2021 10th International Conference on Information and Automation for Sustainability (ICIAfS) | 978-1-6654-4429-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIAfS52090.2021.9605814

Comparative Algorithm Analysis for Machine Learning Based Intrusion Detection System

Sharuka Thirimanne
Faculty of Engineering
Sri Lanka Technological Campus
Padukka, Sri Lanka
sharukat@sltc.edu.lk

Lasitha Jayawardana
Faculty of Engineering
Sri Lanka Technological Campus
Padukka, Sri Lanka
lasithaj@sltc.edu.lk

Pushpika Liyanaarachchi
Faculty of Computing
Sri Lanka Technological Campus
Padukka, Sri Lanka
pushpikal@sltc.ac.lk

Lasith Yasakethu
Faculty of Engineering
Sri Lanka Technological Campus
Padukka, Sri Lanka
lasithy@sltc.ac.lk

***Abstract*—In recent years, various types of new intrusions that differ from existing ones have been identified. Moreover, due to the rapid evolution of cyberattacks, machine learning algorithms require updated datasets that comprise the most recent intrusions. The prime objective of this research is to discover the best machine learning algorithm for intrusion detection trained using the NSL-KDD and the UNSW-NB15 datasets and perform a comparative analysis between six machine learning algorithms classified as supervised, semi-supervised, and unsupervised learning. This study revealed that the performance of supervised and semi-supervised machine learning algorithms outperformed unsupervised machine learning algorithms for both datasets and concluded that Support Vector Machines (SVM) and Deep Neural Network (DNN) perform better for NSL-KDD and UNSW-NB15, respectively.**

***Index Terms*—Intrusion Detection, Supervised Learning, Semi-supervised Learning, Unsupervised Learning**

I. Introduction

Intrusion detection has become a significant concern in corporate and personal networks due to various types of threats. The Intrusion Detection System (IDS) refrains the network from external attacks. In addition, IDSs can identify variations between forms of malicious network communications and computer systems, while conventional firewalls are incapable of doing so. The functions of this system include the supervision of policy violations, malicious activities, and unauthorised access. Machine Learning (ML) algorithms are capable of identifying intrusions upon the arrival of data packets. It is done by utilising the statistical modelling concept for learning the past data patterns of the packets, which will be used later to identify new data patterns.

Most comparative analysis researches that are currently available were done based on the KDD-99 dataset and several based on the NSL-KDD dataset; however, out of all those researches, the majority has been performed using the Weka tool. Nevertheless, this research's uniqueness is that it analyses six ML algorithms based on supervised, semi-supervised, and unsupervised learning to tackle the algorithm that performs best when ingesting unseen data. Moreover, each ML algorithm is trained using the NSL-KDD and the UNSW-NB15 datasets. The NSL-KDD dataset was created using the KDD-99 dataset to overcome the inherent problems such as redundant and duplicate records [1]. The UNSW-NB15 [2] dataset, which consists of nine modern forms of attack, was published in 2015 because current threats are not represented in the KDD-99 dataset[3]. The objective of this research is to comparatively analyse the ML algorithm's performance on each dataset using numerous performance metrics and find the algorithm with the best performance for each dataset.

The research is based on binary classification, and it refers to identifying whether or not a network state represents an intrusion. The research compares the performance metrics of Deep Neural Network (DNN), Support Vector Machines (SVM), K-Nearest Neighbours (KNN), One-class SVM (OCSVM), K-Means, and Expectation-Maximization (EM) ML algorithms, which were trained using the NSL-KDD and the UNSW-NB15 datasets. The confusion matrix, precision, recall, F1-score, and ROC are used as the evaluation metrics to compare each ML algorithm.

The contributions of this paper are explained as follows. This ML algorithm comparison is carried out to find the ML algorithm that outweighs the performances of the other algorithms for both the NSL-KDD and the UNSW-NB15 datasets. Moreover, this conveys a descriptive outlook on how each dataset performs on different ML algorithms. Since precise comparative details are available, suitable algorithm selection will be more accessible in the future when developing a real-time IDS. The performance metrics have shown that SVM and DNN algorithms have outweighed other algorithms in terms of performance for the NSL-KDD and the UNSW-NB15 datasets, accordingly.

