EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Majority Voting and Feature Selection Based Network Intrusion Detection System

Dharmaraj R. Patil[1,*] and Tareek M. Pattewar[2]

[1]Department of Computer Engineering, R.C. Patel Institute of Technology, Shirpur, Maharashtra, India
[2]Department of Computer Engineering, Vishwakarma University, Pune, Maharashtra, India
dharmaraj.patil@rcpit.ac.in, tareek.pattewar@vupune.ac.in

Abstract

Attackers continually foster new endeavours and attack strategies meant to keep away from safeguards. Many attacks have an effect on other malware or social engineering to collect consumer credentials that grant them get access to network and data. A network intrusion detection system (NIDS) is essential for network safety because it empowers to understand and react to malicious traffic. In this paper, we propose a feature selection and majority voting based solutions for detecting intrusions. A multi-model intrusion detection system is designed using Majority Voting approach. Our proposed approach was tested on a NSL-KDD benchmark dataset. The experimental results show that models based on Majority Voting and Chi-square features selection method achieved the best accuracy of 99.50% with error-rate of 0.501%, FPR of 0.005 and FNR of 0.005 using only 14 features.

Keywords: Network Intrusion detection system, Feature selection, Majority voting, Machine learning, NSL_KDD, Network security.

Received on 06 February 2022, accepted on 31 March 2022, published on 04 April 2022

doi: 10.4108/eai.4-4-2022.173780

1. Introduction

The development of malware is a significant problem for network intrusion detection system designers. Malicious attempts have gotten more complicated, and the most difficult task is identifying unknown and obfuscated malware, because malware authors use a variety of information obfuscation escape strategies to avoid detection by an NIDS. Furthermore, security threats have proliferated, such as zero-day attacks on Internet users; as a result, computer security has become more crucial as information consumption has become a part of our daily lives [1-4].

Between network NIDS and HIDS, there is a considerable difference. An intrusion detection system (IDS) is a piece of software or hardware that detects malicious traffic, takes corrective action automatically, and responds automatically to stop intrusions. Despite their use, intrusion detection approaches are constrained by a number of factors, including real-time analysis and detection, generated sensors, and high-quality data, all of which can reduce detection rate and accuracy. As a result, intrusion detection is still a viable and effective research topic [5-12].

In this work, we have presented a multi-model strategy based on feature selection and majority voting to train and create a binary classifier model and make reliable decisions. Feature selection approaches were employed to increase data quality. The results of our experiments on the NSL-KDD dataset reveal that our suggested method works well in terms of accuracy and false alarm rate. The following are the manuscript's major contributions:

- We have suggested a multi-model intrusion detection system based on Majority voting classification and obtained satisfactory intrusion detection utilizing XGBoost, Decision Tree, Random Forest, AdaBoost, and RepTree.
- We have employed feature selection methodologies such as Correlation-based Feature Subset Selection (CFS), Chi-Squared Attribute Evaluation (CHI),

*Corresponding author. Email: dharmaraj.patil@rcpit.ac.in

Gain-Ratio Attribute Evaluation, and Info-Gain Attribute Evaluation to increase data quality. We discovered that by employing feature selection strategies, we were able to gain considerable acceptable attack detection accuracy while incurring minimal system overhead. Using only 14 features and the Majority Voting and CHI features selection methods, the accuracy was 99.50 % with an error rate of 0.501 %, FPR of 0.005, and FNR of 0.005.

- We have conducted a performance comparison of our proposed work with other comparable techniques and discovered that our approach outperforms other approaches in attack detection accuracy.

The remainder of this paper is organized as follows. Section 2 discusses related work on intrusion detection. Section 3 describes the proposed methodology in detail. Section 4 discusses the experimental results, the performance of the proposed model, and its comparison to existing models. In Section 5, the discussion is given. Finally, in Section 6, the conclusion is presented.

2. Related Work

Previously, we reported some research on machine learning approaches in network intrusion detection systems. The issues that each study is concerned with differ, such as feature selection, data reduction, and classification model optimization.

Leevy, J. L., et al. discovered that when available, the top performance ratings for each study were exceptionally high overall, probably due to overfitting. They also noticed that the majority of the studies did not address class inequality, which might bias the findings of a large data research. They discovered that the CSE-CIC-IDS2018 data cleaning information was repeatedly insufficient, raising concerns about experiment repeatability. According to them, their survey found considerable research gaps [1].

Khraisat A. et al. provided a taxonomy of current IDS, as well as a full review of noteworthy recent work and an overview of datasets commonly used for evaluation purposes. They also highlighted future research challenges to fight such approaches and improve the security of computer systems [2].

Laghrissi F. et al. used deep learning approaches to identify dangers in Long Short-Term Memory (LSTM). They employed PCA and mutual information (MI) as techniques for dimension reduction and feature selection. They tested their technique on a standard dataset, KDD99, and the findings show that PCA-based versions attain the highest accuracy for teaching and testing in both binary and multiclass classification [3].

Megantara, A. A. et al. proposed a crossbreed machine learning strategy that blends the specific feature selection methodology indicating supervised understanding with the information reduction method representing unsupervised learning to build an appropriate model. It works by picking acceptable and significant functions using a feature importance decision tree centered recursive feature reduction approach and finding anomaly/outlier data using the Local Outlier Factor technique, according to them. Their experimental results demonstrate that the suggested approach achieves the highest accuracy in identifying R2L (i.e.99. 89 %) and outperforms most previous research in the NSL-KDD dataset for other attack types [4].

Jadhav, A. D., et al. created the Two-Phase Invasion Recognition System (TP-IDS) in two steps to improve accuracy. They employed SVM and kNN in stage of the particular TP-IDS. In order to improve accuracy, Decision Tree and Nave Bayes are used during Phase II of the TP-IDS system validation stage. According them, each phase makes use of the Hadoop distributed system as the primary data storage space and processing structures, which generally permits parallel processing in order to improve system overall performance and so achieve efficiency within TP-IDS [5].

