Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 418 – 433

doi:10.3906/elk-1702-279

Modified stacking ensemble approach to detect network intrusion

Necati DEMİR*, Gökhan DALKILIÇ
Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, Tınaztepe, İzmir, Turkey

Received: 20.02.2017 • **Accepted/Published Online:** 15.11.2017 • **Final Version:** 26.01.2018

Abstract: Detecting intrusions in a network traffic has remained an issue for researchers over the years. Advances in the area of machine learning provide opportunities to researchers to detect network intrusion without using a signature database. We studied and analyzed the performance of a stacking technique, which is an ensemble method that is used to combine different classification models to create a better classifier, on the KDD'99 dataset. In this study, the stacking method is improved by modifying the model generation and selection techniques and by using different classifications algorithms as a combiner method. Model generation is performed using subsets of the dataset with randomly selected features and not all of these models are used as input for the combiner. Various metrics are used in model selection and only selected models are used as input for the combiner method. In our experiments, the stacking technique provided higher accuracy results all the time compared to pure machine learning techniques. The second important result in our experiments was obtaining the highest detection rate for user-to-root attacks compared to other studies.

Key words: Classification, ensemble, machine learning, stacking

1. Introduction

Arthur Samuel described machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed" in 1959 [1]. This field typically deals with data that have already been collected to teach a computer to act on a task.

Ensemble learning is a machine learning approach where more than one learner (classification models or regression models) is used to solve the same classification or regression problem. In contrast to the conventional machine learning approaches that try to construct a model from training data, ensemble methods construct a set of models (learners) and combine them. Ensemble methods have been a major research area since the 1990s after two pioneering studies. Hansen et al. [2] found that if the combination of a set of classifiers is used for prediction, they often provide more accurate results compared to a single classifier's results. In a different study [3], the author proved that weak learners could be boosted to strong learners.

There are three common and widely known types of ensemble techniques: bootstrap aggregating, boosting, and stacking. Bootstrap aggregating, known as bagging, trains each model by drawing random subsets of the training set. The random forest algorithm uses bagging and combines random decision trees. Boosting incrementally builds an ensemble model by training each new model using the misclassified training instances that previous models misclassified. As an example for boosting, the adaboost algorithm uses a boosting technique. Stacking, also known as stacked generalization, is a method where an algorithm is used to combine the

*Correspondence: ndemir@demir.web.tr

outputs of other models' predictions. Stacking is the generalization of other ensemble methods [4] and that is the main reason it was chosen in this study.

Network intrusion detection is defined as the process of detecting attacks and misuse of computer networks [5]. This study uses modified stacking, which is an ensemble approach, to build a network intrusion detection system by using the KDD'99 dataset. Although it has been 16 years since the release of the KDD'99 dataset, it is still in use as the primary source for network intrusion detection studies. For example, [6–10] are some of the papers that were released in 2014 and 2015 in which the KDD dataset was used.

1.1. Motivation

Our motivation for this study is driven by the following:

- Although there are various studies applying ensemble machine learning techniques, to our best knowledge, there is no study that compares linear regression, decision tree, and naïve Bayes when they are used for a combiner method.
- To our best knowledge, there is no study applying stacking ensemble on KDD'99 dataset, which uses various selection methods to be given as input to the combiner algorithm.

1.2. Threat model & detectors

Our threat model assumes that there is a monitoring system that collects information on the packet level. It also assumes that the attacker can do four types of attack:

- The attacker is able to gain a user right on the target host by exploiting various vulnerabilities of the applications running on the target host,
- The attacker already has a user account in the target host and can gain root access by exploiting various vulnerabilities of the applications running on the target host,
- The attacker is able to launch a denial of service (DoS) attack by exploiting the vulnerabilities of the applications running on the target host,
- The attacker probes the target host with various techniques to gain information.

This study is supposed to develop a detector that can be used to detect the attacks explained above.

2. Related works

The conventional way of detecting intrusion is to use signatures, but that can identify only previously seen attacks. To overcome this problem, in the past decades different studies have been published that used machine learning techniques to classify network traffic and detect intrusions. In this section, we investigate the studies that used ensemble machine learning techniques.

Gyanchandani et al. [11] used the C4.5 algorithm in stacking, boosting, and bagging ensemble techniques. Syarif et al. [12] used four different classification algorithms – naïve Bayes, J48 (decision tree), JRip (rule induction), and iBK (instance-based learning with parameter k) – as base classifiers for 3 types of ensemble techniques: bagging, boosting, and stacking. However, these two studies did not provide results for each label in the KDD'99 dataset.

Toosi et al. [13] constructed a custom ensemble method that uses a genetic algorithm and fuzzy algorithm and produces promising results. Although they outperformed most of the previous studies, their user-to-root (U2R) accuracy rate was one of the lowest. Hu et al. [14] used a modified adaboost algorithm that is an ensemble algorithm and used detection rate as the evaluation metric, but they did not provide the accuracy of each labeled attack. Patel et al. [15] used both boosting and bagging methods with a decision tree and support vector machines, but only provided the overall accuracy ratio. Zebarjad et al. [16] used a custom ensemble approach where the dataset is divided into 10 subsets and trained the dataset with three different algorithms, but used only 10 features and did not take the other features into account.

In a study from 2014, Shrivas et al. [17] built an ensemble method by using artificial neural networks and a Bayesian network as base classification algorithms and used the gain ratio for feature selection. In another recent study from 2015, Gaikwad et al. [18] used a bagging ensemble method with a partial decision tree-based classifier and used a genetic algorithm for feature selection. The common disadvantage of these two studies is not considering all the features.

