Pertanika J. Sci. & Technol. 26 (3): 1307 - 1332 (2018)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

On Evaluation of Network Intrusion Detection Systems: Statistical Analysis of CIDDS-001 Dataset Using Machine Learning Techniques

Abhishek Verma* and Virender Ranga

Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana, India

ABSTRACT

In this era of digital revolution, voluminous amount of data are generated from different networks on a daily basis. Security of this data is of utmost importance. Intrusion detection systems have been found to be one of the best solutions in detecting intrusions. Network intrusion detection systems are employed as a defence system to secure networks. Various techniques for the effective development of these defence systems are found in the literature. However, research on the development of datasets used for training and testing purposes of such defence systems is of equal concern. Better datasets improve the online and offline intrusion detection capabilities of detection models. Benchmark datasets like KDD 99 and NSL-KDD cup 99 are obsolete and do not contain network traces of modern attacks like Denial of Service, hence are unsuitable for the purpose of evaluation. In this study, a detailed analysis of CIDDS-001 dataset was conducted and the findings are presented. A wide range of well-known machine learning techniques were used to analyse the complexity of the dataset. Evaluation metrics including detection rate, accuracy, false positive rate, kappa statistics, and root mean squared error were utilised to assess the performance of employed machine learning techniques.

Keywords: Anomaly, decision tree, *k*-means clustering, *k*-nearest neighbour, labelled flow, metrics, random forests, signature

INTRODUCTION

Network security has rapidly become one of the most pressing issues and of concern for web users and service providers with a continual growth in web utilisation (Medaglia & Serbanati, 2010). A secure network can be characterised in terms of its hardware and software immunity against different intrusions. A network can be secured by

ARTICLE INFO

Article history:
Received: 29 December 2017
Accepted: 30 March 2018

E-mail addresses:
abhishek_6170034@nitkkr.ac.in (Abhishek Verma)
virender.ranga@nitkkr.ac.in (Virender Ranga)
*Corresponding Author

incorporating strong observing, examination and safeguard procedures. Network Intrusion Detection System (NIDS) incorporates these procedures to defend against network intrusions (Debar, Dacier & Wespi, 1999). These defence systems perform continuous monitoring of network traffic, analyse and report any intrusions. The major components of this system include traffic collector, analysis engine, signature database and alarm storage, as shown in Figure 1.

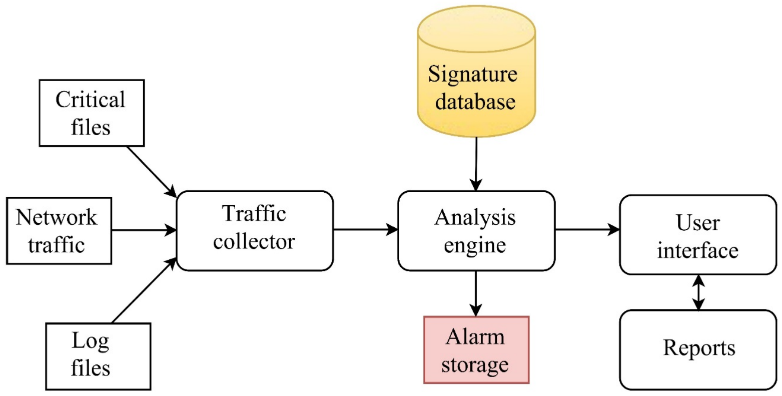


Figure 1. Components of intrusion detection system

Each component plays an important role in intrusion detection. Network traffic is captured by the traffic collector, that is, packet traces, analysis engine conducts a deep analysis of the captured traffic information and sends alarm signals to alarm storage when intrusion is detected. The signature database stores the signatures or patterns of known intruders, and these signatures are used for matching purpose. A typical NIDS is illustrated in Figure 2.

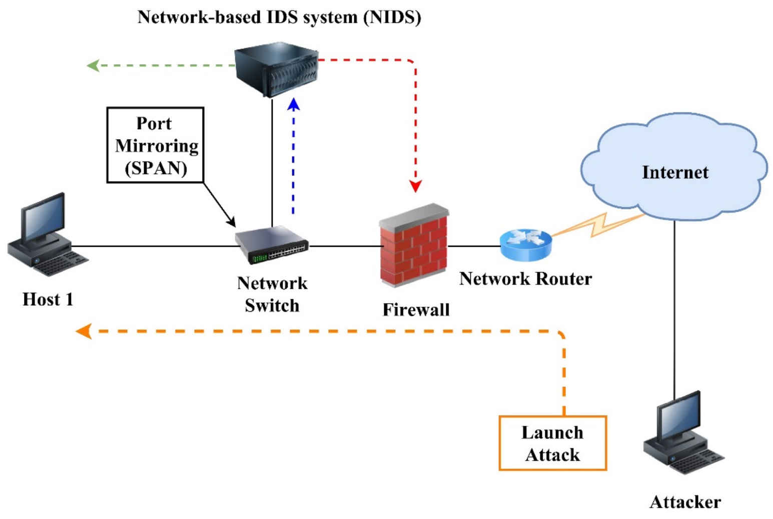


Figure 2. Illustration of network intrusion detection system

NIDS can be classified in two classes, that is, misuse detection (MD) (Zhengbing, Zhitang, & Junqi, 2008) and anomaly detection (AD) (Garcia-Teodoro, Diaz-Verdejo, Macia-Fernandez, & Vazquez, 2009). MD-based systems use traffic patterns of already known attacks for detecting intrusions in the network, while AD-based systems monitor any deviations from normal profiles of network behaviour. MD-based NIDS perform well in accuracy and have a significantly lower false alarm rate (FAR) but they perform poorly for unknown attacks. AD-based NIDS are capable of detecting novel intrusions or attacks, however, they score a higher FAR compared to MD-based NIDS.

Most of the benchmark datasets used for the evaluation of NIDS do not contain network traces of modern attacks (for example, denial of service, port scanning) which make them unsuitable for NIDS. This limitation is solved by CIDDS-001 dataset ("CIDDS-001 dataset", 2017) as it contains modern attack network traces. Machine Learning (ML) has proven to be very effective in the advancement of NIDS (Sommer & Paxson, 2010). It involves a detection system that learns from a dataset consisting of attack and normal packet traces and then classifies incoming network traffic into attack or normal class. The researchers used various well-known supervised and unsupervised learning-based ML models which are exhibited in Figure 3.

