Journal of Computer Science and Cybernetics, V.36, N.2 (2020), 173–185
DOI 10.15625/1813-9663/36/2/14786

EVALUATING EFFECTIVENESS OF ENSEMBLE CLASSIFIERS WHEN DETECTING FUZZERS ATTACKS ON THE UNSW-NB15 DATASET

HOANG NGOC THANH[1,3], TRAN VAN LANG[2,4,*]

[1]*Lac Hong University*
[2]*Institute of Applied Mechanics and Informatics, VAST*
[3]*Information Technology Center, Ba Ria - Vung Tau University*
[4]*Graduate University of Science and Technology, VAST*

Abstract. The UNSW-NB15 dataset was created by the Australian Cyber Security Centre in 2015 by using the IXIA tool to extract normal behaviors and modern attacks, it includes normal data and 9 types of attacks with 49 features. Previous research results show that the detection of Fuzzers attacks in this dataset gives the lowest classification quality. This paper analyzes and evaluates the performance of using known ensemble techniques such as Bagging, AdaBoost, Stacking, Decorate, Random Forest and Voting to detect FUZZERS attacks on UNSW-NB15 dataset to create models. The experimental results show that the AdaBoost technique with the component classifiers using decision tree for the best classification quality with $F - Measure$ is 96.76% compared to 94.16%, which is the best result by using single classifiers and 96.36% by using the Random Forest technique.

Keywords. Machine learning; Ensemble classifier; AdaBoost; Fuzzers; UNSW-NB15 dataset.

1. INTRODUCTION

Due to recent technological advances, network-based services are increasingly playing an important role in modern society. Intruders constantly search for vulnerabilities on the computer system to gain unauthorized access to the system's kernel. 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 that is 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 anomaly behaviors, 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.

Finding a good IDS model from a certain dataset is one of the main tasks when building IDSs to correctly classify network packets as normal access or an attack. A strong classification is desirable, but it is difficult to find. In this work, we approach the ensemble method to initial training then combine these results to improve accuracy.

There are two kinds of ensembles: homogeneous ensemble and heterogeneous ensemble. A multi-classification system is based on learners of the same type, it is called a homogeneous ensemble. In

*Corresponding author.
E-mail addresses: thanhhn@bvu.edu.vn (H.N.Thanh); langtv@vast.vn (T.V.Lang).

contrast, a multi-classification system is based on learners of different types, it is called a heterogeneous ensemble.

In this paper, we use five homogeneous ensemble techniques and two heterogeneous ensemble techniques to train basic classifiers. The homogeneous ensemble techniques include Bagging, AdaBoost, Stacking, Decorate and Random Forest. The heterogeneous ensemble techniques include Stacking and Voting. The basic classifiers use machine learning techniques: Decision Trees (DT), Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), K Nearest Neighbors (KNN) and Random Tree (RT). The these ensemble models are trained and tested on the UNSW-NB15 dataset to detect Fuzzers attacks, a type of attack primarily use to find software coding errors and loopholes in networks and operating system. Fuzzers is commonly use to find for security problems in software or computer systems to discover coding errors and security loopholes in software, operating systems or networks by inputting massive amounts of random data to the system in an attempt to make it crash.

We use these ensemble methods to train multiple classifiers at the same time to solve the same problem. And then we propose a solution that combines them together to improve classification quality in IDS.

The remainder of this paper is organized as follows: Section 2 presents the ensemble machine learning methods use in the experiments; Section 3 presents the datasets, the evaluation metrics and the results obtained by using ensemble techniques when detecting Fuzzers attacks 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 ensemble classification model for greater accuracy than a single classification model. The purpose of aggregation models is to reduce variance and/or bias of algorithms. The 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 [4] introduced Bayesian techniques to reduce variance of learning methods. Wolpert's stacking method [17] aims to minimize the bias of algorithms. Freund and Schapire [8] introduced Boosting, Breiman [2] suggested ArcX4 method to reduce bias and variance, while Breiman's Bagging [1] reduced the variance of the algorithm but did not increase the bias too much. The Random Forest algorithm [3] is one of the most successful 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.

