On evaluation of Network Intrusion Detection Systems: Statistical analysis of CIDDS-001 dataset using Machine Learning Techniques

Abhishek Verma[1*], Virender Ranga[2]

[1]Department of Computer Engineering, NIT Kurukshetra, Haryana, India

abhishek_6170034@nitkkr.ac.in

[2]Department of Computer Engineering, NIT Kurukshetra, Haryana, India

virender.ranga@nitkkr.ac.in

ABSTRACT

In the era of digital revolution, a huge amount of data is being generated from different networks on a daily basis. Security of this data is of utmost importance. Intrusion Detection Systems are found to be one the best solutions towards 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 have been proposed in the literature. However, the research on the development of datasets used for training and testing purpose of such defence systems is equally concerned. Better datasets improve the online and offline intrusion detection capability of detection model. Benchmark datasets like KDD 99 and NSL-KDD cup 99 obsolete and do not contain network traces of modern attacks like Denial of Service, hence are unsuitable for the evaluation purpose. In this work, a detailed analysis of CIDDS-001 dataset has been done and presented. We have used different well-known machine learning techniques for analysing the complexity of the dataset. Eminent evaluation metrics including Detection Rate, Accuracy, False Positive Rate, Kappa statistics, Root mean squared error have been used to show the performance of employed machine learning techniques.

Keywords: Anomaly, Decision tree, *k*-means clustering, *k*-nearest neighbour, Labelled flow, Metrics, Random forests, Signature

Evaluation of Network Intrusion Detection Systems

INTRODUCTION

Network security has turned out to be a standout amongst the most concerning issues for web users and service providers with an extreme increment in the web utilization, Medaglia and Serbanati (2010). A secure network can be characterized in terms of its hardware and software immunity against different intrusions. A network can be secured by incorporating a strong observing, examination and safeguard procedures. Network Intrusion detection system (NIDS) (Debar, Dacier, & Wespi, 1999) incorporates these procedures for providing a defence against network intrusions. These defence systems perform continuous traffic monitoring in a network, analyse and report the intrusions. The major components of Intrusion detection system include traffic collector, the analysis engine, signature database and alarm storage as shown in Figure 1. Each component plays an important role in intrusion detection. Network traffic is captured by traffic collector i.e., packet traces, analysis engine performs the deep analysis of captured traffic and send alarms signal to alarm storage when some intrusion is detected. The signature database has the signatures or patterns of known intruders, these signatures are used for matching purpose. A typical NIDS is illustrated in Figure 2.

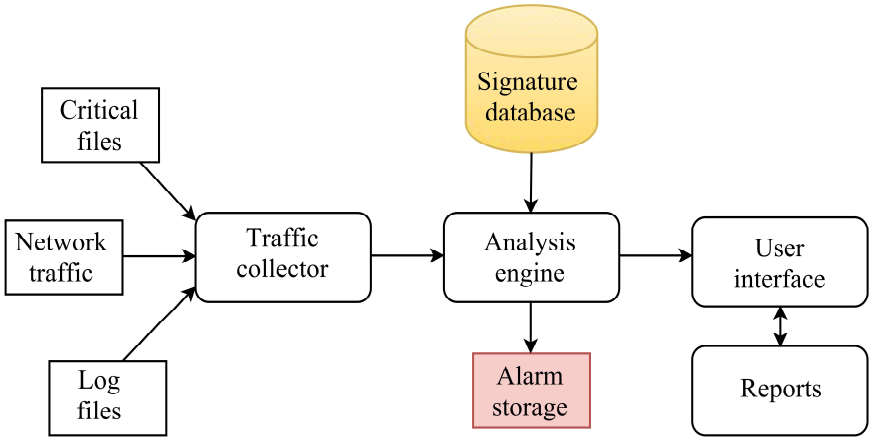

Figure 1. Components of Intrusion detection system.

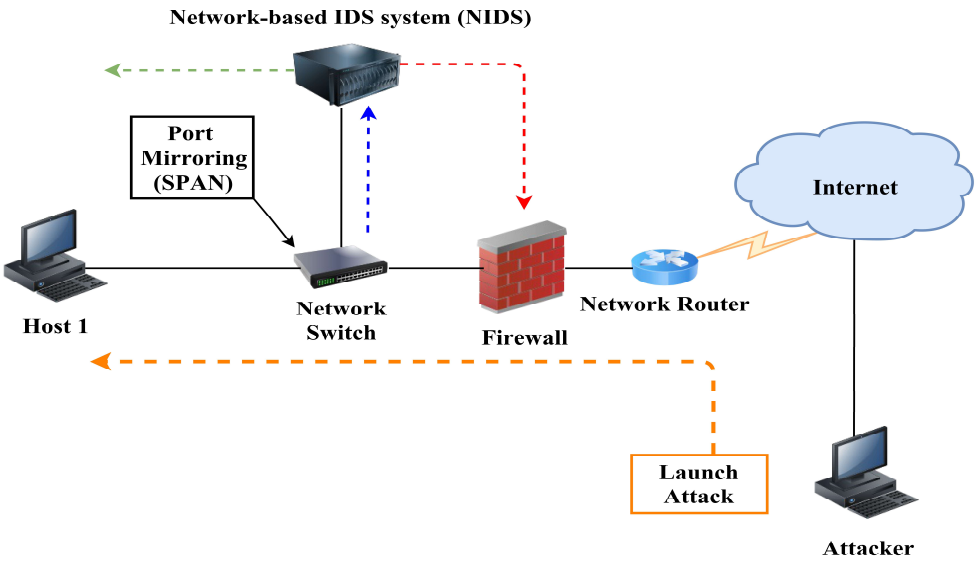

Figure 2. Illustration of Network intrusion detection system.

