applied sciences

MDPI

Article

Ensemble Shields: Optimizing DDoS Detection in IoT Networks Using Machine Learning

Abdul Ghani Ansari [1,*], Fareed Ahmed Jokhio [2] and Aijaz Ahmed Arain [3]

[1] Department of Information Technology, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah 67450, Sindh, Pakistan
[2] Department of Computer Systems Engineering, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah 67450, Sindh, Pakistan; fajokhio@quest.edu.pk
[3] Department of Computer Science, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah 67450, Sindh, Pakistan; aijaz@quest.edu.pk
* Correspondence: agansari@quest.edu.pk

Abstract

DDoS (Distributed Denial-of-Service) attacks are significant cyber threats that affect network infrastructures and cause massive service disruption. DDoS attacks have become even more threatening, owing to the poor security of IoT devices. This paper proposes two ensemble approaches: (1) Optimized Ensemble Stacking Approach 4 (OESA-4) and (2) Optimized Ensemble Voting Approach 4 (OEVA-4). Both proposed approaches use four supervised machine learning classifiers, such as K-Nearest Neighbors (KNN), Random Forest (RF), Extreme Gradient Boosting (XGB), and Decision Tree (DT). OESA-4 combines the base learners' outputs via a meta-learner, and OEVA-4 uses weighted soft-voting, assigning different weights to base learners to improve the output's efficiency. An experimental analysis was conducted on a subset of the CIC-DDoS2019 dataset offered by the Canadian Institute for Cybersecurity to test the performance of state-of-the-art classifiers and proposed approaches. This subset of the dataset contains 325,965 occurrences. The results show that the proposed OESA-4 and OEVA-4 demonstrated reliable performance with accuracies of 99.87% and 99.73%, respectively, on a single split, whereas stratified five-fold cross-validation resulted in mean accuracies of 99.51% $\pm$ 0.08% for OESA-4 and 99.28% $\pm$ 0.11% for OEVA-4 respectively. The relatively low errors in both cases are indicative of consistent performance in all the five folds tested, demonstrating strong and consistent classification capabilities within the experiments. Thus, the presented paper shows that OESA-4 and OEVA-4 approaches achieve higher performance in identifying various types of DDoS attacks compared to individual classifiers.

Keywords: Optimized Ensemble Stacking Approach 4 (OESA-4); Optimized Ensemble Voting Approach 4 (OEVA-4); Extra Tree Classifier (ETC); K-Nearest Neighbors (KNN); Random Forest (RF); Decision Tree (DT); Extreme Gradient Boosting (XGB)

Check for updates

Academic Editor: Christos Bouras

Received: 28 August 2026
Revised: 17 September 2026
Accepted: 20 September 2026
Published: 28 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

In today's digital environment, the rapid growth of IoT technology has led to significant developments in areas such as communication, automation, and data driven services. Enhanced productivity, connectivity, and real-time operation have enabled applications ranging from smart homes and healthcare systems to large-scale critical infrastructure networks [1]. Although there have been many technological innovations, the cyber threat landscape has expanded to present many difficulties related to cybersecurity in today's

world [2,3]. Traditional security methods, which are based on closed and relatively simple networks, are incapable of ensuring safety and security in current conditions. Therefore, today's interconnected systems have become more vulnerable to attacks because of their open nature and the possibility of exploiting live traffic flows. DDoS attacks are one of the most common forms of attack that can be executed against such interconnected systems, and their detrimental effects continue to grow in size, frequency, and complexity [4,5]. They involve sending massive amounts of traffic aimed at networks or services, thereby exhausting computational resources and bandwidth capacity, which leads to prolonged periods of unavailability, significant losses, and serious damage to the reputation of a particular organization [6,7]. In turn, another challenge is related to the vulnerabilities of IoT devices due to their poor protection, low processing power, and lack of security measures [8,9]. As a result, attackers can easily compromise them and gather a botnet consisting of many devices. Once assembled, such botnets can conduct coordinated cyberattacks and harm different industries, critical infrastructures, and people's well-being.

DDoS attacks have different traffic characteristics based on their attack techniques. In terms of volumetric attacks, there is an increase in traffic volume that exhausts the network bandwidth and/or resource capacity, while protocol attacks target weaknesses in the network/transport protocols [10,11]. In contrast, application attacks might generate traffic that appears similar to normal traffic, which makes them more difficult to distinguish [10,12]. Such differences make it extremely difficult to detect DDoS attacks in IoT environments due to the presence of heterogeneity, class imbalance, redundant features, overlapping of traffic patterns for benign and malicious traffic, and different decision-making biases of classifiers [13]. As a result, one classifier will exhibit different performance for different traffic patterns. To overcome these issues, the proposed approaches use SMOTE to solve class imbalance problems, apply an Extra Trees-based feature selection technique to eliminate redundant features and use four heterogeneous classifiers—Decision Tree, Random Forest, KNN, and XGBoost—to cover traffic patterns from different angles. The resulting predictions are combined using two decision fusion approaches: Logistic Regression stacking in OESA-4 and weighted soft voting in OEVA-4. Thus, the proposed techniques specifically address the problems of imbalanced, heterogeneous, and complex DDoS traffic. These proposed approaches are discussed in the Section 3 of this paper.

This paper analyzes several types of DDoS attacks that pose significant threats to the operation of modern networks and require prompt mitigation and protection. A UDP Flood Attack refers to malicious UDP packets being sent by an adversary to specific ports and overwhelming the targeted device with traffic. The MSSQL attack targets the Microsoft SQL database in order to exfiltrate, modify, or disrupt its content. The SYN Flood Attack represents another attack vector that exploits the TCP handshake process through a massive amount of SYN that do not complete connections and use up the server's resources until it becomes unavailable. Apart from these vectors, application-layer attacks, such as LDAP and Portmap attacks, that leverage vulnerabilities in the Lightweight Directory Access Protocol and Remote Procedure Calls, respectively, to gain access or disrupt services and data, and NetBIOS attacks, exploiting flaws in the NetBIOS protocol to extract user account information, pose significant threats to network operation and security. Unfortunately, existing DDoS detection systems based on signatures, rules, thresholds, and statistics are not sufficient to counteract these threats, since they fail to detect new sophisticated malicious activity and cannot operate in dynamic large networks, where false positives and false negatives are common [14,15]. Furthermore, with the help of machine learning algorithms, more accurate DDoS detection is now possible due to their ability to model network traffic flow and detect anomalies. Nonetheless, individual classifiers often have problems with overfitting, instability, and lack of robustness in dealing with IoT

traffic, hence the emergence of various ensemble algorithms. By combining multiple base learners' predictions to compensate for individual algorithms' bias and variance, ensembles significantly improve model performance [16,17].

In this study, approaches to DDoS detection, namely the Optimized Ensemble Stacking Approach 4 (OESA-4), based on a stacking approach, and the Optimized Ensemble Voting Approach 4 (OEVA-4), based on the weighted soft voting framework, are developed. These proposed approaches combine four heterogeneous classifiers, namely K-Nearest Neighbors, Random Forest, Decision Tree, and Extreme Gradient Boosting. In the OESA-4 framework, a meta-learner combines the predictions from various learners to increase performance in diverse traffic conditions, while the OEVA-4 uses weighted soft voting, which includes assigning diverse weights to dissimilar base learners to enhance the effectiveness of the output. The suggested methods are marked by high performance and throughput suitable for IoT environments. The main contributions of this study may be outlined as follows:

1. The OESA-4 model, a heterogeneous stacking framework, is proposed by integrating KNN, RF, DT, and XGB. Logistic Regression acts as the meta-learner for combining predictions from base classifiers to produce an optimized outcome.
2. The OEVA-4 model is proposed, which combines the same heterogeneous classifiers to balance bias and variance. The model emphasizes strong learners and maintains diversity by applying appropriate weights [1,2,3,5]. This weighting yields better accuracy than equal voting or a base classifier.
3. Training time evaluation is conducted to measure computational cost and ensure model applicability for IoT devices with limited resources.
4. A comparison of computational complexities is conducted to evaluate the compromise between the adaptability of the stacking system and the simplicity of the weighted voting system.
5. The leakage control approach is clearly explained in terms of preprocessing, class balancing using SMOTE, feature selection using ETC, training models, creating ensembles, and evaluation.
6. The proposed OESA-4 and OEVA-4 show excellent results in terms of accuracy, precision, recall, and F1-score, achieving 99.87% and 99.73%, respectively, while stratified five-fold cross-validation yields mean accuracies of 99.51% $\pm$ 0.08% and 99.28% $\pm$ 0.11%, respectively, by avoiding false alarms and surpassing both individual classifiers and other ensembles on a limited number of instances. Thus, the proposed approaches can be effectively used in practice for DDoS intrusion detection.

The rest of this paper is structured as follows: Section 2 provides background on DDoS detection techniques and ensemble algorithms. Section 3 presents the architecture and workflow of the OESA-4 and OEVA-4 models. Section 4 describes the experiments, datasets used in this paper, and metrics for measuring results. Section 5 explores findings and comparisons with other methods.

2. Related Work

The Internet of Things (IoT) is a definite evolution of modern technology which allows for the creation and maintenance of systems that achieve better performance and less human interference by means of automation within a range of services. As for IoT environments, security is not limited to DDoS attack detection and availability but also implied confidentiality, integrity, and authentication, as well as multimedia data protection for data such as images or videos [18]. The proposed solutions here concern mostly availability and aim at detection and classification of the relevant traffic; therefore, they can be considered additional to other security measures, rather than complete IoT security solutions. The most relevant related work is discussed here, such as that of Bala and

Kaur [19], who suggested an ensemble learning approach compared with SVM and RF on three datasets. The average accuracy of the ensemble equals 92%, outperforming SVM (88%) and RF (89%). Also, the proposed ensemble achieves a higher sensitivity of 90%, specificity of 93%, and an F1-score of 90% in a computationally efficient way. Okey et al. [20] proposed BoostedEnML, which is an ensemble-based approach based on five boosted classifiers (DT, RF, Extra Trees, LightGBM, and XGBoost) tested on CSE-CICIDS2018 with accuracies ranging from 98.3% to 98.9%, providing reliable results for IoT network security. Venkatraman et al. [21] proposed a self-attention-based weighted ensemble of CNN, XGBoost, LSTM, and RF algorithms used to fuse multiple-scale spatial–temporal features through an attention approach, attaining a high accuracy rate of 98.69%. The approach can be considered a huge improvement compared to existing baselines. Alqarni [22] suggested the use of a majority voting ensemble of Decision Tree, Naive Bayes, SVM, and KNN algorithms in cloud computing for detecting DDoS attacks, with an accuracy of 98.02%, sensitivity of 97.45%, and specificity of 98.65%. Ali et al. [23] introduced a stacking ensemble with MLP and SVM used as base models and RF as the meta-classifier for DDoS attack detection in SDNs, achieving an accuracy of 98.89% on test data. Awad et al. [24] presented an advanced approach to intrusion detection in ensemble learning through majority voting and weighted averaging. The authors tested an advanced approach on the CICIDS2017 and CICIDS2018 datasets. The results showed that the weighted average approach achieved a higher accuracy of 98.78% compared to that of majority voting, 87.34%. Gaur and Kumar [25] developed a hybrid method of feature selection involving chi-square, ANOVA, and Extra Trees together with four classifiers (RF, DT, KNN, and XGBoost) on the CICDDoS2019 dataset in the cloud environment for enabling the early and efficient detection of IoT-targeted DDoS attacks. The method reduced the feature space by 82.5% and achieved 98.34% accuracy. Roopak et al. [26] presented a hybrid of CNN and LSTM models, achieving 97.16% accuracy on CICIDS2017, outperforming both deep learning and conventional ML baselines. Other notable works include Salman et al. [27], who developed a system for identifying IoT devices and detecting malicious traffic; the accuracy achieved on the testing stage was 97% for malicious traffic, 94.5% for device-type classification, and 93.5% for traffic-type classification. Ibrahim et al. [28] developed a multilayer approach to DDoS attack detection based on the CTU-13 dataset by using KNN, SVM, and MLP classifiers, with a maximum accuracy of 91.51%. Yaser et al. [29] employed a stacked autoencoder and feedforward neural networks to detect DDoS attacks with over 99% accuracy on the ISCXIDS-2012 and UNSW-2018 datasets and 97.68% in a real-world scenario, showing generalizability with no overfitting. Abdulrahman and Ibrahem [30] proposed a host-based IDS system with C5.0 and Random Forest classifiers applied to CICIDS2017 and achieved average accuracies of 86.80% and 86.45%, respectively. Mehmood et al. [31] applied an autoencoder plus ML pipeline, achieving over 99% accuracy on CIC-DDoS2019 via combined unsupervised feature extraction and classification. Gupta et al. [32] developed a lightweight, local-router-based ML approach for IoT networks with 99% accuracy, being able to operate robustly. Among other comparative studies, there is the work of Abiramasundari and Ramaswamy [33], who compared various classifiers on CICIDS2017, CICDDoS2019, and CICIDS2018; they found RF to achieve 98.9% accuracy on CICIDS2017, RF and KNN to yield 98.7% on CICDDoS2019, and SVM to show 98.7% on CICIDS2018. Bindraa and Sood [34] evaluated the best supervised ML algorithms on real-world datasets, showing accuracies of 82%, 94%, and 96% for Logistic Regression, KNN, and RF, respectively. Lastly, Elejla et al. [35] assessed ICMPv6-based DDoS attack detection via five classifiers, discovering that KNN yielded the highest accuracy of 85.7% with minimum false positives (0.171) and the fastest processing time (0.12 s), while SVM showed the lowest accuracy and NN yielded the lowest efficiency.

