will have a higher probability of being drawn again in the next bootstraps. The individual classifiers' decisions are combined using weighted majority voting [41].

- A mixture of experts generates several experts (classifiers). The classifiers' decisions are combined using a linear rule. A gating network is used to specify the weights of the combination. The training dataset is required to bother the classifiers and the gating network. This ensemble algorithm is useful when heterogeneous feature subsets are to be used for a data fusion problem or when different classifiers are trained on different feature subsets [41].

2.3.3 Ensemble Combination Rules

There are many combination rules, and some of the ensemble algorithms have their built-in combination rules. For example, bagging uses majority voting, while AdaBoost uses weighted

majority voting. Selecting a combination rule depends on the class label type as well as the classifiers' output. Classifiers' output can be of three types:

- Abstract-level output that is a unique class label.
- A rank-level output that is ranked class labels.
- A measurement-level that is a vector of continuous-values measures standing for the probability of each class label [40].

In abstract-level outputs, the tth classifier's decision is defined as $d_{t,j} \in \{0,1\}$, $t=1,\cdots, T; j=1,\cdots, C$ where T is the number of classifiers and C is the number of classes. If the tth classifier chooses class ω_j, then $d_{t,j}=1$, and 0, otherwise. For those combination rules that require continuous outputs, the classifier outputs are defined as $d_{t,j} \in [0,1]$. Such outputs are usually normalized so that they add up to 1, which can be interpreted as the normalized support given to class j by classifier t, or even as the estimate of the posterior probability $P_t(\omega_j|\mathbf{x})$. Below is a description of the simplest and the lowest computational complexity combination rules:

Algebraic combination works for continuous value output only. It combines the classifiers' outputs using the algebraic expression, such as minimum, maximum, sum, mean, product, median, etc.

Majority voting is used with labels only. Its decision is correct if at least $\lfloor T/2+1 \rfloor$ of classifiers vote for the right class. It is computed as below:

$$d(x) = argmax_{J=1,..,c} \sum_{t=1}^{T} d_{t,j}\,(x)$$

Under the assumption that each individual classifier has a probability p of making a correct decision. Then, the ensemble's probability of making a correct decision has a binomial distribution. Specifically, the probability of choosing $k>\lfloor T/2+1 \rfloor$ correct classifiers out of T is:

$$P_{\text{ens}} = \sum_{k=\left(\frac{T}{2}\right)+1}^{T} \binom{T}{k} P^k (1-P)^{T-k}$$

Then,

$P_{\text{ens}} \to 1$,

as $T \to \infty$ if $p>0.5$

$P_{\text{ens}} \to 0$,

as $T \to \infty$ if $p<0.5$

It is important to note that the requirement of $p>0.5$ is necessary and sufficient for a two-class problem, whereas it is enough but not essential for multi-class problems. This combination method always leads to a performance improvement when having a sufficiently large number of individual classifiers.

In weighted majority voting, weights are assigned to each individual classifier vote. The ensembled decision is computed by:

$$d(x) = argmax_{J=1,..,C} \sum_{t=1}^{T} w_t d_{t,j}\,(x)$$

The optimal weights for the weighted majority voting rule can be shown to be $w_t \propto p_t/(1-p_t)$ if the T classifiers are class-conditionally independent with accuracies $p_1,\cdots, p_T$.

In soft voting, the class labels are predicted based on the predicted probabilities p for the classifier [42]. It is recommended for an ensemble of well-calibrated classifiers [43].

$$d(x) = argmax_{J=1,..,c} \sum_{t=1}^{T} w_t p_{t,j}(x)$$

where w_t is the weight that can be assigned to the tth classifier.

2.4 Feature Selection

Feature selection is a crucial step in the pre-processing data phase [44]. It is defined as selecting the minimal subset of features based on reasonable criteria to improve algorithm performance [45]. Also, it is the process of removing irrelevant, redundant, and noisy features [46]. Sometimes, it is referred to as the relevant feature identification process [47]. Feature selection provides an immediate impact on IDS performance in terms of computational complexity and accuracy [48]. It exclusively selects the most relevant and distinguishing features between traffic types. Thus, it reduces the number of features considered by an IDS and, in turn, reduces the complexity of the problem [46]. Moreover, by eliminating the noisy, redundant, and useless features, the accuracy of IDS increases, and the generated number of false alerts decreases [47].

Feature selection methods (FSM) are classified into three main types: wrapper, filter, and hybrid methods. The wrapper methods try to optimize some predefined criteria concerning the feature set as part of the selection process. The filter methods rely on the training data's general characteristics to select each other's independent features and are highly dependent on the class label. In their turn, the hybrid methods try to exploit the features of both wrapper and filter methods [49].

2.4.1 Feature Selection Process

The feature selection process is divided into three main steps: 1) subset generation, 2) subset evaluation, and 3) subset validation, as shown in Figure 2-9. In subset generation, the method generates/finds features. Two main properties need to be defined in this step:

- Search direction [49]: There can be no previous knowledge about what is the optimal subset of features in the search space. However, the search can be one of the following:
    - Sequential Forward Generation (SFG): It starts with an empty subset, then it adds features.
    - Sequential Backward Generation (SBG): It starts with the full set of features, then it removes features.
    - Bidirectional Generation (BG): it starts from both directions
    - Random Generation (RG): It starts the search in a random direction.
- Search strategy [47]:
    - Complete: It is a brute-force search where it tries all the possible subsets. Its space complexity is $O(2^N)$, where N is the total number of features.
    - Heuristic: It aims to avoid brute-force search by employing heuristics in conducting the search. Its space complexity is $O(N)$. It is clearly faster than the complete search. But, it may compromise on finding the optimal feature subset.
    - Nondeterministic: It searches for the next feature at random.

All feature selection methods can be grouped under two categories based on the two properties above:

- Feature ranking method: This is a simple method that returns a ranked list of ordered features based on an evaluation measurement. It does not tell what the minimum set of features that satisfies an evaluation criterion is. It only tells the importance of features. The run time complexity is $O(N.(n+N^2))$, where N is the number of features, and n is the number of instances. This type of method is considered fast, but it may not be straightforward to choose the number of features in the generated features subset M. It can merely choose the first M feature from the ranked list.
- Minimum subset method: It is used to generate the minimum feature subset. A stoping criterion determines when to stop the search.

In the subset evaluation step, the generated candidate subsets are evaluated based on a predefined evaluation criterion. Two types of evaluation criteria exist; independent and dependent. Independent criteria include distance, information, dependency, and consistency measures. The independent criteria are used with the filter model. On the other hand, dependent criteria depend on a predefined learning algorithm in deciding which feature subset to select based on the learning algorithms' performance. The former type is used in the wrapper model [49]. Another classification of evaluation functions categorizes them into five categories: score-based, entropy or mutual information-based, correlation-based, consistency-based, and detection accuracy-based [47].

The stopping criteria are significant to specify when the feature selection methods should stop. Stopping can be achieved when the research completes; some given bound is reached, such

as a specific number of iterations or features, subsequent deletion or addition of any feature without resulting in an improvement, or a sufficiently proper subset is selected [49].

Subset validation is generally executed by measuring candidate subsets' performance before and after adding/deleting a feature. The performance can be measured using different matrices such as error rate, accuracy, etc. [49]. Furthermore, the validation can take place by using simulation and real-world implementation [47].

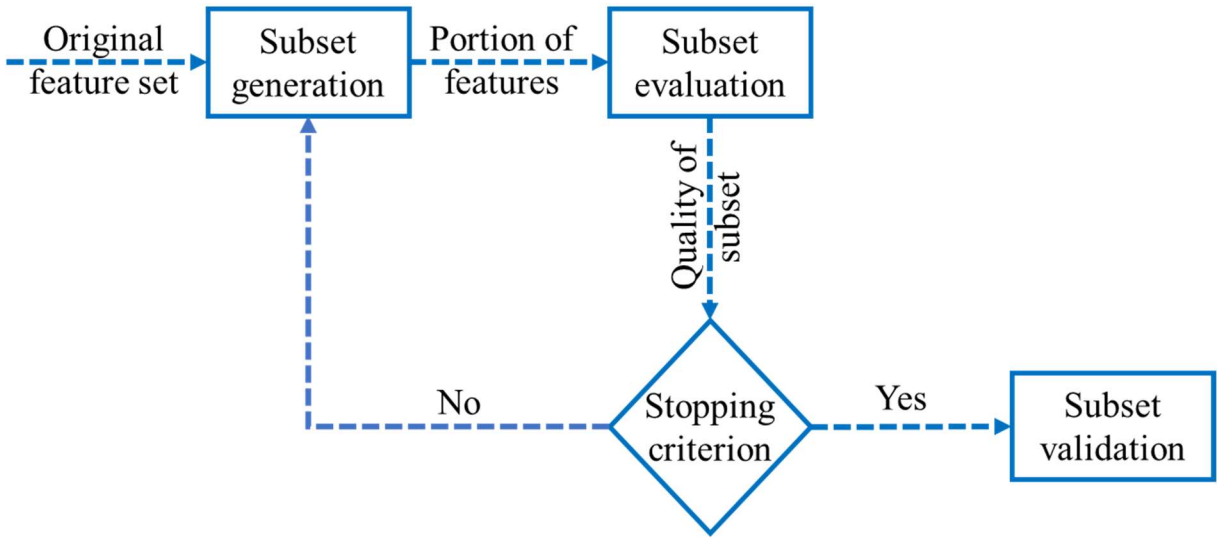


Figure 2-9: Feature selection process main steps.

This thesis considers developing a NIDS because it is practical for IoT networks that connect devices with limited resources. Furthermore, It ensembles a mixture of efficient and simple FSMs and models based on ensemble evaluation methods to capture the heterogeneity of IoT networks by using different methods. Also, to overcome the majority voting threshold, a rank voting technique is used to combine the FSMs' decisions of the ENFSM. Soft voting is used to combine the models' decisions because it is more informative than majority voting.

3 Chapter Three: Literature Review

We conducted a Systematic Literature Review (SLR) [50] method in writing this chapter. All the works considered in this chapter were considered because of having the keywords IoT, IDS, and EL all three together, which are the main three topics related to this research in each literature work. Some works are considered because they proposed EL feature selection in IDS, as well.

Few works in the literature used EL in IDS to improve detection and feature selection methods. This chapter will discuss the most recent IDS practices for IoT networks using ensemble methods for detection and feature selection methods. In section 3.1, a discussion of the literature using FSM with IDS and IoT is presented. Section 3.2 presents a discussion of the literature in terms of the used detection models.

3.1 Feature Selection Methods (FSMs)

Many works in the literature had applied ML methods for detecting botnet attacks. However, most of them did not use any kind of feature selection nor dimensionality reduction, such as the works [3], [7], [8], and [51]–[63].

3.1.1 Single Feature Selection

Table 3-1 FSMs classification in literature.

FSM Category	IDS	FSM	Selection Criteria	Evaluation Measure
Single	Moustafa [46]	Filter	Dependency	NA
	Illavarason [64]	Filter	Dependency, Information	Accuracy
	Alhakami [65]	Filter	Dependency	Accuracy, FPR
	Mukherjee [66]	Wrapper	Accuracy	Accuracy
	Pham [67]	Filter	Information	NA
Ensemble	TAMA [68]	Filter+Wrapper	Information, Accuracy	Accuracy
	Guerra [69]	Filter+Wrapper	Accuracy	F-score
	Nguyen [70]	Filters	Distance	Accuracy

On the other hand, as shown in Table 3-1, some works applied only one FSM, such as [46], which only used the Correlation Coefficient (CC), which measures the correlation between features, like a filter to reduce redundancy. CC is the simplest FSM to be used. It aims to eliminate the most correlated features. When a correlation value is close to +1 or -1, these features are highly correlated and, therefore, may be distracted from the final feature set. A zero-correlation value indicates no correlation. The least *N* correlated features were selected as the most significant features. In [64], the authors tested two FSMs based on correlation and Information Gain Ratio measures, in addition to a single dimensionality reduction method, the PCA. The three methods were evaluated in terms of accuracy using five classifiers Naïve Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Neural Networks (NN), and the k-nearest neighbors (Knn). In [65], the authors used the Bayesian approach proposed in [71]. The Bayesian method selects the relevant features. In effect, the relevancy was measured based on the feature distribution dependency on the class label, which is known as having uncommon density. Accuracy and FPR were used to measure the selected feature quality. [66] proposed the Feature-Vitality Based Reduction Method (FVBRD), which is a wrapper based on using NB classifier and backward search. The feature quality was measured based on NB classifier accuracy. The "Leave-one-out" method was used, which removes a feature at a time until the

accuracy of the classifier is below a predefined threshold. Generally, if NB classifier performance decreases, then the feature is important. If performance increases, then the feature is unimportant, and if no changes are found in performance, the feature is less important. The FVBRD was tested on the NSL-KDD [29] dataset only. Twenty-four features were selected at the end. This method is computationally expensive. Moreover, it considered one classifier to measure the features' quality, which selected features suitable for specific classifiers' type. [67] tested two FSMs on tree-based ensemble classifiers. The first FSM was FVBRD [66]. The second method was a filter based on the Gain Ratio (GR) technique [72]. Both methods were tested on the NSL-KDD dataset. The GR method selected 35 features. These two methods showed the same results because they nearly selected the same features. However, only one dataset was used in the experiments. Applying only one FSM for IDS raised doubts regarding the selected feature set's capacity to represent the optimal feature set.

3.1.2 Ensemble Feature Selection

Few other works have considered ENFSMs. In [68], the authors proposed a two-step FSM. The first step aimed to reduce the data redundancy by measuring the correlation in terms of entropy and information gain. The second step was a wrapper based on the Reduced Error Pruning Tree (REPT) classifier. In the first step, the proposed method considered an evolutionary approach to feature selection, using three different evolutionary techniques, which were particle swarm optimization (PSO), genetic algorithms (GA), and ant colony optimization (ACO). The authors evaluated the features' quality using the REPT [73] classifier's accuracy, which is a tree-based classifier. They considered testing a number of agents for each search algorithm. After that, a feature set was selected based on the highest classification accuracy produced by the REPT classifier indicated by a particular search method. The proposed method was tested on NSL-KDD and UNSW-NB15 datasets only. The datasets' dimensionalities were

reduced to 37 and 19 features for NSL-KDD and UNSW-NB15, respectively. The dimensionality reduction was not significant when considering the usage of complex search algorithms. Moreover, the authors ran several experiments to tune the particle's size, the number of ants, and the population size of PSO, ACO, and GS, which might not be practical for IoT. In [69], a two-step hybrid approach was proposed. The first step used Pearson and Fisher filters to select a feature subset. The second step used the output of the first step as input for a wrapper method. The wrapper step used Knn and Random Forest (RF) tree to select features based on the F-score heuristically. Additionally, the wrapper step considered the sequential Forward feature selection (FW) and sequential Backward feature elimination (BW) with both classifiers. Each filter was combined with one wrapper. The best performances were obtained using the Fisher filter and RF using FW. However, only the same classifiers, which were used in the wrappers, were used to evaluate the ENFSMs. In [70], a linear combination of multiple correlation filter measures was used. The authors used the correlation-feature-selection (CFS), and minimal-redundancy-maximal-relevance (mRMR). The authors combined the two measures by finding the average value to overcome the over-selecting phenomenon. The over-selecting of a feature set was measured using the Jaccard distance measure. Only C4.5 and BayesNet classifiers were used to measure the quality of the selected features based on the accuracy measure. However, the author used the KDD99 dataset only, which is an old dataset that does not represent the current network traffic and suffers from data unbalance, and redundancy issues.

3.1.3 Dimensionality Reduction

Some works in the literature used dimensionality reduction, such as [22], and [74]–[76]. Currently, dimensionality reduction still needs to extract all the original features during runtime, which means that the time reduction is not optimum. Furthermore, it adds computational

complexity to the IDS. In [74], the authors used the PCA to reduce the dataset dimensionality. However, using PCA requires the proposed IDS to extract all its features, followed by applying PCA to reduce the dimensions. Therefore, the PCA requires additional time. Furthermore, the proposed system was evaluated on KDD99 dataset, an old dataset that does not represent the IoT network traffic, and it has old attack types. In [22], the authors used Feature Mapper (FM) based on autoencoders to map *n* features onto a smaller feature space. In [75], the authors used Deep Feed Forward Neural Networks (DFFNN) to reduce the data dimensionality by generating embedded features. The embedded features were the last inner layer after removing the output layer from the DFFNN. While in [76], the authors proposed a dimensionality reduction method called features pairing, which is based on combining two FSMs. Feature pairing stands for generating a new feature represented as a string by concatenating the paired features' strings. The first pairing method was based on a wrapper that uses F-score, and the second method was based on filters, such as Information Gain Ratio (IGR). This method is computationally expensive because it suggests computing the F-score and IGR for all possible pairings.

As can be noted, few studies considered the feature selection matters in IDS, and a few FSMs were used and tested. Moreover, most of them were based on correlation measures. A multitude of studies in the literature gauged the feature sets' quality using one performance measure and a few classifiers. Most of the previous works considered the classifier's accuracy to measure a feature set quality. None of them considered the speed of the feature selection or size reduction. The feature set chosen by one classifier is generally suitable to work with the same classifier type exclusively. It is impractical to use the feature set with different classifiers. In fact, these studies did not provide any justification concerning the reason why they employed those methods and measurements and how they opted for them.

3.2 Detection Models

Various references in the literature used a single detection model, while a few used ensemble models. These works are summarized in Table 3-2 and described in the following subsections.

Table 3-2 Summary of the literature.

IDS	Detection method		Decision combination method	Dataset	Performance metrics		
	Model(s)	Ensemble			Accuracy (%)	F-Score	ROC-AUC
Moustafa [46]	(NB, DT, ANN)+ AbaBoost	Yes	Weighted majority voting	UNSW-NB1	99.2	NA	NA
				NIMS	98.3	NA	NA
Kitsune [22]	(AutoNN, small autoencoders)	Yes	RMSE	BotNetIoT	70	NA	0.7
N-BaIoT [63]	A single autoencoders	No	DL	BotNetIoT	NA	NA	NA
AL-Hawawreh [51]	autoencoder	Yes	MSE	NSL-KDD	98.6	NA	NA
				UNSW-NB15	92.4	NA	NA
Zhou [75]	gradient-boosted trees Knn DT logistic regression gaussianNB SVM	Yes No No No No No	Sigmoid function	NSL-KDD	98.54 98.82 98.77 98.85 98.8 98.86	NA	NA
				UNSW-NB15	91.22 91.9 92.29 90.35 92.52 92.32	NA	NA
ELNIDS [52]	Boosted Trees Bagged Trees Subspace Discriminant RUSBoosted Trees	Yes Yes Yes Yes	Majority voting	RPL-NIDDS17	94.4 93.3 78.6 94	NA	0.97 0.96 0.87 0.98
Pham [67]	DT J48 using bagging	Yes	NA	NSL-KDD	84.25	NA	NA
Jabbar [53]	(Knn+ ADTree)	Yes	weighted majority voting	Gure-KDD	99.93	NA	NA
Miller [54]	NB model+NB	Yes	NB	NSL-KDD	84.1	NA	NA
Putchala [77]	RNN	Yes	DL	KDD99	98.91	0.9959	NA
Lopez [61]	RNN+CNN	Yes	DL	RedIRIS	96	0.96	NA
Amin [62]	DEL	Yes	DL	Local DS	99.68	0.9848	NA
Abeshu [58]	autoencoders	Yes	DL	NSL-KDD	99.2	NA	NA
Feng [60]	DBN	Yes	DL	NSL-KDD	95.25	NA	NA
TAMA [68]	Rotation Forest+bagging	Yes	Majority voting	NSL-KDD	85.7	NA	NA
				UNSW-NB15	72.52	NA	NA
Aloqaily [78]	DBN+DT	Yes	None	NSL-KDD	99.43	NA	NA
TempoCode-IoT [79]	SVMs+Bagging	Yes	Majority voting	BotNetIoT	NA	0.99	NA
				CICIDS2017	NA	0.98	NA

3.2.1 Single Detection Model

Zhou [75] tested Knn, DT, LogisticRegression, GaussianNB, SVM on his proposed feature embedding method. The previous models achieved 98.82%, 98.77%, 98.85%, 98.8%, and 98.86% accuracy using the NSL-KDD dataset, respectively. Moreover, the models achieved 91.9%, 92.29%, 90.35%, 92.52%, and 92.32% accuracy using the UNSW-NB15 dataset.

3.2.2 Ensemble Detection Model

3.2.2.1 Deep learning

Putchala [77] used Deep Recurrent Neural Networks (DNN), which is a Deep Learning (DL) model. This model used a single hidden layer and single node with Gated Recurrent Unit (GRU) to reduce the complexity of the system. However, the author considered the KDD99 dataset, which is not an IoT dataset. The proposed model using all network layers achieved 0.9891 and 99.59% F and accuracy scores, respectively, using all the dataset features.

Lopez [61] proposed a system that combined RNN and Convolutional Neural Networks (CNN) that does not require any feature engineering. The proposed IDS used CNN to extract features from a real network traffic dataset called RedIRIS. Subsequently, the RNN was used as a detection model. The highest accuracy and F scores were 96% and 0.96. However, the used network flows from RedIRIS, which is not an IoT dataset.

Amin [62] proposed a two-phase system: OpCode sequence Graph Generation and Deep Eigenspace Learning (DEL) Phase. The IDS used the OpCode to transform the traffic into a graph. Accordingly, it applied DEL to learn the relations between vertices. The system achieved 0.9848 and 99.68% F and accuracy scores on a local dataset, respectively. Abeshu [58] used a stacked autoencoder with backpropagation and SGD. The proposed IDS achieved 99.2% accuracy on the NSL-KDD dataset. Feng [60] used the Deep Belief Network (DBN). The DBN model was composed of a two-layers restricting Boltzmann machine. Feng IDS achieved 95.25% accuracy using the NSL-KDD dataset.

Mirsky proposed Kitsune [22], Autoencoders Neural Networks (AutoNN). Kitsune used a small number of autoencoders for three reasons; to reduce noise, to mimic different patterns, and increase accuracy. The output autoencoder used the Root Mean Square Error (RMSE) as a non-linear voting mechanism for the ensemble. It achieved a 0.7 average ROC-AUC score using

BotNetIoT. Following that, Mirsky proposed N-BaIoT [63], which used autoencoder to learn the normal traffic features. One autoencoder was assigned for each IoT device to learn its normal traffic features. N-BaIoT is not a practical solution for IoT networks that has a considerable number of devices. N-BaIoT was evaluated by FPR and TPR using BotNetIoT and achieved 100% TPR and 0.7% FPR.

AL-Hawawreh [51] used deep autoencoder (DAE) and Deep Feed Forward Neural Network (DFFNN), which is an unsupervised learning method. AL-Hawawreh IDS used a symmetric architecture with a depth of five layers. Specifically, the outer layers consisted of nodes equal to the number of the extracted features. Thus, it used 41 nodes based on the used dataset. While the inner layers consisted of 10, 3, 10 nodes. As mentioned in the study [22], the authors used SGD for backpropagation and Mean Square Error (MSE) to classify the traffic. The authors did not, however, consider their proposed system complexity. Therefore, they compromised the speed for accuracy. The proposed system achieved 98.6% and 92.4% accuracy on the NSL-KDD and UNSW-NB15, considering all of their features. Zhou [75] tested a Gradient Boosting Tree (GBT) using his proposed feature embedding method. The GBT achieved 98.54% and 91.22% accuracy on the NSL-KDD and UNSW-NB15 datasets.

A major drawback in DL-based models is their high computation complexity that requires a long training time. Additionally, they require a huge amount of data to train, which may not be available when considering signature-based IDS, where malicious traffic is rare. They work as a black box that handles sophisticated calculations that is why it is hard to explain their results.

3.2.2.2 Experts mixture

Moustafa [46] constructed an ensemble model composed of NB, DT, and an Artificial Neural Network (ANN) through applying AdaBoost. The correntropy measure was the key factor in

choosing these models. The proposed model achieved an accuracy of 99.2% and 98.3% using UNSW-NB15 and NIMS datasets, respectively.

ELNIDS [52] aggregated four Boosted, Bagged, Subspace Discriminant, and RUSBoosted Trees using majority voting. ELNIDS had tested on the RPL-NIDDS17 [80] dataset. ELNIDS used all the dataset features. There were droughts regarding the level of diversity archived by the system, where all the examined models were tree-based. Furthermore, assembling complex ensemble models would have high computation overhead, which makes this system unpractical. Controversially, no results of the proposed IDS were shown. Pham [67] used various DTs, such as DT J48, REPTree, and RF using bagging and boosting. The highest accuracy was 84.25% achieved by J48 with bagging using 35 features of the NSL-KDD dataset. In another experiment, Jabbar [53] combined Knn with Alternating Decision Tree (ADTree). They were combined using the majority voting. The ensemble model achieved 99.93% accuracy using the Gure-KDD dataset, which contains connections of KDD99 but with payload to each connection. Thus, the used dataset is not considered an IoT dataset.

Miller [54] tested three decision combination methods. The most accurate method achieved an 84.1% accuracy. It used the NB model to combine the decisions of two NB models. A concern arises regarding the results because of using NB for classification and combining models' decisions has affected the results and thus acquired the highest accuracy. In another study, Tama [68] combined the Rotation Forest and bagging as a two-stage classifier. The Rotation Forest used PCA to create feature subsets. The proposed model achieved only 85.7% and 72.52% accuracy using the NSL-KDD and UNSW-NB15 datasets.

Within the same experimental context, Aloqaily [78] proposed a two-stage IDS using DBN and DT. The proposed IDS used the DBN to reduce the data dimensionality. In this way, it used

the DT for detection. The IDS achieved 99.43% accuracy using the NSL-KDD. However, using DBN caused an additional delay that increased as the number of the DBN nodes increased. TempoCode-IoT [79] used several SVM models with bagging. The number of SVM was defined as equal to the number of classes in the dataset. The authors only considered flow-based features to train the SVMs as well. The proposed IDS achieved 0.99 and 0.98 F scores using the BotNetIoT and CICIDS2017 datasets.

Upon examining all the aforementioned references in the relevant literature, it could be deduced that the authors did not provide any accounts to justify the reasons and the methods according to which the studied models were chosen. In fact, only a limited number of studies focused on speed detection, such as [75]. That is irrespective of the fact that a considerable number of experiments addressed complex models compromising speed for accuracy.

This thesis considers a vast range of FSMs and detection models. The selection of any method is based on the combination of several efficiency measurements' decisions to justify every selection. Moreover, this research aims to generate a feature set suitable for a vast range of detection models, not just one. Thus, it increases the ability to combine more variant types of models and meeting the heterogeneity of IoT networks.

4 Proposed Methods

In this chapter, the proposed methods were categorized into methods related to FSM and for detection model. Section 4.1 presented the proposed methods for FSM, while section 4.2 presented the detection model's proposed methods.

4.1 FSM

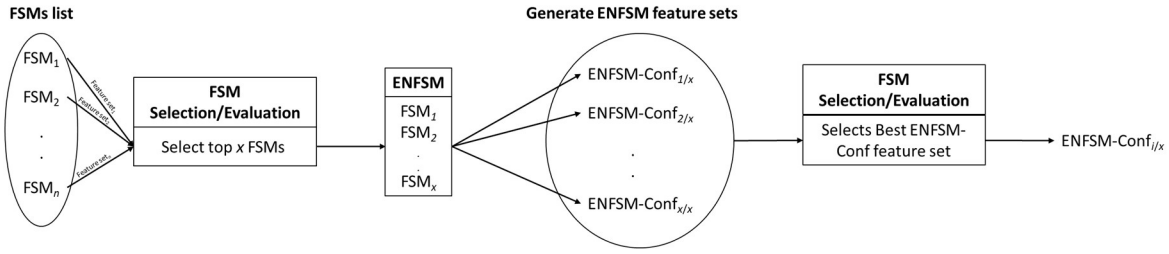


Figure 4-1 Architecture of proposed Ensemble FSM, where x is the total number of the used efficiency measurements.