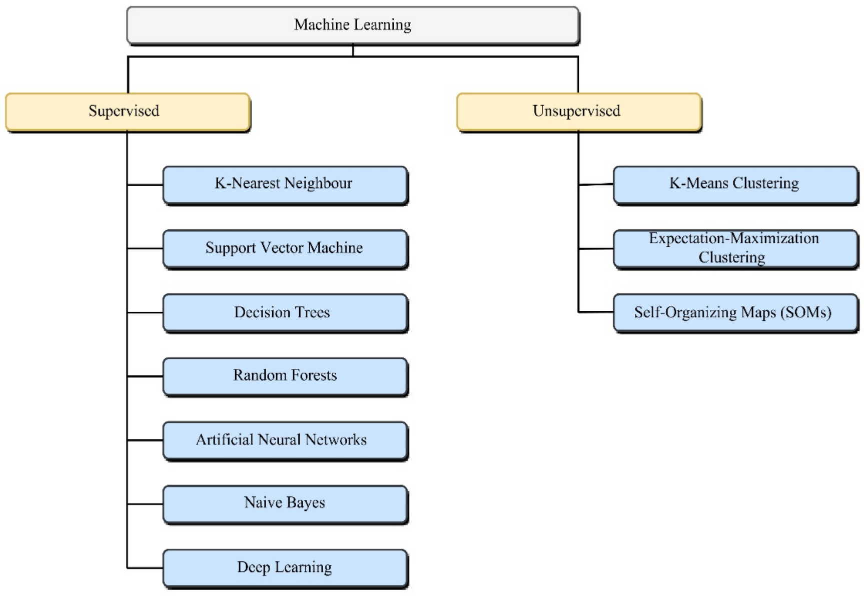


Figure 3. Machine learning techniques used in dataset analysis

CIDDS-001 Dataset

CIDDS-001 ("CIDDS-001", 2017) is a labelled flow-based dataset (Ring, Wunderlich, Grudl, Landes, & Hotho, 2017). It was developed primarily for the evaluation purpose of AD-based NIDS. The dataset consists of traffic from OpenStack and External Servers. CIDDS-001 has 13 features and one class attribute. A total of 11 features were used in this analytical study. The features of Attack ID and Attack Description were neglected in this study because they offer more information about executed attacks. Hence, these attributes did not significantly contribute to the analysis. About 153,026 instances from external servers and 172,839 instances from OpenStack Server data were collected for analysis. Each instance of the dataset was labelled as normal, attacker, victim, suspicious and unknown class. Table 1 provides the description of CIDDS-001 dataset attributes.

Table 1
Details of CIDDS-001

Sl. No.	Attribute Name	Attribute Description
1	Src IP	IP Address of the source node
2	Src Port	Port of the source node
3	Dest IP	IP Address of the destination node
4	Dest Port	Port of the destination node
5	Proto	Protocol
6	Date first seen	Start time flow first seen
7	Duration	Flow duration
8	Bytes	Transmitted bytes
9	Packets	Transmitted packets
10	Flags	TCP Flags
11	AttackDescription	Additional information about attack
12	AttackType	Type of attack
13	AttackID	Unique attack ID (same type of attacks have same ID)
14	Class	Category or label of the instance

Note: Retrieved from CIDDS-001 dataset. Copyright 2017 by CIDDS-001 dataset (Reprinted with permission)

RELATED WORK

An analytical study on NSL-KDD cup 99 dataset was conducted by Aggarwal and Sharma (2015; "NSL-KDD cup 99 dataset", 1999). The attributes of the dataset were categorised in four classes, that is, basic, content, traffic and host. The contribution of every class was evaluated in terms of DR and FAR. Siddiqui and Naahid (2013) performed the analysis of NSL-KDD dataset for Intrusion Detection (ID) using clustering algorithm-based data mining techniques. They used k-means clustering to build 1000 clusters over 494,020 records and focused on establishing a relationship between attack types and protocols used in intrusion. Artificial neural network (ANN) was utilised for the analysis of NSL-KDD dataset (Ingre & Yadav, 2015).

DR for intrusion detection and attack type classification was found to be 81.2% and 72.9%, respectively. According to Moustafa and Slay (2015), the study on irrelevant features of KDD 99 ("KDD 99 dataset", 1999) and UNSW-NB15 ("UNSW-NB15 dataset", 2017) found that they lead to reduction of NIDS efficiency. An association rule mining algorithm was used for the strongest feature selection from the two datasets and subsequently, classifiers did the task of evaluation in accuracy and false alarm rate (FAR). The results indicate that features of UNSW-

NB15 are much more efficient than the KDD 99 dataset. Kayacik and Zincir-Heywood (2005) studied three Intrusion Detection System (IDS) benchmark datasets using ML algorithms. Clustering and neural network algorithms analysed the IDS datasets and distinguished the differences between synthetic and real-world traffic.

Parsazad, Saboori and Allahyar (2012) proposed a fast feature selection method that finds low-quality features in the dataset. The variance of a random variable is used as a measure for finding the quality of a feature. The authors presented a comparison between popular existing similarity-based algorithms like maximal information compression index, correlation coefficient and least square regression error. The output of these algorithms recommend some features which are then fed to naive bayes and k-nearest neighbour classifiers for the purpose of testing the proposed method. This proposed technique outperformed existing similarity-based algorithms in terms of computational cost.

Rampure and Tiwari (2015) suggested a rough set theory-based feature selection on KDD Cup99 dataset. This is based on the premise that the degree of precision in the data is lowered and data pattern visibility is increased. Based on this premise, facts from imperfect data were discovered. Feature selection using Random Forests was documented in Hasan, Nasser, Ahmad and Molla (2016). The researchers derived a new dataset, RRE-KDD, after removing redundant records from KDD99Train + and KDD99Test+ sets of the NSL-KDD dataset. RRE-KDD is then used for the evaluation of Random Forest (RF). RF technique selects the most important features needed for classification and increases accuracy with reduction in time complexity. Janarthanan and Zargari (2017) analysed the UNSW-NB15 dataset using Weka tool. Different attribute selection techniques like CfsSubsetEval (attribute evaluator) with Greedy Stepwise method, and InfoGainAttibuteEval (attribute evaluator) with Ranker method were used for selecting important features. The best selected subset of attributes was utilised for classification using a few machine learning algorithms including RF. It was found that kappa statistics improved due to classification using selected features. A weighted feature selection method for wifi impersonation detection using AWID ("AWID dataset", 2018) dataset was proposed in Aminanto, Choi, Tanuwidjaja, Yoo and Kim (2018). The researchers used deep-feature extraction and selection for feature reduction in the dataset. The proposed approach achieved an accuracy of 99.918% and a FAR of 0.012%. Verma and Ranga (2018) presented an analytical study on CIDDS-001 using distance-based machine learning methods, whereby kNN and k-means algorithms were used for complexity analysis.

EXPERIMENTAL SETUP

Research Methodology

Weka tool was utilised for performing the analysis (Hall et al., 2009).

- Dataset preprocessing involving handling missing values and feature normalization was performed.
- Supervised and unsupervised machine learning algorithms were executed.
- Results of the simulated algorithms were tabulated and analysed.

Supervised Learning Algorithms

***k*-nearest Neighbour (kNN).** kNN is an instance-based learning and classification technique (Cover & Hart, 1967). Basic founding of kNN algorithm is distance function that calculates the correspondence or dissimilarity between two instances or points. There are different distance measures used in *k*NN. The most common distance measure is Euclidean distance. It can be defined as *D (a, b)* as Equation 1 (Kaur, 2014)

$$D(a,b) = \sqrt{\sum_{i=1}^{r}(a_i - b_i)^2} \qquad (1)$$

where a_i is the i^{th} featured element of the instance a, b_i is the i^{th} featured element of the instance b and r is the total number of the features in the dataset.

Support Vector Machine (SVM). SVM aims to find a hyper-plane which classifies all the training instances into different classes (binary classification or multiclass classification) (Suykens & Vandewelle, 1999). SVM algorithm takes observed instances and associated outputs, that is, binary or *N*-ary. Then, it designs a model that can classify new instances into different classes. Training instances are mapped as points in coordinate space, partitioning the instance input sets linearly. There can be the choice of many hyper-planes that can partition the training instance sets but the finest choice will be that with the maximum distance from the nearest instance of any class. In a case of two hyper-planes, *P* which classifies the instances correctly but has less distance from the nearest instance and *Q* which has maximum distance but has a small error in classification, hyper-plane *P* is selected in such case. SVM is effective for high dimensional spaces.

