2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON) | 978-1-6654-7978-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/NIGERCON54645.2022.9803175

Comparative Analysis of Machine Learning Models in Computer Network Intrusion Detection

Edosa Osa
Department of Electrical and Electronic Engineering
Faculty of Engineering University of Benin
Benin City, Nigeria
edosa.osa@uniben.edu
eddyedos@yahoo.com

Ogodo Efevberha Oghenevbaire
Department of Mathematics and Computer Science
College of Natural and Applied Sciences
Western Delta University,
Oghara, Nigeria.
efevbaire@gmail.com

***Abstract*— Network security is a major concern of the modern era. With the rapid development and massive usage of computer networks over the past decade, the vulnerabilities of network systems continue to be of significant concern. Intrusion detection systems are used to monitor networks and identify unauthorized access or malicious traffic over secured networks. The application of machine learning algorithms to the intrusion detection domain could enhance such systems. This paper presents a comparative analysis of selected machine learning algorithms for network intrusion detection. The CICIDS 2017 dataset provided the necessary dataset for training the models. Results show that of all six considered, Decision Tree classifier was the overall best.**

Keywords—Machine Learning, intrusion, accuracy, traffic, algorithm

I. Introduction

The design of secure computer networks has been a subject of research for years. Unauthorized network traffic tend to compromise the integrity, confidentiality, and availability of any information system or the information itself can turn out to be a security attack or an intrusion. Each day new kinds of attack are discovered by cyber industries. One solution to this problem is by implementing an Intrusion Detection System (IDS) in the network. An intrusion detection system (IDS) is set up to analyze the system and network activity for unauthorized and illicit activity. Intrusion Detection System (IDS) is any hardware, software, or a combination of both that monitors a system or a network of systems against any malicious activity. It has three major functions which include monitoring, detecting and generating an alert.

Machine learning (ML) is an area of artificial intelligence (AI) and computer science knowledge that focuses on applying data and algorithms to imitate human learning patterns, with the aim of steadily improving the accuracy of a system. Various algorithms are trained to generate classifications or predictions using statistical approaches. Author in [1] divides the learning process of a machine learning algorithm into three main parts namely: A Prediction or Classification Process, an Error Function and a Model Optimization Process. Machine learning models are also defined by the presence or absence of human influence on raw data, whether a reward is offered, specific feedback is given or labels are utilized. According to [2], the general classification for machine learning models are as follows: Supervised learning, Unsupervised learning, Semi Supervised learning and Reinforcement learning. However, there is a newer subset of machine learning known as deep learning, which learns autonomously from datasets without the explicit use of human rules or understanding. It requires the analysis of large amounts of raw data, and the more data received by such model, the better the prediction model performs. In recent years, Machine Learning based Intrusion Detection systems have been giving high accuracy and good detection of novel network attacks.

II. Literature Review

Based on the mode of Intrusion, Intrusion Detection Systems could be categorized into Signature-based Intrusion Detection Systems OR Anomaly-based Intrusion Detection Systems. Our paper considers an Anomaly-based Intrusion Detection System. An anomaly-based Intrusion Detection System examines ongoing traffic activity, transactions and behaviors in order to identify intrusions by detecting certain anomalies. It operates on the principle that attack behavior differs enough from normal behavior and hence can be detected via catalogue and identification of the differences involved. A useful ingredient for created machine learning solutions for intrusion detection is a dataset. Several datasets exist such as KDD-99, NSL-KDD, DARPA 1998, DARPA 2000, CICIDS 2017, etc. The Canadian Institute for Cybersecurity at the University of New Brunswick developed CICIDS 2017 (Intrusion Detection Evaluation Dataset). This dataset is made up of a 5-day data stream (3rd–7th July 2017) on a network formed by machines running current operating systems like Windows Vista / 7 / 8.1 / 10, Mac, Ubuntu 12/16 and Kali Linux. Information about the CICIDS-2017 dataset can be found at the official website [3]. Various authors have carried out comparative analysis of machine learning algorithms in intrusion detection systems design. Author in [4] implemented the DARPA-Lincoln dataset in evaluating and comparing the performance of four ML classifiers with respect to four attack categories; DoS, R2L, Probe, and U2R. Results show that J48 classifier outperformed the other three classifiers namely IBK, MLP, and NB in prediction accuracy. The authors in [5] used the NSL-KDD dataset to measure the effectiveness of three ML classifiers in detecting the anomalies in network traffic. Their results show that J48 classifier outperforms SVM and NB classifiers regarding accuracy. The work in [6] was carried out as an empirical study to evaluate a comprehensive collection of ML classifiers on the KDD'99 dataset. The work involved detection of attacks from the four classes. Authors in [7] compared different classification techniques such as naive Bayes, OneR, PART, RBF and J48 network algorithms to detect and classify network traffic into normal and abnormal. The authors implemented the NSL-KDD dataset.

