2022-03-22

Detecting IoT Attacks Using an Ensemble Machine Learning Model

Vikas Tomar
Graphic Era Deemed to be University

Sachin Sharma
Technological University Dublin, sachin.sharma@tudublin.ie

Recommended Citation

Tomer V, Sharma S. Detecting IoT Attacks Using an Ensemble Machine Learning Model. Future Internet. 2022; 14(4):102. DOI: 10.3390/fi14040102

Funder: NGIAtlantic H2020 project under agreement no. OC3-292

Article

Detecting IoT Attacks Using an Ensemble Machine Learning Model

Vikas Tomer [1] and Sachin Sharma [2,*]

[1] Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248002, India; vikastomer.cse@geu.ac.in
[2] School of Electrical and Electronic Engineering, Technological University Dublin, D07 EWV4 Dublin, Ireland
* Correspondence: Sachin.Sharma@TUDublin.ie

Abstract: Malicious attacks are becoming more prevalent due to the growing use of Internet of Things (IoT) devices in homes, offices, transportation, healthcare, and other locations. By incorporating fog computing into IoT, attacks can be detected in a short amount of time, as the distance between IoT devices and fog devices is smaller than the distance between IoT devices and the cloud. Machine learning is frequently used for the detection of attacks due to the huge amount of data available from IoT devices. However, the problem is that fog devices may not have enough resources, such as processing power and memory, to detect attacks in a timely manner. This paper proposes an approach to offload the machine learning model selection task to the cloud and the real-time prediction task to the fog nodes. Using the proposed method, based on historical data, an ensemble machine learning model is built in the cloud, followed by the real-time detection of attacks on fog nodes. The proposed approach is tested using the NSL-KDD dataset. The results show the effectiveness of the proposed approach in terms of several performance measures, such as execution time, precision, recall, accuracy, and ROC (receiver operating characteristic) curve.

Keywords: Internet of Things (IoT); machine learning; cybersecurity; DDoS

check for updates

Citation: Tomer, V.; Sharma, S. Detecting IoT Attacks Using an Ensemble Machine Learning Model. *Future Internet* **2022**, *14*, 102. https://doi.org/10.3390/fi14040102

Academic Editor: Paolo Bellavista

Received: 21 February 2022
Accepted: 22 March 2022
Published: 24 March 2022

1. Introduction

Historically, only computers, mobile phones, and tablets were connected to the Internet. The Internet of Things (IoT) today enables many kinds of devices and appliances (e.g., televisions, air conditioners, washing machines) to be connected to the Internet. IoT is being used in several fields today, including healthcare, agriculture, traffic monitoring, energy saving, water supply, unmanned air vehicles, and automobiles.

A three-layer IoT architecture is illustrated in Figure 1; from left to right: (1) thing layer, (2) fog layer, and (3) cloud layer. The thing layer includes IoT devices from several domains, including smart-homes, eHealth, smart vehicles, smart drones, and smart-cities. This layer enables data collection while having limited resources such as bandwidth, processing, energy, and memory. Next comes the fog layer, which is closer to the thing layer and may contain some operational resources to manage real-time operations and rapid decision making. Finally, the cloud layer facilitates the collection, processing, and storage of data in various data centers. However, as it is far away from the thing layer, it may take a long time to incorporate decisions in the thing layer.

According to a recent report from the International Data Corporation (IDC) (https://www.idc.com/, accessed on 20 March 2022), the amount of data generated by IoT devices will reach 73 zeta bytes by 2025, up from 18 zeta bytes in 2019. A massive influx of data opens up a lot of potential threats [1]. The problem is that IoT devices and their networks tend to be insecure since they are typically under-powered, memory-limited, or insufficiently bandwidth-limited to perform basic security functions such as encryption. IBM X-Force (https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/, accessed on 20 March 2022) reported in 2020 that attacks on IoT grew five-fold over the previous year. Currently, IoT-enabled networks are at risk of losing privacy and confidentiality due to malware and botnet attacks [2].

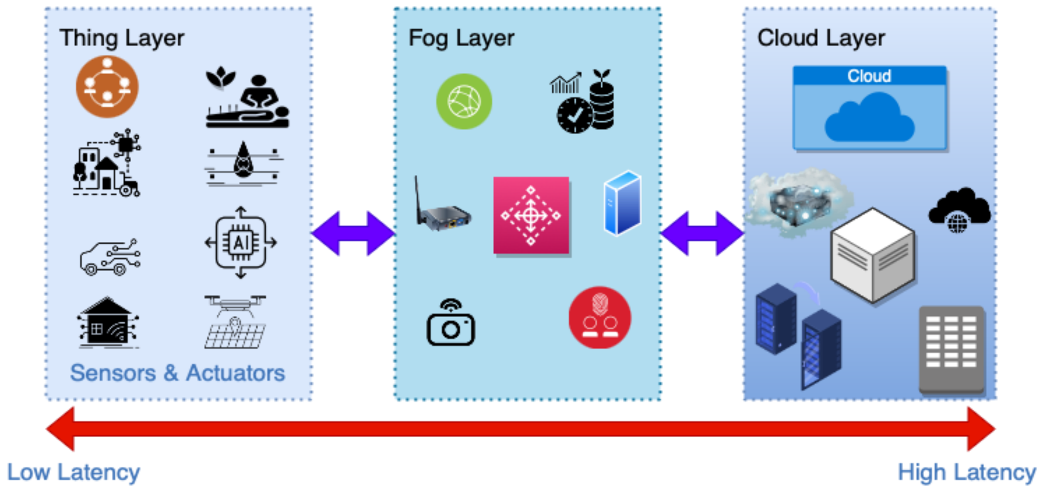

Figure 1. A three-layer Internet of Things (IoT) architecture.

For the IoT, several security solutions have been proposed, such as authentication [3], detection, and prevention [4]. Introducing machine learning (ML) algorithms into the IoT may alleviate concerns about security and privacy [5,6]. Today, it is crucial to decide where to run which algorithms for fast decision making, such as on the cloud or the fog or the thing layer. When all ML decisions are made in the cloud, IoT decisions may be delayed. In other layers, such as the thing or fog layer, it may be difficult to apply ML solutions due to their limited resources, such as bandwidth, processing, and energy.

Current research [7–12] indicates that deep learning algorithms are capable of detecting IoT attacks more effectively than traditional machine learning algorithms. However, only the cloud layer may have the resources to run these algorithms. In addition, these algorithms are not always very effective in some situations, such as remote live operations (e.g., remote surgery), since the system is supposed to make real-time decisions rapidly. Previous work on IoT attacks [9,13] has shown that a machine learning technique such as support vector machine (SVM) can only provide meaningful results if it is combined with a feature extraction/reduction algorithm or optimization algorithm. This combination of algorithms fails to meet the low resource requirement. ML techniques such as decision trees, naïve Bayes, K-nearest neighbors (KNN), and others are extremely robust for applications such as offline or non-interactive predictions between small datasets. These models, however, are considered weak when applied to real-time predictions. Studies conducted in the state of the art [14–16] report that the detection rate is quite low when using these classifiers to detect IoT attacks.

The paper proposes an ensemble model for an IoT system with limited bandwidth, processing power, energy, and memory (e.g., in the fog layer) to detect IoT attacks. Denial of service (DoS), authentication attacks, and probe attacks are taken into account. Moreover, no additional feature extraction or dimensional reduction algorithm is used to increase detection rates. This model is best suited to the real-time, quick detection of IoT attacks. In the proposed approach, there are two important steps: (1) selecting the best ensemble model that has a short execution time and high performance (e.g., accuracy), and (2) running the best model to achieve a short delay when applying the decision. Firstly, we perform the first step in the cloud, as more resources are required for selecting the best ensemble model, and the second step is performed in the fog layer, which has a low delay for real-time applications.

