The impact of using large training data set KDD99 on classification accuracy

Atilla Özgür Corresp., [1] , **Hamit Erdem** [1]

[1] Electrical Engineering, Başkent University, Ankara, Ankara, Turkey

Corresponding Author: Atilla Özgür
Email address: ati.ozgur@gmail.com

This study investigates the effects of using a large data set on supervised machine learning classifiers in the domain of Intrusion Detection Systems (IDS). To investigate this effect 12 machine learning algorithms have been applied. These algorithms are: (1) Adaboost, (2) Bayesian Nets, (3) Decision Tables, (4) Decision Trees (J48), (5)Logistic Regression, (6) Multi-Layer Perceptron, (7) Naive Bayes, (8) OneRule, (9)Random Forests, (10) Radial Basis Function Neural Networks, (11) Support Vector Machines (two different training algorithms), and (12) ZeroR. A well-known IDS benchmark dataset, KDD99 has been used to train and test classifiers. Full training data set of KDD99 is 4.9 million instances while full test dataset is 311,000 instances. In contrast to similar previous studies, which used 0.08%–10% for training and 1.2%–100% for testing, this study uses full training dataset and full test dataset. Weka Machine Learning Toolbox has been used for modeling and simulation. The performance of classifiers has been evaluated using standard binary performance metrics: Detection Rate, True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate, Precision, and F1-Rate. To show effects of dataset size, performance of classifiers has been also evaluated using following hardware metrics: Training Time, Working Memory and Model Size. Test results shows improvements in classifiers in standard performance metrics compared to previous studies.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

The impact of using large training data set on classification accuracy

Atilla Özgür[1] and Hamit Erdem[2]

[1]**Başkent University, Ankara**
[2]**Başkent University, Ankara**

Corresponding author:
Atilla Özgür[1]

Email address: ati.ozgur@gmail.com

ABSTRACT

This study investigates the effects of using a large data set on supervised machine learning classifiers in the domain of Intrusion Detection Systems (IDS). To investigate this effect 12 machine learning algorithms have been applied. These algorithms are: (1) Adaboost, (2) Bayesian Nets, (3) Decision Tables, (4) Decision Trees (J48), (5)Logistic Regression, (6) Multi-Layer Perceptron, (7) Naive Bayes, (8) OneRule, (9)Random Forests, (10) Radial Basis Function Neural Networks, (11) Support Vector Machines (two different training algorithms), and (12) ZeroR. A well-known IDS benchmark dataset, KDD99 has been used to train and test classifiers. Full training data set of KDD99 is 4.9 million instances while full test dataset is 311,000 instances. In contrast to similar previous studies, which used 0.08%–10% for training and 1.2%–100% for testing, this study uses full training dataset and full test dataset. Weka Machine Learning Toolbox has been used for modeling and simulation. The performance of classifiers has been evaluated using standard binary performance metrics: Detection Rate, True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate, Precision, and F1-Rate. To show effects of dataset size, performance of classifiers has been also evaluated using following hardware metrics: Training Time, Working Memory and Model Size. Test results shows improvements in classifiers in standard performance metrics compared to previous studies.

1 INTRODUCTION

Internet, networks and computers form the backbone of modern life; protection of this backbone is very important (Raymond and Choo, 2011). According to Computer Security Institute survey (CSI, 2011), following tools are used to protect these sytems: firewalls, anti-viruses, malware protection programs, intrusion detection systems (IDS), and intrusion detection and prevention systems. IDS is used by 62% of enterprises that use variety of open source and commercial systems. IDS are categorized into the following classes (Scarfone and Mell, 2007):

1. According to deployment of systems
   (a) Network Intrusion Detection Systems (Network IDS)
   (b) Host Intrusion Detection Systems (Host IDS)
2. According to detection methodology.
   (a) Signature Detection
   (b) Anomaly Detection

In the first category, network IDS uses network packets to detect attacks; consequently, it protects many computers in the network. On the other hand, host IDS uses logs and events in host system to detect attacks; therefore, it protects only one host.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

Signature detection or misuse systems use signature database of known attacks to detect intrusions, but this database must be updated regularly. Their performance is low against unknown attacks; while it is very high against known attacks. For that reason, false alarms occur very rarely. Most enterprises prefer signature detection systems since false alarms are costly and resource intensive(CSI, 2011).

However, anomaly detection systems suggest a different approach. They maintain system profiles that define normal activities. When an abnormal activity (different from the stored profiles) occurs, the system flags this activity as an intrusion. Anomaly detection systems produce more false alarms than the signature detection systems, since *the definition of normal* changes in time. But, they are more effective against unknown attacks.

After Denning's first paper (Denning, 1987) about IDS, hundreds of studies have been published. Nonetheless, it still remains an unsolved problem since IDS domain is a evolving problem—Attackers continuously change and improve their capabilities(Sommer and Paxson, 2010).

To introduce importance of the problem and evaluate related studies, results from previous works have been collected for comparison purposes (Table 1). These results indicates that many methods have been applied in IDS domain, and Weka (Hall et al., 2009) is the most-used toolbox in literature.

Based on Table 1 that presents previous 16 studies, nearly all of them utilized only a fraction of available training data — KDD99 benchmark dataset consists of 4898431 training instances and 311029 test instances. But in the literature, minimum training set size is 4,000 instances (0.08%); maximum training 485,000 (10%), minimum test 2,650 (1.2%), maximum test 311,000(100%). In addition, it is common to see claims that KDD99 is too large for research study purposes (Horng et al., 2011; Yi et al., 2011; Chen et al., 2014; Kumar and Kumar, 2013) , justifying usage of small subset of KDD99.

