Decision Analytics Journal 8 (2023) 100307

Contents lists available at ScienceDirect

Decision Analytics Journal

journal homepage: www.elsevier.com/locate/dajour

A novel ensemble method for enhancing Internet of Things device security against botnet attacks

Amina Arshad [a], Maira Jabeen [a], Saqib Ubaid [a], Ali Raza [a,*], Laith Abualigah [b,c,d,e,f,g,h,**], Khaled Aldiabat [i], Heming Jia [j]

[a] Institute of Computer Science, Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan 64200, Pakistan
[b] Computer Science Department, Prince Hussein Bin Abdullah Faculty for Information Technology, Al al-Bayt University, Mafraq 25113, Jordan
[c] Department of Electrical and Computer Engineering, Lebanese American University, Byblos 13-5053, Lebanon
[d] Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan
[e] MEU Research Unit, Middle East University, Amman 11831, Jordan
[f] Applied science research center, Applied science private university, Amman 11931, Jordan
[g] School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang 11800, Malaysia
[h] School of Engineering and Technology, Sunway University Malaysia, Petaling Jaya 27500, Malaysia
[i] Department of Management Information Systems, Ajloun National University, Jordan
[j] School of Information Engineering, Sanming University, Sanming 365004, China

ARTICLE INFO

Keywords:
Botnet attacks
Deep learning
Ensemble learning
IoT devices
Network security
Cyber security

ABSTRACT

The growing number of connected Internet of Things (IoT) devices has led to the daily growth of network botnet attacks. The networks of compromised devices controlled by a single entity can be used for malicious purposes such as denial of service distributed IoT botnet attacks and theft of personal information. The weak security measures of many IoT devices make them easy targets for compromise and inclusion in botnets. In this research, we propose a system for detecting botnet attacks. We develop an ensemble learning system to detect botnets in network traffic with high-performance scores. The system will analyze the traffic and identify any suspicious behavior that may indicate the presence of a botnet. For this purpose, we use the benchmark CTU-13 dataset to build the applied machine learning and deep learning techniques for comparison. We propose a novel ensemble technique, K-neighbors, Decision tree, and Random forest (KDR), to achieve high performance for botnet attack detection. Study results show that the proposed KDR gives 99.7% accuracy in 12.99 s. Hyperparameter optimization and k-fold cross-validation are employed to substantiate the performance. Our research study contributes to the body of knowledge on the detection of botnet attacks and provides a practical solution for securing IoT devices against botnet attacks.

1. Introduction

Botnets are a network of compromised devices, which can be computers, servers, or Internet of Things (IoT) devices infected with malware and controlled by a remote attacker [1,2]. Botnet attacks on IoT devices are becoming increasingly common as IoT devices become more prevalent daily [3]. These attacks usually target vulnerable devices with weak security measures, such as default passwords or unpatched software, to seize command of the device and add it to the botnet. Once the attacker gains control, they can use the device to launch denial-of-service (DoS) distributed attacks, steal private information, or perform other malicious activities [4]. Botnet attacks in IoT devices pose a significant threat to individuals and organizations as they can cause damage to critical infrastructure, compromise sensitive data, and result in financial loss. Therefore, it is crucial to implement strong security measures and keep IoT devices up to date to prevent botnet attacks.

Networking and security are essential to modern life, enabling rapid global communication and easy access to resources through cloud computing [5]. Social media, email, messaging apps, and video conferencing tools connect people worldwide, while firewalls, antivirus software, and two-factor authentication protect against hacking, phishing, and identity theft. These measures are especially critical in healthcare, where patient confidentiality and medical data security are top priorities. Likewise, the financial industry relies on networking and security to safeguard transactions and customer data. With technology