3. Methodological Framework

The proposed approach includes the implementation of a complex machine learning pipeline tailored specifically for DDoS attack detection. The methodological framework can be viewed in Figure 1. The algorithm used in the classification procedure is presented sequentially. Initially, some portion of the CIC-DDoS19 dataset is processed with preprocessing involving such operations as cleansing, noise reduction, and gap filling. The data are then split into training (80%) and testing (20%) sets, thus improving the evaluation process and avoiding overfitting. After that comes the application of the Synthetic Minority Oversampling Technique (SMOTE) to deal with the problem of class imbalance, which is applied to the training dataset. The SMOTE method, unlike oversampling through simple replication, creates synthetic samples by interpolation between existing observations. In this study, SMOTE was applied only to training data to address class imbalance. This technique helped to improve the proportion of less common attacks within the training data, making the machine learning models unbiased towards certain classes [36]. The next step is selection of features via the Extra Trees classifier. From this, four base models—DT, RF, KNN, and XGB—are trained using the selected features. The predictions from these models are then combined via two ensemble approaches, with the use of Logistic Regression as a meta-classifier in OESA-4 and soft voting with classifier class probability weights for better predictions in OEVA-4. Finally, the two ensembles are tested against the unseen data to make their final predictions. All the base classifiers are improved in terms of hyperparameters using the training set and evaluated in terms of accuracy, precision, recall, and F1-scores. A comparative analysis of all classifiers' results highlights their specific strengths and limitations in detecting various types of attacks.

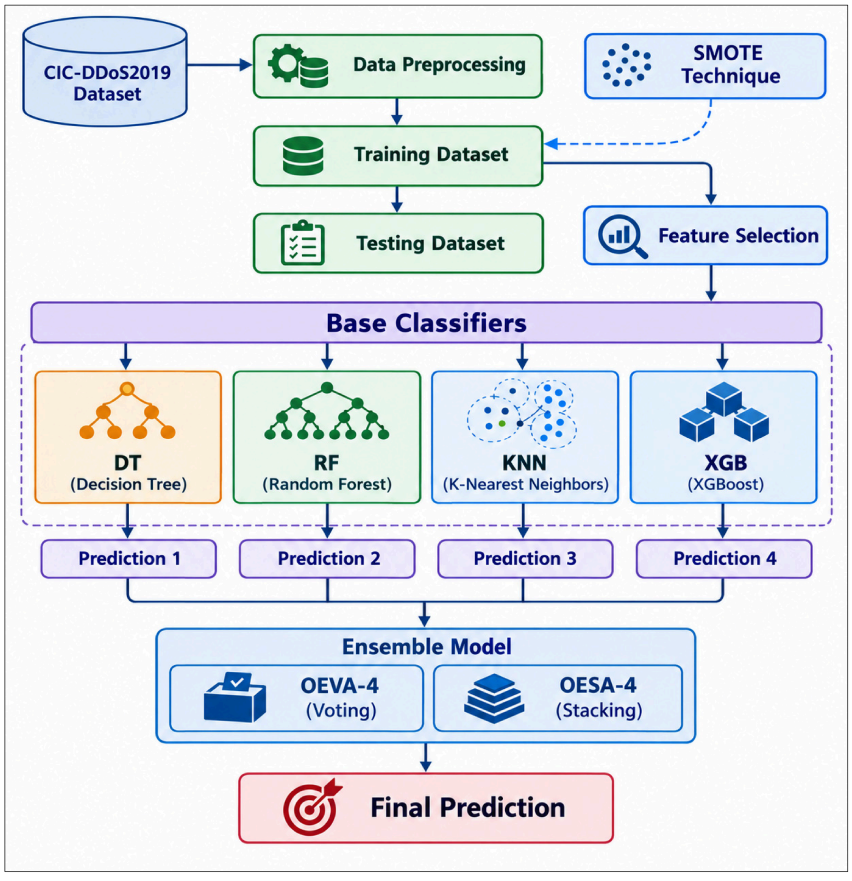


Figure 1. Methodological framework for DDoS detection.

3.1. Dataset

A subset of the CIC-DDoS2019 dataset is used in this work. This dataset was generated by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick, Canada. The dataset includes millions of samples covering a wide variety of DDoS attacks and normal traffic with 88 characteristics. CIC-DDoS2019 was accessed from https://www.kaggle.com/datasets/rodrigorosasilva/cic-ddos2019-30gb-full-dataset-csv-files (accessed on 23 March 2026).

This dataset is regarded as one of the benchmarking datasets for evaluating and designing solutions against DDoS attacks since it enables realistic modeling of current traffic. This dataset offers an accurate representation of contemporary attacks [37]. In the current experiment, a selected part of the full dataset was chosen on the basis of data quantity and the suitability of the dataset for the research objective.

The dataset contains both normal traffic and different DDoS attack samples, including UDP Flood, MSSQL, SYN Flood, Portmap, NetBIOS, and LDAP attacks. The selected part of the dataset contained approximately 325,965 network flows, of which 16.3% of the flows represented normal traffic of the attacks, 15.9% were SYN attacks, 14.6% were MSSQL attacks, 15.2% were Portmap attacks, 9.6% were NetBIOS attacks, 12.8% are LDAP attacks, and 15.6% were UDP attacks. All records in the dataset are flow-based and capture information related to the temporal and statistical behavior of the network traffic. Some of the important attributes in the dataset include Flow ID, Timestamp, Destination Port, Flow Duration, Minimum Packet Length, Forward Packet Length Mean, ACK Flag Count, and Average Forward Segment Size. These features constitute a feature set that is effective in enabling machine learning classifiers to distinguish normal traffic from abnormal behaviors in the network. In particular, they are helpful in detecting LDAP-based abnormalities, UDP attacks, and port scan attacks. The dataset has been widely used in past cybersecurity research work, indicating its usefulness in the evaluation of supervised machine learning models and ensemble techniques [38].

Table 1 indicates the ranking order of the top 20 important features that were used to train the classifier through the use of ETC. Timestamp, ACK Flag Count, and Fwd Packet Length Min are the three most important features, which are very instrumental in helping distinguish benign activity from DDoS attacks. Other supportive features like Min Packet Length, Average Packet Size, and Inbound are equally significant and help in recognizing more subtle behavior.

Table 1. Feature selection using the Extra Trees classifier (ETC).

Rank	Feature Name	Importance Score	Rank	Feature Name	Importance Score
1	Timestamp	0.1811	11	Flow_ID	0.0439
2	ACK_Flag_Count	0.0789	12	Protocol	0.0359
3	Fwd_Packet_Length_Min	0.0724	13	Max_Packet_Length	0.0348
4	Min_Packet_Length	0.0662	14	Fwd_Packet_Length_Max	0.0342
5	Average_Packet_Size	0.0643	15	Flow_Bytes/s	0.0279
6	Inbound	0.0641	16	Flow_Packets/s	0.0179
7	Source_Port	0.0632	17	Subflow_Fwd_Bytes	0.0150
8	Avg_Fwd_segment_Size	0.0563	18	Destination_Port	0.0108
9	Fwd_Packet_Length_Mean	0.0520	19	Fwd_Packets/s	0.0105
10	Packet_Length_Mean	0.0500	20	Total_Length_of_Fwd_Packets	0.0086

3.2. Data Visualization

Figure 2 represents the traffic label distribution in a subset of the CIC-DDoS2019 dataset [39]. This allows us to understand the pattern of distribution of classes within the dataset by enabling us to determine the frequency of every label. The primary objective of this plot is to illustrate the distribution of classes within a dataset. It will enable us to identify whether there is a class imbalance in our dataset by simply looking at the count of each traffic label—that is, at how many of them exist. The number of instances of each traffic label is shown in Figure 2, which is very important for understanding the structure of the data. It helps in detecting any significant class imbalance that could have an impact on the machine learning model's performance. The figure clearly shows that the benign traffic count is greater than the counts of the attack classes.

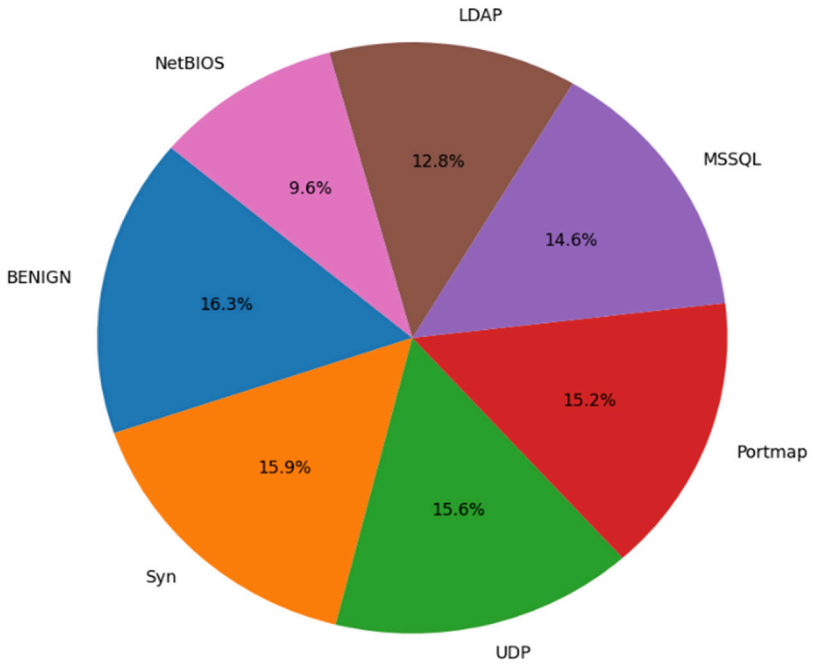


Figure 2. Distribution of traffic labels.