Unlike the work of authors above, our paper focuses on evaluating and comparing the performances of some well-known, supervised ML classifiers over the CICIDS 2017 Dataset. This dataset was chosen above numerous others since it is relatively current and has a larger protocol and attack pool. Intrusion detection is considered along the Feature selection dimension.

III. Methodology

This paper presents a binary classification subject of network traffic as malicious or benign hence supervised machine learning approach was employed. The work was carried out on a computer system with CPU: Intel Core i5, 10th GEN, GPU: Nvidia GeForce GTX 1650, RAM: 8 GB RAM and OS: UBUNTU 20.04 LTS. Python 3.8 programming language was used for writing the necessary codes. Python can also can be used in conjunction with a variety of libraries to create machine learning applications. Libraries such as *Numpy, Matplotlib* and *Pandas* were utilized for general data manipulation. The procedure for the system design is described in figure 1:

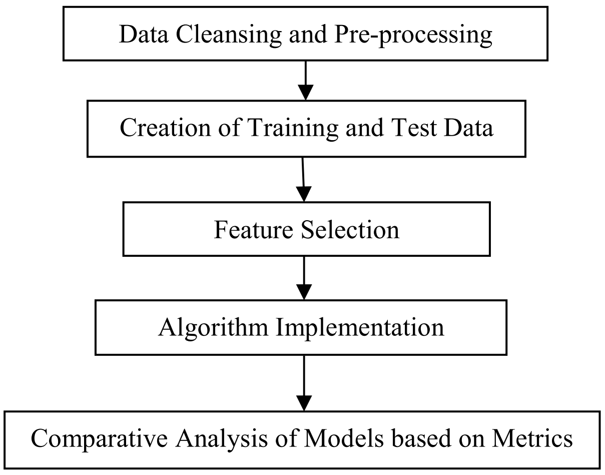
Data Cleansing and Pre-processing
Creation of Training and Test Data
Feature Selection
Algorithm Implementation
Comparative Analysis of Models based on Metrics

Fig. 1. Methodology

A. Data Cleansing

In order to make the training process seamless, it was important to carry out some investigation on the dataset and make some necessary changes before further actions are implemented. The dataset file contains 3119345 network stream records. Table 1 shows the distribution of these network stream records. After taking a view of the data, 288602 of the records contain null and incorrect values and were therefore deleted.

TABLE I. CICIDS-2017 DATASET

TARGET NAME	**COUNT**
Benign	2359289
Faulty	288602
DoS Hulk	231073
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – SQL Injection	21
Heartbleed	11
TOTAL	**3119345**

The columns that make up the features presented another inaccuracy in the dataset. The dataset file has 86 columns that define flow parameters like Flow ID, Source IP, and Source Port, among others. The Fwd Header Length feature, on the other hand, was written twice (it defines the forward direction data flow for total bytes used) (41st and 62nd columns). The recurring column had to be deleted to fix this problem (column 62). "Flow Bytes/s" and "Flow Packets/s" features include the values "Infinity" and "NaN" alongside numerical values, which can be modified to -1 and 0 respectively to make them suitable for machine learning algorithms.

B. Creation of Training and Test Data

The CICIDS2017 dataset does not contain separate training and test data but rather a single unbundled dataset. The train test split [8] Sklearn tool was applied for this process. Generally, a 20 percent test, 80 percent training data split is favored [9], and this ratio was likewise preferred in our case.

