*I. J. Computer Network and Information Security*, **2019, 4, 43-52**
Published Online April 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2019.04.06

# Intrusion Detection using Machine Learning and Feature Selection

**Prachi, Heena Malhotra**
The NorthCap University, Gurgaon, India
E-mail: {prachiah1985, malhotraheena17}@gmail.com

**Prabha Sharma**
The NorthCap University, Gurgaon, India
E-mail: prabhasharma@ncuinda.edu

Received: 13 February 2019; Accepted: 27 February 2019; Published: 08 April 2019

***Abstract*—**Intrusion Detection is one of the most common approaches used in detecting malicious activities in any network by analyzing its traffic. Machine Learning (ML) algorithms help to study the high dimensional network traffic and identify abnormal flow in traffic with high accuracy. It is crucial to integrate machine learning algorithms with dimensionality reduction to decrease the underlying complexity of processing of huge datasets and detect intrusions within real-time. This paper evaluates 10 most popular ML algorithms on NSL-KDD dataset. Thereafter, the ranking of these algorithms is done to identify best performing ML algorithm on the basis of their performance on several parameters such as specificity, sensitivity, accuracy etc. After analyzing the top 4 algorithms, it becomes evident that they consume a lot of time while model building. Therefore, feature selection is applied to detect intrusions in as little time as possible without compromising accuracy. Experimental results clearly demonstrate that which algorithm works best with/without feature selection/reduction technique in terms of achieving high accuracy while minimizing the time taken in building the model.

***Index Terms*—**Network, Intrusion, Machine Learning, NSL-KDD Dataset, Feature Selection.

## I. Introduction

Huge technological advancements in the field of communication industry massively increased the volume of data and its transmission across the globe via the internet. However, such advancements put valuable information and data at risk [1]. In today's era, intrusion happens within a few seconds. This gives rise to the need for a stronger security system. An Intrusion Detection System (IDS) [2] analyzes the network traffic to identify malicious actions. Currently available IDS are divided in 2 major categories [3], namely, anomaly and misused based detection. Misuse detection identifies an intrusion on the basis of already known patterns, popularly called as signatures. Therefore, misuse detection is also referred as signature-based IDS (e.g. Snort [4]). Anomaly detection [5] identifies any unacceptable deviation from normal traffic. Unlike signature-based IDS, anomaly detection identifies zero-day attacks but generates a large number of false alarms. It also faces many challenges while dealing with huge amount of high-dimensional data.

In order to analyze huge volumes of data, most of the existing IDS use Machine Learning (ML) algorithms to identify intrusions in an efficient manner. Although many techniques are available for detection purposes, quite a few are effective in producing high accuracy and low false positives for a huge amount of data [6]. Also, some ML algorithms perform better than others in terms of accuracy but take more training time for building models on large datasets. Hence, this results in an imminent need of consolidating ML algorithms with feature selection/reduction to obtain an accurate classification of reduced dimensional data while taking lesser time in building the model.

An ideal IDS should be able to spot zero-day attacks with high accuracy and low false positives quickly so that intrusions can be prevented as early as possible [7]. Consequently, the Objective behind this paper is to design an intrusion detection model that integrates ML algorithms with the feature selection and feature reduction methods to detect intrusions with high accuracy and low false positives within a short span of time.

This paper evaluates the performance of 10 most popular ML algorithms in WEKA [8] using NSL-KDD dataset [9]. Thereafter, algorithms are ranked based on their performances on certain parameters such as specificity, sensitivity, accuracy, the time taken in building the model, etc. To achieve high accuracy, less false alarms and minimum training time on large data sets, this paper applies dimensionality reduction methods on the best 4 ML algorithms. Later on, the performance of these best 4 ML algorithms is evaluated with/without applying feature selection/reduction methods in order to build an ideal model for intrusion detection.

The organization of this paper is as follows. Work related to intrusion detection is highlighted in Section II.

Section III briefly explains the data set and tool used in this paper. Algorithms related to machine learning and feature selection are discussed in section IV. Section V describes the performance metrics used in evaluating the algorithms. Section VI explains the implementation and analyses the results of the experiments performed. Paper is concluded in section VII.

## II. Related Works

Numerous papers are presented in the literature to discuss and implement various aspects of IDS. In this section, we describe some of the significant literature work.

The first notion of intrusion detection was given in 1980 by James P. Anderson [10]. A model was proposed to keep a security watch on user behavior to detect anomalies. Lee et al. in 1998 [11] designed a framework using data mining techniques for detecting intrusions. Another framework was proposed by Schultz et. al. in 2001 [12] where multiple classifiers are trained on the data set of benign and malicious executable to detect new cases of intrusions. The 3-tier architecture of IDS was proposed by Hwang et. al. in 2007 [13] which consists of three lists, namely, whitelist, blacklist and multi-class representing the normal traffic, known attacks from the traffic and abnormalities that were detected in normal traffic respectively.

In 2009, Tavallaee et. al. [14] studied each feature in KDD'99 dataset. In the same year, Srinivasulu et. al. [15] performed evaluations on the confusion matrix of Naïve Bayesian, CART and Artificial Neural Network. In 2011, Reddy et. al. [16] presented a survey of different techniques in IDS. In 2012, Nadiammai and Hemalatha [17] evaluated the effectiveness of all classifiers on the basis of time, accuracy, error, specificity and sensitivity. Neethu [18], in 2012, gave an IDS framework comprising of Naïve Bayes and PCA classifier.

