*Journal of Advanced Computer Science & Technology*, 9 (2) (2020) 14-20

**Journal of Advanced Computer Science & Technology**

*Website: www.sciencepubco.com/index.php/JACST*

***Research paper***

# Evaluating the performance of machine learning algorithms for network intrusion detection systems in the internet of things infrastructure

**Yaser M. Banadaki ***

*Department of Computer Science, Southern University and A&M College, Baton Rouge, LA 70813, USA*
**Corresponding author E-mail: yaser_banadaki@subr.edu*

**Abstract**

As numerous Internet-of-Things (IoT) devices are deploying on a daily basis, network intrusion detection systems (NIDS) are among the most critical tools to ensure the protection and security of networks against malicious cyberattacks. This paper employs four machine learning algorithms: XGBoost, random forest, decision tree, and gradient boosting, and evaluates their performance in NIDS, considering the accuracy, precision, recall, and F-score. The comparative analysis conducted using the CICIDS2017 dataset reveals that the XGBoost performs better than the other algorithms reaching the predicted accuracy of 99.6% in detecting cyberattacks. XGBoost-based attack detectors also have the largest weighted metrics of F1-score, precision, and recall. The paper also studies the effect of class imbalance and the size of the normal and attack classes. The small numbers of some attacks in training datasets mislead the classifier to bias towards the majority classes resulting in a bottleneck to improving macro recall and macro F1 score. The results assist the network engineers in choosing the most effective machine learning-based NIDS to ensure network security for today's growing IoT network traffic.

***Keywords***: *Network Intrusion Detection Systems; Machine Learning Algorithms; Internet-of-Things; Malicious Cyberattacks; Network Traffic.*

## 1. Introduction

Network intrusion detection systems play an important role in the dramatic growth of the Internet of Things (IoT) that exposes new vulnerabilities in the network. Cyber-attacks are considered as a new remote weapon [1, 2] targeting critical infrastructures such as a presidential campaign [3], a nuclear program [4], government personnel data [5], and software providers [6]. It is vital to distinguish harmful data from normal data while using the internet network efficiently. Intrusion detection was described as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network" [7]. Intrusion Detection System (IDS) [8] is the most critical defense tool against the sophisticated and ever-growing network attacks. An IDS is a device that monitors the traffic of internet-connected devices and attempts to distinguish malicious or normal traffic. Different forms of harmful traffic such as a distributed denial of service (DDoS) attacks can happen on the internet, preventing the proper functioning of a server. Different IDS systems have been developed to detect harmful traffics [8-10]. Different machine-learning algorithms such as naive Bayes [11], neural network regression [12], support vector machine [13], principal component analysis [14], and random forest [15] have been used for attack detection [16]. The anomaly-based intrusion detection approach [17] trains on many normal behaviors in the network to detect abnormal behavior in the network. However, the techniques are mostly suffering from consistent performance evolutions and many false alarms. A reliable IDS with a low false alarm rate can be developed based on a set of signatures of known attacks [18]. Farid et al. used the Bayesian classifier to reach the accuracy of over 99% in detecting DDoS attacks [19]. For instance, ML-based detection of DDoS attack can help to take immediate action to reduce the $2.3 million cost of such attacks [20].

Intrusion Detection Evaluation Dataset (CICIDS2017), available from the Canadian Institute for Cybersecurity, contains the most updated real-world attack scenarios in networks [21]. However, this dataset has ~600,000 benign data from normal traffic, while only 11 and 21 samples are recorded for two attack types on a specific day. While the large size of datasets may contribute to the training process, a class imbalance dataset [7, 18] may mislead the classifier, making it biased towards the majority classes [22, 23]. The small sizes of some known attacks in training datasets are a bottleneck to developing an effective ML-based model. The amount of signature data describing each attack is a limited and very imbalance, making the identification of attacks challenging. Lopez et al. [20] excluded some attack types from the ML-based analysis due to the small number of samples. However, it is critical to evaluate the performance of ML algorithms for different size of the training data for known attacks. This paper evaluates the effect of such a class imbalance on the detection metrics in real network attacks and investigates the impact of training data size on the performance of the ML models.

The paper uses ML models in the IBM platform known as Auto AI [24] to identify the best type of model for the given data and efficiently compare the performance of ML models. Auto AI was described as "a suite of algorithms and feature transformations to automatically

engineer new, high-value features for a given dataset" [25]. The performance of ML models for specific training datasets is subjected to the experience of the data scientists in tuning complex network parameters. The use of Auto AI ensures that the ML process generates the most accurate and optimal predictive results that effectively scales with time and resources. Several supervised classification algorithms, such as XGBoost (XGB) [26], Random Forest (RF) [27], Decision Tree (DT) [28], and Gradient Boosting (GB) [29] are evaluated using metrics like detection accuracy, precision, recall, and F-score. Boosting makes a classifier strongly correlated with the true classification. This paper evaluates ML models in detecting real network attacks to assist the network engineers in choosing the most effective ML-based detection approach for today's network traffic.

The rest of the paper is organized as follows. An overview of the training process, CICIDS2017 dataset, and the attack scenarios are presented in Section 2. Section 3 discusses the performance of machine learning-based attack detectors, including the relative importance of the features for each model and their performance considering precision, recall, F-score, sensitivity, and specificity. The effect of the imbalance and size of the normal and attack classes on detection performance are also studied. The paper is summarized with some conclusions in section 5.

