Annals of Telecommunications
https://doi.org/10.1007/s12243-021-00836-0

Analysis of error-based machine learning algorithms in network anomaly detection and categorization

Samuel A. Ajila[1] · Chung-Horng Lung[1] · Anurag Das[1]

Received: 23 June 2020 / Accepted: 12 February 2021

Abstract
Intrusion and anomaly detection are particularly important to protect computer networks and communication vulnerability. This research aims to experimentally identify the best error-based machine learning algorithm for anomaly detection and anomaly attack categorization with the highest accuracy and fastest build time. A two-stage anomaly and categorization framework has been set up for experimental evaluation. The first stage identifies if a network flow is normal or anomalous and the second stage identifies type of attack if the first stage result is anomalous. The goal is to eventually use the best algorithm in an online stream model of network intrusion detection. To this end, five research propositions are defined, four sets of experiments are set up, and four research questions are asked. The UNSW-NB15 dataset for network anomaly is used for training and testing in the experiments. Machine learning algorithms are classified into four different learning approaches: information-based, similarity-based, probability-based, and error-based. Our focus in this paper is on the error-based learning models, specifically, the following algorithms: *Winnow*, *Logistic*, *Perceptron*, *Support Vector Machine (SVM)*, and *Deep Learning*. The results are also compared with the results of non-error-based machine learning algorithms. The results obtained show that, overall, the error-based machine learning algorithm, *Winnow*, is the best with 100% accuracy and time to build the model of 0.47 s for network anomaly detection. In terms of accuracy only, *SVM* comes top for network anomaly attack categorization but *Simple Logistic* is the best when accuracy and time to build are considered together.

Keywords Network intrusion detection · Machine learning · Error-based learning · Deep learning · Kappa statistics · UNSW-NB15

1 Introduction

Due to the massive growth of computer networks and their many applications, the number of network flows has increased tremendously. The considerable large number of traffic flows leads to a massive amount of data, which eventually leads to the vulnerability of the data and network as a whole. One of the many challenges of cybersecurity research is to identify intrusion/anomaly in the traffic flow. A network Intrusion Detection System (IDS) is one of the solutions to detect such attacks before they compromise the network. An IDS monitors the normal traffic flows and identifies its characteristics or patterns. If a new flow does not follow the same characteristics, it might be an anomaly. Hence, an IDS may help identify and even detect unknown attacks.

In the previous work [1], we have used three categories of machine learning techniques, namely information-, probability-, and partly similarity-based approaches [2] focusing essentially on the following algorithms: Random Forest, Random Tree, Bayes Network, Naïve Bayes, *k*-NN, C4.5, REPT, RIPPER, and PART. In the current work, we are exploring the accuracies of the fourth machine learning category, i.e., error-based machine learning algorithms [2], vis-à-vis network anomaly detection and categorization. The objective and focus of this paper are to extend our previous work by experimentally investigating error-based machine learning algorithms on the same dataset UNSW-NB15 [3]. (See Section 2 for a presentation on those four categories of machine learning techniques.) Specifically, we emphasize on the

✉ Samuel A. Ajila
ajila@sce.carleton.ca

Chung-Horng Lung
chung@sce.carleton.ca

Anurag Das
anuragdas@cmail.carleton.ca

[1] Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario K1S 5B6, Canada

Published online: 30 March 2021

following error-based algorithms and their variants: Winnow, Logistic Regression, Perceptron, Support Vector Machine, and Deep Learning. Similar to our previous work [1], we have set up four sets of experiments based on the following propositions:

Proposition 1 – Using error-based machine learning algorithms for network anomaly detection with training to testing ratio split of 66–34% is likely to generate a better accuracy compared to non-error-based learning approaches.
Proposition 2 – Given proposition 1, the accuracy of certain error-based machine learning algorithms for network anomaly detection can be improved by using 10-fold cross-validation.
Proposition 3 – Using error-based machine learning algorithms for network anomaly categorization detection with training to testing ratio split of 66–34% is likely to give a better accuracy compared to non-error-based learning approaches.
Proposition 4 – Given proposition 3, the accuracy of certain error-based machine learning algorithms for network anomaly categorization detection can be improved by using 10-fold cross-validation.
Proposition 5 –The accuracy of error-based deep machine learning approach is likely to surpass other error-based machine learning algorithms in both network anomaly detection and network attack categorization.

Our experimental framework is a two-staged model with the first stage detecting if the network flow is normal or anomalous. If normal, the network flow is allowed to continue. If not, the second stage analyzes the context of the network flow in order to categorize the anomaly and identify the network attack type.

In order to respond to the five propositions above, four sets of experiments are set up such that we can answer the following four research questions.

Question 1: Considering the accuracy, Kappa statistics, and time to build each learning model, what error-based machine learning algorithm performs the best for detecting network anomaly?
Question 2: Considering the accuracy, Kappa statistics, and time to build each learning model, what error-based machine learning algorithm performs the best for detecting network anomaly categorization?
Question 3: Considering our previous work [1] in terms of accuracy and time to build each learning model, and the error-based machine learning models, what algorithm performs the best for detecting network anomaly?
Question 4: Considering our previous work [1] in terms of accuracy only, and the error-based machine learning models, what algorithm performs the best for detecting network anomaly categorization?

The rest of this paper is structured as follows. Section 2 presents background information and literature review. Section 3 gives an overview of the dataset and the experiment environment. The experimental results are presented in Section 4. Section 5 presents the discussion of the results from the perspectives of the research propositions and questions, and a conclusion and future work is described in Section 6.

2 Background and previous research work

2.1 Machine learning

Machine learnin g is the study of algorithms that can learn complex relationships or patterns from empirical data and make accurate decisions [4]. Other than supervised learning, semi-supervised learning, and unsupervised learning; machine learning can also be divided into four categories based on different ground rules [2]:

- **Information-based machine learning** – this type of learning has its theoretical root in decision trees, entropy model, and information gain. A *decision tree* consists of a root node, intermediate nodes, and leaf nodes. The *entropy model* by Claude Shannon [5] defines a mathematical computational measure of the impurity of the elements in a set. This is based on transformation of probabilities of different outcomes via a random selection of element from a set to entropy values. *Information gain* is the relationship between a measure of heterogeneity of a set and predictive analytics. The foundational approach is ID3 algorithm and examples of information-based learning algorithms are J48, REPTree, Random Tree, Random Forest, HoeffdingTree, and Decision Stump. They also include some rule-based models such as Decision Table, and PART.
- **Similarity-based machine learning** – this type of learning has its theoretical bases from feature space and distance metrics. The fundamental approach is the Nearest Neighbor Algorithm. The basic idea in this type of learning is looking at *what has worked well in the past and make prediction for the future* [2]. Feature space describes the dimension of every descriptive feature in a dataset. The simplest way to measure the similarity between two instances in a dataset is to measure the distance between the instances in a feature space. Formally, a feature space distance metric must have the following properties: *Non-negativity*, *Identity*, *Symmetry*, and *Triangular Inequality*. Examples of feature space distances are Euclidean, Manhattan, Cosine similarity, and Mahalanobis.

