Wireless Personal Communications (2020) 111:2287–2310
https://doi.org/10.1007/s11277-019-06986-8

Machine Learning Based Intrusion Detection Systems for IoT Applications

Abhishek Verma[1] · **Virender Ranga**[1]

Published online: 30 November 2019

Abstract
Internet of Things (IoT) and its applications are the most popular research areas at present. The characteristics of IoT on one side make it easily applicable to real-life applications, whereas on the other side expose it to cyber threats. Denial of Service (DoS) is one of the most catastrophic attacks against IoT. In this paper, we investigate the prospects of using machine learning classification algorithms for securing IoT against DoS attacks. A comprehensive study is carried on the classifiers which can advance the development of anomaly-based intrusion detection systems (IDSs). Performance assessment of classifiers is done in terms of prominent metrics and validation methods. Popular datasets CIDDS-001, UNSW-NB15, and NSL-KDD are used for benchmarking classifiers. Friedman and Nemenyi tests are employed to analyze the significant differences among classifiers statistically. In addition, Raspberry Pi is used to evaluate the response time of classifiers on IoT specific hardware. We also discuss a methodology for selecting the best classifier as per application requirements. The main goals of this study are to motivate IoT security researchers for developing IDSs using ensemble learning, and suggesting appropriate methods for statistical assessment of classifier's performance.

Keywords Internet of Things · Denial of service · Intrusion detection · Anomaly · Significance test · Performance analysis

1 Introduction

Security and privacy aspects of the Internet of Things (IoT) [5, 7, 45, 64, 65] are the key players which drive its potential to become one of the globally adopted technology in the future [34, 58]. However, self-configuring and open nature of IoT makes it vulnerable to various insider and outsider attackers [37, 49]. Attackers may compromise the users' security and privacy in order to gain access to their personal information, create monetary

✉ Abhishek Verma
abhiverma866@gmail.com

Virender Ranga
virender.ranga@nitkkr.ac.in

[1] Department of Computer Engineering, National Institute of Technology Kurukshetra, Kurukshetra, India

losses, and eavesdropping [72]. These factors prevent global adoption of IoT, consequently slowing down its growth [30]. Denial of service (DoS) is one of the most catastrophic attacks that prevent legitimate user to access the service it has paid for [17, 55]. This violates Service Level Agreement (SLA) terms which leads to huge monetary losses for firms and organizations. Moreover, DoS also affects the services of small networks, i.e., smart home, healthcare and smart agriculture etc [43]. DoS attacks on critical smart applications such as healthcare may even lead to death like situations because normal services get delayed [48]. IoT devices (e.g., smart light bulbs, smart door locks, smart television) are an easy target of attackers which exploit their vulnerabilities in order to perform DoS attacks [20, 50, 56, 59, 68]. Thus, securing these devices is one of the important concerns for researchers nowadays [6, 71]. To address this issue, intrusion detection is being heavily researched worldwide [9, 69]. Intrusion detection systems (IDSs) are categorized into three classes based on the detection method, i.e., signature, anomaly, and specification. Among three IDS types, our focus is primarily on anomaly-based IDS [32]. A signature-based IDS matches network traffic patterns with the attack patterns (signatures) already stored in its database. In case a match is found, an alarm is raised. A signature-based IDS has high accuracy and low false alarm rate, however, it is incapable of detecting new attacks. A specification-based network IDS match traffic behavior (parameters) against a predefined set of rules and values (specifications) for detecting malicious activities. These specifications are manually specified by a security expert. In contrast to signature and specification based IDS, anomaly-based IDS continuously checks network traffic for any deviation from normal network profile. In case a deviation exceeds the threshold, an alarm is raised to signify attack detection. The normal network profile is learned using machine learning (ML) algorithms. A anomaly-based IDS are preferred over signature and specification based IDS because of its ability to detect new attacks, but this comes with a cost of high false alarm rate. The effectiveness of anomaly-based IDS depends on the goodness of detection engine (model or classifier), and this goodness comes with the quality of network traffic patterns (dataset instances) being used for engine's training. Once the detection engine has been trained, it can detect new attacks effectively. Intrusion detection in IoT networks is characterized as a binary classification problem in which a trained classifier aims to classify network traffic into normal or attack class with maximum accuracy and minimum false alarms (FAR). The high performance of the classifier in terms of accuracy and FAR solely depends on the choice of classification algorithm and training data. Security experts prefer good performing classifier for the task of intrusion detection. Many solutions for intrusion detection have been proposed in the literature [8, 14, 18, 32, 44, 47] and most of them are dedicated for traditional networking paradigms only, and as far as the literature is concerned, less work has been done towards the development of ML-based intrusion detection for IoT applications. Moreover, we didn't find any work in the literature which statistically analyzed the significance of classifier's performance in particular to IoT based intrusion detection. Also, no work in the literature has realized the execution of classifier on an IoT hardware. Thus, our focus is basically on utilizing ML classification algorithms for building IDS in order to secure IoT against DoS attacks.

In this work, we carry out a performance assessment of ML classifiers for IDS in particular to IoT. The performance of single classifiers including CART and MLP, and classifier ensembles namely Random forest (RF), AdaBoost (AB), Extreme gradient boosting (XGB), Gradient boosted machine (GBM), and Extremely randomized trees (ETC) is measured in terms of prominent metrics, i.e., accuracy, specificity, sensitivity, false positive rate, area under the receiver operating characteristic curve (AUC). Hyper-tuning (finding the set of optimal parameters) of all the classifiers is done using random search[10]. The

significant differences of classifiers are statistically assessed using a well known statistical test. Finally, we have tested the performance of classifiers in terms of average response time on Raspberry Pi, i.e., IoT device [70].

Our primary contributions can be summarized as follows.

- Performance assessment of different ML classifiers on CIDDS-001, UNSW-NB15, NSL-KDD datasets with repeated hold-out and repeated cross fold validation methods is done.
- Statistical assessment of performance results using widely used Friedman test (non-parametric statistical test) and Nemenyi post-hoc test, i.e., Friedman test for classifier significance test and Nemenyi test for pairwise comparison among classifiers is done.
- Implementation and execution of classifiers on Raspberry Pi hardware for realizing actual response time on real IoT hardware is carried out.

The paper organization is as follows. Section 2 discusses recent works in the concerned domain. Section 3 provides a brief discussion on classification algorithms, i.e., single classifiers, and ensembles. Experimental design is discussed in Sect. 4. Discussion related to the classifier's performance and statistical tests is done in Sect. 5. Section 6 concludes the paper.

2 Related Work

There are few works present in the literature which suggest methods for defending IoT against DoS attacks. Like, Misra et al. [46] proposed a specification-based IDS based on Learning Automata for preventing distributed DoS attacks against IoT. The authors considered preventing IoT middle-ware layer rather than a particular device. The proposed security system sets a threshold for the number of requests a middleware layer can service. As soon as the number of incoming requests exceeds the set threshold, an attack is detected. Kasinathan et al. [39] proposed a signature-based IDS framework for detecting DoS attack in IoT. The proposed framework consists of a monitoring and detection modules. These modules are integrated with the network framework of European Union (EU) FP7 project 'ebbits' for securing the network against DoS attacks. A DoS protection manager and IDS are integrated with the 'ebbits' network. A network-based IDS is used for capturing and analyzing the packets sniffed from IDS probe node's that are spread across the network. The evaluation results show that the proposed framework performs well in terms of true positive and false positive rate. Kasinathan et al. [38] proposed an IDS to detect DoS attacks. Suricata [1] (open source IDS) is used for pattern matching and attack detection. A probe node is used to sniff all the packet transmissions in the network and transfer information to IDS for further analysis. Penetration testing tool 'Scapy' is used to test the performance of the proposed IDS. No simulation study is done in support of IDS performance and its usability. Moreover, the authors did not mention any details regarding signature database management (update). Lee et al. [42] proposed a novel IDS for detecting DoS attacks. The key idea behind the proposed IDS is to analyze the node's energy consumption in order to track malicious nodes. The authors proposed various models for normal energy consumption in mesh routing based networks. The proposed security system requires nodes to monitor their own energy consumption at a sampling rate of 0.5 s. The proposed IDS continuously checks the energy consumption of nodes against the defined threshold,