As we see in the previous studies, most of them did not provide accuracy for each label, which makes comparison hard. The KDD'99 dataset contains more DoS attacks than U2R attacks; for example, if you label all DoS and U2R attacks as DoS, you will have more than 90% accuracy. Therefore, the accuracy of each labeled attack, such as U2R, should be given in the results. One of our goals in this study is to provide these results.

In this study, we modified the stacking technique. First, we generated 100 models by using random feature selection and used three different metrics to select the best of them. Then we used three combination methods to reach the best results. The details of these steps are explained in Section 3.

3. Stacking

Stacking is a type of ensemble method. Ensemble learning is a machine learning approach where more than one learner (classification models or regression models) is used to solve the same problem. Ensemble methods try to construct a set of models and combine them, which is different from ordinary machine learning approaches that try to construct a model from training data [19].

Before proceeding, we will describe some key words and how they are used in this study. A classification algorithm or base classification algorithm is an ordinary algorithm used in machine learning for the classification task such as logistic regression, naïve Bayes, etc. The model is the output when the classification algorithm is trained with the training data. This model, after training, is used for prediction.

In the training phase, generally the base classification algorithm(s) and training data are accepted as inputs. Model generation is used to train the algorithm with data and generate models. If the stacking implementation accepts only one algorithm, usually the "model generation" phase generates n models by using the algorithm with randomly drawn subdatasets. If the implementation accepts multiple algorithms, usually the training set is trained with each algorithm. When the models are generated, these models generate the predicted labels. This is the first layer of the two-layered training phase. The predicted labels of each model are given as input to the second layer. In the second layer, a classification algorithm (also called the combiner method) is used to generate a final model while the original labels are still used for labeling the new training data. Figure 1 shows the training phase of the stacking approach.

The prediction phase of the stacking also contains two layers. The first layer uses the input data and previously generated models and makes a prediction, and the second layer uses the model previously generated in the second layer of the training phase. Figure 2 shows the prediction phase of stacking approach.

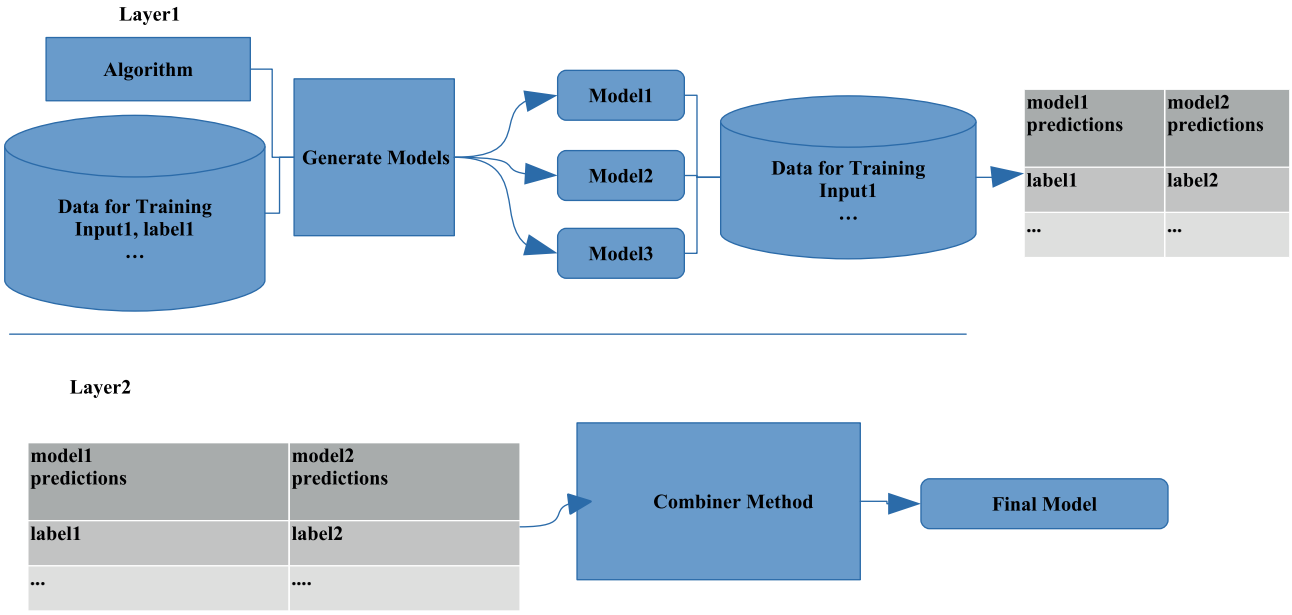

Figure 1. Training phase of stacking approach.