Divyasree T. H. et al. proposed an efficient intrusion detection system based on Ensemble Core Vector Machine (CVM). They employed CVM algorithms based on the notion of the smallest enclosing ball. It detects U2R and R2L attacks, as well as Probe and DoS attacks. They used the KDD Cup99 dataset to train and test the classifiers. They also used the chi-square test to determine the most significant attributes for each attack, and then applied a weighted function to these features to minimize dimensionality. As a consequence, the test results reveal that the model outperformed earlier strategies in all four attacks while needing less processing time [6].

To identify and categorize network threats, Ashiku, L. et al. proposed leveraging heavy learning architectures to construct an adaptive plus robust network incursion detection system. Their emphasis will be on how deep learning, or DNNs, may enable adaptable IDSs with learning capabilities to identify known and unique or even zero-day network behaviour patterns, shut down the system intruder, and limit the chance of penetration. They used the UNSW-NB15 dataset to demonstrate the utility of the model representing real-time network communication behaviour with synthetic attack operations [7].

To safeguard the cloud from potential attacks, Elmasry, W. et al. recommended the creation of a one-of-a-kind integrated cloud-based intrusion detection system (CIDS). The suggested CIDS, according to them, includes of five primary modules that execute the following tasks: monitoring the network, capturing traffic flows, extracting features, analyzing the flows, identifying intruders, responding to and documenting all actions. They employed an upgraded bagging ensemble system with three deep learning models to accurately anticipate intrusions. They demonstrated that the suggested technique resolves all of the issues raised in the cloud threat literature [13].

Sistla, V. P. K., et al. created deep learning algorithms in NIDS prediction models to identify abnormalities and threats automatically. They evaluated the proposed model's performance on the NSL-KDD dataset using metrics such as accuracy, recall, precision, and F1 score. They claim that the experimental findings suggest that the proposed deep learning model outperforms earlier shallow models [14].

RNNIDS was created by Sohi, S. M. et al., who used Persistent Neural Networks (RNNs) to detect detailed

patterns in issues and produce identical patterns. They verified that RNNs work incredibly well to develop new, previously undiscovered variants of attacks, as well as synthetic signatures from the most complicated viruses, to boost intrusion detection performance even more. They have enhanced the appearance of a new NIDS, RNNs function incredibly well to produce malicious datasets including previously concealed virus variants, for example. To evaluate the practicality of their methodologies, they conducted extensive tests using publicly accessible data sets, which revealed a significant increase in the detection rate of commercially available NIDS (up to 16.67 %) [15].

Zhou, Y., et al. created an intrusion detection system that is mostly based on feature extraction and even ensemble techniques. During the first time, they demonstrated the CFS-BA heuristic dimensionality reduction strategy, which picks the ideal subset constructed after feature relationship. They demonstrated an ensemble technique that works with the C4, 5, Random Forest (RF), and Forest by Simply Penalizing Capabilities (Forest PA) methods in this scenario. Finally, for intrusion detection, they used a voting approach to include the majority of the basic learner probability distributions. According on the experimentation findings employing the NSL-KDD, AWID, and CIC-IDS2017 datasets, the suggested CFS-BA collecting approach outperforms other relevant and decreasing edge techniques [16].

Mane, S. et al. employed deep neural networks to detect network intrusions and suggested an explainable AI framework to promote transparency at all stages of the machine learning process. They accomplished this by employing Explainable AI algorithms, which will make ML designs less of a new black box by explaining why a new prediction is made. This information might be generated using column creation from SHAP, LIME, Contrastive Explanations Technique, ProtoDash, and Boolean Decision Guidelines (BRCG). They provide the final results of applying these algorithms to the NSL-KDD information set for Intrusion Detection System [17].

Guezzaz, A., et al. suggested a selection tree-based technique for finding incursions with higher data quality. They used network pre-processing and entropy selection feature collection to increase data quality and suitable training, and they created a selection tree classifier to have dependable intruder individuality. As a result, a learning from mistakes analysis on a handful of data sets reveals that the recommended model provides reliable insights. With the majority of the NSL-KDD and CICIDS2017 data sets, their technique obtained 99.42 % and 98.80 % accuracy, respectively [18].

Li, L., et al. have suggested a unique hybrid approach for efficiently detecting community intruders. In the suggested model, the Gini index is used to choose the best subset of features, the gradient boosted decision tree (GBDT) approach is used to detect network intrusions, and the particle swarm optimization (PSO) methodology is used to fine-tune the GBDT parameters. They used the NSL-KDD dataset to put the suggested models through their tests in terms of accuracy, detection rate, precision, F1 score, and false alarm rate. According to the results, the suggested model outperforms the compared techniques [19].

Using the UNSW-NB15 dataset as a benchmark, Moualla, S. et al. proposed a unique neighbourhood IDS that plays an important role in network security measures and prevents current cyber-attacks on sites. According to them, their suggested system is a learning-based, multi-class system. The method is based on the Synthetic Group Oversampling Technique (SMOTE) approach to deal with imbalanced patterns in the dataset, and then uses a Randomized Trees Classifier (Extra Trees Classifier) to extract the specific key features in the dataset using the Gini select contamination qualifying criterion. Following that, they employed a pretrained extreme learning machine (ELM) model for each attack separately, using "One-Versus-All" as the specific binary classifier associated with them. The specific experimental data show that the proposed approach outperforms similar activities [20].

Xu, W., et al. proposed the Interpretable Intrusion Detection System, a revolutionary intrusion detection system based on model-based interpretability. They coupled Normal and Attack samples rebuilt with AutoEncoder (AE) with training examples to emphasise the Normal and Attack attributes, resulting in an astonishing effect for the classifier. They then employed Additive Tree (AddTree) as a binary classifier, which offered good predictive performance in the particular combined dataset while preserving adequate model-based interpretability. They investigated the suggested approach using the UNSW-NB15 dataset. According to them, I2DS obtained a recognition accuracy of 99.95 %, which is greater than the bulk of current intrusion detection systems [21].

Kabir, E. et al. have suggested a unique intrusion detection system based on Least Square Support Vector Machine sampling. They've split the detecting procedure into two halves. The whole dataset is separated into specified arbitrary subgroups in the first stage. To detect intrusions, the extracted samples are subjected to the least square support vector machine in the second step. They were able to achieve a reasonable level of accuracy and efficiency [22].

