applied sciences

MDPI

Article

A Novel Intrusion Detection Approach Using Machine Learning Ensemble for IoT Environments

Parag Verma [1], Ankur Dumka [2], Rajesh Singh [3], Alaknanda Ashok [4], Anita Gehlot [3], Praveen Kumar Malik [3], Gurjot Singh Gaba [5],* and Mustapha Hedabou [5]

[1] Chitkara University Institute of Engineering and Technology, Chitkara University Punjab, Rajpura 140401, India; parag_verma@yahoo.com
[2] Women Institute of Technology, Dehradun 248007, India; ankurdumka2@gmail.com
[3] School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara 144411, India; rajesh.23402@lpu.co.in (R.S.); anita.23401@lpu.co.in (A.G.); praveen.23314@lpu.co.in (P.K.M.)
[4] Computer Science & Engineering, College of Technology, G.B. Pant University of Agriculture and Technology, Pantnagar 263145, India; alakn@rediff.com
[5] School of Computer Science, Mohammed VI Polytechnic University, Ben Guerir 43150, Morocco; mustapha.hedabou@um6p.ma
* Correspondence: gurjot.singh@um6p.ma

Abstract: The Internet of Things (IoT) has gained significant importance due to its applicability in diverse environments. Another reason for the influence of the IoT is its use of a flexible and scalable framework. The extensive and diversified use of the IoT in the past few years has attracted cyber-criminals. They exploit the vulnerabilities of the open-source IoT framework due to the absentia of robust and standard security protocols, hence discouraging existing and potential stakeholders. The authors propose a binary classifier approach developed from a machine learning ensemble method to filter and dump malicious traffic to prevent malicious actors from accessing the IoT network and its peripherals. The gradient boosting machine (GBM) ensemble approach is used to train the binary classifier using pre-processed recorded data packets to detect the anomaly and prevent the IoT networks from zero-day attacks. The positive class performance metrics of the model resulted in an accuracy of 98.27%, a precision of 96.40%, and a recall of 95.70%. The simulation results prove the effectiveness of the proposed model against cyber threats, thus making it suitable for critical applications for the IoT.

Keywords: machine learning; intrusion detection system; network anomaly detection; Internet of Things; gradient boosting machine (GBM)

check for updates

Citation: Verma, P.; Dumka, A.; Singh, R.; Ashok, A.; Gehlot, A.; Malik, P.K.; Gaba, G.S.; Hedabou, M. A Novel Intrusion Detection Approach Using Machine Learning Ensemble for IoT Environments. *Appl. Sci.* **2021**, *11*, 10268. https://doi.org/10.3390/app112110268

Academic Editor: Donato Cascio

Received: 10 September 2021
Accepted: 14 October 2021
Published: 1 November 2021

1. Introduction

The security of networks is very important as they are widely used to provide communication and information-sharing among individuals, industries, and other parts of society. Various strategies in the form of firewall protection policies and antivirus software have been put into practice for maintaining client security and the protection of their sensitive data [1].

The industry protected intrusion detection system (IDS) recognizes threats or suspicious activities on the network flows by analyzing network traffic or a specific perceived environment. One type of anomaly in a network is an intrusion or threat. Interlopers exploit network vulnerabilities by misusing network faults that bring about a breach of network security and software bugs such as overflowing of the buffer. Interlopers target low privilege clients and expect to find additional access power or programmers that are common web-based clients to hack or damage sensitive data [2].

Although several industry protected IDS algorithms have been proposed by network researchers, most of them are based on traditional mechanisms such as encryption and authentication procedures to protect wireless sensor units and are insufficient. Hence, the existing IDS needs a lot of progressive work to achieve high-performance authentication and security. The increased use of the artificial intelligence–machine learning ensemble technique in industries to classify multi-pattern data was the motivation for this research work to help in the recognition of traffic patterns and the methods of anomaly detection over internet-enabled wireless sensors. There is a growing use of the ensemble techniques of machine learning, a branch of artificial intelligence, and their ability to classify multi-pattern data across industries applications. Accordingly, this research has also applied machine learning ensemble techniques to IDS, so this work is focused on traffic pattern recognition and anomaly detection on Internet-enabled wireless sensors.

To detect novel attacks, the proposed intrusion detection system handles the recently recorded benign information in data packets and analyzes the network traffic for anomalous data flows.

Intrusion detection techniques are categorized into two types: signature-based and anomaly-based. In signature-based intrusion detection techniques, the system monitors the flow of packets within the network and compares the value obtained with previously identified values which are configured as the signature of known attacks. The anomaly-based method detects attacks by capturing user parameters that show deviation from the legitimate user parameters [3].

The major contributions of this research paper are as follows:

- A focus on mobile network appliances such as IoT-enabled platforms for IDSs.
- A machine learning-based binary classifier for identifying network traffic as benign or malicious to provide network security.
- Training, validating, and testing of the model using the random forest and gradient boosting machine (GBM) ensemble approach with a hyperparameter optimizer using the 'CSE-CIC-IDS2018-V2' dataset and demonstrating the performance test with attack categories like infiltration, SQL injection, etc.

This paper consists of six sections. In Section 2, the focus is on the background of the present research and its challenges. Section 3 covers the system architecture of the proposed work. Section 4 summarizes the dataset used for training, validating, and testing of the model to build a binary classifier for IDS. Section 5 of this paper includes the demonstration of the methodology and its results. Section 5 presents the conclusions derived from the proposed work.

2. Literature Review

Communication-based technologies such as the Internet of Things (IoT), mobile networks, and wireless sensor-enabled networks bring a lot of attention to network security. The motivation behind this section is a detailed overview of recent trends in IoT devices and a focus on advancements in machine learning-based solutions for IDSs as a single intrusion has the potential to damage the entire network.

2.1. Machine Learning Enabled Intrusion Detection for IoT Appliances

2.1.1. Intrusion Detection System

An intrusion detection system (IDS) determines the blockage by reviewing the network against violation activities [4]. This model was identified [5] in the 1980s and it can be divided into a set of two classes. The initial two classes are based on the network and host separately, where the behavior of intrusion can be noted. Monitoring the network and analyzing the network traffic are the focused activities in a network-based IDS to serve network security. The host-based IDS monitors the malicious activities happening in the processes and the system events in the software-based environment [6,7].

Another category of IDS relies on the approach of data analytics, which has been used in all types of intrusions such as signature, anomaly, and hybrid-based intrusion. The signature-based approach searches for information from network packets or a specific system (for example, logs) to detect signatures or a similar pattern that support characteristics of behavior for intrusive activity. Such a strategy is essentially more feasible as it uses recently named information from the dataset. Although it is described as a basic and effective technique, it cannot see ambiguous attacks and requires constant information base updates [3,7–9].

An approach based on anomaly intrusion examines the information to look for irregular situations that differ from the general network behavior and system practices. This capability can rely on the recently given information which was used to produce specific algorithms. The strategy depicted is promising as it can enable the search for zero-day attacks. Moreover, it includes a more vigorous adaption to a specific system or network. In such cases, a very significant downside is that these methods are more prone to false-positive alerts that contain a higher level of information, as they are not based on uniquely named information. Recently, some new instructions have been issued for their dependencies, which will search for the given information according to the terms of data discrepancies. However, this information is not enough to detect network security attacks [6–8,10].

The hybrid technique combines both signature and anomaly detection techniques. This kind of strategy was designed by Buczak and Guven to limit the consequences of bogus precautions and further increase recognition for attacks that are known [7].

Liao et al. conducted a comprehensive survey that marked some additional techniques similar to the wireless-based analysis of network behavior, hybrid IDSs, and the analysis of the existing key protocols. Wireless-based IDSs closely resemble the network-based schemes which are capable of capturing wireless traffic flows. A system for analyzing the network behaviors is based on network traffic to search for malicious attacks along with expected flows of network traffic. Hybrid IDSs consolidate various advances to provide more intensive and accurate interrupt detections, while the stateful IDS protocol dissects the explicit conditions of the specific network so that examples of contraceptive harm can be searched [8] (Table 1).

Table 1. Summary of some of IDS types.

Intrusion detection prone area	Based on dedicated host Based on dedicated network Based on remote accessibility Based on analytics of network behavior
Methodologies of intrusion detection	Based on authorized signature values Based on data pattern irregularity or anomaly Based on analytics of stateful protocols

For modeling the IDS, other categories of machine learning and soft computing were introduced by Camastra et al. [6]. Four different groups of mechanisms based on machine learning and soft computing were depicted: supervised learning, unsupervised learning, statistical modeling, and ensemble-based approaches [11]. The dominant approach is used to identify the unique behavior intrusions, while single strategies work for novel intrusions. The statistical modeling-based approach is used to evaluate the behaviors of users and assess whether it is characterized as 'normal', while the ensemble-based approach joins some models to improve proficiency and accuracy.