Using a restricted portion of the training data results in performance reduction in classifier models, since classifier models can only learn the data that they have seen in the training step. From generalization point of view, the models' capabilities reduce as a result of using small datasets; therefore, using a larger training dataset brings *small but non-trivial gains* in generalization performance(Perlich, 2009; Cortes et al., 1994). As a result, using whole available training dataset may improve the generalization capability of classifiers. Considering the effect of using all training data in classification, this study proposes using the full training and the testing dataset with 4,898,431 and 311,029 instances.

Definition of Reproducibility is that other researchers (wikipedia, 2015) should be able to reproduce given study. Reproducibility is one of the corners of scientific method. Nonetheless, most of the published studies are not reproducible(Gentleman and Temple Lang, 2007; Vandewalle et al., 2009; Peng, 2011). Study Reproducibility is needed for wide dissemination of results and comparison of scientific studies. Current study is fully reproducible with all of its code is open sourced in widely known github site. Researches can fully start our experiments with less than 15 minutes of effort. Of course some of our experiments run very long time (more than **56** hours for Multi Layer Perceptron, see Table 4) and takes significant hardware resources; thus, fully reproducing our study will need time and necessary hardware. Since most of the studies in KDD99 is not easily reproducible or comparable, current study can be benchmark study for further studies. Consequently, this study aims five contributions:

- First, while most of studies have used 3–6 algorithms for comparison purposes, 13 supervised machine learning classifiers are trained and evaluated. These classifiers are Adaboost, Bayesian Nets, Decision Tables, Decision Trees (J48), Logistic Regression, Multi-Layer Perceptron, Naive Bayes, OneRule, Random Forests, Radial Basis Function Neural Networks, Stochastic Gradient Descent for Support Vector Machines, Sequential Minimal Optimization for Support Vector Machines and ZeroR.

- Second, large data sets require more computing power. To show efect of large data set size on computing resources, following hardware metrics (training time, working memory, and model size) has been presented in detail, section 2.3 and Table 4. Evaluation is also performed with following standard binary performance metrics: Detection Rate, True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate, Precision and F1-Rate (Table 6 and Table 7).

- Third, using whole training dataset — instead of fraction of it — improves classification results in IDS compared to previous studies, see Table 8. This improvement is predicted by Learning Curves(Perlich, 2009; Cortes et al., 1994).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

- Fourth, this study presents a fully reproducible environment for further KDD99 Dataset studies. Any researcher will be able to start our experiments with less than 15 minutes of time and will be able to add new classifiers to this experiment.

- Last, the proposed study may be a reference study for similar studies in IDS using KDD99 due to previous contributions.

2 EXPERIMENTS

2.1 Dataset KDD99

KDD-DARPA dataset is a simulated dataset, intended to be similar to a network of an US Air Force Base. DARPA sponsored an IDS event in MIT Lincoln Lab in 1998 (Cunningham et al., 1999). This event was repeated in 1999 by using improvements suggested by Computer Security Community (Lippmann et al., 2000). In these events, network dump files and other system files are released to participants. Participants preprocessed these files and used different algorithms to find intrusions.

One of the DARPA IDS teams (Lee and Stolfo, 2000) released their preprocessed dataset to Knowledge Discovery and Data Mining (KDD) conference. This dataset was used in KDD 99 yearly competition (KDDCup, 1999). Even though DARPA dataset is suitable for both Host IDS and Network IDS research, KDD99 is suitable for only Network IDS research.

Public benchmark dataset, KDD99 consists of seven weeks of computer activity simulation. To ease training of anomaly detection algorithms, first two weeks contains no attacks while the remaining five weeks are mostly attacks. These attacks are divided into 4 major groups:

1. Denial of Service (DOS)

2. Probing attacks (Probe)

3. Remote to Local (R2L)

4. User to Root (U2R)

DOS attacks are designed to exhaust the target system; accordingly, they repeat the same attack over and over to consume system resources. Probe attacks are designed to get more information about the target system. R2L attacks are designed to give local access to target system; thus,they are more dangerous than DOS and probe attacks. U2R attacks give root access (super user) to normal user. Since root can do anything in system, U2R attacks are the most dangerous of all attacks in this dataset. Last two attacks, R2L and U2R, are very rare in KDD99.

Applying preprocessing, 41 attributes have been created in KDD99 (Lee and Stolfo, 2000) from DARPA dataset. Training dataset of KDD99 consists of about 4.9 million records of 22 attack and normal instances. This dataset is widely imbalanced and about 80% of data set are attack instances, Table 2. Test dataset consists of 311,029 instances. Test and training datasets have different probability distributions (Elkan, 1999), and test dataset has some new attacks that do not exist in training dataset. These new attacks measure generalization of IDS models. In other words whether **IDS models capable of distinguishing zero day attacks (unknown attacks)** or not.

Interestingly, KDD99 is one of the most used data sets in IDS research(Tavallaee et al., 2010). A recent survey by Tavallaee et al (Tavallaee et al., 2010) reviewed 276 studies (93 Host IDS and 163 Network IDS), mostly in indexed journals. Of these studies: (i) 67 of them (24%) used DARPA, (ii) 77 (28%) KDD99, (iii) 41 (15%) injected other attacks to KDD99, (iv) 86 (31%) did not disclose any dataset information, and (v) only 16 (6%) have used other datasets. In addition, KDD99 dataset is used in 149 articles in 65 different journals with Science Citation Index impact factor(Özgür and Erdem, 2016). This number, 149 , does not include any conference articles, only journals. See Fig 1 for usage by year.