Learning-based classifiers have been proven to be vulnerable to adversarial instances, according to Zhang, F. et al. They illustrate how adversarial inputs adjusted solely based on the model decision outputs may readily evade a discrete-valued random forest classifier. They presented a gradient-free evasion method. Random forests have been shown to be much more vulnerable than SVMs [23].

CyberPulse++, a machine learning-based security system described by Rasool, R. U., et al., uses a pre-trained machine-learning repository to evaluate collected network statistics in real-time to detect abnormal route performance on network links. It efficiently addresses various issues faced by network security solutions, according to them, including the feasibility of large-scale network-level monitoring and data collecting. They have shown that the system can proactively identify and fight against link flooding attacks in real time with little bandwidth and computational overhead [24].

Rasool, R. U., et al. have illustrated the susceptibility of the software-defined networking control layer to link flooding attacks, as well as how the attack technique varies from that

used to attack traditional networks, which mostly entails attacking the connections directly. They introduced CyberPulse, a novel effective countermeasure based on a machine learning-based classifier for preventing link flooding attacks in software-defined networks. They compared CyberPulse to competing techniques for accuracy, false positive rate, and efficacy on actual networks constructed with Mininet. According to them, the results suggest that CyberPulse is capable of accurately classifying harmful traffic and successfully mitigating them [25].

3. Methodology

3.1. Framework of our Proposed Network Intrusion Detection System (NIDS)

Figure 1 show the framework of our proposed network intrusion detection system. It consists of three phases like, feature selection phase, training phase and testing phase. The NSLKDD dataset is given as input to the four feature selection methods like, correlation based feature selection (CFS), information gain ratio (Gain Ratio), chi square (CHI) and information gain (Info gain) [26-28].

3.2. Feature Selection Techniques

(i) Correlation-based feature selection (CFS)

CFS assesses the value of a subset of attributes by taking into account each feature's unique predictive value as well as the degree of redundancy between them. Subsets of features with high correlation with the class but minimal intercorrelation are selected as given in equation (1) [26,29],

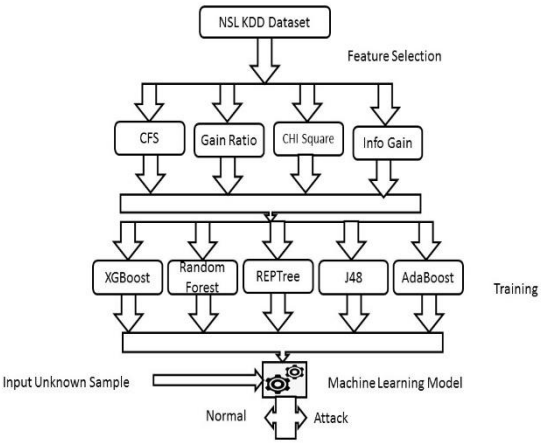

Figure 1. Framework of our proposed NIDS System

$$Merit_s = \frac{k\overline{rcf}}{\sqrt{k + k(k-1)\overline{rff}}} \quad (1)$$

where,

$Merit_s$ = the heuristic ``merit'' of feature subset S containing k features,

$\overline{rcf}$ = the mean feature-class correlation and

$\overline{rff}$ = the average feature-feature intercorrelation

We evaluated CFS with genetic search technique on NSL-KDD training dataset and selected 8 top ranked features as shown in Table 1 out of 41 total features to evaluate the performance of the machine learning classifiers.

Table 1. Top 8 ranked features using Correlation-based feature selection (CFS)

Sr. No.	Attribute No.	Selected Attribute
1	4	flag
2	5	src_bytes
3	6	dst_bytes
4	12	logged_in
5	26	srv_serror_rate
6	29	same_srv_rate
7	30	diff_srv_rate
8	37	dst_host_srv_diff_host_rate

(ii) Chi-squared attribute evaluation (CHI)

To determine the worth of an attribute, CHI computes the value of the chi-squared statistic in relation to the class. The CHI score is typically calculated using an equation (2) [26,30],

$$CHI(a,c) = \frac{N*(XZ - YW)}{(X+W)*(Y+Z)*(X+Y)*(W+Z)} \quad (2)$$

where,
X = the no. of times feature a and class c occur together,
Y = the no. of times feature a occurs without class c,
W = the no. of times class c occurs without feature a,
Z = the no. of times neither a or c occurs and
N = the total size of the training set.
We evaluated chi-squared attribute selection with ranker search technique on NSL KDD training dataset and selected subset 14 top ranked features as shown in Table 2 out of 41 total features to evaluate the performance of the machine learning classifiers.

Table 2. Top 14 ranked features using Chi-squared attribute evaluation (CHI)

Sr. No.	Attribute No.	Selected Attribute
1	5	src_bytes
2	6	dst_bytes
3	3	service
4	4	flag
5	30	diff_srv_rate

6	29	same_srv_rate
7	33	dst_host_srv_count
8	34	dst_host_same_srv_rate
9	35	dst_host_diff_srv_rate
10	12	logged_in
11	23	count
12	25	serror_rate
13	38	dst_host_serror_rate
14	39	dst_host_srv_serror_rate

(iii) Gain Ratio Feature Evaluation

Gain Ratio Feature Evaluation measures the gain ratio in relation to the class to determine the value of an attribute. The solution is provided by the equation (3) [26,29].

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \quad (3)$$

Gain is a criteria for attribute selection in the ID3 approach. It is also referred to as information gain. The property with the largest information gain is chosen as the splitting attribute for the node N in information gain. This function reduces the amount of data required to classify dataset D in a partition and returns the partitions with the lowest impurity. Information gain is defined as the difference in entropy between before and after splitting the dataset D on attribute A. Entropy is used in equation (4) to compute the uncertainty in the dataset D.

$$Entropy(D) = -\sum_{x \in X} p(x) \log_2 p(x) \quad (4)$$

where,
X = the set of classes in dataset D,
p(x) = the proportion of number of elements in class x to the number of elements in dataset D.
SplitInfo describes how equally the attribute splits the dataset and is calculated by equation (5),

$$SplitInfo(A) = -\sum_{j=1}^{n} \frac{|Dj|}{|D|} * \log_2 \left(\frac{|Dj|}{|D|} \right) \quad (5)$$

where,
$\frac{|Dj|}{|D|}$ represents the weight of j^{th} partition.
We evaluated gain ratio feature selection with ranker search technique on NSL KDD training dataset and selected a subset of 14 top ranked features as shown in Table 3 out of 41 total features to evaluate the performance of the machine learning classifiers.