The ensemble techniques in modern machine learning field reviewed in this article include Bagging, Bootsting, Stacking, Random Forest, Decorate, and Voting. From that we have been testing to detect Fuzzers attacks on the UNSW-NB15 dataset, in order to find the optimal solution in classifying attacks.

2.1. Bootstrap

Bootstrap is a very well known method in statistics introduced by Bradley Efron in 1979 [6]. The main goal of this method is that from given dataset, it will generate m samples of identical size with replacement (called bootstrap samples). This method is mainly use to estimate standard errors, bias and calculate confidence intervals for parameters. It is implemented as follows: from an

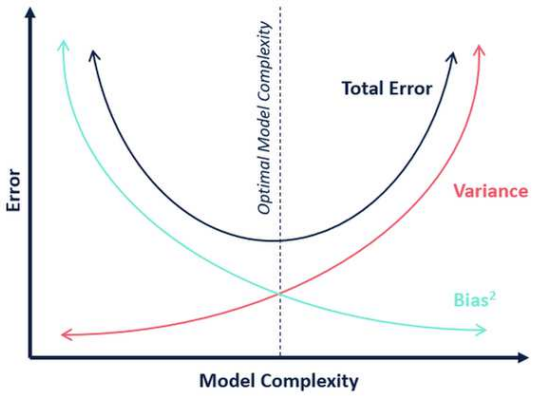

Figure 1. Illustration of the bias-variance tradeoff [12]

initial dataset, D take randomly a sample $D_1 = (x_1, x_2, ..., x_n)$ consisting of n instances to calculate the desired parameters. After that, the algorithm repeated m times to create sample D_i that also consisted of n elements from the sample D_1 by removing randomly some of its instances to add new randomly selected instances from D and calculate the expected parameters of problem.

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: A set of m datasets, each of which consists of n randomly selected elements from D with replacement (like Bootstrap). Therefore $B = (D_1, D_2, ..., D_m)$ looks like a set of cloned training sets; Train a machine or model for each set of D_i $(i = 1, 2, ..., m)$ and collect the predicted results in turn on each set of D_i.

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.

Boosting attempts to build a strong classifier from the number of weak classifiers. It is done by building a model using weak models in series. First, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

AdaBoost, short for "Adaptive Boosting", is one of the first Boosting algorithms to be adapted in practices [16]. There, the output of the weak classifiers is combined into a weighted sum that represents the final output of the boosted classifier.

2.4. Stacking

Stacking is a way to combine multiple models, introducing the concept of meta classifier. It is less widely used than Bagging and Boosting. Unlike Bagging and Boosting, Stacking can be used to combine different models.

In Stacking, training dataset is splited into two disjoint parts. The first part is used to train the base classifiers in layer 1. The second part is used to test the these base classifiers, the output of the base classifiers will be used as training data for meta classifier in layer 2 to produce the most accurate predicted results. Basically, these allow meta classifier to find the best mechanism for combining base 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 m bootstrap samples from the training dataset.

- For each bootstrap sample, an unpruned tree is constructed as follows: At each node, instead of choosing the best division among all predicted variables, a subset of the predicted variables is selected at random, then the best division of these variables is chosen.

- Make predictions by summing up the predictions of m 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.

2.7. Ensemble models for experiments

In this paper, the techniques such as homogeneous ensemble, heterogeneous ensemble, and Random Forest are used to train, test, evaluate and compare the experimental results.

With the homogeneous ensemble technique: Bagging, AdaBoost, Stacking and Decorate ensemble techniques are used on the single classifiers: J48 (DT), NaiveBayes (NB), Logistic (LR), LibSVM (SVM), IBk (KNN) and RandomTree (RT) as depicted in Figure 2. Accordingly, training and testing datasets are used to construct, evaluate and compare between the models. From there, determine which model is best suited to the Fuzzers attack.