On the other hand, this dataset has well known problems (McHugh, 2000; Brugger, 2007), but despite its problems, using this dataset is still useful to IDS domain. Brugger and Chow (Brugger and Chow, 2005) have used Snort on tcpdump data from DARPA data set to show that DARPA data set is still useful for testing IDS. Additionally, good performance against DARPA dataset is a "*necessary but not sufficient*" (emphasis original) condition to show the capabilities of advanced IDS.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

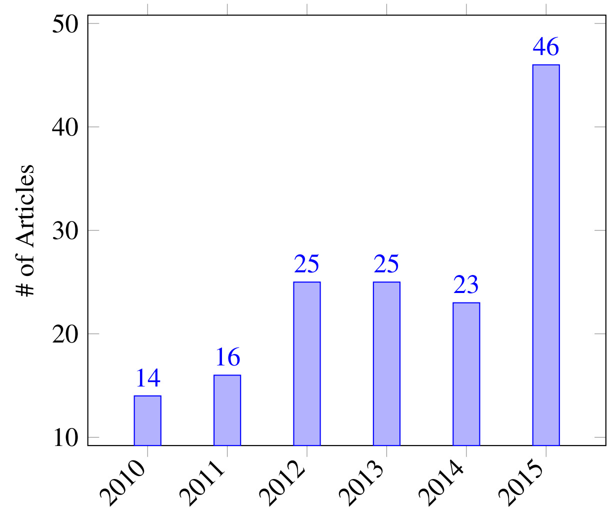

Figure 1. KDD99 Dataset Usage By Years taken from (Özgür and Erdem, 2016).

2.2 Reproducible Build Tool

Vandewalle et al (Vandewalle et al., 2009) proposes six levels of reproducibility. The best levels is:

> The results can be easily reproduced by an independent researcher with at most 15 min of user effort, requiring only standard, freely available tools (C compiler, etc.).

All of our used tools are open source; hence, freely available. Java Development Kit (JDK) should be installed in the machine. In addition our build script downloads most of the necessary libraries using gradle's dependency management. See following url for details: `https://github.com/ati-ozgur/KDD99`.

2.2.1 How a new researcher would use our tool KDD99 Reproducible Test Bench

1. Control that necessary programs are installed (java,sqlite3)
2. Download code from github
3. run startFresh script.

Algorithm 1 How Build Tool Works?

Input: java,sqlite3 works, internet connection
Output: Trained models, training and test results

1. Download Used Gradle Distribution
2. Download dependencies (Groovy,Weka, Other packages) from maven repositories.
3. Download KDD99 Files
4. Unzip KDD99 Files
5. import to sqlite
6. Create necessary views and indexes
7. Create Training and Testing ARFF files for weka
8. **for** $i = 1$ **to** N **do**
9.   Train Classifier Model using Training Dataset and save model file
10.   Test Trained Model on Full KDD99 Training Dataset
11.   Test Trained Model on Full KDD99 Test Dataset
12. **end for**
13. **return** Trained Models and Training Results

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

After running start script, build tool works according to Algorithm 1. Due to cross platform nature of used tools, this experiment is also cross platform and is tested in following platforms.

1. Linux (Linux Mint 16,17.1, Ubuntu 14.04 LTS Desktop)
2. MacOS (10.8,10.9,10.10)
3. Windows (7,8, Server 2012)

Note that: Most of the experiments need a lot of RAM due to large size of KDD99. See section 2.3 for necessary Java heap settings.

2.3 Test Environment

Even though our environment are cross platform, experiments are conducted on only one computer. Its configuration is following.

1. Windows Server 2012 R2
2. Java 1.7.60 64 bit
3. Weka 3.7.12 (Developer Edition)
4. Intel(R) Xeon(R) CPU E5335 @ 2.00GHz (2 Socket-8 Core)
5. 16 GB RAM
6. 80 GB Solid State Disk (SSD) Hard disk that is used as page file memory
7. Java Heap Memory Settings -Xms20096m -Xmx30192m (Start with 20 GB, go to max 30 GB memory)

Especially, SSD and Heap Memory settings are important; since, these settings permit weka to use more memory than available physical memory. Weka is the most-used tool in previous studies, Table 1. Using command line interface of Weka has been preferred in this study to reduce memory usage. When training batch machine learning algorithms with Weka on full KDD99 training dataset, a powerful hardware with high memory and powerful CPU are needed. With lesser hardware, especially with low RAM, it is impossible to train a classifier on full KDD99 dataset.

2.4 Selected Machine Learning Classifiers

Choosing diverse classifiers helps to understand effect of dataset size on following metrics —Training Time, Training Memory, and Model Size. Even though it is obvious that using more training data will increase this metrics, Rate of increase is not so obvious. Classifiers usage of memory, training time and model size are very different, for example high model size of RBFNetwork was unexpected. Classifiers were included according to following 4 criteria, whether classifier: (1) is easy to train, (2) is among the most used, (3) is easy to understand, and (4) is hard to train. Table 3 contains the four groups, classifiers in these groups, and previous studies.

Easy to train classifiers are internally simple algorithms. Their training times are very fast; therefore, they are mostly used for comparison purposes, not as building blocks for real systems. Since they are used for comparison, they are also called baseline algorithms. A good classifier is expected to out-perform these algorithms. ZeroR (Zero Rule) and OneR (One Rule) are used as baseline algorithms.

The following **Most Used Algorithms** (Wu et al., 2008) are selected: (1) Decision Tree, (2) Naive Bayes, (3) Adaboost and other boosting algorithms, (4) Random Forests.

Easy to Understand operating principles are a desirable feature of IDS (Scarfone and Mell, 2007). IDS operators can easily understand operating principles of following classifiers, and can easily comment on why system flagged that instance as an attack. These are (1) Logistic Regression, (2) Decision Table, (3) BayesNet and (4) Decision Tree (DT).

