Acta Scientific COMPUTER SCIENCES

Volume 5 Issue 11 November 2023

Research Article

Applying Machine Learning Techniques in Cybersecurity Field

Saif Rawashdeh*

Department of Computer Science, Jordan University of Science and Technology, Jordan

***Corresponding Author:** Saif Rawashdeh, Department of Computer Science, Jordan University of Science and Technology, Jordan.

Received: October 11, 2023

Published: October 22, 2023

Abstract

Machine learning techniques have been applied in various fields and shown to be effective, like cybersecurity. Machine learning can be used in cybersecurity to detect and defend against network attacks. It can also be used to detect anomalies in system behavior that may indicate an attack is underway. Machine learning is a valuable tool for cybersecurity professionals and can help make systems more secure. This paper aims to develop seven machine learning algorithms (Decision Tree, Random Forest, Gradient Boosting, XGBoost, AdaBoost, Multilayer Perceptron, and Voting) to detect anomaly attacks using a well-known dataset named UNSW-NB15. To assess the performance of these models, there are four popular evaluation metrics: accuracy, precision, recall, and f1-score. Therefore, we applied two experiments and an enchantment experiment to detect several types of attacks: 1) Binary classification into two types of attacks (normal and malicious). 2) Multiclass classification (types of malicious attacks). 3) Enchantment experiment on the second experiment (choose the three most frequent attacks in the dataset out of nine attacks). These experiments are done to see if each algorithm is able to distinguish between the types of malicious attacks in the UNSW_NB15 dataset. The results showed that the voting classifier performed the best in the first experiment. Furthermore, when compared to others, the XGB performed better in the second and enchantment experiments.

Keywords: UNSW_NB15 Dataset; Machine Learning; Cybersecurity Attacks; Detection Attacks

Introduction

Cybersecurity is the process of defending computer systems and networks against information leakage, theft, or damage to their hardware, software, or electronic data, as well as disruption or misdirection of the services they offer [1]. The importance of the field has increased as a result of the growing reliance on computer systems, the Internet, wireless network standards like Bluetooth and Wi-Fi, as well as the spread of "smart" devices like smartphones, televisions, and the numerous items that make up the Internet of Things (IoT) [1]. Cybersecurity is one of the most important issues in the current world because of its complexity, both in terms of political usage and technological development. Its primary goal is to guarantee the dependability, integrity, and data security of the system [2].

A "cybersecurity attack" is any bad thing that happens to IT systems or the people who use them in order to get unauthorized access to the systems and the data or information they hold [3]. In the majority of situations, cyberattacks are carried out by criminals aiming to profit from the attack. In other circumstances, the goal is to disrupt operations by blocking access to IT systems or harming physical equipment directly [3]. The latter is frequently state-sponsored and conducted by state actors or cybercriminals working for them. Cybersecurity attacks can be directed at specific businesses or individuals, or they can be broad in scope, affecting several organizations across multiple regions and countries [4]. Targeted attacks frequently spread beyond their original targets to cause problems for all businesses. The global NotPetya outbreak in June 2017 was most likely a result of a state-sponsored strike against Ukrainian banks and utilities. According to publications documenting the clean-up, it had the intended impact on Ukraine, but it also extended abroad, resulting in around $10 billion in costs to recover IT systems and lost productivity [2-4].

Machine learning (ML) is a field of study that focuses on understanding and making methods that "learn," or use data to get better at a set of tasks. Artificial intelligence is associated with it [5]. Machine learning algorithms create a model based on training data to make predictions or judgments without having to be explicitly programmed to do so. Machine learning algorithms are utilized in a wide range of applications, including medicine, email filtering, speech recognition, and computer vision, where developing traditional algorithms to do the required tasks is difficult or impossible [5].

Machine learning can be used by cybersecurity systems to look for patterns and learn from them, which can help stop attacks from

happening again and again and adapt to changes in behavior. It can help cybersecurity teams be more proactive about stopping threats and responding to attacks that are already happening [5]. It can help firms use their resources more strategically by reducing the amount of time spent on regular tasks. Machine learning could make cybersecurity easier, more proactive, less expensive, and more effective [4,5]. But it can only do that if the data that machine learning is based on gives a full picture of the environment. Garbage in, garbage out, as they say [5].

The goal of this paper is to use several machine learning algorithms on a popular network intrusion dataset to figure out if each sample is normal or not, and then to tell the different attacks in this dataset that are not normal from each other. The remainder of this paper is organized as follows: Section 2 presents some previous works that are related to this paper. Section 3 describes the methodology used in this paper. Section 4 illustrates the results. Section 5 concludes the paper and suggests some future work.

Related Work

Several authors applied machine learning algorithms in the cybersecurity field to detect different types of attacks based on real-time datasets or existing datasets from several resources, as shown in Table 1.