and whenever a deviation is found for any node, such a node is marked as malicious and removed from the routing table. The proposed approach shows promising results in terms of accuracy only. The major concern with this proposed approach is that there is no inbuilt mechanism to verify the integrity of energy consumption values being reported by a node. Sonar et al. [60] proposed an IDS to detect distributed DoS attacks in IoT networks. The authors implemented IDS as software-based manager deployed between network and gateway. The proposed IDS maintains greylist and blacklist or IP addresses in order to control the access to the network. In the proposed IDS, the greylist is updated every 40 s while blacklist is updated every 300 s. Simulation of the proposed IDS is performed on the contiki [23] operating system. The proposed IDS performs do not achieve satisfactory performance in terms of packet delivery ratio, the number of serviced packets, true positives, and false positives. Moreover, the recovery time is larger than agent learning time which adds to the major limitations of the proposed IDS. Diro et al. [21] proposed a deep learning based IDS for defending DoS against IoT networks. The proposed model is evaluated using NSL-KDD dataset. The authors performed the comparison of proposed IDS with the traditional shallow model approach. In addition, the proposed IDS is implemented with centralized and distributed detection scheme. The comparison results show that the distributed attack detection scheme performs better compared to centralized detection scheme in terms of accuracy. Similarly, the deep model shows better results compared to the shallow model in terms of accuracy, precision, recall, and F1 measure. Tama et al. [61] proposed an anomaly based IDS that uses gradient boosted machine (GBM) as a detection engine. The optimal parameters of GBM are obtained using grid search and the performance of the proposed IDS is validated using hold-out and cross fold methods on three different datasets namely UNSW-NB15, NSL-KDD, and GPRS. The authors show that proposed IDS outperforms the fuzzy classifier, GAR forest, tree based ensembles in terms of accuracy, specificity, sensitivity, and area under curve (AUC). Primartha et al. [52] studied the performance of RF based IDS in terms of accuracy and false alarm rate. The authors employed NSL-KDD, UNSW-NB15 and GPRS dataset for model training and testing. The proposed IDS is studied with different tree size ensembles, and statistical analysis based on Friedman ranking showed that the ensemble of 800 trees achieves best results whereas an ensemble of 20 trees showed the worst performance. Moreover, the proposed RF based IDS outperforms ensemble of Random tree + Naive Bayes, and single classifiers like NBTree and Multilayer perceptron.

3 Classification Algorithms

Wolpert et al. [67] stated a theorem popularly known as "no free lunch" theorem that shows the importance of experimenting with different machine classifiers for solving classification tasks. The theorem states that "there is no single learning algorithm that universally performs best across all domains" [22]. Thus, different classifiers should be tested for solving domain specific problems, and in our case, the problem is intrusion detection or classification problem. We consider two types of classification algorithms, i.e., ensembles and single classifiers. Among ensembles, widely studied algorithms [29, 41, 57] like Random forest (RF), AdaBoost (AB), Gradient boosted machine (GBM), Extreme gradient boosting (ETC), and Extremely randomized trees (ETC) are chosen. There are main reasons for selection of mentioned classification algorithms. First, because ensemble-based classification methods are prone to over-fitting in case the number of input features is large we also choose to study some single

classifiers like Classification and regression trees (CART), and Multi-layer perceptron (MLP). Second, the performance of ensembles has not been studied in depth for CIDDS-001 and UNSW-NB15 datasets. Third, the performance of ensembles and single classifiers over real IoT hardware has not been studied yet, which motivated us for carrying this analytical study.

3.1 Classifier Ensembles

This section discusses various classifier ensembles in brief. Ensembles have been proven to be good classification and regression algorithms in the literature. Thus, we have used five different ensembles in this analytical study.

3.1.1 Random Forest (RF)

RF[12] is a collection of trees, i.e., predictors $\{t(x_{in}, \theta_n), n = 1, \ldots\}$ which individually make predictions on a given input x_{in}. Each predictor depends on the random set of variables $\{\theta_n\}$ that are sampled independently with the same distribution. The main idea behind RF is that the number of predictors together might achieve better prediction accuracy while avoiding the over-fitting problem. Each predictor in RF grows to a maximum size without getting pruned. Once a large number of trees are created, they make predictions over the input data by voting for the most popular class at input x_{in}. For the performance assessment the number of estimators (trees) is set to 500 and maximum depth for tree construction is set to 26 as recommended by [12, 61]. The other parameters are obtained using randomized search.

3.1.2 AdaBoost (AB)

AB[25] is an adaptive meta-estimator that learns the initial training weights on the original dataset. These weights act as input to additional copies of the classifier based on incorrectly classified instances. The subsequent classifiers adjust the weights of classified instances, i.e., difficult cases. In this way, AB improves the performance of learning algorithms by boosting weak learners such that final model converges to a strong learner. Equation (1) represents a boost classifier where c_p resembles a weak learner and x resembles an input object.

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

$c_p(x)$ returns the value indicating predicted class. Each c_p generates an output hypothesis $(h(x_i))$ for each instance in the training set. At each iteration p, a c_p is chosen and assigned a coefficient β_p such that the sum training error E_t (represented as Eq. 2) of the resulting p-stage $C_P(x)$ is minimized. $C_{p-1}(x_i)$ represents the boosted classifier built from previous training phase, $E(C)$ is error function which is to be minimized, and $c_p(x) = \beta_p h(x_i)$ is the weak learner that is to be added to the final model, i.e., classifier. The optimal parameters of AB include 50 estimators and 0.1 learning rate.

$$E_t = \sum_i E[C_{p-1}(x_i) + \beta_p h(x_i)] \tag{2}$$

3.1.3 Gradient Boosted Machine (GBM)

GBM [26, 27] is a member of the ensemble family which aims to improve the performance of decision trees (DT). Like other boosting methods it sequentially combines weak classifiers, i.e., DT, and allows them to optimize an arbitrary differential loss function in order to form a strong prediction model. Each present learner (tree) relies on the predictions of previous learners in order to improve the prediction errors. Formally, let us consider a set of random input variables and random output represented by x (Eq. 3) and z respectively.

$$x = \{x_1, x_2, \ldots, x_N\} \tag{3}$$

Our aim is to find an estimate A (approximation) that maps x to z by using training data $\{z, x_i\}_1^N$. A is represented as Eq. 6. Given a dataset (S) with p samples and q features as represented by Eq. (4). Then, a ensemble utilizes M additive function to predict final output Eq. (5).

$$S = \{(x_i, z_i)\}(|D| = p, x_i \in \mathbb{R}^q, z_i \in \mathbb{R}) \tag{4}$$

$$\hat{z}_i = \phi(x_i) = \sum_{m=1}^{M} f_m(x_i), f_k \in A \tag{5}$$

$$A = \{f(x) = w_{r(x)}(r : \mathbb{R}^m \rightarrow K, w \in \mathbb{R}^K)\} \tag{6}$$

A is the instance set of DT, and r represents the tree structure that relates an instance to the correlating leaf index, K indicates the total count of trees, and f_m is a single tree with structure r and leaf weight w. The tree ensemble makes the final prediction by summation of the scores (w) of leaves which are found by classifying given test sample. Suppose, a first classifier (i.e., tree) makes prediction $h_1(x)$ over a sample $\{(x_i, y_i)\}_1^N$. Then, $h_1(x)$ is fed as input to next classifier in order to adjust the weights of previously misclassified instances. Consequently, next classifier makes prediction $h_2(x)$ over $\{(x_i, y_i - h_1(x_i))\}_1^N$. The final prediction $h(x)$ for a given sample data S is the summation of predictions made by the trees while minimizing the prediction error. The hyper-tuned parameters for GBM are: 500 estimators, maximum tree construction depth is 3, minimum samples required for split are 100 and 0.1 learning rate.

3.1.4 Extreme Gradient Boosting (XGB)

XGB [15] is also known as regularized gradient boosting is an improved version of GBM. Like GBM, XGB follows the same principle of gradient boosting. The only key difference between them is in terms of modeling details. XGB uses more regularized model formalization in order to control over-fitting and increase generalization ability while GBM focuses only on the variance. Regularization parameter (ζ) is mathematically expressed as Eq. (7). Where, T_l is the number of leaves in the tree, w_j^2 is the score on the jth leaf, λ represents regularization term the controls model complexity. XGB uses gradient boosting for optimizing the loss function during model training. Typically, for binary classification the LogLoss function (L) [11] is used and represented as Eq. (8). Where, N is the total number of observations, y_i is the binary indicator of whether predicted class c is the correct

classification for particular observation o and p_i is the predicted probability that particular observation o is of class c . Most importantly, L controls the predictive power, and ζ controls the simplicity of the model. The major implementation enhancement of XGB includes usage of sparse matrices (DMatrix) with sparsity aware algorithms, improved data structures, parallelization support. Thus, XGB leverages the hardware to achieve high speed computing with low memory utilization (primary memory and cache). The optimal values of parameters obtained for XGB are: 100 estimators, maximum tree depth is 8, value of minimum child weight is 1, gbtree booster is considered, minimum loss reduction and subsample ratio are 2 and 0.6 respectively.

$$\zeta = \gamma T_l + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{7}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(p_i) + (1 - y_i) log(1 - p_i)) \tag{8}$$