C. Feature Selection

Not all the 79 features were necessary for the training pipeline. In order to measure and sort features with high importance factor, the Random Forest Regressor [10] class of Sklearn was used. A decision-forest is created by this weight of relevance based on how relevant it is in the creation of the decision tree. After the process is complete, these feature importance weights are compared and ordered [11]. By grouping all the attack types under a single label, **"attack,"** the Random Forest Regressor technique was applied on the entire dataset. As a result, only the attack and benign tags were present in the data file. This operation resulted in a feature list for the top 20 most important features. The importance weight was plotted against feature importance as shown in figure 2.

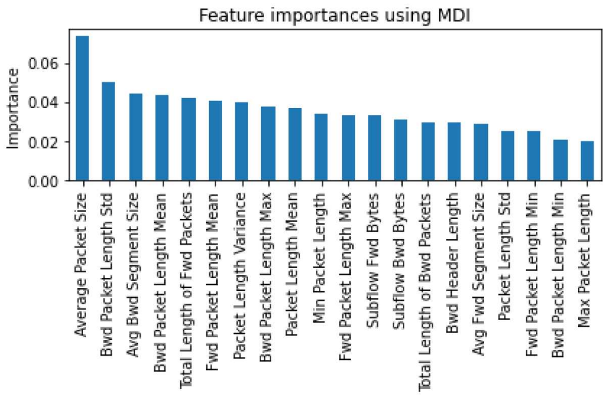

Fig. 2. Importance versus Feature Graph.

It was further decided to drop ten more features. The ten features further selected for training the machine learning models by the Random Forest Regressor are described by the confusion matrix in figure 3.

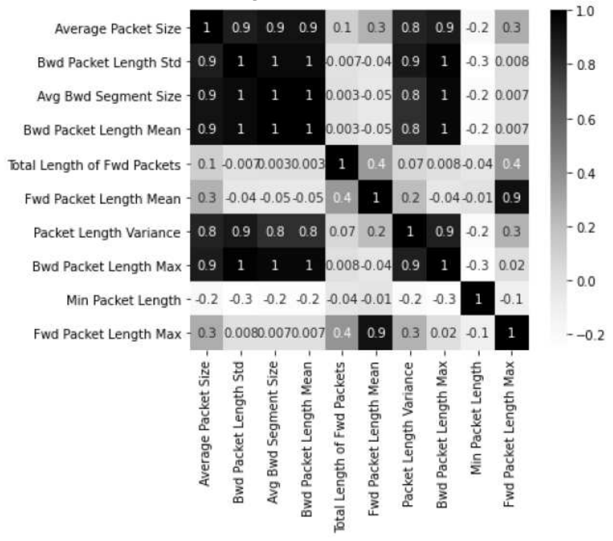

Fig. 3. Confusion Matrix Display for Ten Selected Features.

D. *Machine Learning Algorithm Implementation*

In order to select the best algorithm from a comparative analysis, a python package called *LazyPredict* was used to train on a very small subset of the data (10,000), 5000 safe and 5000 attack. This python package trains the data on 26 different machine learning algorithms automatically and it outputs the results showing the performance of the models and the time taken to train the models. From the results, considering the F1 Score and the execution time, the following models were chosen to train the resulting dataset:

1. XGB Classifier
2. Decision Tree Classifier
3. Random Forest Classifier
4. Multi-Layer Perceptron (MLP)
5. K-Nearest Neighbors (KNN) Classifier
6. Quadratic Discriminant Analysis (QDA)

IV. Results And Discussion

The accuracy, precision, f-measure, recall and time taken for training were the performance metrics used to evaluate the findings of this investigation. All of these criteria have a range of 0 to 1. When it gets close to 1, performance improves, but when it gets close to 0, it gets worse. The six algorithms above were used to train the data with both top ten features and top twenty features described earlier.

It is important to define certain terms by which the performance metrics were deduced as follows:

• TP stands for "True Positive" (Correct Detection). The attack data predicted as an attack.
• FP stands for False Positive (Type-1 Error). The benign data predicted as an attack.
• FN stands for False Negative (Type-2 Error). The Attack data classified as benign.
• TN stands for True Negative (Correct Rejection). Benign data predicted as benign [12].