Hard to Train Algorithms take more time to train than other classifiers, since their underlying mechanisms are more computationally intense. Hard to Train Algorithms are (1) Artificial Neural Network-Multi Layer Perceptron (ANN-MLP), (2) Artificial Neural Network-Radial Basis Function (ANN-RBF), (3) Support Vector Machines (SVM). Our study uses two different training methods for SVM. These training methods are stochastic gradient descent (SGD)(Witten and Frank, 2011) and sequential minimal optimization algorithm(SMO)(Platt, 1998).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

3 RESULTS OF THE APPLIED CLASSIFIERS USING FULL TRAINING DATASET

This study has used full KDD99 dataset and 13 supervised classifiers using Weka to train classifiers on full KDD99 dataset and to test trained classifiers on full test dataset of KDD99. These classifiers have been compared, considering hardware and binary classification metrics.

3.1 Comparison of Classifiers Regarding to Hardware Metrics

First, the classifiers have been compared with the following hardware metrics: Training Time, Training Memory, and Model Size.

Training time of classifiers can be seen in Table 4. Training time of ZeroR (1s) and OneR (67s) is lowest. Since these two classifiers are easy to train baseline classifiers, this short training time is expected. Training Time of other algorithms are mostly between Naive Bayes (75s) and Decision Table (106.75 min). Highest training times are MLP (3368 min) and SMO (1838 min); thus, these two classifiers are clearly outliers. Other interesting fact is training time of Random Forests is 11 minutes, while training time of Decision Trees (J48) is 40 minutes. Even though Random Forests seems to be more complex, its working principles —divide dataset and then train decision trees— makes it faster to train.

Classifiers and required memory information are given in Table 4. The minimum requirement for training memory is 3GB. But, some algorithms need as much as 14GB. The minimum amount of training memory is 3.067GB when Naive Bayes is used; while, the maximum is 13.935GB when Bayes Net is used. Out-of-place algorithms for training memory are ZeroR (7,248GB) and OneR (9,446 GB). These two algorithms are simpler than Naive Bayes; thus, their memory usage should be lower.

Most of the model sizes are fairly consistent, lower than 1 mb. Nonetheless, two models have very large sizes— BayesNet and RBF Network, 1.7 GB and 1.6 GB respectively. Large model size might be a serious restriction in low memory environments. Model size, memory requirements, and training time are not given generally in studies. Therefore, these results should help researchers who work with large data(millions of instances) using Weka regardless of applied domain.

3.2 Comparison of Classifiers Regarding to Binary Metrics

Binary classifier performance can be evaluated by using confusion matrix. Table 5 is a generic confusion matrix of the binary-attack-classification problem and common performance metrics (True Positive, True Negative ...) derived from that confusion matrix. All classifiers are compared using these metrics in Table 6 and Table 7.

Performance of classifiers on training dataset (Table 6) are all about %99. Such high results are due to overfitting; since, training and testing datasets are the same dataset in these experiments. Normally, Results in Table 6 are not given; since, only results on testing dataset should be presented in machine learning research. Yet, most of the results in literature are given for training dataset, instead of testing dataset(Table 1). With this in mind, we present Table 6 so that we can compare our results to previous studies.

ZeroR, the simplest baseline classifier,has detection rate of 80,14% on training dataset. Even though this result may seem high, it is normal since this is the distribution of attack instances to all instances (attack plus normal) in KDD99 training dataset. ZeroR always predicts majority class in instances; that being so, ZeroR is independent of size of the data set. ZeroR has Not Applicable (NA) rate for True Negative since it does not predict any negative class. Similarly, its rates for False Positive and False Negative are %100. OneR, the second baseline classifier, still achieves 98.81%; proving that, there are very similar records in KDD99(Tavallaee et al., 2009). All other results are above 99%, signalling overfitting on training dataset.

Table 7 contains results in testing dataset of KDD99. ZeroR has Detection Rate of 80,52% on test dataset —better than its result on training dataset. This result is due to the distributions of attacks to all instances (attack plus normal) in KDD99 testing dataset. Since the distribution of attack to normal in training data set is lower than that in testing data set, Detection Rate values of ZeroR on training data set is lower. OneR achieves 90% DR in Test dataset. It is such an interesting result since this result is higher than the other two more complex classifiers —RBFNetwork and Logistic Regression. Holte (Holte, 1993) demonstrates that simple classification rules perform very well in most datasets —KDD99 is not immune to this conclusion. Other results are very similar to each other, about 91%-92%. Best result is Decision Table (94.70%) while the second best is J48, Decision Tree (93,49%).

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

Comparison between results of the applied approach and previous studies can be seen in Table 8. Even though different metrics are provided before, Table 8 contains Detection Rate only. This is due to the fact that detection rate is the only consistent metric in previous studies.

Table 8 suggests that the applied approach to use all training data gives very good results. Except for some studies, results in Table 8 are higher than literature. Two classifiers, J48 and Naive Bayes, belong to same study (Benferhat et al., 2013) that use prior expert knowledge to increase detection rate.

4 DISCUSSION

Current study has used full training dataset of KDD99 to train 13 different machine learning algorithms and has tested them on full test dataset. The findings of this study can be summarized as follows:

1. Using more data to train batch learning algorithms needs more hardware resources, and this study gives more information about this resource usage.
2. Considering obtained results, using time-consuming-to-train classifiers on KDD99 is questionable. Relatively simple classifiers (Naive Bayes,Decision Tree,OneR) give comparable results; yet, they use much less computing resources.
3. Using full dataset for training algorithms increases train and test set detection rate, compared to previous studies, see Table 8.
4. Since our study is fully reproducible— with less than 15 minutes effort— it will be a comparison study for further studies that use KDD99.
5. As this study applied 13 classifier on the full training and test dataset of KDD99, improved results of this study can be a reference study for further studies in IDS or similar large datasets.