Kumar., *et al*. [6] proposed a classification model to detect several attacks from malicious activities in the network, which is called a decision tree (DT). They used the UNSW-NB15 dataset, which contains nine attack samples (DoS, reconnaissance, backdoor, fuzzers, analysis, exploits, worms, shellcode, and generic) and 44 features for normal attacks. To assess the performance of these trees, they used many evaluation metrics: accuracy, precision, and recall. This model achieved an accuracy of 90.74%. Amaizu., *et al*. [7] employed a deep learning model for network intrusion detection called Deep Neural Network (DNN). They used the NSL-KDD, UNSW-NB15, and CSECIC-IDS2018 datasets. The model's performance was then assessed using four assessment metrics: accuracy, precision, recall, and F1-Score. The DNN model performed well in each dataset, as shown by the following results: 97.89% in NSL-KDD, 89.99% in UNSW-NB15, and 76.47% in CSECIC-IDS2018.

Kasongo., *et al*. [8] used machine learning algorithms to build intrusion detection systems that could spot attacks on networks. They used a popular dataset called UNSW-NB15 that contains several kinds of network attacks. This dataset contains nine attack samples (DoS, reconnaissance, backdoor, fuzzers, analysis, exploits, worms, shellcode, and generic) and normal attacks with 44 features. These algorithms are SVM, XGBoost, k-Nearest Neighbor (kNN), Logistic Regression (LR), Artificial Neural Network (ANN), and Decision Tree (DT). They have shown that these methods achieve a higher accuracy of 90.55% in the detection process when using XGBoost as a feature selection method. Tuan., *et al*. [9] used five machine learning algorithms to detect different types in a well-known cybersecurity dataset. They used a popular dataset called UNSW-NB15 that contains several kinds of network attacks. This dataset contains nine attack samples (DoS, reconnaissance, backdoor, fuzzers, analysis, exploits, worms, shellcode, and generic) and normal attacks with 44 features. The machine learning algorithms are SVM, ANN, Nave Bayes (NB), DT, and Unsupervised Machine Learning (USML). They showed that the USML obtained good performance in this dataset, with an accuracy of 94.78%.

Shushlevska., *et al*. [10] applied an intrusion detection system based on various machine learning algorithms to detect the network attack on the UNSW-NB15 dataset. This dataset contains nine attack samples (DoS, reconnaissance, backdoor, fuzzers, analysis, exploits, worms, shellcode, and generic) and normal attacks with 44 features. These algorithms are logistic regression (LR), nave Bayes (NB), random forest (RF), and decision tree (DT). The F1 Score and Recall values for the Naive Bayes method are 76.1% and 85.3%; those for the Logistic Regression algorithm are 78.2% and 96.1%; those for the Decision Tree classifier are 88.3% and 95.4%; and those for the Random Forest algorithm are 89.3% and 98.5%.

Ref	Year	Attack	Cybersecurity Dataset	Algorithm	Result
[6]	2020	• Analysis • Backdoor • DoS • Exploits • Fuzzers • Generic • Normal • Reconnaissance • Shellcode • Worms	UNSW-NB15	DT	Accuracy = 90.74%

[7]	2020	* NSL-KDD: • R2L • DoS • U2R • Probe * UNSW-NB15: • Analysis • Reconnaissance • DoS • Exploits • Fuzzers • Generic • Normal • Worms • Backdoor • Shellcode * CSECIC-IDS2018 • Infiltration • Web Attacks • DoS • DDoS • Botnet • Brute-force • Normal	NSL-KDD, UNSW-NB15, and CSECIC-IDS2018	DNN	Accuracy = 97.89%, 89.99% and 76.47 in three datasets, respectively
[8]	2020	• Analysis • Reconnaissance • DoS • Exploits • Fuzzers • Generic • Normal • Worms • Backdoor • Shellcode	UNSW-NB15	SVM, XGBoost, kNN, LR, ANN and DT.	XGBoost accuracy = 90.85%
[9]	2019	• Analysis • Reconnaissance • DoS • Exploits • Fuzzers • Generic • Normal • Worms • Backdoor • Shellcode	UNSW-NB15	SVM, ANN, NB, DT, and USML	USML accuracy = 94.78%.

[10]	2022	• Analysis • Reconnaissance • DoS • Exploits • Fuzzers • Generic • Normal • Worms • Backdoor • Shellcode	UNSW-NB15	RF DT LR NB	F1-score and recall of RF = 89.3% and 98.5%

Table 1: Previous Related Work for Cybersecurity Attacks Detection.

Methodology

Figure 1 shows the proposed methodology used in this study to detect various types of attacks in the UNSW-NB15 cybersecurity dataset. The following sections illustrate the proposed methodology.

