Study of the Effectiveness of ML Algorithm Parameters in Classifying DDoS Attack in SDN Networks

Dr. Boushra Maala•
Dr. Mothana Alkubaily••
Mohammed Abd Al-Hameed•••

(Received 21 / 11 / 2024. Accepted 27 / 5 / 2025)

ABSTRACT

Software Defined Network (SDN) is the new era of networking because of the advantages it offers. But They suffer from multiple security threats and attacks targeting their vulnerabilities. Perhaps the most prominent of these attacks are Distributed Denial-of-Service (DDoS) attacks. Machine learning techniques are increasingly used in predicting security attacks. In this paper, we will present a practical study of a set of machine learning algorithms for predicting DDoS attacks.
The study based on a set of algorithms, and for each algorithm, we defined the best and worst parameters comparing on the basis of the accuracy, f1-score scales and whether the algorithm was appropriate for real-time applications. The results showed that the decision tree algorithm performed the best with an accuracy of 99.99%, while the Multinomial NB algorithm performed the worst with an accuracy of 64.36%. The SVM algorithm had the longest training time at around 76 seconds, while the decision tree algorithm had the best time at 0.018 seconds. In terms of F1 score, the decision tree algorithm was the best at 99.99%, while the worst was the Multinomial NB algorithm at 69.26%.

Keywords: Software Defined Network, Distributed Denial of Service, Machine Learning, Accuracy, F1-score, Real time application

• **Associate Professor, Department of Communication and Electronics, Faculty of Mechanical and Electrical Engineering, Lattakia University, Lattakia, Syria. boushra.maala@gmail.com**
•• **Professor, Department of Communication and Electronics, Faculty of Mechanical and Electrical Engineering, Lattakia University, Lattakia, Syria mothanna.alkubeily@manara.edu.sy**
••• **Postgraduate Student (Ph.D.), Department of Communication and Electronics, Faculty of Mechanical and Electrical Engineering, Lattakia University, Lattakia, Syria mohammedabdolhamed589@gmail.com**

دراسة فعالية بارامترات خوارزميات ML في تصنيف هجوم DDoS في شبكات SDN

د. بشرى معلا*

د. مثنى القبيلي**

محمد عبد الحميد***

(تاريخ الإيداع 21 / 11 / 2024. قُبِل للنشر في 27/ 5 / 2025)

□ ملخّص □

الشبكات المعرفة بالبرمجيات (SDN) هي العصر الجديد للشبكات بسبب المزايا التي تقدمها، لكنها تعاني من تهديدات أمنية متعددة وهجمات تستهدف نقاط الضعف فيها. ربما أكثر هذه الهجمات شهرة هي هجمات حجب الخدمة الموزعة (DDoS).

تستخدم تقنيات التعلم الآلي بشكل متزايد في التنبؤ بالهجمات الأمنية. سنقدم في هذا البحث، دراسة عملية لمجموعة من خوارزميات التعلم الآلي للتنبؤ بهجمات DDoS.

استندت الدراسة إلى مجموعة من الخوارزميات، وحددنا لكل خوارزمية، أفضل وأسوأ البارامترات بالمقارنة على أساس الدقة ونسبة F1 والكفاءة في تطبيقات الزمن الحقيقي. أظهرت النتائج أن خوارزمية شجرة القرار كانت الأفضل أداءً بدقة 99.99% بينما خوارزمية Multinomial NB كانت الأسوأ بدقة 64.36%، وخوارزمية SVM كانت الأسوأ في وقت التدريب بحدود 76 ثانية بينما الأفضل كانت خوارزمية شجرة القرار بزمن 0.018 ثانية ومن ناحية نسبة F1 كانت خوارزمية شجرة القرار الأفضل بنسبة 99.99% بينما الأسوء كانت خوارزمية MultinomialNB بنسبة 69.26%.

الكلمات المفتاحية: الشبكات المعرفة بالبرمجيات، حجب الخدمة الموزع، التعلم الآلي، مقياس F1، تطبيقات الزمن الحقيقي.

* أستاذ مساعد، قسم هندسة الاتصالات والالكترونيات، كلية الهندسة الميكانيكية والكهربائية، جامعة اللاذقية، اللاذقية، سوريا boushra.maala@gmail.com

** أستاذ، قسم هندسة الاتصالات والالكترونيات، كلية الهندسة الميكانيكية والكهربائية، جامعة اللاذقية، اللاذقية، سوريا mothanna.alkubeily@manara.edu.sy

*** طالب دكتوراه، قسم هندسة الاتصالات والالكترونيات، كلية الهندسة الميكانيكية والكهربائية، جامعة اللاذقية، اللاذقية، سوريا mohammedabdolhamed589@gmail.com

Introduction:

Software Defined Network (SDN) are defined by two main characteristics: The first is to separate the level of control from the level of the data, so that intelligence is removed from the elements of the network and concentrated within one point called a controller, that is responsible for the complete management of the network, and the second is the programmability, so that the network can be fully managed through software codes injected into the controller to perform the required tasks[1]. The devices within the network collect information on the status of the network on a continuous basis and send it to the controller who, based on the information sent to him, forms flow bases, injects it into the network hardware and then continuously monitors it.
The controller is at the core of the network and is responsible for all control operations and commands within the network. SDNs suffer from a range of security threats due to their centralized control. The most prominent of these threats are distributed service blocking attacks (DDoS)[2], which typically target central points within the network or their services in order to make service within those networks unavailable to legitimate users. DDoS attacks vary so that many protocols within the network may be targeted. Within this paper, we focused on Transport Layer Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP) based attacks. Machine learning algorithms are widely used to classify security attacks within networks [3]. Accordingly, in this article, we evaluated the performance of a variety of algorithms, namely tree, logistic regression, Support Vector Maching, and Bayes, so that we worked on SDN dataset for DDoS attacks.