3.1.5 Extremely Randomized Trees (ETC)

ETC [33] also known as Extra trees is a tree induction algorithm for performing supervised classification and regression. To be more specific, ETC builds an ensemble of unpruned DTs. The key procedure in ETC involves randomly selecting both features and cut-point irrespective of the target variable. At each tree node, this procedure is followed with totally or partially selecting a certain number of features among which the optimal one is determined. In the worst case, the algorithm selects a single feature and cut-point at each node. In this manner, totally randomized trees are built which are independent of the training sample's target attribute values. The classical top-down methodology is followed while building the ensemble. Unlike other tree-based ensemble algorithms, ETC uses complete training sample rather than bootstrap replicas in order to grow the trees while minimizing bias and variance. The ETC splitting procedure for numeric features has three important parameters. The first parameter is K indicates the number of features selected at each node. The second parameter is n_{min} which represents minimum training set size for splitting a node. The third parameter is T_{count} that is the number of trees in the ensemble. All three parameters play a significant role in the ETC building. Where K is responsible for the strength of feature selection procedure, n_{min} governs the averaging output noise, and T_{count} specifies the reduction in variance. ETC performs almost the same as RF, however with optimal feature selection ETC is computationally faster compared to RF. The hypertuned parameters obtained from randomized search for ETC are: 1788 estimators, maximum tree depth value is 10, minimum sample size for split is 5, number of features considered for best split are $\log_{10} 2$, gini criterion is considered with no bootstrapping.

3.2 Single Classifiers

This section discusses single classifiers in brief. In this comparative study we have used classification and regression trees, and multi-layer perceptron.

3.2.1 Classification and Regression Trees (CART)

CART [13] is one of the widely employed ML methods for predictive modeling problems. It is a non-parametric algorithm with a built-in mechanism to handle missing feature values. CART involves recursive partitioning of training samples and fitting of a simple prediction model within each partition. This partitioning can be represented as DT graphically. CART employs exhaustive search technique, in order to identify the splitting variables such that the total impurity of node's child nodes (two children) is minimized. CART uses the Gini index as its impurity function which makes it computationally efficient over entropy based classification tree algorithms. The number of folds in the internal cross-validation and the minimal number of observations at the terminal nodes considered are 5 and 2, respectively as used in [61]. The optimal value of maximum depth of tree construction obtained is 10.

3.2.2 Multi-layer Perceptron (MLP)

MLP [35] is a logical unit of connected nodes (artificial neurons) that attempts to mimic the biological brain behavior commonly referred as a feed-forward artificial neural network. It learns its expertise towards a particular task using supervised learning approaches. MLP comprises several layers, i.e., input, middle and output. Training of MLP involves learning a mathematical function $f(.)$ shown in Eq. (9), where d, c are the number of inputs and outputs respectively. In order to perform any predictive task, MLP learns a non-linear function approximator over a set of input features $\{I = i_1, i_2, \ldots, i_d\}$ and output variable O, i.e., class. The leftmost layer also termed as input layer consists of many artificial neurons $\{i_p | i_1, i_2, \ldots, i_d\}$ each representing a particular input feature. The second layer, i.e., the middle layer performs the task of transformation. First, the outputs from the former layer are summed using weighted linear summation y represented as Eq. (10). Second, a non-linear activation function ($g(\cdot)$) is applied to y which results into a value that is forwarded to further layers, typically output layer in case single hidden layer is present. The rightmost layer is the output layer which receives values from the last hidden layer and responsible for firing outputs, i.e., final predictions. The optimal parameters values of MLP obtained are: hidden layer size of 100, logistic activation function, *sgd* solver, learning rate of 0.001, and 200 maximum iterations.

$$f(\cdot) = R^d \rightarrow R^c \tag{9}$$

$$y = \sum_{p=1}^{d} (w_p i_p) \tag{10}$$

4 Experimental Design

4.1 Experimental Setup

The performance assessment has been carried out on a machine operated on 64-bit Windows 10 Pro and equipped with Intel^ i7-7700 four core CPU having 3.60 GHz clock

speed and 12GB main memory. The classifiers are implemented in the Python programming language (version 3.6.1). Parameter hyper-tuning is performed on PARAM Shavak system operated on 64-bit Ubuntu 14.04 and equipped with Intel^ Xeon^ Gold 6132 twenty eight-core CPU having 2.6 GHz clock speed and 96 GB main memory. Raspberry Pi 3 Model B+ operated on Raspbian operating system and equipped with 64-bit quad-core ARM CPU running having 1.4 GHz clock speed and 1 GB main memory is used for assessing the response time of classifiers. Popular ML library scikit-learn [51] is utilized for implementing classifiers. In order to perform statistical tests on the performance results, we used the STAC [54] web platform application.

4.2 Datasets

In this study three different datasets, i.e., CIDDS-001 [2], UNSW-NB15 [4], and NSL-KDD [3] are used. We choose CIDDS-001 and UNSW-NB15 dataset as they are most recently generated datasets and contain traffic of real data, and hence can be beneficial for building accurate IDSs for monitoring and detection of new type of DoS attacks in IoT networks. The CIDDS-001 dataset has recently been released for facilitating the development of anomaly-based IDS. The complete dataset contains approximately 32 million records comprising of normal and attacks traffic. CIDDS-001 possesses 12 features and 2 labeling attributes. Random sampling is employed to extract 100,000 instances from the internal server traffic data, compromising of 80,000 normal and 20,000 attacks (DoS) records. The extracted sample is used for carrying out hold-out and cross fold validation tests of classifiers. Our previous works [62, 63] focused on evaluating the performance of various ML classification algorithms on CIDDS-001 dataset.

Further, we have conducted our experiments on newly publicly available dataset known as UNSW-NB15. The dataset possesses 49 features and 1 class attribute. A part of the dataset is used as train and test sets, i.e., UNSW_NB15_Train and UNSW_NB15_Test. The train set comprises of 175,341 instances, and the test set comprises of 82,332 instances. The train set includes 56,000 instances of normal traffic and 119,341 instances of attack traffic. Similarly, the test set includes 37,000 instances of normal traffic and 45,332 instances of attack traffic. Hold-out validation is conducted using the complete train and test sets, whereas for cross-fold validation test only train set is employed.

Subsequently, NSL-KDD dataset is also used for performing validation of classifiers. The dataset contains 41 features and 1 class attribute. In this study, KDDTrain+ (training) and KDDTest+ (testing) sets of NSL_KDD dataset are used. The KDDTrain+ set contains total 25,192 instances comprising of 13,499 instances of attack traffic and 11,743 instances of normal traffic. Whereas, the KDDTest+ set contains total 22,544 instances comprising of 9,711 instances of attack traffic and 12,833 instances of normal traffic. Hold-out and cross fold validation of classifiers is done on each dataset individually. The choice of these sets is done in order to avoid random sampling of instances from complete NSL-KDD dataset.

4.3 Evaluation Metrics and Validation Methods

Selection of input parameter settings influences the overall performance of the classifiers, thus we follow random search [10] procedure to find the optimal input parameters of RF, AB, XGB, GBM, and ETC for different datasets. RandomizedSearchCV implementation in *scikit-learn* package of Python programming language is used for

hyper-tuning of parameters. RandomizedSearchCV finds optimal parameter settings by performing a cross-validated search over candidate parameter values provided by the user. Prominent metrics for evaluating classifier's performance have been used in this study. These metrics include accuracy, specificity or true negative rate, sensitivity or true positive rate, FPR, and AUC, mathematically represented as Eqs. (11)–(15) respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$Specificity = \frac{TN}{TN + FP} \tag{12}$$

$$Senstivity = \frac{TP}{TP + FN} \tag{13}$$

$$FPR = \frac{FP}{TN + FP} \tag{14}$$

$$AUC = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{FP + TN} \tag{15}$$

where true positive (TP) represents the number of correctly classified attack instances, true negative (TN) represents the number of correctly classified normal instances, false positive (FP) is the number of wrongly classified attack instances, and false negative (FN) is the number of wrongly classified normal instances. Accuracy is defined as the total number of correctly classified instances over the total number of instances in the dataset. Specificity is defined as the number of correctly classified normal instances over the total number of normal instances. Sensitivity is defined as the number of correctly classified attack instances over the total number of attack instances. FPR is defined as the number of incorrectly classified attack instances over the total number of normal instances. AUC refers to the area under the receiver operating characteristic (ROC) curve, where ROC curve defined as plotting TPR against FPR.