This research proposes a novel way to select the best features correlated with the class label when considering a filter-based FSM. Furthermore, to increase the selected features' optimality, confidence about it, and suitability for working with different classifiers, an ENFSM was constructed. The proposed method combined five different FSMs. The FSMs were selected by considering different quality measures, including speed, size reduction, F-score, ROC-AUC, feature set entropy, and variance. The final decision was based on all measurements to select the most recommended FSMs. Moreover, the performance measures, such as F and ROC-AUC scores, were generated using a vast range of classifiers to ensure that the chosen feature set could suit a vast range of classifiers. Figure 4-1 provides a global picture of how the final ENFSM feature set is generated.

4.1.1 Cutoff Value For Filter-Based FSMs

The literature that applied filter-based feature selection to reduce data dimensionality has used two approaches to select the corresponding method's best features. These publications either

use a fixed predefined threshold as a cutoff or a fixed number of features to select, such as the top *n* features. Such choices do not meet the IoT network traffic dynamism, which is dynamic, rapidly changing, and heterogeneous. Moreover, it is crucial to have a fast as well as light FSM able to work with big data and make choices in a timely manner.

Every record is assigned an ID; in this research, this ID is considered as a feature, called ID feature. This ID feature is taken into account while evaluating the selection criteria score for each feature. A filter-based FSM selects all features with a score higher than that of the ID feature. Therefore, the FSM selects the features that are more meaningful than the ID feature. The proposed dynamic cutoff also accelerates the selection process and avoids the computationally expensive search.

4.1.2 Ensemble FSM Selection Method

Furthermore, to increase the selected features' optimality, confidence about it, and suitability for working with different classifiers, we proposed an ensemble procedure to select FSMs and construct an ENFSM. The proposed method combined the top five FSMs, which equals the top 25 percentile of the total considered FSMs, to build the ENFSM because the used evaluation method used five evaluation measures. Consequently, more diverse FSMs were selected. Various FSMs were considered and selected by combining the decision of different quality measures, including speed in terms of the elapsed time by an FSM, size reduction, F-score, ROC-AUC, feature set information gain, and variance. Moreover, the performance measures, such as F and ROC-AUC scores, were generated using a vast range of different classifiers to ensure that the chosen feature set suits a vast range of classifiers.

The key procedures for the proposed ENFSM selection method are presented in Algorithm 1. The proposed method ranks the FSMs scores in descending order. Thus, an FSM's score ranked 1 means that this FSM had the highest and best score. Therefore, this FSM is considered as that

measurement recommended FSM. After ranking all measurements' scores and finding each recommended FSM by each measurement, the total number of recommendations for each FSM is counted. In other words, the 1st ranks of each FSM over all the efficiency measures are counted. The top five FSMs with the highest rank summation or the highest number of recommendations are selected to be part of the ENFSM. Thus, the FSMs which are recommended by most of the efficiency measurements are selected. In case of a tie, which may lead to recommending more than five FSMs, an inner ranking between the tied FSMs takes place to select the best FSMs between them. The inner ranking stands for considering only the tied FSMs efficiency scores for re-ranking. If the tie still there, then the FSM with the least ranks summation is selected because it stands for the most recommended FSM by all the evaluation measurements. Thus, the finally selected FSMs have the approval of a majority of efficiency measurements by considering their preferences, apart from their first choice. A detailed explanation with an example is shown in section 5.3.2.2.

Algorithm 1: The algorithm for selecting the top five FSMs

Input: *FSMs*: Set of Feature Selection Methods, *MList:* List of detection models, *A:* training dataset
Output: *ENFSM* : A set of five FSMs

Procedure: FSMsSelection(*FSMs*, *MList, A*)
ENFSM← Ø
Compute efficiency measures scores for all FSMs
Repeat until *ENFSM* length equals 5 or *FSMs* is empty
    For each FSM in *ENFSM*, remove the FSM from *FSMs*
    For each efficiency measure, rank the remaining FSMs based on their scores in descending order
    For each FSM sum the 1st Ranks
    Select the FSMs which have the maximum 1st Rank-Summation among the remaining FSMs //count the total number of recommendations for each FSM
    If number of selected FSMs + *ENFSM length less than or equals 5*
        Add selected FSMs to *ENFSM*
    Else //There is a tie between FSMs 1st rank summation, thus we use the following procedure called inner-ranking to break the tie
        For each efficiency measure, rank the selected FSMs in step 7 based on their scores in descending order //this is the inner-ranking step
        For each selected FSM sum the 1st Ranks
        Select the FSMs which have the maximum 1st Rank-Summation among the selected FSMs
        If *number of selected FSMs in step 13* equals the number of the *previously selected FSMs in step 7* // The tie still not solved, thus select the most recommended FSM based on the FSM ranks sum
            For each selected FSM, sum the ranks values over all the efficiency measurements
            Select the FSMs with the least ranks values summation
            If *number of selected FSMs based ranks summation + ENFSM length* less than or equals 5
                Add the FSM(s) with the minimum ranks' summation to *ENFSM*
            Else
                Add the FSM with minimum feature set size among the selected FSMs in step 16
        Else // Inner ranking solved the tie
            FSMsSelection(*Selected FSMs in step 13*, *MList, A*) //Apply inner ranking
Return ENFSM

4.1.3 Evaluation Measures

This research considers five measures to evaluate a feature set quality. The five measures cover speed, reduction ratio, information gain, density, and accuracy. These measurements are reduction efficiency, feature set information gain, feature set variance, F-score ratio, and ROC-AUC ratio. These measurements provide an informative view of the effectiveness of the generated feature set. Moreover, they are integrated and accumulative measures to overcome the lead to contradictory conclusions. A detailed explanation of how these measures are computed is presented in the following sections. Also, Table 5-7 shows the FSMs scores using NSL-KDD dataset.

4.1.3.1 Reduction efficiency (RE)

The speed/execution time and the amount of data size reduction of an FSM are significant in the selection process when considering IoT networks and cybersecurity. It is essential to generate a

feature set promptly to update the IDS as soon as possible to stop and limit cyber threats harm. Studies in the literature did not address these crucial measures. Moreover, it is difficult to evaluate which FSM is faster with a low data size reduction or vice versa. Thus, having an integrated measure that combines time and reduction percentage overcomes this tradeoff matter [81]. This research proposes a new quality measure called the Reduction Efficiency (RE), which combines the data size reduction percentage, Reduction Ratio (RR), and the time (TM) spent by an FSM to reach that RR based on equation (4-1).

$$RE = \frac{RR}{TM} \tag{4-1}$$

In equation (4-1), RR is computed by dividing the reduced data size by the original data size. The larger RE value is, the better it is. Thus, as the TM value decreases, the RE increases. Thus, the proposed measure meets the requirements.

4.1.3.2 The feature set information gain

Information Gain (IG), also known as the uncertainty function, measures how much "information" a feature provides about the class. It measures the uncertainty or disorder, which is also the goal of machine learning models and data scientists, in general. In other words, the higher the IG, the easier it is to draw any conclusions from that feature. Accordingly, the higher the IG value is, the better. IG is used as an information measure to gauge the relevance of a feature [45].

A model's final decision depends on all the features used to build the model during the learning phase. Accordingly, the objective is to find the IG of the whole chosen feature set by an FSM, not for a single feature. Moreover, as the aim is to measure the FSM quality, not just a feature, accumulative measurements are required in order to describe the FSM output quality.

The PCA is deployed in this effort to convert the feature set into a single dimension and to find the IG of that dimension. This summarized value enables measuring the full feature set uncertainty and the corresponding FSM performance.

4.1.3.3 Feature set variance

The variance depends on the probability density function of feature distribution. If a feature is marked by a low variance or close to zero, then it is nearly constant and may not improve the model's performance. In that case, it should be removed. In this research, the variance will be used to measure the FSM quality via gauging the entire generated feature set variance. Similarly to IG, PCA is used to convert the feature set into one dimension. In this way, this method will lead to finding the variance of that dimension.

4.1.3.4 F-score ratio

The F-score is a combined accuracy measurement based on the harmonic mean. It represents a tradeoff between precision and recall. The F-score reaches its best value at 1 and the worst score at 0. It is calculated as shown in equation (4-2).

$$F - Score = 2 * \frac{Recall\ X\ Precision}{Recall + Precision} = \frac{2 * TP}{(2 * TP + FN + FP)} \tag{4-2}$$

This research tests each FSM over several learning models. It is of utmost importance to measure an FSM quality to generate a feature set suitable for a vast and diverse range of models. To this end, an F-score ratio is introduced as a quality measurement. To calculate the F-score ratio, the number of models that achieved an F-score higher than or equal to 0.95 were considered. Subsequently, the count is divided by the total number of the tested models. The F-score ratio (F-ratio) is therefore calculated, as shown in equation (4-3).

$$F - ratio = \frac{count\ (model\ F - score \geq 0.95)}{n} \quad (4\text{-}3)$$

Where *n* is the total number of the tested models.

4.1.3.5 ROC-AUC score ratio

ROC-AUC score, which is an accuracy measurement that represents the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) computed from prediction scores. By computing the area under the ROC curve, the curve information is summarized in one number. When the integral boundaries are reversed as large threshold *H* has a lower value on the x-axis, the area under the curve is given equation (4-4).

TPR(*H*): $H \rightarrow$y(x)

FPR(*H*): $H \rightarrow$x

$$A = \int_{x=0}^{1} TPR(FPR^{-1}(x))dx \quad (4\text{-}4)$$

Similar to F-ratio, a ROC-AUC-score ratio is postulated in this research (ROC-AUC-ratio) as a quality measurement. To calculate the ROC-AUC-ratio, the number of models that achieved a ROC-AUC-score higher than or equal to 0.95 is calculated. After that, the count is divided by the total number of the tested models. The ROC-AUC-ratio is calculated, as shown in equation (4-5).

$$ROC - AUC - ratio = \frac{count\ (model\ ROC - AUC - score \geq 0.95)}{n} \quad (4\text{-}5)$$

Where *n* is the total number of the tested models.

4.1.4 Decision Combination Method

The proposed ENFSM can generate five possible feature sets, where each FSM generates a feature set. The occurrence frequency of each feature over all the generated feature sets is found. The higher the frequency, the more confidence that feature is. Selecting a feature set by *n* FSMs generates a confidence level equals to *n*/5 for that feature set. For example, if all the features in a feature set are generated by all the FSMs, it will have a confidence level of 100%. The higher the confidence level, the more reliable the individual members of the feature set is. However, the higher the confidence level, the smaller the feature set size will be. As a whole, a feature set with a high confidence level may not be always more reliable than a feature set with a lower confidence level. Thus, a determination of the optimal confidence level is necessary. A detailed explanation is presented in sections 5.3.2.2.2 and 5.3.2.2.3.

4.2 Detection Model

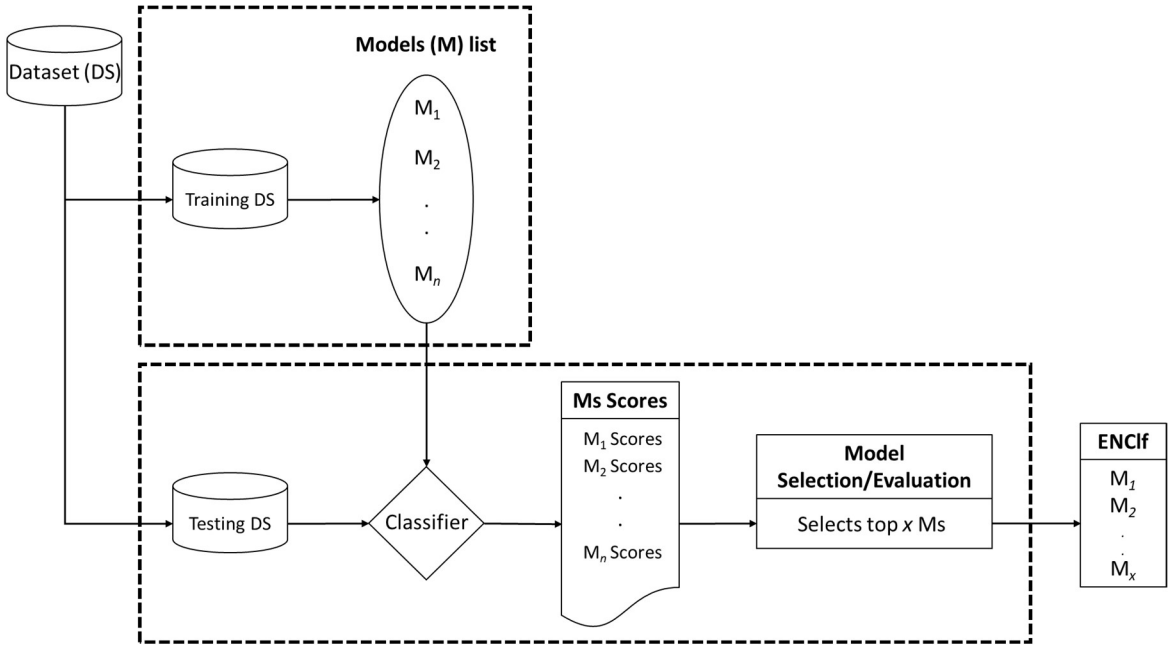


Figure 4-2 Architecture of proposed Ensemble classifier (ENClf), where x is the total number of the used efficiency measurements.

Based on the gaps detected in the literature, this research proposes a novel Model Selection Method (MSM) to construct an ensemble model. The MSM depends on three new integrated efficiency measures. The efficiency measures are based on the ratio between F-score, ROC-AUC, and variance with scoring time (*ST*), which is the time needed by a model to classify records. The *ST* is more significant than the Learning Time (*LT*) for an IDS, because the IDS performance depends on the model *ST* only during runtime. Figure 4-2 provides a big picture of how the final Ensemble Classifier (ENClf) is constructed.

4.2.1 Model Selection Method (MSM)

The proposed MSM combined the top 20 percentile of the considered models in this research. There were fifteen different models considered. Three models were chosen to build the ensemble model because three efficiency measures were used by the MSM. The selection step was based on three efficiency measurements to evaluate each model to increase the MSM confidence. The final decision was made by selecting the most recommended three models by all measures. In case of a tie, an inner ranking took place between the tied models. If the tie could not be resolved, then the model with the least ranks summation is selected. The least model's rank summation stands for the most recommended model by all measures. The key procedures of the proposed MSM algorithm are similar to Algorithm 1, except it uses the three models' efficiency measurements. Further details and explanation by example are available in section 5.4.1. The proposed MSM automatically selects models for the IoT network and updates the ensemble model as the traffic changes. Thus, it meets the IoT traffic heterogeneity and dynamicity.

4.2.2 Ensemble Models

The proposed ensemble model was composed of three different models. By definition, a different model is a model that has an entirely distinct type or kernel function. In this research, the three models' decisions were combined with soft voting. Soft voting can improve on hard

voting because it takes into account more information, where it uses each classifier's uncertainty in the final decision.

This research proposed two ensemble models. The first model was for working at a fog or an edge, called Edge-ENClf. This means that a different ensemble model was created for each network, and each dataset represented that. The second model was a cloud model, which would be the same for all the different networks for a centralized IDS, called Cloud-ENClf. The centralized model was composed of the models that were nominated for two or more datasets. The proposed model complexity is the composing models' maximum complexity when the models run in parallel.

4.2.3 Integrated Evaluation Measures

In cybersecurity, IDSs have to respond as fast as possible without sacrificing accuracy. Response time is significant as stopping the threat at early stages would limit the degree of losses. For this reason, time must be considered when evaluating any detection model along with model accuracy. However, how MSMs interpret this instruction and, consequently, which speed-accuracy tradeoff they choose might vary between methods. Both speed and accuracy are two significant aspects of performance. If they are analyzed separately, sometimes it might lead to contradictory conclusions about the effect of manipulation. To avoid such conflicts, several measures that integrate speed and accuracy have been introduced. An often-suggested solution is the inverse efficiency score (IES) [82], which is a single score that combines speed and accuracy using the equation ($\frac{\text{Mean of Responce Time (RT)}}{Proportion\ of\ Correct\ Responces\ (PC)}$). Furthermore, there are the rate-correct score (RCS) [83], which is found by ($\frac{\text{Number of Correct Responses (NC)}}{\text{Total RT}}$) and the linear-integrated speed-accuracy score (LISAS) [81] that equals ($\text{Mean of RT} + \frac{\text{Standard Deviations of RT}}{\text{Standard Deviations of Proportion Error}} . \text{Proportion Error}$). However, IES, RCS, and LISAS add unequal

weights on speed and accuracy, depending on the accuracy level. Concretely, they are very sensitive to speed-accuracy tradeoffs.

On the other hand, the balanced integration score (BIS) [84], which is found by (Standardized Mean PC − Standardized RT Mean), is devised to integrate speed and accuracy with equal weights. Thus, BIS is relatively insensitive to speed-accuracy tradeoffs. Finding IES and RCS is straightforward by divided the performance score by the time to find the efficiency score. Therefore, the efficiency measures for other performance measures as defined below:

F-Score efficiency: This is the ratio between the F score and ST defined in equation (4-6). F score is an integrated measurement that represents the weighted average of precision and recall. A higher efficiency value means a better model.

$$\mathrm{F-efficiency} = \frac{F-score}{ST} \tag{4-6}$$

ROC-AUC efficiency: This is the ratio between the ROC-AUC score and scoring time defined in equation (4-7). ROC-AUC measures the ability of a classifier to distinguish between classes. A higher efficiency value means a better model.

$$\mathrm{ROC-AUC-efficiency} = \frac{\mathrm{ROC-AUC}-score}{ST} \tag{4-7}$$

Explained variance efficiency: This is the ratio between the explained variance score and scoring time defined in equation (4-8). The explained variance measures the discrepancy between model predictions and actual class labels. A higher efficiency value means a better model.

$$\mathrm{Explained-variance-efficiency} = \frac{\mathrm{Explained\ variance}-score}{ST} \tag{4-8}$$

Section 5.4.1 shows a detailed explanation of how the three proposed efficiency measures are computed using the UNSW-NB15 dataset. In Table 5-10, the three efficiency measures for the used detection models are shown using the UNSW-NB15 dataset.

5 Experiments and Results

Experiments were conducted on Compute Canada Cedar cluster. All experiments used Broadwell type nodes equipped with Intel(R) Xeon(R) CPU E5-2683, v4, 2.10GHz, and 125G RAM. This research used the SciKit-learn [85] package's FSMs and models implementations using Python 3.7.4. This research considered the detection problem as a binary classification problem to develop an IDS for general-purpose detection. All the results presented used the mean of the five folds of the cross-validation. 5-folds cross-validation is chosen by dividing the datasets into 80%-20% ratios for training and testing datasets, respectively. The ShuffleSplit function is used with a random seed to generate independent and different testing and training feature sets in each cross-validation iteration.

5.1 Datasets

Datasets play a crucial role in training and testing IDS. The IDS datasets include network traffic records of normal and malicious traffics. Each dataset contains a set of features generated from the network traffic.

The network data is considered as big data, where it meets the 5V big data features, being volume, velocity, variety, veracity, and value. Volume refers to the massive amount of data generated every second. Velocity refers to the speed at which new data is generated and the speed at which data moves around, considering the rapidly growing number of connected devices and the high network communication speed. Variety stands for the different types of data we can now use, where different protocols use different data formats. Veracity means the messiness or trustworthiness of the data where some packets may get lost, damaged, or repeated due to network failures. Value refers to the ability to turn the IoT data into profit, where

businesses apply AI and data analysis on the collected data to make critical decisions or to monitor vital and significant facilities [86].

In this research, four datasets were used to train and test the ML models. The NSL-KDD and UNSW-NB15 datasets contain general network traffic, while BoTNetIoT and BoTIoT contain IoT network traffic. NSL-KDD and UNSW-NB15 datasets represent a benchmark for intrusion detection. Also, they enable the current research to be compared with previous work. More details about each dataset are presented in the following sub-sections, Table 5-1, and Table 5-2.

5.1.1 NSL-KDD

NSL-KDD [29] was created in 2009 as a corrected version of DARPA KDD99 dataset, which was provided by the Massachusetts Institute of Technology (MIT) Lincoln Laboratory in 1999 [28]. This dataset can be applied as an effective benchmark dataset to help in comparing the performance of different intrusion detection methods.

The dataset solved some of the KDD99 dataset issues. First, it removed the redundant records, which helped in overcoming the issues of training biased detection models due to the existence of repeated records. Secondly, it helped in comparing the results of different research work as they became consistent and comparable, where due to the reduction of records number into a reasonable size, there was no need to select a random training and testing sets anymore [29]. This dataset can be used for IoT IDS, where the advances and expansions of the new IoT networks include a wide variety of devices, in addition to smart devices that act as computers next to the computers themselves. Nevertheless, the same attacks still are valid to occur. Although some researchers criticize its reliability for evaluating IoT IDS [15], many researchers use it as an IoT dataset. Thus, it is considered here to compare the proposed IDS with IDSs in the literature.

Figure 5-1 displays the ratio of normal traffic vs. malicious traffic in the NSL-KDD dataset, while Figure 5-2 represents the distribution of malicious traffic types. The Others category includes the attacks Pod, Land, Nmap, guess_passward, Phf, Imap, Multihop, ftp_write, Spy, warezmaster, Perl, Rootkit, buffer_overflow, and Loadmodule where their occurrences in the dataset are a few; thus, they were summed up.

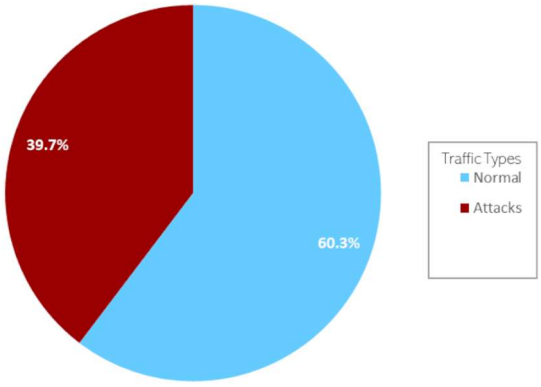


Figure 5-1 NSL-KDD normal vs attack traffics.

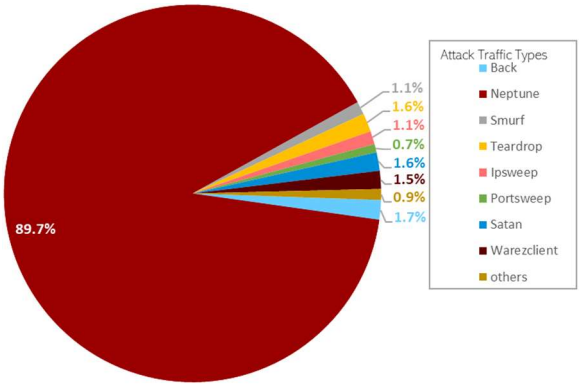


Figure 5-2 NSL-KDD attacks traffic types.

5.1.2 UNSW-NB15

UNSW-NB15 was created in 2015 to overcome the issues of KDD99 and NSL-KDD [30]. It includes newer attacks in addition to generating statistical features to increase the model accuracy. For the same reasons mentioned in section 5.1.1, this dataset is considered in this research as an IoT IDS dataset.

Figure 5-3 displays the fraction of normal traffic vs. malicious traffic. Figure 5-4 represents the fraction of each malicious traffic compared to the total number of malicious traffic in the dataset.

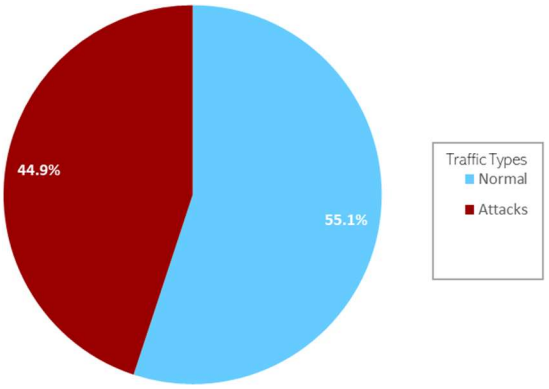


Figure 5-3 UNSW-NB15 normal vs attack traffics.

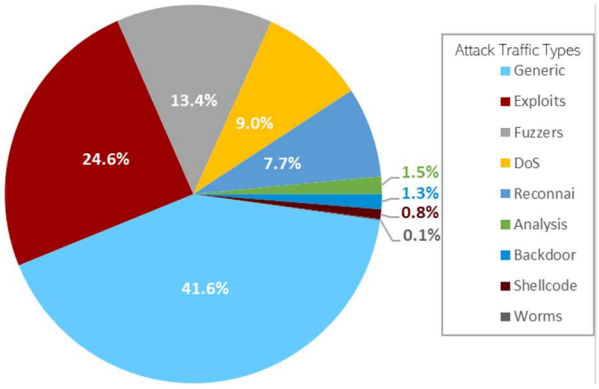


Figure 5-4 UNSW-NB15 attacks traffic types.

5.1.3 BotNetIoT

This is a recent dataset published in 2018 that collected real traffic data gathered from nine commercial IoT devices authentically infected by Mirai and Gafgyt. This dataset addresses the lack of public botnet datasets for the IoT networks.

Figure 5-5 shows the fraction of normal traffic vs. malicious traffic. Figure 5-6 provides the fraction of each malicious traffic compared to the total number of malicious traffic in the dataset.

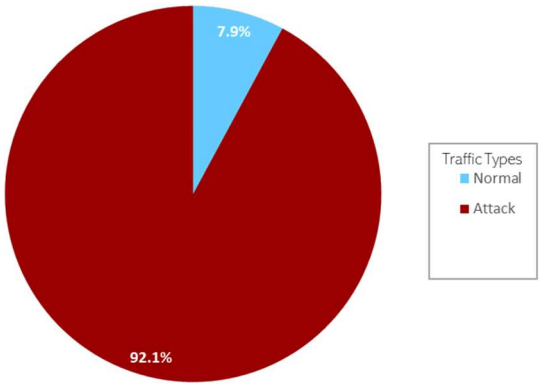


Figure 5-5 BotNetIoT normal vs attack traffic.

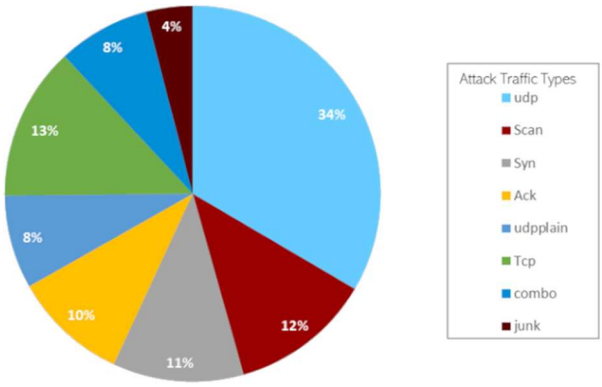


Figure 5-6 BotNetIoT attacks traffic types.

5.1.4 BoTIoT

The BoTIoT [87] dataset was published in 2018. A realistic IoT network environment generated it. It has a recent denial of service and information theft botnet attacks traffic.

Figure 5-7 shows the fraction of normal traffic vs. malicious traffic. There were 477 normal sessions only. Figure 5-8 demonstrates the fraction of each malicious traffic compared to the

total number of malicious traffic in the dataset. There were three types of denial of service attacks TCP, UDP, and HTTP.

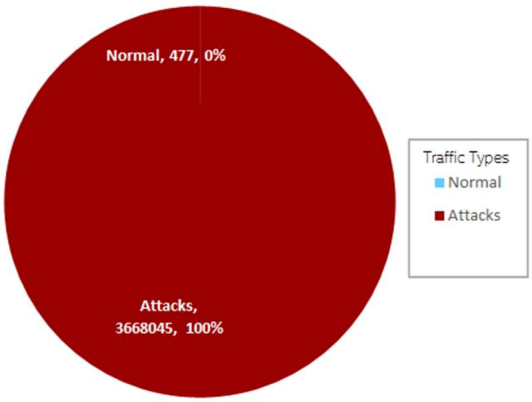


Figure 5-7 BoTIoT normal vs attack traffic.