NIDS can be classified into two classes i.e., Misuse detection (MD) (Zhengbing, Zhitang, & Junqi, 2008) and Anomaly Detection (AD) (Garcia, Diaz, Macia, & Vazquez, 2009). MD based systems use traffic patterns of already known attacks for detecting intrusions in the network. Whereas AD based systems monitor any deviations from normal profiles of network behaviour. MD based NIDS perform well in terms of accuracy and have significantly less false alarm rate (FAR) but they perform poorly for unknown attacks. Whereas AD based NIDS are capable of detecting novel intrusions or attacks, however, they have high FAR as compared to MD based NIDS.

Most of the benchmark datasets used for the evaluation of NIDS do not contain network traces of modern attacks (i.e., Denial of Service, Port Scanning) which makes them unsuitable for NIDS. This limitation is overcome by CIDDS-001 dataset ("CIDDS-001 dataset", 2017) as it contains modern attacks network traces. Machine Learning (ML) (Sommer & Paxson, 2010) has been proved to be very effective in the advancement of NIDS. It involves a detection system to learn from a dataset consisting of attack and normal packet traces and then perform classification of incoming network

traffic into attack or normal class. We have used various well known supervised and unsupervised learning based ML models which are listed in Figure 3.

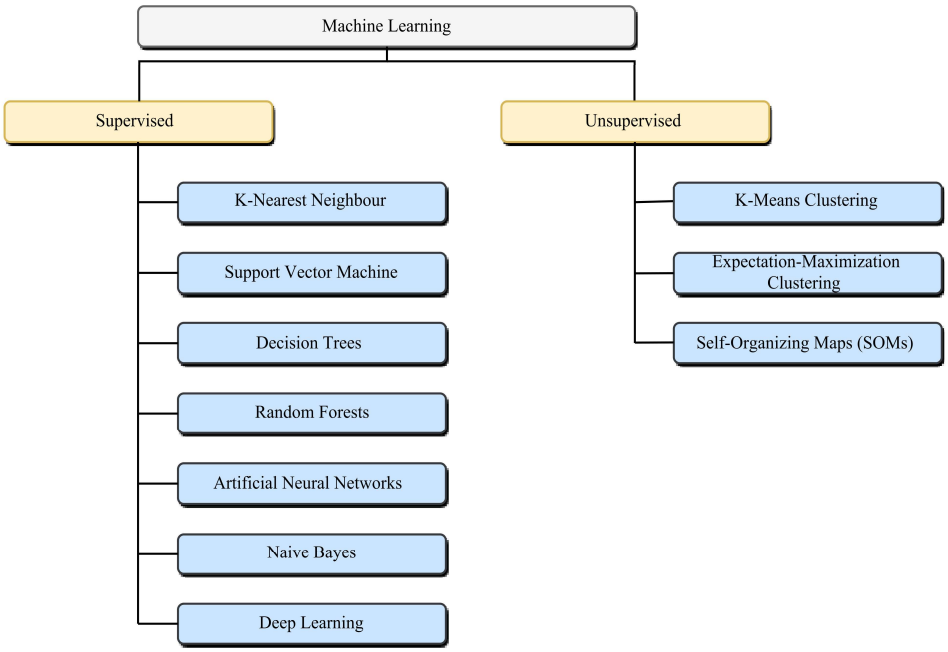

Figure 3. Machine Learning techniques used in dataset analysis.

CIDDS-001 dataset

CIDDS-001 ("CIDDS-001", 2017) is a labelled flow (Ring, Wunderlich, Grudl, Landes, & Hotho, 2017) based dataset. This dataset 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 1 class attribute. 11 features have been used in this analytical study. We have neglected AttackID and AttackDescription features because they just give more information about executed attacks. Hence these attributes do not play an important role in the analysis. 153026 instances from External Server and 172839 instances from OpenStack Server data have been used for the analysis. Each instance of the dataset is labelled as Normal, Attacker, Victim, Suspicious and Unknown class. Table 1 provides the description of CIDDS-001 dataset attributes.

