Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 6 – No.7, February 2017 – www.caeaccess.org

Analysis of Classifier Ensembles for Network Intrusion Detection Systems

Neeraj Bisht
Birla Institute of Applied Sciences, Bhimtal, India

Amir Ahmad
King Abdul Aziz University, Rabigh, Saudi Arabia

A. K. Pant
Birla Institute of Applied Sciences, Bhimtal, India

ABSTRACT
In this paper an ensemble approach to classify network security data have been presented. The experiments are carried out with Decision trees and Naïve Bayes classifiers and ensemble them with methods like Bagging, Adaboost.M1, Random Forests, MultiBoosting, Rotation Forest and Random Sub Space on NSL KDD dataset which is a modified KDD anomaly detection dataset [1]. Results with different performance measures suggest that no single classification method is the best for all types of datasets on all type of performance measures. The results based on the experiments have been tabulated and their comparative performance suggests that the decision trees ensembles performed better than the Naïve Bayes ensembles. Results also suggest that single decision tree is a good classifier for this data as it has reasonable classification accuracy and less training and testing time.

Keywords
network security; NSL KDD; classifier; ensembles; Naïve Bayes; decision trees

1. INTRODUCTION
At the present time an increasing number of commercial and public services are offered trough Internet, so that security is becoming one of the key issues [2]. These "attacks" to internet service providers are carried out by exploiting unknown weakness or bugs always present in system and application software. Computer networks are usually protected against attacks by a number of access restriction policies that act as a coarse grain filter. Intrusion detection systems (IDS) are the fine grain filters placed inside the protected network, looking for known or potential threats in network traffic and/or audit data recorded by hosts. Some of the IDS detection techniques are signature based techniques [3]. In these techniques, the signatures of the known attacks are remembered. When a new pattern comes, its signature is compared with these stored signatures to predict whether the given pattern is normal or attack. This method is fast, however, it can identify only known attacks.

Researchers recently proposed intrusion detection approaches based on data mining algorithms trained on malicious and normal traffic activities [4, 5, 6, 7, 8]. They allow us to design decision "boundaries" 9 network traffic. In these methods, models are trained on the historical data and these models are used to predict the type of the new traffic activity.

Different classifiers like Decision Trees, Naïve Bayes, Neural Networks, Support Vector Machines are used to classify normal and anomalous data [4, 5, 6, 7, 8]. Ensembles are a combination of multiple base models and the final classification depends on the combined outputs of individual models [9, 10].Classifier ensembles have shown to produce better results than a single classifier provided the classifiers are accurate and diverse. Ensembles perform best when base models are unstable-classifiers whose output undergoes significant changes in generalization with small changes in the training data- decision trees and neural networks are in this class.

Several different methods have been proposed to build decision tree ensembles. Randomization is introduced to build diverse decision trees. Bagging [11] and Boosting [12] introduce randomization by manipulating the training data supplied to each classifier. MultiBoosting combines the principle of Bagging with AdaBoost [13]. Ho [14] proposed Random Subspaces which select random subsets of input features for training. Ho proposes the Random Subspace method works well when there is certain redundancy in the dataset, especially in the collections of features. He suggest that for the random subspace method to work on datasets having a small number of features, redundancy need to be introduced artificially by using simple functions of the features. Breiman [15] combines Random Subspaces technique with Bagging to create Random Forests. To build a tree, it uses a bootstrap replica of the training sample, then during the tree growing phase, at each node the optimal split is selected from a random subset of size K of candidate features. In one of the variants of Random Forests, Brieman [15] defines new features by taking the random linear combinations of the features.

During the last few decades, anomaly detection has attracted the attention of many researchers to overcome the limitations of signature-based IDSs in detecting novel attacks, and KDDCUP'99 is the most widely used data set for the evaluation of these systems. Tavallaee et al. [16] conducted a statistical analysis on this data set. They found that the dataset has many redundant data points, which makes data mining methods biased towards these data points. The authors have proposed a new data set, which consists of selected records of the complete KDD data set. This data set is publicly available for researchers through their website and has the following advantages over the original KDD data set:

1. They have removed redundant records in the training set, so the classifiers will not be biased towards more frequent records.
2. There are no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
3. The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range,

which makes it more efficient to have an accurate evaluation of different learning techniques.