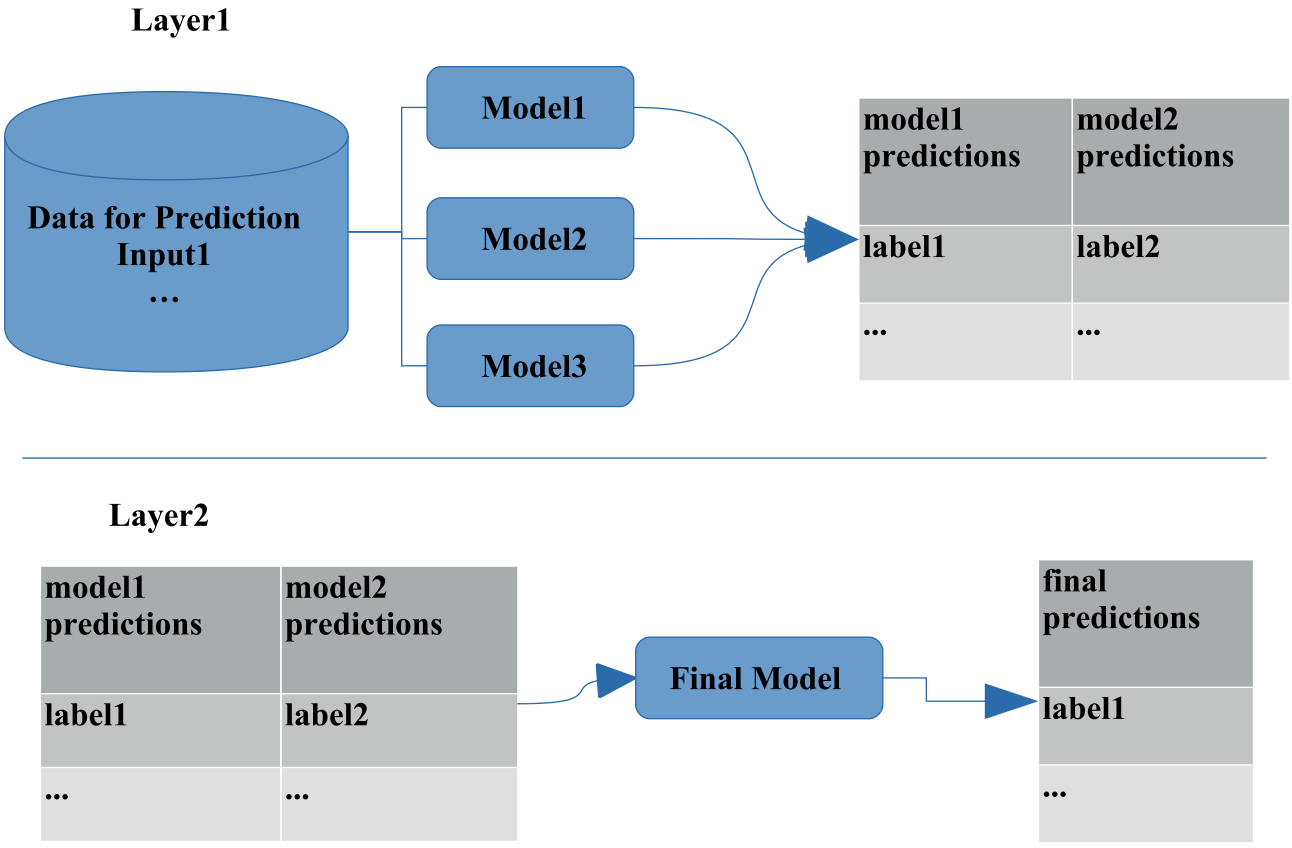

Figure 2. Prediction phase of stacking approach.

3.1. Our approach

In this study, we slightly modified the first layer of training with the stacking approach and used different combiner methods in the second layer. One of our contributions to the first layer is to generate models by using all training data with random features. The second one is selecting the best models according to the three metrics of accuracy, information gain, and recall. The details of our contribution are explained in Section 3.2. Our contribution to the second layer is using different algorithms as combiner methods, which is explained in Section 3.3.

3.2. Model generation

In our approach, we used logistic regression and the training dataset as inputs to the first layer. The modified "model generation" consists of two steps:

- Random feature selection and training,
- Selecting the best 10 models.

In the first step, random subdatasets were not drawn by rows but by features. In the original paper on random forests [20], which is an ensemble algorithm, the author emphasized that a random forest creates the best results when it is used with random feature selection. There are also various papers that investigated the power of random feature selection in ensemble methods. Bryll et al. [21] showed that using random feature selection for ensemble classifiers overperforms and mentioned that using random feature selection helps to create noncorrelated models, and those noncorrelated models help to improve the performance of the ensemble method. Skurichina et al. [22] mentioned that while combining multiple feature subsets, random feature selection may be preferred as it may provide more successful results in obtaining independent feature subsets. In our study, we selected n/2 features, where n equals the number of features, in each draw and generated 100 subdatasets, which led to 100 models.

The second step reduces 100 models to 10 models. The best 10 models are selected according to the three evaluation methods of accuracy, information gain, and recall; the higher the value, the better the model is. The formula of each metric is given in this section and the details of the formulas are explained by using a confusion matrix template.

Using 100 and 10 as the parameters is a common practice in ensemble implementations. For example, the random forest's default 'number of estimators' parameter is 10 and xgboost's default 'number of estimators' is 100 in a widely used Python library called scikit-learn [23].

A confusion matrix is used to show the performance of the model by providing how it performs per label. Table 1 shows the structure of a confusion matrix, which is used to show the performance of our approach. The sum of each row contains the actual corresponding labeled data. As an example, the sum of X_{00}, X_{01}, X_{02}, X_{03}, and X_{04} equals the total number of actual *normal* labeled data, and X_{00} shows the number of data labeled correctly by the model.

Evaluation metrics to select models are as follows:

Accuracy: While using accuracy as an evaluation method for selecting the 10 best models, each model's accuracy is calculated and then the 10 highest scores are selected for the next step. The accuracy is calculated using Eq. (1) [4] by dividing the total number of correctly classified samples by the total number of samples.

Although accuracy is a widely used metric, it can misguide the researcher when the data are unbalanced. As an example, let us assume there are two classes (class a and class b) and 90 of them belong to class a and 10

Table 1. An example of a confusion matrix.