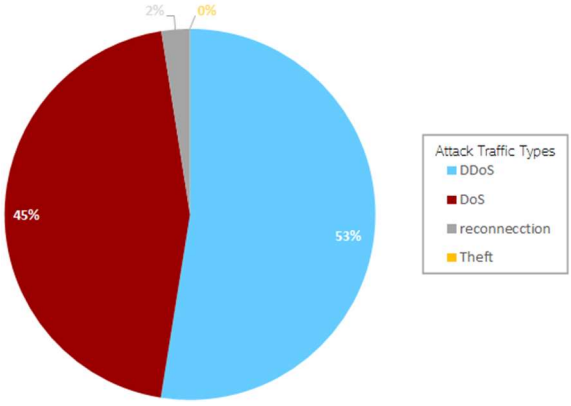


Figure 5-8 BoTIoT attacks traffic types.

5.1.5 Datasets Features Quality

This section aims to study the quality of the features and the impact of feature engineering on IDS performance. Table 5-1 shows the traffic type counts in each dataset, in addition to attacks' classes and sub-types. Furthermore, it represents the number of records for each attack in each dataset. It can be deduced that all the datasets' traffics are not balanced where the number of normal records does not possess equal attack records. The NSL-KDD [29], a corrected version of the original KDD99 dataset [88], was published in 2009 and one of the commonly used datasets in evaluating IDSs. The new version has a label and 41 features. The features are categorized into four main categories: basic, content-based, time-based, and host-based, as indicated in Table 5-2. UNSW-NB15 has nine recent attacks' samples, but the dataset does not provide details about each attack class type. Furthermore, it has 43 features; two label features, fourteen basic features (basic and flow features), and the rest are engineered features related to the content, time, connection as well as a general-purpose, as depicted in Table 5-2. Both NSL-

KDD and UNSW-NB15 have regular network traffic. The BoTNetIoT [31] and BoTIoT [87] are the most recent datasets. BoTNetIoT contains the traffic of nine IoT devices sniffed using Wireshark in a local network using a central switch. It has only time-related features. It includes two Botnet attacks (Mirai and Gafgyt). Five different sub-attack types for each attack are in the dataset, as presented in Table 5-1. The attacks are artificially generated by fetching them in the binaries of Gafgyt from the IoTPOT dataset [89]. The IP address of the C&C server's IP address was extracted from the malware's binaries, and all the traffic corresponding to this IP was directed to the local lab C&C server. While for the Mirai attack, the traffic was generated by using its source code from [22]. The BotNetIoT dataset contains 115 statistically engineered features extracted from the pcap files. These features are based on the seven statistical measures: mean, variance, count, magnitude, radius, covariance, and correlation coefficient. These measures are computed over five different time-windows: 100 ms, 500 ms, 1.5 sec, 10 sec, and 1 minute with decay factors 5, 3, 1, 0.1, and 0.01, respectively. The decay factor indicates the amount of information loss over a time-window. The decay factor value is used in the dataset as well as in this research to refer to its corresponding time-window as L5, L3,.. etc. Using time-windows makes this dataset suitable for stateful IDS/models. Four features were extracted from the pcap: packet count, jitter, size of outbound packets only, and outbound as well as inbound packets together. For each of these four features, one or more statistical measures were computed, resulting in 23 features. These 23 features were computed over the five different time-windows to get the 115 features in this dataset. The author claims that extracting and computing these features is fast and incremental over the time-windows. At last, the BoTIoT [87] is an IoT dataset published by the same research lab of UNSW-NB15 dataset. The BoTIoT dataset has two types of attacked denial of service and information theft. The denial of service category included DDoS and DoS attacks for TPC, UDP, and HTTP. The information theft

included reconnection and theft attacks, as shown in Table 5-1. The BoTIoT dataset had one label and 42 features. Table 5-2 shows there are four types of features: basic, flow, statistical-flow, and time. The majority of features were statistical-flow features.

Table 5-1 Datasets traffic's types and counts.

Dataset															
BotNetIoT				**UNSW-NB15**				**NSL-KDD**				**BoTIoT**			
Traffic	Class	Type	# of Rec.	Traffic	Class	Type	# of Rec.	Traffic	Class	Type	# of Rec.	Traffic	Class	Type	# of Rec.
Normal	---	---	555932	Normal	---	---	37000	Normal	---	---	87832	Normal			477
Attacks	Mirai & Gafgyt	udp	2176365	Attacks	Generic	N/A	18871	Attacks	DoS	Back	968	Attacks	Denial of service	DDoS	1926624
										Land	19			DoS	1650260
										Neptune	51820		Information Theft	reconnection	91082
										Pod	206			Theft	79
										Smurf	641				
										Teardrop	918				
	Mirai	Scan	793090		Exploits	N/A	11132		Probe	Ipsweep	651				
		Syn	733299		Fuzzers	N/A	6062			Nmap	158				
		Ack	643821		DoS	N/A	4089			Portsweep	416				
		udpplain	523304		Reconnai	N/A	3496			Satan	906				
	Gafgyt	Tcp	859850		Analysis	N/A	677		U2L	guess_passwd	53				
					Backdoor	N/A	583			Phf	4				
		combo	515156		Shellcode	N/A	378			Imap	12				
		junk	261789		Worms	N/A	44			Multihop	7				
										ftp_write	8				
										Spy	2				
										Warezclient	893				
										Warezmaster	20				
									U2R	Perl	3				
										Rootkit	10				
										buffer_overflow	30				
										Loadmodule	9				

Table 5-2 Datasets (DS) features.

DS	Features		DS	Features		DS	Features		DS	Features	
	Type	*Names*		*Type*	*Names*		*Type*	*Names*		*Type*	*Names*
BotNetIoT	Temporal-Statistical/ Time-Related	MI_dir_L#_weight, MI_dir_L#_mean, MI_dir_L#_variance H_L#_weight, H_L#_mean, H_L#_variance, HH_L#_weight, HH_L#_mean, HH_L#_std, HH_L#_magnitude, HH_L#_radius, HH_L#_covariance, HH_L#_pcc, HH_jit_L#_weight, HH_jit_L#_mean, HH_jit_L#_variance HpHp_L#_weight, HpHp_L#_mean, HpHp_L#_std, HpHp_L#_magnitude, HpHp_L#_radius, HpHp_L#_covariance, HpHp_L#_pccAll	**UNSW-NB15**	Flow-Related	proto	**NSL-KDD**	Basic	Duration, protocol_type, Service, Flag, src_bytes, dst_bytes, Land, wrong_fragment, Urgent	**BoTIoT**	basic	Flgs,flgs_num,ber,Stat e, state_nu,mberDur, Spkts, Dpkts, Sbytes, Dbytes
				Basic	dur, state, service, sttl, dttl, sload, dload, sloss, dloss, sbytes, dbytes, spkts, dpkts		Content-Related	Hot, num_failed_logins, logged_in, num_compromised, root_shell, su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, is_host_login, is_guest_login		flow	pkSeqID, Proto, proto_numberSaddr, Sport, Daddr, Dport, Pkts, Bytes, Seq, Rate
				Content-Related	swin, dwin, stcpb, dtcpb, smean, dmean, trans_depth, response_body_len		Time-Related	Count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate		statistical-flow	Mean,Stddev,Sum,Mi n,Max,Srate,Drate,Tn BPSrcIP, TnBPDstIP, TnP_PSrcIP, TnP_PDstIP, TnP_PerProto, TnP_Per_Dport, AR_P_Proto_P_SrcIP, AR_P_Proto_P_DstIP, N_IN_Conn_P_SrcIP, N_IN_Conn_P_DstIP, AR_P_Proto_P_Sport, AR_P_Proto_P_Dport, Pkts_P_State_P_Proto col_P_DestIP, Pkts_P_State_P_Proto col_P_SrcIP
				Time-Related	sjit, djit, sinpkt, dinpkt, tcprtt, synack, ackdat		Host-Based-Traffic	dst_host_count, dst_host_srv_count, dst_host_same_srv_ra te, dst_host_diff_srv_rat e, dst_host_same_src_p ort_rate, dst_host_srv_diff_hos t_rate, dst_host_serror_rate, dst_host_srv_serror_r ate, dst_host_rerror_rate, dst_host_srv_rerror_r ate		time	Stime, Ltime
				Generated-General Purpose	is_sm_ips_ports, ct_state_ttl, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd						
				Generated-Connection	ct_srv_src, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst						

Figure 5-9 (a), (b), (c), and (d) displays the traffic plot of the datasets BotNetIoT [31], UNSW-NB15 [30], NSL-KDD [29], and BoTIoT [87], respectively, in addition to the Pearson correlation score of each feature with the class label at the top of each plot. The scores are also

visualized using colors as well to highlight the type of correlation as either positive or negative. The X-axis displays the features in each dataset, and the Y-axis displays the Min-Max normalized values of the features. The red lines represent the attack traffic, while the blue lines represent normal traffic. In Figure 5-9.a, which plots the traffic for all features in the BotNetIoT dataset, it is noticed that some features clearly show a high distinction between normal and attack traffics and its features that provide the best distinguishing pattern among the considered datasets. Furthermore, it is pertinent that the same features have the same pattern over different time-windows, which demonstrates redundancy and necessitates that these features be omitted. In section 5.2, the optimum time-window, which will lead to reducing the redundancy and the dataset dimensionality, is found. Furthermore, interestingly variance is the worst statistical measure correlated with the class label feature over all features. In contrast, the best two statistical measures that are highly correlated either positively and negatively with the class label are the mean and weight, respectively, for all features. Dropping the least correlated features with the class label and considering the highly correlated ones only enables more potentials in dimensionality reduction. In Figure 5-9.b, UNSW-NB15 has some useless features if considered alone, where their values for each traffic type are highly interleaved, such as dur, sload, sjit, stcpd, and dtcpd. Furthermore, there are some features that have the same values for both traffic types such as dttl, smean, dmean, trancs_depth, ct_state_ttl, and others, such as dpkts and dpytes, have the same pattern that is a redundancy. The basic features show a better distinguishing pattern than engineered features, such as state, spkts, dpkts, sbytes, and dbytes. In general, these dataset features suffer from low correlation with the class label, making it challenging to develop an accurate IDS relay on these features without applying any feature engineering techniques. According to the NSL-KDD dataset, shown in Figure 5-9.c, two features, num_outbound_cmds as well as is_hot_login, were removed from the plot because they do not have any values in the

dataset for all packets. Its features show the least distinguishing pattern between normal and attack traffics. The normal and attack packet values are highly interleaved, which makes the detection task more difficult, as shown in SCAN IDS [15]. Interestingly, its features show a better correlation with the class label than UNSW-NB15 dataset's features, even they have a lower level of engineering. Furthermore, the basic features count and srv_count have the same pattern, and thus, they are considered redundant.

Figure 5-9 (d) demonstrates the traffic plot of the BoTIoT dataset. Interestingly, daddr, apkts, dpkts, sbytes, TnBPDstIP, TnBPSrcIP, TnP_PSrcIP, TnP_PDstIP, TnP_PerProto, TnP_Per_DPort, Pkts_P_State_P-Protocol_P-DestIP and Pkts_P_State_P-Protocol_P-SrcIP show the same plot style, where malicious traffic had low values and while normal traffic had higher values. These features may be convenient for candidates to increase the detection models' performance. On the other hand, they are redundant.

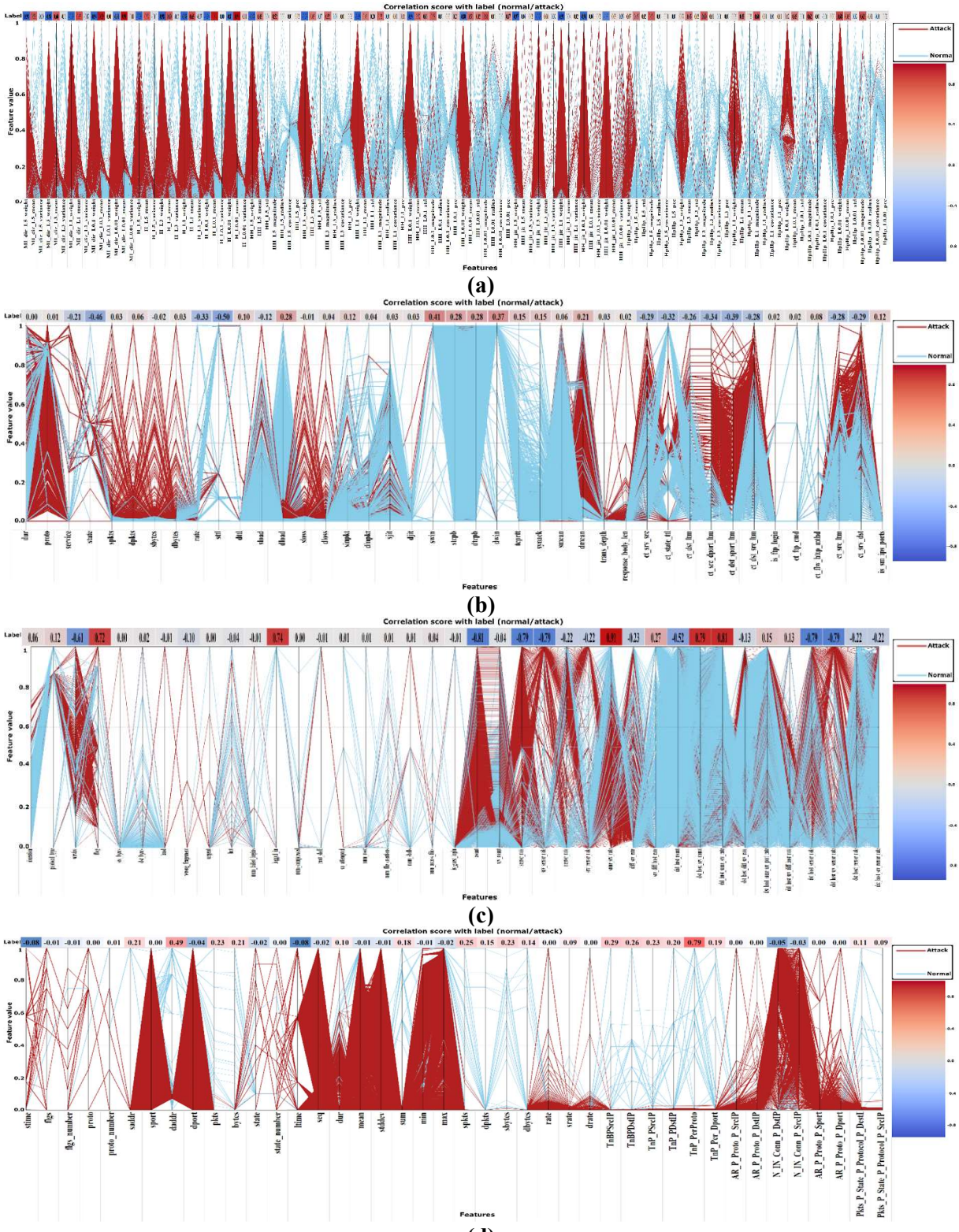


Figure 5-9 Dataset traffic plots, where X-Axis represents features, and Y-Axis represents MiniMax normalized value between [0,1]. a) BotNetIoT dataset, b) UNSW-NB15 dataset, c) NSL-KDD dataset, and d) BoTIoT datasets.

Figure 5-10 depicts the top-10 features highly correlated with the class label using the Pearson correlation score for each dataset as well as the correlation scores between these features. It was found that as the feature correlation with the class label increases, the correlation of the features with each other increases as well. In Figure 5-10.a, the top-10 correlated features with class label are those based on the weight measure, which stands for the number of items observed in recent history. Four features are in the top-10 are MI_dir, H, HH, and HH_jit over the three time-windows L5, L3, and L1. MI_dir, H, HH, and HH_jit are the stats summarizing the recent traffic from the source MAC-IP, the recent traffic from this packet's host (IP), the recent traffic going from this packet's host (IP) to the packet's destination host, and the jitter of the traffic going from this packet's host (IP) to the packet's destination host, respectively. The low diversity of features' types time-windows for a single statistical measure is why the high correlation scores between the features and the similar correlation scores with the class label. This also holds for the other statistical measures' features. Such a fact means that the statistical measures highly overwrite the effect of different features' types and different time-windows. As a result, a high redundancy appears in this dataset. Thus, a feature reduction/selection is necessary. Notably, the weight statistical measure's features show a high correlation with the class label. Thus, they represent potential feature selection/dimensionality reduction without trading the detection method's accuracy.

Figure 5-10.b represents the top-10 features in the UNSW-NB15 dataset that are highly correlated with the class label. The highest correlation score is -0.5 for sttl feature, and all other features' correlation scores are below this value. Furthermore, it is noticed that the basic features, which are state and sttl, have a better correlation with the class label than the engineered statistical features. The low correlation scores reveal the poor quality of this dataset's features.

Thus, applying feature selection to select the most correlated features to the class label becomes mandatory.

Figure 5-10.c presents the top-10 features in NSL-KDD dataset that are highly correlated with the class label. Notably, most of the dataset features have deficient correlation scores with the class label that reaches zero. Therefore, these low correlated features may be excluded from training any detection model. Furthermore, the class label's highest correlated features show high correlation scores between each other, which means a high redundancy in this dataset that requires applying features selection to remove this redundancy. In contrast with UNSW-NB15 dataset's engineered features, NSL-KDD dataset's engineered features showed a higher correlation with the class label than the basic features.

Figure 5-10 (d) demonstrates the top-10 highly correlated features with the class label in the BoTIoT dataset. The highest correlated feature is TnP_PerProto. Interestingly, saddr and daddr had nearly zero correlation with other features, except between themselves. In this way, they were suitable candidates to be selected.

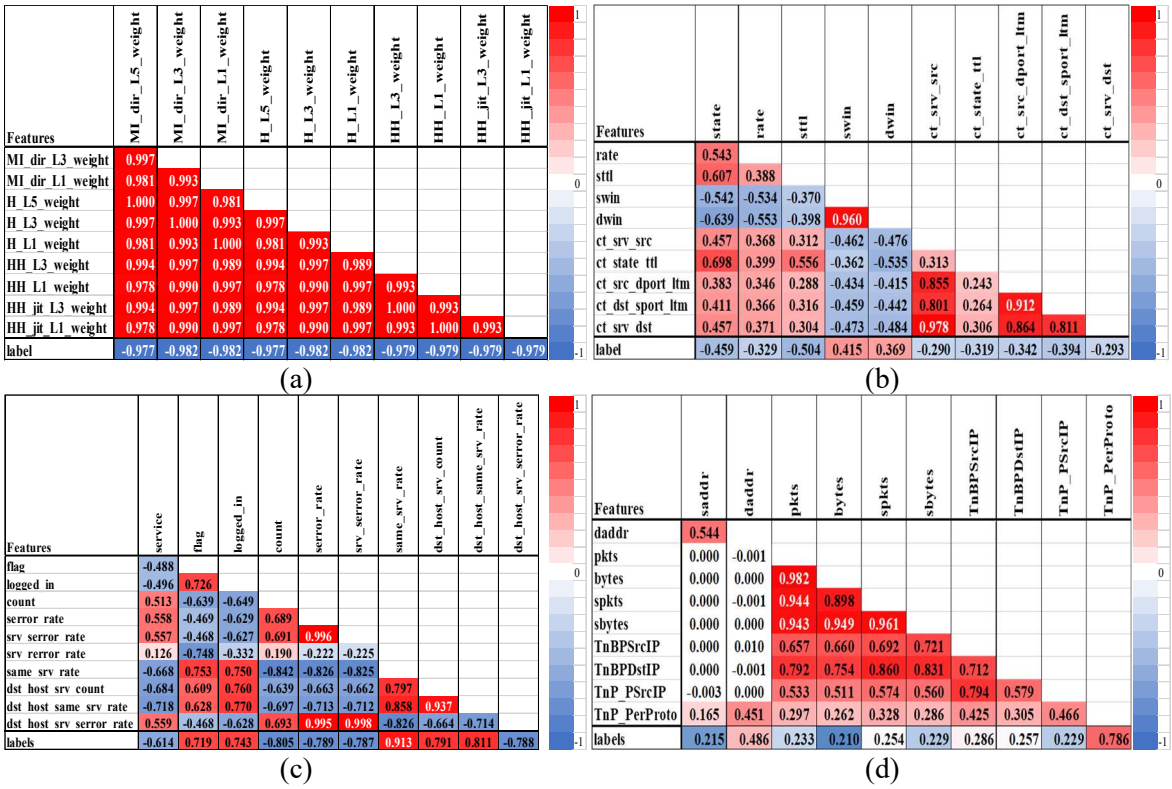


Figure 5-10 Datasets Top-10 highly correlated features with label/class feature heatmap. a) BotNetIoT dataset, b) UNSW-NB15 dataset, c) NSL-KDD dataset, and d) BoTIoT dataset.

5.2 Determining Optimum Time-Window For The BotNetIoT Dataset and Model Heterogeneity Resistance

A time-window refers to the duration of time an IDS collects the network packets for each session before it extracts the features. All the considered datasets extracted their features by considering a single time-window, except the BotNetIoT dataset, which considered 5 time-windows. This section aims to select the best time-window based on detection models' accuracy. Thus, three performance measures are considered, being Accuracy, Precision, and Recall. They provide an informative view of the effectiveness of the model. These metrics are calculated as below:

Accuracy: It stands for the ratio of the number of correctly classified samples to the total number of samples, see equation (5-1).

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{5-1}$$

Precision: It indicates the ratio of correct detections to the model's total actual number of detections, see equation (5-2).

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{5-2}$$

Recall: It is also known as sensitivity or True Positive Rate (TPR), and is the ratio of the correct detections to the total number of actual detections in the dataset, see equation (5-3).

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{5-3}$$

All performance experiments in this section were run on the BotNetIoT dataset [31]. The experiments are divided into two sets: The first set of experiments aims to evaluate the performance of the considered models using all the features in the dataset. The second set intends to assess each model's performance over different traffic heterogeneity levels as we have different time-window sizes to find the optimum time-window size for each model.

All redundant records were removed from the dataset to avoid model overfitting. In addition, the dataset features were normalized before training the model using Z-Score to prevent specific feature domination over other features. All the presented results represent the mean of ten folds of the cross-validation.

Experiments considered five different classifiers in terms of their way of analyzing the problem are: PCA Anomaly Detection, Local Deep SVM (LDSVM), SVM (Pegasos-Linear), Logistic Regression, and Boosted Decision Tree. Since IoT networks have high heterogeneity

of devices that will use a diversity of protocols and standards [79], considering different classifiers will enrich literature to better understand how their classifiers work and knowing which classifier analysis style best works in IoT networks.

The PCA combines the values of the features to generate the principal components, where the features in the used dataset are highly correlated. The Boosted Tree prediction is based on the entire ensembled trees, where a hundred trees were used. The SVM represent the records as points in a hyperplane space, which is the feature space. It then maps them to the traffic types where traffic types are separated with a wide and a clear gap as much as possible. Regarding the network traffic datasets, this gap will not be wide due to the nature of the features and the high level of interleaved values within the same feature. The Local Deep SVM (LDSVM) overcomes this problem by learning decision boundaries that are locally linear where a testing record is efficiently classified by testing it against its local decision boundary rather than testing against the entire set of decision boundaries across the feature space. Lastly, the Logistic Regression uses a function to understand the relation between the traffic type and the features.

5.2.1 Results

Figure 5-11 to Figure 5-19 present the three performance measures considering each time-window's corresponding features in the dataset. This set of experiments strives to discern the effects of time-window size on considered models and define the optimum one as well as to measure each model's resistance to data heterogeneity. Considering all traffic types, each performance measure was found, Mirai's traffic only and Gafgyt's traffic only.

In Figure 5-11, the Boosted Tree demonstrates the best accuracy over all the time-windows and ranges from 99.94% at L0.01 to 100% at L0.1 and L1. The lowest accuracy results over all time-windows were the PCA. In fact, all methods', except the PCA, accuracy increases as the time-window increases until L0.1, when the accuracy decreases. A decrease in the Boosted Tree

accuracy starts at L3, and its highest accuracy is at L0.01. On the other hand, the PCA accuracy increases as the time-window decreases, which refers to its performance on detecting Mirai in Figure 5-12 and Gafgyt in Figure 5-13 attacks that affected its total performance. Both Local Deep SVM and Logistic Regression models show nearly the same performance with a small difference for the Logistic Regression over the Local Deep SVM.

The Boosted Tree has the best accuracy and PCA is the worst, as shown in Figure 5-12. The PCA's accuracy considering the Mirai traffic alone improved compared to its accuracy over all traffic types. However, it is still the lowest among all other algorithms over all time-windows. Furthermore, the Logistic Regression model overcomes the Local Deep SVM's accuracy with a very small difference over all time-windows. All models' accuracy starts decreasing as the time-window equals L1, except for Logistic Regression, which diminishes when its time-window equals L3. Figure 5-13 displays that all models have the same performance behavior, as indicated in Figure 5-12, except for the PCA model, which drops at L3 and increases at L5.

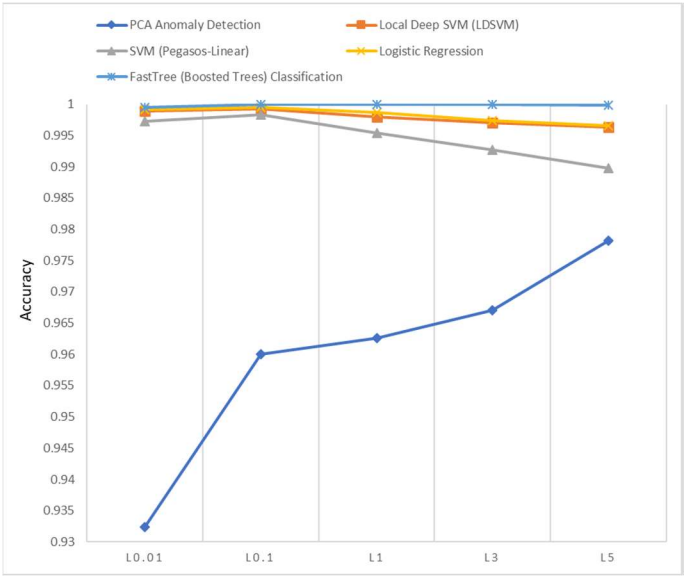


Figure 5-11 Accuracy over considering all traffic types.

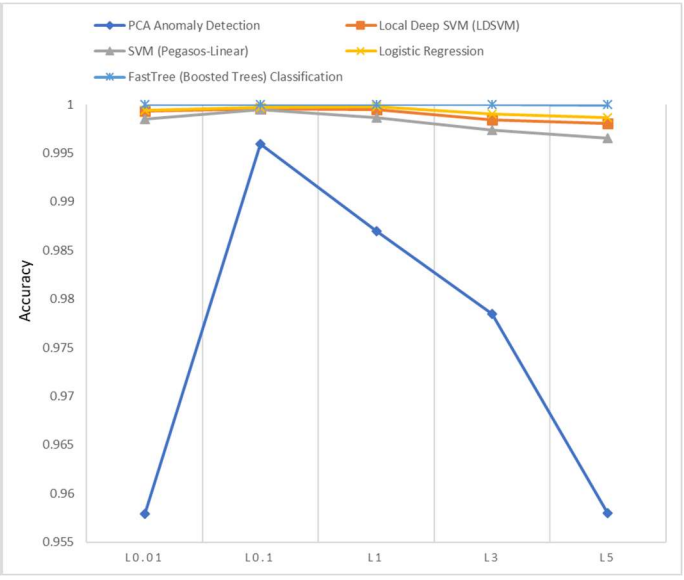


Figure 5-12 Accuracy over considering Mirai attack traffic only.

