JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 9, No. 3, September 2024, Pp. 1180-1197

COMPARISON OF MACHINE LEARNING TECHNIQUES FOR CLASSIFICATION OF DISTRIBUTED DENIAL OF SERVICE ATTACKS BASED ON FEATURE ENGINEERING IN SDN-BASED NETWORKS

Muhammad Ikhwananda Rizaldi*[1], Didih Rizki Chandranegara[2], Denar Regata Akbi[3]
1. Universitas Muhammadiyah Malang, Indonesia
2. Universitas Muhammadiyah Malang, Indonesia
3. Universitas Muhammadiyah Malang, Indonesia

Article Info
Keywords: Correlation-Based Feature Selection; DDoS Attacks; Feature Engineering; Machine Learning; Software-Defined Networking

Article history:
Received 7 June 2024
Revised 21 July 2024
Accepted 14 August 2024
Available online 1 September 2024

DOI :
https://doi.org/10.29100/jipi.v9i3.5262

* Corresponding author.
Muhammad Ikhwananda Rizaldi
E-mail address:
aldirizaldy977@webmail.umm.ac.id

ABSTRACT
Distributed Denial-of-Service (DDoS) attacks present a noteworthy cybersecurity hazard to software-defined networks (SDNs). This investigation presents an approach that depends on feature engineering and machine learning to discern DDoS attacks in SDNs. Initially, the dataset acquired from Kaggle goes through cleansing and normalization procedures, and the optimal subset of features is determined by employing the Correlation-based Feature Selection (CFS) algorithm. Subsequently, the optimal subset of features is trained and evaluated utilizing diverse Machine Learning algorithms, specifically Random Forest (RF), Decision Tree, Adaptive Boosting (AdaBoost), K-Nearest Neighbor (k-NN), Gradient Boosting, Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), and Categorical Boosting (CatBoost). The outcomes demonstrate that XGBoost outperforms the other algorithms in various performance metrics (e.g., accuracy, precision, recall, F1, and AUC values). Furthermore, a comparative analysis was carried out among various models and algorithms, revealing that the technique proposed by the researchers yielded the most favourable outcomes and effectively detected and identified DDoS attacks in SDN. Consequently, this investigation provides a novel perspective and resolution for SDN security.

.

I. INTRODUCTION

The landscape of network infrastructure has experienced a significant transformation due to the rapid advancement and widespread use of state-of-the-art technologies like cloud computing and big data. This transformation has resulted in a tremendous increase in network traffic and an unparalleled dependence on networks for various purposes. It is important to note that 2020 witnessed an extraordinary global crisis in the form of the COVID-19 pandemic, which compelled individuals and organizations to rely heavily on online platforms for work, education, and entertainment. As a result, this surge in online activity placed immense pressure on the stability and security of networks, thereby exacerbating the challenges associated with conventional networking models. It is becoming increasingly evident that these traditional models must be adequately equipped to effectively meet users' diverse and complex needs in today's era. In light of this urgent demand for innovation and adaptability, Software-Defined Networking (SDN) has arisen as the most favored and desired networking technology owing to its incomparable adaptability, programmability, dynamism, and simplicity. SDN embodies a fundamental alteration in the manner in which networks are administered and operated, granting organizations the capability to address the constantly evolving requirements of the contemporary digital environment in a proficient and efficacious manner [1]. The architectural configuration of SDN distinguishes between the control plane and the data plane, which encompasses the application, management, and data planes. The data plane comprises switches and routers that are programmed and managed by the control plane. The interaction between the control plane and the controller is facilitated through the northbound interface. The control plane governs the operation of network devices in the data plane, with the controller functioning as the decision-making element of the network center. The application plane comprises various SDN applications that fulfill users' needs. These applications communicate with the SDN controller through the northbound interface, enabling the transmission of requests for programmable network behavior to the controller. By possessing global control over the network topology, the controller alleviates switches and routers from the burden of intricate traffic processing, thereby augmenting scalability, controllability, and programmability.

This method significantly enhances the control of network traffic and the effective use of resources. However, ensuring network security continues to be a vital issue in the implementation of SDN applications. The separation of the control and data planes makes SDN vulnerable to Distributed Denial of Service (DDoS) attacks. Resiliency to such attacks is a fundamental weakness in most SDN frameworks. Attackers exploit numerous fraudulent requests to overwhelm the target host's resources and disrupt the provision of regular services. These targeted and covert attacks are cost-effective, making them an increasingly potent threat [2]. DDoS attacks in SDN applications are a growing concern due to their potential to disrupt network services.

The attackers are using more sophisticated methods, making it harder to detect Distributed Denial of Service (DDoS) attacks. An attack on the controller can cripple its ability to handle requests, leading to network breakdown. In contrast, an attack on the data plane can cause a massive increase in superfluous traffic, consuming excessive network bandwidth and processing resources. This influx not only disrupts normal traffic but also forces the Software Defined Networking (SDN) controller to create unnecessary flow tables for routing. This depletes the SDN switch's storage and adds strain to the data plane. Thus, effectively detecting and countering DDoS attacks is increasingly critical in SDN research [3]. The prevailing techniques for identification are commonly divided into two separate classifications: approaches centered around statistical analysis and approaches grounded in machine learning [4]. For detection methods based on statistical analysis, scrutinizing particular data from network or application layers is key to identifying Distributed Denial of Service (DDoS) attacks. These strategies create an expected behavior or traffic pattern, spotting irregularities caused by the attack. While these methods are simple and generally effective, they struggle with novel attack types and necessitate individual manual setup for each distinct attack [5]. In contrast, DDoS detection methods using machine learning leverage sophisticated techniques. They apply machine learning algorithms to analyze standard network traffic, identify anomalies, and automatically detect DDoS attacks. This method is adaptable, capable of handling new attack types and revealing hidden patterns in intricate network environments. However, in order to train and test algorithms, they require large datasets and significant computational capabilities. By leveraging SDN technology, these approaches based on machine learning facilitate efficient identification and reduction of DDoS attacks. These methods are characterized by automated procedures, swift responses, and the utilization of deep learning in safeguarding network systems [6].

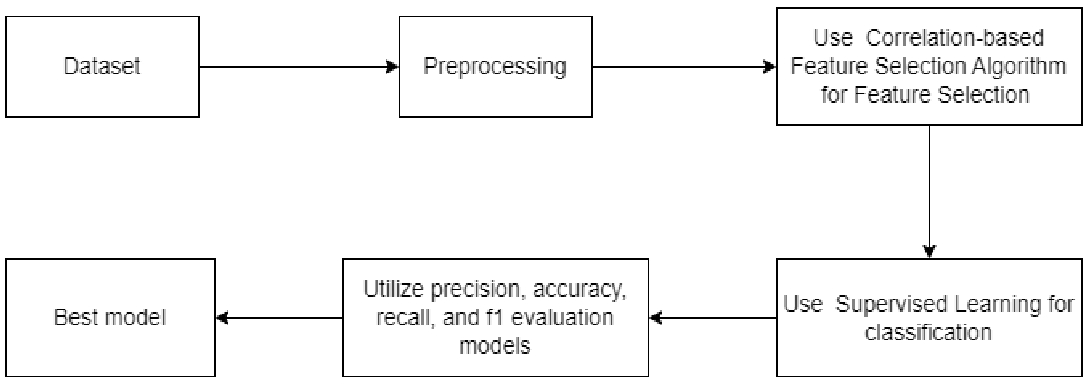
The process of feature engineering plays a critical role in the development of machine learning models. Its main objective is to carefully select pertinent information while discarding redundant or insignificant data [7]. Consequently, effectively managing datasets presents a significant challenge for researchers. The quality and relevance of the data and features directly impact the performance of machine learning systems. Unlike models and algorithms that prioritize achieving the highest possible threshold, the key to success lies in obtaining high-quality datasets to improve the precision of intrusion detection [8]. In the realm of feature selection, numerous researchers have employed established algorithms in their work to accomplish this goal, Polat et al. [9] procured SDN traffic data in both normal and DDoS attack scenarios in order to construct their dataset. They employed filter-based, wrapper-based, and embedded feature selection techniques to discern crucial features. Subsequently, they conducted training and testing of multiple classifiers, encompassing Support Vector Machines (SVM), Naive Bayes (NB), Artificial Neural Networks (ANN), and K-Nearest Neighbors (k-NN). Their findings revealed that the k-NN classifier exhibited the highest level of accuracy, achieving a success rate of 98.3% in detecting DDoS attacks, surpassing the performance of the other classifiers. Beitollahi et al. [10] successfully merged radial basis function (RBF) neural networks with the cuckoo search (CS) algorithm in order to identify and classify DDoS attacks. Initially, a genetic algorithm (GA) was employed to determine the most optimal feature subset. Subsequently, the RBF neural network was trained using this subset alongside the CS optimization algorithm. Comparative analysis was then conducted, contrasting their approach with other established methods such as k-nearest neighbors (k-NN), Bootstrap aggregation, support vector machine (SVM), multilayer perceptron (MLP), and recurrent neural network (RNN). The findings clearly demonstrated the superiority of their method in effectively detecting and managing DDoS traffic. Mishra et al. [11] directed their attention towards the classification of DDoS attacks and implemented a range of training and prediction techniques. The Cicdos2019 dataset underwent thorough data cleaning and transformation procedures as part of their research. In order to determine the most significant attributes, they employed the Extra Tree Classifier and successfully identified 25 key features. Their investigation encompassed the application of six distinct machine learning algorithms, with the AdaBoost Classifier emerging as the most accurate, boasting an exceptional accuracy rate of 99.87%. A strategic framework that integrates feature engineering and machine learning techniques was formulated by Aamir et al. [12]. They utilized statistical tests such as the t-statistic, Chi2 test, and information gain for the purpose of identifying significant features from the dataset. Additionally, five supervised machine learning algorithms were evaluated by them across three distinct datasets. Their findings highlighted the k-NN algorithm as the most optimal choice overall. Maheshwari et al. [13] developed a testing framework utilizing Mininet, POX controllers, and diverse datasets. They presented a novel hybrid meta-heuristic optimization (BHO) algorithm to identify the optimal feature set. Additionally, their investigation encompassed an evaluation of an ensemble approach involving six principal classifiers, consisting of two Support Vector Machines (SVMs), two Random Forests, and two Gradient Boosting Machines. Akgun et al. [14] utilized the CIC-IDDOS2019 dataset and applied an algorithm for attribute evaluation based on information gain to identify 40 significant features. Subsequently, they executed a framework employing a Convolutional Neural Network (CNN) that incorporated a one-dimensional convolutional layer with the specific purpose of detecting instances of Distributed Denial-of-Service (DDoS) attacks. Karatas et al. [15] employed the Synthetic Minority Oversampling Technique (SMOTE) in order to address the issue of imbalanced datasets. By utilizing SMOTE, they were able to generate synthetic data with the aim of enhancing the representation of minority classes and achieving a more balanced dataset size. Subsequently, the researchers proceeded to evaluate the performance of various classification algorithms, namely k-NN, Random Forest (RF), Gradient Boosting, AdaBoost, Decision Tree, and Linear Discriminant Analysis, for the purpose of classification tasks. The outcomes of their experiments demonstrated a noteworthy increase in the detection rate of rare intrusions resulting from the application of this method.