The structure of this paper is organized as follows. Related Works are mentioned in Section II, and Section III includes

the problem statement. Section IV includes the research's system model. Subsequently, the theoretical aspect of the ML algorithms are described in Section V. Section VI describes the evaluation criteria, and the methodologies are discussed in Section VII. Section VIII illustrates the results. Section IX and X include the discussion and the conclusions accordingly.

II. Related Work

NSL-KDD dataset has been used for binary classification and multi class classification for the past few years for intrusion detection. Since the UNSW-NB15 dataset was launched recently, few publications have been launched compared to publications launched using NSL-KDD. A team of researchers from the University of Professionals and Military Institute of Science & Technology in Dhaka, Bangladesh, have performed the research on Network Intrusion Detection using Supervised ML with NSL-KDD dataset. Within that literature, they have analyzed SVM and Artificial Neural Network (ANN) ML techniques. They were able to achieve 82.34% SVM model accuracy and 94.02% ANN model accuracy using the Weka tool [4].

Researchers from George Mason University, USA, have performed research on IoT Network Intrusion Detection using NSL-KDD and KDD Cup 99 datasets. Comparative analysis has been done between various ML classifiers: The Naïve Bayes (NB) Classifier, SVM, Random Forests, and ANN-based on type classification and binary classification methods. For all the classifiers, they were able to obtain more than 90% for precision, recall, and F1-score [5].

Performances of ML algorithms based on binary classification for network intrusion detection analyzed by a team of researchers from UniMAP, Perlis, and UniSZA, Terengganu. Using the Weka tool, three Bayesian family machine learning algorithms were tested using UNSW-NB15: Average One Dependence Estimator (AODE), Bayesian Network (BN), and NB. They achieved an average of 94.5% for True Positive (TP) rate, 92.9% for precisian, 94.5% for recall, and 5.4% for False Positive (FP) rate in AODE. An average of 91% for TP rate, 92% for precision, 91% for recall, and 9% for FP rate in BN. An average of 80.3% for TP rate, 76.4% for precisian, 80.3% for recall, and 19.6% for FP rate in NB [6].

Nour Moustafa and Jill Slay have performed a statistical analysis of the UNSW-NB15 dataset and comparative evaluation of network anomaly detection systems using UNSW-NB15 and KDD99 datasets. Comparative performance analysis has been done between five ML techniques: The Decision Tree (DT), Logistic Regression (LR), NB, ANN, and EM. They were able to achieve better accuracy when using KDD99 compared to the UNSW-NB15 dataset. For the above five techniques, when using KDD99 dataset: 92.3%, 92.75%, 95%, 97.04%, 78.06% were obtained accordingly and when using UNSW-NB15 dataset: 85.56%, 83.15%, 82.07%, 81.34%, 78.47% were achieved accordingly [2].

III. Problem Statement

There are numerous methods, such as firewalls, which have been designed to preserve the networks and systems from attacks. Firewalls are incapable of identifying intrusions; however, ML-based IDS can tackle the intrusions by analysing the inbound data packets. Selecting the best ML algorithm for real-time IDS has been an obstacle due to the lack of comparisons available. The objective of this research is to provide a descriptive comparison to address the above challenge.

IV. System Model

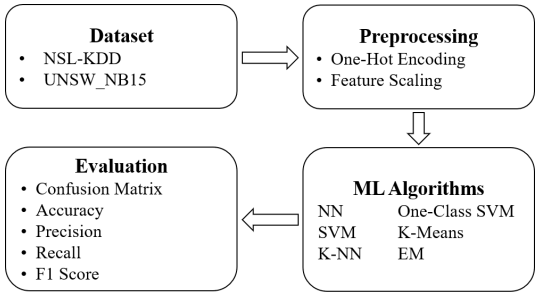

Fig. 1. Diagram of the system model

The above figure 1 describes the system model of this research, which clearly illustrates the entire process's flow—using the NSL-KDD and the UNSW-NB15 datasets. ML algorithms are used for evaluation, and performance metrics are used for comparative analysis.

