electronics

MDPI

Article

Classifier Performance Evaluation for Lightweight IDS Using Fog Computing in IoT Security

Belal Sudqi Khater [1], Ainuddin Wahid Abdul Wahab [1], Mohd Yamani Idna Idris [1,*], Mohammed Abdulla Hussain [2], Ashraf Ahmed Ibrahim [3], Mohammad Arif Amin [4] and Hisham A. Shehadeh [5]

[1] Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia; belal.khater@siswa.um.edu.my (B.S.K.); ainuddin@um.edu.my (A.W.A.W.)
[2] College of Technological Innovation, Zayed University, Dubai 19282, United Arab Emirates; mohammed.hussain@zu.ac.ae
[3] Mozn, Prince Turki Ibn Abdulaziz Al Awwal Rd, Riyadh 12362, Saudi Arabia; ashraf235@gmail.com
[4] Department of Computer and Information System, Faculty of Computer Science and Information Technology, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia; marifamin@gmail.com
[5] Departments of Computer Information System and Computer Science, Faculty of Computer Science and Informatics, Amman Arab University, Amman 21156, Jordan; h.shehadeh@aau.edu.jo
* Correspondence: yamani@um.edu.my; Tel.: +60-379-676-414

Abstract: In this article, a Host-Based Intrusion Detection System (HIDS) using a Modified Vector Space Representation (MVSR) N-gram and Multilayer Perceptron (MLP) model for securing the Internet of Things (IoT), based on lightweight techniques and using Fog Computing devices, is proposed. The Australian Defence Force Academy Linux Dataset (ADFA-LD), which contains exploits and attacks on various applications, is employed for the analysis. The proposed method is divided into the feature extraction stage, the feature selection stage, and classification modeling. To maintain the lightweight criteria, the feature extraction stage considers a combination of 1-gram and 2-gram for the system call encoding. In addition, a Sparse Matrix is used to reduce the space by keeping only the weight of the features that appear in the trace, thus ignoring the zero weights. Subsequently, Linear Correlation Coefficient (LCC) is utilized to compensate for any missing N-gram in the test data. In the feature selection stage, the Mutual Information (MI) method and Principle Component Analysis (PCA) are utilized and then compared to reduce the number of input features. Following the feature selection stage, the modeling and performance evaluation of various Machine Learning classifiers are conducted using a Raspberry Pi IoT device. Further analysis of the effect of MLP parameters, such as the number of nodes, number of features, activation, solver, and regularization parameters, is also conducted. From the simulation, it can be seen that different parameters affect the accuracy and lightweight evaluation. By using a single hidden layer and four nodes, the proposed method with MI can achieve 96% accuracy, 97% recall, 96% F1-Measure, 5% False Positive Rate (FPR), highest curve of Receiver Operating Characteristic (ROC), and 96% Area Under the Curve (AUC). It also achieved low CPU time usage of 4.404 (ms) milliseconds and low energy consumption of 8.809 (mj) millijoules.

Keywords: IoT security; Fog Computing; intrusion detection; N-gram; multilayer perceptron

check for updates

Citation: Khater, B.S.; Abdul Wahab, A.W.; Idris, M.Y.I.; Hussain, M.A.; Ibrahim, A.A.; Amin, M.A.; Shehadeh, H.A. Classifier Performance Evaluation for Lightweight IDS Using Fog Computing in IoT Security. *Electronics* **2021**, *10*, 1633. https://doi.org/10.3390/electronics10141633

Academic Editors: Amir H. Gandomi, Fang Chen and Laith Abualigah

Received: 15 June 2021
Accepted: 4 July 2021
Published: 8 July 2021

1. Introduction

This work proposes lightweight solutions on the basis of standardized resources for the secure connection of IoT devices. This ensures a secure communication between the constrained environment devices and typical Internet hosts, using a lightweight yet standard compliant Internet security anomaly intrusion detection system.

Fog Computing offers a platform that addresses issues linked with the security as well as the privacy of IoT, since the IoT devices have very limited computational resources and those devices might not be able to perform the required computational tasks. Consequently, more development will occur in the area of intrusion detection because computer networks encounter new attacks daily. In the past decades, critical attacks have hit the Internet, leading to appalling consequences for computer users of different levels, such as companies, end-users, and governments [1,2]. The main challenge is to design reliable, secure IoT devices, including the Fog nodes that are located near the network edge. Fog Computing refers to a paradigm that deals with an extension of Cloud Computing as well as network edge services for the purpose of addressing the Cloud's in-built glitches, such as the absence of support for mobility, latency, and awareness of location. The Fog system refers to a medium that is decentralized and has the ability to operate as well as process local data, which can be further installed in a heterogenic hardware, thus ensuring it is suitable for IoT applications. Importantly, Intrusion Detection Systems (IDSs) comprise a vital segment of the security configuration for any IoT or Fog related network, with the aim of ensuring top-notch service quality. Moreover, a lightweight IDS is more reliable, as there is a lack of resources for IoT and Fog related devices. There has been extensive research on Anomaly-based IDSs as protective techniques used in addressing the detection of attacks that are not known, known as zero-day attacks. Dissimilar to the types of IDS that are signature-based, they detect attacks based on pre-determined signature and known attacks. An IDS that is anomaly-based is more concerned with detecting novel kinds of attacks, unknown to the system [3,4]. However, this process is achieved via variation detection in the way the system behaves with regards to a previously defined profile of a normal system. The prediction of the anomalous behavior of a process in execution via the trace of a system call is a normal act among the security-based community and remains an active area of research. In addition, it is a distinctive problem in the recognizing of patterns, and it can be solved through algorithms in machine learning.

Machine Learning Based IDS for IoT Applications, within the computer network security domain, can be described as activities entailing the attempt of outer entities infiltrating a network's ability to access a device so as to steal its information [5]. In previous times, IDS has been employed; however, due to the contemporary challenges associated with IoT new age, most of the IoT devices possess very minimal resources which run difficult security solutions, thus leading to the non-effectivity of the systems. According to Jan and his companions [6], a lightweight IDS probably provides a beneficiary solution for IoT devices; it makes use of machine learning algorithms for the detection of attempts, thus injecting irrelevant data to a network [7] and placing more focus on securing medical devices. Due to the nature of IoT systems, comprising different devices that produce a massive volume of data for overworking IDS, there is a need to combine different machine learning algorithms for the improvement of the detection rate [8].

According to the characteristics of Fog Computing, this article investigates and evaluates a general scheme: the IDS in an IoT environment based on machine learning. Furthermore, the proposed system employs a feature extraction technique based on Perceptron, thus evaluating the performance of the Modified Vector Space Representation (MVSR) technique, referred to as N-gram, which serves as templates for the training phase. In the pre-processing phase of this research, the feature selection process is added innovatively. The reason behind this design is to ensure improvement of the classifier algorithm to enable it to be lightweight during task performances hosted by Fog Computing. Furthermore, in this study, there is a comparison of the performance of some feature selection principal component analysis (PCA) with Mutual Information (MI) using Filter Methods. A simulation study made use of different algorithms for classification. The findings from the experiment revealed that their approach performed well, and it aided an accurate distinctive process behavior via system calls. This research therefore presents the design and performance evaluation of the technique used to represent the Modified Vector Space (MVS) based on raw ADFA-LD system call trace data, which is called N-gram. This is in conjunction with comparing other classifier designs, based on the majority of the classifiers, such as Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM)—Linear, Support Vector Machine (SVM)—Radial Basis Function (RBF), K-Nearest Neighbor (KNN), and Naive Bayes as intrusion detection classifiers, as well as verifying MLP superiority. Furthermore, the model's performance is evaluated using Raspberry Pi, a popular IoT device. On this note, this research is considered to solve issues related to the Evaluation Performance (Recall, F1-Masure, Accuracy, False Positive Rate (FPR), Receiver Operating Characteristic (ROC), and Area Under the Curve (AUC)) and evaluate CPU Usage Testing Time and Energy Consumption. Finally, from the evaluated metrics, the research proposes the best lightweight IDS based on vector space representation (1-gram, 2-gram) as well as compare the presented IDS against the ADFA-LD.

1.1. Motivation

The focus of this research is the build-up of a lightweight security solution for IoT systems. This enhances the security of the Fog devices and reduces the overhead at the cloud side. Figure 1 shows the structure of the lightweight IDS for a Fog Computing environment.

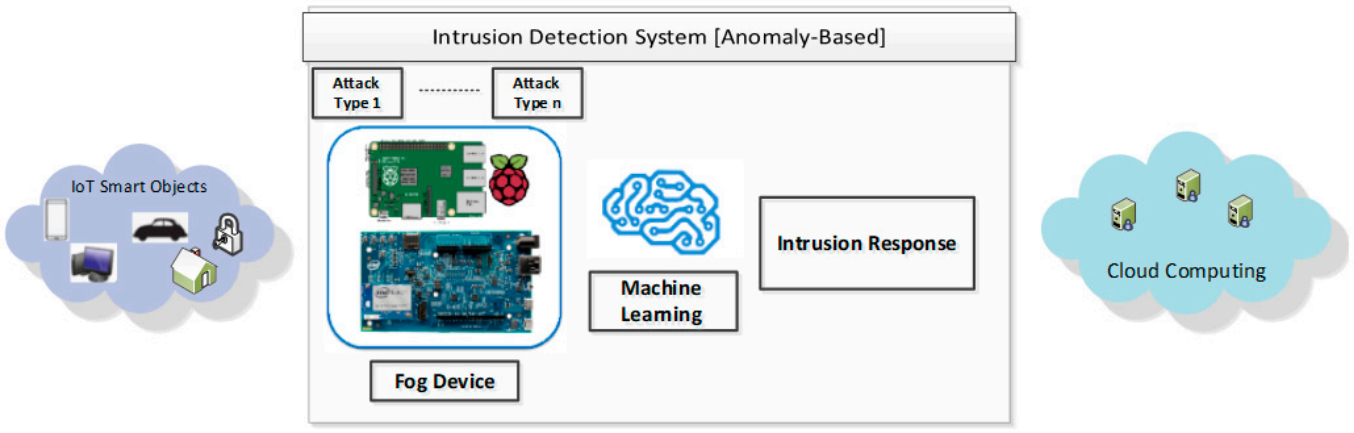


Figure 1. The structure of IDS—lightweight for IoT network using Fog Computing.

The Main Motivations of This Research

- Reducing and mitigating the security challenges in IoT—this work would serve as a general scheme of the Intrusion Detection System, in accordance with the Fog Computing Environment's features.
- Improving and examining different feature extraction and selection methodologies compatible with the constrained resources of Fog Computing.
- Evaluating the performance of different machine learning models based on time, energy consumption, and space complexity.

1.2. The Work Contribution

Regarding the security of IoT systems, issues such as suitability of high-end traditional security solutions exist due to the challenge of low storage capacity as well as low power for processing the IoT devices. Additionally, connectivity between IoT devices lasts for long periods of time without any human intervention [9–11]; the hosting of a Fog node could be done via a large group of devices comprising video surveillance cameras, routers, embedded servers, switches, and industry-based controllers. Most of the aforementioned devices might have insufficient performing computational power; for instance, there is a need for efficient performance as well as the provision of real-time feedback and responses by intrusion detection algorithms, data analysis, and protocols used in cryptography. Thus, lightweight protocols and algorithms that do not require high-end computing resources are needed. An, Zhou, Lü, Lin and Yang [12], with the aim of overcoming the issue of space limitation among Fog nodes, proposed a lightweight IDS, which they referred to as Sample Selected Extreme Learning Machine (SS-ELM). The dataset employed by the researchers was the KDD Cup 99, coupled with SS-ELM, which outperformed the classical

Back Propagation (BP) algorithm with regard to the time spent in training and the accuracy of detection. Consequently, Hosseinpour, Vahdani Amoli, Plosila, Hämäläinen and Tenhunen [13] proposed an Artificial Immune System [14] that was lightweight and distributed IDS, with a detection accuracy reaching 98%. Furthermore, as a means of detecting the behavior of attacks on ADFA-LD, and with an FPR achievement of less than 20%, Xie, Hu, Yu and Chang [15] employed frequency-based models. Invariably, the work was proceeded by Xie, Hu and Slay [16], who implemented a single class SVM model on short sequences. Overall, a better performance was achieved; however, the research encountered a low positive rate of about 20%. In countering these challenge, another researcher [17] implemented a vector space representation and used several classifiers in attaining a detection accuracy above 95%. Among the primary lapses in the work with lightweight IDS are the use of outdated datasets. In addition, IDS models that make use of contemporary datasets are specially designed for cloud-related platforms. The aforementioned challenges reveal the need to develop lightweight smart security solutions that are easily distributed and possess the ability to last long in service. Instead of ensuring individual security for many IoT devices, it is more reasonable to ensure the implementation of network data security solutions. The artificial intelligence theories, like Machine Learning, have already proven their significance when dealing with heterogeneous data of various sizes [18,19]. Thus, in this study, a lightweight IDS via the use of a suitable host dataset was developed. Hence, the core contributions of this work are as follows:

- To review security challenges in IoT. Based on the review, a general architecture of lightweight IDS based on the machine learning method is put forward. We also review the specific aspects and look at the advantages, limitations, and key challenges.
- To explore Intrusion Detection Systems (IDSs) using limited resources, such as a Fog node (Raspberry Pi 3), using a contemporary dataset, ADFA-LD, rather than the outdated KDD Cup 99 dataset.
- To propose a lightweight technique for Intrusion Detection Systems in a Fog Computing environment, which reduces and mitigates the overheads in the Cloud security challenges.
- To examine and improve different feature extraction and selection methodologies that are compatible with the constrained resources of Fog Computing.
- To evaluate the performance of different machine learning models based on time, energy consumption, and space complexity and to evaluate the performance of MLP and its different parameters via a single hidden layer perceptron for practical detection time and energy, deference's number of nodes, and reduced computational complexity for the selection and extraction of features.

The rest of the paper is structured as follows: Section 2 discusses the related works and analyzes the requirements of IoT security events and IDS in Fog Computing. It also presents a brief review of the IDS literature for a foundation of the presented work. It specifies the evolution and history of lightweight IDS based machine learning in the components of the IoT and their drawbacks. Section 3 is an overview of the methodology and the technical background of this work. Section 4 provides the analysis of the benchmark ADFA-LD dataset and details of an experiment in terms of the software and hardware used. We discuss the evaluation of the results in Section 5. Section 6 is a summary of the conclusion, and future work is presented in Section 7.

2. Related Work

This article is aimed at motivating researchers in the field of IoT security to develop IDSs via lightweight-based machine learning for the securing of IoT learning. It proposes suitable approaches for assessing the performance of classifiers statistically.

2.1. Fog Computing and IoT

Figure 1 shows how Fog Computing extends the cloud computing and its services from the core to the edge of the network. The general structure of Fog Computing consists of three main components: (i) end devices; (ii) Fog nodes; and (iii) Back-end cloud infrastructure. Fog computing provides computation and storage in addition to network services between the edge and the cloud servers. Fog computing nodes are heterogeneous in nature and are deployed in a diverse set of environments.

Fog computing is becoming an efficient and cost-effective distributed computing paradigm to support applications on billions of connected devices forming the Internet of Things (IoT). Fog computing is proposed to help resolve issues in IoT applications that require low latency, geolocation, and mobility support in addition to location awareness. The infrastructure of Fog Computing allows applications to run in an environment close to the edge of the network and to reduce the overhead in the cloud [20]. Fog computing is a platform that, similar to the cloud, provides computing resources, storage, and application services to the end user [21]. As compared to Cloud Computing, which empowers client computers via the distribution of resources for the computing and storage of data that are resident in a remote server, or the "cloud", Fog Computing ensures the availability of such resources to connected clients closer to the data source.

2.1.1. Fog Computing and Similar Technologies

Though Cisco was the first to suggest the name "Fog Computing", related ideas have been researched and proposed by other parties. The subsections below provide details regarding such related technologies, with the inclusion of how they are different from Fog systems. In addition, a more comprehensive comparison of these similar technologies is available in the literature [22,23] for edge computing.

Edge Computing: this deals with performing localized device processing via PAC, a controller that helps in the handling of data communication, processing, and storage [23]. It has more advantages than Fog Computing due to its ability to reduce failure points as well as ensure the independent nature of individual devices. Nevertheless, this same feature is responsible for its difficulty in managing and accumulating data, especially in large scale networks, for example, the IoT [24].

Cloudlet: this refers to the middle aspect of the three-tier hierarchy referred to as "mobile device; cloudlet; cloud". Cloudlet comprises four main characteristics, namely the ability to totally self-manage, possession of adequate computing power, built on standard cloud technology, and less end-to-end latency [25]. However, there are differences between Fog Computing and Cloudlet, such as non-suitability of application virtualization to the environment, consumption of more resources, and the inability to work in offline modes [25,26].

Micro-Data Center: this refers to a minute fully functioning data center that houses numerous servers that are capable of supporting many virtual machines [27]. Numerous technologies, including Fog computing, are able to benefit from centers of micro data, due to their aptitude for latency reduction, conservation of bandwidth consumption, inbuilt security protocols, relative portability, and enhanced reliability, via compression, and the potential of accommodating numerous novel services.

Fog computing ensures that storage and computation are provided apart from the services occurring in the network within cloud servers and the edge. The nodes of Fog Computing are quite heterogeneous and thus are deployed in different environments.

2.1.2. Fog Computing Mitigating Security Challenges in IoT

The introduction of Fog Computing was as an means of extending that of cloud computing; thus, it inherited many of the privacy and security challenges faced by the cloud [28,29]. Solutions to security issues that were designed for the cloud could actually be applied in the Fog Computing area; nevertheless, limitations could exist due to the special characteristics of Fog Computing [30]. In addition, cloud computing offers a medium that helps to resolve issues linked to the privacy and security of the IoT due to the limitations in the computational resources of IoT devices as well as issues such as inability of those devices to perform their required tasks for computation [31]. Thus, Fog Computing can be likened to an extra IoT security tool, wherein it assists in reducing some concerns regarding the security of IoT [31,32]. Some of the major issues regarding the security of Fog Computing are Access Control, Authentication, Privacy, Intrusion Detection.

2.1.3. Advantages of Fog Computing over Traditional IoT

The features of Fog Computing [33–35] are as follows: (1) Low latency: less access time needed for computing and Fog node storage resources, also referred to as smart gateways. (2) Location awareness: the location of the Fog at the network edge makes it aware of where the applications are located as well as their specificity. This is of benefit as the awareness of context is a very essential aspect of IoT applications. (3) Distributed nodes: dissimilar to centralized cloud nodes, Fog nodes are usually distributed. Thus, there is a need for the deployment of multiple Fog nodes in distributed geographical areas so as to ensure the provision of services to the mobile devices in such areas. An example could be with regards to vehicular networks, which deploy Fog nodes in highways, thus providing less latency of data or even ensuring video streaming to vehicles. (4) Mobility: mobility is supported by the Fog, wherein direct communication is allowed between the smart gateways and the devices at a close proximity. (5) Real-time response: Unlike the cloud, a more instant response can be achieved via Fog nodes, as they have less latency than the cloud. (6) Interaction with the cloud: interaction can take place among Fog nodes and the cloud by communicating directly with just the data required to be received by the cloud, as seen in Table 1.

Table 1. How Fog can help cloud computing [33,34].

Cloud Challenge	How Fog Can Help
Critical Latency Requirements and Data Rich Mobility	Less network hopping; less concentrated load. Data located at optimal depth; local caches. Significant mitigation in the movement of data within networks, thus leading to lessened expenditure, latency, and congestion, alongside the removal of bottlenecks that occur due to centralized computing systems.
Geographic Diversity	Intelligence localized as appropriate. Provides high levels of reliability, the tolerance of faults, scalability, and the provision of sub-secondly responding to end users.
Network Bandwidth Constraints	Local processing reduces core network load. Lesser bandwidth consumption.
Reliability/Robustness and Analytics Challenges	Fast fail-over; local response in emergency. Appropriate level of analysis and storage. Elimination of the core environment for computing, thus mitigating the chances of failure and major blockages.
User Data/Geographic Privacy	Fog can aggregate/ anonymize user data. The security of encrypted data can be more reliable due to its closeness to end users, thus lessening exposure to elements that are hostile. Better scalability of virtualized systems as well as encoding of data due to its close proximity to the network edge.

2.2. Relevant Work

Approaches for anomaly detection can be classified generally into two categories: supervised learning and unsupervised learning. For a supervised approach of detecting anomalies, a labelled dataset is used in constructing the system or network's normal behavior. Consequently, an unsupervised technique assumes greater frequency of normal behaviors, thus leading to the establishment of the model on assumptions, wherein there is no need of any data for training [36].

Furthermore, as informed by [37,38], a supervised anomaly detection technique has the ability of leveraging the measurement of distance as well as density of clusters for the detection of intrusions.

Consequently, some researchers developed an anomaly detection technique based on deep packets, with the sole aim of executing it on IoT devices that are resource constrained [39]. Arguments were made by the authors as to the fact that IoT devices of small sizes make use of relatively simple-to-use protocols, thus leading to network payloads that are very similar in nature. Hinging on the aforementioned idea, the researchers employed the use of a technique named bit-pattern matching for performing the selection of features. Worthy of note is the treatment of payloads as a sequence of bytes as well as the operation of feature selection via overlying byte tuples, referred to as N-grams. Thus, matching an N-gram and a bit-pattern takes place in situations wherein there is a match among corresponding bits across all positions. Thus, an experimental evaluation was proposed by the scholars via the use of dual devices enabled by Internet, though there was a low FPR for the four types of attack researched, namely directory traversal attacks, worm propagation, tunnelling, and SQL code injection. Furthermore, a brief introduction on an IoT-based internally distributed anomaly detection system was presented in another study [40]. The essence of this IDS proposal was to find the availability of network discrepancies via the monitoring of the features of one hop neighbor nodes, examples of which include data rate and the size of a packet. As informed by the authors, learning is undergone by the system, leading to the derivation of normal behaviors from the information being monitored. Nevertheless, there was no information regarding the approach used in constructing the profile of normal behaviors. Additionally, there was no clear justification as to the working techniques of the detection algorithms on low-capacity-based IoT devices. In a presentation by Pongle and Chavan [41], an IDS was designed for detecting wormhole attacks found in IoT devices. The authors assumed that the symptoms of a wormhole attack are always left on the system, for example, the exchange of a large amount of control packets within the tunnel's dual ends or the formation of a large number of neighbors subsequent to a successful attack. Employing the aforementioned logic, three algorithms were further proposed by the authors for the detection of such anomalies within the network. Based on their experiment, for the detection of wormhole, a 94% positive rate was achieved by the system, followed by 87% for the detection of both the attack and the attacker. Furthermore, a study related to the consumption of memory and power for the nodes was conducted by the authors. Thus, the proposed system is of suitability for IoT devices due to the low consumption of power and memory. However, there is a need to make comparisons between the achieved results and the literature so as to establish a baseline between both.

Security decisions are normally made with regard to the collection of data from IoT devices as well as from the environment of IoT systems. An instance of this is the learning of attack models by machine learning algorithms from the attack detection data. Thus, trustworthiness as well as the quality of the data is of importance in ensuring appropriate decisions are made. The design of reliable protocols is, however, not trivial for the collection of a high-quality dataset [42].