* Corresponding author.
** Computer Science Department, Prince Hussein Bin Abdullah Faculty for Information Technology, Al al-Bayt University, Mafraq 25113, Jordan.
E-mail addresses: aminaarshad340@gmail.com (A. Arshad), mairajabeen.1mr@gmail.com (M. Jabeen), saqib.ubaid@yahoo.com (S. Ubaid), ali.raza.scholarly@gmail.com (A. Raza), aligah.2020@gmail.com (L. Abualigah), khaledmis@anu.edu.jo (K. Aldiabat), jiaheming@fjsmu.edu.cn (H. Jia).

https://doi.org/10.1016/j.dajour.2023.100307
Received 14 June 2023; Received in revised form 21 July 2023; Accepted 18 August 2023
Available online 23 August 2023

advancing rapidly, staying up-to-date with the latest practices and developments in networking and security is crucial.

Machine learning, deep learning, and ensemble learning are increasingly used to detect botnet attacks due to their ability to analyze large amounts of data and identify patterns that may indicate botnet activity [6]. Machine learning methods can be trained to recognize the characteristics of normal network traffic and differentiate it from the abnormal traffic generated by botnets. Neural networks are used in deep learning, a form of machine learning to learn representations of data automatically and have shown promising results in detecting botnets. Ensemble learning [7] combines multiple machine learning models to improve accuracy and can be used to enhance botnet detection by combining the strengths of different algorithms. These techniques can identify botnet command-and-control servers, detect communication patterns among botnet nodes, and identify botnet malware on infected machines. Overall, machine learning [8], deep learning, and ensemble learning have shown great potential for improving the detection and prevention of botnet attacks.

Our everyday lives have benefited greatly from the growth of Internet of Things (IoT) gadgets, but it has also created new security issues. IoT security is particularly vulnerable to botnet assaults, which may hack many devices and use them to execute DDoS attacks or steal confidential information. Researchers and business people have created several methods to identify and stop botnet assaults in order to meet this difficulty. Ensemble learning is a promising strategy integrating many machine learning models to increase the detection system's durability and accuracy.

The following research questions are raised in this study:

1. How does the performance of botnet attack detection improve compared to state-of-the-art methods by combining multiple models as an ensemble?
2. What are the most efficient machine learning and deep learning approaches built and evaluated for botnet attack detection?

This article provides an improved ensemble learning strategy to improve IoT device security by efficiently identifying botnet assaults. The three key components of our method are feature extraction, model training, and ensemble learning. We employ a set of representative features that accurately describe the network traffic behavior of IoT devices during the feature extraction step. These characteristics include payload data entropy, packet size, and packet interarrival time. Throughout the model training stage, we train several machine learning models, such as Random Forest, Support Vector Machines, and Gradient Boosting Decision Trees, on a sizeable dataset of labeled network traffic data.

We use an optimization technique to choose the top-performing models and integrate them into a single ensemble model during the ensemble learning stage. In comparison to previous ensemble models and individual models, our optimized ensemble model outperforms them all by identifying botnet assaults on IoT devices with a high accuracy of 99.7%. Additionally, we demonstrate that our method has a low false positive rate and can identify a variety of botnet assaults, such as Mirai, IoT Reaper, and Gafgyt. We assess our method's resilience to adversarial attacks and demonstrate that it can reliably identify and categorize assaults even when network traffic data has been altered.

The scientific contributions of our research study for botnet attack detection are followed as:

- A novel ensemble learning-based KDR technique is proposed for efficient botnet attack detection. The advanced machine learning-based k-neighbors, decision tree, and random forest methods are combined to create the proposed ensemble method. Study results indicate that the proposed approach achieved high-performance scores for botnet attack detection compared to the state-of-the-art studies.
- Ten advanced machine learning and one deep learning-based method are applied in comparisons to evaluate the performance. All applied techniques are fully hyperparameter optimized, and performance validation is conducted through k-fold cross-validation. The computation complexity analysis is also applied.

Overall, our suggested method shows how ensemble learning can be used to improve IoT device security and identify botnet assaults. Our method may be included in current IoT security frameworks to offer an extra layer of defense against botnet assaults and guarantee IoT devices' secure and reliable operation.