## 2. Training process

Figure 1 shows the procedure to train ML-based models to detect attacks from the network traffic. Raw data for training ML models are adopted from the CICIDS2017 dataset [21] that contains a mix of benign traffic and the most up-to-date common attacks. The dataset covers a diverse set of attack scenarios resembling the true real-world data that can be used for network security and intrusion detection purposes. In this paper, a subset of data from CICIDS2017 is taken to optimize the ML model that can be used to detect eight attack profiles: Brute Force attacks, DoS (Slowloris, and Slowhttptest), Heartbleed, Web attacks, Botnet and DDoS. SQL injection web attack includes a string of SQL commands that are created to force the database to reply to the request required to reveal information to find the administrator's password. Brute Force Attacks are used for cracking passwords and discovering hidden content in a web application. DoS Attack makes a machine or network resource unavailable temporarily by flooding the targeted machine or resource with superfluous requests that overload the machine leading to a denial of some legitimate requests. DoS Slowloris and DoS Slowhttptest keep a single machine's connection open with minimal bandwidth to consume the web server resources. DoS Hulk generates volumes of unique and obscure traffic at a web server to bypass caching engines and hit the server's direct resource pool. The heartbleed attack originates from a bug in the OpenSSL cryptography library. Following Transport Layer Security (TLS) protocol, a malformed heartbeat request is normally exploited with a small payload and large length field to a vulnerable server to extract the victim's response. Botnet includes several internet-connected devices to allow the attacker access to the device and network to steal data and send spam. The DDoS attack includes flooding the bandwidth or resources of the targeted system by generating huge network traffic.

The features of traffic flow are generated by the network traffic analyzer known as CICFlowMeter [30]. Recursive Feature Elimination (RFE) techniques are used to extract 85 appropriate features based on the time-stamp, source and destination IPs, source and destination ports, protocols, and type of attack in a CSV format file. The ML algorithms need to be kept generic so that a trained algorithm can predict an unseen instance correctly. As such, the available dataset is split into training and test dataset where the algorithm is trained using a training dataset with known attack labels, and a test dataset is used to evaluate the model performance in predicting the attack labels. A confusion matrix can be generated using the number of correct predictions on the test dataset to find the actual class label against the predicted class label for each category and to extract the classification metrics.

Figure 2 shows the training progress pipelines of four ML-based attack detectors, including XGBoost, random forest, decision tree, and gradient boosting classifiers that have been chosen as the top-performing ML algorithms. These algorithms are best suited to the CICIDS2017 data resulting in more significant accuracies among six available classifiers in Auto AI. XGBoost is a strong learner because of the optimizing step for every new tree that attaches, that reduces false alarms and improve the classification accuracy. Random forests have many trees combined using averages or majority rule at the end of the process. Decision trees are a series of sequential steps designed to provide probabilities, costs, or other consequences of making a particular decision. While random forests build each tree independently, gradient boosting builds one tree at a time, improving the shortcomings of existing weak learners. Also, gradient boosting combines results along the way, while random forests combine results at the end of the process.

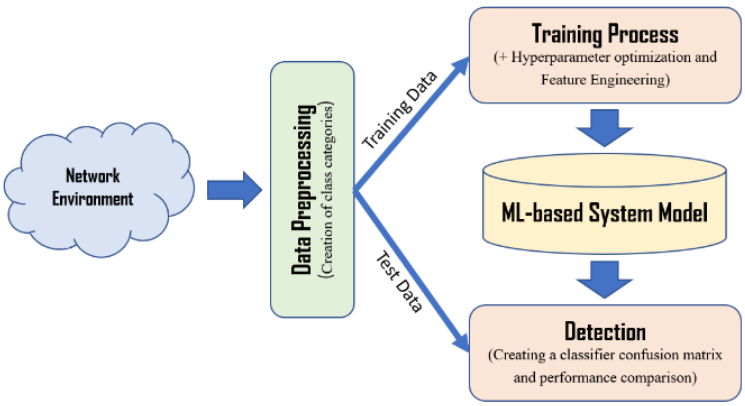



**Fig. 1:** Training Procedure of Intrusion Detection Including Data Preprocessing, Training and Optimizing the Training Algorithms, Deployment of ML-Based Intrusion Detectors, and Testing of the Model to Extract the Classification Performance Metrics.

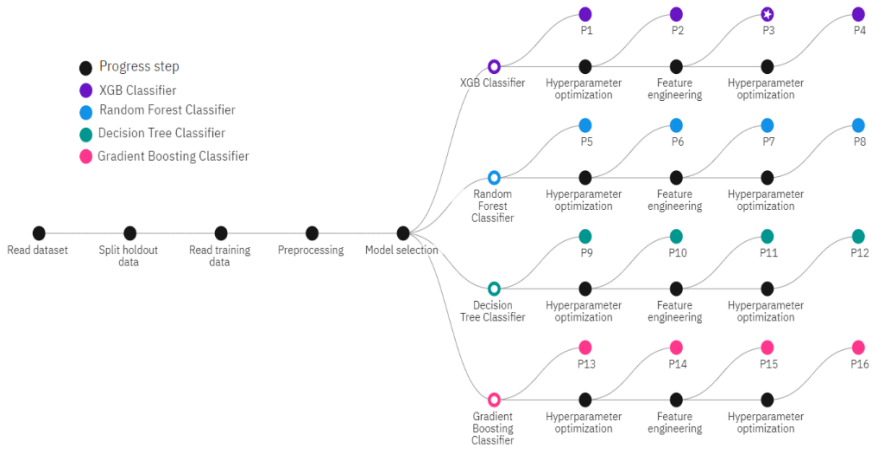



**Fig. 2:** Training Progress Pipelines for Four ML Models: Xgboost Classifier, Random Forest Classifier, Decision Tree Classifier, and Gradient Boosting Classifier.