Table 3. Top 14 ranked features using Gain Ratio Feature Evaluation

Sr. No.	Attribute No.	Selected Attribute
1	12	logged_in
2	26	srv_serror_rate
3	4	flag
4	25	serror_rate
5	39	dst_host_srv_serror_rate
6	30	diff_srv_rate
7	38	dst_host_serror_rate
8	6	dst_bytes
9	29	same_srv_rate
10	5	src_bytes
11	3	service
12	37	dst_host_srv_diff_host_rate
13	33	dst_host_srv_count
14	34	dst_host_same_srv_rate

(iv) InfoGain Feature Evaluation

InfoGain Feature Evaluation Measures the information obtained with regard to the class to determine the value of an attribute. The InfoGain score is derived using an equation (6) [26,29],

$$InfoGain(Class, Attribute) = H(Class) - H(Class \mid Attribute) \quad (6)$$

where,
H is the information entropy.
We evaluated InfoGain feature selection with ranker search technique on NSL KDD training dataset and selected a subset of 14 top ranked features as shown in Table 4 out of 41 total features to evaluate the performance of the machine learning classifiers.

Table 4. Top 14 ranked features using InfoGain Feature Evaluation

Sr. No.	Attribute No.	Selected Attribute
1	5	src_bytes
2	6	dst_bytes
3	3	service
4	4	flag
5	30	diff_srv_rate
6	29	same_srv_rate
7	33	dst_host_srv_count
8	34	dst_host_same_srv_rate
9	35	dst_host_diff_srv_rate
10	12	logged_in
11	23	count
12	38	dst_host_serror_rate
13	25	serror_rate
14	39	dst_host_srv_serror_rate

3.3. Classification Methods

Six machine learning methods were used for training and testing. We tested the suggested technique using XGBoost, J48 Decision Tree, AdaBoost, Random Forest, REPTree, and Majority Voting. Similarly, we used the Majority Voting method to create a multi-model classification system. These are the most extensively used and efficient classification methods. To perform the experiments, we employed the WEKA API implementation for these learning approaches. The specifics of each algorithm are as follows [31].

These are the commonly used machine learning algorithms to solve the task of classification. These are widely used and researched algorithms, with applications in a broad variety of fields such as text classification, image classification, intrusion detection, malware detections etc. These algorithms have provided significant and promising classification results in different domains.

(i) XGBoost

XGBoost (eXtreme Gradient Boosting) is a well-known and effective method. Gradient boosting is a supervised learning strategy that integrates an ensemble of estimates from a number of simpler and weaker models in order to anticipate a target variable accurately. The XGBoost approach performs well in machine learning challenges due to its robust handling of a wide range of data types, relationships, and distributions, as well as the vast range of hyperparameters that can be fine-tuned. XGBoost [32] can address regression, classification (binary and multiclass), and ranking problems.

(ii) J48 Decision Tree

One of the most popular classification approaches is J48 decision tree learning. It is highly efficient and has classification accuracy comparable to other learning methods. A decision tree is a tree that reflects the classification model that has been learned. It's an easy-to-understand decision tree classification paradigm. At WEKA, J48 is a modified C4.5. By recursively partitioning data, the C4.5 technique generates a categorization decision tree for a given data set. The depth-first strategy is used to broaden the selection. The method evaluates all feasible data split tests and chooses the one with the highest information gain [33].

(iii) AdaBoost

AdaBoost is the most widely used and researched algorithm, with applications in a broad variety of fields. Freund and Schapire developed the AdaBoost algorithm in 1995. Abstract Boosting is a machine learning approach that combines a large number of weak and wrong classifiers to create a highly accurate classifier. It's easy to use, fast, and simple to understand. It does not need any prior information from the weak learner, therefore it may be used in conjunction with any weak hypothesis identification approach [34].

(iv) Random Forest

As the name indicates, a random forest is made up of a large number of individual decision trees that work together as an ensemble. For each tree, the random forest creates a class prediction, and the class with the most votes becomes our model's forecast. The core concept of The Random Forest is the knowledge of the community, and it is a simple yet powerful one. The random forest model is particularly effective because it consists of a large number of generally uncorrelated models (trees) that work together to outperform each of the individual constituent models [35].

(v) REPTree

The REPTree classifier is a rapid decision tree learner that builds classification and regression trees using the C4.5 method. It constructs a regression/decision tree using information gain/variance and truncates it using error-reduced pruning [36].

(vi) Majority Voting

Voting is the most fundamental ensemble approach, and it is typically pretty effective. It may be used to solve classification and regression problems. It breaks down a model into two or more sub-models, in this case five. The majority voting process is used to integrate predictions from each sub-model. The majority voting method is depicted in Figure 2. It is a meta-classifier that uses a majority vote to identify machine learning classifiers that are conceptually similar or dissimilar. We use majority voting to forecast the final class label, which is the class label that classification models most usually predict. We predict the class label y using equation (7) and the majority vote of each classifier Cj. [26, 37-38],

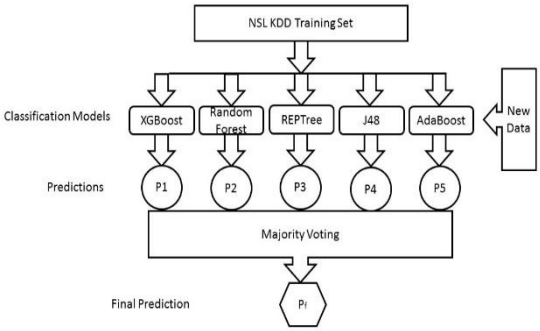

Figure 2. Majority Voting Algorithm

$$y = mode\{C1(x), C2(x), ..., Cm(x)\} \quad (7)$$

where,

y = predicted class label and
C1(x), C2(x),..., Cm(x)=classification models

3.4. Motivations for using the multi-model strategy

Following are the motivations to use multi-model strategy for the detection of network intrusions,

- It's a strategy for improving model performance, with the goal of outperforming any single model in the ensemble.
- Majority Voting is based on the performance of many models, therefore huge mistakes or misclassifications from one model will not be a hindrance.
- A model's poor performance can be compensated for by the good performance of other models.
- When you combine models to produce a forecast, you reduce the chances of one model making an incorrect prediction by having several models that can make the correct prediction.
- Majority Voting makes the estimator more robust and less prone to overfitting.

