Use the ensemble methods when detecting DoS attacks in Network Intrusion Detection Systems

Hoang Ngoc Thanh[1,*], Tran Van Lang[2]

[1]Lac Hong University, Vietnam
[2]Institute of Applied Mechanics and Informatics, VAST, Vietnam

Abstract

Building a good IDS model from a certain dataset is one of the main tasks in machine learning. Training multiple classifiers at the same time to solve the same problem and then combining their outputs to improve classification quality, called ensemble method. This paper analyzes and evaluates the performance of using known ensemble techniques such as Bagging, AdaBoost, Stacking, Decorate, Random Forest and Voting to detect DoS attacks on UNSW-NB15 dataset, created by the Australian Cyber Security Center 2015. The experimental results show that the Stacking technique with heterogeneous classifiers for the best classification quality with $F - Measure$ is 99.28% compared to 98.61%, which is the best result are obtained by using single classifiers and 99.02% by using the Random Forest technique.

Received on 13 November 2019; accepted on 28 November 2019; published on 29 November 2019

Keywords: Machine Learning, Ensemble Classifier, Stacking, DoS, UNSW-NB15 dataset.

doi:10.4108/eai.29-11-2019.163484

1. Introduction

For companies that are constantly connected to the internet in today's technology age, the terms DoS and DDoS are not too strange. Especially in recent years, news that the information and technology division of large organizations around the world has been hacked and the stolen data always contains the terms DoS and DDoS.
The full name of "DoS" is "Denial of Service" and DDoS is "Distributed Denial of Service", is a form of denial of service attack. This is a fairly common form of attack today, it makes the target computer can not handle the task and lead to overload. These DoS attacks often target virtual servers (VPS) or web servers of large businesses such as banks, governments or e-commerce websites, ... A fourth quarter security report produced by Kaspersky [1] indicated that the source attack of DDoS originated from 86 nations with attack duration up to 279 hours. The most common type of attack is still SYN flooding (79.7%), with UDP flooding in second place (9.4%). The least popular is ICMP flooding (0.5%). Johnson Singh et al. [2] claimed that 540Gbps DDoS attack occurred on 31st August 2016 against a federal government official website of Rio Olympic 2016 and the Ministry of Brazilian Sport. Accordingly, DDoS attacks have been around for decades and are not particularly sophisticated, at least considering the most modern network attack standards. But DDoS attacks continue to be a popular method for attackers.
An Intrusion Detection System (IDS) is an important tool used to monitor and identify intrusion attacks. To determine whether an intrusion attack has occurred or not, IDS depends on several approaches. The first is a signature-based approach, in which the known attack signature is stored in the IDS database to match the current system data. When IDS finds a match, it recognizes it as an intrusion. This approach provides a quick and accurate detection. However, the disadvantage of this is having to update the signature database periodically. Additionally, the system may be compromised before the latest intrusion attack can be updated. The second approach is based on anomalies or behavior-based, in which IDS will identify an attack when the system operates without rules. This approach can detect both known and unknown attacks. However, the disadvantage of this method is low accuracy with a high false alarm rate.

*Corresponding author. Email: thanhhn@bvu.edu.vn

Building a good IDS model from a certain dataset is one of the main tasks in machine learning. A strong classification is desirable, but it is difficult to find. Marwane Zekri et al. [3] designed a DDoS detection system based on the C.4.5 algorithm to mitigate the DDoS threat. This algorithm, coupled with signature detection techniques, generates a decision tree to perform automatic, effective detection of signatures attacks for DDoS flooding attacks. In another study by Pei et al. [4], the random forest algorithm is used to train the attack detection model. The experimental results show that the proposed DDoS attack detection method based on machine learning has a good detection rate for the current popular DDoS attack.
The idea of training multiple classifiers at the same time to solve the same problem, and then combine their output to improve accuracy, called the ensemble method. When a combination, also known as a multi-classification system, is based on learners of the same type, it is called a homogeneous ensemble. When it is based on learners of different types, it is called a heterogeneous ensemble. Generally, the generalization ability of a ensemble classifier is better than a single classifier, because it can increase weak classifiers to produce better results than a single strong classifier.
In this paper we evaluated and analyzed six different ensemble classifier techniques, called Bagging, AdaBoost, Stacking, Decorate, Random Forest and Voting, using various basic classifiers such as Decision Trees (DT), Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), k nearest neighbors (KNN) and Random Tree (RT); These were applied on the UNSW-NB15 dataset.
The remainder of this paper is organized as follows: Section 2 presents the ensemble machine learning methods used in the experiments; Section 3 presents the datasets, the evaluation metrics and the results obtained by using ensemble techniques when detecting a DoS attack on the UNSW-NB15 dataset; and Section 4 is discussions and issues need to be further studied.

