Applied Intelligence
https://doi.org/10.1007/s10489-019-01436-1

An intelligent intrusion detection system

Nevrus Kaja[1] · Adnan Shaout[1] · Di Ma[2]

Abstract

With the introduction of emerging technologies cybersecurity has become an inherited and amplified problem. New technologies bring significant developments but also come with new challenges in the cybersecurity area. The fight against malicious attacks is an everyday battle for every company. Challenges brought by security breaches can be devastating for a company and sometimes bring un-survivable circumstances. In this paper, we propose a novel two-stage intelligent intrusion detection system (IDS) to detect and protect from such malicious attacks. Intrusion Detection Systems are feasible solutions for cybersecurity problems, but they come with implementation challenges. Anomaly based IDS usually have a high rate of false positives (FP) and they require considerable computational requirements. The approach proposed in this paper consists of a two-stage architecture based on machine learning algorithms. In the first stage, the IDS uses K-Means to detect attacks and the second stage uses supervised learning to classify such attacks and eliminate the number of false positives. The implementation of this approach results in a computationally efficient IDS able to detect and classify attacks at a 99.97% accuracy while lowering the number of false positives to 0. The paper also evaluates the performance results and compares them with other relevant research papers. The performance of this proposed IDS is superior to the current state of the art.

Keywords Intrusion detection systems (IDS) · Cyber-security · Machine learning · Supervised learning

1 Introduction

Emerging technologies, connectivity and innovations have brought many dependencies in network structures. Increasing connectivity among business applications brings an immediate consideration for cybersecurity breaches. Such breaches often impact business operations and usually result in significant financial losses. Therefore, cybersecurity challenges are nowadays a business priority [10, 11].

From the other spectrum, artificial intelligence and machine learning technologies are enabling researches to achieve breakthroughs in many problems. The purpose of this paper is to leverage some of these technologies in solving cybersecurity problems. To detect attacks, we look and analyze the transactional data, build models and make predictions out of them [10]. Machine leaning algorithms and a Knowledge Discovery Database (KDD) is used to build a two-stage intrusion detection system. This data is gathered from Massachusetts Institute of Technology - Lincoln labs and simulates a typical US Air Force Local Area Network (LAN). A set of attacks are injected to the data such as Denial of Service, Remote Unauthorized access, Local Unauthorized access and probing [17].

The structure of the paper is organized as follows: Section 2 talks about intrusion detection systems, Section 4 provides an analysis of the data set and the pre-processing methodologies. Section 5 gives a description of the IDS architecture and Section 6 provides performance results.

2 Intrusion detection system

Intrusion Detection System are algorithms that look for malicious attacks or activity in a network system. They are categorized into two main categories: anomaly-based systems and signature-based systems [5, 15, 21].

Anomaly-based systems model the normal behavior of the system and compare it with the monitored behavior to create a baseline model [29]. If a certain action differs from the normal behavior, then the IDS analyzes such action and

✉ Nevrus Kaja
nkaja@umich.edu

[1] Electrical and Computer Engineering Department, The University of Michigan-Dearborn, Dearborn, MI 48128, USA

[2] Computer and Information Science Department, The University of Michigan-Dearborn, Dearborn, MI 48128, USA

Published online: 25 March 2019

classifies it into a certain anomaly [3]. The main benefits of such systems is the ability to detect malicious attacks which were unknown to the system prior and their ability to operate in a real-time environment. However, due to noise or other elements in the data, anomaly-based IDS are poised to have problems with high false positives [5, 12]. Figure 1 shows a generic architecture of an anomaly-based Intrusion Detection System.

Signature-based systems scan the data for predetermined and preconfigured attack patterns called probes and sweeps. These elements are used as a basis for raising an alert when an attack behavior occurs [17]. Signature-based IDS are efficient when attack signatures are known in advance but do not work well when there is no prior knowledge of the attacks. Therefore, the downside of such systems is the need to continuously update the database for attack signatures [3]. In additions, signature-based IDS are not optimal for attacks with self-modifying behavior.

3 Dataset and adversary model

In order to build, train and test the proposed IDS we use a Knowledge Discovery dataset. This dataset is one of the most widely used datasets in the evaluation of Intrusion Detection Systems. It was built from Massachusetts Institute of Technology (MIT) and used in the International Knowledge Discovery and Data Mining Tool Competition [11].

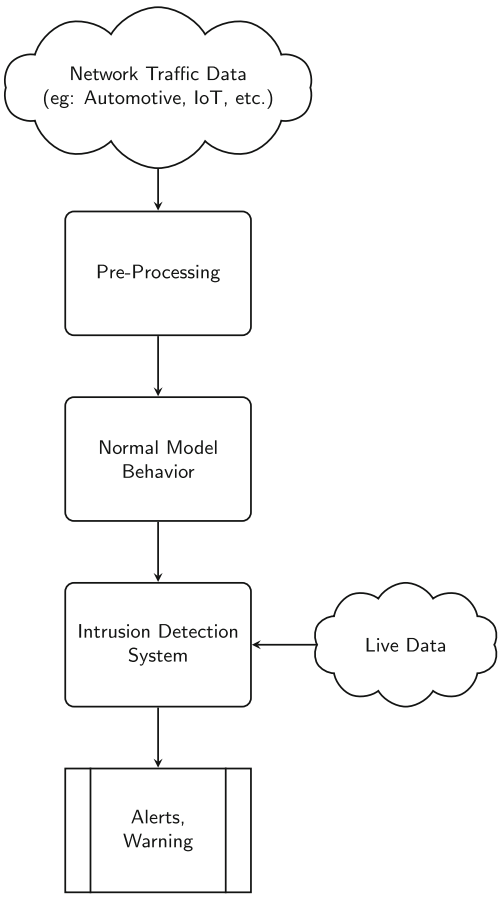

Fig. 1 IDS Architecture

The actual data was gathered as TCP data dumps in an environment set up from MIT Lincoln Lab. The environment simulated a local area network (LAN) of a US Air Force network. The raw data was gathered in a seven weeks' time span and multiple attack were injected to it [17]. It has around 4,900,000 connections with a size of about 4 gigabytes. In addition, the lab also collected two weeks' worth of testing data with around 2 million connections.