Dataset description

We used in our experiment a well-known dataset that has many cybersecurity attacks: the UNSW_NB15[1] dataset. Table 2 shows the name of each dataset, the number of instances, the number of features, and what the cybersecurity attacks are.

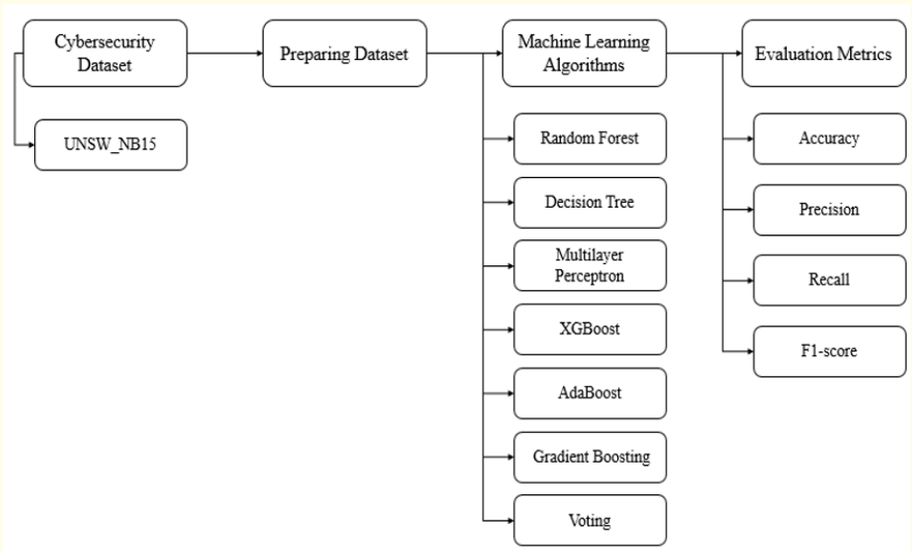

Figure 2: Flow-chart of Proposed Methodology.

Table 2: Information of Cybersecurity Datasets.

Dataset Name	No. of Instances	No. of Features	Cybersecurity Attacks
UNSW_NB15	257,673	45	Analysis, Reconnaissance, DoS, Exploits, Fuzzers, Generic, Normal, Worms, Backdoor, Shellcode

This dataset was made by the IXIA traffic generator [11], which uses three servers: two for normal attacks and one for malicious attacks. This dataset contains 45 features and 257,673 instances, divided into 93,000 normal attacks and 164,673 malicious attacks. The malicious attacks included nine types of attacks: analysis, reconnaissance, DoS, exploits, fuzzers, generic, normal, worms, backdoors, and shellcode. Table 3 shows the features of this dataset. In addition, Table 4 and Figure 2 present these attacks and the counts for each attack.

[1]UNSW_NB15: https://www.kaggle.com/mrwellsdavid/unsw-nb15.

Table 3: UNSW_NB15 Features.

No.	Feature	No.	Feature	No.	Feature
1	id	16	dloss	31	response_body_len
2	dur	17	sinpkt	32	ct_srv_src
3	proto	18	dinpkt	33	ct_state_ttl
4	service	19	sjit	34	ct_dst_ltm
5	state	20	djit	35	ct_src_dport_ltm
6	spkts	21	swin	36	ct_dst_sport_ltm
7	dpkts	22	stcpb	37	ct_dst_src_ltm
8	sbytes	23	dtcpb	38	is_ftp_login
9	dbytes	24	dwin	39	ct_ftp_cmd
10	rate	25	tcprtt	40	ct_flw_http_mthd
11	sttl	26	synack	41	ct_src_ltm
12	dttl	27	ackdat	42	ct_srv_dst
13	sload	28	smean	43	is_sm_ips_ports
14	dload	29	dmean	44	attack_cat
15	sloss	30	trans_depth	45	label

Attack	Label	Frequency
Normal	Normal	93000
Analysis	Malicious	2677
Backdoor	Malicious	2329
DoS	Malicious	16353
Exploits	Malicious	44525
Probing	Malicious	23,389
Fuzzers	Malicious	24246
Generic	Malicious	58871
Reconnaissance	Malicious	13987
Shellcode	Malicious	1511
Worms	Malicious	174

Table 4: Attack Classes in NSL-KDD Dataset.

