Virginia Journal of Science
Volume 68, Issue 3 & 4
Fall & Winter 2017
doi: 10.25778/PEXS-2309

A Comparative Study on Machine Learning Algorithms for Network Defense

Abdinur Ali, Yen-Hung Hu, Chung-Chu (George) Hsieh, Mushtaq Khan
Norfolk State University, Center of Excellence in Cyber Security
700 Park Avenue, Norfolk, VA, 23504, USA
{amali, yhu, ghsieh, mkhan}@nsu.edu

ABSTRACT

Network security specialists use machine learning algorithms to detect computer network attacks and prevent unauthorized access to their networks. Traditionally, signature and anomaly detection techniques have been used for network defense. However, detection techniques must adapt to keep pace with continuously changing security attacks. Therefore, machine learning algorithms always learn from experience and are appropriate tools for this adaptation. In this paper, ten machine learning algorithms were trained with the KDD99 dataset with labels, then they were tested with different dataset without labels. The researchers investigate the speed and the efficiency of these machine learning algorithms in terms of several selected benchmarks such as time to build models, kappa statistic, root mean squared error, accuracy by attack class, and percentage of correctly classified instances of the classifier algorithms.

Keywords: Machine learning, KDD dataset, network defense

INTRODUCTION

The number of Internet security threats are growing astronomically. In 2016, the report (Symantec, 2016) revealed that, more than 430 million new malwares were discovered in 2015, up 36 percent more from the year before; a new zero day vulnerability was discovered on average each week in 2015, up 125 percent from the year before; more than 429 million personal identities were stolen in 2015, up 23 percent from the year before; over one million web attacks each and every day in 2015; spear-phishing attacks increased 55 percent in 2015; ransomware increased 35 percent in 2015; and more than 100 million fake technical support scams were blocked in 2015. Meanwhile, in 2016, the report (FBI, 2016) from the FBI disclosed that approximately 300 thousand cybercrime complaints were filed and the total loss of those victims was about 1.33 billion dollars. These data demonstrate that it is crucial and very challenging to protect end users from security threats and cybercrimes while they are connecting to the Internet via private or public networks.

Many approaches have been developed in the past few years for defending networks against security threats. Such technologies include IP traceback (Gong & Sarac, 2008) (Murugesan, Shalinie, & Neethimani, 2014) (Song & Perrig, 2001), IP traffic classification (Crotti, Gringoli, Pelosato, & Salgarelli, 2006) (Nguyen & Armitage, 2008) (Callado, et al., 2009), intrusion detection (Zhou & Lang, 2003) (Dharmapurikar & John W. Lockwood, 2006)

(Chen & Leneutre, 2009) (Das, Nguyen, Zambreno, Memik, & Choudhary, 2008) (Hu, Hu, & Maybank, 2008) (Mabu, Chen, Lu, Shimada, & Hirasawa, 2011) having been widely discussed and implemented. Their performance, advantages and disadvantages have been thoroughly investigated as well. In addition, cybersecurity issues caused by human factors are also addressed and discussed (Hadlington, 2017). Recently, machine learning technology has been used to detect network threats and malicious executables (Kolter & Maloof, 2006) (Siddiqui, Khan, Ferens, & Kinsner, 2016). Machine learning gives computers the ability to learn without being explicitly programmed (Samuel, 1959). In 1997, Tom Mitchell (Tom Mitchell, 1997) gave the definition: "*A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.*" Furthermore, machine learning is a data analysis approach that builds automatic models for processing, predicting data, and decision making (Mannila, 1996) (Chandiok & Chaturvedi, 2015).

We focus on applying machine learning algorithms for network defense. However, the efficiencies of those algorithms vary quite a bit. In this paper, we investigate various machine learning algorithms. Our goal is to help researchers to select the appropriate algorithm based on speed and efficiency. In this research, we have used the KDD99 dataset (KDD Cup 1999 Data, n.d.) and the Weka software (Weka 3: Data Mining Software in Java, n.d.). A number of authors have pointed out some drawbacks of the KDD99 dataset (A. Olusola, S. Oladele, & O. Abosede, 2010) (Tavallaee, Bagheri, Lu, & Ghorbani, 2009) (McHugh, 2000) (Kayacik & Zincir-Heywood, 2005). Each of the proposals has its merits and demerits. However, we have used the original KDD99 dataset for training and testing the classifier algorithms. The focus of our research is benchmarking the machine learning algorithms used for network defense.

The remainder of this paper is organized as follows: Section 2 reviews related literature. Section 3 introduces research materials and methods. Section 4 discusses our research results. Section 5 concludes this paper and points out future work. Section 6 expresses our thanks to the research sponsors.

LITERATURE REVIEW

Several articles have studied and investigated machine learning algorithms. In (Solanki & Dhamdhere, 2015), the authors compared the performance of the support vector machines (SVM) and the C 4.5 algorithm. Accuracy of the two algorithms was tested for four different computer network attacks. The findings indicate that C 4.5 outperforms SVM. This finding agrees with the result we obtained with the J48 tree which is a Weka implementation of the C 4.5 algorithm.

In (Nguyen & Choi, 2008), the authors investigated a set of classifier algorithms for denial of service (DoS), remote to local (R2L), user to root (U2R), and surveillance (PROBE) attacks. The authors tried to determine the best algorithms for each attack category. The results show that Naïve Bayes, Bayes Net, One-R are the best algorithms for PROBE, U2R, and R2L attacks, respectively. Most of the algorithms were reported to have significant performance for DoS attack category.