In 2013, Revathi et al. [19] identified the best classifier in terms of performance and accuracy. In 2015, Sumouli Choudhary et. al. [20] presented a comparative analysis of machine learning algorithms such as Logistic, BayesNet, Random Forest, IBK, PART, J48, Random Tree and JRip for network intrusion detection. In 2016, Murthy et. al. [21] compared the performances of 4 classifiers, namely, Naïve Bayes, J48, OneR and RandomTree and used the best classifier to select important features and build a model with better accuracy.

Latha et al. [22] proposed feature selection algorithm for intrusion detection. However, authors used KDD Cup'99 which suffers from lots of drawbacks, mentioned in the next section.

Biswas S. K. compared 5 classifiers and feature selection techniques and concluded that kNN classifier performed better in comparison to other classifiers [23].

In most of the above-mentioned methods, the time taken in building the model is not taken into consideration. However, time is an important parameter for predicting intrusion in real world scenarios. Therefore, while building an ideal model, we consider model building time as an important aspect whilst achieving high accuracy and low false alarms.

Consequently, the motivation of this paper is to design a model that analyzes high dimensional network traffic to identify intrusion within real-time with high accuracy. Machine learning and feature selection techniques are used in a 2-step process in order to attain this objective. Authors have taken special care about the model building time parameter while selecting the best candidate for task.

## III. Description of the Dataset and Tool Used

In literature, the KDD Cup'99 dataset is extensively used by researchers. However, Tavallaee et al. [14] highlighted that KDD Cup'99 suffers from many drawbacks. According to them, the KDD Cup'99 dataset was built using a closed network and suffer from duplicate data entries, hand injected attacks and non-validation. NSL-KDD dataset [9] has emerged from KDD Cup'99 dataset and provides solution to abovementioned problems. It has many advantages over KDD dataset in terms of duplicate data entries and better detection rates. These advantages make the NSL-KDD dataset much better than original KDD dataset.

NSL-KDD dataset divides the network traffic as normal and anomaly. The entire dataset consists of 41 attributes along with 1 class label attribute and 125973 instances. It is divided into training set (80%) and test dataset (20%).

Performance of different ML algorithms is evaluated on NSL-KDD dataset using WEKA tool [24]. WEKA comprises of ML algorithms used for performing various data mining tasks. Earlier, it was coded in C language but later, it was revised and written again in JAVA. The algorithms available in WEKA can be applied directly to the datasets. It also provides a visualization tool for analyzing the results. WEKA is distributed under the GNU General Public License [8]. WEKA GUI chooser provides 4 options:

### A. *Explorer*

It is an environment containing several panels for exploring data.

### B. *Experimenter*

It allows the user to create, run and analyze the experiment and conduct statistical test between learning schemes.

### C. *Knowledge Flow*

It is an interface based on Java-beans used for setting and running different machine learning algorithms.

### D. *Simple CLI*

It allows direct execution of WEKA commands by providing a simple command line interface.

## IV. ML Algorithms and Feature Selection/Reduction Methods

### *A. ML Algorithms*

ML algorithms help in analyzing a large amount of network traffic in an efficient manner. Out of the several algorithms available in WEKA, this paper evaluates 10 algorithms that are compatible with NSL-KDD dataset. These algorithms are described in brief below:

BayesNet – It acts as a foundation for a Bayes Network classifier and constructs a Bayesian Network [25] by determining conditional probability on every node. It draws a model and represents it graphically using a directed acyclic graph by depicting random variables along with their conditional dependencies.

Naïve Bayes – Naive Bayes is a simple but powerful algorithm for predictive modeling. In machine learning, Naive Bayes classifiers comprise of simple probabilistic classifiers when Bayes' theorem is applied with strong independence assumptions among the features. The representation for Naïve Bayes is done using probabilities.

RandomTree – It is an algorithm for tree generation and uses a certain number of attributes at each node of the decision tree. Classification is quick once rules are designed. It does not perform any pruning. It performs well on large datasets. However, it ignores the correlation among attributes.

RandomForest – RandomForest [26] is an ensemble learning method works by combining many decision trees at training time and delivers output by constructing separate trees. This classifier has higher accuracy as it produces low classification errors.

J48 – It is an improved version of C4.5 algorithm based on the notion of information entropy [27]. It constructs decision trees from the training data and then applies a heuristic criterion. This technique consumes lots of space and time.

Bagging – Bagging [28], also known as Bootstrap aggregating, is an ensemble meta-algorithm developed for enhancing the stability and precision of machine learning algorithms by combining models of a similar type. Bagging helps in reducing variance and avoids overfitting.

PART – It is based on separate-and-conquer approach for constructing a partial C4.5 decision tree in every iteration and takes the "best" leaf into a rule. It basically performs instance based learning by building the tree using heuristics of C4.5 with similar parameters as defined in *J48*.

OneR – It is the 1R classifier based on one parameter, which is the required minimum bucket size for performing discretization. It may use internal cross validation (the no. of folds being a parameter) or the training data for evaluation.

ZeroR - It is the simplest classification method which makes a table of frequency for the target and selects its most recurring value. Having no foretelling power, ZeroR is useful in deciding the baseline performance. It has the overfitting problem.