This dataset provides a general adversary model. The training data contains labels which identify if a connection is "normal" or "attack" along with the specific attack type. Every connection is about 100 bytes and is defined as a sequence of TCP packets from a source IP to a target IP with a predefined start and end time [17, 20]. According to this model, the attacker has mounted 22 attacks in the following categories [17]:

- Unauthorized port scanning, surveillance or other probing
- Unauthorized access to local root privileges
- Unauthorized access from a remote machine
- Denial of service attacks

To make the problem more realistic, the test data contains attack connections not found in the training data. The test dataset is used to measure the intrusion detection systems performance and contains 17 additional unknown type of attacks.

4 Data pre-processing methodologies

After analyzing and performing preliminary experiments on the dataset described above, a significant amount of redundancies and similarities were discovered [27]. Therefore, when this data was used "as-is" to build an IDS model, it resulted with a biased model towards the more frequent data dumps. The dataset is also too large, and this makes the initial results to be overfitting. Overfitting causes the model to perform well on the training set, but not as well on the test data. These data characteristics made it necessary to pre-process the data prior to using it for building the IDS model [16].

In order to build an accurate IDS model, four pre-processing steps were done to the training data:

First, the variance of feature values was calculated to measure the spread between features in this dataset. Features with low variances usually do not provide

significant Kullback-Leibler divergence during training; therefore, the objective is to filter them out. Kullback-Leibler divergence is a measure of the similarity between probability distributions of two different features.

Table 1 shows all of the initial features in the dataset, along with their variance values. This process was calculated using Matlab during the pre-train process therefore; it does not affect the computational requirements or the performance in real time.

From Table 1 results we can see that "*su_attempted*", "*land*", "*is_host_login*" and "*num_outbound_cmds*" have considerable low variances, therefore they can be removed from the training without impacting the knowledge derived from the model.

Secondly, we calculated and removed correlated features to avoid overfitting. Correlated features are closely associated with each other and do not allow the system to infer distinct knowledge from the data [8]. Correlation coefficient shows how much the two features are associated with each other.

After calculating correlation coefficients for each feature pair, we identify the pairs that are closely correlated. Table 2 shows pairs of features which have a correlation coefficient higher than 0.9. When the correlation coefficient between two features is close to 1, then these two features are similar to each other. Removing closely correlated features helps with removing bias from the model.

Third, Least Square Regression Error (LSQE) was used to minimize feature similarity and maximize dimensionality reduction [23]. (LSQE) or residual variance is the error of predicting y from y = bx + a while a and b are the regression coefficients which can be calculated by minimizing $e(x, y)^2$ in (1).

Equation (2), (3) and (4) show the derivation of them,

$$e(x, y)^2 = \frac{1}{n} \sum e(x, y)_i^2 \tag{1}$$

$$e(x, y)^2 = y_i - a - bx_i \tag{2}$$

$$a = \bar{y} \tag{3}$$

$$b = \frac{cov\,(x, y)}{var(x)} \tag{4}$$

Once a and b are calculated then the mean square error $e(x, y)$ can be calculated by (5).

$$e(x, y) = var(x) \times (1 - relation(x, y)^2) \tag{5}$$

Equation (5) essentially measures the relationship between two features x and y. If these features have a linear relationship then $e(x, y)$ will be 0 and if they don't have a relationship then $e(x, y)$ will be equal to $var(x)$.

Table 1 Dataset features

Name	Variance	Name	Variance
time_duration:	1.98E+006	guest_login:	2.76E-02
type_of_protocol:	1.57E-001	count_nr:	1.65E+04
srv:	2.18E+002	srv_nr:	7.93E+03
flg:	1.03+E01	serror:	8.72E-02
source_bytes:	2.23E+011	srv_serror:	8.90E-02
destination_bytes:	4.50E+008	rerror:	1.73E-01
land:	3.10E-004	srv_rerror:	1.73E-01
error_fragment:	2.03E-002	same_srv:	1.70E-01
high_urgent:	1.33E-003	df_srv:	6.72E-02
warned:	8.61E-001	srv_df_host:	6.43E-02
nr_failed_log_in:	2.26E-002	destination_count:	8.84E+03
log_in:	2.47E-001	destination_srv_count:	1.25E+04
nr_compromised:	2.30E-003	destination_same_srv_rate:	1.89E-01
root:	2.43E-003	destination_diff_srv_rate:	4.87E-02
su_attempt:	4.44E-004	destination_same_src_port_rate:	9.38E-02
nr_root:	6.47E+001	destination_srv_diff_host_rate:	7.29E-03
nr_file_new:	4.58E-001	destination_serror_rate:	7.46E-02
nr_shells:	2.30E-003	destination_srv_serror_rate:	7.95E-02
nr_access_files:	4.60E-003	destination_rerror_rate:	1.45E-01
nr_outbound_cmds:	0.00E+000	destination_srv_rerror_rate:	1.61E-01
host_login:	4.88E-004		

Table 2 Correlated features

Feature A	Feature B	Correlation between A and B
destination_srv_serror_rate:	destination_serror_rate:	0.95
destination_srv_rerror_rate	rerror:	0.93
dst_srv_rerror_rate	server_rerror_rate:	0.95
rerror:	service:	0.91
nr_compromised:	nr_root	0.99
serror	server_serror	0.97
server_rerror	rerror	0.97

Lastly, Maximal Information Compression Index (MICI) was also used to analyze feature relationships. Denoted by $\lambda_2(x, y)$ which is calculated using (6).

$$\lambda_2(x, y) = min(eigv(\Sigma)) \tag{6}$$

where Σ provides the covariance of x and y while $eigv$ is the Eigen values vector of that covariance. When MICI is 0, then the features have linear relation. A higher MICI means that the relationship does not exist.

After calculating variance, correlation coefficient, LSQE and MICI, 11 features were removed from the data. These features did not provide distinct knowledge for the Intrusion Detection System model. Therefore, the number of features was reduced from 41 to 30, which is more manageable and more meaningful for training the model [23].