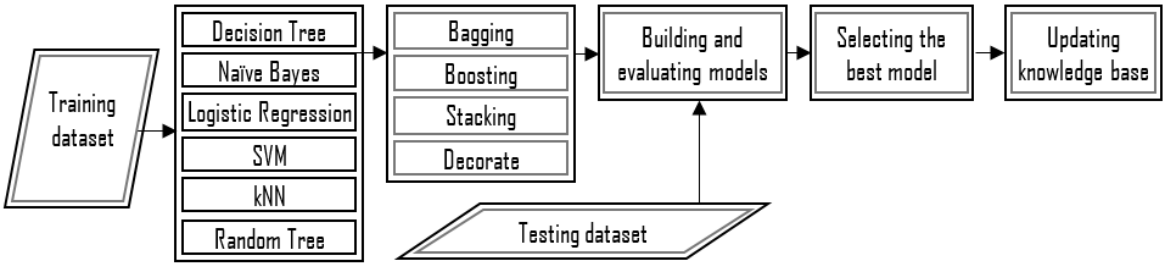

Figure 2. The Fuzzers attacks detection model using Homogeneous techniques

Similarly, with heterogeneous ensemble technique: Stacking and Voting techniques are used on the single classifiers: DT, NB, LR, SVM, KNN and RT as depicted in Figure 3 and Figure 4. Accordingly, the predicted results of the classifiers on the first stage are used as the inputs for voting or classified by the meta classifier on the second stage.

The Random Forest technique is also used to compare results with the above homogeneous and heterogeneous ensemble techniques.

Algorithm 1 Choose the best ensemble classifier using Homogeneous ensemble techniques

Input: D: a training dataset, k: k-fold, n: the number of classifiers in the ensemble, M: a set of machine learning techniques, E: a set of homogeneous ensemble techniques.
Output: the best homogeneous ensemble classifier.

```
1: begin
2:     for each: e ∈ E do
3:         for each: m ∈ M do
4:             begin
5:                 split D into k equal sized subsets D_k;
6:                 for i ← 1 to k do
7:                     begin
8:                         use the D_k as testing dataset
9:                         use the remaining (k − 1) subsets as training dataset
10:                        train ensemble using ensemble technique e and ML method m
11:                        test ensemble using dataset D_k
12:                        calculate the evaluation indexs
13:                    end
14:                calculate the average of the evaluation indexs
15:                update the best homogeneous ensemble classifier
16:            end
17:    return the best homogeneous ensemble classifier
18: end
```

Algorithm 2 Choose the best ensemble classifier using Heterogeneous ensemble techniques

Input: D: a training dataset, k: k-fold, n: the number of classifiers in the ensemble, M: a set of machine learning techniques, E: a set of heterogeneous ensemble techniques.
Output: the best heterogeneous ensemble classifier.

```
1: begin
2:     for each: e ∈ E do
3:         begin
4:             split D into k equal sized subsets D_k;
5:             for i ← 1 to k do
6:                 begin
7:                     use the D_k as testing dataset
8:                     use the remaining (k − 1) subsets as training dataset
9:                     train ensemble using n/|M| classifiers each type of ML
10:                    test ensemble using dataset D_k
11:                    calculate the evaluation indexs
12:                end
13:            calculate the average of the evaluation indexs
14:            update the best heterogeneous ensemble classifier
15:        end
16:    return the best heterogeneous ensemble classifier
17: end
```