For each of these four ML algorithms, AutoAI generates the following pipelines: automated model selection (Pipeline 1), hyperparameter optimization (Pipeline 2), automated feature engineering (Pipeline 3), hyperparameter optimization (Pipeline 4). The Hyper-parameter optimization (HPO) process includes finding a set of optimal parameters for the learning procedure to enable fast convergence to a better performing solution. To reduce ML bias, features are equally scaled, and the raw data is transformed into the combination of features that best represents the intrusion detection problem to achieve the most accurate detection of attacks. In the model selection process, small subsets of the data are tested to rank candidate ML algorithms, and gradually, the size of the subsets for the most promising algorithms are increased to find the best algorithm that matches CICIDS2017 data. The models use transfer learning (TL), in which the knowledge gained while solving one problem is applied to a different but related problem. The approach extracts existing knowledge learned from one environment to solve new problems. The pre-trained models take advantage of training with a lower amount of data for the new problem, and significantly shortens the training procedure.

## 3. Results and discussion

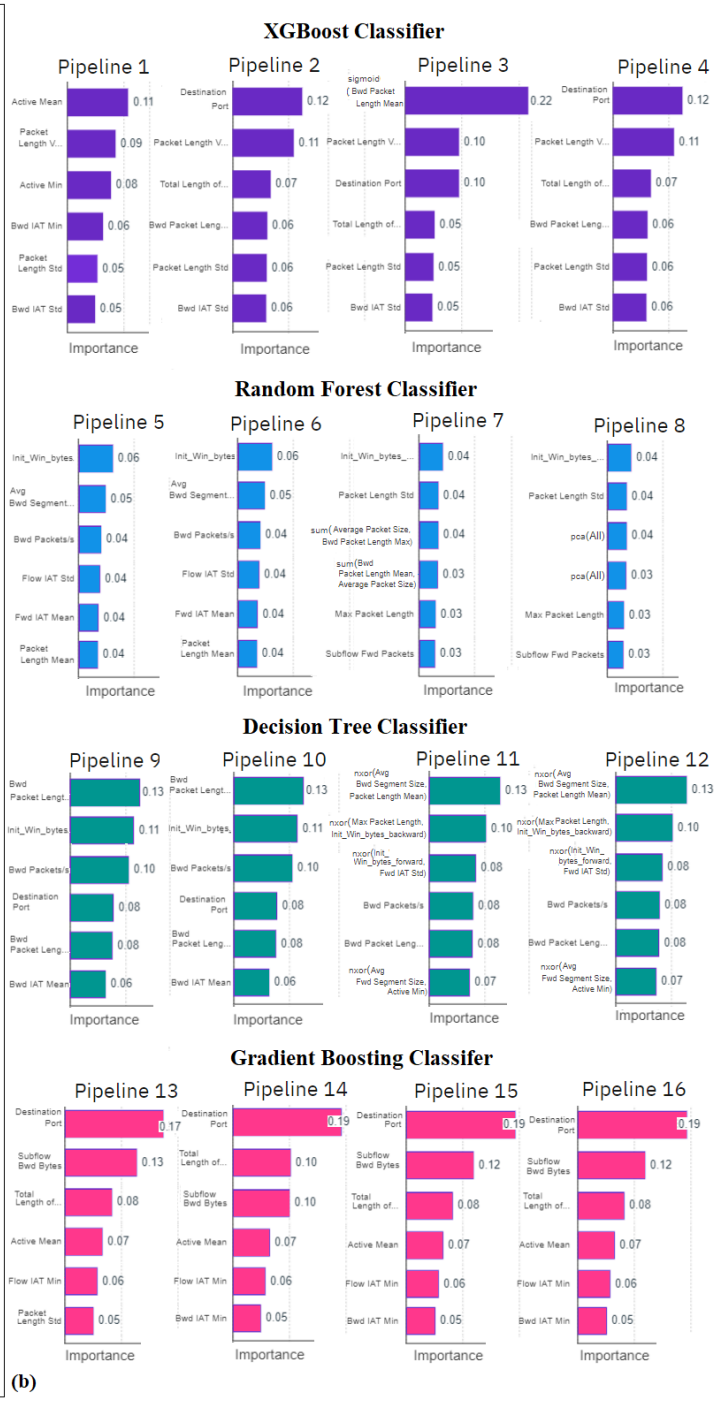
Figure 3 shows the relative importance of the first six features in predicting the attacks. The figure shows features in the order of its importance for each pipeline (i.e., ML models). It can be noticed that classification models evaluated the importance of their appropriate features differently to ensure the highest accuracy. Correlation between a pair of features is analyzed to eliminate features that contribute the same information about the data. The destination port is the key feature in both XGB classifiers (pipeline 2 and 4) and gradient boosting classifiers (pipeline 13-16). The importance of destination port is ~[0.10 - 0.12] for XG boost classifier and ~[0.17 - 0.19] for gradient boosting classifier. Boosting makes a strong learner by optimizing step for every new tree, allowing the classification model to generate less False Alarms and higher accuracy of classification. XGB algorithm has a regularization aspect to avoid data overfitting problems that make the classifier fast in dealing effectively with the system overwhelming with a float of attack. As such, XGB outperforms many existing models to deal with the majority of the attacks in a real-world network.