5 Two stage IDS

After data pre-processing, we design an intelligent, two-stage, intrusion detection system architecture build upon machine-learning algorithms. The novelty of this work is the use of a two stage machine learning approach in the design of IDS after a four step efficient data pre-processing. This is done to increase the accuracy and lower the number of false positives when detecting an attack [12]. The first stage detects whether there is an attack or not while the second stage classifies that attack and provides an alert [24]. We have already published Fig. 2 provides the block diagram architecture of the proposed IDS.

5.1 First stage: Detect!

The first stage uses a K-Means model to cluster the data and detect an attack [4, 9]. Clustering in this phase simply classifies the data into two categories: "attack" connections or "normal" connections. The design objective for this stage is to simply detect if there is an attack or not. The algorithm is purposely designed to show bias towards "attack" connections because at this stage we can afford false positive attacks but do not want to miss any true positives. To accomplish this classification we build two clusters with low inter cluster similarity and good intra-cluster similarity [9].

K-Means clustering is used in machine learning to processes n observations into k clusters [19]. This algorithm looks at the pre-processed data features and places them

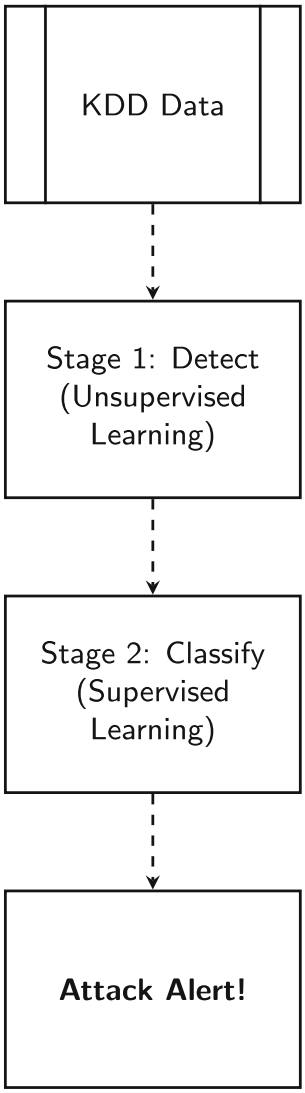

Fig. 2 Intrusion detection system architecture

into clusters. The reason why we choose k-Means in this stage is due to its computational requirements and the ability to produce tighter and more controlled clusters rather then hierarchical ones. In addition k-Means is an easy to implement and flexible algorithm re-classifying connections continuously as centroids (cluster centers) are re-calculated.

To further explain the k-Means algorithm, let's consider the following example:

The training set consists of random connections $X^{(1)}$, ... ,$X(n)$. The objective of the algorithm is to separate this in a few clusters – in our case we want to split it in simply two clusters (attack and normal). In addition, we also have some feature vectors for each connection $X^{(i)} \in R^m$. K-Means attempts to predict k centroids (k=2 in this case) and a label $C^{(i)}$ for each connection $X^{(i)}$ K-Means does the following steps to determine the predictions for every connection.

1. Initialize cluster centroids $m_1, m_2, \ldots m_k$
2. Repeat under convergence

For every i, set

$$C(i) \;=\; min_j \;||\, X_(i) - \;--m_j||^2 \tag{7}$$

For every j, set

$$m_j = \frac{\sum_{i=1}^{m} 1\left\{C^{(i)} = j\right\} X^{(i)}}{\sum_{i=1}^{m} 1\left\{C^{(i)} = j\right\}} \tag{8}$$

The Euclidian distance between the given connection and the cluster center decides the placement of the connection between the two clusters. The cluster centroid is recalculated every time the connection is placed in the cluster [28]. The algorithm tries to minimize the objective function. Objective function is donated by E and known as the squared error function (9).

$$E = \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N} z_{nk} \|\mathbf{x}_n - \mathbf{m}_k\|^2 \tag{9}$$

where $\mathbf{m}_k$ is the center of the cluster K, N is the total number of data points, $\|\ .\ \|$ give the Euclidian norm (L2 norm) of the vector.

Figure 3 provides a visual view of the two data clusters generated by the K-Means algorithm.

As shown from the figure there are two clusters (with different colors - red and blue) which are well defined. Table 3 shows the results of the first stage.

After building the model, the detection accuracy for the first stage is 96.61%. After detecting attacks in this stage, then we attempt to classify those attacks in stage two with a series of machine learning supervised algorithms.

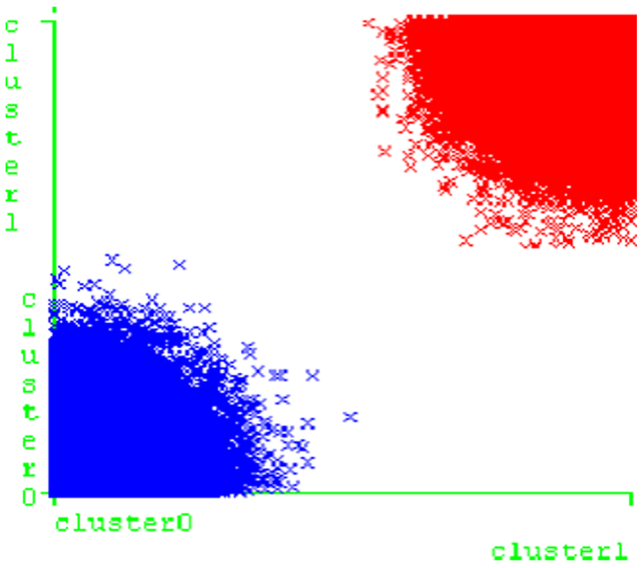

Fig. 3 Visual of the K-Means Clusters

5.2 Second stage: Classify!

In this stage, the objective is to lower the rate of false positives and improve accuracy. By classifying these attacks into a certain category, we increase the confidence that an attack is detected and appropriately classified. The following algorithms are used to test the IDS performance regarding classifying the attacks [25]:

5.3 J48

J48 is a decision tree-based algorithm which classifies data. This algorithm builds decision trees by using information entropy and is based on C4.5 decision tree [19]. This algorithm is often referred as a statistical classifier because it bases its decision tree on labelled input data. When building the tree, J48 chooses the attributes based on the information gain, or whichever attribute results in the most efficient split of the data [22]. The following steps describe how the algorithm works:

1. Calculate the entropy using (10)

$$Entropy = \sum_{i=1}^{C} -p_i \;*\; \log_2(p_i) \tag{10}$$

Table 3 Stage 1 results (Detect)

	Clustering	True labels	Accuracy
Normal	23%	19.61%	96.61%
Attack	77%	80.39%	

Table 4 J48 results

Performance variable	Score
# Leaves	711
Tree size	833
Classification Accuracy	99.9512%
True Positives	1.000
False Positives	0.000
F-Measure	0.999
Computational Time for train	48.13 (s)
Computational Time for test packet	4.02e-6 (s)

2. Calculate information gain rate using (11). The gain is basically the difference of prior entropy (T) and the entropy of the selected branch (X).

$$Gain(T, X) = Entropy(T) - Entropy(X) \quad (11)$$

3. After calculating the Gain for each candidate attribute, then the data is split based on the attribute with the highest information gain.
4. Repeat steps 1-3 until the leaf level [22].

After applying the provided data to the first stage and using J48 as the second stage, the following performance results were obtained as shown in Table 4.

The following is the explanation for some of the performance variables:

- *Classification accuracy*: % of connections that are classified correctly
- *True positives*: proportion of instances predicted positive that are actually positive
- *False positives*: proportion of instances predicted positive but are actually negative
- *F-Measure*: measure of test's accuracy
- *Computational time for train* - time it takes to train the model
- *Computational time for test packet* - time it takes to test a single packet

J48 tree produced the following features with the highest information:

- *server_count* - number of connections to the same service in the past two seconds
- *same_server_rate* - percentage of connections to the same service
- *destination_different_server_rate* - percentage of connections to different services

As observed from the results of Table 2, stage 2 was able to decrease the number of false positives (FP) to 0 and increase the accuracy results to 99.9512%. This is a significant improvement from stage 1 results, which was 96.61%.

Figure 4 visualizes the performance of the J48 algorithm. The X-axis represents the actual label of the class while the Y-axis represents the predicted label of the class. Receiver Operating Characteristic (ROC) curves were not used in these experiments because the False Positive rate is too low and the visualization is skewed. Colors of the figure show the different types of attacks.

Each quadratic dot in the figure represents a classification error and since errors are too few in numbers compared with correctly classified records (crosses), we have emphasized those using squares. The attacks, which are not classified (detected) in this stage, are scattered around due to having outlier characteristics. J48 does not perform well once the feature with the most information gain of an attack falls into the wrong branch. J48 tree propagates that error for the rest of the leaves. Some Tree Pruning is used in this case to minimize such errors.

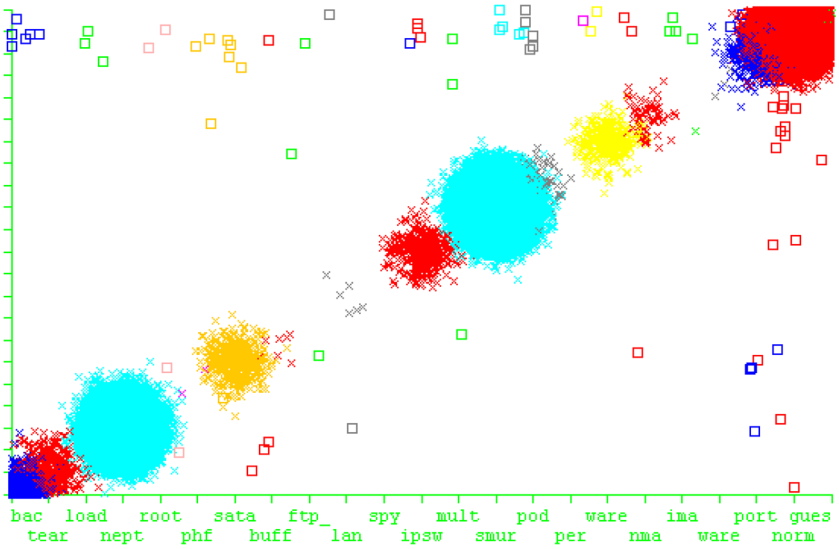

Fig. 4 J48 classifier errors curve

After analyzing J48 results, the following can be concluded about J48:

5.3.1 Advantages

J48 is a fairly easy algorithm to implement and visualize. Since it is based on C4.5, it performs well in discrete data with more than two classes, which is also one of the main reasons why J48 is chosen as one of the candidate algorithms in stage two. In addition, computational requirements for decision making are fairly low compered with other algorithms used in this paper. Looking at the literature survey available, J48 is used commonly in medical and clinical applications, weather prediction and banking data.

5.3.2 Disadvantages

When looking at the training computational requirements, generally J48 takes more time and memory to be trained. If a J48 decision tree is not able to be configured properly, it results in a large tree and the algorithm denigrates pretty easy. If J48 outputs a complex tree it gives a poor performance and requires high computational power. That is why, it is recommended to apply tree pruning which helps with complexity and sometime avoids over-fitting and other classification errors. J48 and decisions trees in general have limits when dealing with continuous data, or decisions which require more than one output per attribute.

5.4 Random forest

Random forest uses an ensemble learning method to combine decision trees, similar to those explained previously. This algorithm is similar to a technique known as bagging. Bagging is a machine learning ensemble meta-algorithm aimed at improving accuracy, reducing variance and avoiding over-fitting. In a single decision tree, the predictions are sensitive due to certain data characteristics or noise, bagging takes the average performance of multiple trees so it eliminates such sensitivity and gives a more accurate performance. Random Forest improves bagging with a multitude of decision trees [7]. The mode output among the decision trees is output of random forest. Table 5 provides the IDS performance results while using Random Forest in the second stage.