V. Machine Learning Algorithms

A. Neural Netwok (NN)

Human biology influenced the ANN concept and how neurons of the human brain function together to comprehend signals from human senses. Neural Network (NN) models can act as nonlinear discriminating functions since NN can form any decision boundary classification in feature space [7]. NNs are used for regressions problems as well as classification problems. In classification problems, each element of the feature vector is connected to a single input node. This NN model contains one output node and multiple hidden layers with multiple nodes within each hidden layer since this research is related to binary classification. All the hidden layers in this network were activated using the ReLU activation function [5].

$$ReLU(x) = max(0, x) \tag{1}$$

Equation 1 depicts the ReLU activation function. It is a piecewise linear function, and whenever the input is positive, it output the input directly; otherwise, the output will be zero. The nodes in which this activation function is implemented are referred to as a rectified linear activation unit.

The Sigmoid function, which is depitcted below, is used as the activation function of the output node.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

According to the Equation 2, the sigmoid function exits between 0 and 1. Consequently, this activation function is beneficial when probability prediction is being carried out.

B. Support Vector Machines (SVM)

SVM is the most common and accessible approach for binary classification for ML tasks [7]. SVM maps linear algorithms into non-linear space. The function called the 'kernel' is used for this mapping. Kernel functions, such as linear, polynomial, and radial basis functions, are used to divide the feature space by creating a hyperplane. Support vectors along that function's surface can be identified at the time of classifier training using the kernel functions. Each example of this approach is labelled with several training examples belonging to one of two categories. In this, 'rbf' and 'poly' kernels are used to identify which kernel performs best for both datasets. The hyperparameters related to SVM, such as C, gamma, and degree, were assigned after performing hyperparameter tuning using the grid search method.

C. K-Nearest Neighbors (K-NN)

K-NN algorithm is often used to solve classification problems. Based on the attributes and the training data K-NN algorithm classifies new objects. Based on the Euclidian equation, K-NN algorithms work by calculating the closest distance to a new object [8]. The value K is the number of neighbours which the algorithm will be searched, and based on their output, the prediction for the newly arrived data will be given. K-NN algorithm is known as a lazy method since it only makes predictions by memorizing the dataset [9].

D. One-Class Support Vector Machines (OCSVM)

OCSVM algorithm is a semi-supervised ML algorithm since it uses the 'normal' class's training data without the target label. This algorithm is suitable when the relevant dataset only has examples from the 'normal' class. Once the algorithm is trained using the 'normal' class, the model can classify whether the new data are normal or abnormal (outliers). The OCSVM uses the hyperparameter 'nu' instead of hyperparameters such as C in regular SVM, which is used to monitor the support vectors' sensitivity and be calibrated to the approximate ratio of outliers in the dataset.

E. K-Means

K-means is a partitional clustering algorithm that partitions the data into clusters of K's, and the user determines the K value. There is a cluster core for each cluster, called a centroid. The K-means algorithm begins with the first group of randomly chosen centroids, which are used for each cluster as the starting points, and then performs iterative calculations to optimize the centroid positions. When the centroids have stabilized, and there is no change in their values because clustering has been optimal or the defined number of iterations has been completed, clustering stops creating and optimizing clusters [10].

F. Expectation Maximization (EM)

A mixture model involving a set of Gaussian (Normal) probability distributions is the Gaussian Mixture Model (GMM), which involves calculating the mean and standard deviation parameters for each one. A practical method to use to evaluate the distribution parameters is the EM algorithm. In the presence of latent variables, the EM Algorithm is an approach to find Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP). Latent variables cannot be detected explicitly but are derived from the variables observed instead. An iterative method follows EM, and that requires two steps. Firstly, it calculates the missing or latent variables, which is called the estimation step or E-step. The second stage aims to optimize the distribution parameters using maximum likelihood to understand the data, called the maximization-step or M-step [11].

VI. Evaluation Criteria

Each ML classifier includes two trials for each dataset, and the performance is evaluated using test set accuracy, confusion matrix, precision, recall, F1 score, and the Receiver Operating Characteristics (ROC). Precision, recall, and f1-score are dependent on the number of predicted false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN). These are well-known metrics used for evaluating binary classification problems.

1. TP : The model predicts 1 and the actual class is 1.
2. FP : The model predicts 1 and the actual class is 0.
3. TN : The model predicts 0 and the actual class is 0.
4. FN : The model predicts 0 and the actual class is 1.