Table 8 indicates that our results highly exceeds literature. This result is not unexpected, as Domingos (Domingos, 2012) claims that "More Data Beats Cleverer Algorithm." In addition, numerous studies assert that more training examples increase test detection rate (Kalayeh and Landgrebe, 1983; Fukunaga and Hayes, 1989; Raudys and Jain, 1991; Cortes et al., 1994; Lenth, 2001; Last, 2007; Perlich, 2009; Halevy et al., 2009; Figueroa et al., 2012; Beleites et al., 2013).

Relationship between training data and test detection rate is hypothesized in Learning Curves (Perlich, 2009; Cortes et al., 1994). Learning Curves obey power law; as a result, small detection rate increases in the test set may be obtained using large amount of training data. For instance, an increase in test set detection rate from 0.91 to 0.92 may need 100.000 instances of new training data. According to Learning Curve hypothesis, using more training data converges test set detection rate to a point; for example, 0.85 or 0.91. This convergence point differs from dataset to dataset.

Learning Curves may be used to estimate training size requirements for desired test detection rate. For this study, best test detection rate is 94,70% for Decision Table. This shows that, using more data might increase test detection rate of other algorithms also; but, it might never go above 95.00%, since Training and Test Set have different probability distributions on KDD99.

As a final note, several limitations exist in this study. First, KDD99 dataset is very old; thus, its applicability to real world IDS is limited. However the focus of our study is especially about classification performance versus dataset size. Second, our study considers only metric detection rate when comparing classifiers; this is due to the fact that most of the previous studies report consistently only detection rate. Besides, our study provides other performance metrics. Third, our study does not consider rare attack performance.

5 CONCLUSION

This study proposed that using full training dataset instead of subset improves the performance of machine learning classifiers on IDS benchmark dataset KDD99. Using Weka, 13 machine learning classifiers have been trained with full KDD99 training dataset and tested on full dataset, Table 7. Results of the current study have been compared against previous studies. Additionally, the results also gave information about difficulties in training of large data in hardware metrics: training time, working memory, and model size. This study has found that generally using more training data brings performance benefits in test detection

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

rate in IDS domain, as predicted by Learning Curves. But some of the algorithms costs too much in the hardware resources while bringing less improvement. The findings of this study suggests that studies that are based on KDD99 should use more training data to obtain better results. KDD99 training requirements were given in this study; some of these requirements are significantly higher than those of in standard PCs, but these requirements are reachable nowadays. As the results shows, the finding of this study, using the most used machine learning methods, the full data set, and comparing the classifiers with respect to binary and hardware criteria, can be a reference study for further studies in IDS or similar large datasets.

REFERENCES

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., and Popp, J. (2013). Sample size planning for classification models. *Analytica Chimica Acta*, 760(0):25 – 33.

Benferhat, S., Boudjelida, A., Tabia, K., and Drias, H. (2013). An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge. *Applied Intelligence*, 38(4):520–540.

Brugger, S. (2007). Kdd cup 99 dataset (network intrusion) considered harmful.

Brugger, S. T. and Chow, J. (2005). An assessment of the darpa ids evaluation dataset using snort.

Chen, T., Zhang, X., Jin, S., and Kim, O. (2014). Efficient classification using parallel and scalable compressed model and its application on intrusion detection. *Expert Systems with Applications*, 41(13):5972 – 5983.

Chung, Y. Y. and Wahid, N. (2012). A hybrid network intrusion detection system using simplified swarm optimization (sso). *Applied Soft Computing*, 12(9):3014–3022.

Cortes, C., Jackel, L. D., Solla, S. A., Vapnik, V., and Denker, J. S. (1994). Learning curves: Asymptotic values and rate of convergence. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *NIPS 1994*, volume 6, pages 327–334. Morgan Kaufmann Publishers, Inc.

CSI (2011). 2010-2011 computer crime and security survey. Technical report, CSI.

Cunningham, R. K., Lippmann, R. P., Fried, D. J., Garfinkel, S. L., Graf, I., Kendall, K. R., Webster, S. E., Wyschogrod, D., and Zissman, M. A. (1999). Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB.

Denning, D. E. (1987). An intrusion-detection model. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 13(2):222–232.

Domingos, P. (2012). A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87.

Eesa, A. S., Orman, Z., and Brifcani, A. M. A. (2015). A new feature selection model based on id3 and bees algorithm for intrusion detection system. *Turk J Elec Eng & Comp Sci*, 23:615–622.

Elkan, C. (1999). Results of the kdd'99 classifier learning contest. `http://cseweb.ucsd.edu/users/elkan/clresults.html`.

Figueroa, R., Zeng-Treitler, Q., Kandula, S., and Ngo, L. (2012). Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12(1):8.

Fukunaga, K. and Hayes, R. R. (1989). Effects of sample size in classifier design. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(8):873–885.

Gentleman, R. and Temple Lang, D. (2007). Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, 16(1):1–23.

Gowrison, G., Ramar, K., Muneeswaran, K., and Revathi, T. (2013). Minimal complexity attack classification intrusion detection system. *Applied Soft Computing*, 13(2):921 – 927.

Guo, C., Zhou, Y., Ping, Y., Zhang, Z., Liu, G., and Yang, Y. (2014). A distance sum-based hybrid method for intrusion detection. *Applied Intelligence*, pages 1–11.

Halevy, A., Norvig, P., and Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90. 10.1023/A:1022631118932.

Horng, S.-J., Su, M.-Y., Chen, Y.-H., Kao, T.-W., Chen, R.-J., Lai, J.-L., and Perkasa, C. D. (2011). A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38(1):306 – 313.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

Kalayeh, H. M. and Landgrebe, D. A. (1983). Predicting the required number of training samples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(6):664–667.