Examples of algorithms are *k*-NN and *k*-means (e.g., simple *k*-means, Hierarchical, Canopy).

- **Probability-based machine learning** – this type of learning has its root in Bayes' Theorem. Bayes' Theorem can be defined as "the probability of an event occurring given the evidence, is equal to the probability of the evidence given the event, multiplied by the probability of the event." [2] Reasoning from event to evidence (forward reasoning) is less difficult than reasoning from evidence to event (backward reasoning). However, Bayes' Theorem allows for swapping between the two modes of reasoning. The standard approach in this type of learning is the Naïve Bayes Model.
- **Error-based machine learning** – this type of learning approach is based on Multivariable Linear Regression with Gradient Descent. The fundamentals are Simple Linear Regression, Error Measuring, and Error Surfaces. Simple Linear Regression is a relationship model that has its root in a simple geometry, that is, the equation of a line: $y = mx + b$ where m is the slope of the line, and b is known as the *y-intercept* of the line. This equation of a line predicts a *y value* for every *x value* given the slope and the *y*-intercept. Error measuring is needed to formally measure the fit of a linear regression model with a set of training data; thus, an *error function* is needed in the definition of a regression model. Example algorithms in error-based learning are Winnow, Logistic, Perceptron, Support Vector Machine (SVM), and Deep Learning.

2.2 Error-based machine learning

In error-based machine learning, searching is done for a set of parameters in a parameterized model that minimizes the error for the predictions made by the model for a set of training instances [2, 5]. From this, we can identify three important factors: the parameterized model, how to measure the error, and the error surface. The general approach in this type of learning is to build a predictive model that uses multivariable linear regression with Gradient Descent [5, 6]. Extension to this is Logistic Regression for nonlinear models, multinomial models, and SVMs [2, 6].

Linear regression is modeling the relationship between dependent variable and independent variables. The independent variables are referred to as explanatory variables [5]. When there is only one explanatory variable, it is called simple linear regression, or multiple linear regression if more than one independent variable. Multivariable linear regression is where multiple correlated dependent variables are predicted compared to a single variable.

A linear combination is generally expressed as: $y = w_0 + w_1\lambda_1 + w_2\lambda_2 + \ldots + w_k\lambda_k$ where y is the class, $\lambda_1, \lambda_2, \ldots, \lambda_k$ are the attribute values, and $w_0, w_1, w_2, \ldots, w_k$ are the weights. Generally, an extra attribute λ_0 is assumed and its value is always 1. Therefore, the predicted value for an instance class "*i*" is written as: $w_0\lambda_0^i + w_1\lambda_1^i + w_2\lambda_2^i + \ldots + w_k\lambda_k^i = \sum_{j=0}^{k} w_j\lambda_j^i$. Hence, for a simple linear regression, that is, $k = 1$, the predicted output $M(\lambda) = w_0 + w_1\lambda_1^i$. Using vector representation, we can redefine the predicted output as $M_w[\lambda] = w[0] + w\ [1] \times \lambda\ [1]$ where w is the vector and $w[0]$ and $w\ [1]$ are weights, and λ is an instance defined by a single descriptive feature $\lambda\ [1]$. We need an *error function* to measure the fit of a linear model. An error function captures the difference between the predicted value and the actual value. There are many different error functions, but the most commonly used is the *sum of squared errors*. In this case, what is calculated is to minimize the sum of the squares of the difference between the actual value and the predicted value, that is, $\sum_i^n (y^i - \sum_{j=0}^{k} w_j\lambda_j^i)^2$, where $\sum_{j=0}^{k} w_j\lambda_j^i$ is the predicted value and y^i is the actual. In the literature [2, 5, 6], the notation $\mathbf{L_2}$ is used to represent the error function. Using vector, $L_2(M_w, \lambda) = \frac{1}{2}\sum_{i=1}^{n} (y_i - (w[0] + w\ [1] \times \lambda_i\ [1])^2$.

Linear regression can be used easily for numeric attributes. It can also be used for any classification as long as regression is performed for each class by setting to "1" for those instances that belong to the class and "0" for those that do not. The problem with linear regression is that the membership values produced are not proper probabilities, because they can fall outside the range of 0 and 1. In addition, it assumes that the errors are statistically independent and that they are normally distributed with the same delta. These assumptions are violated when applied to classification problem because the observations only take the values 0 and 1. However, a related technique called *Logistic Regression* does not have these problems because it builds a linear model based on transformed target variable instead of approximating the values 0 and 1 directly [2, 5, 6].

Gradient Descent is an approach for learning weights based on the fact that the error surfaces that correspond to the high-dimensional weight spaces have the convex shape and that a single global minimum exists [5, 6]. This approach is based on a control search from a random starting point.

In *Logistic Regression*, an attempt is made to produce accurate probability estimates that lead to accurate classifications. However, this may not be necessary if the sole purpose is to predict class labels (i.e., 0 and 1). Another approach is to "learn" the *hyperplane* that separates the instances into two classes, that is, if the data can be separated perfectly into two groups using the hyperplane. In this case, it is said to be *linearly separable*. A simple algorithm for finding a separating hyperplane is called *perceptron learning rule*. Looking again at the equation of a hyperplane $w_0\ \lambda_0 + w_1\lambda_1 + w_2\lambda_2 + \ldots + w_k\lambda_k$, where the additional attribute λ_0 always has the value 1. In *perceptron learning rule*, this extension, i.e., λ_0, is called the *bias*. This means that we can predict the first class if the sum is greater than 0 and there is no need to include an additional constant in the sum. Therefore, a perceptron computes a

single output from multiple real valued inputs by forming a linear combination to its input weights and putting the output through a nonlinear activation function. A *Multilayer Perceptron (MLP)* is a network of simple neurons that are called perceptron. *Perceptron* is not the only algorithm guaranteed to find a separating hyperplane, an alternative known as *Winnow* can also be used [5, 6]. *Winnow* is "mistaken driven" because it only updates the weight vector when a misclassified instance is encountered [7].