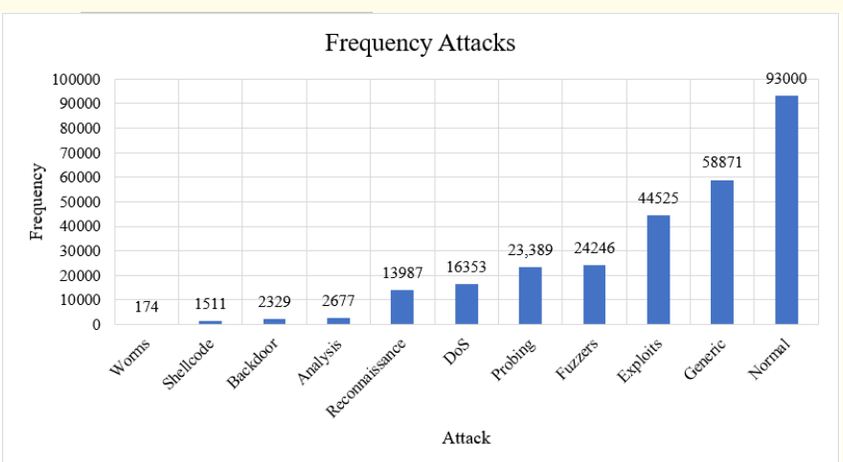

Figure 2: UNSW_NB15 Frequency Attacks.

Preparing dataset

To use different types of machine learning algorithms on these datasets, we use a common encoding technique [12] to turn the non-numerical features in each dataset into numerical features. This method is known as "Label Encoding," and it turns non-numerical data into machine-readable forms by replacing each value with a unique number starting at 0 [12]. In the dataset, named UNSW_NB15, the proto, service, and state are categorical features. Therefore, we must convert them to numerical features. After that, the dataset was split into two datasets: a training dataset to build the models and a testing dataset to assess the performance of these models. The size of each dataset is as follows: 0.10 of the whole dataset is for testing, and the remainder is a training dataset.

Machine learning algorithms

After the dataset is ready, it is put through seven machine learning algorithms that look for the above cybersecurity attacks. We used a hold-out technique to split the dataset into training and testing datasets with a test size of 0.1, which means that 0.9 of the whole dataset is a training dataset and 0.1 of the whole dataset is a testing dataset. The training dataset is used to build several machine learning models, while the testing dataset is used to assess the performance of these models.

Random forest algorithm (RF)

Random Forest is an ensemble-supervised learning method for regression and classification tasks in which a large number of decision trees are trained. For classification tasks, the prediction result is the class picked by the most trees. For regression tasks, the mean or average forecast of each tree is returned [13]. Decision trees have a tendency to overfit their training set, which is corrected by random decision forests. As a result, the Random Forest classifier is used to classify each cybersecurity dataset, which incorporates several attacks [14,15]. In our experiment, we used the random forest with classification type because the label is discrete, and the parameters of the RF that we used were as follows: n_estimators (number of decision trees) = 100, max_features = sqrt, max_depth = None, random_state = 42.

Decision tree algorithm (DT)

Decision Tree is a supervised machine learning algorithm that makes judgments based on a set of rules, similar to what people do [16]. Decision tree learning, also known as induction of decision trees, is one of the predictive modeling approaches used in three fields: data mining, statistics, and machine learning [17]. It goes from observations about a sample (represented in the branches) to inferences about the sample's target value using a decision tree (represented in the leaves that are attack types) [18].

Classification trees are tree models with a discrete target variable; in these tree structures, leaves represent class labels (types of attacks), and branches represent features in the dataset that lead to predicting the class labels [19,20]. Regression trees are decision trees with a continuous target variable (typically real numbers) [19]. Decision trees are one of the best-known machine learning algorithms due to their comprehensibility and simplicity. In our experiment, we used the decision tree with classification type because the label is discrete, and the parameters of the DT that we used are as follows: criterion = gini and random_state = 42.

Multilayer perceptron algorithm (MLP)

A multilayer perceptron (MLP) is a type of feedforward artificial neural network. In some contexts, the term MLP refers to networks made up of multiple layers of perceptrons (with threshold activation), whereas in others, it refers to any feedforward ANN. The term "vanilla" neural networks refers to multilayer perceptrons, particularly those having a single hidden layer [21]. An MLP has at least three node layers: an input layer, a hidden layer, and an output layer. Each node is a neuron with a nonlinear activation function, with the exception of the input nodes [21,22]. MLP employs backpropagation as a supervised learning technique during training. The multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can tell the difference between data that isn't linearly separable [23].

In our experiment, we used the MLP with classification type because the label is discrete, and the parameters of the MLP that we used are as follows: activation = relu and random_state = 42.

eXtreme gradient boosting algorithm (XGBoost)

Extreme Gradient Boosting, abbreviated as XGBoost, is used for classification and regression tasks. The gradient-boosting algorithm has been parallelized and carefully optimized [24]. The training time is greatly reduced by parallelizing the entire boosting procedure. We train hundreds of models on different subsets of the training dataset and then vote on the best-performing model [24], rather than creating the best model possible on the data (as in traditional approaches). In many cases, XGBoost outperforms classic gradient-boosting approaches [25]. The Python implementation gives you access to a slew of inner parameters that you may modify for better precision and accuracy [26].

