Journal of Theoretical and Applied Information Technology
15th July 2021. Vol.99. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENSEMBLE LEARNING BASED FEATURE REDUCTION AND SELECTION METHODS FOR NETWORK INTRUSION DETECTION SYSTEM

[1]SARA TAMY, [2]HICHAM BELHADAOUI, [3]NABILA RABBAH, [4]MOUNIR RIFI

[1,2,4] Laboratory of Network, Computing, Telecommunications and Multimedia, ESTC, Hassan II University, BP. 8012, Casablanca, Morocco
[3]Laboratory of Structural Engineering, Intelligent Systems and Electrical Energy, ENSAM, Hassan II University, BP. 20000, Casablanca, Morocco
E-mail: [1]saratamy@yahoo.fr

ABSTRACT

The use of network intrusion detection systems based on machine learning algorithms is currently emerging as one of the most effective solutions for monitoring high dimensional network traffic and identifying anomalous flows with high accuracy. Integrating feature reduction/selection techniques is also essential to reduce the undserlying complexity of processing big data sets and detect intrusions in real time. The purpose of this paper is to investigate the possibility of using hybrid network intrusion detection system based feature reduction/selection techniques and ensemble algorithms. First, we compare the performance of six classifiers namely Naïve Bayes, Support Vector Machine, Simple Logistic Regression, JRip, Part and J48 using the NSL-KDD dataset. After analyzing the results, it is obvious that the algorithms take a lot of time to build the model. Therefore, we applied three dimensionality reduction methods namely: Information gain evaluation, correlation attribute evaluation and OneRule attribute evaluation, to detect intrusions in the minimum possible time without compromising accuracy. Then, we compared the performance of these methods based on the time taken to build the model, accuracy, error rate and other metrics to select the best one and associate it with Artificial Bee Colony algorithm. Based on the experimental results the three best classifiers are selected to be combined into a stacking model and a majority voting model. We then evaluate them using several detection measures including accuracy, precision, F-Measure, recall, time to build model, attack detection rate through true positive rate and false positive rate, and confusion matrices. The results are given and analyzed for each category of attack including R2L, Probes, DOS and U2R to identify the weaknesses of each algorithm, in order to make it more robust against new intrusions. Overall, no algorithm in the model of attack detection performed very well in detecting new U2R and R2L intrusions, nevertheless, the outcomes of our study demonstrate that stacking model, with J48 as the model learner and Part with JRip as the base classifiers, has allowed to increase the detection accuracy of R2L to 15. 20%, U2R up to 29.85%, Probes to 84.55%, DOS to 84.04% and an accuracy score of 91.17% for normal traffic, while reducing the time needed to build the model.

Keywords: *Machine Learning, Feature Reduction, Feature Selection, Ensemble Classifier, Naïve Bayes, Support Vector Machine, Simple Logistic, JRip, Part, J48, Network Intrusion Detection System*

1. INTRODUCTION

The digitization carried by the fourth industrial revolution is currently occurring in most organizations and companies, in order to provide a more reliable services to their customers and improve production processes. This transformation is mainly related strengthening connectivity and interaction between systems. However, this connectivity increases the vulnerabilities of networks and systems and makes them privileged targets for threats.

Cybersecurity is one of the core elements of industry 4.0 and digitalization. Therefore, it is essential to provide convenient solutions to cybersecurity issues. In concrete terms, to ensure an optimal level of security, companies have to apply the best cyber practices, such as the use of effective and optimized intrusion detection and prevention systems.

Intrusion detection systems monitor and analyze the events in network traffic to detect, identify and predict attacks. Generally, IDSs can be classified, according to the target they will survey, in two categories: network intrusion detection systems (NIDS) and host intrusion detection systems (HIDS) [1], and they are designed based on different approaches, namely Scenarios-based approach in which the IDS attempts to match data collected from the system's information sources with data already known using database of signatures. and Behavioral approach: the IDS detects violations of the system's security policy by analyzing the behavior of users, then comparing it with a pattern of normal behavior, known as a profile [2].

Building an efficient intrusion detection model is a challenging task, as the entire dataset contains a large number of redundant and irrelevant features. using all the features in the data to detect intrusions reduces the performance of learning and makes analysis difficult. Thus, applying feature reduction and selection methods to decrease the dataset dimensionality is required, in order to reduce the time taken for processing and to improve the detection rate and precision [3]. On the other hand, the use of ensemble classifier, that combine many algorithms, is also an important aspect of data mining that increases efficiency and performances.
The main objective of this paper is to improve the network intrusion detection performance by applying the feature reduction and selection methods to different classification algorithms, namely: Naïve Bayes, Support Vector Machine, Simple logistic, JRip, Part and J48. The three best classifiers are then selected and combined in an ensemble learning. The experimental study is conducted on the NSL KDD dataset which is an advanced version of KDD CUP99.
The main contributions of this paper are:

- Assessment of different data mining algorithms on the NSL KDD dataset without applying feature selection/reduction methods.
- Apply three dimensionality reduction methods namely information Gain evaluation, correlation attribute evaluation and OneRule attribute evaluation, to remove redundant and irrelevant features. The algorithms are ranked based on their performances using a different metrics like precision, recall, F-measure, time to build the model, etc.
- Comparison of performances of these methods to select the best one and associated it with Artificial Bee Colony (ABC) optimization method.
- Combining the three best algorithms in a stacking model and majority voting model.
- Identification of the best performance based model for network intrusion detection system.

Paper outline:
The remainder of this article is organized as follows: Some recent works in this area is addressed in Section 2. Section 3 gives a detailed description of our proposed approach. Section 4 discusses in detail the experimental outcomes based on the proposed machine learning schemes. Finally, Section 5 summarizes this study and identifies some future research directions.

2. RELATED WORKS

The first concept of intrusion detection was proposed in 1980 by James P. Anderson [4]. He defined a model for monitoring user behavior in order to detect anomalies. Thereafter, a several works have been done in this field, the initial purpose was to protect the corporate network. Currently, with the fourth industrial revolution, Industry 4.0, connectivity is becoming more and more ubiquitous in our personal lives and enterprise activities, which requires a higher level of security than before. Thus the use of IDS based on machine learning techniques is considered as one of the main areas of research in cyber security. In this section, we present some approaches proposed to tackle the problem of detecting network attacks.

The paper [5] focuses on the importance of IDS and evaluates the performance of the 10 most popular machine learning algorithms using the NSL-KDD dataset and ranks them on the basis of their performance. The experimental analysis of the top four algorithms namely: Random Forest, PART, J48 and Bagging reveals that they take a long time to build the model. Therefore, they are chosen for an evaluation in combination with different selection and reduction feature methods. The experimental results show that the time to build model is significantly reduced using a small set of features without affecting accuracy. In this work, the top four algorithms performed well in terms of time to build the model with selection and reduction features, but Random Tree is the only algorithm that reaches good accuracy in a comparatively shorter time without using feature selection and reduction methods.

In the paper [6], the authors examined some specific attacks in the NSL KDD intrusion detection dataset that can have an impact on IoT environment components. In addition, in order to detect attacks, they studied eleven machine learning algorithms namely: Decision Tree, XGBoost, Bagging Tree, Random Forest, Bayes Net, Support Vector Machine, Naïve Bayes, AdaBoost, Expectation Maximization, DBSCAN and K-Means. In this study, authors demonstrated that tree-based and ensemble methods outperformed the other machine learning methods studied. Experimental results showed that XGBoost achieves the highest accuracy of 97%, a MCC (Matthews correlation coefficient) of 90.5%, and an AUC (area under the curve) of 99.6%. and they found also that the Expectation Maximization algorithm performs well in detecting attacks in the NSL KDD dataset and outperforms the accuracy of Naïve Bayes by 22.0%.

In [7], the authors designed a hierarchical intrusion detection system based on the original symmetric combination of the machine learning method with the knowledge based approach in order to support the detection of both the existing and the new types of network attacks. They evaluated the IDS using the KDD99 dataset.

The objective of their work is to combine the Hierarchical Multi-Layer Model approach by employing multiple predictive models in order to detect intrusions at several levels in the taxonomy of attack types and then combining them with the knowledge generated from ontology's field specific knowledge.

The predictive models distinguish normal connections from attacks and then predict classes and concrete attack types. While the knowledge model allows to navigate through the attack and to select the most relevant model to make a prediction on the selected level.

The authors describe clearly the high complexity of the hierarchical and ensemble models in terms of training. Because each one is trained separately, the resource and time requirements for training are considerably higher compared to the other models, and if this approach is deployed in a real environment with dynamic and continuously changing data flows, the limitation will be significant. Thus, it is necessary to use either concept drift detection approaches, in order to add new classes, or make a periodic retraining of the models. Particularly in the case of ensemble learning, to update the predictive models at the earliest possible time, in order to prevent missing new attacks. For complex ensemble approaches, if a model of domain knowledge is used, updates of the structure of model have to be applied to add new types of attacks to the taxonomy.

In paper [8], the authors proposed a machine learning-based security model, the IntruDTree (Intrusion Detection Tree), which considers the ranking of security features according to their importance, and then builds a tree based generalized intrusion detection model with the selected important features.

In the first step the authors proceed with the preparation of data which includes both encoding and scaling feature of the intrusion dataset. Then, they showed the score of importance for each attribute in the given dataset to reduce the irrelevant features. Therefore, this can help to make a data-driven generalized of security model using a reduced feature set.