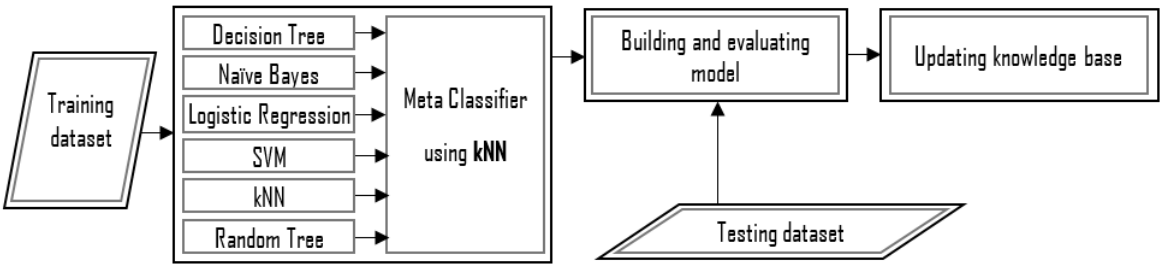

Figure 3. The Fuzzers attacks detection model using Mix Stacking technique

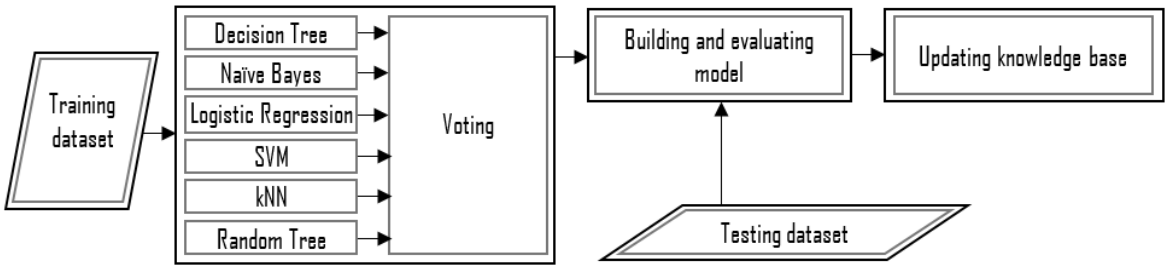

Figure 4. The Fuzzers attacks detection model using Voting technique

To solve the problem, we propose two main computational solutions that are expressed through Algorithms 1 and 2 below.

To solve the problem, we propose two computational solutions that are expressed through Algorithms 1 and 2 below. These Algorithms 1 and 2 describe in detail the choice of the best ensemble classifier using homogeneous and heterogeneous ensemble techniques. Accordingly, the training dataset is divided into 10 disjoint folds of the same size (10-folds). In the first iteration, the first fold is used as the testing dataset, and the remaining 9 folds are used as training dataset. These training and test datasets are used to train and test ensemble classifiers using homogeneous and heterogeneous ensemble techniques. In the next iteration, the second fold is used as the testing dataset, and the remaining folders are used as the training dataset, training and testing are repeated. This process is repeated 10 times. The classification results of ensemble classifiers are presented as the average of the evaluation indexs after 10 iterations, used to compare and chose the best ensemble classifier when classifying Fuzzers attack on UNSW-NB15 dataset.

3. EXPERIMENTS

The experimental computer program is implemented in the Java language with the Weka library.

3.1. Datasets

According to the statistics in [9], NSL-KDD, KDD99 and UNSW-NB15 datasets were commonly used in IDS systems.

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

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%

The UNSW-NB15 dataset contains 2,540,044 instances [10]. A part of this dataset is divided into training and testing datasets, which are used extensively in scholars' experiments. The detailed information about the datasets is presented in Table 1. In these training and testing datasets, there are normal data and a total of 9 types of attacks are as follows: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms. The UNSW-NB15 dataset was used for experiments in this paper.

3.2. Evaluation metrics

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

$$Accuracy_i = (TP_i + TN_i)/(TP_i + FP_i + TN_i + FN_i),$$
$$Sensitivity_i = TPR_i = TP_i/(TP_i + FN_i),$$
$$Specificity_i = TNR_i = TN_i/(TN_i + FP_i),$$
$$Efficiency_i = (Sensitivity_i + Specificity_i)/2,$$
$$Precision_i = TP_i/(TP_i + FP_i),$$
$$FNR_i = FN_i/(FN_i + TP_i),$$
$$FPR_i = FP_i/(FP_i + TN_i).$$

In there:

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 use of $Accuracy$ to evaluate the quality of classification has been used by many scholars. However, the class distribution in most nonlinear classification problems is very imbalanced, the use of $Accuracy$ is not really effective [13]. The more effective evaluation metrics such as $F - Measure$

and $G - Means$ are calculated as follows [7, 5]

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

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

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

AUC - ROC (Area Under The Curve- Receiver Operating Characteristics) curve is a other performance measurement for classification problem. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0 which means it has worst measure of separability. However, according to the research results of John Muschelli [11] and many other scholars, the AUC - ROC curve can lead to misleading results.