The general purpose of this algorithm is to convert weak learners (decision trees) into strong learners, which means that the strong learner produces the final prediction label (the average of each prediction by week classifier) [26]. The XGBoost [24-26] has a number of significant features: 1) Parallelization: The model is designed to run simultaneously on several CPU cores. 2) Regularization: XGBoost offers a variety of regularization penalties to prevent overfitting. Regularizations with penalties result in successful training, allowing the model to generalize successfully. 3) Non-linearity: XGBoost can recognize and learn from non-linear data patterns. 4) Cross-validation is built-in and immediately available. 5)

Scalability: XGBoost can run in a distributed fashion, allowing you to manage huge volumes of data thanks to distributed servers and clusters like Hadoop and Spark. Many programming languages are supported, including C++, Java, Python, and Julia.

In our experiment, we used this algorithm with the classification type because the label is discrete, and the parameters of XGBoost that we used are as follows: colsample_bylevel = 1, learning_rate = 0.1, gamma = 0, n_estimators = 100, and random_state = 42.

AdaBoost algorithm

The statistical classification meta-algorithm AdaBoost (short for "adaptive boosting") is a statistical classification meta-algorithm. It can be combined with a variety of other learning algorithms to boost performance. Other learning algorithms' output (from "weak learners') is blended into a weighted total that represents the boosted classifier's final output [27]. AdaBoost is adaptive in that it tweaks succeeding weak learners in favor of instances misclassified by earlier classifiers. It may be less prone to the overfitting problem than other learning algorithms in some situations. Individual learners may be poor, but as long as their performance is marginally better than random guessing, the final model will converge to a powerful learner [28]. Although AdaBoost is most commonly used to combine weak base learners (such as decision 28), it has been demonstrated that it can also be used to combine strong base learners (such as deep decision trees), resulting in a more accurate model [29].

Every learning algorithm has many different parameters and configurations to tweak before it reaches optimal performance on a dataset, and most of them fit some problem types better than others. [27-29] they often say that AdaBoost is the best out-of-the-box classifier (with decision trees as weak learners). When combined with decision tree learning, data obtained at each stage of the AdaBoost algorithm on the relative 'hardness' of each training sample is fed into the tree-growing process, causing later trees to focus on more difficult-to-classify samples [29].

In our experiment, we used the AdaBoost with classification type because the label is discrete, and the parameters of GB that were used were as follows: algorithm = SAMME.R, learning_rate = 1.0, n_estimators = 50 and random_state = 42.

Gradient boosting algorithm (GB)

Gradient boosting is a machine learning approach that can be used for regression and classification, among other things. It returns a prediction model in the form of a group of weak prediction models, which are decision trees [30]. Gradient boosting is a method for making one strong learner out of several weak learners (decision trees).In this instance, individual decision trees are poor learners [31]. Each tree in the sequence is related to the one before it, with each tree striving to correct the error of the one before it. Due to this sequential relationship, boosting algorithms are often slow to train yet incredibly exact. In statistical learning, models that learn slowly perform better [30,31]. The weak learners are fitted in such a way that each new learner fits into the residuals of the previous stage as the model improves. The final model brings together the outcomes of each phase to produce a strong learner. A loss function is used to detect residuals. For example, in a regression work, mean squared error (MSE) can be utilized, and in a classification task, logarithmic loss (log loss) can be employed. It's worth noticing that when a new tree is added to the model, nothing changes. The added decision tree fits the current model's residuals [30-32].

In our experiment, we used the GB with classification type because the label is discrete, and the parameters of the GB that we used are as follows: subsample= 1.0, learning_rate = 0.1, criterion= friedman_mse, n_estimators = 100, and random_state = 42.

Voting algorithm

A voting classifier is an ensemble machine learning model that learns from a group of models and predicts an output (class) based on the output's highest chance of being the desired class [33]. It simply adds up the results of each classifier fed into the voting classifier and predicts the output class with the most votes [34]. We propose a single model that trains on numerous models and predicts output based on the cumulative majority of votes for each output class, rather than building separate specialized models and determining their performance. Two forms of voting are supported by the Voting Classifier [35,36]. 1) Hard voting: the projected output class with the most votes, i.e., the one having the highest likelihood of being predicted by each of the classifiers, is the one with the best chance of being predicted by each of the classifiers. 2) Soft voting: the output class is a forecast based on an average of the likelihood provided to that class.

In our experiment, we used voting with classification type because the label is discrete, and the parameters of voting that were used were as follows: estimators = DT, RF, and XGB, and voting type = hard.