2.1.2. CSE-CIC-IDS2018-V2 Dataset

The dataset has a huge impact on the performance evaluation of a machine learning algorithm [12]. A review of the literature suggests that most of the intrusion detection methods that are based on machine learning have been trained by the DAROA 98/991 and KDD CUP 992 datasets. However, the method of intrusion detection using these datasets is not very effective, and many researchers [13–17] are against the recommendation of their use. This is why a new dataset, CSE-CIC-IDS 2018 (https://registry.opendata.aws/CSE-CIC-IDS2018-V2/, accessed on 15 March 2021), collected by the Canadian Institute of

Cyberspace (CIC) on Amazon's AWS LAN network, was used in our research work [18]. This dataset is also termed as CIC-AWS-2018. This dataset provides TCP/IP level traffic data details like IP address, port number, detection of CIC-AWS dataset, and statistical data of traffic based on the flow of data in the network. A network flow generator and analyzer, CICFlowMeter, was used to calculate the statistical data.

The dataset includes six types of infiltration conditions: DoS attack, Brute Force attack, botnet attack, DDoS attack, web attack, and infiltration, which are within the network, along with 14 different types of intrusions: FTP-Brute Force, Brute Force-Web, botnet, SSH Brute Force, DDoS-HOIC attacks, DDoS-LOIC-UDP attacks, DDoS-LOIC-HTTP attacks, SQL injections, Brute Force-XSS, DoS GoldenEye attacks, DoS Hulk attacks, DoS slow HTTP test attacks, infiltration, and DoS Slowloris attacks. The data is labeled according to three criteria, namely, the number of transmitted packets, data flow duration, and counts of certain flags. The CIC-AWS-2018 datasets also contain the records of network traffic and event logs.

However, considerable research work has been done on CIC-IDS-2017, an older version of the CIC-AWS-2018 dataset. Some researchers [19] have implemented a classifier based on unsupervised learning. Sharafaldin et.al, the creator of the CIC-IDS-2017 dataset, used the random forest regression model, developed by CIC, to select only the top four features that are fully capable of detecting intrusion derived from machine learning techniques [20]. In our analysis, we used the CSE-CIC-IDS 2018 because it is more recent than the other datasets like CIDDS-001 (CIDDS-001 dataset. https://www.hs-coburg.de/forschungkooperation/forschungsprojekte-oeffentlich/ingenieurwissenschaften/cidds-coburg-intrusion-detectiondata-sets.html, accessed on 20 April 2021), UNSW-NB15 (UNSW-NB15 dataset. https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/, accessed on 20 April 2021), and NSLKDD (NSL-KDD dataset. http://nsl.cs.unb.ca/nsl-kdd/, accessed on 20 April 2021). In addition, this dataset has the latest information about traffic attacks that supports building IDSs to detect novel attacks.

2.1.3. IoT-Enabled Networks

For secure communications to transmit data over IoT-enabled networks, various studies have been carried out that define the vulnerabilities in IoT systems. This section explores the contributions of many researchers suggesting different methods for defending the IoT against malicious attacks.

Misra et al. [21] suggested an IDS based on the specification for learning automata to prevent DoS attacks against the IoT. In this proposed system, an IoT center product layer is used instead of a specific tool and is helpful in the decision-making process. This system sets an edge for the number of solvents a middleware layer can support. When the number of convulsions exceeds the set limit, an attack is recognized. Kasinathan et al. [12] proposed a signature-based intrusion detection system that consists of monitoring and recognizing the modules. In that study, the module was coordinated with the European Union (EU) FP7 enterprise project's network system, called 'ebbits', to ensure that networks were subject to DoS attacks. A DoS assurance administrator and IDS were included in the proposed network. An IDS was used to catch and inspect the sniffing packets from IDS testing centers and spread them across the networks. The estimation results indicated that the proposed system exhibits true positive and false-positive rates.

Suricata [13] put forward an Open Source IDS that provided the functionality of matching patterns and detecting attacks in the network. A test node was used to sniff all packet transmissions in the network and the transmission of the data to IDS for additional analysis. The performance of the proposed IDS was tested by penetration testing tools called scape. No simulation-based studies were conducted in favor of IDS performance and its convenience. Additionally, the authors did not refer to any information about the signature database management.

Lee et al. [14] came up with a novel IDS to spot DoS attacks. The main idea behind the proposed IDS was to reduce the energy of the nodes by detecting malicious nodes.

In this research, it was suggested that various models use cross-section direction-based networks. The security framework proposed expects the nodes to demonstrate the use of their energy at an inspection rate of 0.5 s. The suggested approach continuously monitored the energy usage by the nodes and any node that was found to deviate at any point was set as a malicious node. Finally, such nodes were removed, and the routing table of the connected nodes was also updated.

An IDS proposed by Sonar and Upadhyay [15] operates as a software-based supervisor between the network and the gateway. This helps to protect the IP addresses in the grey list or the ones that control access to the network. At every 300 s interval, the blacklist is refreshed, and the grey list is refreshed every 40 s. The simulated framework of this IDS is reproduced on the Contiki operating system by Dunkels et al. [16]. Amount of packet delivery ratio, packets served, true positives, and false positives are not performed at an acceptable rate in this IDS. Another significant limitation of the proposed IDS is that it has a larger retrieval time as compared to the learning time.

Similarly, machine learning techniques are recommended in modern-day IDSs to achieve accurate prediction, automation, speed, and scalability. In the same direction, machine learning for intrusion detection in the industrial IoT (IIoT) was applied through federated learning (FL) in [22]. The federated learning assistant application is primarily designed to train an algorithm that enables multiple edge devices to connect without sharing private data. Distributed denial of service (DDoS) attacks pose a significant threat to IIoT applications. The adversarial nodes overwhelm the network servers, SCADA systems, and other IoT-enabled devices, causing the crash of industrial systems and resulting in operational disruption. The authors have suggested the use of FL to protect the IIoT framework from DDoS attacks. They trained the model locally so that edge nodes can include modules that perform analysis and detection for DDoS. These modules include traffic policies that all the network traffic must adhere to before a connection can be established. As per their scheme, upon sensing any malicious cyber activity, the IDS moves the application to the sandbox, and related information is updated to the cloud to make other edge nodes aware of this type of intrusion.

Data security is a big concern in smart healthcare industry applications. Gope et al. [11] addressed the problems related to machine learning-based modeling attacks by proposing a physically-unclonable function (PUF)-based authentication mechanism. This proposed scheme has two phases: registration and authentication, to ensure secure connections and protection against man-in-middle attacks, replay, etc. However, weak PUF (WPUF) and one-time PUF (OPUF) can put the entire network at risk. Although their proposed scheme has shown decent protection against a few attacks, it does not protect against DDoS, Brute force, and bot attacks. These inferences and findings motivated us to develop a robust IDS for enhanced protection against potential attacks.

Deep learning-based intrusion detection was proposed by Tama and Rhee [17] to protect IoT networks. This model was evaluated using the NSLKDD dataset. Through this model, the author examined an IDS with the customary shallow model methods. This proposed IDS is executed with focused and appropriated validations. Test results showed that the distributed attacks detection schemes performed better in terms of accuracy. Additionally, the deep model is much better than the shallow model concerning the precision, recall, reliability, and F1 measures. Additionally, Primartha and Tama [18] proposed an IDS based on anomalies that make use of a gradient boosting machine (GBM) as a recognized automated machine. The GBM model ideal parameters are used in the grid-based search and the IDS uses three datasets, namely, UNSW-NB15, NSL-KDD, and GPRS, to approve using hold-out and cross-fold strategies. The proposed model was compared with previous approaches like fuzzy classifiers, GAR forest, and tree-based ensembles. It was found that the suggested model is better concerning accuracy, specificity, sensitivity, and area under the curve (AUC). A detailed summary of the highlighted paper is given in Table 2.

Table 2. Summary of some important previously proposed schemes.

Author	Attacks Coverage	Dataset	Methodology	Performance Metric
Gope et al. [11]	Man-in-middle, replay, DoS, security-against-any-learner, tracking, scalability	-	Physically unclonable function (PUF), registration, and authentication	-
Li et al. [22]	DDoS, bot, attacker-centric mitigation	UNSW NB15 dataset	FLEAM architecture	Mitigation response time; detection accuracy
Sonar and Upadhyay [15]	DoS, DDoS, volumetric flooding attack	-	Attack recover algorithm along with traditional Grey List and Black List	Packet delivery ratio
Lee et al. [14]	DoS, wormhole, selective forwarding	-	6LoWPAN Jammer	Energy consumption

2.2. *Classification Algorithm*

A hypothesis, popularly known by the name 'no free lunch', stated by Douglas et al. [19], reflects the importance of trying different things with different classifiers based on machines to solve classification tasks. The hypothesis expresses that "all optimization algorithms perform equally well when their performance is averaged across all possible problems [20]".