Decision Trees (DT). These are a type of supervised learning algorithms that are mostly used for solving classification problems of ML. Tree models in which the target variable can take discrete values as input are known as classification trees. DT consists of entities like leaves and branches. Leaves signify class labels and branches signify aggregations of attributes that lead to those class labels. It works with both discrete and continuous data. DT algorithm splits the samples into two or more homogeneous sets based on a most significant splitter in input variables. DT suffer from overfitting problem which can be handled by Bagging and Boosting (Quinlan, 1996). DT works effectively over discrete data. Figure 4 shows a typical example of DT (Bhargava, Sharma, Bhargava, & Mathuria, 2013).

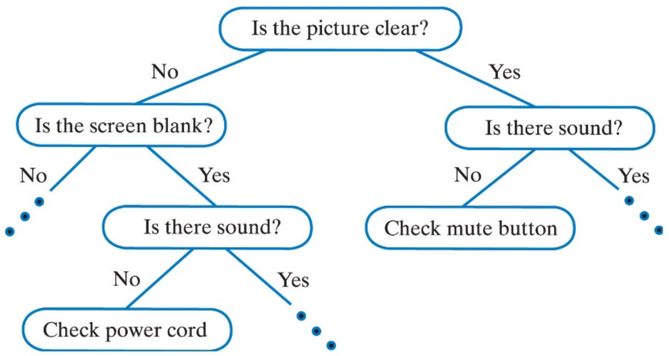


Figure 4. Decision tree [From "Decision Trees", para. 2, by Henrich 2018 (http://www.sfs.uni-tuebingen.de/~vhenrich/ss12/java/homework/hw7/decisionTrees.html). In the public domain.]

Random Forests (RF). As mentioned earlier DT suffers from overfitting problem. RF corrects this problem efficiently by averaging multiple deep decision trees (Breiman, 2001). RF is the ensemble learning algorithm used to solve classification and regression problems. Their operation involves the building of multiple DT during training time. The output is the mode of the classes of the distinct DT when classification task is being performed. RF gives better results than DT. A simple illustration of RF is shown in Figure 5.

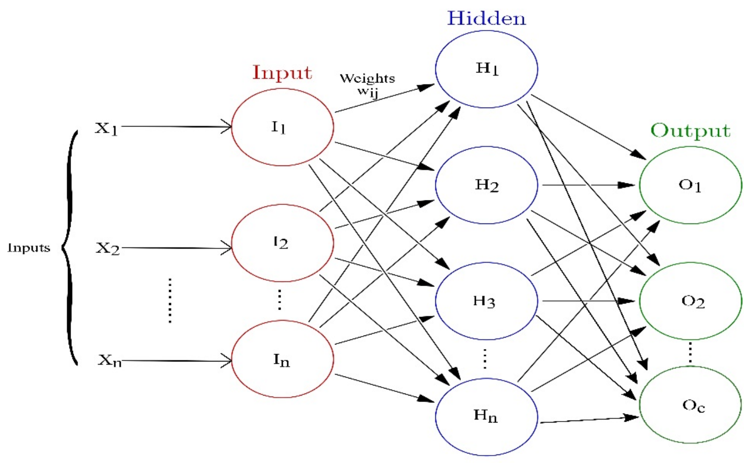


Figure 5. Artificial neural network

Artificial Neural Networks (ANN). ANN can be visualised as a weighted directed graph which consists of nodes and edges (Schalkoff, 1997). Nodes represent artificial neurons and directed edges with weights (strength between neurons) represent connections between artificial neurons. The output of one neuron acts as input to another neuron. ANN receives input from external world in the form of vector, that is, resembling some pattern or image. The weights are adjusted during learning of ANN which further help to solve the classification problems. ANN architecture consists of the input layer, output layer and hidden layer, each layer consists of neurons. Input layer receives input from the external world, output layer responds to the input fed to input layer on the basis of its learning capability. Hidden layer is intermediary between the input layer and output layer; it transforms the input in some manner such that output layer can utilise. These layers can be partially or fully connected. In this study, the researchers used multilayer perceptron model with back propagation learning.

General ANN architecture (I-H-O) for c class is shown in Figure 6, where I represents the count of input nodes, H represents the count of hidden layer nodes and O represent the count of output nodes.

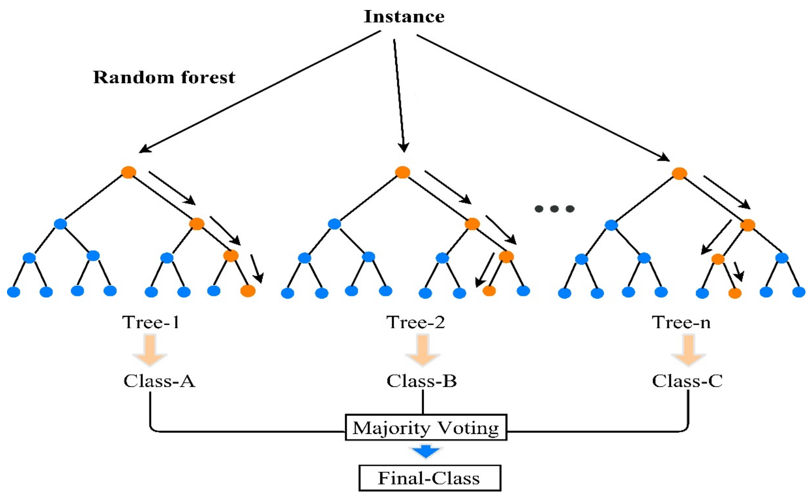


Figure 6. Random forests

Naive Bayes (NB). NB approaches are a family of simple probabilistic classifiers constructed by applying Bayes theorem. NB considers naive assumption of independence between every pair of features or attributes (Lewis, 1998). By applying a suitable pre-processing of training data NB can compete with most of the advanced approaches in its domain, that is, SVM and ANN. NB is easy to be trained using supervised learning configuration. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood. In other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods. Equation 2 represents Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

where *A* represents target attribute or dependent event, *B* represents predictor attribute or prior event. *P(A)* is said to be priori probability of *A* and *P(A|B)* is called as posteriori probability of *B* and *P(B|A)* is likelihood of *B* if the hypothesis *A* is true.

Deep Learning (DL). It is a method based on the learning of data representations in contrast to task definite methods without getting stuck to local minima. DL comprises ANN with more number of hidden layers making it more dense and complex (LeCun, Bengio, & Hinton, 2015).

It can be trained using supervised, semi-supervised or unsupervised learning. In this work, supervised learning is used. Cascaded multiple layers of neurons for feature extraction and transformation are used. It learns multiple representations of data that correspond to different levels of abstraction. Deep learning is applicable to many real-world problem solving situations. Figure 7 illustrates the deep learning model.

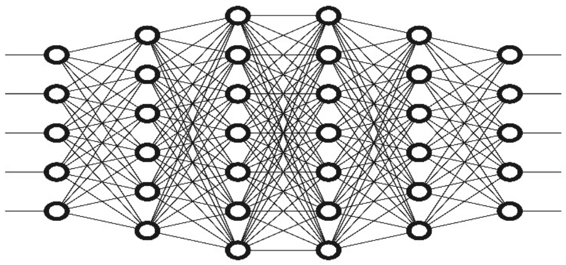


Figure 7. Deep learning network [From "Machine Learning, Deep Learning, and AI: What's the Difference?", para. 1, by Woodie, 2017 (https://www.datanami.com/2017/05/10/machine-learning-deep-learning-ai-whats-difference/). In the public domain.]