1- Methodology and Material

The study relied on a dataset [Dataset_sdn] of DDOS attacks in SDN networks [4]. We applied a set of machine learning algorithms using the Python programming language [5], primarily relying on the sklearn [6], matplot [7], and pandas [8] libraries. The process went through several stages, including:

1. Pre-processing the data to encode textual data and eliminate empty values within the data.
2. Normalizing the data to prevent values from being scattered and affecting the performance of the studied classifiers.
3. Dividing the data into training and testing sets.
4. Training the classifiers on the training data to build the model.
5. Testing the models and evaluating their performance, as well as comparing them.

The programming operations were performed using the Spyder program [9], and Google Colab [10] environment was utilized to implement the classifiers.

2- Distributed Denial of Service (DDoS):

A SYN flood attack [11] is the most common type of a DDoS attack. In the first quarter of 2019, this type was classified as the most frequent type of DDoS attacks. An attacker takes advantage of the three-way handshake method in TCP to establish a connection with the server. In a normal scenario, the client initiates the request to share data with the server by sending the SYN packet. The request is then assigned to a queue (i.e. memory storage). The server responds by sending the SYN/ACK package to the client and then waits until
the semi-open connection is completed or the TCP connection expires. In the event of a SYN flood attack, the victim server receives an enormous number of SYN packages, but the ACK that is last and required for the termination of the TCP triple handshake. As a result, the server's queue is depleted, resulting in the cancelation of all applications from legitimate customers. The attacker is the tool and determines the IP address, port address, and data payload size. In addition, an attacker can hide the IP address of the real source and use the

false host name option to set a fake IP address, ensuring that the target will never know the real address. Figure 1 shows how to execute an attack.

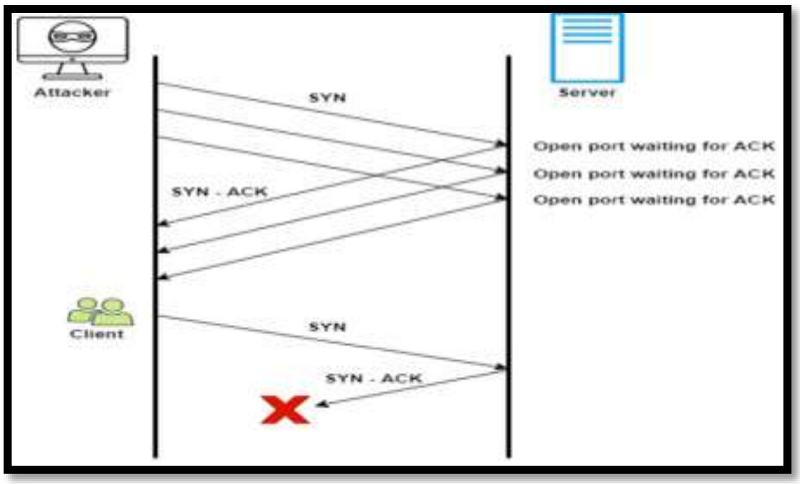
Attacker
Server
SYN
Open port waiting for ACK
Open port waiting for ACK
Open port waiting for ACK
SYN - ACK
Client
SYN
SYN - ACK

Fig-1: TCP-SYN flood attack.

A protocol flood attack (UDP) [12], as figure2 shows, is a type of DDoS attack in which an attacker targets random ports in the server through IP packages including UDP packages. In this type of attack, the host checks for apps that listen to a particular port. If the host does not find applications, it responds with ICMP messages stating that the destination is not accessible. When a large number of UDP packages target the victim, the host is forced to send many inaccessible packets of ICMP. As a result, the server's resources are consumed with these UPD packets, which makes the host unresponsive to legitimate client**.**

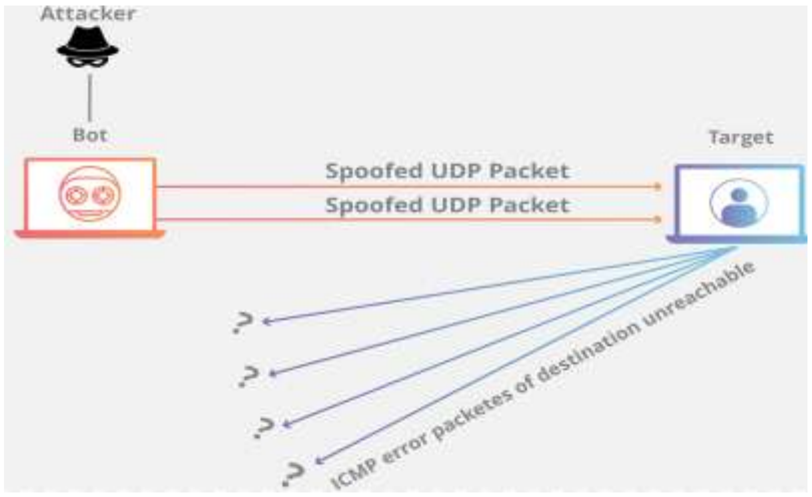

Fig-2: UDP flood attack

The ICMP flood attack [13], also known as the ping attack, is intended to target the victim's employer with a large number of echo requests. The victim's target server must send a response packet for each request from the sender. Each ICMP request requires the server to use its resources to process the request and send the response to the sender. This type of attack drains the server's resources with a large number of echo requests, making the server inaccessible to legitimate users, as shown in figure3.