3.3. Preprocessing

An appropriate data preprocessing technique determines the accuracy, robustness, and effectiveness of machine learning models for DDoS attack detection. In this study, the following data preprocessing steps were employed on a subset of the CIC-DDoS2019 dataset. First, a subset of the dataset was uploaded and imported as a DataFrame to verify the consistency of data types and feature names, as well as the correctness of its structure. To create a diverse dataset reflecting different attack patterns, any column names containing unnecessary spaces at the beginning and the end were corrected by using df.columns.str.strip(). Missing data was checked with df.isnull(), while df.dropna (inplace = True) was used to get rid of any rows containing missing values. While this led to a slight decrease in the size of the data, it provided certainty that no complicated imputation of traffic features would be needed. Additional redundant columns, like "Unnamed: 0," created during CSV exporting, were dropped. Then, all categorical variables were encoded by means of the Label Encoding technique, allowing for the conversion of any non-numeric value into its numeric representation required for use in supervised learning. Further on, the data was separated into features and target labels by splitting it into X and Y parts, where the Label column represented the target variable for classification. For further splitting into train and test samples, a stratified train–test split was performed using the

train_test_split function with random_state = 42. The training data underwent K-fold cross-validation. The SMOTE method was shaped with k_neighbors = 5 and it was applied to each training fold to solve the problem of unbalanced classes. As a result, a larger number of patterns could be learned by the classifiers, which increased their predictive capabilities. Then, to increase effectiveness, feature selection was performed with the help of ETC on each training fold.

3.4. Supervised Learning Models

Four popular machine learning (ML) classifiers were applied for the purposes of detecting DDoS attacks. The models were initially tested independently before being incorporated into the two ensemble techniques, OESA-4 and OEVA-4, where hyperparameters were tuned to increase the classifiers accuracy. The following brief explanations provide an insight into each method.

3.4.1. Decision Tree (DT)

A decision tree refers to a decision-based approach that divides the dataset into increasingly smaller sets according to decisions made at each level. Internal nodes are used for making decisions, while leaf nodes are used to predict the final class. In this case, at each node, the algorithm chooses the attribute xj (e.g., Gini index). The Gini index is given by

$$\text{Gini(S)} = \sum_{i=1}^{c} p_i^2 \quad (1)$$

where p_i represents the value that denotes the ratio of sample points within class i out of all c classes. The construction of the decision tree follows the strategy of recursively choosing the split on a feature that produces the maximum amount of information gain. The DT is easy to interpret and can capture nonlinear relationships [40].

3.4.2. Random Forest (RF)

The Random Forest approach represents an ensemble learning method where a number of decision trees work together to enhance the performance of the model and avoid overfitting [41]. The training process involves a bootstrapped sample of the data, and for every split, only a random selection of the features is used to increase diversity. The predicted class is made through a majority vote and mathematically formulated as

$$\hat{y} = \text{mode}\{\text{T1(x)}, \text{T2(x)}\ldots, \text{TM(x)}\} \quad (2)$$

In this model, $\hat{y}$ denotes the final prediction of an ensemble of decision trees. Where $TM\ (x)$ denotes the prediction made by the mth decision tree for input instance x, and M is the total number of trees in the ensemble. This ensemble approach provides stability against noise and performs well with high-dimensional data [42,43].

3.4.3. K-Nearest Neighbors (KNN)

KNN is a kind of non-parametric prediction algorithm that works on the basis of similarity to existing observations. It selects the k-nearest points to a new observation in the feature space, which are called neighbors, and predicts the most common class of those neighbors [44]. The distances between observations are calculated through Euclidean distance. The Euclidean distance between a query point x and a training point xi is expressed as

$$d_E(\text{x}, \text{xi}) = \sqrt{\sum_{j=1}^{n} (xj - xij)^2} \quad (3)$$

where x = ($x1$, $x2$, . . ., xn) represents the feature vector of the query sample and xi = ($xi1$, $xi2$, . . ., xin) is the feature vector of the ith training sample. The expression ($xj - xij$)2 computes the squared difference between the jth feature of the query sample and the respective feature of the training sample. Summation of those over all n features leads us to the squared distance, and taking its root will make the distance similar to the scale of the initial features. KNN's efficiency means it is the better choice for such purposes as anomaly detection or pattern recognition. The efficiency of the algorithm relies on choosing k and a metric for defining locality in the decision-making process [45].

3.4.4. Extreme Gradient Boosting (XGB)

Extreme Gradient Boosting, also known as XGBoost, is an optimized implementation of gradient boosting where the decision trees are generated in sequence, each one attempting to correct for the mistakes of the previous one. When used for classification tasks, XGBoost utilizes a multi-class log-loss function (SoftMax objective), allowing for differentiating between classes by generating probabilities [46,47]. It optimizes a normalized objective function that combines prediction loss and model intricacy:

$$l = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t) \tag{4}$$

where $l(y_i, \hat{y}_i)$ is the loss function (softmax cross-entropy for multi-class classification), f_t is the tth tree, T is the number of trees, and $\Omega(f_t)$ is the regularization term, which determines the model's complexity. The regularization term is described as

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_j^2 \tag{5}$$

where γ penalizes the number of leaf nodes, T is the number of leaf nodes in the tree, w_j is the weight of the jth leaf node, and λ is the L2 regularization parameter to avoid overfitting. It uses Newton's approximation, which means it considers both the gradient and the Hessian when making split decisions. It further improves performance through shrinkage (learning rate), subsampling, and column sampling. Missing values are automatically handled by learning the optimal default split direction. After constructing all trees, XGBoost predicts multi-class probabilities using the SoftMax function

$$P(y = k|x) = \frac{e^{\hat{y}k(x)}}{\sum_{j=1}^{K} e^{\hat{y}j(x)}} \tag{6}$$

where $\hat{y}k(x)$ is the total score for the class k obtained by summing the contributions of all trees, and K is the number of classes [48,49]. XGBoost is widely used in large-scale multi-class tasks, including network intrusion detection and DDoS attack classification.

3.5. Proposed Ensemble Approaches

The proposed OESA-4 and OEVA-4 ensemble approaches combine the predictions of diverse individual base models to get the highest accuracy and robustness compared to other state-of-the-art models.

3.5.1. The Proposed OESA-4 Ensemble Approach

The OESA-4 approach can be considered as a suggested methodology for DDoS attack detection, as it is based on a stacking ensemble method to leverage the combined advantages of various supervised learning models. Stacking is a sophisticated ensemble method where four base models are initially trained and then a meta-learner is trained

based on the predictions of these base models to make the final prediction [50,51]. To avoid information leakage when building the meta-classifier, OESA-4 uses K-fold cross-validation to generate out-of-fold (OOF) predictions. In each fold, the base classifier trains the model on K − 1 folds and makes predictions only on the held-out fold. The collected predictions in all folds are joined to get the final set of OOF meta-features. Consequently, the meta-classifier is trained on predictions of samples which have not been used to build base classifiers [52].

$$\text{Let} \quad h_m(\text{x}),\ m = 1, 2, \ldots, \text{M} \tag{7}$$

denote the probability vector generated by the mth base classifier on a particular input instance. For OESA-4, $M = 4$, corresponding to the base learners. The concatenation of the probability values outputted by the four base classifiers constitutes the meta-feature vector in the out-of-fold (OOF) phase:

$$\mathbf{z} = [h_{TD}(\text{x}), h_{RF}(\text{x}), h_{KNN}(\text{x}), h_{XGB}(\text{x})] \tag{8}$$

where z represents the meta-feature vector formed from the probability outputs of the four base classifiers.

The meta-classifier H is trained using these OOF meta-features and their corresponding true class labels. The general stacking formulation can be expressed as

$$\hat{\text{Y}} = H(\text{z}) = [h_{DT}\ (\text{x}), h_{RF}\ (\text{x}), h_{KNN}\ (\text{x}), h_{XGB}\ (\text{x})] \tag{9}$$

where H stands for the Logistic Regression meta-classifier and $\hat{\text{Y}}$ denotes the output class.

In cases where the meta-classifier is Logistic Regression, the probability for class c is found using SoftMax as follows:

$$P(y = c|x)\frac{exp\left(\beta_c^T z + b_c\right)}{\sum_{r\in C} exp\ \left(\beta_c^T\ z + b_r\right)},\ \text{c}\ \in\ \text{C} \tag{10}$$

where z refers to the meta-features vector, β_c is the coefficients vector for class c, b_c is the intercept for class c, and C refers to the set of candidate classes.

The final prediction is obtained by selecting the class with the highest probability:

$$\hat{\text{y}} = \underset{c\in C}{\text{argmax}} P(y\ = c|z) \tag{11}$$

If $\hat{\text{y}}$ is the predicted class at the end, then the argmax function chooses the class with the highest probability, whereas P (y = c | z) indicates the probability that the meta-learner assigns to a class c for sample z, where z is the meta-feature vector constructed by combining the outputs of the base classifiers. The stacking method leverages the complementarity of the base classifiers, which helps the meta-model to correct the weakness of each classifier and identify nonlinear relationships between them [53,54]. The OESA-4 approach integrates base learners due to their diverse learning strategies, where DT and RF efficiently capture nonlinear interactions, KNN identifies localized anomalies and clusters, and Logistic Regression serves as the meta-learner for final decision aggregation due to its simplicity and interpretability. This model has improved detection accuracy, and overfitting issues are taken care of. Despite being computationally expensive, the computational cost involved with the stacking model is not too high given the level of improvement in accuracy. The experimental results show that this proposed approach beats other individual models on accuracy, precision, recall, and F1-score. Figure 3 shows the operation process of the proposed OESA-4 model.

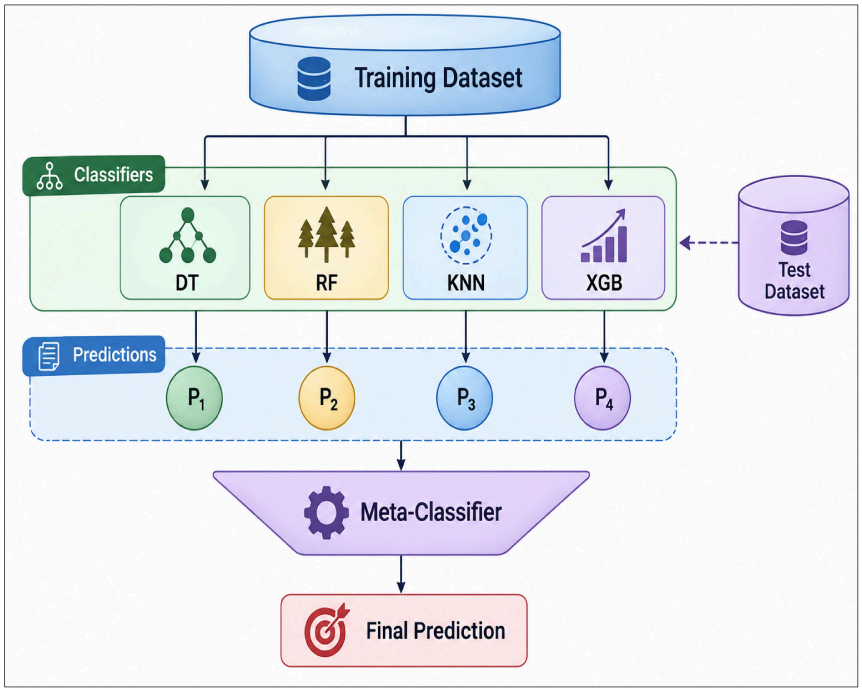


Figure 3. Proposed OESA-4 ensemble model.