Five metrics are calculated: precision, recall, F-score, sensitivity, and specificity to evaluate the classification performance of the predictive models to detect each attack. The accuracy is calculated as a fraction of true positive among all the positive's recalled and can be viewed as a measure of a classifier's exactness. Recall (or sensitivity) is a fraction of true positives among all the true events and can be viewed as a measure of a classifier's completeness. Low precision and recall indicate many false positives and many false negatives, respectively. The F-score considers both precision and recall as the harmonic mean of the Precision and Recall indicating the worst accuracy when it becomes 0, while the best accuracy corresponds to 1.

In the micro-averaged F1-score or the micro-F1, micro-averaged precision and micro averaged recall is calculated over all the samples, and then combine the two. Micro qualifier calculates metrics globally by counting the total true positives, false negatives, and false positives. In other words, a micro-average look at all the samples together that is a proper measure of classification when the size of datasets is variable. Table 1 shows the classification metrics for all the 16 pipelines of four classifiers. Each model pipeline is scored for a variety of classification metrics. The accuracy is used as the ranking metric for multi-class classification models. It can be observed the XGB classifier has generated the best possible accuracy of attack detection. The accuracy of the XGB model reaches the maximum value of 98.4% using training data size 3 in Fig. 6. XGB-based attack detectors also have the largest weighted metrics of F1-score, precision, and recall equal to 0.979, 0.987, and 0.984 for pipeline 1. Micro qualifier does not take label imbalance into account by calculating metrics for each label and finds their unweighted mean. While each class has equal weights in macro-averaged metrics, the metrics of each class are weighted by the number of samples from that class in weighted-average metrics. Decision tree-based attack detectors show the largest F1 macro and recall macro are 0.911 and 0.912 in pipelines 9 and 10. Macro-averaged F1-score or the macro-F1 is an arithmetic mean of the per-class F1-scores, and macro-averaged recall is the arithmetic mean of the per-class recall. Macro-averaged metrics evaluate the performance of attack detectors across different datasets. Figure 4. provides a visual comparison of how each of these models from each pipeline performs based on various metrics. It can be observed that XGB-based attack detectors are the best in accuracy, followed by Gradient Boosting-based attack detector. Both detectors outperform RT and DT in F1 weighted, precision macro, precision weighted, and recall weighted.

| No | Feature |
|---|---|
| 1 | Flow ID |
| 2 | Source IP |
| 3 | Source Port |
| 4 | Destination IP |
| 5 | Destination Port |
| 6 | Protocol |
| 7 | Time stamp |
| 8 | Flow Duration |
| 9 | Total Fwd Packets |
| 10 | Total Backward Packets |
| 11 | Total Length of Fwd Pck |
| 12 | Total Length of Bwd Pck |
| 13 | Fwd Packet Length Max |
| 14 | Fwd Packet Length Min |
| 15 | Fwd Pck Length Mean |
| 16 | Fwd Packet Length Std |
| 17 | Bwd Packet Length Max |
| 18 | Bwd Packet Length Min |
| 19 | Bwd Packet Length Mean |
| 20 | Bwd Packet Length Std |
| 21 | Flow Bytes/s |
| 22 | Flow Packets/s |
| 23 | Flow IAT Mean |
| 24 | Flow IAT Std |
| 25 | Flow IAT Max |
| 26 | Flow IAT Min |
| 27 | Fwd IAT Total |
| 28 | Fwd IAT Mean |
| 29 | Fwd IAT Std |
| 30 | Fwd IAT Max |
| 31 | Fwd IAT Min |
| 32 | Bwd IAT Total |
| 33 | Bwd IAT Mean |
| 34 | Bwd IAT Std |
| 35 | Bwd IAT Max |
| 36 | Bwd IAT Min |
| 37 | Fwd PSH Flags |
| 38 | Bwd PSH Flags |
| 39 | Fwd URG Flags |
| 40 | Bwd URG Flags |
| 41 | Fwd Header Length |
| 42 | Bwd Header Length |
| 43 | Fwd Packets/s |
| 44 | Bwd Packets/s |
| 45 | Min Packet Length |
| 46 | Max Packet Length |
| 47 | Packet Length Mean |
| 48 | Packet Length Std |
| 49 | Packet Len. Variance |
| 50 | FIN Flag Count |
| 51 | SYN Flag Count |
| 52 | RST Flag Count |
| 53 | PSH Flag Count |
| 54 | ACK Flag Count |
| 55 | URG Flag Count |
| 56 | CWE Flag Count |
| 57 | ECE Flag Count |
| 58 | Down/Up Ratio |
| 59 | Average Packet Size |
| 60 | Avg Fwd Segment Size |
| 61 | Avg Bwd Segment Size |
| 62 | Fwd Avg Bytes/Bulk |
| 63 | Fwd Avg Packets/Bulk |
| 64 | Fwd Avg Bulk Rate |
| 65 | Bwd Avg Bytes/Bulk |
| 66 | Bwd Avg Packets/Bulk |
| 67 | Bwd Avg Bulk Rate |
| 68 | Subflow Fwd Packets |
| 69 | Subflow Fwd Bytes |
| 70 | Subflow Bwd Packets |
| 71 | Subflow Bwd Bytes |
| 72 | Init_Win_bytes_fwd |
| 73 | Act_data_pkt_fwd |
| 74 | Min_seg_size_fwd |
| 75 | Active Mean |
| 76 | Active Std |
| 77 | Active Max |
| 78 | Active Min |
| 79 | Idle Mean |
| 80 | Idle Packet |
| 81 | Idle Std |
| 82 | Idle Max |
| 83 | Idle Min |
| 84 | Label |