For the case we considered, we focused on only one type of classification algorithm: an ensemble classifier. Among the ensemble techniques, widely explored algorithms [23–25] are random forest (RF), gradient boosted machine (GBM), AdaBoost (AB), extreme gradient boosting classifier, and extremely randomized trees. Referenced classification is the fundamental explanation behind the choice of algorithm. Single classifiers such as classification and regression tree (CART) and multi-layer perceptron (MLP) have mainly been discussed as the number of input features was very large. Also, the presentation of ensembles for the CIDDS-001 and UNSW-NB15 datasets has not been read in and out. Lastly, the performance of ensemble classifiers in IoT-enabled appliances has rarely been focused upon, which prompted us to express this analytical study.

2.2.1. Ensemble Classifiers

The ensemble has been demonstrated for acceptable classification and regression algorithms. Subsequently, in this proposed strategy, we have used random forest and gradient boosting machine ensembles for analysis purposes.

2.2.2. Gradient Boosting Machine (GBM)

Also termed as gradient tree boosting or gradient boosted regression tree (GBRT) [24,25], GBM is a classifier technique that enhances the performance of classification and regression trees (CART) [26].

The classifier working with GBM requires different hyperparameters in the tuned format. This classifier requires that the same parameters be used for each dataset, so a grid-based search model was used to select the best parameters for the dataset. For experimental fulfillment, we used the GBM model which is implemented in the CatBoost (https://catboost.ai/docs/, accessed on 20 April 2021) library in a Python environment.

2.2.3. Random Forest

The random forest (RF) model uses the property of the decision tree-based model to build and aggregate a set of learning trees. It selects variables that are capable of being put into each model for random selection [27] for examples of predictors $\{t(x_{in}, \theta_n), n = 1, \cdots\}$ that have individual predictive outputs at a given input x_{in}. This predictive model relies only on a set of variables, θ_n, that admits to random selection, freely sampled with similar prevalence values. The core concept behind building a random forest model is that it accepts the simultaneous number of indicators that helps to achieve better accuracy while keeping away the issues of over-fitting values. Every indicator in a random forest develops to the largest size without sorting. When innumerable trees are created, they predict the information by deciding in favor of the most mainstream class on the input x_{in}. For the performance evaluation, the quality of estimators (trees) was set to 500 and the maximum depth for tree growth was 26 as suggested by Breiman [26,28] and Tama and Rhee [17]. Various parameters were obtained by using random search operations.

The novelty of the proposed model is the bagging technique, which uses the random forest to train different sets of models in a sequential manner. This bagging technique is ensembled with a gradient boosting machine. This ensemble technique learns from the model's citations of its previous residual errors and misclassifications to produce a strong learning model, whereas the previously introduced models try to enrich the model by updating the weights of the data points, which affects the performance of the model. Additionally, the performance of the proposed ensemble technique was estimated to be 2.9% more than that of the earlier models [18]. The proposed model was implemented with polynomial and standard scalar feature selection, rather than a grid search-based process, which enhanced the overall performance of the model by extracting the most relevant features from the input malicious attack data values.

3. Proposed Work

As presented in Figure 1, the traffic classification scheme consisting of four phases, including data preparation, data sampling, feature selection, and traffic classification, is proposed. In particular, the objective of data preparation was to enhance data quality, data integration, data cleaning, data transformation, and data normalization. After preparing the data, an under-sampling process was conducted to reduce the irregularity of the benign and malicious traffic samples, as the amount of specific data transmission traffic was much larger than that of regular transmission. The selection of features was done in such a way as to eliminate the unnecessary characteristics of attack behavior, improving the effectiveness of the data training and testing. Finally, the purpose of the GBM-based ensemble machine learning model was to identify malicious attacks.

To train a model, it is necessary to have a well-organized intrusion detection dataset. The network-based dataset should include modern network traffic scenarios, large varieties of low footprint intrusions, and deeply structured information on network traffic. Selecting such a dataset proved to be a significant challenge. Before finalizing the dataset for our research, we conducted a comparison-based study between the latest available intrusion detection datasets: UNSW-NB15-v2, TON-IoT-V2, BOT-IoT-V2, and CSE-CIC-IDS2018-V2, as shown in Table 3 and Figure 2.

The proportion of classes, i.e., benign, DoS, and DDoS, varies in each dataset. The CSE-CIC-IDS2018 dataset has incredibly vast coverage and a very high benign-to-attack ratio, which drew the attention of this research to the CSE-CIC-IDS2018-V2 dataset.

Table 3. Comparison of datasets on coverage of latest attacks [29–31].

Class	UNSW-NB15	ToN-IoT	BoT-IoT	CSE-CIC-IDS2018
Benign	1,550,712	270,279	13,859	7,373,198
DoS	5051	17,717	56,833	269,361
DDoS	0	326,345	56,844	380,096
Web Attacks	0	0	0	4394
Infiltration	0	0	0	62,072
Brute Force	0	0	0	287,597
Bot	0	0	0	15,683

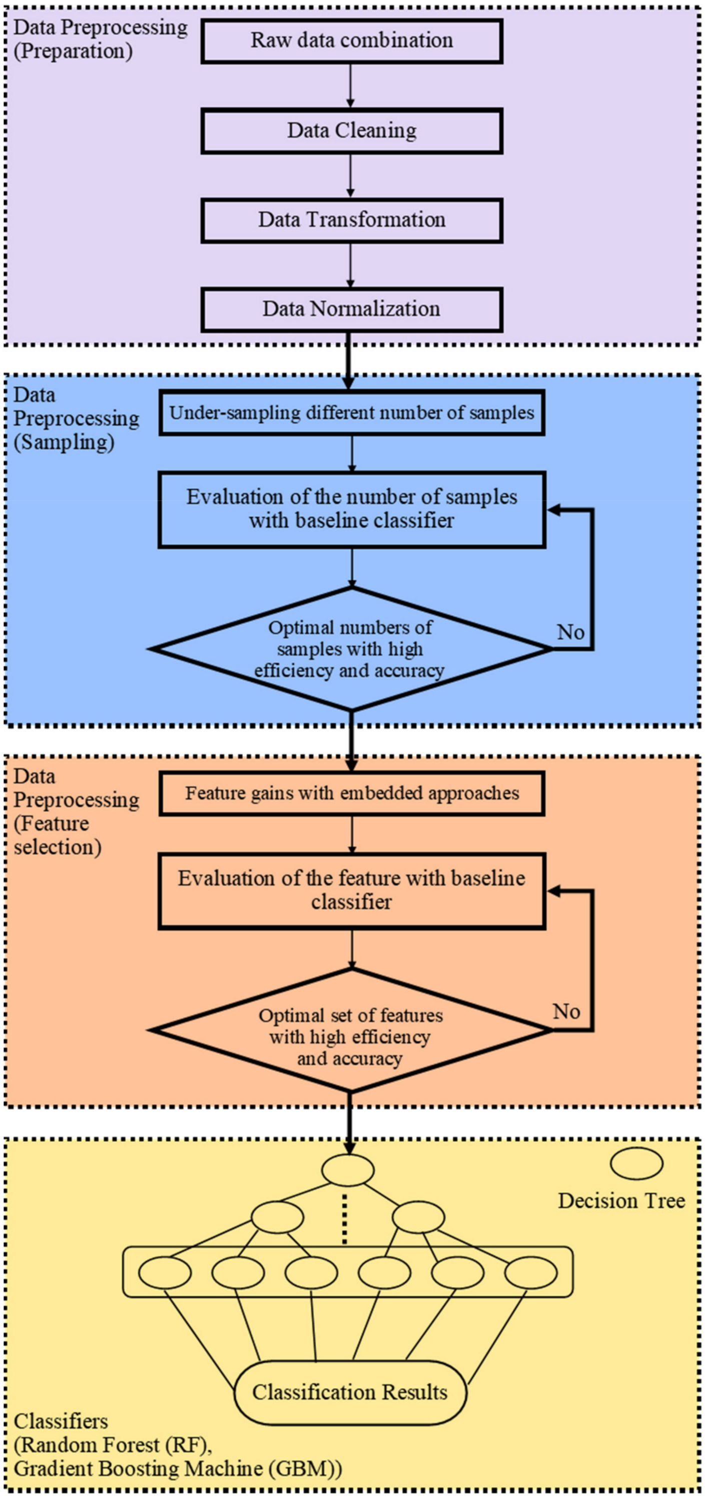
Data Preprocessing (Preparation)
Raw data combination
Data Cleaning
Data Transformation
Data Normalization
Data Preprocessing (Sampling)
Under-sampling different number of samples
Evaluation of the number of samples with baseline classifier
Optimal numbers of samples with high efficiency and accuracy
No
Data Preprocessing (Feature selection)
Feature gains with embedded approaches
Evaluation of the feature with baseline classifier
Optimal set of features with high efficiency and accuracy
No
Decision Tree
Classification Results
Classifiers (Random Forest (RF), Gradient Boosting Machine (GBM))

Figure 1. Proposed methodology.