3.3. Experimental results

First, the classification results for single classifiers are shown 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 $Recall$.

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

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9372	0.9732	0.7731	0.2617	0.9480	0.9097
Specificity	0.9847	0.7253	0.9731	0.9956	0.9839	0.9778
Precision	0.9377	0.4655	0.8761	0.9355	0.9353	0.9096
F-Measure	0.9375	0.6298	0.8214	0.4090	**0.9416**	0.9097
G-Means	0.9607	0.8402	0.8674	0.5105	0.9657	0.9431
AUC	0.9737	0.8892	0.9720	0.6286	0.9934	0.9438
Training time	5565 ms	551 ms	14.66 s	565.99 s	19 ms	444 ms
Testing time	49.35 s	6615 ms	237.32 s	5246.4 s	593.9 s	5.59 s

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 refers 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 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).

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

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9496	0.9730	0.7726	0.2597	0.9487	0.9522
Specificity	0.9919	0.7246	0.9733	0.9956	0.9842	0.9920
Precision	0.9665	0.4648	0.8769	0.9358	0.9366	0.9668
F-Measure	0.9580	0.6291	0.8215	0.4066	0.9426	**0.9594**
G-Means	0.9705	0.8396	0.8672	0.5085	0.9663	0.9719
AUC	0.9975	0.8894	0.9720	0.6593	0.9956	0.9975
Training time	42992ms	8969ms	233.03s	4653.7s	2202.5s	5884ms
Testing time	368.3s	91.1s	2193s	43262.6s	23479s	51.7s

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

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9604	0.9732	0.7709	0.5032	0.9542	0.9041
Specificity	0.9939	0.7253	0.9729	0.9835	0.9852	0.9760
Precision	0.9750	0.4655	0.8751	0.8823	0.9406	0.9027
F-Measure	**0.9676**	0.6298	0.8197	0.6409	0.9473	0.9034
G-Means	0.9770	0.8402	0.8660	0.7035	0.9696	0.9394
AUC	0.9975	0.8612	0.9577	0.9470	0.9806	0.9401
Training time	119.97s	3396ms	164.67s	23894s	1529.7s	328ms
Testing time	1005.47s	29.39s	1339.2s	240178s	18634.5s	2372ms

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

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9305	0.5016	0.7762	0.2619	0.9547	0.9101
Specificity	0.9858	0.8949	0.9611	0.9956	0.9840	0.9765
Precision	0.9417	0.5397	0.8305	0.9355	0.9361	0.9049
F-Measure	0.9361	0.5199	0.8025	0.4092	**0.9453**	0.9075
G-Means	0.9578	0.6699	0.8637	0.5106	0.9692	0.9427
AUC	0.9726	0.8575	0.9632	0.6286	0.9935	0.9423
Training time	58.86 s	6388 ms	256.12 s	5358.75 s	613.48 s	5.65 s
Testing time	632.66 s	170.98 s	2215.6 s	44002.8 s	5660.7 s	243.7 s

With Decorate technique, the 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).

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

(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 (Figure 4).

(3) Mix Stacking is a Stacking technique with heterogeneous base classifiers using DT, NB, LR,

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

Evaluation metrics	DT	NB	LR	SVM	KNN	RT
Sensitivity	0.9507	0.9732	0.7731	0.2617	0.9384	0.9590
Specificity	0.9906	0.7253	0.9731	0.9956	0.9773	0.9928
Precision	0.9615	0.4655	0.8761	0.9355	0.9104	0.9704
F-Measure	0.9560	0.6298	0.8214	0.4090	0.9242	**0.9647**
G-Means	0.9705	0.8402	0.8674	0.5105	0.9576	0.9758
AUC	0.9948	0.8892	0.9720	0.6286	0.9907	0.9968
Training time	434.7 s	57.2 s	167.1 s	73850 s	17144 s	21 s
Testing time	3578 s	513.8 s	1508 s	510263 s	184541 s	192.3 s