RELATED WORK

In (Aggarwal & Sharma, 2015) the analytical study on NSL-KDD cup 99 ("NSL-KDD cup 99 dataset", 1999) dataset is done. The attributes of the dataset are categorized into four classes i.e., Basic, Content, Traffic and Host. The contribution of every class is evaluated in terms of DR and FAR. (Siddiqui & 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 494020 records and focused on building a relationship between attack types and protocols used in performing intrusion. Artificial neural network (ANN) is used in (Ingre & Yadav, 2015) for the analysis of NSL-KDD dataset. DR for intrusion detection and attack type classification was found

Table 1
Detail of CIDDS-001 ("CIDDS-001", 2017)

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

to be 81.2% and 72.9% respectively. In (Moustafa & Slay, 2015) the study on irrelevant features of KDD 99 ("KDD 99 dataset", 1999) and UNSW-NB15 ("UNSW-NB15 dataset", 2017) which leads to the reduction of NIDS efficiency. An Association Rule Mining algorithm is used for strongest feature selection from the two datasets and then classifiers are used for the evaluation in terms of accuracy and False alarm rate (FAR).

In results, it is shown that features of UNSW-NB15 are much efficient than the KDD 99 dataset. (Kayacik & Zincir, 2005) studied three Intrusion Detection System (IDS) benchmark datasets using ML algorithms. Clustering and Neural Network algorithms are used to analyse the IDS datasets and find the differences between synthetic and real-world traffic. (Parsazad, Saboori, & Allahyar, 2012) proposed a fast feature selection method which is based on finding the features with low quality in the dataset. The variance of a random variable is used as a measure for finding the quality of a feature. Authors presented a comparison with exiting prominent similarity based algorithms like Maximal Information Compression Index, Correlation coefficient and Least Square Regression Error. The output of these algorithms are some recommended features which are then fed to naive bayes and k-nearest neighbour classifiers for testing the proposed method. Proposed technique outperforms existing similarity-based algorithms in terms of computational cost. (Rampure & Tiwari, 2015) proposed a rough set theory based feature selection on KDD Cup99 dataset. They have used the premise that if the degree of precision in the data is lowered then data pattern visibility is increased. Based on this premise, facts from imperfect data are discovered. Feature selection using Random Forests is presented in (Hasan, Nasser, Ahmad, & Molla, 2016). They 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 the reduction in time complexity. (Janarthanan & Zargari, 2017) analyzed the UNSW-NB15 dataset using Weka tool. They used different attribute selection techniques like CfsSubsetEval (attribute evaluator) with the GreedyStepwise method and InfoGainAttibuteEval (attribute evaluator) with Ranker method for selecting important features. The selected best subset of attributes is used for classification using few machine learning algorithms including RF. It is shown that the Kappa statistics is improved from the classification using selected features. A weighted feature selection method for Wi-Fi Impersonation detection using AWID (“AWID dataset”, 2018) dataset is proposed in (Aminanto, Choi, Tanuwidjaja, Yoo, & Kim, 2018). They used a 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 & Ranga, 2018) presented an analytical study on CIDDS-001 using distance based machine learning methods. They used *k*NN and *k*-means algorithms for complexity analysis.

EXPERIMENTAL SETUP

Research methodology

- Weka tool (Hall et al., 2009) is utilized for performing the analysis.
- Dataset preprocessing is performed which involves handling missing values and feature normalization.
- Supervised and unsupervised machine learning algorithms are executed.
- Results of the simulated algorithms are tabulated and analyzed.

Supervised Learning Algorithms

***k*-nearest neighbour (*k*NN).** *k*NN (Cover & Hart, 1967) is an instance-based learning and classification technique. Basic founding of *k*NN algorithm is a 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).

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

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 (Suykens & Vandewelle, 1999) aims to find a hyper-plane which classifies all the training instances into different classes (binary classification or multiclass classification). SVM algorithm takes observed instances and associated outputs i.e., 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. Suppose there are 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). They 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 a 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).

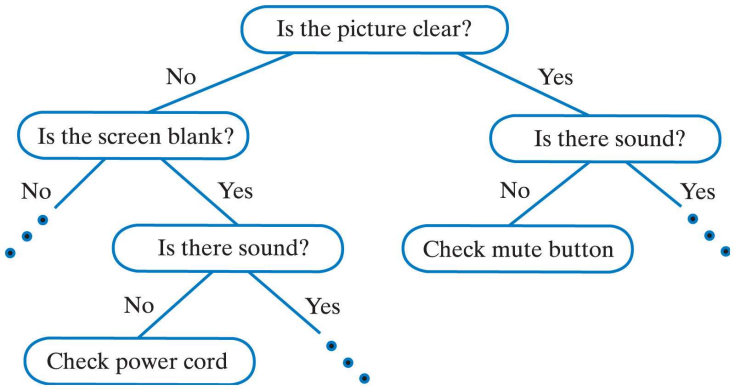

Figure 4. Decision Tree (Henrich, 2018, "Decision Trees", para. 2).

Random Forests (RF). As mentioned earlier DT suffers from overfitting problem. RF (Breiman, 2001) correct this problem efficiently by averaging multiple deep decision trees. 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.