The remaining of our study is arranged as the network attack-related research gaps are identified in Section 2. The attack detection methodology is described in Section 3. The results and discussion related to the applied technique are comparatively evaluated in Section 4. Our proposed study winds up in Section 5.

2. Literature review

The network attack detection-related studies are comparatively evaluated in this section. The related analysis is based on the previously used techniques, dataset, and performance metrics scores to identify the research gaps, as discussed in Table 1.

This paper [9] proposed a machine learning-based multilayer perceptron approach for botnet detection that addresses the challenges of detecting unseen botnets that can evade traditional signature-based analysis. The framework consists of filtering and classification modules that use machine learning algorithms to detect the botnet's attack command and take control of the server. To detect botnets, the scientists used behavior-based analysis using flow-based characteristics that analyzed the packet header, even in encapsulated techniques such as a VPN tunnel. The proposed framework achieved a high accuracy score of 92% for f1 and a low false-negative rate of 1.5%. The study results demonstrate the potential of using machine learning methods for botnet detection and highlight the importance of behavior analysis-based modern botnet attack detection.

Several research papers have been published in recent years to tackle the problem of detecting HTTP-based botnets using machine learning algorithms. One such paper [10] proposed solving the problem using deep learning techniques to classify HTTP traffic data packets. The study aimed to improve detection accuracy and reduce false positives. The proposed method achieved an accuracy rate of 96.3% on the test dataset. Another paper used machine learning methods, including a random forest classifier and support vector machine, to detect botnet traffic. The researchers used a dataset of network traffic features based on HTTP and HTTPS protocols. The proposed approach obtained an accuracy rate of 93.4% on the test dataset. Additionally, another study used machine learning algorithms to detect botnets using packet features extracted from network traffic data. The proposed method obtained an accuracy rate of 92.5% on the test dataset. Overall, these studies proposed effective solutions to the problem of detecting HTTP-based botnets using machine learning algorithms, with high accuracy rates achieved through various datasets and classifiers.

Related work in the field of botnet attack detection using a honeypot combined with machine learning has been conducted by several researchers. One such [11] study aimed to address the problem of botnet attack detection in IoT smart factories, which results in major production delays and financial damages for suppliers. The study proposed a honeypot and machine learning method to detect botnet attacks. The researchers built a smart industrial environment based on IoT device hardware architecture. They used the Weka machine learning program with the random forest algorithm to achieve a high accuracy score above 96%. The dataset used for the experiment was not explicitly mentioned. The proposed model showed high feasibility in detecting botnet attacks for the security network of smart factories.

Table 1
The summary analysis of studied related literature.

Ref.	Year	Approach	Dataset	Performance accuracy (%)
[9]	2021	Multi-Layer framework	CTU-13	92
[10]	2018	Machine Learning Classifier	Network Traffic	92
[11]	2021	Honeypot combined with ML	10 feature botnet	96
[12]	2019	Multi-Layer	CTU and ISOT	98
[13]	2021	hybrid deep learning, long short-term memory, convolutional neural network	N-BaIoT	90
[14]	2022	decision tree, an XgBoost model, and a logistic regression model	UNSW-NB15	94
[15]	2022	Meta-learning botnet detection models	Aposemat IoT-23 GarciaApose, UC Irvine KDD99Dua: 2019KDD99, and UNSW TONbooijTON dataset	97

This study [12] proposed a method that aims to address the challenge of identifying P2P botnets, which have unique characteristics that make them difficult to detect. Machine learning classifiers are used to analyze network traffic features and detect botnets related to P2P. The multilayer approach was the proposed approach. The ISOT and CTU-13 datasets were utilized for machine learning classifier model training and testing. The proposed method uses a decision tree algorithm for feature selection and achieves an accuracy score of 98.7%. A dataset of network traffic captured from a real-world network environment is utilized to validate the proposed method's performance.