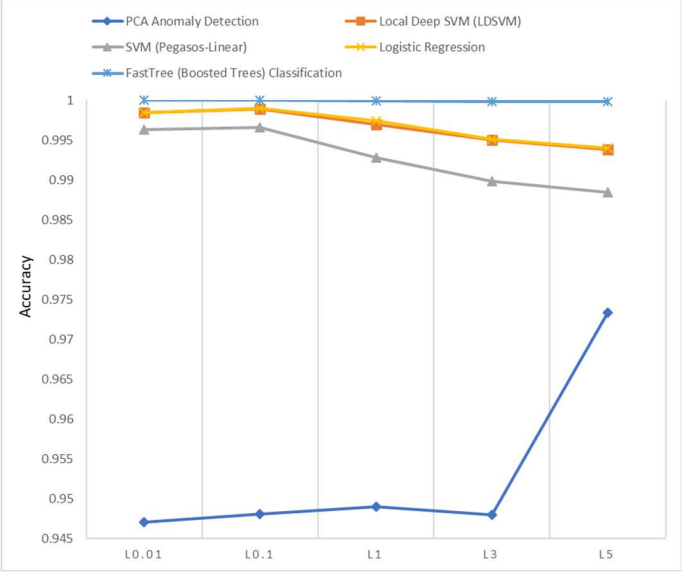


Figure 5-13 Accuracy over considering Gafgyt attack traffic only.

It is recognizable that the highest accuracy resulted from the Boosted Tree and the lowest was produced by the PCA over all three experiment sets and over all time-windows. The accuracy of Logistic Regression slightly overcomes the Local Deep SVM over all three experiments and for all time-windows. The optimum time-window size for detecting each of Mirai or Gafgyt separately is L0.1 for all models in terms of accuracy, and when considering all traffic types is L3.

Figure 5-14 manifests that the highest precision value resulted from the Boosted Tree model, which ranged from 0.9928 at L0.01 to 0.9999 at L0.1. The lowest precision, in contrast, value resulted from the PCA model, which ranged from 0.5337 at L0.01 to 0.7862 at L5. The Logistic Regression model slightly overcame the Local Deep SVM over all time-window sizes. Again, the precision value resulted from the PCA that increases as the time-window size decreases, while for the rest of the models, the precision value decreases for time-window sizes equal to or smaller than L1.

Figure 5-15 shows the highest precision value resulting from the Boosted Tree model, which ranges from 0.99994 at L0.01 to 0.99999 at L1, while the lowest precision over all time-windows is the PCA model, which ranges from 0.7584 at L0.01 to 0.9714 at L0.1. Similarly, the Logistic Regression model slightly overcomes the Local Deep SVM over all time-window sizes. The precision value of PCA diminishes for all time-windows smaller than L0.1. For the Boosted Tree, the precision value decreases when the time-window size is smaller than L1. For the rest of the models, the reduction occurs when the time-window size is smaller than L0.01 except for Logistic Regression, where the precision value increases at time-window L1 then drops again.

Similar to Figure 5-15, in Figure 5-16 the highest precision value resulted from the Boosted Tree model, which ranges from 0.9999 at L0.01 to 0.9996 at L3. The lowest precision value resulted from the PCA model, which ranges from 0.7556 at L0.01 to 0.8632 at L5. The Logistic Regression model slightly overcomes the Local Deep SVM in the experiment for the time-window sizes L0.01 and L0.1, while Local Deep SVM overcomes the rest's logistic regression. Interestingly, the precision value resulted from the PCA increases as the time-window size decreases. For Local Deep SVM and Boosted Tree, the precision value decreases as the time-window size decreases. The Local Deep SVM and Logistic Regression precision values decline

as the time-window declines, except for time-window L0.1 where it slightly increases then drops.

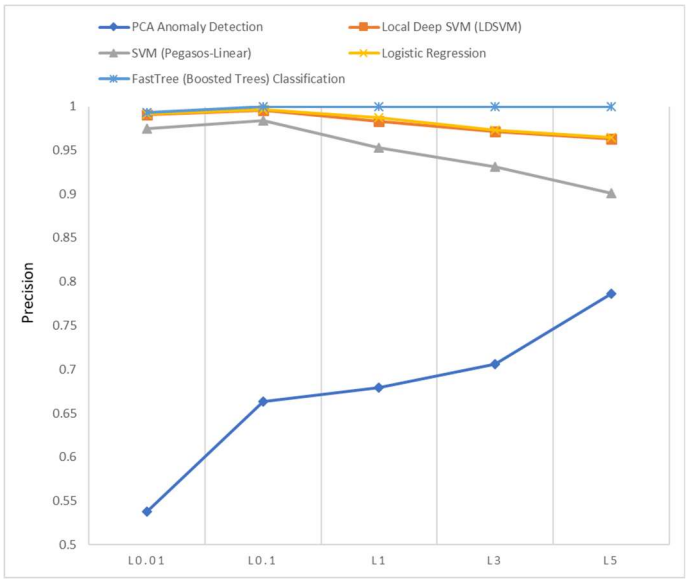


Figure 5-14 Precision over considering all traffic types.

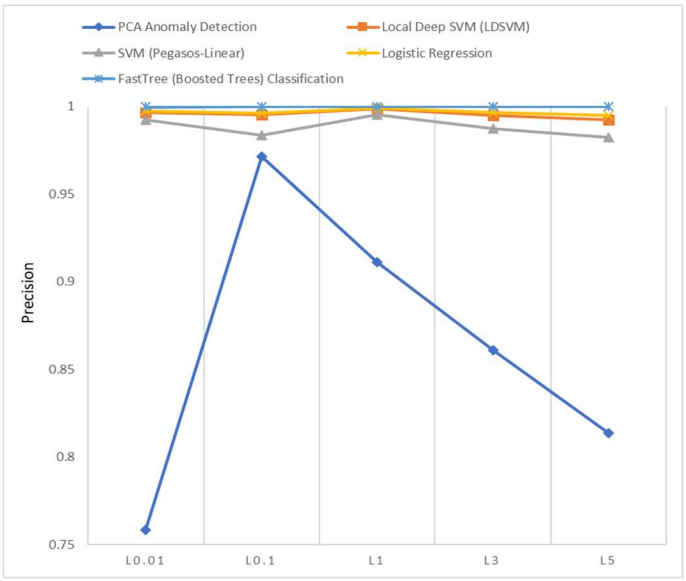


Figure 5-15 Precision over considering Mirai attack traffic only.

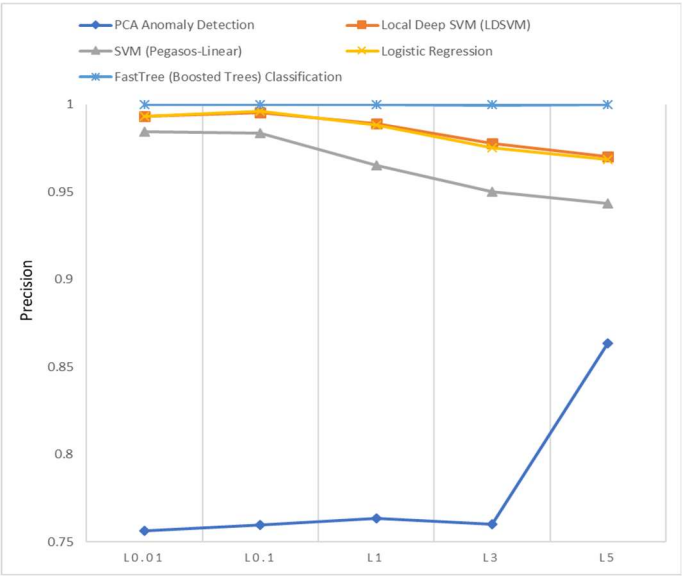


Figure 5-16 Precision over considering Gafgyt attack traffic only.

The highest precision resulted from the Boosted Tree and the lowest by the PCA over all three experiment sets and over all time-windows. The precision of Logistic Regression slightly overcomes the Local Deep SVM over all three experiment sets and for all time-windows except

for Gafgyt traffic for time-windows L1-L5. Yet the optimum time-window size for detection, on average, for all three experiment sets is L0.1.

The Boosted Tree has the best recall results over all time-windows, and the lowest resulted from the SVM, as indicated in Figure 5-17. All the models show the same behavior as demonstrated in the previous figures, where the recall values decrease as the time-window size decreases, except for PCA. Most of the maximum recall values are generated at time-window L0.1.

In Figure 5-18, the best recall results over all time-windows are generated by the Boosted Tree, and the lowest result from the SVM for time-window L0.01-L3 and PCA at L5. All the models show the same behavior, again, where the recall values decrease as the time-window size becomes smaller than L0.1. The Logistic Regression overcomes the Local Deep SVM over all time-windows, and the difference increases as the time-window size decreases. Most of the maximum recall values are generated at time-window L0.1.

Same as Figure 5-18, Figure 5-19 represents the Boosted Tree that generates the best recall results over all time-windows, and the lowest recall resulted from the SVM. All the models show the same behavior, where the recall values decrease as the time-window size decreases after L0.1.

The Logistic Regression clearly surpasses the Local Deep SVM over all time-windows except at L5. Most of the maximum recall values are generated at time-window L0.1. The highest recall results were by the Boosted Tree, and the lowest were by the SVM for the three experiment sets and over all time-windows. The recall of Logistic Regression overcomes the Local Deep SVM over all three experiment sets and for all time-windows. The optimum time-window size for recall, on average, for all the three experiments sets is L0.1.

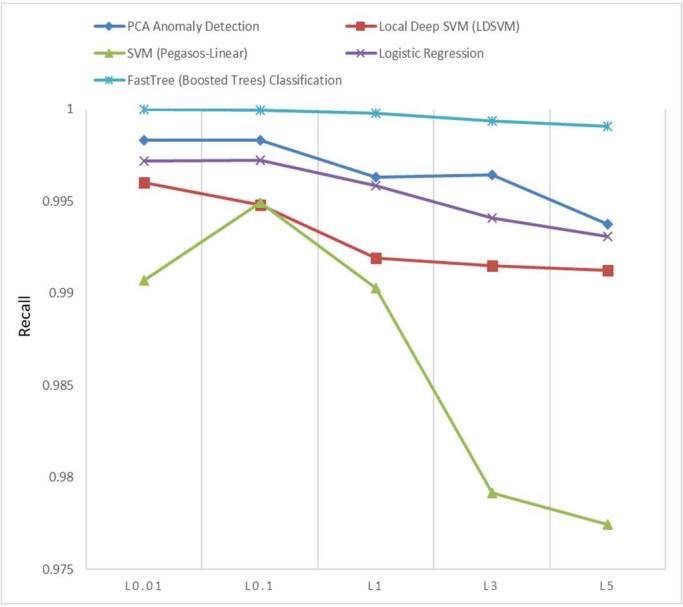


Figure 5-17 Recall over considering all traffic.

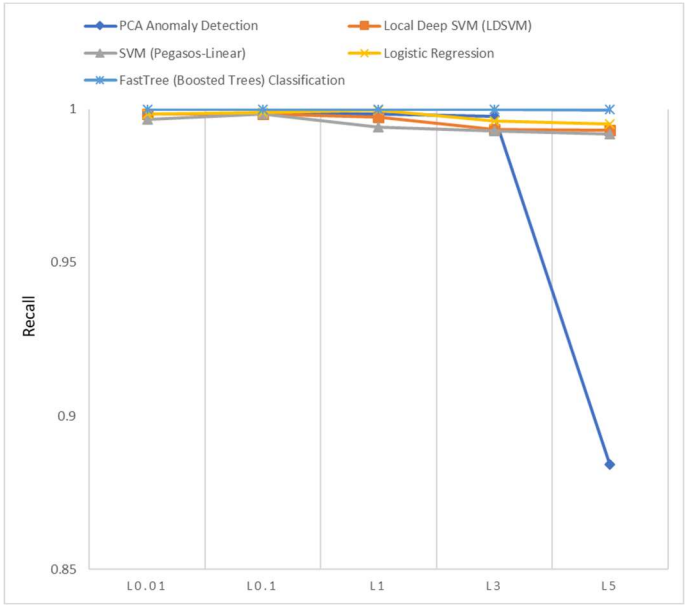


Figure 5-18 Recall over considering Mirai attack traffic only.

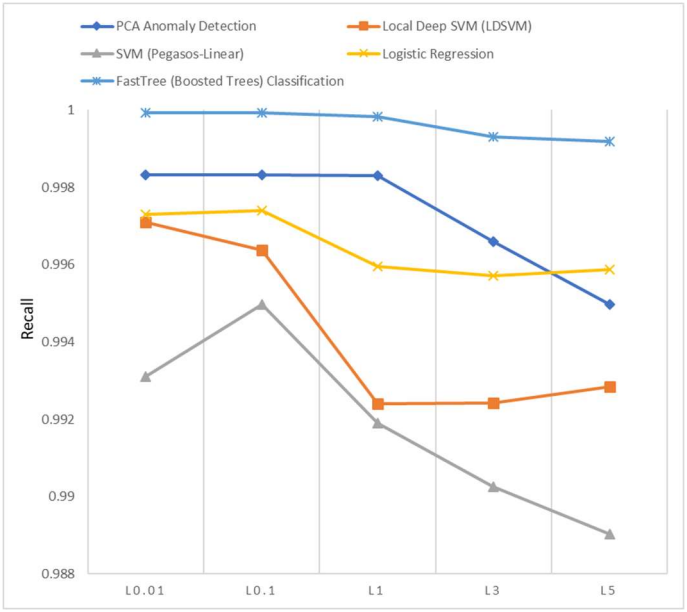


Figure 5-19 Recall over considering Gafgyt attack traffic only.

5.2.2 Conclusion

The PCA has the worst performance for all the performance metrics, except for the recall metric, where it is the second-best model for all time-windows except for L5; its recall drops drastically. The Boosted Tree is the best for all metrics overall time-windows and all traffic types. In general, all models' performance decreases as the time-window decreases. Thus, the optimum time-window is L0.1, which stands for 10 seconds time-window. The L0.1 is the best because more information is collected than the shorter time-windows and less noise than the largest time-

window. Thus, the following sections considered the 23 features at a time-window 10 seconds only. Importantly, the performance of all models diminishes when the heterogeneity of the traffic increases.

Considering Figure 5-11 to Figure 5-19, the Boosted Tree, which is an ensemble detection method, shows the least performance diminution as the time-window shrinks. Additionally, it barely demonstrates any performance infection as the traffic heterogeneity level increases. However, other models' performance was significantly affected, where their performance diminished as time-window shrunk, and the level of heterogeneity level increased. Moreover, experiments show that some models' performance and behavior cannot be expected by changing any environmental parameter, such as the PCA detection model, which is undoubtedly an unfortunate choice for IDS.

5.3 FSM

The FSM experiments in this section aim to construct an ENFSM, and finding the best confidence level of a generated feature set by the proposed ENFSM. This section experiments used fifteen different classifiers. These were SVM using Radial Basis Function (rbf) and sigmoid kernel functions, stochastic gradient descent (SGD) using logistic regression loss (log), Bernoulli-NB, Multi-layer Perceptron (MLP) using lbfgs, SGD, and adam solvers for weight optimization, DT, Knn, Gaussian-NB, Logistic Regression, Bag-DT, Random-Forest, AdaBoost, and Gradient Boosting. The last four classifiers are ensemble classifiers. The ensemble classifiers were tested by considering 1, 10, and 100 estimators. The FSM experiments only considered the highest ensemble classifier configuration's performance in terms of F and ROC-AUC scores to contribute the F and ROC-AUC ratios, respectively. Thus, the 100 estimator's configuration was chosen. The previous consideration was made to give each

model's type the same weight in contributing the efficiency measurements to avoid overfitting and bias results.

In the literature, FSMs are divided into filters and wrappers. Moreover, FSMs are classified based on their selection criteria, which measure the features' quality. There are five selection criteria categories: information, distance, dependence, consistency, and accuracy measures. This research considered criteria measures to select features that are strongly correlated with the class label. Furthermore, this research considered a vast and various range of filters and wrappers FSMs combined with FW and BW. Below is the definition of the considered FSMs:

5.3.1.1 Filter-based FSM

- Kendal: It measures the ordinal correlation between two features [90].
- Spearman: It is a correlation and statistical measure that computes the strength of a monotonic relationship between two continuous or ordinal features [91]. The Spearman correlation coefficient is based on the ranked values for each variable rather than the raw data.
- Pearson: It is a correlation measure that finds the linear relationship between two continuous variables. A linear relationship occurs when a change in one variable is associated with a proportional change in the other variable [92].
- Mutual Information (MI): It is an information measure that computes the statistical dependence between two non-negative features. MI equals zero if and only if two features are independent, and higher values mean higher dependency [93].
- Chi-squared (Chi2): It is a non-parametric measure that evaluates the independence between two non-negative categorical features [94].

- L1-SVM (linear SVM): After learning a linear SVM, it uses the model coefficients as weights that stand for the importance of the features. The weights represent the hyperplane that separates the classes as best as possible. The less important features they have, the lower the variance.
- ANOVA (Analysis of Variance) F-value (F): It is a statistical measure that measures the independence between two or more features in terms of density. Specifically, it finds the means of two or more groups of quantitative features that are significantly different from each other. It examines if, when we group the numerical feature by the target feature, each group's means are significantly different [95]. If the variance is low, it implies there is no impact of this feature on the target feature and vice-versa.

All the aforementioned measures are considered dependency measures that use bivariate analysis to measure the strength of association between two features, except for MI, which is considered an information measure. In addition to relationship strength, Kindall, Spearman, and Pearson specify the direction of the relationship. The experiments tested these filters using the feature ranking algorithm. Additionally, we applied the proposed cutoff as well to four different percentiles 10%, 25%, 50%, and 75%. A 10% percentile stands for selecting the highest 10% scoring percentage of the nominated features.

5.3.1.2 Wrapper-based FSM

This research examined three commonly used classifiers in the literature: Knn, RF, and SVM. They were under scrutiny because they use different measurements and ways to select features, where Knn, SVM, and RF use distance, variance, and information gain, respectively. The Knn is a supervised classification model that captures the idea of similarity by calculating the distance between points on a graph. It classifies them based on a simple majority vote of each

point's nearest neighbors: a query point is assigned to the data class with the most representatives within the nearest neighbors of the point. The SVM is a classifier that learns a hyperplane that best separates the data points. The RF is an ensemble classifier that learns some decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The RF model used 100 estimators..

This research focused on each of the wrappers with FW and BW. Similar to filters, it considered the four different percentiles 10%, 25%, 50%, and 75% for features selection.

5.3.2 Results

5.3.2.1 Cutoff value for filter-based FSM

Table 5-3 to Table 5-6 show the features selected by the filter-based FSMs, which used the proposed cutoff value, using NSL-KDD, UNSW-NB15, BotNetIoT, and BoTIoT datasets, respectively. The features' count is displayed in the right column of each category. In Table 5-3, the FSMs MI-ID and Chi2-ID did not select any features because all the features' scores were below the ID feature's score. Thus, they were removed from the table. Significantly, Kendal-ID, Pearson-ID, Spearman-ID, F-Class-If-ID, and L1-SVM-ID reduced the NSL-KDD dimensionality into 19, 12, 19, 12, and 12, respectively, out of 41 features. Notably, all filters' most selected features were host-based features, except for the L1-SVM-ID, which were content and time-based features. Interestingly, Kendal-ID and Spearman-ID selected the same feature set. Similarly, Pearson-ID and F-Class-If-ID did select the same feature set. Moreover, the Kendal-ID and Spearman-ID selected the same features as Pearson-ID and F-Class-If-ID, in addition to the two basic features are dst_bypes and src_bytes features and two time-related features are diff_srv_rate and srv_diff_host_rate.

Table 5-3 Feature-ID FSMs' selected features using NSL-KDD dataset.

	FSM									
Feature type	**Kendal-ID**	**#**	**Pearson-ID**	**#**	**Spearman-ID**	**#**	**F-Class-If-ID**	**#**	**L1-SVM-ID**	**#**
Basic	dst_bytes flag service src_bytes	4	flag service	2	dst_bytes flag service src_bytes	4	flag service	2	protocol_type urgent	2
Content-Related	logged_in	1	logged_in	1	logged_in	1	logged_in	1	is_guest_login num_file_creations num_root su_attempted	4
Host-Based-Traffic	dst_host_count dst_host_diff_srv_rate dst_host_same_src_port_rate dst_host_same_srv_rate dst_host_serror_rate dst_host_srv_count dst_host_srv_diff_host_rate dst_host_srv_serror_rate	8	dst_host_count dst_host_same_srv_rate dst_host_serror_rate dst_host_srv_count dst_host_srv_serror_rate	5	dst_host_count dst_host_diff_srv_rate dst_host_same_src_port_rate dst_host_same_srv_rate dst_host_serror_rate dst_host_srv_count dst_host_srv_diff_host_rate dst_host_srv_serror_rate	8	dst_host_count dst_host_same_srv_rate dst_host_serror_rate dst_host_srv_count dst_host_srv_serror_rate	5	dst_host_rerror_rate dst_host_srv_count	2
Time-Related	srv_serror_rate count diff_srv_rate same_srv_rate serror_rate srv_diff_host_rate	6	count same_srv_rate serror_rate srv_serror_rate	4	count diff_srv_rate same_srv_rate serror_rate srv_diff_host_rate srv_serror_rate	6	count same_srv_rate serror_rate srv_serror_rate	4	diff_srv_rate same_srv_rate serror_rate srv_count	4

In Table 5-4, only MI-ID did not select features. There were 14, 4, 10, 5, 4, and 16 features selected by Kendal-ID, Pearson-ID, Spearman-ID, Chi2-ID, F-Class-If-ID, and L1-SVM-ID, respectively. Interestingly, the FSMs did not select flow-based features, except for the L1-SVM-ID, which selected a single feature only. Similarly, only Chi2-ID and L1-SVM-ID selected one and three time-based features, respectively. Pearson-ID and F-Class-If-ID selected the same features, while Kendal-ID selected four more features than Spearman-ID: dloss, spkts, dwin, ct_scr_dport_Itm. The feature set selected by the L1-SVM-ID just shared selecting swin feature with other feature sets selected by other FSMs, except Chi2-ID which didn't select it.

Table 5-4 Feature-ID FSMs'' selected features using UNSW-NB15 dataset.

	FSM											
Feature Type	**Kendal-ID**	**#**	**Pearson-ID**	**#**	**Spearman-ID**	**#**	**Chi2-ID**	**#**	**F-Class-If-ID**	**#**	**L1-SVM-ID**	**#**
Basic	state sttl dload dpkts dbytes sloss dloss spkts	8	state sttl	2	state sttl dload dpkts dbytes sloss	6	sload dload	2	state sttl	2	dur spkts dpkts	3
Content-Related	swin dmean dwin	3	swin	1	swin dmean	2	stcpb dtcpb	2	swin	1	swin trans_depth	2
Generated-Connection	ct_dst_sport_ltm ct_src_dport_ltm	2	ct_dst_sport_ltm	1	ct_dst_sport_ltm	1	None	0	ct_dst_sport_ltm	1	ct_dst_ltm ct_dst_src_ltm ct_src_ltm	3
Generated-General Purpose	ct_state_ttl	1	None	0	ct_state_ttl	1	None	0	None	0	is_ftp_login ct_ftp_cmd ct_flw_http_mthd is_sm_ips_ports	4
Time-Related	None	0	None	0	None	0	rate	1	None	0	sinpkt tcprtt synack	3
Flow-Related	None	0	None	0	None	0	None	0	None	0	proto	1

In Table 5-5, the Kendal-ID, Spearman-ID, Chi2-ID, L1-SVM-ID, and MI-ID selected 3, 1, 2, 10, and 1 features, respectively, out of 23 features. The Pearson-ID and F-Class-If-ID did not select any features. Notably, Kendal-ID selected two additional features to HpHp_L0.1_radius, which was the only feature selected by Spearman-ID, being HpHp_L0.1_std, HpHp_L0.1_weight.

Table 5-5 Feature-ID FSMs' selected features using BotNetIoT dataset.

	FSM									
Feature Type	**Kendal-ID**	**#**	**Spearman-ID**	**#**	**Chi2-ID**	**#**	**L1-SVM-ID**	**#**	**MI-ID**	**#**
Temporal-Statistical/ Time-Related	HpHp_L0.1_radius HpHp_L0.1_std HpHp_L0.1_weight	3	HpHp_L0.1_radius	1	HH_jit_L0.1_mean HH_jit_L0.1_variance	2	H_L0.1_mean H_L0.1_variance HH_L0.1_mean HH_L0.1_magnitude HH_L0.1_radius HH_L0.1_covariance HpHp_L0.1_weight HpHp_L0.1_mean HpHp_L0.1_std HpHp_L0.1_pcc	10	HpHp_L0.1_magnitude	1

In Table 5-6, all FSMs were able to select features. The FSMs selected 17, 27, 14, 8, 27, 27, and 21 features out of 43 features by Kendal-ID, Pearson-ID, Spearman-ID, Chi2-ID, F-Class-If-ID, L1-SVM-ID, and MI-ID, respectively. Interestingly, the most selected features were of statistical-flow features, with the exception MI-ID mostly selected basic features. All FSMs selected the same time features, the ltime and stime, excluding Chi2-ID, which did not select any. Kendal-ID selected the same feature set as Spearman-ID, in addition to dbytes, daddr, and spkts. Pearson-ID and F-Class-If-ID selected the same feature set. Furthermore, Pearson-ID and F-Class-If-ID selected the same features.

Table 5-6 Feature-ID FSM's selected features using BoTIoT dataset.

	FSM													
Feature Type	**Kendal-ID**	**#**	**Pearson-ID**	**#**	**Spearman-ID**	**#**	**Chi2-ID**	**#**	**F-Class-If-ID**	**#**	**L1-SVM-ID**	**#**	**MI-ID**	**#**
Basic	state dbytes daddr spkts sbytes	5	state spkts pkts sbytes bytes dpkts dbytes dur	8	state sbytes	2	srate Bytes dpkts sbytes dbytes	5	pkts bytes state dur spkts dpkts sbytes dbytes	8	flgs_number sbytes dbytes	3	flgs flgs_number state state_number spkts dpkts sbytes dbytes	8
Flow	dport seq	2	daddr saddr dport seq	4	dport seq	2	None	0	saddr daddr dport seq	4	proto proto_number saddr daddr dport pkts rate	7	proto proto_number saddr daddr dport pkts bytes	7
Statistical-Flow	N_IN_Conn_P_DstIP N_IN_Conn_P_SrcIP Pkts_P_State_P_Protocol_P_DestIP TnP_PDstIP TnBPDstIP Pkts_P_State_P_Protocol_P_SrcIP stddev TnP_Per_Dport	8	TnP_PerProto TnBPSrcIP TnBPDstIP TnP_PSrcIP TnP_PDstIP TnP_Per_Dport sum Pkts_P_State_P_Protocol_P_DestIP Pkts_P_State_P_Protocol_P_SrcIP N_IN_Conn_P_DstIP N_IN_Conn_P_SrcIP max srate	13	N_IN_Conn_P_DstIP N_IN_Conn_P_SrcIP Pkts_P_State_P_Protocol_P_DestIP TnP_PDstIP TnBPDstIP Pkts_P_State_P_Protocol_P_SrcIP stddev TnP_Per_Dport	8	TnBPSrcIP TnBPDstIP TnP_PerProto	3	sum max srate TnBPSrcIP TnBPDstIP TnP_PSrcIP TnP_PDstIP TnP_PerProto TnP_Per_Dport N_IN_Conn_P_DstIP N_IN_Conn_P_SrcIP Pkts_P_State_P_Protocol_P_DestIP Pkts_P_State_P_Protocol_P_SrcIP	13	min TnBPSrcIP TnBPDstIP TnP_PDstIP TnP_PerProto TnP_Per_Dport AR_P_Proto_P_SrcIP AR_P_Proto_P_DstIP Pkts_P_State_P_Protocol_P_SrcIP TnP_PerProto TnP_Per_Dport N_IN_Conn_P_DstIP N_IN_Conn_P_SrcIP Pkts_P_State_P_Protocol_P_DestIP Pkts_P_State_P_Protocol_P_SrcIP	15	TnBPSrcIP TnBPDstIP TnP_PSrcIP TnP_PDstIP	4
Time	stime ltime	2	stime ltime	2	stime ltime	2	None	0	stime ltime	2	stime ltime	2	stime ltime	2

From Table 5-3 to Table 5-6, it should be noted that Spearman-ID's feature set is a subset of Kendal-ID's feature set. Moreover, Pearson-ID's feature set is equal to F-Class-If-ID's feature set. MI-ID was not able to select features in NSL-KDD and UNSW-NB15 datasets.