Logistic - It is another way for constructing and using a multinomial logistic regression model with a ridge estimator to protect itself from overfitting by imposing large coefficients [29]. It is easy to understand and implement but suffers from the problem of overfitting.

### *B. Feature Selection/Reduction Methods*

Feature selection/reduction is a methodology in which a subset of features is selected by removing surplus features for generating an accurate learning model [30] while reducing the model building time and complexity. Two methods for feature selection and reduction [31] are:

Wrapper Method: In this method, feasible subsets are created with the help of subset evaluator. Different classifiers are generated using a classification algorithm and features of every subset to find out which subset of features performs the best with the classification algorithm.

Filter Method: In this method, instead of taking selected attributes, ranks are assigned to all the attributes in the dataset by using an attribute evaluator and a ranker method. The attribute ranked first has the highest priority. Features having lower rank are omitted one at a time to evaluate the accuracy of the classifier at that point of time. The number of features can be omitted one after the other till the global minimum is reached. Beyond global minima, the dataset will start over-fitting and generating more number of incorrectly classified instances.

Feature/Attribute Discretization: Discretization is the process of transforming numeric data into nominal data, by dividing attribute's numeric values into a number of intervals. Thus, discretization combines the attribute values as per those interval values and helps in reducing the learning complexity of the classifier with which it is used. In order to do that, the data is discretized by applying unsupervised discretize filter to attributes.

## V. Performance Measures

Performance parameters such as accuracy, specificity, sensitivity, ROC area, the time taken to build model etc. are used for evaluating and comparing the performance of different classifiers. Description of all these parameters is discussed below:

Confusion Matrix: A confusion matrix is a visualization tool that acts as a basis for calculating all other parameters. The confusion matrix comprises of 4 values, namely, TP, FP, FN and TN as shown in Table 1.

Table 1. Confusion Matrix

| | | Predicted | |
|---|---|---|---|
| Actual | | Normal | Anomaly |
| | Normal | TP | FN |
| | Anomaly | FP | TN |

The above parameters are described in brief below:

*True Positive (TP)* - It denotes the normal instances that are identified correctly.

*False Negative (FN)* - It points out the abnormal instances as normal incorrectly i.e. identifies instances as normal which are attacks in reality.

*False Positive (FP)* - It indicates anomaly instances that are identified as normal incorrectly.

*True Negative (TN)* - It indicates anomaly instances that are identified rightly as an attack.

**Accuracy:** It indicates the total number of correct predictions. It can be determined from Table 2 using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (1)$$

**Specificity:** It is the measure of actual negatives that are identified correctly. It can be measured from Table 2 using the following formula:

$$Specificity = \frac{TN}{FP+TN} \quad (2)$$

**Sensitivity:** It is the measure of actual positives that are identified correctly. It can be determined from Table 2 using the following formula:

$$Sensitivity = \frac{TP}{TP + FN} \quad (3)$$

**ROC (Receiver Operating Characteristics):** It is a graphical way of evaluating machine learning schemes. ROC curves outline the performance without any reference to class distribution. On the vertical axis, the true positive rate is plotted and on the horizontal axis, the true negative rate is plotted. Larger the area of the curve, better the model and greater is the value of ROC.

**TP Rate:** It points out the possibility of an algorithm to foretell positive instances as correct and normal. A high TP is desirable and can be calculated as mentioned below:

$$TP\ rate = \frac{TP}{TP + FN} \quad (4)$$

**FP Rate:** It signifies the possibility of an algorithm to forecast normal instances as an attack. A low FP is desirable and can be calculated as mentioned below:

$$FP\ rate = \frac{FP}{FP + TN} \quad (5)$$

**Correctly and Incorrectly classified instances:** number of instances which are correctly and incorrectly identified.

**Time taken to build model:** time taken by the classifier to generate the model, measured in seconds. Lesser the time taken in building the model, better is the classifier chosen.

## VI. Result Analysis

### A. Classifier Comparison

The above discussed classifiers are implemented with the NSL-KDD training dataset comprising of 125973 instances and 42 attributes using 10 fold cross-validation. 10 most popular algorithms compatible with our dataset are chosen for the experimental purpose.

Table 2 illustrates the performance on the basis of correctly and incorrectly classified instances, TP Rate, FP Rate, ROC, Specificity, Sensitivity, Accuracy and the time taken in building the model. Table 2 helps in predicting which algorithm based classifier is statistically significant. Based on the values obtained in classifying the instances correctly, the highest accuracy (99.91%) is achieved by Random Forest. It also produces a high TP rate and low FP positives as desirable. Moreover, it has the highest specificity and sensitivity percentage when compared to other algorithms. However, it also takes the largest time in building the model which is 191 sec.

It is clear from Table 2 that the second highest accuracy, second highest TP rate and second lowest FP rate is produced by Bagging. Moreover, it has the second highest sensitivity percentage. However, it also occupies second place in consuming more time for building the model.

PART algorithm achieves third highest accuracy, second highest specificity percentage, third highest TP rate and third lowest FP rate. However, it also consumes the third highest time in building the model.

J48 occupies the fourth position in detection accuracy (99.78%), fourth highest TP rate and fourth lowest FP rate. It has also got the fourth highest specificity percentage. Despite its proficiency in achieving high accuracy, it takes more model building time which is approx. 62 sec.