This model was compared with several popular traditional machine learning methods, namely: naive bayes, k-nearest neighbor, logistic regression and support vector machines. The results show its effectiveness in terms of prediction accuracy, also, reducing dimensions of the features allows to minimize the computational complexity of the model. However, they used an intrusion data set composed of two class categories: normal and anomaly and did not apply their model for each category of anomaly or intrusion.

From the previous works cited above, it can be seen that intrusion detection systems are largely used to detect and predict intrusions in networks, for protecting them from attacks and vulnerabilities. Currently, machine learning techniques are widely employed to construct an effective IDS, such as classification, attribute selection, ensemble methods and several others.

The majority of previous works have examined machine learning algorithms to predict and detect network intrusions. Nevertheless, these researches have mainly focused on the specific impacts of different machine learning methods. In these works, the authors use either feature reduction methods, optimization methods, or ensemble learning techniques. There is no comprehensive framework that includes all these techniques for network intrusion detection systems, hence our motivation to propose a hybrid approach that combines feature reduction/selection methods, and ensemble classifiers. On the other hand, in our previous work [9-10-11], different machine learning algorithms were tested, and then the particle swarm optimization method was applied in a hybrid approach, in the same context of intrusion detection

in the industry 4.0 environment, and we have achieved a very interesting and motivating results. However, we have never applied ensemble learning models, and from the research we have done, which is cited in the section 3.3, the ensemble learning approaches provide new advantages such as improving detection accuracy and reducing error rate.

In this research paper, we have applied the following algorithms: Naïve Bayes, Support Vector Machine, Simple Logistic regression, JRip, Part and J48, as they are the most popular and used ones, also by the fact that certain of them (Naïve Bayes, Support Vector Machine and J48) have been evaluated in our previous studies [9-10-11], and they have shown good performances.

The choice of the suitable methods is essential to build an effective IDS, for this purpose we will test the three feature reduction methods, that are currently most used in the research works, in order to integrate the most efficient one in our approach. Then we will join it with Artificial Bee Colony (ABC) algorithms to optimize feature selection. We have chosen to apply the ABC algorithm in our approach as it has been largely used and adapted to solve optimization problems in many fields of application. Moreover, ABC is simple, easy to be implemented and it is capable of producing extremely good results with low computational cost [12]. Afterwards we will combine the three best classifiers into an ensemble classifier to improve the classification efficiency and achieve more effective results.

To evaluate our model, we will use the KDD NSL dataset, since it has solved some of the problems inherent in the KDD'99 dataset, and it is considered to be an effective benchmark dataset that helps researchers and scientists to evaluate different intrusion detection algorithms.

3. PROPOSED APPROACH

3.1 Machine Learning Algorithms used for Anomaly Detection:

For our study we will use the following classifiers:

3.1.1 Naïve Bayes

A Naïve Bayes (NB) is a simplified and widely used Bayesian probability model. This classifier is based on a strong hypothesis of independence. This means that the probability of an attribute does not affect the probability of another. The NB classifier often performs well, and has been effective in several practical applications, like system performance management, medical diagnosis and text classification [13].

3.1.2 Support Vector Machine

Support vector machine (SVM) is one of the most used and powerful algorithm nowadays, this classifier is part of supervised learning methods, under which various types of data from different subjects are trained. In a large dimensional space, the SVM sets up one or multiple hyperplanes. The hyperplane that optimally separates the data into different classes with the largest partition is considered the best one. To assess the margins between hyperplanes, a nonlinear classifier implements several kernel functions [14-15]. The concept is to maximize the margin between the closest points of the classes to find the optimal hyperplane of separation between two classes. Consider the case of a linear discriminant function obtained by linear combination of the input vector [16],

$x=(x_1,x_2,..x_n)^Y$, and weight vector $w=(w_1,w_2,.. w_n)^Y$:

$$f(x)= w^Yx+w0. \quad (1)$$

The hyperplane will satisfy:

x is class 1 if $w^Yx+w0 \geq 0$;

x is class -1 if $w^Yx+w0 < 0$.

while f(x)=0 is a separating hyperplane.

Usually, a classification task mainly involves dividing the data into two sets, namely training and test data sets. In the latter, the class label will be defined as "target variable" and the attributes will be defined as features or "observed variables" [17].

3.1.3 Simple Logistic Regression

Simple Logistic Regression (SL) creates the best fitting model to build a relationship or dependence between the features and class variable. For a given test case involving only two classes: 0 and 1, it essentially predicts a value between 0 and 1 as the probability that the class is 1 for a specific observation. The simple logistic model is only appropriate for the binary classification, but it can, with some effort, be extended to multi-class purposes. A linear expression for x is formed as follows:

$$Øy=Ø_0+Ø_1y_1+Ø_2y_2+…….Ø_nx_n \quad (2)$$

Where θ and y are vectors [$Ø_0,Ø_1,Ø_2,Ø_3….Øn$] and [$y_0,y_1,y_2,y_3…….yn$] respectively [18].

3.1.4 JRip

JRip classifier uses the repeated incremental pruning to provide method of error reduction. It includes the association rules with the error reduction delimitation. JRip divides the dataset into incremental and pruning sets, by generating rules for a subset of the training samples and removing

all samples covered by these rules for the training set over all samples [19].

3.1.5 Partial Decision List

Partial Decision List (Part) is a decision list algorithm based on a partial decision tree, combining the advantages of C4.5 classifier and PIPPER. for all instances a pruned decision tree is built, by constructing a rule for the leaf node corresponding to the largest coverage, then eliminating the tree and continuing [20].

3.1.6 J48

C4.5 (J48) is a popular algorithm in classification and data mining, developed by Ross Quinlan. it is employed to create a decision tree. This algorithm uses The gain ratio technique as a criterion for dividing the data set. Some standardization methods are implemented on the information gain through a value of " split information" [20].

3.2 Ranking and Determining Feature Importance

Feature selection methods reduce the size of the sample set, the features are ranked and those that are most appropriate for application in the machine learning algorithm are filtered out while irrelevant features are removed. Thus, the performance of the learning algorithms is improved. In our approach, we applied and tested correlation attribute evaluation, information Gain attribute evaluation and OneR attribute evaluation to choose the best one and combine it with ABC algorithm.

3.2.1 Correlation-based Attribute Ranking

CFS is a simple filtering algorithm that evaluates subsets composed of vectors of attributes, using a correlation-based heuristic evaluation function. The evaluation function favors subsets that are correlated with the class label independently of each other. The CFS method considers that irrelevant features have to be ignored because they will have a low correlation with the class, while redundant features have to be eliminated because they will be highly correlated with one or more of the remaining features.
Based on this concept, the criteria used to evaluate a set of characteristics can be expressed as follows:

$$M = \frac{k\overline{avg\left(corr_{fc}\right)}}{\sqrt{k + k(k-1)\overline{avg\left(corr_{ff}\right)}}} \qquad (3)$$

Where M is the ranking criterion for evaluating the feature set containing K features, it presents the correlation between the feature set and the dependent class. avg($corr_{fc}$) is the average feature-class correlation, and avg($corr_{ff}$) is the average feature-feature correlation [21].

3.2.2 Information Gain Attribute Evaluation

Information gain (InfoGain) is a widely used measure in the areas of machine learning and information theory. It is a feature selection method based on scoring techniques for the rating or weighting of continuous attributes that are discredited using maximum entropy.
The information gain of feature X is given in the following equation:

$$InfoGain(X) = F(Y) - F(Y|X) \qquad (4)$$

Where F(Y) is the entropy of Y and F(Y |X) is the conditional entropy of Y for given X. The level of importance of a feature is determined by the magnitude of the decrease in the class entropy when considered individually with the corresponding feature [22].

3.2.3 OneR attribute evaluation

One Rule (OneR) is a simple classification algorithm that assesses features based on error rate. This algorithm uses an accurate rule with a very simple approach, it creates a rule for each feature in the dataset and then selects the rule with the lowest error rate. It can handle categorical features, thus if the features have numerical values, it uses a simple method to split the set of values into several distinct intervals. It deals with missing values by treating "missing" as a legitimate value [23].

In the first step of the study we demonstrate the importance score of each feature in the NSL KDD dataset using the three filter methods mentioned above. From Table 1 we observe that the values of the importance score of all features are not the same for the given dataset and can be varied depending on the filter method used. For example the feature dst_host_same_srv_rate has the score of 0.57626629 using the information gain evaluation method while it has a score of 0.62451using the correlation attribute evaluation method. Table 1 shows the values of the10 best ranked features with their importance score in a descending order, where the score values are ranked from the highest to the lowest number.

Table 1:Top 10 ranked features with corresponding importance score values for NSLKDD dataset using filter methods.