The deployment of IoT-based IDS mechanisms on low-capacity nodes and networks with less power is quite intricate and challenging. Worthy of consideration for signature-based methods is the high cost of storing signatures in the databases as well as computing running learning algorithms used in checking individual signatures.

Many NIDS mechanisms have been developed in the IoT via the use of attack signatures; however, there have been issues of non-identification of unknown attacks, as well as higher FPR, also referred to as false alarms [43]. Furthermore, for monitoring traffic within a network, there is a need to connect IDS hosts to any components, such as nodes, routers, or computer devices. Dissimilar to NIDS, the system calls, file system modifications, and the operating system processes are scanned by the HIDS [44]. Nevertheless, due to the many limiting factors, HIDS is not preferable.

2.3. Lightweight IDS-Based Machine Learning

IDSs, in general, collect information about the host and network resources of a system and attempt to analyze it in order to detect any probable intrusion in the system. This provides a mechanism for security specialists in an organization to monitor activities across the computer systems and network infrastructure. Figure 1 depicts a general state of IDSs on the basis of implemented techniques for detection alongside the deployed Machine Learning lightweight devices in the IoT environment. The concept of IDS has also expanded to include the monitoring and analysis of host and network features together. The simple concept behind this convergence is to assess the interconnection of all features that may act as an indicator of compromise for the whole system. As shown in Figure 2, the implementation of lightweight IDS-based Machine Learning can be achieved via employing several techniques and approaches. Thus, different mechanisms of detection have been developed for detecting abnormalities and are placed into categories such as data mining methods, machine-learning-based methods, and statistical methods [45]. Finally, a brief discussion on the lightweight IDS-based Machine Learning approaches is included.

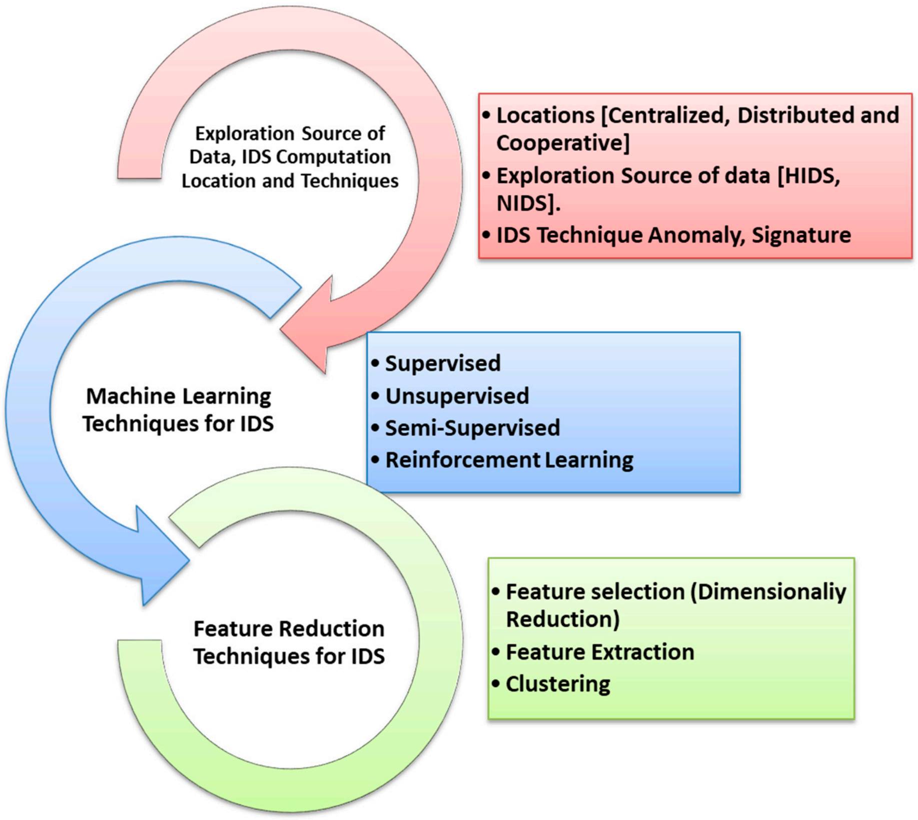

Exploration Source of Data, IDS Computation Location and Techniques
• Locations [Centralized, Distributed and Cooperative]
• Exploration Source of data [HIDS, NIDS].
• IDS Technique Anomaly, Signature
Machine Learning Techniques for IDS
• Supervised
• Unsupervised
• Semi-Supervised
• Reinforcement Learning
Feature Reduction Techniques for IDS
• Feature selection (Dimensionaliy Reduction)
• Feature Extraction
• Clustering

Figure 2. General architecture of a lightweight IDS based on the machine learning approach.

2.3.1. IDS Techniques—Location and Exploration of Source of Data

IDSs are developed into host-based intrusion detection systems (HIDS) and Network Intrusion Detection Systems (NIDS). They ensure the collection of data from diverse sources of networks and systems, thereby analyzing the data for probable threats [46]. For example, the implementation of NIDS can be done via detection techniques such as anomaly-based detection and signature-based detection [47]. For a signature-based NIDS, there is a limit in the detection of malicious threats that are already known. Furthermore, there is a need to apply a mix of both packet content inspection rules as well as packet header to the detection system via the specification of the signature and via the anomalous traffic flows. The design of anomaly detection techniques is such that they can have an automatic understanding of the attacks that are not known as well those that are not predictable for signature-based NIDS, as shown in Table 2 [45–47].

Table 2. Classification of IDS methods based on their aspects, advantages, and disadvantages.

Features/Aspects	Methods	Advantages	Disadvantages
Audit source location	Host-Based IDS	• Do not require additional hardware. • Cost-effective. • Easy to deploy due to not affecting existing infrastructure. • Ability of viewing low-level activities such as access to files and altering file permissions. • Ability of dealing with switched and encrypted environs. • Good detection and deterrence for inside intruders (Intruder Detection and deterrence) and good prevention of inner intruders (Intruder Prevention). • Outstanding determination of the extent of the damage (Assessing Damage). • Good at trending and the detection of behavior patterns that are suspicious (Threat Anticipation) and strong capabilities for supporting prosecution (Prosecution Support). • Weak real-time response but better long-term attack performance (Threat response time).	• Very limited network view. • More disposed to illegal tampering due to being close to users.
	Network-Based IDS	• Quick response. • Less exposed to false positives as compare to host-based systems. • Able to detect attacks missed by host-based systems due to network traffic monitoring at the transport layer. • Good detection and deterrence for intruders from outside and strong response time against outer intruders.	• Not aware of implementation of each host's protocol due to being far from individual hosts. • No capability to decrypt encrypted data. • Difficulty in removing evidence. • Very weak damage assessment capabilities. • Prosecution supports very weak due to a lack of data source integrity. • Threat anticipation (None).
Detection methods	Anomaly-based (behavior-based)	• Effective in detecting novel and unforeseen vulnerabilities and the ability to detect most new attacks. • Uses fewer rules compared to the signature-based techniques. Less dependent on OS. • The ability to facilitate the detection of privilege abuse, thus increasing detection rate and effectiveness.	• Weak profile accuracy due to observed events being constantly changed, thus difficulty in adapting to unceasing changes in normal behaviors as well as a dynamic anomaly. • Unavailable during the rebuilding of behavior profiles, thus difficult to discover the boundaries between abnormal and normal behavior. • Difficult to trigger alerts at the right time, thus higher false-positive alarms.

Table 2. *Cont.*

Features/Aspects	Methods	Advantages	Disadvantages
	Signature-based (knowledge-based)	• Reliable, efficient, and generates a very small rate of false alarm for the detection of well-known and definite intrusions. • Detailed contextual analysis, thus the simplest and effective method to detect known attacks.	• False alarms due to poorly constituted signatures, thus the limitation in unknown attacks. • Matching signatures is properly conducted for individual attacks; however, the majority of attacks are multi-connected, thus leading to little understanding of states and protocols. • Non-effectiveness in detecting attacks that are not known, known attack variances, and evasion attacks. • Hard to keep signatures/patterns up to date. • Time-consuming to maintain the knowledge.
	Stateful protocol analysis (specification-based)	• Knows and traces the protocol states. • Distinguishes unexpected sequences of commands.	• Resource consuming for protocol state tracing and examination. • Unable to inspect attacks looking like benign protocol behaviors.
Data distribution modes	Centralized IDS	• Direct control via a central console of all response, monitoring, and detecting activities. • Fault-Tolerant A DIDS state, resulting in more difficulty in storing consistently as well as the manner of recovery. The intrusion detection system is in a state of central storage, resulting in easier recovery after a crashing incidence. • Minute or no overhead is imposed on the systems, apart from those running analysis components, wherein the imposing of large loads is necessary. In addition, the hosts might be required to dedicate themselves to the task of the analysis. • Execution: there is a need for continuous running of a relatively small number of components. • Resist subversion: a smaller number of components are required to undergo monitoring. However, these components are larger and more complex, thus leading to more difficulty in monitoring.	• Data can be destroyed or modified by an attacker. • An intruder can modify or disable running programs on a system. • There is a limit to the size of the intrusion detection system with regard to its fixed number of components. However, as there is an increase in the number of monitored hosts, there will be a need for more storage and computing resources for analyzing the components, so as to ensure a balance of the load. • Dynamic Reconfiguration: a minute number of components are responsible for the data analysis. Reconfiguring them would probably require the restart of the intrusion detection system.

Table 2. *Cont.*

Features/Aspects	Methods	Advantages	Disadvantages
	Distributed/Decentralized IDS	• The distributed data utilize the traffic information from various sources to investigate the security status. • Scalability A DIDS has the ability of scaling into a higher number of hosts via additional components, as necessart. Scalability might be restricted to the necessity of communication between components, coupled with the existence of central components used for coordinating. • Dynamic Reconfiguration: restarting and reconfiguration of new components might occur without affecting the remaining systems of intrusion detection.	• The data flow between host monitors and the director agent has the tendency to generate high overheads of network traffic. • Small overheads are imposed on the systems due to the fact that the running components are smaller. Thus, imposing extra load is executed on majority of the monitored systems. • Execution is more difficult due to the need for continuous running of larger number of components. • Resist subversion: a larger number of components need to be monitored. Nevertheless, due to the larger number, components can cross-check each other. The components are normally of small size and less complexity.

IDSs can be classified into two categories: Network-based IDS (NIDS) and Host-based IDS (HIDS). In NIDS, several network segments are connected to each other, and network traffic is being monitored to detect malicious activities. Meanwhile, HIDS are attached to a computer device, whereby malicious activities that occur within the systems are monitored. Dissimilar to NIDS, analyses are made by HIDS on the system calls, file system changes, application logs, inter-process communication, and running processes, apart from just the network traffic. Furthermore, system call traces are used in detecting intrusions with HIDS. These intrusions could be in the form of anomalous sub-sequential traces. The collective anomalies are further translated into malicious programs, violators of policy, and unauthorized behaviors [48]. A system call trace refers to the system call sequences that have been ordered, performed by a process while executing, such as write or read, and open. It is important for HIDS applied anomaly detection techniques to put the data context into consideration. Moreover, there is a clear difference between a compromised system call trace and a normal program process [48]. Due to this, it is impossible to apply point anomaly detection techniques that consider personal data instance anomalously in this domain [44]. Additionally, IDS approaches can be classified based on specifications, anomaly, or signature, as shown in Table 2, which presents the advantages and disadvantages of intrusion detection techniques [38,49].

Early IDSs are mainly isolated single instances used in monitoring single networks via the conducting of local analysis in finding attacks. Thus, among stand-alone IDS instances of such a manner, there is no occurrence of interaction as well as communication. It is, however, obvious that such solutions will not be able to detect attacks that are highly distributed and sophisticated. The establishment of connectivity between isolated IDSs will not be possible within the occurrence of malicious events at once in different places, in order to ensure that these large networks and IT ecosystems are well protected [50]. A distributed IDS refers to one that denotes the performance of data analysis in different locations proportionally to the amount of monitored hosts. Previous research only considered the amount of data analysis components and locations rather than the components of data collection. Some distributed systems of intrusion detection are classified into Distributed Intrusion Detection System (DIDS) [51,52].

A centralized IDS is one that involves the analysis of data performed in a fixed number of locations, not minding the amount of monitored hosts. Similarly, for a decentralized IDS, there is no need to consider the data retrieval component location; the only location to be considered should be that of the analysis components. A comparison between centralized and decentralized intrusion detection systems, based on respective conditions, is illustrated in Table 2. The distribution of the IDS can occur across many nodes within a network. However, in the case of intrusion, decisions can be made collaboratively or in an independent way. Regarding the collaborative way, a single decision is shared by many nodes. Collaboration can employ statistical techniques, including game theory and voting; however, using an independent approach, individual nodes make their decisions separately over the network. In addition, a situation whereby all nodes work together under the same capacity is referred to as a flat infrastructure, in a distributed way, as compared to clustered infrastructure, where all nodes are under clusters of diverse capabilities, with each of them making contributions to the decisions uniquely. Thus, the location for computing is another aspect of a distributed IDS. Here, there is a centralized location for computation works on the retried data from the entire network. Dissimilar to the centralized computation location, the location of a stand-alone computation works on the local data, thus disregarding any decisions made by other nodes. A mix of both stand-alone and centralized locations is achievable via cooperative computation, wherein individual nodes are able to detect intrusions personally yet provide contributions to the final decision. Additionally, an IDS computation can be operated hierarchically, wherein all detected intrusions are sent to the root node by a cluster, after which a decision is taken [49].

Some of the research in this context is associated with detecting anomalies in industrial data [53,54] and the industrial IoT [55,56]. In this class, a real application service of

IoT, like in smart homes [57] or in industry [58] is considered, and according to context and application, different techniques are employed for the detection of data anomalies. Intrusion detection refers to the activity involving the detection of actions carried out by intruders against information systems. These intruders might be either internal or external. The internal intruders refer to users within the network that possess some extent of legitimate access and try to raise their accessibility privileges, so as to cause a misuse of privileges that are unauthorized for them. On the other hand, the external intruders refer to those outside the targeted network, who try to gain access in an unauthorized manner to information in the system [38,59]. A typical IDS is made up of a system for reporting, sensors, and an engine for analysis. The deployment of sensors occurs at several network hosts. The major task of these sensors is the retrieval of data from the host or network, such as changes to file system, system call operations, packet headers, service requests, and statistics of traffic. Moreover, the sensors transmit the retrieved data to the analysis engine, whose duty is to ensure retrieved data are being investigated as well as to coordinate the detection of ongoing intrusions. Thus, when there is a detection of intrusion in the analysis engine, an alert is generated to the network administrator by the reporting system.

Table 2 presents a comparison of methods used in detection, based on several criteria for IDS performance, thus classifying them into diverse IDS features such as threat to security, detection method, and placement strategy. More focus was placed on the state-of-the-art method review with regard to the implementation of IDS [38].

2.3.2. Datasets in IDS

In this section, a summary is provided of the popular benchmarks in measuring a dataset in the field of intrusion detection. From previous research, it has been discovered that data are generally retrieved using three means: User Behavior, System Call Sequences (HIDS), and Network Traffic (NIDS), some researchers comprehensively evaluated existing dataset via some proposed criterions, alongside an evaluation and design framework for IPS and IDS datasets, namely CPU/memory usage, log files, network data packages, user input command sequences, low-level information of systems, sequences of system call, and error in system logs. Table 3 presents some commonly used benchmarks, all of which are used in the detection of anomalies or misuse [60].

Table 3. Frequently used datasets in IDSs [60,61].

Data Source	Dataset
User Behavior	UNIX User Dataset (UNIXDS)
System Call Sequences HIDS	DARPA Files Datasets (BSM99)
	University of New Mexico Dataset (UNM) 2009
	ADFA IDS Datasets (2014)
Network Traffic NIDS	DARPA TCPDump Files Datasets (DARPA 2000)
	KDD99 Dataset (KDD99)
	KDD99 (10%) Dataset (KDD99-10)
	Internet Exploration Shootout Dataset (IES)
	University of New Brunswick, Information Security Centre of Excellence Datasets (UNB ISCX IDS 2012)
	Canadian Institute of Cybersecurity IDS CICIDS2017 Datasets (2018)

ADFA IDS datasets are used on both Windows and Linux operating systems and are designed for evaluating system calls, such as HIDS. Whereas ADFA-LD datasets offer present day datasets of Linux for traditional HIDS-based evaluation, ADFAWD datasets offer novel datasets powered by Windows, for HIDS evaluation [60].

Numerous datasets are in existence, such as ADFA13, ISC2012, KDD99, and DARPA98, and they have been employed by scholars for evaluating intrusion detection performance and prevention techniques. Nevertheless, there is a gap in research with regard to assessing and evaluating datasets, coupled with the lack of any befitting datasets in the domain.

2.3.3. Machine Learning Techniques for IDS on IoT

An IDS is designed for monitoring occurring events within a computer network or system, thus leading to the analysis of possible incident signs as well as frequent prohibition of unauthorized access [57,62]. There are two methods of differentiating between a malicious and normal system behavior. They are the misuse detection approach and the anomaly detection approach. Misuse detection is capable of detecting a sequence of actions as an attack if there is a match between former complete descriptions of the series of actions, also referred to as the signature, executed by the attacker. The second approach helps in discovering data patterns, which are not conformable to the awaited behavioral expectations. A major technique that has proven to be practically valuable with regard to anomaly-based detection is machine learning. The foundational principle for machine learning approaches lies in the data prediction and learning [36]. Machine learning can be split into two categories, namely Artificial Intelligence techniques, examples of which include SVM, KNN, DT, k-means clustering and MLP, and Computational Intelligence (CI) techniques, examples of which include Fuzzy Logic, Generic Algorithms, Artificial Immune Systems (AIM), and Artificial Neural Networks (ANNs) [63].

It is important to note that the field of machine learning deals with the development of systems with the capability of automatic data learning [36,64], including the identification of patterns that are hidden automatically [65]. Machine learning algorithms are characterized based on their styles of learning as well as the functionality in the similar nature of their manner of engagement [46]. Figure 2 presents an overview of machine learning approaches based on their learning styles in lightweight techniques. The techniques of machine learning tend to be effective with regard to the improvement of detection rate and the reduction of false alarm rates and offer decreased communicating and computing costs [56,66]. The techniques of machine learning can be split into different phases, including supervised, unsupervised, and semi-supervised learning [67,68] and Reinforcement Learning (RL) [69].

Supervised Learning

Here, the algorithms are able to learn representations to predict cases that are not known, from labelled inputted data. Examples of supervised machine learning algorithms include SVMs for solving issues regarding classification and RFs for solving both regression and classification issues [64,70]. SVMs are extensively used in IDS-based research as a result of their powerful classifying and their computing practicality. Thus, this makes them of great suitability for high dimensional data, though with criticality in the selection of a reasonable kernel function; therefore they are resource insatiable and demand memory as well as processing units for computations [46]. The RF algorithm [71] is an ensemble approach of supervised learning, which powerfully deals in an effective manner with uneven data yet is prone to issues of overfitting.

Unsupervised Learning

In the unsupervised learning scheme, representations as well as the structure from inputted data that are not labelled are learned by the algorithms. The aim of this algorithm is the modeling of foundational data structures for the prediction of data that are not known [64,70]. Some examples of these unsupervised learning algorithms include techniques used in reducing features such as clustering techniques, PCAs, and Self-Organizational Maps (SOMs). A PCA refers to an algorithm used in efficiently speeding up feature learning that is not supervised [72]. Many researchers have employed PCA in their work, before classification application [73]. Thus, clustering algorithms like K-means, among other algorithms for distance learning, are employed in the detection of anomalies. Furthermore, an SOM refers to an ANN employed for the reduction of IDS payloads [74]. Clustering algorithms for anomaly detection have their own drawbacks, such as subjectivity to initial conditions, instances including centroids, and the production of high rates of false positives [75]. Furthermore, an approach was proposed by Oh, Kim and Ro [76], wherein an SOM intrusion detection system in real time is able to classify related data and at the same time ensure cluster visualization. Via correlating features, the detection can perform map labelling produced by SOMs. Moreover, there is wide use of SOMs in anomaly detection systems. Some researchers have applied SOMs as HIDS [77].

Semi-Supervised Learning

This refers to a kind of supervised learning that makes use of data that are not labelled, in training. The data used for training comprises a small amount of labelled data and a huge amount of data that are not labelled. Thus, such learning could be useful in situations where there is an absence of large amounts of labelled data; an example is the case of a photo archive where not all images are labelled, thus consisting of some unlabeled data, one of which is used to enhance the accuracy of IDS [78,79]. In another study, two semi-supervised Spectral Graph Transducers used for classification and the Gaussian Fields technique were employed for detecting attacks that are not known, while a semi-unsupervised clustering approach, known as the Metric Pairwise Constrained K-Means (MPCK-Means), was employed for improving the manner in which detection systems perform [80].

Reinforcement Learning (RL) Methods

The ability to learn from nearby environments is among the initial methods of learning experienced by humans. On a natural note, human beings commence learning through interaction with their environment. Therefore, the inspiration behind RL is based on the neuroscientific as well as psychological aspects of animal behaviors, coupled with the mechanisms through which the control of environments can be enhanced by agents [81]. Furthermore, RL has the ability to ensure an agent is able to have knowledge on mapping situations to actions in an appropriate manner, so as to attain maximal rewards [82]. This agent has no memory of the expected actions to execute, but rather needs to first learn about the actions that can produce top-notch rewards; this is done via trial and error. The trial and error feature is the unique and main RL characteristic. Thus, the agent continuously learns from its past experience so as to attain higher rewards. Among recently successful methods in RL is the "deep Q network" [81]. In the literature, there are proposals regarding extending deep Q networks, with the inclusion of double Q-learning [82], non-stop control with deep RL [83], and prioritized experience replay [84].

Machine-Learning-Based IDS Observation for IoT Security

It is quite challenging to implement conventional approaches such as machine-learning-based IDS observation due to their complex structure of technologies, protocol stacks, and layers. Thus, there is a great need for IDSs for regulation of secure and reliable networking among IoT devices. Machine learning or deep learning techniques have been employed for developing IDS, some of which are self-organizing map (SOM), Naïve Bayesian (NB), Random Forest (RF), Support Vector Machine (SVM), as well as Artificial Neural Networks (ANNs), among others [47]. A past study proposed that the system call of a Fog-based system can assist in securing IoT devices [85]. The aforementioned work utilized DT for analyzing traffic across a network, with the sole aim of detecting traffic sources of suspicion as well as consequently detecting the behavior of DDoS. DT has been explored in different scenarios that deal with IoT system security. Some studies [86] have mentioned how DT can be used for the detection of intrusions, whereas others [87] informed on DT's use in identifying types of operating systems that are not known in IoT. Another study focused on the authentication of the physical player of IoT wireless devices [7], comparing DT performance with SVM and K-means when experimenting with Machine Learning algorithms aimed at discovering targeted attacks on medical devices. The authors concluded by stating the following: DT possesses the highest rate of detection, low FPR first training, and speed in prediction as compared to that of K-means and SVM. Nevertheless, it was also stated that they experienced a failure in the ability of the algorithm in detecting and providing results similar to those previously achieved; in cases where there is a familiarity between the attacker and the device, it has knowledge of the schedules as well as the patterns of data. DT achieved the highest overall accuracy, and it was also noted that supervised learning performed more suitably with regard to threat detection compared to that of unsupervised learning.