Correlation-based feature selection (CFS) is a machine learning technique used to identify the most relevant features within a given dataset. Its primary goal is to detect and eliminate irrelevant or redundant features, thereby reducing the dataset's dimensionality and improving the performance of classification algorithms. This approach proves particularly beneficial when dealing with high-dimensional datasets, as it helps reduce computational complexity, runtime, and storage demands. Correlation-based feature selection entails assessing the correlation between attributes and class labels, as well as between attributes themselves. The selected features exhibit strong correlations with the target variable while demonstrating minimal correlations with other features [16].

The research concentrates on classifying DDoS attacks using machine learning within SDN-based networks. It utilizes the Correlation-based Feature Selection (CFS) algorithm to select the most relevant features from network traffic data, aiding in the creation of more effective attack detection models. The study leverages the adaptability and programmability of SDN to improve the precision and efficiency of DDoS attack classification [17].

II. RESEARCH METHODOLOGY

This investigation employs information acquired via the Kaggle website [18]. This study used an extensive dataset with 1,188,333 entries, covering details on DDoS and website attacks. The dataset includes five types of data packets: Benign, DDoS, Web Attack Brute Force Traffic, Web Attack XSS Traffic, and Web Attack SQL Injection, featuring a total of 79 attributes. The main goal is to develop a framework for detecting DDoS attacks, enriched by the core concepts of machine learning in SDN networks. The development of this approach is illustrated in Figure 1 of the manuscript.


Dataset
Preprocessing
Use Correlation-based Feature Selection Algorithm for Feature Selection
Use Supervised Learning for classification
Utilize precision, accuracy, recall, and f1 evaluation models
Best model
Figure 1. Research flow diagram

In the initial phase, the dataset undergoes an initial preprocessing procedure. Following this, the CFS algorithm is utilized to choose the most optimized subset of features. Afterwards, this selected subset of features is employed for the purposes of training and testing using eight supervised learning classifiers, specifically Random Forest (RF), Decision Tree, AdaBoost, k-Nearest Neighbor (k-NN), Gradient Boosting, XGBoost, LightGBM, and CatBoost algorithms. These classifiers are utilized to train and test the chosen feature subset. The classifier that exhibits the highest level of effectiveness is determined as the optimal classifier within this specific set.

A. Data Preprocessing and Feature Extraction

1) Dataset Selection

A specific dataset, comprising records of various types of network traffic data such as DDoS, XSS Intrusions, Brute Force Intrusions, SQL Injection, and benign traffic, has been selected. This chosen dataset contains a total of 1,188,333 rows, which include network intrusion observations and logged traffic. Additionally, it encompasses 79 features. The distribution of traffic types within this dataset can be described as follows: benign traffic accounts for 798,322 observations, DDoS traffic represents 383,439 observations, Brute Force Web attack traffic is observed in 45,500 instances, XSS Web attack traffic is documented in 1,962 instances, and SQL Injection Web attack traffic is recorded in 60 instances. These statistical details are visually depicted in Figure 2.

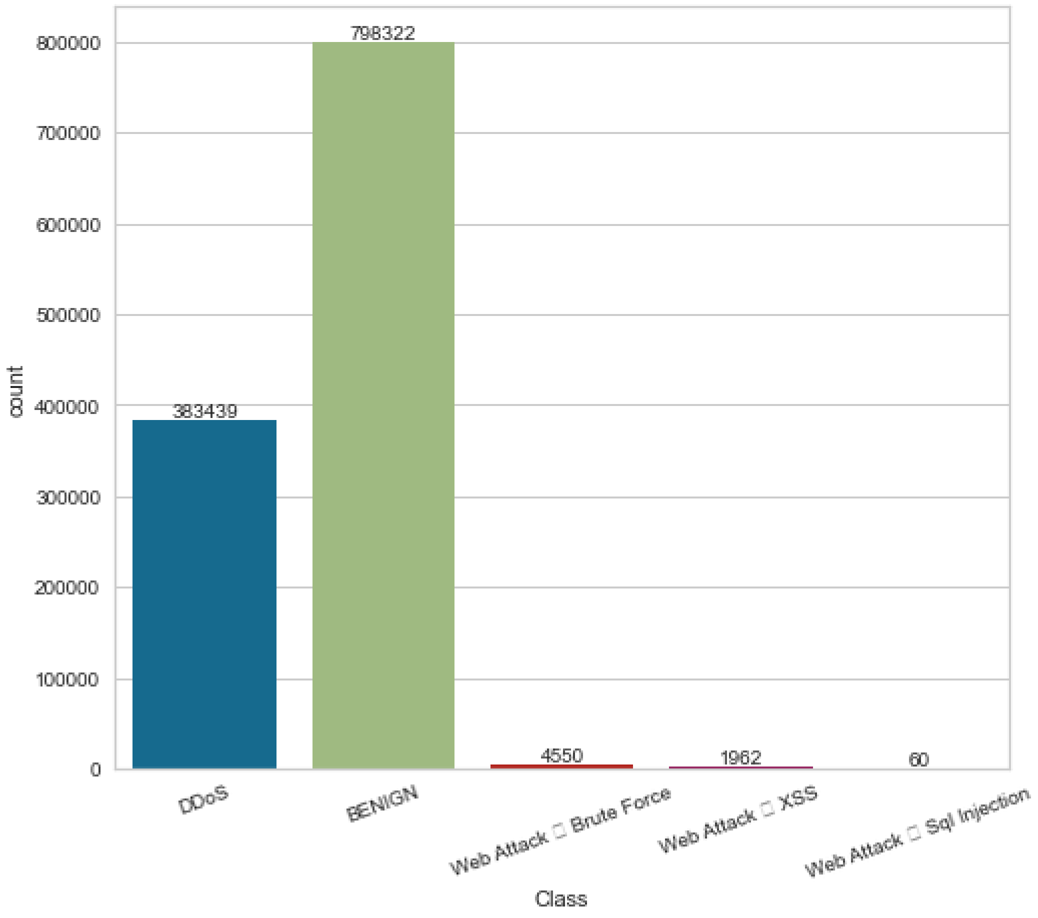


Figure 2. Distribution of data types in the dataset

2) Data Cleaning for Datasets

During the phase of preparing the dataset, it is not uncommon to come across various issues related to the quality of the data, such as values that are not a number (NaN), outliers, and duplicate entries. The process of cleaning the data plays a crucial role in the preprocessing of the data, with the aim of minimizing any potential negative impacts that may arise from these issues and ultimately maintaining the quality of the dataset. In this particular study, the researcher made the decision to remove all instances that contained values that were empty or infinite, and also excluded features that showed no correlation with the target variable. As a result of these efforts to clean the data, the total number of samples was reduced from 1,188,333 to 365,725, which involved the removal of 822,608 samples. Additionally, the number of features was reduced from 79 to 26.

3) Data Transformation

To accommodate the variations in magnitude present in the dataset, this study utilizes the MinMax scaling technique in order to standardize all of the features. The process of MinMax scaling entails applying a linear transformation to the original data. By denoting the initial data as x and the transformed data as x', the formula for scaling can be succinctly expressed as follows:

$$x' = \frac{(x - min)}{(max - min)} \tag{1}$$

In this context, the terms '*min*' and '*max*' represent the minimum and maximum values observed within the column where the variable 'x' is situated. The *MinMax* scaling process finds extensive applications in data standardization. Through this method, the data is transformed to fit within the numerical range [0,1], preserving the original data structure. This property sets it apart from the Z-Score standardization method. Consequently, *MinMax* scaling enables swift and straightforward data normalization within the specified range.

4) Balancing Data

The technique of data balancing guarantees a uniform allocation of the two categories, specifically standard and attack, across the dataset. Figure 2 portrays the inequitable distribution of these categories, with a cumulative count of 798,322 entries for average records and 390,011 admissions for attack records. In order to rectify this incongruity, scholars have implemented the SMOTE oversampling approach. Consequently, this balancing procedure has significantly enhanced the accuracy of data classification.