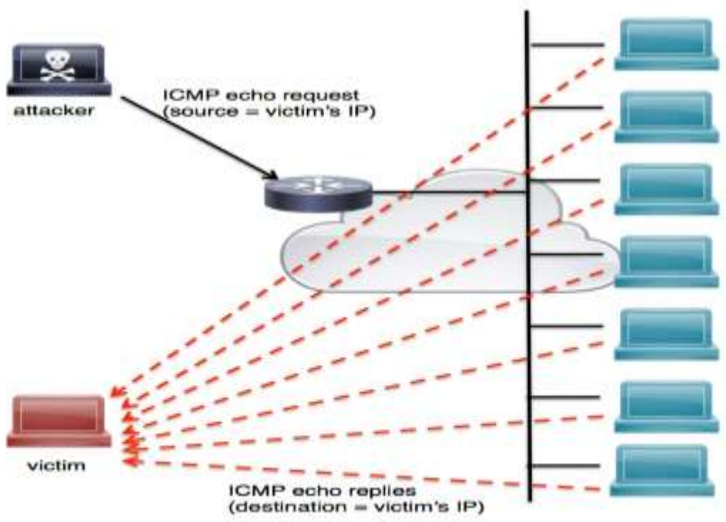

Fig-3: ICMP flood attack.

3- ML Algorithms:

In this section, we provide a description of the algorithms used in this paper, as shown in figure 4, along with an explanation of the most important parameters that directly affect the performance of the algorithm.

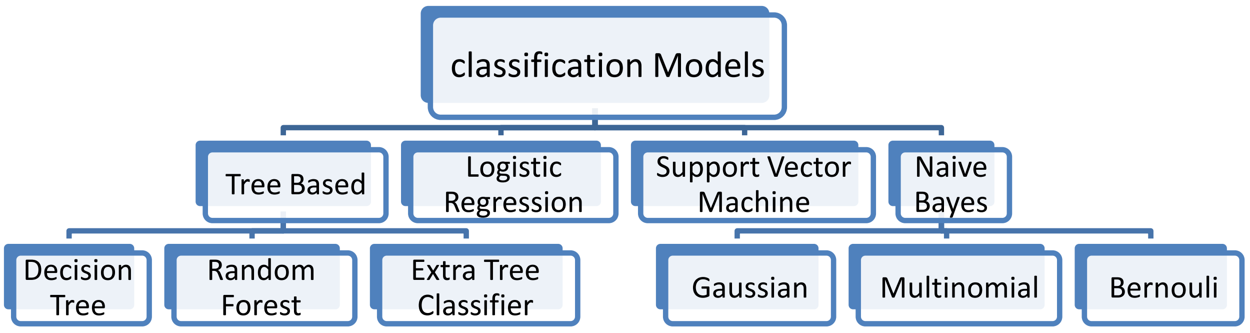

Fig-4: Classification Models in ML

These algorithms are:

3-1 Decision Tree Classifier:

The decision tree is a supervised learning technique commonly used for classification problems [14], although it can also be applied to regression problems. It functions as a structured classifier, where the internal nodes represent dataset features, the branches represent decision rules, and the leaf nodes represent the outcomes. Within the decision tree, there are two types of nodes: decision nodes and leaf nodes. Decision nodes are responsible for making decisions and have multiple branches, while leaf nodes are the end results of those decisions and do not have any further branches. The decisions or tests made are based on specific features of the dataset. The name "decision tree" stems from its resemblance to a tree, starting with a root node that expands into other branches, forming a tree-like structure. The CART algorithm, which stands for Classification and Regression Tree algorithm, is commonly used to build the decision tree. This algorithm prompts questions and, based on the answers (yes/no), divides the tree into subtrees.

3-2 Random Forest Classifier:

The Random Forest classifier is a powerful multi-classifier that exhibits a low misclassification error rate [15]. It operates by combining multiple independent decision trees into a single ensemble, enabling it to effectively handle large datasets and outliers present in the data. Each decision tree within the ensemble contributes to the classification process, ultimately leading to maximum accuracy. However, it is important to note that the computational time required for predictions can be relatively high due to the involvement of multiple trees. On a positive note, the Random Forest classifier excels in detecting attacks within networks, showcasing a high performance in this domain.

3-3 Extra Tree Classifier :

Extra Trees is a collective machine learning approach that, similar to Random Forests, trains multiple decision trees and combines their results to make predictions [16]. However, there are a few distinctions between Extra Trees and Random Forests. Random Forests employ bootstrapping to select different subsets of training data, ensuring diversity among decision trees. In contrast, Extra Trees utilize the entire dataset to train the decision trees. To maintain diversity among individual decision trees, Extra Trees randomly select values to split features and create child nodes. Conversely, Random Forests employ an algorithm to greedily search and select the optimal value for splitting the feature. Despite these differences, Random Forests and Extra Trees share many similarities, and the influential parameters in each classifier remain the same.

Within the three algorithms mentioned above, there are a set of parameters that had an impact on the results given by the classifier. The following parameters had an effect on the three algorithms:

- **criterion**: refers to the measure used to evaluate the quality of a split at a node. A split is a decision made by the algorithm to divide the data into two or more subsets based on a feature and a threshold value. The goal of the split is to maximize the separation between the classes or the reduction of impurity in the subsets.
- **min_weight_fraction_leaf:** is a hyperparameter that controls the minimum fraction of the total weight of the data required to be at a leaf node. The weight of a sample can be used to assign more importance to certain samples or to balance imbalanced datasets. The weight of a sample is typically proportional to its frequency or inverse to its rarity. It is used to prevent overfitting by controlling the minimum number of samples required to be at a leaf node. If the fraction of the total weight of the data at a leaf node is less than min_weight_fraction_leaf, the node is considered a leaf and the prediction is based on the majority class or the regression value of the samples at the node.