Relating to the IoT environment, a study [88] developed an Android malware detection system that had the ability to provide security for IoT systems; the scholars also applied a linear SVM to the developed system. There are some studies that indicate that SVM is a commonly used approach for issues relating to IoT security [5,6,89–93]. Wei, Luo, Weng, Zhong, Zhang and Yan [94] compared Naive Bayes and DT algorithms for the detection of malicious mobile malware in android applications, inclusive of an Androidetect system. The Androidetect system refers to an automated tool for detecting malicious applications. Furthermore, some researchers [90,93,95] compared Naive Bayes with other Machine Learning algorithms. The comparison of SVM, Naive Bayes, NN, and KNN in [90] in detecting on-off attack on IoT devices via the use of smart trust management approaches revealed that a high rate of precision was found in Naive Bayes as well as the recall; however, this was not the same with the F1-score. Consequently, as informed by Kotenko, Saenko and Branitskiy [93], the testing data results revealed that the algorithm with the least training time was Naive Bayes, as compared to that of SVM and ANN, which had much longer training times. The reason behind the lesser time in training for Naïve Bayes is due to the use of primitive operations. Correspondingly, as informed by Park, You and Lee [95], there is a lower accuracy in detection with Naïve Bayes, as compare to SVM, Linear, or DT, due to the kind of data used in their study.

Moreover, RF has also been considered as an integrated learning, which entails the selection of diverse input samples from actual training sets via a technique of resampling using bootstrapping [89]. In the aforementioned reference, the scholars compared RF with other techniques, such as KNN, NN, and SVM, for the detection of crypto ransoms within IoT networks. The analysis of their performance of classifying algorithms concluded that RF had the second highest detection, accuracy, precision rate, and F-measure, behind KNN. In addition, another study, after comparing RF with NN, DT, SVM, and KNN techniques, in an experiment trying to detect DDoS within consumer IoT devices, found that all algorithms performed quite high, apart from that of SVM, whose performance was slightly lower [92].

Based on the previous descriptions, it is established that anomaly detection consists of two types of techniques: namely, supervised and unsupervised. The majority of the algorithms from the literature were used in achieving excellent results for the respective techniques applied. However, the application of diverse algorithms for detecting intrusions was done for the purpose of enhancing the performance of IDS in all facets, including clustering, classifying, and the selection of features. Table 4 presents a comparison between commonly used algorithms.

Table 4. Pros and cons of techniques for common classifiers for the use of IDSs.

Method	Technique	Pros	Cons	Potential Application in IoT Security
Decision Tree DT	DT is used in establishing a model (i.e., a prediction model), to learn from training samples via representations in the form of branches and leaves (leading to the construction of a data structure in the form of a tree). The pre-trained model is later used for predicting new sample classes (where calculations in the error rate from the learned tree occur). It can be employed for any task relating to supervised learning. This is because it is easier to understand data more appropriately using trees.	- Simple, easy-to-use and transparent method. Easily understandable and interpretable. - Data preparation is minimal. - Ability of handling data of both categorical as well as numerical forms. - Validating models are possible statistically. - Robust. - Good performance with large data within short period of time. - Automatically selects features; strong interpretation.	- Requires large storage space due to constructive nature. Thus, to understand this method, not many DTs should be involved. - Issues regarding the learning of an optimal decision tree, is most times NP-complete, comprising optimality failures as well as issues for simple concepts. - Creation of overly complex trees by decision tree learners, which are not able to achieve proper data generalization. - Difficult concepts exist, which are challenging for learning, as they are not easily expressible by the DT. - Most of the classes are trending within results obtained from classification, thus ignoring relationships between the data. - Prone to overfitting.	Detection of intrusion [96] and suspicious traffic sources [85].
Support Vector Machine SVM	- A splitting hyperplane in a dual class feature dimension is formed, such that distance between the hyperplane (which aids in visualizing separated hyperplanes) as well as a large fraction of the time will be spent together. - During this phase, two parameters can be adjusted, while the most adjacent point of samples is maximized for each class.	- Generalization capability and suitability for data that involves huge number of features yet very little sample points [97]. - Low generalization error. - Computationally inexpensive. - Easy results interpretation. - Optimal separation hyper-plane discovery. - Ability to deal with data of quite high dimensions. - Some kernels have infinite Vapnik-Chervonenkis dimension, inferring their ability to learn concepts in an elaborative manner [98]. - Usually works very well. - Able to learn important info from little training, alongside a very firm capability of generation.	- Challenging optimal selection process for kernels. - Difficulty in understanding and interpreting SVM-centered models. - A combination of negative and positive instances is needed. - Selection of a reliable kernel function is necessary. - Needs enough memory as well as CPU time. - Some issues regarding stability of numerals are faced during the solving of the QP constraint. - Poor performance for tasks related to big data as well as multiple classifications. - Sensitivity to parameters from kernel function; sensitive to how parameters are tuned and the selection of kernels, handling only native binary classifications. - The need to write more codes for solving challenges above two classes.	Detection of intrusion [99].

Table 4. *Cont.*

Method	Technique	Pros	Cons	Potential Application in IoT Security
K-Nearest Neighbor KNN	- New samples are classified based on number of votes from nearest neighbors. - Decides classes of samples not yet known by the most voted from the closest neighbors.	- Popular. - Effective for detecting intrusion. - Easy understanding, including cases of few variables to predict. - Can be used in situations of model build-up, comprising data types, such as text, which are not standard. - Applicable to huge data amount. - Suitable for non-linear data types. - Fast training of data. - Handles noise during training.	- Variations occur in the optimal K value among diverse datasets. - Determination of optimal K value could be challenging and take much time. - Large storage requirements. - Sensitivity to selecting functions of similarity used for instance comparison. - Lack of well-organized methods of choosing K, apart from similarity check or cross validating. - Not computationally cost friendly. - Minority class suffers low accuracy issues. - Spends much time in performing long tests. - Sensitivity issues with the K parameter.	Detection of intrusions [100] and anomalies [101].
Random Forest RF	Reconstruction of diverse DTs, combined for the purpose of acquiring well-established as well as adequate models of prediction to ensure enhanced results.	- Resilient to overfitting. - Ability to bypass selecting features, thus needing fewer parameters for input.	- Hinged on construction of many DTs. - Impracticality with regard to particular real-time applications needing huge dataset training.	Detection of intrusion, anomalies [102] and unauthorized IoT devices [103].
Neural Network NN	An ANN works on three layers. The input layer receives data, in a likely manner, just as dendrites. Processing of the input is conducted by the hidden layer, in similarity to axon and soma. The output later is responsible for producing the calculated outputs, similar to the terminals within a dendrite; this is with the aim of achieving an independent outcome of the inputted data set.	- Able to perform tasks not performable by linear programs. - Able to ensure ongoing process, even at instances of neural network failure by elements, without having challenges in parallel nature. - Able to learn; no need for reprogramming. - Ability to deal with non-linear data due to strong fitting attributes.	- The need for training time to operate. - The need for processing time to deal with neural networks of huge capacity.	Adopted improved optimizers, activation functions, and loss functions.

Table 4. *Cont.*

Method	Technique	Pros	Cons	Potential Application in IoT Security
k-Means clustering	k-Means clustering is an unsupervised learning approach that identifies clusters in the data according to feature similarities. k refers to the number of clusters to be generated by the algorithm.	Unsupervised algorithms are generally a good choice when generating the labelled data is difficult. k-Means clustering can be used for private data anonymization in an IoT system because it does not require labelled data.	k-Means clustering is less effective than supervised learning methods, specifically in detecting known attacks [104].	Sybil detection in industrial WSNs [105] and private data anonymization in an IoT system [106].
Self-Organizational Map SOM	Unsupervised learning neural network maps multidimensional data onto a two-dimensional grid. Geometric relationships between image points indicate similarity. Neurons are arranged in a two-dimensional grid; each neuron contains a weight vector.	- Data mapping is easily interpreted. - Capable of organizing large, complex datasets.	- Difficult to determine what input weights to use. - Mapping can result in divided clusters. - Requires that nearby points behave similarly.	Used the Self-Organizing Map (SOM) neural network IDS to perform RPL attack classification [107].

2.3.4. Feature Reduction (Feature Selection) Techniques for IDS on IoT

Machine learning can be categorized into two phases: the training and classification phases. The training phase has the ability to learn how to distribute features, while in the classification phase, features that are already learned are applied as normal profiles for the detection of abnormalities [108–110]. In research by Jain, Duin and Mao [111], a model of statistical pattern recognition was developed. Furthermore, data for training are normalized during processing, and noise is removed from the data. The training phase ensures the extraction of features, a representative feature set used in training the classifiers of the processed training data. However, the classification phase ensures the application of the trained classifier in assigning test data to the selected training phase features [100,112].

Training data of high quality is necessary to achieve the most outstanding performance of machine learning IDS; thus it must contain a mixture of abnormal and normal patterns [113,114]. Features refer to the vital information that is extracted from raw data; they are essential for classification purposes as well as detection, and they influence the effectiveness of a machine learning IDS.

The selection of features for IDSs processes large amounts of audit data, containing numerous features. Nevertheless, not all the features are necessary for classifying the audit data network. Consequently, the selection of features is undertaken for discovering essential features or feature sets from all audit data features. Thus, feature selection could result in the improvement of performance during classification as well as make IDSs lightweight. One study [115] offered the following benefits of feature selection: (1) Reduced dimensionality of feature space, to limit storage requirements and increase algorithm speed. (2) The removal of noisy, redundant, or unimportant data. (3) The immediate effect of data analysis tasks speeding up, reducing the time necessary for learning by an algorithm. (4) Data quality improvement. (5) The resulting model's accuracy is increased. (6) A feature set reduction helps in saving resources in the process of utilization or for future data collection. (7) Improvement in performance for predictive accuracy gain. (8) The adequate comprehension of data for gaining knowledge of the data generation process or data visualization.

Feature Selection

"Feature Learning" plays a special role in the build-up of IDSs, such that the accuracy is highly affected by the selected features [116]. Thus, diverse representations of features can help in addressing several threat detection aspects. In fact, some seem to be naive, if they carry fundamental details of a network or the software, while others seems to be rich if they represent deeper information [117]. Thus, the obtaining of features can be achieved via some processes or by a mix of these processes, namely selection, construction, and extraction.

Further objectives of feature selection as informed by [115,118] include (i) increasing the speed of classifiers and reducing the capacity of storage; (ii) conserving resources for upcoming data collection processes; (iii) obtaining additional information and comprehensive knowledge of the process of data generation. In the area of IDS, feature selection has been utilized in improving processes of classification. Three common approaches for selecting features are genetic algorithm, principal component analysis (PCA), and information gain. In addition, the selection of features can hinge on the embedded, wrapper, and filter methods [66,119]. The wrapper, filter, and embedded techniques, are discussed briefly in Table 5.

Filters methods can be further categorized into two groups, namely the feature weighting procedure and subset search procedure. The feature weighting procedure assigns weights to features individually and ranks them based on their relevance to the target concept [115,120]. In filter methods, the feature selection process is optimized for the classification procedure. Therefore, filter methods are much faster than wrapper methods and are better suited to highly dimensional datasets.

Table 5. Merits and demerits of filter, wrapper, and embedded feature selection methods [121].

Feature Selection Method	Description	Merits	Demerits
Filter	Selects the most meaningful features regardless of the model	Computationally less expensive when contrasted with wrapper and embedded methods. Quickest running time. Reduce the risk of overfitting. The ability of good generalization. Handily scaled to high-dimensional datasets.	There is no interaction with the classification model used for the selection of features. In the case of univariate techniques, feature dependency is mostly ignored and each feature is considered separately. Compared to other feature selection techniques, this can lead to lower computational performance. can select redundant variables.
Wrapper	Combines related variables to obtain subsets	Consider interaction: interact with classifier for feature selection. More comprehensive feature set search. Consider feature dependency. Generalization is better than the filtering method.	The computational cost is high. Longer duration. Compared with filtering and embedding methods, the risk of overfitting is higher. As the number of features increases, it becomes more computationally unfeasible. If another classifier is used to predict, the optimization of the solution cannot be guaranteed.
Embedded	Investigates interaction more deeply than wrapper method	Compared to the wrapper method, it is less computationally intensive, the running time is faster, and the risk of overfitting is less. As the number of data points increases, it is better than filtering methods in terms of generalizability.	Identifying a small group of features can be problematic.

Principle components (PCs) are new variables with two properties: (1) each PC is a linear combination of the original variables; (2) the PCs are not correlated to each other, and also the redundant information is removed [122]. The major areas of PCA application comprise analysis of image, compression of data, recognition of patterns, regression, and prediction of time series, as well as visualization. However, there are limitations in using PCA, such as (1) the assumption of linear relationship between variables. (2) sensible interpretations are based on assumptions of numerically levelled scaled variables, (3) a lack of probabilistic structure for models, such as those relevant in most contexts, like Bayesian decision as well as mixture modeling. Furthermore, PCA is a technique for the reduction of features that is applicable for the transformation of large variable sets into a mitigated set, thus ensuring that the bulk of the information represented is preserved in the large set. The technique also ensures the conversion of several probably related features to a smaller number of features that are not related, referred to as "principal components" [123,124]. Therefore, the main working principle of PCA can be utilized for feature selection to realize real-time intrusion detection for IoT systems; a previous work proposed a model that uses PCA for feature reduction and adopts SoftMax regression and KNN algorithm as classifiers. The author reported that the combination of PCA with these classifiers provided a time- and computing-efficient system that can be utilized in real time in IoT environments by reducing the model features [125].

Feature Extraction

Feature extraction involves transforming original features for the generation of additional, more significant features. As informed by Zheng, Vanderbeek, Daniel, Stambolian, Maguire, Brainard and Gee [126], "Feature extraction is being defined as the construction of linear combinations αTx of continuous features which have good discriminatory power between classes". One of the major challenges for the research on Neural Networks and other areas such as Artificial Intelligence is finding suitable representative multi-variate data. Feature extraction can fill this gap by mitigating complexity issues, thus proffering a simple data representation, wherein each feature space variable is denoted as a linear combination from the initial inputted variable.

System calls are of utmost importance in any operating system, as they ensure the depicting of computer processes, thus constituting a large amount of fragmented as well as unstructured texts, used by a regular HIDS for detecting cyber-attacks and intrusions. In the literature, some researchers have considered text representation techniques for classifying behaviors of processes via system call traces. A classical machine learning method works in the direction of representing features, feature extraction, and feature engineering. Nevertheless, due to advancement in embedded approaches used in machine learning, including deep learning, the importance of feature extraction and the engineering of features cannot be overemphasized. Thus, researchers usually adopt advanced deep learning techniques and methods of representing text for the purpose of capturing information related to sequence and contexts from system calls [125]. Some feature representation methods in the area of NLP used to convert system calls into feature vectors exist in literature and are discussed here. Techniques such as term frequency inverse document frequency (TF-IDF) and Term Document Matrix (TDM) are used for the estimation of vectors in the feature. However, they have their drawbacks, which include the inability to capture system call information in a sequential order [127,128]. The N-gram method of text representation has process capability of preserving the system call information in a sequential order. For example, N size can be 1, denoting unigram; 2, denoting a bigram; 3, denoting a tri-gram [17,129]. In terms of exploiting N-gram location for intrusion detection, there exist many IDSs that use the N-gram technique to analyze network packets payload; two examples are specified in [130–132]. The N-grams are used to model the language that characterizes a network traffic profile, since each different N-gram is interpreted as a different feature space used to represent the traffic. The N-gram technique has been used previously in fields like information retrieval [133] and statistical NLP [134]. With this technique, it is possible to extract sequences of symbols from a given input flow by using a sliding window of length n. At each position, a sequence of length n is considered.

Clustering

Clustering refers to a learning method that is not under supervision, and its major task is that of data mining. Furthermore, it divides the data into look-alike groups for the purpose of simplifying them; however, it might lead to loss of information [135]. Some researchers, such as [136], discovered that clustering techniques perform via splitting observed data into clusters, based on the measures of distance or provided similarities. Other approaches regarding anomaly detection based on clustering are of two categories. The first category ensures that training is provided for the model of anomaly detections via the use of unlabeled data, consisting of attacks as well as normal traffic. The second approach ensures that training is provided for the model via the regular data only, thus leading to the creation of a normal activity profile. The foundation behind the first approach lies in the fact that attack or anomalous data make up a small percentage of the overall data. Thus, if the assertion is correct, the detecting of attacks as well as anomalies can be done on the basis of cluster sizes, as the large clusters normally correspond with the normal data, leaving the other data points, referred to as outliers, corresponding to attacks.

The Advantage of Feature Extraction and Selection

Methods of extracting features have been proposed as a step prior to the processing state, to ensure a diminishing impact of class noise on the process of learning. It is thus essential for future data analysis, be it visualization, recognition of patterns, de-noising, and compression of data. Diverse methods have been developed for discovering more appropriate transformations.

One of the benefits attached to feature selection is the preservation of relevant related data of individual features; however, this depends on the size of the features; if the set of diversified original features is too small, there is a tendency to lose some feature information.

Furthermore, dimensionality reduction, referred to also as the extraction of features, entails that there would be a decrease in the feature space size, without any loss of information from the initial feature space. One of the disadvantages of extracting features is the lack of interpretability and details of the lost features [115]. Some vital features that could be possessed by IDS as informed by the literature are as follows [115,118]: (1) It is tolerant to fault, and continually runs with little or no need for human supervision. Furthermore, the IDS has the ability to recover from accidental or malicious activity causing system crashes. (2) It has the ability to resist subversion, thus making it impossible for an attacker to cause disabling or modification of the IDS. In addition, the IDS has the capability of detecting forceful modifications by attackers. (3) It reduces the overhead on systems for the avoidance of interference with normal methods of system operation. (4) It is configurable, thus being able to ensure correct implementation of security policies on systems under monitoring. The IDS also possesses the capability of adapting to changes in user and system behavior. (5) It is easily deployable; this is achievable via portability to diverse operating systems and architectures by simply installing mechanisms as well as making operation easy for users. (6) It is generic for the detecting of diverse kinds of attacks and is able to distinguish legitimate activities from an attack (false positives). In addition, the IDS does not fail to recognize actual attacks, referred to as false negatives.

The use of feature selection in detecting intrusion is mainly intended to eradicate data that are not important and those that are redundant. It is seen as a process whereby a subset of important features is selected, thus giving a full description of the problem with minimal degradation in performance [137]. Using all the features of a dataset does not necessarily guarantee the best performance from the IDS. It might increase the computational cost as well as the error rate of the system [115]. As the number of selected features increases, the required computation power, which is needed to process the data, increases, and vice versa. Therefore, due to correlations between features, selected features can achieve similar or better results in comparison to using all the features of a dataset.

The irrelevant or redundant data might contain false correlations that obstruct the learning process of the classifiers. Therefore, feature selection has a significant impact on intrusion detection system performance, as it reduces the computation cost, removes information redundancy, increases the accuracy of the detection algorithm, facilitates data understanding, and improves generalization [137,138].

2.4. Key Challenges

Dissimilar to computer networks that function by users, IoT networks are object driven, thus making it challenging for applying contemporary computer network security mechanisms. Thus, there is a need to ensure the securing of specialized tools as well as ensure the management and preservation of IoT networks and devices from vulnerabilities and threats that are surfacing. Some key challenges and directions for future research as taken from the literature outlined in this chapter are as follows: (1) The design of a singular-based-mechanism IDS is inappropriate for present-day complex and heterogeneous IoT networks. Thus, the proposal of a more suitable mechanism, which is comprised of a combination of diverse approaches might produce better results. Nevertheless, it is important to note that rather than centralizing a model, it could be more reliable to use the approach of distributed attack detection. (2) There is need for specialization of the "Rule-based/Specification-based method", which deals with the definition of laid-down rules for the behavior of benign networks. Additionally, this specialization needs to be done topologically, thus resulting in the creation of an overhead at frequent changes of topologies. (3) There is a gap in the use of signature-based intrusion detection techniques with regard to present-day evolving zero-day attacks, wherein there is no advance definition of signatures or patterns. (4) Contrary to signature and rule-based approaches, there is a need to use anomaly-based IDS with intelligent techniques for learning, which has the capability of uniquely identifying intrusions that are either known or not known, thus producing more suitable speed and accuracy. (5) Due to constraints in resources, it is quite challenging to apply machine learning as well as other intelligent techniques in IoT. There is a lack of developed approaches that make use of the aforementioned techniques for IoT; there is need to validate current approaches diligently via diverse datasets in order to verify their level of effectiveness in real-time situations using IoT. (6) It is important to note that performance might be affected by intelligent techniques, such as Machine Learning, which needs higher processing power and memory to function effectively.

3. Proposed Method

The proposed methods as depicted in Figure 3, are split into different phases, as follows:

- Phase 1: In this work, more focus is given to a simple and efficiently computable feature extraction technique, involving the pre-processing of data, comprising methods for extracting features based on Modified Vectors Space, formed from a system call trace raw data of fixed size, referred to as N-gram. The phase also compares techniques used in selecting between PCA, MI for memory use, and CPU time.
- Phase 2: This phase covers the application of various anomaly-detection algorithms on intrusion detection, for enhancing the performance of IDS at all levels, including classification, clustering, and feature selection. Anomaly detection includes both supervised and unsupervised techniques. Several algorithms are employed for the purpose of achieving more suitable results for the detection of anomalies. Various algorithms for detecting anomalies have been applied to detect intrusion, so as to enhance the performance of IDSs at all levels, including classification, selection of features, and clustering. Based on the preceding descriptions of several algorithms for detecting anomalies, Table 6 presents a comparison of the most relative algorithms. This comparison provides a summary of the pros and cons of each respective algorithm. It is important to note that weaknesses exist for knowledge-based detection techniques. Furthermore, the detection of anomalies consists of both supervised and unsupervised techniques. This research proposes an overview of the techniques used in machine learning for the detection of anomalies. Experiments carried out have demonstrated that supervised learning methods significantly outperform their counterparts, the unsupervised methods. This is based on the criterion that the attacks in the test data are not known. Thus, of the supervised methods, the non-linear methods achieved the best performance; examples of these are multilayer and SVM methods. This is the model selection phase.
- Phase 3: this refers to the evaluation performance stage, with regard to the rate of detection using Accuracy, FPR, Recall, FI-measure, ROC, AUC, CPU time, and energy consumption for testing on a Raspberry Pi.

Table 6. The pros and cons of various shallow models.