Atribute evaluator & search method	Ranking	Feature selected	Score value
CorrelationAttributeEval +Ranker	01	29 same_srv_rate	0.69958
	02	39 dst_host_srv_serror_rate	0.65511
	03	38 dst_host_serror_rate	0.65256
	04	25 serror_rate	0.65103
	05	26 srv_serror_rate	0.64978
	06	33 dst_host_srv_count	0.62863
	07	34 dst_host_same_srv_rate	0.62451
	08	4 flag	0.60828
	09	12 logged_in	0.6049
	10	23 count	0.53715
InfoGainAttributeEval +Ranker	01	5 src_bytes	1.03221019
	02	3 service	0.86007429
	03	30 diff_srv_rate	0.73046279
	04	4 flag	0.70572521
	05	6 dst_bytes	0.66215121
	06	29 same_srv_rate	0.66015685
	07	35 dst_host_diff_srv_rate	0.64983015
	08	23 count	0.60058163
	09	33 dst_host_srv_count	0.59783364
	10	34 dst_host_same_srv_rate	0.57626629
OneRAttributeEval+ Ranker	01	5 src_bytes	92.6032
	02	3 service	86.967
	03	30 diff_srv_rate	86.5717
	04	4 flag	85.5199
	05	29 same_srv_rate	85.0778
	06	35 dst_host_diff_srv_rate	83.2583
	07	25 serror_rate	81.523
	08	6 dst_bytes	81.4397
	09	38 dst_host_serror_rate	81.3825
	10	34 dst_host_same_srv_rate	80.9793

3.2.4 Artificial Bee Colony Algorithm

Artificial Bee Colony (ABC) is an intelligent swarm algorithm proposed by Karaboga [24], and it has been extensively applied in several areas to solve optimization issues. The ABC algorithm is a novel metaheuristic based on the natural behavior of bees when foraging for food.

Bees use a very efficient mechanism of movement for searching food. They use a set of wriggling dances as a means of communication with each other. These processes allow the bees to share information about the quantity of nectar, direction and distance found by the different members.

The Artificial Bee Colony Algorithm is composed of three types of bees: Employed Bees (EB), Onlooker Bees (OB) and Scout Bees (SB). First, the positions of food source are determined, and each EB is allocated to a food source and transmit the nectar information to the OB. OBs wait for the return of the EB to the dance field to observe their dance and collect information about the nectar sources they have found. According to the information provided by the EBs, the OBs exploit the food sources and their surroundings until the food sources are completely exhausted. The EB of depleted food source becomes a scout, and tries to make changes to its current position to discover randomly a new food source, and to change its status again from scout to EB.

After determining the new location of each food source, another process of the ABC algorithm begins. The entire of this process is iterated repeatedly until the final condition is satisfied.

Information about nectar determines the quality of the available solution of a food source. A greater amount of nectar enhances the probability of choosing a particular food source by bees [25-26].

In the ABC algorithm, the food source position corresponds to a possible solution for the optimization problem and the nectar quantity of a food source represents the quality of the related solution.

The principal steps of the algorithm are as follows [27]:

- Start the process and initialize the population of solutions using the following equation:

$$Y_{ij} = Y_{min,j} + rand(0,1)(Y_{max,j} - Y_{min,j}) \quad (5)$$

Where Y_{ij} presents the food source values, i=1,…..S/2, S is the size of colony, j=1,….V, V is the variable number which will be optimized.

- Assess the population using a pre-defined function.
- Reiterate the following process until achieve the maximum iteration.
- Generate and evaluate new solutions N_{ij} in the neighborhood of Y_{ij} for the EBs using the equation mentioned below:

$$N_{ij} = Y_{ij} + Ø_{ij} (Y_{ij} - Y_{kj}) \quad (6)$$

here $Ø_{ij}$ presents a random value in the range of [1,1].

- Perform the process of selection between Yi and Ni.
- Determine the fitness measurements of the solutions.
- Generate and evaluate new solutions Ni for OBs using the solutions Yi given in the previous section.
- Apply the process of selection between Yi and Ni for the OBs.
- Replace SB with a new randomly generated solution using equation (5).
- Memorize the optimal position of the food source obtained so far.

The flowchart of the ABC algorithm is shown in Figure1.

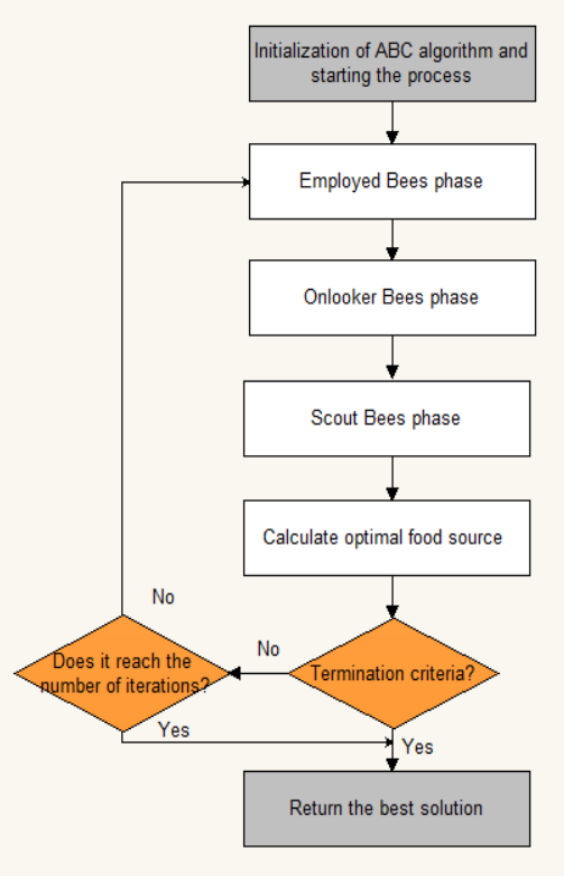

Figure.1 The flowchart of the ABC algorithm

In our scenario, the features selected by the ABC algorithm are: src_bytes, service, diff_srv_rate, flag, same_srv_rate, serror_rate, dst_bytes, dst_host_serror_rate, with population size of 30 and the number of iterations is 20.

3.3 Ensemble learning

An ensemble classifier is an approach that involves combining multiple machine learning algorithms to improve the performance of the classification, and to make a more effective and efficient technique. In ensemble methods, a problem is divided into smaller sub-problems that are easy to be analyzed and solved. The main advantage of this approach is that it achieves more accurate results than using a single algorithm [28].

In this article, after selecting the three best algorithms we combined them using two different ensemble classifier techniques called majority voting and stacking. We then evaluated them and analyzed the results in order to choose our definitive model to build a hybrid and efficient NIDS.

3.3.1 Majority Voting

Majority voting was considered as one of the most popular and useful voting techniques. As the combination of several classifiers has several advantages, such as improving robustness, achieving better accuracy and providing very high generalization. Majority voting has been utilized by several researchers who use base classifiers to achieve better outcomes. Voting for a class is done by each base classifier, and the final class label is the one which gets more than half of the votes. If no class label receives more than half of the votes, the majority voting method makes no prediction, or the option of one of the base classifiers is selected [29].

Many algorithms in the literature construct a majority vote from different kinds of voters. Boosting algorithms such as AdaBoost, typically combine decision stumps. Whereas random forests iteratively build more complex decision trees and then use a vote, which gives the same weight to all votes. The last layer of a neural network can also be seen as a majority vote, whose voters are defined by the structure of the network. Finally, kernel machines algorithms such as support vector machines can also be seen as algorithms returning a majority voting, since they return a classifier formed by a weighted sum of kernel functions [30].

3.3.2 Stacking

Stacking is one of the ensemble methods that have been successful in a variety of issues and have shown that it is able to tackle several machine learning challenges such as recall bias and accuracy. This process involves combining multiple classifiers to improve efficiency. The classifiers are combined step by step, in which the output of the first classifier is given as input of the second one. The stacking approach is implemented in two steps: The first one is a base learner in which the dataset is used on different models. A new dataset is acquired and its instances are used for prediction. In the second step, the new dataset is considered as input and the model gives the final output [31].

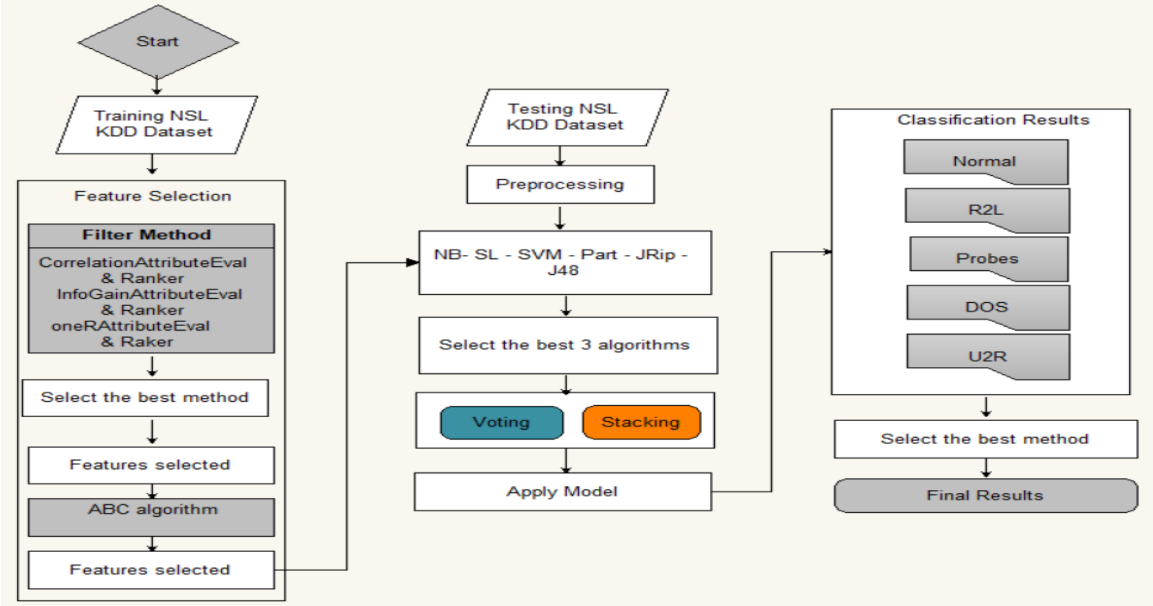

Figure 2. The proposed architecture

4. EXPERIMENT

4.1. NSL-KDD intrusion dataset

In this experience, we use the NSL KDD dataset, which was provided to solve some issues in KDD99 dataset, in that its training and test sets had a massive number of redundant records. Thus the classification algorithms may therefore be biased by these redundant records, preventing them from classifying the remaining records [32].