In this paper, extensive data analysis experiments are performed on the NSL-KDD dataset (https://www.unb.ca/cic/datasets/nsl.html, accessed on 20 March 2022). The dataset represents IoT attacks on a network in real time, and it is an upgraded version of the original KDD-99 dataset. The results show a high level of accuracy in a minimum amount of time with the fewest possible resources needed. The paper is organized as follows: Section 2 presents the related work and the background, Section 3 presents our proposed approach, Section 4 presents simulation scenarios, Section 5 provides the results and, finally, Section 6 concludes the paper.

2. Background and Related Work

2.1. IoT-Specific Attacks Overview

From IoT devices, data can be collected which can then be processed and monitored, depending on an application (e.g., e-healthcare or industrial) located in a cloud or fog layer. There are several attacks related to the IoT in the literature. Denial of service (DoS) attacks, authentication attacks, and probe attacks are presented below:

1. A denial of service (DoS) attack poses the greatest threat to IoT devices and servers with open ports [17,18]. There are several types of DoS attacks such as Smurf, Neptune, and Teardrop;
2. An authentication attack is an attack against privileged access. A remote to the user (R2U) attack (such as HTTPtunnel and FTP_write) occurs when an intruder sends malformed packets to a computer or server to which he/she does not have access. User-to-root (U2R) attacks (such as Rootkit) occur when a malicious intruder attempts to gain access to a network resource by posing as a normal user and then accessing it using full permission;
3. In a probe attack, an intruder runs a scan of a network device to determine potential vulnerabilities in the design of its topology or port settings and then exploits those in the future to gain illegal access to confidential information. There are several types of probe attacks, such as IPsweep, Nmap, and Portsweep.

2.2. ML-Specific Related Work on Security and Privacy

A comparison of related work on ML-specific attack detection can be seen in Table 1, including the ML (machine learning)/DL (deep learning) used, the pre-processing features, and performance analysis performed. During the pre-processing step, encoding (E), scaling (S), normalization (N), and dimensionality reductions (D) are taken into account. Furthermore, as part of the performance analysis, accuracy, receiver operating characteristic (ROC) curve, F-score, Matthews correlation coefficient (MCC), and detection rate (DR) are considered.

In [13,19,20], decision trees and rule induction are used to explain under what conditions a specific type of attack (DoS, authentication attacks, and probe attacks) occurs on a network. In this approach, encoding is used as a pre-processing technique, while accuracy is used to evaluate the effectiveness of the method. Although this is a valuable state-of-the-art approach, it cannot guarantee that any rules from decision trees will be applicable for large sets of data because overfitting poses the greatest risks. Further, in [21], principal component analysis (PCA) is utilized with a decision tree to detect and investigate the reason of the anomalies.

The previous works of [7,8,13,22,23] show that attacks can be predicted with high accuracy by using deep learning neural networks, either as a standalone technology [7,8] or in combination with optimization [22,23] or machine learning algorithms [9,13]. More precisely, [9,13] combine artificial neural networks (ANNs) with support vector machines (SVMs), which provide significantly higher detection rates than standalone deep learning or machine learning algorithms. Particularly, [13] develops the hybridization by including the SVM with ANN but also combining that fusion with a genetic algorithm (GA) and particle swarm optimization (PSO). This hybridization achieves a 99.3% accuracy rate.

Table 1. Related Work. The letters E, S, N, and D stand for encoding, standardization, normalization, and dimensional reduction, respectively. Further, accuracy, Matthews correlation coefficient, and detection rate are denoted as A, MCC, and DR, respectively.

Reference	**ML/DL Algorithm Used**	**Features Used (✓) or Not (×)**	**Analysis Performed (✓) or Not Performed (×)**
[19,20]	Decision Tree + Rule Induction	E(✓), S(×), N(×), D(×)	A(✓), ROC(×), FScore(×), MCC(×), DR(×)
[7,8]	Deep Neural Network (DNN)	E (×), S(×), N(✓), D(×)	A(×), ROC (×), FScore(✓), MCC(×), DR(×)
[22,23]	Optimization + DNN	E(✓), S(×), N(✓), D(✓)	A(✓), ROC (×), FScore(✓), MCC(×), DR(×)
[9,13]	SVM-ANN + hybrid optimization	E(×), S(×), N(✓), D(×)	A(×), ROC (×), FScore(✓), MCC(✓), DR(✓)
[21]	PCA + Random Decision	E(×), S(×), N(✓), D(✓)	A(✓), ROC (✓), FScore(×), MCC(×), DR(✓)
[10,11]	Dimensionality Reduction + DNN	E (×), S(×), N(✓), D(✓)	A(✓), ROC(×), FScore(✓), MCC(×), DR(✓)
[24]	GA-based Latent Dirichlet Allocation	E(✓), S(×), N(×), D(×)	A(✓), ROC (×), FScore(✓), MCC(×), DR(✓)
[25]	Autoencoder based LSTM classifier	E (✓), S(✓), N(✓), D(✓)	A(×), ROC (×), FScore(✓), MCC(×), DR(×)
[26]	Multinomial Logistic Regression	E(×), S(×), N(×), D(×)	A(×), ROC (✓), FScore(×), MCC(×), DR(×)
[27]	Ensemble Learning with XGboost	E (✓), S (×), N(×), D(×)	A(✓), ROC (×), FScore(×), MCC(×), DR(×)

The dimensionality reduction factor is also explored in a wide variety of works. The studies of [10] and refs. [11,12] used principal component analysis (PCA) with ANN and showed an efficacy of 91 percent F1-scores. Researchers from [28] have also explored dimensionality reduction with one-hot encoder and combined outlier analysis, which increased performance by 2.96 percent and 4.12 percent higher than CNN and RNN. This approach to dimensionality reduction with machine learning yields a mix of higher and average results. In addition, it is still unclear how many dimensionality reduction algorithms will fit within a single model to provide an optimal outcome. A combination of latent Dirichlet allocation (LDA) and a genetic algorithm is used in [24], which provides a below-average accuracy rate of 88.5 percent and a false positive rate of 6 percent.

The results are improved even more by techniques such as logistic regression and autoencoder. The study of [25] uses an autoencoder with LSTM and carries out experiments on a number of autoencoders, hitting the AUC score of 96 percent. Multinomial logistic regression provided a 99 percent ROC for finding anomalies in [26]. The idea of ensemble learning has also been explored by several authors. One of the appealing results, with 99.6 AUC, is provided by using XGBoost in [27].

The literature review covered almost all taxonomies of machine learning, from decision trees to neural networks, and from regression (logistic) techniques to ensemble learning. Following an extensive assessment, it was determined that a deep neural network with some optimization algorithm or ensemble learning could provide an impressive detection rate and the least false alarm rate of attacks. Additionally, feature engineering is also required to improve this model.

2.3. Voting and Stacking Techniques

The voting process, as its name suggests, ensembles the results of a number of weak classifiers by choosing the classifier with the greatest number of common traits as the final one. The advantage of this method is that it ignores errors of misclassified classifiers. As an example, to solve a classification problem through voting, a range of weak classifiers is selected, including K-nearest neighbor (KNN) classifiers and decision trees. Both naïve Bayes and K-nearest neighbour classifiers yield the same class label as a result, which differs from naïve Bayes. Following this, the maximum number of common votes from the K-nearest neighbor classifier and decision tree will be considered.

Stacking is a method of ensemble learning that takes into account heterogeneous weak classifiers, which means that different machine learning algorithms are combined. In addition, in stacking, there is the concept of a meta-layer that combines the classifier results from the base layers using a meta-layer model. For instance, to solve a classification problem through stacking, a range of weak classifiers, such as K-nearest neighbour classifiers, decision trees, and naïve Bayes classifiers are selected at base layers, and their results are combined through a neural network classifier as a meta-layer model. In the meta-layer model, the neural network will take inputs from the base layer and provide the outputs of these three weak classifiers with a final prediction.