Figure 5-20 to Figure 5-27 demonstrate the F and ROC-AUC scores for the fifteen considered models using all the filter-based FSMs. Figure 5-24 to Figure 5-27, the SVM models were removed because they failed to give results. These figures aimed to show the impact of feature selection on models' performance. Furthermore, they aimed at comparing the filter-based FSMs using the proposed cutoff with other filter-based FSMs. Experiments used four percentiles, 10%, 25%, 50%, and 75%, to select the *k* best features. Moreover, experiments used a single estimator for the ensemble models, which showed the lowest performance. Also, experiments showed that as the number of estimators increases, the model accuracy increases.

In Figure 5-20, Chi2-ID and MI-ID were removed because they did not select any features. Interestingly, all models using any FSMs showed F scores higher than or equal to using all features, with the exception of SGD-log and LogisticRegression which used MI-0.1. Most of the models showed F scores ranging from 0.75 to 1 using FSMs. The Pearson-ID showed the best results, on average, over all other filter-based FSMs used the ID-cutoff. In contrast, the worst performance was by the Spearman-ID. Remarkably, the Kendal-ID showed performance close to Kendal-0.75. The Pearson-ID, F-Class-If-ID, and Spearman-ID showed performance close to the corresponding FSMs using percentile 0.25. Spearman-ID showed the lowest performance using the MLPClf-lbfgs model. The Knn, GaussianNB, LogisticRegression, Bag-DT(1), Random-Forest(1), Ada-Boost(1), and Gradient-Boosting(1) showed a stable performance, which was barely affected by the feature set. The Bag-DT(1) demonstrated the best results of all FSMs.

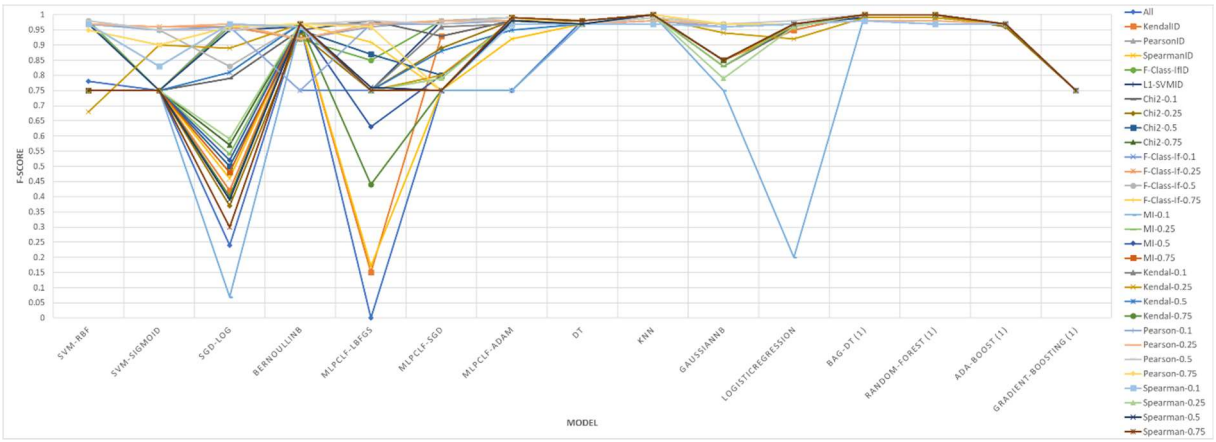


Figure 5-20 F-scores of filter-based FSMs using NSL-KDD dataset.

Figure 5-21 presents the ROC-AUC scores using the NSL-KDD dataset. The Chi2-ID and MI-ID were removed from the figure because of their failing to select any feature. The Pearson-ID showed the highest score between all filter-based FSMs that used ID-cutoff. In contrast, the Spearman-ID displayed the lowest results. The Kendal-ID showed results close to Kendal-0.1, the Person-ID's results were close to Pearson-0.5, the Spearman-ID's results were close to Spearman-0.1, and F-Class-If-ID's results were close to F-Class-If-0.5. In fact, SVM-sigmoid conveyed the worst scores, while the Bag-DT(1) showed the best score among the FSMs. All the models using the filters with ID-cutoff showed scores higher than or equal to the score using all the features, except for the MLPClf-lbfgs using Kendal-ID.

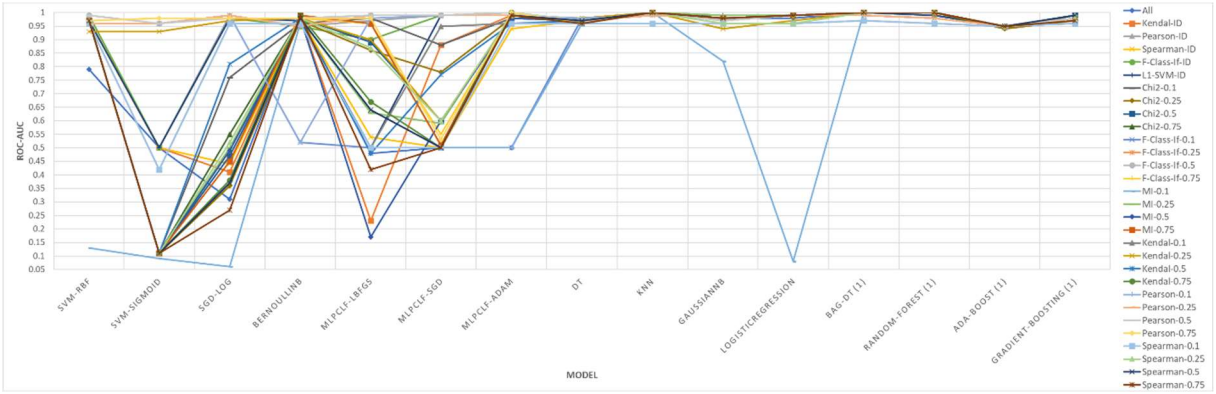


Figure 5-21 ROC-AUC scores of filter-based FSMs using NSL-KDD dataset.

Figure 5-22 displays the F scores using the UNSW-NB15 dataset. The MI-ID was removed given its failure to select features. F-Class-If-ID and L1-SVM-ID showed scores higher than

using all-features method, while the rest of the filters used ID-cutoff presented results close to all-features method's scores. Significantly, the F-Class-If-ID overcame all F-Class-If using percentiles. In effect, Kendal-ID score was close to Kendal-0.5, Pearson-ID score was close to Pearson-0.75, Spearman-ID score was close to Spearman-0.25, and Chi2-ID score was close to Chi2-0.1. Interestingly, Bag-DT(1) model showed the highest scores, while MLPClf-sgd model exhibited the lowest scores over all the FSMs.

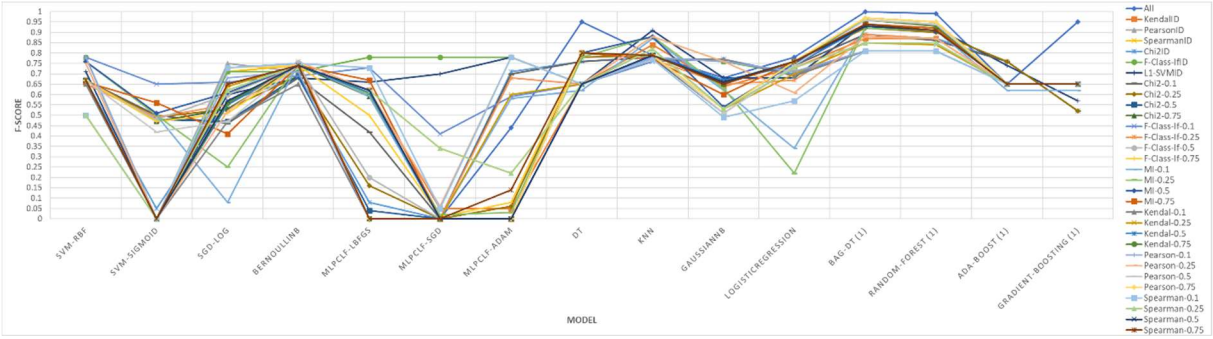


Figure 5-22 F-scores of filter-based FSM using UNSW-NB15 dataset.

In Figure 5-23, the F-Class-If-ID scores were higher than all FSMs. Moreover, Kendal-ID, Spearman-ID, and Chi2-ID displayed scores higher than the corresponding filter percentiles. The Pearson-ID showed scores similar to Pearson-0.1. The Bag-DT(1) model showed the highest scores, while SVM-sigmoid model showed the lowest.

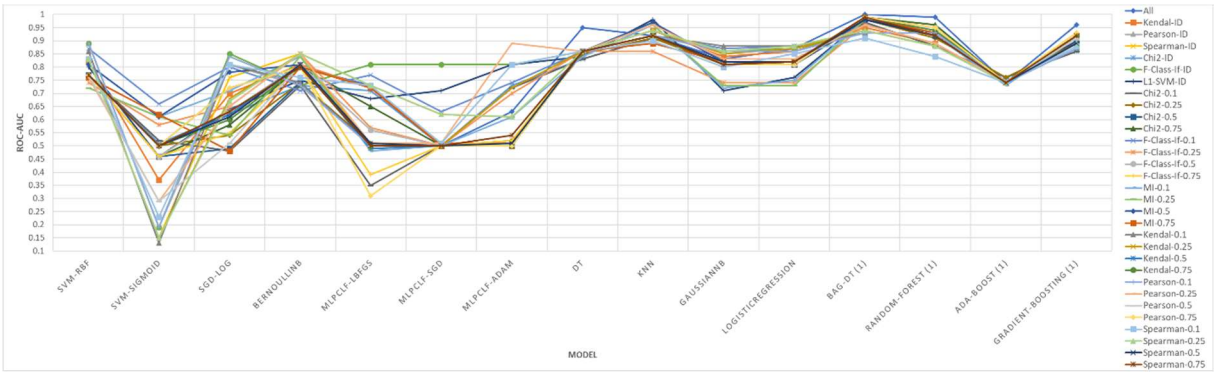


Figure 5-23 ROC-AUC scores of filter-based FSM using UNSW-NB15 dataset.

Figure 5-24 reveals the F scores using the BotNetIoT dataset. Pearson-ID and F-Class-If-ID did not select any features. Thus, they were removed from the figure. L1-SVM-ID showed the highest scores over all filters using ID-cutoff. Moreover, Kendal-ID and L1-SVM-ID presented

scores higher than using all-features method. However, all the filters using ID-cutoff showed scores lower than the corresponding filters using percentiles. MI-0.1 showed the highest scores. Interestingly, Gradient-Boosting(1) was the worst model, while Bag-DT(1) was the best.

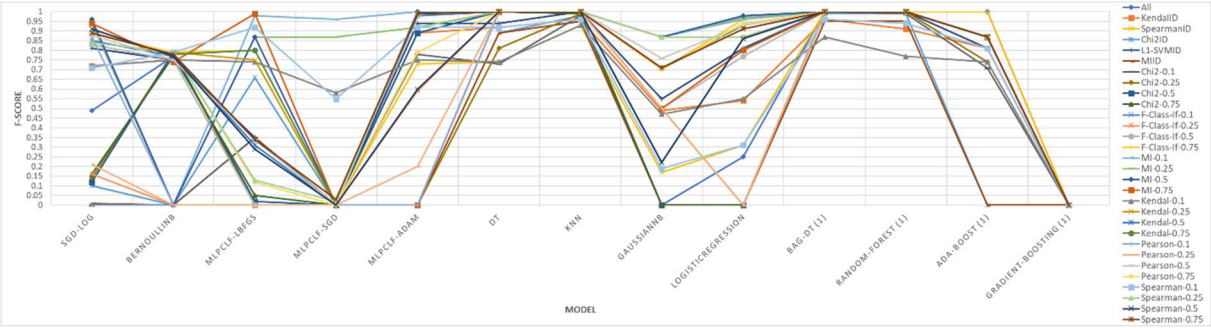


Figure 5-24 F-scores of filter-based FSM using BotNetIoT dataset.

Figure 5-25 shows the same behavior as Figure 5-24, with the exception of Chi2-ID scores, which were similar to Chi2-0.1. Interestingly, Knn models showed the highest scores. In contrast, MLPClf-sgd model showed the lowest scores.

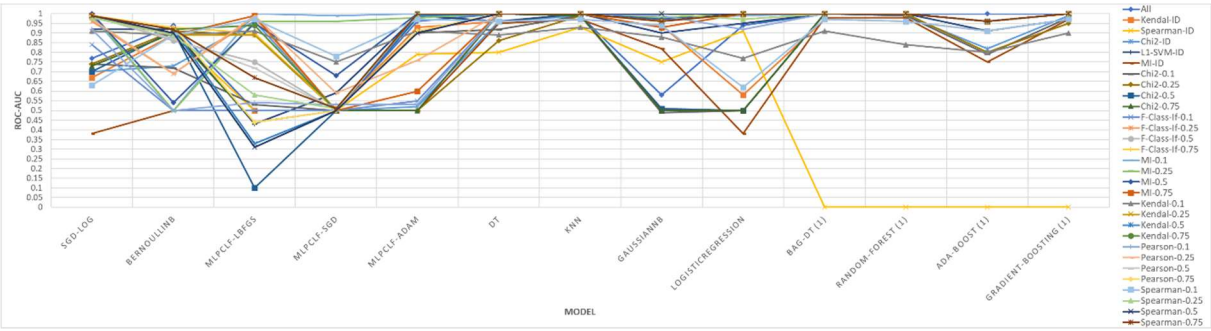


Figure 5-25 ROC-AUC scores of filter-based FSM using BotNetIoT dataset.

Figure 5-26 introduces the F scores using the BoTIoT dataset. The SVMs models were removed for the same reasons that occurred using the BotNetIoT dataset. Also, BernoulliNB showed zero for all FSMs models, which was the worst score and was removed to increase the figure readability. F-Class-If-ID showed the highest scores between all the filters using ID-cutoff. However, the filters using the ID-cutoff did not present scores higher than using all features. MI-0.5 and Chi2-0.5 showed the highest scores. Moreover, F-Class-If-0.1 and F-Class-

If-0.25 showed scores higher than using all-features method. The Bag-DT(1) model showed the highest scores of all FSMs.

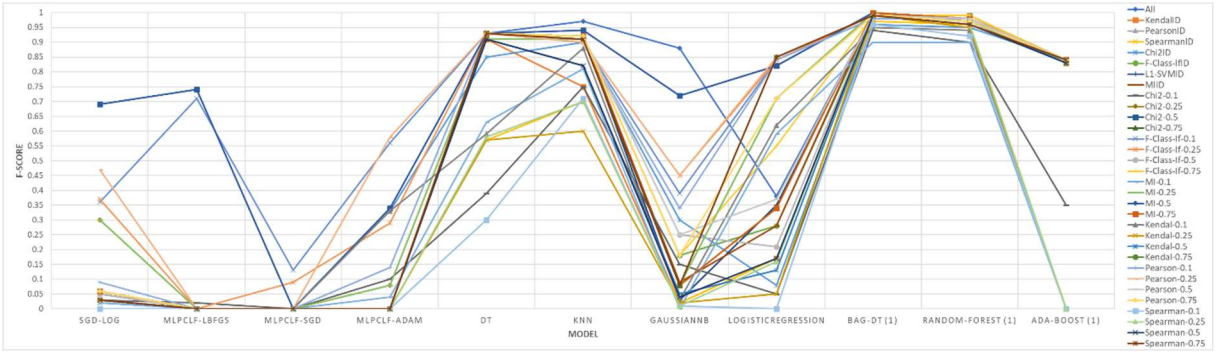


Figure 5-26 F-scores of filter-based FSM using BoTIoT dataset.

Figure 5-27 showed the ROC-AUC scores using the BoTIoT dataset. MI-0.5 and Chi2-0.5 showed the highest scores over all other FSMs. In their turn, Pearson-ID and F-Class-If-ID showed the highest score over all filters that used ID-cutoff and all the same filters using percentiles. Their scores were also close to using all features. Specifically, Kendal-ID displayed scores similar to Kendal-0.5. Spearman-ID showed scores similar to Spearman-0.25. It is noticeable that MLPClf-sgd model showed the lowest scores, while the Bag-DT(1) model showed the highest scores over all FSMs.

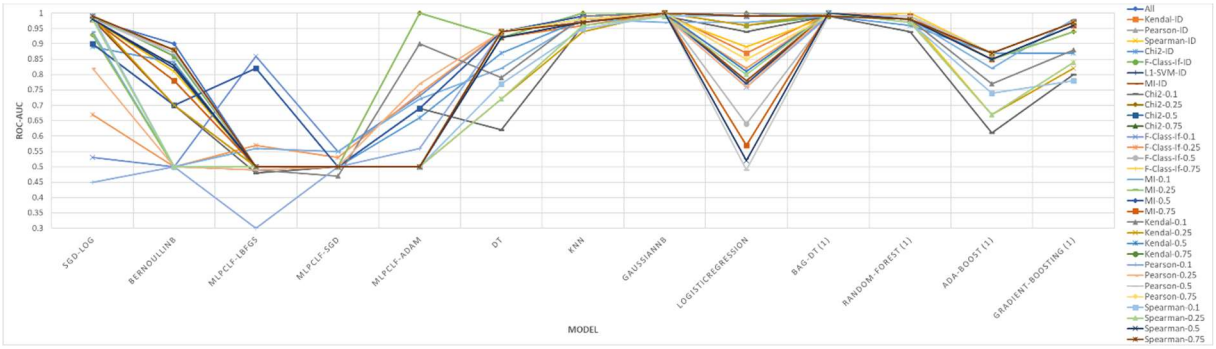


Figure 5-27 ROC-AUC scores of filter-based FSM using BoTIoT dataset.

5.3.2.2 Ensemble FSM selection method

5.3.2.2.1 FSMs selection

To meet diversity and give each FSM the same weight in contributing to the ENFSM, only the most efficient FSM percentile was selected. Thus, the final total number of FSMs considered in

constructing ENFSM was nineteen. Table 5-7 shows performance measures using the NSL-KDD dataset for all FSMs examined in this research. The scores were ranked in descending order. Thus, an FSM's score ranked 1 means that this FSM had the highest and the best score. Therefore, this FSM is considered as that measurement recommended FSM. After ranking all measurements' scores and finding each recommended FSM by each measurement, the total number of measurement recommendations for each FSM was counted. The FSM with the highest rank summation or the highest number of recommendations was selected to be part of the ENFSM. In Table 5-7, Kendal-ID had the highest ranks summation that equals to 2, where it was recommended by two measurements are F-ratio and ROC-AUC-Ratio. While Pearson-ID, Spearman-ID, F-Class-If-ID, L1-SVM-ID, and Knn-BW-0.1 had the same number of recommendations, that is one. Accordingly, an inner ranking between them took place, as shown in Figure 5-28. In the first inner ranking step, Spearman-ID and L1-SVM-ID were selected. The other FSMs still have the same number of recommendations which equals one. Thus, another inner ranking step took place between them. In the second inner ranking step, Pearson-ID was selected. Again, there is a tie between the F-Class-If-ID and Knn-BW-0.1. Thus, the fourth and the last inner ranking step took place in which F-Class-If-ID was selected. If there was a tie in the last inner ranking, then the proposed FSM selection method selects the FSM with the least ranks summation, which stands for the most recommended FSM by all measurements. Finally, the five FSMs selected to be part of the ENFSM for the NSL-KDD dataset were Kendal-ID, Spearman-ID, L1-SVM-ID, Pearson-ID, F-Class-If-ID.

Table 5-7 FSMs scores using NSL-KDD dataset.

FSM	RE	Rank	PCA-IG	Rank	PCA-Var	Rank	F-Ratio	Rank	ROCAUC-Ratio	Rank	Ranks Sum
Kendal-ID	2.07	3	0.59	6	0.79	11	0.93	1	1.00	1	**2**
Pearson-ID	6.14	1	0.59	4	0.75	12	0.80	8	0.87	5	**1**
Spearman-ID	0.63	5	0.57	11	0.89	5	0.93	1	0.93	2	**1**
F-Class-If-ID	0.00	13	0.61	2	1.00	1	0.47	12	0.67	11	**1**

L1-SVM-ID	0.00	15	0.58	9	0.86	8	0.93	1	0.93	2	**1**
Knn-BW-0.1	0.00	10	0.63	1	0.89	6	0.46	15	0.47	13	**1**
Chi2-0.1	1.05	4	0.59	5	0.75	12	0.87	4	0.93	2	0
F-Class-If-0.25	0.06	7	0.59	7	0.70	14	0.53	11	0.67	11	0
MI-0.1	0.00	16	0.00	16	0.00	16	0.00	16	0.00	16	0
Kendal-0.1	0.04	8	0.60	3	0.70	15	0.87	4	0.87	5	0
Pearson-0.1	0.00	16	0.00	16	0.00	16	0.00	16	0.00	16	0
Spearman-0.1	4.37	2	0.59	8	0.91	4	0.67	10	0.80	8	0
Knn-FW-0.1	0.03	9	0.55	14	0.82	9	0.87	4	0.87	5	0
RF-BW-0.1	0.25	6	0.55	15	0.82	10	0.80	8	0.80	8	0
RF-FW-0.1	0.00	12	0.57	12	0.93	3	0.47	13	0.47	13	0
SVM-BW-0.1	0.00	11	0.57	13	0.93	2	0.47	13	0.47	13	0
SVM-FW-0.1	0.00	14	0.57	10	0.89	7	0.87	4	0.80	8	0

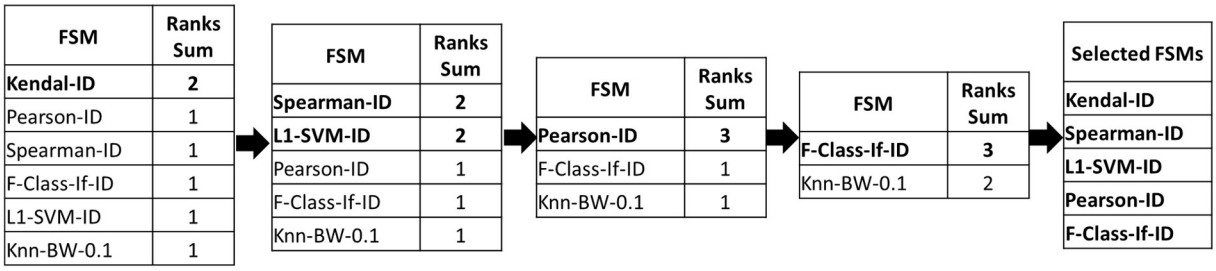


Figure 5-28 FSMs selections method using NSL-KDD dataset.

Table 5-8 summarizes the selected FSMs, feature sets, and feature counts for each dataset's ENFSM. Interestingly, all the selected FSMs for the NSL-KDD dataset's ENFSM were filters using the proposed ID-cutoff. Moreover, in each dataset's ENFSM, there was a filter using ID-cutoff. It is noticeable that the Pearson filter was selected in each ENFSM. Remarkably, only wrappers using RF and SVM were selected. BotNetIoT ENFSM had RF wrapper, BoTIoT ENFSM had SVM wrapper, and UNSW-NB15 ENFSM had RF and SVM wrappers. In addition to that, all the selected wrappers used the BW, except in BoTIoT used FW.

Table 5-8 Selected FSMs for ENFSM for each dataset.

Dataset	Ensemble FSM	Feature Set	# D
NSL	Kendal-ID	'same_srv_rate' 'diff_srv_rate' 'serror_rate' 'flag' 'srv_serror_rate' 'dst_host_serror_rate' 'logged_in' 'dst_host_srv_serror_rate' 'dst_host_same_srv_rate' 'dst_bytes''src_bytes' 'count' 'dst_host_srv_count' 'dst_host_diff_srv_rate''service' 'dst_host_count' 'dst_host_srv_diff_host_rate''dst_host_same_src_port_rate' 'srv_diff_host_rate'	19
	Pearson-ID	'same_srv_rate' 'dst_host_same_srv_rate' 'count' 'dst_host_srv_count' 'serror_rate' 'dst_host_serror_rate' 'dst_host_srv_serror_rate' 'srv_serror_rate' 'logged_in' 'flag''service' 'dst_host_count'	12
	Spearman-ID	'same_srv_rate' 'diff_srv_rate' 'flag' 'serror_rate''srv_serror_rate''count''dst_host_serror_rate''src_bytes''dst_bytes''dst_host_srv_serror_rate''logged_in''dst_host_same_srv_rate''dst_host_srv_count''dst_host_diff_srv_rate''dst_host_count''service''dst_host_same_src_port_rate''dst_host_srv_diff_host_rate'' srv_diff_host_rate'	19
	F-Class-lf-ID	'service' 'flag' 'logged_in' 'count' 'serror_rate''srv_serror_rate' 'same_srv_rate' 'dst_host_count' 'dst_host_srv_count' 'dst_host_same_srv_rate' 'dst_host_serror_rate''dst_host_srv_serror_rate'	12
	L1-SVM-ID	'protocol_type''urgent''su_attempted''num_root''num_file_creations''is_guest_login''srv_count''serror_rate''same_srv_rate''diff_srv_rate''dst_host_srv_count''dst_host_rerror_rate'	12
UNSW-NB15	L1-SVM-ID	'dur' 'proto' 'spkts' 'dpkts' 'sinpkt' 'swin' 'tcprtt' 'synack''trans_depth' 'ct_dst_ltm' 'ct_dst_src_ltm' 'is_ftp_login' 'ct_ftp_cmd' 'ct_flw_http_mthd' 'ct_src_ltm' 'is_sm_ips_ports'	16
	MI-0.1	'sbytes' 'dbytes' 'sload' 'smean'	4
	Pearson-0.25	SttlStateSwinct_dst_sport_ltmdwinct_src_dport_ltmratect_state_ttlct_srv_dstct_srv_src	10
	RF-BW-0.25	'proto' 'state' 'dttl' 'swin' 'dwin' 'ct_src_dport_ltm' 'ct_dst_src_ltm' 'ct_ftp_cmd' 'ct_flw_http_mthd' 'is_sm_ips_ports'	10
	SVM-BW-0.5	'dur' 'proto' 'service' 'state' 'spkts' 'dpkts' 'sbytes' 'dbytes' 'rate' 'sttl' 'dttl' 'sload' 'dload' 'sloss' 'dloss''sinpkt' 'dinpkt' 'sjit' 'djit' 'swin' 'dwin'	21
BoTNeTIoT	MI-0.1	'MI_dir_L0.1_weight' 'H_L0.1_weight'	2
	Chi2-ID	'HH_jit_L0.1_mean' 'HH_jit_L0.1_variance'	2
	MI-ID	'HpHp_L0.1_magnitude'	1
	Pearson-0.1	MI_dir_L0.1_weightH_L0.1_weight	2
	RF-BW-0.1	'MI_dir_L0.1_weight' 'HH_jit_L0.1_mean'	2
BoTIoT	Pearson-ID	'TnP_PerProto' 'daddr' 'TnBPSrcIP' 'TnBPDstIP' 'spkts' 'pkts''TnP_PSrcIP' 'sbytes' 'saddr' 'bytes' 'TnP_PDstIP' 'TnP_Per_Dport' 'sum' 'dpkts' 'dbytes' 'Pkts_P_State_P_Protocol_P_DestIP' 'dur' 'srate' 'Pkts_P_State_P_Protocol_P_SrcIP' 'stime' 'ltime' 'N_IN_Conn_P_DstIP' 'dport' 'N_IN_Conn_P_SrcIP' 'state' 'seq' 'max'	27
	F-Class-lf-ID	'stime' 'saddr' 'daddr' 'dport' 'pkts' 'bytes' 'state' 'ltime' 'seq' 'dur' 'sum' 'max' 'spkts' 'dpkts' 'sbytes' 'dbytes' 'srate' 'TnBPSrcIP' 'TnBPDstIP' 'TnP_PSrcIP' 'TnP_PDstIP' 'TnP_PerProto' 'TnP_Per_Dport' 'N_IN_Conn_P_DstIP' 'N_IN_Conn_P_SrcIP' 'Pkts_P_State_P_Protocol_P_DestIP' 'Pkts_P_State_P_Protocol_P_SrcIP'	27
	SVM-FW-0.25	'flgs' 'flgs_number' 'proto' 'proto_number' 'saddr' 'daddr' 'dport' 'TnP_PSrcIP' 'TnP_PDstIP' 'TnP_PerProto'	10
	MI-ID	'stime' 'flgs' 'flgs_number' 'proto' 'proto_number' 'saddr' 'daddr' 'dport' 'pkts' 'bytes' 'state' 'state_number' 'ltime' 'spkts' 'dpkts' 'sbytes' 'dbytes' 'TnBPSrcIP' 'TnBPDstIP' 'TnP_PSrcIP' 'TnP_PDstIP' 'TnP_PerProto' 'TnP_Per_Dport' 'N_IN_Conn_P_DstIP' 'N_IN_Conn_P_SrcIP' 'Pkts_P_State_P_Protocol_P_DestIP' 'Pkts_P_State_P_Protocol_P_SrcIP'	27
	Chi2-0.5	'stime' 'pkts' 'bytes' 'ltime' 'seq' 'dur' 'spkts' 'dpkts''sbytes' 'dbytes' 'rate' 'srate' 'TnBPSrcIP' 'TnBPDstIP' 'TnP_PSrcIP' 'TnP_PDstIP' 'TnP_PerProto' 'TnP_Per_Dport' 'AR_P_Proto_P_DstIP' 'Pkts_P_State_P_Protocol_P_DestIP' 'Pkts_P_State_P_Protocol_P_SrcIP'	21

5.3.2.2.2 ENFSM feature sets confidence level

Figure 5-29 to Figure 5-36 show the F and ROC-AUC scores for all the used datasets using the ENFSM feature sets. In this set of experiments, five confidence levels were considered 20%, 40%, 60%, 80%, 100% standing for features selected by one, two, three, four, and five FSMs within the ENFSM.