For detecting intrusions in high-speed network, time plays a crucial role. The aforementioned classifiers take longer model building time while achieving higher accuracy. Although ZeroR is the most time efficient classifier out of all 10 classifiers as it consumes minimum time in the model building but it offers the least amount of accuracy percentage (53.45%), TP rate and highest FP rate. So, it cannot be considered as a good classifier.

Thus, it becomes clear from Table 2 that if feature selection/reduction method is not applied, then out of all evaluated algorithms, Random Tree is the algorithm that achieves good amount of accuracy (99.76%), high TP rate and low FP rate whilst consuming lesser time in building the model (3.24 sec.) in comparison with other algorithms.

Thereafter, the ranking of the aforementioned algorithms has been done in Table 3 on the basis of their performance. In case, the values of a set of classifiers are same or very close on a set of parameters then the classifier in the set with smaller building time will be ranked lower. In our ranking system, the classifier with lower rank is preferred over another with higher rank. The classifier with rank 1 is considered as best and the classifier with rank 10 is considered as worst.

Table 3 clearly shows that Random Forest Classifier outperforms all other classifiers on all of the performance parameters while ZeroR classifier performs worst on our chosen dataset and occupies the last position in the ranking table. Other classifiers such as Bagging, PART, J48 and Random Tree have managed to secure their position in the top 5 classifiers. The other 4 classifiers, namely, Naïve Bayes, Bayes Net, Logistic and OneR, come under the bottom 5 categories of classifiers.

Table 2. Performance of Different Classifiers using 10-fold cross Validation

| Classifiers → Parameters ↓ | **Naïve Bayes** | **Bayes Net** | **Logistic** | **Random Tree** | **Random Forest** | **J48** | **Bagging** | **OneR** | **PART** | **ZeroR** |
|---|---|---|---|---|---|---|---|---|---|---|
| Correctly Classified Instances | 113858 (90.38%) | 122409 (97.17 %) | 122329 (97.10 %) | 125678 (99.76%) | 125869 (99.91%) | 125698 (99.78%) | 125776 (99.84%) | 121406 (96.37%) | 125769 (99.83%) | 67343 (53.45 %) |
| Incorrectly Classified Instances | 12115 (9.61%) | 3564 (2.89%) | 3644 (2.89% ) | 295 (0.23%) | 104 (0.08%) | 275 (0.218) | 197 (0.15%) | 4567 (3.62%) | 204 (0.16%) | 58630 (46.54 %) |
| TP Rate | 0.904 | 0.972 | 0.971 | 0.998 | 0.999 | 0.998 | 0.998 | 0.964 | 0.998 | 0.535 |
| FP Rate | 0.101 | 0.032 | 0.029 | 0.002 | 0.001 | 0.002 | 0.002 | 0.033 | 0.002 | 0.535 |
| ROC Area | 0.966 | 0.997 | 0.993 | 0.998 | 1.000 | 0.999 | 1.000 | 0.965 | 0.999 | 0.500 |
| Specificity (%) | 86.8 | 94.6 | 96.81 | 99.73 | 99.86 | 99.77 | 99.80 | 98.8 | 99.81 | 0 |
| Sensitivity (%) | 93.6 | 99.3 | 97.3 | 99.79 | 99.96 | 99.78 | 99.87 | 94.1 | 99.85 | 1.0 |
| Accuracy (%) | 90.38 | 97.17 | 97.1 | 99.76 | 99.91 | 99.78 | 99.84 | 96.3 | 99.83 | 53.4 |
| Time taken to build model (in sec.) | 2.88 | 14.02 | 87.44 | 3.24 | 191.06 | 61.68 | 109.9 | 2.66 | 99.1 | 0.13 |

Table 3. Ranking of Different Classifiers based on Different Parameters

| Classifiers → Parameters ↓ | **Naïve Bayes** | **Bayes Net** | **Logistic** | **Random Tree** | **Random Forest** | **J48** | **Bagging** | **OneR** | **PART** | **ZeroR** |
|---|---|---|---|---|---|---|---|---|---|---|
| Correctly Classified Instances | 9 | 6 | 7 | 5 | 1 | 4 | 2 | 8 | 3 | 10 |
| Incorrectly Classified Instances | 9 | 6 | 7 | 5 | 1 | 4 | 2 | 8 | 3 | 10 |
| TP Rate | 9 | 6 | 7 | 5 | 1 | 4 | 2 | 8 | 3 | 10 |
| FP Rate | 9 | 7 | 6 | 5 | 1 | 4 | 2 | 8 | 3 | 10 |
| ROC Area | 8 | 6 | 7 | 5 | 1 | 4 | 2 | 9 | 3 | 10 |
| Specificity | 9 | 8 | 7 | 5 | 1 | 4 | 3 | 6 | 2 | 10 |
| Sensitivity | 9 | 6 | 7 | 4 | 1 | 5 | 2 | 8 | 3 | 10 |
| Accuracy | 9 | 6 | 7 | 5 | 1 | 4 | 2 | 8 | 3 | 10 |

It is clear from the ranking in Table 3 that Random Forest, Bagging, PART and J48 are best 4 classifiers. However, all these algorithms consume huge time in building the model. Therefore, in the next section these algorithms are further evaluated using feature selection and feature reduction methods to determine whether it is possible to reduce model building time for these algorithms while achieving high accuracy and low FP rate.