Unsupervised Learning Algorithms

***k*-means Clustering.** *k*-means is known to be one of the simplest unsupervised learning algorithm from distance-based perspective. It partitions n instances into k clusters, where each instance is grouped with the cluster having the nearest mean. Given a set of instances ($p_1, p_2, ..., p_n$), where each instance is a d-dimensional real vector. *k*-means clustering aims to partition p instances into k ($\leq p$) sets $Z = \{Z_1, Z_2, ..., Z_k\}$ in order to minimise the variance. *k*-means can be illustrated as Equation 3 (Kriegel, Schubert, & Zimek, 2016).

$$\arg_Z \min \sum_{i=1}^{k} \sum_{p \in Z_i}^{\max} \|p - \mu_i\|^2 = \arg_z \min \sum_{i=1}^{k} |Z_i| Var Z_i \tag{3}$$

where μ_i represents the mean of points in set Z_i.

Expectation-Maximisation Clustering (EM). EM clustering technique is very similar to k-means clustering (Moon, 1996). EM clustering extends the basic methodology of k-means clustering in two ways. EM algorithm calculates the probabilities of cluster memberships based on one or more probability distributions. EM aims to maximisze the overall probability of the data, given the final clusters.

Self-Organising Maps (SOM). It is based on unsupervised learning class of neural network models. SOM can perform clustering of data without prior knowledge of class categories of input data (Kohonen, 1998). SOM provides a topology preserving mapping from the high dimensional data space to map neurons (units). This mapping preserves the distance between points. Points which are near to each other are mapped to nearby maps units in the SOM. SOM network can recognise inputs which it has encountered before. Figure 8 represents SOM.

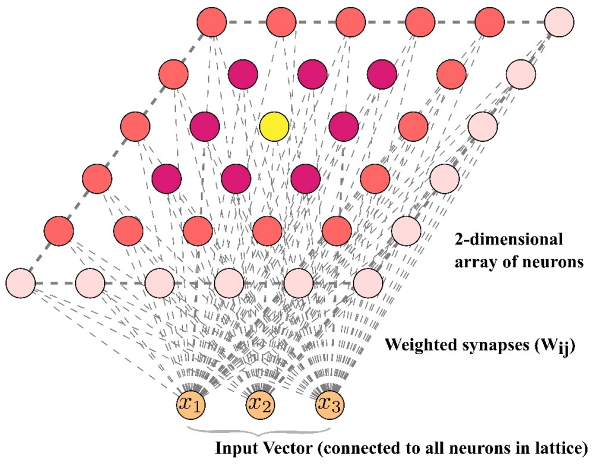

2-dimensional array of neurons
Weighted synapses (W_{ij})
x_1
x_2
x_3
Input Vector (connected to all neurons in lattice)

Figure 8. Self-Organising Maps (SOM)

Table 2 lists different Weka classes used for the analysis of CIDDS-001 dataset.

Table 2
Weka classes used for analysis of CIDDS-001

Machine Learning Techniques		Weka Class
Supervised Learning based Techniques	*k*-Nearest Neighbour	weka.classifiers.lazy.Ibk
	Support Vector Machine	weka.classifiers.functions.SMO
	Decision Trees	weka.classifiers.trees.J48
	Random Forests	weka.classifiers.trees.RandomForest
	Artificial Neural Networks	weka.classifiers.functions.MultilayerPerceptron
	Naive Bayes	weka.classifiers.bayes.NaiveBayes
	Deep Learning	weka.classifiers.functions.Dl4jMlpClassifier
Unsupervised Learning based Techniques	*k*-Means Clustering	weka.clusterers.SimpleKMeans
	Expectation-Maximization Clustering	weka.clusterers.EM
	Self-Organizing Maps	weka.clusterers.SelfOrganizingMap

Evaluation Metrics

Performance of machine learning classifiers in this analytical study were evaluated using eminent metrics, such as detection rate (DR), false positive rate (FPR), f-measure, accuracy, precision, root mean squared error and kappa statistics. All these metrics are evaluated from the elements of the confusion matrix. The elements of confusion matrix are true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Typically, TP represents the number of instances that are correctly classified as the attack. TN represents the number of instances that are correctly classified as normal. FP is the count of incorrectly classified normal instances as attack instances. Similarly, FN is the count of incorrectly classified attack instances as normal instances. Accuracy is defined as the ratio of all correctly classified instances (TP, TN) to all the instances (TP, TN, FP, and FN). Accuracy is denoted by Equation 4. DR (true positive rate) is the ratio of correctly classified instances (TP) as attacks to all the correctly classified attacks (TP) and normal instances (TN). DR is represented by Equation 5. Precision (positive predictive value) is the ratio of TP to a total of TP and FP. Equation 6 represents precision. The harmonic mean of precision and DR is known as f-measure. It is denoted by Equation 7.

Root mean squared error (RMSE) is a quadratic scoring rule which measures the average magnitude of the error (Levinson, 1946). It indicates the accuracy of the probability estimates that are generated by the classification model. Equation 8 represents the RMSE, where P is the original value or forecast value, *O* represents observed value and *n* is the number of samples.

In case of multi-class classification, evaluation measures like accuracy, precision and detection rate do not provide a full view of the classifier performance. Precision and detection rate are used in contrast to accuracy when there are imbalanced classes. Kappa statistics (K) is used in such case as it handles multi-class and imbalanced class like problems (Viera & Garrett, 2005). Kappa is defined in Equation 9, where Pr(*a*) is observed agreement and Pr(*e*) is expected agreement. K has value less than or equal to 1. Value of 0 or less represents that classifier is useless.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Detection \quad Rate = \frac{TP}{TP + TN} \tag{5}$$

$$\Pr ecision = \frac{TP}{TP + FP} \tag{6}$$

$$F - measure = \frac{2TP}{2TP + FP + FN} \tag{7}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (P_i - O_i)^2}{n}} \tag{8}$$

$$K = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \tag{9}$$

RESULTS AND DISCUSSION

Various supervised and unsupervised machine learning techniques were utilised for examining the complexity of CIDDS-001. In this study, 10 techniques were used which included classification techniques like *k*NN, SVM, DT, ANN, DL, RF, NB, and clustering techniques like *k*-means, EM, SOMs. All the experiments were done on Weka (version 3.9.1) using Intel(R) 7700 having a clock speed of 3.60 GHz processor with 8 GB primary memory running on Windows 10 Pro. Accuracy was given in scale between 0 and 1, that is, 0.36 would be 36% accuracy (multiplied by 100).

Analysis of Supervised Learning Algorithms

Analysis Using *k*NN. Firstly, *k*NN classifier is used for the analysis of external server traffic data. Features named "flows" and "tos" are not considered for the analytical study. Results of *k*NNclassifier execution are shown for 1, 2, 3, 4, and 5 neighbours in Table 3. Secondly, *k*NN classifier is analysed on OpenStack Server traffic data. The researchers selected 172839 instances from week 1 traffic data using reservoir sampling (Vitter, 1985). Results of kNN classifier execution are shown in Table 4. Approximately for every execution of *k*NN classifier on the external server traffic data, models average accuracy is 99%. Maximum accuracy of 99.6% was achieved with 2NN and minimum 99.3% with 5NN. Similarly, for *k*NN classifier execution on OpenStack traffic data models, average accuracy was 100% in each case, this may be due to a random sampling of instances from the dataset file which can lead to some biased instance selections. Dataset can be analysed using other evaluation metrics like ROC curve and FAR (Hand, 2009).