5) Feature Extraction

In this research effort, the CFS algorithm is utilized to select the most suitable feature subset from among many feature subsets. The process can be understood as follows:

a. Correlation computation: The correlation between each feature and the target variable is computed in this analysis. Pearson's correlation coefficient is typically employed for continuous variables, whereas other methods, such as Spearman's rank correlation or point-biserial correlation, are used for ordinal and binary data, respectively. The Pearson correlation coefficient between two variables, denoted as (X) and (Y), is calculated using the following formula:

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - X)(Y_i - Y)}{\sqrt{\sum_{i=1}^{n}(X_i - X)^2}\sqrt{\sum_{i=1}^{n}(Y_i - Y)^2}} \tag{2}$$

b. Inter feature correlation: the correlation between each pair of features is calculated. The goal is to retain features that show lower correlations with each other to avoid redundancy.
c. Feature subset evaluation: various combinations of features are evaluated to identify a subset that maximizes a correlation-based heuristic. The heuristic assesses the feature subset's value or quality by considering each feature's individual predictive ability and the degree of redundancy between them. The heuristic for a feature subset (S) with (k) features is given by:

$$Meri\,t_S = \frac{k \cdot r_{cf}}{\sqrt{k + k \cdot (k-1) \cdot r_{ff}}} \tag{3}$$

d. Search method: search strategies are used to explore different feature subsets. This can involve techniques such as first-best search, greedy search, or more complex methods such as genetic algorithms.
e. Best subset selection: after assessing the subsets, the one with the highest merit score is chosen as the ultimate feature set for model training

As a whole, ten features were chosen by the aforementioned algorithms out of the initial dataset comprising of 26 elements. A depiction of the methodology employed for the research algorithms can be observed in Figure 3. A thorough exposition of the achieved outcomes is presented in Table 1. Additionally, the process of selecting the algorithms also assisted in ascertaining the significance of each chosen feature, as demonstrated in Figure 4.

```
Data: $F(X_1,X_2,\ldots,X_n,C)$          // $F$ is the training data set, $X_i$ are the features and $C$ is the class feature
Result: $S'$                                                                      // Optimal subset of features
$S' = \emptyset$                                            // $S'$ is the set of optimal features; initially it is empty
                          // Steps 1 to 8 produce a list of features in decreasing order of their correlation to $C$
begin
    New_merit $= -1$
    Current_merit $= 0$
    while Current_merit $>$ New_merit or $S$ <> Null do
        Current_merit $=$ New_merit
        New_merit $= -1$                          // This variable will help us to identify which feature we should include
        for $i = 1$ until $n$                                  // From all features of $S$, find that with the highest merit
         do
            Maximum_value = Obtain the merit of $S'UX_i$                      // In order to get the merit, is necessary
            • if Maximum_value $>$ New_merit                  // to calculate the correlations as described in equation 1
             then
                New_merit = Maximum_value
                Feature = $X_i$
            end
        end
        if Current_merit $<$ New_merit then
            Add Feature to $S'$
            Remove Feature from $S$ and subtract 1 from $n$
        end
    end
    return $S'$
end
```

Figure 3. CFS algorithm [19]

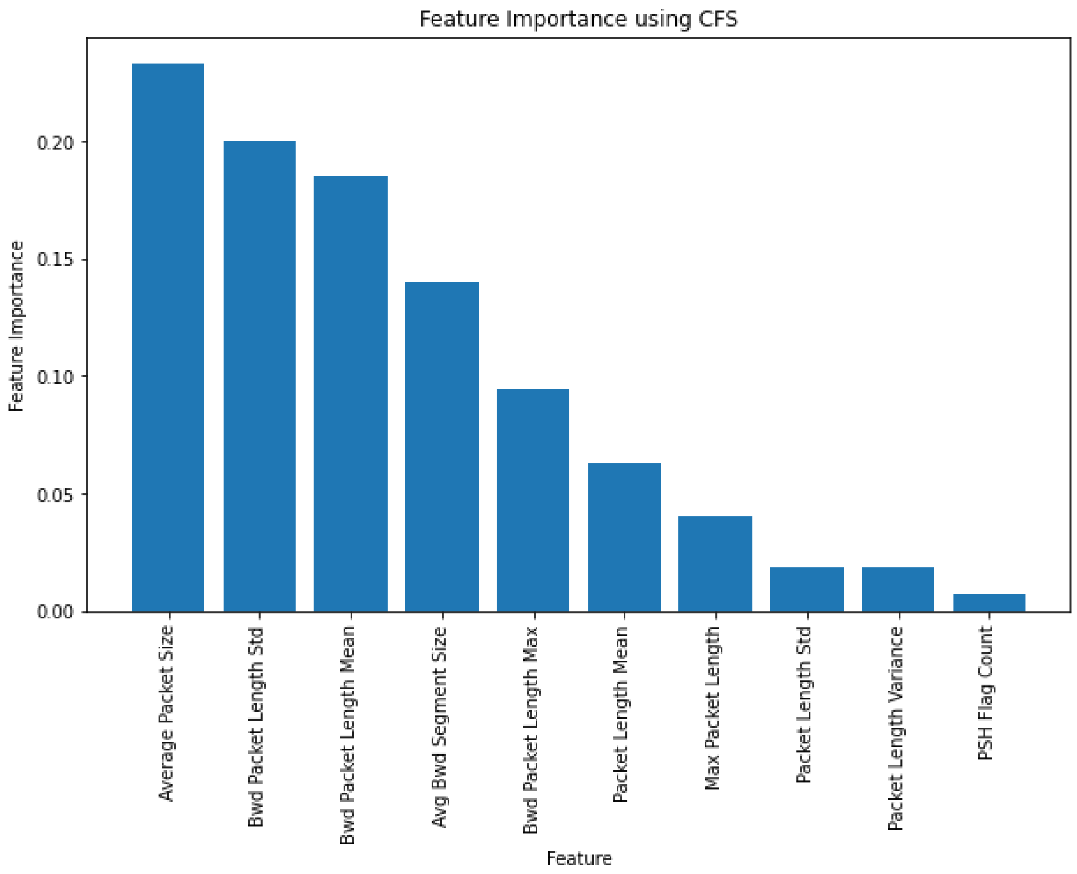


Figure 4. The importance of each feature

Table 1.
A comprehensive examination of the extracted characteristics

Features	Description
Average Packet Size	This feature represents the mean packet size within the network.
Bwd Packet Length Std	This feature quantifies the standard deviation of packet lengths in the reverse direction within the network.
Bwd Packet Length Mean	This feature represents the mean packet length in the reverse direction within the network.
Avg Bwd Segment Size	This feature quantifies the mean size of backward segments in the reverse direction within the network.
Bwd Packet Length Max	This feature represents the maximum packet length in the reverse direction within the network.
Packet Length Mean	This feature quantifies the mean packet length within the network.
Max Packet Length	This feature represents the maximum packet length within the network.
Packet Length Std	This feature quantifies the standard deviation of packet lengths within the network.
Packet Length Variance	This feature calculates the variance in packet lengths within the network.
PSH Flag Count	This feature represents the count of packets with the PSH (Push) flag set in the network.

B. Classifiers Used

In the realm of Machine Learning-driven intrusion detection systems, the central aim revolves around discerning the normalcy or abnormality of network traffic, demanding the exploration of a diverse array of machine learning algorithms. The literature extensively contains documentation on a multitude of machine learning algorithms [20]. As such, this investigation leverages a suite of classifiers, including Random Forest (RF), Decision Tree, AdaBoost, k-Nearest Neighbor (k-NN), Gradient Boosting, XGBoost, LightGBM, and CatBoost. Moreover, it remains essential to evaluate the performance of these classifiers to pinpoint the most suitable one for effectively detecting DDoS attacks.

1) Random Forest (RF)

Random Forest (RF) is a composite model that consists of a substantial ensemble of Decision Trees, where each tree is trained on a distinct subset of the dataset. While individually considered weak classifiers, their collective utilization often results in notable levels of accuracy. When RF is utilized to classify or predict new data samples, each Decision Tree performs classification based on the data's features, and the final outcome is determined through averaging or majority voting, as depicted in Figure 5. In the training phase, each Decision Tree within the RF framework undergoes independent sampling using the Bootstrap method and employs random feature selection. This specific methodology effectively prevents overfitting, thereby enhancing the stability and generalization capabilities of the model. These unique characteristics have contributed to the increasing adoption of RF in research endeavors focused on attack detection and addressing regression challenges [21].

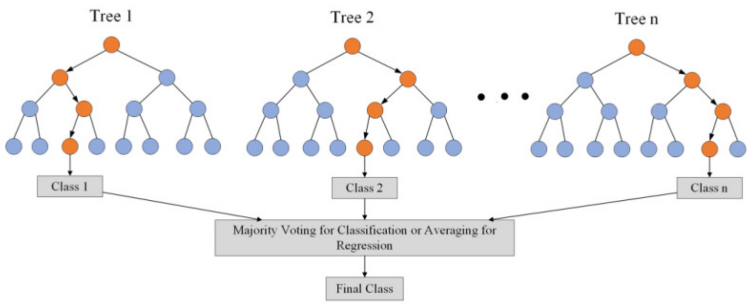

Tree 1
Tree 2
Tree n
Class 1
Class 2
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

Figure 5. Random Forest (RF) [22]

2) *Decision Trees (DT)*

The Decision Tree algorithm, a tool widely employed in the realm of machine learning, allows for the establishment of a pathway from the root to the leaf nodes. This methodology organizes data in a tree-like structure, dividing a dataset such that every node represents a characteristic and each branch signifies a different value of said characteristic. The final leaf node dictates the ultimate classification outcome. The Decision Tree possesses notable merits such as its simplicity, which renders it easily comprehensible and analyzable, its capacity to handle nonlinear attributes, its suitability for managing voluminous datasets, and its adaptability to multi-class predicaments. Moreover, the Decision Tree can be integrated with other machine learning techniques as part of an ensemble model, thereby augmenting classification accuracy. Consequently, the Decision Tree often emerges as a commendable choice for classification tasks across a wide range of applications. Notably, its popularity is on the rise in the domain of DDoS detection [11].

3) *Adaptive Boosting (AdaBoost)*

AdaBoost, short for Adaptive Boosting, is a machine learning approach that builds a strong classifier from multiple weaker ones. It's part of ensemble learning, where several models are combined to enhance the overall performance. AdaBoost trains weak classifiers on different data subsets iteratively, then merges them into a more robust classifier. Each iteration focuses more on previously misclassified samples by assigning them greater weight, allowing the next weak classifier to pay more attention to these samples. The final classifier is a composite of these weak classifiers, with their influence based on performance in training. AdaBoost is effective in various fields like face detection, object recognition, and text classification, significantly boosting system accuracy, particularly when used alongside other machine learning methods. An example is its integration with Decision Trees to form the AdaBoost M1 algorithm, a popular implementation. Extensively researched, AdaBoost has proven its value in multiple areas, including face detection, object recognition, and text classification [23].