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	X_{00}	X_{01}	X_{02}	X_{03}	X_{04}
	Probe	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}
	DoS	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}
	U2R	X_{30}	X_{31}	X_{32}	X_{33}	X_{34}
	R2L	X_{40}	X_{41}	X_{42}	X_{43}	X_{44}

of them belong to class b. If the model predicts all of them as class a, the accuracy score will be 90% although it detects none of the data labeled as class b.

$$accuracy = \frac{x_{00} + x_{11} + x_{22} + x_{33} + x_{44}}{\sum (X)} \tag{1}$$

Information gain: As in accuracy calculation, information gain of each model is calculated and the 10 best scores are chosen. Information gain is calculated using Eq. (4) [4] by the difference between two entropies. In our case the first entropy is always 1, because 1 is the default value in the initial state. Eq. (2) shows how entropy is calculated for a binary class, and in a binary class problem a class has value 0 or 1; as an example, in this formula, $p(y = 0)$ denotes the probability of the classes where the value is equal to 0. Eq. (2) can be generalized as seen in Eq. (3), and $p(y_i)$ denotes the probability of each class. Eq. (4) shows the final calculation of information gain for our study and the main idea of this formula is to find the information gain after predictions. In this formula $y_{predictions}$ denotes all unique labels in the predicted data, y_p denotes each unique value in $y_{predictions}$, y denotes all original labels, and $y \vee y_p$ denotes the value of y where the predicted label is equal to y_p. Information gain is a well-known technique used in algorithms such as decision trees and the goal of using information gain in this algorithm is to find out which feature contains more information about the result. In our study, we use information gain to select the best models, which affects the results.

$$Entropy = -p(c = 0) * log_2(p(c = 0)) - p(c = 1) * log_2(p(c = 1)) \tag{2}$$

$$E(Y) = -\sum_{i=1}^{n} p(y_i) \, log_2 p(y_i) \tag{3}$$

$$InformationGain = 1 - \sum_{y_p \in vals(y_{predictions})} p(y_p) \, Entropy(y \vee y_p) \tag{4}$$

Recall (true positive rate) mean: Recall mean, also known as true positive rate mean, is calculated as the mean of the recall value for each label (Probe, DoS, U2R, R2L). In the second step, as in the previous evaluation methods, the 10 best scores are chosen. Recall mean formula is calculated using Eq. (5) and the formula for each label is calculated using Eqs. (6)–(10) [4]. Recall is calculated by dividing the number of correctly classified samples by the number of true labeled samples in the original dataset. The main reason for using recall as a model selection criterion is that it allows us to understand how a model performs in terms of detecting a certain type of attack. We have five different labels (four types of attack traffic and one label as the normal traffic) and we use averaging to get an insight of how a model does well to detect these labels. Another

option could be precision, but recall helps us to understand if the model generalization is working or not, while precision allows us to understand if an attack is a real attack. For example, if a model detects only one attack and it is a real attack, the precision will be 100% but the generalization will be very poor.

$$recall_{mean} = \frac{(recall_{normal} + recall_{probe} + recall_{dos} + recall_{u2r} + recall_{r2l})}{5} \tag{5}$$

$$recall_{normal} = \frac{X_{00}}{\sum x_{0i}} \tag{6}$$

$$recall_{probe} = \frac{X_{11}}{\sum x_{1i}} \tag{7}$$

$$recall_{dos} = \frac{X_{22}}{\sum x_{2i}} \tag{8}$$

$$recall_{u2r} = \frac{X_{33}}{\sum x_{3i}} \tag{9}$$

$$recall_{r2l} = \frac{X_{44}}{\sum x_{4i}} \tag{10}$$

3.3. Combiner method

In our study, we used three different classification algorithms as combiner methods. These are logistic regression, decision tree, and naïve Bayes algorithms.

Logistic regression: Logistic regression (also known as logit regression) is a regression model where the outcome is a categorical variable. Logistic regression is used to estimate the probability of the outcome based on the features given. Although there is a history of logistic regression with one feature, David Cox, with his studies [24], is known as the founder of logistic regression.

Logistic regression measures the relationship between the input variables and the outcome by estimating probabilities. We can define logistic regression as shown in Eq. (11), which finds the probability of Y by given X.

$$P\left(Y = 1 \,|\, X = x\right) \tag{11}$$

The main idea behind logistic regression is to use the linear regression, where the equation P of linear regression is shown in Eq. (12).

$$w_1 x_1 + w_2 x_2 + ... = \sum w_i x_i \tag{12}$$

Logit transformation is used to convert linear regression to logistic regression using Eq. (13) and solving for P gives Eq. (14) [4].

$$log \frac{P\left(x\right)}{1 - P\left(x\right)} \tag{13}$$

$$P = \frac{e^{\sum w_i x_i}}{1 + e^{\sum w_i x_i}} = \frac{1}{1 + e^{-\sum w_i x_i}} \tag{14}$$

Decision tree: Decision tree learning is a method where a learned tree can be represented as if-then rules, which allows it to be interpreted easily, and that is the main reason for decision trees to be used widely. A decision tree is composed of a set of nodes, where each node contains a rule for an attribute. To classify an instance, starting from the root node, the instance is tested against the rule of that node by moving down the tree branch according to the rule. This is repeated for each subtree until the leaf nodes. Each leaf is labeled as the class that represents the best or it may contain the probability of the target class.

Decision trees have several advantages. Decision trees are nonparametric, which means that no assumption is needed for the distribution of the data. They can find nonlinear relations between features and labels (classes) and also work fine when there are missing values. Finally, they can handle both numeric and categorical inputs [25]. Decision trees are easily interpreted, but complexity plays an important role according to [26] for the accuracy of decision trees. Tree complexity is determined by the stopping criteria. The complexity of a tree is expressed in terms of total number of leaves, the total number of nodes, and tree depth.

There are various algorithms to create decision trees such as Iterative Dichotomiser 3 (ID3) [27], C4.5 [27], and classification and regression trees (CART) [26]. ID3 is a simple algorithm and the splitting criterion for ID3 is information gain. The growth of the tree continues while information gain is greater than zero or when all of the instances are in the same class. ID3 cannot handle numeric values or missing values. C4.5 is the successor of ID3 from the same creator of ID3. The growth of the tree stops when the number of instances to be split is below a predefined threshold. Besides handling categorical values, C4.5 can handle numeric values. CART is similar to C4.5, but also accepts numerical values as a target, which means that it can also produce regression trees in addition to classification trees. In this study, we used the CART algorithm for building decision trees.