4. The number of records in the train and test sets are reasonable, which makes it convenient to run the experiments on the complete set without the need to randomly select a small portion. As a result, evaluation results of different research works will be consistent and comparable.

In this paper, an approach to intrusion detection in computer networks based on multiple classifier systems is studied. This approach is motivated by the observation that generally a combination of classifiers performs better than a single classifier. This paper focuses on the use of classifier ensemble to predict intrusion detection.

In the Section 2, the classifier methods and the data sets used for the study are discussed. The section 3 has experiments and discussion. Section 4 has conclusion and future work.

2. METHODS AND MATERIAL

Decision Trees and Naïve Bayes are two popular classification methods. In this paper, authors carried out experiments with these classifiers and ensembles of these classifiers. In this section, different methods used in the paper have been discussed.

2.1 Decision Trees

Decision trees are very popular tools for classification [17, 18, 19]. The attractiveness of decision trees is due to the fact that one can represent rules with the help of decision trees. Rules can readily be expressed so that humans can understand them. A decision tree is in the form of a tree structure, where each node is either a leaf node (it indicates the value of the target class of examples) or a decision node (it specifies some test to be carried out on a single attribute-value), with two or more than two branches and each branch has a sub-tree. One can classify an example by starting from the root of the decision tree and moving through it until a leaf node is found, which provides the rules for classification of the example.

2.2 Naïve Bayes

A Naive Bayes classifier is a probabilistic classifier [20]. It is based on conditional probabilities computed by using Naïve Bayes theorem. Despite a strong independence assumption (all attributes are independent), it has shown excellent performance over a variety of datasets.

2.3 Bagging

Bagging (Bootstrap Aggregation) [11, 18] generates different bootstrap training datasets from the original training dataset and uses each of them to train one of the classifiers in the ensemble. For example, to create a training set of N data points, it selects one point from the training dataset, N times without replacement. Each point has equal probability of selection. In one training dataset, some of the points get selected more than once, whereas some of them are not selected at all. Different training datasets are created by this process. When different classifiers of the ensemble are trained on different training datasets, diverse classifiers are created. Bagging does more to reduce the variance part of the error of the base classifier than the bias part of the error.

2.4 Adaboost.M1

Boosting [12, 18] generates a sequence of classifiers with different weight distribution over the training set. In each iteration, the learning algorithm is invoked to minimize the weighted error, and it returns a hypothesis. The weighted error of this hypothesis is computed and applied to update the weight on the training examples. The final classifier is constructed by a weighted vote of the individual classifiers. Each classifier is weighted according to its accuracy on the weighted training set that it has trained on. The key idea, behind Boosting is to concentrate on data points that are hard to classify by increasing their weights so that the probability of their selection in the next round is increased. In subsequent iteration, therefore, Boosting tries to solve more difficult learning problems. Boosting reduces both bias and variance parts of the error. As it concentrates on hard to classify data points, this leads to the decrease in the bias. At the same time classifiers are trained on different training data sets so it helps in reducing the variance. Boosting has difficulty in learning when the dataset is noisy. In each iteration, the weights assigned to the noisy data points increases so in subsequent iteration it concentrates more on the noisy data points; it leads to over fitting of the data.

2.5 Random Forests

Random Forests are very popular decision tree ensembles [15]. It combines Bagging with Random Subspace. For each decision tree, a dataset is created by bagging procedure. During the tree growing phase, at each node, k attributes are selected randomly and the node is split by the best attribute from these k attributes. Breiman showed that Random Forests are quite competitive to Adaboost. However, Random Forests can handle mislabeled data points better than Adaboost can. The Random Forests are widely used due to their robustness.

2.6 MultiBoosting

MultiBoosting is an extension to the highly successful daboost technique for forming decision committees. MultiBoosting can be viewed as combining Adaboost with wagging. It is able to harness both Adaboost's high bias and variance reduction with wagging's superior variance reduction. It offers the further advantage over AdaBoost of suiting parallel execution [13].