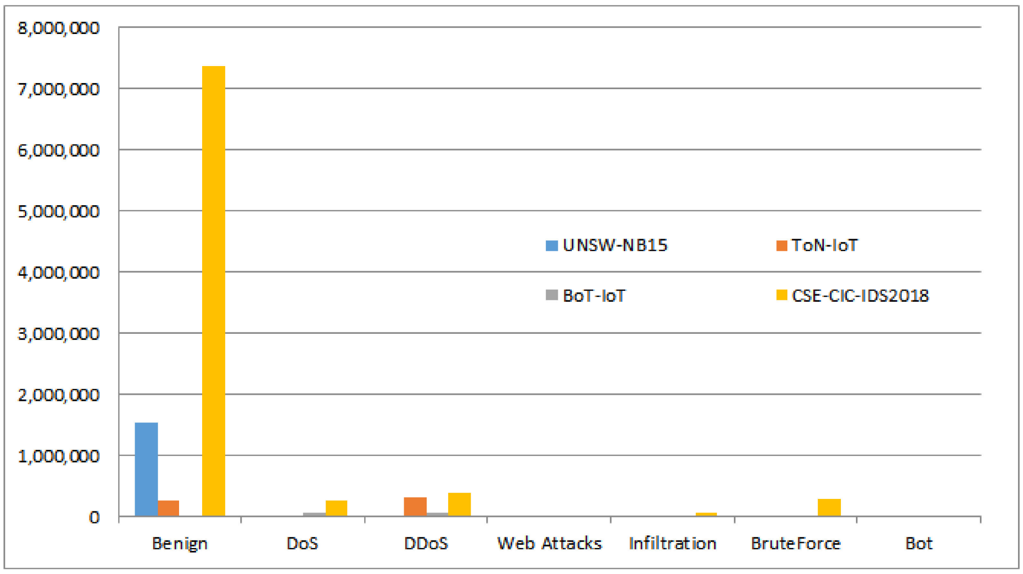

Figure 2. Summary of attacks covered by available datasets.

3.1. Data Preparation

In this section, the process of data preparation, including data integration, data cleaning, data transformation, and data normalization using feature scaling, is presented in detail.

3.1.1. Data Integration

The considered dataset "CSE-CIC-IDS2018-V2" on AWS was assumed to capture all available network traffic during 10 consecutive days of activities under a controlled network environment. All appropriate base traffic that was conducted on specific attack scenarios was controlled. The dataset also captured benign network traffic and the most well-known attacks. The unusual dataset consisted of 10 raw-data files, containing 16 million (approximately) individual network flows that cover common and different types of attacks, including Brute Force attacks, DoS attacks, DDoS attacks, Heartbleed attacks, web attacks, intrusions, and botnets. For the subsequent training phase, we combined all these files into the dataset.

3.1.2. Data Cleaning

A few samples of the raw dataset had some values missing. Therefore, medina imputation was used as a strategy to find those missing values. The mean/median of the non-missing values in each column is calculated and then the missing values within each column are individually substituted.

3.1.3. Data Transformation

As per the attack conditions study done by Sharafaldin et al. [27], we turned 15 types of different traffic flows into 9 types only: infiltration, SQL injection, Brute Force-Web, Brute Force-XSS, DoS Slowloris attacks, benign, bot, DDOS-HOIC attacks, and DDoS-LOIC-HTTP attacks [32,33]. Table 4 lists the sample numbers of these nine types of network traffic after changes.

The string labels were converted into integer labels in the range of 0 to 8. As the first integer number may cause halfway requesting in vector space-based estimation algorithms, we further utilized one-hot encoding to map discrete labels to Euclidean space, which made them continuous in each dimension.

Table 4. Distribution of 9 different types of traffic flow.

	Training Labels	Validation Labels	Test Labels
Infiltration	129,547	16,193	16,194
SQL Injection	70	9	8
Brute Force-Web	489	61	61
Brute Force-XSS	184	23	23
DoS Slowloris Attacks	8792	1099	1099
Benign	10,787,766	1,348,471	1,348,471
Bot	228,953	28,619	28,619
DDOS-HOIC Attacks	548,809	68,601	68,602
DDoS-LOIC-HTTP Attacks	460,953	57,619	57,619

3.1.4. Data Normalization Utilizing Feature Scaling

As the range of estimation was a very broad one, (the scope of port number varied from 1 to 65,535 and the range of the packet size varied from 1 to 1500), we scaled the data by utilizing a method of standard scaler as formulated in Equation (1). A huge number of samples, as much as 100,000, were required in every category of attack for the machine learning algorithms. For achieving this target, we performed upsampling by utilizing the method of synthetic minority oversampling provided by the SMOTE library:

$$standardization\ z = \frac{(x-\mu)}{\sigma} \tag{1}$$

where x is the original value of the features, mean is $\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$, and standard deviation is $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$.

3.2. *Undersampling*

The conversion ratio of benign traffic to absolute numbers was 84.14%, as shown in Table 4, and it caused an unbalanced classification problem. To avoid this issue, we randomly performed undersampling for benign traffic samples as given in Equation (2), where α_{us} represents the ratio of undersampling, N_{rM} is the quantity of the sample after undersampling, and N_m denotes the original samples:

$$N_{rM} = \alpha_{us} \times N_M \tag{2}$$

3.3. *Feature Selection*

We used XGBoost as the ensemble algorithm to select the dominant features for anomaly detection as it performs better in accuracy and robustness. The random forest method was comprised of multiple decision trees, and each decision tree is based upon randomly selected samples and features. When features selection was applied, all the samples were divided into two buckets. The results of classification were compared for these two buckets after randomly changing the value of a certain feature. The importance score of each feature was calculated using the Gini impurity [34] shown in Equation (3), where C denotes the data category and $p(i)$ represents the probability of each type of traffic that a random sample belongs to:

$$G = \sum_{i+1}^{c} p(i) \times (1 - p(i)) \tag{3}$$

3.4. Target Grouping

The binary target variable was created by target grouping all malicious network traffic into a single attack category represented by the positive class 1, whereas benign traffic was represented as the negative class 0. Through class weight calculation for each class, the proportion of samples of this class was given in the training set. For the majority class, the class weight was set to be 1 while for the minority class the class weight was calculated with the formula given in Equation (4):

$$Class - weight_{minority} = \frac{sum\_negative\_class}{sum\_positive\_class} \tag{4}$$

4. Experimental Results and Discussion

4.1. Experimental Setup

The machine used was a 64-bit Windows 10 Pro operating system with the following specifications: an Intel i7 vPro four-core processor, 3.60 GHz clock speed, and 16 GB main memory. For implementation and performance evaluation of the classifier, the Python (version 3.7.2) environment was used. Additionally, Python library Scikit-learn [35] was used to implement the model of the classifiers, as it highly supports the machine learning algorithms. The PARAM system operating on a 64-bit Ubuntu 14.04 was used for tuning the hyperparameters, and was equipped with an Intel Xeon Gold 6132 twenty-eight core processing CPU unit with a 2.64 GHz clock speed and 16 GB main memory. The IoT application is supported by the Raspberry Pi 3B+ model, which was enabled on an operating system named Raspbian equipped with a 64-bit quad-core ARM processing unit running at a 1.4 GHz clock speed and 1 GB of main memory. It was used to estimate the response time of the classifier. STAC is a web-based application that was used for a statistical test of the classes and the calculation of the performance results [36]:

$$avg.\ response\ time = \frac{\sum_{i=1}^{n_{test\_sample}} t_i}{n_{test\_sample}} \tag{5}$$

The computation formula of average response time represented in Equation (5), where i is the instance number; t_i is the total time consumed by the applied classifier to classify the ith test simple instance into two classes, i.e., attack or normal category; and n_{test_sample} represents the overall test instances.

4.2. Metrics and Method of Model Evaluation and Validation

The chosen range of the input parameters affected the general performance of the classifiers. The method of random search was used to explore the best optimal input parameters picked from the hyperparameters of the random forest technique and gradient boosting machine method for the CSE-CIC-IDS 2018 dataset. The method of randomized search CVs was used for the hyper-tuning of parameters, implemented in the Python programming environment using the Scikit-learn library. Randomized search CV finds the best optimal parameters settings by performing a method of cross-validation search given by the client. This test uses specific metrics to access the evaluation of the classifiers that are accuracy, specificity, or the rate of true negatives, and sensitivity, or the rate of true positives [36]. The metrics FPR and AUC were mathematically represented through Equations (6) and (7) as follows:

$$PR = \frac{FP}{(TN + FP)} \tag{6}$$

$$AUC = \int_0^1 \frac{TP}{(TP + FN)} d \frac{FP}{(FP + TN)} \tag{7}$$

In Equations (6) and (7), the term TP represents the true positive, which is the amount of truly classified attack instances and the term TN represents the true negative, which is the amount of true classified specified instances. The amount of incorrectly classified attack instances is defined by the term FP, false positive, and the FN, false negative, is the quantity of incorrectly classified normal instances. Accuracy is the absolute number of examples accurately classified to the total number of cases in the dataset. The sensitivity of the model represents the number of instances of general attacks accurately classified on all specific instances. The FPR represents the number of instances of misclassified attacks over the total number of common instances, and the AUC represents the referenced area under the receiver operating characteristics (ROC) curve, wherein the ROC curve is plotted against the FRP of the defined TRP.