There is an additional parameter that has an impact on the DT algorithm, which is:

- **splitter**: It refers to the rule used to split the data at a node into two or more subsets. The splitter is determined by selecting the best feature and threshold that maximize a criterion, such as Gini impurity, information gain, or gain ratio. The splitter is used to divide the data into subsets that are more homogeneous in terms of the target variable (e.g., class labels). The goal of the splitter is to create a tree that can generalize well to new, unseen data by capturing the underlying patterns and relationships in the data.

There is an additional parameter that has an impact on both RF and ETC algorithms, which is:

- **min_impurity_decrease** is a parameter in decision tree classifiers that determines the minimum impurity decrease required to split an internal node. In other words, it sets the threshold for the amount of improvement in the model's impurity measure that must be achieved to justify splitting a node.

3-4 Support Vector Machine classifier:

Support Vector Machines (SVM) is a supervised learning model that excels in classifying non-linear data [17]. It achieves this by utilizing kernels to transform the input into a higher-dimensional feature space. The core principle of SVM involves creating a boundary, often referred to as a hyper-plane, to separate the predicted halves of the data. One of the notable advantages of this classifier is its ability to effectively detect outliers even when trained with a limited amount of data.

The most influential parameters in SVM classifier are:

- **C**: is a regularization parameter that controls the tradeoff between achieving a low training error and a low testing error. It determines the penalty for misclassifying data points and influences the width of the margin and the number of support vectors. A high C value allows a smaller margin and more support vectors, leading to overfitting, while a low C value allows a larger margin and fewer support vectors, leading to underfitting. Finding the optimal value of C is essential for obtaining a well-performing SVM model.
- **kernel**: is a function that transforms the input data from the original feature space into a higher-dimensional space, where it becomes easier to separate the data with a linear boundary. The kernel function calculates the similarity between two data points in the feature space and maps them to a higher-dimensional space, where the data points become more separable.

3-5 Naïve Bayes classifier:

The NB classifier, short for Naïve Bayes classifier [18], is a supervised learning algorithm based on Bayesian theory. It is commonly employed to address classification problems, particularly in text classification scenarios involving high-dimensional training datasets. The Naïve Bayes classifier is recognized as one of the simplest and most effective classification algorithms, enabling the creation of fast machine learning models capable of making quick predictions. As a probabilistic classifier, it predicts outcomes based on the probability of an object's existence. There are three types of NB classifiers:

1) Gaussian: The Gaussian model assumes that the features in the dataset follow a normal distribution. This implies that if the predictors take continuous values instead of discrete values, the model assumes that these values are sampled from a Gaussian distribution.

2) Multinomial: The Multinomial Naive Bayes classifier is utilized when the data is distributed with multiple categories or classes. It is commonly employed in document classification tasks, where a document is assigned to a specific category such as sports, politics, education, etc. This classifier leverages word repetition as a basis for predictions.

3) Bernoulli: The Bernoulli classifier operates similarly to the Multinomial classifier, but it considers predictors as independent logical variables, such as whether a particular word is present or absent in a document. This model is also commonly used in document classification tasks.

The most influential parameter in NB classifier is:

- **binarize:** is a technique used to convert continuous feature values into binary values. This technique is used when the features are not binary and have continuous values. Binarization is done by selecting a threshold value and assigning a value of 0 or 1 to each feature based on whether it is below or above the threshold value.

3-6 Logistic Regression classifier:

Logistic Regression is a widely used machine learning algorithm that falls under the supervised learning technique [19]. Its primary purpose is to predict a dependent categorical variable based on a specific set of independent variables. Unlike linear regression, which is used for regression problems, logistic regression is specifically designed for classification problems. It predicts the output of a categorical dependent variable, resulting in a discrete or categorical value. This value can be binary, such as yes or no, 0 or 1, true or false, and so on. However, instead of providing exact values of 0 or 1, logistic regression produces probability values ranging between 0 and 1. These probabilities indicate the likelihood of a particular outcome. Logistic regression shares similarities with linear regression but differs in its application and purpose.

Parameters that affect the Logistic Regression algorithm:

− **Penalty**: It controls the amount of regularization applied to the model. Regularization helps to prevent overfitting and improve the generalization ability of the model.

− **Solver**: It specifies the optimization algorithm used to find the optimal weights for the logistic regression model. Different solver algorithms have different strengths and weaknesses, and the choice of solver algorithm can affect the performance of the model.

4- Dataset_sdn:

The dataset was created using the Mininet simulator for the purpose of classifying traffic using machine learning and deep learning algorithms [4]. Ten topologies were created in Mininet, each containing switches connected to a Ryu controller, which is one of the most popular controllers used in SDN networks and is built using Python. The topologies contain normal traffic for TCP, UDP, and ICMP protocols, as well as attack traffic for the three protocols: TCP Syn, UDP Flood, and ICMP Flood. The dataset consists of 23 features, some of which were extracted from the switches and the rest were calculated. The network simulation lasted for 250 minutes and contains 104345 data samples. We noted the dataset components in table1.

Table 1: Dataset components

	Feature	Description
Extracted Features	Switch ID	The ID of switch that sends the packet to controller
	Packet count	Number of packets sent from switch to controller
	Byte count	Number of bytes sent from switch to controller
	Duration	Duration in seconds and nanoseconds,
	Source IP	Sender of packets IP
	Destination IP	Receiver of packets IP
	Port number	Defines the port used in the communication
	Transmitted & received bytes	Define the number of bytes at the switch
	Date and time	Date/time for sending/receiving a packet
calculated	pktperflow	the number of packets in a flow
	byteperflow	the number of bytes in a flow
	pktrate	the number of packets sent per second and is calculated by dividing pktperflow by the 30-second time interval
	number of packet_i	represents the total number of entries in the switch,
	tx_kbps and rx_kb	represent the transmitted and received data rates, respectively
	PortBandwidth	the sum of tx_kbps and rx_kbps
	Label	which indicates whether the traffic is normal (0) or attack (1).