2.7 Random subspace

Random subspace method (or attribute bagging) is an ensemble classifier that consists of several classifiers and outputs the class based on the outputs of these individual classifiers. Random subspace method is a generalization of the random forest algorithm. Whereas random forests are composed of decision trees, a random subspace classifier can be composed from any underlying classifiers. Random subspace method has been used for linear classifiers, support vector machines, nearest neighbors and other types of classifiers. This method is also applicable to one-class classifiers.

2.8 Rotation Forest

Rotation Forest is a recently proposed method for building classifier ensembles using independently trained decision trees. Rotation Forest is an ensemble method which trains L decision trees independently, using a different set of extracted features for each tree. It was found to be more accurate than Bagging, AdaBoost and Random Forest ensembles across a collection of benchmark data sets [21].

2.9 Dataset

In this paper, the modified KDD anomaly detection datasets have been used. Tavallaee et. al.[16] presented one modified

KDD training datasets and two testing datasets .Here, these are named viz training dataset, type1 testing dataset and type 2 testing dataset. The details of these datasets are presented below.

2.9.1 Training Dataset

This is obtained by removing all the redundant records from the original KDD Cup'99 training dataset. This new training dataset has been used for training the classifiers.

2.9.2 Type 1 Testing Dataset

This dataset is created by removing all the redundant records from the testing dataset, after that 21 classifiers are used to divide the testing dataset and into 5 groups on the basis of prediction difficulty. A new testing dataset is created by selecting data points from each group such that the number of data points selected from each group were inversely proportional to the number of data points in that group.

2.9.3 Type 2 testing dataset

Any data point which is correctly classified by all 21 classifiers is not included in this dataset. This testing dataset was expected to be the most difficult dataset.

3. EXPERIMENTS AND DISCUSSION

All the experiments were carried out by using WEKA software [22]. The authors performed the experiments with Bagging, Adaboost.M1, Random Forests, MultiBoostAB, Rotation Forest and Random Sub Space modules. For the ensembles other than Random Forest , experiments are carried out with J48 tree (the implementation of C4.5 tree) and Naïve Bayes classifier as the base classifier. As the training dataset was large, the size of the ensembles was set to 10. All the other default parameters were used in the experiments. We also carried out experiment with single J48 tree and single Naïve Bayes classifier. The following performance measures have been used to compare different classifiers.

3.1 Performance measures

Various parameters to evaluate the performances of various classification techniques have been defined.

(a) $\text{Sensitivity / Recall(r)} = \frac{TP}{TP+FN} \times 100$

(b) $\text{Specificity} = \frac{TN}{FP+TN} \times 100$

(c) $\text{Accuracy} = \frac{TP+TN}{n} \times 100$ where, n= Total number of data points.

(d) $\text{Precision(p)} = \frac{TP}{TP+FP} \times 100$

(e) F-Measure is given by,

$$F(r,p) = \frac{2rp}{r+p},$$

(a) TP is the number of true positive (Attack is predicted correctly).
(b) TN is the number of true negative (Normal is predicted correctly).
(c) FP is the number of false negative (Normal is predicted as Attack).
(d) FN is the number of false negative(Attack is predicted as Normal).

The high sensitivity is most desirable as we do not want any attacks go unnoticed.

3.2 Results and discussion

We carried out testing with three types of testing datasets discussed in the last section.

3.2.1 Training data as the testing data

The experiments have been carried out with the training dataset as the testing datasets. Results are presented in Table 1 and Table 2. As expected, all the classifier performed well for this testing dataset. However, it is clear from Figure 1 and Figure 2 that the performance of ensembles based on decision trees are better than ensembles based on naïve Bayes classifiers. AdaBoost.M1 with decision trees, MultiBoostAB with decision tree and Random Forests are the best classifier for this problem. Our results show that Adaboost.M1 is the best method among all ensembles based on naïve Bayes classifier. Hence, AdaBoost.M1 is the best ensemble method among the ensemble methods studied. It was expected because AdaBoost.M1 creates the classifiers such that the training error is minimum. In this case the testing error is same as the training error, hence, the testing error is minimum.