4) *K-Nearest Neighbors (k-NN)*

The k-NN (k-Nearest Neighbors) algorithm functions on the principle of acquiring knowledge from observed instances, thereby enabling it to make classifications and predictions without imposing any defined assumptions regarding the distribution of the data. During the classification procedure, k-NN assigns a given data point to a specific category based on the class that occurs most frequently among its k nearest neighbors. As a result, it establishes the class for a new data point by evaluating the classes of its closest neighbors. This process of classifying unfamiliar instances unfolds in a sequence of consecutive steps.

a. The determination of the distance between the sample to be classified and every sample in the training set is typically accomplished by utilizing distance metrics like the Euclidean or Manhattan distance.
b. Choose the K samples that are closest in proximity from the dataset, but belong to different classes, to be used in the classification model. Arrange these samples in ascending order based on their level of proximity.
c. By scrutinizing the classifications of the K nearest samples via a majority vote mechanism, one can ascertain the categories to which the samples should be assigned. It is generally advisable to select an odd value for K in order to prevent potential deadlocks during the voting procedure.

$$d_{euclidean} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

$$d_{manhattan} = \sum_{i=1}^{n} | x_i - y_i | \tag{5}$$

The k-NN algorithm presents numerous benefits, such as its simplicity, ease of comprehension, and lack of necessity for pre-trained models. It effectively handles both classification and regression tasks, yielding remarkable outcomes even in scenarios with non-linearly separable data. Due to its practicality and straightforward nature, it has garnered widespread acceptance, especially in the domain of multi-class classification challenges. Additionally, various research inquiries have emphasized its exceptional performance in situations involving attack detection. [17],[23].

5) *Gradient Boosting (GB)*

Gradient Boosting is an ensemble machine learning technique utilized in tasks like regression and classification. It stands out for its adaptability to specific applications, including the capability to learn various loss functions. An advancement of the boosting method, Gradient Boosting builds on the concept of amalgamating several weak models to form a stronger one. It progressively refines new models for a more precise prediction of the response variable. The core principle of this algorithm is to construct a new base-learner that aligns closely with the negative gradient of the loss function related to the entire ensemble. Often, a fixed-size Decision Tree, particularly CART, is employed as the base-learner in Gradient Boosting. The algorithm revolves around three key elements: a loss function that needs optimization, a weak learner for making predictions, and an additive approach to enhance the weak learner in reducing the loss function. Gradient Boosting is effective because it merges multiple frail learners into a potent one. Each new model focuses on minimizing a loss function, like mean squared error or cross-entropy, from the preceding model through gradient descent. With each iteration, it calculates the loss function's gradient against the current ensemble prediction and trains a new weak model to minimize this gradient. Gradient Boosting is especially robust as it adjusts weights based on the gradient, making it less susceptible to outliers [25].

6) *Extreme Gradient Boosting (XGBoost)*

The XGBoost algorithm is a form of ensemble learning that is based on the principles of Gradient Boosting Trees. It trains a set of weak classifiers, such as Decision Trees, through an iterative process that improves the model's ability to make predictions, ultimately resulting in a strong classifier. During the construction of each tree, this technique considers the residual error from the previous tree while also minimizing a loss function, such as mean square error. XGBoost improves both the accuracy and efficiency of the model by optimizing the Gradient Boosting Tree algorithm, incorporating methods for regularization and parallel processing to address the risks of overfitting and speed up the training process. The algorithm's effectiveness is due to its exceptional accuracy and speed, making it a suitable choice for identifying attacks in various research efforts [26]. This study evaluates the performance of XGBoost compared to other classifiers in its ability to detect DDoS attacks.

7) *Light Gradient Boosting Machine (LightGBM)*

LightGBM is an advanced implementation of gradient boosting designed to be more efficient and scalable than traditional gradient boosting methods. LightGBM utilizes gradient-based one-sided sampling (GOSS) to prioritize examples that provide more informative data and exclusive feature bundling (EFB) to reduce feature dimensionality. As a result, the system is highly skilled in efficiently handling large data sets and minimizing memory usage. Unlike the level-wise approach used by other methods, LightGBM builds trees leaf-wise, enabling faster convergence and potentially more accurate models. In addition, LightGBM directly supports categorical features, so it does not require one-off coding. To prevent overfitting, LightGBM incorporates regularization techniques. As a result of its efficient computation and high speed, LightGBM has gained wide popularity in machine-learning applications that prioritize performance and speed while maintaining model accuracy [27].

8) *Categorical Boosting (CatBoost)*

CatBoost is an algorithm in machine learning that utilizes gradient-boosting techniques in Decision Trees. It has been specifically designed to handle categorical data effectively. The name "CatBoost" is a combination of the words "Category" and "Boosting" and stands for Categorical Boosting. Unlike traditional coding methods that require pre-processing, such as one-off coding, CatBoost transforms categorical variables into numerical values by utilizing the combination statistics of flat features and target variables. This transformation process ensures that the intrinsic properties of categorical variables are preserved while maintaining efficiency. This is achieved through "ordered boosting," an alternative approach to the classic boosting method based on permutations. In addition, CatBoost uses an unconscious Decision Tree, which enforces the same division decision across tree levels. This leads to improved model stability and faster computation. In addition, CatBoost has a robust mechanism to handle missing data and supports GPU acceleration. These features make CatBoost a high-performance and accurate choice for boosting, especially suitable for datasets with many categorical features [28].

III. RESULT AND DISCUSSION

This division offers a comprehensive explanation of the utilized experimental approaches, the display of the acquired findings, and the subsequent examination and discourse, making connections with pertinent investigations in the discipline.

A. *Evaluation Criteria*

The results of the predictions are categorized into four distinct types, as depicted in Figure 6: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). TP signifies the number of normal samples accurately identified by the algorithm, while TN represents the correct identification of attack samples. Conversely, FP indicates normal samples that were incorrectly classified as attack samples, and FN denotes attack samples that were erroneously classified as normal by the algorithm. This research evaluates the performance of the proposed model utilizing designated metrics:

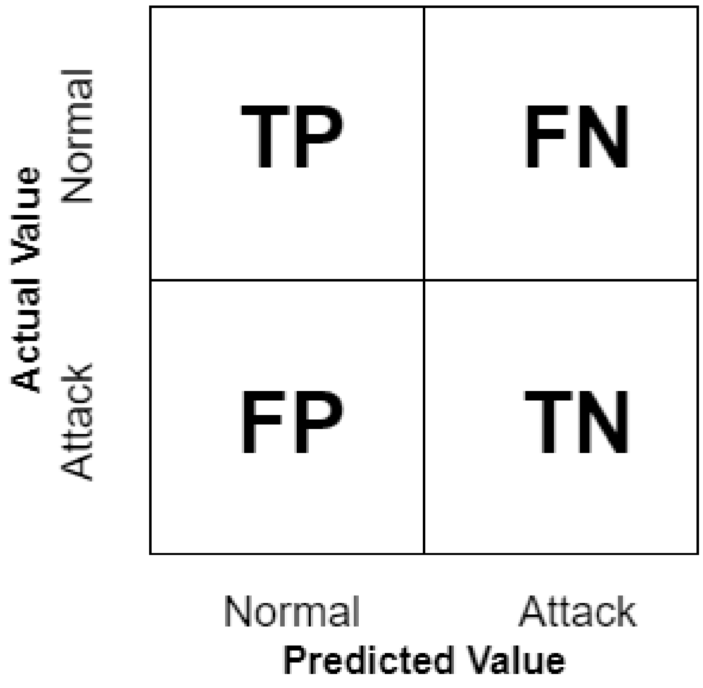


Figure 6. Confusion matrix

a. Accuracy: accuracy gauges the overall correctness of the classifier by computing the proportion of correctly classified instances in relation to the total number of instances.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

b. Precision: precision assesses the ratio of accurate positive predictions among all positive predictions, revealing the classifier's capability to minimize false positives.

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

c. Recall: recall, often referred to as sensitivity, evaluates the ratio of true positive predictions among all genuine positive examples, illustrating the classifier's capacity to minimize false negatives.

$$Recall = \frac{TP}{TP+FN} \quad (8)$$

d. F1: the F1 score represents the harmonic mean of precision and recall, furnishing a well-balanced measure of classifier performance.

$$F1 - Score = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \quad (9)$$

e. ROC curve: AUC, which stands for Area Under the Curve, is a metric employed in binary classification to assess a classifier's capacity to differentiate between positive and negative examples. It quantifies the area beneath the receiver operating characteristic (ROC) curve, which graphically portrays a model's classification prowess. More proficient models exhibit higher ROC curves with larger areas underneath them. The model's performance is commonly evaluated using AUCROC (Area Under the ROC Curve), where a value of one signifies nearly flawless classification, while a lower value indicates a lower level of model performance.

B. Evaluation of Outcomes

1) Evaluation of the Performance of Individual Classification Models

In this study, we employed eight distinct machine learning algorithms: Random Forest (RF), Decision Tree, Adaptive Boosting (AdaBoost), k-Nearest Neighbors (k-NN), Gradient Boosting (GB), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), and Categorical Boosting (CatBoost). Each of these classifiers was trained and evaluated using their default settings through K-fold cross-validation. Prior research [28],[29] recommends a K value of 10. The performance of each classifier on the dataset is visually presented in Figures 7 and 8, encompassing metrics such as Accuracy, Precision, Recall, and F1 score. For a comprehensive analysis of the classification results, please refer to Table 2.