Hidden Markov models are usually used for speech recognition (Rabiner, 1989). However, Gao, Ma and Yang have used hidden Markov models to model the normal and abnormal behavior of computer networks to detect network intrusions (Gao, Ma, & Yang, 2002). Comparing detection anomalies of different experiments, the highest accuracy achieved was 63.2% detection. Some other authors (Zhou & Lang, 2003) have used Fourier series methods for network intrusion detection. They used such methods to extract network traffic periodic patterns from raw network traffic and classify those periodic patterns as normal or abnormal traffic behavior. Gomez and Dasgupta (Gomez & Dasgupta, 2001) have proposed the use of fuzzy logic for network intrusion detection. The authors claim that their results are comparable to 95.4% detection rate.

Unlike most researchers who focused on a single audit of the data, Ye et. al. (Ye, Li, Chen, Emran, & Xu, 2001) introduced a multiple audit method based on frequency properties of the traffic data. However, the data they used for testing was "pure data" which was not noisy network data and did not reflect real case scenario of network traffic. In addition, Goonatilake, et. al. (Goonatilake, Herath, Herath, Herath, & Herath, 2007) used Chi-square goodness of fit tests for network intrusion detection. They used the test to determine abnormal network traffic activities which deviate from normal traffic patterns. Related research shows that all the existing algorithms found in the literature have challenges to determine and estimate the optimal parameters which will give us the highest detection rates within the fastest time. However, the goal for network security is still better profiling, better classification and better prediction (Wang, 2008).

In (Dao & Vemuri, 2002), the authors compared and analyzed the performance measures for five neural networks. The result of the study indicates quasi-Newton and conjugate gradient descent produced better detection rates. Our contribution is to use rigorous statistical analysis such as tests of normality, hypothesis testing, and nonparametric tests of statistical significance to compare the differences in true positive rates for ten classifier algorithms and propose benchmarks for their performance measures.

MATERIALS AND METHODS

In this simulation study, we use multiple learning algorithms and test benchmarks for each of them. The computer we selected is DELL PRECISION T3610. It equips with a 64-bit Windows 7 Pro Operating System and an Intel Xeon CPU E5-1607 V2 @ 3.00 GHZ processor with 16 gigabytes memory.

In this investigation, we have used twenty three sub-attacks. We kept the number of attacks the same for each classifier algorithm. We used ten different machine learning algorithms. Then we compared and contrasted the results of the classifiers. We have used a sample of classifiers instead of the whole population of classifiers to reduce the computational complexity. Whether this sample is representative sample or not we leave it to others.

Nevertheless, we made careful observations of the variables within the selected sample of classifiers. The ten machine learning algorithms used in this research are J48 Tree (Patil & Sherekar, 2013), Naïve Bayes (Patil & Sherekar, 2013) (Wu, et al., 2008) (Ashari, Paryudi, &

Tjoa, 2013), Random Forest (Ho, 1995) (Dietterich, 2000), Support Vector Machines (Wu, et al., 2008) (Collobert & Bengio, 2004), Multilayer Perceptron (Collobert & Bengio, 2004), Radial Basis Function (Que & Belkin, 2016), Bayes Net (Heckerman, Geiger, & Chickering, 1994), Bagging (Dietterich, 2000), AdaBoostM1 (Wu, et al., 2008), and Stacking (Sikora & Al-laymoun, 2014).

RESULTS AND DISCUSSION

In this study, the total number of instances used for testing each algorithm is 494020. The ratio of number of correctly classified instances and the total number of instances used was converted into a percentage of correctly classified instances. Random Forest was found to have the highest percentage of correctly classified instances and lowest value of root mean squared error. Therefore, it is more precise than others. Furthermore, Random Forest has the highest Kappa Statistic. This result supports that this classifier has highest agreement between the actual and predicted values. The results are given in Tables 1 - 3 and in Figures 1 - 2.

We have investigated the performance of ten classifier algorithms. We used twenty three sub-attacks and determined the true positive (TP) rates, false positive (FP) rates, precision, recall, F-measure, and the receiver operating characteristic (ROC) area of each algorithm. The twenty three sub-attacks can be reduced into four major categories; namely, denial of service (DoS) attack, remote to local attack (R2L), user to root attack (U2R), surveillance (PROBE). However, instead of reduction, we decided to retain the twenty three sub-attacks and find their accuracies using the ten classifier algorithms. The results of the accuracy by class are given in Tables 4 - 13.

In this simulation, we compared and contrasted the true positives of all ten classifier algorithms. However, before we compared the true positive rates of machine algorithms, tests of normality were performed. As shown in Table 14, Kolmogorov-Smirnov and Shapiro-Wilk tests were used to test the assumption of normality. Lilliefors significance correction was used for Kolmogorov-Smirnov test. The results indicate that the model violates the assumption of normality. Therefore, parametric statistics cannot be used to analyze this data. We decided to use non-parametric statistics. The results for the normality tests are given in Table 14.

When we selected non-parametric statistics, the Kruskal-Wallis test was used to compare the differences in true positive rates of the machine learning algorithms. We input identical twenty three sub-attacks into ten different classifiers. The findings show that the differences in true positive rates for the ten classifiers are statistically significant. The result for the overall model is significant and is given in Table 15.

The pairwise comparison logically follows if the overall model is statistically significant. Only significant pairwise comparisons are given in Tables 16 & 17. The findings indicate that J48 Tree outperforms AdaBoostM1 and Stacking. Naïve Bayes outperforms Radial Basis Function, AdaboostM1 and Stacking. Random Forest outperforms Multilayer Perceptron, Radial Basis Function, AdaBoostM1 and Stacking. Multilayer Perceptron outperforms AdaboostM1 and Stacking. Support Vector Machines outperform AdaBoostM1 and Stacking. Radial Basis Function outperforms Bagging, AdaBoostM1 and Stacking. Bayes Net outperforms AdaBoostM1 and Stacking. Bagging outperforms AdaBoostM1 and Stacking.