Algorithms	Advantages	Disadvantages	Improvement Measures
ANN	- Can work on non-linear data. - Strong fitting ability.	- Suffers overfitting issues. - Can get stuck in local optimum situation. - Too much time in training models.	- Adopted improved optimizers, activation functions, and loss functions.
SVM	- Relevant information can be learned from a minute training set. - String capability of generation.	- Inability to perform on large data. - Sensitivity to kernel function parameters during a classification task.	- Optimized parameters by particle swarm optimization (PSO) [139].
KNN	- Can be applicable for large data capacity. - Fast training. - Can deal with noise.	- Minority class results are not accurate enough. - Takes much time in testing. - Sensitivity issues of the K parameter.	- Time spent on comparison can be reduced via trigonometric inequality techniques. - Parameters can be optimized by particle swarming. - Optimization (PSO) [140]; balanced datasets using the SMOTE [141].
Naïve Bayes	- Can deal with noisy situations. - Ability for incremental learning.	- Poor performance on attribute-related data.	- Imported latent variables to relax the independent assumption [142].
RF	- Simplistic nature (voting result). - Ability to accommodate large data. - Automatic scaling of features.	- Poor performance on non-linear data. - Initialization sensitivity. - Inability to perform well on non-convex data; however, for a small/medium-sized dataset, it is a very good way to build a sophisticated model. - Overfitting issues. - Training time (slow training).	- It has a method to balance error when dataset is not balanced. - Imported regularization to avoid overfitting; thus, basically, RF is used when looking for high performance with less need for interpretation. - The bootstrapping resampling technique [89]
DT	- Feature selection is automatic. - Strong interpretation.	- Trending of classification result only on the majority class, leaving the minority helpless. - Correlation of data is ignored. - Overfitting issues.	- Balancing datasets with SMOTE; introduction of latent variables. - DT suffers from overfitting problem, which can be handled by Bagging and Boosting.

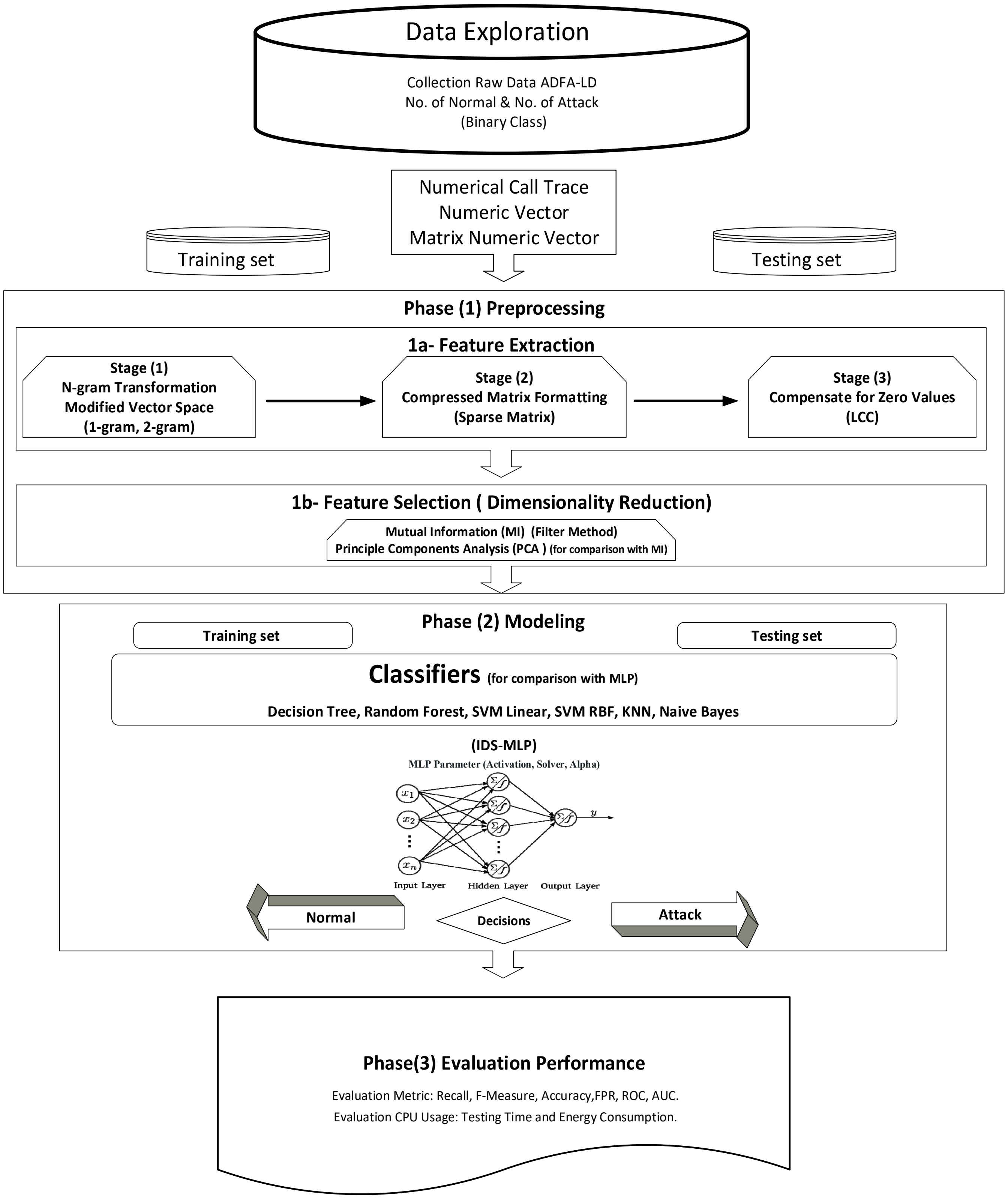

Data Exploration
Collection Raw Data ADFA-LD
No. of Normal & No. of Attack
(Binary Class)
Numerical Call Trace
Numeric Vector
Matrix Numeric Vector
Training set
Testing set
Phase (1) Preprocessing
1a- Feature Extraction
Stage (1)
N-gram Transformation
Modified Vector Space
(1-gram, 2-gram)
Stage (2)
Compressed Matrix Formatting
(Sparse Matrix)
Stage (3)
Compensate for Zero Values
(LCC)
1b- Feature Selection (Dimensionality Reduction)
Mutual Information (MI) (Filter Method)
Principle Components Analysis (PCA) (for comparison with MI)
Phase (2) Modeling
Training set
Testing set
Classifiers (for comparison with MLP)
Decision Tree, Random Forest, SVM Linear, SVM RBF, KNN, Naive Bayes
(IDS-MLP)
MLP Parameter (Activation, Solver, Alpha)
Input Layer
Hidden Layer
Output Layer
Normal
Decisions
Attack
Phase(3) Evaluation Performance
Evaluation Metric: Recall, F-Measure, Accuracy,FPR, ROC, AUC.
Evaluation CPU Usage: Testing Time and Energy Consumption.

Figure 3. Proposed method.

3.1. Phase 1: Pre-Processing

3.1.1. Feature Extraction

The data used in this article comprise system call traces. These traces refer to relatively shortened series of position integers, referred to as system call numbers; these are inappropriate for inputting into a regular model of machine learning. For the purpose of transforming the inputted data's format into a form of matrix, there is a need to implement the extraction of features during the stage of pre-processing.

Stage (1)

In this stage, the aforementioned approach is used. For the system call encoding, only 1-gramam and 2-gram as features are considered. In this stage, vectors from features are constructed via the consideration of only N-grams, which appear in the data that is under training. Also at this stage, a dense form of the data matrix is considered, where it is discovered to be too large, including being large for ample resource-based machines. Feature extraction refers to the process whereby imputed data is being represented in a compatible format with the algorithm used for modeling. Specifically, this paper is concerned about creating a simple technique for the extraction of features, which can as well be computed in an efficient manner.

Modified Vector Space Representation (MVSR) [143] serves as an extension to prior representations and takes into consideration the unique terms, only from the data used to train, via incorporation of the mechanisms needed to handle all unforeseen terms in the course of testing.

In this work, a feature set is considered as a correspondence of vector sets into traces of system call applications. There is a need to consider tuples of terms (or calls), not just the individual term in the model of vector space. The *n*-gram is defined as tuple *c = (c1, c2, ..., cn)* of *n* terms where *n* generically denotes a positive integer of small value. This modified vector space model is recurrently used in NLP and DNA and the analysis of protein sequences [128,144]. For a given *n*-gram, *c*, its weight *w*(*c*) in trace *T* is given, where *f(c)* denotes the frequency of *c* in *T*, and ∣*T*∣ represents the trace length in Equation (1).

$$\omega(c) = \frac{f(c)}{|T|} \tag{1}$$

Example (1)

For illustration of the above notion, the following trace is considered:

$$T = (6, 174, 174, 174, 6, 45, 33, 192, 33, 192, 174, 174, 6, 174) \tag{2}$$

Note that ∣*T*∣ = 14. For *n* = 2, we calculate, for every 2-gram, *c*, its frequency, and its weight (see Table 7).

Table 7. The weights and frequencies of all 2-gram.

c	*f*(*c*)	*w*(*c*)
(6, 174)	2	0.143
(174, 174)	3	0.214
(174, 6)	2	0.143
(6, 45)	1	0.071
(45, 33)	1	0.071
(33, 192)	2	0.143
(192, 33)	1	0.071
(192, 174)	1	0.071

End Example 1 [21].

The advantage of using this feature type is the inexpensive nature of computationally calculating the weight of an N-gram. Consequently, the N-gram approach results in an additional issue: the data matrix becomes more sparse. Thus, if the data matrix's dense form is taken into consideration, it would tend to be very large, even for machines with ample resources.

Stage (2)

The second stage in feature extraction in this study is the use of compressed matrix formation [145–147]. Here, the data are entered in the form of a sparse matrix, where only the weights of features are visible in the trace, thus ignoring zero weights and keeping the non-zero values. Additionally, algorithms requiring a minimal runtime are implemented. Regarding the encoding of system calls, which has been done in this research, consideration was given only to 1-gram and 2-gram, which appear in training data, to be represented as features for experimentation.

Stage (3)

The following step is used in compensating for all missing N-grams found in the test data. In this research, a Linear Correlation Coefficient (LCC) was used for the compensation test, between values of all N-grams, as follows:

$$LCC(T) = \frac{A - B}{\sqrt{CD}} \tag{3}$$

where the definitions of A, B, C, and D are as follows:

$$A = N \sum_{c} \omega(c)\overline{\omega}(c) \tag{4}$$

$$B = \sum_{c} \omega(c) \sum_{c} \overline{\omega}(c) \tag{5}$$

$$C = N \sum_{c} \omega(c)^2 - \left(\sum_{c} \omega(c) \right)^2 \tag{6}$$

$$D = N \sum_{c} \overline{\omega}(c)^2 - \left(\sum_{c} \overline{\omega}(c) \right)^2 \tag{7}$$

where $\overline{\omega}$ (c) denotes the mean value of n-gram; c is the positive training dataset; and the sum is over all n-grams in the training dataset.

3.1.2. Feature Selection

Feature selection is needed for the simplification of the model, so it can achieve a more suitable performance and ensure reduction in the time used to train. This is performed by selecting a particular subset from a series of relevant features. In this research, after the analysis and comparison of feature selection techniques, there are three main methodologies of feature selection and also a PCA approach. A Filer method was used in ranking features according to their level of relevance via several criteria, such as the value of information, correlation, and the value of $\chi 2$. Furthermore, due to the low space and complexity in time encountered in the study, a common feature selection approach, called MI, was used for classification on the basis of the information value. According to Manning, Raghavan and Schütze [127], MI is defined as follows:

$$MI(V;C) = \sum_{e_v e_c \in \{0,1\}} P(V = e_v\ ,\ C = e_c)\ \log_2 \frac{P(V = e_v\ ,\ C = e_c)}{P(V = e_v)\ P(V = e_v)} \tag{8}$$

where V denotes a random variable, taking the values of e_v = 1 if the trace contains n-gram v and e_v = 0 if the trace does not contain v. C is a random variable that takes values of e_c = 1 if the n-gram belongs to class c and e_c = 0 if the trace is not in class c. Connecting this with the area of information theory, MI measures how much information an attribute contains about the class, of which the N-gram used in this scenario. The more similarity existing between the N-gram and distributions within the class, the less information obtained when this N-gram is considered.

3.2. Phase 2: Modeling

In this work, Multi-Layer Perceptron (MLP) modeling was used. A presentation on the comparison of deference classifiers using 1-gram and 2-gram is given. The classifier used in modeling was chosen based on its wide acceptance rate as compared to other classifier algorithms. Table 7 provides a comparison of common classifiers used. Various algorithms used in detecting anomalies have been applied for the detection of intrusion for enhancing the performance of IDS. In reference to the prior descriptions of diverse algorithms used in detecting anomalies, comparisons have been made, and such comparisons summarize the pros and cons of the respective algorithms. In addition, the comparison deduces the weaknesses of knowledge-based detection techniques. Anomaly detection comprises supervised techniques; nevertheless, the design of the classifier must be in accordance with how most classifiers used at the base are designed, such as DT, NB, SVM, RF, and KNN algorithms for classification. The classifier's design is based on the majority, with base model algorithm for classification.

3.2.1. Artificial Neural Network (ANN)

An ANN works in three layers. The input layer receives data, in manner similar to dendrites. Processing of the input is conducted by the hidden layer, similar to axon and soma. The output later is responsible for producing the calculated outputs, similar to the terminals within a dendrite [148,149]. There are basically three types of ANN: supervised, unsupervised, and reinforcement [69,150].

A perceptron is the founding ground of an Artificial Neural Network, based on inspirations from models relating to biological aspects of machine learning. Furthermore, a perceptron mathematically represents the neuron, which is a fundamental unit of computation within the brain. As in the nervous system of humans, input signals are received by neurons from the dendrites, while the output produces signals in line with the axons, which in turn divert and create a connection between other neurons via a synapse of the dendrites, as depicted in Figure 4. ANN refers to a paradigm for information processing, based on biological structures and complex model relationships between inputs and outputs.

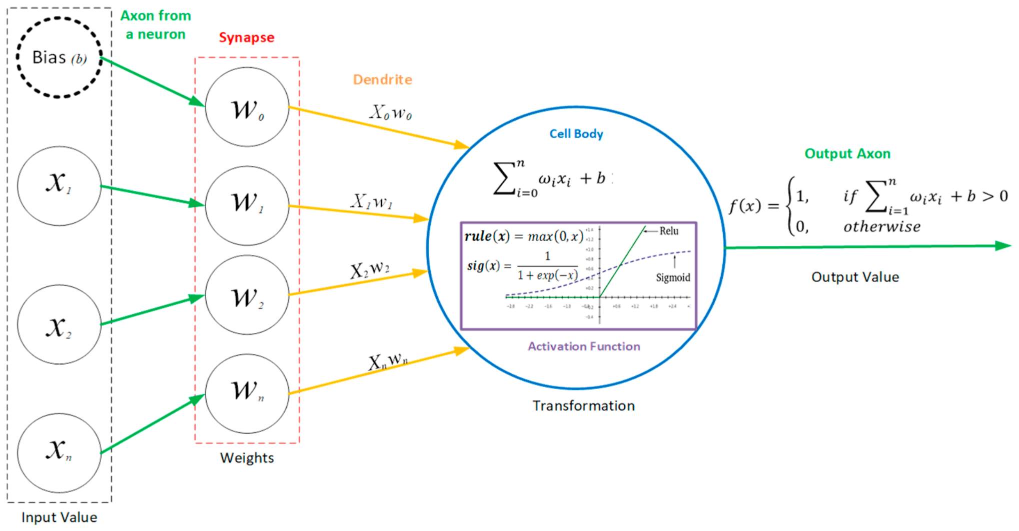


Figure 4. Structure of a single neuron.

Thus, it is possible for a single perceptron-based mathematical model to denote a linear classifier. Mapping of the input is as follows: x = (x_1, ..., x_n), a real-valued vector, to an output value $f(x)$ in accordance with the following rule:

$$f(x) = \begin{cases} 1, & if\ \sum_{i=1}^{m} \omega_i x_i + b > 0 \\ 0, & otherwise \end{cases} \tag{9}$$

where w = (ω_1, ..., ω_n) denotes the weights of real-values; b denotes bias (ω_0). Thus, generalization can be made on the perceptron via application of non-linear transformation sets, φ_i, in concordance with the inputted vector x. Thus, the output, y, is represented as

$$y(x, w) = g\left(\sum_{i=1}^{m} \omega_i \varnothing_i(x)\right) \tag{10}$$

where m denotes the number of variables or numbers; g represents the function of a non-linear activation, an example of which is a Rectified Linear Unit (relu), or a sigmoid, as depicted in Figure 4. The perceptron is a simple model, which is of much benefit with regard to the approximation of numerous non-stopping functions. A way of improving the performance of the model entails the consideration of MLP, which is achieved via the addition of a minimum of a singular layer that is hidden to the above-stated perceptron model, as shown in Figure 4. However, a commonly used learning algorithm is backpropagation, due its generalist nature regarding algorithms of minimal mean squares within a linear perceptron [151]. Thus, the sigmoid and the relu functions are two functions commonly used in activating MLP learning models and are given by

$$sig(x) = \frac{1}{1 + exp(-x)} \tag{11}$$

$$rule(x) = max(0, x) \tag{12}$$

3.2.2. Machine Learning Shallow Models

Different algorithms used in detecting anomalies have been applied for the detection of intrusion, with the sole purpose of enhancing the performance of IDS at all levels, inclusive of classification, clustering, and selection of features. More information is provided in the previous chapter on the literature review.

The conventional models used in machine learning (also referred to as shallow models) for IDS basically comprise the ANN, support vector machine (SVM), K-nearest neighbor (KNN), naïve Bayes (NB), decision tree (DT), and clustering, as well as combined or hybrid methods. Some of these methods have been studied for many years, and their methodology is proven to perform. Their focus is not just on the effect of detection but also on practical issues, which include management of data and effectiveness of detection. The pros and cons of various shallow models are presented in Table 6.

3.3. Phase 3: Evaluation Performance

In machine learning, the measuring of performances is very important. Thus, regarding issues of classification, the following can be considered: (1) Evaluation of Metrics: Recall (R), F-Measure (F1), Accuracy (ACC), False Positive Rate (FPR), Receiver Operating Characteristics Curve (ROC), Area Under the ROC Curve (AUC); (2) Evaluation of CPU Usage (Lightweight): Testing Time and Energy Consumption.

To evaluate the model in this study, several metrics were taken into consideration for evaluation. These metrics are used generally for diverse classification model applications. First, the following definitions were used: True Positive (TP): amount of detected positive traces (attack) as positive traces (attack). True Negative (TN): amount of detected negative traces (normal) as negative traces (normal). False Positive (FP): amount of detected positive traces (attacks) as negative traces (normal). False Negative (FN): amount of detected negative traces (normal) as positive traces (attack). Table 8 presents the confusion matrix that can be used for deriving additional measures. The definitions presented above are for purposes of deriving important measures, inclusive of recalls and precision. Precision (*P)* refers to the positive predictive values, recall refers to the true positive rate, and F1-measure represents the harmonic mean for both recall and precision.

Table 8. The confusion matrix for a binary classifier.

Predicted Class		**True Class**	
		Positive	Negative
	Positive	*TP*	*FP*
	Negative	*FN*	*TN*

Consequently, accuracy refers to the adequate classification ratio of the overall number of examples existing in a dataset test. In this research, all of the available contemporary evaluation metrics widely used in the information retrieval area were used [8].

Furthermore, in evaluating the performance of an IDS, several metrics for evaluation need to be calculated via the confusion matrix values [17]. Based on the above values, the most-used metrics in evaluation are given in the following formulas.

3.3.1. Precision or Positive Predictive Value

This is the ratio of the amount of predicted attack traces as actual attack traces, from the overall number of traces predicted as attack traces.

Precision is defined as

$$P = \frac{TP}{TP + FP} \tag{13}$$

3.3.2. Recall or Sensitivity

This is also referred to as the True Positive Rate (TPR). Moreover, it is a representation of the ratio of the number of attacks being predicted as actual attack traces, from the overall number of traces that are true attacks.

Recall is defined as

$$R = \frac{TP}{TP + FN} \tag{14}$$

3.3.3. Accuracy

This refers to the proportion of true results (number of attack traces and normal traces detected correctly) from the overall number of samples.

Accuracy defined as

$$A = \frac{TP + TN}{N} \tag{15}$$

where $N = TP + TN + FP + FN$, is the total number of samples.

3.3.4. F1-Measure

This refers to a measure that combines precision and recall into a single measure. It is calculated as the harmonic mean of precision and recall.

$F1$-measure is defined as

$$F1 = \frac{1}{(1/P + 1/R)} \tag{16}$$

3.3.5. FP Rate

False Positive Rate (FPR) is a measure involving the amount of normal trace labelled as attack trace by a classifier.

FP Rate is defined as

$$FPR = \frac{FP}{FP + TN} \tag{17}$$

Finally, regarding the study methodology, this work considered the ROC.

3.3.6. Curve and AUC

- Receiver Operating Characteristics (ROC) Curve: This refers to a graph of true positive rate against FPR. Thus, it gives a representation of how binary performs, based on the variations in the discriminated threshold.
- Area under the ROC Curve (AUC): This refers to the coverage area surrounding the ROC curve. Thus, it is consonant to the possibilities of ranking by the classifier in a randomly selected positive instance, above the randomly selected negative instance [152,153].

AUC-ROC curve refers to a measure of performance for issues of classifying binary within diverse thresholds. ROC refers to the curve of probability, while AUC is a representation of the measure or degree of separation. Furthermore, it gives detailed information as to the amount of the model needed to carry out variances across classes. Thus, the higher the AUC, the better the model becomes for the prediction of 0 s and 1 s as 1 s. Furthermore, the rate at which ROC curves are steeped is of utmost relevance due to it being the best for the maximization of true positive rate while ensuring the minimization of the FPR. Normally, ROC curves, with regard to the classification of binary, are used for the purpose of understanding the outputs from a classifier. Thus, to ensure the extension of the ROC area as well as the curves to a multi-labelled classification, there is a need for output conversion to binary. This can be done via the drawing of one ROC curve at each label, though an ROC curve can also be drawn by taking into consideration each element from the label matrix as binary predictions (or micro-averaging).

In addition, there is a need to consider the consumption of power, CPU consumption, and throughput, as these are essential metrics for evaluating IDS, which are being run on numerous hardware based on a specific setting, including the network's high speed nature or limited hardware resources.

4. Experimental Setup

This work was based on evaluating the results of benchmark datasets. Nevertheless, currently available datasets lack real-life properties, thus explaining why the majority of systems for detecting anomalies in the intrusion are not applicable in the production environment [154]. Consequently, this leads to an inability to adapt to non-stop variances within networks, such as changes in topology, new nodes, and variations in traffic loads, among others. Therefore, to ensure the evaluation of the proposed framework and algorithm based on their level of effectiveness for detecting attacks in IoT, this research utilized an intrusion detection dataset. According to the literature, the most preferred datasets for intrusion detection are NSLKDD and KDDCUP'99 [154–156]. Furthermore, in this work, a more suitable and modern dataset, the Australian Defence Force Academy Linux Dataset (ADFA-LD) [157,158], was employed to train and evaluate the performance of the system.

4.1. ADFA-LD Dataset

The collection and generation of traces in system call based on a large scale for the purposes of research for simplistic tasks, as well as the number of available datasets for evaluating IDS, are quite limited. Recently, some researchers proposed ADFA-LD datasets as a novel alternative for outdated datasets [157,159]. From their proposal, it was revealed that there exists a higher level of complexity in ADFA-LD as compared to other contemporary datasets. A preliminary performance analysis was conducted on ADFA-LD by some scholars [160,161], which revealed the ineffectiveness of Euclidean distance as a medium for separating normal behaviors from those that are instigated by attacks.

Table 9 presents the distribution between system call traces, splitting them into normal and different types of attacks under three ADFA-LD groups.