Artificial neural networks (ANN). ANN (Schalkoff, 1997) can be visualized as a weighted directed graph which consists of nodes and edges. 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

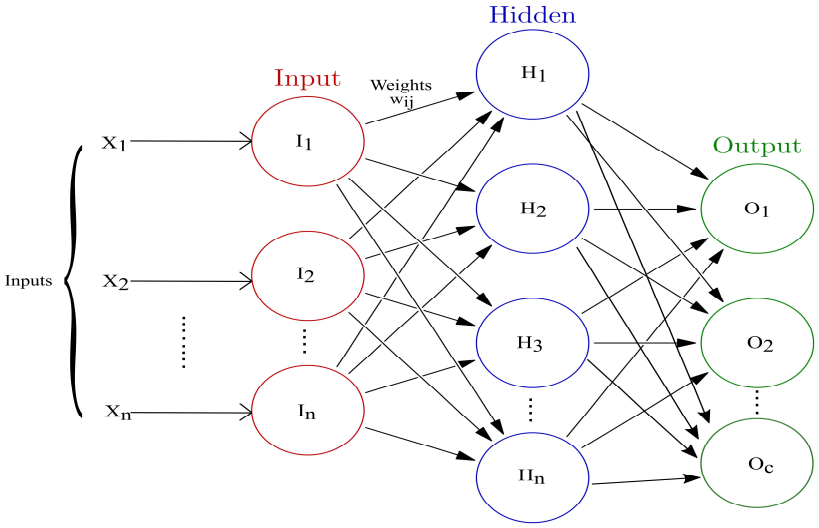

Figure 5. Artificial Neural Network.

another neuron. ANN receives input from external world in the form of vector i.e., 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 utilize. These layers can be partially or fully connected. In this study, we have used multilayer perceptron model with back propagation learning. General ANN architecture (I-H-O) for *c* class is shown in Figure 6, where I represent the count of

input nodes, H represents the count of hidden layer nodes and O represent the count of output nodes.

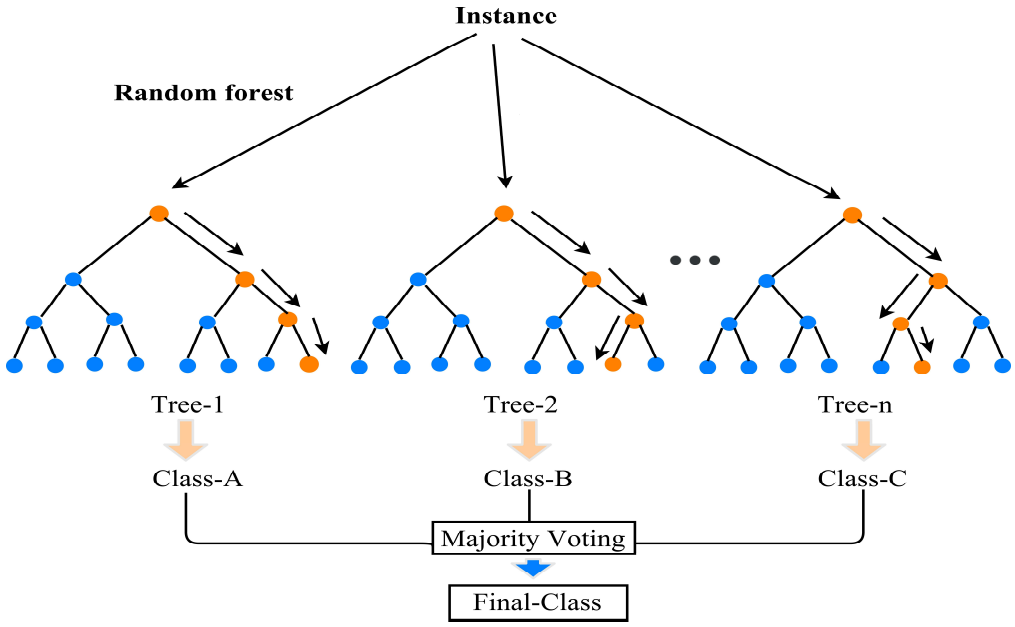

Figure 6. Random Forests.

Naive Bayes (NB). NB (Lewis, 1998) 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. By applying a suitable pre-processing of training data NB can compete with most of the advanced approaches in its domain i.e., 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.

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

Where A represents target attribute or dependent event. B represent 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 (LeCun, Bengio, & Hinton, 2015) comprises of ANN with more number of hidden layers making it more dense and complex. It can be trained using supervised, semi-supervised or unsupervised learning. In this work, we have used supervised learning. 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 problems solving. Figure 7 illustrates a Deep learning model.