2.4. Ensemble Machine Learning-Based Attack Detection

The authors of [29] demonstrate how ensemble machine learning, neural networks, and kernel methods can be used to detect abnormal behavior in an IoT intrusion detection system. In this study, ensemble methods outperform kernel and neural networks in terms of accuracy and error detection rates.

To detect webshell-based attacks, ensemble machine learning is used in [30]. In webshell attacks, a malicious script installed on a web server for remote administration executes malicious code written in popular web programming languages. Ensemble techniques, including random forest and extremely randomized trees, are applied in this work, and voting is used in order to improve their performance. The study concluded that random forests and extremely randomized trees are best for IoT scenarios involving moderate resources (CPU, memory, etc). Nevertheless, voting is proved to be most effective in scenarios requiring heavy resources. In [31], cyberattacks are detected using ensemble methods for IoT-based smart cities. Ensemble methods were found to be more accurate than other machine learning algorithms, including linear regression, support vector machines, decision trees, and random forests.

Further, anomalies are detected using ensemble methods applied to software-defined networking (SDN) in IoT at [32]. In SDN, IoT networks could be controlled from a central server called a controller [33,34]. Further, in [35], DDoS attacks are detected by using an ensemble method that uses traffic flow metrics to classify attacks. The applied approach yields fewer false alarms and a high degree of accuracy. Moreover, cyberattacks are detected by enabling cloud–fog architecture on the Internet of Medical Things (IoMT) using ensemble machine learning, in [36]. In this work, decision trees, naïve Bayes, and random forest machine learning techniques are used as a base classifier, and XGBoost is used at the next level. This method achieved a high detection rate of 99.98% on the NSL-KDD dataset.

The detection of anomalies in the smart home is carried out by ensemble machine learning rather than binary classification in [37]. Ensemble machine learning was able to detect anomalies in categorical datasets with minimal false positives. In [38], adaptive learning is used to boost the intelligence of ensemble machine learning for the Internet of Industrial Networks. This approach proved effective under ROC curve calculations.

2.5. IoT System with Cloud and Fog

Figure 1 illustrates the benefits of using the cloud for data processing because it may have the resources necessary to perform complex computations. The cloud, however, has several inherent weaknesses, including high costs, long latency, and limited bandwidth [39].

Further, due to proximity to IoT devices, fog is well suited for solving a variety of issues including long latency, communication, control, and computation [40]. With fog computing, time-sensitive data can be stored and analyzed locally [41]. Furthermore, by reducing the amount and distance of data sent to the cloud, IoT applications can be made more secure and private [42,43].

Researchers have employed a number of approaches and techniques to overcome data transfer challenges in fog, including encryption-based data transfer, as described in [44,45]. Furthermore, several researchers have proposed methods to improve security in fog, including game-based security [46]. However, these works do not have the advantage of functioning in real time. Currently, researchers are developing a method for predicting real-time scenarios and minimizing the overall time factor by balancing cloud computing with fog computing and optimizing the trade-off between the two (e.g., [47]). Likewise, this approach is used in our paper to move resource-intensive and time-sensitive tasks to the cloud and real-time tasks to the fog layer.

3. Proposed Approach

Our objective is to use ensemble machine learning techniques for detecting attacks in an IoT system. This is because deep neural networks require substantial resources, such as memory. The goal is to come up with the best ensemble method and to apply it for real-time attack detection. Figure 2 outlines the proposed approach with three layers: thing, fog and cloud. It involves the following three steps (also shown in Figure 2): (1) data collection at the cloud layer, (2) running the ensemble algorithm on the cloud and selecting the best model, and (3) running the best selected algorithm in the cloud. The description of the above tasks is given below.

1. **Data collection at Cloud Layer**
   This step involves collecting data from the thing layer and passing it to the cloud layer. To accomplish this, data from the thing layer can first be transported to the fog layer. The fog layer can then transport it to the cloud layer. While transporting the data to the cloud layer, the fog layer can also filter data to decide which data to be transported to the cloud. IoT attacks can be predicted using the following attributes: (1) login details, (2) the fields of network data packets, such as fragment details, protocol type, source and destination address, (3) service type, (4) flags, and (5) duration. We provide detailed information about the data used in our simulation in the next section.
2. **Selecting a best model on the cloud**
   The objective of this step is to combine various basic machine learning classifiers (such as naïve Bayes, KNN, and decision trees) with ensemble techniques (such as stacking, bagging, and voting) to obtain optimal results (accuracy, precision, execution time). As this is a time-consuming step, we recommend running it in the cloud. In addition, we simply apply the basic machine learning classifiers, as they require a short execution time.
   Figure 3 illustrates this step by including four layers: (1) the data layer, (2) the base layer, (3) the meta-layer, and (4) the method selection layer. In the data layer, collected data from the previous step is pre-processed and fed into the base layer. The base layer applies different combinations of base classifiers, such as naïve Bayes (B_1), decision trees (B_2), and KNN (B_3). The results of these combinations are then fed into the meta layer, where ensemble methods, such as stacking (E_1), bagging (E_2), and voting (E_3), aggregate the outcomes. Each ensemble method is evaluated in terms of accuracy, precision, recall, and ROC and execution time. Further, the model with a combination of base classifiers and an ensemble method that yields the best results is selected.
   Algorithm 1 describes the above-proposed approach in detail. The input parameters of the algorithms are: (1) base classifiers (i.e., $B = B_1, B_2, B_3, \ldots B_n$), (2) ensemble methods (i.e., $E = E_1, E_2, E_3, \ldots E_m$), and (3) training dataset (D). At the first two lines of the algorithm, the output and the result (i.e., variable OUTPUT and Result in

Algorithm 1) are initialized to *NULL*. The third line initializes the execution time to the maximum value.

In the fourth line, we store all the combinations of the base classifiers (i.e., using the function findAllCombinations) in variable *C*. The proposed approach aims to determine the best combination and the best ensemble method. Therefore, in line 5, we iterate each of the combinations, and then, again, in line 7, each base classifier in the corresponding combination is iterated. Each base classifier is applied to the training dataset (D) with the outcome being stored in *o* (line 8).

Line 10 involves an iteration of the ensemble methods and the application of each ensemble method to the outcome (*o*) at step 11. At line 12, the ensemble result is calculated in terms of accuracy, precision, recall, etc. Further, at line 13, the execution time of the combination of base classifiers and an ensemble method is calculated. The new result (r) and execution time (time) is then compared to the previous best result (Result) and time (ExcecutionTime). If this is the best result so far, the corresponding combination and ensemble method is stored in the output (OUTPUT); see line 14. Further, the result is stored in line 15. In the end, the best output is returned at line 21.

3. **Running the best model on the fog layer**
   This step involves executing the model selected in the previous step over the fog layer with the real-time data collected from the thing layer. The model consists of a combination of base classifiers and an ensemble method.