In order to perform a comprehensive performance assessment of different classifiers, we conducted experiments by using repeated hold-out as well as repeated k-fold cross validation (10f) method [40]. As suggested in [53], the repeated version stabilizes the error estimation and minimizes the variance of the validation approach. For hold-out validation, we divided sample dataset into 60:40 ratio (60% training instances and 40% testing instances) in order to create train and test set. Similarly, for k-fold cross validation the value of k is considered as 10. We considered 100 rounds of repeated 10f and hold-out validation as the classification models are observed to be stable, i.e., same prediction for the same test data. 10f is performed in order to asses the classifier's performance while avoiding the effect of instance sampling (i.e in case of hold-out validation). In order to avoid bias, all the performance results reported in this paper are the mean value of outputs from 10 iterations of each repeated validation approach. Each experiment is repeated by using different seed (an input to a random number generator) for avoiding biased results.

4.4 Statistical Significance Tests

In ML studies, comparison of multiple algorithms over multiple datasets is an essential issue [19]. An algorithm may show better performance over one dataset whereas may fail to achieve similar result over another dataset. The reason for this may be the presence of outliers, feature distribution or algorithm characteristics. Thus, it becomes quite difficult to compare different algorithms among themselves. This consequently makes it challenging to decide which algorithm is better than others. To address this issue, the statistical assessment is needed to statistically validate the performance results. In this study, two statistical significance tests [16] are utilized in order to perform the comparison of classifiers in a correct way. Friedman [28] and Nemenyi [24] tests are selected for this purpose. The significance tests help in finding whether the classifiers are significantly different from each other or not [19, 31]. The null hypothesis (H_0) considered in this case is that there is no performance difference among classifiers. While alternative hypothesis (H_1) is that there is at least one classifier that performs significantly different that at least one other classifier. The main reason behind choosing the Friedman test is that it is the most powerful statistical test in case the number of entities being compared are greater than five [16, 61]. Friedman test helps in determining whether at least one classifier performs significantly better than the others in case of all the datasets. In any one such classifier is found, then Nemenyi post-hoc test is performed for pairwise multiple comparisons. As suggested in [19], it is crucial to conduct post-hoc test so as to identify the performance differences among the classifiers.

To be more specific, Friedman test checks for the significant difference among the classifiers being tested, whereas the Nemenyi test pinpoints where that difference lies. The further discussion assumes d as the number of datasets, and k as the number of classifiers. In Friedman test, initially the performance results (X_{ij}) of classifiers are ranked ($R(X_{ij})$) for all the datasets. Then, sum of the $R(X_{ij})$ is computed for each classifier in order to obtain R_j (Eq. 16), where $j = 1, 2, \ldots, k$. The Friedman statistic (F-Statistic) is computed as Eq. (17), where Q is calculated as Eq. (18).

$$R_j = \sum_{i=1}^{d} R(X_{ij}) \tag{16}$$

$$F\text{-}Statistic = \frac{(d-1)Q}{d(k-1)-Q} \tag{17}$$

$$Q = \frac{12}{dk(k+1)} \sum_{j=1}^{k} \left(R_j - \frac{d(k+1)}{2} \right)^2 \tag{18}$$

The F-Statistic is tested against the F-quantiles for a given α with degree of freedom, $f_1 = k - 1$ and $f_2 = (d-1)(k-1)$, where α is the significance level being considered. In this study the values of d, k are 4 and 7 respectively. Nemenyi post-hoc test is performed by calculating test statistic γ_{xy} (represented as Eq. 19) for all classifier pairs, where $\overline{R_x}$ and $\overline{R_y}$ are mean ranks of classifiers x and y respectively on all datasets, and computed as Eq. (20). After all the γ_{xy} are calculated and those which exceed a critical value are said to indicate a significant difference between classifiers x and y at α significance level. In this study, two values of α are considered, i.e., 0.05 and 0.1. The statistical analysis of both hold-out and 10f validation results is carried out in this experimental study.

$$\gamma_{xy} = \frac{\overline{R_x} - \overline{R_y}}{\sqrt{\frac{k(k+1)}{6d}}} \tag{19}$$

$$\overline{R_j} = \frac{1}{d} \sum_{i=1}^{d} R(X_{ij}) \tag{20}$$

5 Results and Analysis

In this section, a detailed discussion on performance analysis of ensembles (RF, AB, GBM, XGB and ETC) and single classifiers (CART and MLP) specific to CIDDS-001, UNSW-NB15, and NSL-KDD datasets is done. The results are compared and statistically analyzed. We have shown that the classifiers used in this study are suitable for intrusion detection in IoT applications. First, we analyze the performance results of hold-out validation. Figure 1 indicates the average value of all prominent metrics other than FPR, achieved with hold-out validation across CIDDS-001, UNSW-NB15, KDDTrain+, and KDDTest+ datasets. It is observed that RF outperforms other classifiers in terms of accuracy (94.94%) and specificity (91.6%). GBM performs best in terms of sensitivity (99.53%). In terms of AUC metric, XGB performs best by achieving 98.76%. MLP is the worst performer in terms of accuracy (82.76%), whereas AB performs worst in terms of specificity (86.72%) and sensitivity (97.94%). CART achieves lowest AUC value (94.01%). Figure 2 shows the average FPR values of classifiers across four datasets with hold-out validation. It is observed that RF performs best whereas AB performs worst among all the classifiers in terms of FPR by achieving 8.89% and 13.26% respectively. Table 1 lists out model building time (MBT) of different classifiers across four datasets with hold-out validation. The main reason behind computing MBT is that it is very important to consider the training time a model takes, as it would directly impact the resources usage, which is an important criterion for resource-constrained devices [66]. Thus, MBT helps in making a good trade-off between resource usage and classification performance of a classifier, i.e., IDS. RF and CART take approximately 2 s for training on all four datasets. The highest time for model training is taken by GBM and ETC in case of KDDTrain+ dataset. MBT of all the classifiers is calculated for hold-out validation only.

Figure 3 shows the average value of all prominent metrics other than FPR, achieved with 10f validation across CIDDS-001, UNSW-NB15, KDDTrain+, and KDDTest+ datasets. It is observed that the performances of all the used classifiers improve with 10f validation in comparison to classifier's performances with hold-out validation. This is due to the effect of sampling which results in the selection of random instances that leads to poor classification. This phenomenon advocates the use of 10f validation over hold-out validation. The 10f validation results show promising performance for all the classifiers. However, from the point of comparison, CART performs best in terms of accuracy (96.74%). AB achieves the highest average value of specificity (97.5%) metric. RF and XGB perform best in terms of sensitivity by achieving 97.31% performance measure. For AUC, the best performing classifier is XGB which achieves 98.77%. Figure 4 shows the average FPR values of classifiers across four datasets with 10f validation. It is observed that CART performs best

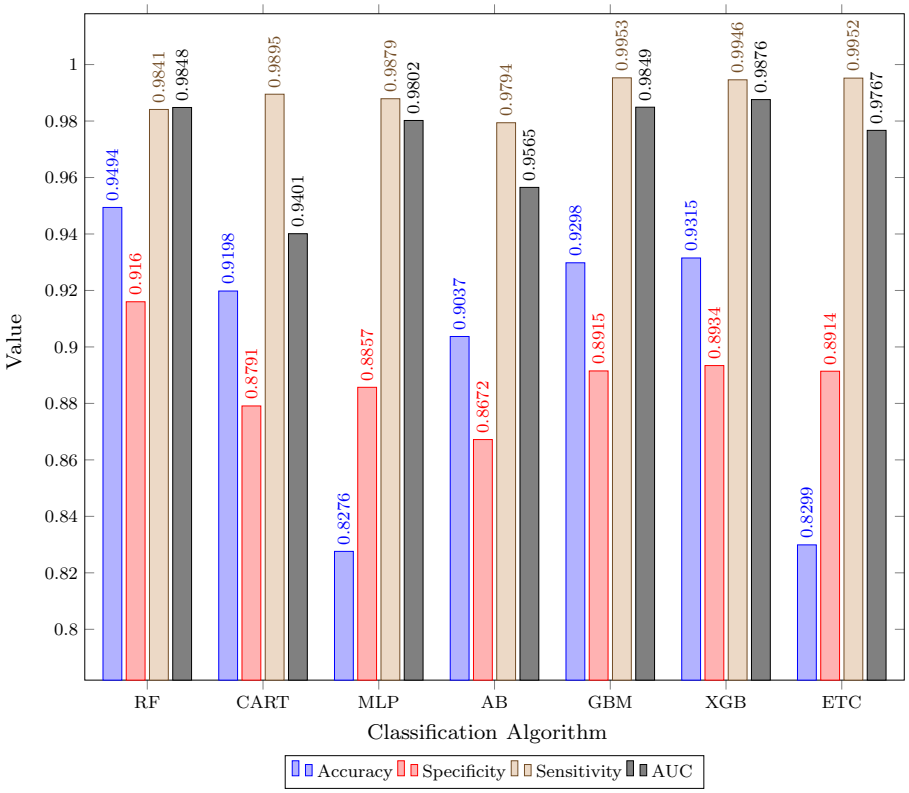

Fig. 1 The average value of prominent metrics per classifier across four datasets with hold-out validation