5- Evaluation Metrics:

The dataset was normalized using the Scalar data processor. Scalar values were calculated mathematically according to the following equation [20]:

$$x_{new} = \frac{x_{old} - mean(x)}{standard\ deviation(x)} \qquad (1)$$

For the Naïve Bayes classifier, the Min-Max data processor was used to handle transformations within the dataset. Standardization was used within the range of [0, 1] using the following equation [20]:

$$x' = \frac{x - min(x)}{\max(x) - \min(x)} \qquad (2)$$

Performance and efficiency are evaluated in many cases using a set of metrics, including:

- True Positive (TP): the number of attack records correctly classified
- True Negative (TN): the number of normal records correctly classified
- False Positive (FP): the number of normal records incorrectly classified
- False Negative (FN): the number of attack records incorrectly classified

For the evaluation purpose, Accuracy and F1_score (F1) metrics are applied. These metrics are calculated as follows [21]:

- Accuracy: shows the percentage of true detection over total predictions made by the model:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (3)$$

- Precision: shows how many attacks predicted are actual attack:

$$Precision = \frac{TP}{TP + FP} \times 100\% \qquad (4)$$

- Recall: shows the percentage of predicted attacks versus all attack presented:

$$Precision = \frac{TP}{TP + FN} \times 100\% \qquad (5)$$

- F1_score: gives a better measure of the accuracy by considering both the precision and the recall

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \times 100\% \qquad (6)$$

- Training time: the time required for the classifier to fit the model

Results and Discussion:

Classifiers performance

1) Decision Tree Classifier:

Initially, the algorithm was applied with default parameters on the entire dataset, which contains 22 features, and the algorithm gave an accuracy of 99.98% and an F1 score of 99.98%, and the training time of the algorithm was only 0.43 seconds. We reduced the features to the top 3 features within the algorithm using the feature_importance property, and the most important 3 features were bytecount, packetins, and byteperflow. The algorithm was also applied with default parameters, and it gave an accuracy of 99.96%, which decreased by 0.0002, while the F1 score was 99.97%, which decreased by 0.0001. The training time became 0.075 seconds, which decreased by 0.355 seconds. We then worked on adjusting the parameters of the DT algorithm after reducing the number of features, and we were able to obtain an accuracy of 99.99% and an F1 score of 99.99%, which is similar to the results obtained when working on the entire dataset, but with a clear difference of 0.82 seconds, and the training time required was 0.018 seconds, which decreased by 0.412 seconds. During the study, it was found that the parameters that affect the algorithm's results are criterion with a value of entropy, splitter = random, and random_state = 33, while the parameters that negatively affect the performance of the DT classifier are min_weight_fraction_leaf. In conclusion, the DT classifier can be used with 3 features with precise parameter tuning to achieve an accuracy of 99.99% and an F1 score of 99.99% with a very low training time of around 0.018 seconds.

2) Random Forest Classifier:

Initially, we applied the algorithm with default parameters on the entire dataset, which contains 22 features, and the algorithm gave an accuracy of 99.98% and an F1 score of 99.98%, and the training time of the algorithm was 6.57 seconds. Using the featureselection class, we reduced the features to 6 features, which are pktcount, bytecount, packetins, pktperflow, byteperflow, and pktrate, out of 22 features, and the accuracy and F1 score were 100%. This is due to the fact that the RF algorithm in the default state builds 100 trees, and for a low number of features, the accuracy is perfect, which is an overfitting case, meaning that all samples were classified correctly. The algorithm took 3.57 seconds to train. We then worked on reducing the processing time by reducing the number of trees from 100 to 20, and the accuracy and F1 score were 100%, and the training time was 0.71 seconds, which decreased by 2.86 seconds. When obtaining all parameter values of 1, this means that there is an overfitting case, which can be solved by controlling the maximum depth parameter (max_depth). It was found that leaving it at the default value of none or using values of 15 and above will result in an overfitting case, but the best accuracy was obtained for a value of 14, where the accuracy and F1 score were 99.98%, and the training time was 0.67 seconds. During the study, it was found that the parameters that negatively affect the performance of the RF classifier are min_weight_fraction_leaf and min_impurity_decrease. In conclusion, the RF classifier can be used with 6 features with precise parameter tuning to eliminate overfitting and achieve the highest possible accuracy for a depth of 14, which is the threshold for overfitting. This can result in an accuracy of 99.98% and an F1 score of 99.98% with a training time of 0.67 seconds.

3) Extra Tree Classifier:

Initially, the algorithm was applied to the entire dataset and achieved an accuracy of 99.97% and an f1-score of 99.98%. The algorithm took approximately 4 seconds to train, which is about 2.5 seconds faster than the RF algorithm. Similar to the RF algorithm, the featureSelection class from the sklearn library was applied, and we noticed that when the algorithm chose 7, 8, or 9 features from the dataset, it was the best for the classifier. However, after feature reduction, the algorithm suffered from overfitting. When the process was repeated with max_depth = 4, the algorithm chose 8 features: pktcount, bytecount, dur, tot_dur, pktperflow, byteperflow, pktrate, and protocol. Overfitting was eliminated, and all features were removed from the dataset, leaving only the 8 features. The results for the default classifier parameters were: acc=99.93%, which decreased by 0.0004, and f1-score=99.95%, which decreased by 0.0003. The training time for the algorithm decreased to 1.87 seconds .