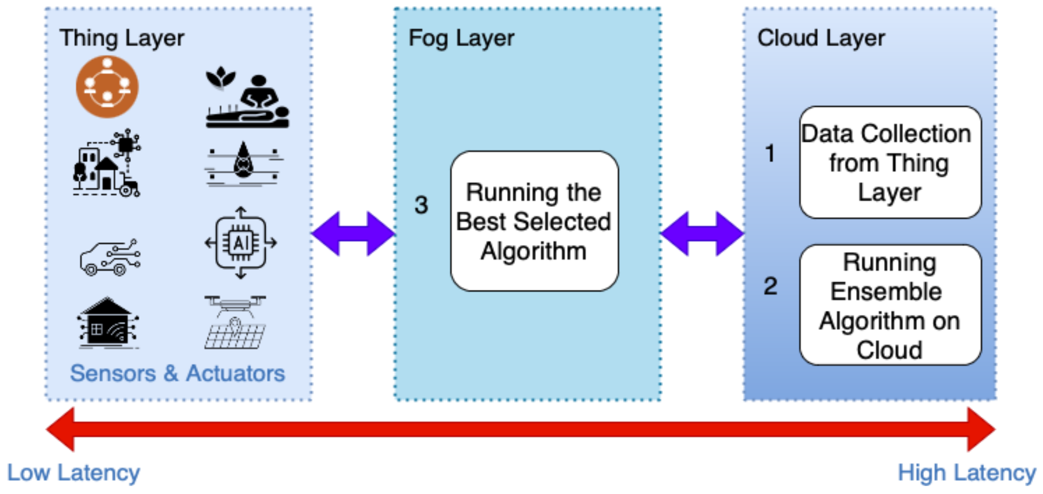
Thing Layer
Sensors & Actuators
Fog Layer
3
Running the Best Selected Algorithm
Cloud Layer
1
Data Collection from Thing Layer
2
Running Ensemble Algorithm on Cloud
Low Latency
High Latency

Figure 2. Proposed approach.

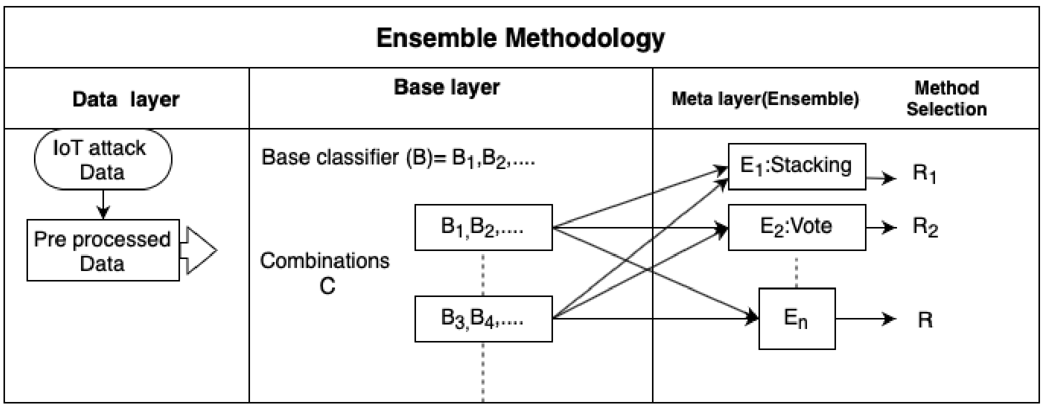

Figure 3. Selection of an ensemble method.

Algorithm 1: Find a best model

```
procedure FINDABESTMODEL($B, E, D$)
    // $B \leftarrow B_1, B_2, B_3,$ ... $B_n$. Here, $B_1, B_2, B_3,$ ... $B_n$ are Base Classifiers.
    // $E \leftarrow E_1, E_2, E_3,$ ... $E_m$. Here, $E_1, E_2, E_3,$ ... $E_m$ are Ensemble methods.
    // D is the training dataset.
 1  $OUTPUT \leftarrow NULL;$
 2  $Result \leftarrow NULL;$
 3  $ExecutionTime \leftarrow MAX;$
 4  $C \leftarrow findAllCombinations(B);$
    // Find all the combinations of the Base Classifiers (B).
 5  foreach $c \in C$ do
        // Iterate each combination.
 6      $o = 0;$ // Initialize an outcome.
 7      foreach $B_a \in c$ do
            // Iterate each Base Classifier in c.
 8          $o \leftarrow o, ApplyBaseClassfier(B_a, D);$
            // Apply $B_a$ over dataset D and store the outcome of each $B_a$ in the form of the ROC curve or any performance measure in o.
 9      end
10      foreach $E_a \in E$ do
            // Iterate each Ensemble method.
11          $e \leftarrow applyEnsembleMethod(E_a, o)$ ;
            // Apply Ensemble Method $E_a$ on $o$.
12          $r \leftarrow findResult(e);$
            // Find the result in form of ROC or any other Performance measures
13          $time \leftarrow findExcecutionTime(c, E_a);$
            // The execution time of the combination of base classifiers (c) and an ensemble method ($E_a$)is calculated
14          if isResultBetter(r, time, Result, ExecutionTime) then
                // If r and time is better than Result and ExecutionTime.
15              $OUTPUT \leftarrow c, E_a;$
                // Store the base classifiers and Ensemble method over OUTPUT.
16              $Result \leftarrow r;$
17              $ExecutionTime \leftarrow time;$
18          end
19      end
20  end
21  $Return : OUTPUT;$
    // Return the best model with base classifiers and an ensemble method
```

The proposed approach to include cloud–fog/edge architecture is derived from the analysis of an NGIAtlantic EU project [48], in which cross-Atlantic experimental validation is proposed for intelligent SDN-controlled IoT networks. In this project, IoT devices transmit data to an IoT application in the cloud over the Internet via a gateway (located at edge/fog devices) whose security and latency are enhanced by running secure network functions. Our approach is a practical solution in real-time for such a scenario since, in production IoT networks, fog/edge nodes do not have a lot of resources to run heavyweight algorithms that require a lot of resources. Therefore, if only the trained model is run in the fog layer (step 3, above), the fog node's resource requirements will be lowered, which is practical. Furthermore, since the cloud layer has plenty of resources, it makes sense to train the data there, as described in steps 1 and 2.

4. Simulation Environment

This section presents the simulation environment in terms of server configuration, dataset description, cloud and fog data separation, and simulated base classifier and ensemble methods.

4.1. Server Configuration

The proposed framework with fog and cloud nodes is tested on a server with a CPU Core E7400 processor and 3.00 GB of RAM and a 32-bit operating system with 2.80 GHz. The proposed ensemble algorithm is implemented on the cloud node and the best model is run on the fog node. The Weka platform is used to run the experimentation at the cloud layer and the real-time detection of IoT attacks at the fog layer.

4.2. Dataset Description

The NSL-KDD dataset (https://www.unb.ca/cic/datasets/nsl.html, accessed on 20 March 2022) is used for the simulation of this work. It contains 41 features to describe each specific entity in an IoT network. Details on network intrusions with these 41 features can be segmented into computational information (service, flag, land, etc.), content-based information (login information, root shell information, etc.), duration-based (such as duration from host to destination transfer, error rates), and host-based information (host and destination ports and counts information).

In Figure 4, the NSL-KDD dataset is represented by two layers: (1) the inner layer represents different types of IoT attacks in the dataset, such as Probe, DoS, U2R, and R2L; (2) the outer layer represents examples of attacks within each category. Attacks such as Saint, Satan, Nmap, and portsweep, which can be found in Figure 4, come under the Probe IoT attack category. In these attacks, the attacker scans a network device to determine potential weaknesses in its design, which are subsequently exploited in order to gain access to confidential information, as described in Section 4.

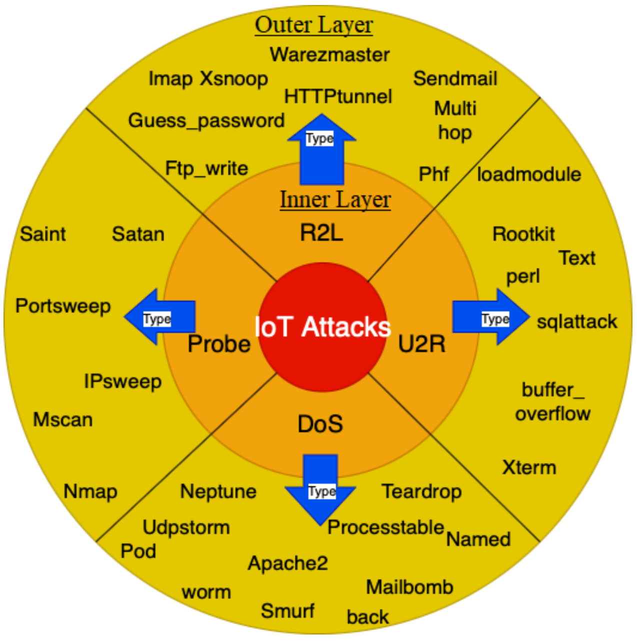

Figure 4. Layerwise NSL-KDD dataset description.