Results and Discussion

This section shows the experimental results for each cybersecurity dataset in each machine learning algorithm based on four evaluation metrics.

Evaluation metrics

There are a number of ways to evaluate the machine learning algorithms used, such as accuracy, precision, recall, and f1-score [37]. These metrics' formulas are as follows: TP = true positives; TN = true negatives; FP = false positives; and FN = false negatives.

Accuracy: is the most intuitive performance metric is the ratio of properly predicted samples to total samples, which is simply a ratio of correctly predicted samples to total samples.

$$Accuracy = \frac{TP+TN}{TP + TN + FP + FN} \quad \text{---------- (1)}$$

Precision: is the ratio of correctly predicted positive tweets to the total predicted positive samples.

$$Precision = \frac{TP}{TP+FP} \quad \text{---------- (2)}$$

Recall: is the proportion of accurately anticipated positive samples to the total number of positive samples predicted.

$$Recall = \frac{TP}{TP+FN} \quad \text{---------- (3)}$$

F1-score: is the weighted average of Precision and Recall.

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad \text{---------- (4)}$$

In this dataset, we applied three experiments to detect several types of attacks: 1) Binary classification (normal and malicious attack types). 2) Multiclass classification (malicious attack types). 3) the enhancement experiment on the second experiment. These experiments are done to see if each algorithm is able to distinguish between the types of malicious attacks in the UNSW_NB15 dataset.

Binary Classification experiment

In this experiment, machine learning algorithms are applied to detect if the sample in this dataset is normal or the result of a malicious attack. As shown in Table 5 and Figure 3, the performance results for the machine learning algorithms used are based on four metrics: precision, accuracy, f1-score, and recall. The voting classifier outperforms the others in terms of performance results compared with the detection attack process.

Table 5: Performance Results of Binary Classification.

Models	Accuracy	Precision	Recall	F1-score
DT	0.986184	0.990992	0.987386	0.989185
RF	0.984904	0.988175	0.988235	0.988205
GB	0.968372	0.974165	0.976469	0.975316
XGB	0.988901	0.995838	0.986779	0.991288
AdaBoost	0.962706	0.97066	0.971072	0.970866
MLP	0.970661	0.980338	0.973679	0.976998
Voting	0.989716	0.994453	0.989448	0.991944

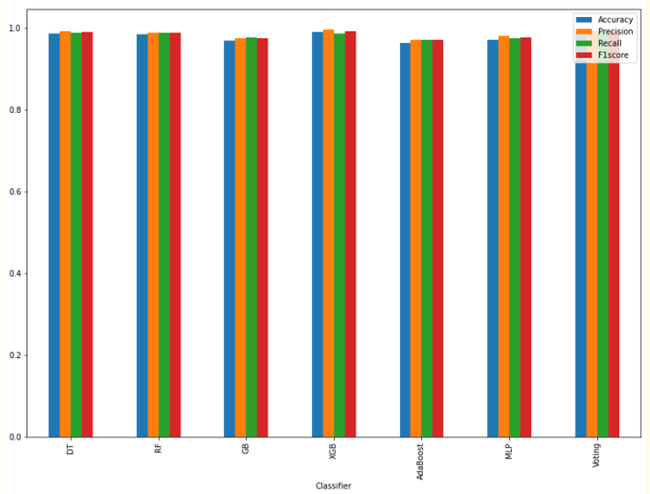

Figure 3: Performance Results of Binary Classification.

Multiclass Classification experiment

In this experiment, machine learning algorithms are applied to detect the types of malicious code in this dataset: analysis, backdoors, DoS, exploits, fuzzers, generics, reconnaissance, shellcode, and worms. As shown in Table 6 and Figure 4, the performance results for the machine learning algorithms used are based on four metrics: precision, accuracy, f1-score, and recall. The XGB algorithm achieved better performance results in these metrics in the detection process compared with the others.

Table 6: Performance Results of Multiclass Classification.

Models	Accuracy	Precision	Recall	F1-score
DT	0.802222	0.623885	0.626894	0.625035
RF	0.820561	0.662426	0.612215	0.630347
GB	0.820197	0.773805	0.617132	0.65556
XGB	0.831977	0.805031	0.653065	0.688423
AdaBoost	0.574508	0.502255	0.400902	0.344738
MLP	0.800644	0.739343	0.484815	0.530402
Voting	0.823901	0.665136	0.649436	0.655904