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 nearest mean. Given a set of

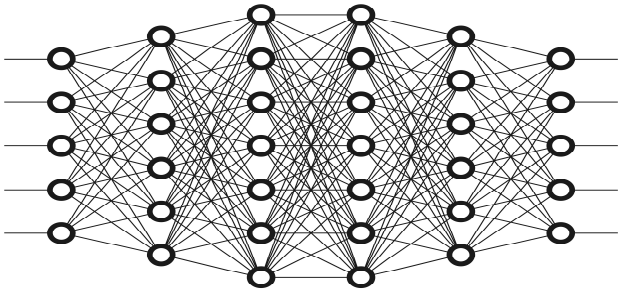

Figure 7. Deep Learning Network (Woodie, 2017, "Machine Learning, Deep Learning, and AI: What's the Difference?", para. 1).

instances ($p_1, p_2, \ldots, p_n$), where each instance is a *d*-dimensional real vector. *k*-means clustering aims to partition *p* instances into *k* ($\leq p$) sets $Z = \{Z_1, Z_2, \ldots, Z_k\}$ in order to minimize the variance. *k*-means can be illustrated as Equation 3 (Kriegel, Schubert, & Zimek, 2016).

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

Where μ_i represents the mean of points in set Z_i.

Expectation-Maximization clustering (EM). EM (Moon, 1996) clustering technique is very similar to *k*-means clustering. 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 maximize the overall probability of the data, given the final clusters.

Self-Organizing Maps (SOM). It is based on unsupervised learning class of neural network models. SOM (Kohonen, 1998) can perform clustering of data without having the prior knowledge of class categories of input data. 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 recognize inputs which it has encountered before. Figure 8 represents SOM.

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

Evaluation metrics

Performance of machine learning classifiers is evaluated using some eminent metrics. In this analytical study, 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 denoted by Equation 7.

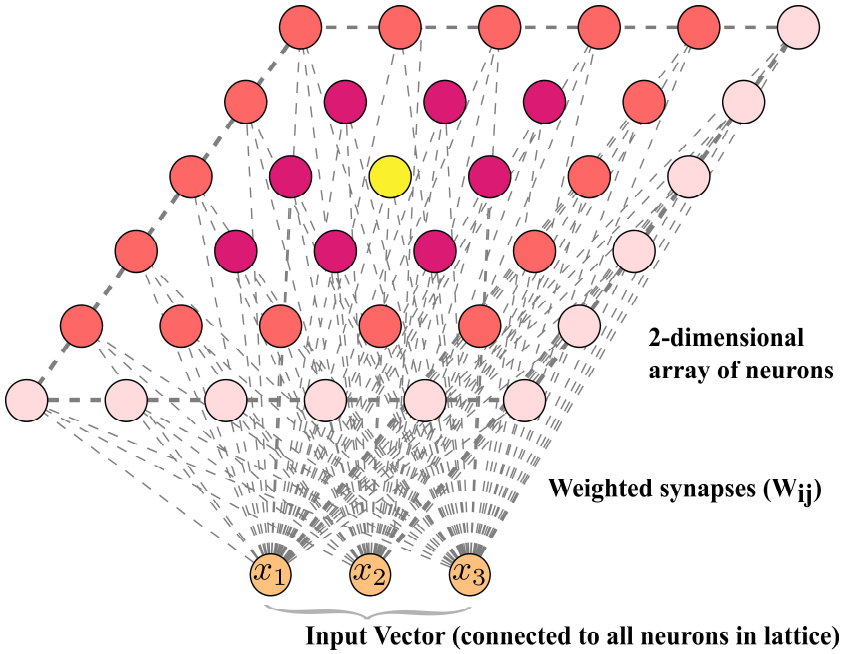

Figure 8. Self-Organizing Maps (SOM).

Root mean squared error (RMSE) (Levinson, 1946) is a quadratic scoring rule which measures the average magnitude of the error. It indicates the accuracy of the probability estimates that are generated by the classification model. Equation 8 represents the RMSE, where *P* is original value or forecast value, *O* represents observed value and *n* is a 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 statistic (K) (Viera & Garrett, 2005) is used in such case as it handles multi-class and imbalanced class like problems. 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.

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

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

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

$$\Pr ecision = \frac{TP}{TP + FP} \quad (6)$$

$$F - measure = \frac{2TP}{2TP + FP + FN} \quad (7)$$

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

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

RESULTS AND DISCUSSION

Various supervised and unsupervised machine learning techniques have been utilized for examining the complexity of CIDDS-001. Ten techniques have been used in this study which includes classification techniques like *k*NN, SVM, DT, ANN, DL, RF, NB,

and clustering techniques like *k*-means, EM, SOMs. All the experiments are 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 is given in scale between 0 and 1 i.e., 0.36 will 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*NN classifier 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. We selected 172839 instances from week 1 traffic data using Reservoir Sampling (Vitter, 1985). Results of *k*NN 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% is achieved with 2NN and minimum 99.3% with 5NN. Similarly, for *k*NN classifier execution on OpenStack traffic data models average accuracy is 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 (Hand, 2009) and FAR also.

Analysis using Support vector machine. John Platt's sequential minimal optimization algorithm (Platt, 1998) is used to train SVM classifier. Firstly, SVM is trained over Week 1 External Server data. Accuracy of 95.3% is achieved in this case with RMSE of 0.320. Performance of SVM classifier on External Server traffic is shown in Table 5. Secondly, SVM is trained over OpenStack Server data.