5.4.1 Advantages

Random Forest does a great job at correctly classifying and identifying malicious attacks from the data as shown in Fig. 5 and Table 5. Although the dataset is huge, the results show that errors are countable and the accuracy is improved to 99.9708%. Based on literature survey, Random Forest is widely known to be one of the most accurate learning algorithms and that is also the reason why we selected Random Forest to be one of the algorithm candidates for stage two [7]. Accurate results received from this algorithm simply validated the previous claim. Random Forest in the second stage give us the best accuracy. This accuracy does not denigrate even with a large data set or even at times when a large portion of the pockets are missing. Once trained, this algorithm can also be re-used in other models with similar data which is another feature making random forest a predominant algorithm to be used for classification problems. Random Forest is commonly used in areas of Medicine, E-commerce and stock market application.

Table 5 Random forest results

Performance variable	Score
# of Iterations	100
Classification Accuracy	99.9708%
True Positives	1.000
False Positives	0.000
F-Measure	0.999
Computational Time for train	235.52s
Computational Time for test packet	4.29e-5

5.4.2 Disadvantages

A high performance in Random Forest comes also with an increase in computational cost. Since Random Forest is another decision tree-based algorithm, the same path of scattered errors is repeated in Fig. 5. This relates to the same issues described in J48. Another disadvantage of random forest is the fact that it is hard to interpret it. In addition a careful analysis is needed in deciding its configuration parameters (e.g numbers of trees) according to the data-set used, otherwise the performance accuracy will suffer.

5.5 Adaptive boosting

Adaptive Boosting is another ensemble of machine learning algorithms developed by Yoav Freund and Robert Schapiro [19]. In Adaptive Boosting (AdaBoost) the ensemble is built in such a way that prediction errors are improved at every layer. The subsequent models focus on fixing errors made by prior models. This is similar to regular boosting algorithms. Adaptive Boosting adds short decision trees in series until the performance is not subsequently improved [22]. Performance results for IDS with Adaptive Boosting are shown in Table 6.

Figure 6 shows the Adaptive Boosting classifier errors results.

Fig. 5 Random forest classifier errors curve

Adaptive Boosting performance visualization is somehow skewed. It is interesting to see that most of the errors are around unauthorized access from a remote machine type of attacks. These kinds of attacks are usually associated with ”duration of connection”, ”service requested” and ”number of failed attempts”. Adaptive Boosting miss-classifies these due to having a high information gain in the dataset [22].

After looking at AdaBoost performance we can conclude the following about the algorithm:

5.5.1 Advantages

AdaBoost is not a complicated algorithm to implement. Generally, the algorithm is known to give a good generalization and is used in many classification problems. AdaBoost is usually not prone to over-fitting due to its ”boosting” technique. This algorithm is used in many classification problems and it is known to improve classification errors through boosting. In our case, AdaBoost does not give the best performance in terms of accuracy but it does give the best timing performance, therefore making it one of the most efficient algorithms for implementation. Its implementation efficiency and over-fitting avoidance are the reasons why AdaBoost was selected as the third algorithm candidate. There are a few papers which have evaluated the use of AdaBoost in different applications such as [2, 6]. Some papers actually consider AdaBoost as one of the best ”off the shelf” algorithms. [2]. Applications where AdaBoost has been implemented successfully mainly focus on optical character recognition, pedestrian detection systems, speech and facial recognition, etc.

Table 6 Adaptive boosting results

Performance variable	Score
Classification Accuracy	97.8597%
True Positives	0.979
False Positives	0.005
F-Measure	0.970
Computational Time for train	45.65s
Computational Time for test packet	3.71e-6

5.5.2 Disadvantages

One of the main disadvantages for AdaBoost is its sensitivity due to noise in the data-set and potential outliers. This property is also shown when applied to our data. When we injected additional unknown attacks or outlier packages, AdaBoost suffered in their classification. As we will analyze below, this algorithm is not the most optimal solution for our problem, this is also often the case for other complex classification problems.