Fig. 2 The average value of FPR per classifier across four datasets with hold-out validation

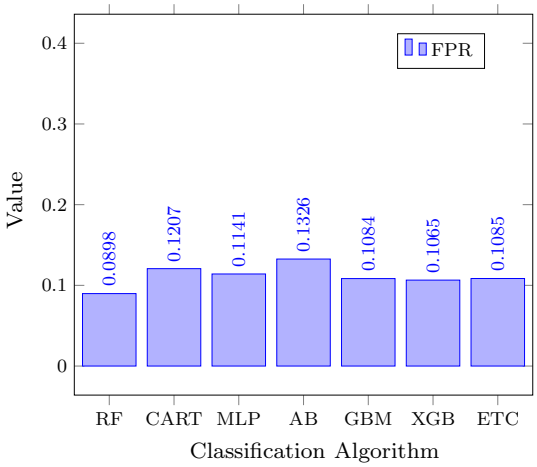

Table 1 MBT (seconds) of classifiers across four datasets

Dataset	RF	CART	MLP	AB	GBM	XGB	ETC
CIDDS-001	0.4124	0.2353	1.0160	0.9557	20.1139	7.3965	17.5031
UNSW-NB15	1.4657	0.6260	4.3782	7.9092	12.7477	23.7196	44.4775
KDDTrain+	0.4087	0.2653	16.7050	2.6928	318.8914	14.0115	143.0728
KDDTest+	0.0601	0.0337	5.3062	0.4866	22.7327	2.9957	1.5041

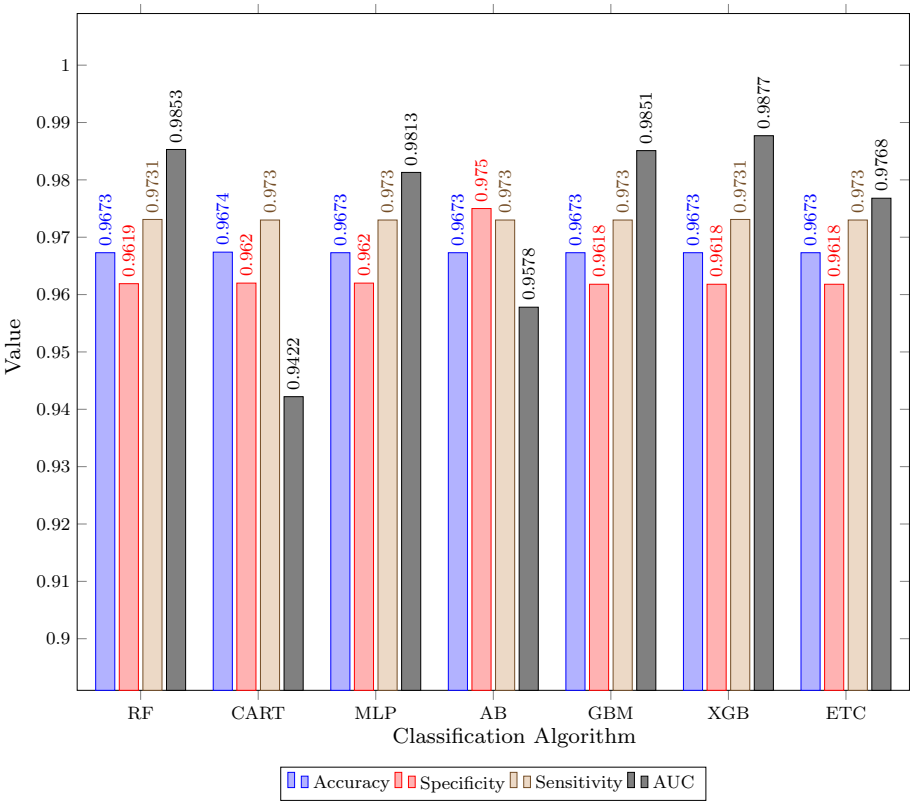

Fig. 3 The average value of prominent metrics per classifier across four datasets with 10f validation

whereas RF performs worst among all classifiers in terms of FPR by achieving 3.78% and 21.85% respectively.

The performance results are statistically assessed using Friedman and Nemenyi post-hoc test. Both the statistical tests are performed for two values of significance level α (i.e., 0.05 and 0.1). For $\alpha = \{0.05, 0.1\}$ the value of f_1, f_2 are 6 and 18 respectively, the F-Statistic and p value for each performance metric is computed. Table 2 shows Friedman test statistics for hold-out validation results. From the results it is observed that the performance of the classifiers is significantly different ($p < 0.05$ and $p < 0.1$) in terms of all the considered performance metrics. Thus, it is concluded that there is

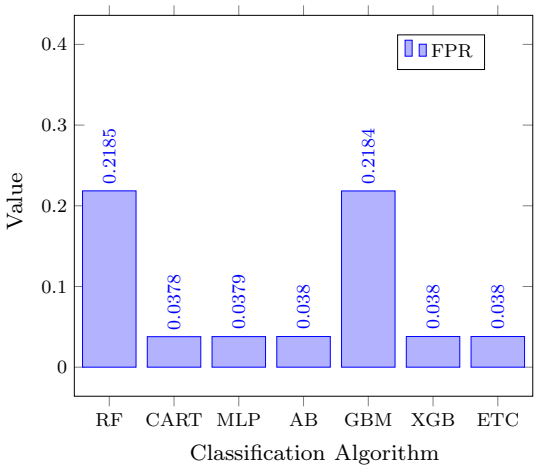

Fig. 4 The average value of FPR per classifier across four datasets with 10f validation

Table 2 Friedman test statistics for hold-out validation

	Accuracy	Specificity	Sensitivity	FPR	AUC
F-Statistic	6.7745	7.7091	7.1434	7.7091	4.7020
p value	0.0007	0.0003	0.0005	0.0003	0.0048
$\alpha = 0.05$	R	R	R	R	R
$\alpha = 0.1$	R	R	R	R	R

Table 3 Friedman test (mean ranks for hold-out validation)

	Accuracy	Specificity	Sensitivity	FPR	AUC
RF	4.875	5.750	2.875	2.250	3.750
CART	2.250	2.000	3.250	6.000	1.500
MLP	3.250	3.250	2.250	4.750	3.000
AB	1.250	1.500	2.000	6.500	2.875
XGB	6.000	6.375	5.500	1.625	6.000
GBM	5.750	4.625	6.250	3.375	5.875
ETC	4.625	4.500	5.875	3.500	5.000

at least one classifier that performs significantly different that one another classifier. Because the results of Friedman test are highly significant ($p < 0.05$ and $p < 0.1$), the null hypothesis H_0 is rejected (represented by R in Table 2) and alternative hypothesis H_1 is accepted. In Table 3, the mean ranks of all the classifiers for hold-out validation are shown. In order to find which classifier pairs perform significantly different, Nemenyi post-hoc test is performed. For this purpose, the p value of all the pairwise comparisons is tested against the considered significance level α.

Table 4 presents the results of the Nemenyi test (pairwise comparison) over accuracy, specificity and sensitivity. As shown in Table 4, the classifier's accuracy is highly significant ($p < 0.05$) in case of AB-XGB pair, whereas less significant ($p < 0.1$) in case