Figure 3 describes the working process flow of the proposed OESA-4 model that starts from preprocessed data from a subset of the CIC-DDoS2019 dataset. Features in this dataset help differentiate between normal and abnormal traffic flows. Then, the preprocessed data is sent to the base classifiers, each having its own analytical purpose. The first one, the Decision Tree classifier, works in the context of rule-based machine learning, and the second one, Random Forest, consists of many decision trees. To build a robust algorithm, the K-Nearest Neighbors classifier classifies samples based on similarity; XGBoost integrates multiple model forecasts to generate its final prediction. All the forecasts made by the base classifiers are used as features in creating a new feature vector, which is passed through to the Logistic Regression classifier, serving as the meta-classifier. It serves as the heart of the stacking ensemble and uses base classifier results in its final decision-making process. Thus, at the end of the procedure, the system classifies the flow of network traffic as DDoS or normal traffic, resulting in an increased detection rate and decreased number of false positives, as well as more generalized output.

As shown in Algorithm 1, the proposed methodology follows a three-step classification pipeline to improve the performance of the suggested ensemble classifier. First the training data is divided into stratified folds such that the original class distribution is preserved. During each iteration, the training folds are first balanced using SMOTE and then the most relevant features are selected using ETC. The four base classifiers are then trained on the selected features and predict the corresponding hold-out fold. These class probability predictions form the out-of-fold (OOF) predictions which are used to build the meta-dataset. After all folds have been exhausted, the OOF predictions are then used to train the meta-classifier that learns to combine predictions from the four base models. Finally, the base models are re-trained on the entirety of the preprocessed training data. For every test instance, the probability predictions are then fed to the meta-classifier to get the final DDoS classification.

Algorithm 1: The proposed OESA-4 approach.

1: Input: Split into Training data, (TD_{train}), Test data (TD_{test}), oversampling method SMOTE, Feature Selector *ETC*, Base classifiers T = {T_{DT}, T_{RF}, T_{KNN}, T_{XGB}), Meta-Classifier M_C and K-fold stratified cross-validation.
2. Output: Optimized Stacking Ensemble Classifier *EC*
3. ***Step 1:*** *Divide* TD $_{\text{train}}$ into *K* stratified folds.
4. ***Step2:*** *for each fold k* = 1, . . ., *K*
5. *Use K* − 1 folds for training and the remaining fold for validation.
6: Apply ***SMOTE*** *to* K − 1 training folds to obtain TD$^{(k)}$ $_{\text{bal}}$
7: Apply *ETC on* TD$^{(k)}$ $_{\text{bal}}$ for feature selection and obtain TD$^{(k)}$ $_{\text{final}}$
8. Apply k-fold cross-validation on the training data for each base model
9. Train M_{DT}, M_{RF}, M_{KNN}, *and* M_{XGB} using TD$^{(k)}$ $_{\text{final}}$
10. Use the trained base classifiers to predict the corresponding held-out fold.
11. For each validation sample, generate the OOF probability vector:

$$z_i = [\boldsymbol{p_{DT}}(\mathrm{y}|\boldsymbol{a_i},), \boldsymbol{p_{RF}}(\mathrm{y}|\boldsymbol{a_i}), \boldsymbol{p_{KNN}}(\mathrm{y}|\boldsymbol{a_i}), \boldsymbol{p_{XGB}}(\mathrm{y}|\boldsymbol{a_i})]$$

12. Append $(\boldsymbol{Z_i}, \boldsymbol{a_i})$ to D $_{\text{meta}}$
13. End For
14. step3: Train the meta-classifier *MC* using the complete OOF meta-dataset D $_{\text{meta}}$
15. Apply SMOTE to the complete TD $_{\text{train}}$ and perform ETC feature selection to obtain the final training dataset TD $_{\text{final}}$.
16. Train M_{DT}, M_{RF}, M_{KNN}, *and* M_{XGB} on the complete TD $_{\text{final}}$
17. For each test sample $\boldsymbol{x_j} \in$ TD $_{\text{test}}$, generate the probability vector:

$$z_j = [\boldsymbol{p_{DT}}(\mathrm{y} \mid \boldsymbol{x_j},), \boldsymbol{p_{RF}}(\mathrm{y} \mid \boldsymbol{x_j}), \boldsymbol{p_{KNN}}(\mathrm{y} \mid \boldsymbol{x_j}), \boldsymbol{p_{XGB}}(\mathrm{y} \mid \boldsymbol{x_j})]$$

18. Provide $\boldsymbol{z_j}$ to the trained meta-classifier *MC*.
19. Generate the final prediction:

$$\hat{\mathbf{y}}_j = \underset{c \in C}{\mathbf{argmax}} \boldsymbol{P}(\boldsymbol{y} = \boldsymbol{c} | \boldsymbol{z_j})$$

20. return to *EC* = {M_{DT}, M_{RF}, M_{KNN}, M_{XGB}, *MC*}

3.5.2. The Proposed OEVA-4 Ensemble Approach

The proposed OEVA-4 approach involves the combination of outputs from four selected supervised learning algorithms via a soft voting classifier. Through the combination of predictions obtained from different base learners, this method allows for the creation of an ultimate prediction, taking advantage of the strong characteristics of the base learners while taking care of the disadvantages of each of them [55,56]. Class probabilities calculated by each learner in the ensemble are combined through soft voting that uses weighted average probabilities of classifiers with the following weight values [1,2,3,5]:

$$P_M\ (y = c|x) = \frac{\sum_{m=1}^{4} w_m\ P_m\ (y = c|x)}{\sum_{m=1}^{4} w_m} \tag{12}$$

where $P_M\ (y = c|x)$ refers to the final probability value obtained using the proposed OEVA-4 model to predict the class c for the given input traffic instance x. The notation $P_m\ (y = c|x)$ stands for the posterior probability prediction from the m-th base classifier, which predicts the given input x being from class c. The parameter w_m denotes the weight associated with the mth base learner. The term $\sum_{m=1}^{4} w_m\ P_m\ (y = c|x)$ denotes the sum of the weighted probabilities produced by all four base classifiers, whereas $\sum_{m=1}^{4} w_m$ represents the total

weight used to normalize the weighted probability. The class with the maximum probability is taken as the output:

$$\hat{y} = \underset{c \in C}{\text{argmax}} P_M\ (y = c|x) \tag{13}$$

Here, $\hat{y}$ denotes the final class predicted by OEVA-4. The term argmax represents to the maximum probability, while C refers to the target classes. The soft voting method results in better performance due to the use of probability generated by the model [55,57]. The performance of the proposed OEVA-4 approach in terms of accuracy, precision, recall and F1-score is higher than the individual learners. Figure 4 provides the working process flow of the proposed OEVA-4 model.

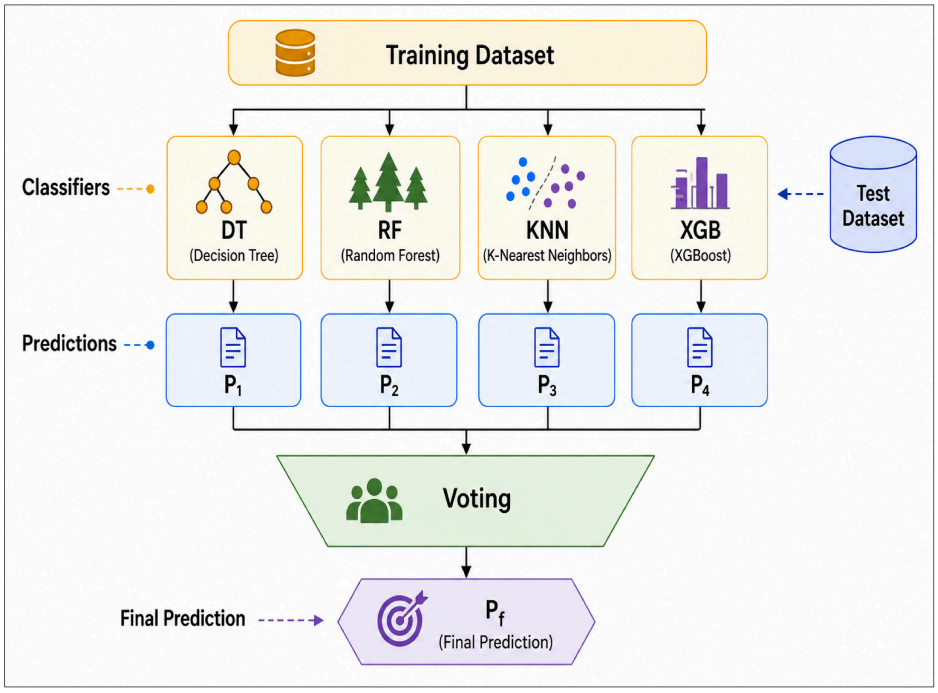


Figure 4. Proposed OEVA-4 DDoS detection model.

Figure 4 shows the OEVA-4 model, designed to improve performance through integrating classifiers using a voting classifier. The process starts with a training dataset used to train classifiers. Once trained, a test dataset is fed into these models to obtain separate predictions. These predictions are then aggregated using a voting technique, where a weighted compromise is determined to generate a robust final prediction denoted by P_f. Individual algorithm biases are less likely to affect the ensemble. This ensemble approach reduces the risk of biases and improves the overall performance compared to other single models.

As shown in Algorithm 2, the proposed methodology consists of a three-step classification pipeline that aims at resolving problems related to class imbalance, dimensionality, and variance in prediction. First, we split the dataset into training and testing sets; then SMOTE is employed to balance the minority classes of attacks compared to the majority class of traffic. This is followed by utilizing an ETC that detects and selects the k best features out of all possible ones to filter out noise and reduce the processing time. The four base classifiers are trained on the selected training data. Once a new sample of network traffic is available, each classifier produces an output probability for all the possible classes. The obtained probabilities are combined according to the weights [1,2,3,5] that have been established in advance, thus assigning different weights to different classifiers. The class with the largest weighted probability is selected as the final one. OEVA-4 makes use of

four different classifiers and combines their output in order to provide DDoS classification.

Algorithm 2: The proposed OEVA-4 approach.

1. Input: Training data **(TD_{train}), Test data (TD_{test}),** oversampling method SMOTE, Feature Selector *ETC*, Base classifiers T = {T_{DT}, T_{RF}, T_{KNN}, T_{XGB})
2. Output: Optimized Soft Voting Ensemble Classifier *EC*
3. **Step 1:** Apply *SMOTE to* TD to balance the minority classes → TD $_{\text{balanced}}$
4. Apply *ETC on* TD $_{\text{balanced}}$ to identify top *K* features
5: Transform TD $_{\text{balanced}}$ to TD $_{\text{test}}$ selected *K* features to obtain TD $_{\text{final}}$ and TD $_{\text{test}}$, *respectively*
6. **Step 2:** for each $T_j \in \{T_{DT}, T_{RF}, T_{KNN}, T_{XGB}\}$ do:
7. Train Model M_j on TD $_{\text{final}}$
8. Append M_j to the ensemble list *BL*
9. End for
10. **Step 3:** for a new input sample x (with *k* features) do:
11. Generate class-probability vectors P_{DT}, P_{RF}, P_{KNN}, P_{XGB}
12. Calculate weighted probability across classifiers using weights [1,2,3,5]:
$P_M\ (y\ =\ c|x) = \frac{\sum_{m=1}^{4} w_m\ P_m\ (y=c|x)}{\sum_{m=1}^{4} w_m}$
13. Final class prediction: $\hat{y} = \underset{c \in C}{\text{argmax}} P_M\ (y\ =\ c|x)$
14. Return *EC* = {M_{DT}, M_{RF}, M_{KNN}, M_{XGB}} with weighted soft-voting prediction

3.6. Evaluation Metrics

Performance assessment of the ensemble learning approaches and individual learners is done using the metrics discussed below. These performance assessment metrics can be applied on the basis of a confusion matrix, which will help to record correct predictions and errors in classification, describing classifier performance in terms of a group of observations that have known outcomes. Four main elements constitute a confusion matrix: True Negatives (TNs), meaning items whose both real and predicted values are negative; TPs (True Positives), examples where the actual and predicted values are both positive; FPs (False Positives), examples where there was a positive prediction although the actual value was negative; and FNs (False Negatives), situations where there was a negative prediction although the actual value was positive [58,59]. Using the confusion matrix, the following performance metrics can be used to measure how well a classification model performs.