2. Ensemble techniques

Since the 1990s, the machine learning community has been studying ways to combine multiple classification models into a set of classification models for greater accuracy than a single classification model. The purpose of aggregation models is to reduce variance and / or bias of algorithms. Bias is a conceptual error of geometric models (not related to learning data) and variance is an error due to the variability of the model compared to the randomness of the data samples (Figure 1). Buntine [5] introduced Bayesian techniques to reduce variance of learning methods. Wolpert's stacking method [6] aims to minimize the bias of algorithms. While Freund and Schapire [7] introduced Boosting, Breiman [8] suggested ArcX4 to reduce bias and variance, while Breiman's Bagging [9] reduced the variance of the algorithm but did not increase the bias too much. Approach to random forests [10] one of the most successful model collection methods. The random forest algorithm builds trees without branches to keep the bias low and uses randomness to control the low correlation between trees in the forest.

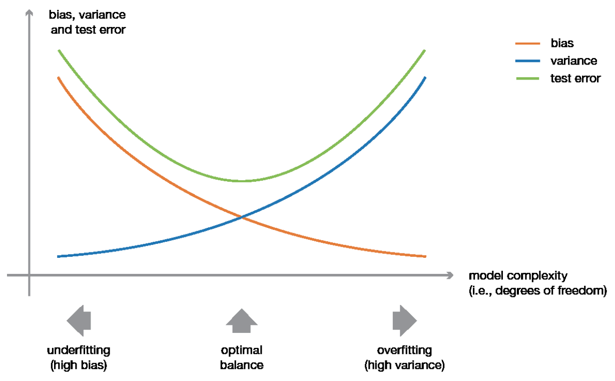

Figure 1. Illustration of the bias-variance tradeoff.

2.1. Bootstrap

A very well known method in statistics introduced by Bradley Efron in 1979 [11]. This method is mainly used to estimate standard errors, deviations and calculate confidence intervals for parameters. This method is performed as follows: From an initial population, a sample $L = (x_1, x_2, ..., x_n)$ consisting of n instances, calculation of the desired parameters. In the following steps, repeat b times the creation of the L_b pattern also includes n instances from L by retrieving the sample with the replacement of the instances in the original sample and then calculating the desired parameters.

2.2. Bagging (Bootstrap Aggregation)

This method is considered as a method of summarizing the results obtained from Bootstrap. The main ideas of this method are as follows: Give a training set $D = \{(x_i, y_i) : i = 1, 2, ..., n\}$ and suppose we want to make a prediction for the variable x. A sample of B datasets, each of which consists of n randomly selected elements from D with substitution (like Bootstrap). Therefore $B = (D_1, D_2, ..., D_B)$ looks like a set of cloned training sets; Train a machine or model for each set of $D_b (b = 1, 2, ..., B)$ and collect the predicted results in turn on each set of D_b; The final aggregate results are calculated by means of regression or classification of the most votes (classification).

2.3. Boosting

Unlike the Bagging method, which builds up a classifier in ensemble with training instances of equal weight, the Boosting method builds a classifier in ensemble with different weighted training instances. After each iteration, the incorrectly anticipated training instances will be weighted, and the correctly predicted training instances will be rated smaller. This helps Boosting focus on improving accuracy for instances that are incorrectly predicted after each iteration.

An Boosting algorithm was originally defined as an algorithm used to convert a weak machine learning algorithm into a strong machine learning algorithm. This means that it converts a machine learning algorithm that solves a binary classification problem better than a random solution into an algorithm that solves the problem well. Schapire's original Boosting algorithm is a recursive algorithm. At the end of recursion, it combines the hypotheses generated by weak machine learning algorithms. The error probability of this ensemble is proven to be smaller than the error probability of weak hypotheses.

Adaboost is an algorithm that combines a diverse set of classifiers by running machine learning algorithms with different distributions on the training set [12].