A confusion matrix could be used to visualize these terms as shown in figure 4.

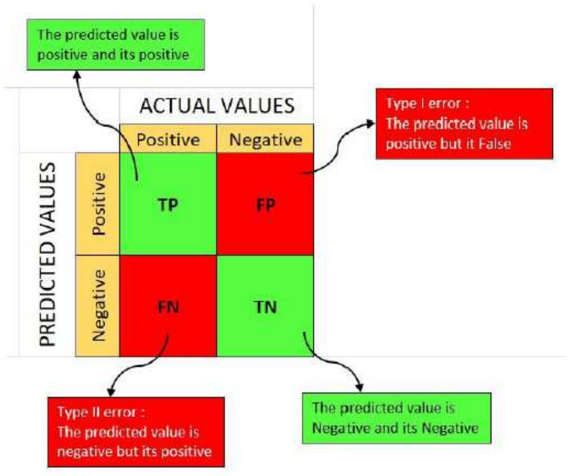
The predicted value is positive and its positive
ACTUAL VALUES
Positive
Negative
Type I error : The predicted value is positive but it False
PREDICTED VALUES
Positive
Negative
TP
FP
FN
TN
Type II error : The predicted value is negative but its positive
The predicted value is Negative and its Negative

Fig. 4. Confusion Matrix for Binary Classification.

XGB Classifier:
Tables 2 and 3 display a summary of the algorithm performance for ten and twenty features respectively.

Table 2: XGB Ten Features

Classifier	XG_BOOSTS
Number of Features	10
Accuracy	95.13%
Precision	95.08 %
Recall	95.57%
F1-Score	95.10%
Time Taken	29 Seconds

Table 3: XGB Twenty Features

Classifier	XG_BOOSTS
Number of Features	20
Accuracy	98.3%
Precision	98.18%
Recall	98.42%
F1-Score	98.28%
Time Taken	59 Seconds

Decision Tree Classifier:

Tables 4 and 5 display a summary of the algorithm performance for ten and twenty features respectively.

Table 4: Decision Tree Ten Features

Classifier	Decision Tree
Number of Features	10
Accuracy	94.96%
Precision	94.91%
Recall	95.41%
F1-Score	94.9%
Time Taken	2.24 Seconds

Table 5: Decision Tree Twenty Features

Classifier	Decision Tree
Number of Features	20
Accuracy	98.1%
Precision	98.0%
Recall	98.2%
F1-Score	98.1%
Time Taken	4.567 Seconds

Random Forest Classifier:

Tables 6 and 7 display a summary of the algorithm performance for ten and twenty features respectively.

Table 6: Random Forest Ten Features

Classifier	Random Forest
Number of Features	10
Accuracy	93.86%
Precision	93.91%
Recall	93.41%
F1-Score	93.9%
Time Taken	3.85 Seconds

Table 7: Random Tree Twenty Features

Classifier	Random Forest
Number of Features	20
Accuracy	97.3%
Precision	97.0%
Recall	97.2%
F1-Score	97.1%
Time Taken	3.95 Seconds

Multi-Layer Perceptron (MLP):

Tables 8 and 9 display a summary of the algorithm performance for ten and twenty features respectively.

Table 8: MLP Ten Features

Classifier	MLP
Number of Features	10
Accuracy	93.03%
Precision	93.1%
Recall	93.6%
F1-Score	93.0%
Time Taken	46.15 Seconds

Table 9: MLP Twenty Features

Classifier	MLP
Number of Features	20
Accuracy	97.4%
Precision	97.29%
Recall	97.6%
F1-Score	97.4%
Time Taken	45.29 Seconds

KNN Classifier:

Tables 10 and 11 display a summary of the algorithm performance for ten and twenty features respectively.

Table 10: KNN Ten Features

Classifier	KNN
Number of Features	10
Accuracy	93.65%
Precision	94.91%
Recall	95.41%
F1-Score	94.9%
Time Taken	75.23 Seconds

Table 11: KNN Twenty Features

Classifier	KNN
Number of Features	20
Accuracy	98.3%
Precision	98.0%
Recall	98.2%
F1-Score	98.1%
Time Taken	1344 Seconds

Quadratic Discriminant Analysis (QDA):

Tables 12 and 13 display a summary of the algorithm performance for ten and twenty features respectively.