### *B. Performance Evaluation using Feature Selection and Feature Reduction Method*

Selection of features to reduce unwanted features helps in increasing the efficiency of the classifier and building an effective model. Therefore, the top 4 classifiers are now evaluated using feature selection and feature reduction. In this paper, two combinations of feature selection methods are tried: Wrapper method (CfsSubsetEval + BestFirst) and Filter method (InfoGainAttributeEval + Ranker). The details of the features selected by each combination are described in Table 4.

Table 4. Feature Selection by using Different Methods

| **Attribute Evaluator + Search Method** | **Feature Selected** | **Method Used** |
|---|---|---|
| CfsSubsetEval + BestFirst | 4,5,6,12,26,30 | Wrapper Method |
| InfoGainAttribute Eval + Ranker | 5,3,6,4,30,29,33,34,35,38,12,39,25,23,26,37,32,36, 31,24,41,2,27,40,28,1,10, 8,13,16,19,22,17,15,14,18,11, 7,21,20,9 | Filter Method |

Wrapper based feature selection method is applied to select the best subset of features. With NSL-KDD dataset, Wrapper method selected optimal subset (Table 4) as 4, 5, 6, 12, 26, 30 features. Table 4 also shows the ranking of features by the filter method. In this paper, ranker is run on the dataset and low ranked features are omitted one by one until the overfitting problem occurs. If the feature/attribute is further removed, the model starts to overfit and the percentage of correctly classified instances begins to decrease. So, in filter method features are removed one after the other till global minima is achieved. In NSL-KDD dataset, global minimum is achieved when we are left with top 10 features 5,3,6,4,30,29,33,34,35,38 for detecting attacks.

Performance of 4 top ranked classifiers is evaluated after removing all redundant features as suggested by wrapper and filter methods and results are tabulated in Table 5-8. Moreover, discretize filter is also used individually and in conjunction with filter and wrapper method to evaluate the performance of those classifiers.

Table 5 shows the evaluation of J48 classifier based on different methods. It is observed from the table that the accuracy of J48 was initially 99.78% but model building time was quite high. However, after applying the different combination of feature selection/reduction techniques, the time taken in building the model is reduced significantly but with compromising the accuracy. In fact, in case of filter method, J48 achieves maximum accuracy i.e. 99.79% while the time taken in building the model is 17% of the time taken by the J48 when executed without applying feature selection/reduction methods. Other methods are also able to achieve nearly equivalent amount of accuracy while reducing the model building time significantly. Fig. 1 clearly demonstrates the accuracy of the J48 algorithm with different feature selection/reduction methods.

Table 6 shows the evaluation of Random Forest classifier using different feature selection and reduction methods. It is evident from the table that even after applying wrapper or filter method, the time taken in building the model is still very high. However, model building time reduces significantly either when discretize filter is used individually with the classifier or in conjugation with filter or wrapper method. The accuracy achieved with the help of discretize filter is similar to the accuracy achieved while running Random Forest individually but the time taken in building the model is 15.5% of actual.

So, Random Forest classifier achieves the highest accuracy without reducing any features but requires a lot of model building time. On the other hand, performing detection with the most important features results in saving a lot of computation time and helps in building high speed network intrusion detection.

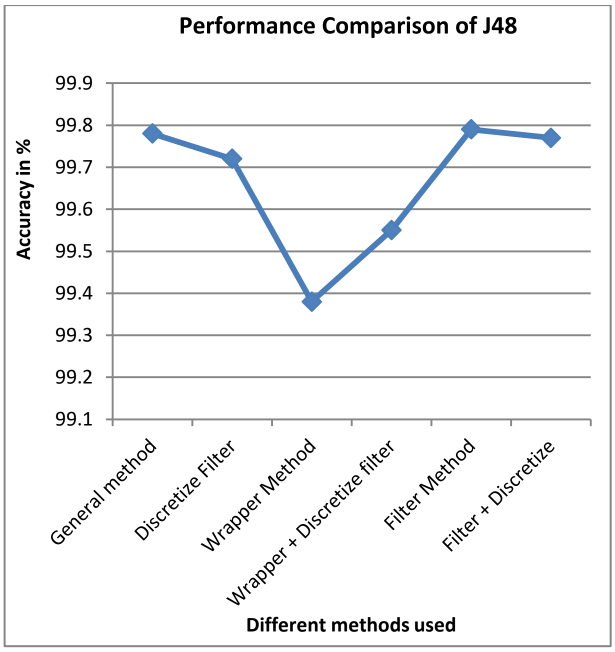

Fig.1. Performance Comparison of J48 based on Different Methods

Fig. 2 illustrates the performance of Random Forest classifier in terms of accuracy when used with different feature selection/reduction techniques.

Table 7 shows the evaluation of Bagging classifier based on different methods. The accuracy of Bagging was initially 99.84% but model building time was quite high. However, when used with the filter method, the time required in building the model is only 20% of the actual whilst achieving similar accuracy. Discretize filter based methods are also able to achieve nearly equivalent amount of accuracy while reducing the model building time significantly. Fig. 3 shows the accuracy of Bagging using different methods.