2.4. Stacking

Stacking is a way to combine multiple models, introducing the concept of meta classifiers. It is less widely used than Bagging and Boosting. Unlike Bagging and Boosting, Stacking can be used to combine different models. The process is as follows:

(1) Divide the training into two separate sets.

(2) Train the base classiers at the beginning.

(3) Test the base classifier in the second part.

(4) Use the predictions in (3) as inputs, and the correct classification results as outputs to train a meta classifier.

In Stacking, the combined mechanism is that the output of the classifiers (level 0 classifiers) will be used as training data for another classifier (level 1 classifier) to produce the result most accurate forecasts. Basically, we allow level 1 classifier (meta classifier) to find the best mechanism for combining level 0 classifiers on their own.

2.5. Random Forest

Random Forest (RF) is a classification method developed by Leo Breiman at the University of California, Berkeley. The summary of the RF algorithm for stratification is explained as follows:

- Get K bootstrap instances from the training set.
- For each bootstrap pattern, an unpruned tree is constructed as follows: At each node, instead of choosing the best division among all predicted variables, we randomly select a sample m of the predicted variable then chooses the best division among these variables.
- Make predictions by summing up the predictions of K trees.

The learning of the RF includes the random use of input values or a combination of those values at each node in the process of constructing a decision tree. RF has some strong points:

(1) High precision;

(2) The algorithm solves problems with lots of noise data;

(3) The algorithm runs faster than other ensemble machine learning algorithms;

(4) There are intrinsic estimates such as the accuracy of the conjecture model or the strength and relevance of the features;

(5) Easy to perform in parallel.

However, to achieve these strengths, the execution time of the algorithm is quite long and requires a lot of system resources.

Through the above findings about RF algorithm, we have commented that RF is a good classification method because:

(1) In RF, the errors (variance) are minimized because the results of RF are synthesized through many learners;

(2) Random selection at each step in the RF will reduce the correlation between learners in summing up the results.

2.6. Decorate

In Decorate (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples), a combination is created repeatedly, first learning a classifier and then adding it to the current combination. We initialize the association to contain the trained classifier for the given training data. Classifiers in each successive iteration are trained on initial training data in conjunction with some artificial data. In each iteration, artificial training instances are created from data distribution; in which the number of instances created is determined to be a part, *Rsize*, of the training file size. The labels for these artificially created training instances are chosen to be the maximum different from the predictions of the current population. The creation of artificial data is explained in more detail later. We refer to the labeled training set that is labeled as diverse data. We train a new classifier on the combination of initial training data and diverse data, thus forcing it different from the current suits. Therefore, adding this category to the mix will increase its diversity. While forced to diversity, we still want to maintain training accuracy. We do this by rejecting a new classifier if adding it to an existing collection reduces its accuracy. This process is repeated until we reach the desired committee size or exceed the maximum number of iterations.

3. Experiments

Programs and algorithms in the experiment using the Java programming language, based on the library, Weka machine learning framework developed by Waikato University, New Zealand [13].
Part 3.1 presents the datasets used in the experiment.
Part 3.2 presents the evaluation metrics used in the experiment.
Part 3.3 presents the results obtained by using homogeneous and heterogeneous ensemble techniques when detecting a Buzzers attack on the UNSW-NB15 dataset.

3.1. Datasets

About the datasets used in IDS; KDD99, NSL-KDD and UNSW-NB15 are the most popular datasets, according to statistics from 2015 to 2018, showing that the NSL-KDD dataset is used 38%, KDD99 is 23% and UNSW-NB15 is 12% [14].

Types of attacks	Testing dataset		Training dataset	
Normal	56.000	31,94%	37.000	44,94%
Analysis	2.000	1,14%	677	0,82%
Backdoor	1.746	1,00%	583	0,71%
DoS	12.264	6,99%	4.089	4,97%
Exploits	33.393	19,04%	11.132	13,52%
Fuzzers	18.184	10,37%	6.062	7,36%
Generic	40.000	22,81%	18.871	22,92%
Reconnaissance	10.491	5,98%	3.496	4,25%
Shellcode	1.133	0,65%	378	0,46%
Worms	130	0,07%	44	0,05%
Total	175.341	100,00%	82.332	100,00%

Table 1. Information about UNSW-NB15 dataset [9].

UNSW-NB15 dataset contains 2,540,044 instances. Part of this dataset is divided into training and testing datasets, which are used extensively in scholars' experiments, detailed information about the datasets is presented in Table 1. This training and testing datasets contains a total of 9 types of attacks: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms. In this paper, the UNSW-NB15 dataset was used for experiments because [15]:

(1) It contains modern normal behavior and contemporary synthesis attack activities.