The classification metrics are as follows.

$$Precision(P) = \frac{TP}{TP + FP} \tag{3}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - score = 2.\frac{P.R}{P + R} \tag{5}$$

Equations 3-5 depict the equations associated with precision, recall, and F1-score. Precisions quantifies the number of positive predictions that are actually positive. Moreover, recall quantifies the number of positive predictions made utilizing the positive examples in the dataset. The weighted average between precision and recall is known as the F1-score, and it can be used to seek a balance between precision and recall [12]. The performance of the model is better when the F1-score is higher. ROC is utilized as a performance measurement for classification problems. Moreover, the ROC and Area Under Curve (AUC) tell the model's capability distinguishing between classes. Higher the AUC, better the accuracy of the prediction. The above performance metrics are generally more effective than only comparing accuracy, especially when the class distribution is skewed.

VII. Methodology

A. Data Preprocessing

Data pre-processing is an essential procedure, which prepares the data before ingesting them into the ML model, and it consists of several essential tasks: dropping duplicated data, handling missing values, categorical data encoding and feature scaling. Both the datasets have two separate datasets for training and testing purposes, and those have undergone the above procedures before training the algorithms. Firstly, the attribute 'id' is dropped from both datasets since duplicated records cannot be identified due to the unique numbering in that attribute. Afterwards, using the in-built functions of the Pandas library, duplicated data were located and dropped from both training and testing datasets in the NSL-KDD and the UNSW-NB15 datasets. Table I displays the number of duplicates in each dataset for both training and test datasets.

TABLE I
Number of duplicates in each dataset

	NSL-KDD	UNSW-NB15
Training set duplicates	9	67,601
Test set duplicates	3	26,387

Subsequently, the One-Hot Encoding (OHE) technique was utilized to transform categorical data into numerical values since the ML model usually performs optimally when numerical values are being used. OHE was used since the categorical data in the NSL-KDD dataset and the UNSW-NB15 dataset is nominal; an ordered numerical list will be created once integer encoding is being used, which will mislead the ML algorithms creating unnecessary importance considering the magnitude of the assigned values. In addition, the attribute which included the attack types in the UNSW-NB15 dataset was removed because it is unnecessary for binary classification problems. The drawback involved with OHE is, it generates a new column per category, known as the "curse of dimensionality". Several training dataset categories were not in the testing dataset, leading to a dimensionality mismatch after encoding both datasets. In order to prevent the curse of dimensionality and dimensionality mismatch, categories with less frequency are grouped into a single category. Table II and Table III reveal the number of categories before and after grouping in each categorical feature in both datasets.

TABLE II
NSL-KDD, before and after category reduction

Feature Name	Category count before	Category count after
protocol_type	3	3
service	70	25
flag	11	11
Total	**84**	**39**

TABLE III
UNSW-NB15, before and after category reduction

Feature Name	Category count before	Category count after
proto	133	8
service	13	13
state	9	5
Total	**155**	**26**

Eventually, OHE is performed for the features which have categorical data after the category reduction process is being carried out. The final step of the data pre-processing is the feature scaling, and it is done to convert the numerical values of the entire dataset to a standard scale. The feature scaling process is carried out using 'MinMaxScaler' and 'StandardScaler' functions in Scikit-Learn. The feature scaling method for each ML algorithm is selected based on the performance metrics after comparative analysis.

$$X_{normalized} = \frac{X - min(X)}{max(X) - min(X)} \tag{6}$$

$$X_{standardized} = \frac{X - mean(X)}{standard\ deviation(X)} \tag{7}$$

The normalization equation is depicted in Equation 6—it is a scaling technique utilized to rescale the features to the range between 0 and 1. Equation 7 is the standardization equation, a scaling mechanism capable of rescaling the attributes to zero mean and the distribution with unit standard deviation.

B. Machine Learning Algorithm Implementation

The functions and classifiers used to build all the ML algorithms were taken from Scikit-Learn, Keras, and TensorFlow libraries. Moreover, Python was the only programming language used to build ML algorithms. The Scikit-Learn function called 'GridSeachCV' was used for the majority of the ML algorithms when tuning the hyperparameters.