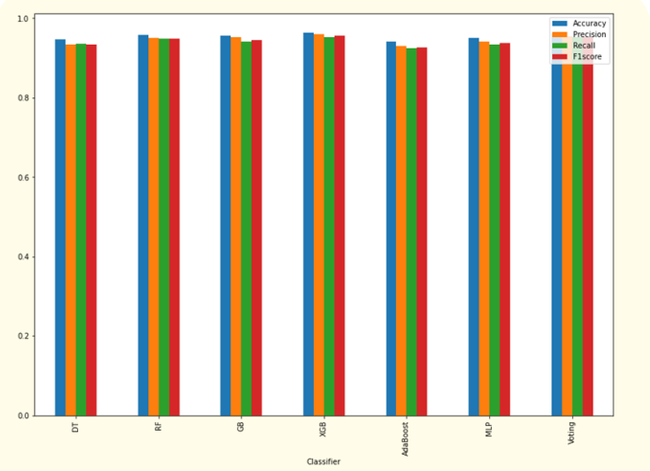

Figure 4: Performance Results of Multiclass Classification with three Attacks.

Enhancement experiment

To improve the results of the second experiment, we applied the machine learning algorithms to three types of attacks (generic, fuzzy, and exploit) because the models found it difficult to detect one of the nine attacks and the number of samples in deleted attacks was small. Table 7 and Figure 5 show the performance results of these algorithms for three network attacks. It is still true that the XGB algorithm achieved better performance results in these metrics in the detection process compared with the others.

Table 7: Performance Results of Multiclass Classification with three Attacks.

Models	Accuracy	Precision	Recall	F1-score
DT	0.946181	0.933022	0.934954	0.933975
RF	0.958559	0.950125	0.948758	0.949297
GB	0.95613	0.95187	0.941505	0.945738
XGB	0.963886	0.959218	0.952679	0.955509
AdaBoost	0.941559	0.930602	0.924622	0.927133
MLP	0.949628	0.941601	0.934617	0.937439
Voting	0.961144	0.953394	0.951498	0.9523

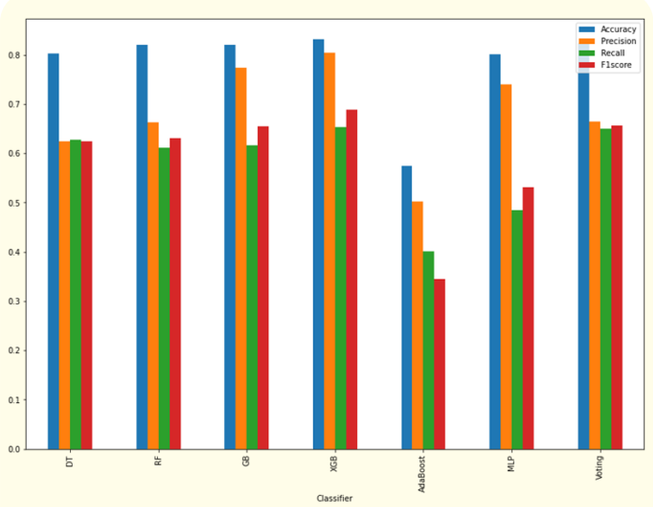

Figure 5: Performance Results of Multiclass Classification.

Conclusion and Future Work

In this research paper, we made several machine learning algorithms (Decision Tree, Random Forest, Gradient Boosting, XGBoost, AdaBoost, Multilayer Perceptron, and Voting) to find anomaly attacks. We used a popular dataset to carry out our experiments, which is called the UNSW-NB15 dataset. Most of the time, there are four ways to judge how well these models work: accuracy, precision, recall, and f1-score.On this dataset, we used two experiments and an enchantment experiment to detect several forms of attack: 1) Binary classification into two categories (normal and malicious attacks). 2) Multiclass classification into numerous categories (malicious attack types). 3) The enchantment experiment in the second experiment (choose the three most frequent attacks in the dataset out of nine attacks). The goal of these tests is to see if each algorithm can discriminate between the different forms of cybersecurity attacks in the UNSW NB15 dataset. According to the results, the voting classifier performed the best in the first experiment. Furthermore, the XGB outperformed in the second and enchantment experiments.

In future work, we will plan to apply other machine learning and deep learning algorithms to this dataset. In addition, we will also plan to apply the aforementioned machine learning algorithms to the other dataset to evaluate the performance of these algorithms.

Bibliography

1. Seemma PS., *et al*. "Overview of cyber security". *International Journal of Advanced Research in Computer and Communication Engineering 7*.11 (2018): 125-128.

2. Ervural B C and Ervural B. "Overview of cyber security in the industry 4.0 era". In Industry 4.0: managing the digital transformation (2018): 267-284.

3. Chowdhury A. "Recent cyber security attacks and their mitigation approaches–an overview". In International conference on applications and techniques in information security (2016): 54-65.

4. El-Rewini Z., *et al*. "Cybersecurity challenges in vehicular communications". *Vehicular Communications 23* (2020): 100214.