1. Accuracy: Proportion of all correctly classified instances among total instances.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{14}$$

2. Precision: Proportion of correctly predicted positive instances among all predicted positive instances.

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{15}$$

3. Recall/Sensitivity: Proportion of actual positive instances correctly identified by the model.

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{16}$$

4. F1-Score: Harmonic mean of precision and recall, providing a balanced measure of both.

$$\text{F1-score} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (17)$$

5. ROC-AUC: Area Under the ROC (Receiver Operating Characteristic) Curve (AUC) is a measure that evaluates the discriminative ability of the classifier based on true positive rate vs. false positive rate for different thresholds. The higher the AUC value, the more reliable the system is [58,59]. The mathematical formulation of AUC is as follows:

$$\text{AUC} = \int_0^1 TPR(FPR)d(FPR) \quad (18)$$

These performance metrics allow a detailed examination of classification performance and enable comparative analysis of the performance achieved by the ensemble frameworks against individual classifiers.

3.7. Experimental Setup and Reproducibility Settings

All the experiments were performed using Python 3.12.5. The experiments were conducted using Google Colab, a cloud service that allows collaborative work and reproducibility of actions. Scikit-learn, which is one of the most popular public libraries for Python, offers efficient tools for the implementation and evaluation of classification algorithms. In order to conduct the experiments, we utilized a computer with 8 GB of memory and an Intel Core i5 CPU, which was quite sufficient for the current experiment. The key experimental settings and configuration parameters are provided in Table 2. These settings and this configuration provide a clear description of the experimental protocol used for both proposed approaches.

Table 2. Reproducibility settings and hyperparameter configuration.

Category	Component	Model/Technique	Configuration/Hyperparameters
Dataset	Dataset	CIC-DDoS2019	Subset containing 325,965 instances
Data Split	Train/Test Split	Stratified Split	80% training/20% testing; test_size = 0.20, stratify = y, random_state = 42
Preprocessing	Feature Scaling	StandardScaler	Standardization applied to input features
Preprocessing	Class Balancing	SMOTE	k_neighbors = 5, random_state = 42
Preprocessing	Feature Selection	ExtraTreesClassifier + SelectFromModel	n_estimators = 100, random_state = 42; top 20 features selected
Evaluation	Outer Cross-Validation	StratifiedKFold	n_splits = 5, shuffle = True, random_state = 42
Evaluation	Stacking Inner CV	StratifiedKFold	n_splits = 5, shuffle = True, random_state = 42; used for OOF meta-features
Optimization	Hyperparameter Tuning	GridSearchCV	5-fold inner stratified K-fold; scoring = 'f1_macro'
Hyperparameter Search	Decision Tree	DecisionTreeClassifier	max_depth = [none, 3, 5, 7, 10]; min_samples_split = [2, 4, 6]; min_samples_leaf = [1, 3, 5]; splitter = ['best', 'random']
Final Configuration	Decision Tree	DecisionTreeClassifier	max_depth = 5; min_samples_split = 4; min_samples_leaf = 3; splitter = 'random'; random_state = 42
Hyperparameter Search	Random Forest	RandomForestClassifier	n_estimators = [50, 100, 200]; max_depth = [None, 5, 10, 20]; max_features = ['sqrt', 'log2']

Table 2. *Cont.*

Category	Component	Model/Technique	Configuration/Hyperparameters
Final Configuration	Random Forest	RandomForestClassifier	n_estimators = 100; max_depth = None; max_features = 'sqrt'; random_state = 42
Hyperparameter Search	KNN	KNeighborsClassifier	n_neighbors = [5, 10, 20, 40, 50]; p = [1, 2]; leaf_size = [30, 50]
Final Configuration	KNN	KNeighborsClassifier	n_neighbors = 40; p = 2 (Euclidean); leaf_size = 50
Hyperparameter Search	XGBoost	XGBClassifier	n_estimators = [15, 50, 100]; max_depth = [1, 3, 5]; learning_rate = [0.01, 0.08, 0.1]; booster = ['gbtree', 'dart']
Final Configuration	XGBoost	XGBClassifier	n_estimators = 15; max_depth = 1; learning_rate = 0.08; booster = 'dart'; random_state = 42
Hyperparameter Search	OESA-4	Logistic Regression Meta-Learner	max_iter = [100, 500, 1000]
Ensemble Model	OESA-4	Stacking Classifier	Logistic Regression meta-learner; selected max_iter = 1000, random_state = 42
Hyperparameter Search	OEVA-4	Soft Voting Classifier	Candidate weights: [[1, 1, 1, 1], [1, 2, 3, 5], [2, 2, 1, 5]]
Ensemble Model	OEVA-4	Soft Voting Classifier	voting = 'soft'; selected weights [1, 2, 3, 5] mapped to DT, RF, KNN, XGB
Environment	Python	Python	Version 3.12.5
Environment	Libraries	Software Packages	scikit-learn 1.6.1; imbalanced-learn 0.14.2; XGBoost 3.4.1; NumPy 2.1.3; pandas 2.2.3

4. Results and Discussion

This study employs proposed stacking and voting ensemble approaches to classify both normal and attack traffic. The training and testing processes of the models were performed using a subset of the first 20 features obtained from ETC. Feature space reduction aims to minimize redundancy and focus on important, discriminative features of network traffic. Figure 5 shows the confusion matrices for these models. It can be stated that OESA-4 and OEVA-4 show a significant advantage over competing approaches in terms of accuracy, precision, recall, and F1-score. These results are explained by the higher competence and variance reduction provided by the combination of various learning approaches. Thus, it is shown that the OESA-4 and OEVA-4 models provide efficient and robust detection of DDoS attacks. In addition, the high efficiency of OESA-4 is very effective and practical, as demonstrated by its reliability through five-fold cross-validation.

Evaluation of Results

Performance of the proposed models was analyzed in comparison with the individual algorithms. Based on the findings, it is evident that the OESA-4 and OEVA-4 models perform better than the individual algorithms in terms of all performance parameters, providing more reliable combinations of precision and recall, and hence accuracy in DDoS attack detection. In the case of cross-validation, OESA-4 achieves an accuracy value of 99.51% and OEVA-4 obtains an accuracy of 99.28%. These values are slightly lower than those obtained via the single-split approach, which are 99.87% and 99.73% for OESA-4 and OEVA-4, respectively. This is expected and desirable since it shows that the single-split accuracy values can be a bit optimistic while cross-validation provides a realistic estimate of the model's accuracy. The training and test times are reasonable when compared to accuracy values.

Figure 5 shows the effectiveness of six classifiers used for multi-class classification, based on the confusion matrix. As can be seen from the matrix, the actual classes are shown in rows, whereas the predicted values appear on columns. The entries on the diagonal of the matrix indicate the correctly predicted instances for each class. The darker the cell is colored, the higher the number of predictions; off-diagonal cells are used to demonstrate classification errors of the corresponding model for a particular traffic type. As shown in Figure 5a, Decision Tree demonstrates satisfactory performance in classifying certain traffic types, such as BENIGN (5178) and SYN (5052). However, this classifier demonstrates the highest level of class-specific error within this study. Namely, it misclassified 968 NetBIOS instances as Portmap. Figure 5b presents the accuracy achieved by the KNN model, which showed very good results when classifying the BENIGN class (5179), but experienced problems with recognizing Portmap instances (248 NetBIOS were identified as Portmap), which was not characteristic of tree-based classifiers. Random Forest, in turn, presented in Figure 5c, solves the issue of NetBIOS/Portmap confusion demonstrated by Decision Tree (correctly identifies 3137 NetBIOS). Still, the biggest problem is its struggle with MSSQL and LDAP classes because of their overlap: there were 109 cases in which MSSQL traffic was identified as LDAP. The XGBoost classifier, shown in Figure 5d, gives virtually the same results as Random Forest; it successfully distinguishes between SYN (5052) and UDP (4876) while being troubled by the overlap of the MSSQL/LDAP classes (109 instances). Figure 5e,f contain the confusion matrices for the two proposed models—the voting classifier (OEVA-4) and stacking classifier (OESA-4), respectively. OEVA-4 shows better results: the classifier managed to increase the number of accurately predicted instances of BENIGN traffic to 5251 and decreased MSSQL/LDAP misclassification to 14 instances. The best result among the examined models was provided by OESA-4: the classifier achieves the same BENIGN accuracy (5251 instances) and resolves the issue of the MSSQL/LDAP boundary (only six misclassifications, i.e., the lowest rate among the models). The most effective model is OESA-4. It demonstrates the highest overall robustness by leveraging the strengths of each base model while using a meta-classifier to resolve the final remaining ambiguities. It produces the cleanest diagonal, achieving the highest true positive counts while maintaining the lowest false positive rates across all network traffic categories.

Figure 6 shows the ROC curve for the proposed models and other state-of-the-art models in multi-class classification. In Figure 6a–d, we can see that the proposed models OESA-4 and OEVA-4, as shown in Figure 6e,f, outperform the other single models, demonstrating higher performance for all classes: BENIGN, LDAP, NetBIOS, Portmap, SYN, UDP, and MSSQL. The ROC curves of the proposed approaches remain close to the upper-left region across the evaluated classes, indicating high true positive rates and comparatively low false positive rates over a range of decision thresholds. This demonstrates strong discriminative capability relative to the individual base classifiers. Each ROC curve closely hugs the plot's upper-left corner, representing the maximum true positive rates and lowest false positive rates compared to other single models, which show slight dips in AUC for certain classes. It should be noted that XGB significantly outperforms the DT, RF, and KNN models. All the suggested models perform extremely well, which implies efficient handling of features and modeling itself.