We then worked on improving the classifier performance by studying the classifier parameters. We started with the n_estimator parameter, which represents the number of trees (default value is 100), and reduced it to 20. The accuracy increased to 99.94%, and the f1-score increased to 99.95%. The training time for the algorithm decreased to 0.38 seconds, a 1.49-second improvement. We also noticed an inverse relationship between the n_estimator parameter and the max_depth parameter. If the number of trees is high (e.g., 100), the value of

max_depth should be reduced; otherwise, overfitting occurs. However, when the number of trees was reduced to 20 and the maximum depth was increased to 75, we achieved an accuracy of 99.97% and an f1-score of 99.97%, with a training time of 0.36 seconds. The parameter with a negative impact on the classifier was min_weight_fraction_leaf, where increasing it by 0.1 from its default value of 0.0 resulted in an acc=73.09% and f1-score=81.84% .
In conclusion, the ETC classifier with 8 features can be used with precise parameter tuning to eliminate overfitting and achieve the highest possible accuracy of 99.97% and f1-score of 99.97%, with a training time of 0.36 seconds.

4) Logistic Regression Classifier:

The default LR classifier was applied to the dataset, and an accuracy of 77.17% and an f1-score of 81.81% were achieved, with a training time of 0.39 seconds. When the features used by the algorithm were removed, the accuracy decreased significantly to 69.20%, a decrease of 0.0797, and the training time decreased to 0.04 seconds. The LR classifier depends on a set of parameters, the most important of which are the penalty and solver parameters. All cases did not improve the accuracy, but when penalty=l1 and solver=sage were used, the worst training time was 326.51 seconds, and the best performance of the classifier in terms of time was for penalty=l2 and solver=lbfgs, with a training time of 0.35 seconds.
The parameter that affects the accuracy the most is the C parameter, which has a default value of 1 and adjusting it affects the classifier evaluation parameters. When its value was raised to 100, the accuracy improved by 0.0028, reaching 77.45%. The rest of the classifier parameters did not significantly affect the accuracy. The LR classifier gave worse performance than all tree-based classification algorithms and the SVM classifier, but its performance was better than some Naïve Bayes algorithms.

5) Naïve Bayes Classifier:

All three types of the NB algorithm were applied to the entire dataset, and the following results were obtained:

Table 2: NB algorithms comparison

Algorithm	Accuracy	F1-score	Training time
Gaussian NB	65.98%	71.62%	0.037
Bernoulli NB	83.76%	87.66%	0.034
Multinomial NB	64.36%	69.26%	0.013

In the GaussianNB classifier, adjusting its parameter values did not affect either metric. In the BernoulliNB classifier, there is a parameter that affects both metrics, which is binarize, which has a default value of 0. When its value was raised to 1, the accuracy increased from 61.26% to 83.67%, with an improvement of 0.2241, and the f1-score improved from 63.93% to 87.66%, with an improvement of 0.2373. In the MultinomialNB classifier, MinMaxScalar scaling was used because this classifier rejects negative values in the data it is trained on. There is also a fit_prior parameter that affects the metrics. When its value was changed to True, the accuracy increased from 59.35% to 64.36%, with an increase of 0.0501, while the f1-score decreased from 70.13% to 69.26%, with a decrease of 0.0087. It can be concluded that with the correct parameter settings, the BernoulliNB classifier was the best type of NB classifier among them.

6) Support Vector Classifier:

After applying the SVM algorithm to the dataset with all available features, we achieved an accuracy of 97.52% and an F1 score of 97.96%, with a training time of 62.32 seconds. Comparing it to the previous algorithms, the SVC classifier achieved higher accuracy than

Naïve Bayes and Logistic Regression classifiers. However, it required a longer training time compared to these classifiers.
After starting the normalization process, we observed the following:
-The regularization parameter (C) has a direct impact on the performance metrics. When decreasing its value to 0.1, the accuracy decreased to 95.73% and the F1 score decreased to 96.47%, while the training time increased to 101.02 seconds. When further decreasing C to 0.01, the accuracy dropped to 92.66% and the F1 score dropped to 93.97%, while the training time increased to 296.18 seconds. It seems that the minimum value for C is not suitable. On the other hand, increasing C to 10 resulted in an accuracy of 98.69%, an F1 score of 98.93%, and a training time of around 69 seconds. Increasing C to 100 further improved the accuracy to 99.21%, the F1 score to 99.35%, with a training time of around 76 seconds. Increasing the value of C seems to improve the classifier's performance without significantly affecting the training time.
-The choice of the kernel also significantly affects the classifier's performance. Using a linear kernel with non-linearly distributed data leads to very high training time. Therefore, we tried the sigmoid and rbf kernels. We observed that the best accuracy of 99.21% was achieved for C=100 and kernel=rbf. The highest F1 score of 99.35% was also achieved for C=100 and kernel=rbf. On the other hand, the worst accuracy of 58.16% was obtained for C=100 and kernel=sigmoid, and the lowest F1 score of 65.65% was obtained for C=100 and kernel=sigmoid. The worst training time of 463.35 seconds was observed when using kernel=sigmoid and C=0.01, while the training time was 62.32 seconds when using kernel=rbf and C=1.
Therefore, it seems that increasing the value of the regularization parameter (C) can improve the classifier's performance without significantly affecting the training time. Additionally, choosing the appropriate kernel is crucial, and in this case, the rbf kernel outperformed the sigmoid kernel in terms of accuracy and F1 score.