SVM is a learning machine that implements **Structural Risk Minimization (SRM)** [4, 8]. There are two categories—*Support Vector Classification (SVC)* and *Support Vector Regression (SVR)*. Like linear regression, Support Vector finds a function that approximates the training points by minimizing the prediction error. If the training dataset is not linear, SVR uses a kernel function to map the data to a higher dimension feature space and then performs a linear regression in the higher dimensional feature space. The goal here is to find the optimal weights w and threshold b, i.e., bias. SVR attempts to minimize flatness of the weights (i.e., minimize $||w||$) and to minimize the error generated by the estimation process of the values at the same time [4].

There are different types of kernels. In general, any function of the form $\mathbf{F_k(x,y)}$ is a kernel function if written as $\mathbf{F_k(x,y) = \lambda(x).\lambda(y)}$, where $\boldsymbol{\lambda}$ is a function that maps an instance into a high-dimensional feature space [6, 7]. The kernel function represents a dot product in the feature space produced by $\boldsymbol{\lambda}$. The three most widely used kernels are Linear kernel, Polynomial kernel (PolyKernel), and Gaussian radial basis kernel (RBF). Simplifying the $\mathbf{F_k(x,y)}$ by dropping the function $\boldsymbol{\lambda}$, Linear kernel is defined as $\mathbf{F_k(x,y) = x.\ y + c}$, where c is an optional constant; Polynomial kernel is defined as $\mathbf{F_k(x,y) = (x.\ y + 1)^p}$, where p is the degree of a polynomial function; and RBF is defined as $\mathbf{F_k(x,y)} = \exp.(-\phi\ ||\mathbf{x - y}||^2)$, where ϕ is a manually chosen tuning parameter. In this paper, we have experimented with Polykernel, Normalized Polykernel, RBF, and PUK. PUK is Pearson Universal Kernel that can adapt its behavior to various kernel functions (from Linear to Gaussian).

Deep Learning techniques or simply *Deep Neural Nets* have been used effectively in image processing, speech recognition, natural language processing, etc. In contrast to a simple three-layer perceptron that learns a representation by adapting the hidden layer, "deep" multilayer perceptron adds further hidden layers that are subject of further transformations. Each hidden layer consists of further computational steps that are in themselves neural nets. Examples of *Deep Learning* approaches are *Feedforward Deep Multilayer Perceptrons*, *Deep Convolutional Neural Networks (CNN)*, and *Deep Recurrent Neural Networks (RNN)*. Each of these approaches does have variants, for example, Long Short-Term Memory (LSTM) and Seq2Seq in RNN [5, 7].

The deep learning library used in this research is a fully connected feedforward networks [7, 8] with 30 epochs, early stopping set at ten epochs, and the iteration sets at five epochs. The idea here is that if there is no improvement in error rate after five epochs, the training can stop; else, the next check is set at the second iteration which is ten epochs. The CustomNet zooModel is used with Softmax as the activation function, and LossMCXENT as the loss function [8]. In what follows, we will simply refer to this setting as "*deepLearning*."

2.3 Cross-validation

The hold-out method is to partition the dataset into training and testing sets [5–7]. The hold-out method is good enough if the dataset is very big and if the sample dataset is representative. The problem, however, is that it is difficult to tell if the sample is representative even though the dataset may be big. If, for some reason, the training sample part of the dataset has missing classes, then the training model produced is not a "good" representative of the dataset. A solution here is to apply stratification by using random sampling and this could provide a safeguard against uneven representation in both the training and testing sets. However, a more general approach to mitigate any bias is to repeat the whole sampling process several times and this is called cross-validation. In a 10-fold cross-validation, the data is divided randomly into 10 parts in which the class is represented approximately in the same proportion of the full dataset. Each part is held-out in turn and the model trained on the 9-tenths and the error rate is calculated on the hold-out set. Why 10-fold? Different datasets with different learning models have shown that 10-fold is about the right number that returns relatively good estimate of error [5, 7].

2.4 Evaluation

The evaluation criteria considered in this paper are precision, recall, and accuracy. The output from a model is probabilistic and a threshold can be set above which all the probabilistic values will result in classifying all the corresponding data into a particular class and vice versa [1]. Precision and recall are the measures that are directly computed from the confusion matrix (Table 1) formed as a result of classification [5]. Precision is calculated as (TP)/(TP + FP) and recall as (TP)/

Table 1 Confusion matrix for binary classification

		Actual class	
		1	**0**
Predicted class	**1**	True positive (TP)	False positive (FP)
	0	False negative (FN)	True negative (TN)

(TP + FN). The classification accuracy is then calculated as (TP + TN)/(TP + FP + TN + FN) expressed as a percentage. In reality, recall represents true positive rate (TPR).

In addition, we also used Kappa statistics in our evaluation. The Kappa statistics (also called Cohen's Kappa coefficient (k)) is used to measure inter- and intra-rater reliability of features. A Kappa of 1 shows perfect agreement, while a Kappa of 0 indicates agreement to chance [5].

2.5 Previous efforts and results on network anomaly detection

A number of research efforts [1, 3, 9–15] have been conducted for network anomaly or intrusion detection using the UNSW-NB15 dataset [3, 15]. These approaches have certain limitations. Some research papers considered one or two machine learning algorithms. For instance, only the research works in [11, 16] use more than one machine learning technique on the UNSW-NB15 dataset. Furthermore, research works [11–13, 16] do not identify the types of attacks. They only detect if a flow is normal or an anomaly. Research works in [3, 9, 10, 14, 17] do classify the attack types, but they only investigate a single machine learning technique. This paper does not present a detailed literature review of previous works on anomaly detection. However, a detailed presentation of an analysis of previous work review is given in our previous paper [1] which is a precursor for the current paper.

In this section, we focus on our previous work on which the current research work is extended. In the previous work [1], we investigated the detection of anomalies and the types of attacks, and made a comparison between the effectiveness of nine different machine learning techniques as well as the impact of different feature selection techniques on those machine learning algorithms. Furthermore, two sets of experiments were conducted in [1] on the UNSW-NB15 dataset. The result of the Experiments-set 1 is presented in Table 2. The methodology in this experiment is that all the attributes, i.e., features, are considered and all nine machine learning algorithms are used to build anomaly detection models. Those nine machine learning algorithms are Random Forest, Random Tree, Bayes Network, Naive Bayes, k-NN, C4.5, Reduced Error Pruning Tree, RIPPER, and PART.