Naïve Bayes: Bayes' theorem is used to find the probability of an event where conditions related to that event are used as the input. Naïve Bayes is a classification algorithm that is based on Bayes' theorem with independence assumptions between features of the instances. It may seem that assuming independence between features is a bad idea, but naïve Bayes often performs well.

Given a class variable y and an instance X, where $X = x_1 x_2$ and x_i is a feature of the instance, Bayes' theorem states the relationship shown in Eq. (15).

$$P(y \mid X) = \frac{P(X \mid y)\, P(y)}{P(X)} \tag{15}$$

The naïve Bayes classifier assumes that features are independent of each other, as shown in Eq. (16).

$$P(X \mid y) = \prod P(X \mid y, x_1, ..., x_{i-1}, x_{i+1}, ...) \tag{16}$$

We can rewrite the Bayes' theorem with independence assumption as the equation shown in Eq. (17).

$$P(y \mid X) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(X)} \tag{17}$$

Since P (X) is constant, we can simplify the equation as shown in Eq. (18) [4].

$$P(y \mid X) \approx P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{18}$$

By using Eq. (18), we calculate the probability of each y class and we can say that the class of the instance is the class of y that has the maximum value.

4. Experiments

In this study, we used a stacking ensemble approach by using different base classification algorithms as stacking algorithm. In this section, we will describe the dataset, which is known as the KDD'99 dataset. After describing the dataset, we will describe the steps of the experiments. Finally, we will describe evaluation metrics.

4.1. The dataset

We used the KDD'99 dataset, which is widely used in network intrusion detection studies. The KDD'99 dataset was used for the 3rd International Knowledge Discovery and Data Mining Tools Competition. The competitors were given a task and the task was to build a network intrusion detection system. This dataset was acquired from the 1998 Defense Advanced Research Projects Agency (DARPA) intrusion detection evaluation program. DARPA set up an environment to collect raw TCP/IP data for a local area network (LAN) that simulates a typical US Air Force LAN, and that LAN was operated as if it was a true environment and was blasted with multiple attacks.

The attacks in this dataset are divided into 4 different categories:

- *Probe:* The goal of the attacker is to get information about the target host.
- *Denial of Service (DoS):* The goal of the attacker is to prevent normal users from using a service. This can be accomplished by two methods:
  - Keeping the resources of the target host busy by launching excessive numbers of normal communication transactions, which is a widely used method;
  - Exploiting the services.
- *User to Root (U2R):* Attackers can access the target machine with the goal of gaining root privileges.
- *Remote to Local (R2L):* Attackers cannot access the target host and their goal is to access it.

In 1999, the DARPA dataset was revised to create the KDD'99 dataset that contains 265 units of summarized information of TCP connections, not TCP dumps. Each connection is summarized with 41 features, which can be grouped in 4 categories:

- *Basic features:* Basic features are extracted from headers and it is not necessary to investigate the payload.
- *Content features:* Content features are extracted from payload and domain knowledge is needed to extract features.
- *Time-based traffic features:* These features are computed features over a 2-s temporal window. For example, error rate is a time-based traffic feature, which describes the percentage of connections having SYN errors.
- *Host-based traffic features:* These features are computed features over a 100-connections temporal window. Some probing attacks scan the hosts where computing these features gives insight.

The KDD'99 dataset consists of three components as seen in Table 2 and all these components are extracted from the same source. In the International Knowledge Discovery and Data Mining Tools Competition '99, only 10% of the KDD, which contains 22 attack types, was used for training, as in our study. The corrected KDD dataset contains different statistical distributions for attacks and also contains 14 additional attacks. We used the corrected KDD dataset for testing purposes.

Table 2. Components of KDD'99.

Dataset	Probe	DoS	U2R	R2L	Normal
10% of KDD	4107	391,458	52	1126	97,277
Corrected KDD	4166	229,853	70	16,347	60,593
Whole KDD	41,102	3,883,370	52	1126	972,780

4.2. The steps of the experiment

In this study, we conducted 13 different experiments by using 4 main methods. In the first method, we did not use any stacking and used logistic regression for classification of data, which is compared against other methods. The other three methods share three common steps, but in the last step, each method uses a different algorithm as a combiner method. The following steps are the shared common steps among the three methods:

- using logistic regression to generate models,
- selecting models by using 4 different methods,
- using a combiner method.

The model selection methods used in the second step and the algorithms used by each method have been discussed in Section 3.2.

In our study, we did not create a new set of models in each experiment and used all 100 models in all experiments. In other words, we created 100 models once and used them again and again in each experiment. These 100 models were reduced to 10 models in each experiment by using one selection method at a time. This gave us the chance to understand how the combiner and selection methods perform.

We conducted 13 experiments. In the first experiment, we used logistic regression without stacking. For the rest of the experiments, we used three different combination algorithms and four different model selection methods for each combination algorithm, which results in 12 experiments. The details of each experiment can be found in Table 3 where the results are reported.

We implemented this experiment with the Python programming language and used the scikit-learn machine learning library [23] on Ubuntu Linux OS. The scikit-machine learning library is a widely used library for academic and commercial purposes. It contains implementations for both supervised learning (classification and regression) and unsupervised learning (clustering) algorithms. It allows developers to use a common interface for all algorithms, which makes it easier to use and develop prototypes quickly.

4.3. Experimental results

We evaluated each experiment with six metrics, which led to 78 results. The evaluation metrics are listed below:

- tp_normal: True positive percentage of normal labeled data,
- tp_probe: True positive percentage of probe labeled data,
- tp_dos: True positive percentage of DoS labeled data,
- tp_u2r: True positive percentage of U2R labeled data,
- tp_r2l: True positive percentage of R2L labeled data,
- accuracy: Accuracy of the experiment.