This paper [13] proposed botnet attack detection using the hybrid CNN-LSTM Method for Internet of Things application forms. Extensive experimental research was performed using a real N-BaIoT dataset, including benign and malicious patterns, extracted from a real system. The present research proposes a novel algorithm called deep hybrid learning, which combines a convolutional neural network and long short-term memory (CNN-LSTM) to detect botnet attacks. The experimental outcomes demonstrate the CNN-LSTM model's effectiveness in detecting botnet attacks from doorbell devices (Danminin and Ennio brands) with accuracy rates of 90.88% and 88.61%. Similarly, the proposed algorithm achieves an acceptable accuracy rate of 88.53% for identifying botnet attacks from thermostat devices. Regarding accuracy metrics, the proposed system performs reasonably well in detecting botnet attacks from security cameras, with accuracy rates of 87.19%, 89.23%, 87.76%, and 89.64%.

The detection of botnet attacks in IoT devices Using machine learning methods was proposed in this study [14]. The proposed methodology involved utilizing a decision tree, XgBoost model, and logistic regression model that were trained, tested, and evaluated on the provided dataset. The UNSW-NB15 dataset was utilized for training and testing. The decision tree method in this study outperformed with 94% test accuracy.

The lightweight meta-learning-based methods for botnet attack detection were proposed in this study [15]. The meta-learning botnet detection methods were the proposed approach with high-performance scores. Aposemat IoT-23 GarciaApose, UC Irvine KDD99Dua: 2019KDD99, and UNSW TONbooijTON datasets were utilized for training and performance evaluations. The applied study model results show they achieved 97.9%

The article [16] under examination focuses on the pressing issue of IoT Botnets and their potential impact on the dependability of IoT systems [17], which is exacerbated by the limited resources of IoT devices. The study's main objective was to deploy the state-of-the-art CTGAN model, a Generative Adversarial Networks approach specialized in tabular data modeling and generation. In pursuit of this goal, the researchers worked with an imbalanced dataset called Bot-IOT and devised practical techniques to address the problem. Encouragingly, the findings demonstrate promising outcomes, revealing that after employing CTGAN for data augmentation, the Multi-layer Perceptron (MLP) achieved an impressive accuracy of 98.93% in successfully detecting IoT Botnet attacks. The performance scores for attack detection are good; however, further enhancements are still needed.

This study [18] introduces a novel Intelligent IoT-BOTNET attack detection model that utilizes an optimized hybrid classification technique. The research leverages the IoT-botnet dataset for model training and testing. An enhanced Information Gain (IIG) model is employed to identify the most reliable features from the data. The detection model is a hybrid classifier that integrates the optimized Bi-GRU with the Recurrent Neural Network (RNN). Additionally, the research proposes a novel hybrid optimization approach called SMIE (Slime Mould with Immunity Evolution), which combines two established optimization methods: Coronavirus herd immunity optimizer (CHIO) and the Slime mould algorithm. Remarkably, the projected model achieves a detection accuracy of 97%, but there is still a three percent error in accuracy that needs to be addressed.

IoT security and botnet attack detection are the two basic areas into which the related activities may be divided. Diverse strategies have been put forth in the domain of IoT security to deal with the security issues provided by IoT devices. One strategy to prevent unwanted people or devices from connecting to IoT networks is to deploy access control techniques. Utilizing encryption methods is another strategy for preserving the integrity and secrecy of IoT data. In several research, it has also been suggested to employ blockchain technology to guarantee the security and reliability of IoT networks.

Researchers have created several methods to identify and stop botnet attacks on Internet of Things (IoT) devices under the area of botnet attack detection. One method is to examine network traffic data using machine learning techniques to spot unusual activity that could point to a botnet assault. Another strategy is to identify known botnet assaults using approaches that rely on pre-established patterns or signatures. Furthermore, several researchers have suggested using honeypots or decoy gadgets to entice botnets and learn more about their activity.