Table 4 Nemenyi pairwise comparison (hold-out validation) Part I

A_1 versus A_2	Accuracy				Specificity				Senstivity			
	F-statistic	p value	α		F-statistic	p value	α		F-statistic	p value	α	
			0.05	0.1			0.05	0.1			0.05	0.1
AB versus XGB	3.1096	0.0393	R	R	3.1914	0.0297	R	R	2.2912	0.4608	A	A
AB versus GBM	2.9459	0.0676	A	R	2.0457	0.8563	A	A	2.7822	0.1133	A	A
AB versus RF	2.3731	0.3704	A	A	2.7822	0.1134	A	A	0.5728	1.0	A	A
AB versus ETC	2.2094	0.57	A	A	1.9639	1.0	A	A	2.5367	0.2349	A	A
AB versus MLP	1.3093	1.0	A	A	1.1456	1.0	A	A	0.4091	1.0	A	A
AB versus CART	0.6546	1.0	A	A	0.3273	1.0	A	A	0.8183	1.0	A	A
CART versus ETC	1.5548	1.0	A	A	1.6366	1.0	A	A	1.7184	1.0	A	A
CART versus MLP	0.6546	1.0	A	A	0.8183	1.0	A	A	0.6546	1.0	A	A
ETC versus MLP	0.9001	1.0	A	A	0.8183	1.0	A	A	2.3731	0.3704	A	A
GBM versus CART	2.2912	0.4608	A	A	1.7184	1.0	A	A	1.9639	1.0	A	A
GBM versus MLP	1.6366	1.0	A	A	0.9001	1.0	A	A	2.6186	0.1854	A	A
GBM versus ETC	0.7364	1.0	A	A	0.0818	1.0	A	A	0.2455	1.0	A	A
GBM versus RF	0.5728	1.0	A	A	0.7364	1.0	A	A	2.2094	0.57	A	A
GBM versus XGB	0.1636	1.0	A	A	1.1456	1.0	A	A	0.4909	1.0	A	A
RF versus CART	1.7184	1.0	A	A	2.4549	0.2959	A	A	0.2455	1.0	A	A
RF versus MLP	1.0638	1.0	A	A	1.6366	1.0	A	A	0.1636	1.0	A	A
RF versus ETC	0.1636	1.0	A	A	0.8183	1.0	A	A	1.9639	1.0	A	A
XGB versus CART	2.4549	0.2959	A	A	2.8641	0.0878	A	R	1.4729	1.0	A	A
XGB versus MLP	1.8003	1.0	A	A	2.0457	0.8563	A	A	2.1276	0.7007	A	A
XGB versus ETC	0.9001	1.0	A	A	1.2274	1.0	A	A	0.2455	1.0	A	A
XGB versus RF	0.7364	1.0	A	A	0.4091	1.0	A	A	1.7184	1.0	A	A

Table 5 Nemenyi pairwise comparison (hold-out validation) Part II

A_1 versus A_2	FPR				AUC			
	F-statistic	p value	α		F-statistic	p value	α	
			0.05	0.1			0.05	0.1
AB versus XGB	3.1914	0.0297	R	R	2.0457	0.8563	A	A
AB versus GBM	2.0457	0.8563	A	A	1.9639	1.0	A	A
AB versus RF	2.7822	0.1133	A	A	0.5728	1.0	A	A
AB versus ETC	1.9639	1.0	A	A	1.3911	1.0	A	A
AB versus MLP	1.1456	1.0	A	A	0.0818	1.0	A	A
AB versus CART	0.3273	1.0	A	A	0.9001	1.0	A	A
CART versus ETC	1.6366	1.0	A	A	2.2912	0.4608	A	A
CART versus MLP	0.8183	1.0	A	A	0.9819	1.0	A	A
ETC versus MLP	0.8183	1.0	A	A	1.3093	1.0	A	A
GBM versus CART	1.7184	1.0	A	A	2.8641	0.0878	A	R
GBM versus MLP	0.9001	1.0	A	A	1.8821	1.0	A	A
GBM versus ETC	0.0818	1.0	A	A	0.5728	1.0	A	A
GBM versus RF	0.7364	1.0	A	A	1.3911	1.0	A	A
GBM versus XGB	1.1456	1.0	A	A	0.0818	1.0	A	A
RF versus CART	2.4549	0.2959	A	A	1.4729	1.0	A	A
RF versus MLP	1.6366	1.0	A	A	0.4909	1.0	A	A
RF versus ETC	0.8183	1.0	A	A	0.8183	1.0	A	A
XGB versus CART	2.8641	0.0878	A	R	2.9459	0.0676	A	R
XGB versus MLP	2.0457	0.8563	A	A	1.9639	1.0	A	A
XGB versus ETC	1.2274	1.0	A	A	0.6546	1.0	A	A
XGB versus RF	0.4091	1.0	A	A	1.4729	1.0	A	A

Table 6 Friedman test statistics for 10f validation

	Accuracy	Specificity	Sensitivity	FPR	AUC
F-Statistic	0.1698	0.2346	0.2740	0.4242	4.5294
p value	0.9816	0.9594	0.9418	0.8532	0.0057
$\alpha = 0.05$	A	A	A	A	R
$\alpha = 0.1$	A	A	A	A	R

of AB-GBM pair. While the remaining pairs are not significant ($p < 0.1$). Moreover, in terms of specificity, the highly significant pair is AB-XGB, whereas the less significant pair is XGB-CART, whilst no other pair is found to be significant. Furthermore, no pair is significant in terms of sensitivity. Table 5 shows the results of the Nemenyi test (pairwise comparison) over FPR, AUC and MBT. It is observed that for FPR metric the classifier's performance is highly significant in the case of AB-XGB and less significant in the case of XGB-CART, whereas remaining pairs are not significant. While, in the case of AUC metric there are no pairs which are highly significant, whilst GBM-CART and XGB-CART are the only less significant pairs among all the classifier pairs, every other pair is not significant.

Table 7 Friedman test (mean ranks for 10f validation)

	Accuracy	Specificity	Sensitivity	FPR	AUC
RF	4.000	4.000	4.500	4.625	3.625
CART	4.375	4.250	3.375	3.500	1.500
MLP	4.250	5.000	3.375	2.750	3.125
AB	4.625	4.000	4.500	4.750	2.875
XGB	3.125	3.375	4.500	4.250	6.125
GBM	4.000	3.250	4.500	4.625	5.625
ETC	3.625	4.125	3.250	3.500	5.125

Table 8 Nemenyi test (10f validation)

A_1 versus A_2	AUC			
	F-statistic	p value	α	
			0.05	0.1
AB versus XGB	2.1276	0.7007	A	A
AB versus GBM	1.8003	1.0	A	A
AB versus ETC	1.4729	1.0	A	A
AB versus CART	0.9001	1.0	A	A
AB versus RF	0.4909	1.0	A	A
AB versus MLP	0.1636	1.0	A	A
CART versus GBM	2.7004	0.1454	A	A
CART versus ETC	2.3731	0.3704	A	A
CART versus MLP	1.0638	1.0	A	A
ETC versus MLP	1.3093	1.0	A	A
ETC versus GBM	0.3273	1.0	A	A
MLP versus GBM	1.6366	1.0	A	A
RF versus CART	1.3911	1.0	A	A
RF versus GBM	1.3093	1.0	A	A
RF versus ETC	0.9819	1.0	A	A
RF versus MLP	0.3273	1.0	A	A
XGB versus CART	3.0277	0.0517	A	R
XGB versus MLP	1.9639	1.0	A	A
XGB versus RF	1.6366	1.0	A	A
XGB versus ETC	0.6546	1.0	A	A
XGB versus GBM	0.3273	1.0	A	A

Table 6 shows Friedman test statistics for 10f validation results. From the results it is observed that the performance of the classifiers is significantly different ($p < 0.05$ and $p < 0.1$) in terms of AUC only. Thus, it is concluded that there is at least one classifier that performs significantly different than one another classifier. Because the result of Friedman test is highly significant ($p < 0.05$ and $p < 0.1$) for AUC metrics, the null hypothesis H_0 is rejected and alternative hypothesis H_1 is accepted (represented by A in Table 6). In Table 7 the mean ranks of all the classifiers for 10f validation results are shown. Table 8 presents the results of the Nemenyi test (pairwise comparison) over AUC values. As shown in Table 8, the

classifier's AUC measure is found less significant in case of XGB-CART pair, whereas all other pairs are not significant.

In addition, we have analyzed the average response time (seconds) that a classifier takes to classify an instance. The main reason to perform this experiment is that the knowledge of classifier's response time plays an important role in its selection as a intrusion detection system [36]. The classifiers with quick (small) response time are favored over classifier with slow (large) response time. To accomplish this task, all the classifiers with test data as input were executed on Raspberry Pi 3 Model B+. The average time is computed by dividing the total time taken by the classifier for classifying all the test instances by the total number of test instances.

$$Average\ response\ time = \frac{\sum_{i=1}^{n_{test}} t_i}{n_{test}} \tag{21}$$

Equation (21) represents the mathematical expression of average response time, where i represents an instance number, t_i represents time taken by a classifier to classify ith test instance into attack or normal category, and n_{test} is the total number of test instances.