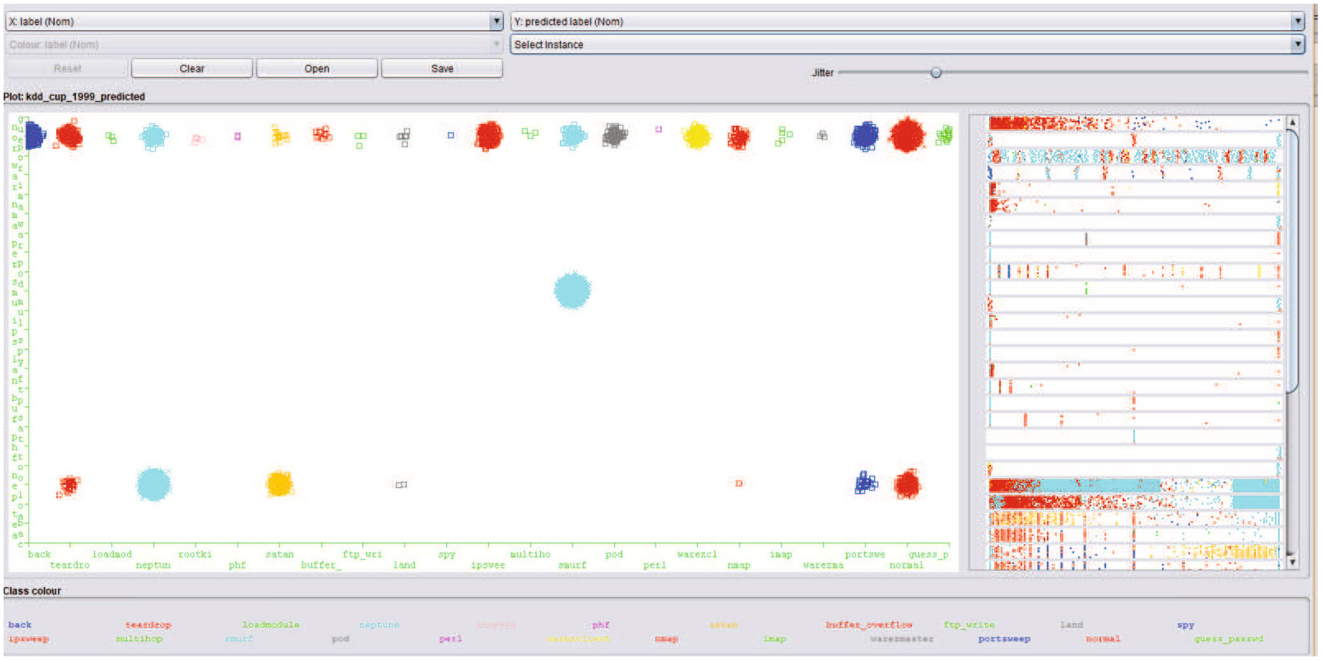

Fig. 6 Adaptive Boosting Classifier Errors Curve

5.6 Naïve Bayes

Naïve Bayes is another algorithm selected for use in the second stage of the intrusion detection system. This algorithm uses a probabilistic classification and is based on Bayes theorem. The objective of this algorithm is to determine the probability of the features happening in every class and return the highest probable class.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{12}$$

Equation (12) calculates the probability of an event (A) considering prior probability of conditions (B) that might be related to event (A).

Table 7 provides the performance results of the IDS with Naïve Bayes as the algorithm in the second stage.

Figure 7 shows the Naive Bayes classifier results.

Table 7 Naïve Bayes results

Performance variable	Score
Classification Accuracy	92.748%
True Positives	0.927
False Positives	0.000
F-Measure	0.949
Computational Time for train	40.69s
Computational Time for test packet	1.48e-6

5.6.1 Advantages

Naive Bayes is primarily based on the conditional independence assumption as shown by (12). When the data-set holds the conditional independence assumption as true, then the algorithm converges quickly, making it more efficient than other logistic regression algorithms. In this scenario, the algorithm can also be trained with less data. One of the most common applications that uses Naive Bayes is e-mail spam detection and news categorization. Its ability to infer based on the independence assumption is also the reason why we selected Naive Bayes in this paper. This property makes this algorithm to be suitable with other applications such as sentiment analysis, digit recognition, etc.

5.6.2 Disadvantages

Similar to Adaptive Boosting, Naive Bayes, in our case has a problem with unauthorized access type of attacks as well. In Adaptive Boosting we had a low prediction for such attacks, while here these attacks are incorrectly classified resulting in poor accuracy results. These kind of attack features have a high information gain so it is easy to skew the training model. Naive Bayes' classifies those features true in many cases due to these features being true for other attacks as well [1]. The accuracy performance for Naive Bayes is the lowest among the the algorithms, suggesting that the independence assumption is not a strong characteristic of our the attacks found in our data. This means that Naive Bayes is an algorithm that does nor perform well in data-sets where features are not independent of each other.

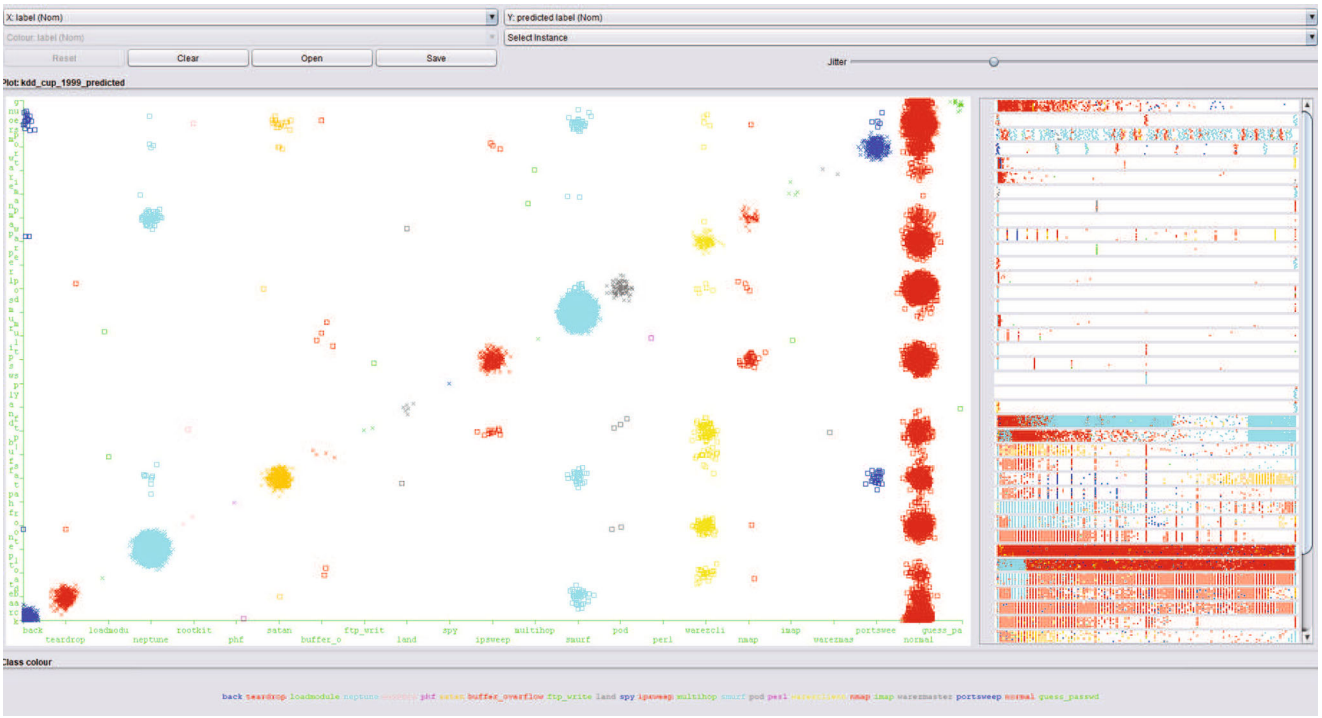

Fig. 7 Naïve Bayes classifier errors curve

6 Performance summary

6.1 Performance based rank

Figure 8 shows the performance summary results from the four algorithms tested in the second stage.