As presented in Table 2, in terms of accuracy, Random Forest is the most accurate anomaly detection model with 97.91% closely followed by C4.5 (97.3194%), then PART (97.3109). Naïve Bayes is the least accurate considering the nine algorithms. Judging by the false positive rates and the precision (Table 2), there are little or no significant statistical differences between the following algorithms in terms of accuracy—Random Forest, Random Tree, C4.5, REPT, RIPPER, and PART. In terms of time to build, Naive Bayes is the fastest model with 0.69 s although the least accurate. Random Tree is equally fast with a build time of just 0.93 s and generates a high accuracy of 96.10%.

The second set of experiments in the previous work [1] was conducted to identify the types of anomaly. This experiment set has nine sub-experiments with different feature sets. The experiments are based on nine different feature selection methods [15, 18–20]. Please refer to our previous paper [1] for the details of these different feature selection methods.

In Table 3, C4.5 generates the best accuracy for *All Features* (87.66%) followed by Random Forest (87.08%) then comes PART (87.05%). The rest are REPT (86.62%), Random Tree (84.18%), k-NN (80.62%), Bayes Network (65.28%), and finally Naïve Bayes (46.16%). In general, C4.5 came out to be the top in terms of accuracy for anomaly categorization. This is closely followed by Random Forest. Note that our interest in Table 3 vis-à-vis the current research work is the row for "All Features." These results (i.e., "All Features" row) will be compared with the experimental results presented in this paper using the error-based machine learning algorithms.

3 Dataset and experiment setup

We have set up four additional sets of experiments based on the five propositions defined in Section 1. The

Table 2 Experiments-set 1: anomaly detection results for UNSW-NB15 [1]

Machine learning algorithms	Accuracy (%)	False positive rate (%)	Precision (%)	Recall (%)	Time to build (s)
Random Forest	97.9121	2.0	97.9	97.9	57.25
Random Tree	96.1036	4.0	96.1	96.1	0.93
Bayes Network	81.6961	17.2	82.7	81.7	4.93
Naive Bayes	76.1952	21.4	79.1	76.2	0.69
k-NN	93.4691	6.5	93.5	93.5	1.51
C4.5	97.3194	2.7	97.3	97.3	15.63
REPT	97.0680	2.9	97.1	97.1	3.43
RIPPER	96.7582	3.2	96.8	96.8	185.36
PART	97.3109	2.7	97.3	97.3	53.69

Table 3 Accuracies of network attack categorization for different feature sets for UNSW-NB15 [1]

Feature sets	Random Forest	Random Tree	Bayes	N. Bayes Network	*k*-NN	C4.5	REPT	RIPPER	PART
All features	87.08	84.17	65.28	46.16	80.62	87.66	86.62	80.24	87.05
FBSE+WBSE	82.85	80.92	74.25	17.95	76.59	82.56	82.33	76.67	82.30
PCA+feature normalization	85.85	83.48	71.55	41.87	81.27	85.78	85.13	79.58	85.65
Weka feature selection	82.99	82.85	74.55	57.57	82.5	83.00	82.8	76.44	82.9
Association rule mining	77.70	75.18	62.72	51.07	77.04	78.22	77.99	72.69	77.83
Correlation feature selection	74.31	71.93	60.32	43.05	73.59	74.58	74.28	71.15	74.3
10-fold CV	85.57	83.8	66.55	56.59	84.17	86.16	85.10	79.24	85.91
ICA	85.68	83.59	74.27	37.87	80.12	85.31	84.86	77.6	85.03
Genetic algorithm + SVM	77.35	74.18	69.68	35.83	71.67	76.05	76.80	71.86	75.78

experimental environment consists of *AMD Opteron ™ Processor 4386, 16 CPUs, 64 cores, 32 G RAM, and 1 TB of storage.* An example of a 60-s snapshot of the CPUs, memory, and network history in the case of running multinomial Logistic Regression model is given below in Fig. 1.

Similar to the previous work [1], the UNSW-NB15 [3, 15] dataset is used for the experiments. This dataset is a mixture of real normal traffic flow and synthetic attacks. The types of attacks and the number of each attack in the training dataset are shown in Fig. 2. The dataset has 82,332 records (37,000 normal and 45,332 anomalies) and 45 attributes or features (details in [1]).

Figure 3 illustrates the proposed anomaly and categorization framework. The first stage of the framework identifies if the network flow data is normal or anomalous by extracting the data context and doing classification using the best classifier. The first stage consists of seven algorithms (Winnow, Simple Logistic, MLR, MLP, Voted Perceptron, SVM, and *deepLearning*) and the purpose is to find the best error-based machine learning technique for anomaly detection vis-à-vis our research dataset. If normal, the processing continues and the access is allowed. If not, the second stage is engaged where the anomaly is categorized and type of attack identified. The second stage components include four machine learning algorithms: Simple logistic, MLP, SVM, and *deepLearning*. The second stage categorizes the anomaly types and extract the attack type.

4 Experimental results

The first set of experiments based on *proposition 1* for network anomaly detection consists of seven experiments with seven different error-based learning algorithms using training to testing ratio of 66 to 34%. Table 4 below shows the experimental results. Using accuracy and Kappa statistics, the following algorithms Winnow, Simple Logistic, Multilayer Perceptron (MLP), and SVM return the best (in fact perfect) results. Multinomial Logistic Regression (MLR) and