(**a**) Confusion Matrix of DT

(**b**) Confusion Matrix of KNN

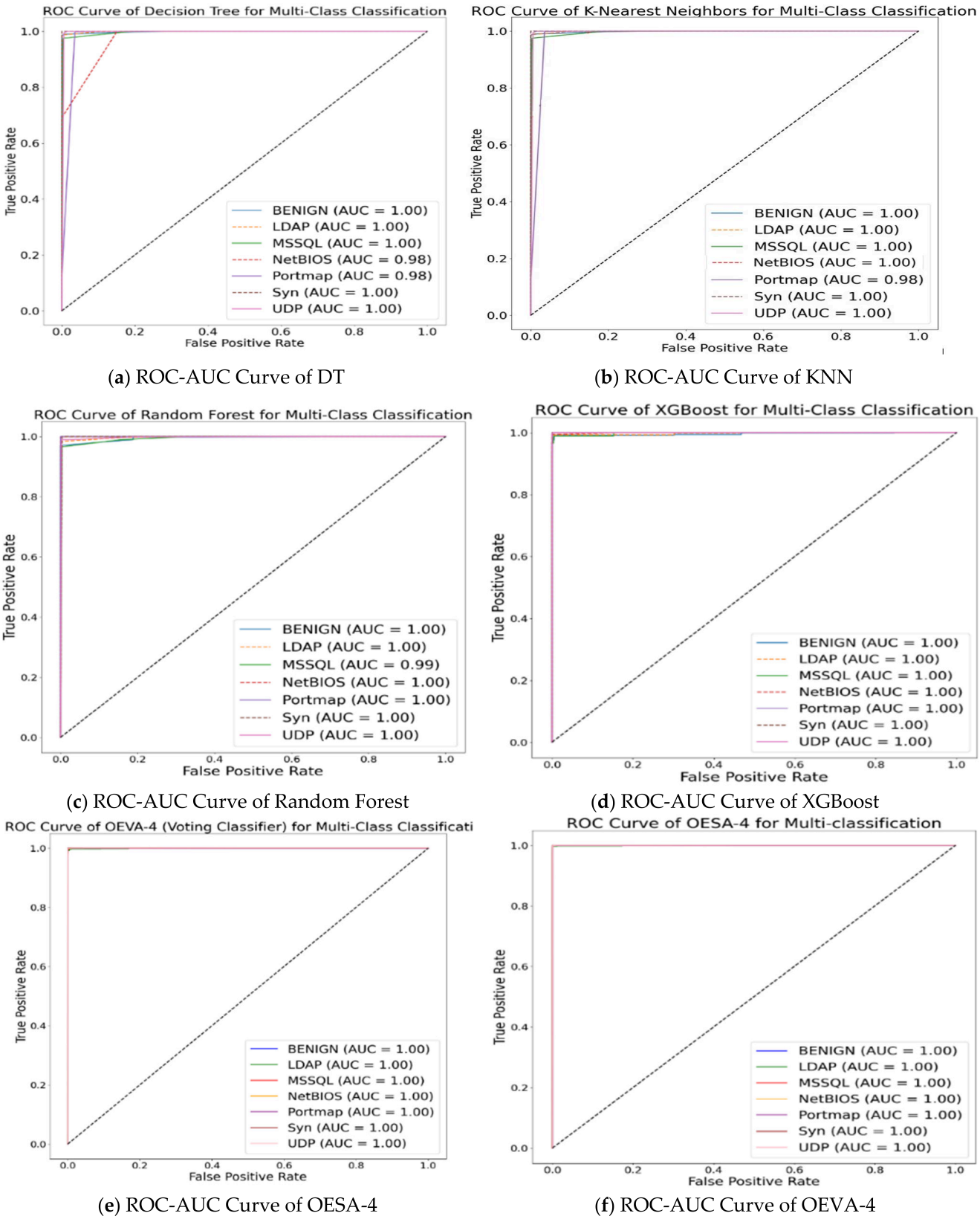
(**c**) Confusion Matrix of Random Forest

(**d**) Confusion Matrix of XGBoost

(**e**) Confusion Matrix of OESA-4

(**f**) Confusion Matrix of OEVA-4

Figure 5. Confusion matrices of the classifiers.



(**a**) ROC-AUC Curve of DT

(**b**) ROC-AUC Curve of KNN

(**c**) ROC-AUC Curve of Random Forest

(**d**) ROC-AUC Curve of XGBoost

(**e**) ROC-AUC Curve of OESA-4

(**f**) ROC-AUC Curve of OEVA-4

Figure 6. ROC-AUC curves of OESA-4, OEVA-4 and other classifiers.

Table 3 presents the results of the performance metrics for the DT classifier. As can be seen from the table, the model showed good performance for the majority of traffic types. To be more precise, the model classified the SYN category excellently (F1-score equal to 1.00). Benign, LDAP, UDP, and MSSQL traffic are also classified effectively (F1-score

varies between 0.98 and 0.99). In the case of the Portmap category, the model identified all attack instances (recall equals 1.00). However, precision for this category is low (0.84). This means that the model considers certain instances of normal traffic as attacks. As regards the NetBIOS traffic type, the model performs worse compared to other categories, as accuracy decreased to 69.07%, while the F1-score dropped to 0.70. This poor result could be explained by the fact that certain features are characteristic of other traffic types as well.

Table 3. Classification report of the Decision Tree classifier.

Normal/Attack	Accuracy (%)	Precision	Recall	F1-Score
BENIGN	98.61	1.00	0.99	0.99
LDAP	98.64	1.00	0.99	0.99
NetBIOS	69.07	0.69	0.69	0.70
Portmap	99.70	0.84	1.00	0.91
SYN	100.00	1.00	1.00	1.00
UDP	99.39	0.97	0.99	0.98
MSSQL	97.50	0.98	0.98	0.98

Table 4 highlights the output of the KNN classifier. Based on the outcomes obtained, it can be concluded that the classifier shows stable performance regardless of the traffic type. Firstly, the classifier successfully recognizes all data belonging to the category of SYN and obtains the maximum possible F1-score (1.00). Next, the performance of KNN for the BENIGN, LDAP, and UDP traffic types is also good since it gets 0.99 for all of them. Furthermore, the classifier shows significantly better performance compared to previous classifiers when dealing with NetBIOS data; the algorithm achieves a 0.95 F1-score and 99.17% overall accuracy rate. Portmap and MSSQL data are also identified successfully, obtaining F1-scores of 0.97 and 0.98, respectively. Nevertheless, the recall score is slightly lower for Portmap (0.94), showing that some traffic was missed by the classifier. KNN shows better performance compared to Decision Tree.

Table 4. Classification report of the KNN classifier.

Normal/Attack	Accuracy (%)	Precision	Recall	F1-Score
BENIGN	98.63	1.00	0.99	0.99
LDAP	98.30	0.99	0.98	0.99
NetBIOS	99.17	0.92	0.99	0.95
Portmap	94.45	0.99	0.94	0.97
SYN	99.78	1.00	1.00	1.00
UDP	98.82	0.99	0.99	0.99
MSSQL	97.84	0.98	0.98	0.98

Table 5 shows the results for the RF classifier. In contrast with the other individual models (DT, and KNN) the RF model shows more consistency when it comes to all different types of attacks. The F1-scores for NetBIOS and Portmap equal 1.00, which means that the model splendidly detects all the samples for those two attack types. Additionally, there is a recall rate of 1.00 achieved when dealing with SYN data. BENIGN, LDAP, and UDP attacks have F1-scores of 0.98. Although the MSSQL class has a slightly lower F1-score of 0.97, the model still maintains high performance overall. Because Random Forest combines several decision trees, it produces well-balanced precision and recall, which help it handle complex network traffic more effectively.

Table 5. Classification report of the Random Forest classifier.

Normal/Attack	Accuracy (%)	Precision	Recall	F1-Score
BENIGN	97.01	1.00	0.97	0.98
LDAP	98.37	0.97	0.98	0.98
NetBIOS	99.94	1.00	1.00	1.00
Portmap	99.54	1.00	1.00	1.00
SYN	100.00	0.98	1.00	0.99
UDP	98.96	0.98	0.99	0.98
MSSQL	96.59	0.98	0.97	0.97

Table 6 shows the results of the XGBoost classifier. The model performs well across traffic types. The F1-score for both NetBIOS and Portmap is 1.00, suggesting that the model correctly classifies these types. Likewise, the performance of SYN and BENIGN traffic is remarkable, with F1-scores of 0.99 and 0.98, respectively. In addition, the LDAP and UDP traffic types both have F1-scores of 0.98. Even though the MSSQL type represents the worst-performing group, it still achieves a relatively high F1-score of 0.97. XGBoost provides well-balanced results, reducing both false alarms and missed attacks across the dataset compared to DT, RF, and KNN.

Table 6. Classification report of the XGBoost classifier.

Normal/Attack	Accuracy (%)	Precision	Recall	F1-Score
BENIGN	97.01	1.00	0.97	0.98
LDAP	98.37	0.97	0.98	0.98
NetBIOS	99.39	1.00	0.99	1.00
Portmap	100.00	0.99	1.00	1.00
SYN	100.00	0.99	1.00	0.99
UDP	99.01	0.97	0.99	0.98
MSSQL	96.61	0.98	0.97	0.97

Table 7 depicts the performance of OESA-4. The output shows that this model performs very well for all kinds of traffic. Particularly, this model demonstrates an F1-Score of 1.00 for the categories of BENIGN, LDAP, NetBIOS, Portmap, SYN, UDP, and MSSQL. It indicates an exceptional harmony between precision and recall. The model correctly detects attack occurrences and avoids any sort of false alarm. Likewise, the accuracy is very high. For instance, the model predicts attacks with higher accuracy for almost all types of traffic. Still, the accuracy score is slightly low for the LDAP category at 99.56 percent. In terms of performance consistency, the OESA-4 model works better than other existing models, especially on NetBIOS and Portmap. Results advocate that the OESA-4 affords an effective and reliable solution.

Table 7. Classification report of the OESA-4 classifier.

Normal/Attack	Accuracy (%)	Precision	Recall	F1-Score
BENIGN	100.00	1.00	1.00	1.00
LDAP	99.56	1.00	1.00	1.00
NetBIOS	99.90	1.00	1.00	1.00
Portmap	100.00	1.00	1.00	1.00
SYN	100.00	1.00	1.00	1.00
UDP	99.82	1.00	1.00	1.00
MSSQL	99.75	1.00	1.00	1.00

Table 8 presents the performance of the OEVA-4 classifier. As the results show, the classifier's performance remains quite good, with an F1-score of 1.00 for the categories BENIGN, NetBIOS, Portmap, SYN, and UDP. However, the F1-scores for the LDAP and MSSQL categories are slightly lower, at 0.99. The accuracy values reach 100% for several traffic categories, showing that the model classifies most instances correctly. Precision is also very high for nearly all classes, which means that the attacks identified by the model are rarely false alarms. Overall, the proposed OEVA-4 shows stable and reliable performance and performs almost as well as the proposed OESA-4, particularly in handling traffic types that were more difficult for earlier baseline models.

Table 8. Classification report of the OEVA-4 classifier.

Normal/Attack	Accuracy (%)	Precision	Recall	F1-Score
BENIGN	100.00	1.00	1.00	1.00
LDAP	99.00	1.00	0.99	0.99
NetBIOS	99.46	1.00	0.99	1.00
Portmap	100.00	1.00	1.00	1.00
SYN	100.00	1.00	1.00	1.00
UDP	99.80	1.00	1.00	1.00
MSSQL	99.58	0.99	1.00	0.99

Table 9 displays a comparison among the six models of accuracy, precision, recall, F1-score, training time, and testing time. With the use of only 20 features, the proposed OESA-4 and OEVA-4 classifiers show significantly better results compared to other models in terms of the main metrics, achieving 99.87% and 99.73%. This makes them highly balanced and accurate classifiers. However, the training and testing times of OESA-4 and OEVA-4 classifiers are relatively large (708.1186 s and 23.4857 s, respectively). Nevertheless, testing time can be considered appropriate (70.6844 s and 70.7797 s). On the contrary, the XGB and RF models are close to the proposed models, with 98.60% and 98.57% accuracy, respectively; KNN is shown to be quite a good classifier with 98.09% accuracy. DT demonstrates a lower performance compared to XGB, Random Forest, and KNN, with 96.07% accuracy. DT and RF have computational costs, but DT underperforms in terms of classification.

Table 9. Performance of the classifiers with twenty (20) features.