Likewise, attacks such as Neptune, Teardrop, Worm, and Smurf fall into the category of DoS attacks. These attacks cause a denial of service when an attacker consumes resources unnecessarily, making the service unavailable for legitimate users. Moreover, Sendmail, Multihop, and phf belong to R2L (remote-to-user) attacks, while Perl, text, and sqlattack belong to U2R (user-to-root) attacks. In Figure 4, variables are underlined according to their segment. Most variables in this dataset are nominal. There are three basic protocol types, TCP (transmission control protocol), UDP (user datagram protocol), and FTP (file transfer protocol), that exist in the dataset.

4.3. Data Separation for the Cloud and Fog Layers

Our proposed scheme uses the cloud layer to keep track of historical data about network connections associated with IoT attacks, while the fog layer analyzes real-time data. Furthermore, the cloud layer consists of the target variable and its associated labels, whereas the fog layer requires this variable to be predicted for new entries or labels. Training and testing data segments are provided in the NSL-KDD dataset source. For experimentation, training data is used as cloud data, and testing data as fog data. Further, a significant subset of the NSL-KDD dataset is used in the cloud layer for training and validation, while the rest of the unlabeled data is considered for real-time processing in the fog layer for testing. Moreover, K-cross validation is used with an 80:20 ratio at the cloud layer.

4.4. Simulated Base Classifiers and Ensemble Methods

Simulating the proposed approach included the use of five machine learning classifiers and two ensemble methods. The classifiers used are: (1) decision tree (DT), (2) random forest (RF), (3) K-nearest neighbors (KNN), (4) logistic regression (LR), and (5) naïve Bayes (NB), while ensemble techniques are voting and stacking. Table 2 shows the detail of each combination of base classifiers in the base layer. A total of 10 different model combinations are tested. The models are listed in Table 2. This is because we selected five base classifiers, and we created combinations of two. Therefore, we end up with 10 models (i.e., ${}^{5}C_{2}$).

Table 2. Base classifier combinations: decision tree (DT), random forest (RF), K-nearest neighbor (KNN), logistic regression (LR), naïve Bayes (NB).

Model	Base Classifier Combinations		
1	DT	RF	KNN
2	RF	KNN	LR
3	KNN	LR	NB
4	LR	NB	DT
5	NB	DT	RF
6	DT	KNN	LR
7	RF	LR	NB
8	KNN	NB	DT
9	LR	DT	RF
10	NB	RF	KNN

5. Results and Analysis

Here, we evaluate the results of the proposed approach for the cloud and fog layers using three factors: (1) execution time, (2) performance measures, and (3) error associated with the final model. On the cloud layer, a larger amount of data (training) is used to build models and conduct experiments. Testing data is considered new data and is tested on the fog layer. In the cloud layer, the best model is selected, and in the fog layer, it is evaluated using real-time data. Our first objective is to summarize the results, including the cloud layer, and the method by which model 8 (distributed in Table 2), with an ensemble method, was selected to be applied to the fog layer. Following that, we show the results obtained from the real-time data in the fog layer.

5.1. Cloud Layer Result Analysis

5.1.1. Execution Time

Figure 5 displays the execution time for voting and stacking ensemble methods over all the models described in Table 2. The X-axis in Figure 5 refers to the duration in seconds to execute a model, while the Y-axis refers to the model number. Compared to the voting ensemble method, stacking takes a much higher execution time. According to our results, model number 8, with the voting technique, shows minimal execution time (9.96 s), with KNN, NB, and DT used as base classifiers.

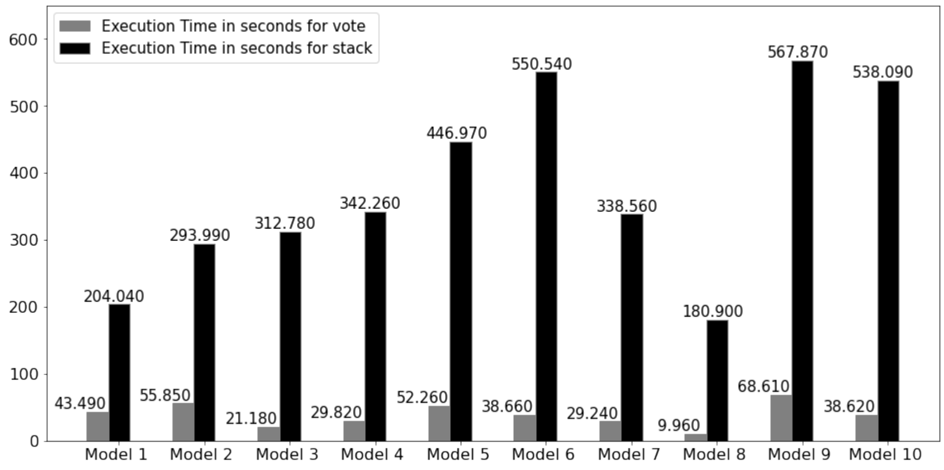

Figure 5. Execution times of all models.

5.1.2. Performance Measures

Figure 6 shows overall performance as measured by kappa, F-measure, and the ROC area. It shows that all the models have values greater than 0.99, with model 8 providing the kappa value 0.991, the F-measure value 0.995, and the ROC area 0.999. Figure 7 shows the errors with voting as an ensemble method in terms of mean absolute error, root mean square error, relative absolute error, and root-relative squared error. Model 1, with voting, exhibits significantly fewer errors than any other model. In this model, DT, RF, and K-NN are used as base classifiers, and voting is used as an ensemble technique. In spite of this, we selected model 8 with voting to run in the fog layer, as it performed well in terms of execution times and other performance parameters, as shown in Figure 6. Based on Figure 7, the root-relative squared error in model 8 with voting has the greatest impact, of 27.94 percent, and the mean absolute error has the least impact, of 0.6 percent.

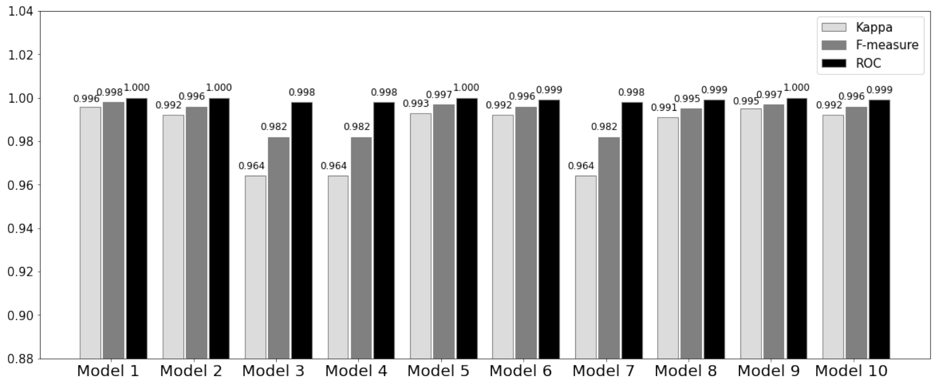

Figure 6. Performance of all models.