Figure 5-29 shows the F score using the NSL-KDD dataset. The models achieved the best scores using ENFSM-Conf-0.6 feature set. In contrast, the models achieved the worst scores using ENFSM-Conf-0.2 feature set, which has more features. Surprisingly, the models' scores dropped drastically using ENFSM-Conf-1.0 feature set, which includes the least number of features. The models Knn, GaussianNB, LogisticRegression, Bag-DT (1), Bag-DT (10), Bag-

DT (100), Random-Forest (1), Random-Forest (10), Random-Forest (100), Ada-Boost (1), Ada-Boost (10), Ada-Boost (100), Gradient-Boosting (1), Gradient-Boosting (10), and Gradient-Boosting (100) achieved the highest scores all over the feature sets. In the contrary, SGD-log and Gradient-Boosting (1) achieved the worst scores.

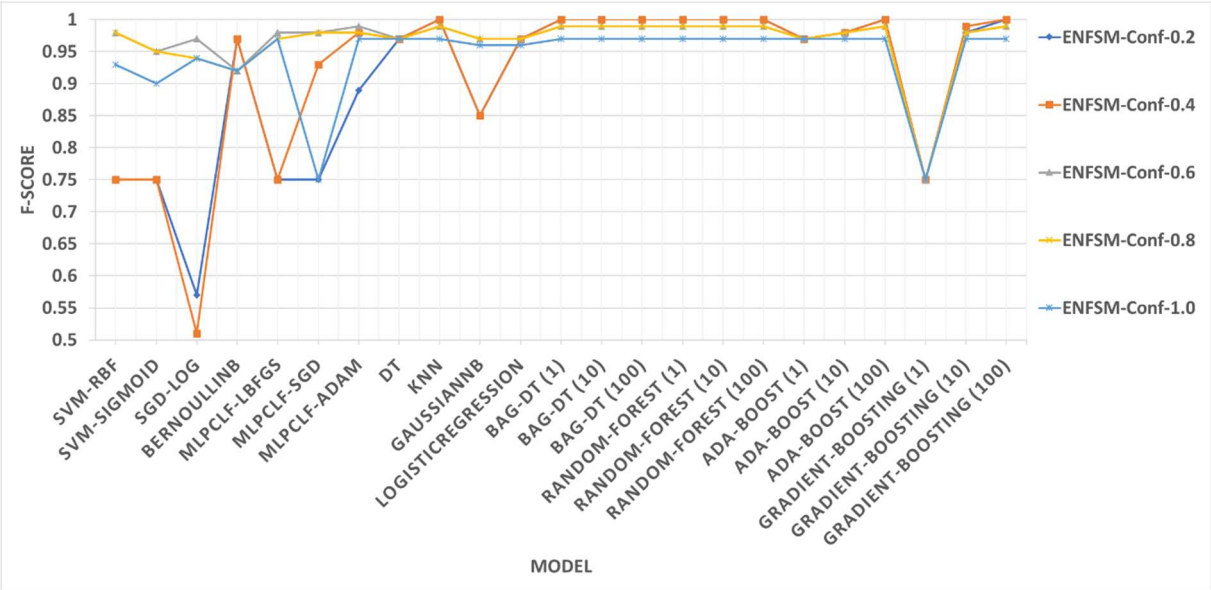


Figure 5-29 ENFSM feature sets F-scores using NSL-KDD dataset.

Figure 5-30 shows the ROC-AUC scores using NSL-KDD. The models achieved the best scores using ENFSM-Conf-0.6 feature set. In contrast, the models achieved the worst scores using ENFSM-Conf-0.2 feature set. Significantly, the models' scores dropped drastically using ENFSM-Conf-1.0 feature set. The models Knn, GaussianNB, LogisticRegression, Bag-DT (1), Bag-DT (10), Bag-DT (100), Random-Forest (1), Random-Forest (10), Random-Forest (100), Ada-Boost (1), Ada-Boost (10), Ada-Boost (100), Gradient-Boosting (1), Gradient-Boosting (10), and Gradient-Boosting (100) achieved the highest scores all over the feature sets. In contrast, SVM-sigmoid achieved the worst scores.

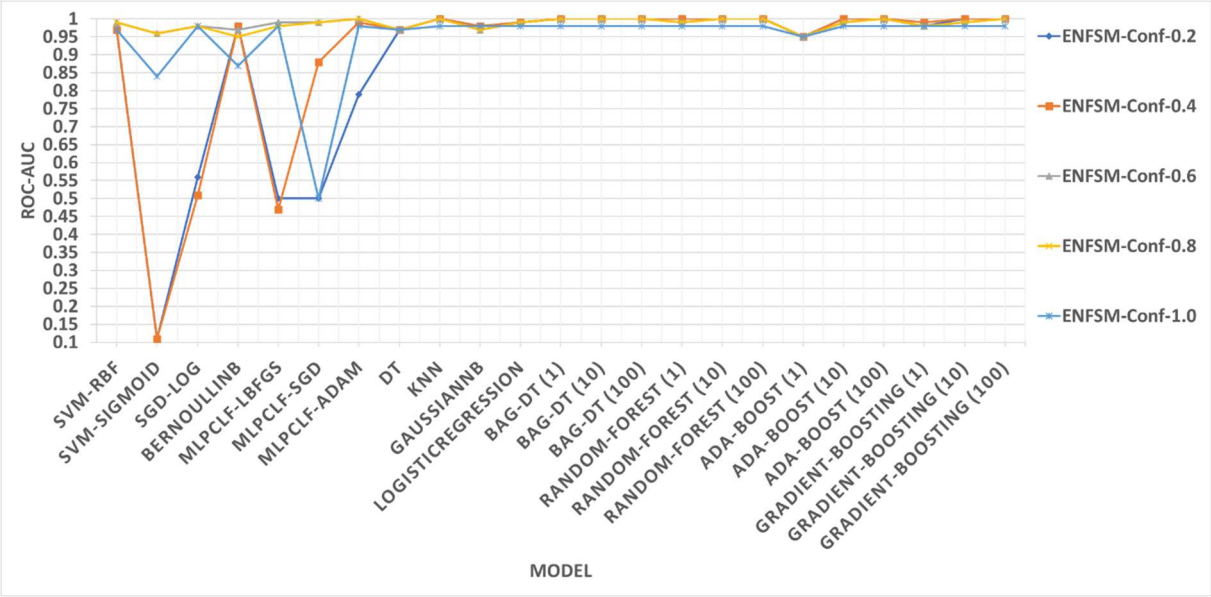


Figure 5-30 ENFSM feature sets ROC-AUC scores using NSL-KDD dataset.

Figure 5-31 shows the F scores of the considered models using ENFSM over UNSW-NB15 dataset. The ENFSM-Conf-1.0 was removed from the figure, since there were no features selected by all FSMs. Notably, the models had the highest F scores using ENFSM-Conf-0.2, which had the largest number of features. In contrast, ENFSM-Conf-0.8 had the lowest models' scores, in general. However, it should be noted that the models' scores slightly dropped using ENFSM-Conf-0.4 when comparing with ENFSM-Conf-0.2. All models' scores dropped as confidence level increased, except for SGD-log, , MLPClf-adam, GaussianNB, LogisticRegression, and Ada-Boost(1). The Random-Forest(100) had the highest scores.

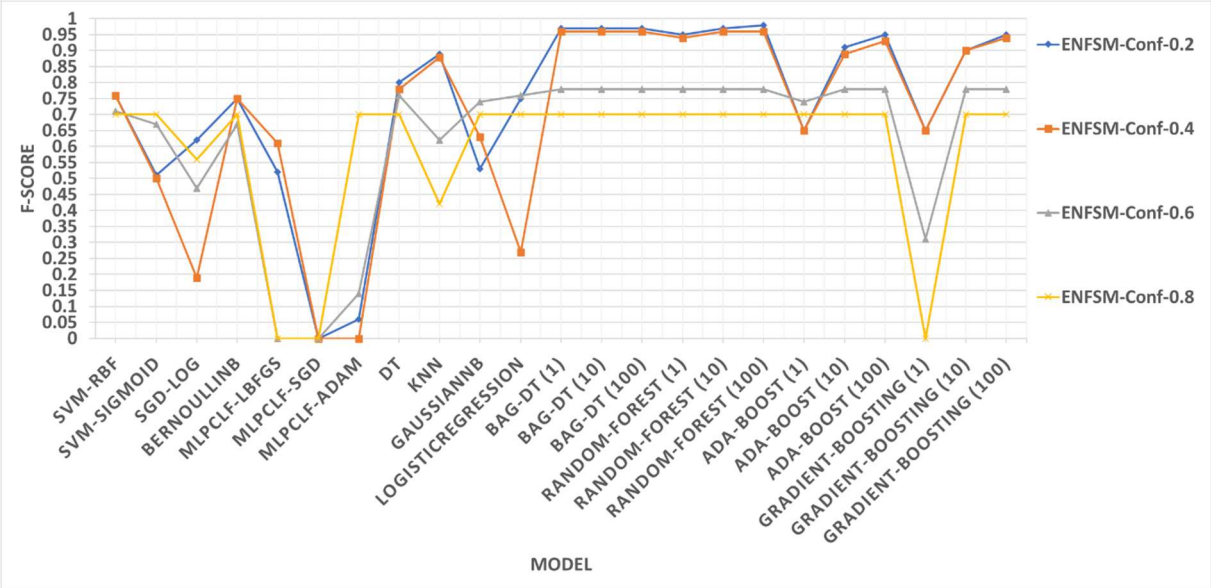


Figure 5-31 ENFSM feature sets F-scores using UNSW-NB15 dataset.

In Figure 5-32, the models had the highest ROC-AUC scores using ENFSM-Conf-0.2, which had the largest number of features. Nevertheless, ENFSM-Conf-0.8 generally had the lowest models' scores. Interestingly, The models' scores slightly dropped using ENFSM-Conf-0.4 when comparing with ENFSM-Conf-0.2. All models' scores dropped as confidence level increased, except for SVM-rbf, SVM-sigmoid, MLPClf-lbfgs, MLPClf-adam, and LogisticRegression. The Bag-DT(100), Random-Forest(10), and Random-Forest(100) showed the highest scores.

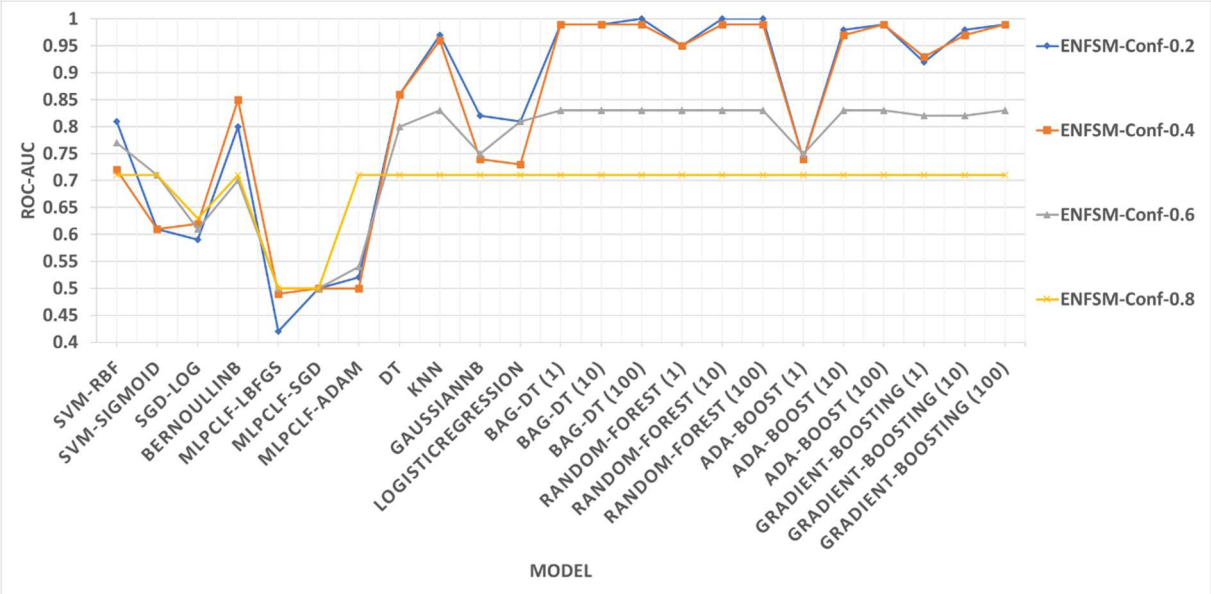


Figure 5-32 ENFSM feature sets ROC-AUC scores using UNSW-NB15 dataset.

Figure 5-33 shows the F scores for the BotNetIoT dataset. No features were selected for the confidence levels of 80% and 100%. That is why they were removed from the figure. ENFSM-Conf-0.4 had the highest models' scores. However, ENFSM-Conf-0.2 had the lowest modes' scores. Interestingly, all the ensemble models had a 100% score, except Ada-Boost(1) had 87% and Gradient-Boosting(1) zero for all feature sets. The ensemble models showed a stable behavior over all the feature sets. The worst scores were by BernoulliNB and Gradient-Boosting(1) for all feature sets.

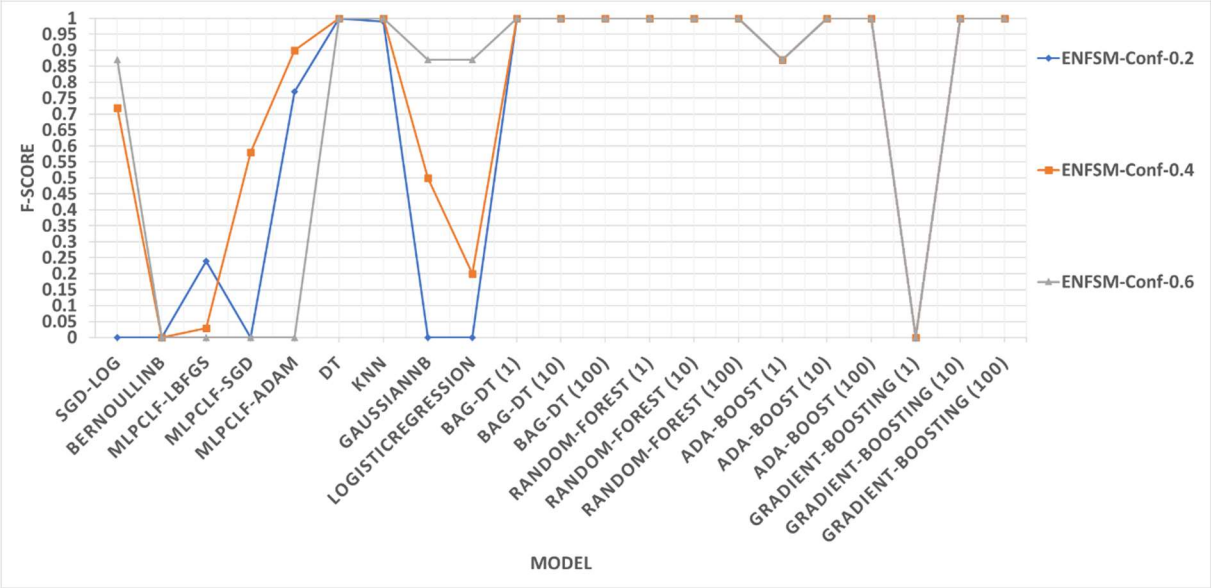


Figure 5-33 ENFSM feature sets F-scores using BotNetIoT dataset.

Figure 5-34 shows the ROC-AUC scores for the BotNetIoT dataset. ENFSM-Conf-0.4 had the highest models' scores. In contrast, ENFSM-Conf-0.2 had the lowest modes' scores. Interestingly, all the ensemble models had a score of 1, except for Ada-Boost(1) had 0.96 for all feature sets. The ensemble models showed a stable behavior over all the feature sets. The worst scores were by MLPClf-lbfgs for all feature sets.

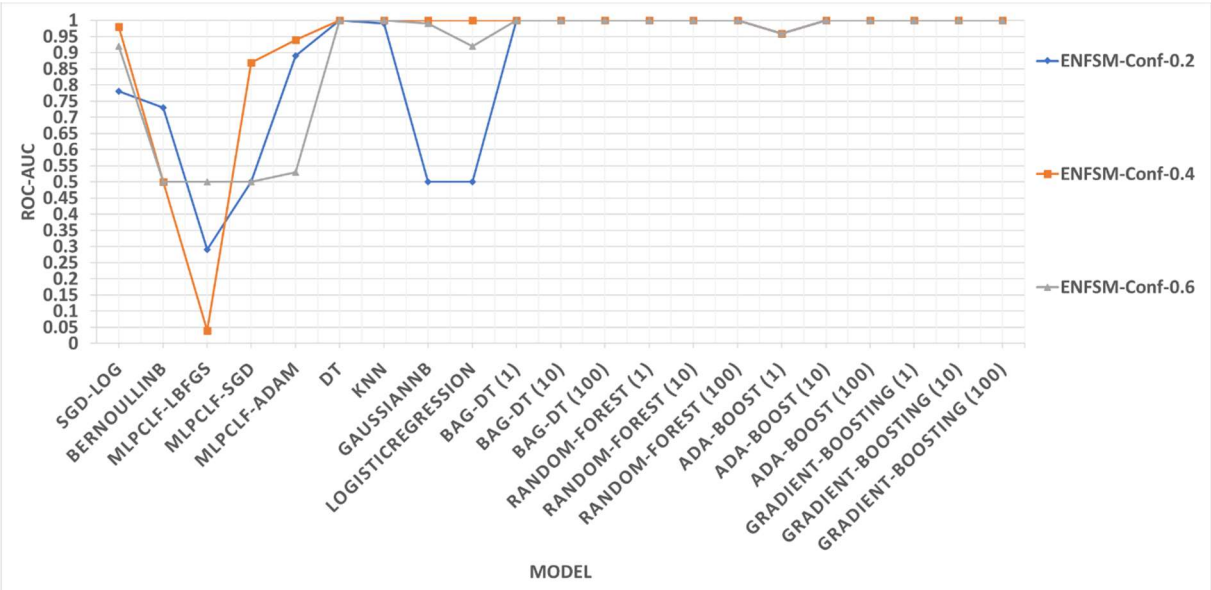


Figure 5-34 ENFSM feature sets ROC-AUC scores using BotNetIoT dataset.

In Figure 5-35 the F scores using BoTIoT are shown. Features were selected for all the confidence levels. ENFSM-Conf-0.4 showed the best scores for all models, on average. The worst scores were using ENFSM-Conf-1.0 for all models, except for MLPClf-sgd, MLPClf-adam, GaussianNB, and LogisticRegression. The Random-Forest(100) showed the highest scores and the most stable performance over all feature sets. In contrast, BernoulliNB and MLPClf-lbfgs showed the lowest scores.

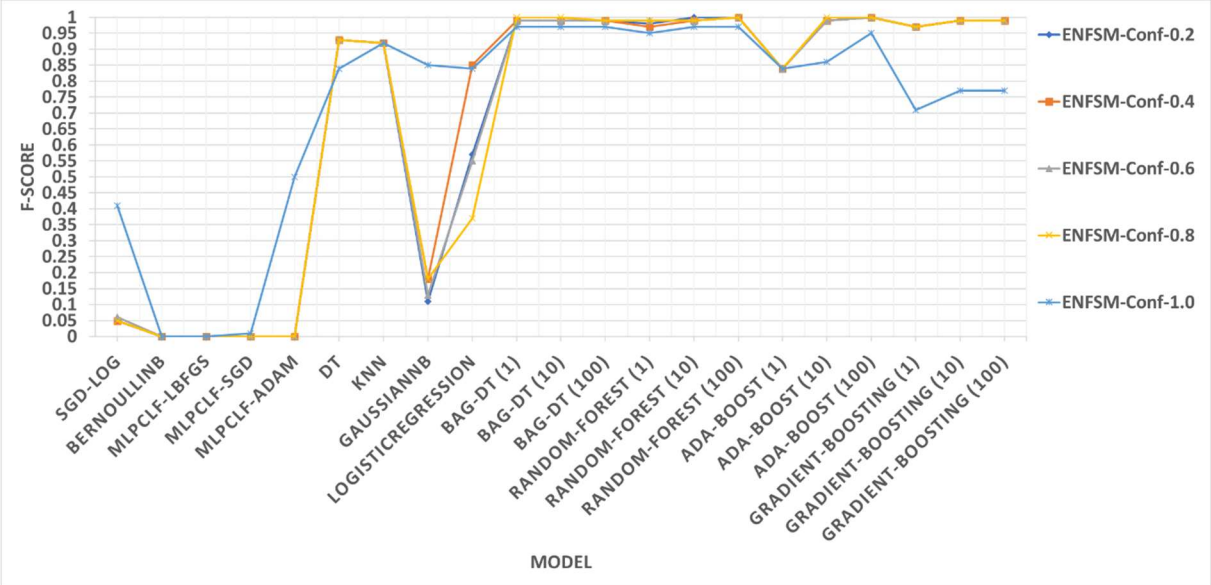


Figure 5-35 ENFSM feature sets F-scores using BoTIoT dataset.

Figure 5-36 shows the ROC-AUC scores using BoTIoT. ENFSM-Conf-0.4 had the best scores for all models. The worst scores were achieved by ENFSM-Conf-1.0 for all models, except MLPClf-adam. The Random-Forest(100), Ada-Boost(10), and Ada-Boost(100) showed the highest scores and most stable performance over all feature sets. The MLPClf-lbfgs and MLPClf-sgd, in their turn, had the lowest scores.

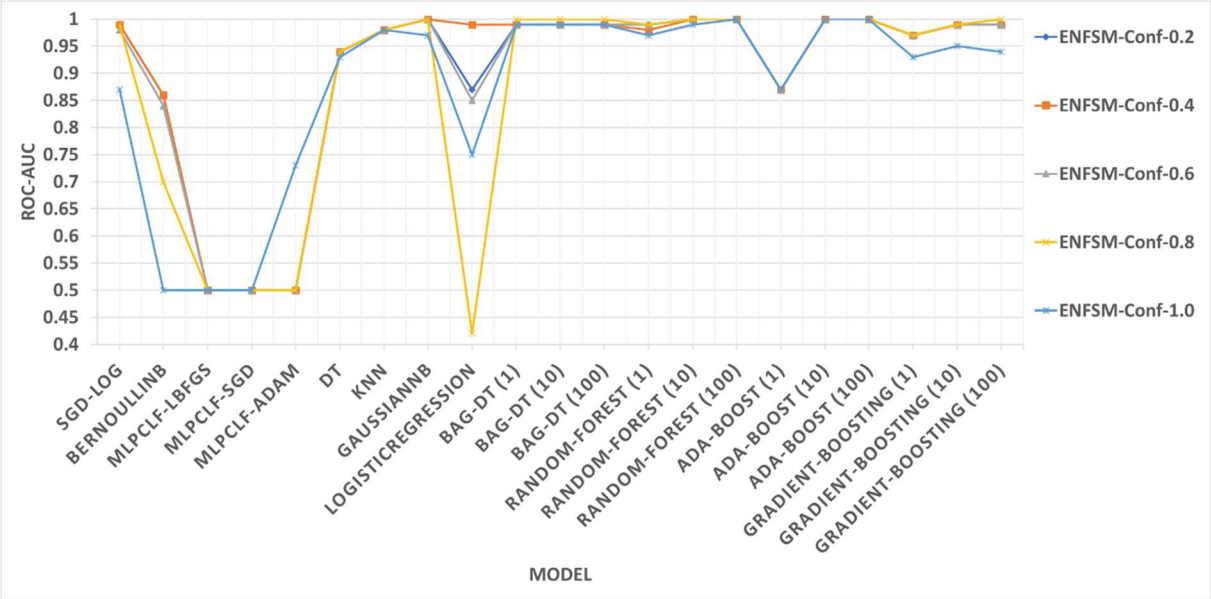


Figure 5-36 ENFSM feature sets ROC-AUC scores using BoTIoT dataset.

5.3.2.2.3 The ENFSM feature set confidence level selection

The proposed ENFSM generated five feature sets with different confidence levels. In practice, only one feature set is required to train the detection model. Thus, the proposed FSM selection method was used to select the best feature set's confidence level generated by the ENFSM for each dataset. Table 5-9 presents the selected feature sets for each dataset. The largest reduction was for the BotNetIoT datasets for 79%. In contrast, the lowest reduction was for UNSW-NB15 by 51%. The highest confidence feature sets were selected for the NSL-KDD, and BoTIoT datasets of 80% confidence, which means four out of five FSMs selected these features.

Table 5-9 Selected confidence feature sets generated by the proposed ENFSM for all datasets.

Dataset	Confidence feature set	Features	# features	Reduction %
NSL	ENFSM-Conf-0.8	same_srv_rate, serror_rate, flag, srv_serror_rate, dst_host_serror_rate, logged_in, dst_host_srv_serror_rate, dst_host_same_srv_rate, count, dst_host_srv_count, service, dst_host_count	12	0.67
UNSW-NB15	ENFSM-Conf-0.4	dur,proto,spkts,dpkts,sinpkt,swin,ct_dst_src_ltm,ct_ftp_cmd,ct_flw_http_mthd,is_sm_ips_ports,sbytes,dbytes,sload,ct_state_ttl,sttl,state,dwin,ct_src_dport_ltm,rate,dttl	20	0.51
BoTNeTIoT	ENFSM-Conf-0.4	MI_dir_L0.1_weight, H_L0.1_weight, HpHp_L0.1_magnitude, HH_jit_L0.1_mean	4	0.79
BoTIoT	ENFSM-Conf-0.8	TnP_PerProto, daddr, TnBPSrcIP, TnBPDstIP, spkts, pkts, TnP_PSrcIP, sbytes, saddr, bytes, TnP_PDstIP, TnP_Per_Dport, dpkts, dbytes, Pkts_P_State_P_Protocol_P_DestIP, Pkts_P_State_P_Protocol_P_SrcIP, stime, ltime, dport	19	0.53

5.3.3 Conclusion

Feature selection has a significant role in reducing data dimensionality. The data volum is reduced by reducing the data dimentionality. Thus, it speeds up the detection step. Moreover, it positively affects the models' performance. The "ID" feature was used as a cutoff to track relevant from non-relevant features. Using ID-cutoff highly improved the efficiency of filter-based FSMs. The filter methods, which used ID-cutoff, showed F and ROC-AUC scores equal to the highest score obtained using the same filters with percentiles. Some of the filters, which used ID-cutoff, could not select any of the features because of the bad quality of the features in the corresponding dataset. However, the proposed ENFSM overcame this issue. The ENFSM F and ROC-AUC scores represented the best scores in most of the experiments. Furthermore, the generated feature set suited a vast range of classifiers.