KDDCup (1999). Kdd cup 1999 data task description. `http://kdd.ics.uci.edu/databases/kddcup99/task.html`.

Khor, K.-C., Ting, C.-Y., and Phon-Amnuaisuk, S. (2012). A cascaded classifier approach for improving detection rates on rare attack categories in network intrusion detection. *Applied Intelligence*, 36(2):320–329.

Koc, L., Mazzuchi, T. A., and Sarkani, S. (2012). A network intrusion detection system based on a hidden naïve bayes multiclass classifier. *Expert Systems with Applications*, 39(18):13492 – 13500.

Kumar, G. and Kumar, K. (2013). Design of an evolutionary approach for intrusion detection. *The Scientific World Journal*, 2013:14.

Last, M. (2007). Predicting and optimizing classifier utility with the power law. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 219–224.

Lee, W. and Stolfo, S. J. (2000). A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3:227–261.

Lenth, R. V. (2001). Some practical guidelines for effective sample size determination. *The American Statistician*, 55(3):187–193.

Lin, S.-W., Ying, K.-C., Lee, C.-Y., and Lee, Z.-J. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, 12(10):3285 – 3290.

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000). The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 34:579–595.

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*, 3(4):262–294.

Nguyen, H. and Choi, D. (2008). Application of data mining to network intrusion detection: Classifier selection model. In Ma, Y., Choi, D., and Ata, S., editors, *Challenges for Next Generation Network Operations and Service Management*, volume 5297 of *Lecture Notes in Computer Science*, pages 399–408. Springer Berlin / Heidelberg.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, 334(6060):1226–1227.

Perlich, C. (2009). Learning curves in machine learning. Technical report, IBM.

Pfahringer, B. (2000). Winning the kdd99 classification cup: bagged boosting. *SIGKDD Explor. Newsl.*, 1:65–66.

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. Technical report, Microsoft.

Raudys, S. and Jain, A. (1991). Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264.

Raymond, K.-K. and Choo (2011). The cyber threat landscape: Challenges and future research directions. *Computers and Security*, 30(8):719 – 731.

Scarfone, K. and Mell, P. (2007). *Guide to intrusion detection and prevention systems (IDPS)*, volume 800. NIST.

Sheikhan, M. and Sharifi Rad, M. (2013). Gravitational search algorithm–optimized neural misuse detector with selected features by fuzzy grids–based association rules mining. *Neural Computing and Applications*, pages 1–13.

Sindhu, S. S. S., Geetha, S., and Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*, 39(1):129 – 141.

Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 305–316, Washington, DC, USA. IEEE Computer Society.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, CISDA'09, pages 53–58.

Tavallaee, M., Stakhanova, N., and Ghorbani, A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE*

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

Transactions on, 40(5):516–524.
Vandewalle, P., Kovacevic, J., and Vetterli, M. (2009). Reproducible research in signal processing. *Signal Processing Magazine, IEEE*, 26(3):37–47.
wikipedia (2015). Reproducibility. `http://en.wikipedia.org/wiki/Reproducibility`.
Witten, I. and Frank, E. (2011). *Data Mining: Practical machine learning tools and techniques (Third Edition)*. Morgan Kaufmann Pub.
Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14:1–37.
Yi, Y., Wu, J., and Xu, W. (2011). Incremental svm based on reserved set for network intrusion detection. *Expert Systems with Applications*, 38(6):7698 – 7707.
Zeng, J., Liu, X., Li, T., Li, G., Li, H., and Zeng, J. (2011). A novel intrusion detection approach learned from the change of antibody concentration in biological immune response. *Applied Intelligence*, 35(1):41–62.
Zhang, J., Zulkernine, M., and Haque, A. (2008). Random-forests-based network intrusion detection systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(5):649 –659.
Özgür, A. and Erdem, H. (2016). A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ Preprints*.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