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

Evaluation metrics	KNN	RF	Heterogeneous ensemble		AdaBoost
			Voting	Mix Stacking	
Sensitivity	0.9480	0.9554	0.9291	0.9669	0.9604
Specificity	0.9839	0.9932	0.9904	0.9918	0.9939
Precision	0.9353	0.9719	0.9596	0.9668	0.9750
F-Measure	0.9416	0.9636	0.9441	0.9668	**0.9676**
G-Means	0.9657	0.9741	0.9593	0.9793	0.9770
AUC	0.9934	0.9983	0.9968	0.9928	0.9975
Training time	19 ms	45.95 s	912.6 s	8987.55 s	119.97 s
Testing time	593.9 s	419.8 s	8844.3 s	77900.5 s	1005.47 s

SVM, KNN and RT (Figure 3); The meta classifier selected for use is KNN.

(4) Adaboost is a homogeneous ensemble algorithm that produces the best results among the ensemble algorithms: Bagging, Adaboost, Stacking and Decorate (Figure 2).

Accordingly, Adaboost technique using component classifiers as DT is selected because of the highest $F - Measure$ index (0.9676). In terms of training and testing time, AdaBoost is higher than KNN and RF, but not much.

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 use of AdaBoost technique with base classifiers using DT algorithm is to help to improve the best classification quality compared to other ensemble algorithms when classifying Fuzzers attack on UNSW-NB15 dataset.

(2) The Decorate technique helps to improve classification quality with small training datasets such as: NSL-KDD, KDD99 [15]. However, for large datasets such as UNSW-NB15, this algorithm is not effective.

(3) The use of $F - Measure$ to evaluate classification quality is to improve the harmonious relationship between $Precision$ and $Recall$. This is especially true for imbalanced datasets as in IDS.

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

(1) Combination with feature reduction techniques [14] to have a more effective classification system on both criteria: computational cost and classification quality.

(2) The ability to process data as well as the computing power of the machine system plays an important role in the operation of algorithms of machine learning methods.

REFERENCES

[1] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996. [Online]. Available: https://doi.org/10.1007/BF00058655

[2] ——, "Arcing classifier (with discussion and a rejoinder by the author)," *Ann. Statist.*, vol. 26, no. 3, pp. 801–849, 06 1998. [Online]. Available: https://doi.org/10.1214/aos/1024691079

[3] ——, "Random forests," in *Machine Learning*, 2001, pp. 5–32.

[4] Buntine and Wray, "Learning classification trees," *Statistics and Computing*, vol. 2, no. 2, pp. 63–73, Jun 1992. [Online]. Available: https://doi.org/10.1007/BF01889584

[5] Y. Y. Chung and N. Wahid, "A hybrid network intrusion detection system using simplified swarm optimization," in *Applied Soft Computing 12*. Elsevier, 2012, pp. 3014–3022.

[6] B. Efron, "Bootstrap methods: Another look at the jackknife," *Ann. Statist.*, vol. 7, no. 1, pp. 1–26, 01 1979. [Online]. Available: https://doi.org/10.1214/aos/1176344552

[7] 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*, vol. 35. WIT Press, 2005.

[8] Freund, Yoav, Schapire, and R. E., "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*, P. Vitányi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 23–37.

[9] S. Kok, A. Abdullah, NZJhanjhi, and M. Supramaniam, "A review of intrusion detection system using machine learning approach," *International Journal of Engineering Research and Technology, ISBN 0974-3154*, vol. 12, no. 1, pp. 8–15, 2019.

[10] 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.

[11] J. Muschelli, "Roc and auc with a binary predictor: a potentially misleading metric," 03 2019.

[12] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas, "A modern take on the bias-variance tradeoff in neural networks," *ArXiv*, vol. abs/1810.08591, 2018.

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

[14] H. N. Thanh and T. V. 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)*, Vietnam, 2017, pp. 823–831.

[15] ——, "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)*, Vietnam, 2018, pp. 489–496.

[16] Tharwat and Alaa, "Adaboost classifier: an overview," 02 2018.

[17] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 12 1992.

Received on January 17, 2020
Revised on April 17, 2020