Classifiers performance

The following table shows a comparison between the algorithms used in the study and the values obtained for each algorithm in the best performance:

Table 3: classifier comparison

Algorithm	Accuracy	F1-score	Training Time (sec	Number of features
DT	99.99%	99.99%	0.018	3
RF	99.98%	99.98%	0.67	6
ETC	99.97%	99.97%	0.36	8
LR	77.45%	81.81%	0.39	21
SVC	99.21%	99.35%	76	21
GaussianNB	65.98%	71.62%	0.037	21
BernouliNB	83.76%	87.66%	0.034	21
MultinomialNB	64.36%	69.26%	0.013	21

In summary, the table provides a comparison of the algorithms used in the study, showcasing their performance in terms of accuracy, F1-score, training time, and the number of features used.
Among the algorithms, Decision Tree (DT), Random Forest (RF), and Extra Trees Classifier (ETC) achieved the highest accuracy and F1-scores, with values close to 99.99%. These algorithms also demonstrated very low training times, ranging from 0.018 to 0.67 seconds. However, they utilized a smaller number of features, ranging from 3 to 8. Logistic Regression (LR) achieved a relatively lower accuracy of 77.45% and an F1-score of 81.81%, with a moderate training time of 0.39 seconds. It utilized all 21 features available. Support Vector Classifier (SVC) achieved an accuracy of 99.21% and an F1-score of 99.35%, but with a

higher training time of 76 seconds. It also utilized all 21 features. Gaussian Naive Bayes (GaussianNB), Bernoulli Naive Bayes (BernouliNB), and Multinomial Naive Bayes (MultinomialNB) achieved lower accuracies and F1-scores, ranging from 64.36% to 83.76%. These algorithms had relatively low training times, ranging from 0.013 to 0.037 seconds, and utilized all 21 features. Overall, the decision tree-based algorithms (DT, RF, ETC) performed exceptionally well in terms of accuracy and F1-score, with faster training times and fewer features used compared to logistic regression and the Naive Bayes algorithms.
The chart illustrates a comparison between algorithms in terms of accuracy

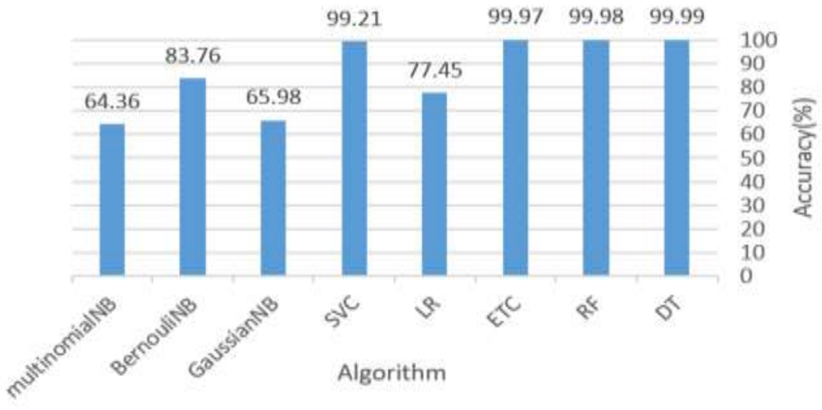

Fig-5: Accuracy of ML Algorithm

The table clearly shows that tree-based classifiers (DT, RF, ETC) achieved high accuracy results ranging from 99.97% to 99.99%. It is also worth noting that the SVC classifier achieved a similar level of accuracy. However, the lowest accuracy was observed for the MultinomialNB classifier, which scored 64.36%. On the other hand, one of the Bayes classifiers, BernouliNB, performed better than logistic regression with an accuracy of 83.76%. In summary, the tree-based classifiers achieved high accuracy results ranging from 99.97% to 99.99% compared to other types of classifiers. The SVC classifier also achieved a close accuracy level. The worst accuracy was observed for the MultinomialNB classifier, but one of the Bayes classifiers, BernouliNB, performed better than logistic regression.
When comparing based on the f1_score, we obtained the same evaluation where tree-based classifiers outperformed the other classifiers and the worst performance was observed for the multinomial classifier.

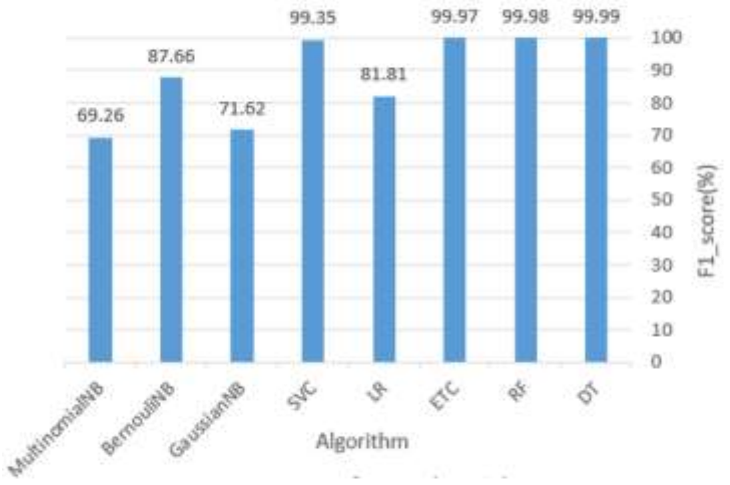

Fig-6: F1_Score of ML Algorithm

Regarding training time, the worst time was for the SVC classifier with a time of 76 seconds, while all other classifiers achieved very low times. The best time was for the multinomial classifier, followed by the DT classifier.