Table 3
Performance of kNN on external server data

Neighbours	Evaluation metrics					Class	Accuracy
	TP rate	FP rate	Precision	Detection rate	F-measure		
1NN	0.995	0.004	0.998	0.995	0.996	suspicious	0.995
	0.993	0.004	0.986	0.993	0.990	unknown	
	1.000	0.000	0.999	1.000	0.999	normal	
	1.000	0.000	1.000	1.000	1.000	attacker	
	1.000	0.000	1.000	1.000	1.000	victim	
2NN	0.997	0.006	0.997	0.997	0.997	suspicious	0.996
	0.990	0.003	0.991	0.990	0.990	unknown	
	1.000	0.000	0.999	1.000	1.000	normal	
	1.000	0.000	1.000	1.000	1.000	attacker	
	1.000	0.000	1.000	1.000	1.000	victim	
3NN	0.994	0.006	0.997	0.994	0.995	suspicious	0.994
	0.991	0.005	0.983	0.991	0.987	unknown	
	1.000	0.000	0.996	1.000	0.998	normal	
	1.000	0.000	1.000	1.000	1.000	attacker	
	1.000	0.000	0.996	1.000	0.998	victim	
4NN	0.996	0.007	0.996	0.994	0.996	suspicious	0.995
	0.989	0.003	0.988	0.989	0.988	unknown	
	1.000	0.000	0.996	1.000	0.998	normal	
	1.000	0.000	1.000	1.000	1.000	attacker	
	1.000	0.000	1.000	1.000	1.000	victim	
5NN	0.993	0.006	0.996	0.993	0.995	suspicious	0.993
	0.989	0.005	0.982	0.989	0.986	unknown	
	1.000	0.000	0.996	1.000	0.998	normal	
	1.000	0.000	1.000	1.000	1.000	attacker	
	1.000	0.000	1.000	1.000	1.000	victim	

Table 4
Performance of kNN on OpenStack server data

Neighbours	Evaluation metrics					Class	Accuracy
	TP rate	FP rate	Precision	Detection rate	F-measure		
1NN	1.000	0.000	1.000	1.000	1.000	victim	1.000
	1.000	0.001	1.000	1.000	1.000	normal	
	0.999	0.000	1.000	0.999	1.000	attacker	
2NN	1.000	0.000	1.000	1.000	1.000	victim	1.000
	1.000	0.001	1.000	1.000	1.000	normal	
	0.999	0.000	1.000	0.999	0.999	attacker	
3NN	1.000	0.000	1.000	1.000	1.000	victim	1.000
	1.000	0.001	1.000	1.000	1.000	normal	
	0.999	0.000	1.000	0.999	0.999	attacker	
4NN	1.000	0.000	1.000	1.000	1.000	victim	1.000
	1.000	0.001	1.000	1.000	1.000	normal	
	0.998	0.000	1.000	0.998	0.999	attacker	
5NN	1.000	0.000	1.000	1.000	1.000	victim	1.000
	1.000	0.001	1.000	1.000	1.000	normal	
	0.999	0.000	1.000	0.999	1.000	attacker	

Analysis Using Support Vector Machine. John Platt's sequential minimal optimisation algorithm was used to train SVM classifier (Platt, 1998). Firstly, SVM was trained over week 1 external server data. Accuracy of 95.3% was achieved in this case with RMSE of 0.320. Performance of SVM classifier on external server traffic is shown in Table 5. Secondly, SVM was trained over OpenStack server data.

In this case classifier achieved accuracy of 95.3% with RMSE of 0.272. Considerably good accuracy was achieved with SVM, hence a SVM based NIDS can be built. Modified algorithms for SVM training can be used to reduce model building time (Tsang, Kwok, & Cheung, 2005). Other variants of SVM can also be used to perform analysis of CIDDS-001. Performance of SVM classifier on OpenStack Traffic is tabulated in Table 6.

Table 5
Performance of SVM on external server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
0.976	0.088	0.951	0.976	0.964	suspicious	0.953
0.860	0.018	0.933	0.860	0.895	unknown	
0.981	0.001	0.968	0.981	0.974	normal	
1.000	0.000	0.999	1.000	1.000	attacker	
0.999	0.000	1.000	0.999	1.000	victim	

Table 6
Performance of SVM on OpenStack server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
0.997	0.000	0.999	0.997	0.998	victim	0.999
1.000	0.000	1.000	1.000	1.000	normal	
0.998	0.000	0.997	0.998	0.998	attacker	

Analysis Using Decision Trees. DT J48 (C4.5) is analysed over external server traffic data. It takes 4.61 seconds to build model for testing. Due to pruning characteristics of J48, model size significantly decreased the training and testing time. The accuracy of 99.7 % was achieved in the first case.

In second run J48 is trained over OpenStack server data. Model building time in this case is 1.27 seconds which is an acceptable time for NIDS training. Fortunately J48 provides 100% correct classifications. Efficient split decides the correctness of DT. Hence it can be concluded that J48 with pruning characteristics not only achieves a good accuracy but also manages space complexity. Table 7 shows the performance of DT on external server traffic data. Table 8 represents the performance of DT classifier on OpenStack server traffic data.

Table 7
Performance of DT on external server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
1.000	0.000	1.000	1.000	1.000	suspicious	0.997
1.000	0.000	0.999	0.999	0.000	unknown	
1.000	0.000	1.000	1.000	0.000	normal	
1.000	0.000	1.000	1.000	0.000	attacker	
1.000	0.000	1.000	1.000	0.000	victim	

Table 8
Performance of DT on OpenStack Server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
1.000	0.000	1.000	1.000	1.000	victim	1.000
1.000	0.000	1.000	1.000	1.000	normal	
1.000	0.000	1.000	1.000	1.000	attacker	

Analysis Using Random Forests. RF works by building many small classifiers and then collects votes from each one to decide the class of the test instance. This works on the voting method where small classifiers vote and the majority vote was selected as the output class. Firstly, RF was used for External Server data. It took 46.98 seconds to build the model. The accuracy of 99% was achieved in the first run. Performance of RF classifier on external server traffic is presented in Table 9.

Secondly, RF was used over OpenStack Server data. 100% accuracy was achieved second run model building time of 30.07 seconds. It can be observed that tree based algorithms perform well on CIDDS-001 dataset. RMSE is almost negligible in both cases. Performance of RF classifier on OpenStack server traffic is shown in Table 10.

Table 9
Performance of RF on external server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
1.000	0.000	1.000	1.000	1.000	suspicious	0.999
1.000	0.000	1.000	1.000	1.000	unknown	
1.000	0.000	1.000	1.000	1.000	normal	
1.000	0.000	1.000	1.000	1.000	attacker	
1.000	0.000	1.000	1.000	1.000	victim	

Table 10
Performance of RF on OpenStack server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
1.000	0.000	1.000	1.000	1.000	victim	1.000
1.000	0.000	1.000	1.000	1.000	normal	
1.000	0.000	1.000	1.000	1.000	attacker	

Analysis using Artificial Neural Networks. Analysis results show a very poor performance of ANN over both external and OpenStack server data as compared to other techniques employed. This may be possible due to improper dataset preprocessing. However, this can be improved by employing proper feature preprocessing methods (binary feature encoding). In the first run, ANN is tested over External Server data. About 63.8% accuracy is achieved while the model building time is 303.85 seconds which is very high. In second test ANN is tested on OpenStack server data which shows an accuracy of 8.26% with model building time of 413.63 seconds. Hence, ANN is unsuitable for NIDS development based on CIDDS-001. Table 11 and 12 show performance of ANN classifier on external and OpenStack server traffic data respectively.