The formula of each metric is the same as described in Section 3.2.

Table 3. Results of the experiments.

Combiner/stacking algorithm	Model selection method	Exp. #	tp_normal %	tp_probe %	tp_dos %	tp_u2r %	tp_r2l %	Accuracy %
Logistic regression (no stacking)	-	1	98.1	68.7	82.8	8.6	5.6	81.53
Logistic regression	all	2	98.3	83.5	97.3	12.9	5.2	92.43
	accuracy_score	3	98.4	72.9	97.3	8.6	5.6	92.33
	information_gain	4	98.4	73.7	97.3	7.1	5.6	92.36
	recall_mean	5	98.2	72.3	97.3	8.6	5.6	92.31
Decision tree	all	6	99.4	77.0	97.3	14.3	2.5	92.47
	accuracy_score	7	99.0	73.7	97.3	7.1	5.6	92.46
	information_gain	8	99.0	78.3	97.3	8.6	5.7	92.55
	recall_mean	9	99.0	75.2	97.3	18.6	5.5	92.46
Naïve Bayes	all	10	97.3	78.9	93.2	47.1	6.0	89.19
	accuracy_score	11	97.9	80.2	97.2	30.0	5.7	92.29
	information_gain	12	97.9	79.9	97.2	30.0	7.6	92.39
	recall_mean	13	98.0	79.0	96.7	35.7	5.6	91.90

We also used McNemar's test to assess the performance of experiments. McNemar's test [28] is a statistical test used to compare the two classifiers, where $\mathbf{f_1}$ denotes the first classifier and $\mathbf{f_2}$ denotes the second classifier. McNemar's test is based on the chi-square (χ^2) test. Eq. (19) is used to calculate McNemar's test, where $\mathbf{n_{01}}$ denotes the number of examples misclassified by $\mathbf{f_1}$ and not by $\mathbf{f_2}$, and $\mathbf{n_{10}}$ denotes number of examples misclassified by $\mathbf{f_2}$ and not by $\mathbf{f_1}$. If χ^2 is more than 3.841459, we can see that the P-value is less than 0.05 in the chi-square distribution table and conclude that these two classifiers have different performances.

$$\chi^2 = \frac{(n_{01} - n_{10} \vee -1)^2}{n_{01} + n_{10}} \tag{19}$$

We provide the results in Table 3 and compare the results with other studies in Table 4. We also provide four bar plots (Figures 3–6) to show the results of true positive rates of attacks (probe, dos, U2R, and R2L) in experiments.

Table 4. Comparison of the results.

Study	Normal (%)	Probe (%)	DoS (%)	U2R (%)	R2L (%)	Accuracy (%)	Training data	Testing data
[13]	98.2	84.1	99.5	14.1	31.5	N/A	10% of KDD	Corrected KDD
[28]	99.5	73.2	96.9	6.6	10.7	92.59	Whole KDD	Corrected KDD
[29]	92.4	72.8	96.5	22.9	11.3	N/A	10% of KDD	Corrected KDD
[30]	99.5	83.3	97.1	13.2	8.4	93.3	Whole KDD	Corrected KDD
[31]	98.55	80.9	99.6	16.2	31.6	95.71	10% of KDD	Corrected KDD
Our approach	99.4	83.5	97.3	47.1	7.6	92.55	10% of KDD	Corrected KDD

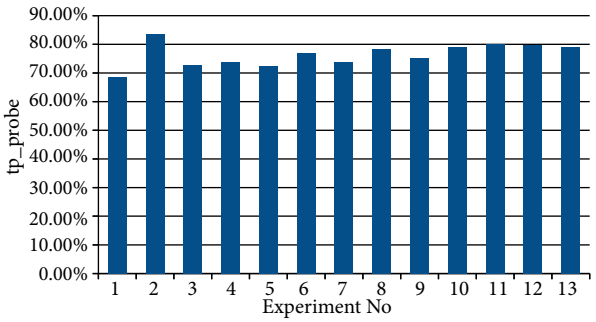

Figure 3. Bar plot of tp_probe values from experimental results.

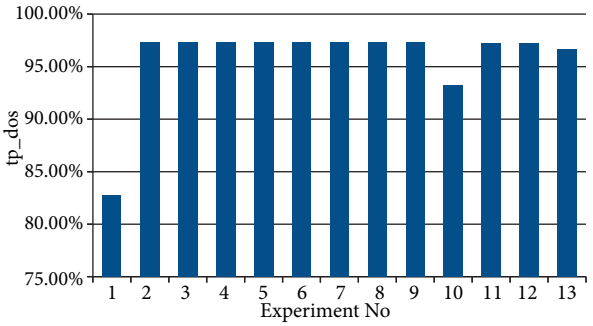

Figure 4. Bar plot of tp_dos values from experimental results.

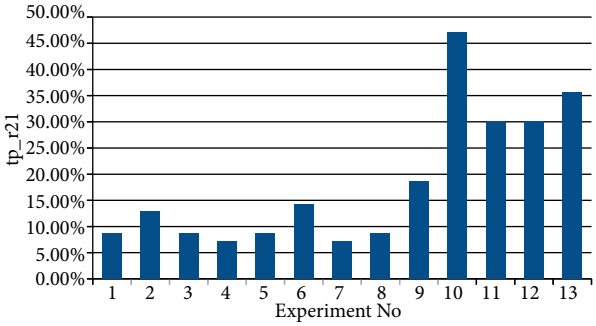

Figure 5. Bar plot of tp_u2r values from experimental results.