*1) **DNN**:* The DNN algorithm was executed several times under different parameters using a k-fold cross-validation mechanism. Moreover, different hyperparameter values for learning rate, regularization coefficient, mini-batch size, number of hidden layers, number of neurons per hidden layer, optimizer (SGD, RMSprop, Adam)and activation function (Sigmoid, Relu, SoftMax, LeakyRelu) were tested to enhance the performance. DNN algorithms were trained for 200 epochs together with mini-batches to reduce the training period. In addition, the overfitting was prevented using regularization and early stopping methods.

*2) **SVM**:* The SVM algorithm was created based on two different kernels known as the 'rbf' and the 'poly' kernels. Two significant hyperparameters were associated with both kernels, namely 'C' and 'gamma'. Both kernels were tested using the k-fold cross-validation mechanism assigning a range of different values for both hyperparameters. Although 'gamma' values vary between 0 - 1, values between 0 - 0.5 were assigned, since upon the increment of 'gamma' the model tends to overfit.

*3) **KNN**:* The hyperparameter known as the number of neighbours (K) acts as the major contributing factor for the performance. In order to find the optimal K value for a K-NN classifier based on a particular dataset, the algorithm was executed several times, assigning different values to K. However, the gradual decrement in the 'k' value decreased the stability of the predictions.

*4) **OCSVM**:* The OCSVM algorithm was trained using a pipeline containing OHE for categorical data encoding, 'StandardScaler' for feature scaling, and the OCSVM classifier. Moreover, the inbuilt functions of the Scikit-Learn library were used for the above process. Once the training data is being sent into the ML pipeline, data transformation will be initiated, and subsequently, the ML model will be trained. In addition, testing the system using a cross-validation set and the unseen testing dataset was done using the same pipeline. Furthermore, testing the trained OCSVM model for binary classification using the test dataset requires changing the target labels from 0 and 1 to +1 and -1 accordingly.

*5) **K-Means and EM**:* In unsupervised learning, since unlabeled data is fed into the algorithms, there is no explicit method to determine whether the resulting clusters are normal or intrusions. Thus, clusters were labelled implicitly by mapping two randomly labelled clusters given by K-Mean and EM to 1 and 0, and vice versa for the NSL-KDD dataset. Those algorithms produce promising results when the number of clusters is two. The resultant will have two Y label sets and using those sets, accuracy is checked against the training set Y label. Then the mapping, which gives maximum accuracy, is used to evaluate training and testing set scores. Since the UNSW-NB15 dataset contains one normal class and nine intrusions [2], the 10 clusters were mapped into two classes to perform binary classification. That was done by mapping 0 to one class and the remaining nine clusters to 1 using the one-vs-all method to identify the normal cluster. Out of ten mappings, the mapping with maximum accuracy was used to evaluate the performance metrics. In addition, when Principal Component Analysis (PCA) was used, the K-Means and the EM performed well for the NSL-KDD dataset.

VIII. Results

Table IV displays the test set accuracies for each ML algorithm for the NSL-KDD and the UNSW-NB15 datasets.

TABLE IV
TEST SET ACCURACIES OF EACH ML ALGORITHM

			Test Accuracy
DNN	NSL-KDD		0.7847
	UNSW-NB15		0.8026
SVM	NSL-KDD	RBF	0.8025
		POLY	0.8076
	UNSW NB15	RBF	0.7573
		POLY	0.7581
K-NN	NSL-KDD		0.7886
	UNSW-NB15		0.7934
One-Class SVM	NSL-KDD		0.8414
	UNSW-NB15		0.6019
K-Means	NSL-KDD		0.7375
	UNSW-NB15		0.5900
EM	NSL-KDD		0.7418
	UNSW-NB15		0.6789

Table V displays the precision, recall, f1-score and ROC values of each ML algorithms for the NSL-KDD and the UNSW-NB15 datasets.