Table 11
Performance of ANN on external server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
1.000	1.000	0.638	1.000	0.779	suspicious	0.638
0.000	0.000	0.000	0.000	0.000	unknown	
0.000	0.000	0.000	0.000	0.000	normal	
0.000	0.000	0.000	0.000	0.000	attacker	
0.000	0.000	0.000	0.000	0.000	victim	

Table 12
Performance of ANN on OpenStack server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
1.000	1.000	0.083	1.000	0.153	victim	0.083
0.000	0.000	0.000	0.000	0.000	normal	
0.000	0.000	0.000	0.000	0.000	attacker	

In this study, NB was tested first on external server traffic data. In the first case, NB yielded accuracy of 87.1% with 0.226 RMSE. NB took 0.27 seconds to build a model from training data. Secondly, NB is tested over OpenStack traffic. An accuracy of 99% was achieved with 0.074 RMSE. It took 0.34 seconds to build a model from training data in the second case. Results show the effectiveness of probabilistic classifiers, that is, NB takes less time in the model building while showing acceptable accuracy with minimum RMSE. Performance of NB classifier on external and OpenStack server traffic data is presented in Tables 13 and 14 respectively.

Table 13
Performance of NB on external server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
0.999	0.354	0.832	0.999	0.908	suspicious	0.871
0.426	0.001	0.994	0.426	0.597	unknown	
0.977	0.000	1.000	0.977	0.988	normal	
1.000	0.000	0.999	1.000	0.999	attacker	
0.999	0.000	0.999	0.999	0.999	victim	

Table 14
Performance of NB on OpenStack server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
0.998	0.000	1.000	0.998	0.999	victim	0.991
0.989	0.002	1.000	0.989	0.994	normal	
0.998	0.010	0.906	0.998	0.950	attacker	

Analysis using Deep Learning (deeplearning4j). Using the Java based deep learning class (deeplearning4j (http://Deeplearning4j.org)) CIDDS-001 was analysed. In the first run external server traffic data was analysed using DL. Model from training data was built in 916.07 seconds. An accuracy of 94.05% was achieved with RMSE of 0.139. Table 15 shows the performance of DL based classifier on external server traffic data.

In the second test OpenStack server traffic is analysed. The model is built in 457.47 seconds and instances are classified with 99.96% accuracy with 0.015 RMSE. It is quite clear that although it takes a long time to build the model, accuracy achieved is acceptable. DL can be used in high computation capable systems which aim to achieve higher accuracy in the long run. Performance of DL classifier on OpenStack server traffic data is presented in Table 16.

Table 15
Performance of DL on external server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
0.951	0.078	0.956	0.951	0.953	suspicious	0.941
0.874	0.039	0.864	0.874	0.869	unknown	
0.994	0.001	0.980	0.994	0.987	normal	
1.000	0.000	1.000	1.000	1.000	attacker	
1.000	0.000	1.000	1.000	1.000	victim	

Table 16
Performance of DL on OpenStack server data

Evaluation metrics					Class	Accuracy
TP rate	FP rate	Precision	Detection rate	F-measure		
0.997	0.000	0.999	0.997	0.998	victim	0.999
1.000	0.000	1.000	1.000	1.000	normal	
0.999	0.000	0.997	0.999	0.998	attacker	

Analysis of Unsupervised Learning Algorithms

Analysis using *k*-means Clustering. Firstly, *k*-means clustering was used for analysis of external server traffic data. Features named "flows" and "tos" were not considered for analytical study. Results of execution of k-means algorithm are shown in the form of Multi-class confusion matrix and tabulated in Table 17. A total 38.1086% instances are correctly clustered by the *k*-means algorithm. Secondly, *k*-means clustering is used over OpenStack server traffic data. About 150,000 instances from week 1 traffic data are selected using reservoir sampling. The results of the second execution are in Table 18. In this experiment 99.6627 instances are correctly clustered.

Table 17
Confusion matrix for k-means on external server data

k-means External server		Predicted class					Accuracy
		Suspicious	Unknown	Normal	Attacker	Victim	
Actual Class	suspicious	28952	3788	28061	17218	19833	0.381
	unknown	1977	14045	330	2545	14940	
	normal	32	20	3038	32	3058	
	attacker	0	4	719	8532	0	
	victim	2153	0	0	0	3749	

Table 18
Confusion matrix for k-means on OpenStack server data

k-means External server		Predicted class			Accuracy
		Victim	Attacker	Normal	
Actual class	victim	57955	0	0	0.997
	attacker	0	57963	0	
	normal	90	416	33576	

Analysis using Expectation-Maximisation Clustering. Firstly, EM algorithm was used to analyse external server traffic data. The same set of features as used in k-means clustering were used. In this experiment, accuracy of 45.9% was achieved with model building time of 32.07 seconds. The results of this experiment are shown in the form of multi-class confusion matrix and tabulated in Table 19. In the second experiment, EM was tested over OpenStack server traffic. In this analysis, accuracy of 49.3% was achieved and the model was built in 10.18 seconds. As compared to previously mentioned techniques, this method not only does time costly model building but also performs very badly. Confusion matrix for the second experiment is presented in Table 20.

Table 19
Confusion matrix for EM on external server data

EM External server		Predicted class					Accuracy
		Suspicious	Unknown	Normal	Attacker	Victim	
Actual Class	suspicious	2880	45238	5636	25	44073	0.459
	unknown	715	16202	15848	0	1072	
	normal	199	2	0	5898	81	
	attacker	0	0	0	8877	378	
	victim	2	0	0	5819	81	

Table 20
Confusion matrix for EM on OpenStack server data

EM OpenStack server		Predicted class			Accuracy
		Victim	Attacker	Normal	
Actual Class	victim	13094	0	1142	0.493
	normal	58905	71414	13083	
	attacker	0	14579	622	

Analysis using Self-Organising Maps. In the first experiment, SOM was used to analyse external server traffic data. SOM takes 601.37 seconds to build a model using training data. After applying testing data it was found that SOM correctly clustered 38.4% test instances. In the second experiment SOM was used to analyse OpenStack server traffic data. In this test, SOM built a model in 719.59 seconds. About 46.3% test instances were correctly clustered in this experiment. Table 21, and 22 show confusion matrix for first and second experiment respectively.