Table 9. ADFA-LD dataset composition based on binary classification of the types of system call traces.

Dataset	No. of Traces ADFA-LD	Class of Traces
TRAINING_DATA_MASTER	833	Normal
VALIDATION_DATA_MASTER	4372	Normal
ATTACK_DATA_MASTER	746	Attack

Python workbench was employed for evaluating the demonstration of the modified vector space on the dataset of ADFA-LD. This workbench has the ability to host concurrent algorithms of machine learning, which also are used for the selected datasets, under different sizes of terms. A total of nine algorithms for classification were selected out of six diverse groups [17]. Furthermore, the datasets were gathered and converted to representations of modified vector space for diverse sizes of terms. The term sizes one and two were chosen for this experiment. The experiment on the ADFA-LD dataset was executed in favor of the classification of binary. Out of the two labels, one was used for each trace, either attack or normal. Additionally, each of the selected algorithms was run independently according to the chosen options for data conversion. Table 9 gives a description of a number of extracted features from the ADFA-LD dataset for the size terming, via a representation of modified spaces within the vector.

In comparison with aforementioned dataset benchmarks, ADFA-LD is seen to have the most representative nature for handling present-day cyber-attacks. In addition, it offers a well-organized framework for evaluating performance and developing intrusion detection systems (IDSs). Furthermore, many system call traces are contained in ADFA-LD, some of which are retrieved based on diverse situations for the purpose of simulating real-life scenarios. Additionally, they comprise a quite concise collection of system call traces, which represents present-day vulnerabilities on system levels as well as attacks. Thus, the generating of ADFA-LD is from a local Linux server (having a Linux kernel of 2.6.38), offering advanced as well as popularly used computer systems. ADFA-LD generated attacks comprise Webshell, Adduser, Hydra-FTP, Meterpreter, and Java-Meterpreter. Each of these attacks is capable of generating up to 8–20 traces of attack. Table 10 presents the number of retrieved traces according to individual attack type. Additional information with regard to the technical aspects of the dataset is available in the literature [157].

Table 10. Attack vectors used to generate the ADFA-LD attack dataset.

Attack Type	Attack Payload Description	Vector	Trace Count
Adduser	Add new superuser using poisoned executables	Client side poisoned executable	91
Hydra-FTP	Bruteforce password guess on FTP port	FTP by Hydra	162
Hydra-SSH	Bruteforce password guess on SSH port	SSH by Hydra	176
Java-Meterpreter	Java-based Meterpreter exploit	TikiWiki vulnerability exploit	124
Meterpreter	Linux Meterpreter exploit	Client side poisoned executable	75
Webshell	Privilege escalation using C100 Webshell	PHP remote file inclusion vulnerability	118

4.2. Software and Platform

The experiments are implemented using Python 2.7 softwar; Scipy and Numpy are used for experiments and the reprocessing phase. In addition, for the aspect of visualization, the Matplotlib library was used, while the ScikitLearn library was used for machine learning [162–165].

In this work, Python programming language is used to implement machine learning techniques. Python is a great language for machine learning for numerous reasons. First, it has a clear syntax; second, with python, the manipulation of text is very easy. Thus, the majority of researchers, including those associated with private and public organizations, make use of Python, therefore leading to its rapid documentation and development. Additionally, the compilation of scientific libraries, such as Scikit Learn, an open-source library for machine learning, developed as part of a project for the Google summer code, established around 2007 by David Cournapeau [166,167]. The language used in writing this code is Python, which incorporates both the scientific and numerical libraries of python, such as NumPy, SciPy, Panda, and matplotlib. Furthermore, it also makes available provisions for effective machine learning tools, such as regression algorithms, classification, and clustering. Furthermore, it supports feature extraction approaches and offers tutorials for understanding the respective concepts [19,164].

The testing aspect of the experiment was done via Raspberry Pi 3, a small, low-powered single board computer made by the UK Raspberry Pi Foundation. The Raspberry Pi has gained popularity among computer enthusiasts and has been employed for diverse projects on robotics and IoT, thus making it a perfect choice for IoT-device-related applications. Among a lot of applications for IoT, Raspberry Pi has been deployed in the form of a Fog node, referred to in [168–170]. Thus, this experiment makes use of Raspberry Pi 3, the third generation of Raspberry Pi.

Raspberry Pi 3 comes with a low-powered yet high-speed CPU, named BCM2835 (1.2 GHz 64-bit quad-core ARMv8 CPU), with approximately 40 General Purpose Input Output (GPIO) pins, which are used in connecting sensors, 1 GB of RAM, SD card 16 GB class 10, four USB ports, Video Core IV 3D graphics core, full HDMI port, and Ethernet port.

Figure 5 presents the hardware components' location on the Model B of Raspberry Pi 3. Correspondingly, Raspberry Pi is installed with Debian Linux based OS, referred to as Raspbian, which serves as an optimization for the Raspberry Pi hardware. Furthermore, Raspberry Pi helps in evaluating the time taken in response by a classifier, based on specific IoT hardware [4]. Thus, the consumption of energy by Raspberry Pi is calculated via a commonly used technique, not significantly different from a conventional computer system. The voltage value of Raspberry Pi model 3 B is around +5.1 V. With regards to the requirements for current, a keyboard and HDMI port were connected to the Raspberry Pi, resulting in a 400 mA electrical current requirement. The power consumption is yielded by the combination of the products of both the Voltage (V) and the Current (I), which is then multiplied by the CPU time (t) to calculate the consumed energy [21,171].

$$P\ (Power) = I * V \tag{18}$$

$$\text{Energy Consumption} = P * t \tag{19}$$

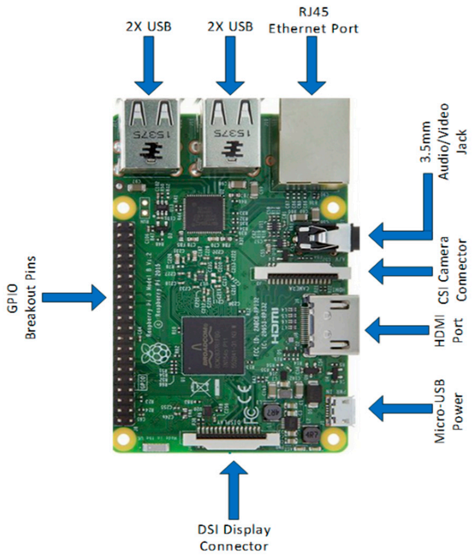


Figure 5. The hardware components of Raspberry Pi 3 Model B [21].

5. Results and Discussion

The experiment conducted in this paper was aimed at addressing the need for the development of an HIDS anomaly based on light weight, which is responsive to most recent attacks via the usage of ADFA-LD datasets, to employ in IoT-based Fog computing. Thus, the research tries to address the following issues:

- What techniques that support HIDS anomaly based on light weight can be developed in IoT-based Fog Computing?
- What Machine Learning computable devices with limited resources can help create lightweight HIDS in Fog Computing?
- What feature extraction and selection approaches for pre-processing data can produce high performance?
- Which classifiers can give high accuracy?
- Which classifiers can give low CPU usage and energy consumption in testing data?

Many algorithms were employed for the purpose of achieving adequate results using these techniques. Thus, the study proposes a general viewpoint for the detection of anomalies via machine learning techniques. The conducted experiments also proved that supervised learning approaches perform far better than unsupervised approaches, provided there are no unknown attacks in the test data. Previous studies revealed that out of numerous supervised approaches, the best performance can be achieved by RF, SVM-RBF, and MLP; however, there are differences with regard to their individual capabilities to detect attack class effectively. In order to compare with other machine learning methods, contrast experiments are designed at the same time. This is done in the binary classification experiments, for using more loading instances for lightweight evaluation.

The first step in this experiment involves the computation of 1-gram and 2-gram, for the system call traces, whereby selections for the top 60 and 80 features for 1-gram, followed by the top 80 and 120 features for the 2-gram, were made on the basis of the value of their information using MI, which were combined to train the model. Comprehensively, the numerals 60, 80, and 120 were selected for the reduction of the data size as well as data representation [21].

Furthermore, the classifiers were compared with regard to their accuracy, recall, F1-measure, FPR, ROC Curve, and AUC. In addition, a comparison was made of the testing time for CPU usage and energy consumption between the different classifiers using 1-gram and 2-gram, which are the maximum features on the Raspberry Pi platform.

This work did not focus on memory usage testing time, because after the collection of results and analysis, it was discovered that consumers for each classifier did not exceed 350 Bytes, as shown in Figure 6, for the KNN classifier. Therefore, it is not considered important in the Raspberry Pi platform.

5.1. Performance of the Feature Extraction and Feature Selection

In this section, the performance of the feature extraction and feature selection methods are explored in terms of CPU time and memory usage. The 1-gram and 2-gram transformation modified vector spaces (feature extraction) are tested with two feature selection methods: Principle Component Analysis (PCA) and Mutual Information (MI) (filter method). Based on the test, the best feature selection method in term of its “light weight” was selected for implementation with the MLP classification method. Figure 7a,b show the CPU time and memory usage, respectively. Based on this figure, we can notice that (MI) outperforms PCA, which on the contrary has a lesser performance for CPU time with the consumption of less memory.

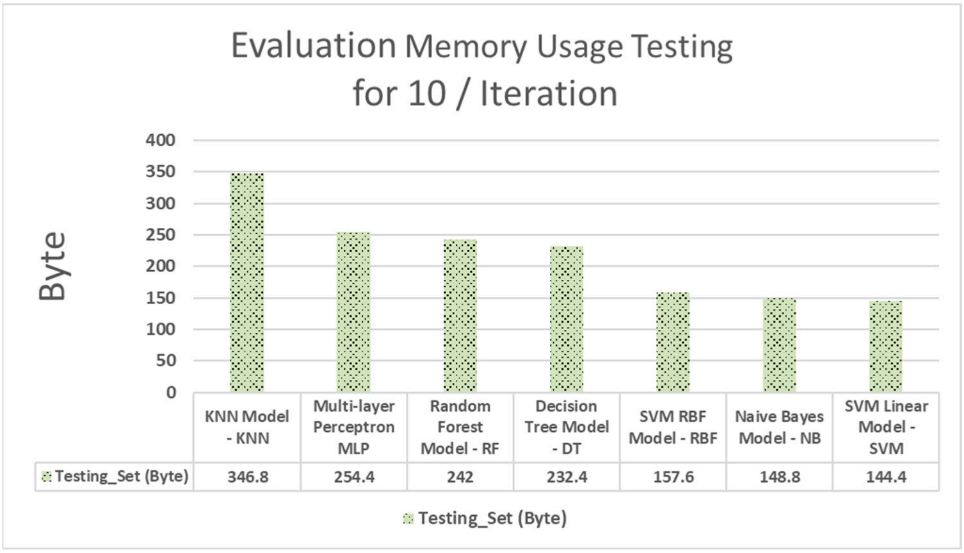
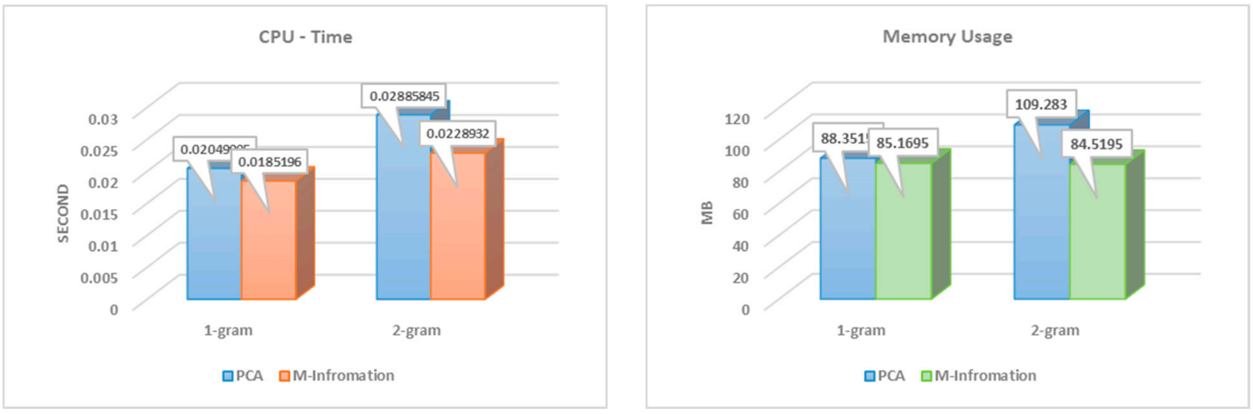


Figure 6. Comparison testing memory usage between different classifiers using 1-gram and 2-gram.



(**a**) (**b**)

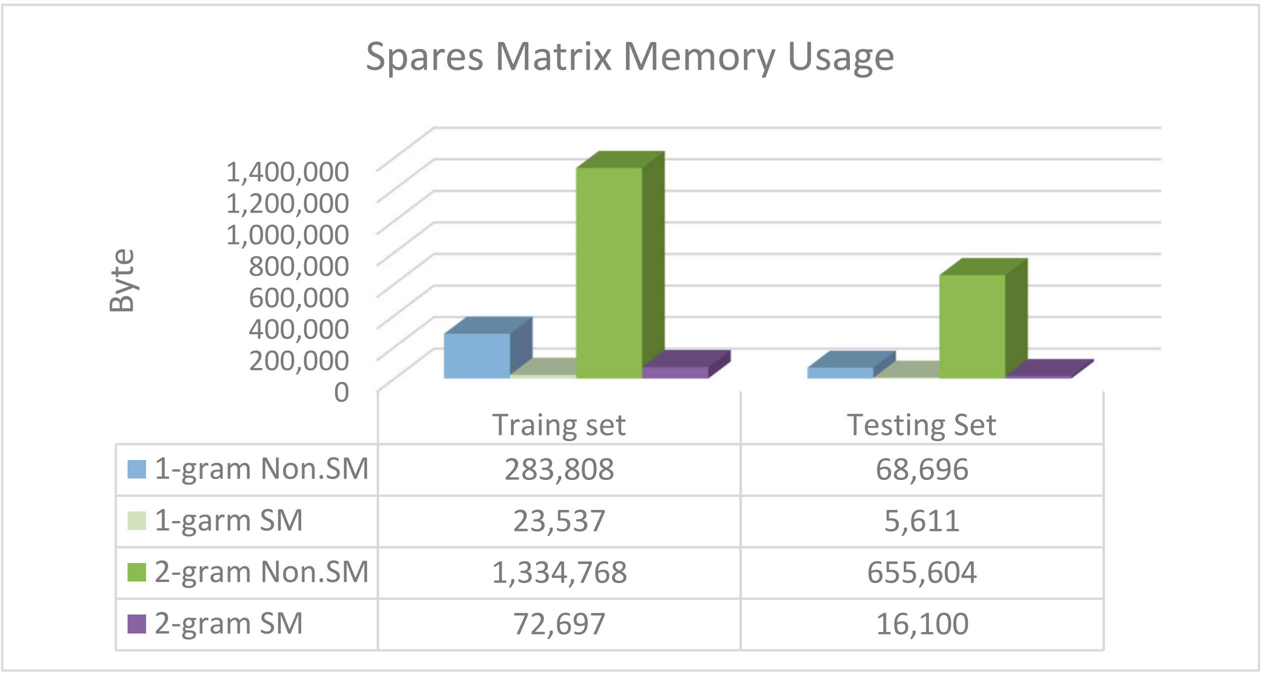
Figure 7. Feature selection comparison between PCA and MI using 1-gram and 2-gram: (**a**) CPU time performance; (**b**) memory usage performance.

The following stage, traces in the system call, makes use of a sparse matrix formatting technique from Python to ensure reduction in the size of the memory. This approach to formatting is of much relevance, as it was able to reduce a reasonable amount of data in the matrix, as 98.2% of the 2-gram matrix entries were zero values. Figure 8 depicts the matrix's compressed form, which is around 4% of the initial dense matrix memory space.

5.2. *Performance of the Classification Methods*

Subsequent to the normalization of the processed data for more suitable performance, Figure 9a presents a comparison of recall, F1-measure, and accuracy between the different classifiers, using 80 features for 1-gram and 120 features for 2-gram, thus selecting the highest number of features to provide more efficiency and to help in determining the classifiers of good performance as well those fit for lightweight techniques.

Based on Figure 9a, it is noticeable that MLP outperforms the other classifiers, in accordance with their accuracy, recall, and F1-measure. Correspondingly, the Naive Bayes and Decision Tree Models achieved the worst results for these metrics.



	Traing set	Testing Set
🟦 1-gram Non.SM	283,808	68,696
⬜ 1-garm SM	23,537	5,611
🟩 2-gram Non.SM	1,334,768	655,604
🟪 2-gram SM	72,697	16,100

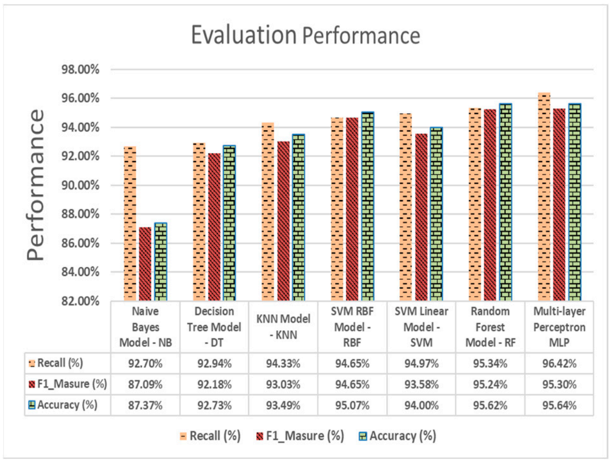
Figure 8. Sparse matrix memory usage.

In terms of FPR, Figure 9b reveals that RF, SVM-RBF, and MLP models outperform by approximately 5% compared to others, which achieved 17.13%. It is also noticeable that RF and SVM-RBF models achieved higher FPR than MLP, up to a difference of nearly 0.08% FPR. This value is meaningful with reference to the rate of detection; however, it is observed in Figure 9c, which shows a comparison between ROC curves, that it is a graph of true positive rate against FPR. It is also noticeable that MLP outperforms the other classifiers, as it has the highest range over curve. In addition, the AUC value is equivalent to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. MLP showed the highest value (95.7%), and this represents that it was able to achieve a higher performance than the other classifiers.

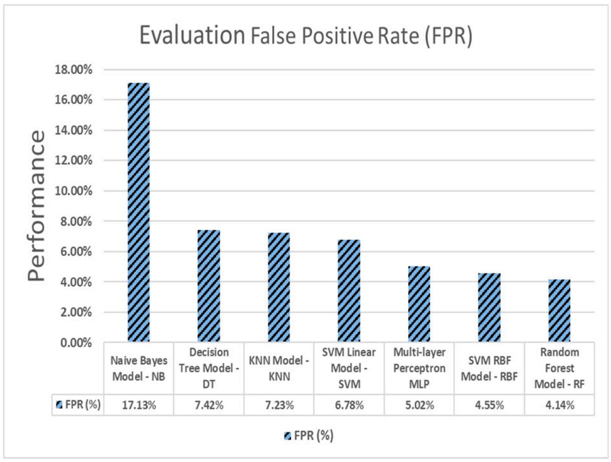
Furthermore, Figure 9d shows a comparison of CPU usage for the testing time between the different classifiers. Based on this figure, it is noticeable that Decision Tree (DT) and MLP outperform the other classifiers in terms of the testing time. However, SVM-RBF, RF, and KNN models had the worst values for CPU usage for the testing time.

In terms of energy consumption, Figure 9e depicts the Decision Tree (DT), whereby MLP models outperformed by up to around (0.008) Joules the others models, including the KNN model (4.3 Joules). In other words, the MLP outperforms the other classifiers in terms of accuracy, recall, F1-measure, and FPR and achieved low values for CPU usage and energy consumption.

According to the above results, the best performance is given by the MLP model. Furthermore, MLP also did well with regard to the amount of CPU time and consumption of energy achieved for all instances during the testing phase, as compared to other classifiers. There were only a few cases where simple performances were provided by these techniques. However, observing the MLP model, it is obvious that it achieved a good performance and is also compatible with lightweight devices.

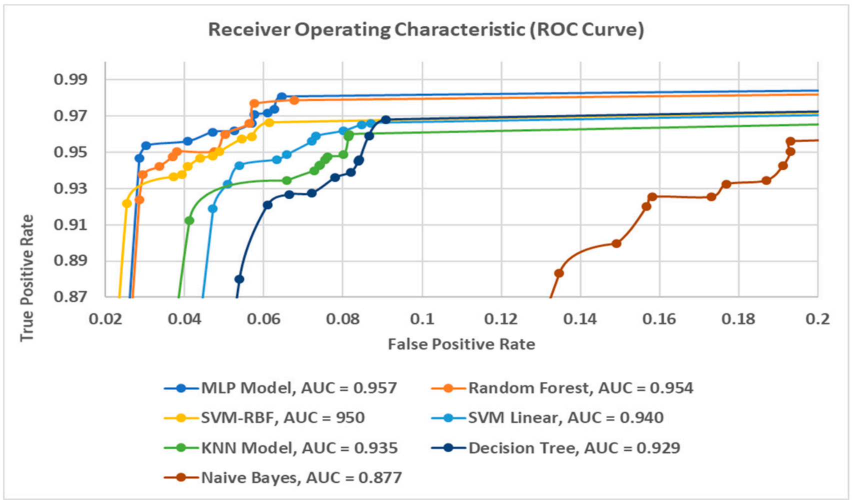


	Naive Bayes Model - NB	Decision Tree Model - DT	KNN Model - KNN	SVM RBF Model - RBF	SVM Linear Model - SVM	Random Forest Model - RF	Multi-layer Perceptron MLP
▨ Recall (%)	92.70%	92.94%	94.33%	94.65%	94.97%	95.34%	96.42%
▨ F1_Masure (%)	87.09%	92.18%	93.03%	94.65%	93.58%	95.24%	95.30%
▨ Accuracy (%)	87.37%	92.73%	93.49%	95.07%	94.00%	95.62%	95.64%

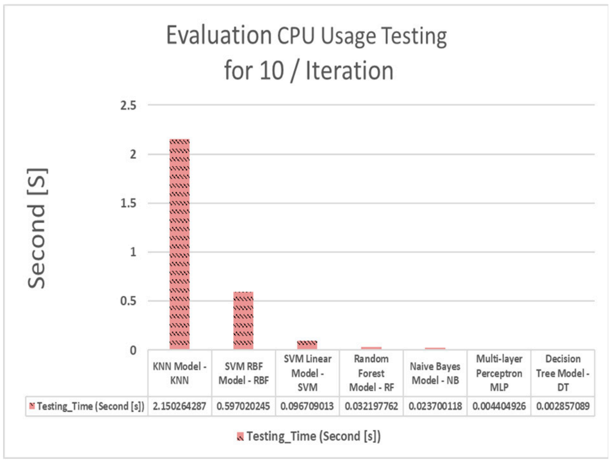


	Naive Bayes Model - NB	Decision Tree Model - DT	KNN Model - KNN	SVM Linear Model - SVM	Multi-layer Perceptron MLP	SVM RBF Model - RBF	Random Forest Model - RF
▨ FPR (%)	17.13%	7.42%	7.23%	6.78%	5.02%	4.55%	4.14%

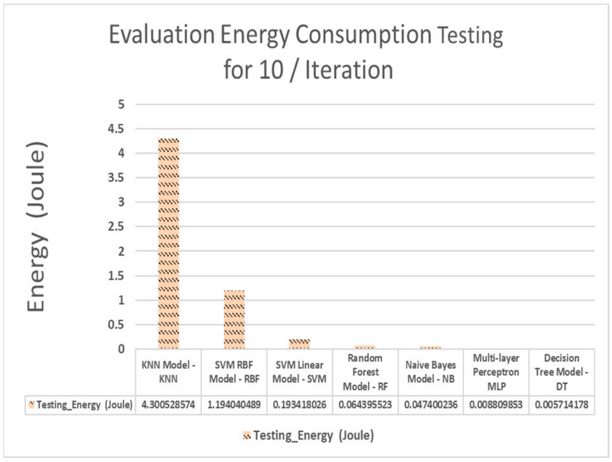
(**a**) (**b**)



(**c**)



	KNN Model - KNN	SVM RBF Model - RBF	SVM Linear Model - SVM	Random Forest Model - RF	Naive Bayes Model - NB	Multi-layer Perceptron MLP	Decision Tree Model - DT
▨ Testing_Time (Second [s])	2.150264287	0.597020245	0.096709013	0.032197762	0.023700118	0.004404926	0.002857089



	KNN Model - KNN	SVM RBF Model - RBF	SVM Linear Model - SVM	Random Forest Model - RF	Naive Bayes Model - NB	Multi-layer Perceptron MLP	Decision Tree Model - DT
▨ Testing_Energy (Joule)	4.300528574	1.194040489	0.193418026	0.064395523	0.047400236	0.008809853	0.005714178

(**d**) (**e**)

Figure 9. Performance of the classification methods: (**a**) comparison of recall, F1-Measure, and accuracy between different classifiers; (**b**) comparison of FPR between different classifiers using 1-gram and 2-gram; (**c**) comparison of ROC between the different classifiers using 1-gram and 2-gram; (**d**) comparison of CPU usage for testing time between the different classifiers using 1-gram and 2-gram; (**e**) comparison of energy consumption for testing time between the different classifiers using 1-gram and 2-gram.