TABLE V
TESTING SET PRECISION, RECALL, F1-SCORE AND ROC

			Precision	Recall	F1-score	ROC
DNN	NSL-KDD		0.9237	0.6775	0.7817	0.80
	UNSW-NB15		0.6635	0.9330	0.7755	0.83
SVM	NSL-KDD	RBF	0.9276	0.7081	0.8031	0.82
		POLY	0.9248	0.7203	0.8098	0.82
	UNSW NB15	RBF	0.6034	0.9804	0.7470	0.80
		POLY	0.6054	0.9717	0.7460	0.80
K-NN	NSL-KDD		0.9587	0.6566	0.7794	0.81
	UNSW-NB15		0.6508	0.9382	0.7685	0.82
One-Class SVM	NSL-KDD		0.9739	0.6497	0.7794	0.82
	UNSW-NB15		0.9188	0.4088	0.5658	0.67
K-Means	NSL-KDD		0.9847	0.5473	0.7036	0.76
	UNSW-NB15		0.4865	0.9896	0.6523	0.66
EM	NSL-KDD		0.9837	0.5557	0.7102	0.76
	UNSW-NB15		0.5478	0.9953	0.7067	0.73

Fig. 2-8 displays the normalized confusion matrices of DNN, K-NN, SVM, OCSVM, K-Means and EM.

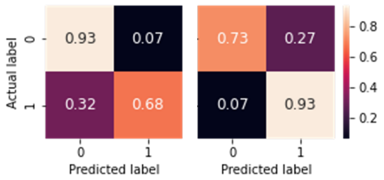

Fig. 2. Confusion Matrix of DNN for NSL-KDD and UNSW-NB15.

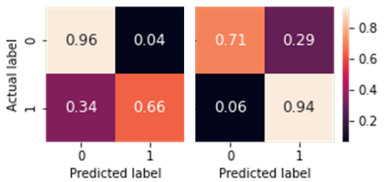

Fig. 3. Confusion Matrix of KNN for NSL-KDD and UNSW-NB15.

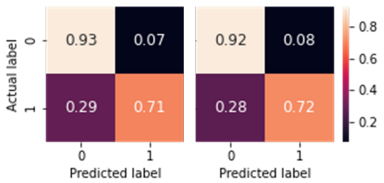

Fig. 4. Confusion Matrix of SVM RBF and POLY kernels for NSL-KDD.

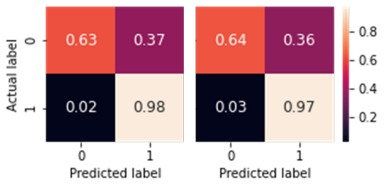

Fig. 5. Confusion Matrix of SVM RBF and POLY kernels for UNSW-NB15.

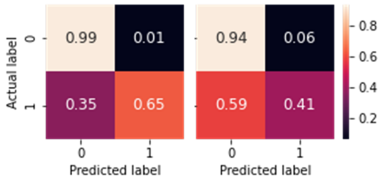

Fig. 6. Confusion Matrix of OCSVM for NSL-KDD and UNSW-NB15.

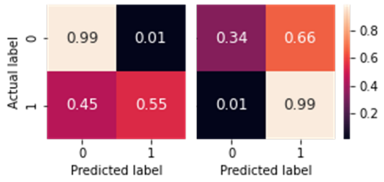

Fig. 7. Confusion Matrix of K-Mean for NSL-KDD and UNSW-NB15.

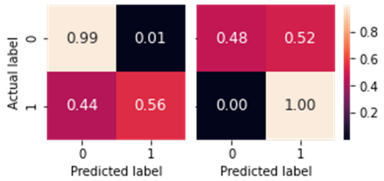

Fig. 8. Confusion Matrix of EM for NSL-KDD and UNSW-NB15.

IX. Discussion

The results depicted in Table IV, Table V and Fig.2-8 were used to identify the ML algorithm with the best performance for each dataset. All the performance metrics are taken using the testing set of each dataset. Furthermore, the performance metrics results: accuracy, precision, recall, f1-score, ROC and confusion matrix (CM) of the training set are higher than the testing results. Overall, even though the UNSW-NB15 dataset contains modern intrusions, according to Table IV, Table V and Fig. 2-8, the majority of the ML algorithms, which performed well, were trained using the NSL-KDD dataset.