There is a set of data files proposed by the Canadian Institute for Cybersecurity, in this experiment we have used:

- KDDTrain+.ARFF which contains the complete NSL KDD train set including binary labels in ARFF format.
- KDDTrain+.TXT which contains the complete NSL KDD train set with labels for attack type and difficulty level in CSV format.

- KDDTest+.ARFF that represents the complete NSL KDD test set including binary labels in ARFF format.
- KDDTest+.TXT that represents the complete NSL KDD test set with labels for attack type and difficulty level in CSV format [32].

The dataset contains 42 attributes (Table3), with 125973 instances in KDDTrain+.ARFF and 22544 instances in KDDTest+.ARFF itemized in Table2, which represents the number of records for each attack category, namely: R2L (Remote to Local Attack), Probing Attack, DoS (Denial of Service) and U2R (User to Root Attack).

Table 2: Number of records in the train and test sets

Class	NSL KDD Train+	NSL KDD Test+
Normal	67343 (53.45%)	9711 (43.07%)
R2L	995 (0.79%)	2887 (12.81%)
Probes	11656 (9.25%)	2421 (10.74%)
DOS	45927 (36.47%)	7458 (33.08%)
U2R	52 (0.04%)	67 (0.30%)

Some unknown attacks that are not in the training set are assigned to the test set to evaluate the detection capability of these attacks, as shown in Table 4.

Table3:NSLKDD Dataset attributes

Feature No	Feature name	Feature No	Feature name
1	duration	22	is_guest_login
2	protocol_type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login	42	Class

Table 4: List of attack in NSL KDD Train+ and NSL KDD Test+

Attack Caterogy	Attack type which exist in both NSL KDD Train+ and NSL KDD Test+	Attack which is only in NSL KDD Test+
R2L	ftp_write, guess_passwd, imap, multihop, phf, warezmaster	Httptunnel, named, sendmail, snmpgetattack, snmpguess, worm, xlock, xsnoop
Probes	Ipsweep, nmap, portsweep, satan	Mscan, saint
DOS	**Back,** land, neptune, pod, smurf, teardrop	apache2, mailbomb, processtable, udpstorm
U2R	buffer_overflow, loadmodule, rootkit, perl	Ps, sqlattack, xterm

The experiment is performed on a computer with an Intel(R) Core™ i7 - 2760QM CPU, 16GB of RAM, and Windows 10 Professional operating system.

In order to evaluate the performance of the proposed approach, we have conducted several simulations using the Weka (Waikato Environment for Knowledge Analysis) software tools. which is a set of machine learning algorithms designed to facilitate the application of machine learning methods to solve challenging real world problems. It is supplied with tools for data preparation, classification, clustering, regression, association rule mining and visualization [33].

4.2. Results and discussion

Our approach reduces the dimensionality considerably and eliminates the irrelevant features from the dataset, for this purpose we will first identify the ten most important features by using three approaches namely correlation attribute evaluation, Information Gain attribute evaluation and OneR attribute evaluation. Table 1 shows the names of the best features selected by each method for the NSL-KDD dataset, next we select the best

technique and consider its result as the initial values of the ABC optimization approach. Finally, we select the three best algorithms and combine them into a stacking model and a majority voting model to improve significantly the predictive performance of the NIDS afterwards we evaluate them using several measures of detection, including accuracy, precision, detection rate, F-Measure, recall, time to build model, confusion matrix, attack detection rate through true positive rate and false positive rate. The metrics used are described as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (7)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad (8)$$

$$Specificity = \frac{TN}{FP+TN} \quad (9)$$

$$Precision = \frac{TP}{TP+FP} \quad (10)$$

$$F-measure = \frac{2*Precision*Sensitivity}{Precision+Sensitivity} \quad (11)$$

where:

TP (True Positive): when predicted network attack is in fact an attack,

TN (True Negative): when predicted normal record is in fact normal record,

FN (False Negative): when predicted normal record is in fact an attack,

FP (False Positive): when predicted network attack is in fact a normal record.

In the first step, we evaluate the efficiency of all classifiers in terms of time to build the model, correctly classified instances, incorrectly classified instances and accuracy. The results are presented in Table5 without optimization, Table6 optimized by Correlation attribute evaluation, Table 7 optimized by Information Gain attribute evaluation, Table 8 optimized by OneR attribute evaluation and Table 9 optimized by OneR attribute evaluation and ABC algorithm.

As we can see from Table 5, the algorithms consume a lot of time to build the model, SL consumes 1538.93 seconds followed by SVM with 406,4 seconds, JRip with 250,46 seconds, then Part which tikes 45,62 seconds, J48 tikes 32,67 seconds and finally NB takes the shortest time 0,75seconds, which is unacceptable in the field of intrusion detection. In order to tackle this issue, we have applied the feature selection and reduction methods cited above, and from the obtained results (as shown in Tables 6,7 and 8) we can clearly select the method OneR since it gives the most efficient and optimal results and combine it with ABC method to optimize the results again (Table9).

In terms of accuracy, we can notice that the correlation attribute evaluation method did not give optimized results for any classifier, whereas information gain method allowed to optimize the detection of intrusions from 71.21% to 71.30 for NB, from 76.40% to 76.97% for JRip, and from 75.25% to 75.93% for J48. The OneR method has not only reduced the time to build the model but also improved the accuracy of the algorithms, which increased from 74.64% to 77.15% for SL, and from 75.25% to 76.70% for J48.

From the results shown in table 9, we can see that the classifiers optimized by OneR and ABC methods give more efficient results both for the time to build the model and for the efficiency of the algorithms, as it improved the precision of JRip and J48, which have achieved an accuracy score of 77.99% and 78.47% respectively.

Table5.Classifiers Performance without optimization

Classifiers	NB	SL	SVM	Part	JRip	J48
Time to build model (s)	0,75	1538.93	406,4	45,62	250,46	32,67
Correctly Classified instances	16053 (71.21%)	16826 (74.64%)	17388 (77.13%)	17389 (77.13%)	17224 (76.40 %)	16966 (75.25 %)
Incorrectly classified instance	6491 (28.79%)	5718 (25.36%)	5156 (22.87%)	5155 (22.87) %	5320 (23.60 %)	5578 (24.75 %)

Table6.Classifiers performance optimized by Correlation Attribute Evaluation

Classifiers	NB	SL	SVM	Part	JRip	J48
Time to build model (s)	0,19	62,54	1169,54	22,7	147,71	5,09
Correctly classified instances	12846 (56,98%)	14628 (64,89%)	14751 (65,43%)	15710 (69,69%)	15746 (69,85%)	15927 (70,65%)
Incorrectly classified instance	9698 (40,02%)	7916 (35,11%)	7793 (34,57%)	6834 (30,31%)	6798 (30,15%)	6617 (29,35%)

Table7.Classifiers performance optimized by Information Gain Attribute Evaluation

Classifiers	NB	SL	SVM	Part	JRip	J48
Time to build model (s)	0,22	484,4	681,76	9,2	145,84	5,24
Correctly classified instances	16076 (71,30%)	16779 (74,43%)	16826 (74,64%)	16811 (74,57%)	17352 (76,97%)	17117 (75,93%)
Incorrectly classified instance	6468 (28,70%)	5765 (25,57%)	5718 (25,36%)	5733 (25,43%)	5192 (23,03%)	5427 (24,07%)

Table8.Classifiers performance optimized by OneR Attribute Evaluation

Filter Method	NB	SL	SVM	Part	JRip	J48
Time to build model (s)	0,22	468,48	305,01	7,16	118,33	4,16
Correctly classified instances	15864 (70,37%)	17392 (77,15%)	17308 (76,77%)	17123 (75,95%)	17083 (75,78%)	17291 (76,70%)
Incorrectly classified instance	6680 (29,63%)	5152 (22,85%)	5236 (23,23%)	5421 (24,05%)	5461 (24,22%)	5253 (23,30%)

Table9.Classifiers optimized by OneR Attribute Evaluation and ABC method

Classifiers	NB	SL	SVM	Part	JRip	J48
Time to build model (s)	0,14	296,76	228	5,7	84,48	3,23
Correctly classified instances	15965 (70,82%)	17078 (75,76%)	16579 (73,54%)	17269 (76,60%)	17581 (77,99%)	17689 (78,47%)
Incorrectly classified instance	6579 (29,18%)	5466 (24,24%)	5965 (26,46%)	5275 (23,40%)	4963 (22,01%)	4855 (21,53%)

To verify the effectiveness of our model, we evaluate the precision measures of normal traffic and each category of attack based on the values of TP rate, FP rate, precision, recall and F-Measure. Tables 10,11,12,13,14 and 15 provide the performance comparison of NB, SL, SVM, Part, JRip and J48 respectively.