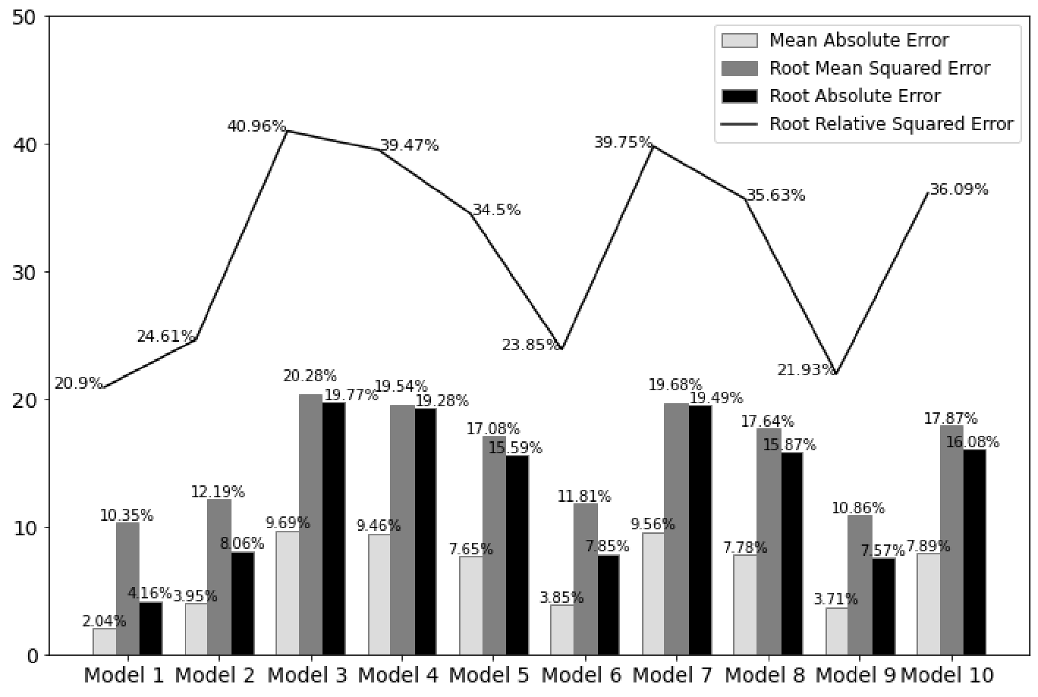

Figure 7. Errors associated with all the models.

To verify further, we calculate the performance of model 8 in terms of precision, F-measure, MCC, and PRC area (Figure 8), in addition to all other metrics. Through the Y-axis, the result is accurate to three decimal places. The most significant performance metric is MCC, which indicates how random or real the prediction is. It ranges from −1 to 1. Model 8's values in the experiment are typically closer to 99.99 percent. In general, model 8 with voting is highly optimized to run on the fog layer, according to the requirements of real-time execution and excellent performance.

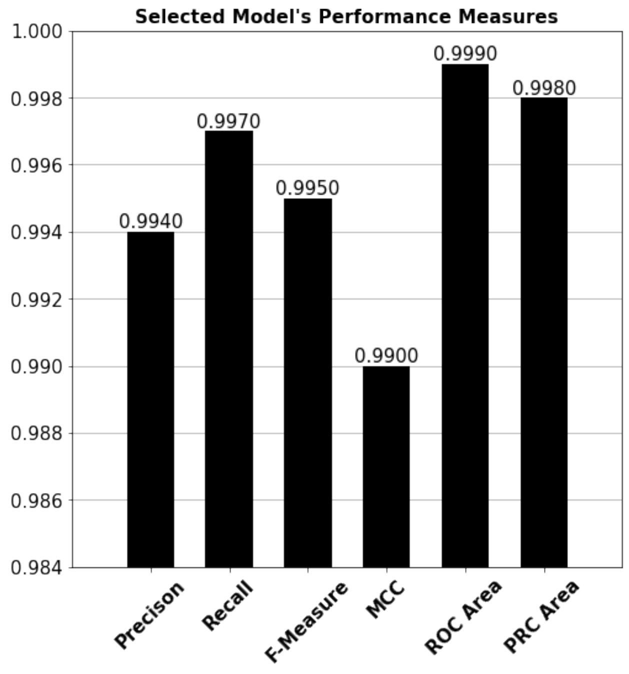

Figure 8. Performance of the selected model.

We found that model number 8, using K-nearest neighbor, naïve Bayes, and decision trees as the base classifiers outperforms all other models with respect to execution time and performance metrics (such as kappa, F-measure, ROC, and MCC). Since time is an important factor in the selection of any model, the voting ensemble technique determines that model 8 takes the least time: 1.15 s. Additionally, kappa, F-measure, ROC, and MCC have maximum values of 6.39, 98.20, 99.60, and 96.40, respectively. There is also a mean absolute error of 7.78 percent, a root mean square error of 17.64 percent, a relative absolute error of 15.87 percent, and a root-relative squared error of 35.63 percent. Further, the root-relative squared error of model 8 is 27.94 percent, and the minimum impact is 0.6 percent. In fact, model 8 is the most time-efficient and resource-intensive model, which is why it has the greatest impact.

5.2. Fog Layer Result Analysis

With the new data now being included, we measure the performance of model 8, with this model having KNN, NB, and DT as the base classifiers, as well as voting as an ensemble model.

5.2.1. Performance Measures

Performance measures such as kappa, F-measure, and ROC indicate how well the model performs in the fog layer. Figure 9 illustrates that all performance indicators in the selected model are almost equal and at the top. The values are 96.39, 98.20, 99.60, and 96.40 for kappa, F-measure, ROC, and MCC, respectively.

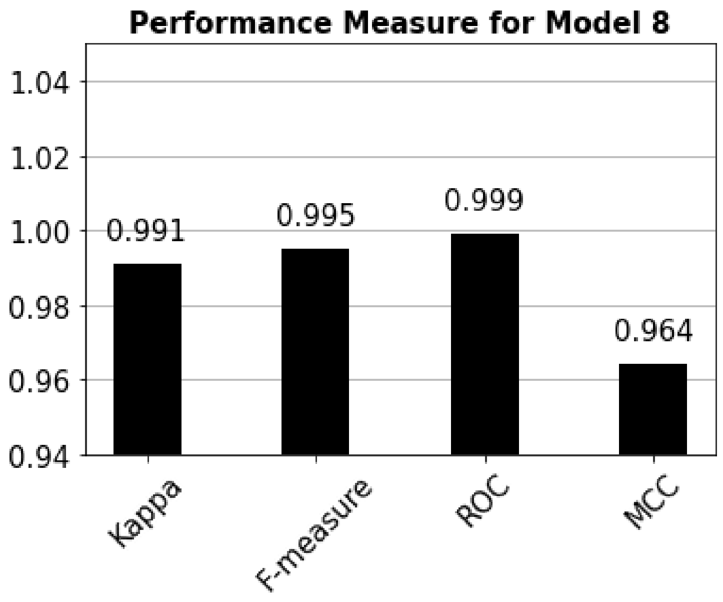

Figure 9. Performance on the fog node (using a model with KNN, NB, and DT as the base classifiers as well as voting as an ensemble method).

5.2.2. Errors Associated

Figure 10 represents the mean absolute error (MAE), root mean square error (RMSE), relative absolute error (RAE), and root-relative squared error (RRSE). Our experiment yielded mean absolute error, root mean square error, relative absolute error, and root-relative squared error values of 7.78, 17.64, 15.87, and 35.63 percent, respectively.