According to the CMs related to the NSL-KDD dataset in fig. 2-8, the number of TNs was higher while TPs are considerably lower. In general, the models have predicted the 'normal' data accurately, while misclassifications have occurred when predicting intrusions. Nevertheless, the CMs of the UNSW-NB15 reveal higher TPs than TNs except in the OCSVM. Thus, the ML models trained using the UNSW-NB15 dataset have predicted the intrusions precisely. On the other hand, over 50% of misclassified predictions were observed in the OCSVM and the EM, which were trained using the UNSW-NB15 dataset.

Table IV reveals that the highest testing dataset accuracy is attached to the OCSVM (NSL-KDD), which is 84%. However, the K-Means (UNSW-NB15) model has the lowest accuracy. In general, the majority of the algorithms have passed the accuracy level of 70% except for OCSVM, K-Means and EM, which were trained using the UNSW-NB15 dataset. Overall, most ML models' performance trained using the NSL-KDD dataset has outweighed the ML model trained using the UNSW-NB15, except in the DNN and the KNN.

Based on the results of Table V, the highest values for precision, recall, f1-score and ROC were observed in OCSVM(NSL-KDD), K-Means(UNSW-NB15), SVM-POLY(NSL-KDD) and DNN(UNSW-NB15), accordingly. Overall, the 'poly' kernel-based SVM (NSL-KDD) algorithm performed best with higher values of 0.9248, 0.7203, 0.8098, 0.82 for precision, recall, f1-score, and ROC accordingly. In addition, for the UNSW-NB15, the DNN model has shown promising results by achieving 0.6635, 0.9330, 0.7755, 0.83 for precision, recall, f1-score, and ROC accordingly.

X. Conclusion

This research presents a descriptive analysis of six ML algorithms for the NSL-KDD and the UNSW-NB15 datasets. In conclusion, the NSL-KDD dataset gives promising results than the UNSW-NB15 dataset. These results reveal that supervised and semi-supervised algorithms perform well compared to unsupervised algorithms. While supervised learning algorithms have been more effective than the other two groups, a rich dataset containing all forms of intrusion to train the ML model is needed. However, the OCSVM ML algorithm is the best choice as it only needs the 'normal' data, and it performs well using the anomaly detection technique. Finally, the SVM algorithm obtained the highest result for the NSL-KDD dataset, and for the UNSW-NB15 data set, DNN has achieved the highest results.

XI. Future Works

A DNN based real-time IDS will be built using the UNSW-NB15 dataset since it contains modern intrusion types. Features will be extracted using the packet sniffing method and fed into the trained ML model through a pipeline. Detection of an intrusion will be reported to the user.

References

[1] R. Thomas and D. Pavithran, "A Survey of Intrusion Detection Models based on NSL-KDD Data Set," 2018 Fifth HCT Information Technology Trends (ITT), 2018, pp. 286-291, doi: 10.1109/CTIT.2018.8649498.

[2] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.

[3] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), 2017, pp. 1881-1886, doi: 10.1109/ISIE.2017.8001537.

[4] K. A. Taher, B. M. Y. Jisan and M. M. Rahman, "Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection," 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), pp. 643-646, 2019.

[5] S. Sapre, P. Ahmadi and K. Islam, "A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms," 2019.

[6] M. Nawir, A. Amir, O. B. Lynn, N. Yaakob and R. Ahmad, "Performances of Machine Learning Algorithms for Binary Classification of Network Anomaly Detection System," Journal of Physics, 2017.

[7] Z. C. Hua TANG, "Machine Learning-based Intrusion Detection Algorithms," Journal of Computational Information Systems5, 2009.

[8] Okfalisa, Mustakim, I. Gazalba and N. G. I. Reza, "Comparative Analysis of K-Nearest Neighbor and Modified K-Nearest Neighbor Algorithm for Data Classification," 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 2017.

[9] K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[10] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 881-892, July 2002, doi: 10.1109/TPAMI.2002.1017616.

[11] T. K. Moon, "The expectation-maximization algorithm," in IEEE Signal Processing Magazine, vol. 13, no. 6, pp. 47-60, Nov. 1996, doi: 10.1109/79.543975.

[12] Y. Zhang, P. Ma and Q. Gao, "Multiple Classification Models Based Student's Phobia Prediction Study," 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019, pp. 526-531, doi: 10.1109/IRC.2019.00109.