Table 5. Evaluation of J48 Classifier based on Different Methods

| **Methods →** <br> **Parameters ↓** | **General method** | **Discretize Filter classifier** | **Wrapper method** | **Wrapper method + Discretize filter** | **Filter method** | **Filter method + Discretize filter** |
|---|---|---|---|---|---|---|
| Correctly Classified Instances | 125698 (99.78%) | 125621 (99.72%) | 125200 (99.38%) | 125412 (99.55%) | 125720 (99.79%) | 125689 (99.77%) |
| Incorrectly Classified Instances | 275 (0.218%) | 352 (0.279%) | 773 (0.613%) | 561 (0.445%) | 253 (0.200%) | 284 (0.225%) |
| Model Building Time | 61.68 sec. | 5.75 sec. | 7.95 sec. | 9.25 sec. | 10.45 sec. | 10.85 sec. |

Table 6. Evaluation of Random Forest Classifier based on Different Methods

| **Methods →** **Parameters ↓** | **General method** | **Discretize Filter classifier** | **Wrapper method** | **Wrapper method + Discretize filter** | **Filter method** | **Filter method + Discretize filter** |
|---|---|---|---|---|---|---|
| Correctly Classified Instances | 125869 (99.91%) | 125861 (99.90%) | 125238 (99.41%) | 125485 (99.61%) | 125818 (99.87%) | 125842 (99.89%) |
| Incorrectly Classified Instances | 104 (0.082%) | 112 (0.088%) | 735 (0.583%) | 488 (0.387%) | 155 (0.123%) | 131 (0.104%) |
| Model Building Time | 191.06 sec. | 30.34 sec. | 66.82 sec. | 14.89 sec. | 69.55 sec. | 20.06 sec. |

Table 7. Evaluation of Bagging Classifier based on Different Methods

| **Methods →** **Parameters ↓** | **General method** | **Discretize Filter classifier** | **Wrapper method** | **Wrapper method + Discretize filter** | **Filter method** | **Filter method + Discretize filter** |
|---|---|---|---|---|---|---|
| Correctly Classified Instances | 125776 (99.84%) | 125708 (99.78%) | 125163 (99.35%) | 125404 (99.54%) | 125742 (99.81%) | 125692 (99.77%) |
| Incorrectly Classified Instances | 197 (0.154%) | 265 (0.210%) | 810 (0.643%) | 569 (0.451%) | 231 (0.183%) | 281 (0.223%) |
| Model Building Time | 109.9 sec. | 39.21 sec. | 17.7 sec. | 12.2 sec. | 21.99 sec. | 16.16 sec. |

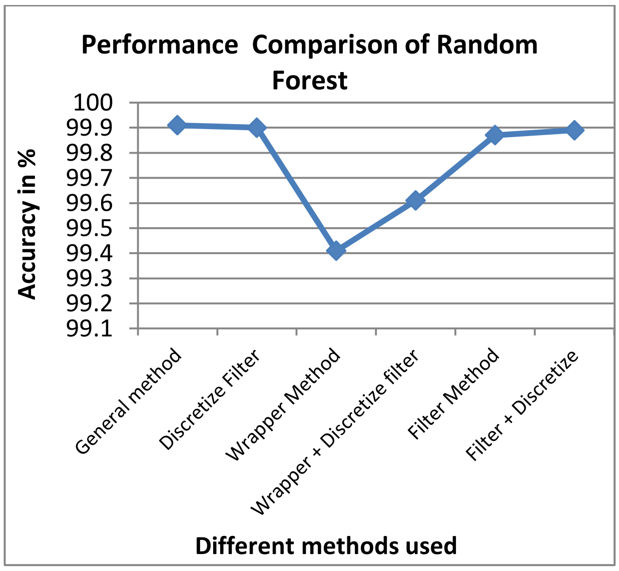

Fig.2. Performance Comparison of Random Forest based on Different Methods

Table 8 shows the evaluation of PART classifier based on different methods. As observed from the table, the accuracy of PART was initially 99.83% but with high model building time. However, on performing feature selection using filter method, model building time was reduced to its one-fifth while accuracy remains almost the same. The model building time was further reduced when feature reduction using wrapper method was applied but with a sizeable reduction in accuracy percentage. Least model building time is observed when feature selection/reduction methods are used in conjugation with discretize filter. Fig. 4 shows the accuracy of PART using different methods.

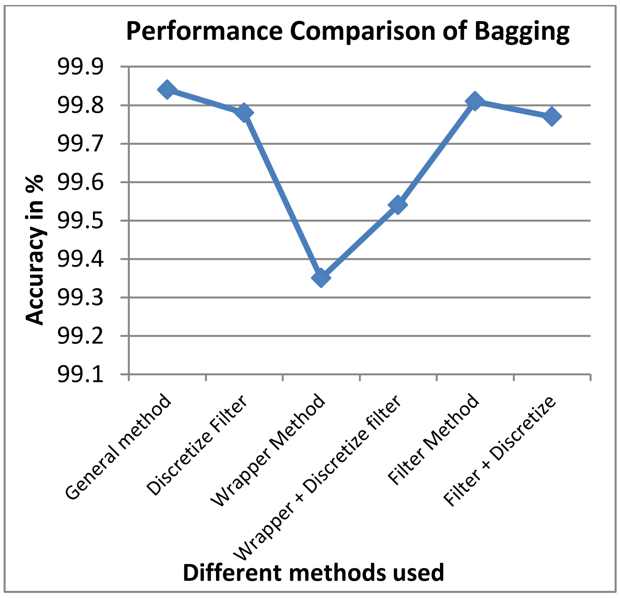

Fig.3. Performance Comparison of Bagging based on Different Methods

Thus, with fewer features, the computational time required during model building can be greatly saved without compromising on detection accuracy. This also indicates that out of 42 features, not all of them are relevant and fewer features are sufficient in detecting attacks whilst achieving high accuracy in little time.