Table 12: QDA Ten Features

Classifier	QDA
Number of Features	10
Accuracy	73.03%
Precision	80.68%
Recall	75.5%
F1-Score	72.33%
Time Taken	0.59 Seconds

Table 13: QDA Twenty Features

Classifier	QDA
Number of Features	20
Accuracy	63.9%
Precision	76.78%
Recall	67.24%
F1-Score	61.53%
Time Taken	1.56 Seconds

Table 14 displays a summary of results for all six algorithms considering accuracy and time for training as performance metrics.

Table 14: Result Summary

Algorithms	Accuracy% (10features)	Accuracy% (20features)	Time in s. (10features)	Time in s. (20features)
XG-Boost	95.13	98.3	29	59
Decision Tree	94.96	98.1	2.24	4.567
Random Forest	93.86	97.3	3.85	3.95
MLP	93.03	97.4	45.15	45.29
KNN	93.65	98.3	75.23	1344
QDA	73.03	63.9	0.59	1.56

Of the six algorithms, XG-Boost has the highest accuracy for both 10 features and 20 features, but took a relative long time for training and inference. QDA took the shortest time to train but had the least accuracy values. KNN presented the longest time for training on both 10 and 20 features dataset.

Random forest had very high accuracy with relatively small training time. The time of training as seen for both Random Forest and MLP is approximately the same for both the 20 feature dataset and the 10 feature dataset. Decision Tree presents the best balance between accuracy and training time. It has a very high accuracy for both feature data cases and took just a few seconds to train and do inference. It therefore stands out as the best classifier of all six models.

V. Conclusion

This paper presents a comparative analysis of the performance of six machine learning algorithms in detecting anomalies in network traffic. Owing to challenges in finding a live network to analyze, a CICIDS2017 dataset was used to train the machine learning models. Decision Tree was found to give the overall best result of all six algorithms considered. It is therefore recommended for network security specialists that deploy machine learning solutions for intrusion detection systems.

Acknowledgment

The authors would like to thank Elotech Solutions Benin City and EDYSYS Solutions Benin City for their support in this research.

References

[1] *IDS 2017*. (n.d.). Available at: https://www.unb.ca/cic/datasets/ids-2017.html.

[2] Nvidia. (2018). *NVIDIA blog: Supervised vs. Unsupervised learning*. The Official NVIDIA Blog. https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/ saravanan. (n.d.).

[3] *What is machine learning? - I school online*. (2020a). UCB-UMT. Available at: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning.

[4] G. MeeraGandhi. "Machine learning approach for attack prediction and classification using supervised learning algorithms." Int. J. Comput. Sci. Commun 1.2., (2010).

[5] L. Dhanabal and S. P. Shantharajah. "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms." International Journal of Advanced Research in Computer and Communication Engineering 4.6: 446-452. (2015).

[6] H. A. Nguyen and C. Deokjai. "Application of data mining to network intrusion detection: Classifier Selection Model." APNOMS '08: Proceedings of the 11th Asia-Pacific Symposium on Network Operations and Management: Challenges for Next Generation Network Operations and Service Management. October 2008. Pages 399–408. https://doi.org/10.1007/978-3-540-88623-5_41.

[7] G. Kalyani, A. J. Lakshmi. Performance assessment of different classification techniques for intrusion detection. IOSR J. Comput. Eng. (IOSRJCE) 7(5), 25–29 (2012).

[8] "train_test_split," scikit-learn 0.19.1 documentation. [Online]. Available: http://scikitlearn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[9] A. Géron, A. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc." (2017).

[10] "sklearn.preprocessing.LabelEncoder," 1.4. Support Vector Machines - scikit-learn 0.19.1 documentation. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.

[11] J. Brownlee, "Feature Importance and Feature Selection With XGBoost in Python," Machine Learning Mastery, 10-Mar-2018. [Online]. Available: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/.

[12] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," IEEE communications surveys & tutorials, vol. 16, no. 1, pp. 303-336, 2014.