As shown, the Random Forest (99.9708%) and J48 (99.9512%) have the best classification accuracies when it comes to performance. Both of these algorithms - assisted from K-means model in stage one are able to eliminate false positives. As we mentioned false positives are a syndrome of anomaly-based intrusions, and this proposed IDS architecture in this paper is able to cope with them in an easy and efficient way.

6.2 Computation performance

In order to evaluate the performance of the algorithms, the computational time required to train and test the algorithms are used. The models were built and tested using a Windows

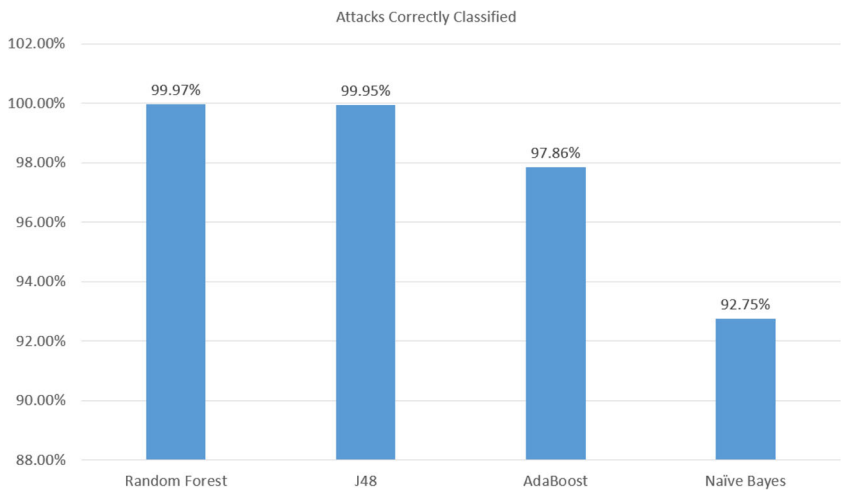

Fig. 8 Attacks correctly classified

Table 8 Summarized results

Algorithm	Performance	Computational time
Random Forest	**99.97**%	4.29e-5
J48	99.95%	**4.02e-6**
Adaptive Boosting	97.86%	3.71e-6
Naïve Bayes	92.75%	1.48e-5

HP Desktop with an Intel Core i7-6700 CPU @ 3.40 GHz. Table 8 shows computational times versus classification performance.

When it comes to classification performance, Random Forest is the best algorithm to perform. Looking at computational time required, J48 delivers similar performance results (within 0.02% of Random Forest) but with almost 10 times better computational time requirement. Therefore, considering implementation feasibility, computational limitations and other factors, J48 is a feasible algorithm to select for the second stage of the new proposed Intrusion Detection Architecture in this paper [20, 26].

6.3 State of the art comparison

Other research papers have used the same dataset (KDD) to build Intrusion Detection Systems and tested their performance. Their approaches vary but based on our literature survey we have not identified another intelligent, two-stage IDS architecture as proposed in this work. Generally, all the other research papers have an agreement regarding the problematic false positives when building an IDS [13]. Among many papers who have used IDS on the knowledge discovery dataset we have selected papers in Table 9 since they use the same dataset, leverage machine-learning algorithms, are published recently and have some of the best performance results. Table 9 compares our results with these papers performances.

As shown from the table comparison, our proposed algorithm stands out with the best performance in both accuracy and F-Measure.

7 Summary

This paper initially provided a background in intrusion detection systems and their importance in the cybersecurity space. Then, a standardized dataset was analyzed and pre-processed using variance, correlation coefficients, Least Square Regression Error and Maximal Information Compression Index. Pre-processing the data was necessary to reduce the number of features and help with avoiding bias and overfitting. After such pre-processing, the data was used to build an intelligent, two-stage intrusion detection system.

The first stage uses the K-means algorithm to detect an attack. The second stage tests four different supervised learning algorithms (J47, Random Forest, Naïve Bayes, and Adaptive Boosting) to classify the attacks. After building and testing the two-stage model we achieved a high accuracy in performance results. In addition, the proposed IDS was able to fully eliminate the number of false positives which are usually a syndrome of anomaly-based intrusion detection systems.

Lastly, we compared the performance results from this proposed IDS with other state of the art research papers. Based on the comparison we showed that results generated from this IDS are one of the best performance results achieved using the KDD dataset.

References

1. Abar T, Letaifa AB, El Asmi S (2017) Machine learning based QoE prediction in SDN networks. In: 2017 13th international wireless communications and mobile computing conference (IWCMC), IEEE, pp 1395–1400
2. Bauer B, Kohavi R (1999) An empirical comparison of voting classification algorithms: Bagging, boosting, and varian. In: Machine learning, pp 105–139
3. Chandolikar N, Nandavadekar V (2012) Efficient algorithm for intrusion attack classification by analyzing KDD cup 99. In: 2012 9th international conference on wireless and optical communications networks (WOCN), IEEE, pp 1–5
4. Dave D, Richhariya V (2012) Intrusion detection with KNN classification and DS-theory. Int J Comput Sci Inf Technol Secur(IJCSITS) 2(2):274–281

Table 9 Related work

Research paper	Algorithm	Accuracy (%)	F-Measure
2017: Meena, Choudhary [20]	Bayes	92.72	0.916
2017: Meena, Choudhary [20]	J48	99.45	0.993
2016: Subba et al. [26]	C4.5 (best)	98.74	Nor reported
2017: Kushwaha et al. [18]	SVM	99.63	0.99
Proposed Algorithm	**Two Stage**	**99.95**	**0.999**