We can see that the best results are those generated by the Classifiers optimized by OneR and ABC methods. JRip, Part and J48 show the best results compared to the other classification algorithms.

The average precision value of the best performance without optimization is for SVM with 80.90%, then J48 with 80.50%, followed by JRip with 80.40% then Part with 77.80%, NB occupies the fifth position in detection precision with76%, and the SL algorithm offers the least percentage of precision (74.80%). After the optimization by OneR and ABC methods, we find the best precision performance for JRip with 81.60% then Part with 79.30%, followed by J48 witch achieves 74.20%. NB, SL and SVM offer the least amount of precision percentage.

With this model, Part improves the detection of R2L attacks by 20%, JRip enhances the detection of Probes attacks from 64.80% to 85.10%, DOS from 95.50% to 97.20%, and U2R from 80.00% to 88.90%, J48 increases the detection of DOS from 95.10% to 97.00%, and U2R from 69.20% to 80.00%. Nevertheless, each of these algorithms has some weaknesses in detecting other categories of attacks, as clearly demonstrated by the detailed results produced in the tables 10,11,12,13,14 and 15 which indicate that Part produces a TP rate of 4.40% for R2L and 19.40% for U2R, JRip has a rate of 11.70% for R2L, 69.40% for Probes and 23.90% for U2R, while J48 gives the lowest TP rate for R2L with 1.20% and a score of 11.90% for U2R. To overcome this limitation and to improve the accuracy and the TP rate of all categories of attacks, we have used two ensemble classifiers and we will present the obtained results in the next section.

Table10. Performance Comparison of NB based on Different Methods

	Class	TP Rate	FP Rate	Precision	Recall	F-Measure
NB without optimization	**normal**	86,90%	23,90%	73,30%	86,90%	79,50%
	R2L	10,00%	1,30%	52,40%	10,00%	16,80%
	Probes	81,60%	3,20%	75,40%	81,60%	78,30%
	DOS	71,10%	4,10%	89,60%	71,10%	79,30%
	U2R	70,10%	8,40%	2,40%	70,10%	4,70%
	Weighted Avg.	71,20%	12,20%	76,00%	71,20%	71,10%
NB optimized by InfoGainAttributeEval	**normal**	88,90%	32,00%	67,70%	88,90%	76,90%
	R2L	39,70%	4,00%	59,30%	39,70%	47,60%
	Probes	59,30%	4,00%	64,00%	59,30%	61,60%
	DOS	65,00%	1,80%	94,80%	65,00%	77,10%
	U2R	14,90%	2,20%	2,00%	14,90%	3,50%
	Weighted Avg.	71,30%	15,30%	75,00%	71,30%	71,40%
NB optimized by OneRAttributeEval	**normal**	96,60%	41,10%	64,00%	96,60%	77,00%
	R2L	0,10%	0,10%	10,00%	0,10%	0,20%
	Probes	57,70%	2,20%	76,10%	57,70%	65,60%
	DOS	68,10%	1,80%	94,80%	68,10%	79,30%
	U2R	6,00%	2,90%	0,60%	6,00%	1,10%
	Weighted Avg.	70,40%	18,60%	68,40%	70,40%	66,50%
NB optimized by CorrelationAttributeEval	**normal**	84,10%	23,40%	73,10%	84,10%	78,20%
	R2L	45,50%	6,20%	52,10%	45,50%	48,60%
	Probes	54,30%	21,80%	23,10%	54,30%	32,40%
	DOS	27,50%	2,50%	84,70%	27,50%	41,50%
	U2R	3,00%	3,20%	0,30%	3,00%	0,50%
	Weighted Avg.	57,00%	14,00%	68,70%	57,00%	57,10%
NB optimized by OneRAttributeEval and ABC	**normal**	97,00%	42,00%	63,60%	97,00%	76,90%
	R2L	0,10%	0,10%	10,50%	0,10%	0,10%
	Probes	51,90%	0,80%	88,10%	51,90%	65,30%
	DOS	70,80%	2,40%	93,70%	70,80%	80,70%
	U2R	0,00%	2,90%	0,00%	0,00%	0,00%
	Weighted Avg.	70,80%	19,00%	69,20%	70,80%	66,80%

Table11. Performance Comparison of SL based on Different Methods

	Class	TP Rate	FP Rate	Precision	Recall	F-Measure
SL without optimization	**normal**	93,00%	37,30%	65,40%	93,00%	76,80%
	R2L	3,80%	0,50%	53,90%	3,80%	7,20%
	Probes	72,20%	1,30%	86,80%	72,20%	78,80%
	DOS	79,30%	3,70%	91,30%	79,30%	84,90%
	U2R	28,40%	0,00%	67,90%	28,40%	40,00%
	Weighted Avg.	74,60%	17,50%	74,80%	74,60%	70,70%
SL optimized by InfoGainAttributeEval	**normal**	92,50%	36,70%	65,60%	92,50%	76,80%
	R2L	13,10%	0,70%	73,90%	13,10%	22,20%
	Probes	57,20%	1,30%	84,50%	57,20%	68,20%
	DOS	81,00%	4,50%	90,00%	81,00%	85,20%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	74,40%	17,50%	76,60%	74,40%	71,40%
SL optimized by OneRAttributeEval	**normal**	97,20%	35,00%	67,80%	97,20%	79,90%
	R2L	15,50%	0,60%	78,10%	15,50%	25,80%
	Probes	60,10%	1,10%	87,00%	60,10%	71,10%
	DOS	81,10%	2,10%	95,00%	81,10%	87,50%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	77,10%	16,00%	80,00%	77,10%	74,30%
SL optimized by CorrelationAttributeEval	**normal**	93,60%	43,80%	61,80%	93,60%	74,40%
	R2L	0,00%	0,00%	0,00%	0,00%	0,00%
	Probes	21,60%	3,40%	43,20%	21,60%	28,80%
	DOS	67,30%	10,70%	75,70%	67,30%	71,20%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	64,90%	22,80%	56,30%	64,90%	58,70%
SL optimized by OneRAttributeEval and ABC	**normal**	97,30%	37,50%	66,30%	97,30%	78,80%
	R2L	0,00%	0,50%	1,00%	0,00%	0,10%
	Probes	65,30%	1,30%	86,10%	65,30%	74,30%
	DOS	81,10%	2,00%	95,30%	81,10%	87,60%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	75,80%	17,00%	69,50%	75,80%	70,90%

Table12. Performance Comparison of SVM based on Different Methods

	Class	TP Rate	FP Rate	Precision	Recall	F-Measure
SVM without optimization	**normal**	95,80%	35,60%	67,10%	95,80%	78,90%
	R2L	10,10%	0,30%	84,60%	10,10%	18,10%
	Probes	67,30%	1,80%	82,20%	67,30%	74,00%
	DOS	82,40%	1,10%	97,30%	82,40%	89,20%
	U2R	35,80%	0,00%	68,60%	35,80%	47,10%
	Weighted Avg.	77,10%	15,90%	80,90%	77,10%	73,90%

SVM optimized by InfoGainAttributeEval	**normal**	92,50%	37,30%	65,30%	92,50%	76,60%
	R2L	11,00%	0,10%	92,40%	11,00%	19,60%
	Probes	60,00%	1,10%	86,40%	60,00%	70,80%
	DOS	81,40%	4,50%	89,90%	81,40%	85,40%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	74,60%	17,70%	79,00%	74,60%	71,40%
SVM optimized by OneRAttributeEval	**normal**	96,70%	35,90%	67,10%	96,70%	79,20%
	R2L	13,70%	0,10%	95,90%	13,70%	24,00%
	Probes	62,60%	1,30%	85,30%	62,60%	72,20%
	DOS	80,60%	2,30%	94,60%	80,60%	87,00%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	76,80%	16,40%	81,60%	76,80%	73,70%
SVM optimized by CorrelationAttributeEval	**normal**	92,50%	42,40%	62,30%	92,50%	74,50%
	R2L	0,00%	0,00%	0,00%	0,00%	0,00%
	Probes	28,70%	4,00%	46,40%	28,70%	35,50%
	DOS	68,00%	10,30%	76,60%	68,00%	72,00%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	65,40%	22,10%	57,20%	65,40%	59,70%
SVM optimized by OneRAttributeEval and ABC	**normal**	90,20%	32,30%	67,90%	90,20%	77,50%
	R2L	0,10%	0,10%	8,30%	0,10%	0,10%
	Probes	70,50%	7,60%	52,70%	70,50%	60,30%
	DOS	81,90%	1,80%	95,70%	81,90%	88,30%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	73,50%	15,30%	67,70%	73,50%	69,10%