Models	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Training Time	Testing Time
DT	96.07	96.56	96.07	95.93	0.7044	0.0108
KNN	98.09	98.15	98.9	98.10	0.0759	71.5266
RF	98.57	98.58	98.57	98.57	0.7158	0.0140
XGB	98.60	98.60	98.59	98.59	36.3548	0.7165
OEVA-4	99.73	99.73	99.73	99.73	23.4857	70.7797
OESA-4	99.87	99.87	99.87	99.87	708.1186	70.6844

Table 10 presents the results of the five-fold stratified cross-validation of the four base classifiers and the two proposed ensembles. For each model, the margin of error is presented based on the standard deviation of accuracy obtained across five folds, providing a measure of result stability rather than a single-split estimate. The results show that OESA-4 (stacking) achieved the highest aggregate accuracy of 99.51%, with a margin of error of $\pm$0.08%. OEVA-4 (voting) achieved 99.28%, with a margin of error of $\pm$0.11%. This indicates that both ensemble approaches outperform all other methods consistently across the five folds of the five-fold stratified cross-validation test. Amongst the base classifiers, the XGBoost

classifier performs best, with an aggregate accuracy of 98.34% ± 0.16%, while the Random Forest model comes second with an aggregate accuracy of 98.21% ± 0.19%. The third-best-performing model is the KNN classifier with an aggregate accuracy of 97.65% ± 0.24%, and last is the Decision Tree model with an aggregate accuracy of 95.42% ± 0.40%. Overall, the results demonstrate that OESA-4 provides the most powerful and consistent cross-validation performance, while OEVA-4 also improves substantially over the individual base classifiers.

Table 10. Five-Fold cross-validation results.

Category	Model	Fold1	Fold2	Fold3	Fold4	Fold5	Aggregate (%)	Margin of Error (±%)
Base Classifier	Decision Tree	94.95	95.80	95.25	95.90	95.20	95.42	0.40
Base Classifier	K-Nearest Neighbors	97.35	97.95	97.45	97.80	97.70	97.65	0.24
Base Classifier	Random Forest	97.95	98.30	98.45	98.10	98.25	98.21	0.19
Base Classifier	XGBoost	98.15	98.45	98.20	98.55	98.35	98.34	0.16
Ensemble Model	OEVA-4 (Voting)	99.15	99.39	99.20	99.40	99.26	99.28	0.11
Ensemble Model	OESA-4 (Stacking)	99.41	99.60	99.52	99.45	99.57	99.51	0.08

Figure 7 provides a comparative study of the performances of the six classification models. The OESA-4 and OEVA-4 approaches provide the highest performances among the six classification models with higher scores of 99.87% and 99.73%, respectively, while outperforming the other existing approaches, thus setting a benchmark for future performance standards. XGBoost and Random Forest have the second-highest performances, with scores of 98.59% and 98.57%, whereas K-Nearest Neighbors (KNN) and Decision Tree have scores of 98.09% and 96.07%, respectively. Out of all six models used, the Decision Tree model gives the lowest performance. The OESA-4 and OEVA-4 methods utilize the strength of different models for better performance, enhancing accuracy and lowering error rates.

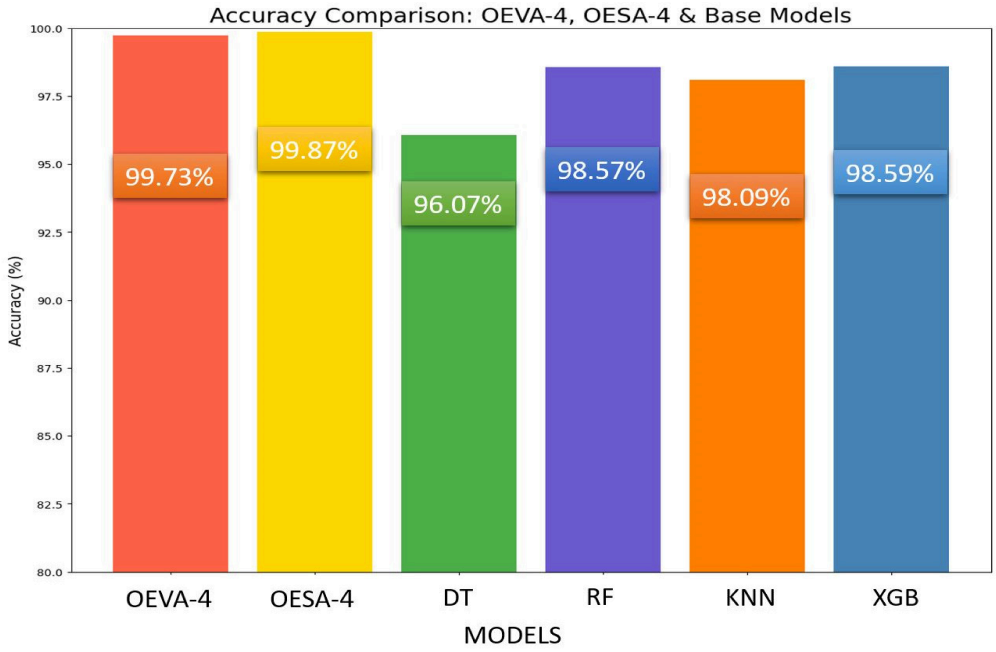


Figure 7. Accuracy comparison of classifiers.

5. Conclusions and Future Work

This research work proposed OESA-4 (Optimized Ensemble Stacking Approach-4) and OEVA-4 (Optimized Ensemble Voting Approach-4) for improving the effectiveness of the detection of DDoS attacks. The proposed frameworks purposely integrate K-Nearest Neighbors (KNN), Random Forest (RF), Extreme Gradient Boosting (XGB), and Decision Tree (DT) classifiers, which are employed as base learners to strengthen the performance of the proposed models. OESA-4 employed a Logistic Regression algorithm as the meta-learner that integrates its outputs to enhance decision-making, while OEVA-4 leveraged classifier diversity by assigning optimal weights to stronger learners, enhancing DDoS attack detection accuracy and reducing misclassification. The proposed models enhance detection accuracy and robustness against diverse attack patterns using the complementary strengths of base learners. The operationalization of the framework is realized by implementing it in Python using Scikit-learn libraries, which incorporate a preprocessing step of systematic scaling and normalization to increase the models' reliability. The results obtained from experimentation on a selected subset of CIC-DDoS2019 were 99.87% accuracy for OESA-4 and 99.73% accuracy for OEVA-4, based on one train–test split. In order to verify these results, five-fold cross-validation was used, yielding an average accuracy of 99.51% $\pm$ 0.08% for OESA-4 and 99.28% $\pm$ 0.11% for OEVA-4. These results indicate a low margin of error, which signifies reliable performance for both models in DDoS attack detection. Potential future research may focus on the real-time implementation of the proposed frameworks in live networks, the implementation of adaptive incremental learning to cope with changing cyber threats, and developing the ability to analyze encrypted traffic in response to increased encryption use by attackers. Additionally, future efforts may include implementing lightweight optimization to adapt to IoT and edge computing devices. Moreover, extending the platform's functionality to cover other types of attacks apart from DDoS and applying deep learning to analyze the dynamics of the traffic can be considered further directions of research.

Author Contributions: A.G.A. contributed to the conception of the study, implementation and analysis of the experiments and writing the manuscript. F.A.J. provided proper guidance in the overall process of writing the manuscript. A.A.A. provided assistance in the review and editing process. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. The CIC-DDoS2019 dataset is publicly available in the Kaggle repository: R.R. Silva, "CIC-DDoS2019 30GB Full Dataset—CSV Files," 2022. The dataset can be accessed at: https://www.kaggle.com/datasets/rodrigorosasilva/cic-ddos2019-30gb-full-dataset-csv-files (accessed on 18 September 2026).

Acknowledgments: The authors sincerely acknowledge the valuable contributions of Mohammad Siraj, email: siraj@ksu.edu.sa; Saif A. Alsaif, email: saalsaif@ksu.edu.sa; and Kashan Memon, email: kashan02@mail.ustc.edu.cn, to this research work. Their support and contributions to the writing and refinement of the manuscript are gratefully acknowledged.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhukabayeva, T.; Zholshiyeva, L.; Karabayev, N.; Khan, S.; Alnazzawi, N. Cybersecurity Solutions for Industrial Internet of Things–Edge Computing Integration: Challenges, Threats, and Future Directions. *Sensors* **2025**, *25*, 213. [CrossRef] [PubMed]
2. Mallick, M.A.I.; Nath, R. Navigating the cyber security landscape: A comprehensive review of cyber-attacks, emerging trends, and recent developments. *World Sci. News* **2024**, *190*, 1–69.
3. Djenna, A.; Harous, S.; Saidouni, D.E. Internet of things meet internet of threats: New concern cyber security issues of critical cyber infrastructure. *Appl. Sci.* **2021**, *11*, 4580. [CrossRef]
4. Abualhassan, A.; Rashid, I.; Binbeshr, F.; Imam, M. DDoS Attack Detection in IoT: A Comparative Resource and Performance Analysis of Deep Learning and Machine Learning Models. *IEEE Access* **2025**, *13*, 116529–116547. [CrossRef]
5. Ansari, A.G.; Jokhio, F.A.; Syed, M.S.S.; Dharejo, H.; Memon, F.A. Machine Learning-Driven Approach for DDOS Attacks Detection using Neural-Based Networks: A Proficiency Study. *Mehran Univ. Res. J. Eng. Technol.* **2025**, *44*, 206–218. [CrossRef]
6. Poonia, L.; Tinker, S. A Comprehensive Analysis of the Types, Impacts, Prevention, and Mitigation of DDoS Attacks. *Recent Pat. Eng.* **2024**, *19*, 1–18. [CrossRef]
7. Agrawal, N.; Tapaswi, S. Defense Mechanisms against DDoS Attacks in a Cloud Computing Environment: State-of-the-Art and Research Challenges. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3769–3795. [CrossRef]
8. Riggs, H.; Tufail, S.; Parvez, I.; Tariq, M.; Khan, M.A.; Amir, A.; Vuda, K.V.; Sarwat, A.I. Impact, Vulnerabilities, and Mitigation Strategies for Cyber-Secure Critical Infrastructure. *Sensors* **2023**, *23*, 4060. [CrossRef] [PubMed]
9. Fei, W.; Ohno, H.; Sampalli, S. A Systematic Review of IoT Security: Research Potential, Challenges, and Future Directions. *ACM Comput. Surv.* **2024**, *56*, 1–40. [CrossRef]
10. Orosz, P.; Nagy, B.; Varga, P. Real-Time Detection and Mitigation Strategies Newly Appearing for DDoS Profiles. *Future Internet* **2025**, *17*, 400. [CrossRef]
11. Ahmed, A.H.; Fida, M.-R.; Arsalaan, A.S. DDoShield: In-Network Defensive Architecture Against Volumetric and Non-Volumetric DDoS Attacks. *Techrxiv* **2024**. [CrossRef] [PubMed]
12. Staddon, E.; Loscri, V.; Mitton, N. Attack categorisation for iot applications in critical infrastructures, a survey. *Appl. Sci.* **2021**, *11*, 7228. [CrossRef]
13. Mbow, M.; Koide, H.; Sakurai, K. Handling class imbalance problem in intrusion detection system based on deep learning. *Int. J. Netw. Comput.* **2022**, *12*, 467–492. [CrossRef] [PubMed]
14. Najafimehr, M.; Zarifzadeh, S.; Mostafavi, S. A hybrid machine learning approach for detecting unprecedented DDoS attacks. *J. Supercomput.* **2022**, *78*, 8106–8136. [CrossRef] [PubMed]
15. Owusu, E.; Rahouti, M.; Jagatheesaperumal, S.K.; Xiong, K.; Xin, Y.; Lu, L.; Hsu, D.F. Online Network DoS/DDoS Detection: Sampling, Change Point Detection, and Machine Learning Methods. *IEEE Commun. Surv. Tutor.* **2024**, *27*, 2543–2580. [CrossRef]
16. Hossain, A. Enhanced Ensemble-Based Distributed Denial-of-Service (DDoS) Attack Detection with Novel Feature Selection: A Robust Cybersecurity Approach. *Artif. Intell. Evol.* **2023**, *4*, 165–186. [CrossRef]
17. Alotaibi, Y.; Ilyas, M. Ensemble-Learning Framework for Intrusion Detection to Enhance Internet of Things' Devices Security. *Sensors* **2023**, *23*, 5568. [CrossRef] [PubMed]
18. Lin, Y.; Wei, Y.; Chen, D.; Li, Y.; Erkan, U.; Toktas, A.; Gao, S.; Zhang, Y. Cryptanalysis and Improvement of a Video Cryptosystem via Chaos and S-Box. *ACM Trans. Multimed. Comput. Commun. Appl.* **2026**, *22*, 1–28. [CrossRef]
19. Bala, S.; Kaur, H. Enhancing DDoS Attack Detection: A Comparative Analysis of Ensemble-Based Approach with SVM and Random Forest. *J. Emerg. Technol. Innov. Res.* **2023**, *10*, 363–370.
20. Okey, O.D.; Maidin, S.S.; Adasme, P.; Rosa, R.L.; Saadi, M.; Melgarejo, D.C.; Rodríguez, D.Z. BoostedEnML: Efficient Technique for Detecting Cyberattacks in IoT Systems Using Boosted Ensemble Machine Learning. *Sensors* **2022**, *22*, 7409. [CrossRef] [PubMed]
21. Venkatraman, S.; Kanthimathi, S.; Jayasankar, K.S.; Jiljith, T.P.; Jashwanth, R. A Novel Self-Attention-Enabled Weighted Ensemble-Based Convolutional Neural Network Framework for Distributed Denial of Service Attack Classification. *IEEE Access* **2024**, *12*, 151515–151531. [CrossRef]
22. Alqarni, A.A. Majority Vote-Based Ensemble Approach for Distributed Denial of Service Attack Detection in Cloud Computing. *J. Cyber Secur. Mobil.* **2022**, *11*, 265–278. [CrossRef]
23. Ali, T.E.; Chong, Y.-W.; Manickam, S.; Yusoff, M.N.; Yau, K.-L.A.; Zoltan, A.D. A Stacking Ensemble Model with Enhanced Feature Selection for Distributed Denial-of-Service Detection in Software-Defined Networks. *Eng. Technol. Appl. Sci. Res.* **2025**, *15*, 19232–19245. [CrossRef]
24. Awad, Z.; Zakaria, M.; Hassan, R. An enhanced ensemble defense framework for boosting adversarial robustness of intrusion detection systems. *Sci. Rep.* **2025**, *15*, 14177. [CrossRef] [PubMed]
25. Gaur, V.; Kumar, R. Analysis of Machine Learning Classifiers for Early Detection of DDoS Attacks on IoT Devices. *Arab. J. Sci. Eng.* **2022**, *47*, 1353–1374. [CrossRef]