4. Experimental Setup and Evaluation

4.1. Dataset

The NSL-KDD dataset was created to address some of the drawbacks of the KDD99 dataset. Despite the fact that this revised version of the KDD dataset has significant flaws and may not be a perfect representation of real-world networks, it can still be used as a benchmark dataset to help academics evaluate different intrusion detection systems and address the lack of public records for network-based IDS. The NSL-KDD train and test sets also include a large corpus of data. This advantage allows tests to be run on the complete collection rather than a random sample. As a result, the findings of multiple research studies are consistent and similar.

In this work, the NSL-KDD dataset is used, which contains the datasets KDDTrain+ and KDDTest+. There are a total of 125,973 instances in the KDDTrain+ dataset with 58,630 attack traffic and 67,343 normal traffic. The KDDTest+ set has a total of 22,544 instances. A detailed overview of the instances is shown in Table 5 [39-40].

Table 5. The breakdown of the three sets of the NSL-KDD dataset.

Class	KDDTrain+	KDDTest+
Normal	67343	9711
DoS	45927	7458
PRB	11656	2421
R2L	995	2754
U2R	52	200
Attacks	58630	12833
Total	125973	22544

4.2. Evaluation Measures

To evaluate the performance of classifiers, we used the following metrics. A binary classifier assigns a positive or negative label to all data items in a test dataset. This classification (or prediction) produces four outcomes true positive (TP), true negative (TN), false positive (FP) and false negative (FN) [41].

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (8)$$

$$Error - Rate = \frac{FP + FN}{TP + TN + FN + FP} \quad (9)$$

$$FPR = \frac{FP}{TN + FP} \quad (10)$$

$$FNR = \frac{FN}{TP + FN} \quad (11)$$

4.3. Performance Evaluation of XGBoost classifier using feature selection techniques on NSL-KDD dataset

Table 6 shows the performance evaluation of XGBoost classifier using feature selection techniques on NSL-KDD dataset. Without using feature selection (using all 41 features) XGBoost achieved accuracy of 99.36%, error-rate of 0.6377%, FPR of 0.007 and FNR of 0.006. It takes 8.82 seconds to train the model and 24.1 seconds to test the model. On the other hand, by using feature selection techniques like CFS using 8 features, GainRatio using 14 features, Chi-squared using 14 features and InfoGain using 14 features the classifier achieved acceptable results with minimum system overhead. To train the model, by using CFS takes 2.71 seconds, GainRatio takes 5.21 seconds, Chi-squared takes 5.43 seconds and InfoGain takes 4.3 seconds. To test the model, by using CFS takes 1.37 seconds, GainRatio takes 1.78 seconds, Chi-squared takes 1.52 seconds and InfoGain takes 2.96 seconds. It shows that by using feature selection techniques the classifier achieved acceptable results with minimum system overhead.

4.4. Performance Evaluation of Random Forest classifier using feature selection techniques on NSL-KDD dataset

Table 7 shows the performance evaluation of Random Forest classifier using feature selection techniques on NSL-KDD dataset. Without using feature selection (using all 41 features) Random Forest achieved accuracy of 99.6%, error-rate of 0.4%, FPR of 0.004 and FNR of 0.004. It takes 165.98 seconds to train the model and 2.67 seconds to test the model. On the other hand, by using feature selection techniques like

CFS using 8 features, GainRatio using 14 features, Chi-squared using 14 features and InfoGain using 14 features the classifier achieved acceptable results with minimum system overhead. To train the model, by using CFS takes 95.99 seconds, GainRatio takes 83.22 seconds, Chi-squared takes 83.05 seconds and InfoGain takes 88.75 seconds. To test the model, by using CFS takes 2.67 seconds, GainRatio takes 2.47 seconds, Chi-squared takes 2.03 seconds and InfoGain takes 1.86 seconds. It shows that by using feature selection techniques the classifier achieved acceptable results with minimum system overhead.

4.5. Performance Evaluation of REPTree classifier using feature selection techniques on NSL-KDD dataset

Table 8 shows the performance evaluation of REPTree classifier using feature selection techniques on NSL-KDD dataset. Without using feature selection (using all 41 features) REPTree achieved accuracy of 99.49%, error-rate of 0.5109%, FPR of 0.005 and FNR of 0.005. It takes 9.73 seconds to train the model and 0.25 seconds to test the model. On the other hand, by using feature selection techniques like CFS using 8 features, GainRatio using 14 features, Chi-squared using 14 features and InfoGain using 14 features the classifier achieved acceptable results with minimum system overhead. To train the model, by using CFS takes 2.9 seconds, GainRatio takes 3.24 seconds, Chi-squared takes 3.42 seconds and InfoGain takes 3.16 seconds. To test the model, by using CFS takes 0.08 seconds, GainRatio takes 0.06 seconds, Chi-squared takes 0.06 seconds and InfoGain takes 0.06 seconds. It shows that by using feature selection techniques the classifier achieved acceptable results with minimum system overhead.

4.6. Performance Evaluation of J48 classifier using feature selection techniques on NSL-KDD dataset

Table 9 shows the performance evaluation of J48 classifier using feature selection techniques on NSL-KDD dataset. Without using feature selection (using all 41 features) J48 achieved accuracy of 99.37%, error-rate of 0.6298%, FPR of 0.005 and FNR of 0.007. It takes 57.27 seconds to train the model and 0.09 seconds to test the model. On the other hand, by using feature selection techniques like CFS using 8 features, GainRatio using 14 features, Chi-squared using 14 features and InfoGain using 14 features the classifier achieved acceptable results with minimum system overhead. To train the model, by using CFS takes 7.92 seconds, GainRatio takes 12.96 seconds, Chi-squared takes 16.28 seconds and InfoGain takes 16.14 seconds. To test the model, by using CFS takes 0.05 seconds, GainRatio takes 0.07 seconds, Chi-squared takes 0.07 seconds and InfoGain takes 0.07 seconds. It shows that by using feature selection techniques the classifier achieved acceptable results with minimum system overhead.