CONCLUSION

A fundamental goal of network security is to use statistical models with the highest true positive rates. In general, network security engineers have to choose the most appropriate machine learning algorithms to prevent abnormal uses of their computer systems. However, they are faced with many competing parameters such as speed and the efficiency of the selected algorithm.

We used the KDD99 dataset as a benchmark for testing ten classifier algorithms. We found Stacking algorithm is the fastest for the time to build models. Additionally, for the correctly classified instances, Random Forest is the winner. Random Forest has the highest true positive rates. However, we infer based on our simulation results that AdaBoostM1 and the Stacking classifiers have the lowest true positives rates out of the ten classifiers. With regard to the model performance measures, our data supports that J48 Tree, Naïve Bayes, Random Forest and Support Vector Machines are the top four best performers in true positive rates.

ACKNOWLEDGEMENT

This material is based on research sponsored by the Office of the Assistant Secretary of Defense for Research and Engineering (OASD(R&E)) under agreement number FAB750-15-2-0120. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Office of the Assistant Secretary of Defense for Research and Engineering (OASD(R&E)) or the U.S. Government. We thank the editor and the reviewers for constructive suggestions for improvement of the paper.

REFERENCES

A. Olusola, A., S. Oladele, A., & O. Abosede, D. (2010). Analysis of KDD '99 Intrusion Detection Dataset for Selection of Relevance Features. *Proceedings of the World Congress on Engineering and Computer Science (WCECS), 1*, pp. 20-22. San Francisco, USA.

Ashari, A., Paryudi, I., & Tjoa, A. M. (2013). Performance Comparison Between Naive Bayes, Decision Tree and K-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool. *International Journal of Advanced Computer Science and Applications ((IJACSA)), 4*(11), 33-39.

Callado, A., Kamienski, C., Szabó, G., Ger˝o, B. P., Kelner, J., Fernandes, S., & Sadok, D. (2009). A Survey on Internet Traffic Identification. *IEEE Communications Surveys & Tutorials, 11*(3), 37-52.

Chandiok, A., & Chaturvedi, D. K. (2015). Machine Learning Techniques for Cognitive Decision Making. *Proceedings of 2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI),* (pp. 1-6). Kanpur, India.

Chen, L., & Leneutre, J. (2009). A Game Theoretical Framework on Intrusion Detection in Heterogeneous Networks. *IEEE Transactions On Information Forensics And Security, 4*(2), 165-178.

Collobert, R., & Bengio, S. (2004). Links between Perceptrons, MLPs and SVMs. *Proceedings of the 21st International Conference on Machine Learnin.* Banff, Canada.

Crotti, M., Gringoli, F., Pelosato, P., & Salgarelli, L. (2006). A Statistical Approach to IP-level Classification of Network Traffic. *Proceedings of 2006 IEEE International Conference on Communications (ICC 2006)*, (pp. 170-176). Istanbul, Turkey.

Dao, V. N., & Vemuri, V. R. (2002). Computer Network Intrusion Detection: A Comparison of Neural Network Methods. *Differential Equations and Dynamical Systems, 10*(1&2), 201-214.

Das, A., Nguyen, D., Zambreno, J., Memik, G., & Choudhary, A. (2008). An FPGA-Based Network Intrusion Detection Architecture. *IEEE Transactions On Information Forensics And Security, 3*(1), 118-132.

Dharmapurikar, S., & John W. Lockwood. (2006). Fast and Scalable Pattern Matching for Network Intrusion Detection Systems. *IEEE Journal On Selected Areas In Communications, 24*(10), 1781-1792.

Dietterich, T. (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 139-157.

FBI. (2016). *2016 Internet Crime Report.* Retrieved from https://pdf.ic3.gov/2016_IC3Report.pdf

Gao, B., Ma, H.-Y., & Yang, Y.-H. (2002). HMMs (Hidden Markov models) based on anomaly intrusion detection method. *Proceedings of 2002 International Conference on Machine Learning and Cybernetics.* Beijing, China.

Gomez, J., & Dasgupta, D. (2001). Evolving Fuzzy Classifiers for Intrusion Detection. *Proceedings of the 2002 IEEE Workshop on Information Assurance.* West Point, New York, USA.

Gong, C., & Sarac, K. (2008). A More Practical Approach for Single-Packet IP Traceback using Packet Logging and Marking. *IEEE Transactions on Parallel and Distributed Systems, 19*(10), 1310-1324.

Goonatilake, R., Herath, A., Herath, S., Herath, S., & Herath, J. (2007). Intrusion Detection Using the Chi-Square Goodness-of-Fit Test for Information Assurance, Network, Forensics and Software Security. *Journal of Computing Sciences in Colleges - Papers of*

the Fourteenth Annual CCSC Midwestern Conference and Papers of the Sixteenth Annual CCSC Rocky Mountain Conference, 255-263.

Hadlington, L. (2017). Human Factors in Cybersecurity; Examining the Link Between Internet Addition, Impulsivity, Attitudes Towards Cybersecurity, and Risk Cybersecurity Behaviours. *Heliyon, 3*(7).

Heckerman, D., Geiger, D., & Chickering, D. M. (1994). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Proceedings of AAAI-94 Workshop on Knowledge Discovery in Databases*, (pp. 85-96).

Ho, T. K. (1995). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, (pp. 278-282). Montreal, Quebec, Canada.

Hu, W., Hu, W., & Maybank, S. (2008). AdaBoost-Based Algorithm for Network Intrusion Detection. *IEEE Transactions On Systems, Man, And Cybernetics - Part B: Cybernetics, 38*(2), 577-583.