The other interesting point is the performance of single decision tree which is almost similar to the performance of the classifier ensembles. The training of ensembles takes a lot of time and huge memory is required to store the trained classifiers. Hence, only a single classifier is used if the training dataset and testing dataset are expected to be similar.

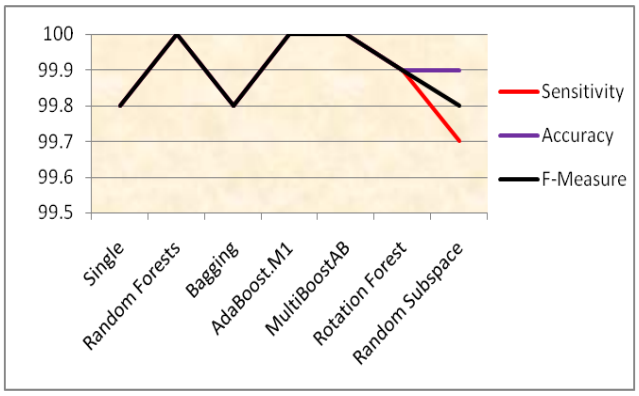

Fig.1 Classification results for training data as testing data for classifiers based on decision tree. Results are presented in % classification accuracy

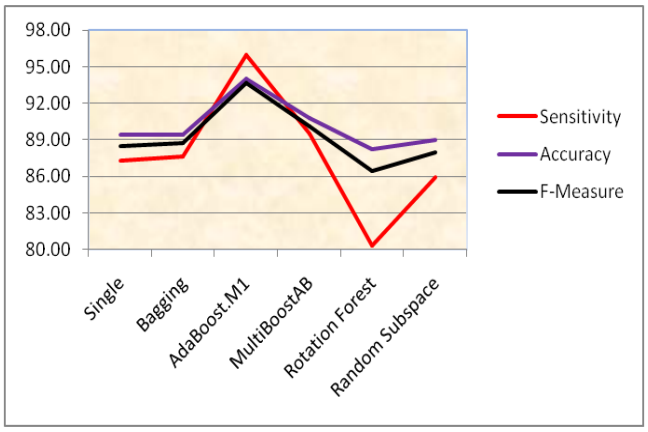

Fig.2 Classification results for training data as testing data for classifiers based on naïve Bayes. Results are presented in % classification accuracy

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 6 – No.7, February 2017 – www.caeaccess.org

3.2.2 Type 1 testing data

Results for these experiments are presented in Table 3 and Table 4 as well as in Figure 4 and Figure 5. The performance of all classifiers is not as good as in the previous case (the training dataset as the testing dataset).

It is clear in Figure 3 that sensitivity has decreased as some of the attacks in the testing datasets are not exactly the same as in the training dataset so the classifiers have difficulty in predicting these attacks. Comparison of the Table 3 and Table 4 shows that decision trees ensembles perform better than Naïve Bayes ensembles. The accuracy of the ensembles created by using the AdaBoost.M1 is less than the accuracy of single classifier. As AdaBoost.M1 has problem in learning the mislabeled training data points, the low accuracy of these ensembles suggests that the training dataset has mislabeled data points. Interestingly, single decision tree has the best accuracy and sensitivity. However, the AUC under ROC is best for Rotation Forests. This suggests that no single classifier is best for all the performance measures. In other words, one has to decide about the performance measure to select the classifier.