5. Handa A., *et al*. "Machine learning in cybersecurity: A review". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9*.4 (2019): e1306.

6. Kumar V., *et al*. "An integrated rule based intrusion detection system: analysis on UNSW-NB15 data set and the real time online dataset". *Cluster Computing* 23.2 (2020): 1397-1418.

7. Amaizu GC., *et al*. "Investigating Network Intrusion Detection Datasets Using Machine Learning". In 2020 International Conference on Information and Communication Technology Convergence (ICTC) (2020): 1325-1328.

8. Kasongo S M and Sun Y. "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset". *Journal of Big Data* 7.1 (2020): 1-20.

9. Tuan T A., *et al*. "Performance evaluation of Botnet DDoS attack detection using machine learning". *Evolutionary Intelligence* (2019): 1-12.

10. Shushlevska M., *et al*. "Anomaly detection with various machine learning classification techniques over UNSW-nb15 dataset" (2022).

11. Moustafa N and Slay J. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In 2015 military communications and information systems conference (MilCIS) (2015): 1-6.

12. Hancock J T and Khoshgoftaar T M. "Survey on categorical data for neural networks". *Journal of Big Data* 7.1 (2020): 1-41.

13. Pal M. "Random forest classifier for remote sensing classification". *International Journal of Remote Sensing* 26.1 (2005): 217-222.

14. Farnaaz N and Jabbar M A. "Random forest modeling for network intrusion detection system". *Procedia Computer Science* 89 (2016): 213-217.

15. Idhammad M., *et al*. "Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest". *Security and Communication Networks* (2018).

16. Kingsford C and Salzberg SL. "What are decision trees?". *Nature Biotechnology* 26.9 (2008): 1011-1013.

17. Quinlan J R. "Induction of decision trees". *Machine Learning* 1.1 (1986) 81-106.

18. De Ville B. "Decision trees". *Wiley Interdisciplinary Reviews: Computational Statistics* 5.6 (2013): 448-455.

19. Kotsiantis SB. "Decision trees: a recent overview". *Artificial Intelligence Review 39*.4 (2013): 261-283.

20. Amor N B., *et al*. "Naive bayes vs decision trees in intrusion detection systems". In Proceedings of the 2004 ACM symposium on Applied computing (2004): 420-424.

21. Noriega L. "Multilayer perceptron tutorial". *School of Computing. Staffordshire University* (2005).

22. Tang J., *et al*. "Extreme learning machine for multilayer perceptron". IEEE transactions on neural networks and learning systems 27.4 (2015): 809-821.

23. Ramchoun H., *et al*. "Multilayer perceptron: Architecture optimization and training" (2016).

24. Mitchell R and Frank E. "Accelerating the XGBoost algorithm using GPU computing". *Peer Journal of Computer Science 3* (2017): e127.

25. Pan B. "Application of XGBoost algorithm in hourly PM2. 5 concentration prediction". In IOP conference series: earth and environmental science 113.1 (2018): 012127.

26. Dong W., *et al*. "XGBoost algorithm-based prediction of concrete electrical resistivity for structural health monitoring". *Automation in Construction* 114 (2020): 103155.

27. Hu W and Hu W. "Network-based intrusion detection using Adaboost algorithm". In The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05) (2005): 712-717.

28. Jabri S., *et al*. "Moving vehicle detection using Haar-like, LBP and a machine learning Adaboost algorithm". In 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS) (2018): 121-124.

29. Yuan L and Zhang F. "Ear detection based on improved adaboost algorithm". In 2009 International Conference on Machine Learning and Cybernetics (2009): 2414-2417.

30. Son J., *et al*. "Tracking-by-segmentation with online gradient boosting decision tree". In Proceedings of the IEEE international conference on computer vision (2015): 3056-3064.

31. Peter S., *et al*. "Cost efficient gradient boosting". *Advances in Neural Information Processing Systems* 30 (2017).

32. Lusa L. "Gradient boosting for high-dimensional prediction of rare events". *Computational Statistics and Data Analysis* 113 (2017): 19-37.

33. Kumar U K., *et al*. "Prediction of breast cancer using voting classifier technique". In 2017 IEEE international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM) (2017): 108-114.

34. El-Kenawy E S M., *et al*. "Novel feature selection and voting classifier algorithms for COVID-19 classification in CT images". *IEEE Access 8* (2020): 179317-179335.

35. Khan M A., *et al*. "Voting classifier-based intrusion detection for iot networks". In Advances on Smart and Soft Computing (2022): 313-328.

36. Mahabub A. "A robust technique of fake news detection using Ensemble Voting Classifier and comparison with other classifiers". *SN Applied Sciences 2*.4 (2020): 1-9.

37. Dalianis H. "Evaluation metrics and evaluation". In Clinical text mining (2018): 45-53.