Additionally, this research highlighted that some models had unstable performance over different feature sets, where their performance was strongly affected by features and the dataset. Some detection models achieved speed improvement without compromising their performance by using smaller feature sets. Nonetheless, ensemble models showed the best scores and stable performance over all the datasets and feature sets. Therefore, ensemble models are convenient candidates for IDS for IoT networks.

5.4 Detection Model

As shown in section 5.3, feature selection improved the IDS performance, and it is also mandatory in practice, as shown in section 2.4. The MSM considered the selected feature sets' confidence levels in Table 5-9 to select the best three models to construct the ensemble model. On the whole, the proposed models were tested on all the used and proposed ENFSM in section

5.3.2.2.2. The aim is to figure out the reliability of the proposed ensemble detection models on various feature sets and meet IoT networks' dynamism.

The Experiments considered eleven different models, in addition to the four ensemble models. Moreover, they considered three different estimator values that were 1, 10, and 100 estimators for the used ensemble models to analyze the impact of estimators' numbers on the efficiency of the ensemble models. A 1 estimator was used to meet the main idea of ensemble learning of combining simpler models, which is not an ensemble model, trained using different methods. Thus in total, there were twenty-three models considered. To give each model's type the same weight in contributing to the ensemble model construction, only different models were used. Thus, the most efficient ensemble model configuration was selected from the three considered configurations based on the proposed efficiency measurements.

The proposed MSM was applied to each ensemble model category to opt for the best model configuration during the ensemble model construction. In total, fifteen different models were used in the MSM to select from to build the proposed ensemble model. An ensemble model for each dataset called Edge-ENClf was constructed. Additionally, the Cloud-ENClf was constructed from all the proposed Edge-ENClf models. The Cloud-ENClf model generalization level is more than the Edge-ENClf model because it was constructed based on information from all the used datasets.

5.4.1 Results

The MSM computed the efficiency scores for each dataset to select the best three models. Table 5-10 introduces the efficiency scores of the models using the UNSW-NB15 dataset. The MSM selected the most recommended models by the efficiency measurements. Two efficiency measurements recommended the LogisticRegression model. Only one efficiency measurement recommended the Random-Forest (1) and DT models. The efficiency measurements did not

recommend any of the other models. Consequently, no further analysis took place, and the MSM selected the LogisticRegression, Random-Forest (1), and DT as the Edge-ENClf model for the UNSW-NB15 dataset.

Table 5-10 Detection models efficiency scores using UNSW-NB15 dataset.

Classifier-Con0.4	F-ScoreEfc	Rank	ROC-AUCEfc	Rank	Explained-varianceEfc	Rank	Rank-Sum
LogisticRegression	**2.814815**	**1**	**3**	**1**	**0.333333**	**4**	**2**
Random-Forest (1)	**2.785714**	**3**	**2.964286**	**2**	**0.535714**	**1**	**1**
DT	**2.814815**	**1**	**2.962963**	**3**	**0.333333**	**4**	**1**
Gradient-Boosting (10)	2.689655	5	2.827586	4	0.482759	3	0
Bag-DT (1)	2.6	7	2.766667	6	0.5	2	0
GaussianNB	2.740741	4	2.777778	5	0	6	0
Ada-Boost (1)	2.642857	6	2.678571	7	0	6	0
Knn	0.04267	11	0.057123	13	0	6	0
BernoulliNB	2.481481	8	2.592593	8	0	6	0
MLPClf-lbfgs	0	14	2.173913	11	0	6	0
MLPClf-adam	0.583333	10	2.25	9	0	6	0
SGD-log	1.678571	9	2.178571	10	0	6	0
MLPClf-sgd	0	14	2.083333	12	0	6	0
SVM-SVC	0.009404	12	0.010199	14	0	6	0
SVM-sigmoidSVC	0.00683	13	0.007238	15	0	6	0

Table 5-11 summarizes the final selection of models for each Edge-ENClf model for each dataset. Furthermore, it shows the Cloud-ENClf model for all the datasets. Interestingly, the MSM selected the DT and Random-Forest (1) models for three datasets. Notably, the NSL-KDD Edge-ENClf model is the same as the Cloud-ENClf model. Interestingly, the MSM selected the simplest ensemble models, which had a single estimator to be part of the final ensemble model.

Table 5-11 Selected Models for all datasets and a centralized ensemble model.

NSL-KDD	UNSW-NB15	BotNetIoT	BoTIoT	Cloud Model
DT	**DT**	**DT**	MLPClf-lbfgs	DT
Random-Forest (1)	LogisticRegression	**Random-Forest (1)**	**Random-Forest (1)**	Random-Forest (1)
Gradient-Boosting (1)	**Gradient-Boosting (1)**	Bag-DT (10)	Gradient-Boosting (10)	Gradient-Boosting (1)

Figure 5-37 to Figure 5-44 provide the F and ROC-AUC scores for the Edge and Cloud ensemble models for all the datasets using all FSMs discussed in section 5.3. Figure 5-37 shows that as the cloud and edge models are the same, it only shows the F scores for the Edge-ENClf model. The F scores range from 0.97 to 0.99. The proposed model has shown the highest scores

using Chi2-0.1 and Knn-BW FSMs. The F scores using ENFSM are 0.98 for all confidence levels, except for the feature set with a confidence of 100%.

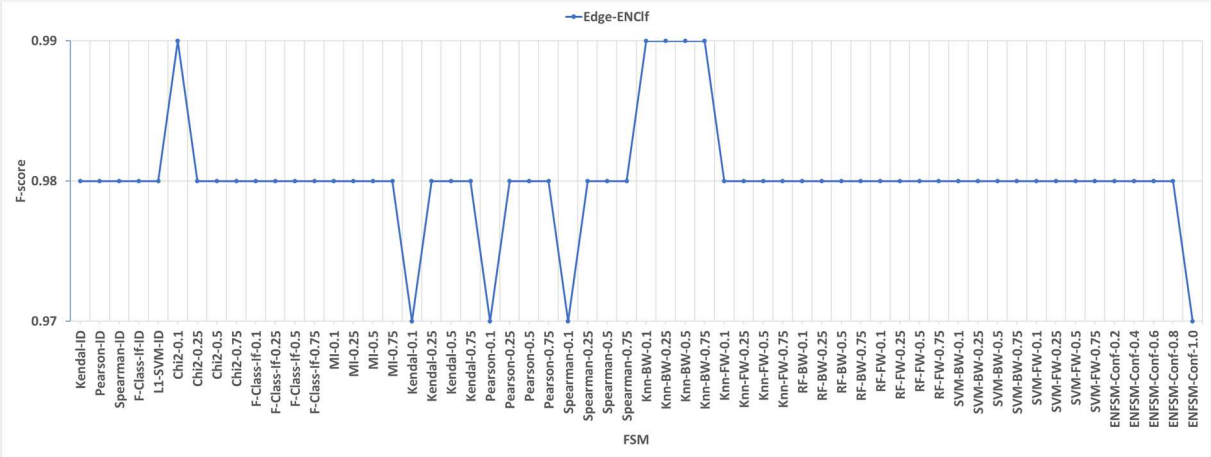


Figure 5-37 EnClf F scores using the NSL-KDD dataset.

Figure 5-38 sketches the ROC-AUC scores using the NSL-KDD dataset. The ROC-AUC scores range from 0.96 to 1. The proposed model generated the lowest scores using Kendal-0.1 and Spearman-0.1 FSMs. The proposed ensemble model generated 1 using all the ENFSMs except for ENFSM-Conf-1.0.

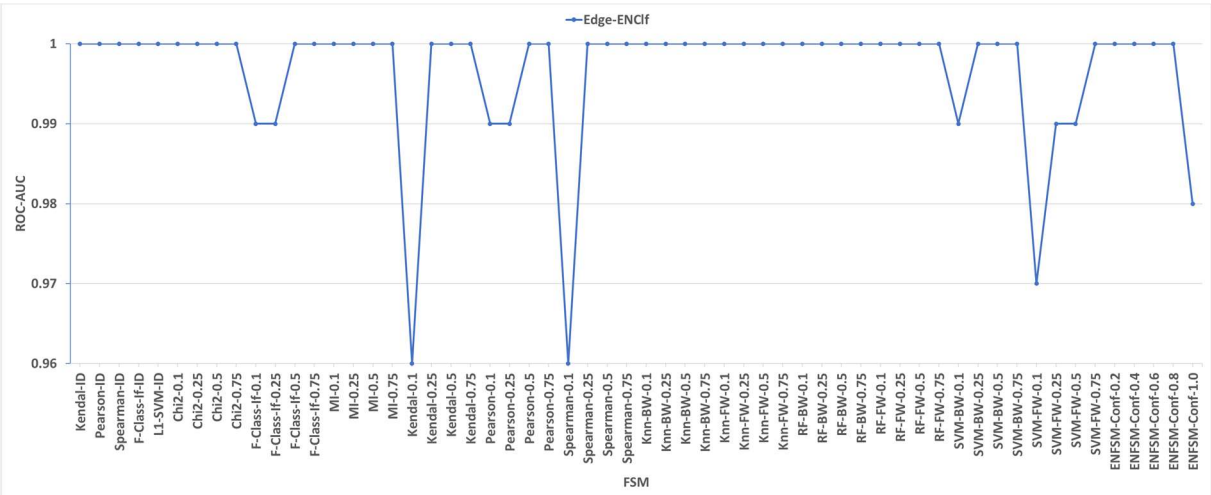


Figure 5-38 EnClf ROC-AUC scores using the NSL-KDD dataset.

Figure 5-39 presents the F scores using the UNSW-NB15 dataset. Interestingly, the Edge and cloud models showed close scores using all FSMs. The Cloud-ENCLF model slightly overcame

the Edge-ENClf model using all FSMs, except for Spearman-ID, Kendal-0.1, Spearman-0.25, and Knn-FW-0.25. The F score ranges from 0.75 to 0.95. The Cloud-ENClf mode generates the highest scores using Chi2-0.75, If-Class-If-0.75, Pearson-0.75, RF-BW-0.5, RF-BW-0.75, SVM-FW-0.75, and ENFSM-Conf-0.2. The models showed higher scores when using large feature sets than using small feature sets.

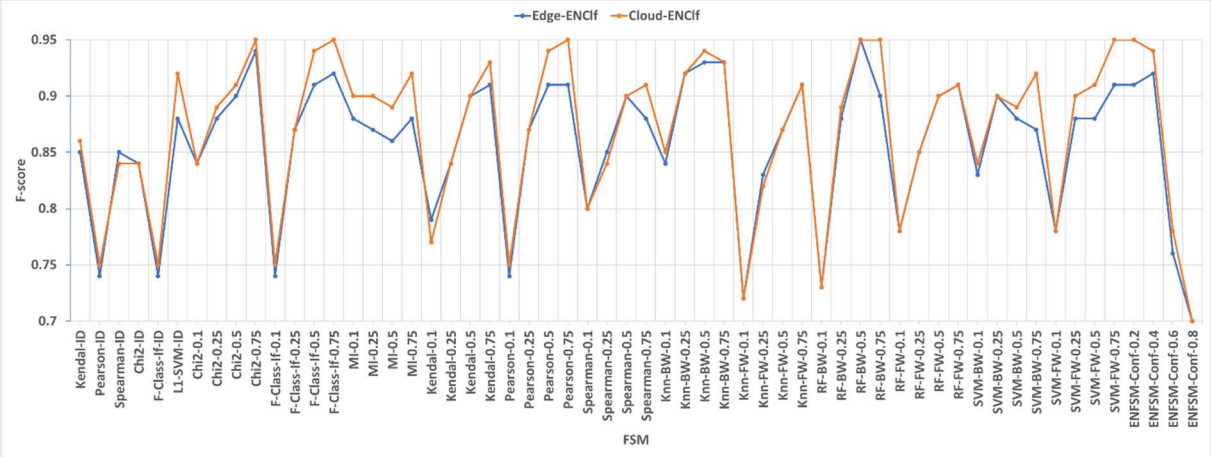


Figure 5-39 EnClf F scores using the UNSW-NB15 dataset.

In terms of ROC-AUC, as shown in Figure 5-40, the Cloud-ENClf model overcame the Edge-ENClf model using all FSM, except when using Kendal-ID. Both models generate the lowest score using ENFSM-0.8. Notably, the Cloud-ENClf model has the highest score using ENFSM-0.2 and ENFSM-0.4, while the Edge-ENClf model has the same score using ENFSM-0.2. Mostly, the ROC-AUC scores range between 0.90 and 0.98, except for using five of the FSMs.

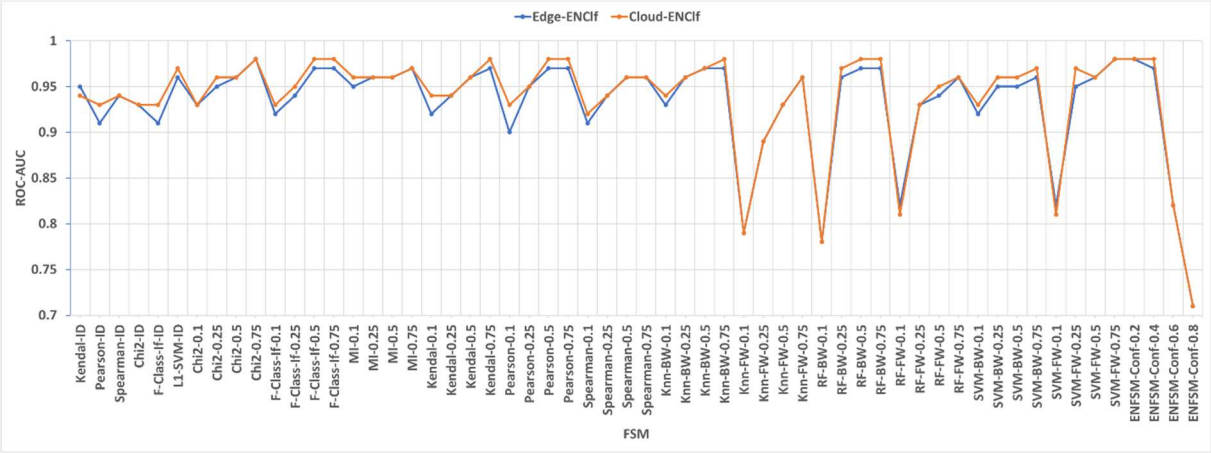


Figure 5-40 EnClf ROC-AUC scores using the UNSW-NB15 dataset.

Figure 5-41 shows the F scores using the BotNetIoT dataset. Interestingly, the Edge-ENFClf model overcomes the Cloud-ENClf model using Kendal-ID, Chi2-ID, L1-SVM-ID, MI-ID, Chi2-0.1, Chi2-0.25, Kendal-0.1, and Spearman-0.1. In this representation, the Edge-ENClf model shows higher scores than the Cloud-ENClf model using smaller feature sets. Almost both models have a score of 1 using all FSMs.

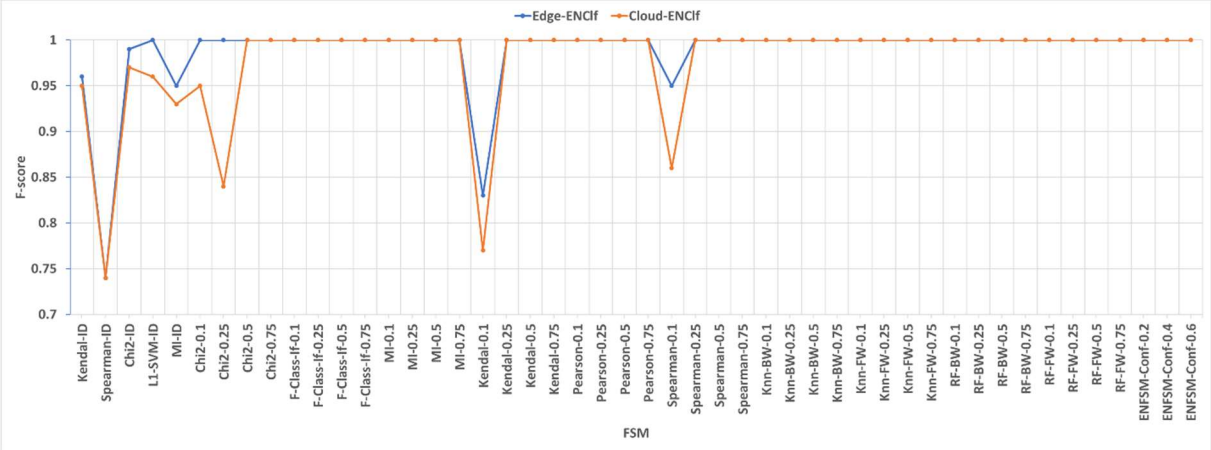


Figure 5-41 EnClf F scores using the BotNetIoT dataset.

Figure 5-42 shows the ROC-AUC score. Both models had the same scores, except for using Chi2-ID. The Edge-ENClf overcomes the Cloud-ENClf model. Nearly a score of 1 is achieved using all FSMs.

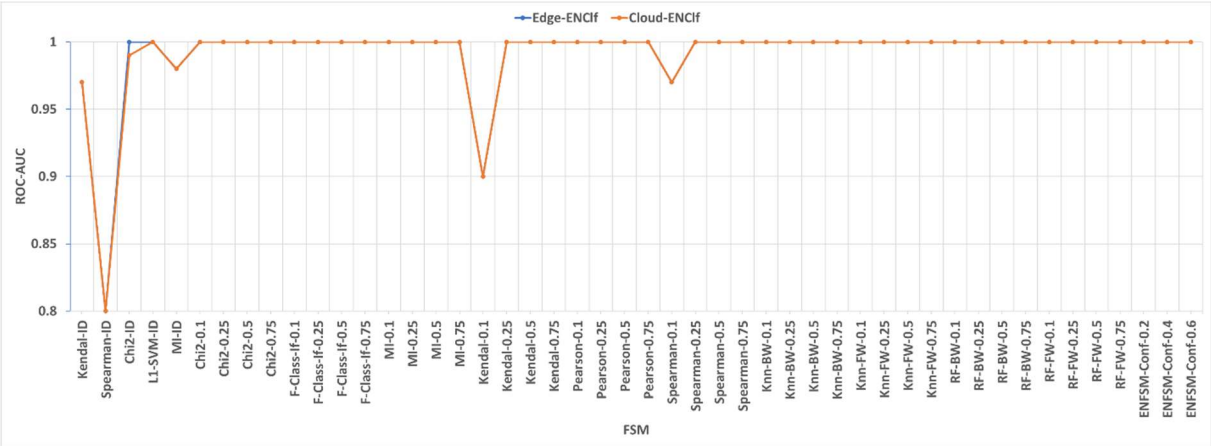


Figure 5-42 EnClf ROC-AUC scores using the BotNetIoT dataset.

Figure 5-43 shows the F scores using the BoTIoT dataset. Interestingly, the Edge-ENClf model slightly overcomes the Cloud-ENClf model using most of the FSMs. However, the Edge-

ENClf model has the lowest score. Generally, the scores range between 0.90 and 0.99. The cloud model shows a more stable performance over all FSMs.

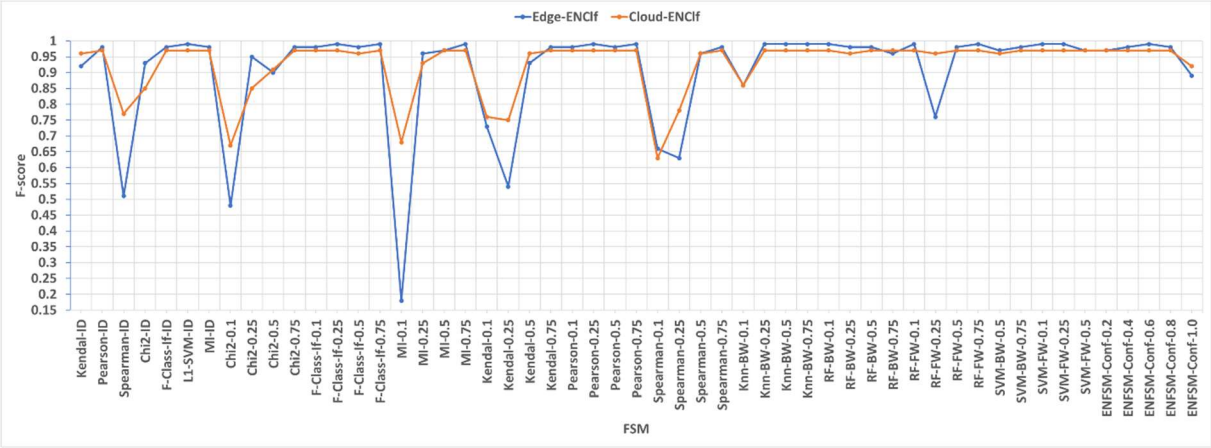


Figure 5-43 EnClf F scores using the BoTIoT dataset.

Figure 5-44 shows the ROC-AUC using the BoTIoT dataset. Still, the Edge-ENClf model overcomes the Cloud-ENClf model using most of the FSMs. Significantly, the Cloud-ENClf model has the lowest score using Chi0.1. The scores range from 0.96 to 1.

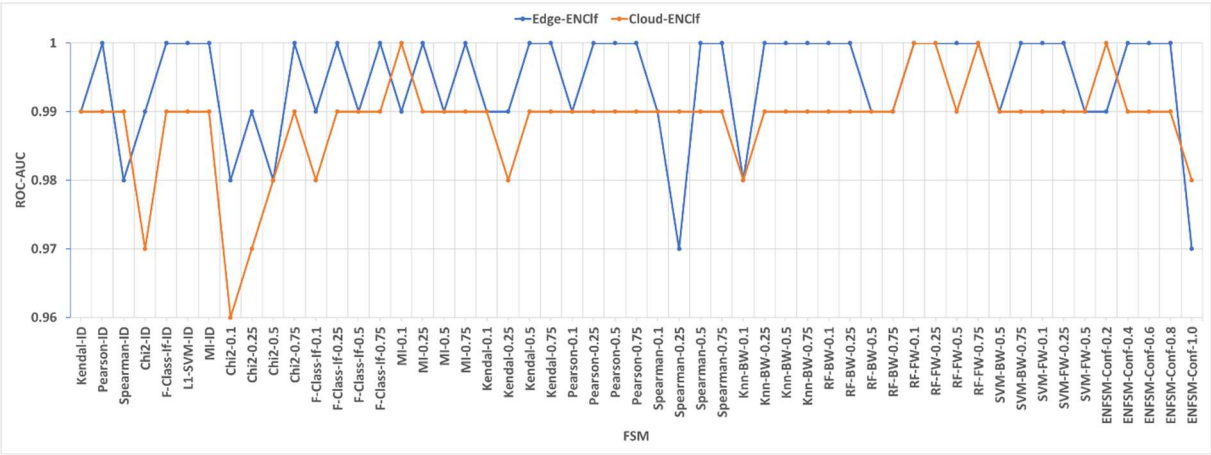


Figure 5-44 EnClf ROC-AUC scores using the BoTIoT dataset.

The proposed ensemble models showed comparable performance to each other because of sharing some models. The Edge-ENClf using BoTIoT dataset showed a better and more different performance than the Cloud-ENClf model because it had two models different from the Cloud-ENClf model.

Table 5-12 summarizes the proposed models' ranks when compared with all the considered models in this research using the efficiency measurements. The proposed models achieved the ranks in Table 5-12 using the chosen ENFSM feature sets in Chapter 5. The Cloud and Edge models were the first and the second models, respectively, using UNSW-NB15 and BoTIoT datasets. Both were the best-recommended models using the NSL-KDD dataset. The worst ranking was using the BotNetIoT dataset. It is noticeable that the proposed ensemble models overcame all the classifiers using NSL-KDD, UNSW-NB15, and BoTIoT datasets in terms of efficiency. Significantly, the two proposed models can work in different deployment levels without compromising the detection efficiency. Moreover, they showed a stable performance, which enhanced their reliability when they are in practice.

Table 5-12 Overall proposed ensemble classifiers ranks.

Dataset	Model	Rank
NSL-KDD	Edge-ENClf	1st
	Cloud-ENClf	1st
UNSW-NB15	Edge-ENClf	2nd
	Cloud-ENClf	1st
BotNetIoT	Edge-ENClf	7th
	Cloud-ENClf	3th
BoTIoT	Edge-ENClf	2nd
	Cloud-ENClf	1st

5.4.2 Conclusion

IoT network is currently witnessing a remarkable expansion, which can also be detected in its traffic data. A reliable and efficient IDS is, therefore, mandatory to secure it. The efficiency is measured in terms of detection speed and accuracy. In this research, twenty-three models were evaluated using fifty-five FSMs on four datasets. Three novel efficiency measurements were proposed. Furthermore, a new MSM using the efficiency measurements to select and build efficient ensemble detection models was proposed. The models can be deployed on different IoT network infrastructure levels without compromising the detection efficiency.

Ensemble learning has a significant role in learning reliable models. It increases the confidence in the model decision. Moreover, it might positively affect the models' efficiency. The F, ROC-AUC, and variance efficiency scores were used to select models. The ensemble models achieved high-efficiency scores over all the datasets.

Additionally, this research highlighted that ensemble models have stable performance over different feature sets, where their performance was slightly affected by features and the dataset. The ensemble models showed the best efficiency scores, and thus some were selected to construct a more stable and efficient ensemble model. Therefore, ensemble models can be combined with the right configuration.

5.5 Comparison Study

Table 5-13 shows the proposed models' performance metrics using the proposed ENFSM for all the used datasets. The PR-AUC is used because it works better than ROC-AUC in imbalanced datasets. The PR-AUC focuses on the minority class. The ROC-AUC, which covers both classes, is used because, in IDS evaluation, the negative and positive classes have the same importance. Noticeably, the proposed models' performance is accurate for both classes.

Table 5-13 The proposed ensemble models results for each dataset using the proposed ENFSM.

Dataset, ENFSM	Model	Performance metrics			
		Accuracy (%)	F-Score	ROC-AUC	PR-AUC
NSL-KDD, ENFSM-Conf0.8	Edge-ENClf	98(+/- 0.005)	0.9(+/- 0.005)	1(+/- 0.000)	1(+/- 0.000)
	Cloud-ENClf	98(+/- 0.005)	0.98(+/- 0.005)	1(+/- 0.000)	1(+/- 0.000)
UNSW-NB15, ENFSM-Conf0.4	Edge-ENClf	95(+/- 0.005)	0.94(+/- 0.005)	0.98(+/- 0.005)	0.91(+/- 0.005)
	Cloud-ENClf	93(+/- 0.000)	0.92(+/- 0.000)	0.97(+/- 0.000)	0.97(+/- 0.000)
BotNetIoT, ENFSM-Conf0.4	Edge-ENClf	100(+/- 0.000)	1(+/- 0.000)	1(+/- 0.000)	1(+/- 0.000)
	Cloud-ENClf	100(+/- 0.000)	1(+/- 0.000)	1(+/- 0.000)	1(+/- 0.000)
BoTIoT, ENFSM-Conf0.8	Edge-ENClf	100(+/- 0.000)	0.97(+/- 0.01)	0.99(+/- 0.000)	1(+/- 0.005)
	Cloud-ENClf	100(+/- 0.000)	1(+/- 0.01)	1(+/- 0.005)	0.99(+/- 0.005)

Table 5-14 Shows that the AL-Hawawreh [51], Zhou [75], Abeshu [58], and Aloqaily [78] slightly overcome the proposed model using the NSL-KDD dataset in terms of accuracy. On the

other hand, the proposed models overcome Pham [67], Miller [54], Feng [60], and TAMA [68] models.