5. Depren O, Topallar M, Anarim E, Ciliz MK (2005) An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Systems with Applications 29(4):713–722
6. Ditterich T (2000) An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. In: Machine Learning, pp 139–157
7. Dong Y, Du B, Zhang L (2018) Target detection based on random forest metric learning. In: Ground vehicle systems engineering and technology symposium, national defense industrial association
8. Dubey GP, Gupta N, Bhujade RK (2011) A novel approach to intrusion detection system using rough set theory and incremental SVM. Int J Soft Comput Eng (IJSCE) 1(1):1448
9. Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) The KDD process for extracting useful knowledge from volumes of data. Commun ACM 39(11):27–34
10. Iguer H, Medromi H, Sayouti A, Elhasnaoui S, Faris S (2014) The impact of cyber security issues on businesses and governments: A framework for implementing a cyber security plan. In: 2014 international conference on future internet of things and cloud (FiCloud), IEEE, pp 316-321
11. Kaja N, Shaout A, Borovikov M (2014) Security solution for cloud computing using a hardware implementation of aes. In: The international arab conference on information technology, ACIT
12. Kaja N, Shaout A, Dehzangi O (2017) Two stage intelligent automotive system to detect and classify a traffic light. In: 2017 international conference on new trends in computing sciences, ICTCS, IEEE, pp 30–35
13. Kaja N, Shaout A, Ma D (2017) A two stage intrusion detection intelligent system. In: The international arab conference on information technology, IEEE-ACIT
14. Kaja N, Nasser A, Shaout A, Ma D (2018) Automotive security. In: Encyclopedia of wireless networks. Springer
15. Kayacik HG, Zincir-Heywood N (2005) Analysis of three intrusion detection system benchmark datasets using machine learning algorithms. In: International conference on intelligence and security informatics, Springer, pp 362–367
16. Kayacik HG, Zincir-Heywood AN, Heywood MI (2005) Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. In: Proceedings of the 3rd annual conference on privacy, security and trust
17. KDD Cup (1999) Intrusion detection data set. The UCI KDD Archive Information and Computer Science University of California, Irvine. http://kddicsuciedu/databases/kddcup99
18. Kushwaha P, Buckchash H, Raman B (2017) Anomaly based intrusion detection using filter based feature selection on KDD-CUP 99. In: Region 10 Conference, TENCON 2017-2017, IEEE, IEEE, pp 839-844
19. MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA, vol 1, pp 281–297
20. Meena G, Choudhary RR (2017) A review paper on ids classification using KDD 99 and NSL KDD dataset in WEKA.In: 2017 international conference on computer, communications and electronics (Comptelix), IEEE, pp 553–558
21. Northcutt S, Novak J (2002) Network intrusion detection. Sams Publishing
22. Pandey P, Prabhakar R (2016) An analysis of machine learning techniques (J48 & AdaBoost)-for classification. In: 2016 1st India international conference on information processing (IICIP), IEEE, pp 1–6
23. Pervez MS, Farid DM (2014) Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In: 2014 8th international conference on software, knowledge, information management and applications (SKIMA), IEEE, pp 1–6
24. Shaout A, Kaja N (2015) A smart traffic sign recognition system. In: 2015 11th international computer engineering conference, ICENCO, IEEE, pp 57–162
25. Singh J, Nene MJ (2013) A survey on machine learning techniques for intrusion detection systems. International Journal of Advanced Research in Computer and Communication Engineering 2(11):4349–4355
26. Subba B, Biswas S, Karmakar S (2016) A neural network based system for intrusion detection and attack classification. In: 2016 22nd national conference on communication (NCC), IEEE, pp 1–6
27. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications, 2009, CISDA, IEEE, pp 1–6
28. Velmurugan T, Santhanam T (2010) Performance evaluation of k-means and fuzzy c-means clustering algorithms for statistical distributions of input data points. Eur J Sci Res 46(3):320–330
29. Zhang L, Shi L, Kaja N, Ma D (2015) A two-stage deep learning approach for can intrusion detection. In: IEEE journal of selected topics in applied earth observations and remote sensing, IEEE, pp 1830–1838

Nevrus Kaja is a PhD Candidate in the Electrical and Computer Engineering Department at the University of Michigan – Dearborn. He is the Vice-Chair of IEEE-Region 4- Southeastern Michigan Section and Chair of Student Activities for IEEE Region 4. His current research is in applications of artificial intelligence, machine learning and cyber-security. He has won many academic, research and industry awards including Honor Scholar Award and Difference Maker from University of Michigan-Dearborn and "Thirty under 30" Fellow from Ford Motor Company. He received his B.S.E and M.S.E in Computer Engineering from University of Michigan Dearborn.

Adnan Shaout is a full professor and a Fulbright Scholar in the Computer Science Department at the Electrical and Computer Engineering Department at the University of Michigan – Dearborn. His current research is in applications of software engineering methods, cloud computing, embedded systems, fuzzy systems, real time systems, cyber security and artificial intelligence. Dr. Shaout has more than 36 years of experience in teaching and conducting research in the Computer Science, Electrical and Computer Engineering fields at Syracuse University and the University of Michigan - Dearborn. Dr. Shaout has published over 260 papers in topics related to Computer Science, Electrical and Computer Engineering fields. Dr. Shaout has obtained his B.S.c, M.S. and Ph.D. in Computer Engineering from Syracuse University, Syracuse, NY, in 1982, 1983, 1987, respectively.

Di Ma is an Associate Professor in the Computer and Information Science Department at the University of Michigan-Dearborn (UM-Dearborn), where she serves as the director of the Cybersecurity Center for Education, Research, and Outreach (CCERO) and the Security and Forensics Research Lab (SAFE). She is broadly interested in the general area of security, privacy, and applied cryptography. Her research spans a wide range of topics, including data privacy, automotive and connected vehicle security, smartphone and mobile device security, RFID and sensor security, and so on. Her research is supported by NSF, NHTSA, AFOSR, Intel, Ford, and Research in Motion. She received the PhD degree from the University of California, Irvine, in 2009. She was with IBM Almaden Research Center in 2008 and the Institute for Infocomm Research, Singapore in 2000-2005. She was the recipient of the 2018 Trevor O. Jones Outstanding Paper Award from SAE, the Distinguished Research Award from the College of Engineering and Computer Science of UM-Dearborn in 2017, and the Tan Kah Kee Young Inventor Award in 2004.