For the computation process of the model and evaluation of full performance, we directed the experiment using a cyclic hold-out as in the cyclic *k*-fold cross-validations (in our case 10-fold) method [37]. As proposed by Rodriguez et al. [38], the cyclic version makes the error estimation stable and reduces the variance of the validation approach. For hold-out validation purposes, we divided the considered dataset into the 7:3 ratio, meaning 70% of the dataset was considered as training instances, while 30% of the dataset was considered as a testing instance. Additionally, for *k*-fold cross-validation, *k* is assumed to be 10. Cyclic 10f and 100 rounds of hold-out validation are considered as the models of classification which are assumed stable, indicating uniform prediction for the same dataset. To derive accurate information, all performance results reported in this paper were outputs estimated by 10 iterations of each cyclic validation approach and each experiment was repeated using an alternate seed which was also a contributing input to the random number generator.

The performance evaluation of the machine learning ensemble technique using the random forest model and the gradient boosting machine model with a hyperparameter search space, specifically with the CSE-CIC-IDS2018-V2 dataset, is discussed in this section. The results provide an optimal parameter configuration for both random forest and gradient boosting machine models and statistically analyze the performance of the models. To conduct the hyperparameter search to accomplish the task, Hyperopt was used, a distributed hyperparameter optimization [29,37,39,40] library providing a standardized optimization over awkward search space for serial or parallel searches. Through this experimental study, it was shown that the classifier used in the study is useful for detecting the intrusion of IoT appliances. First, we analyzed the performance results of hyperparameter optimization with the utilization of a random forest classifier.

4.3. Hyperparameter Optimization for the Random Forest Model

To optimize the parameters, a search-space needs to be defined, so for the random forest model the defined search space is as follows in Table 5.

Table 5. Hyperparameter search space for random forest.

Parameter Description	Parameter Identifier	Hyperparameter Search Space Training	Hyperparameter Search Space Result
Number of tree estimators	(n_estimators)	Uniform integer space in the interval of [10, 100].	The optimal number of tree estimators seems to be in the interval of [60, 100].
Information gain criterion	(criterion)	Choice of Gini or entropy.	The entropy criterion yields a better performance than the Gini criterion in our search run.
Maximum depth of a single tree	(max_depth)	Uniform integer space in the interval of [10, 100] or None resulting in an unbounded tree.	Restricting the depth of the trees seems to perform better than using unrestricted trees. A value in the interval of [20, 50] for the maximum tree depth yields a good performance.
Maximum features for the best split	(max_features)	Choice of sqrt or log2.	The sqrt option seems to perform better than log2.

Table 5. *Cont.*

Parameter Description	Parameter Identifier	Hyperparameter Search Space Training	Hyperparameter Search Space Result
Minimum samples to split a node	(min_samples_split)	Uniform integer space in the interval of [1, 10].	The optimal number of minimum samples to split a node is in the interval of [4, 6].
Minimum samples at a leaf node	(min_samples_leaf)	Uniform integer space in the interval of [2, 10].	Given our search run, there is no conclusive best option for the minimum numbers of samples in a leaf node. However, the values {3, 6, 7, 8} seem to perform well.

Before applying the random forest model over the dataset, a preprocess was applied to compute the best optimal parameters of the data as illustrated in Figure 3. The performance of the random forest model yielded a precision/recall score of 0.9810, with a precision of 0.967 and a recall of 0.955 with respect to the positive class as presented in Figure 4. The model performs quite well, except for the attack category infiltration, as presented in Tables 6 and 7.

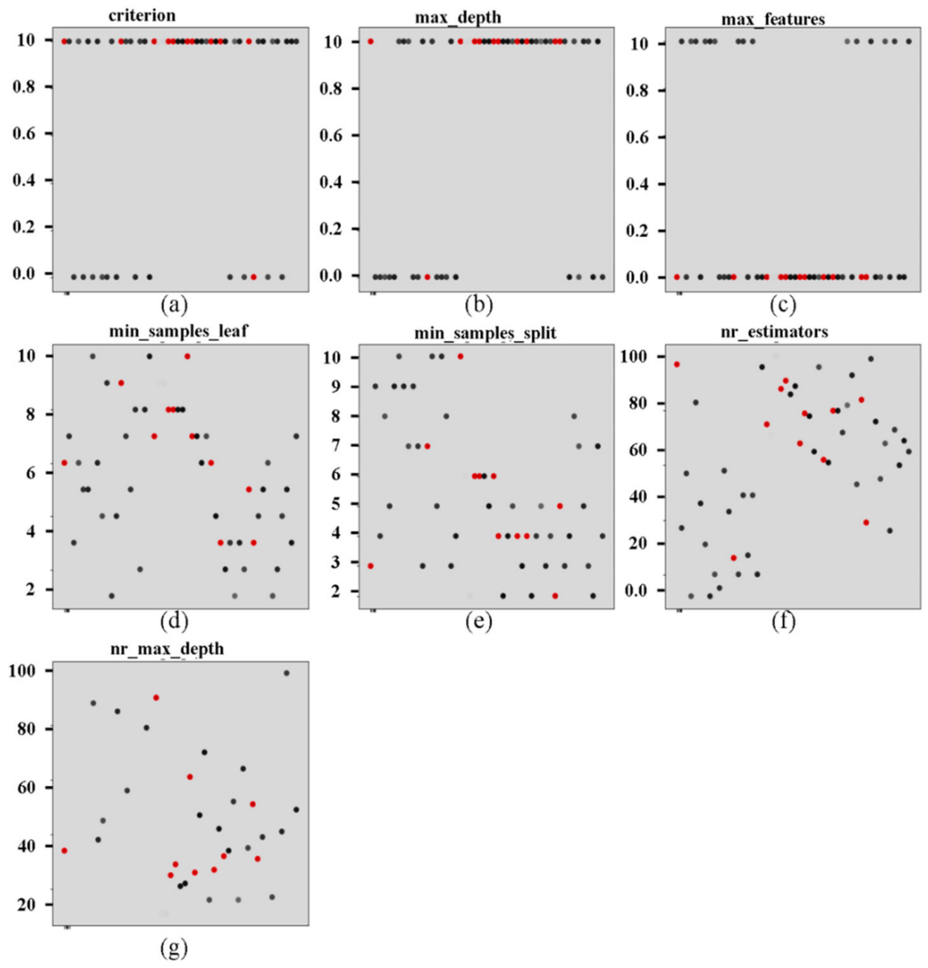

Figure 3. Scatter plot of best optimal model parameter with (**a**) 'criterion': 1, (**b**) 'max_depth': 1, (**c**) 'max_features': 0, (**d**) 'min_samples_leaf': 3.0, (**e**) 'min_samples_split': 4.0, (**f**) 'nr_estimators': 80.0, and (**g**) 'nr_max_depth': 32.0.

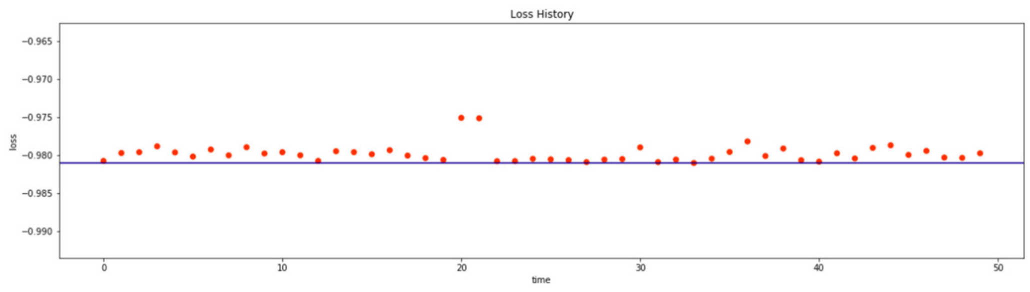

Figure 4. The loss-time graph with a best average error of random forest of −0.9810 and the finite loss range of −0.9810, −0.9751, and −0.9805.

Table 6. Classification report of model before using random forest model.

	Precision	Recall	f1-Score	Support
0	0.991	0.993	0.992	1,348,471
1	0.967	0.955	0.961	274,823
Accuracy			0.987	1,623,294
Macro avg	0.979	0.974	0.977	1,623,294
Weighted avg	0.987	0.987	0.987	1,623,294

Table 7. Misclassification by attack category.

	Misclassified	Total	Percent_Misclassified
Infiltration	12,252	16,193	0.756623
SQL Injection	1	9	0.111111
Brute Force–Web	4	61	0.065574
Brute Force–XSS	1	23	0.043478
DoS attacks-Slowloris	8	1099	0.007279
Benign	8820	1,348,471	0.006541
Bot	17	28,619	0.000594
DDOS attack-HOIC	22	68,601	0.000321
DDoS attacks-LOIC-HTTP	17	57,619	0.000295

The accuracy score of the random forest model, with consideration of precision and recall as visualized in Figure 5 with an average ratio of precision and recall, was 98.10%, and the computed confusion matrix over the true and predicted values is presented in Figure 6, which represents the number of true positives, true negatives, false positives, and false negatives.