(2) The probability distribution of training and testing datasets is similar.

(3) It includes a set of features from payload and header of packages to reflect effective network packets.

(4) The complexity of evaluating UNSW-NB15 for existing classification systems shows that this dataset has complex patterns [14], especially when classifying DoS attacks [16].

This means that the dataset can be used to evaluate current and new classification methods effectively and reliably. However, because the UNSW-NB15 dataset is still quite new, it has not been used by many scholars in their studies. Therefore, there are limitations when comparing results with other studies.

3.2. Evaluation metrics

We denote:

- TP_i : the number of correctly classified instances for class c_i
- FP_i : the number of instances that were incorrectly classified to the class c_i
- TN_i : the number of correctly classified instances that do not belong to the class c_i
- FN_i : the number of instances that were not classified as belonging to the class c_i

The performance evaluation of the classifiers is done by measuring and comparing metrics:

- $Accuracy_i = (TP_i + TN_i)/(TP_i + FP_i + TN_i + FN_i)$
- $Sensitivity_i = TPR_i = TP_i/(TP_i + FN_i)$
- $Specificity_i = TNR_i = TN_i/(TN_i + FP_i)$
- $Efficiency_i = (Sensitivity_i + Specificity_i)/2$
- $Precision_i = TP_i/(TP_i + FP_i)$
- $FNR_i = FN_i/(FN_i + TP_i)$
- $FPR_i = FP_i/(FP_i + TN_i)$
- Time for training and testing

The use of *Accuracy* to assess the quality of classification has been used by many scholars. However, the class distribution in most nonlinear classification problems is very imbalanced. Therefore, using *Accuracy* to evaluate the quality of classification of a model is not really effective [17]. Therefore, the more comprehensive metrics recommended for the evaluation of $F-Measure$ and $G-Means$ are calculated as follows [18], [19]:

$$F-Measure_i = \frac{(1+\beta^2) \times Precision_i \times Recall_i}{\beta^2 \times Precision_i + Recall_i}$$

Here, β is the coefficient that adjusts the relationship between Precision and Recall and usually $\beta = 1$. $F-Measure$ shows the harmonious correlation between $Precision$ and $Recall$. $F-measure$ values are high when both $Precision$ and $Recall$ are high. And the $G-Means$ indicator is calculated:

$$G-Means_i = \sqrt{Sensitivity_i \times Specificity_i}$$

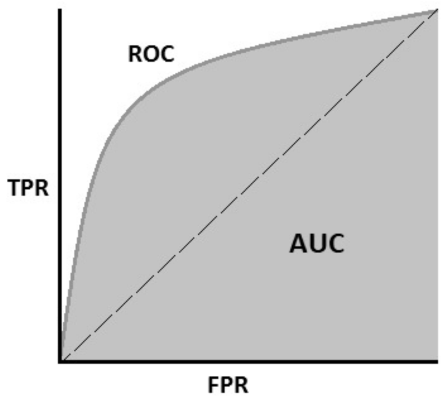

Figure 2. AUC - ROC Curve.

ROC (Receiver Operating Characteristics) is a method of calculating the performance of a classification model according to different classification thresholds. Assuming a binary classification problem (2 classes) using Logistic Regression, the selection of different classification thresholds [0..1] will affect the classification ability of the model and the level of influence of thresholds needs to be calculated. ROC is a probability and Area Under The Curve (AUC) representing the degree of classification of the model. Meaning of the interpretable AUC: It is the probability that a randomly sampled positive sample will be ranked higher than a randomized negative sample, $AUC = P((score(x^+) > score(x^-))$ The higher the AUC, the more accurate the model is in classifying classes. The ROC curve represents the pair of metrics (TPR, FPR) at each threshold with TPR is the vertical axis and FPR is the horizontal axis (Figure 2).
The evaluation metrics used in the experiments of this paper are: $Sensitivity$, $Specificity$, $Precision$, $F-Measure$, $G-Means$, AUC, Training time and Testing time.