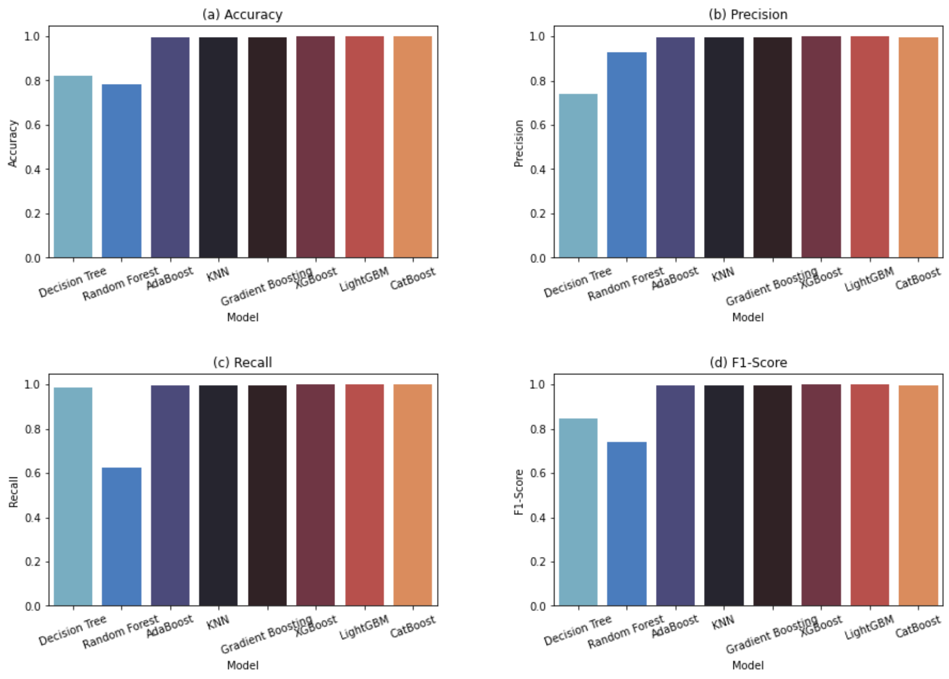


Figure 7. Assessment Metrics for Each Classifier Prior to Feature Extraction

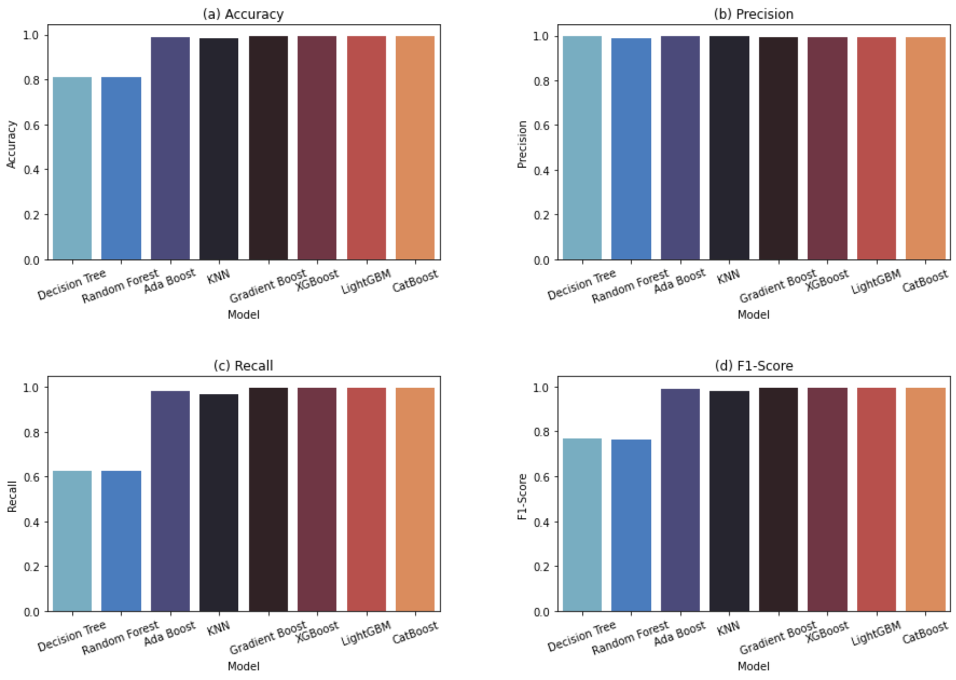


Figure 8. Assessment Metrics for Each Classifier Following Feature Extraction

Table 2.
Assessing and Comparing Performance Metrics across Various Classifiers for both the Original Dataset and the Dataset After Feature Extraction

	Dataset before Feature Extraction				**Dataset after Feature Extraction**			
	Accuracy	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
DT	0.819968	0.739917	0.986808	0.845708	0.811193	0.996727	0.624436	0.767831
RF	0.784220	0.926899	0.624746	0.742344	0.808260	0.987848	0.624570	0.765336
AdaBoost	0.994731	0.993531	0.995947	0.994737	0.989087	0.996931	0.981195	0.989000
k-NN	0.997609	0.997296	0.997924	0.997610	0.982574	0.999698	0.965442	0.981477
GB	0.996628	0.995866	0.997403	0.996631	0.994542	0.991805	0.997310	0.994550
XGBoost	0.998939	0.999478	0.998399	0.998938	0.994868	0.992078	0.997702	0.994882
LightGBM	0.998588	0.998771	0.998405	0.998588	0.994819	0.992103	0.997578	0.994833
CatBoost	0.998048	0.997822	0.998275	0.998049	0.994751	0.992046	0.997501	0.994766

Table 2 presents a comparative analysis of the performance of various machine learning models, focusing on four key metrics: Accuracy, Precision, Recall, and F1 Score. Initially, before feature extraction, the Decision Tree model exhibited a high accuracy rate (0.819968) but displayed lower Precision values, suggesting challenges in accurately identifying all positive instances. However, following feature extraction with CFS, there was a marginal decrease in accuracy (0.811193), alongside an increase in Precision values. This shift indicates that post feature extraction, the Decision Tree model improved in correctly identifying positive instances.

Before feature extraction, the Random Forest (RF) model exhibited lower accuracy (0.784220), with also low recall and F1 scores. Post feature extraction using Correlation Feature Selection (CFS), there was an improvement in accuracy (0.808260), as well as in recall and F1 scores, indicating the positive impact of CFS on RF's performance.

The AdaBoost model showed strong performance both before and after feature extraction, albeit with a minor decrease in accuracy following feature extraction. Similarly, the k-Nearest Neighbors (k-NN) model maintained high performance in both scenarios, though it too experienced a slight dip in accuracy post feature extraction.

Boosting models, including Gradient Boosting, XGBoost, Light Gradient Boosting Machine (LightGBM), and Categorical Boosting (CatBoost), demonstrated outstanding performance before and after feature extraction, maintaining high performance levels throughout. Notably, models like AdaBoost, k-NN, Gradient Boosting, XGBoost, LightGBM, and CatBoost showed near-perfect performance in all metrics both before and after feature extraction. These models proved highly effective in classifying positive cases, with AdaBoost particularly excelling in balancing Precision and Recall, as reflected in its high F1 score. Similarly, k-NN, Gradient Boosting, and ensemble models like XGBoost, LightGBM, and CatBoost displayed high accuracy, along with a strong balance between Precision and Recall.

Here is an expanded analysis of the performance difference before and after feature extraction for each classification model:

1. Decision Tree (DT):
   - Accuracy decreased slightly from 0.819968 to 0.811193 after feature extraction. This indicates that some useful information may have been lost during feature extraction
   - Precision increased significantly from 0.739917 to 0.996727. This shows feature extraction helped the model better identify true positives.
   - Recall dropped noticeably from 0.986808 to 0.624436. The model now struggles more to detect all positive cases. There is a trade-off between improved precision and reduced recall.
2. Random Forest (RF):
   - Accuracy improved from 0.784220 to 0.808260 after feature extraction. The selected features contain useful information for classification.
   - Precision, recall and F1 score all increased. Feature extraction helped improve overall performance
3. AdaBoost:
   - Accuracy dropped slightly from 0.994731 to 0.989087. Some useful patterns may exist in the discarded features.

- Other scores stayed high, so the performance trade-off is minor.

4. k-NN:
   - Minor performance decrease after feature extraction. Useful information in discarded features may have been lost.
5. Gradient Boosting (GB):
   - Almost identical high performance before and after feature extraction. Selected features retain the predictive power.
6. XGBoost:
   - Similar story. No significant difference in metrics before and after feature extraction. Selected features are sufficient.
7. LightGBM:
   - Again, performance remained excellent after feature extraction. No noticeable disadvantage from discarding features.
8. CatBoost:
   - Consistent near-perfect performance with minimal decrease after feature extraction. Useful patterns may exist in removed features.

In summary, feature extraction leads to a trade-off between improved precision and reduced recall in some models, while others show consistent performance. Overall, the core predictive information is retained during feature selection.

Overall, the boosting-based models (AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost) demonstrated superior performance compared to the Decision Tree and Random Forest (RF) models. The choice of the most suitable model depends on the specific application and the priority given to Precision, Recall, or a balanced consideration of both (F1 Score). For instance, in medical diagnostics, a higher Recall might be preferred to reduce the risk of missing positive cases. Additionally, Figures 9 and 10 display the Confusion Matrix for each classifier applied to the dataset. The values in these matrices reflect the number of correct and incorrect predictions made by each model. The values along the central diagonal (from the top left to the bottom right corner) represent the count of accurate predictions for each class, providing a clear insight into the performance of each classifier. To illustrate, before performing feature extraction, the Decision Tree algorithm accurately classified 43.27% of the data as Benign and 33.21% as Attack. Instances that fall outside the main diagonal in the classification matrix indicate misclassification. For example, the Decision Tree algorithm mislabeled 0.59% of Benign instances and 23.93% of Attack instances as Benign.

Among the algorithms considered, the performance in detecting attacks appears to be relatively similar, with percentages ranging from 21.19% to 33.74%. However, there is minimal variation in identifying benign cases, with rates ranging from 43.27% to 66.17%. The k-NN, XGBoost, LightGBM, and CatBoost algorithms show very similar performance, with little difference in the number of misclassifications for both classes. After performing feature extraction, the decision tree correctly identified 66.05% of the data as benign and 21.19% as an attack. Values that lie outside the main diagonal indicate misclassification. For example, Decision Tree inaccurately labeled 12.62% of Benign instances as Attack and 0.15% of Attack instances as Benign.