4.7. Performance Evaluation of AdaBoost classifier using feature selection techniques on NSL-KDD dataset

Table 10 shows the performance evaluation of AdaBoost classifier using feature selection techniques on NSL-KDD dataset. Without using feature selection (using all 41 features) AdaBoost achieved accuracy of 94.41%, error-rate of 5.5945%, FPR of 0.084 and FNR of 0.003. It takes 39.05 seconds to train the model and 0.09 seconds to test the model. On the other hand, by using feature selection techniques like CFS using 8 features, GainRatio using 14 features, Chi-squared using 14 features and InfoGain using 14 features the classifier achieved acceptable results with minimum system overhead. To train the model, by using CFS takes 6.46 seconds, GainRatio takes 11.85 seconds, Chi-squared takes 12.33 seconds and InfoGain takes 12.07 seconds. To test the model, by using CFS takes 0.06 seconds, GainRatio takes 0.07 seconds, Chi-squared takes 0.07 seconds and InfoGain takes 0.08 seconds. It shows that by using feature selection techniques the classifier achieved acceptable results with minimum system overhead.

4.8. Performance Evaluation of Majority Voting classifier using feature selection techniques on NSL-KDD dataset

Table 11 shows the performance evaluation of multi-model training using Majority Voting classifier using feature selection techniques on NSL-KDD dataset. Without using feature selection (using all 41 features) Majority Voting achieved accuracy of 99.55%, error-rate of 0.4515%, FPR of 0.005 and FNR of 0.004. It takes 350.38 seconds to train the model and 201.46 seconds to test the model. On the other hand, by using feature selection techniques like CFS using 8 features, GainRatio using 14 features, Chi-squared using 14 features and InfoGain using 14 features the classifier achieved acceptable results with minimum system overhead. To train the model, by using CFS takes 157.92 seconds, GainRatio takes 143.58 seconds, Chi-squared takes 123.08 seconds and InfoGain takes 126.16 seconds. To test the model, by using CFS takes 165.56 seconds, GainRatio takes 159.04 seconds, Chi-squared takes 13.97 seconds and InfoGain takes 6.73 seconds. It shows that by using feature selection techniques the classifier achieved acceptable results with minimum system overhead.

Table 6. Performance evaluation of XGBoost classifier using and without using feature selection techniques on NSL-KDD dataset

FS	ST	#Features	Accuracy (%)	Error-rate (%)	FPR	FNR	Train time (sec.)	Test time (sec.)
Without feature selection		41	99.36	0.6377	0.007	0.006	8.82	24.1
CFS	GreedyStepwise	8	98.82	1.1843	0.011	0.013	2.71	1.37
GainRatio	Ranker	14	99.22	0.7822	0.009	0.007	5.21	1.78
CHI	Ranker	14	99.28	0.7208	0.008	0.007	5.43	1.52
InfoGain	Ranker	14	99.28	0.7208	0.008	0.007	4.3	2.96

Table 7. Performance evaluation of Random Forest classifier using and without using feature selection techniques on NSL-KDD dataset

FS	ST	#Features	Accuracy (%)	Error-rate (%)	FPR	FNR	Train time (sec.)	Test time (sec.)
Without feature selection		41	99.6	0.4	0.004	0.004	165.98	2.67
CFS	GreedyStepwise	8	99.11	0.8912	0.008	0.01	95.99	2.67
GainRatio	Ranker	14	99.53	0.4654	0.004	0.005	83.22	2.47
CHI	Ranker	14	99.58	0.4238	0.005	0.004	83.05	2.03
InfoGain	Ranker	14	99.58	0.4238	0.005	0.004	88.75	1.86

Table 8. Performance evaluation of REPTree classifier using and without using feature selection techniques on NSL-KDD dataset

FS	ST	#Features	Accuracy (%)	Error-rate (%)	FPR	FNR	Train time (sec.)	Test time (sec.)
Without feature selection		41	99.49	0.5109	0.005	0.005	9.73	0.25
CFS	GreedyStepwise	8	98.96	1.0357	0.01	0.011	2.9	0.08
GainRatio	Ranker	14	99.33	0.6654	0.007	0.007	3.24	0.06
CHI	Ranker	14	99.44	0.5565	0.006	0.005	3.42	0.06
InfoGain	Ranker	14	99.44	0.5565	0.006	0.005	3.16	0.06

Table 9. Performance evaluation of J48 classifier using and without using feature selection techniques on NSL-KDD dataset

FS	ST	#Features	Accuracy (%)	Error-rate (%)	FPR	FNR	Train time (sec.)	Test time (sec.)
Without feature selection		41	99.37	0.6298	0.005	0.007	57.27	0.09
CFS	GreedyStepwise	8	98.98	1.0179	0.009	0.011	7.92	0.05
GainRatio	Ranker	14	99.42	0.5822	0.006	0.006	12.46	0.07
CHI	Ranker	14	99.45	0.5525	0.005	0.006	16.28	0.07
InfoGain	Ranker	14	99.45	0.5525	0.005	0.006	16.14	0.07

Table 10. Performance evaluation of AdaBoost classifier using and without using feature selection techniques on NSL-KDD dataset

FS	ST	#Features	Accuracy (%)	Error-rate (%)	FPR	FNR	Train time (sec.)	Test time (sec.)
Without feature selection		41	94.41	5.5945	0.084	0.03	39.05	0.09
CFS	GreedyStepwise	8	91.64	8.3571	0.089	0.078	6.46	0.06
GainRatio	Ranker	14	93.41	6.5867	0.09	0.044	11.85	0.07
CHI	Ranker	14	92.53	7.4739	0.083	0.067	12.33	0.07
InfoGain	Ranker	14	92.53	7.4739	0.083	0.067	12.07	0.08

Table 11. Performance evaluation of Majority Voting classifier using and without using feature selection techniques on NSL-KDD dataset