In this case classifier achieves accuracy of 95.3% with RMSE of 0.272. Considerably good accuracy is achieved with SVM hence a SVM based NIDS can be built. Modified algorithms (Tsang, Kwok, & Cheung, 2005) for SVM training can be used to reduce model building time. 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 3

Performance of *k*NN on External Server data

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

Analysis using Decision trees. DT J48 (C4.5) is analyzed over External Server traffic data. It takes 4.61 seconds to build model for testing. Due to pruning characteristics of J48, it reduces model size significantly decreases the training and testing time. The accuracy of 99.7 % is achieved in the first case.

Table 4

Performance of *k*NN on OpenStack Server data

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

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	
0.860	0.018	0.933	0.860	0.895	unknown	
0.981	0.001	0.968	0.981	0.974	normal	0.953
1.000	0.000	0.999	1.000	1.000	attacker	
0.999	0.000	1.000	0.999	1.000	victim	

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 gives 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 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	
1.000	0.000	1.000	1.000	1.000	normal	0.999
0.998	0.000	0.997	0.998	0.998	attacker	

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	
1.000	0.000	0.999	0.999	0.000	unknown	
1.000	0.000	1.000	1.000	0.000	normal	0.997
1.000	0.000	1.000	1.000	0.000	attacker	
1.000	0.000	1.000	1.000	0.000	victim	

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 is selected as the output class. Firstly, RF is used for External Server data. It takes 46.98 seconds to build the model. The accuracy of 99% is achieved in the first run. Performance of RF classifier on External Server traffic is presented in Table 9.

Secondly, RF is used over OpenStack Server data. 100% Accuracy is second run with model building time of 30.07 seconds. We can observe that tree based algorithms perform well on CIDDS-001 dataset. RMSE is almost negligible in both the cases. Performance of RF classifier on OpenStack Server Traffic is shown in Table 10.

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	0.000	1.000	1.000	1.000	normal	1.000
1.000	0.000	1.000	1.000	1.000	attacker	

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	
1.000	0.000	1.000	1.000	1.000	unknown	
1.000	0.000	1.000	1.000	1.000	normal	0.999
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	0.000	1.000	1.000	1.000	normal	1.000
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. 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 the accuracy of 8.26% with model building time of 413.63 seconds. Hence ANN is not suitable for NIDS development based on CIDDS-001. Table 11, 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	
0.000	0.000	0.000	0.000	0.000	unknown	
0.000	0.000	0.000	0.000	0.000	normal	0.638
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	
0.000	0.000	0.000	0.000	0.000	normal	0.083
0.000	0.000	0.000	0.000	0.000	attacker	

Analysis using Naive bayes classifier. Probabilistic classifiers have been found to be a quite suitable choice for NIDS development. In our study, NB is tested first on External Server traffic data. In the first case, NB gives the accuracy of 87.1% with 0.226 RMSE. NB takes 0.27 to build a model from training data. Secondly, NB is tested over OpenStack traffic. The accuracy of 99% is achieved with 0.074 RMSE. It takes 0.34 seconds to build a model from training data in the second case. Results show the effectiveness of probabilistic classifiers i.e., NB which takes less time in the model building while giving an 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	
0.426	0.001	0.994	0.426	0.597	unknown	
0.977	0.000	1.000	0.977	0.988	normal	0.871
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	
0.989	0.002	1.000	0.989	0.994	normal	0.991
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 is analyzed. In first run External Server traffic data is analyzed using DL. Model from training data is built in 916.07 seconds. The accuracy of 94.05% is achieved with RMSE of 0.139. Table 15 shows the performance of DL based classifier on External Server traffic data.

In second test OpenStack Server traffic is analyzed. 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 large time to build model but 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	
0.874	0.039	0.864	0.874	0.869	unknown	
0.994	0.001	0.980	0.994	0.987	normal	0.941
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	
1.000	0.000	1.000	1.000	1.000	normal	0.999
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 is used for analysis of External Server traffic data. Features named “Flows” and “Tos” are 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. Total 38.1086% instances are correctly clustered by the *k*-means algorithm. Secondly, *k*-means clustering is used over OpenStack Server traffic data. 150000 instances from week 1 traffic data are selected using Reservoir Sampling. Results of second execution Table 18. In this experiment 99.6627 instances are correctly clustered.

Analysis using Expectation-maximization clustering. Firstly, EM algorithm is used to analyse External Server traffic data. We have used the same set of features as used in k-means clustering. In this experiment accuracy of 45.9% is achieved with

model building time of 32.07 seconds. . Results of this experiment are shown in the form of Multi-class confusion matrix and tabulated in Table 19. In second experiment EM is tested over OpenStack Server traffic. In this analysis, the accuracy of 49.3% is

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	

achieved and model is 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.