Kayacik, H. G., & Zincir-Heywood, N. (2005). Analysis of three intrusion detection system benchmark datasets using machine learning algorithms. *Proceedings of the 2005 IEEE international conference on Intelligence and Security Informatics (ISI'05)*, (pp. 362-367). Atlanta, Georgia, USA.

KDD Cup 1999 Data. (n.d.). Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Kolter, J. Z., & Maloof, M. A. (2006). Learning to Detect and Classify Malicious Executables in the Wild. *The Journal of Machine Learning Research, 7*, 2721-2744.

Mabu, S., Chen, C., Lu, N., Shimada, K., & Hirasawa, K. (2011). An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining Using Genetic Network Programming. *IEEE Transactions On Systems, Man, And Cybernetics - Part C: Applications And Reviews, 41*(1), 130-139.

Mannila, H. (1996). Data Mining: Machine Learning, Statistics, and Databases. *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management*, (pp. 2-9). Stockholm, Sweden.

McHugh, J. (2000). Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security (TISSEC), 3*(4), 262-294.

Murugesan, V., Shalinie, M., & Neethimani, N. (2014). A Brief Survey of IP Traceback Methodologies. *Acta Polytechnica Hungarica, 11*(9), 197-216.

Nguyen, H. A., & Choi, D. (2008). Application of Data Mining to Network Intrusion Detection: Classifier Selection Model. *Proceedings of the 11th Asia-Pacific Symposium on Network Operations and Management: Challenges for Next Generation Network Operations and Service Management (APNOMS '08)*, (pp. 399-408). Beijing, China.

Nguyen, T. T., & Armitage, G. (2008). A Survey of Techniques for Internet Traffic Classification using Machine Learning. *IEEE Communications Surveys & Tutorials, 10*(4), 56-76.

Patil, T. R., & Sherekar, S. S. (2013). Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification. *International Journal of Computer Science and Application, 6*(2), 256-261.

Que, Q., & Belkin, M. (2016). Back to the Future: Radial Basis Function Networks Revisited. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS), 51*, pp. 1375-1383. Cadiz, Spain.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Moels and Selected Application in Speech Recognition. *Proceedings of the IEEE*, 257-286.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development, 3*(3), 210-229.

Siddiqui, S., Khan, M. S., Ferens, K., & Kinsner, W. (2016). Detecting Advanced Persistent Threats using Fractal Dimension based Machine Learning Classification. *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics (IWSPA '16)*, (pp. 64-69). New Orleans, Louisiana, USA.

Sikora, R., & Al-laymoun, O. H. (2014). A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms. *Journal of International Technology and Information Management, 23*(1).

Solanki, M., & Dhamdhere, V. (2015). Intrusion Detection System Using Means of Data Mining By Using C 4.5 Algorithm. *International Journal of Application or Innovation in Engineering & Management (IJAIEM), 4*(5).

Song, D. X., & Perrig, A. (2001). Advanced and Authenticated Marking Schemes for IP Traceback. *Proceedings of IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), 2*, pp. 878-886. Anchorage, Alaska, USA.

Symantec. (2016, April). *Internet Security Threat Report.* Retrieved from https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Application (CISDA 2009)*, (pp. 53-58). Ottaws, Ontario, Canada.

Tom Mitchell. (1997). *Machine Learning.* McGraw Hill.

Wang, Y. (2008). *Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection.* IGI Global.

Weka 3: Data Mining Software in Java. (n.d.). Retrieved from http://www.cs.waikato.ac.nz/ml/weka/

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., . . . Steinberg, D. (2008). Top 10 Algorithms in Data Mining. *Knowledge Information System, 14*, 1-37.

Ye, N., Li, X., Chen, Q., Emran, S., & Xu, M. (2001). Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 266-274.

Zhou, M., & Lang, S.-d. (2003). Mining Frequency Content of Network Traffic for Intrusion Detection. *Proceedings of IASTED International Conference on Communication, Network and Information Security (CNIS 2003)*, (pp. 101-107). New York City, New York, USA.