5.3. Performance on the Raspberry Pi

Raspberry Pi has a significantly slower speed as compared to other regularly used computers; thus its limited resources for computing makes it an appropriate platform to test the performance of the model. The model's performance for this experiment was evaluated via a Raspberry Pi 3 of the Model Type B. The performance showed the median for each of the 10 iterations tested. Further observation showed that there were 350 instances (i.e., "normal" = 200, "attack" = 150) for the experimental testing of the test set.

Furthermore, the optimization of the CPU's energy and time could be achieved via the utilization of Fog Computing's distributive nature. It was observed that the average time used in testing was around 0.004 s, thus indicating a reasonable delay, as applicable to many IoT-inclined applications, and the energy consumption rate was 0.008 Joules.

Based on the benchmark, using the ADFA-LD training set and testing set, the experimental results show that for binary classification, the intrusion detection model of MLP-IDS through the training set and testing set has higher accuracy than other machine learning methods and maintains a high recall, F1-measure, and accuracy rate as well as the highest ROC curve with a high value of AUC, even in the case of FPR. The FPR rate was low for the other models, such as SVM-RBF and RF; however, though they outperformed, they had higher CPU time and energy consumption for the testing process and thus were inappropriate for the lightweight model. Consequently, the DT model outperformed the MLP model with respect to CPU time and energy consumption for the testing process, but maintained a lower rate of recall, F1-measure, and accuracy as well as a low ROC curve and low value of AUC; however, it also achieved a high value of FPR as compared to the other machine learning methods. Of course, the model proposed from this experiment will reduce the time and energy consumption for the testing process more effectively, as it boasts a higher accuracy than the other machine learning methods.

5.4. Performance of the MLP Methods (Parameter)

This work employed an ANN approach to serve as the computational model. This is due to its influence on biological neural network characteristics for incorporating intelligence into the proposed methods. Thus, a particular type of ANN is represented in the form of a direct graph, for the purpose of transmitting diverse information from the system over the edge and across nodes, without the formation of a cycle. To achieve this, a MLP model was adopted. However, in order to achieve an outstanding performance, it is important to find an optimal parameter when modeling an MLP [162,165].

Fully connected layer: This layer is referred to as a layer that is fully connected due to the connectivity of the layer's unit with all other units around the succeeding layer. From a general perspective, high dimension mapping can be easily achieved via layers that are fully connected. This will result in more accuracy of the output. The function for non-linear activation used in this experiment is ReLU.

Batch Normalization and Regularization: Dropout (0.01) and Batch Normalization were employed amidst layers that were fully connected, for the purpose of over-fitting obviation as well as to ensure a speed-up of the model training of the Deep Neural Network (DNN). Regarding the alternative architectures, it was discovered that DNNs were able to over-fit the data used for training very easily, without the need for regularization, including situations where training was conducted for a large number samples.

Classification: This is the last layer, and it is fully connected as it makes use of the activation function from sigmoid for the classification of binary, as well as the activation function from softmax for the classification of multi-classes. Thus, the definition of the function for prediction loss in sigmoid is achieved via binary cross entropy; moreover, the definition for the prediction loss of softmax is achieved via categorical cross entropy as follows:

The estimation for the prediction loss of Binary classification is achieved via binary cross entropy, given by

$$\text{loss}(pd,\ ed) = -\frac{1}{\text{N}} \sum_{\text{i}=1}^{\text{N}} [ed_{\mathbf{i}} \log pd_i + (1 - ed_i) \log(1 - pd_i)] \tag{20}$$

where *pd* denotes a vector of the predicted probability for all the testing dataset samples; *ed* denotes a vector of expected class label, for which the values lie between 0 and 1

The prediction loss for Multi-class classification is estimated using categorical cross entropy, given by

$$loss(pd,\ ed) = -\sum_{x} pd(x) \log(ed(x)) \tag{21}$$

where *ed* represents the true probability distribution; *pd* denotes the predicted probability distribution. Furthermore, in this research, adam was used as an optimizer for ensuring the minimization of the loss of categorical cross entropy as well as binary cross entropy [129,172].

As MLPs are parameterized, as shown in Table 11, the performance is dependent on the optimized parameters. Determining the parameter that is optimized for both MLP network topologies and network parameters was achieved via the ADF-LD dataset. For identifying which parameter is suitable for MLP, this research employed the use of an architecture of medium size, for the experimentations within specific hidden units, rate of learning, and the functions of activation, as follows: default (Activation = 'relu', Solver = 'adam', Alpha = 0.0001), a small node size, and a single hidden layer. The architecture of MLP contains three layers. The first is the input layer, the second is the hidden layer or fully connected layer, and the third is the output later.

Table 11. MLP parameters (activation, solver, alpha).

Function	Parameters
Activation: for the hidden layer	'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
	'logistic', the logistic sigmoid function, returns f(x) = 1/(1 + exp(2212x))
	'tanh', the hyperbolic tan function, returns f(x) = tanh(x)
	default 'relu', the rectified linear unit function, returns f(x) = max(0, x)
Solver: for weight optimization	'lbfgs' is an optimizer in the family of quasi-Newton methods
	'sgd' refers to stochastic gradient descent
	default 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Ba
Alpha: L2 penalty (regularization term) parameter	Alpha float, optional, default 0.0001
	Both MLP Regressor and MLP Classifier use parameter alpha for regularization (L2 regularization) term, which helps in avoiding overfitting by penalizing weights with large magnitudes

The advantages of Multi-layer Perceptron (MLP) are that it is able to (1) learn non-linear models and (2) learn models in real-time (online learning) via partial fit.

The disadvantages of Multi-Layer Perceptron (MLP) include the following: (1) MLP has hidden layers consisting of a non-convex loss function and comprising the existence of more than a single local minimum. Thus, accuracy of diverse validations can occur due to the initialization of various random weights. (2) MLP requires the tuning of several hyperparameters, including numerous hidden neurons, iterations, and layers. (1) MLP has a sensitivity to feature scaling [129].

The evaluation performance for MLP parameters (Recall, F1-Measure, Accuracy, FPR) is illustrated in Figure 10. The evaluation of light weight is based on the following parameters, as illustrated in Figure 11: Evaluation of CPU Usage MLP Parameter; Evaluation of Memory Usage MLP Parameter; Evaluation of Energy (Consumption). This process includes exploring all factors that help to reach the goals of achieving a light weight. This study obtained performance under the default values of the MLP model that were deemed the most appropriate; however, some parameters were used to help provide the light weight. The performance from comparing both alpha values at an equal point of 10.0 shows that there is an improvement in both the processor usage and energy consumption. in addition, when using activation "identity", it contributes to the support of light weight, while the performance of both "identity" and the alpha 10.0 is poor.

5.5. Performance of the MLP Methods (N-gram)

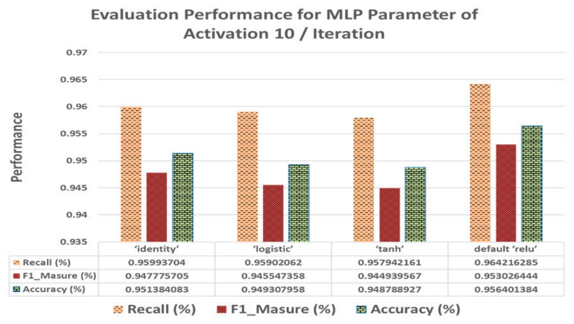
The experiment is performed via combining the top 60 and 80 for the 1-gram, whereas for the 2-gram, it combined the top 80 and 120, with several numbers of nodes at the hidden layer, after which the results underwent evaluation. Figure 12a illustrates the model performance based on accuracy, via the use of combining four top features and their N-grams, accordingly. From the results, the best accuracy value was achieved for 4 nodes at the hidden layer, which used 80 1-gram and 120 2-gram.

With regards to the recall values, as depicted in Figure 12b, the best value achieved around 97%, which was possible under three cases, namely the 2 node model via 80 1-gram and 80 2-gram; the 4 node model via 60 1-gram and 120 2-gram; the 6 node model via 60 1-gram, and 120 2-gram. The F1 measure values for total combinations of 1-gram as well as 2-gram are illustrated in Figure 12c. The best value for the F1-measure was achieved on the 6 node model via 80 1-gram and 80 2-gram.

In terms of FPR, Figure 12d shows that when increasing the number of nodes and the number of combinations of 1-gram and 2-gram, the models outperform. Figure 12e shows that the best Area Under Curve (AUC) value achieved during the experiment was for 6 nodes at the hidden layer, with 80 1-gram and 120 2-gram, which achieve a value of 94.95%.

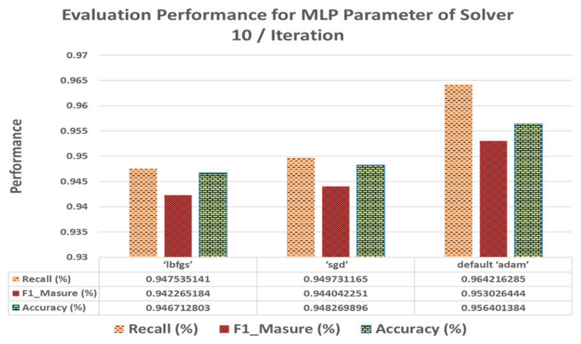
In terms of CPU usage and energy consumption, Figure 12f,g depicts the low values. When using a small number of nodes and small numbers for all combinations of 1-gram and 2-gram, high values of CPU usage and energy consumption result. When increasing the number of nodes and the number of combinations of 1-gram and 2-gram, the best consumption value is achieved on the 2 node model using 60 1-gram and 80 2-gram, which outperformed by up to around (0.0025) seconds and by a (0.005) Joule energy consumption rate.

Furthermore, comparisons were made among all performances, such as for accuracy, recall, F1-measure, FPR, AUC, CPU usage, and energy consumption of the models via diverse nodes. These comparisons revealed that the overall outstanding performance was attained at the point where 2 or 4 nodes were used at the hidden layer. However, it is worth noting that the 5 and 6 node models also produced good values for the recall rate, similar to that of the 4 node model. Thus, conclusions can be drawn from the above analysis as to the fact that the model can perform well, provided the hidden layer contains a minute number of nodes. The use of 4 nodes at the hidden layer is adequate for the majority of applications. Overall, it can be postulated that the 4 node model of a hidden layer performs far better than models with a greater number of nodes, working on a mix of several 1-gram and 2-gram. The implication of this is that the addition of more nodes will yield an insignificant improvement in the performance of the model.



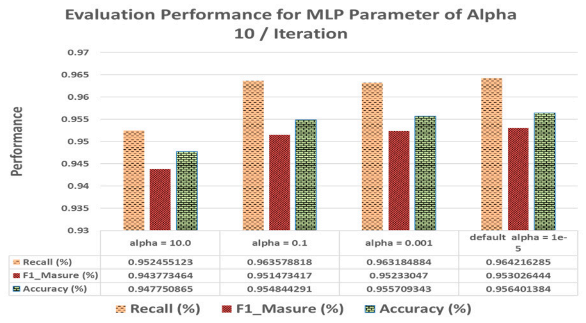
	'identity'	'logistic'	'tanh'	default 'relu'
▨ Recall (%)	0.95993704	0.95902062	0.957942161	0.964216285
▨ F1_Masure (%)	0.947775705	0.945547358	0.944939567	0.953026444
▨ Accuracy (%)	0.951384083	0.949307958	0.948788927	0.956401384

(**a**)



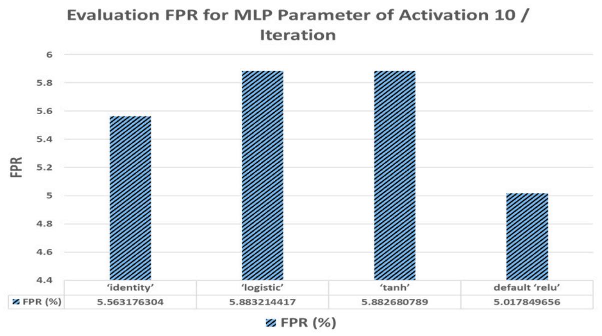
	'lbfgs'	'sgd'	default 'adam'
▨ Recall (%)	0.947535141	0.949731165	0.964216285
▨ F1_Masure (%)	0.942265184	0.944042251	0.953026444
▨ Accuracy (%)	0.946712803	0.948269896	0.956401384

(**b**)



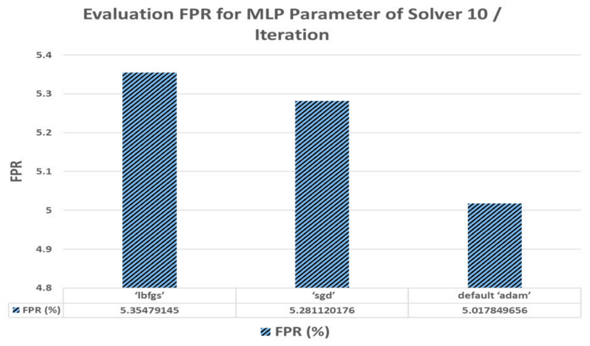
	alpha = 10.0	alpha = 0.1	alpha = 0.001	default alpha = 1e-5
▨ Recall (%)	0.952455123	0.963578818	0.963184884	0.964216285
▨ F1_Masure (%)	0.943773464	0.951473417	0.95233047	0.953026444
▨ Accuracy (%)	0.947750865	0.954844291	0.955709343	0.956401384

(**c**)



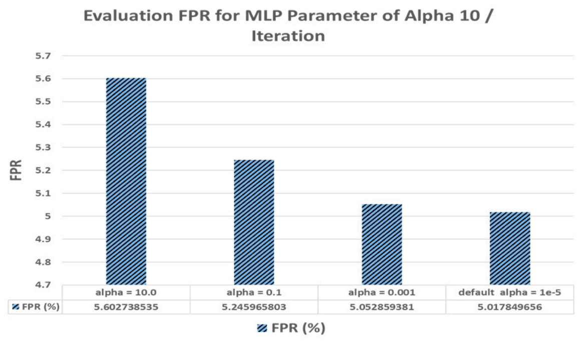
	'identity'	'logistic'	'tanh'	default 'relu'
▨ FPR (%)	5.563176304	5.883214417	5.882680789	5.017849656

(**d**)



	'lbfgs'	'sgd'	default 'adam'
▨ FPR (%)	5.35479145	5.281120176	5.017849656

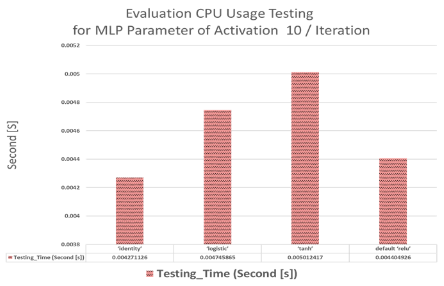
(**e**)



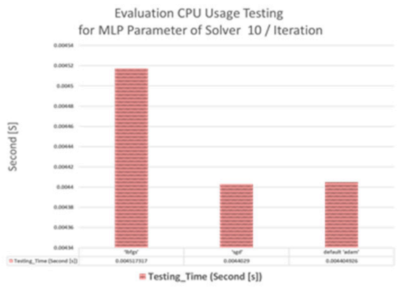
	alpha = 10.0	alpha = 0.1	alpha = 0.001	default alpha = 1e-5
▨ FPR (%)	5.602738535	5.245965803	5.052859381	5.017849656

(**f**)

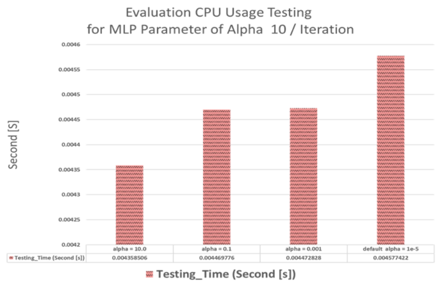
Figure 10. Performance of MLP parameters: (**a**) Evaluation Performance of Activation; (**b**) Evaluation Performance of Solver; (**c**) Evaluation Performance of Alpha; (**d**) Evaluation FPR for Activation; (**e**) Evaluation FPR of Solver; (**f**) Evaluation FPR of Alpha.



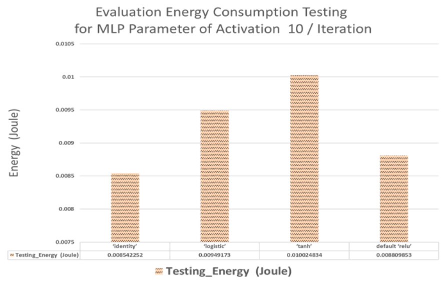
(**a**)



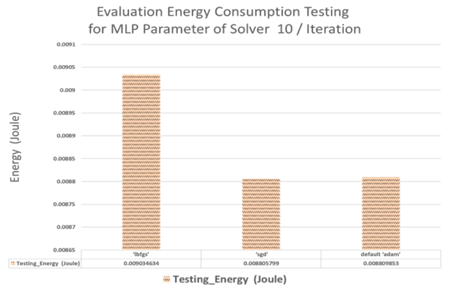
(**b**)



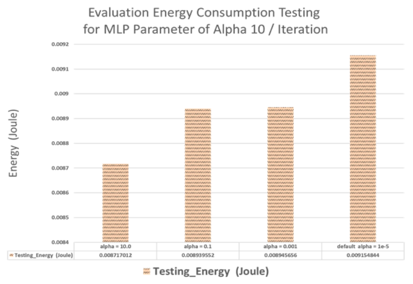
(**c**)



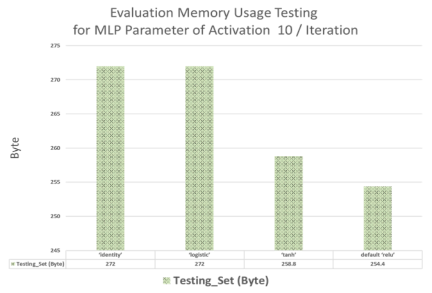
(**d**)



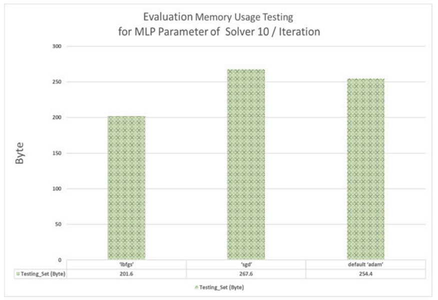
(**e**)



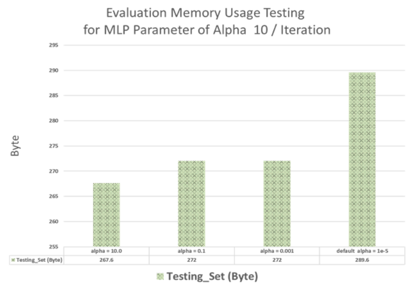
(**f**)



(**g**)

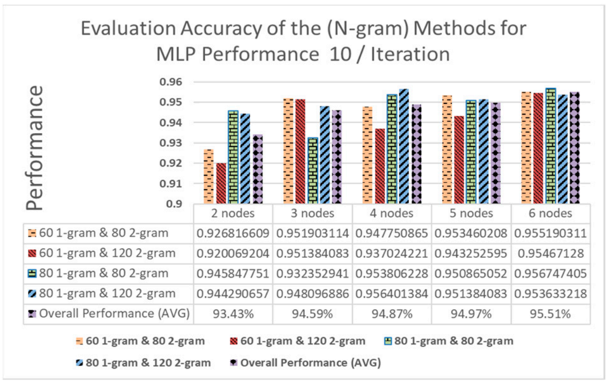


(**h**)



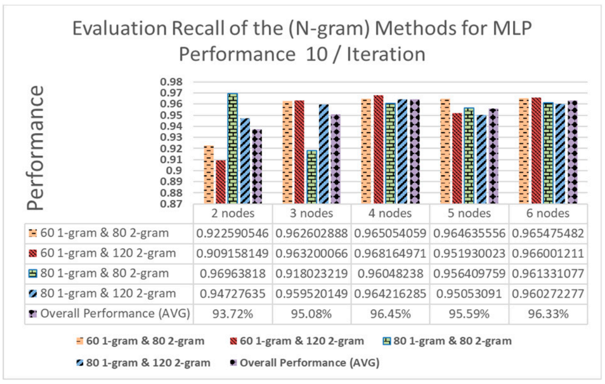
(**i**)

Figure 11. Evaluation lightweight of MLP parameter 10/Iteration: (**a**) Evaluation CPU Usage Testing for Activation; (**b**) Evaluation CPU Usage Testing for Solver; (**c**) Evaluation CPU Usage Testing for Alpha; (**d**) Evaluation Energy Consumption Testing for Activation; (**e**) Evaluation Energy Consumption Testing for Solver; (**f**) Evaluation Energy Consumption Testing for Alpha; (**g**) Evaluation Memory Usage Testing for Activation; (**h**) Evaluation Memory Usage Testing for Solver; (**i**) Evaluation Memory Usage Testing for Alpha.