Classification Algorithm	Author(s) [Reference]	Detection Rate (Detection Rate)	Train Set Size instance number	Test Set Size instance number	Software Used
AdaBoost	Gowrison et al 2013 (Gowrison et al., 2013)	0.9844	6,983	6,983	Weka
AdaBoost	Zeng et al, 2011 (Zeng et al., 2011)	0.9004–0.9088	60,000–12,000	2650	NI
Bagged Boosting (KDD99 Winner)	Pfahringer 1999 (Pfahringer, 2000)	0.9271	485,178	311,029*	NI
Cascaded NB-J48	Khor et al 2012 (Khor et al., 2012)	0.9480	127,955	311,029*	NI
Decision Stump	Sindhu et al, 2012 (Sindhu et al., 2012)	0.7973	444,617	49,401	Weka
Decision Tree	Lin et al, 2012 (Lin et al., 2012)	0.9885	444,618	49,402	NI
Decision Tree	Benferhat et al, 2013 (Benferhat et al., 2013)	0.9341–0.9401	494,019	311,029*	NI
Decision Tree	Guo et al, 2014 (Guo et al., 2014)	0.9170	20,752	311,029*	NI
Decision Tree	Sindhu et al, 2012 (Sindhu et al., 2012)	0.9666	444,617	49,401	Weka
DSSVM (Distance Sum-based SVM)	Guo et al, 2014 (Guo et al., 2014)	0.9206	20,752	311,029*	NI
Elman NN	Sheikhan et al 2012 (Sheikhan and Sharifi Rad, 2013)	0.8790	49,402	31,103*	NI
Ensemble NN	Sindhu et al, 2012 (Sindhu et al., 2012)	0.9676	444,617	49,401	Weka
Hidden Naive Bayes (HNB)	Benferhat et al, 2013 (Benferhat et al., 2013)	0.9419–0.9506	494,019	311,029*	NI
Hidden Naive Bayes (HNB)	Koc et al, 2012 (Koc et al., 2012)	0.9372	444,617	49,401	Weka
HMM	Zheng et al, 2011 (Zeng et al., 2011)	0.9516–0.9600	60,000–12,000	2650	NI
K–Nearest Neighbor	Guo et al, 2014 (Guo et al., 2014)	0.9109	20,752	311,029*	NI
ID3-Bee	Eesa et al, 2015 (Eesa et al., 2015)	0.9314	4947	3117	C#
MLP	Gowrison et al, 2013 (Gowrison et al., 2013)	0.9390	6,983	6,983	Weka
MLP	Sheikhan et al, 2012 (Sheikhan and Sharifi Rad, 2013)	0.8000	49,402	31,103*	NI
Naive Bayes	Chung & Wahid 2012 (Chung and Wahid, 2012)	0.8680	4,000	4,000	NI
Naive Bayes	Zheng et al, 2011 (Zeng et al., 2011)	0.9681–0.9721	60,000–12,000	2650	NI
Naive Bayes	Benferhat et al, 2013 (Benferhat et al., 2013)	0.9345–0.9515	494,019	311,029*	NI
Naive Bayes	Guo et al, 2014 (Guo et al., 2014)	0.9148	20,752	311,029*	NI
NBTree	Sindhu et al, 2012 (Sindhu et al., 2012)	0.9227	444,617	49,401	Weka
Neuro Tree	Sindhu et al, 2012 (Sindhu et al., 2012)	0.9838	444,617	49,401	Weka java neurotree
NIDAAC	Zheng et al, 2011 (Zeng et al., 2011)	0.9562–0.9606	60,000–12,000	2650	NI
PSO	Chung & Wahid 2012 (Chung and Wahid, 2012)	0.8830	4,000	4,000	NI
Random Forests	Zhang et al, 2008(Zhang et al., 2008)	0.9808(a) 0.995(b) 0.9203(c)	494,020–60,620	60,620(a) 60,620(b) 311,029(c)*	Weka
Random Forests	Sindhu et al, 2012 (Sindhu et al., 2012)	0.8921	444,617	49,401	Weka
Random Tree	Sindhu et al, 2012 (Sindhu et al., 2012)	0.8898	444,617	49,401	Weka
Representative Tree	Sindhu et al, 2012 (Sindhu et al., 2012)	0.8911	444,617	49,401	Weka
RNN	Sheikhan et al, 2012 (Sheikhan and Sharifi Rad, 2013)	0.9410	49,402	31,103*	NI
Rule Based	Gowrison et al, 2013 (Gowrison et al., 2013)	0.999	6,983	6,983	Weka
SSO–WLS	Chung & Wahid, 2012 (Chung and Wahid, 2012)	0.9330	4,000	4,000	Weka
SVM	Chung & Wahid, 2012 (Chung and Wahid, 2012)	0.9218	4,000	4,000	Weka
SVM	Lin et al, 2012 (Lin et al., 2012))	0.9903	444,618	49,402	NI
SVM	Guo et al, 2014 (Guo et al., 2014)	0.9137	20,752	311,029*	NI
Tree Augmented Naive Bayes (TAN)	(Benferhat et al, 2013 (Benferhat et al., 2013)	0.9436–0.9740	494,019	311,029*	NI

Table 1. Classifiers Results from Literature (***** means KDD99 original test dataset used, **NI** means No Information)

Dataset Type	Attack	Normal	Total
Training	3,925,650 (80.15%)	972,781 (19.85%)	4,898,431
Test	250,436 (80.52%)	60,593 (19.48%)	311,029

Table 2. KDD99 Properties Attack/Normal

Easy to Use	Most Used	Easy to Understand	Hard to Train
ZeroR	Adaboost (Zeng et al., 2011; Gowrison et al., 2013)	BayesNet (Nguyen and Choi, 2008)	MLP (Gowrison et al., 2013; Sheikhan and Sharifi Rad, 2013)
OneR (Nguyen and Choi, 2008)	Decision Tree (Nguyen and Choi, 2008; Sindhu et al., 2012; Guo et al., 2014; Khor et al., 2012; Lin et al., 2012; Benferhat et al., 2013)	Decision Table (Nguyen and Choi, 2008)	RBF
	Naive Bayes (Guo et al., 2014; Nguyen and Choi, 2008; Zeng et al., 2011; Benferhat et al., 2013)	Decision Tree (Nguyen and Choi, 2008; Sindhu et al., 2012; Guo et al., 2014; Khor et al., 2012; Lin et al., 2012; Benferhat et al., 2013)	SVM (Chung and Wahid, 2012; Lin et al., 2012; Guo et al., 2014)
	Random Forest (Zhang et al., 2008; Sindhu et al., 2012)	Logistic Regression	

Table 3. Four Groups of Applied Classifiers

Classifier Name	Working Memory Giga Bytes	Training Time Minutes	Model Sizes Bytes	Model Sizes Mega Bytes
Logistic	7,233	10.37	39,661	0,038
BayesNet	13,935	5.22	1,787,990,728	1,705,161
NaiveBayes	3,067	1.80	14,936	0,014
SMO	7,724	1,128.90	40,124	0,038
J48	4,573	30.88	142,345	0,136
MLP	8,861	722.78	68,035	0,065
RBFNetwork	13,381	8.52	1,748,805,636	1,667,791
AdaBoostM1	5,549	12.87	9,714	0,009
SGD	12,253	34.58	38,225	0,036
DecisionTable	8,461	41.42	443,696	0,423
OneR	9,446	0.42	1,767	0,002
RandomForest	9,015	11.98	5,0012,917	47,696
ZeroR	7,248	0.02	1,110	0,001