Table 21
Confusion matrix for SOM on external server data

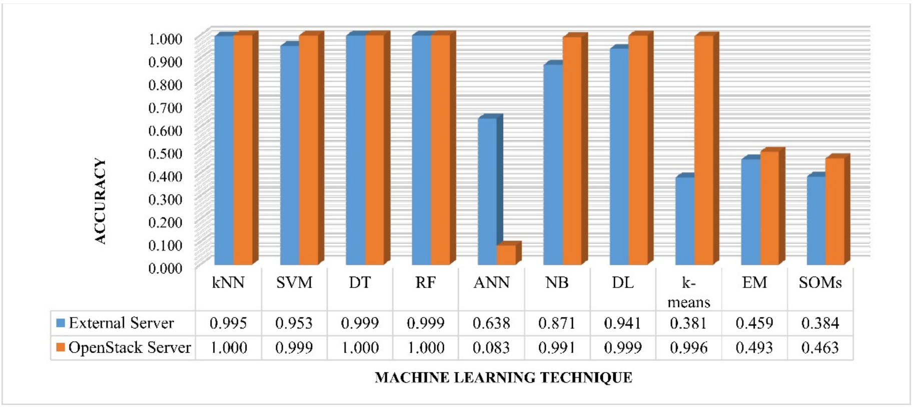
SOM External server		Predicted class				Accuracy
		Attacker	Suspicious	No class	Unknown	
Actual Class	suspicious	15399	34252	339938	14263	0.384
	unknown	16030	913	934	15960	
	normal	3090	0	0	3090	
	attacker	8754	0	0	501	
	victim	215	0	0	5684	

Table 22
Confusion matrix for SOM on OpenStack server data

SOM OpenStack Server		Predicted class				Accuracy
		Attacker	Suspicious	No class	Unknown	
Actual Class	victim	0	62	14174	0	0.463
	attacker	50877	20672	21781	50072	
	normal	0	14918	283	0	

Overall Evaluation

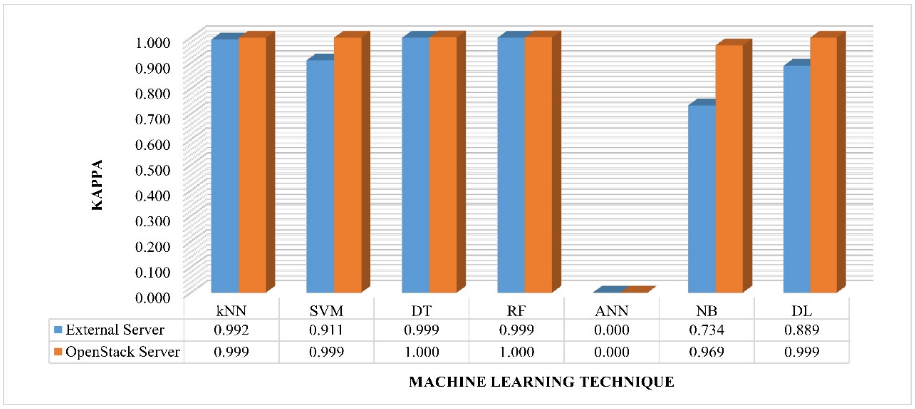
From the analysis results, it can be interpreted that most of the supervised learning based classification ML techniques perform better, only ANN fails to give acceptable accuracy. Figure 9 shows the performance of all the used techniques in accuracy. Almost all unsupervised learning based clustering techniques perform poorly. However *k*-means clustering gives good accuracy on OpenStack Server traffic data. These can be improved by proper data cleaning, binary feature encoding, normalisation and data standardisation methods. As clustering techniques require input data to follow a normal distribution for achieving better accuracy, in CIDDS-001 case features do not follow normal distribution and hence, poor accuracy is achieved. Performance of clustering techniques can be improved by capping, flouring and normalisation of attributes. Removal of outliers from the dataset can also improve the clustering performance. Kappa statistics for all classification techniques other than ANN is above average and hence it can be said that anomaly based NIDS using kNN, SVM, DT, RF, NB and DL can be developed.



	kNN	SVM	DT	RF	ANN	NB	DL	k-means	EM	SOMs
🟦 External Server	0.995	0.953	0.999	0.999	0.638	0.871	0.941	0.381	0.459	0.384
🟧 OpenStack Server	1.000	0.999	1.000	1.000	0.083	0.991	0.999	0.996	0.493	0.463

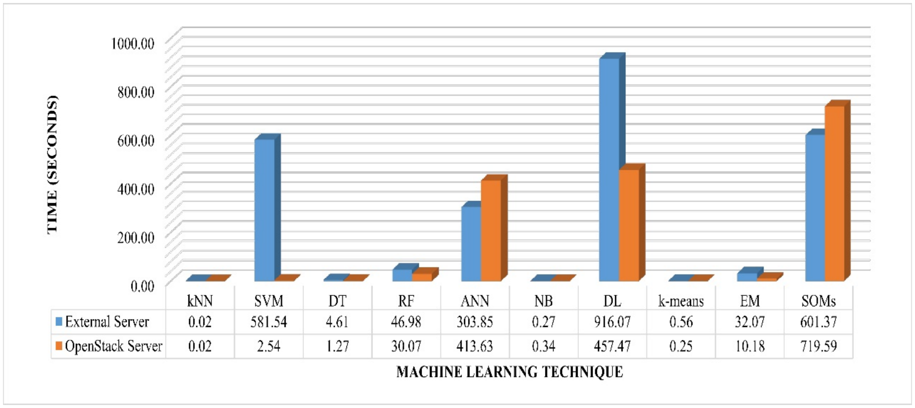
Figure 9. Performance of techniques in accuracy

Figure 10 and Figure 11 show the kappa statistics and model building time for different classification techniques used for the analysis. Model building time is the amount of time an algorithm (ML) takes to build a trained model from training data. This time should be less so that trained model can be employed for intrusion detection in minimum possible time. It can be observed that model building time for *k*NN, DT, RF, NB, *k*-means and EM methods is much less while ANN, DL, SOM methods take a much higher time to build a model. In case of DL, although it takes a long time to build a trained model, DL gives better accuracy once a model gets completely built. Root mean squared error is one of the important factors to analyse the performance of classifiers on a particular dataset.



	kNN	SVM	DT	RF	ANN	NB	DL
🟦 External Server	0.992	0.911	0.999	0.999	0.000	0.734	0.889
🟧 OpenStack Server	0.999	0.999	1.000	1.000	0.000	0.969	0.999

Figure 10. Performance of techniques in Kappa Statistics

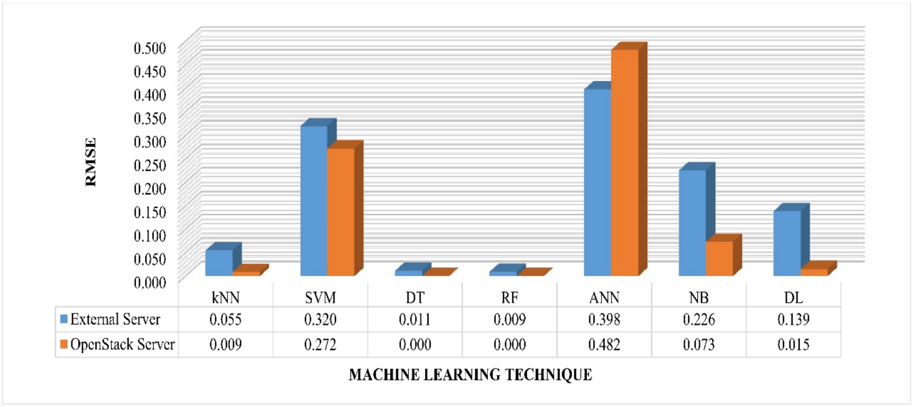


	kNN	SVM	DT	RF	ANN	NB	DL	k-means	EM	SOMs
🟦 External Server	0.02	581.54	4.61	46.98	303.85	0.27	916.07	0.56	32.07	601.37
🟧 OpenStack Server	0.02	2.54	1.27	30.07	413.63	0.34	457.47	0.25	10.18	719.59

Figure 11. Performance of techniques in model building time

DL can be a suitable choice if target system (where NIDS is to be deployed, that is, a router or some dedicated analysing machine) is having high computational power. Tree model-based techniques like DT and RF can be a suitable choice if the target system is not capable of performing high computations and lack higher storage capabilities.