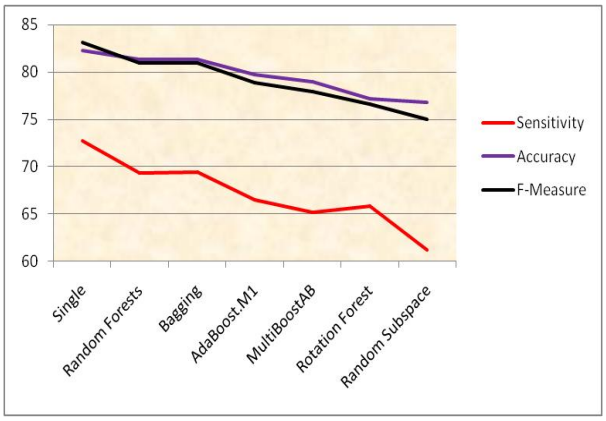

Fig.3 Classification results for type 1 testing data for classifiers based on decision trees. Results are presented in % classification accuracy

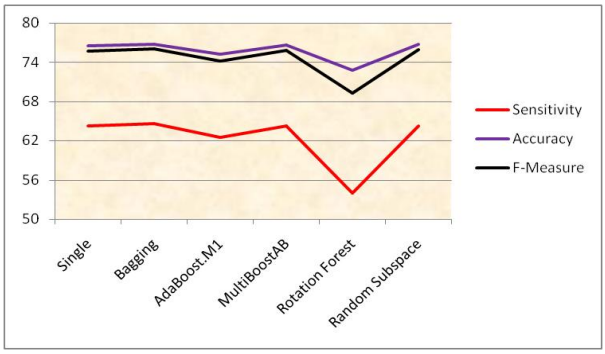

Fig.4 Classification results for type 1 testing data for classifiers based on naïve Bayes. Results are presented in % classification accuracy

3.2.3 Type 2 testing data

Results are presented in Table 5 and Table 6. As discussed in the last section, this is the most difficult testing dataset to predict. The performances of all classifiers are quite poor as compared to the other two cases (Figure 5 and Figure 6). This is due to the fact that the level of difficulty (for prediction) is more for this dataset. In this case also, single decision tree has the best accuracy and sensitivity whereas, MultiBoostAB method has the best AUC under ROC. Classifiers based on decision trees performed better than classifiers based on Naïve Bayes classifiers.

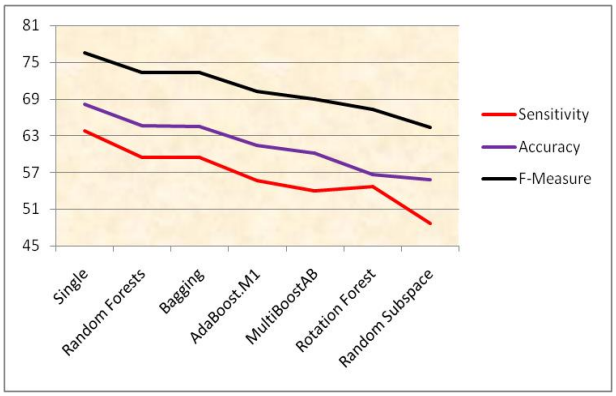

Fig.5 Classification results for type 2 testing data for classifiers based on decision tree. Results are presented in % classification accuracy

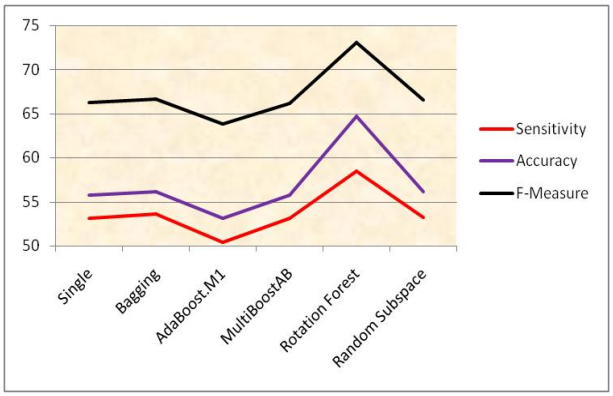

Fig.6 Classification results for type 2 testing data for classifiers based on naïve Bayes. Results are presented in % classification accuracy

3.3 Summary of the Results

We observed following points from our experiments:

1. No classifier (among the classifiers we studied) is best for all the performance measure. Hence, one has to decide the performance measure carefully in order to compare different classifiers.
2. Decision tree ensembles perform better than Naive Bayes ensembles. Hence, decision tree ensembles should be preferred.
3. Performance of Random Sub Space ensemble is poor in all the cases because the dataset does not contain redundant data points.
4. Performance of Random Forest is better than Adaboost.M1 as it handles mislabeled data points more efficiently.
5. In real life, most of the attacks are similar but not exactly same. In these cases (type 1 and type 2 testing datasets), Random Forests and Bagging performed best. Hence, Random Forests and Bagging are the better choices as compared to AdaBoost.M1. However Rotation Forest performed

best for type 2 test data with Naïve Bayes ensembles.