Table 4. Classifiers Training Information

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

		Actual	
		Attack	Normal
Predicted	Attack	**T**rue **P**ositives	**F**alse **P**ositives
	Normal	**F**alse **N**egatives	**T**rue **N**egatives

Actual Number Performance Metrics		Rate Performance Metrics	
True **P**ositive	How many actual attacks classifier predicted as true attack (number).	tp rate	$\frac{TP}{TP+FN}$
True **N**egative	How many actual normal instances classifier predicted as normal(number).	fp rate	$\frac{FP}{FP+TN}$
False **P**ositive	How many actual normal instances classifier predicted as attack, in other words false alarm (number)	precision	$\frac{TP}{TP+FP}$
False **N**egative	How many actual attacks classifier predicted as normal, that is missed an attack (number)	recall	$\frac{TP}{TP+FP}$
		Detection Rate (Accuracy)	$\frac{TP+TN}{TP+FP+FN+TN}$
		F-Measure	$\frac{2TP}{2TP+FP+FN}$

Table 5. Confusion Matrix for Binary Attack Classification

Algorithms	**Detection Rate**	**True Positive Rate**	**True Negative Rate**	**False Positive Rate**	**False Negative Rate**	**precision**	**F1-Rate**
AdaBoostM1	99,20%	99,46%	97,82%	1,86%	1,86%	99,54%	99,50%
BayesNet	99,64%	99,55%	98,21%	0,01%	0,01%	99,99%	99,77%
DecisionTable	99,99%	99,99%	99,96%	0,03%	0,03%	99,99%	99,99%
J48	99,99%	99,99%	99,99%	0,01%	0,01%	99,99%	99,99%
Logistic	99,48%	99,76%	99,01%	1,63%	1,63%	99,60%	99,68%
MLP	99,92%	99,91%	99,62%	0,02%	0,02%	99,99%	99,95%
NaiveBayes	99,19%	99,24%	97,00%	1,00%	1,00%	99,75%	99,50%
OneR	98,81%	99,96%	99,84%	5,82%	5,82%	98,58%	99,26%
RandomForest	99,99%	99,99%	99,99%	0,01%	0,01%	99,99%	99,99%
RBFNetwork	99,36%	99,32%	97,31%	0,46%	0,46%	99,88%	99,60%
SGD	99,90%	99,91%	99,63%	0,13%	0,13%	99,97%	99,94%
SMO	99,88%	99,87%	99,48%	0,10%	0,10%	99,97%	99,92%
ZeroR	80,14%	100,00%	NA	100,00%	100,00%	80,14%	88,98%

Table 6. Training Set Base Metrics

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017

Algorithms	Detection Rate	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	precision	F1-Rate
AdaBoostM1	91,49%	90,11%	70,39%	2,80%	2,80%	99,25%	94,46%
BayesNet	91,61%	89,78%	70,13%	0,82%	0,82%	99,78%	94,52%
DecisionTable	94,70%	93,98%	79,69%	2,32%	2,32%	99,41%	96,61%
J48	93,49%	92,03%	75,13%	0,49%	0,49%	99,87%	95,79%
Logistic	81,52%	77,68%	51,35%	2,62%	2,62%	99,19%	87,13%
MLP	91,82%	90,23%	70,90%	1,60%	1,60%	99,57%	94,67%
NaiveBayes	91,47%	89,95%	70,18%	2,25%	2,25%	99,40%	94,44%
OneR	90,74%	88,81%	68,10%	1,27%	1,27%	99,66%	93,92%
RandomForest	92,43%	90,85%	72,34%	1,03%	1,03%	99,73%	95,08%
RBFNetwork	85,28%	82,11%	57,09%	1,62%	1,62%	99,53%	89,98%
SGD	92,29%	90,82%	72,16%	1,63%	1,63%	99,57%	94,99%
SMO	91,92%	90,36%	71,16%	1,65%	1,65%	99,56%	94,74%
ZeroR	80,52%	100,00%	NA	100,00%	100,00%	80,52%	89,21%

Table 7. Test Set Base Metrics

Classification Algorithm	DR Literature on Training	DR Literature on Testing	DR Train	DR Test
AdaBoostM1	0.9844(Gowrison et al., 2013)	NA	0.9920 ↑	0.9149
BayesNet	0.9062(Nguyen and Choi, 2008)	NA	0.9964 ↑	0.9161 ↑
DecisionTable	0.9166(Nguyen and Choi, 2008)	NA	0.9999 ↑	0.9470 ↑
J48	0.9885(Lin et al., 2012)	0.9401(Benferhat et al., 2013)	1.0000 ↑	0.9349 ↓
Logistic Regression	NA	NA	0.9948	0.8152
MLP	0.9390(Gowrison et al., 2013)	0.8000(Sheikhan and Sharifi Rad, 2013)	0.9992 ↑	0.9182 ↑
NaiveBayes	0.9721 (Zeng et al., 2011)	0.9515(Benferhat et al., 2013)	0.9919 ↑	0.9147 ↓
OneR	0.8931(Nguyen and Choi, 2008)	NA	0.9881 ↑	0.9074
RandomForest	0.9808(Zhang et al., 2008)	0.9203(Zhang et al., 2008)	0.9999 ↑	0.9243 ↑
RBFNetwork	NA	NA	0.9936 ↑	0.8528
SGD	NA	NA	0.9990	0.9229
SMO	0.9903(Lin et al., 2012)	0.9137(Guo et al., 2014)	0.9988 ↑	0.9192 ↑
ZeroR	NA	NA	0.8014	0.8052

Table 8. Comparison of Applied Classifiers with Previous Studies using Detection Rate

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2838v1 | CC BY 4.0 Open Access | rec: 1 Mar 2017, publ: 1 Mar 2017