Table 5-14 Comparison of proposed ensemble models with the relevant literature using the NSL-KDD dataset.

IDS		Performance metrics		
		Accuracy (%)	F-Score	ROC-AUC
Cloud-ENClf		98	0.98	1
AL-Hawawreh		**98.6**	NA	NA
Zhou [75]	gradient-boosted trees	**98.54**	NA	NA
	Knn	**98.82**		
	DT	**98.77**		
	logistic regression	**98.85**		
	gaussianNB	**98.8**		
	SVM	**98.86**		
Pham [67]		84.25	NA	NA
Miller [54]		84.1	NA	NA
Abeshu [58]		**99.2**	NA	NA
Feng [60]		95.25	NA	NA
TAMA [68]		85.7	NA	NA
Aloqaily [78]		**99.43**	NA	NA

Table 5-15 demonstrates that the proposed models overcome all the relevant literature references using the UNSW-NB15 dataset in terms of accuracy, except Moustafa [46]. Significantly, the proposed model overcomes AL-Hawawreh [51] and Zhou [75] models.

Table 5-15 Comparison of proposed ensemble models with the relevant literature using the UNSW-NB15 dataset.

IDS		Performance metrics		
		Accuracy (%)	F-Score	ROC-AUC
Cloud-ENClf		95	0.94	0.98
Edge-ENClf		93	0.92	0.97
Moustafa [46]		**99.2**	NA	NA
AL-Hawawreh [51]		92.4	NA	NA
Zhou [75]	gradient-boosted trees	91.22	NA	NA
	Knn	91.9		
	DT	92.29		
	logistic regression	90.35		
	gaussianNB	92.52		
	SVM	92.32		
TAMA [68]		72.52	NA	NA

Table 5-16 reveals that the proposed models significantly overcome Kitsune [22] in terms of ROC-AUC using the BotNetIoT dataset. Furthermore, it overcomes TempoCode-IoT [79] model in terms of F score. The proposed models achieved 100% for all performance metrics. However, some models in this research overcome the proposed models in terms of efficiency measurements. That is why the proposed models were placed in the 3rd and the 7th ranks.

Table 5-16 Comparison of proposed ensemble models with the relevant literature using the BotNetIoT dataset.

IDS	Performance metrics		
	Accuracy (%)	**F-Score**	**ROC-AUC**
Cloud-ENClf	100	**1**	**1**
Edge-ENClf	100	**1**	**1**
Kitsune [22]	NA	NA	0.7
TempoCode-IoT [79]	NA	0.99	NA

Table 5-17 presents the performance metrics of the proposed model using the BoTIoT dataset. This dataset has been recently published. That is to say, it is not referenced in the relevant literature yet. Both models achieved 100% accuracy. The Edge-ENClf slightly overcomes the Cloud-ENClf model on the other metrics with a slight difference.

Table 5-17 Proposed ensemble models' performance results using the BoTIoT dataset.

IDS	Performance metrics		
	Accuracy (%)	**F-Score**	**ROC-AUC**
Cloud-ENClf	100	0.97	0.99
Edge-ENClf	100	1	1

6 Conclusion and Future Works

6.1 Introduction

This thesis made several contributions to the IoT IDS field, in particular, hybrid IDS. Most of the existing IDS methodologies were designed without addressing the explosion of data and FS's primary need. Consequently, some studies considered FS. But they only examined one method without justification or used a complex FS without observing the field's time requirements. Furthermore, an IDS performance highly depends on the detection model. A few experiments justified why a model was chosen, but the response time of these models was not addressed thoroughly in previous studies.

This research study's contributions address the challenges above to a considerable degree, with some of its limitations offering directions for future research, as discussed in sections 6.3.

Firstly, chapter four analyzed four commonly used datasets' features quality to supply a comprehensive view of network traffic. A single correlation measure was used, the Pearson correlation measure, to highlight the features' quality and the redundancy in each of the datasets. This correlation displayed many potentials in reducing the datasets' dimensionalities. Furthermore, different types of classifiers were evaluated to enhance the malicious traffic prediction. This evaluation also aimed to distinguish this traffic from a benign one to reduce the examined data from 115 to 23 features dimensionally. This goal was achieved by finding the optimum time-window. The results were compared with the benchmarked reference that provided BotNetIoT dataset, Kitsune [22]. To ensure unbiased findings, various performance metrics were evaluated that considered different prediction accuracy tradeoff systems.

Secondly, for the first time in the feature selection field, a novel automatic ensemble evaluation methodology for feature selection methods was proposed. The automatic evaluation

and selection methodology used several and variant FSM evaluation measures to highlight the features' quality and the redundancy in each of the datasets. Moreover, a novel cutoff threshold was proposed for filter-based FSMs using the "ID" feature score. Proposing an ENFSM, which generated feature sets based on five different confidence levels, helped adjust the most reliable and optimal feature set. Furthermore, a vast range of FSMs was evaluated, which allowed these research results to be comparable with other relevant references in the literature.

Finally, for the first time in the ensemble models field, a novel model selection method was proposed to construct ensemble detection models. Two ensemble detection models were constructed—one for each dataset, representing an edge model, and the other for cloud detection constructed for all datasets. The proposed method depends on three novel efficiency measurements. A vast variant range of classifiers and FSMs were used to evaluate the research's results significance.

The rest of this chapter is organized as follows. Section 6.2 elaborates on the key contributions of this research. Section 6.3 provides a discussion of future research directions.

6.2 Key Contributions

The key contributions of this research consist in:

- **The analysis of four network datasets was conducted** using traffic statistics, visual plotting, and correlation measuring. This analysis highlighted the features' quality and the redundancy in each of the datasets. It equally displayed many potentials in reducing the datasets' dimensionalities.
- **The design of a novel ensemble methodology to automatically select feature selection methods**. This methodology combined five measurements to select FSMs. The five measures cover the speed, reduction ratio, information gain, density, and

accuracy. These measurements are reduction efficiency, feature set's information gain, feature set variance, F-score ratio, and ROC-AUC ratio.

- **The cutoff value for filter-based**. The ID feature's score was used as a cutoff. It improved the efficiency of the filter-based FSMs. Also, it satisfied the dynamicity of IoT network traffic.
- **The reduction efficiency measurement** is an integrated measure that combines the time and the reduction percentage to overcomes the corresponding tradeoff matter.
- **The whole feature set information gain measurement**. The PCA was used to convert the feature set into a single dimension. Consequently, the IG of that dimension could be found. It measures the whole feature set uncertainty.
- **Whole feature set information gain measurement**. The PCA was used to convert the feature set into a single dimension. In this way, the variance of that dimension can be calculated.
- **F and ROC-AUC ratios score**. It stands for the percentage of the models that achieved a corresponding score higher than or equal to 0.95%.
- **Ensemble feature selection methods and confidence levels**. An ensemble feature selection method was proposed for each dataset. Each ENFSM generated several feature sets based on confidence. The best feature sets, based on efficiency and reliability, were chosen for each dataset.
- **The model selection method.** Used three efficiency measurements to evaluate each model in order to increase the MSM confidence and suitability for working with

different model types. The final decision was made by selecting the most recommended three models by all measures.

- **Efficiency measurements** were generated using three performance measurements divided by the model scoring time. These integrated efficiency measurements helped in solving tradeoff issues.

- **Edge and Cloud ensemble models**. To answer the choice of IDS deployment matter, two ensemble detection models were proposed. The edge model was constructed using the MSM. This model was trained using one dataset and aimed at overseeing a single homogeneous IoT network. It would be operated at the edge of the network. The cloud model is constructed by combining the shared models constructed by the MSM. This model is a centralized model to operate on a cloud and to oversee several heterogeneous IoT networks.

6.3 Future Works

While this research provides significant advances in IoT IDSs, the following are suggested directions for future research.

The proposed ENClf models and ENFSMs were tested on session-based datasets. It will be interesting to test the proposed methods and models on packet-based datasets.

This research considered cyber-threats as a binary classification problem to detect zero-day attacks. Therefore, it is interesting to know how the proposed models would perform if they are considered a signature-based IDS or the cyber-threat is a multi-classification problem.

This research addressed the experts-mixture ensemble learning technique. A study on using the different ways of ensemble learning and their impact will be beneficial for the literature. Moreover, it may lead to new efficient ways of ensembling.

Feature extraction plays a critical role in the accuracy and efficiency of an IDS. Based on the dataset analysis, detection models' performance varied based on the features' quality and their correlation with the target class. Different datasets have different features without any justification regarding why these features are selected to be extracted. Thus, it will be extremely useful to develop fast and reliable feature extraction methods.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015, doi: 10.1109/COMST.2015.2444095.

[2] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab, "Cybercrime: The Case of Obfuscated Malware," in *Global Security, Safety and Sustainability & e-Democracy*, Berlin, Heidelberg, 2012, pp. 204–211, doi: 10.1007/978-3-642-33448-1_28.

[3] T. Mohamed, T. Otsuka, and T. Ito, "Towards Machine Learning Based IoT Intrusion Detection Service," *Recent Trends and Future Technology in Applied Intelligence. IEA/AIE 2018. Lecture Notes in Computer Science*, vol. 10868, May 2018, doi: https://doi.org/10.1007/978-3-319-92058-0_56.

[4] E. Elbasi, "Reliable abnormal event detection from IoT surveillance systems," in *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, France, Dec. 2020, pp. 1–5, doi: 10.1109/IOTSMS52051.2020.9340162.

[5] A. Futter, "*War Games* redux? Cyberthreats, US–Russian strategic stability, and new challenges for nuclear security and arms control," *European Security*, vol. 25, no. 2, pp. 163–180, Apr. 2016, doi: 10.1080/09662839.2015.1112276.

[6] S. Prabavathy, K. Sundarakantham, and S. M. Shalinie, "Design of cognitive fog computing for intrusion detection in Internet of Things," *J. Commun. Netw.*, vol. 20, no. 3, pp. 291–298, Jun. 2018, doi: 10.1109/JCN.2018.000041.

[7] B. F. L. M. Sousa, Z. Abdelouahab, D. C. P. Lopes, N. C. Soeiro, and W. F. Ribeiro, "An intrusion detection system for denial of service attack detection in internet of things," in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing - ICC '17*, Cambridge, United Kingdom, Mar. 2017, pp. 1–8, doi: 10.1145/3018896.3018962.

[8] S. Chawla and G. Thamilarasu, "Security as a service: real-time intrusion detection in internet of things," in *Proceedings of the Fifth Cybersecurity Symposium on - CyberSec '18*, Coeur d' Alene, Idaho, Apr. 2018, pp. 1–4, doi: 10.1145/3212687.3212872.

[9] L. Atzori, A. Lera, and G. Morabito, "The Internet of Things: A survey | Elsevier Enhanced Reader," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010, doi: 10.1016/j.comnet.2010.05.010.

[10] K. J. Kaur and A. Hahn, "Exploring ensemble classifiers for detecting attacks in the smart grids," in *Proceedings of the Fifth Cybersecurity Symposium*, Coeur d' Alene Idaho, Apr. 2018, pp. 1–4, doi: 10.1145/3212687.3212873.

[11] S. Shen, L. Huang, H. Zhou, S. Yu, E. Fan, and Q. Cao, "Multistage Signaling Game-Based Optimal Detection Strategies for Suppressing Malware Diffusion in Fog-Cloud-Based IoT Networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1043–1054, Apr. 2018, doi: 10.1109/JIOT.2018.2795549.

[12] K. Lueth, “IoT 2019 in Review: The 10 Most Relevant IoT Developments of the Year.” https://iot-analytics.com/iot-2019-in-review/ (accessed May 27, 2020).

[13] K. Lueth, “State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating.” https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/ (accessed May 27, 2020).

[14] “A Perfect Storm: the Security Challenges of Coronavirus Threats and Mass Remote Working,” *Check Point Software*, Apr. 07, 2020. https://blog.checkpoint.com/2020/04/07/a-perfect-storm-the-security-challenges-of-coronavirus-threats-and-mass-remote-working/ (accessed May 27, 2020).

[15] M. Aldwairi, W. Mardini, and A. Alhowaide, “Anomaly Payload Signature Generation System Based on Efficient Tokenization Methodology,” *International Journal on Communications Antenna and Propagation (IRECAP) (2018)*, Nov. 2018.

[16] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges,” in *2012 10th International Conference on Frontiers of Information Technology*, Dec. 2012, pp. 257–260, doi: 10.1109/FIT.2012.53.

[17] Zhihong Yang, Yingzhao Yue, Yu Yang, Yufeng Peng, Xiaobo Wang, and Wenji Liu, “Study and application on the architecture and key technologies for IOT,” in *2011 International Conference on Multimedia Technology*, Jul. 2011, pp. 747–751, doi: 10.1109/ICMT.2011.6002149.

[18] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du, “Research on the architecture of Internet of Things,” in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, Aug. 2010, vol. 5, pp. V5-484-V5-487, doi: 10.1109/ICACTE.2010.5579493.

[19] M. A. Chaqfeh and N. Mohamed, “Challenges in middleware solutions for the internet of things,” in *2012 International Conference on Collaboration Technologies and Systems (CTS)*, May 2012, pp. 21–26, doi: 10.1109/CTS.2012.6261022.

[20] Lu Tan and Neng Wang, “Future internet: The Internet of Things,” in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, Aug. 2010, vol. 5, pp. V5-376-V5-380, doi: 10.1109/ICACTE.2010.5579543.

[21] M. Miettinen and A. Sadeghi, “Keynote: Internet of Things or Threats? On Building Trust in IoT,” in *2018 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Sep. 2018, pp. 1–9, doi: 10.1109/CODESISSS.2018.8525931.

[22] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection,” *arXiv:1802.09089 [cs]*, Feb. 2018, Accessed: Oct. 24, 2019. [Online]. Available: http://arxiv.org/abs/1802.09089.

[23] A. Shameli-Sendi, M. Cheriet, and A. Hamou-Lhaj, “Taxonomy of intrusion risk assessment and response system | Elsevier Enhanced Reader,” *Computers & Security*, vol. 45, pp. 1–16, Sep. 2014, doi: 10.1016/j.cose.2014.04.009.

[24] I. Alsmadi, R. Burdwell, A. Aleroud, A. Wahbeh, M. Qudah, and A. Al-Omari, *Practical Information Security: A Competency-Based Education Course*. Springer International Publishing, 2018.

[25] N. Moustafa, J. Hu, and J. Slay, "A holistic review of Network Anomaly Detection Systems: A comprehensive survey | Elsevier Enhanced Reader," *Journal of Network and Computer Applications*, vol. 128, pp. 33–55, Feb. 2019, doi: 10.1016/j.jnca.2018.12.006.

[26] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building Auto-Encoder Intrusion Detection System based on random forest feature selection," *Computers & Security*, vol. 95, p. 101851, Aug. 2020, doi: 10.1016/j.cose.2020.101851.

[27] M. E. Whitman and H. J. Mattord, *Principles of Information Security*. Cengage Learning EMEA, 2009.

[28] "1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory." http://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset (accessed Nov. 20, 2019).

[29] "NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB." https://www.unb.ca/cic/datasets/nsl.html (accessed Nov. 20, 2019).

[30] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Nov. 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[31] "UCI Machine Learning Repository: detection_of_IoT_botnet_attacks_N_BaIoT Data Set." https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT (accessed Nov. 27, 2019).

[32] I.-L. Yen, F. Bastani, N. Solanki, Y. Huang, and H. San-Yih, "Trustworthy Computing in the Dynamic IoT Cloud," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, Jul. 2018, pp. 411–418, doi: 10.1109/IRI.2018.00067.

[33] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *Proc. ACM MCC*, pp. 13–6, 2012.

[34] D. Welch and S. Lathrop, "Wireless security threat taxonomy," in *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003.*, Jun. 2003, pp. 76–83, doi: 10.1109/SMCSIA.2003.1232404.

[35] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT)," in *Recent Trends in Network Security and Applications*, Berlin, Heidelberg, 2010, vol. 89, pp. 420–429, doi: 10.1007/978-3-642-14478-3_42.

[36] K. Kendall, *A database of computer attacks for the evaluation of intrusion detection systems*. Ph.D. Dissertation, 1999.

[37] H. Hindy *et al.*, "A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets," *arXiv:1806.03517 [cs]*, Jun. 2018, Accessed: Nov. 12, 2019. [Online]. Available: http://arxiv.org/abs/1806.03517.

[38] I. Alsmadi and F. Mira, "IoT security threats analysis based on components, layers and devices," *http://ajse.us/*, vol. 1, no. 1, pp. 1–10, 2019.

[39] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in *Proceedings of the 2003 ACM workshop on Rapid Malcode - WORM'03*, Washington, DC, USA, 2003, p. 11, doi: 10.1145/948187.948190.

[40] R. Polikar, "Ensemble learning," *Scholarpedia*, vol. 4, no. 1, p. 2776, Jan. 2009, doi: 10.4249/scholarpedia.2776.

[41] C. Zhang and Y. Ma, Eds., *Ensemble Machine Learning: Methods and Applications*. New York: Springer-Verlag, 2012.

[42] S. Raschka, *Python Machine Learning - Second Edition*. Packt Publishing, 2017.

[43] "sklearn.ensemble.VotingClassifier — scikit-learn 0.23.2 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html (accessed Nov. 19, 2020).

[44] A. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning," *Artificial Intelligence,* vol. 97, pp. 245–271, 1997, doi: https://doi.org/10.1016/s0004-3702(97)00063-5.

[45] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, vol. 454. Springer Science & Business Media, 2012.

[46] N. Moustafa, 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*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019, doi: 10.1109/JIOT.2018.2871719.

[47] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, First 2014, doi: 10.1109/SURV.2013.052213.00046.

[48] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer," *Expert Systems with Applications*, vol. 148, p. 113249, Jun. 2020, doi: 10.1016/j.eswa.2020.113249.

[49] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo, "Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System," *International Conference on Information Security and Cryptology, Springer*, vol. 4318, pp. 153–167, 2006, doi: https://doi.org/10.1007/11937807_13.

[50] C. Okoli and K. Schabram, "A Guide to Conducting a Systematic Literature Review of Information Systems Research," *SSRN Journal*, 2010, doi: 10.2139/ssrn.1954824.

[51] M. AL-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, Aug. 2018, doi: https://doi.org///doi.org/10.1016/j.jisa.2018.05.002 ID: 287016.

[52] A. Verma and V. Ranga, "ELNIDS: Ensemble Learning based Network Intrusion Detection System for RPL based Internet of Things," in *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Apr. 2019, pp. 1–6, doi: 10.1109/IoT-SIU.2019.8777504.

[53] M. A. Jabbar, R. Aluvalu, and S. S. S. Reddy, "Cluster Based Ensemble Classification for Intrusion Detection System," in *Proceedings of the 9th International Conference on Machine Learning and Computing - ICMLC 2017*, Singapore, Singapore, 2017, pp. 253–257, doi: 10.1145/3055635.3056595.

[54] S. T. Miller and C. Busby-Earle, "Multi-Perspective Machine Learning a Classifier Ensemble Method for Intrusion Detection," in *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing - ICMLSC '17*, Ho Chi Minh City, Vietnam, 2017, pp. 7–12, doi: 10.1145/3036290.3036303.

[55] V. V. Kumari and P. R. K. Varma, "A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Feb. 2017, pp. 481–485, doi: 10.1109/I-SMAC.2017.8058397.

[56] M. Rebbah, D. E. H. Rebbah, and O. Smail, "Intrusion detection in Cloud Internet of Things environment," in *2017 International Conference on Mathematics and Information Technology (ICMIT)*, Dec. 2017, pp. 65–70, doi: 10.1109/MATHIT.2017.8259697.

[57] R. Fu, K. Zheng, D. Zhang, and Y. Yang, "An intrusion detection scheme based on anomaly mining in internet of things," in *4th IET International Conference on Wireless, Mobile Multimedia Networks (ICWMMN 2011)*, Nov. 2011, pp. 315–320, doi: 10.1049/cp.2011.1014.

[58] A. Abeshu and N. Chilamkurti, "Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-Things Computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, Feb. 2018, doi: 10.1109/MCOM.2018.1700332.

[59] L. F. Maimó, Á. L. P. Gómez, F. J. G. Clemente, M. G. Pérez, and G. M. Pérez, "A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks," *IEEE Access*, vol. 6, pp. 7700–7712, Feb. 2018, doi: 10.1109/ACCESS.2018.2803446.

[60] F. Qu, J. Zhang, Z. Shao, and S. Qi, "An Intrusion Detection Model Based on Deep Belief Network," in *Proceedings of the 2017 VI International Conference on Network, Communication and Computing - ICNCC 2017*, Kunming, China, Dec. 2017, pp. 97–101, doi: 10.1145/3171592.3171598.

[61] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, Sep. 2017, doi: 10.1109/ACCESS.2017.2747560.

[62] A. Azmoodeh, A. Dehghantanha, and K. R. Choo, "Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, Jan. 2019, doi: 10.1109/TSUSC.2018.2809665.

[63] Y. Meidan *et al.*, "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, Oct. 2018, doi: 10.1109/MPRV.2018.03367731.

[64] P. Illavarason and B. Kamachi Sundaram, "A Study of Intrusion Detection System using Machine Learning Classification Algorithm based on different feature selection approach," in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Dec. 2019, pp. 295–299, doi: 10.1109/I-SMAC47947.2019.9032499.

[65] W. Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea, and N. Bouguila, "Network Anomaly Intrusion Detection Using a Nonparametric Bayesian Approach and Feature Selection," *IEEE Access*, vol. 7, pp. 52181–52190, Apr. 2019, doi: 10.1109/ACCESS.2019.2912115.

[66] Mukherjee, Saurabh and Neelam Sharma, "Intrusion Detection using Naive Bayes Classifier with Feature Reduction | Elsevier Enhanced Reader," *Procedia Technology*, vol. 4, pp. 119–128, Jan. 2012, doi: 10.1016/j.protcy.2012.05.017.

[67] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and 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 on - ACSW '18*, Brisband, Queensland, Australia, Jan. 2018, pp. 1–6, doi: 10.1145/3167918.3167951.

[68] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: A Two-Stage Classifier Ensemble for Intelligent Anomaly-Based Intrusion Detection System," *IEEE Access*, vol. 7, pp. 94497–94507, Jul. 2019, doi: 10.1109/ACCESS.2019.2928048.

[69] A. Guerra-Manzanares, H. Bahsi, and S. Nõmm, "Hybrid Feature Selection Models for Machine Learning Based Botnet Detection in IoT Networks," in *2019 International Conference on Cyberworlds (CW)*, Oct. 2019, pp. 324–327, doi: 10.1109/CW.2019.00059.

[70] H. T. Nguyen, K. Franke, and S. Petrović, "A new ensemble-feature-selection framework for intrusion detection," in *2011 11th International Conference on Intelligent Systems Design and Applications*, Nov. 2011, pp. 213–218, doi: 10.1109/ISDA.2011.6121657.

[71] C. Constantinopoulos, M. K. Titsias, and A. Likas, "Bayesian feature and model selection for Gaussian mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 1013–1018, Jun. 2006, doi: 10.1109/TPAMI.2006.111.

[72] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[73] Q. J. Ross, "Simplifying decision trees," *International Journal of Human-Computer Studies*, vol. 51, no. 2, pp. 497–510, Aug. 1999, doi: https://doi.org/10.1006/ijhc.1987.0321.

[74] L. Deng, D. Li, X. Yao, D. Cox, and H. Wang, "Mobile network intrusion detection for IoT system based on transfer learning algorithm," *Cluster Comput*, vol. 22, no. 4, pp. 9889–9904, Jul. 2019, doi: 10.1007/s10586-018-1847-2.

[75] Y. Zhou, M. Han, L. Liu, J. S. He, and Y. Wang, "Deep learning approach for cyberattack detection," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 262–267, doi: 10.1109/INFCOMW.2018.8407032.

[76] M. Milliken, Y. Bi, L. Galway, and G. Hawe, "Ensemble learning utilising feature pairings for intrusion detection," in *2015 World Congress on Internet Security (WorldCIS)*, Oct. 2015, pp. 24–31, doi: 10.1109/WorldCIS.2015.7359407.

[77] M. Putchala, "Deep Learning Approach for Intrusion Detection System (IDS) in the Internet of Things (IoT) Network using Gated Recurrent Neural Networks (GRU)," *Browse all Theses and Dissertations*, Jan. 2017, [Online]. Available: https://corescholar.libraries.wright.edu/etd_all/1848.

[78] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Networks*, vol. 90, p. 101842, Jul. 2019, doi: 10.1016/j.adhoc.2019.02.001.

[79] A. J. Siddiqui and A. Boukerche, "TempoCode-IoT: temporal codebook-based encoding of flow features for intrusion detection in Internet of Things," *Cluster Comput*, Sep. 2020, doi: 10.1007/s10586-020-03153-8.

[80] Abhishek Verma and V. Ranga, "RPL-NIDDS17- A Data set for Intrusion Detection in RPL based 6LoWPAN Networks (Internet of Things),"." https://doi.org/10.5281/zenodo.1406034.

[81] A. Vandierendonck, "A comparison of methods to combine speed and accuracy measures of performance: A rejoinder on the binning procedure," *Behav Res*, vol. 49, no. 2, pp. 653–673, Apr. 2017, doi: 10.3758/s13428-016-0721-5.

[82] J. Townsend and A. Gregory, *Stochastic modelling of elementary psychological processes*. CUP Archive, 1983.

[83] D. J. Woltz and C. A. Was, "Availability of related long-term memory during and after attention focus in working memory," *Memory & Cognition*, vol. 34, no. 3, pp. 668–684, Apr. 2006, doi: 10.3758/BF03193587.

[84] Liesefeld, R. Heinrich, F. Xiaolan, and Z. Hubert D., "Fast and careless or careful and slow? Apparent holistic processing in mental rotation is explained by speed-accuracy trade-offs," *Journal of experimental psychology: learning, memory, and cognition*, vol. 41, no. 4, p. 1140, 2015.

[85] "scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation." https://scikit-learn.org/stable/ (accessed Apr. 19, 2021).

[86] X. Cheng, L. Fang, L. Yang, and S. Cui, "Mobile Big Data: The Fuel for Data-Driven Wireless," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1489–1516, Oct. 2017, doi: 10.1109/JIOT.2017.2714189.

[87] "The BoT-IoT Dataset." https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php (accessed Dec. 12, 2019).

[88] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[89] S. García, A. Zunino, and M. Campo, "Survey on network-based botnet detection methods," *Security and Communication Networks*, vol. 7, no. 5, pp. 878–903, 2014, doi: 10.1002/sec.800.

[90] "Kendall rank correlation coefficient," *Wikipedia*. Jun. 17, 2020, Accessed: Jun. 30, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Kendall_rank_correlation_coefficient&oldid=963057707.

[91] "spearmans.pdf." Accessed: Jun. 30, 2020. [Online]. Available: http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf.

[92] "Pearson correlation coefficient," *Wikipedia*. Jun. 22, 2020, Accessed: Jun. 30, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pearson_correlation_coefficient&oldid=963975000.

[93] "sklearn.feature_selection.mutual_info_classif — scikit-learn 0.23.1 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html?highlight=feature%20selection#sklearn.feature_selection.mutual_info_classif (accessed Jun. 30, 2020).

[94] K. Yeager, "LibGuides: SPSS Tutorials: Chi-Square Test of Independence." https://libguides.library.kent.edu/SPSS/ChiSquare (accessed Jun. 30, 2020).

[95] sampath kumar gajawada, "ANOVA for Feature Selection in Machine Learning," *Medium*, Oct. 20, 2019. https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476 (accessed Jun. 30, 2020).