26. Roopak, M.; Tian, G.Y.; Chambers, J. Deep learning models for cyber security in IoT networks. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019, Las Vegas, NV, USA, 7–9 January 2019; pp. 452–457. [CrossRef]
27. Qasim, S.S.; Nsaif, S.M. Advancements in Time Series-Based Detection Systems for Distributed Denial-of-Service (DDoS) Attacks: A Comprehensive Review. *Babylon. J. Netw.* **2024**, *2024*, 9–17. [CrossRef]
28. Ibrahim, W.N.H.; Anuar, S.; Selamat, A.; Krejcar, O.; Crespo, R.G.; Herrera-Viedma, E.; Fujita, H. Multilayer Framework for Botnet Detection Using Machine Learning Algorithms. *IEEE Access* **2021**, *9*, 48753–48768. [CrossRef]
29. Yaser, A.L.; Mousa, H.M.; Hussein, M. Improved DDoS Detection Utilizing Deep Neural Networks and Feedforward Neural Networks as Autoencoder. *Futur. Internet* **2022**, *14*, 240. [CrossRef]
30. Abdulrahman, A.A.; Ibrahem, M.K. Evaluation of DDoS attacks detection in a new intrusion dataset based on classification algorithms. *Iraqi J. Inf. Commun. Technol.* **2018**, *1*, 49–55. [CrossRef]
31. Mehmood, S.; Amin, R.; Mustafa, J.; Hussain, M.; Alsubaei, F.S.; Zakaria, M.D. Distributed Denial of Services (DDoS) attack detection in SDN using Optimizer-equipped CNN-MLP. *PLoS ONE* **2025**, *20*, e0312425. [CrossRef] [PubMed]
32. Gupta, B.; Chaudhary, P.; Chang, X.; Nedjah, N. Smart defense against distributed Denial of service attack in IoT networks using supervised learning classifiers. *Comput. Electr. Eng.* **2022**, *98*, 107726. [CrossRef]
33. Abiramasundari, S.; Ramaswamy, V. Distributed denial-of-service (DDOS) attack detection using supervised machine learning algorithms. *Sci. Rep.* **2025**, *15*, 13098. [CrossRef] [PubMed]
34. Bindra, N.; Sood, M. Detecting DDoS Attacks Using Machine Learning Techniques and Contemporary Intrusion Detection Dataset. *Autom. Control. Comput. Sci.* **2019**, *53*, 419–428. [CrossRef]
35. Elejla, O.E.; Belaton, B.; Anbar, M.; Alabsi, B.; Al-Ani, A.K. Comparison of classification algorithms on ICMPv6-based DDoS attacks detection. In *Computational Science and Technology*; Lecture Notes in Electrical Engineering; Springer: Berlin/Heidelberg, Germany, 2019; pp. 347–357. [CrossRef]
36. Mohammed, A.J.; Kadir, D. Improving Classification Performance for a Novel Imbalanced Medical Dataset using SMOTE Method. *Int. J. Adv. Trends Comput. Sci. Eng.* **2020**, *9*, 3161–3172. [CrossRef]
37. Al-Khayyat, A.T.K.; Ucan, O.N. A Multi-Branched Hybrid Perceptron Network for DDoS Attack Detection Using Dynamic Feature Adaptation and Multi-Instance Learning. *IEEE Access* **2024**, *12*, 192618–192638. [CrossRef]
38. Aktar, S.; Nur, A.Y. Towards DDoS attack detection using deep learning approach. *Comput. Secur.* **2023**, *129*, 103251. [CrossRef]
39. Elubeyd, H.; Yiltas-Kaplan, D. Hybrid Deep Learning Approach for Automatic DoS/DDoS Attacks Detection in Software-Defined Networks. *Appl. Sci.* **2023**, *13*, 3828. [CrossRef]
40. Bouke, M.A.; Abdullah, A.; Alshatebi, S.H.; Abdullah, M.T.; El Atigh, H. An intelligent DDoS attack detection tree-based model using Gini index feature selection method. *Microprocess. Microsyst.* **2023**, *98*, 104823. [CrossRef]
41. Salman, H.A.; Kalakech, A.; Steiti, A. Random Forest Algorithm Overview. *Babylon. J. Mach. Learn.* **2024**, *2024*, 69–79. [CrossRef] [PubMed]
42. Butt, H.A.; Al Harthy, K.S.; Shah, M.A.; Hussain, M.; Amin, R.; Rehman, M.U. Enhanced DDoS Detection Using Advanced Machine Learning and Ensemble Techniques in Software Defined Networking. *Comput. Mater. Contin.* **2024**, *81*, 3003–3031. [CrossRef]
43. Alsumaidaie, M.; Alheeti, K.; Alaloosy, A. An Assessment of Ensemble Voting Approaches, Random Forest, and Decision Tree Techniques in Detecting Distributed Denial of Service (DDoS) Attacks. *Iraqi J. Electr. Electron. Eng.* **2023**, *20*, 16–24. [CrossRef]
44. Rejwan Bin Sulaiman, A.; Khraisat, A.; Bin Sulaiman, R. Metaheuristic-Driven Feature Selection with SVM and KNN for Robust DDoS Attack Detection: A Comparative Study. *J. Cyber Secur. Risk Audit.* **2025**, *2025*, 182–203. [CrossRef]
45. Halder, R.K.; Uddin, M.N.; Uddin, A.; Aryal, S.; Khraisat, A. Enhancing K-nearest neighbor algorithm: A comprehensive review and performance analysis of modifications. *J. Big Data* **2024**, *11*, 113. [CrossRef]
46. Hambali, M.A.; Ayati, B.A.; Olasupo, Y.A.; Dalhatu, M.; Akinyemi, A.A. An Intelligent Feature-Optimized Framework for Ddosattack Detection Using SVM and XGBoost. *FULafia J. Sci. Technol.* **2026**, *10*, 132–147. [CrossRef]
47. Ahmed, M.A.O.; AbdelSatar, Y.; Alotaibi, R.; Reyad, O. Enhancing Internet of Things security using performance gradient boosting for network intrusion detection systems. *Alex. Eng. J.* **2025**, *116*, 472–482. [CrossRef]
48. Hu, Y.; Xiao, K.; Luo, L.; Chen, L. An XGBoost-Based Intrusion Detection Framework with Interpretability Analysis for IoT Networks. *Appl. Sci.* **2026**, *16*, 980. [CrossRef]
49. Suvra, D.K. An Efficient Real Time DDoS Detection Model Using Machine Learning Algorithms. *arXiv* **2025**, arXiv:2501.14311. [CrossRef]
50. Rahman, A.; Khan, S.I.; Eidmum, Z.A.; Shaha, P.; Muiz, B.; Hasan, N.; Debnath, T.; Kundu, D.; Tamanna, J.T.; Sayduzzaman, M.; et al. Stacked Ensemble Method: An Advanced Machine Learning Approach for Anomaly-based Intrusion Detection System. *Stat. Optim. Inf. Comput.* **2025**, *14*, 434–453. [CrossRef]
51. Mall, R.; Abhishek, K.; Shankar, A.; Kumar, A. Stacking ensemble approach for DDoS attack detection in software-defined cyber–physical systems. *Comput. Electr. Eng.* **2023**, *107*, 108635. [CrossRef]

52. Abraham, A.M.; Kumar, M.S. An Imbalance-Resilient Network Intrusion Detection Framework Using TVAE-Based Data Augmentation and Stacked Ensemble. 2026. Available online: https://www.researchsquare.com/article/rs-10412382/v1 (accessed on 18 September 2026).
53. Chen, C.-L.; Lee, W.-J. Constructing an Ensemble Stacking Model for Detecting DDoS Attacks. *Telecom* **2026**, *7*, 51. [CrossRef]
54. Ganesh, L.S.; Kumar, P.M. HydraGuard-ID: A Hybrid Ensemble Framework with Stratified Feature Selection for High-Accuracy DDoS Attack Detection. *IEEE Access* **2026**, *14*, 58985–59000. [CrossRef]
55. Jabbar, H.G. Advanced Threat Detection Using Soft and Hard Voting Techniques in Ensemble Learning. *J. Robot. Control (JRC)* **2024**, *5*, 1104–1116.
56. Swami, R.; Dave, M.; Ranga, V. Voting-based intrusion detection framework for securing software-defined networks. *Concurr. Comput. Pr. Exp.* **2020**, *32*, e5927. [CrossRef]
57. Kamaruddin, A.; Chin, T.S. An Enhanced Learning Voting-Based Framework for Time-Efficient DDoS Detection with Dataset Consistency in SDN-IoT Enabled Smart Homes. In *Mobile Web and Intelligent Information Systems. MobiWIS 2025*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2025; Volume 16066, pp. 144–158. [CrossRef]
58. Alserhani, F.; Aljared, A. Evaluating Ensemble Learning Mechanisms for Predicting Advanced Cyber Attacks. *Appl. Sci.* **2023**, *13*, 13310. [CrossRef]
59. Farouk, M.; Sakr, R.H.; Hikal, N. Identifying the most accurate machine learning classification technique to detect network threats. *Neural Comput. Appl.* **2024**, *36*, 8977–8994. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.