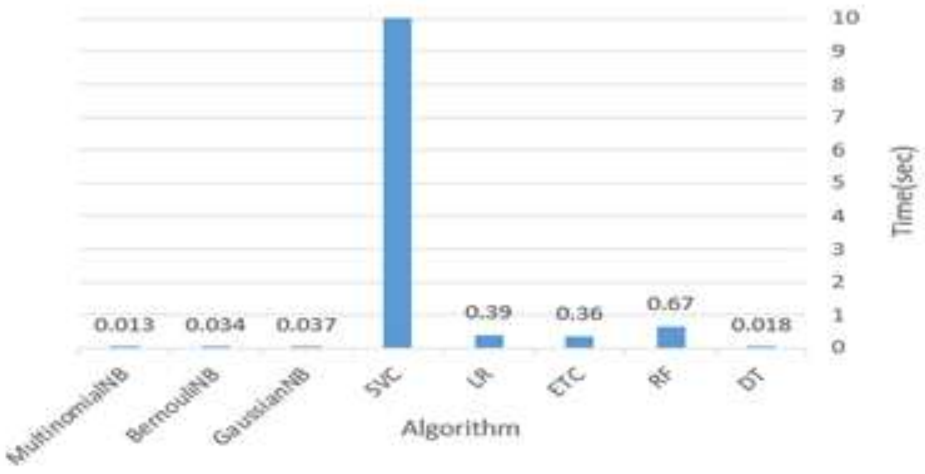

Fig-7: Training Time of ML Algorithm

Conclusion and Recommendations:

In terms of tree-based algorithms, the DT classifier is considered the fastest because it relies on building only one decision tree. The parameter min_weight_fraction_leaf has a negative impact on the performance metrics of tree-based algorithms and should be left at a value of 0, as changing it can result in a decrease in classifier performance evaluation metrics. In the ETC classifier, both the number of trees and the depth are inversely proportional and give different results in each execution. Its concept is based on building a random forest with additional trees within the branches of each tree in the forest. The features bytecount and byteperflow are common and relied upon by the DT, RF, and ETC classifiers. The tree-based classification algorithms achieved high accuracy and low training times. They performed better than the various types of NB classifiers and outperformed the LR and SVC classifiers, making them the best choice for the dataset. Among the NB classifiers, the BernouliNB classifier achieved the best performance with a low training time. The SVC classifier outperformed the NB classifiers in terms of accuracy but requires a longer training time compared to them. In the DT classifier, it is possible to reduce the number of features to 3 out of the original 21 without affecting the results or training time. Based on the above, we recommend using tree-based classifiers when studying DDoS attacks in SDN networks as they provide the best possible accuracy with the lowest training time suitable for real-time applications.

References:

[1] B.Smitha, E.Zoraida and G.Indumathi, A Comparative Study on Software-Defined Network with Traditional Networks, TEM Journal. Volume 13, Issue 1, pages 167-176, ISSN 2217-8309, DOI: 10.18421/TEM131-17, February 2024.

[2] M. Setitra, M.Fan, I.Benkhaddra and .Bensalem, DoS/DDoS attacks in Software Defined Networks: Current situation, challenges and future directions, Computer Communications, Volume 222, Pages 77-96, 1 June 2024

[3] F. Shradha, G. Rutuja, C.Sakshi, A.Khushi and K.Srushti, Detection of cyber-attacks and network attacks using Machine Learning, World Journal of Advanced

Engineering Technology and Sciences, 2024, 12(01), 128–132 Publication history: Received on 28 March 2024; revised on 17 May 2024; accepted on 20 May 2024 Article DOI: https://doi.org/10.30574/wjaets.2024.12.1.0184

[4] https://www.kaggle.com/datasets/aikenkazin/ddos-sdn-dataset, last visit 11-Oct-24

[5] https://www.python.org/, last visit 11-Oct-24

[6] https://scikit-learn.org/stable/, last visit 11-Oct-24

[7] https://matplotlib.org/, last visit 11-Oct-24

[8] https://pandas.pydata.org/, last visit 11-Oct-24

[9] https://www.spyder-ide.org/, last visit 11-Oct-24

[10] https://colab.research.google.com/, last visit 11-Oct-24

[11] B. Bouyeddou, F. Harrou, Y. Sun, and B. Kadri; “Detecting SYN flood attacks via statistical monitoring charts: A comparative study"; in Proc. 5th Int. Conf. Electr. Eng. Boumerdes (ICEE-B), Boumerdes, Algeria, October 2017.

[12] M. S. Khaing, Y. M. Thant, T. Tun, C. S. Htwe, and M. M. S. Thwin; “IoT botnet detection mechanism based on UDP protocol"; in Proc. IEEE Conf. Comput. Appl. (ICCA), Yangon, Myanmar, February 2020.

[13] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, “Poseidon: Mitigating volumetric DDoS attacks with programmable switches"; in Proc. Netw. Distrib. Syst. Secur. Symp., 2020

[14] B. Jijo and A.Abdulazeez, Classification Based on Decision Tree Algorithm for Machine Learning, Journal OF APPLIED SCIENCE AND TECHNOLOGY TRENDS,Vol. 02, No. 01, pp. 20 – 28 (2021) ISSN: 2708-0757

[15] A.Cutler, D.Cutler and J.Stevens, Random Forest, publication at: https://www.researchgate.net/publication/236952762, January 2011 DOI: 10.1007/978-1-4419-9326-7_5 · Source: DBLP

[16] A.Sharaff and H.Gupta, Extra-Tree Classifier with Metaheuristics Approach for Email Classification, Advances in Computer Communication and Computational Sciences (pp.189-197), May 2019 DOI:10.1007/978-981-13-6861-5_17

[17] M. Hossain, Support Vector Machine, publication at: https://www.researchgate.net/publication/365892847, December 2022

[18] M. Alahmar, Naïve Bayes Algorithms, publication at: https://www.researchgate.net/publication/369067312, February 2023 DOI: 10.13140/RG.2.2.15378.73921

[19] S. Sperandei, Understanding logistic regression analysis, publication at: https://www.researchgate.net/publication/260810482, February 2014 DOI: 10.11613/BM.2014.003 · Source: PubMed

[20] https://www.javatpoint.com/data-preprocessing-machine-learning, last visit 11-Oct-24

[21] P. Singh, S. Jha, S. Nandi and S. Nandi; “ML-Based Approach to Detect DDoS Attack in V2I Communication Under SDN Architecture “; in “ 2018 IEEE Region 10 Conference“.