This time, before applying the gradient boosting machine learning modeling technique over the dataset, a preprocess was applied to compute the best optimal parameters of the data as visualized in Figure 7. The performance of the gradient boosting machine model over loss time had an average error score of −0.9826, including the finite loss range pf −0.9827, −0.9720, and −0.9825 as presented in Figure 8. The model performed quite well, again except for the attack category infiltration. Tables 6 and 7 cover the classification report and the range of misclassification attack of the model before using the random forest model, respectively.

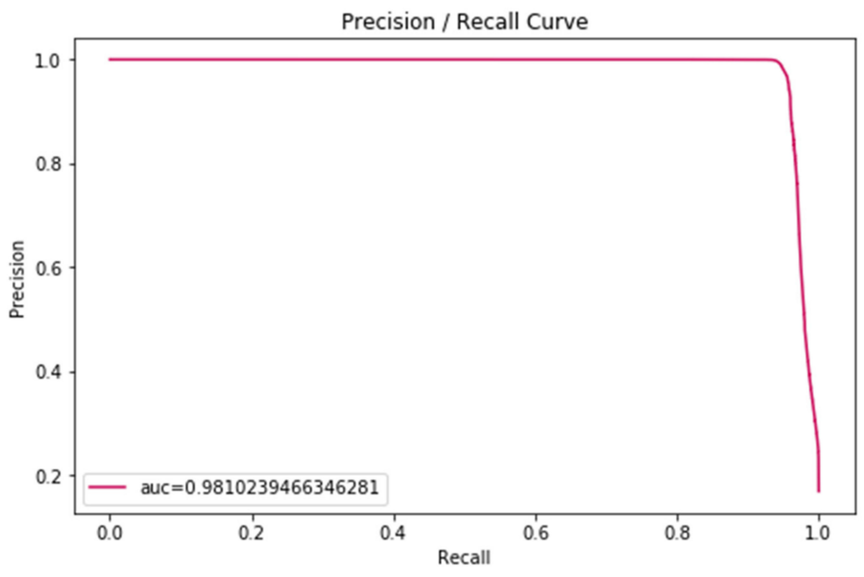

Figure 5. Precision-recall curve with average precision/recall score ≈ 98.10%.

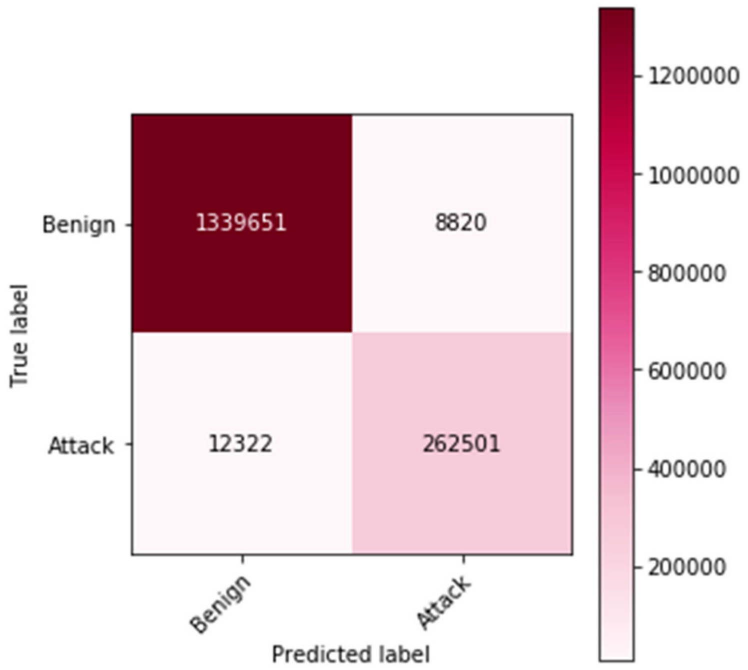

Figure 6. Confusion matrix without normalization.

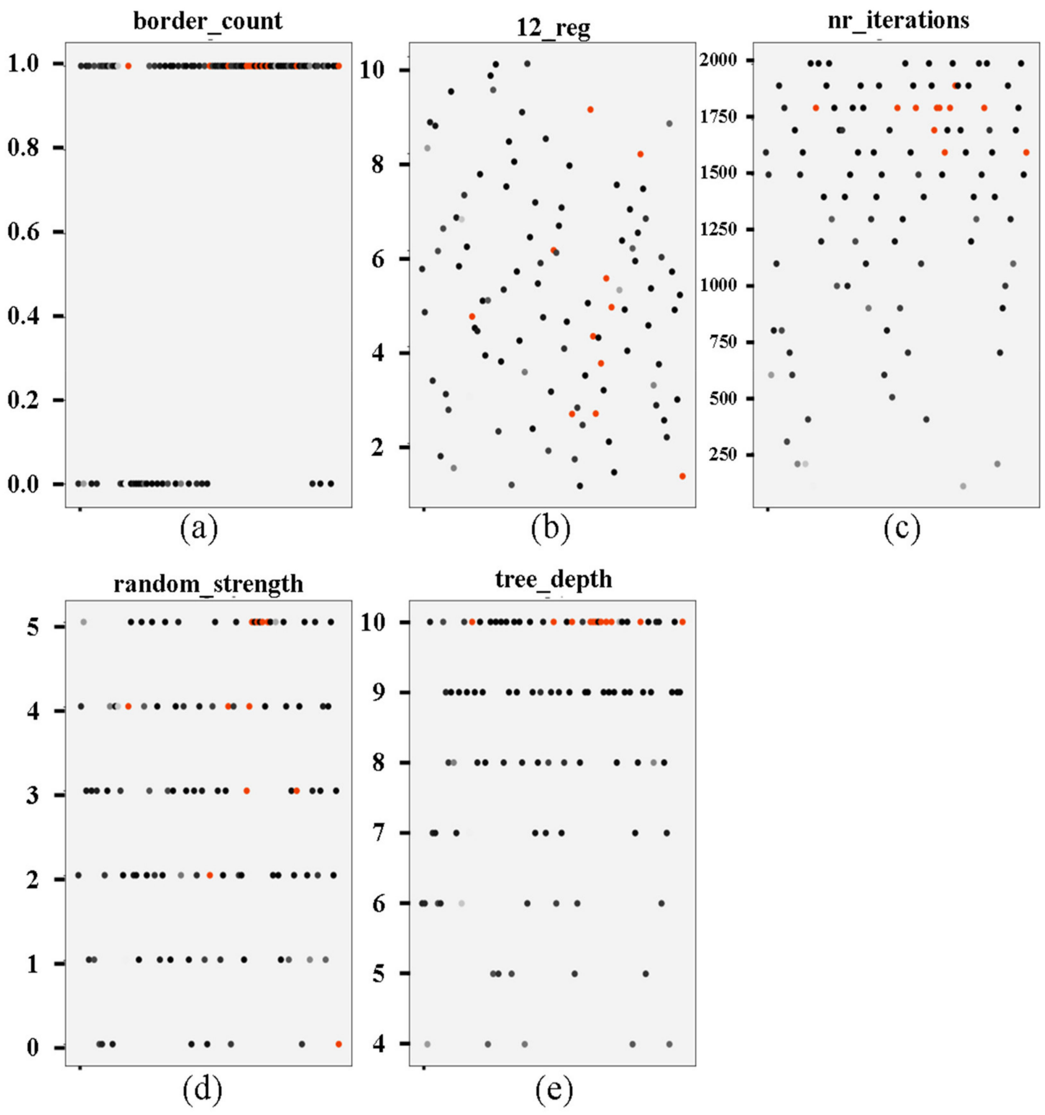

Figure 7. Scatter plot of best optimal model parameter with (**a**) 'border_count': 1, (**b**) 'l2_reg': 4.8139, (**c**) 'nr_iterations': 1900.0, (**d**) 'random_strength': 5.0, and (**e**)' tree_depth': 10.0.

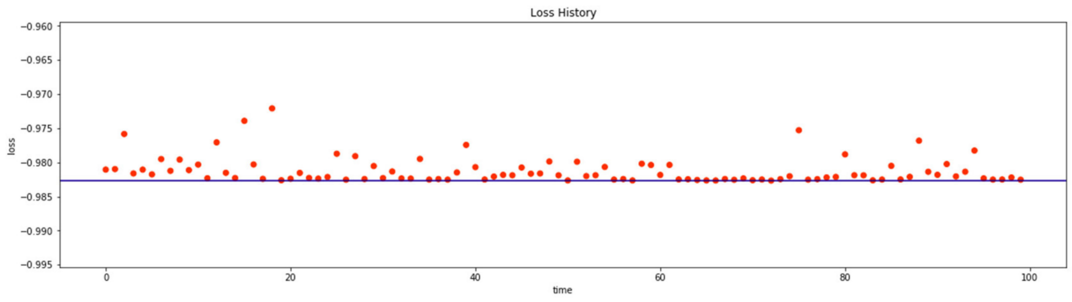

Figure 8. Loss-time graph with a best average error of gradient boosting machine of −0.9826 and a finite loss range of −0.9827, −0.9720, and −0.9825.