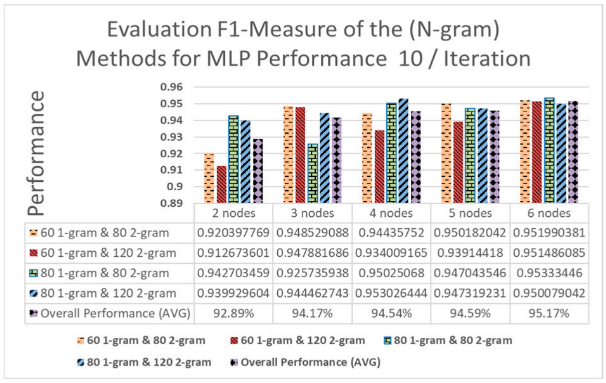
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	0.926816609	0.951903114	0.947750865	0.953460208	0.955190311
60 1-gram & 120 2-gram	0.920069204	0.951384083	0.937024221	0.943252595	0.95467128
80 1-gram & 80 2-gram	0.945847751	0.932352941	0.953806228	0.950865052	0.956747405
80 1-gram & 120 2-gram	0.944290657	0.948096886	0.956401384	0.951384083	0.953633218
Overall Performance (AVG)	93.43%	94.59%	94.87%	94.97%	95.51%

(**a**)



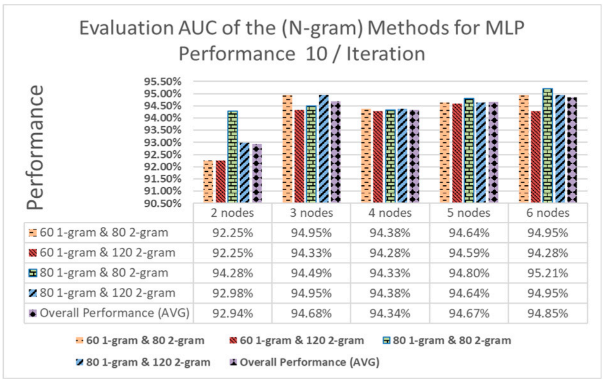
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	0.922590546	0.962602888	0.965054059	0.964635556	0.965475482
60 1-gram & 120 2-gram	0.909158149	0.963200066	0.968164971	0.951930023	0.966001211
80 1-gram & 80 2-gram	0.96963818	0.918023219	0.96048238	0.956409759	0.961331077
80 1-gram & 120 2-gram	0.94727635	0.959520149	0.964216285	0.95053091	0.960272277
Overall Performance (AVG)	93.72%	95.08%	96.45%	95.59%	96.33%

(**b**)



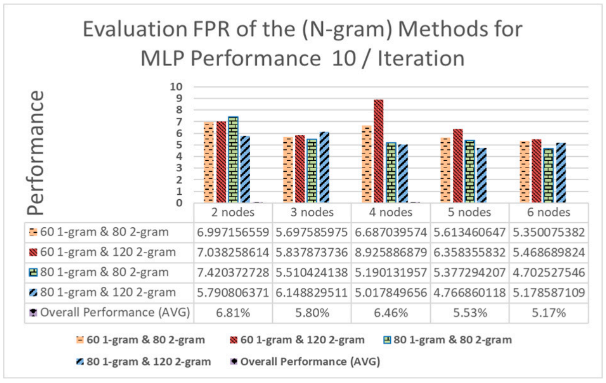
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	0.920397769	0.948529088	0.94435752	0.950182042	0.951990381
60 1-gram & 120 2-gram	0.912673601	0.947881686	0.934009165	0.93914418	0.951486085
80 1-gram & 80 2-gram	0.942703459	0.925735938	0.95025068	0.947043546	0.95333446
80 1-gram & 120 2-gram	0.939929604	0.944462743	0.953026444	0.947319231	0.950079042
Overall Performance (AVG)	92.89%	94.17%	94.54%	94.59%	95.17%

(**c**)



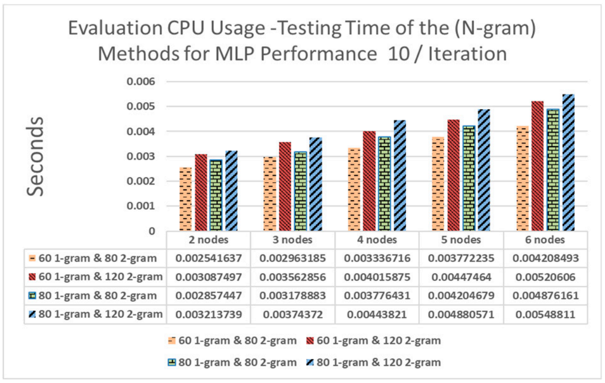
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	92.25%	94.95%	94.38%	94.64%	94.95%
60 1-gram & 120 2-gram	92.25%	94.33%	94.28%	94.59%	94.28%
80 1-gram & 80 2-gram	94.28%	94.49%	94.33%	94.80%	95.21%
80 1-gram & 120 2-gram	92.98%	94.95%	94.38%	94.64%	94.95%
Overall Performance (AVG)	92.94%	94.68%	94.34%	94.67%	94.85%

(**d**)



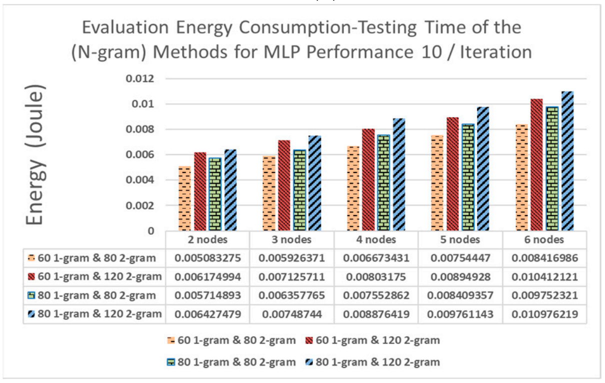
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	6.997156559	5.697585975	6.687039574	5.613460647	5.350075382
60 1-gram & 120 2-gram	7.038258614	5.837873736	8.925886879	6.358355832	5.468689824
80 1-gram & 80 2-gram	7.420372728	5.510424138	5.190131957	5.377294207	4.702527546
80 1-gram & 120 2-gram	5.790806371	6.148829511	5.017849656	4.766860118	5.178587109
Overall Performance (AVG)	6.81%	5.80%	6.46%	5.53%	5.17%

(**e**)



	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	0.002541637	0.002963185	0.003336716	0.003772235	0.004208493
60 1-gram & 120 2-gram	0.003087497	0.003562856	0.004015875	0.00447464	0.00520606
80 1-gram & 80 2-gram	0.002857447	0.003178883	0.003776431	0.004204679	0.004876161
80 1-gram & 120 2-gram	0.003213739	0.00374372	0.00443821	0.004880571	0.00548811

(**f**)



	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	0.005083275	0.005926371	0.006673431	0.00754447	0.008416986
60 1-gram & 120 2-gram	0.006174994	0.007125711	0.00803175	0.00894928	0.010412121
80 1-gram & 80 2-gram	0.005714893	0.006357765	0.007552862	0.008409357	0.009752321
80 1-gram & 120 2-gram	0.006427479	0.00748744	0.008876419	0.009761143	0.010976219

(**g**)

Figure 12. The performance of the model with respect to value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations by 10/Iteration: (**a**) accuracy, (**b**) recall, (**c**) F1-measure, (**d**) FPR, (**e**) AUC, (**f**) CPU usage—testing, (**g**) energy consumption—testing.

6. Conclusions

In this article, a novel and intelligent architecture of an IDS is presented. To address the aforementioned limitations of current systems, the IDS presented here includes the main function, type, and normal behavior profile of each IoT device connected to the network; the type of attack deployed is classified. Evaluation of the performance of applying a supervised machine learning approach to automate each function demonstrates that the proposed architecture can successfully distinguish between IoT devices on the network and whether network activity is malicious or benign, and it can detect which attack was deployed on which device connected to the network automatically. In addition to the experimental results, this study provides resources that can further support research into automating IoT-based cyber-attack detection.

In this work, a methodology was presented to develop a lightweight IDS-based machine learning model that can be used to identify potential attacks against IoT security, considering their impacts and the ways of mitigating as well as recovering from these attacks. In this work, the Fog Computing concepts and its security challenges were discussed.

- The proposed method is tested using the ADFA-LD dataset for anomaly HIDS.
- Spares Matrix (Memory Usage)—the compressed form of the matrix is around 4% of memory space of the original dense matrix.
- Based on these results, we can notice that Mutual Information (MI) outperforms PCA, with less CPU time and with less memory consumption.
- The research found that MLP outperforms the other classifiers in terms of 96% accuracy, 97% recall, 96% F1-Measure, 5% False Positive Rate (FPR), the highest curve of ROC, and 96% Area Under The Curve (AUC). It also achieved low CPU time usage of 4.404 ms and a low energy consumption of 8.809 mj.
- This study obtained performance under the default values of the MLP model that were deemed the most appropriate; however, some parameters were used to help provide the light weight. The performance from comparing both alpha values at an equal point of 10.0 shows that there is an improvement in both the processor usage and energy consumption. in addition, when using activation "identity", it contributes to the support of light weight, while the performance of both "identity" and the alpha 10.0 is poor.
- The test is executed with 2, 3, 4, 5, and 6 nodes of the MLP single hidden layer. From the experiment, we found that 2 nodes of the MLP single hidden layer with 80 1-gram and 80 2-gram were able to achieve 95% accuracy, 97% recall rate, 94% F1-measure, and FPR of 7%, and an AUC of 95%. This is comparable to the higher number of nodes (i.e., 3 nodes to 6 nodes), which give similar results. Since the 2 node model has lower complexity, and there is no significant advantage when a higher number of nodes is implemented, the 2 node implementation is suggested for the Fog device.
- Raspberry Pi, which acts as the Fog device with its limited computational resources, is a suitable platform for testing the model performance. The performance that shows the median for the CPU time usage is approximately 5 ms, and the energy consumption is approximatively 9 mj. The testing time could be optimized by utilizing the distributed nature of Fog Computing.

IoT end devices need novel solutions to protect their privacy. The current paper employed the application of the concepts of "A Lightweight Perceptron-based Intrusion Detection System" using Fog Computing to build a distributed lightweight security solution with high lifespan for IoT security.

7. Future Work

There is a wide gap in the area of IoT device capacity as well as intelligence, as these IoT systems are known to produce large heterogeneous data every second. Thus, efficient approaches for managing huge data produced by IoT systems is a tentative drive for future work. The application of technologies like Fog Computing, big data, and the cloud can be useful in ensuring a large exchange of information across networks. Furthermore, there is hope with regards to efficiency of the technology in being able to manage huge heterogeneous data in an efficient and secure manner. The concept of a holistic architecture for IDS in the IoT has begun to be explored. Still in its infancy, this trend will grow, with continued evaluation of IDS implementation to improve security of alert traffic and management and further development of applications such as alert correlation and autonomic management systems.

Author Contributions: Conceptualization, B.S.K.; M.Y.I.I. and A.W.A.W.; methodology, B.S.K.; M.Y.I.I.; software, A.A.I.; validation, B.S.K., A.W.A.W. and M.A.H.; formal analysis, M.Y.I.I.; A.A.I.; investigation, B.S.K.; M.Y.I.I. and A.W.A.W.; data curation, B.S.K.; M.Y.I.I. and A.A.I.; writing—original draft preparation, B.S.K., M.Y.I.I. and M.A.H.; writing—review and editing, M.A.A.; and H.A.S.; visualization, B.S.K. and M.A.A.; supervision, M.Y.I.I. and A.W.A.W.; M.A.H. All authors provided critical feedback and helped shape the research, analysis and manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Universiti Malaya Impact Oriented Interdisciplinary Research Grant (IIRG008A-19IISS).

Data Availability Statement: This study made use of the ADFA-LD dataset which can be retrieved from: https://research.unsw.edu.au/projects/adfa-ids-datasets (accessed on 2 February 2017) [158].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khan, S.; Parkinson, S.; Qin, Y. Fog computing security: A review of current applications and security solutions. *J. Cloud Comput.* **2017**, *6*, 19. [CrossRef]
2. Sfar, A.R.; Natalizio, E.; Challal, Y.; Chtourou, Z. A roadmap for security challenges in the Internet of Things. *Digit. Commun. Netw.* **2018**, *4*, 118–137. [CrossRef]
3. Sun, F.; Guo, G. Research of Immunity-based Anomaly Intrusion Detection and Its Application for Security Evaluation of E-government Affair Systems. *Int. J. Digit. Content Technol. Its Appl.* **2012**, *6*, 429.
4. Verma, A.; Ranga, V. Machine learning based intrusion detection systems for IoT applications. *Wirel. Pers. Commun.* **2020**, *111*, 2287–2310. [CrossRef]
5. Perez, D.; Astor, M.A.; Abreu, D.P.; Scalise, E. Intrusion detection in computer networks using hybrid machine learning techniques. In Proceedings of the 2017 XLIII Latin American Computer Conference (CLEI), Cordoba, Argentina, 4–8 September 2017; pp. 1–10.
6. Jan, S.U.; Ahmed, S.; Shakhov, V.; Koo, I. Toward a lightweight intrusion detection system for the internet of things. *IEEE Access* **2019**, *7*, 42450–42471. [CrossRef]
7. Gao, S.; Thamilarasu, G. Machine-learning classifiers for security in connected medical devices. In Proceedings of the 2017 26th International Conference on Computer Communication and Networks, Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–5.
8. Tian, Q.; Li, J.; Liu, H. A Method for Guaranteeing Wireless Communication Based on a Combination of Deep and Shallow Learning. *IEEE Access* **2019**, *7*, 38688–38695. [CrossRef]
9. Alharbi, S.; Rodriguez, P.; Maharaja, R.; Iyer, P.; Bose, N.; Ye, Z. FOCUS: A fog computing-based security system for the Internet of Things. In Proceedings of the 2018 15th IEEE Annual Consumer Communications & Networking Conference, Las Vegas, NV, USA, 12–15 January 2018; pp. 1–5.
10. Arrington, B.; Barnett, L.; Rufus, R.; Esterline, A. Behavioral modeling intrusion detection system (bmids) using internet of things (iot) behavior-based anomaly detection via immunity-inspired algorithms. In Proceedings of the 2016 25th International Conference on Computer Communication and Networks, Waikoloa, HI, USA, 1–4 August 2016; pp. 1–6.
11. Javed, F.; Afzal, M.K.; Sharif, M.; Kim, B.-S. Internet of things (IoT) operating Systems support, networking technologies, applications, and challenges: A comparative review. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2062–2100. [CrossRef]
12. An, X.; Zhou, X.; Lü, X.; Lin, F.; Yang, L. Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC. *Wirel. Commun. Mob. Comput.* **2018**. [CrossRef]
13. Hosseinpour, F.; Vahdani Amoli, P.; Plosila, J.; Hämäläinen, T.; Tenhunen, H. An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach. *Int. J. Digit. Content Technol. Its Appl.* **2016**, *10*, 34–46.
14. Alrawais, A.; Alhothaily, A.; Hu, C.; Cheng, X. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Comput.* **2017**, *21*, 34–42. [CrossRef]
15. Xie, M.; Hu, J.; Yu, X.; Chang, E. Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to adfa-ld. In *Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to Adfa-ld*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 542–549.

16. Xie, M.; Hu, J.; Slay, J. Evaluating host-based anomaly detection systems: Application of the one-class svm algorithm to adfa-ld'. Preceedings of the International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Xiamen, China, 19–21 August 2014; pp. 978–982.
17. Borisaniya, B.; Patel, D. Evaluation of modified vector space representation using adfa-ld and adfa-wd datasets. *J. Inf. Secur.* **2015**, *6*, 250. [CrossRef]
18. Da Costa, K.A.; Papa, J.P.; Lisboa, C.O.; Munoz, R.; de Albuquerque, V.H.C. Internet of Things: A survey on machine learning-based intrusion detection approaches. *Comput. Netw.* **2019**, *151*, 147–157. [CrossRef]
19. Hussain, F.; Hussain, R.; Hassan, S.A.; Hossain, E. Machine learning in IoT security: Current solutions and future challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1686–1721. [CrossRef]
20. Computing, F. *The Internet of Things: Extend the Cloud to Where the Things Are*; Cisco White Paper; Cisco: San Francisco, CA, USA, 2015.
21. Sudqi Khater, B.; Abdul Wahab, A.W.B.; Idris, M.Y.I.B.; Abdulla Hussain, M.; Ahmed Ibrahim, A. A lightweight perceptron-based intrusion detection system for fog computing. *Appl. Sci.* **2019**, *9*, 178. [CrossRef]
22. Dinh, H.T.; Lee, C.; Niyato, D.; Wang, P. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wirel. Commun. Mob. Comput.* **2013**, *13*, 1587–1611. [CrossRef]
23. Jararweh, Y.; Doulat, A.; AlQudah, O.; Ahmed, E.; Al-Ayyoub, M.; Benkhelifa, E. The future of mobile cloud computing: Integrating cloudlets and mobile edge computing. In Proceedings of the 2016 23rd International Conference on Telecommunications, Thessaloniki, Greece, 16–18 May 2016; pp. 1–5.
24. Pierson, R. How Does Fog Computing Differ from Edge Computing. Available online: https://readwrite.com/2016/08/05/fog-computing-different-edge-computing-pl1/ (accessed on 12 June 2021).
25. Ha, K.; Satyanarayanan, M. *Openstack++ for Cloudlet Deployment*; School of Computer Science Carnegie Mellon University Pittsburgh: Pittsburgh, PA, USA, 2015.
26. Jaiswal, A.; Thakare, V.; Sherekar, S. Performance based Analysis of Cloudlet Architectures in Mobile Cloud Computing. *Int. J. Comput. Appl.* **2015**, *975*, 8887.
27. Bahl, V. Emergence of micro datacenter (cloudlets/edges) for mobile computing. *Microsoft Devices Netw. Summit* **2015**, *2015*, 23.
28. Lee, K.; Kim, D.; Ha, D.; Rajput, U.; Oh, H. On security and privacy issues of fog computing supported Internet of Things environment. In Proceedings of the 2015 6th International Conference on the Network of the Future, Montreal, QC, Canada, 30 September–2 October 2015; pp. 1–3.
29. Wang, Y.; Uehara, T.; Sasaki, R. Fog computing: Issues and challenges in security and forensics. In Proceedings of the 2015 IEEE 39th annual computer software and applications conference, Taichung, Taiwan, 1–5 July 2015; pp. 53–59.
30. Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.* **2016**, *3*, 854–864. [CrossRef]
31. Calabretta, M.; Pecori, R.; Vecchio, M.; Veltri, L. MQTT-Auth: A token-based solution to endow MQTT with authentication and authorization capabilities. *J. Commun. Softw. Syst.* **2018**, *14*, 320–331. [CrossRef]
32. Napiah, M.N.; Idris, M.Y.I.B.; Ramli, R.; Ahmedy, I. Compression header analyzer intrusion detection system (cha-ids) for 6lowpan communication protocol. *IEEE Access* **2018**, *6*, 16623–16638. [CrossRef]
33. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 169–186.
34. Roman, R.; Lopez, J.; Mambo, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 680–698. [CrossRef]
35. Wang, X.; Han, Y.; Leung, V.C.; Niyato, D.; Yan, X.; Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 869–904. [CrossRef]
36. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
37. Ahmed, M.; Mahmood, A.N.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [CrossRef]
38. Fernandes, G.; Rodrigues, J.J.; Carvalho, L.F.; Al-Muhtadi, J.F.; Proença, M.L. A comprehensive survey on network anomaly detection. *Telecommun. Syst.* **2019**, *70*, 447–489. [CrossRef]
39. Summerville, D.H.; Zach, K.M.; Chen, Y. Ultra-lightweight deep packet anomaly detection for Internet of Things devices. In Proceedings of the 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), Nanjing, China, 14–16 December 2015; pp. 1–8.
40. Thanigaivelan, N.K.; Nigussie, E.; Kanth, R.K.; Virtanen, S.; Isoaho, J. Distributed internal anomaly detection system for Internet-of-Things. In Proceedings of the 2016 13th IEEE Annual Consumer Communications & Networking Conference, Las Vegas, NV, USA, 9–12 January 2016; pp. 319–320.
41. Pongle, P.; Chavan, G. Real time intrusion and wormhole attack detection in internet of things. *Int. J. Comput. Appl.* **2015**, *121*, 5–7. [CrossRef]
42. Sha, K.; Yang, T.A.; Wei, W.; Davari, S. A survey of edge computing based designs for IoT security. *Digit. Commun. Netw.* **2019**, *6*, 195–202. [CrossRef]
43. Fadlullah, Z.M.; Tang, F.; Mao, B.; Kato, N.; Akashi, O.; Inoue, T.; Mizutani, K. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2432–2455. [CrossRef]