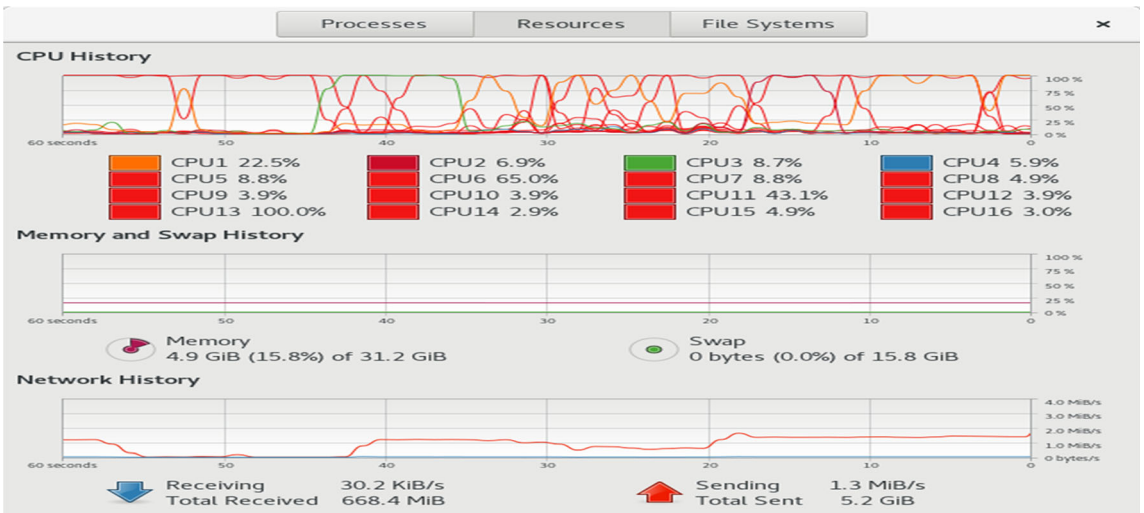

Fig. 1 A 60-s snapshot of running the Logistic Regression algorithm for the dataset

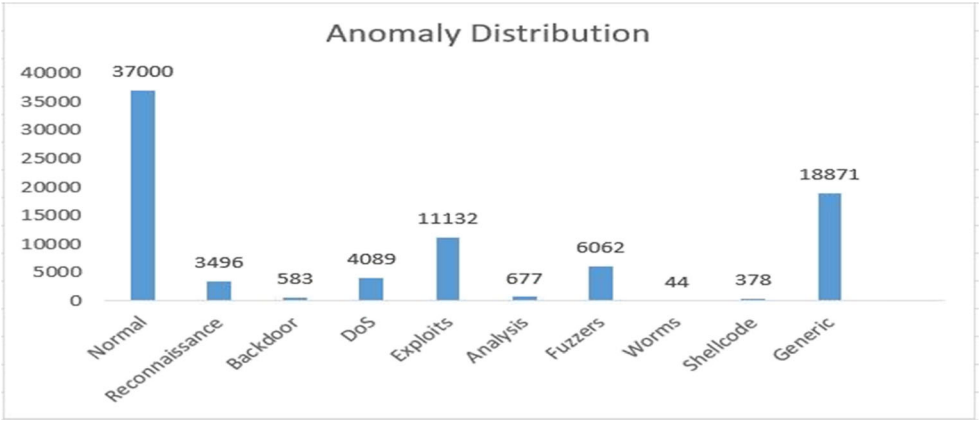

Fig. 2 Distribution of anomalies (attack types) in the UNSW-NB15 training dataset

deepLearning [7, 8, 21]are not far behind. The worst of the seven algorithms is Voted Perceptron. This is evident from the accuracy of 74.64% and Kappa statistics of 0.5015.

The thesis for proposition 2 is that "Given proposition 1, can we improve the accuracy of certain error-based algorithms using 10-fold cross-validation?" In Table 4, we already have perfect results in term of accuracy and Kappa statistics for four algorithms—Winnow, Simple Logistic, MLP, and SVM. There is nothing to improve in these four algorithms. However, we have three other algorithms—MLR, Voted Perceptron, and *deepLearning* that do not have "perfect" results. Hence, we carried out our second set of experiments consisting of three experiments for these three algorithms. The results of the experiments are shown in Table 5. There are no significant statistical differences between the results shown in Table 4 for these three algorithms compared to the results in Table 5.

Propositions 1 and 2 deal with identifying if a network instance is normal or anomalous. However, in propositions 3 and 4, we are interested in the attack category of network anomaly. That is, identifying the different types of anomaly. We cannot use three of the seven algorithms for propositions 3 and 4 because of their limitations. Winnow capabilities are for binary class and missing class values. It works very well for anomaly detection because it is simply a binary class (i.e., normal or anomalous) problem. Winnow deals with classifi-

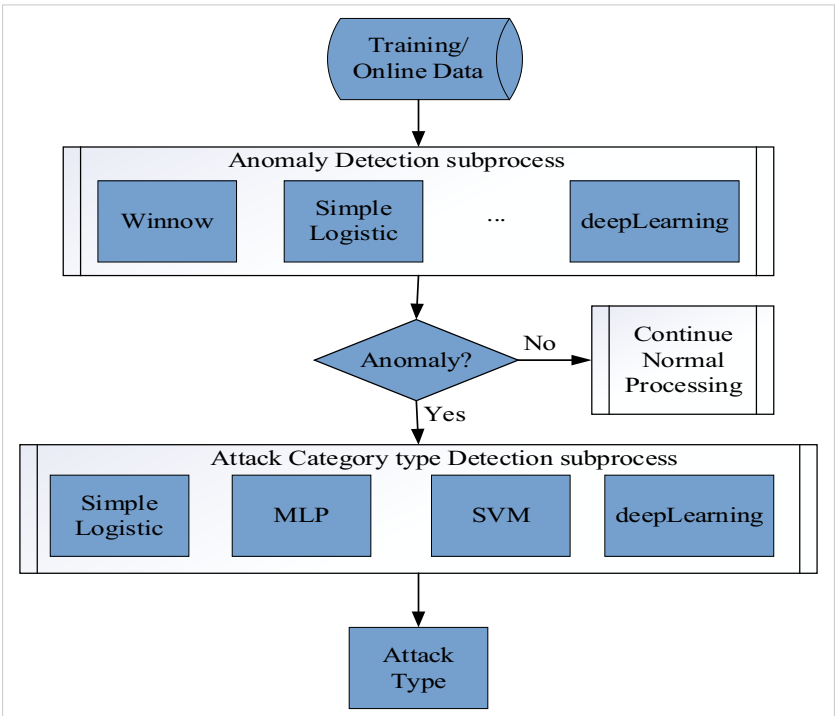

Fig. 3 Anomaly detection and categorization framework

Table 4 Simple anomaly detection with training to testing ratio of 66–34 data split