Among the algorithms considered, the performance seems similar in detecting attacks, with percentages ranging from 21.19% to 33.72%. Nevertheless, there is a slight variation in identifying benign cases, with rates ranging from 65.05% to 65.99%. The Gradient Boosting, XGBoost, LightGBM, and CatBoost algorithms showed very similar performance, with little difference in the number of misclassifications for both classes. The AUC-ROC curves for all classifiers are presented in Figure 11 and Figure 12 before performing Decision Tree feature extraction with a value of 0.82, RF with 0.78, AdaBoost with 0.99, k-NN with 1.00, Gradient Boosting with 1.00, XGBoost with 1.00, LightGBM with 1.00 and CatBoost with 1.00. After extracting Decision Tree features as much as 0.81, RF as much as 0.81, AdaBoost as much as 0.99, k-NN as much as 0.99, Gradient Boosting as much as 0.99, XGBoost as much as 0.99, LightGBM as much as 0.99, and CatBoost as much as 0.99. The AUC-ROC curve graphically represents how well a classification model performs over every possible threshold for classification. It displays the relationship between the true positive rate (also known as recall) and the false positive rate at various thresholds. In an ideal scenario, a flawless model would attain an AUC (Area Under the Curve) score of 1.

Within the scope of this study focused on detecting DDoS attacks:

- An AUC closer to 1 indicates the model is correctly identifying more real attacks as attacks (true positives) while minimizing false alarms (false positives). This demonstrates good discrimination ability between the attack and benign classes.

- Before feature extraction, the AUC scores ranged from 0.78 to 1.00, with most models over 0.99. This shows extremely strong classification performance across all models initially.
- After CFS-based feature extraction, the AUC scores remained high between 0.81 to 0.99. So while some useful information may have been discarded, the core predictive power was retained post-extraction.
- The consistency of high AUC scores even after reducing the feature space dimensionality validates that the selected subset contains the most informative attributes for discrimination.
- Top models like XGBoost, LightBGM and CatBoost maintained AUC values equal or close to 1 after extraction. This perfect or near-perfect AUC proves their robustness and suitability for this DDoS detection task.

In summary, the preservation of strong AUC-ROC curve scores from pre to post-extraction demonstrates that:

1. The models show excellent ability to differentiate between attack and benign traffic.
2. The CSF method successfully extracted the most relevant predictive features.
3. Features discarded by CFS were likely redundant or non-informative.
4. Top ensemble models are perfectly or near-perfectly suited for this classification problem.

The high AUC value validated the effectiveness of the proposed approach combining CFS and machine learning models, especially boosting-based algorithms, for DDoS detection in SDN environments.

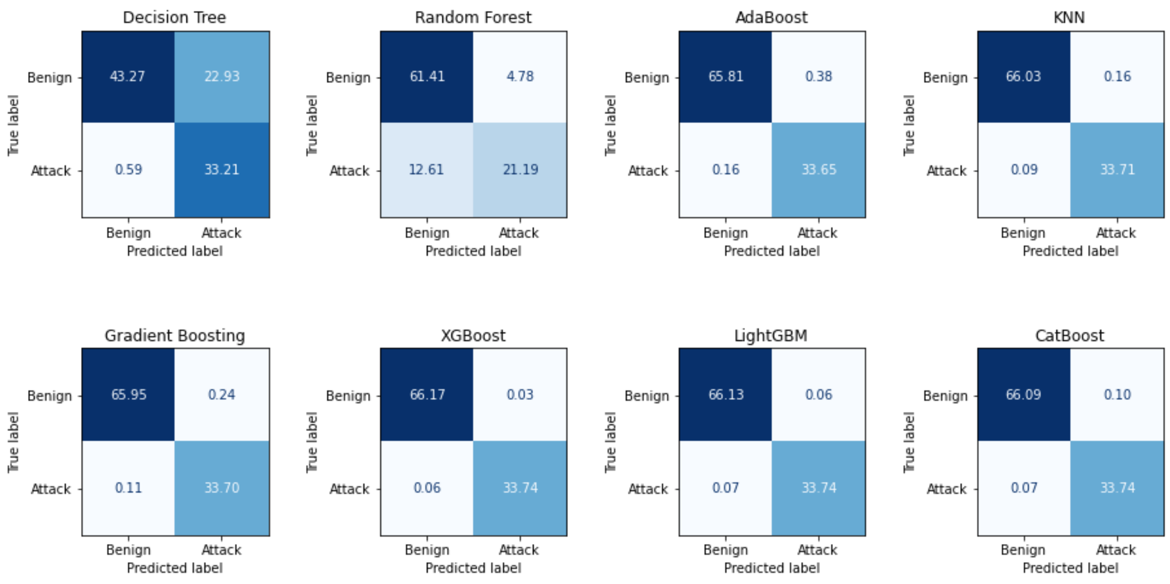


Figure 9. Confusion matrix of each classifier before feature extraction in percent

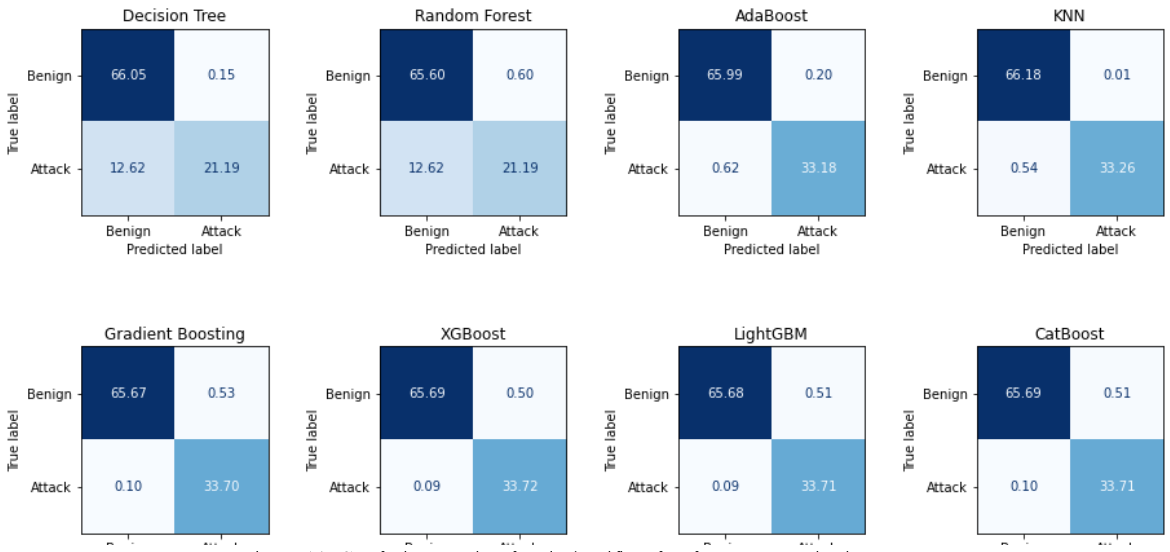


Figure 10. Confusion matrix of each classifier after feature extraction in percent