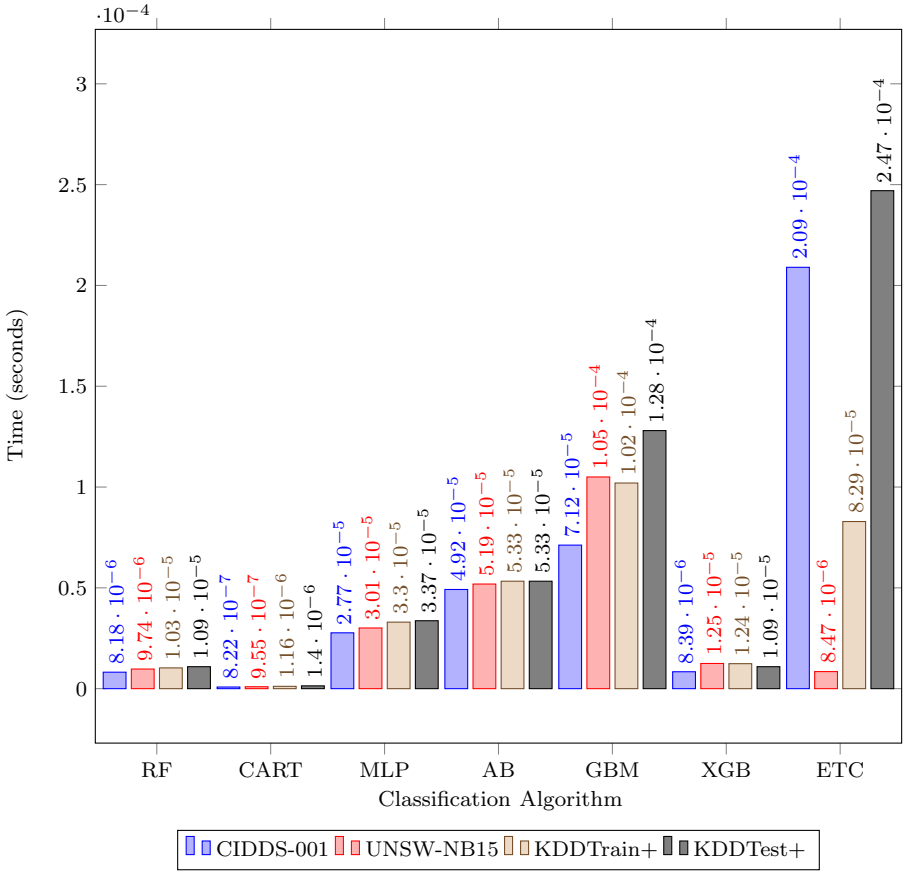

Fig. 5 Average response time of classifiers

Figure 5 shows the average response time taken by different classifiers for classifying a single instance. From the experiment results, it is observed that CART takes minimum time to classify an instance of CIDDS-001, UNSW-NB15, KDDTrain+, and KDDTest+ in comparison to other classifiers. RF and XGB show almost similar results in terms of average response time for all four datasets. ETC takes maximum time for classifying an instance of CIDDS-001 and KDDTest+ dataset in comparison to other classifiers. Moreover, GBM is the worst performer in the case of KDDTrain+ dataset. The experimental results show promising solutions for the choice of different classifiers suitable for performing the task of intrusion detection (DoS specific) in IoT applications. The classifiers have been validated using hold-out and 10f validation methods. Both the methods show promising performance results in terms of accuracy, specificity, sensitivity, FPR, AUC. These results can be used to select the suitable classifier as per the requirement of the application. Like, if an application demands high accuracy and low FPR, then CART, MLP, AB, XGB, or ETC can be used. Whereas, if an application demands quick response time, then CART, RF, or XGB can be selected. Similar trade-offs can be considered in the selection of the best suitable classifier for an IoT application. The real-time performance of IDS depends on the dataset selected for model training. Thus, a dataset containing traffic patterns of recent types of DoS attacks must be used for achieving the best real-time results. CIDDS-001 and UNSW-NB15 are suitable choices for this purpose. It can be observed from the experimental results that classifiers show promising performance results with CIDDS-001 and UNSW-NB15 dataset thus, we suggest these datasets for training IDSs to achieve the best classification results.

In this study only supervised learning based ML classifiers are used. It is one of the popular ML approaches in which the classifier uses known target values for training. The result shown by different ML classifiers shows the effectiveness of using supervised learning for the intrusion detection task. The main reason for choosing supervised learning is that the network characteristics (traffic patterns) can be effectively used to train ML models for further predictions. These patterns can be differentiated between normal and attack based on different network based features. The other ML approach like unsupervised learning can also be used to perform a similar task. Clustering (i.e., unsupervised learning algorithm) can be used to train the different classifiers. In this paper, the emphasis is made particularly on the performance assessment of supervised ML algorithms. The performance assessment of unsupervised ML algorithms for intrusion detection in IoT will be considered in our future work.

6 Conclusion

In this paper, a study on anomaly-based IDS suitable for securing IoT against DoS attacks is carried out. The performance assessment of seven machine learning classification algorithms including random forests, adaboost, gradient boosted machine, extremely randomized trees, classification and regression trees, and multi-layer perceptron is done. The optimal parameters of classifiers are obtained using a random search algorithm. Performance of all the classifiers is measured in terms of accuracy, specificity, sensitivity, false positive rate, and area under the receiver operating characteristic curve. Benchmarking of all the classifiers is performed on CIDDS-001, UNSW-NB15, and NSL-KDD datasets. Moreover, in order to find significant differences among classifiers the statistical analysis of performance measures is done using Friedman and Nemenyi post host tests. In addition

to this, the average response time of all classifiers is evaluated on Raspberry Pi hardware device. From the performance results and statistical tests, it is concluded that classification and regression trees, and extreme gradient boosting classifier show the best trade-off between prominent metrics and response time, thus both are the suitable choice for building IoT specific anomaly-based IDS. Our future goal is to design an IDS for defending routing attacks in IoT networks.

Acknowledgements This research was supported by the Ministry of Human Resource Development, Government of India.

Compliance with Ethical Standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. (2014). Suricata: Open-source ids/ips/nsm engine. Retrieved November 3, 2019, from https://suricata-ids.org/.
2. (2017). CIDDS-001 dataset. Retrieved November 3, 2019, from https://www.hs-coburg.de/forschung-kooperation/forschungsprojekte-oeffentlich/ingenieurwissenschaften/cidds-coburg-intrusion-detection-data-sets.html.
3. (2017). NSL-KDD dataset. Retrieved November 3, 2019, from http://nsl.cs.unb.ca/nsl-kdd/.
4. (2017). UNSW-NB15 dataset. Retrieved November 3, 2019, from https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/.
5. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, *17*(4), 2347–2376.
6. Arış, A., Oktuğ, S. F., & Yalçın, S. B. Ö. (2015). Internet-of-things security: Denial of service attacks. In *IEEE 23th signal processing and communications applications conference (SIU)* (pp. 903–906).
7. Ashton, K. (2009). That 'Internet of Things' thing. *RFID Journal*, *22*(7), 97–114.
8. Axelsson, S. (2000). *Intrusion detection systems: A survey and taxonomy*. Technical report.
9. Baykara, M., & Das, R. (2017). A novel hybrid approach for detection of webbased attacks in intrusion detection systems. *International Journal of Computer Networks and Applications*, *4*(2), 62–76.
10. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(Feb), 281–305.
11. Bishop, C. M. (2006). *Pattern recognition and machine learning (Information science and statistics)*. Berlin: Springer.
12. Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.
13. Breiman, L. (2017). *Classification and regression trees*. London: Routledge.
14. Butun, I., Morgera, S. D., & Sankar, R. (2014). A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, *16*(1), 266–282.
15. Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *ACM, proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794).
16. Conover, W. J., & Conover, W. J. (1980). *Practical nonparametric statistics*. New York: Wiley.
17. Das, R., Tuna, A., Demirel, S., & Yurdakul, M. K. (2017). A survey on the Internet of Things solutions for the elderly and disabled: Applications, prospects, and challenges. *International Journal of Computer Networks and Applications*, *4*(3), 84–92.
18. Debar, H., Dacier, M., & Wespi, A. (2000). A revised taxonomy for intrusion-detection systems. *Annales Des Télécommunications*, *55*(7), 361–378.
19. Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*(Jan), 1–30.
20. Dhanjani, N. (2013). Hacking lightbulbs: Security evaluation of the philips hue personal wireless lighting system. Retrieved November 3, 2019, from https://www.dhanjani.com/docs/Hacking.

21. Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, *82*, 761–768.
22. Douglas, P. K., Harris, S., Yuille, A., & Cohen, M. S. (2011). Performance comparison of machine learning algorithms and number of independent components used in FMRI decoding of belief vs. disbelief. *Neuroimage*, *56*(2), 544–553.
23. Dunkels, A., Gronvall, B., & Voigt, T. (2004). Contiki—A lightweight and flexible operating system for tiny networked sensors. In *IEEE 29th annual IEEE international conference on local computer networks* (pp. 455–462).
24. Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, *56*(293), 52–64.
25. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.
26. Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232.
27. Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.
28. Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701.
29. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(4), 463–484.
30. Gao, L., & Bai, X. (2014). A unified perspective on the factors influencing consumer acceptance of Internet of Things technology. *Asia Pacific Journal of Marketing and Logistics*, *26*(2), 211–231.
31. Garcia, S., & Herrera, F. (2008). An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, *9*(Dec), 2677–2694.
32. Garcia-Teodoro, p, Diaz-Verdejo, j, & Maciá-Fernández, G. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1–2), 18–28.
33. Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.
34. Granjal, J., Monteiro, E., & Silva, J. S. (2015). Security for the Internet of Things: A survey of existing protocols and open research issues. *IEEE Communications Surveys Tutorials*, *17*(3), 1294–1312.
35. Haykin, S. (1994). *Neural networks: A comprehensive foundation*. Englewood Cliffs: Prentice Hall PTR.
36. Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P. L., Iorkyase, E., Tachtatzis, C., et al. (2016). Threat analysis of IoT networks using artificial neural network intrusion detection system. In *International symposium on networks, computers and communications (ISNCC)* (pp. 1–6). IEEE.
37. Hwang, Y. H. (2015). Iot security & privacy: Threats and challenges. In: *Proceedings of the 1st ACM workshop on IoT privacy, trust, and security* (pp. 1–1). New York, NY: ACM
38. Kasinathan, P., Costamagna, G., Khaleel, H., Pastrone, C., & Spirito, M. A. (2013). Demo: An ids framework for Internet of Things empowered by 6lowpan. In *Proceedings of the 2013 ACM SIGSAC conference on computer & communications security (CCS '13)* (pp. 1337–1340). New York, NY: ACM.
39. Kasinathan, P., Pastrone, C., Spirito, M. A., & Vinkovits, M. (2013). Denial-of-service detection in 6lowpan based Internet of Things. In *IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)* (pp. 600–607).
40. Kim, J. H. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, *53*(11), 3735–3745.
41. Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156.
42. Lee, T. H., Wen, C. H., Chang, L. H., Chiang, H. S., & Hsieh, M. C. (2014). *A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan* (pp. 1205–1213)., Advanced technologies, embedded and multimedia for human-centric computing Dordrecht: Springer.
43. Li, X., Lu, R., Liang, X., Shen, X., Chen, J., & Lin, X. (2011). Smart community: An Internet of Things application. *IEEE Communications Magazine*, *49*(11), 68–75.
44. Lunt, T. F. (1993). A survey of intrusion detection. *Computers & Security*, *12*, 405–418.
45. Medhat, M., Elshafey, K., & Rashed, A. (2019). Evaluation of optimum NPRACH performance in NB-IoT systems. *International Journal of Computer Networks and Applications*, *6*(4), 55–64.
46. Misra, S., Krishna, P. V., Agarwal, H., Saxena, A., & Obaidat, M. S. (2011). A learning automata based solution for preventing distributed denial of service in Internet of Things. In *IEEE, 4th*

international conference on cyber, physical and social computing, Internet of Things (ithings/cpscom) (pp. 114–122).
47. Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, *36*(1), 42–57.
48. Moosavi, S. R., Gia, T. N., Rahmani, A. M., Nigussie, E., Virtanen, S., Isoaho, J., et al. (2015). Sea: A secure and efficient authentication and authorization architecture for IoT-based healthcare using smart gateways. *Procedia Computer Science*, *52*, 452–459.
49. Mosenia, A., & Jha, N. K. (2017). A comprehensive study of security of internet-of-things. *IEEE Transactions on Emerging Topics in Computing*, *5*(4), 586–602.
50. Notra, S., Siddiqi, M., Gharakheili, H. H., Sivaraman, V., & Boreli, R. (2014). An experimental study of security and privacy risks with emerging household appliances. In *2014 IEEE conference on communications and network security* (pp. 79–84). https://doi.org/10.1109/CNS.2014.6997469.
51. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.
52. Primartha, R., & Tama, B. A. (2017). Anomaly detection using random forest: A performance revisited. In *2017 International conference on data and software engineering (ICoDSE)* (pp. 1–6). IEEE.
53. Rodriguez, J. D., Perez, A., & Lozano, J. A. (2010). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(3), 569–575.
54. Rodríguez-Fdez, I., Canosa, A., Mucientes, M., & Bugarín, A. (2015). Stac: A web platform for the comparison of algorithms using statistical tests. In *IEEE international conference on fuzzy systems (FUZZ-IEEE)* (pp. 1–8).
55. Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed Internet of Things. *Computer Networks*, *57*(10), 2266–2279.
56. Ronen, E., & Shamir, A. (2016). Extended functionality attacks on IoT devices: The case of smart lights. In *2016 IEEE European symposium on security and privacy (EuroS P)* (pp. 3–12). https://doi.org/10.1109/EuroSP.2016.13.
57. Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1249.
58. Sfar, A. R., Natalizio, E., Challal, Y., & Chtourou, Z. (2018). A roadmap for security challenges in the Internet of Things. *Digital Communications and Networks*, *4*(2), 118–137.
59. Sivaraman, V., Gharakheili, H. H., Vishwanath, A., Boreli, R., & Mehani, O. (2015). Network-level security and privacy control for smart-home IoT devices. In *IEEE 11th international conference on wireless and mobile computing, networking and communications (WiMob)* (pp. 163–167). https://doi.org/10.1109/WiMOB.2015.7347956.
60. Sonar, K., & Upadhyay, H. (2016). An approach to secure Internet of Things against DDOS. In *Springer proceedings of international conference on ICT for sustainable development* (pp. 367–376).
61. Tama, B. A., & Rhee, K. H. (2019). An in-depth experimental study of anomaly detection using gradient boosted machine. *Neural Computing and Applications*, *31*(4), 955–965.
62. Verma, A., & Ranga, V. (2018a). On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques. *Pertanika Journal of Science & Technology*, *26*(3), 1307–1332.
63. Verma, A., & Ranga, V. (2018). Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. *Procedia Computer Science*, *125*, 709–716.
64. Verma, A., & Ranga, V. (2019a). ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things. In *2019 4th International conference on Internet of Things: Smart innovation and usages (IoT-SIU)* (pp. 1–6). IEEE.
65. Verma, A., & Ranga, V. (2019). Evaluation of network intrusion detection systems for RPL based 6LoWPAN networks in IoT. *Wireless Personal Communications*, *108*(3), 1571–1594.
66. Williams, N., Zander, S., & Armitage, G. (2006). A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Computer Communication Review*, *36*(5), 5–16.
67. Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.
68. Zahoor, S., & Mir, R. N. (2018). Virtualization and IoT resource management: A survey. *International Journal of Computer Networks and Applications*, *5*(4), 43–51.
69. Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications*, *84*, 25–37.
70. Zhao, C. W., Jayanand, J., & Son, C. L. (2015). Exploring IoT application using Raspberry Pi. *International Journal of Computer Networks and Applications*, *2*(1), 27–34.

71. Zhao, K., & Ge, L. (2013). A survey on the Internet of Things security. In *IEEE 9th international conference on computational intelligence and security (CIS)* (pp. 663–667).
72. Ziegeldorf, J. H., Morchon, O. G., & Wehrle, K. (2014). Privacy in the Internet of Things: Threats and challenges. *Security and Communication Networks*, *7*(12), 2728–2742.

Abhishek Verma is Ph.D. research scholar in the Department of Computer Engineering at National Institute of Technology Kurukshetra, Haryana, India. He received his B.Tech. degree (2014) in Computer Science & Engineering from Uttar Pradesh Technical University, Lucknow, India and M.Tech. degree (2016) in Computer Engineering from National Institute of Technology Kurukshetra, India. He is an IEEE student member. He has more than 3 years' experience in research and published several international journal and conference papers. He is an active reviewer of many reputed journals of IEEE, Springer, Wiley, and Bentham Science. His current areas of interests include network security, in particular, Internet of Things, Wireless Sensor Networks using Statistical methods and Machine Learning mechanisms.

Virender Ranga is working as Assistant Professor in the Department of Computer Engineering at National Institute of Technology Kurukshetra, India. He received his Ph.D. degree in 2016 from National Institute of Technology Kurukshetra, Haryana, India. He has published more than 60 research papers in various International SCI Journals as well as reputed International Conferences in the area of Computer Communications. He has been conferred by Young Faculty Award in 2016 for his excellent contributions in the field of Computer Communications. He has been acted as a member of TPC in various International conferences of repute. He is a member of editorial board various reputed journals like Journal of Applied Computer Science & Artificial Intelligence, International Journal of Advances in Computer Science and Information Technology(IJACSIT), Circulation in Computer Science (CCS), International Journal of Bio Based and Modern Engineering (IJBBME) and International Journal of Wireless Networks and Broadband Technologies. Currently, he has been selected Guest Editor for a special issue to be published in the International Journal of Sensors, Wireless Communications and Control (Bentham Science Publication). He is an active reviewer of many reputed journals of IEEE, Springer, Elsevier, Talyor & Francis, Wiley, and InderScience. His current area of interest include Wireless Sensor & Ad hoc Networks, Internet of Things, Flying Ad hoc networks, and Software Defined Networking.