Hyperparameter optimization for gradient boosting machine (GBM) model. Again, the defined search space for the GBM model is as follows in Table 8.

Table 8. Hyperparameter search space for gradient boosting machine.

Parameter Description	Parameter Identifier	Hyperparameter Search Space Training	Hyperparameter Search Space Result
Maximum number of trees	(nr_iterations)	Uniform integer space in the interval of [100, 2000].	A value in the interval of [1600, 1900] seems to perform best.
Maximum depth of a tree	(depth)	Uniform integer space in the interval of [4, 10].	The optimal value for the maximum dept of trees is 10 in all best performing cases. This suggests that another round of hyperparameter search with a higher value might yield a better result.
L2 regularization coefficient	(l2_leaf_reg)	Uniform space in the interval of [1, 10].	A value in the interval of [2, 6] yields good results.
Number of splits for numerical features	(border_count)	A choice of 128 and 254.	A border count of 254 was chosen for all the best models.
Amount of randomness used for scoring splits	(random_strength)	Uniform integer space in the interval of [0, 5].	The optimal value seems to be in the interval of [3, 5].

The accuracy score of the gradient boosting machine model, with consideration of the precision and recall covered in Table 9, additionally Table 10 cover the range of misclassification attack of the model after using ensemble technique with random forest, and as visualized in Figure 9 with an average ratio of precision and recall, was 98.27%, and the computed confusion matrix over the true and predicted values is presented in Figure 10, which represents the number of true positives, true negatives, false positives, and false negatives.

Table 9. Classification report of model after applying ensemble technique with random forest.

	Precision	Recall	f1-Score	Support
0	0.991	0.993	0.992	1,348,471
1	0.964	0.957	0.961	274,823
Accuracy			0.987	1,623,294
Macro avg.	0.978	0.975	0.976	1,623,294
Weighted avg.	0.987	0.987	0.987	1,623,294

Table 10. Misclassification by attack category.

	Misclassified	Total	Percent_Misclassified
Infiltration	11,690	16,193	0.721917
SQL Injection	1	9	0.111111
Brute Force-XSS	1	23	0.043478
Brute Force-Web	2	61	0.032787
Benign	9784	1,348,471	0.007256
DoS-Slowloris Attacks	6	1099	0.00546
Bot	35	28,619	0.001223
DDoS-LOIC-HTTP Attacks	13	57,619	0.000226

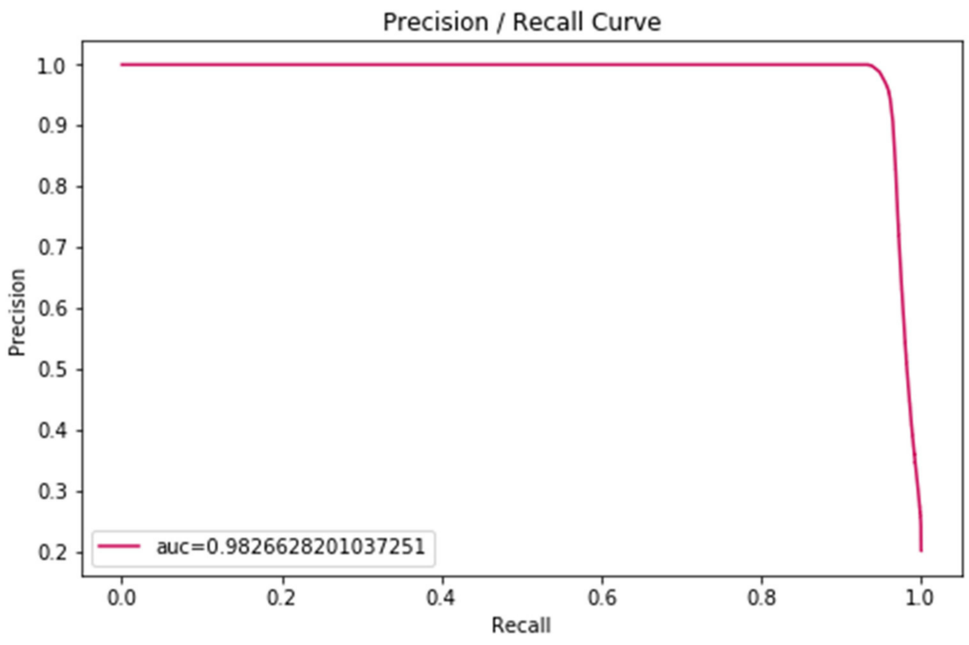

Figure 9. Precision-recall curve with average precision/recall score ≈ 98.27%.

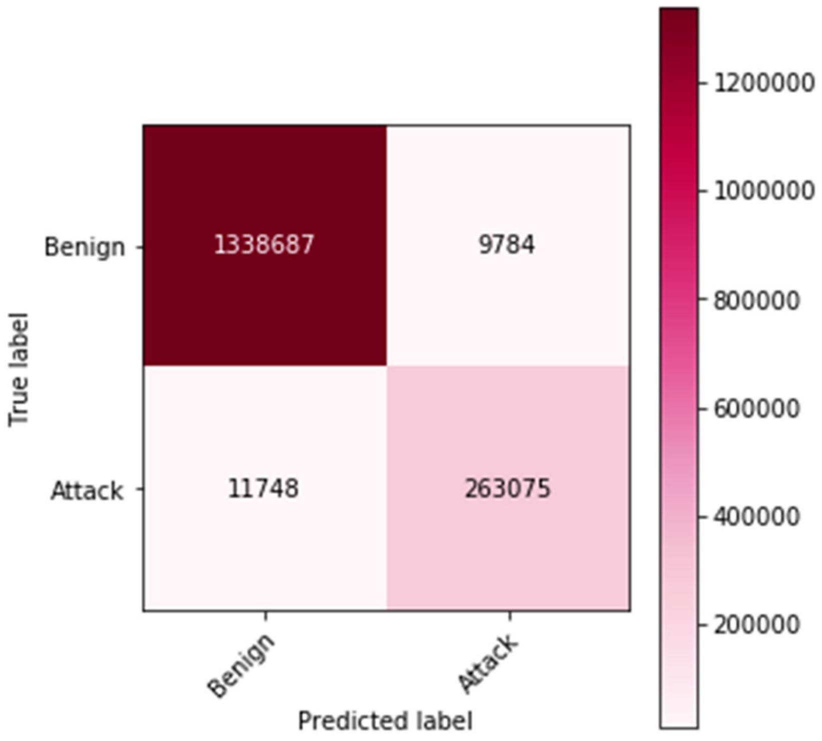

Figure 10. Confusion matrix without normalization.

Figure 11 presents the average response times that were taken by the classifiers individually and with the ensemble technique as the proposed model for classifying a single instance. From the results, it can be observed that the ensemble technique through bagging with random forest and boosting with gradient boosting machine took the least time to classify the instances of the CSE-CIC-IDS2018-V2 dataset.

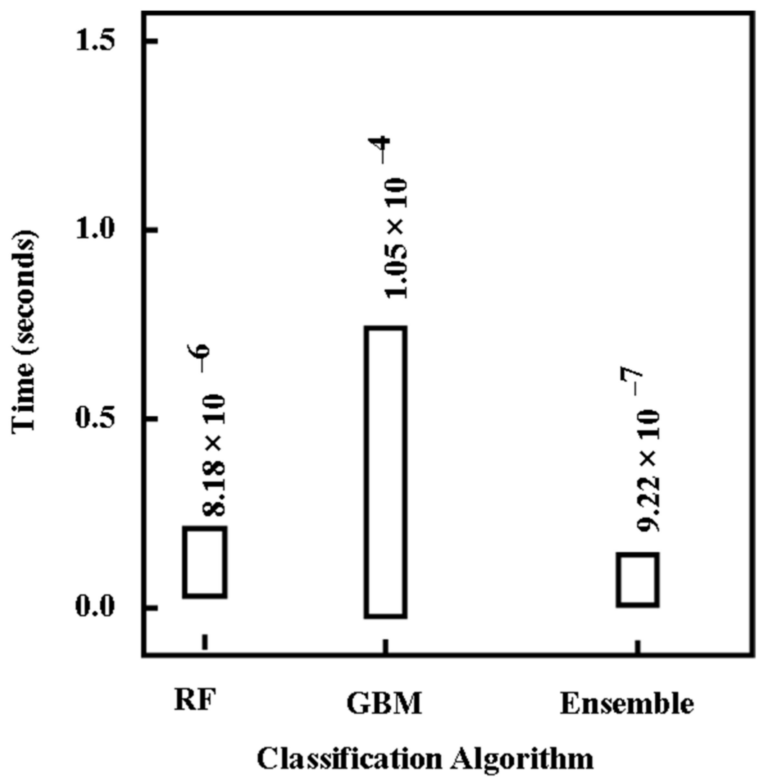

Figure 11. Average response times of classifiers.