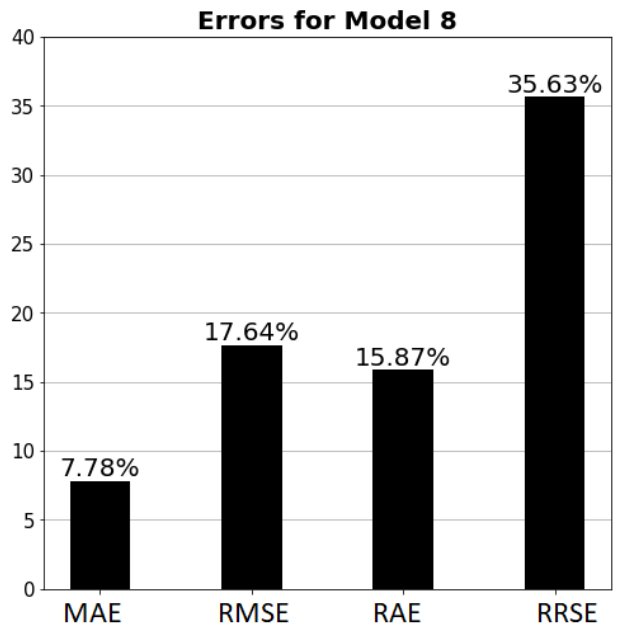

Figure 10. Associated errors on the fog node (using a model with KNN, NB, and DT as the base classifiers as well as voting as an ensemble method). Here, MAE stands for mean absolute Eeror, RMSE stands for root mean square error, RAE stands for root absolute error, and RRSE stands for root-relative squared error.

5.2.3. Execution Time and CPU Usage

Along with the previously discussed performance metric, we also calculated the execution time of the chosen model, as well as all other models (not selected at the cloud layer) using voting as an ensemble method on the fog node. This execution time is shown in Figure 11. This is to determine whether we selected the correct model in terms of execution time. The fog node execution time of model 8 with voting was the fastest of all models.

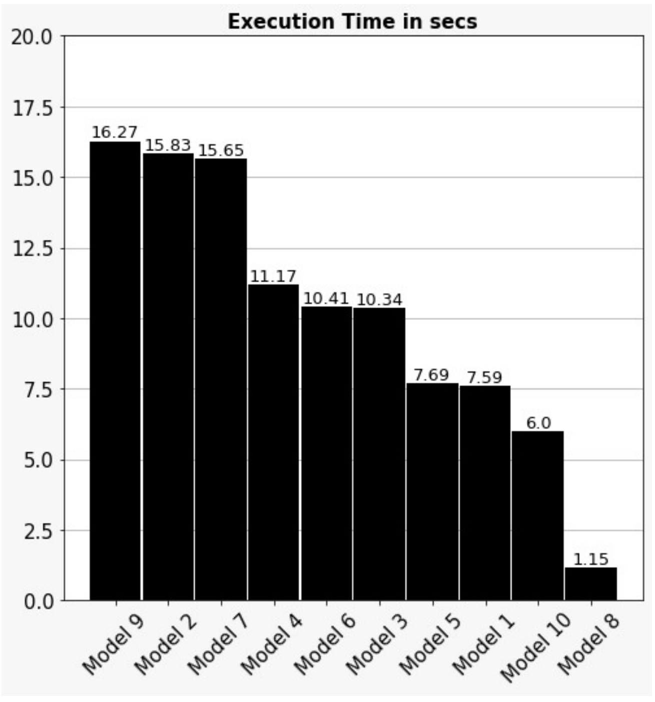

Figure 11. Execution time of all the models on the fog node.

Additionally, we calculated the CPU consumption within the fog layer. Less than 10% of the CPU is consumed by the fog layer. Therefore, our method does not require additional resources from fog nodes. Moreover, our approach has a low execution time. This shows that our approach is highly cost-effective.

6. Conclusions

This study proposes an approach to offload the ensemble machine learning model selection task to the cloud and the real-time prediction task to fog nodes. Using this technique, the cloud can handle more resource-intensive tasks and the fog nodes can handle real-time computations to simplify and reduce real-time attack detection. The proposed approach has been tested on the NSL-KDD dataset. Using a range of performance indicators, such as kappa, F-measure, ROC, and MCC, our results showed that the selected model in the cloud layer performed well in the fog layer. Moreover, the selected model in the fog node took a minimum of 1.15 s in the experiments. The research also shows that the ensemble method with voting takes less time to execute than stacking.

Our study used the NSL-KDD dataset. Our future plans are to collect data from real testbed emulation. Currently, there are several testbeds available in the EU and the US [49,50], such as Fed4Fire (https://www.fed4fire.eu/, accessed on 20 March 2022), COSMOS (https://cosmos-lab.org/, accessed on 20 March 2022) (Cloud-Enhanced Open Software-Defined Mobile Wireless Testbed for City-Scale Deployment), and POWDER (https://powderwireless.net/, accessed on 20 March 2022) (Platform for Open Wireless Data-Driven Experimental Research). We will create an edge/fog computing use case on these testbeds and run our proposed approach in an IoT scenario presented in an NGIAtlantic project [48].

Author Contributions: Formal analysis, V.T. and S.S.; Methodology, V.T. and S.S.; Supervision, S.S.; Validation, V.T.; Writing, original draft, V.T. and S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the EU H2020 NGIAtlantic project under agreement No. OC3-292.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was carried out with the support of the EU H2020 NGIAtlantic project under agreement No. OC3-292.

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

References

1. Abdulghani, H.A.; Nijdam, N.A.; Collen, A.; Konstantas, D. A Study on Security and Privacy Guidelines, Countermeasures, Threats: IoT Data at Rest Perspective. *Symmetry* **2019**, *11*, 774. [CrossRef]
2. Wang, A.; Liang, R.; Liu, X.; Zhang, Y.; Chen, K.; Li, J. An Inside Look at IoT Malware. In *Industrial IoT Technologies and Applications*; Chen, F., Luo, Y., Eds.; Industrial IoT 2017; Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer: Cham, Switzerland, 2017.
3. Razdan, S.; Sharma, S. Internet of Medical Things (IoMT): Overview, Emerging Technologies, and Case Studies. *IETE Tech. Rev.* **2021**, 1–14. [CrossRef]
4. 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]
5. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
6. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Process. Mag.* **2018**, *35*, 41–49. [CrossRef]
7. Giacinto, G.; Roli, F.; Bruzzone, L. Combination of neural and statistical algorithms for supervised classification of remote-sensing images. *Pattern Recognit. Lett.* **2000**, *21*, 385–397.