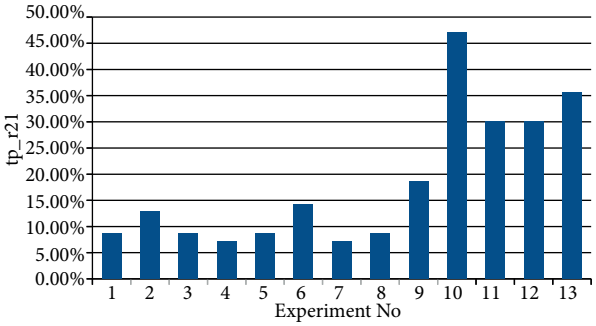

Figure 6. Bar plot of tp_r2l values from experimental results.

5. Discussion

In Table 3, the results of all experiments are given. The table contains the results of 13 experiments and each experiment is evaluated by six different metrics. As seen in Table 3, the tp_normal metric provides the highest value when the decision tree is used as the combiner algorithm with all models (experiment #6), and the tp_probe metric provides the highest value when logistic regression is used as a combiner algorithm with all models (experiment #2). Tables 5 and 6 show the confusion matrix of experiment #6 and experiment #2, respectively. The highest value of tp_dos is 97.3% and this is achieved in more than one experiment; for example, using logistic regression and the decision tree as the combiner algorithm, regardless of the model selection method, provides the value of 97.3%. The tp_u2r metric provides the highest value of 47.1% when naïve Bayes is used as the combiner algorithm with all models (experiment #10). The confusion matrix of experiment #10 is shown in Table 7. One conclusion to mention for tp_u2r is the high difference between the first and the second best values compared to other metrics. The tp_r2l metric provides the highest value when naïve Bayes is used as the combiner algorithm with the information gain-based model selection method (experiment #12). The confusion matrix of experiment #12 is shown in Table 8.

We see the best results of tp_r2l and tp_u2r when naïve Bayes is used as the combiner method. Naïve Bayes assumes that each feature in a dataset is independent. When naïve Bayes is used as the combiner method, the output of each model in the first layer is used and we know that each model is built independently. As seen in Table 2, the number of R2L and U2R attacks in the dataset is the lowest and an algorithm that less tends to overfit may provide better results for an unbalanced dataset. Naive Bayes tends to overfit less than other classification algorithms because it does not converge at all; rather, it learns its parameters by calculating.

Table 5. Confusion matrix of experiment #6.

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	60,255	244	97	1	16
	Probe	779	3208	167	0	12
	DoS	5783	227	223,721	0	122
	U2R	48	1	2	10	9
	R2L	15,909	17	11	8	402

Table 6. Confusion matrix of experiment #2.

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	59,572	233	778	4	6
	Probe	431	3477	257	0	1
	DoS	6182	83	223,584	0	4
	U2R	48	1	5	9	7
	R2L	15,487	6	3	3	848

Table 7. Confusion matrix of experiment #10.

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	58,940	201	692	719	41
	Probe	188	3287	107	584	0
	DoS	6925	41	214,157	8730	0
	U2R	29	0	0	33	8
	R2L	14,920	1	1	443	982

Table 8. Confusion matrix of experiment #12.

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	59,305	208	59	973	48
	Probe	449	3330	155	232	0
	DoS	6099	25	223,474	255	0
	U2R	36	0	4	21	9
	R2L	14,840	0	1	265	1241

Experiment #6 provides an outlier for tp_r2l, which is 2.5%. The decision tree algorithm tends to overfit data when the depth of the tree increases. We are using all 100 models for experiment #6 and that is causing the tree to create more subtrees and increase the depth compared to experiments #7–9. Since the number of tp_r2l is smaller than other labeled data and the decision tree tends to overfit, tp_r2l is disadvantaged in this situation.

In Table 4, the results of our approach are compared to other studies and this table also shows which study uses which component of the KDD dataset for training and testing purposes. In this table, we used the studies that provided detection rates for each label for comparison. Toosi and Kahani [13] constructed a custom ensemble that uses a genetic algorithm and fuzzy algorithm. Compared to Toosi and Kahani's study, our ensemble technique that combines all generated models with naïve Bayes provides a better detection rate for U2R, but not for others. Agarwal and Joshi [29] introduced a rule-based framework, Kayacik et al. [30] introduced a self-organizing map (SOM)-based technique, and Pfahringer [31] explained the technique, which was the winner of KDD'99, that used a bagged boosting ensemble technique. Our study provides better accuracy results for all types of attacks compared to [27–29]. Al-Yaseen et al. [32] used a modified k-means technique and our study provides better detection rates for U2R and probes compared to [32].

6. Conclusion

In this study, we provided the results for 12 different stacking implementations. Each stacking implementation provided a different strength in different attack types. In real-world applications, this study should be used as the second layer of a network intrusion detection system. The first layer detects the type of the attack and then triggers our best approach for the detected attack type. For example, if a network intrusion detection system detects that there is a probe attack, it can trigger our best approach to find which connections are the malicious traffic and then can block those connections. If the first layer of this approach misses the attack, the first layer should be investigated further. If the first layer of this approach misclassifies the attack, the first layer will use one of the best results mentioned in this paper, which are from experiments #2, #10, and #12; this will lead to the average success rates of 79.4%, 95.2%, 21.45%, and 5.6% for probe, DoS, U2R, and R2L attacks, respectively.

We also applied McNemar's test [28] to assess the performance of experiments compared to experiment #1. We assessed experiment #2 for tp_probe and tp_dos, experiment #10 for tp_u2r, and experiment #12 for tp_r2l. We found $P < 0.05$ for all these experiments, which indicates that those experiments yield statistically significant results when compared to experiment #1.