**(a)**




**(b)**

**Fig. 3:** (A) List of 84 Features of CICIDS2017 Dataset. (B) Feature Importance of Attack Detectors for Four Training Algorithms in Sixteen Pipelines.

**Table 1:** Pipeline Leaderboard of Four Attack Classifiers and Their Classification Metrics

| Name | Algorithm | Accuracy | F1 | | | Precision | | | Recall | | | Enhancements |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | macro | micro | weighted | macro | micro | weighted | macro | micro | weighted | |
| Pipeline 1 | XGB Classifier | 0.984 | 0.888 | 0.984 | 0.979 | 0.967 | 0.984 | 0.987 | 0.897 | 0.984 | 0.984 | *None* |
| Pipeline 2 | XGB Classifier | 0.984 | 0.900 | 0.984 | 0.982 | 0.910 | 0.984 | 0.982 | 0.901 | 0.984 | 0.984 | HPO-1 |
| Pipeline 3 | XGB Classifier | 0.983 | 0.896 | 0.983 | 0.981 | 0.905 | 0.983 | 0.981 | 0.898 | 0.983 | 0.983 | HPO-1 FE |
| Pipeline 4 | XGB Classifier | 0.983 | 0.896 | 0.983 | 0.981 | 0.905 | 0.983 | 0.981 | 0.898 | 0.983 | 0.983 | HPO-1 FE HPO-2 |
| Pipeline 14 | Gradient Boosting Classifier | 0.983 | 0.900 | 0.983 | 0.981 | 0.910 | 0.983 | 0.981 | 0.901 | 0.983 | 0.983 | HPO-1 |
| Pipeline 15 | Gradient Boosting Classifier | 0.983 | 0.899 | 0.983 | 0.981 | 0.908 | 0.983 | 0.981 | 0.900 | 0.983 | 0.983 | HPO-1 FE |
| Pipeline 16 | Gradient Boosting Classifier | 0.983 | 0.899 | 0.983 | 0.981 | 0.908 | 0.983 | 0.981 | 0.900 | 0.983 | 0.983 | HPO-1 FE HPO-2 |
| Pipeline 5 | Random Forest Classifier | 0.981 | 0.896 | 0.981 | 0.980 | 0.901 | 0.981 | 0.979 | 0.896 | 0.981 | 0.981 | *None* |
| Pipeline 6 | Random Forest Classifier | 0.981 | 0.896 | 0.981 | 0.980 | 0.901 | 0.981 | 0.979 | 0.896 | 0.981 | 0.981 | HPO-1 |
| Pipeline 13 | Gradient Boosting Classifier | 0.981 | 0.851 | 0.981 | 0.978 | 0.897 | 0.981 | 0.978 | 0.836 | 0.981 | 0.981 | *None* |
| Pipeline 7 | Random Forest Classifier | 0.980 | 0.865 | 0.980 | 0.979 | 0.904 | 0.980 | 0.979 | 0.846 | 0.980 | 0.980 | HPO-1 FE |
| Pipeline 8 | Random Forest Classifier | 0.980 | 0.865 | 0.980 | 0.979 | 0.904 | 0.980 | 0.979 | 0.846 | 0.980 | 0.980 | HPO-1 FE HPO-2 |
| Pipeline 9 | Decision Tree Classifier | 0.980 | 0.911 | 0.980 | 0.980 | 0.911 | 0.980 | 0.980 | 0.912 | 0.980 | 0.980 | *None* |
| Pipeline 10 | Decision Tree Classifier | 0.980 | 0.911 | 0.980 | 0.980 | 0.911 | 0.980 | 0.980 | 0.912 | 0.980 | 0.980 | HPO-1 FE HPO-2 |
| Pipeline 11 | Decision Tree Classifier | 0.980 | 0.854 | 0.980 | 0.980 | 0.836 | 0.980 | 0.980 | 0.909 | 0.980 | 0.980 | HPO-1 FE |
| Pipeline 12 | Decision Tree Classifier | 0.980 | 0.854 | 0.980 | 0.980 | 0.836 | 0.980 | 0.980 | 0.909 | 0.980 | 0.980 | HPO-1 FE HPO-2 |