Machine learning algorithms	Accuracy (%)	False positive rate (%	Precision (%)	Recall (%)	Kappa statistic	Time to build(s)
Winnow	100.00	0.00	100.00	100.00	1.00	0.47
Simple Logistic	100.00	0.00	100.00	100.00	1.00	50.11
MLR	99.9857	0.00	100.00	100.00	0.9997	246.94
MLP	100.00	0.00	100.00	100.00	1.0000	18,171.65
Voted Perceptron	74.64	35.40	66.70	86.90	0.5015	136.44
SVM	100.00	0.00	100.00	100.00	1.000	290.43
deepLearning	97.9352	2.00	97.90	97.90	0.9583	1435.66

cation problem by converting nominal attributes to binary attributes. It maintains a non-negative weight initially set to 1 and the linear classifier rule is [22, 23] defined as "$if \sum_{i=1}^{k} w_i \lambda_i > threshold, then\ 1; otherwise\ 0$. " The threshold defines a dividing hyperplane. In the same way, MLR model capabilities are similar to that of Winnow. It works well on binary class, missing class, and nominal class. It also converts nominal values to binary. Voted Perceptron's capabilities are exactly the same as that of Winnow. In addition, its performance in terms of accuracy and Kappa statistics is not good even for a binary classification for network anomaly detection. As a result, we set up the third set of four experiments consisting of four algorithms: Simple Logistic, MLP, SVM, and *deepLearning*. The results are shown in Table 6.

From Table 6, we can see that the best result (i.e., algorithm) in terms of accuracy (89.4831%) and Kappa statistics (0.8533) is SVM. This is followed by Simple Logistic with 87.4933% accuracy and Kappa value of 0.8257, respectively. *deepLearning* came close with 84% and MLP roughly 83%. One important point to note here is that the four algorithms correctly classified all the "normal" classes; that is, they correctly partitioned the network instances 100% into whether they are normal or anomalous. See Table 7, an example confusion matrix produced by MLP which has the least accuracy among the four algorithms. It is evident that the normal class of 12,567 instances (i.e., a = Normal) is 100% correctly classified.

The thesis for proposition 4 is that "Given proposition 3, the accuracy of certain error-based machine learning algorithms for network anomaly attack categorization can be improved by using 10-fold cross-validation." The results shown in Table 8 indicate that there is no significant statistical difference between the results in Tables 6 and 8. Hence, the 10-fold cross-validation did not make the results statistically better than the 66–34% split for training and testing[24–27].

However, the confusion matrix (Table 9) shows that the classification between instances that are normal and those that are anomalous is 100% classified correctly. An example of confusion matrix for attack category for 10-fold cross-validation produced by Simple Logistic is given in Table 9. It is evident that the normal class of 37,000 instances (i.e., a = Normal) is 100% correctly classified.

Table 5 Simple anomaly detection with 10-fold cross-validation

Machine learning algorithms	Accuracy (%)	False positive rate (%)	Precision (%)	Recall (%)	Kappa statistic	Time to build(s)
MLR	99.9939	0.00	100.00	100.00	0.9999	244.35
Voted Perceptron	74.8385	35.20	66.90	87.10	0.5053	64.29
deepLearning	97.8672	2.00	97.90	97.90	0.9570	1583.50

Table 6 Attack category detection with training to testing ratio of 66–34 data split

Machine learning algorithms	Accuracy (%)	False positive (%)	Precision (%)	Recall (%)	Kappa	Time to build(s)
Simple Logistic	87.4933	1.20	?	87.50	0.8257	356.34
MLP	82.7671	2.40	?	82.80	0.7582	17,160.02
SVM	89.4831	1.10	89.90	89.50	0.8533	22,193.33
deepLearning	83.9103	1.60	85.90	83.90	0.7768	1716.73

Table 7 64–36% split confusion matrix for multilayer perceptron (MLP) for attack category

a	b	c	d	e	f	g	h	i	j	<–- classified as
12,567	0	0	0	0	0	0	0	0	0	| a=Normal
0	396	0	149	143	0	121	0	0	368	| b=Reconnaissance
0	0	0	188	7	0	4	0	0	9	| c=Backdoor
0	23	0	1015	241	0	11	0	0	104	| d=DoS
0	18	0	1107	2199	0	86	0	0	384	| e=Exploits
0	0	0	224	22	0	0	0	0	0	| f=Analysis
0	48	0	682	27	0	758	0	0	556	| g=Fuzzers
0	0	0	0	10	0	1	0	0	2	| h=Worms
0	38	0	13	0	0	18	0	0	46	| i=Shellcode
0	6	0	54	103	0	11	0	0	6234	| j=Generic

5 Discussions

The discussion of the experimental results is partitioned into two subsections based on the five propositions and the four research questions in the introduction.

5.1 Research propositions

The research proposition 1 focusing on the error-based learning algorithms can be confirmed based on the results presented in Table 4 compared to that in Table 2. There are four "perfect" scores for both the accuracy and Kappa statistics for the UNSW-NB15 dataset in the case of error-based learning methods—*Winnow*, *Simple Logistic*, MLP, and *SVM*. Looking at Table 2, the results of non-error-based machine learning algorithms from previous research work [1], there is no single learning model with 100% accuracy.

Research proposition 2 cannot be confirmed because there are no significant statistical differences between the results shown in Table 4 for the three algorithms (*MLR*, *Voted Perceptron*, and *Deep Learning*) compared to the results in Table 5 obtained from 10-fold cross-validation.

Research propositions 3 and 4 deal with the case of anomaly categorization. Comparing the first row of Table 3 from previous research work [1] with the highest accuracy of 87.66% for C4.5 algorithm and the results shown in Table 6, we may accept proposition 3 because the accuracy for SVM in Table 6 is 89.4831%. We may not confirm research proposition 4 although there is slight increase in the result of SVM in Table 8 (89.8484% vs. 89.4831%), since this increase is not statistically significant.

In the case of research proposition 5, that is, "The accuracy of error-based deep machine learning approach is likely to surpass other error-based machine learning algorithms in both network anomaly detection and network attack categorization." Starting with the results in Table 4, *deepLearning* has the sixth best accuracy and Kappa value. Table 5 shows that *deepLearning* model has the second-best accuracy and Kappa statistics. *deepLearning* is the third best prediction accuracy in Table 6, and finally, Table 8 shows that *deepLearning* model has the third best accuracy and Kappa value. From the forgoing, we may not accept research proposition 5 with respect to both network anomaly detection and network attack categorization.