Table13. Performance Comparison of Part based on Different Methods

	Class	TP Rate	FP Rate	Precision	Recall	F-Measure
Part without optimization	**normal**	97,40%	34,60%	68,10%	97,40%	80,10%
	R2L	3,80%	0,40%	60,00%	3,80%	7,20%
	Probes	69,50%	2,00%	80,70%	69,50%	74,70%
	DOS	82,10%	1,60%	96,30%	82,10%	88,60%
	U2R	32,80%	0,00%	78,60%	32,80%	46,30%
	Weighted Avg.	77,10%	15,70%	77,80%	77,10%	72,90%
Part optimized by InfoGainAttributeEval	**normal**	96,60%	35,80%	67,10%	96,60%	79,20%
	R2L	5,60%	0,10%	89,40%	5,60%	10,50%
	Probes	65,70%	3,90%	67,00%	65,70%	66,30%
	DOS	76,10%	2,00%	94,90%	76,10%	84,40%
	U2R	6,00%	0,10%	15,40%	6,00%	8,60%
	Weighted Avg.	74,60%	16,50%	79,00%	74,60%	70,50%
Part optimized by OneRAttributeEval	**normal**	91,30%	28,80%	70,60%	91,30%	79,60%
	R2L	5,70%	0,10%	89,10%	5,70%	10,70%
	Probes	75,90%	7,20%	56,00%	75,90%	64,50%
	DOS	83,80%	1,70%	96,00%	83,80%	89,50%

	U2R	14,90%	0,00%	76,90%	14,90%	25,00%
	Weighted Avg.	76,00%	13,80%	79,80%	76,00%	72,30%
Part optimized by CorrelationAttributeEval	**normal**	92,50%	40,60%	63,30%	92,50%	75,10%
	R2L	2,10%	0,90%	25,10%	2,10%	3,90%
	Probes	63,70%	3,80%	66,80%	63,70%	65,20%
	DOS	68,70%	4,50%	88,40%	68,70%	77,30%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	69,70%	19,50%	66,90%	69,70%	65,50%
Part optimized by OneRAttributeEval and ABC	**normal**	91,20%	29,20%	70,30%	91,20%	79,40%
	R2L	4,40%	0,20%	80,00%	4,40%	8,40%
	Probes	85,90%	6,20%	62,40%	85,90%	72,30%
	DOS	83,00%	1,60%	96,30%	83,00%	89,20%
	U2R	19,40%	0,00%	65,00%	19,40%	29,90%
	Weighted Avg.	76,60%	13,80%	79,30%	76,60%	72,60%

Table14. Performance Comparison of JRip based on Different Methods

	Class	**TP Rate**	**FP Rate**	**Precision**	**Recall**	**F-Measure**
JRip without optimization	**normal**	93,10%	32,30%	68,60%	93,10%	79,00%
	R2L	9,70%	0,10%	94,30%	9,70%	17,60%
	Probes	65,50%	4,30%	64,80%	65,50%	65,10%
	DOS	84,60%	2,00%	95,50%	84,60%	89,70%
	U2R	11,90%	0,00%	80,00%	11,90%	20,80%
	Weighted Avg.	76,40%	15,00%	80,40%	76,40%	73,00%
JRip optimized by InfoGainAttributeEval	**normal**	97,20%	36,20%	67,00%	97,20%	79,40%
	R2L	7,80%	0,00%	97,00%	7,80%	14,40%
	Probes	66,60%	2,00%	80,30%	66,60%	72,80%
	DOS	81,30%	1,00%	97,60%	81,30%	88,80%
	U2R	9,00%	0,00%	100,00%	9,00%	16,40%
	Weighted Avg.	77,00%	16,10%	82,50%	77,00%	73,30%
JRip optimized by OneRAttributeEval	**normal**	97,10%	36,20%	67,00%	97,10%	79,30%
	R2L	1,40%	0,10%	80,40%	1,40%	2,80%
	Probes	64,40%	1,60%	83,00%	64,40%	72,60%
	DOS	81,00%	3,20%	92,60%	81,00%	86,40%
	U2R	14,90%	0,00%	90,90%	14,90%	25,60%
	Weighted Avg.	75,80%	16,80%	79,00%	75,80%	71,00%
JRip optimized by CorrelationAttributeEval	**normal**	96,10%	45,70%	61,40%	96,10%	74,90%
	R2L	2,00%	0,10%	73,40%	2,00%	3,90%
	Probes	52,70%	2,90%	69,00%	52,70%	59,80%
	DOS	68,10%	2,20%	93,80%	68,10%	78,90%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	69,80%	20,80%	74,30%	69,80%	65,30%
JRip optimized by OneRAttributeEval and	**normal**	96,50%	34,50%	67,90%	96,50%	79,70%
	R2L	11,70%	0,30%	84,30%	11,70%	20,60%

ABC	**Probes**	69,40%	1,50%	85,10%	69,40%	76,50%
	DOS	82,80%	1,20%	97,20%	82,80%	89,40%
	U2R	23,90%	0,00%	88,90%	23,90%	37,60%
	Weighted Avg.	78,00%	15,40%	81,60%	78,00%	74,90%

Table15. Performance Comparison of J48 based on Different Methods

	Class	**TP Rate**	**FP Rate**	**Precision**	**Recall**	**F-Measure**
J48 without optimization	**normal**	97,00%	36,50%	66,80%	97,00%	79,10%
	R2L	6,20%	0,00%	95,70%	6,20%	11,70%
	Probes	65,50%	2,90%	72,90%	65,50%	69,00%
	DOS	77,40%	2,00%	95,10%	77,40%	85,30%
	U2R	13,40%	0,00%	69,20%	13,40%	22,50%
	Weighted Avg.	75,30%	16,70%	80,50%	75,30%	71,30%
J48 optimized by InfoGainAttributeEval	**normal**	96,90%	32,00%	69,60%	96,90%	81,00%
	R2L	1,20%	0,00%	87,20%	1,20%	2,30%
	Probes	83,20%	6,00%	62,60%	83,20%	71,50%
	DOS	75,80%	0,80%	98,00%	75,80%	85,50%
	U2R	6,00%	0,00%	80,00%	6,00%	11,10%
	Weighted Avg.	75,90%	14,70%	80,50%	75,90%	71,20%
J48 optimized by OneRAttributeEval	**normal**	97,10%	31,70%	69,90%	97,10%	81,30%
	R2L	1,70%	0,00%	94,10%	1,70%	3,30%
	Probes	67,00%	4,00%	67,10%	67,00%	67,00%
	DOS	82,90%	2,50%	94,30%	82,90%	88,30%
	U2R	9,00%	0,10%	24,00%	9,00%	13,00%
	Weighted Avg.	76,70%	14,90%	80,60%	76,70%	71,90%
J48 optimized by CorrelationAttributeEval	**normal**	91,90%	37,50%	65,00%	91,90%	76,10%
	R2L	1,70%	0,10%	69,40%	1,70%	3,40%
	Probes	62,20%	4,20%	64,00%	62,20%	63,10%
	DOS	73,00%	6,30%	85,20%	73,00%	78,70%
	U2R	0,00%	0,00%	0,00%	0,00%	0,00%
	Weighted Avg.	70,60%	18,70%	72,00%	70,60%	66,00%
J48 optimized by OneRAttributeEval and ABC	**normal**	96,20%	27,20%	72,80%	96,20%	82,90%
	R2L	1,20%	0,50%	27,30%	1,20%	2,30%
	Probes	82,60%	5,30%	65,00%	82,60%	72,70%
	DOS	84,50%	1,30%	97,00%	84,50%	90,30%
	U2R	11,90%	0,00%	80,00%	11,90%	20,80%
	Weighted Avg.	78,50%	12,80%	74,20%	78,50%	73,80%

In order to improve the performance measurement of the classifier, the simulation error is also addressed in this research. For this purpose, we evaluate the effectiveness of the classifiers in terms of mean absolute error and root mean squared error. The results are presented in Figures 3,4,5,6 and 7.

Journal of Theoretical and Applied Information Technology
15th July 2021. Vol.99. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