8. Bansal, A.; Mahapatra, S. A Comparative Analysis of Machine Learning Techniques for Botnet Detection. In Proceedings of the 10th International Conference on Security of Information and Networks SIN '17, New York, NY, USA, 13–15 October 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 91–98. [CrossRef]
9. Jaber, A.N.; Rehman, S.U. FCM–SVM based intrusion detection system for cloud computing environment. *Clust. Comput.* **2020**, 23, 3221–3231.
10. Zhang, Y.; Ren, Y.; Wang, J.; Fang, L. Network forensic computing based on ANN-PCA. In Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007), Harbin, China, 15–19 December 2007; pp. 942–945.
11. Hemavathi, D.; Srimathi, H. Effective feature selection technique in an integrated environment using enhanced principal component analysis. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 3679–3688.
12. 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.
13. Hosseini, S.; Zade, B.M.H. New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN. *Comput. Netw.* **2020**, *173*, 107168.
14. Amor, N.B.; Benferhat, S.; Elouedi, Z. Naive bayes vs. decision trees in intrusion detection systems. In Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 14–17 March 2004; pp. 420–424.
15. Ingre, B.; Yadav, A. Performance analysis of NSL-KDD dataset using ANN. In Proceedings of the 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96. [CrossRef]
16. Zhang, C.; Ruan, F.; Yin, L.; Chen, X.; Zhai, L.; Liu, F. A Deep Learning Approach for Network Intrusion Detection Based on NSL-KDD Dataset. In Proceedings of the 2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 25–27 October 2019; pp. 41–45. [CrossRef]
17. Wang, H.; Sayadi, H.; Sasan, A.; Rafatirad, S.; Mohsenin, T.; Homayoun, H. *Comprehensive Evaluation of Machine Learning Countermeasures for Detecting Microarchitectural Side-Channel Attacks*; GLSVLSI '20; Association for Computing Machinery: New York, NY, USA, 2020, pp. 181–186. [CrossRef]
18. Ahmad, R.; Alsmadi, I. Machine learning approaches to IoT security: A systematic literature review. *Int. Things (IoT)* **2021**, *14*, 100365. [CrossRef]
19. Ambedkar, C.; Babu, V.K. Detection of probe attacks using machine learning techniques. *Int. J. Res. Stud. Comput. Sci. Eng. (IJRSCSE)* **2015**, *2*, 25–29.
20. Sabhnani, M.; Serpen, G. Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. *Intell. Data Anal.* **2004**, *8*, 403–415.
21. Abdelkefi, A.; Jiang, Y.; Sharma, S. SENATUS: An Approach to Joint Traffic Anomaly Detection and Root Cause Analysis. In Proceedings of the 2018 2nd Cyber Security in Networking Conference (CSNet), Paris, France, 24–26 October 2018; pp. 1–8. [CrossRef]
22. Khare, N.; Devan, P.; Chowdhary, C.L.; Bhattacharya, S.; Singh, G.; Singh, S.; Yoon, B. Smo-dnn: Spider monkey optimization and deep neural network hybrid classifier model for intrusion detection. *Electronics* **2020**, *9*, 692. [CrossRef]
23. Manimurugan, S.; Majdi, A.Q.; Mohmmed, M.; Narmatha, C.; Varatharajan, R. Intrusion detection in networks using crow search optimization algorithm with adaptive neuro-fuzzy inference system. *Microprocess. Microsyst.* **2020**, *79*, 103261.
24. Kasliwal, B.; Bhatia, S.; Saini, S.; Thaseen, I.S.; Kumar, C.A. A hybrid anomaly detection model using G-LDA. In Proceedings of the 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 21–22 February 2014; pp. 288–293.
25. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing* **2020**, *387*, 51–62.
26. Chan, Y.H. Biostatistics 305. Multinomial logistic regression. *Singap. Med. J.* **2005**, *46*, 259.
27. Liu, J.; Kantarci, B.; Adams, C. Machine learning-driven intrusion detection for contiki-NG-based IoT networks exposed to NSL-KDD dataset. In Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning, Linz, Austria, 13 July 2020; pp. 25–30.
28. Su, T.; Sun, H.; Zhu, J.; Wang, S.; Li, Y. BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. *IEEE Access* **2020**, *8*, 29575–29585.
29. Abu Al-Haija, Q.; Al-Badawi, A. Attack-Aware IoT Network Traffic Routing Leveraging Ensemble Learning. *Sensors* **2022**, *22*, 241. [CrossRef]
30. Yong, B.; Wei, W.; Li, K.C.; Shen, J.; Zhou, Q.; Wozniak, M.; Połap, D.; Damaševičius, R. Ensemble machine learning approaches for webshell detection in Internet of things environments. In *Transactions on Emerging Telecommunications Technologies*; Wiley: Hoboken, NJ, USA, 2020; p. e4085. [CrossRef]
31. Rashid, M.M.; Kamruzzaman, J.; Hassan, M.M.; Imam, T.; Gordon, S. Cyberattacks Detection in IoT-Based Smart City Applications Using Machine Learning Techniques. *Int. J. Environ. Res. Public Health* **2020**, *17*, 9347. [CrossRef]
32. Tsogbaatar, E.; Bhuyan, M.H.; Taenaka, Y.; Fall, D.; Gonchigsumlaa, K.; Elmroth, E.; Kadobayashi, Y. SDN-Enabled IoT Anomaly Detection Using Ensemble Learning. In *Artificial Intelligence Applications and Innovations*; Maglogiannis, I., Iliadis, L., Pimenidis, E., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 268–280.

33. Sharma, S. Towards Artificial Intelligence Assisted Software Defined Networking for Internet of Vehicles. In *Intelligent Technologies for Internet of Vehicles*; Magaia, N., Mastorakis, G., Mavromoustakis, C., Pallis, E., Markakis, E.K., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 191–222. [CrossRef]
34. Latif, S.A.; Wen, F.B.X.; Iwendi, C.; Li, F.; Wang, L.; Mohsin, S.M.; Han, Z.; Band, S.S. AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems. *Comput. Commun.* **2022**, *181*, 274–283. [CrossRef]
35. Rambabu, K.; Venkatram, N. Ensemble classification using traffic flow metrics to predict distributed denial of service scope in the Internet of Things (IoT) networks. *Comput. Electr. Eng.* **2021**, *96*, 107444. [CrossRef]
36. Kumar, P.; Gupta, G.P.; Tripathi, R. An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks. *Comput. Commun.* **2021**, *166*, 110–124. [CrossRef]
37. Khare, S.; Totaro, M. Ensemble Learning for Detecting Attacks and Anomalies in IoT Smart Home. In Proceedings of the 2020 3rd International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 24–26 June 2020; pp. 56–63. [CrossRef]
38. Hung, Y.H. Improved Ensemble-Learning Algorithm for Predictive Maintenance in the Manufacturing Process. *Appl. Sci.* **2021**, *11*, 6832. [CrossRef]
39. Wang, J.; Pan, J.; Esposito, F.; Calyam, P.; Yang, Z.; Mohapatra, P. Edge cloud offloading algorithms: Issues, methods, and perspectives. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–23.
40. Zhang, P.; Zhou, M.; Fortino, G. Security and trust issues in Fog computing: A survey. *Future Gener. Comput. Syst.* **2018**, *88*, 16–27.
41. Hu, P.; Dhelim, S.; Ning, H.; Qiu, T. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl.* **2017**, *98*, 27–42.
42. Tariq, N.; Asim, M.; Al-Obeidat, F.; Zubair Farooqi, M.; Baker, T.; Hammoudeh, M.; Ghafir, I. The Security of Big Data in Fog-Enabled IoT Applications Including Blockchain: A Survey. *Sensors* **2019**, *19*, 1788. [CrossRef]
43. Alzoubi, Y.I.; Osmanaj, V.H.; Jaradat, A.; Al-Ahmad, A. Fog computing security and privacy for the Internet of Thing applications: State-of-the-art. *Secur. Priv.* **2021**, *4*, e145. [CrossRef]
44. Alrawais, A.; Alhothaily, A.; Hu, C.; Xing, X.; Cheng, X. An attribute-based encryption scheme to secure fog communications. *IEEE Access* **2017**, *5*, 9131–9138.
45. Hu, P.; Ning, H.; Qiu, T.; Song, H.; Wang, Y.; Yao, X. Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things. *IEEE Int. Things J.* **2017**, *4*, 1143–1155.
46. Li, Z.; Zhou, X.; Liu, Y.; Xu, H.; Miao, L. A non-cooperative differential game-based security model in fog computing. *China Commun.* **2017**, *14*, 180–189.
47. Osanaiye, O.; Chen, S.; Yan, Z.; Lu, R.; Choo, K.K.R.; Dlodlo, M. From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access* **2017**, *5*, 8284–8300.
48. ATLANTIC-eVISION: Cross-Atlantic Experimental Validation of Intelligent SDN-controlled IoT Networks 2021–2022. Available online: https://ngiatlantic.eu/funded-experiments/atlantic-evision-cross-atlantic-experimental-validation-intelligent-sdn (accessed on 20 March 2022).
49. Berman, M.; Demeester, P.; Lee, J.W.; Nagaraja, K.; Zink, M.; Colle, D.; Krishnappa, D.K.; Raychaudhuri, D.; Schulzrinne, H.; Seskar, I.; et al. Future Internets Escape the Simulator. *Commun. ACM* **2015**, *58*, 78–89. [CrossRef]
50. Suñé, M.; Bergesio, L.; Woesner, H.; Rothe, T.; Köpsel, A.; Colle, D.; Puype, B.; Simeonidou, D.; Nejabati, R.; Channegowda, M.; et al. Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed. *Comput. Netw.* **2014**, *61*, 132–150. [CrossRef]