FS	ST	#Features	Accuracy (%)	Error-rate (%)	FPR	FNR	Train time (sec.)	Test time (sec.)
Without feature selection		41	99.55	0.4515	0.005	0.004	350.38	201.46
CFS	GreedyStepwise	8	99.03	0.9625	0.009	0.01	157.92	165.56
GainRatio	Ranker	14	99.48	0.5208	0.006	0.005	143.58	159.04
CHI	Ranker	14	99.5	0.501	0.005	0.005	123.08	13.97
InfoGain	Ranker	14	99.5	0.501	0.005	0.005	126.16	6.73

4.8. Comparison of our proposed approach using Majority Voting classifier with other works on NSL-KDD dataset

Table 12 and Figure 3 shows the comparative performance evaluation of our proposed approach using Majority Voting classifier with other similar works on the NSL-KDD dataset. It is found that our approach outperforms the other approaches in the attack detection accuracy of 99.50% using only 14 selected features out of 41 features.

Table 12. Comparison of our approach using Majority Voting classifier with other works on NSL-KDD dataset

Approach	Method	Accuracy (%)
Azidine Guezzaz et al. [18]	Decision Tree with Enhanced Data Quality (DTE)	99.42
Masdari and Khezri [12]	Five classification	96.70
Ahmim et al. [42]	Tree algorithm	89.24
Fang et al. [43]	Random Forest	99.33
Proposed approach	Majority Voting	99.50

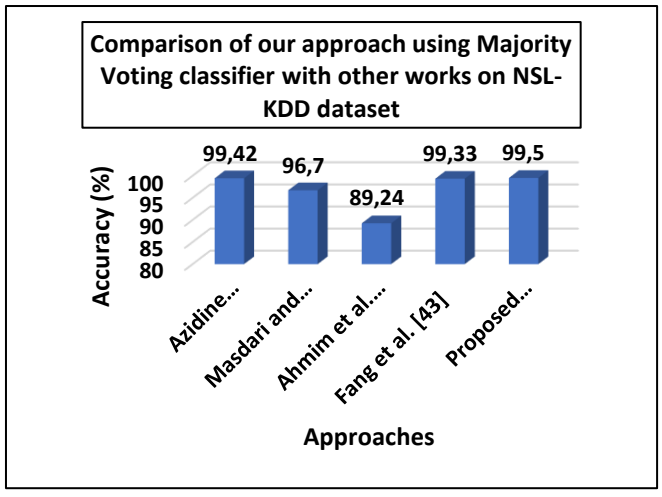

Figure 3. Comparison of our approach using Majority Voting classifier with other works on NSL-KDD dataset

5. Discussion

5.1. Positive impacts of Network Intrusion Detection System

Following are some of the positive impacts of network intrusion detection systems.

- NIDS can be configured to display the specific information included within the packets. This feature can be used to detect intrusions such as exploitation attacks and botnet-infected packets.
- NIDS looks at the number and types of attacks. This information can be utilised to improve security measures. It can also be examined for flaws with network settings.
- To satisfy certain standards, NIDS logs can be used as documentation.
- This increased efficiency can help a business to save money on employees while also covering the costs of installing the NIDS.

5.2. Negative impacts of Network Intrusion Detection System

Following are some of the negative impacts of network intrusion detection systems.

- NIDS does not stop or prevent attacks; rather, it aids in their detection.
- NIDS is extremely valuable for network monitoring, but the value of the information it provides is entirely dependent on what you do with it.
- Intruders can employ encrypted packets to sneak into the network since an NIDS can't see them.
- Although a NIDS reads the data from an IP packet, the network address may still be faked.
- One major drawback of a NIDS is that it frequently alerts false positives.
- Because an NIDS examines protocols as they are collected, they are vulnerable to the same protocol-based attacks as network hosts.
- An NIDS's signature library determines how effective it is. It won't register the latest attacks if it isn't updated often, and it won't be able to warn about them.

5.3. Benefits for the academic community and government

Following are the benefits of NIDS for the academic community and government.

- Academic community can use NIDS as a research platform for further study to solve open challenges in thus field.
- They can provide more significant solutions to detect the new threats in an efficient way with minimum false positives.
- Government agencies and organizations can effectively use NIDS to detect the threats before they can expose the systems.

6. Conclusions

We provided feature selection and majority voting-based classification for detecting attacks using the NSL-KDD dataset. Correlation-based feature selection (CFS) with GreedyStepwise, Chi-squared attribute evaluation (CHI) with Ranker, Gain Ratio Feature Evaluation with Ranker, and InfoGain Feature Evaluation with Ranker search strategies were used for feature selection. Using these feature selection strategies, we picked 8 features with CFS, 14 with CHI, 14 with GainRatio, and 14 with InfoGain. To evaluate the performance of various feature selection techniques and selected features, we used six machine learning classifiers: XGBoost, Random Forest, AdaBoost, REPTree, J48 Decision Tree, and Majority Voting. We built a multi-model classification model using the Majority voting classifier. According to the testing findings, our suggested solution, which employs a Majority Voting classifier and feature selection algorithms such as CHI and InfoGain, achieved a 99.50 % attack detection accuracy with minimum system overhead. Furthermore, we compared our suggested technique to other comparable approaches and found that it outperforms in terms of attack detection accuracy.

References

[1] Leevy, J. L., Khoshgoftaar, T. M. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. Journal of Big Data, 2020, 7(1), 1-19.

[2] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2019, 2(1), 1-22.

[3] Laghrissi, F., Douzi, S., Douzi, K., Hssina, B. Intrusion detection systems using long short-term memory (LSTM). Journal of Big Data, 2021, 8(1), 1-16.

[4] Megantara, A. A., & Ahmad, T. A hybrid machine learning method for increasing the performance of network intrusion detection systems. Journal of Big Data, 2021, 8(1), 1-19.

[5] Jadhav, A. D., Pellakuri, V. Highly Accurate and Efficient Two Phase-Intrusion Detection System (TP-IDS) using Distributed Processing of HADOOP & Machine Learning Techniques, 2021.

[6] Divyasree, T. H., Sherly, K. K. A network intrusion detection system based on ensemble CVM using efficient feature selection approach. Procedia computer science, 2018, 143, 442-449.