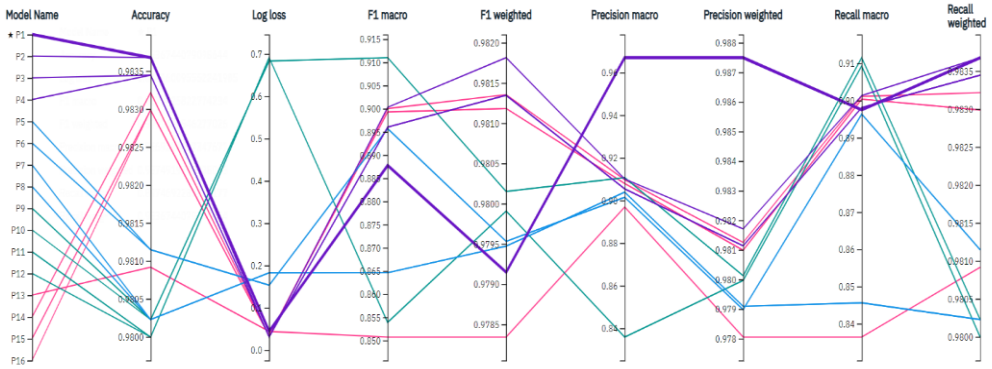



**Fig. 4:** View of Classification Metric Chart for 16 Pipelines of Four Attack Classifiers.

The ROC (Receiver Operating Characteristics) curve is a fundamental tool used for diagnostic test evaluation providing measures of overall predictive accuracy of the models. Figure 5 shows the ROC curve for four ML models. The curve depicts the proportion of positive outcomes that are correctly predicted, also known as the true positive rate (i.e., sensitivity), against the proportion of negative outcomes that are falsely predicted to be positive, also known as the false positive rate (i.e., 1-Specificity). The sensitivity is a measure of a classifier's completeness, and the specificity measures the proportion of correctly identified negatives. The area under the ROC curve represents the measure of separability, demonstrating how much models are capable of distinguishing between classes. ROC curves can be used to evaluate the performance of the four classification models. A ROC curve that passes through the upper left corner has 100% sensitivity and 100% specificity, indicating an ideal case with no overlap between the classes. The perfect area under the ROC curve is built on the truth value of '1' and '0', resulting in the angle-shaped elbow seen in the ROC curve. The accuracy of the classifier is higher when the ROC curve is closer to the upper left corner. The average ROC curves of attack classes show that there is an extremely concave curve for the XGB classifier and gradient boosting classifier. When the predictions of attacks overlap, the errors are introduced, which are the cases for the random forest classifier and decision tree classifier. From comparing the curves for four classification algorithms, it is seen that XGB and GB perform very well compared to RF and DT. All paragraphs must be justified alignment. With justified alignment, both sides of the paragraph are straight.