3.3. Experimental results

We apply various machine learning algorithms in the misuse detection module in order to find the best method for detecting DoS attacks based on $F-Measure$, $G-Means$, AUC and speed (computation time). We use six single algorithms from the Weka Data Mining Tools: J48 (DT), NaiveBayes (NB), Logistic (LR), LibSVM (SVM), IBk (KNN) and RandomTree (RT), then apply these algorithms into six different ensemble classifiers, which are Bagging, AdaBoost, Stacking, Decorate, Voting and Random Forest, as shown in Figure 3 below.

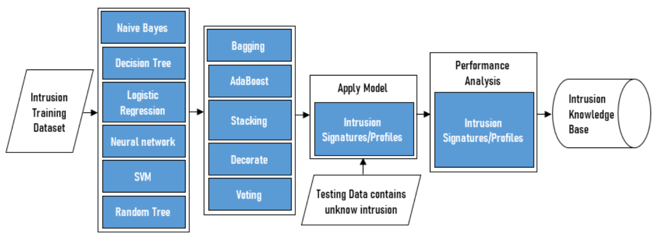

Figure 3. The DoS attacks detection model using a ensemble techniques.

The classification results are presented in Table 2, whereby KNN was selected as the best single classification solution because of the highest $F-Measure$ index (0.9416). As a reminder, $F-Measure$ shows a good correlation between $Precision$ and $Recall$.

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9767	0.7012	0.8581	0.7558	0.9895	0.9733
Specificity	0.9949	0.8756	0.9821	0.9551	0.9954	0.9944
Precision	0.9806	0.6002	0.9272	0.8177	0.9828	0.9787
F-Measure	0.9787	0.6468	0.8913	0.7855	0.9861	0.9760
G-Means	0.9857	0.7836	0.9180	0.8496	0.9925	0.9838
AUC	0.9900	0.8866	0.9846	0.8555	0.9992	0.9838
Training time	10.1 s	608 ms	12.67 s	517.44 s	20 ms	753 ms
Testing time	91.19 s	6207 ms	137.9 s	4672 s	723.72 s	6.69 s

Table 2. Results of using single classifier when classifying DoS on UNSW-NB15 dataset

With Bagging technique, classification results are presented in Table 3, whereby Bagging technique using component classifiers as RT is selected because of the highest $F-Measure$ index (0.9594). Here, RT usually refer to randomly constructed trees that have nothing to do with machine learning. However, the Weka machine learning framework uses this term to refer to decision trees built on a random subset of features.
With AdaBoost technique, classification results are presented in Table 4, whereby AdaBoost technique using component classifiers as DT is selected because of the highest $F-Measure$ index (0.9676).
With Stacking technique, the meta classifier chosen to use is KNN (this is the best result chosen after

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9847	0.7017	0.8584	0.7552	0.9870	0.9858
Specificity	0.9974	0.8734	0.9821	0.9551	0.9953	0.9979
Precision	0.9901	0.5962	0.9274	0.8174	0.9824	0.9919
F-Measure	0.9874	0.6447	0.8915	0.7851	0.9847	0.9889
G-Means	0.9910	0.7829	0.9181	0.8493	0.9911	0.9918
AUC	0.9996	0.8867	0.9847	0.8659	0.9994	0.9996
Training time	93s	11.8s	187.6s	5132s	2796s	6.7s
Testing time	815.4s	117s	1630.6s	43822s	29086s	58.97s

Table 3. Results of using Bagging when classifying DoS on UNSW-NB15 dataset

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9899	0.9076	0.8581	0.8411	0.9936	0.9752
Specificity	0.9986	0.7718	0.9821	0.9619	0.9963	0.9940
Precision	0.9947	0.5144	0.9272	0.8545	0.9860	0.9774
F-Measure	0.9923	0.6566	0.8913	0.8478	0.9898	0.9763
G-Means	0.9943	0.8370	0.9180	0.8995	0.9949	0.9845
AUC	0.9996	0.9164	0.9735	0.9783	0.9958	0.9846
Training time	109.99s	15.82s	42.06s	31347s	11041s	536ms
Testing time	883.17s	186.13s	377.74s	258518s	91224s	2755ms

Table 4. Results of using AdaBoost when classifying DoS on UNSW-NB15 dataset

using many different machine learning techniques). Classification results are presented in Table 5, whereby Stacking technique using component classifiers as KNN is selected because of the highest $F-Measure$ index (0.9453).