Figure 12 shows the RMSE in case of different ML techniques used in this study. In this case, ANN attains highest RMSE over both external server and OpenStack server data. SVM outputs RMSE are between 0.250 and 0.350. This is observed by running SVM multiple times. All the performance parameters are closely related and affect each other. Hence, it can be concluded that by making a trade-off between different performance parameters the best technique can be selected for developing anomaly based or signature-based NIDS.



	kNN	SVM	DT	RF	ANN	NB	DL
🟦 External Server	0.055	0.320	0.011	0.009	0.398	0.226	0.139
🟧 OpenStack Server	0.009	0.272	0.000	0.000	0.482	0.073	0.015

Figure 12. Performance of techniques in root mean squared error

CONCLUSION

In this paper, the statistical and complexity analysis of CIDDS-001 dataset is presented and discussed. Supervised and unsupervised machine learning techniques are utilised to analyse the complexity of the dataset in eminent evaluation metrics. Evaluation results show that *k*-nearest neighbour, decision trees, random forests, naive bayes and deep learning based classifiers can be used to develop an efficient network intrusion detection systems. Based on the evaluation results it is concluded that CIDDS-001 dataset is suitable for the evaluation of Anomaly-based Network Intrusion Detection Systems. As follow-up, the researchers aim to carry out an in-depth comparative study of the CIDDS-001 dataset with existing benchmarking datasets.

ACKNOWLEDGMENTS

This paper is an extended version of the paper published in The 6th International Conference on Smart Computing and Communications, NIT Kurukshetra, India, 2017.

REFERENCES

Aggarwal, P., & Sharma, S. K. (2015). Analysis of KDD dataset attributes-class wise for intrusion detection. *Procedia Computer Science, 57*, 842-851.

Aminanto, M. E., Choi, R., Tanuwidjaja, H. C., Yoo, P. D., & Kim, K. (2018). Deep abstraction and weighted feature selection for Wi-Fi impersonation detection. *IEEE Transactions on Information Forensics and Security, 13*(3), 621-636.

AWID. (2018). *AWID dataset.* Retrieved January 2, 2018, from http://icsdweb.aegean.gr/awid/download.html

Bhargava, N., Sharma, G., Bhargava, R., & Mathuria, M. (2013). Decision tree analysis on j48 algorithm for data mining. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(6), 1114-1119.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.

CIDDS-001 dataset. (2017). CIDDS – Coburg intrusion detection data set. Retrieved January 22, 2018, from https://www.hs-coburg.de/forschung-kooperation/forschungsprojekte-oeffentlich/ingenieurwissenschaften/cidds-coburg-intrusion-detection-data-sets.html

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21-27.

Debar, H., Dacier, M., & Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks, 31*(8), 805-822.

Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., & Vazquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security, 28*(1), 18-28.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10-18.

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning, 77*(1), 103-123.

Hasan, M. A. M., Nasser, M., Ahmad, S., & Molla, K. I. (2016). Feature selection for intrusion detection using random forest. *Journal of Information Security,* 7(03), 129-140.

Henrich, V. (2018). *Decision trees.* Retrieved January 8, 2018, from http://www.sfs.uni-tuebingen.de/~vhenrich/ss12/java/homework/hw7/decisionTrees.html

Ingre, B., & Yadav, A. (2015). Performance analysis of NSL-KDD dataset using ANN. In *Proceedings of International Conference on Signal Processing and Communication Engineering Systems (SPACES)* (pp. 92-96). Guntur, India: IEEE.

Janarthanan, T., & Zargari, S. (2017). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In *Proceedings of 26th International Symposium on Industrial Electronics (ISIE)* (pp. 1881-1886). Edinburgh, UK: IEEE.

Kaur, D. (2014). A comparative study of various distance measures for software fault prediction. *International Journal of Computer Trends and Technology, 17*(3), 117-120.

Kayacik, H. G., & Zincir-Heywood, N. (2005). Analysis of three intrusion detection system benchmark datasets using machine learning algorithms. In *Proceedings of International Conference on Intelligence and Security Informatics* (pp. 362-367). Atlanta, GA, USA: Springer.

KDD 99 dataset. (1999). *Index of /ml/machine-learning-databases/kddcup99-mld.* Retrieved January 5, 2018, from http://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing, 21*(1), 1-6.

Kriegel, H. P., Schubert, E., & Zimek, A. (2016). The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems, 52*(2), 1-38.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444.

Levinson, N. (1946). The Wiener (root mean square) error criterion in filter design and prediction. *Studies in Applied Mathematics, 25*(1-4), 261-278.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of European Conference on Machine Learning* (pp. 4-15). Chemnitz, Germany: Springer.

Medaglia, C. M., & Serbanati, A. (2010). An overview of privacy and security issues in the internet of things. *The Internet of Things* (pp. 389-395). New York, USA: Springer.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine, 13*(6), 47-60.

Moustafa, N., & Slay, J. (2015). The significant features of the UNSW-NB15 and the KDD99 data sets for Network Intrusion Detection Systems. In *Proceedings of 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)* (pp. 25-31). Kyoto, Japan: IEEE.

NSL-KDD dataset. (1999). *KDD Cup 1999 Data Data Set*. Retrieved January 18, 2012, from http://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data.

Parsazad, S., Saboori, E., & Allahyar, A. (2012). Fast feature reduction in intrusion detection datasets. In *Proceedings of the 35th International Convention MIPRO* (pp. 1023-1029). Opatija, Croatia: IEEE.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. *Technical Report MSR-TR, 98*(14), 1-21. Retrieved from https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/.

Quinlan, J. R. (1996, August 04-08). Bagging, boosting, and C4. 5. In *Proceedings of Fourteenth National Conference on Artificial Intelligence* (pp. 725-730). Providence, Rhode Island: ACM.

Rampure, V., & Tiwari, A. (2015). A rough set based feature selection on KDD CUP 99 data set. *International Journal of Database Theory and Application, 8*(1), 149-156.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., & Hotho, A. (2017). Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security* (pp. 361-369). Dublin, Ireland: ACPI.

Schalkoff, R. J. (1997). *Artificial neural networks* (Vol. 1). New York: McGraw-Hill.

Siddiqui, M. K., & Naahid, S. (2013). Analysis of KDD CUP 99 dataset using clustering based data mining. *International Journal of Database Theory and Application, 6*(5), 23-34.

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of IEEE Symposium on Security and Privacy (SP), 2010* (pp. 305-316). Berkeley/Oakland, CA, USA: IEEE.

Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters, 9*(3), 293-300.

Tsang, I. W., Kwok, J. T., & Cheung, P. M. (2005). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research, 6*(Apr), 363-392.

UNSW-NB15 dataset. (2017). *The UNSW-NB15 data set description.* Retrieved January 12, 2018, from http://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/.

Verma, A., & Ranga, V. (2018). Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using distance-based machine learning. *Procedia Computer Science, 125*, 709-716.

Viera, A. J., & Garrett, J. M. (2005). Understanding interobserver agreement: The kappa statistic. *Family Medicine, 37*(5), 360-363.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS), 11*(1), 37-57.

Woodie, A. (2017). *Machine learning, deep learning, and AI: What's the difference?* Retrieved January 6, 2018, from https://www.datanami.com/2017/05/10/machine-learning-deep-learning-ai-whats-difference/

Zhengbing, H., Zhitang, L., & Junqi, W. (2008). A novel Network Intrusion Detection System (NIDS) based on signatures search of data mining. In *Proceedings of First International Workshop on the Knowledge Discovery and Data Mining, WKDD 2008.* (pp. 10-16). Adelaide, SA, Australia: IEEE.