Analysis using Self-organizing maps. In first experiment, SOM is used to analyse External Server traffic data. SOM takes 601.37 seconds to build a model using training data. After applying testing data it is found that SOM correctly clusters 38.4% test instances. In second experiment SOM is used to analyse OpenStack Server traffic data. In this test, SOM builds a model in 719.59 seconds. 46.3% test instances are correctly clustered in this experiment. Table 21, 22 show confusion matrix for first and second experiment respectively.

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 terms of 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, normalization and data standardization methods. As clustering techniques require input

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	normal	attacker	
Actual Class	victim	13094	0	1142	0.493
	normal	58905	71414	13083	
	attacker	0	14579	622	

data to follow a normal distribution for achieving better accuracy, in CIDDS-001 case features do not follow a normal distribution and hence poor accuracy is achieved. Performance of clustering techniques can be improved by capping, flouring and normalization 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 *k*NN, SVM, DT, RF, NB and DL can be developed.

Figure 10, 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 very less while ANN, DL, SOM methods take a much higher time to build a model. In case of DL, although it takes a large time to build a trained model, DL gives better accuracy once a model gets completely build. Root mean squared error is one of the important factors to analyse the performance of classifiers on a particular dataset. 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 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.

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
		normal	attacker	victim	No class	
Actual Class	victim	0	62	14174	0	0.463
	attacker	50877	20672	21781	50072	
	normal	0	14918	283	0	

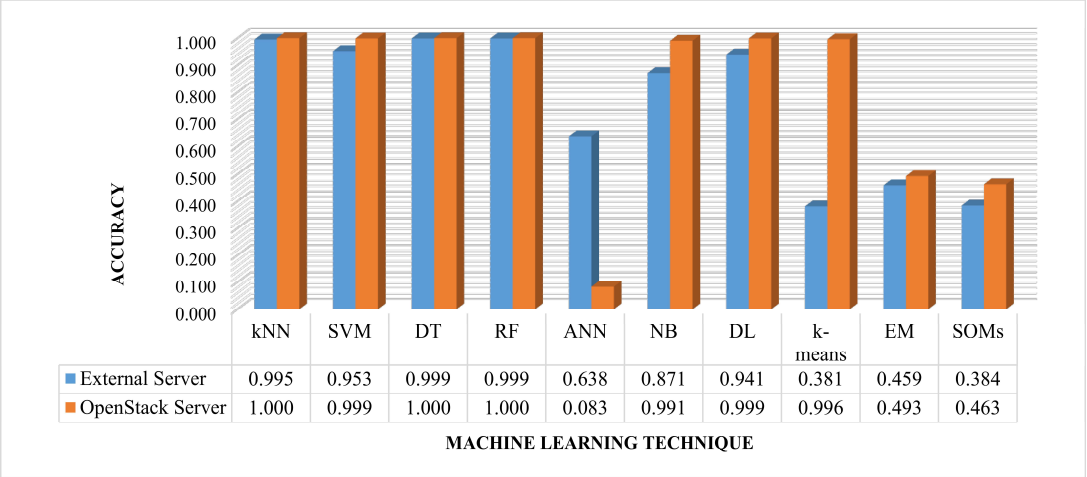

Figure 9. Performance of techniques in terms of accuracy.

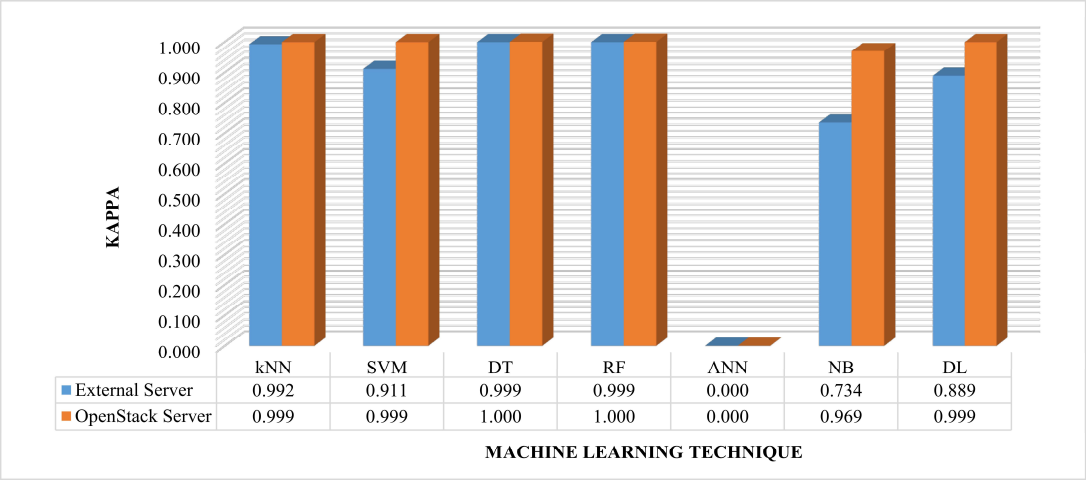

Figure 10. Performance of techniques in terms of kappa statistic.