In this study, we modified the stacking technique to select the best models that were combined by using a combiner method. We used 3 different combiner methods and 13 different experiments. As seen in Table 3, using the stacking technique is always better than using pure logistic regression according to tp_probe, tp_dos, tp_u2r, tp_r2l, and accuracy metrics. Selecting only the best models instead of using all models as input for combiner methods mostly performs better when using the decision tree and naïve Bayes for the combiner method if we evaluate accuracy as the performance metric. Detection of U2R and R2L attacks is achieved well when using naïve Bayes as the combiner method. In future work, more techniques will be tested and the best models will be selected automatically. As another future work, this study can be improved by using statistical justification of selecting the number of models instead of fixing it as 10.

References

[1] Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. Contemp Phys 2015; 5: 172-185.

[2] Hansen LK, Salamon P. Neural network ensembles. IEEE T Pattern Anal 1990; 12: 993-1001.

[3] Schapire RE. The strength of weak learnability. Mach Learn 1990; 5: 197-227.

[4] Zhou ZH. Ensemble Methods: Foundations and Algorithms. Boca Raton, FL, USA; CRC Press, 2012.

[5] Retnaswamy B, Ponniah KK. A new ontology-based multi agent framework for intrusion detection. Int J Commun Syst 2016; 29: 2490-2502.

[6] Upadhyay S, Singh RR. Comparative analysis based classification of KDD'99 Intrusion Dataset. Int J Comp Sci Inf Sec 2015; 13: 14-20.

[7] Nalavade K, Meshram BB. Comprehensive study of KDD99 dataset and data mining tools for intrusion detection. i-Manager's Journal on Information Technology 2014; 3: 28-37.

[8] Saxena H, Richariya V. Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain. Int J Comput Appl 2014; 98: 6-19.

[9] Selvi R, Kumar SS, Suresh A. An intelligent intrusion detection system using average Manhattan distance-based decision tree. In: Suresh LP, Dash SS, Panigrahi BK, editors. Artificial Intelligence and Evolutionary Algorithms in Engineering Systems. New Delhi, India: Springer India, 2015. pp. 205-212.

[10] Harbola A, Harbola J, Vaisla KS. Improved intrusion detection in DDoS applying feature selection using rank & score of attributes in KDD-99 data set. In: International Conference on Computational Intelligence and Communication Networks; 14–16 November 2014; Bhopal, India. pp. 840-845.

[11] Gyanchandani M, Yadav RN, Rana JL. Intrusion detection using C4.5: performance enhancement by classifier combination. In: International Conference on Advances in Computer Science; 21–22 June 2010; Bangalore, India. pp. 130-133.

[12] Syarif I, Zaluska E, Prugel-Bennett A, Wills G. Application of bagging, boosting and stacking to intrusion detection. In: International Conference on Machine Learning and Data Mining in Pattern Recognition; 13–20 July 2012; Berlin, Germany. pp. 593-602.

[13] Toosi AN, Kahani M. A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. Comput Commun 2007; 10: 2201-2212.

[14] Hu W, Hu W. Network-based intrusion detection using Adaboost algorithm. In: 2005 IEEE/WIC/ACM International Conference on Web Intelligence; 19–22 September 2005; Compiegne, France. pp. 712-717.

[15] Patel A, Tiwari R. Bagging ensemble technique for intrusion detection system. International Journal for Technological Research in Engineering 2014; 4: 1350-1355.

[16] Zebarjad AR, Lotfinejad MM. Combination of three machine learning algorithms for intrusion detection systems in computer networks. Journal of Academic and Applied Studies 2012; 2: 9-18.

[17] Shrivas AK, Dewangan AK. An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set. Int J Comp App 2014; 15: 8-13.

[18] Gaikwad DP, Thool RC. Intrusion detection system using bagging with partial decision treebase classifier. Procedia Comp Sci 2015; 49: 92-98.

[19] Liu L, Zsu M. Encyclopedia of Database Systems. New York, NY, USA: Springer, 2009.

[20] Breiman L. Random forests. Mach Learn 2001; 45: 5-32.

[21] Bryll R, Gutierrez-Osuna R, Quek F. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recogn 2003; 36: 1291-1302.

[22] Skurichina M, Duin RP. Combining feature subsets in feature selection. In: 6th International Workshop, Multiple Classifier Systems; 13–15 June 2005; Seaside, CA, USA. pp. 165-175.

[23] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011; 12: 2825-2830.

[24] Cox DR. The regression analysis of binary sequences. J Roy Stat Soc B Met 1958; 20: 215-242.

[25] Fayyad UM, Irani KB. On the handling of continuous-valued attributes in decision tree generation. Mach Learn 1992; 8: 87-102.

[26] Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and Regression Trees. London; UK: Chapman and Hall/CRC; 1984.

[27] Quinlan JR. C4.5: Programs for Machine Learning. San Francisco, CA, USA; Morgan Kaufman Publishers, 1993.

[28] Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. Neur Comp 1998; 10: 1895-1923.

[29] Agarwal R, Joshi MV. PNrule: A new framework for learning classifier models. In: 2001 SIAM International Conference on Data Mining; 5–7 April 2001; Chicago, IL, USA. pp. 1-17.

[30] Kayacik HG, Zincir-Heywood AN, Heywood MI. A hierarchical SOM-based intrusion detection system. Eng Appl Artif Intel 2007; 20: 439-451.

[31] Pfahringer B. Winning the KDD99 classification cup: bagged boosting. SIGKDD Explor 2000; 2: 65-66.

[32] Al-Yaseen WL, Othman ZA, Nazri MZ. Intrusion detection system based on modified k-means and multi-level support vector machines. In: International Conference on Soft Computing in Data Science; 2–3 September 2015; Putrajaya, Malaysia. pp. 265-274.