Figure 1: Kappa Statistic

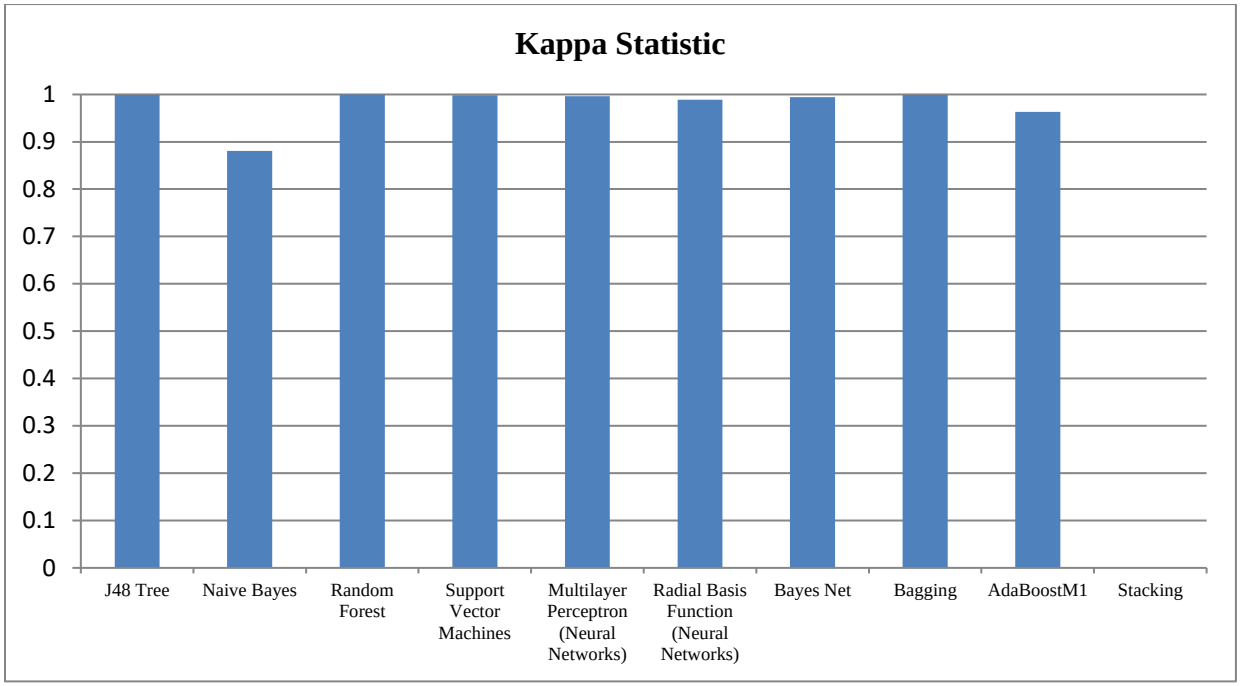

Figure 2: Root Mean Squared Error

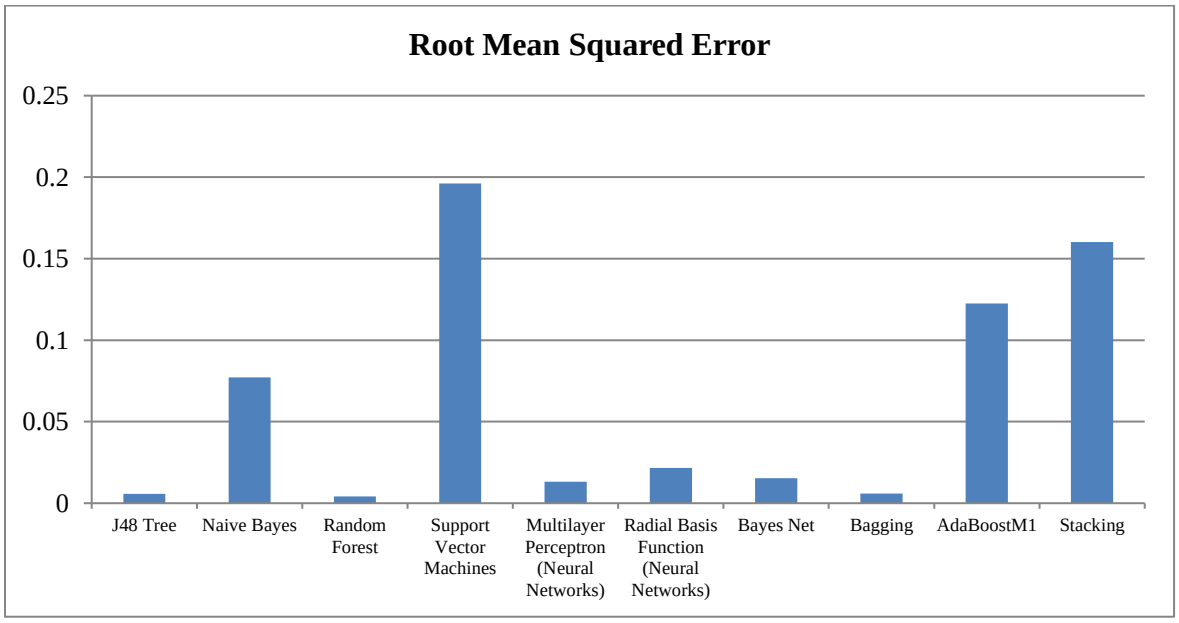

Table1: Benchmark comparison of various algorithms

Algorithm	Time to Build Model (seconds)	Correctly Classified Instances (%)	Kappa Statistic	Root Mean Squared Error
J48 Tree	69.47	99.9603 %	0.9993	0.0057
Naïve Bayes	2.89	92.7794 %	0.8806	0.0772
Random Forest	588.21	99.9794 %	0.9997	0.0041
Support Vector Machines (SVM)	254.31	99.9245 %	0.9987	0.1961
Multilayer Perceptron (Neural Networks)	46212.93	99.7911 %	0.9965	0.0132
Radial Basis Function (Neural Networks)	41623.78	99.3243%	0.9886	0.0217
Bayes Net	17.42	99.667 %	0.9944	0.0153
Bagging	206.18	99.9524 %	0.9992	0.0059
AdaBoostM1	65.12	97.8576 %	0.9635	0.1225
Stacking	2.31	56.8378 %	0	0.1603

Table 2: Benchmark comparison of various algorithms based on Time to Build Model (seconds)

Algorithm	Ranks based on Time to Build Model (seconds)
Stacking	2.31
Naïve Bayes	2.89
Bayes Net	17.42
AdaBoostM1	65.12
J48 Tree	69.47
Bagging	206.18
Support Vector Machines	254.31
Random Forest	588.21
Radial Basis Function (Neural Networks)	41623.78
Multilayer Perceptron (Neural Networks)	46212.93

Table 3: Benchmark comparison of various algorithms based on Correctly Classified Instance (%)

Algorithm	Ranks based on Correctly Classified Instances (%)
Random Forest	99.9794 %
J48 Tree	99.9603 %
Bagging	99.9524 %
Support Vector Machines	99.9245 %
Multilayer Perceptron (Neural Networks)	99.9245 %
Bayes Net	99.667 %
Radial Basis Function (Neural Networks)	99.3243 %
AdaBoostM1	97.8576 %
Naive Bayes	92.7794 %
Stacking	56.8378 %