Table 8. Evaluation of PART Classifier based on Different Methods

| Methods Parameters | General method | Discretize Filter classifier | Wrapper method | Wrapper method + Discretize filter | Filter method | Filter method + Discretize filter |
|---|---|---|---|---|---|---|
| Correctly Classified Instances | 125769 (99.83%) | 125680 (99.76%) | 125181 (99.37%) | 125390 (99.53%) | 125761 (99.82%) | 125698 (99.78%) |
| Incorrectly Classified Instances | 204 (0.161%) | 293 (0.232%) | 792 (0.628%) | 583 (0.462%) | 212 (0.168%) | 275 (0.218%) |
| Model Building Time | 99.1 sec. | 14.91 sec. | 8.07 sec. | 2.22 sec. | 20.35 sec. | 3.21 sec. |

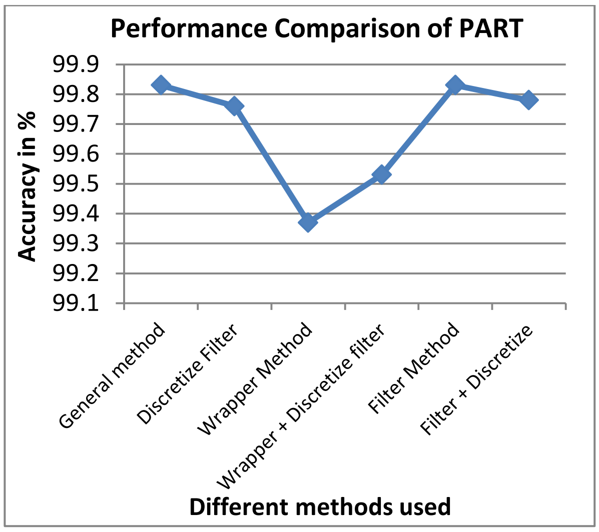

Fig.4. Performance Comparison of PART based on Different Methods

Following observations are made from the above analysis of Tables 5-8:

Wrapper method performs detection with the least accuracy rate in comparison to other methods. Hence, it is not suitable for detecting intrusion in the real world.

Filter method achieves the highest accuracy when compared with other methods except the general method. However, its model building time is dramatically less in comparison to the general method.

Owing to time as the most important consideration in attack detection, the discretize filter works best when used in conjugation with the filter method.

However, it is worth noting from the above analysis that even after applying feature selection/reduction methods, the best 4 ML algorithms consumed more time for model building in comparison to Random Tree (without feature selection/reduction) while achieving the almost same amount of accuracy. Therefore, it becomes evident that Random Tree achieves a good amount of accuracy within a very short span of time even without feature selection/reduction methods.

## VII. Conclusions

Current network intrusion detection systems face the challenge of processing large volumes of data. Hence, for achieving high detection accuracy in lesser possible time, it becomes necessary to remove irrelevant data and reduce its dimensionality by carefully selecting the most important features. This paper emphasizes on the significance of IDS and assesses the performance of 10 most popular ML algorithms using NSL-KDD intrusion detection dataset and ranks them according to their performance. Empirical analysis of best 4 algorithms (Random Forest, Bagging, PART and J48) shows that they consume a lot of time in building the model. So, they are selected for further evaluation in conjugation with different feature selection and feature reduction schemes. Results clearly show that the model building time is significantly reduced with a smaller set of features without compromising accuracy.

However, it is important to note that the best 4 algorithms performed well in terms of model building time after feature selection/reduction but Random Tree is the only classifier which achieves good accuracy in a comparative smaller span of time without using feature selection/reduction methods. The above conclusions are based on the performances of the ML algorithms on NSL-KDD dataset. However, performance of these algorithms may vary for other datasets. We, therefore, feel that by selecting and reducing the features should help a number of ML algorithms to perform well.

Our results will help future researchers as well as industrialists to understand the suitable feature selection and reduction schemes for designing good machine learning algorithms. In future work, we will generate the most acceptable and applicable set of features from large and complex data and build a practical real time system for high speed network intrusion detection. We also plan to formulate a scheme for comparing ML algorithms on the basis of their requirements of space.

## References

[1] Summers R. C., “Secure computing: Threats and safe-guards” in Computers, New York: McGraw-Hill, 2000, pp. 1-688.

[2] Intrusion Detection Systems: Definition, Need and Challenges, SANS Institute 2001. https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-systems-definition-challenges-343

[3] Benferhat S., Tabia K., “Integrating Anomaly-Based

Approach into Bayesian Network Classifiers" in e-Business and Telecommunications, 2009, vol.8, eds. Joaquim Filipe, Mohammad S. Obaidat, pp. 127-139.

[4] Snort (2014), the open source network intrusion detection system [online]. Available at: http://www.snort.org/.

[5] Ranjan R, Sahoo G., "A new clustering approach for anomaly intrusion detection" in International Journal of Data Mining and Knowledge Management Process, 2014 Mar; 4(2), pp. 29–38.

[6] McHugh J., "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory" in ACM Transactions on Information and System Security, vol. 3, no. 4, 2000, pp. 262–294.