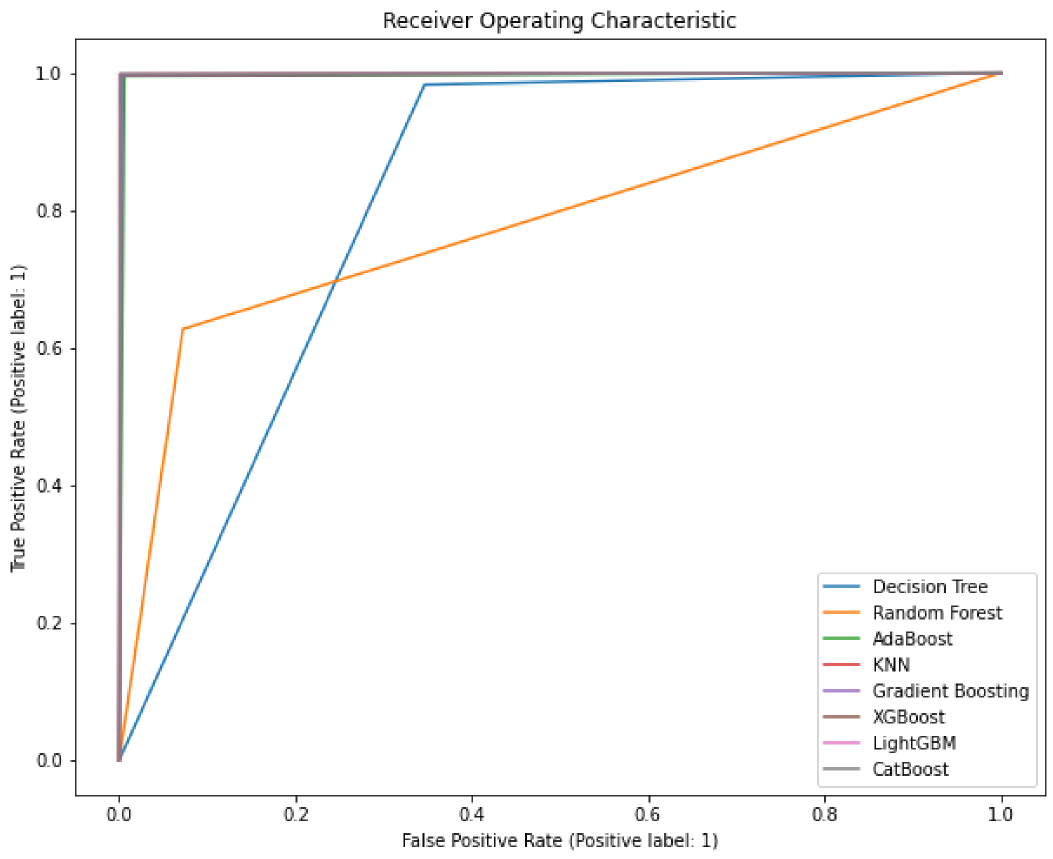


Figure 11. AUC-ROC curves for all classifiers before feature extraction

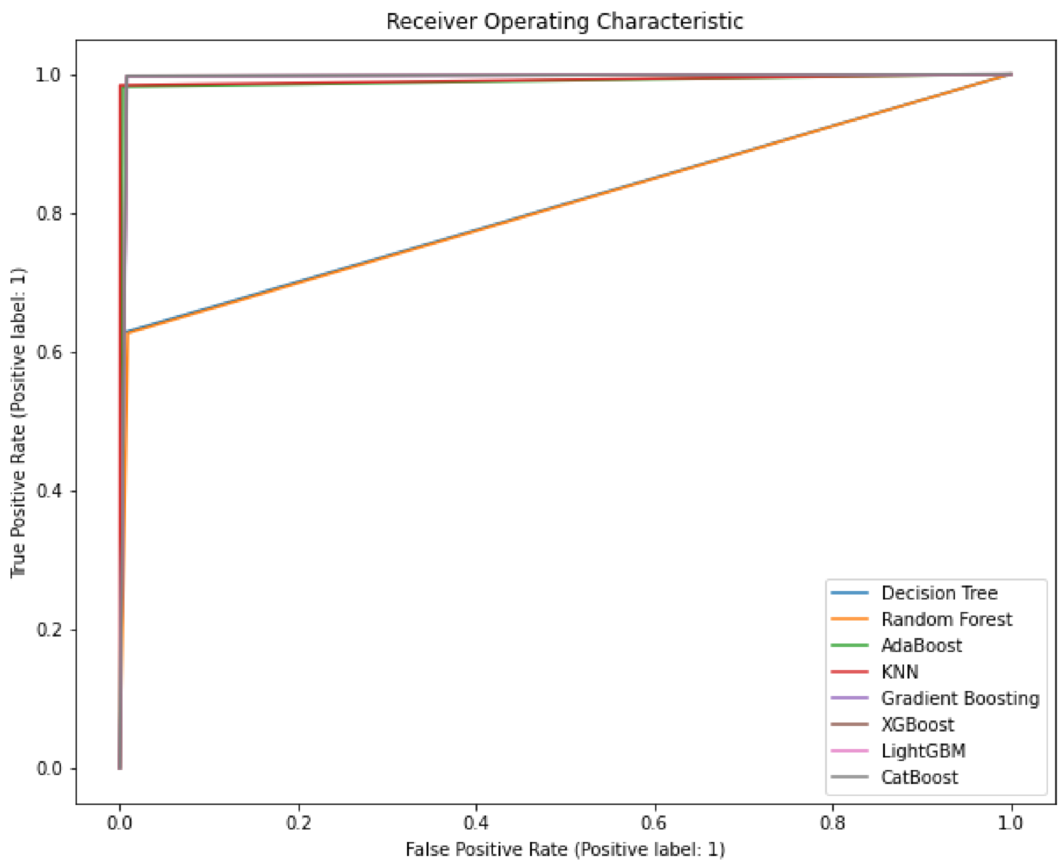


Figure 12. AUC-ROC curves for all classifiers after feature extraction

2) Comparison with Others Research

Table 3 delineates a comparative analysis between the findings presented in this research and those of related studies

Table 4. Comparison with others research

References	Year	Model	Accuracy	Precision	Recall	F1-Score
[9]	2020	Wrapper-Based and k-NN	0.983	0.9772	0.9773	0.9770
[12]	2019	Chi2 and k-NN	0.9351	NA	NA	NA
[31]	2022	Parallel RNN-based SVM Model	0.9762	0.9772	0.9679	0.9719
[32]	2023	Deep belief network feature extraction and PSO-LSTM	0.98	0.97	0.95	0.96
[33]	2021	SVC-RF Improved binary grey wolf	0.988	0.9827	0.979	0.9765
[22]	2023	Wolf optimization algorithm and RF	0.9913	0.9843	0.9992	0.9913
Our study	2023	CFS algorithm and XGBoost	0.994868	0.992078	0.997702	0.994882

Polat et al. [9] utilized a combination of three distinct wrapper feature selection techniques, namely Filter, Wrapper, and Embedded, in conjunction with a variety of classification models, such as SVM, NB, ANN, and k-NN, for both training and testing purposes. Their investigation revealed that when the Wrapper-based feature extraction method was employed in tandem with the k-NN classifier, it demonstrated exceptional performance, achieving accuracy, precision, recall, and F1 scores of 0.983, 0.9772, 0.9773, and 0.977, respectively. In contrast, our study has exhibited enhancements in accuracy, precision, recall, and F1 score by 0.0083, 0.0071, 0.0219, and 0.0143, correspondingly, compared to these aforementioned findings.

Aamir et al. [12] utilized feature selection techniques, including before-and-after elimination, Chi2, and information gain scores, along with several supervised machine learning models for classification. Their study indicated that the most successful combination was Chi2 paired with the k-NN classifier, resulting in an accuracy of 0.9351. However, their data preprocessing approach was reported as time-consuming, and they did not provide comprehensive details on other evaluation criteria. In contrast, our current research showcases a significant accuracy improvement of 0.0562 when compared to their prior study.

Polat et al. [31] utilized Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models for feature extraction, and they employed the Support Vector Machine (SVM) model for classification. Their approach yielded commendable results, with an accuracy rate of 0.9762, a precision rate of 0.9772, a recall rate of 0.9679, and an F1 score of 0.9719. However, it is regrettable that their feature extraction methodology and outcome details remained relatively under-explored. In contrast, our own research demonstrates notable enhancements in terms of accuracy, precision, recall, and F1 score, improving by 0.0083, 0.0071, 0.0219, and 0.0143, respectively.

Thangasamy et al. [32] harnessed deep belief networks for feature extraction and incorporated the PSO-LSTM model for classification, resulting in a remarkable accuracy rate of 0.98. Furthermore, they reported a precision of 0.97, a recall of 0.95, and an F1 score of 0.96. Despite their effective utilization of a feature extraction algorithm, the authors did not provide a detailed exposition of the extracted features. In this current investigation, we aim to bridge this gap by showcasing substantial improvements in accuracy, precision, recall, and F1 score. Our findings reveal enhancements of 0.0113, 0.0143, 0.0492, and 0.0313, respectively, when compared to the aforementioned study.

Ahuja and his colleagues [33] conducted an experimental study in which they generated various types of network attacks, including UDP, TCP, and ICMP, alongside regular network traffic. They extracted a total of 23 distinct features from the collected data, focusing on eight of them for a more comprehensive analysis. Employing a hybrid machine learning model known as SVM-RF, they evaluated the classification performance in terms of accuracy, recall, precision, and F1 score, obtaining values of 0.988, 0.9827, 0.979, and 0.9765, respectively. Importantly, the authors' investigation demonstrated improvements in accuracy, precision, recall, and F1 scores by 0.0033, 0.0016, 0.0202, and 0.0148, respectively.

Liu et al. [22] presented a novel approach for detecting DDoS attacks within SDN environments, relying on a combination of feature engineering and machine learning techniques. Their methodology, tested using the CSE-CIC-IDS2018 dataset and an improved binary gray wolf optimization algorithm, demonstrated exceptional performance when assessed using the RF algorithm. Notably, their system achieved remarkable values for accuracy, recall, precision, and F1 score, reaching 0.9913, 0.9843, 0.9992, and 0.9913, respectively. Notably, this approach marked a substantial improvement of 0.0278 when compared to prior methods.

As proposed in this study, the CFS method shows superior performance when used in conjunction with the XGBoost model. It is essential to recognize that existing literature on this subject conducts similar investigations, albeit using different data sets and models. In the realm of classifying distributed denial of service (DDoS) attacks in software-defined networks (SDNs) via machine learning, our study stands out in its approach, particularly when contrasted with existing research. Unlike other studies that employ a variety of datasets ranging from real-time traffic data in SDNs to specific DDoS datasets like CIC-IDDOS2019, our study utilized a Kaggle dataset, uniquely optimized through the Correlation-based Feature Selection (CFS) algorithm. This optimization is significant as it tailors the feature subsets specifically for DDoS attack detection in SDNs, potentially enhancing the efficacy of our chosen machine learning algorithms. The diversity in our algorithm selection, encompassing techniques from Random Forest to advanced algorithms like XGBoost, LightGBM, and CatBoost, offers a comprehensive analysis, highlighted by the superior performance of the XGBoost algorithm. This stands in contrast to other studies which might focus on a narrower range of models or different feature selection techniques, such as SVM combined with wrapper-based feature selection or hybrid LSTM techniques. The strength of our study lies in its high accuracy, precision, and recall, particularly post-feature extraction, which indicates an effective balance in identifying true positives while minimizing false negatives. This is a notable improvement over other studies, which may demonstrate lower accuracy and F1 scores, potentially due to their dataset nature or model choice. Moreover, our study's focused application to SDN environments specifically addresses the challenges posed by DDoS attacks in these networks. This specialization potentially makes our methodology more tailored and effective compared to broader approaches, underlining the importance of context-specific strategies in cybersecurity challenges. Hence, the disparities in datasets, models, and methodologies not only underscore the distinctiveness of our research but also hint at its possible supremacy in achieving higher accuracy and efficiency within the realm of DDoS attack classification in SDNs.

IV. CONCLUSION

This research paper presents a novel approach for the detection of DDoS attacks in the context of SDNs. The proposed method combines feature engineering techniques with a variety of machine learning algorithms. Specifically, it utilizes the CFS algorithm to extract relevant features from the dataset. Furthermore, eight distinct machine learning models, namely Random Forest (RF), Decision Tree, AdaBoost, k-Nearest Neighbors (k-NN), Gradient Boosting, XGBoost, LightGBM, and CatBoost, are deployed to assess and determine the optimal classifier for both the original and feature-extracted datasets. The experimental findings highlight that XGBoost attains the highest accuracy score of 0.998939 when applied to the original dataset.