[7] Ashiku, L., Dagli, C. Network Intrusion Detection System using Deep Learning. Procedia Computer Science, 2021, 185, 239-247.

[8] Di Mauro, M., Galatro, G., Fortino, G., Liotta, A. (2021). Supervised feature selection techniques in network intrusion detection: A critical review. Engineering Applications of Artificial Intelligence, 2021, 101, 104216.

[9] Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S. H. T., Rashidi, S., Rahmani, A. M. Deep learning-based intrusion detection systems: a systematic review. IEEE Access, 2021, 9, 101574-101599.

[10] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 2021, 32(1), e4150.

[11] Hamid, Y., Balasaraswathi, V. R., Journaux, L., Sugumaran, M. Benchmark Datasets for Network Intrusion Detection: A Review. Int. J. Netw. Secur., 2018, 20(4), 645-654.

[12] Masdari, M., Khezri, H. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. Applied Soft Computing, 2020, 92, 106301.

[13] Elmasry, W., Akbulut, A., Zaim, A. H. A Design of an Integrated Cloud-based Intrusion Detection System with Third Party Cloud Service. Open Computer Science, 2021, 11(1), 365-379.

[14] Sistla, V. P. K., Kolli, V. K. K., Voggu, L. K., Bhavanam, R., Vallabhasoyula, S. Predictive Model for Network Intrusion Detection System Using Deep Learning. Rev. d'Intelligence Artif., 2020, 34(3), 323-330.

[15] Sohi, S. M., Seifert, J. P., Ganji, F. RNNIDS: Enhancing network intrusion detection systems through deep learning. Computers & Security, 2021, 102, 102151.

[16] Zhou, Y., Cheng, G., Jiang, S., Dai, M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. Computer networks, 2020, 174, 107247.

[17] Mane, S., Rao, D. Explaining Network Intrusion Detection System Using Explainable AI Framework. arXiv preprint arXiv:2103.07110. 2021.

[18] Guezzaz, A., Benkirane, S., Azrour, M., Khurram, S. A Reliable Network Intrusion Detection Approach Using Decision Tree with Enhanced Data Quality. Security and Communication Networks, 2021.

[19] Li, L., Yu, Y., Bai, S., Cheng, J., Chen, X. Towards effective network intrusion detection: A hybrid model integrating gini index and GBDT with PSO. Journal of Sensors, 2018.

[20] Moualla, S., Khorzom, K., Jafar, A. Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset. Computational Intelligence and Neuroscience, 2021.

[21] Xu, W., Fan, Y., Li, C. I^2DS: Interpretable Intrusion Detection System Using Autoencoder and Additive Tree. Security and Communication Networks, 2021.

[22] Kabir, E., Hu, J., Wang, H., Zhuo, G. A novel statistical technique for intrusion detection systems. Future Generation Computer Systems, 79, 303-318, 2018.

[23] Zhang, F., Wang, Y., Liu, S., Wang, H. Decision-based evasion attacks on tree ensemble classifiers. World Wide Web, 23(5), 2957-2977, 2020.

[24] Rasool, R. U., Ahmed, K., Anwar, Z., Wang, H., Ashraf, U., Rafique, W. CyberPulse++: A machine learning-based security framework for detecting link flooding attacks in

software defined networks. International Journal of Intelligent Systems, 36(8), 3852-3879, 2021.
[25] Rasool, R. U., Ashraf, U., Ahmed, K., Wang, H., Rafique, W., & Anwar, Z. Cyberpulse: a machine learning based link flooding attack mitigation system for software defined networks. IEEE Access, 7, 34885-34899, 2019.
[26] Patil, D. R., Patil, J. B. Malicious web pages' detection using feature selection techniques and machine learning. International Journal of High Performance Computing and Networking, 2019, 14(4), 473-488.
[27] Patil, D. R., Pattewar, T. M. A comparative performance evaluation of machine learning-based NIDS on benchmark datasets. International Journal of Research in Advent Technology, 2014, 2(2), 101-106.
[28] Kshirsagar, V. P., Patil, D. R. An overview of adaboost-based NIDS and performance evaluation on NSL-KDD dataset. International Journal of Computer Engineering and Computer Application, 2010, 1.
[29] Basnet, R. B., Sung, A. H., Liu, Q. Feature selection for improved phishing detection. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, Berlin, Heidelberg, 2012, pp. 252-261
[30] Rajab, K. D. New hybrid features selection method: A case study on websites phishing. Security and Communication Networks, 2017.
[31] Weka 3.9: Data Mining Software in Java [online] http://www.cs.waikato.ac.nz/ml/weka/ (accessed 15 November 2021).
[32] A Brief Introduction to XGBoost, https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6 / (accessed 15 November 2021).
[33] Quinlan, J. R. Induction of decision trees. Machine learning, 1986, 1(1), 81-106.
[34] Schapire, R. E. Explaining adaboost. In: Empirical inference. Springer, Berlin, Heidelberg, 2013, pp. 37-52.
[35] Houtao Deng, An Introduction to Random Forest, https://towardsdatascience.com/random-forest-3a55c3aca46d (accessed 15 November 2021).
[36] REPTree:http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html (accessed 15 November 2021).
[37] EnsembleVoteClassifier:https://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/, (accessed 15 November 2021).
[38] EnsembleWeka: How to Use Ensemble Machine Learning Algorithms in Weka, http://machinelearningmastery.com/use-ensemble-machine-learning-algorithms-weka/ (accessed 15 November 2021).
[39] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications, IEEE, 2009, pp. 1-6.
[40] NSL-KDD dataset, https://www.unb.ca/cic/datasets/nsl.html, (accessed 15 November 2021).
[41] Saito, T. and Rehmsmeier, M. Basic Evaluation Measures from the Confusion Matrix. https://classeval.wordpress.com/%20introduction/basic-evaluation-measures/, (accessed 15 November 2021).
[42] Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., Janicke, H. A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, 2019, pp. 228-233.
[43] Fang, W., Tan, X., Wilbur, D. Application of intrusion detection technology in network safety based on machine learning. Safety Science, 2020, 124, 104604.