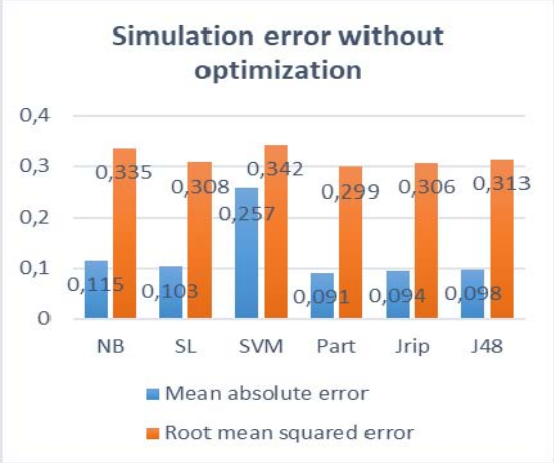

Figure3. Simulation error without optimization

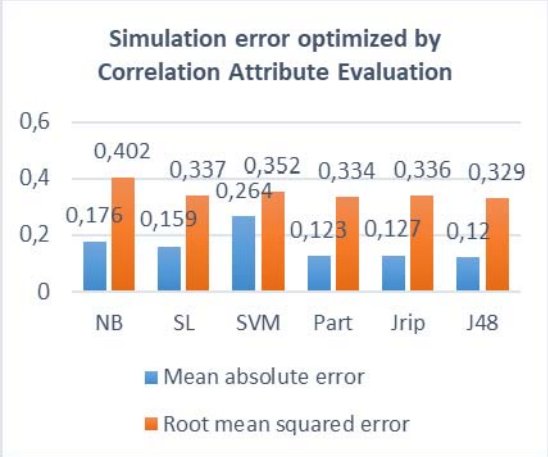

Figure4. Simulation error optimized by Correlation Attribute Evaluation

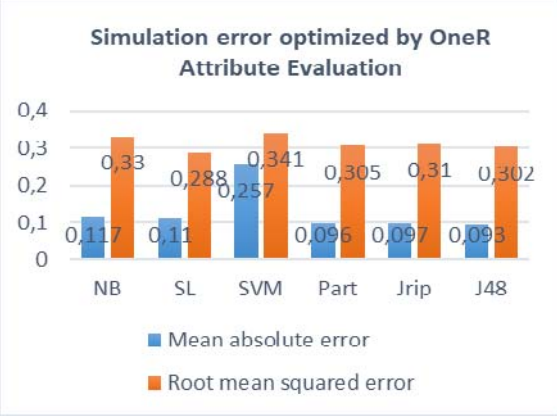

Figure5. Simulation error optimized by OneR Attribute Evaluation

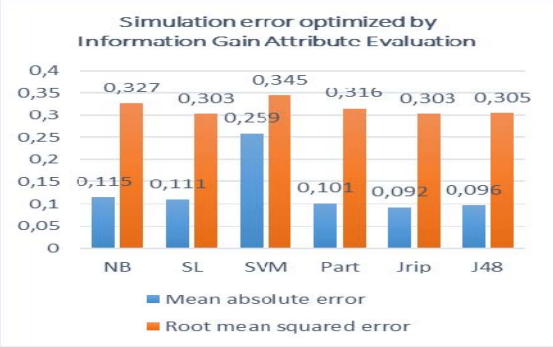

Figure6. Simulation error optimized by Information Gain Attribute Evaluation

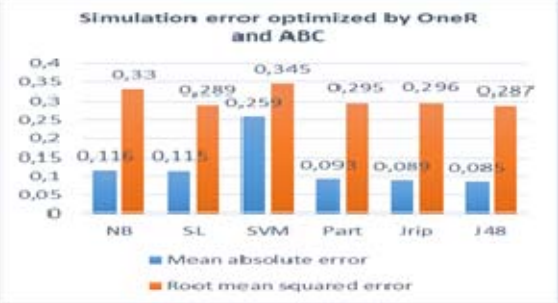

Figure7. Simulation error optimized by OneR and ABC Methods

To assess the performance of our proposed ensemble classifier, several experiments have been carried out using different classification algorithms. First, a stacking method is chosen, we use two algorithms as base learners and an algorithm as a stacking model learner. We use various combinations of the three best classifiers namely: Part, JRip and J48. The predicted classifications from the base learners will become the input variables in a stacking model learner.

The stacking model learner will experiment with learning from data how to combine the different models' predictions to reach high classification precision. The experiment results of the stacking algorithm are given in Table 16.

In the second step, the ensemble classifier based on majority voting is proposed, constructed with the same three classifiers. the detailed results are given in Table 17, and a performance comparison based time to build model, correctly classified instances, incorrectly classified instances, mean absolute error and root mean squared error is presented in Table 18 and Figure 8.

Table16.The performance of stacking algorithm optimized by OneR and ABC

Base Learner	Stacking Model Learner	Class	TP Rate	FP Rate	Precision	Recall	F-Measure
JRip	J48	normal	91,20%	27,00%	71,90%	91,20%	80,40%
		R2L	15,20%	0,50%	**83,10%**	15,20%	25,70%
		Probes	84,60%	5,80%	63,60%	84,60%	72,60%
Part		DOS	84,00%	1,20%	**97,10%**	84,00%	90,10%
		U2R	29,90%	0,00%	**80,00%**	29,90%	43,50%
		Weighted Avg.	78,20%	12,70%	**80,80%**	78,20%	75,70%
Part	Jrip	normal	91,20%	31,00%	69,00%	91,20%	78,60%
		R2L	4,50%	0,20%	**80,70%**	4,50%	8,50%
		Probes	82,50%	6,10%	62,10%	82,50%	70,80%
J48		DOS	82,30%	1,20%	**97,10%**	82,30%	89,10%
		U2R	11,90%	0,00%	**100,00%**	11,90%	21,30%
		Weighted Avg.	76,00%	14,40%	79,20%	76,00%	72,10%
JRip	Part	normal	96,10%	26,10%	73,60%	96,10%	83,40%
		R2L	11,70%	0,30%	**84,30%**	11,70%	20,50%
		Probes	75,80%	6,60%	58,10%	75,80%	65,80%
J48		DOS	81,80%	1,20%	**97,10%**	81,80%	88,80%
		U2R	23,90%	0,00%	**88,90%**	23,90%	37,60%
		Weighted Avg.	78,20%	12,40%	**81,10%**	78,20%	75,10%

Table17.Performance based Majority voting of JRip, Part and J48

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
normal	96,30%	31,60%	69,70%	96,30%	80,90%
R2L	1,00%	0,40%	26,10%	1,00%	2,00%
Probes	82,20%	3,40%	74,50%	82,20%	78,20%
DOS	82,50%	1,20%	97,20%	82,50%	89,20%
U2R	19,40%	0,00%	86,70%	19,40%	31,70%
Weighted Avg.	77,80%	14,40%	73,80%	77,80%	73,10%

Table18. Performance Comparison of ensemble methods

Evaluation criteria	Majority voting	Stacking by Part	Stacking by Jrip	Stackin by J48
Time to build model (s)	90,23	835,74	111,17	94,93
Correctly classified instances	17539 (77,80%)	17623 (78,17%)	17129 (75,98%)	17628 (78,20%)
Incorrectly classified instance	5005 (22,20%)	4921 (21,83%)	5415 (24,01%)	4916 (21,80%)

Journal of Theoretical and Applied Information Technology
15th July 2021. Vol.99. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

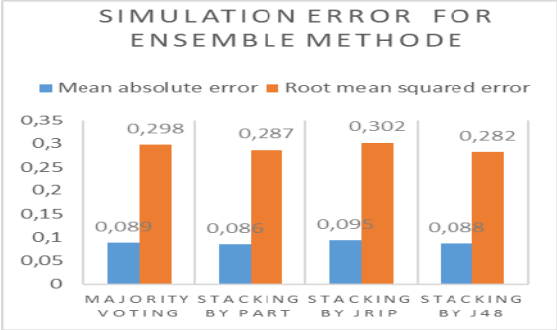

Figure8. Simulation error for ensemble methods

Confusion matrix

Confusion matrix is a useful tool for visualization, it allows to evaluate classifiers using the four basics values namely, TP, FP, TN and FN. Each row of confusion matrix shows the rates in a real class while each column represents the predictions.

From confusion matrices (Tables 19, 20, 21 and 22), we can clearly see that none of classifiers in the attack detection model performed very well in the detection of new R2L and U2R intrusions. The stacking model with J48 as a model learner and two other algorithms (Part and JRip) as base classifiers gives the best results compared to the others ensemble classifiers. It allows to increase the detection accuracy of R2L attacks to 15.20%, U2R to 29.85% while maintaining a detection accuracy of normal traffic of 91.17%, probes attacks of 84.55% and DOS with 84.04%.

Table19.Confusion matrix of Stacking by J48

normal	R2L	Probes	DOS	U2R	
8854	4	777	76	0	**normal**
2091	**439**	351	1	5	**R2L**
184	83	**2047**	107	0	**Probes**
1146	0	44	**6268**	0	**DOS**
45	2	0	0	**20**	**U2R**

Table20.Confusion matrix of Stacking by JRip

normal	R2L	Probes	DOS	U2R	
8859	2	779	71	0	**normal**
2406	**130**	351	0	0	**R2L**
290	24	**1997**	110	0	**Probes**
1233	0	90	**6135**	0	**DOS**
54	5	0	0	**8**	**U2R**

Table21.Confusion matrix of Stacking by Part

normal	R2L	Probes	DOS	U2R	
9337	3	295	76	0	**normal**
1843	**337**	705	0	2	**R2L**
419	60	**1835**	107	0	**Probes**
1039	0	321	**6098**	0	**DOS**
51	0	0	0	**16**	**U2R**

Table22.Confusion matrix of majority voting

normal	R2L	Probes	DOS	U2R	
9355	3	280	73	0	**normal**
2547	**30**	308	0	2	**R2L**
244	82	**1989**	106	0	**Probes**
1214	0	92	**6152**	0	**DOS**
54	0	0	0	**13**	**U2R**

We tested the proposed classifiers namely: Naïve Bayes, Support Vector Machine, Simple Logistic Regression, JRip, Part and J48, optimized by OneR attribute evaluation and ABC method, and then selected the three best classifiers to combine them using ensemble models (the stacking and the majority voting models) for the detection and classification of network intrusions according to their category. Table 23 compares our proposed classification approach with the results of previous research. Based on the existing methods and experimental findings, we observe that our optimized model performs better than other models alone for the prediction and classification of network intrusions. We cannot compare it to all the previous works since we did not use the same dataset, and even for the works based on NSL KDD dataset most of them did not use the data with the attack category label. Our model shows that no algorithm is very efficient in the detection of U2R and R2L attacks, however it has improved the accuracy of detection of these types of intrusion while reducing the time to build the model.