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9754	0.5932	0.8531	0.7558	0.9900	0.9772
Specificity	0.9950	0.9547	0.9738	0.9551	0.9960	0.9944
Precision	0.9811	0.7773	0.8968	0.8178	0.9852	0.9789
F-Measure	0.9782	0.6729	0.8744	0.7856	0.9876	0.9781
G-Means	0.9851	0.7526	0.9115	0.8497	0.9930	0.9858
AUC	0.9900	0.8475	0.9579	0.8549	0.9990	0.9855
Training time	121.08 s	6831 ms	131.61 s	4568.05 s	745.68 s	7.74 s
Testing time	1138.1 s	280.86 s	1237 s	39027.4 s	6711.2 s	347.3 s

Table 5. Results of using Stacking when classifying DoS on UNSW-NB15 dataset

With Decorate technique, classification results are presented in Table 6, whereby Decorate technique using component classifiers as RT is selected because of the highest $F-Measure$ index (0.9647).

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9835	0.6190	0.8581	0.7558	0.9900	0.9889
Specificity	0.9971	0.9434	0.9821	0.9551	0.9947	0.9982
Precision	0.9890	0.7445	0.9272	0.8177	0.9803	0.9932
F-Measure	0.9862	0.6760	0.8913	0.7855	0.9851	0.9911
G-Means	0.9903	0.7642	0.9180	0.8496	0.9924	0.9935
AUC	0.9989	0.8697	0.9846	0.8555	0.9988	0.9995
Training time	268.35s	185.3s	299.98s	68606s	21845s	26.39s
Testing time	2199.93s	854.5s	3731.55s	590171s	224640s	222.8s

Table 6. Results of using Decorate when classifying DoS on UNSW-NB15 dataset

Table 7 compares the results of using KNN, Random Forest, Voting, Mix Stacking and Adaboost. Here:

Evaluation metrics	KNN	RF	Voting	Mix Stacking	AdaBoost
Sensitivity	0.9895	0.9876	0.9269	0.9922	0.9899
Specificity	0.9954	0.9981	0.9943	0.9983	0.9986
Precision	0.9828	0.9928	0.9773	0.9934	0.9947
F-Measure	0.9861	0.9902	0.9514	0.9928	0.9923
G-Means	0.9925	0.9928	0.9600	0.9952	0.9943
AUC	0.9992	0.9999	0.9968	0.9983	0.9996
Training time	20 ms	58.17 s	904.68 s	9894.38 s	109.99s
Testing time	723.72 s	587.7 s	8904.35 s	82937.9 s	883.17s

Table 7. Compare results of algorithms when classifying DoS on UNSW-NB15 dataset

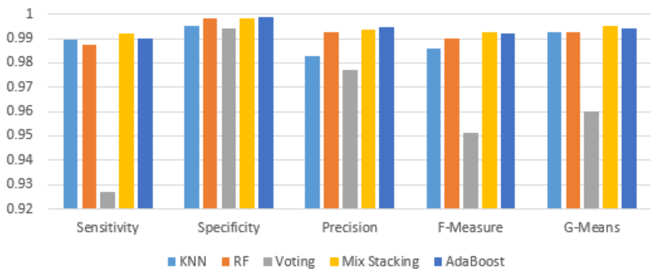

Figure 4. Comparison chart of algorithms when classifying DoS on UNSW-NB15 dataset.

(1) KNN is the single classifier for the best classification results in Table 1.

(2) Voting is a technique for building multiple models (with component classifiers using DT, NB, LR, SVM, KNN and RT) and a simple statistic based on the majority of votes used to combine predictions.

(3) Mix Stacking is a Stacking technique with heterogeneous base classifiers using DT, NB, LR, SVM, KNN and RT; The meta classifier selected for use is KNN ($k = 3$).

(4) Adaboost is a homogeneous ensemble algorithm that produces the best results among the ensemble algorithms: Bagging, Adaboost, Stacking and Decorate.

Accordingly, the Mix Stacking is selected because of the highest $F-Measure$ index (0.9928). Figure 4 compares the evaluation metrics of the Mix-Stacking algorithm with other algorithms, whereby the Mix-Stacking algorithm gives better results than the decision tree and the random forest as suggested by the other authors [3], [4].

4. Conclusions

From the results of experiments with homogeneous and heterogeneous ensemble techniques on the UNSW-NB15 dataset above, we have some comments:

(1) The ensemble classifiers give better classification quality than the case of the use of single classifiers.