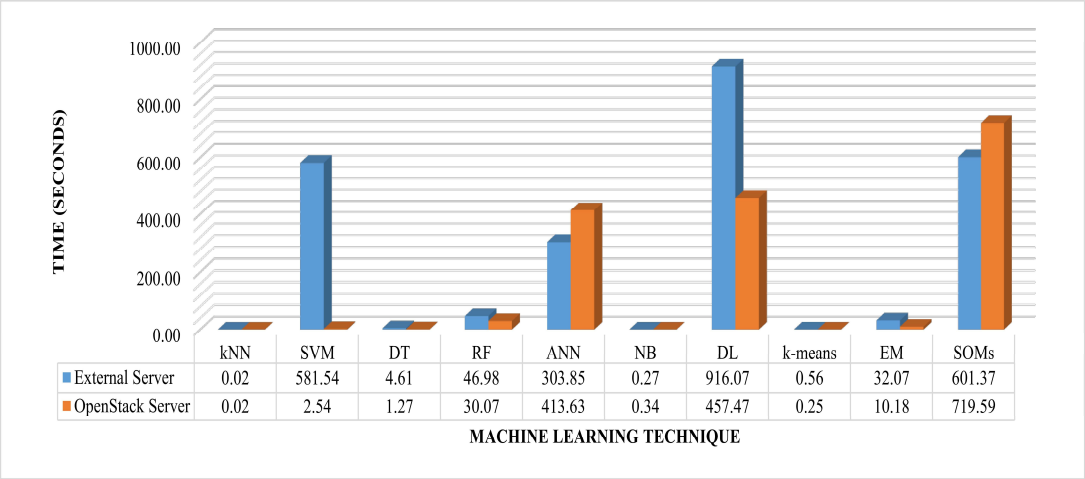

Figure 11. Performance of techniques in terms of model building time.

DL can be a suitable choice if target system (where NIDS is to be deployed i.e. 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.

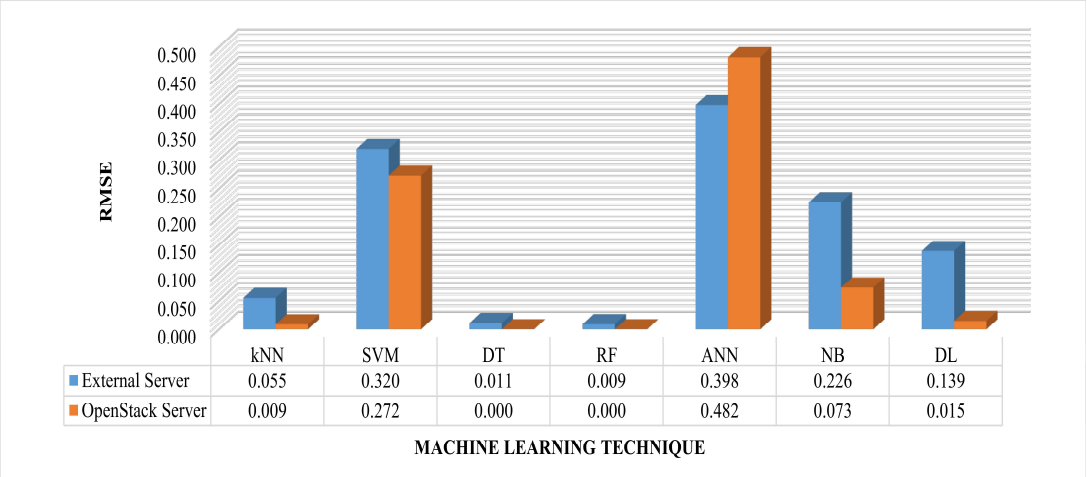

Figure 12. Performance of techniques in terms of root mean squared error.

CONCLUSION

In this paper, the statistical and complexity analysis of CIDDS-001 dataset is presented and discussed. Supervised and unsupervised machine learning techniques are utilized to analyse the complexity of the dataset in terms of 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. In future, we have planned to do an in-depth comparative study of the CIDDS-001 dataset with existing benchmarking datasets.

ACKNOWLEDGEMENT

This paper is the extended version of paper published in "*The 6th International Conference on Smart Computing and Communications*".

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 dataset. (2018). 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. Proceedings of *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) 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 & 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.

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. *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. *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. Retrieved from the Cornell University Library website: https://arxiv.org/abs/1411.7474.

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

KDD 99 dataset. (1999) 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*, 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. *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*. 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. *Proceedings of 4th International Workshop on* Building *Analysis Datasets and Gathering Experience Returns for Security (BADGERS)* (pp. 25-31). Kyoto, Japan: IEEE.

NSL-KDD cup 99 dataset. (1999). 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. *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. Retrieved from the Microsoft website: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/.

Quinlan, J. R. (1996). Bagging, boosting, and C4. 5. *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. *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. *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). 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. *Proceedings of First International Workshop on the Knowledge Discovery and Data Mining, WKDD 2008.* (pp. 10-16). Adelaide, SA, Australia: IEEE.