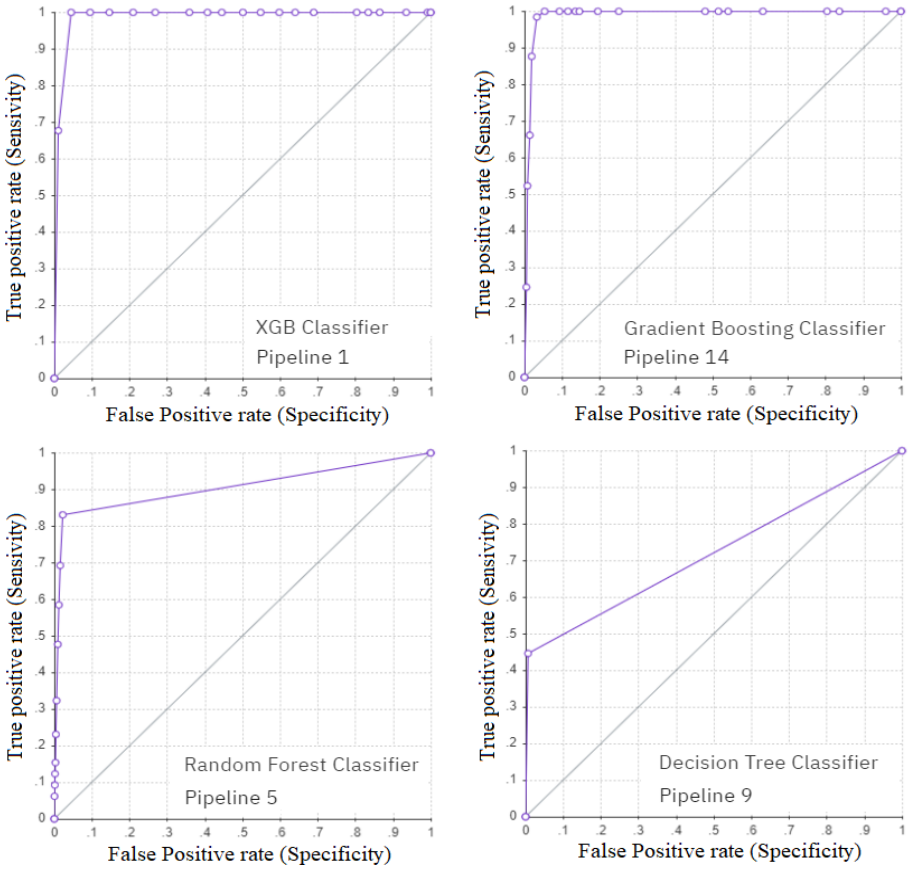



**Fig. 5:** ROC Curve for Four ML-Based Attack Detectors.

Figure 6 shows the classification measures of the XGB classifier for four different sizes of attack datasets. It can be noticed that the distribution of records for each type of attack is not equal. The smallest dataset has 100 records in normal traffics and in each of 6 attack classes. Two attack types have only 11 and 21 records that make the dataset slightly imbalance. The accuracy of the intrusion detection reaches 90.5%, and all other classification measures such as precision, recall, and F1 score are larger than 89%. The size of normal and attack datasets is increased in the largest dataset (i.e., Dataset Size 4), for example, reaching 105 for normal traffic and DoS Hulk attacks. However, there are still two attack types that have as low as 11 and 21 records in this dataset, making it very imbalance. It can be noticed that the accuracy, weighted recall, weighted precision, and weighted F1 score increases to ~ 99.6% for the largest dataset. However, the macro recall and macro F1 scores are calculated as 88.9% and 87.8%, respectively. The metrics decrease by increasing the imbalance in the larger datasets as it misleads the classifier to bias towards the majority classes. When a random attack sample in this dataset is ascertained for training and testing of the attack detector, the possibility of finding attack labels as "Web Attack–sql Injection" or "DoS slow loris" is minimal. While the large size of datasets may contribute to the training process, the small quantities of some known attacks in training datasets are the bottleneck to improve macro recall and macro F1 score.

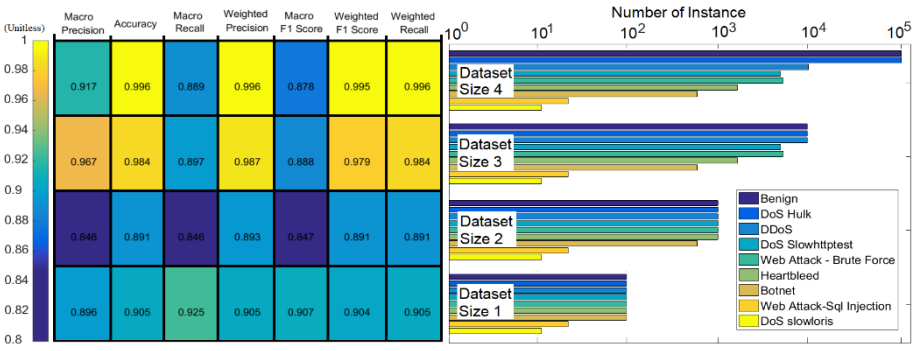



**Fig. 6:** Classification Measures of XGB Classifier for Four Different Sizes of Attack Datasets.

## 4. Conclusion

Reliable network intrusion detection systems are critical tools for the protection and security of IoT networks against malicious cyberattacks. In this paper, four machine learning algorithms are employed to evaluate their performance in detecting intrusions considering the accuracy, precision, recall, and F-score. The analysis of ML-based NIDS conducted using the CICIDS2017 dataset that contains the most updated real-world attack scenarios in networks. The results show that XGBoost performs better than the other ML algorithms leading to higher accuracy and higher weighted metrics of F1-score, precision, and recall. The accuracy, weighted F1-score, weighted precision, and weighted recall of XGB-based attack detectors reach the maximum values of 99.6%, 97.9, 98.7, and 98.4, respectively. The paper investigates how the class imbalance and the shortage of the attack instances mislead the classifier to bias towards the majority classes leading to

a bottleneck to improve the performance metrics. The paper assists the network engineers in choosing the most effective ML-based NIDS, ensuring network security for IoT devices.

## References


[1] A. Ahmim, N. Ghoualmi–Zine, A new adaptive intrusion detection system based on the intersection of two different classifiers, *International Journal of Security and Networks* 9(3) (2014) 125-132. https://doi.org/10.1504/IJSN.2014.065710.

[2] A. Ahmim N.G. Zine, A new hierarchical intrusion detection system based on a binary tree of classifiers, *Information & Computer Security* 23(1) (2015) 31-57. https://doi.org/10.1108/ICS-04-2013-0031.

[3] S. Detrow, Obama on Russian Hacking: We Need to Take Action. And We Will, *NPR News*, 2016. https://www.npr.org/2016/12/15/505775550/obama-on-russian-hacking-we-need-to-take-action-and-we-will, December 15, 2016.

[4] R. Langner, Stuxnet: Dissecting a cyberwarfare weapon, *IEEE Security & Privacy* 9(3) (2011) 49-51. https://doi.org/10.1109/MSP.2011.67.

[5] I. Bouteraa, M. Derdour, A. Ahmim, Intrusion Detection using Data Mining: A contemporary comparative study, *3rd* IEEE *International Conference on Pattern Analysis and Intelligent Systems (*2018) 1-8. https://doi.org/10.1109/PAIS.2018.8598494.

[6] N. Kshetri, Kaspersky Lab: from Russia with anti-virus, *Emerald Emerging Markets Case Studies* 1(3) (2011) 1-10. https://doi.org/10.1108/20450621111180954.

[7] R. Bace, P. Mell, NIST special publication on intrusion detection systems, Booz-Allen and Hamilton Inc McLean (2001). https://www.nist.gov/publications/intrusion-detection-systems. https://doi.org/10.6028/NIST.SP.800-31.

[8] H.J. Liao, C.H.R. Lin, Y.C. Lin, K.Y. Tung, Intrusion detection system: A comprehensive review, *Journal of Network and Computer Applications*, 36(1) (2013) 16-24. https://doi.org/10.1016/j.jnca.2012.09.004.