Table 4: Accuracy by Class of Attacks (J48 Tree)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0.998	0	0.998	0.998	0.998	0.999	Back
0.999	0	1	0.999	0.999	0.999	teardrop
0	0	0	0	0	0.729	load module
1	0	1	1	1	1	Neptune
0	0	0	0	0	0.622	rootkit
1	0	0.667	1	0.8	1	phf
0.989	0	0.992	0.989	0.991	0.996	Satan
0.7	0	0.656	0.7	0.677	0.895	Buffer_overflow
0	0	0	0	0	0.578	ftp_write
0.81	0	0.895	0.81	0.85	0.905	land
0	0	0	0	0	0.453	spy
0.993	0	0.994	0.993	0.993	0.999	ipsweep
0	0	0	0	0	0.857	multihop
1	0	1	1	1	1	smurf
1	0	0.996	1	0.998	1	pod
0.667	0	0.5	0.667	0.571	0.997	perl
0.979	0	0.994	0.079	0.987	0.998	warezclient
0.952	0	0.991	0.333	0.421	0.995	nmap
0.333	0	0.571	0.333	0.421	0.818	imap
0.8	0	0.889	0.8	0.842	0.923	warezmaster
0.986	0	0.993	0.986	0.989	0.997	portsweep
1	0	0.999	1	0.999	1	normal
0.943	0	0.962	0.943	0.952	0.972	Guess_passwd

Table 5: Accuracy by Class of Attacks (Naïve Bayes)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0.975	0.005	0.485	0.975	0.648	0.999	Back
0.995	0.001	0.638	0.995	0.777	0.999	teardrop
0.556	0	0.022	0.556	0.041	0.999	load module
0.996	0	1	0.996	0.998	1	Neptune
0.5	0.003	0.003	0.5	0.006	0.977	rootkit
0.75	0	0.071	0.75	0.13	0.995	phf
0.954	0.002	0.58	0.954	0.722	0.996	Satan
0.133	0	0.018	0.133	0.032	0.999	Buffer_overflow
0.75	0.003	0.004	0.75	0.009	0.997	ftp_write
0.952	0	0.323	0.952	0.482	1	land
1	0	1	1	1	1	spy
0.966	0.009	0.205	0.966	0.339	0.994	ipsweep
0.429	0	0.068	0.429	0.118	1	multihop
0.999	0	1	0.999	0.999	1	smurf
0.985	0.037	0.014	0.985	0.028	0.998	pod
0.333	0	0.333	0.333	0.333	1	perl
0.478	0.006	0.143	0.478	0.22	0.988	warezclient
0.446	0.001	0.167	0.446	0.243	0.995	nmap
0.917	0	0.141	0.917	0.244	0.942	imap
0.9	0.001	0.055	0.9	0.103	0.994	warezmaster
0.907	0.001	0.65	0.907	0.757	0.998	portsweep
0.652	0.001	0.997	0.652	0.788	0.999	normal
0.943	0.001	0.07	0.943	0.13	0.989	Guess_passwd

Table 6: Accuracy by Class of Attacks (Random Forest)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
1	0	1	1	1	1	Back
1	0	1	1	1	1	teardrop
0.333	0	1	0.333	0.5	1	load module
1	0	1	1	1	1	Neptune
0.1	0	1	0.1	0.182	1	rootkit
0.75	0	1	0.75	0.857	1	phf
0.99	0	0.999	0.99	0.995	0.999	Satan
0.833	0	0.833	0.833	0.833	1	Buffer_overflow
0.375	0	1	0.375	0.545	0.937	ftp_write
0.857	0	0.947	0.857	0.9	1	land
0	0	0	0	0	1	spy
0.991	0	0.996	0.991	0.994	1	ipsweep
0.429	0	0.6	0.429	0.5	0.929	multihop
1	0	1	1	1	1	smurf
0.996	0	0.996	0.996	0.996	1	pod
0.667	0	1	0.667	0.8	1	perl
0.993	0	0.994	0.993	0.994	1	warezclient
0.974	0	1	0.974	0.987	1	nmap
1	0	1	1	1	1	imap
0.8	0	0.889	0.8	0.842	1	warezmaster
0.995	0	0.996	0.995	0.996	1	portsweep
1	0	0.999	1	1	1	normal
0.962	0	1	0.962	0.981	1	Guess_passwd

Table 7: Accuracy by Class of Attacks (Support Vector Machines)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0.997	0	0.991	0.997	0.994	1	Back
0.998	0	1	0.998	0.999	1	teardrop
0	0	0	0	0	0.994	load module
1	0	1	1	1	1	Neptune
0	0	0	0	0	0.882	rootkit
0	0	0	0	0	1	phf
0.972	0	0.998	0.972	0.985	0.998	Satan
0.6	0	0.75	0.6	0.667	0.999	Buffer_overflow
0.5	0	0.667	0.5	0.571	0.999	ftp_write
1	0	0.955	1	0.977	1	land
0	0	0	0	0	1	spy
0.982	0	0.987	0.982	0.984	1	ipsweep
0	0	0	0	0	0.998	multihop
1	0	1	1	1	1	smurf
0.992	0	0.992	0.992	0.992	1	pod
0	0	0	0	0	1	perl
0.915	0	0.925	0.915	0.92	1	warezclient
0.965	0	0.933	0.965	0.949	0.997	nmap
0.833	0	1	0.833	0.909	1	imap
0.75	0	0.789	0.75	0.769	0.999	warezmaster
0.994	0	0.998	0.994	0.996	0.998	portsweep
0.999	0.001	0.998	0.999	0.998	0.999	normal
0.943	0	0.98	0.943	0.962	1	Guess_passwd