Table 23: Performance of different methods

Model	Techniques	Data set	Accuracy
Appraoch [5]	J48	NSL-KDD	99.79%
	Random Forest		99.91%
	Bagging		99.84%
	PART		99.83%
Appraoch [6]	XGBoost	NSL-KDD	97%
Appraoch [7]	Machine learning & Knowledge based approach	KDD 99	99.80%
Appraoch [8]	Feature encoding, Feature scaling & Features selection	Cybersecurity dataset	98%
Appraoch [9]	Random Forest	10% Random Sample Gas Pipeline	99.30%
	J48		99.16%
	SVM		94.20%
	Naive Bayes		94.15%
Appraoch [10]	Naïve Bayes	KDD 99	98.9%
	RandomTree		100%
Appraoch [11]	Hybrid Technique (PSO + RF)	Gas Pipeline	99.30%
Our proposed model	OneR, ABC and stacking model	NSL-KDD	80.80%

5. CONCLUSION AND FUTURE WORK

Many machine learning techniques have been applied to improve the efficiency of NIDSs, however, the existing intrusion detection algorithms are still challenged to perform well. In this article, we propose a new intrusion detection approach based on feature reduction/selection methods to deal with highly dimensional network traffic, and ensemble learning techniques to improve the detection accuracy. First, we have evaluated the Naïve Bayes, Support Vector Machine, Simple Logistic regression, JRip, Part and J48 algorithms using NSL KDD dataset. The experimental results show that these classifiers consume a lot of time to build the model, however it becomes necessary to remove irrelevant and redundant features and select the most important ones. Thus, we have tested three dimensionality reduction methods namely: information gain evaluation, correlation attribute evaluation and OneRule attribute evaluation. After the comparative analysis of the results, we have selected the best technique which is OneR attribute evaluation, and we have associated it with Artificial Bee Colony method, which has clearly allowed the algorithms to decrease the time of building the model, as well as to improve the efficiency of detection of Probes and DOS attacks. The challenge was to optimize the efficiency of detection of U2R and R2L intrusions. For this purpose, we have selected the three best classifiers and we have combined them in a stacking model, and another model based on majority voting. After this study we can conclude that no algorithm performs very well in detection of U2R and R2L attacks. However, the results of our research show that the stacking model, using J48 as the model learner and JRip with Part as the base classifiers, has improved the detection accuracy of R2L to 15. 20%, U2R to 29.85%, Probes to 84.55%, DOS to 84.04% and for normal traffic an accuracy score of 91.17%, while minimizing the time required to build the model.

Our outcomes will help future authors and searchers to figure out the appropriate feature reduction/selection schemes required to conceive effective machine learning approaches.

Still passionate about cybersecurity and artificial intelligence, in our future work, we will propose a convenient framework based on deep learning for network intrusion detection systems.

REFRENCES

[1] SINGH, Amrit Pal et SINGH, Manik Deep. "Analysis of Host-Based and Network-Based Intrusion Detection System". *International Journal of Computer Network & Information Security*, 2014, vol. 6, no 8.

[2] TAMY, Sara, NABILA, Rabbah, MAHMOUD ALMOSTAFA, Rabbah, et al. "Study of Strategies for Real-Time Supervision of Industrial Network Security". *Smart Application and Data Analysis for Smart Cities (SADASC'18)*, 2018.

[3] CAI, Jie, LUO, Jiawei, WANG, Shulin, et al. "Feature selection in machine learning: A new perspective". *Neurocomputing, 2018*, vol. 300, p. 70-79.

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

[5] : "Intrusion Detection using Machine Learning and Feature Selection", *I. J. Computer Network and Information Security*, 2019, 4, 43-52 Published Online April 2019 in MECS (http://www.mecs-press.org/)

[6] LIU, Jixin, KANTARCI, Burak, et ADAMS, Carlisle. "Machine learning-driven intrusion

detection for contiki-NG-based IoT networks exposed to NSL-KDD dataset". *In : Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*. 2020. p. 25-30.

[7] SARNOVSKY, Martin et PARALIC, Jan. "Hierarchical intrusion detection using machine learning and knowledge model". *Symmetry*, 2020, vol. 12, no 2, p. 203.

[8] SARKER, Iqbal H., ABUSHARK, Yoosef B., ALSOLAMI, Fawaz, et al. "Intrudtree: a machine learning based cyber security intrusion detection model". *Symmetry*, 2020, vol. 12, no 5, p. 754.

[9] TAMY, Sara, BELHADAOUI, Hicham, RABBAH, Mahmoud Almostafa, et al. An "evaluation of machine learning algorithms to detect attacks in SCADA network". *In : 2019 7th Mediterranean Congress of Telecommunications (CMT)*. IEEE, 2019. p. 1-5

[10] TAMY[1], Sara, BELHADAOUI, Hicham, RABBAH[1], Mahmoud Almostafa, et al. "Select the Best Machine Learning Algorithms for Prediction and Classification of Intrusions using KDD99 Intrusion Detection Dataset". *Indian Journal of Science and Technology*, 2019, vol. 12, p. 37.

[11] SARA TAMY, HICHAM BELHADAOUI, NABILA RABBAH,MOUNIR RIFI. "cyber security based machine learning algorithms applied to industry 4.0 application case: development of network intrusion detection system using hybrid method". *Journal of Theoretical and Applied Information Technology,2020,* Vol.98, No 12

[12] KARABOGA, Dervis, GORKEMLI, Beyza, OZTURK, Celal, *et al.* "A comprehensive survey: artificial bee colony (ABC) algorithm and applications". *Artificial Intelligence Review*, 2014, vol. 42, no 1, p. 21-57

[13] MUKHERJEE, Saurabh et SHARMA, Neelam. "Intrusion detection using naive Bayes classifier with feature reduction". *Procedia Technology*, 2012, vol. 4, p. 119-128.

[14] Lopez Perez, R., Adamsky, F., Soua, R., & Engel, T. "Forget the Myth of the Air Gap: Machine Learning for Reliable Intrusion Detection in SCADA Systems". *EAI Endorsed Transactions on Security and Safety*.2019

[15] ALJARAH, Ibrahim, ALA'M, Al-Zoubi, FARIS, Hossam, *et al.* "Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm".*Cognitive Computation*, 2018, vol. 10, no 3, p. 478-495.

[16] SALEH, Ahmed I., TALAAT, Fatma M., et LABIB, Labib M. "A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers". *Artificial Intelligence Review*, 2019, vol. 51, no 3, p. 403-443

[17] HALIMAA, Anish et SUNDARAKANTHAM, K. "Machine learning based intrusion detection system". *In: 2019 3rd International conference on trends in electronics and informatics (ICOEI)*. IEEE, 2019. p. 916-920.

[18] GHOSH, Partha et MITRA, Rajarshee. "Proposed GA-BFSS and logistic regression based intrusion detection system". *In: Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*. IEEE, 2015. p. 1-6.

[19] HUSSAIN, Jamal et LALMUANAWMA, Samuel. "Feature analysis, evaluation and comparisons of classification algorithms based on noisy intrusion dataset". *Procedia Computer Science*, 2016, vol. 92, p. 188-198.

[20] ABDULLAH, Manal, ALSHANNAQ, A., BALAMASH, A., et al. "Enhanced intrusion detection system using feature selection method and ensemble learning algorithms". *International Journal of Computer Science and Information Security (IJCSIS)*, 2018, vol. 16, no 2, p. 48-55.

[21] Hall, M.A. "Correlation-Based Feature Selection for Machine Learning"; *The University of Waikato: Hamilton, New Zealand*, 1999

[22] KARIMI, Zahra, KASHANI, Mohammad Mansour Riahi, et HAROUNABADI, Ali. "Feature ranking in intrusion detection dataset using combination of filtering methods". *International Journal of Computer Applications*, 2013, vol. 78, no 4.

[23] AL JANABI, Kadhim BS et KADHIM, Rusul. Data reduction techniques: "a comparative study for attribute selection methods". *International Journal of Advanced Computer Science and Technology*, 2018, vol. 8, no 1, p. 1-13

[24] KARABOGA, Dervis. " An idea based on honey bee swarm for numerical optimization". *Technical report-tr06, Erciyes university, engineering faculty, computer engineering department*, 2005.

[25] SHUNMUGAPRIYA, P. et KANMANI, S. "A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid)". *Swarm and Evolutionary Computation*, 2017, vol. 36, p. 27-36.

[26] WANG, Jun, LI, Taihang, et REN, Rongrong. "A real time IDSs based on artificial bee colony-support vector machine algorithm". *In: Third International Workshop on Advanced Computational Intelligence*. IEEE, 2010. p. 91-96.

[27] MAHMOD, Mahmod S., ALNAISH, Zakaria A. Hamed, et AL-HADI, Ismail Ahmed A. "Hybrid intrusion detection system using artificial bee colony algorithm and multi-layer perceptron". *International Journal of Computer Science and Information Security*, 2015, vol. 13, no 2, p. 1.

[28] CHOUDHARY, Roshani et SHUKLA, Sanyam. "A clustering based ensemble of weighted kernelized extreme learning machine for class imbalance learning". *Expert Systems with Applications*, 2021, vol. 164, p. 114041

[29] ALOTAIBI, Bandar et ELLEITHY, Khaled. "A majority voting technique for wireless intrusion detection systems". *In: 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE, 2016. p. 1-6

[30] ROY, Jean-Francis. " Apprentissage automatique avec garanties de généralisation à l'aide de méthodes d'ensemble maximisant le désaccord". 2018

[31] VINUTHA, H. P. et POORNIMA, B. "An ensemble classifier approach on different feature selection methods for intrusion detection". *In: Information systems design and intelligent applications. Springer, Singapore*, 2018. p. 442-451.

[32] https://www.unb.ca/cic/datasets/nsl.html

[33] https://www.cs.waikato.ac.nz/ml/weka/