[7] Hofmann A., Sick B., "Online Intrusion Alert Aggregation with Generative Data Stream Modeling," in IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 2, 2011, pp. 282-294.

[8] WEKA Machine Learning Project: http://www.cs.waikato.ac.nz/~ml/weka/index.html.

[9] NSL-KDD dataset for network based intrusion detection systems. Available at: http://nsl.cs.unb.ca/NSL-KDD/, December 2016.

[10] Anderson J. P., "Computer security threat monitoring and surveillance," Technical Report, Fort Washington, Pennsylvania, USA, 1980.

[11] Lee W. and Stolfo S. J., "Data mining approaches for intrusion detection" in Proceedings of the 7th conference on USENIX Security Symposium, vol. 7, San Antonio, TX, 1998.

[12] Schultz M. G., Eskin E., Zadok E., Stolfo S. J., "Data Mining Methods for detection of New Malicious Executables", in IEEE Symposium on Security and Privacy, Columbia University, 14-16 May 2001, pp.38-49.

[13] Hwang T., Lee T., and Lee Y., "A Three-tier IDS via Data Mining Approach" in Proceedings of the 3rd annual ACM workshop on Mining network data, 2007, pp. 1-6.

[14] Tavallaee M., Bagheri E., Lu W., and Ghorbani A., "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[15] Srinivasulu P., Nagaraju D., Kumar P. R., and Rao K. N., "Classifying the Network Intrusion Attacks using Data Mining Classification Methods and their Performance Comparison" in IJCSNS International Journal of Computer Science and Network Security, vol. 9, no.6, 2009, pp. 11-18.

[16] Reddy K., Iaeng M., Reddy V. N., and Rajulu P. G., in "A Study of Intrusion Detection in Data Mining" in World Congress on Engineering, vol. III, 2011, July 6-8.

[17] Nadiammai G. V. and Hemalatha M., "Perspective analysis of machine learning classifiers for detecting network intrusions" in IEEE Third International Conference on Computing Communication & Networking Technologies (ICCCNT), India, 26-28 July 2012, 2012, pp. 1-7.

[18] Neethu B., "Classification of Intrusion Detection Dataset using machine learning Approaches" in International Journal of Electronics and Computer Science Engineering, vol. 1, 2012, pp. 1044-51.

[19] Revathi S., Malathi A., "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection", in International Journal of Engineering Research & Technology (IJERT), vol. 2 no. 12, 2013, pp. 1848-1853.

[20] Choudhary S. and Bhowal A., "Comparative Analysis of Machine Learning Algorithms along with Classifiers for Network Intrusion Detection" in IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Chennai, T.N., India, 6-8 May 2015, pp. 89-95.

[21] Murthy P. C., Manjunatha A. S., Jaiswal A., Madhu B. R., "Building Efficient Classifiers For Intrusion Detection With Reduction of Features" in International Journal of Applied Engineering Research, vol. 11, no. 6, 2016, pp. 4590-4596.

[22] Latha S., Prakash S. J., "HPFSM - A High Pertinent Feature Selection Mechanism for Intrusion Detection System," International Journal of Pure and Applied Mathematics, vol 118, no. 9, 77-83

[23] Biswas. S. K., "Intrusion Detection Using Machine Learning: A Comparison Study," International Journal of Pure and Applied Mathematics, vol 118, no. 19, 101-114

[24] WEKA 3.9 - Data Mining with Open Source Machine Learning Software in Java. [Online] Available at: http://www.cs.waikato.ac.nz/ml/weka/ [Accessed: July, 2016].

[25] John G.H., P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers" in Proc. Of the 11th Conference on Uncertainity in Artificial Intelligence, August 18 - 20, 1995, pp. 338-345.

[26] Kaur H., "Algorithm used in Intrusion Detection System: A Review", International Journal of Innovative Research in Computer and Communication Engineering, 2014, vol. 2, issue 5, May 2014.

[27] Quinlan J. R., "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[28] Witten I. H., Frank E. and Hall M. A., "Data Mining: Practical Machine Learning Tools and Techniques", 3rd edition, eds. J. Geller, E. Davis, P.A. Flach, Morgan Kaufmann Publishers Inc, 2011, pp. 1-558.

[29] Cessie S. L., Houwelingen J. C., "Ridge Estimators in Logistic Regression" in Applied Statistics, vol. 41, no. 1, 1992, pp. 191-201.

[30] John G.H., "Irrelevant Features and the Subset Selection Problem" in proc. of the 11th Int. Conf. on Machine Learning, Morgan Kaufmann Publishers, 1994, pp.121-129.

[31] Dash M. & Liu H., "Feature Selection for Classification" in Intelligent Data Analysis, vol.1(3), 1997, pp. 131–56.

## Authors' Profiles

**Dr. Prachi,** Ph. D. and associate professor in The NorthCap University. Her main research interests include cyber security, intrusion detection and prevention.

**Ms. Heena,** pursuing Ph. D. in The NorthCap University. Her main research interests include intrusion detection and prevention.

**Prof. Prabha Sharma,** Professor and Ph. D. supervisor in The NorthCap University. Her main research interests include optimization.

**How to cite this paper:** Prachi, Heena Malhotra, Prabha Sharma,"Intrusion Detection using Machine Learning and Feature Selection", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.4, pp.43-52, 2019.DOI: 10.5815/ijcnis.2019.04.06