The comparison results are shown in Table 11. The proposed model outperformed the most recent functions of intuition detection trained with the CSE-CIC-IDS2018-V2 dataset, that is, random forest + gradient boosting machine, in terms of accuracy. It was also better than the previous models: the border/router node convection model [15], the GBM model [17,18], and the random tree + NB tree model [18].

Table 11. Comparison results of proposed model with other models.

Method	Feature Selection	Accuracy Rate (%)	Significance Test Results
Border/router node convection [15]	No	73.88	No
GBM [17,18]	No	93.64	No
Random tree [18]	Yes	95.23	Yes
Proposed (RF + GBM)	**Yes**	**98.27**	**Yes**

5. Conclusions

The research work conducted used an experimental study on an anomaly-based intrusion detection system that is suitable for providing network security against DoS attacks to an IoT-enabled framework. The performance evaluation was done using two machine learning classification algorithms: random forest and gradient boosting machines. The hyperparameter optimization technique was used to find the best optimal parameters for the classifiers to train the models and compare their performances, respectively. The valuated performance of both the classifiers was measured in terms of accuracy, sensitivity, specificity, rate of false positives, and the area under the receiver operating characteristic curve. Benchmarking of both the classifiers was performed on the CSE-CIC-IDS-2018 dataset. The performance results indicate that the gradient boosting machine ensemble classifier performs best for binary classification, yielding a very attractive performance accuracy of 98.27%. Whereas, even on the validation and test sets, the classification of intrusion types was incorrectly falling for malicious network traffic. Additionally, the results show the trade-off between prominent metrics and response time, implying that both can be used for building IoT-specific anomaly-based intrusion detection systems.

Author Contributions: All authors contributed equally to this work. All authors have read and agree to the published version of the manuscript.

Funding: This work was not funded by any agency or organization.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset presented and during this work were collected by the Canadian Institute of Cyberspace (CIC) on Amazon's AWS LAN network. The data is publicly available in the 'CSE-CIC-IDS2018-V2 Dataset' at the link: https://registry.opendata.aws/CSE-CIC-IDS2018-V2/, accessed on 15 March 2021.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Choo, K.-K.R. The cyber threat landscape: Challenges and future research directions. *Comput. Secur.* **2011**, *30*, 719–731. [CrossRef]
2. Gaikwad, D.P.; Thool, R.C. Intrusion detection system using bagging with partial decision treebase classifier. *Procedia Comput. Sci.* **2015**, *49*, 92–98. [CrossRef]
3. Lin, W.-C.; Ke, S.-W.; Tsai, C.-F. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowl.-Based Syst.* **2015**, *78*, 13–21. [CrossRef]
4. Tran, N.N.; Sarker, R.; Hu, J. An approach for host-based intrusion detection system design using convolutional neural network. In Proceedings of the International Conference on Mobile Networks and Management, Melbourne, Australia, 13–15 December 2017; pp. 116–126.
5. Denning, D. An intrusion detection system. In Proceedings of the Symposium on Security and Privacy, Oakland, CA, USA, 7–9 April 1986; IEEE Computer Society Press: Los Alamitos, CA, USA, 1986; pp. 118–131.
6. Camastra, F.; Ciaramella, A.; Staiano, A. Machine learning and soft computing for ICT security: An overview of current trends. *J. Ambient Intell. Humaniz. Comput.* **2013**, *4*, 235–247. [CrossRef]
7. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [CrossRef]
8. Liao, H.-J.; Lin, C.-H.R.; Lin, Y.-C.; Tung, K.-Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
9. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A survey of intrusion detection techniques in cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57. [CrossRef]
10. Besharati, E.; Naderan, M.; Namjoo, E. LR-HIDS: Logistic regression host-based intrusion detection system for cloud environments. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 3669–3692. [CrossRef]
11. Gope, P.; Sikdar, B.; Millwood, O. A Scalable Protocol Level Approach to Prevent Machine Learning Attacks on PUF-based Authentication Mechanisms for Internet-of-Medical-Things. *IEEE Trans. Ind. Inform.* **2021**. [CrossRef]
12. Kasinathan, P.; Pastrone, C.; Spirito, M.A.; Vinkovits, M. Denial-of-Service detection in 6LoWPAN based Internet of Things. In Proceedings of the 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Lyon, France, 7–9 October 2013; pp. 600–607.
13. Suricata, I.D.S. Open-Source IDS/IPS/NSM Engine. 2014. Available online: https://cybersecurity.att.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview (accessed on 15 October 2021).
14. Lee, T.-H.; Wen, C.-H.; Chang, L.-H.; Chiang, H.-S.; Hsieh, M.-C. A lightweight intrusion detection scheme based on energy consumption analysis in 6LowPAN. In *Advanced Technologies, Embedded and Multimedia for Human-Centric Computing*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1205–1213.
15. Sonar, K.; Upadhyay, H. An approach to secure internet of things against DDoS. In Proceedings of the International Conference on ICT for Sustainable Development, Ahmedabad, India, 3–4 July 2015; Springer: Singapore, 2016; pp. 367–376.
16. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004; pp. 455–462.
17. Tama, B.A.; Rhee, K.-H. An in-depth experimental study of anomaly detection using gradient boosted machine. *Neural Comput. Appl.* **2019**, *31*, 955–965. [CrossRef]
18. Primartha, R.; Tama, B.A. Anomaly detection using random forest: A performance revisited. In Proceedings of the 2017 International Conference on Data and Software Engineering (ICoDSE), Palembang, Indonesia, 1–2 November 2017; pp. 1–6.
19. Douglas, P.K.; Harris, S.; Yuille, A.; Cohen, M.S. Performance comparison of machine learning algorithms and number of independent components used in fMRI decoding of belief vs. disbelief. *Neuroimage* **2011**, *56*, 544–553. [CrossRef] [PubMed]

20. Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; Herrera, F. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2011**, *42*, 463–484. [CrossRef]
21. Misra, S.; Krishna, P.V.; Agarwal, H.; Saxena, A.; Obaidat, M.S. A learning automata based solution for preventing distributed denial of service in internet of things. In Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, Dalian, China, 19–22 October 2011; pp. 114–122.
22. Li, J.; Lyu, L.; Liu, X.; Zhang, X.; Lv, X. FLEAM: A federated learning empowered architecture to mitigate DDoS in industrial IoT. *IEEE Trans. Ind. Inform.* **2021**. [CrossRef]
23. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
24. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
25. Breiman, L.; Friedman, J.; Stone, C.J.; Olshen, R.A. *Classification and Regression Trees*; CRC Press: Boca Raton, FL, USA, 1984.
26. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
27. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), Funchal, Portugal, 22–24 January 2018; pp. 108–116.
28. Louppe, G.; Wehenkel, L.; Sutera, A.; Geurts, P. Understanding variable importances in forests of randomized trees. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 431–439.
29. Nour, M.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
30. Smirti, D.; Pujari, M.; Sun, W. A Comparative Study on Contemporary Intrusion Detection Datasets for Machine Learning Research. In Proceedings of the 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Arlington, VA, USA, 9–10 November 2020; pp. 1–6.
31. Mohanad, S.; Layeghy, S.; Moustafa, N.; Portmann, M. Towards a standard feature set of nids datasets. *arXiv* **2021**, arXiv:2101.11315.
32. Krawczyk, B.; Minku, L.L.; Gama, J.; Stefanowski, J.; Woźniak, M. Ensemble learning for data stream analysis: A survey. *Inf. Fusion* **2017**, *37*, 132–156. [CrossRef]
33. Dietterich, T.G. Ensemble methods in machine learning. In Proceedings of the International Workshop on Multiple Classifier Systems, Cagliari, Italy, 21–23 June 2000; Springer: Berlin/Heidelberg, Germany, 2000; pp. 1–15.
34. Undercofer, J. Intrusion Detection: Modeling System State to Detect and Classify Aberrant Behavior. 2004. Available online: https://ebiquity.umbc.edu/person/html/J./Undercofer (accessed on 15 October 2021).
35. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
36. Ting, K.M. Sensitivity and Specificity. In *Encyclopedia of Machine Learning*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2011. [CrossRef]
37. Gaba, G.S.; Kumar, G.; Kim, T.-H.; Monga, H.; Kumar, P. Secure device-to-device communications for 5g enabled internet of things applications. *Comput. Commun.* **2021**, *169*, 114–128. [CrossRef]
38. Rodriguez, J.D.; Perez, A.; Lozano, J.A. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *32*, 569–575. [CrossRef] [PubMed]
39. Gaba, G.S.; Kumar, G.; Monga, H.; Kim, T.-H.; Liyanage, M.; Kumar, P. Robust and lightweight key exchange (lke) protocol for industry 4.0. *IEEE Access* **2020**, *8*, 132808–132824. [CrossRef]
40. Bergstra, J.; Yamins, D.; Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 115–123.