Table 8: Accuracy by Class of Attacks (Multilayer Perceptron)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0.727	0	0.979	0.727	0.835	0.992	Back
1	0	0.998	1	0.999	1	teardrop
0	0	0	0	0	0.648	load module
1	0	1	1	1	1	Neptune
0	0	0	0	0	0.751	rootkit
0	0	0	0	0	0.931	phf
0.987	0	0.994	0.987	0.99	0.996	Satan
0.2	0	0.286	0.2	0.235	0.885	Buffer_overflow
0	0	0	0	0	0.846	ftp_write
0	0	0	0	0	0.93	land
0	0	0	0	0	0.736	spy
0.982	0	0.964	0.982	0.973	0.999	ipsweep
0	0	0	0	0	0.94	multihop
1	0	1	1	1	1	smurf
0.981	0	0.974	0.981	0.977	0.997	pod
0	0	0	0	0	0.769	perl
0.917	0	0.936	0.917	0.926	0.995	warezclient
0.792	0	0.91	0.792	0.847	0.936	nmap
0	0	0	0	0	0.918	imap
0	0	0	0	0	0.953	warezmaster
0.996	0	0.984	0.996	0.99	0.998	portsweep
0.999	0.002	0.992	0.999	0.995	0.999	normal
0.925	0	0.69	0.925	0.79	1	Guess_passwd

Table 9: Accuracy by Class of Attack (Radial Basis Function)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0.907	0	0.978	0.907	0.941	0.998	Back
0.994	0	1	0.994	0.997	1	teardrop
0.333	0	0.6	0.333	0.429	0.868	load module
0.998	0	0.999	0.998	0.999	1	Neptune
0	0	0	0	0	0.66	rootkit
0.75	0	1	0.75	0.857	0.796	phf
0.897	0.001	0.824	0.897	0.859	0.999	Satan
0.567	0	0.81	0.567	0.667	0.944	Buffer_overflow
0	0	0	0	0	0.993	ftp_write
0.19	0	0.444	0.19	0.267	0.961	land
0	0	0	0	0	0.182	spy
0.52	0	0.992	0.52	0.682	0.998	ipsweep
0	0	0	0	0	0.722	multihop
0.999	0	1	0.999	0.999	1	smurf
0.841	0	0.892	0.841	0.865	1	pod
0.333	0	1	0.333	0.5	0.455	perl
0	0	0	0	0	0.994	warezclient
0.442	0	1	0.442	0.613	0.992	nmap
0.583	0	0.875	0.583	0.7	0.727	imap
0.15	0	0.5	0.15	0.231	0.931	warezmaster
0.88	0	0.963	0.88	0.92	0.997	portsweep
0.996	0.007	0.972	0.996	0.984	0.999	normal
0.83	0	0.917	0.83	0.871	0.969	Guess_passwd

Table 10: Accuracy by Class of Attacks (Bayes Net)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0.997	0	1	0.997	0.998	1	Back
0.999	0	0.924	0.999	0.96	1	teardrop
0	0	0	0	0	0.999	load module
0.999	0	1	0.999	1	1	Neptune
0.4	0	0.033	0.4	0.06	0.994	rootkit
0.5	0	1	0.5	0.667	1	phf
0.913	0	0.982	0.913	0.946	1	Satan
0.767	0	0.288	0.767	0.418	1	Buffer_overflow
0.25	0	0.087	0.25	0.129	0.998	ftp_write
1	0	0.189	1	0.318	1	land
0	0	0	0	0	0.999	spy
0.929	0.001	0.729	0.929	0.817	1	ipsweep
0	0	0	0	0	0.994	multihop
1	0	1	1	1	1	smurf
0.989	0	0.779	0.989	0.871	1	pod
0	0	0	0	0	1	perl
0.996	0	0.815	0.996	0.896	1	warezclient
0.472	0	0.341	0.472	0.396	0.998	nmap
0.833	0	0.286	0.833	0.426	0.998	imap
0.75	0	0.221	0.75	0.341	0.999	warezmaster
0.977	0	0.905	0.977	0.939	1	portsweep
0.989	0	0.999	0.989	0.994	1	normal
0.962	0	0.85	0.962	0.903	1	Guess_passwd

Table 11: Accuracy by Class of Attacks (Bagging)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
1	0	1	1	1	1	Back
1	0	1	1	1	1	teardrop
0.444	0	0.571	0.444	0.5	0.944	load module
1	0	0.999	1	1	1	Neptune
0	0	0	0	0	0.899	rootkit
0.5	0	1	0.5	0.667	1	phf
0.981	0	0.996	0.981	0.989	0.994	Satan
0.867	0	0.722	0.867	0.788	0.967	Buffer_overflow
0	0	0	0	0	0.875	ftp_write
0.952	0	0.952	0.952	0.952	0.976	land
0	0	0	0	0	1	spy
0.974	0	0.997	0.974	0.985	0.996	ipsweep
0	0	0	0	0	0.928	multihop
1	0	1	1	1	1	smurf
1	0	0.992	1	0.996	1	pod
0.333	0	1	0.333	0.5	1	perl
0.99	0	0.979	0.99	0.984	1	warezclient
0.965	0	0.97	0.965	0.967	0.989	nmap
0.833	0	1	0.833	0.909	0.958	imap
0.85	0	0.81	0.85	0.829	0.95	warezmaster
0.951	0	0.999	0.951	0.974	0.988	portsweep
1	0	0.999	1	0.999	1	normal
0.925	0	0.942	0.925	0.933	1	Guess_passwd