Furthermore, the feature extraction procedure results in a reduction in the feature count from 26 to 10, and all classifiers display enhancements across a range of evaluation metrics. Significantly, the XGBoost classifier stands out with outstanding performance in Accuracy, Precision, Recall, and F1-score, achieving scores of 0.994868, 0.992078, 0.997702, and 0.994882, respectively. When comparing the evaluation metric values between models that employ CFS feature selection and those that do not, the following improvements become apparent:

a. Precision: models employing CFS exhibit enhancements ranging from 0.256719 to 0.259614.
b. Recall: models employing CFS demonstrate improvements varying from 0.340162 to 0.373482.
c. F1: models employing CFS display enhancements ranging from 0.222894 to 0.249213.
d. Accuracy: the utilization of CFS in models leads to improvements ranging from 0.026225 to 0.209837.

These findings provide a comprehensive overview of the performance enhancements achievable through applying CFS in feature selection. Employing CFS in machine learning models yields significant benefits, such as dimensionality reduction and improved model performance, effectively addressing challenges posed by infinite problems or large values. However, it is essential to consider potential risks associated with information loss when applying this approach. In conclusion, using CFS in the context of large and complex datasets can significantly enhance the efficiency and effectiveness of machine learning models. Still, it necessitates careful consideration of feature selection and handling of extreme values.

REFERENCES

[1] C. Raju, S. Rajagopal, K. Venusamy, K. Suriyan, and M. Alagarsamy, 'SDSFLF: fault localization framework for optical communication using software digital switching network', *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 12, no. 1, p. 113, Mar. 2023, doi: 10.11591/ijres.v12.i1.pp113-124.

[2] L. Shi, Y. Song, Z. Xue, Y. Liu, and H. Chen, 'SACT: A New Model of Covert Communication Based on SDN', *Sensors*, vol. 20, no. 24, p. 7300, Dec. 2020, doi: 10.3390/s20247300.

[3] P. Radoglou Grammatikis, P. Sarigiannidis, G. Efstathopoulos, and E. Panaousis, 'ARIES: A Novel Multivariate Intrusion Detection System for Smart Grid', *Sensors*, vol. 20, no. 18, p. 5305, Sep. 2020, doi: 10.3390/s20185305.

[4] A. O. Alzahrani and M. J. F. Alenazi, 'Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks', *Future Internet*, vol. 13, no. 5, p. 111, Apr. 2021, doi: 10.3390/fi13050111.

[5] S. Ramesh, S. Nirmalraj, S. Murugan, R. Manikandan, and F. Al-Turjman, 'Optimization of Energy and Security in Mobile Sensor Network Using Classification Based Signal Processing in Heterogeneous Network', *J Signal Process Syst*, vol. 95, no. 2–3, pp. 153–160, Mar. 2023, doi: 10.1007/s11265-021-01690-y.

[6] S.-J. Young, S.-J. Chang, S. D. Prior, and L.-W. Ji, 'Special Issue: Selected Papers from IEEE ICASI 2019', *Applied Sciences*, vol. 10, no. 8, p. 2652, Apr. 2020, doi: 10.3390/app10082652.

[7] Y. Song and L. Zhao, 'Skill Movement Trajectory Recognition of Freestyle Skiing U-Shaped Field Based on Deep Learning and Multitarget Tracking Algorithm', *Comput Intell Neurosci*, vol. 2022, pp. 1–12, Aug. 2022, doi: 10.1155/2022/7992045.

[8] Jay Kumar Jain, Akhilesh A. Waoo, and Dipti Chauhan, 'A Literature Review on Machine Learning for Cyber Security Issues', *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 374–385, Dec. 2022, doi: 10.32628/CSEIT228654.

[9] H. Polat, O. Polat, and A. Cetin, 'Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models', *Sustainability*, vol. 12, no. 3, p. 1035, Feb. 2020, doi: 10.3390/su12031035.

[10] H. Beitollahi, D. M. Sharif, and M. Fazeli, 'Application Layer DDoS Attack Detection Using Cuckoo Search Algorithm-Trained Radial Basis Function', *IEEE Access*, vol. 10, pp. 63844–63854, 2022, doi: 10.1109/ACCESS.2022.3182818.

[11] A. Mishra, N. Gupta, and B. B. Gupta, 'Defensive mechanism against DDoS attack based on feature selection and multi-classifier algorithms', *Telecommun Syst*, vol. 82, no. 2, pp. 229–244, Feb. 2023, doi: 10.1007/s11235-022-00981-4.

[12] M. Aamir and S. M. A. Zaidi, 'DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation', *Int J Inf Secur*, vol. 18, no. 6, pp. 761–785, Dec. 2019, doi: 10.1007/s10207-019-00434-1.

[13] A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, 'An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment', *Microprocess Microsyst*, vol. 89, p. 104412, Mar. 2022, doi: 10.1016/j.micpro.2021.104412.

[14] D. Akgun, S. Hizal, and U. Cavusoglu, 'A new DDoS attacks intrusion detection model based on deep learning for cybersecurity', *Comput Secur*, vol. 118, p. 102748, Jul. 2022, doi: 10.1016/j.cose.2022.102748.

[15] G. Karatas, O. Demir, and O. K. Sahingoz, 'Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset', *IEEE Access*, vol. 8, pp. 32150–32162, 2020, doi: 10.1109/ACCESS.2020.2973219.

[16] H. A. Yanti, H. Sukoco, and S. N. Neyman, 'Pemodelan Identifikasi Trafik Bittorrent Dengan Pendekatan Correlation Based Feature Selection (CFS) Menggunakan Algoritme Decision Tree (C4.5)', *CESS (Journal of Computer Engineering, System and Science)*, vol. 6, no. 1, p. 1, Jan. 2021, doi: 10.24114/cess.v6i1.20855.

[17] S. Dong and M. Sarem, 'DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks', *IEEE Access*, vol. 8, pp. 5039–5048, 2020, doi: 10.1109/ACCESS.2019.2963077.

[18] subhajournal, 'https://www.kaggle.com/datasets/subhajournal/sdn-intrusion-detection/data'.

[19] A. H. Márquez, A. G. Arenas, and G. L. M. Luna, 'Feature Selection Ordered By Correlation - FSOC', *Computación y Sistemas*, vol. 27, no. 1, Mar. 2023, doi: 10.13053/cys-27-1-3982.

[20] B. Alhijawi, S. Almajali, H. Elgala, H. Bany Salameh, and M. Ayyash, 'A survey on DoS/DDoS mitigation techniques in SDNs: Classification, comparison, solutions, testing tools and datasets', *Computers and Electrical Engineering*, vol. 99, p. 107706, Apr. 2022, doi: 10.1016/j.compeleceng.2022.107706.

[21] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij, and F. Malik, 'Machine-Learning-Based DDoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method', *Symmetry (Basel)*, vol. 14, no. 6, p. 1095, May 2022, doi: 10.3390/sym14061095.

[22] Z. Liu, Y. Wang, F. Feng, Y. Liu, Z. Li, and Y. Shan, 'A DDoS Detection Method Based on Feature Engineering and Machine Learning in Software-Defined Networks', *Sensors*, vol. 23, no. 13, p. 6176, Jul. 2023, doi: 10.3390/s23136176.

[23] W. Chen, Q. Sun, J. Wang, J.-J. Dong, and C. Xu, 'A Novel Model Based on AdaBoost and Deep CNN for Vehicle Classification', *IEEE Access*, vol. 6, pp. 60445–60455, 2019, doi: 10.1109/ACCESS.2018.2875525.

[24] Z. Ma and B. Li, 'A DDoS attack detection method based on SVM and K-nearest neighbour in SDN environment', *International Journal of Computational Science and Engineering*, vol. 23, no. 3, p. 224, 2020, doi: 10.1504/IJCSE.2020.111431.

[25] H. A. Alamri and V. Thayananthan, 'Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-Defined Networks Against DDoS Attacks', *IEEE Access*, vol. 8, pp. 194269–194288, 2020, doi: 10.1109/ACCESS.2020.3033942.

[26] Z. Mei, W. Yu, W. Tang, J. Yu, and Z. Cai, 'Attention mechanism-based model for short-term bus traffic passenger volume prediction', *IET Intelligent Transport Systems*, vol. 17, no. 4, pp. 767–779, Apr. 2023, doi: 10.1049/itr2.12302.

[27] M. H. Rahmatul Kholiq, W. Wiranto, and S. Widya Sihwi, 'News classification using light gradient boosted machine algorithm', *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 1, p. 206, Jul. 2022, doi: 10.11591/ijeecs.v27.i1.pp206-213.

[28] D. Tu, F. Luo, D. Wang, and Y. Cai, 'flexCAT: Computerized Adaptive Test Development Platform', *Chinese/English Journal of Educational Measurement and Evaluation*, vol. 4, no. 1, Jun. 2023, doi: 10.59863/GXZD9076.

[29] Z. Wang, C. Cao, and Y. Zhu, 'Entropy and Confidence-Based Undersampling Boosting Random Forests for Imbalanced Problems', *IEEE Trans Neural Netw Learn Syst*, vol. 31, no. 12, pp. 5178–5191, Dec. 2020, doi: 10.1109/TNNLS.2020.2964585.

[30] O. Almomani, 'A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms', *Symmetry (Basel)*, vol. 12, no. 6, p. 1046, Jun. 2020, doi: 10.3390/sym12061046.

[31] H. Polat, M. Türkoğlu, O. Polat, and A. Şengür, 'A novel approach for accurate detection of the DDoS attacks in SDN-based SCADA systems based on deep recurrent neural networks', *Expert Syst Appl*, vol. 197, p. 116748, Jul. 2022, doi: 10.1016/j.eswa.2022.116748.

[32] A. Thangasamy, B. Sundan, and L. Govindaraj, 'A Novel Framework for DDoS Attacks Detection Using Hybrid LSTM Techniques', *Computer Systems Science and Engineering*, vol. 45, no. 3, pp. 2553–2567, 2023, doi: 10.32604/csse.2023.032078.

[33] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, 'Automated DDOS attack detection in software defined networking', *Journal of Network and Computer Applications*, vol. 187, p. 103108, Aug. 2021, doi: 10.1016/j.jnca.2021.103108.