6. The network security datasets are quite large. The training of these ensembles on these datasets takes a lot of time, whereas the storage of these ensembles take a lot of space. As we observed that the performance of single decision tree is quite comparable with decision trees ensembles, one may use the single decision tree if the performance requirements are not very strict (the best performance).

4. CONCLUSION AND FUTURE WORK

KDD 98 data has redundant data points. A new dataset is presented to overcome this weakness. In this paper, a detailed study of this dataset is carried out by using different classifier ensembles. The study suggests that decision trees ensembles performed better than Naïve Bayes ensembles. It is also clear from the experiments that Random Subspace ensemble dose not perform well for non redundant data sets. Even the performance of single decision tree is quite competitive. It is suggested that a single decision tree is a useful classifier for network security data. In future, the other ensemble methods and other classifiers like support vector machines [16] can be used for our study.

Table 1- Classification results for training data as testing data for classifiers based on decision tree. Results are presented in % classification accuracy

Classifier	Sensitivity	Specificity	Accuracy	Precision	F-Measure
Single	99.8	99.8	99.8	99.8	99.8
Random Forests	100	100	100	100	100
Bagging	99.8	99.9	99.8	99.9	99.8
AdaBoost.M1	100	100	100	100	100
MultiBoostAB	100	100	100	100	100
Rotation Forest	99.9	100	99.9	100	99.9
Random Subspace	99.7	100	99.9	99.9	99.8

Table 2- Classification results for training data as testing data for classifiers based on naïve Bayes. Results are presented in % classification accuracy

Classifier	Sensitivity	Specificity	Accuracy	Precision	F-Measure
Single	87.3	91.3	89.43	89.8	88.5
Bagging	87.6	91	89.39	89.5	88.7
AdaBoost.M1	96	92.3	94	91.5	93.7
MultiBoostAB	89.5	91.9	90.8	90.6	90.1
Rotation Forest	80.3	95.1	88.2	93.5	86.4
Random Subspace	85.9	91.8	89	90.2	88

Table 3- Classification results for type 1 testing data for classifiers based on decision trees. Results are presented in % classification accuracy

Classifier	Sensitivity	Specificity	Accuracy	Precision	F-Measure
Single	72.7	97.2	82.3	97.1	83.1
Random Forests	69.3	97.2	81.3	97	80.9
Bagging	69.4	97.2	81.3	97	80.9
AdaBoost.M1	66.5	97.1	79.7	96.8	78.9

MultiBoostAB	65.2	97.2	79	96.8	77.9
Rotation Forest	65.8	92.3	77.2	91.8	76.6
Random Subspace	61.2	97.4	76.8	96.9	75

Table 4- Classification results for type 1 testing data for classifiers based on naïve Bayes. Results are presented in % classification accuracy

Classifier	Sensitivity	Specificity	Accuracy	Precision	F-Measure
Single	64.3	92.8	76.56	92.2	75.7
Bagging	64.6	92.7	76.71	92.1	76
AdaBoost.M1	62.5	92	75.22	91.2	74.2
MultiBoostAB	64.3	92.9	76.6	92.2	75.8
Rotation Forest	54	97.6	72.8	96.7	69.3
Random Subspace	64.3	93.2	76.8	92.6	75.9

Table 5- Classification results for type 2 testing data for classifiers based on decision tree. Results are presented in % classification accuracy