Table 12: Accuracy by Class of Attacks (AdaBoostM1)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0	0	0	0	0	0.692	Back
0	0	0	0	0	0.661	teardrop
0	0	0	0	0	0.643	load module
0.995	0.007	0.975	0.995	0.985	0.998	Neptune
0	0	0	0	0	0.663	rootkit
0	0	0	0	0	0.738	phf
0	0	0	0	0	0.856	Satan
0	0	0	0	0	0.665	Buffer_overflow
0	0	0	0	0	0.661	ftp_write
0	0	0	0	0	0.586	land
0	0	0	0	0	0.71	spy
0	0	0	0	0	0.588	ipsweep
0	0	0	0	0	0.662	multihop
0.999	0	1	0.999	0.999	1	smurf
0	0	0	0	0	0.584	pod
0	0	0	0	0	0.638	perl
0	0	0	0	0	0.691	warezclient
0	0	0	0	0	0.598	nmap
0	0	0	0	0	0.617	imap
0	0	0	0	0	0.599	warezmaster
0	0	0	0	0	0.608	portsweep
0.99	0.02	0.925	0.99	0.956	0.995	normal
0	0	0	0	0	0.688	Guess_passwd

Table 13: Accuracy by Class of Attacks (Stacking)

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC area	Class of Attacks
0	0	0	0	0	0.5	Back
0	0	0	0	0	0.5	teardrop
0	0	0	0	0	0.45	load module
0	0	0	0	0	0.5	Neptune
0	0	0	0	0	0.5	rootkit
0	0	0	0	0	0.2	phf
0	0	0	0	0	0.5	Satan
0	0	0	0	0	0.5	Buffer_overflow
0	0	0	0	0	0.4	ftp_write
0	0	0	0	0	0.479	land
0	0	0	0	0	0.1	spy
0	0	0	0	0	0.499	ipsweep
0	0	0	0	0	0.35	multihop
1	1	0.568	1	0.725	0.5	smurf
0	0	0	0	0	0.495	pod
0	0	0	0	0	0.15	perl
0	0	0	0	0	0.5	warezclient
0	0	0	0	0	0.498	nmap
0	0	0	0	0	0.433	imap
0	0	0	0	0	0.5	warezmaster
0	0	0	0	0	0.5	portsweep
0	0	0	0	0	0.5	normal
0	0	0	0	0	0.48	Guess_passwd

Table 14: Tests of Normality

Algorithms	Kolmogorov-Smirnov Test (KS) Statistic	KS df	KS Sig.	Shapiro-Wilk Test (Sh-W) Statistic	Sh-W df	Sh-W Sig.
J48 Tree	0.287	23	0.000	0.697	23	0.000
Naïve Bayes	0.266	23	0.000	0.833	23	0.001
Random Forest	0.279	23	0.000	0.729	23	0.000
Support Vector Machines	0.281	23	0.000	0.702	23	0.000
Multilayer Perceptron	0.287	23	0.000	0.711	23	0.000
Radial Basis Function	0.173	23	0.073	0.873	23	0.007
Bayes Net	0.245	23	0.001	0.767	23	0.000
Bagging	0.310	23	0.000	0.708	23	0.000
AdaBoostM1	0.517	23	0.000	0.403	23	0.000
Stacking	0.539	23	0.000	0.215	23	0.000

Table 15: Kruskal-Wallis Test

	True Positive Rates
Chi-Square	62.965
df	9
Asymp. Signif.	0.000

Table 16: Pairwise comparison for true positive rates

Algorithms	Mann-WhitneyU	WilcoxonW	Z Statistic	Asymp. Signif. Two-tailed	Mean Rank
J48 Tree vs AdaboostM1	98.500	374.500	-3.983	0.000	J48 Tree > AdaboostM1
J48 Tree vs Stacking	75.500	351.500	-4.654	0.000	J48Tree > Stacking
Naïve Bayes vs Radial Basis Function	170.500	446.500	-2.067	0.039	Naïve Bayes > Radial Basis Function
Naïve Bayes vs AdaBoostM1	60.000	336.000	-4.690	0.000	Naïve Bayes > AdaBoostM1
Naïve Bayes vs Stacking	22.500	298.500	-5.633	0.000	Naïve Bayes > Stacking
Random Forest vs Multilayer Perceptron	169.500	445.500	-2.109	0.035	Random Forest > Multilayer Perceptron
Random Forest vs Radial Basis Function	150.500	426.500	-2.510	0.012	Random Forest > Radial Basis Function
Random Forest vs AdaBoostM1	55.000	331.000	-4.844	0.000	Random Forest > AdaBoostN1
Random Forest vs Stacking	31.000	307.000	-5.494	0.000	Random Forest > Stacking
Support Vector Machines vs AdaBoostM1	111.500	387.500	-3.713	0.000	Support Vector Machines > AdaBoostM1
Support Vector Machines vs Stacking	87.500	363.500	-4.420	0.000	Support Vector Machines > Stacking

Table 17: Pairwise comparison for true positive rates

Algorithms	Mann-Whitney U	Wilcoxon W	Z Statistic	Asymp. Sig. (2-tailed)	Mean Rank
Multilayer Perceptron vs AdaBoostM1	155.500	431.500	-2.817	0.005	Multilayer Perceptron > AdaBoostM1
Multilayer Perceptron vs Stacking	131.500	407.500	-3.589	0.000	Multilayer Perceptron > Stacking
Radial Basis Function vs Bagging	173.000	449.000	-2.020	0.043	Radial Basis Function > Bagging
Radial Basis Function vs AdaBoostM1	111.500	387.500	-3.668	0.000	Radial Basis Function > AdaBoostM1
Radial Basis Function vs Stacking	78.000	354.000	-4.587	0.000	Radial Basis Function > Stacking
Bayes Net vs AdaBoostM1	94.000	370.000	-4.044	0.000	Bayes Net > AdaBoostM1
Bayes Net vs Stacking	66.000	342.000	-4.818	0.000	Bayes Net > Stacking
Bagging vs AdaBoostM1	90.500	366.500	-4.132	0.000	Bagging > AdaBoostM1
Bagging vs Stacking	64.00	340.000	-4.876	0.000	Bagging > Stacking