5.2 Research questions

Research questions 1 and 2 deal with finding the best algorithm in terms of accuracy, Kappa statistics, and time to build the model. The best error-based learning algorithm for anomaly detection in response to question 1 is Winnow with accuracy of 100%, Kappa value of 1, and the model building time of 0.47 s (Table 3). This is evident in Fig. 4 below showing bar charts for the three parameters: accuracy, Kappa statistics, and time to build. In response to question 2, the best algorithm with respect to accuracy, Kappa statistics, and time to build for anomaly categorization is Simple Logistic. Although, looking at Table 6, SVM has the better accuracy (89.4831%) compared to Simple Logistic (87.4933%), the time for SVM is 22,193.33 s which is much higher compared to 356.34 s for Simple Logistic.

The subject of research questions 3 and 4 is to compare the work presented in this paper and the results of our previous work [1] in terms of the best algorithm for both anomaly detection and categorization detection. Examining both Tables 2 and 4, the response to research question 3 is the error-based learning algorithm *Winnow*. This algorithm has the best accuracy (100%) and the best time to build the model (0.47 s). For research question 4, looking at the first row of

Table 8 Attack category detection with 10-fold cross-validation

Machine learning algorithms	Accuracy (%)	False positive (%)	Precision (%)	Recall (%)	Kappa	Time to build(s)
Simple Logistic	87.4702	1.2	?	87.50	0.8252	361.67
MLP	82.3021	1.6	?	82.30	0.7533	19,499.54
SVM	89.8484	1.00	90.20	89.80	0.8584	22,979.75
deepLearning	83.4244	1.50	85.70	83.40	0.7704	1307.97

Table 9 10-fold confusion matrix for simple logistic for attack category

a	b	c	d	e	f	g	h i	j	<–- classified as
37,000	0	0	0	0	0	0	0 0	0	| a=Normal
0	2449	0	245	477	0	313	0 0	12	| b=Reconnaissance
0	33	0	26	237	0	286	0 0	1	| c=Backdoor
0	161	0	1843	1611	0	436	0 2	36	| d=DoS
0	398	0	1664	7767	0	1242	1 0	60	| e=Exploits
0	1	0	124	269	3	280	0 0	0	| f=Analysis
0	346	0	240	696	0	4763	0 1	16	| g=Fuzzers
0	1	0	0	29	0	10	0 0	4	| h=Worms
0	273	0	0	4	0	99	0 2	0	| i=Shellcode
0	50	0	13	430	0	186	3 0	18,189	| j=Generic

Table 3 and the results presented in Tables 6 and 7, SVM has the best accuracies—89.4831% for training to testing ratio (Table 6) and 89.8484% for 10-fold cross-validation (Table 7). In this case, SVM learning algorithm performs the best in terms of accuracy for anomaly categorization.

6 Conclusion and future work

In this paper, five research propositions were defined and four research questions were raised in the beginning. We have set up four sets of experiments to analyze the research propositions and to answer the questions.

Analyzing the results in Tables 4 and 5 for propositions 1 and 2, four error-based learning machine algorithms produce 100% accuracy for anomaly detection and there is no significant statistical improvement using 10-fold cross-validation for those algorithms. Likewise, for propositions 3 and 4, the error-based algorithm for network attack anomaly categorization is *SVM* (Tables 6 and 8) for both training to testing ratio and 10-fold cross-validation. However, *deepLearning*, a highly popular machine learning technique, does not perform better than other linear models in error-based classification. In fact, *deepLearning* has the sixth best accuracy and Kappa value as depicted in Table 4, the second-best accuracy and Kappa statistics as shown in Table 5, the third best category prediction accuracy as presented in Table 6, and finally, the third best accuracy and Kappa value as shown in Table 8. As a result, we may not confirm proposition 5.

In respect of questions 1 and 2, the responses are similar to that of propositions 1 to 4. For question 1, the best error-based machine learning algorithm for anomaly detection is *Winnow*—it has 100% accuracy, perfect Kappa value, and build time of 0.47 s. For question 2, *Simple Logistic* is the best error-based machine learning model overall for network attack categorization considering accuracy, Kappa value, and time to build together. On the other hand, if we look at accuracy only, *SVM* will be the best model for attack categorization. Comparing the results in this paper with our previous research work [1] using the same dataset with respect of questions 3 and 4, the best overall machine learning model for anomaly detection in terms of accuracy and time to build is the simple Error-based learning algorithm, *Winnow*, with a build time of 0.47 s and 100% accuracy. In terms of accuracy

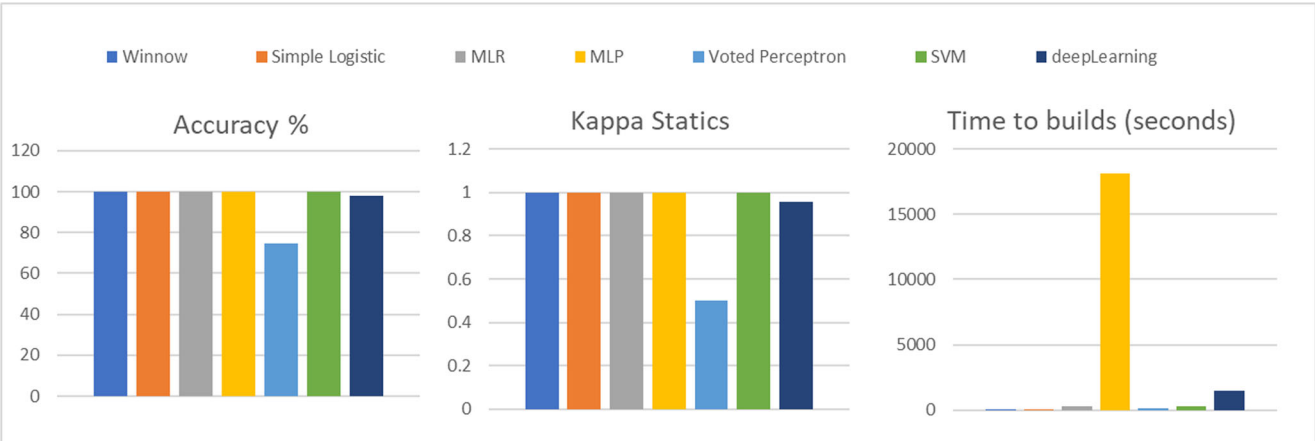

Fig. 4 Accuracy, Kappa statistics, and model building time for anomaly detection

only, the best overall machine learning model for network anomaly categorization is *SVM*.