Classifier	Sensitivity	Specificity	Accuracy	Precision	F-Measure
Single	63.8	87.3	68.1	95.8	76.6
Random Forests	59.4	87.1	64.6	95.7	73.3
Bagging	59.4	87.4	64.5	95.5	73.3
AdaBoost.M1	55.7	87	61.4	95.1	70.2
MultiBoostAB	54	87.4	60.1	95.1	68.9
Rotation Forest	54.7	65.1	56.6	87.6	67.3
Random Subspace	48.6	88.2	55.8	94.9	64.3

Table 6- Classification results for type 2 testing data for classifiers based on naïve Bayes. Results are presented in % classification accuracy

Classifier	Sensitivity	Specificity	Accuracy	Precision	F-Measure
Single	53.1	67.8	55.77	88.2	66.3
Bagging	53.6	67.4	56.1	88.1	66.6
AdaBoost.M1	50.4	65.1	53.08	86.7	63.8
MultiBoostAB	53.1	67.9	55.7	88.2	66.2
Rotation Forest	58.5	92.9	64.7	97.4	73.1
Random Subspace	53.2	69.5	56.1	88.7	66.5

5. REFERENCES

[1] KDD Cup'99 Data. http://www.sigkdd.org. Accessed 15 July 2012

[2] Pfleeger CP (1997) Security in Computing. Prentice Hall, Upper Saddle River, NJ

[3] Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J (May 2003) A comparative study of anomaly detection schemes in network intrusion detection. In Proceedings of the SIAM International Conference on Data Mining, San Francisco

[4] Amor NB, Benferhat S, Elouedi Z (2004) Naïve Bayes vs. Decision Trees in Intrusion Detection Systems. In Proceedings of ACM Symposium on Applied Computing, Nicosia, Cyprus

[5] Gaddam SR, Phoha VV, Balagani KS (2007) Means+id3 a novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods. IEEE Trans Knowl and Data Engg 19(3):345-354

[6] Horng SJ, Su MY, Chen YH, Kao TW, Chen RJ, Lai JL, Perkasa CD (2011) A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Syst with Appl 38(1):306-313

[7] Sabhnani M, Serpen G (2003) Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context. In Proceedings of Conference on Machine Learning Models, Technology and Application (pp. 209-215), MLMTA

[8] Tajbakhsh A, Rahmati M, Mirzaei A (2009) Intrusion detection using fuzzy association rules. Appl Soft Comput 9(2):462-469

[9] Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Patt Anal Mach Intel 12: 993-1001

[10] Kuncheva LI (2004) Combining pattern classifiers: Methods and Algorithms. Wiley-IEEE Press, New York

[11] Breiman L (1996) Bagging predictors. Machine Learning 24(2):123–140

[12] Freund Y, Schapire RE (1997) A decision theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55:119-139

[13] Webb GI (2000) MultiBoosting: A Technique for Combining Boosting and Wagging. Machine Learning 40(2):159–196

[14] Ho TK (1998) The Random Subspace Method for Constructing Decision Forests. IEEE Trans Patt Anal Mach Intell 20(8):832–844

[15] Breiman L (2001) Random Forests. Machine Learning 45(1):5–32

[16] Tavallaee ME, Bagheri WL, Ghorbani A (2009) A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA) (pp. 53-58), Piscataway, NJ, USA

[17] Breiman L, Friedman JH, Olshen R, Stone C (1984) Classification and Regression Trees. Chapman and Hall, London

[18] Halawani SM, Alhaddad M, Ahmad A (2012) A study of digital mammograms by using clustering algorithms. J Sci Ind Res 71:594-600

[19] Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann, San Mateo

[20] Bishop CM (2006) Pattern Recognition and Machine Learning. Springer-Verlag, New York

[21] Kuncheva LI, Rodriguez JJ (2007) An Experimental Study on Rotation Forest Ensembles. In Proceedings of 7th International Workshop on Multiple Classifier Systems, MCS'07 (pp. 459-468), Prague, Czech Republic, LNCS 4472

[22] Witten IH, Frank E (2000) Data Mining: Practical Machine Learning Tools with Java Implementations. Morgan Kaufmann, San Francisco