(2) The use of AdaBoost algorithm with component classifiers using DT helps to improve the best classification quality compared to other ensemble algorithms when classifying DoS attack on UNSW-NB15 dataset.

(3) The training and testing time of ensemble classifiers is larger than that of single classifiers, especially when using Stacking and Decorate techniques.

(4) Decorate technique helps to improve classification quality with small training datasets such as: NSL-KDD, KDD99 [20]. However, for large datasets such as UNSW-NB15, this algorithm is not effective.

(5) The use of $F-Measure$ to evaluate classification quality improves the harmonious relationship between $Precision$ and $Recall$.

At the same time, the experimental results also set out issues that need to be further studied, especially the contents:

(1) Combined with feature reduction techniques [21], [16] to have a more effective classification system on both criteria: training time and $F-Measure$.

(2) The ability to process data as well as calculation of machine systems plays an important role in the operation of algorithms as well as machine learning methods. Since then, improve processing efficiency and access to artificial intelligence.

References

[1] Oleg Kupreev, Ekaterina Badovskaya, and Alexander Gutnikov. Ddos attacks in q4 2019. *https://securelist.com/ddos-report-q4-2019/96154/*, 2020.

[2] K. J. Singh, K. Thongam, and T. De. Entropy-based application layer ddos attack detection using artificial neural networks. *Entropy*, 18:350, 2016.

[3] Marwane Zekri, Said El Kafhali, Noureddine Aboutabit, and Youssef Saadi. Ddos attack detection using machine learning techniques in cloud computing environments. pages 1–7, 10 2017.

[4] Jiangtao Pei, Yunli Chen, and Wei Ji. A ddos attack detection method based on machine learning. *Journal of Physics: Conference Series*, 1237:032040, 06 2019.

[5] Buntine and Wray. Learning classification trees. *Statistics and Computing*, 2(2):63–73, Jun 1992.

[6] David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 12 1992.

[7] Freund, Yoav, Schapire, and Robert E. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[8] Leo Breiman. Arcing classifier (with discussion and a rejoinder by the author). *Ann. Statist.*, 26(3):801–849, 06 1998.

[9] Breiman and Leo. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[10] Leo Breiman Statistics and Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[11] B. Efron. Bootstrap methods: Another look at the jackknife. *Ann. Statist.*, 7(1):1–26, 01 1979.

[12] Tharwat and Alaa. Adaboost classifier: an overview. 02 2018.

[13] E. Frank, M. A. Hall, and I. H. Witten. *The WEKA workbench, Online Appendix for "Data Mining: Practical Machine Learning Tools and Tech-niques"*. Morgan Kaufmann, Fourth Edition, 2016.

[14] SH Kok, Azween Abdullah, NZJhanjhi, and Mahadevan Supramaniam. A review of intrusion detection system using machine learning approach. *International Journal of Engineering Research and Technology, ISBN 0974-3154*, 12(1):8–15, 2019.

[15] N. Moustafa and J. Slay. Unsw-nb15: A comprehensive dataset for network intrusion detection. In *Paper presented at the Military Communications and Information Systems Conference*, 2015.

[16] Hoang Ngoc Thanh and Tran Van Lang. An approach to reduce data dimension in building effective network intrusion detection systems. *EAI Endorsed Transactions on Context-aware Systems and Applications: Online First*, 1 2020.

[17] Powers, David, and Ailab. Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation. *J. Mach. Learn. Technol*, 2:2229–3981, 01 2011.

[18] R. P. Espíndola and N. F. F. Ebecken. On extending f-measure and g-mean metrics to multi-class problems. In *WIT Transactions on Information and Communication Technologies*, volume 35. WIT Press, 2005.

[19] Yuk Ying Chung and Noorhaniza Wahid. A hybrid network intrusion detection system using simplified swarm optimization (sso). In *Applied Soft Computing 12*, pages 3014–3022. Elsevier, 2012.

[20] Hoang Ngoc Thanh and Tran Van Lang. Creating rules for firewall use of decision tree based ensemble techniques. In *Proceedings of the 11th National Conference on Fundamental and Applied IT Research (FAIR'11)*, pages 489–496, Vietnam, 2018.

[21] Hoang Ngoc Thanh and Tran Van Lang. Feature selection based on information gain to improve performance of network intrusion detection systems. In *Proceedings of the 10th National Conference on Fundamental and Applied IT Research (FAIR'10)*, pages 823–831, Vietnam, 2017.