[9] F. Ertam, L.F. Kilincer, O. Yaman, Intrusion detection in computer networks via machine learning algorithms, *IEEE International Artificial Intelligence and Data Processing Symposium (IDAP)* (2017) 1-4. https://doi.org/10.1109/IDAP.2017.8090165.

[10] A. Lazarevic, V. Kumar, J. Srivastava, Intrusion detection: A survey, *Managing Cyber Threats: Springer* (2005) 19-78. https://doi.org/10.1007/0-387-24230-9_2.

[11] W. Li, Q. Li, Using naive Bayes with AdaBoost to enhance network anomaly intrusion detection, *3rd International Conference on Intelligent Networks and Intelligent Systems* (2010) 486-489. https://doi.org/10.1109/ICINIS.2010.133.

[12] S.K. Gautam, H. Om, Computational neural network regression model for Host based Intrusion Detection System, *Perspectives in Science* 8 (2016) 93-95. https://doi.org/10.1016/j.pisc.2016.04.005.

[13] J. Jha, L. Ragha, Intrusion detection system using support vector machine, *International Journal of Applied Information Systems (IJAIS)* 3 (2013) 25-30. 10.5120/icwac1342.

[14] G. Liu, Z. Yi, S. Yang, A hierarchical intrusion detection model based on the PCA neural networks, *Neurocomputing* 70(7) (2007) 1561-1568. https://doi.org/10.1016/j.neucom.2006.10.146.

[15] J. Zhang, M. Zulkernine, A. Haque, Random-forests-based network intrusion detection systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 38(5) (2008) 649-659. https://doi.org/10.1109/TSMCC.2008.923876.

[16] C.F. Tsai, Y.F. Hsu, C.Y. Lin, W.Y. Lin, Intrusion detection by machine learning: A review, *Expert systems with applications*, 36(10) (2009) 11994-12000. https://doi.org/10.1016/j.eswa.2009.05.029.

[17] D. Alexander, 5.6 million fingerprints stolen in US personnel data hack: government, ed: Reuters (2015). Online: https://www.reuters.com/article/us-usa-cybersecurity-fingerprints-idUSKCN0RN1V820150923.

[18] J. Peng, K.-K. R. Choo, and H. Ashman, User profiling in intrusion detection: A review, *Journal of Network and Computer Applications* 72 (2016) 14-27. https://doi.org/10.1016/j.jnca.2016.06.012.

[19] D. M. Farid, M. Z. Rahman, Anomaly network intrusion detection based on improved self adaptive bayesian algorithm, *Journal of Computers*, 5(1) (2010) 23-31. https://doi.org/10.4304/jcp.5.1.23-31.

[20] A. D. Lopez, A. P. Mohan, S. Nair, Network traffic behavioral analytics for detection of DDoS attacks, *SMU data science review* 2(1) (2019) 14. https://scholar.smu.edu/datasciencereview/vol2/iss1/14?utm_source=scholar.smu.edu%2Fdatasciencereview%2Fvol2%2Fiss1%2F14&utm_medium=PDF&utm_campaign=PDFCoverPages. I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, *ICISSP*, (2018) 108-116. https://doi.org/10.5220/0006639801080116.

[21] S. X. Wu, W. Banzhaf, The use of computational intelligence in intrusion detection systems: A review, *Applied soft computing* 10(1) (2010) 1-35. https://doi.org/10.5220/0006639801080116.

[22] H. Chauhan, V. Kumar, S. Pundir, E. S. Pilli, A comparative study of classification techniques for intrusion detection, *IEEE International Symposium on Computational and Business Intelligence* (2013) 40-43. https://doi.org/10.1016/j.asoc.2009.06.019.

[23] R. E. Hoyt, D. H. Snider, C. J. Thompson, S. Mantravadi, IBM Watson analytics: automating visualization, descriptive, and predictive statistics," *JMIR public health and surveillance* 2(2) (2016) e157. https://doi.org/10.1109/ISCBI.2013.16.

[24] G. Regkas, Empowering Citizen Data Scientists with IBM Watson AutoAI, Online: *https://towardsdatascience.com/empowering-citizen-data-scientists-with-watson-autoai-49a087df99e5*, (2020). https://doi.org/10.2196/publichealth.5810.

[25] S. S. Dhaliwal, A.-A. Nahid, R. Abbas, Effective intrusion detection system using XGBoost, *Information*, 9(7) (2018) 149. https://doi.org/10.3390/info9070149.

[26] N. Farnaaz, M. Jabbar, Random forest modeling for network intrusion detection system, *Procedia Computer Science* 89(1) (2016) 213-217. https://doi.org/10.1016/j.procs.2016.06.047.

[27] X. Li, N. Ye, Decision tree classifiers for computer intrusion detection, *Journal of Parallel and Distributed Computing Practices* 4(2) (2001) 179-190. https://doi.org/10.1145/1167253.1167288.

[28] P. Verma, S. Anwar, S. Khan, S. B. Mane, Network intrusion detection using clustering and gradient boosting, *9th IEEE International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (2018) 1-7. https://doi.org/10.1109/ICCCNT.2018.8494186.

[29] A. H. Lashkari, A. Seo, G. D. Gil, A. Ghorbani, CIC-AB: Online ad blocker for browsers, *International Carnahan Conference on Security Technology (ICCST)* (2017) 1-7. https://doi.org/10.1109/CCST.2017.8167846.