The results obtained in this paper are significant when designing an online data streaming to detect network anomaly and categorize the attack anomaly. Time to build the model and the accuracy of the model are two most important criteria in online streaming of network anomaly detection. It is evident from this paper that *Winnow* is perfect for anomaly detection because of its accuracy and only 0.47 s to build the model. *Simple Logistic* with window sizing will be a good option for anomaly categorization due to its shorter build time, although its accuracy is less than that of *SVM*.

We have also evaluated error-based machine learning techniques with the BoT-IoT dataset as we did for information-, probability-, and partly similarity-based approaches in [1]. The results share high similarities with that of the current dataset used. Due to the page limit, the results are not reported in the paper. Nevertheless, a threat to validity that can be observed from this research work is the effect of classifiers' configuration on the accuracy and the time to build. Each of the machine learning algorithms can be configured differently when it comes to the choice of configuration parameters and this could affect the accuracy and the various statistical evaluations. In addition, the training sample size could change the accuracy or the variations of features of different datasets could affect also the result [27–30]. Finally, our general assumption in this paper is based on the fact that anomaly detection is a binary class problem. This assumption will fail if the problem cannot be reduced to a binary classification.

A future work in this wise is to investigate the use of auto-configuration in order to determine the best machine learning model especially for attack categorization.

Funding The authors received support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Das A, Ajila SA, Lung C-H (2020) A comprehensive analysis of accuracies of machine learning algorithms for network intrusion detection, IFIP international federation for information processing 2020, Published by Springer Nature Switzerland AG 2020, S. Boumerdassi et al. (Eds.): MLN 2019, LNCS 12081, pp 40–57
2. Kelleher JD, Mac Namee B, D'Arcy A (2015) Fundamentals of machine learning for predictive data analytics. The MIT Press, Cambridge ISBN 978-0-262-02944-5
3. Moustafa N, Slay J, Creech G (2019) Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. IEEE Trans Big Data 5(4):481–494
4. Nikravesh AY, Ajila SA, Lung C-H (2017) An autonomic prediction suite for cloud resource provisioning. J Cloud Comput 6(3):1–20
5. Witten IH, Frank E, Hall MA, Pal CJ (2017) Data mining – practical machine learning tools and techniques, 4th edn. Morgan Kaufmann, Elsevier, New York ISBN 978-0-12-804291-5
6. Bifet A, Gavaldà R, Holmes G, Pfahringer B (2017) Machine learning for data streams with practical examples in MOA. The MIT Press, Cambridge ISBN 978-0-262-03779-2
7. Charniak E (2018) Introduction to deep learning. The MIT Press, Cambridge ISBN 978-0-262-03951-2
8. S. Langa, F. Bravo-Marquezb, C. Beckhamc, M. Halld, and E. Franke, WekaDeeplearning4j: a deep learning package for Weka based on DeepLearning4j, Knowledge-Based Systems, 2019
9. N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, "Towards the development of realistic Botnet dataset in the Internet of things for network forensic analytics: Bot-IoT Datasethttps://arxiv.org/abs/1811.00701, 2018
10. T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets", Proc. of IEEE 26th Int'l Symposium on Industrial Electronics, 2017
11. N. Moustafa; J. Slay, "A hybrid feature selection for Network intrusion detection systems: central points", Proc. of the 16th Australian Information Warfare Conf., 2015
12. Idhammad M, Afdel K, Belouch M (2017) DoS detection method based on artificial neural networks. Int J Adv Comput Sci Appl 8(4)
13. M. Al-Zewairi, S. Almajali and A. Awajan, "Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system", Proc. of Int'l Conf on New Trends in Computing Sciences, 2017, pp. 167–172
14. H. Gharaee and H. Hosseinvand, "A new feature selection IDS based on genetic algorithm and SVM", Proc. of the 8th Int'l Symp. on Telecommunications, 2016, pp. 139–144
15. Moustafa N, Slay J The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 Data set. Inf Secur J 25(1):18–31
16. S. Marsland, Machine learning: an algorithmic perspective, 2, Chapman and Hall/CRC, 2014
17. N. Moustafa, and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data Set)", Proc. of IEEE Military Communications and Information Systems Conf. (MilCIS), 2015
18. Ajila SA, Bankole AA (2016) Using machine learning algorithms for cloud prediction models in a web VM resource provisioning environment. Trans Mach Learn Artif Intell 4(1):29–51
19. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell 2(1):41–50
20. S. Kumar and A. Yadav, "Increasing performance of intrusion detection system using neural network", Proc. of IEEE Int'l Conf on Advanced Communications, Control and Computing Technologies, 2014, pp. 546–550
21. Bouckaert RR (2008) Bayesian Network Classifiers in Weka for Version 3–5-7. https://www.cs.waikato.ac.nz/~remco/weka.bn.pdf, Access June 2020
22. Quinlan JR (1986) "Induction of decision trees", Machine Learning. 6 Kluwer academic publishers, Boston
23. H. Nguyen, K. Franke and S. Petrovic, "Improving effectiveness of intrusion detection by correlation feature selection", Proc. of Int'l Conf. on Availability, Reliability and Security, 2010, pp. 17–24
24. M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs", Proc. of Int'l Conf on Software, Knowledge, Information Management and Applications, 2014, pp. 1–6
25. H. T. Nguyen, K. Franke and S. Petrovic, "Towards a generic feature-selection measure for intrusion detection", Proc. of Int'l Conf on Pattern Recognition, 2010, pp.1529–1532

26. A. Zainal, M. A. Maarof and S. M. Shamsuddin, “Feature selection using rough set in intrusion detection”, Proc. of TENCON IEEE Region 10 Conference, Hong Kong, 2006, pp. 1–4
27. Z. Muda, W. Yassin, M. N. Sulaiman and N. I. Udzir, “Intrusion detection based on K-means clustering and Naïve Bayes classification”, Proc. of 7th Int’l Conf on Information Technology in Asia, 2011, pp. 1–6
28. Ingre B, Yadav A (2015) Performance analysis of NSL-KDD dataset using ANN. Proc. of Int’l Conf on Signal Processing and Communication Eng. Systems, Guntur, pp 92–96
29. T. Garg and S. S. Khurana, “Comparison of classification techniques for intrusion detection dataset using WEKA”, Proc. of Int’l Conf on Recent Advances and Innovations in Engineering, 2014, pp. 1–5
30. N. Paulauskas and J. Auskalnis, “Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset," Proc. of Open Conf. of Electrical, Electronic and Information Sciences (eStream), 2017, pp. 1–5