44. Jose, S.; Malathi, D.; Reddy, B.; Jayaseeli, D. A survey on anomaly based host intrusion detection system. In *A Survey on Anomaly Based Host Intrusion Detection System*; IOP Publishing: Bristol, UK, 2018; p. 012049.
45. Sultana, N.; Chilamkurti, N.; Peng, W.; Alhadad, R. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer Peer Netw. Appl.* **2019**, *12*, 493–501. [CrossRef]
46. Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Atkinson, R. Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv* **2017**, arXiv:1701.02145. preprint.
47. Wang, L.; Jones, R. Big data analytics for network intrusion detection: A survey. *Int. J. Netw. Commun.* **2017**, *7*, 24–31.
48. Bridges, R.A.; Glass-Vanderlan, T.R.; Iannacone, M.D.; Vincent, M.S.; Chen, Q. A survey of intrusion detection systems leveraging host data. *ACM Computing Surveys (CSUR)* **2019**, *52*, 1–35. [CrossRef]
49. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. [CrossRef]
50. Vasilomanolakis, E.; Daubert, J.; Luthra, M.; Gazis, V.; Wiesmaier, A.; Kikiras, P. On the security and privacy of Internet of Things architectures and systems. In Proceedings of the 2015 International Workshop on Secure Internet of Things, Vienna, Austria, 21–25 September 2015; pp. 49–57.
51. Ghribi, S.; Makhlouf, A.M.; Zarai, F. C-DIDS: A Cooperative and Distributed Intrusion Detection System in Cloud environment. In Proceedings of the 2018 14th International Wireless Communications & Mobile Computing Conference, Limassol, Cyprus, 25–29 June 2018; pp. 267–272.
52. Shterenberg, S.; Poltavtseva, M.A. A distributed intrusion detection system with protection from an internal intruder. *Autom. Control Comput. Sci.* **2018**, *52*, 945–953. [CrossRef]
53. Goodman, D.L.; Hofmeister, J.; Wagoner, R. Advanced diagnostics and anomaly detection for railroad safety applications: Using a wireless, IoT-enabled measurement system. In Proceedings of the 2015 IEEE AUTOTESTCON, National Harbor, MD, USA, 2–5 November 2015; pp. 273–279.
54. Han, M.L.; Lee, J.; Kang, A.R.; Kang, S.; Park, J.K.; Kim, H.K. A statistical-based anomaly detection method for connected cars in internet of things environment. In *A Statistical-Based Anomaly Detection Method for Connected Cars in Internet of Things Environment*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 89–97.
55. Da Xu, L.; He, W.; Li, S. Internet of things in industries: A survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243.
56. Hajiheidari, S.; Wakil, K.; Badri, M.; Navimipour, N.J. Intrusion detection systems in the Internet of things: A comprehensive investigation. *Comput. Netw.* **2019**, *160*, 165–191. [CrossRef]
57. Elrawy, M.F.; Awad, A.I.; Hamed, H.F. Intrusion detection systems for IoT-based smart environments: A survey. *J. Cloud Comput.* **2018**, *7*, 1–20. [CrossRef]
58. Ukil, A.; Bandyoapdhyay, S.; Puri, C.; Pal, A. IoT healthcare analytics: The importance of anomaly detection. In Proceedings of the 2016 IEEE 30th international conference on advanced information networking and applications, Crans-Montana, Switzerland, 23–25 March 2016; pp. 994–997.
59. Borkar, A.; Donode, A.; Kumari, A. A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS). In Proceedings of the 2017 International conference on inventive computing and informatics, Coimbatore, India, 23–24 November 2017; pp. 949–953.
60. Bijone, M. A survey on secure network: Intrusion detection & prevention approaches. *Am. J. Inf. Syst.* **2016**, *4*, 69–88.
61. Wu, S.X.; Banzhaf, W. The use of computational intelligence in intrusion detection systems: A review. *Appl. Soft Comput.* **2010**, *10*, 1–35. [CrossRef]
62. Kishan, B.; Reddy, A.S.; Datta, M.; Raghunath, B.; Vinay, C.; Reddy, K.G. Intrusion Detection Systems for Iot-Based Smart Environments: A Survey. *Complex. Int.* **2020**, *24*, 3–4.
63. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 686–728. [CrossRef]
64. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
65. Ahmadian Ramaki, A.; Rasoolzadegan, A.; Javan Jafari, A. A systematic review on intrusion detection based on the Hidden Markov Model. *Stat. Anal. Data Min. ASA Data Sci. J.* **2018**, *11*, 111–134. [CrossRef]
66. Fenanir, S.; Semchedine, F.; Baadache, A. A Machine Learning-Based Lightweight Intrusion Detection System for the Internet of Things. *Rev. D'intelligence Artif.* **2019**, *33*, 203–211. [CrossRef]
67. Aburomman, A.A.; Reaz, M.B.I. Survey of learning methods in intrusion detection systems. In Proceedings of the 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering, Putrajaya, Malaysia, 14–16 November 2016; pp. 362–365.
68. Al-Garadi, M.A.; Mohamed, A.; Al-Ali, A.K.; Du, X.; Ali, I.; Guizani, M. A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1646–1685. [CrossRef]
69. Chinnamgari, S.K. *R Machine Learning Projects: Implement Supervised, Unsupervised, and Reinforcement Learning Techniques Using R 3.5′*; Packt Publishing Ltd.: Birmingham, UK, 2019.
70. Thakkar, A.; Lohiya, R. A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges. *Arch. Comput. Methods Eng.* **2020**, *28*, 3211–3243. [CrossRef]
71. De Andrade, B.M.; de Gois, J.S.; Xavier, V.L.; Luna, A.S. Comparison of the performance of multiclass classifiers in chemical data: Addressing the problem of overfitting with the permutation test. *Chemom. Intell. Lab. Syst.* **2020**, *201*, 104013. [CrossRef]

72. Heba, F.E.; Darwish, A.; Hassanien, A.E.; Abraham, A. Principle components analysis and support vector machine based intrusion detection system. In Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, Cairo, Egypt, 29 November–1 December 2010; pp. 363–367.
73. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. *Eai Endorsed Trans. Secur. Saf.* **2016**, *3*, e2.
74. Zanero, S.; Savaresi, S.M. Unsupervised learning techniques for an intrusion detection system. In Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 14–17 March 2004; pp. 412–419.
75. Syarif, I.; Prugel-Bennett, A.; Wills, G. Unsupervised clustering approach for network anomaly detection. In *Unsupervised Clustering Approach for Network Anomaly Detection*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 135–145.
76. Oh, D.; Kim, D.; Ro, W.W. A malicious pattern detection engine for embedded security systems in the Internet of Things. *Sensors* **2014**, *14*, 4188. [CrossRef]
77. Qu, X.; Yang, L.; Guo, K.; Ma, L.; Sun, M.; Ke, M.; Li, M. A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mob. Netw. Appl.* **2019**, *26*, 808–829. [CrossRef]
78. Haweliya, J.; Nigam, B. Network intrusion detection using semi supervised support vector machine. *Int. J. Comput. Appl.* **2014**, *85*. [CrossRef]
79. Li, W.; Meng, W.; Au, M.H. Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments. *J. Netw. Comput. Appl.* **2020**, *161*, 102631. [CrossRef]
80. Al-Jarrah, O.Y.; Al-Hammdi, Y.; Yoo, P.D.; Muhaidat, S.; Al-Qutayri, M. Semi-supervised multi-layered clustering model for intrusion detection. *Digit. Commun. Netw.* **2018**, *4*, 277–286. [CrossRef]
81. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
82. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
83. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971. preprint.
84. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952. preprint.
85. Alharbi, S.; Rodriguez, P.; Maharaja, R.; Iyer, P.; Subaschandrabose, N.; Ye, Z. Secure the internet of things with challenge response authentication in fog computing. In Proceedings of the 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC), San Diego, CA, USA, 10–12 December 2017; pp. 1–2.
86. Shafi, Q.; Basit, A.; Qaisar, S.; Koay, A.; Welch, I. Fog-assisted SDN controlled framework for enduring anomaly detection in an IoT network. *IEEE Access* **2018**, *6*, 73713–73723. [CrossRef]
87. Xuan, S.; Man, D.; Yang, W.; Wang, W.; Zhao, J.; Yu, M. Identification of unknown operating system type of Internet of Things terminal device based on RIPPER. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1550147718806707. [CrossRef]
88. Ham, H.-S.; Kim, H.-H.; Kim, M.-S.; Choi, M.-J. Linear SVM-based android malware detection for reliable IoT services. *J. Appl. Math.* **2014**, *2014*. [CrossRef]
89. Azmoodeh, A.; Dehghantanha, A.; Conti, M.; Choo, K.-K.R. Detecting crypto-ransomware in IoT networks based on energy consumption footprint. *J. Ambient Intell. Humaniz. Comput.* **2018**, *9*, 1141–1152. [CrossRef]
90. Caminha, J.; Perkusich, A.; Perkusich, M. A smart trust management method to detect on-off attacks in the internet of things. *Secur. Commun. Netw.* **2018**, *2018*. [CrossRef]
91. Chiu, W.; Su, C.; Fan, C.-Y.; Chen, C.-M.; Yeh, K.-H. Authentication with what you see and remember in the internet of things. *Symmetry* **2018**, *10*, 537. [CrossRef]
92. Doshi, R.; Apthorpe, N.; Feamster, N. Machine learning ddos detection for consumer internet of things devices. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 29–35.
93. Kotenko, I.; Saenko, I.; Branitskiy, A. Framework for mobile Internet of Things security monitoring based on big data processing and machine learning. *IEEE Access* **2018**, *6*, 72714–72723. [CrossRef]
94. Wei, L.; Luo, W.; Weng, J.; Zhong, Y.; Zhang, X.; Yan, Z. Machine learning-based malicious application detection of android. *IEEE Access* **2017**, *5*, 25591–25601. [CrossRef]
95. Park, W.; You, Y.; Lee, K. Detecting Potential Insider Threat: Analyzing Insiders' Sentiment Exposed in Social Media. *Secur. Commun. Networks* **2018**, *2018*, 1–8. [CrossRef]
96. Goeschel, K. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis. In Proceedings of the SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–6.
97. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176. [CrossRef]
98. Abe, S. Minimal Complexity Support Vector Machines. In Proceedings of the IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Winterthur, Switzerland, 2–4 September 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 89–101.
99. Liu, Y.; Pi, D. A Novel Kernel SVM Algorithm with Game Theory for Network Intrusion Detection. *KSII Trans. Internet Inf. Syst.* **2017**, *11*. [CrossRef]

100. Pajouh, H.H.; Javidan, R.; Khayami, R.; Dehghantanha, A.; Choo, K.-K.R. A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. *IEEE Trans. Emerg. Top. Comput.* **2016**, *7*, 314–323. [CrossRef]
101. Li, L.; Zhang, H.; Peng, H.; Yang, Y. Nearest neighbors based density peaks approach to intrusion detection. *Chaos, Solitons Fractals* **2018**, *110*, 33–40. [CrossRef]
102. Chang, Y.; Li, W.; Yang, Z. Network intrusion detection based on random forest and support vector machine. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 635–638.
103. Meidan, Y.; Bohadana, M.; Shabtai, A.; Ochoa, M.; Tippenhauer, N.O.; Guarnizo, J.D.; Elovici, Y. Detection of unauthorized iot devices using machine learning techniques. *arXiv* **2017**, arXiv:1709.04647. preprint.
104. Laskov, P.; Düssel, P.; Schäfer, C.; Rieck, K. Learning intrusion detection: Supervised or unsupervised? In Proceedings of the International Conference on Image Analysis and Processing, Cagliari, Italy, 6–8 September 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 50–57.
105. Li, Q.; Zhang, K.; Cheffena, M.; Shen, X. Channel-based sybil detection in industrial wireless sensor networks: A multi-kernel approach. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
106. Xie, M.; Huang, M.; Bai, Y.; Hu, Z. The anonymization protection algorithm based on fuzzy clustering for the ego of data in the internet of things. *J. Electr. Comput. Eng.* **2017**. [CrossRef]
107. Kfoury, E.; Saab, J.; Younes, P.; Achkar, R. A Self Organizing Map Intrusion Detection System for RPL Protocol Attacks. *Int. J. Interdiscip. Telecommun. Netw.* **2019**, *11*, 30–43. [CrossRef]
108. Janarthanan, T.; Zargari, S. Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In Proceedings of the 2017 IEEE 26th international symposium on industrial electronics (ISIE), Edinburgh, UK, 19–21 June 2017; pp. 1881–1886.
109. Dua, M. Machine Learning Approach to IDS: A Comprehensive Review. In Proceedings of the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Tamil Nadu, India, 2–14 June 2019; pp. 117–122.
110. Soe, Y.N.; Feng, Y.; Santosa, P.I.; Hartanto, R.; Sakurai, K. Implementing Lightweight iot-ids on Raspberry pi Using Correlation-Based Feature Selection and Its Performance Evaluation. 2019. Available online: https://kyushu-u.pure.elsevier.com/en/publications/implementing-lightweight-iot-ids-on-raspberry-pi-using-correlatio (accessed on 30 May 2021).
111. Jain, A.K.; Duin, R.P.W.; Mao, J. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 4–37. [CrossRef]
112. Li, D.; Deng, L.; Lee, M.; Wang, H. IoT Data Feature Extraction and Intrusion Detection System for Smart Cities Based on Deep Migration Learning. *Int. J. Inf. Manag.* **2019**, *49*, 533–545. [CrossRef]
113. Ramaki, A.A.; Rasoolzadegan, A.; Bafghi, A.G. A systematic mapping study on intrusion alert analysis in intrusion detection systems. *ACM Computing Surveys (CSUR)* **2018**, *51*, 1–41. [CrossRef]
114. Zhang, K.; Luo, S.; Xin, Y.; Zhu, H.; Chen, Y. Online Mining Intrusion Patterns from IDS Alerts. *Appl. Sci.* **2020**, *10*, 2983. [CrossRef]
115. Manikandan, G.; Abirami, S. A survey on feature selection and extraction techniques for high-dimensional microarray datasets. In *Knowledge Computing and Its Applications*; Springer: Singapore, 2018; pp. 311–333.
116. Aminanto, M.E.; Choi, R.; Tanuwidjaja, H.C.; Yoo, P.D.; Kim, K. Deep abstraction and weighted feature selection for Wi-Fi impersonation detection. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 621–636. [CrossRef]
117. Ghaffarian, S.M.; Shahriari, H.R. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. *ACM Computing Surveys (CSUR)* **2017**, *50*, 1–36. [CrossRef]
118. Manzoor, I.; Kumar, N. A feature reduced intrusion detection system using ANN classifier. *Expert Syst. Appl.* **2017**, *88*, 249–257.
119. Vergara, J.R.; Estévez, P.A. A review of feature selection methods based on mutual information. *Neural Comput. Appl.* **2014**, *24*, 175–186. [CrossRef]
120. Yu, L.; Liu, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In Proceedings of the 20th International Conference on Machine Learning, Fort Lauderdale, FL, USA, 21–24 August 2003; pp. 856–863.
121. Biswas, S.; Bordoloi, M.; Purkayastha, B. Review on Feature Selection and Classification using Neuro-Fuzzy Approaches. *Int. J. Appl. Evol. Comput. (IJAEC)* **2016**, *7*, 28–44. [CrossRef]
122. Cateni, S.; Vannucci, M.; Vannocci, M.; Colla, V. Variable selection and feature extraction through artificial intelligence techniques. *Multivar. Anal. Manag. Eng. Sci.* **2012**, 103–118. [CrossRef]
123. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52. [CrossRef]
124. Salo, F.; Nassif, A.B.; Essex, A. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Comput. Netw.* **2019**, *148*, 164–175. [CrossRef]
125. Zhao, S.; Li, W.; Zia, T.; Zomaya, A.Y. A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things. In Proceedings of the 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, Orlando, FL, USA, 6–10 November 2017.
126. Zheng, Y.; Vanderbeek, B.; Daniel, E.; Stambolian, D.; Maguire, M.; Brainard, D.; Gee, J. An automated drusen detection system for classifying age-related macular degeneration with color fundus photographs. In Proceedings of the 2013 IEEE 10th International Symposium on Biomedical Imaging, San Francisco, CA, USA, 7–11 April 2013; pp. 1448–1451.

127. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008.
128. Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text classification algorithms: A survey. *Information* **2019**, *10*, 150. [CrossRef]
129. Vinayakumar, R.; Alazab, M.; Soman, K.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
130. Rai, K.; Devi, M.S.; Guleria, A. Packet-based Anomaly Detection using n-gram Approach. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 6. [CrossRef]
131. Khreich, W.; Khosravifar, B.; Hamou-Lhadj, A.; Talhi, C. An anomaly detection system based on variable N-gram features and one-class SVM. *Inf. Softw. Technol.* **2017**, *91*, 186–197. [CrossRef]
132. Subba, B.; Biswas, S.; Karmakar, S. Host based intrusion detection system using frequency analysis of n-gram terms. In Proceedings of the TENCON 2017-2017 IEEE Region 10 Conference, Penang, Malaysia, 5–8 November 2017; pp. 2006–2011.
133. Gaydhani, A.; Doma, V.; Kendre, S.; Bhagwat, L. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. *arXiv* **2018**, arXiv:1809.08651. preprint.
134. Rumez, M.; Lin, J.; Fuchß, T.; Kriesten, R.; Sax, E. Anomaly Detection for Automotive Diagnostic Applications Based on N-Grams. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference, Madrid, Spain, 13–17 July 2020; pp. 1423–1429.
135. Tran, C.T.; Zhang, M.; Andreae, P.; Xue, B.; Bui, L.T. Improving performance of classification on incomplete data using feature selection and clustering. *Appl. Soft Comput.* **2018**, *73*, 848–861. [CrossRef]
136. Scherer, P.; Vicher, M.; Drazdilova, P.; Martinovic, J.; Dvorsky, J.; Snasel, V. *Using Svm and Clustering Algorithms in Ids Systems*; Citeseer: Princeton, NJ, USA, 2011.
137. Pham, N.T.; Foo, E.; Suriadi, S.; Jeffrey, H.; Lahza, H.F.M. Improving performance of intrusion detection system using ensemble methods and feature selection. In Proceedings of the Proceedings of the Australasian Computer Science Week Multiconference, Sydney, Australia, 29–31 January 2019; pp. 1–6.
138. Boutaba, R.; Salahuddin, M.A.; Limam, N.; Ayoubi, S.; Shahriar, N.; Estrada-Solano, F.; Caicedo, O.M. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *J. Internet Serv. Appl.* **2018**, *9*, 16. [CrossRef]
139. Kuang, F.; Zhang, S.; Jin, Z.; Xu, W. A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection. *Soft Comput.* **2015**, *19*, 1187–1199. [CrossRef]
140. Syarif, A.R.; Gata, W. Intrusion detection system using hybrid binary PSO and K-nearest neighborhood algorithm. Preceedings of the 11th International Conference on Information, Communication Technology and System (ICTS 2017), Surabaya, Indonesia, 31 October 2017; pp. 181–186.
141. Pajouh, H.H.; Dastghaibyfard, G.; Hashemi, S. Two-tier network anomaly detection model: A machine learning approach. *J. Intell. Inf. Syst.* **2017**, *48*, 61–74. [CrossRef]
142. Mahmood, H.A. Network intrusion detection system (NIDS) in cloud environment based on hidden Naïve Bayes multiclass classifier. *Al-Mustansiriyah J. Sci.* **2018**, *28*, 134–142. [CrossRef]
143. Borisaniya, B.; Patel, K.; Patel, D. Evaluation of applicability of modified vector space representation for in-VM malicious activity detection in Cloud. In Proceedings of the 2014 Annual IEEE India Conference (INDICON), Pune, India, 11–13 December 2014; pp. 1–6.
144. Leslie, C.; Eskin, E.; Noble, W.S. The spectrum kernel: A string kernel for SVM protein classification. In *Biocomputing*; World Scientific: Singapore, 2001; pp. 564–575.
145. Bunch, J.R.; Rose, D.J. *Sparse Matrix Computations*; Academic Press: Cambridge, MA, USA, 2014.
146. D'Azevedo, E.F.; Fahey, M.R.; Mills, R.T. Vectorized sparse matrix multiply for compressed row storage format. In Proceedings of the International Conference on Computational Science, Amsterdam, The Netherlands, 3–5 June 2005; pp. 99–106.
147. Jamalmohammed, S.B.; Lavanya, K.; Thaseen, S.; Biju, V. Review on Sparse Matrix Storage Formats With Space Complexity Analysis. In *Applications of Artificial Intelligence for Smart Technology*; IGI Global: Hershey, PA, USA, 2020; pp. 122–145.
148. Zheng, J.; Hu, M.-Z.; Zhang, H.-L. A new method of data preprocessing and anomaly detection. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, 26–29 August 2004; pp. 2685–2690.
149. Tian, C.; Fei, L.; Zheng, W.; Xu, Y.; Zuo, W.; Lin, C.-W. Deep learning on image denoising: An overview. *Neural Netw.* **2020**, *131*. [CrossRef]
150. Wang, J.; Hong, X.; Ren, R.-R.; Li, T.-H. A real-time intrusion detection system based on PSO-SVM. In Proceedings of the The 2009 International Workshop on Information Security and Application, Jeju Island, South Korea, 26–28 August 2020; p. 319.
151. Ahmed, H.I.; Elfeshawy, N.A.; Elzoghdy, S.F.; El-Sayed, H.S.; Faragallah, O.S. A neural network-based learning algorithm for intrusion detection systems. *Wirel. Pers. Commun.* **2017**, *97*, 3097–3112. [CrossRef]
152. Fawcett, T. Introduction to Receiver Operator Curves. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]
153. Sachs, M.C. plotROC: A tool for plotting ROC curves. *J. Stat. Softw.* **2017**, *79*. [CrossRef] [PubMed]
154. Viegas, E.K.; Santin, A.O.; Oliveira, L.S. Toward a reliable anomaly-based intrusion detection in real-world environments. *Comput. Netw.* **2017**, *127*, 200–216. [CrossRef]
155. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. *arXiv* **2018**, arXiv:1806.03517. preprint.

156. Sharafaldin, I.; Gharib, A.; Lashkari, A.H.; Ghorbani, A.A. Towards a reliable intrusion detection benchmark dataset. *Softw. Netw.* **2018**, *2018*, 177–200. [CrossRef]
157. Creech, G.; Hu, J. Generation of a new IDS test dataset: Time to retire the KDD collection. In Proceedings of the 2013 IEEE Wireless Communications and Networking Conference, Shanghai, China, 7–10 April 2013; pp. 4487–4492.
158. Available online: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-IDS-Datasets/ (accessed on 2 February 2017).
159. Haider, W.; Creech, G.; Xie, Y.; Hu, J. Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks. *Future Internet* **2016**, *8*, 29. [CrossRef]
160. Abubakar, A.I.; Chiroma, H.; Muaz, S.A.; Ila, L.B. A Review of the Advances in Cyber Security Benchmark Datasets for Evaluating Data-Driven Based Intrusion Detection Systems. *Procedia Comput. Sci.* **2015**, *62*, 221–227. [CrossRef]
161. Xie, M.; Hu, J. Evaluating host-based anomaly detection systems: A preliminary analysis of adfa-ld. In Proceedings of the 2013 6th International Congress on Image and Signal Processing, Hangzhou, Chia, 16–18 December 2013; Manning Publications Co.: Helter Island, NY, USA, 2012.
162. Harrington, P. Machine learning in action. 2012. Available online: https://www.accenture.com/hk-en/services/ai-artificial-intelligence-index?c=acn_glb_brandexpressiongoogle_12238967&n=psgs_0621&gclid=EAIaIQobChMI19OiroXT8QIVFwkrCh0HYAR3EAAYASAAEgILTfD_BwE (accessed on 2 June 2021).
163. Ranjani, J.; Sheela, A.; Meena, K.P. Combination of NumPy, SciPy and Matplotlib/Pylab-a good alternative methodology to MATLAB-A Comparative analysis. In Proceedings of the 2019 1st International Conference on Innovations in Information and Communication Technology, Hennai, India, 25–26 April 2019; pp. 1–5.
164. Müller, A.C.; Guido, S. *Introduction to Machine Learning with Python: A Guide for Data Scientists*; O'Reilly Media, Inc.: Newton, MA, USA, 2016.
165. Bisong, E. The Multilayer Perceptron (MLP). In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 401–405.
166. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
167. Saleh, H. *Machine Learning Fundamentals: Use Python and Scikit-Learn to Get Up and Running with the Hottest Developments in Machine Learning*; Packt Publishing: Birmingham, UK, 2018.
168. Borthakur, D.; Dubey, H.; Constant, N.; Mahler, L.; Mankodiya, K. Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things. In Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing, Montreal, QC, Canada, 14–16 November 2017; pp. 472–476.
169. Constant, N.; Borthakur, D.; Abtahi, M.; Dubey, H.; Mankodiya, K. Fog-assisted wiot: A smart fog gateway for end-to-end analytics in wearable internet of things. *arXiv* **2017**, arXiv:1701.08680. preprint.
170. Lavassani, M.; Forsström, S.; Jennehag, U.; Zhang, T. Combining fog computing with sensor mote machine learning for industrial IoT. *Sensors* **2018**, *18*, 1532. [CrossRef]
171. Learning, U. *Raspberry Pi 3: Get Started with Raspberry Pi 3 a Simple Guide TO Understanding and Programming Raspberry Pi 3 (Raspberry Pi 3 User Guide, Python Programming, Mathematica Programming)*; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2016.
172. Castro, W.; Oblitas, J.; Santa-Cruz, R.; Avila-George, H. Multilayer perceptron architecture optimization using parallel computing techniques. *PLoS ONE* **2017**, *12*, e0189369. [CrossRef] [PubMed]