By utilizing an ensemble learning technique to increase the accuracy and resilience of the detection system, the suggested approach in this research expands upon earlier work in the botnet attack detection field. The suggested strategy is able to achieve high accuracy in identifying various forms of botnet attacks on IoT devices with a low false positive rate thanks to the employment of numerous machine learning models and optimization approaches. The suggested method also overcomes the drawbacks of earlier methods by detecting botnet assaults even when the network traffic data is altered by adversarial attacks.

Overall, the related studies in IoT security and botnet attack detection offer a useful framework for the strategy suggested in this research and emphasize the significance of IoT device security measures. Through the network attacks-related literature analysis, we have identified numerous research gaps to solve. The following are research points that need to cover:

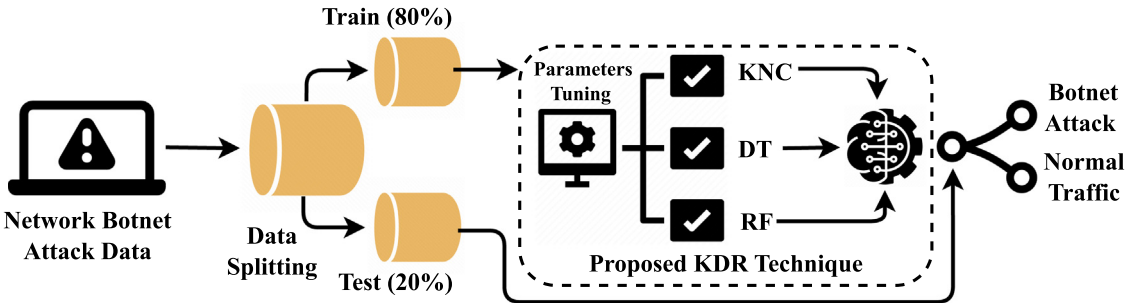

Fig. 1. The methodology analysis of our proposed study for botnet attack detection.

- Previously published studies primarily used classical machine learning and deep learning approaches for network attack detection. Advanced ensemble learning-based approaches must need to implement for effective network attack detection.
- The performance accuracy scores for network attack detection in previously published studies are also low. This analysis's highest performance score is 99.7%, which can be further improved for efficient network attack detection.

3. Attack detection methodology

In this section, we have provided a detailed explanation of the study methods and their workflow. Additionally, we have analyzed the dataset that is utilized to build the applied methods and evaluate their performance with different hyperparameters. Furthermore, we have described the proposed ensemble learning approach architecture and its mathematical algorithm to provide a clear understanding of its functionality. By doing so, we aim to provide a comprehensive overview of our methodology and enable readers to replicate and further develop our approach.

Fig. 1 shows our proposed study methodology analysis. The botnet attack-based benchmark dataset is used to carry out our proposed study experiments. The imported dataset is then split into two portions. One portion is used for training, and the other is used for testing the performance of applied artificial intelligence techniques for unseen data. We proposed a novel ensemble approach that combines the KNC, DT, and RF techniques. The proposed ensemble approach is fully optimized hyperparameters, showing efficient botnet attack detection results. The performance of the proposed ensemble model is assessed and used for detecting the botnet attack.

3.1. Botnet attack data

The publicly available benchmark CTU-13 dataset [19] is used for our study experiments. In 2011, the CTU-13 dataset was obtained from a university located in the Czech Republic known as CTU University. This dataset's main goal is to capture real botnet attack traffic and normal traffic samples on the network. The thirteen scenarios related to malware were executed to create the CTU-13 dataset. The types of network attacks included in the CTU-13 dataset are IRC, Click Fraud, DoS, Port Scan, Spam traffic, and Fast Flux. The used dataset is based on the 58 features related to normal and botnet attack traffic. The dataset contains the labels normal traffic (0) and botnet attack traffic (1) related features as illustrated in Fig. 2. The dataset distribution analysis shows that the dataset is imbalanced. The normal traffic contains a sample of 53,314, and botnet attack traffic contains samples of 38,898 in this study.

3.2. Data splitting

Data splitting includes partitioning research data into multiple subsets, including a training set for model training and a testing set for performance evaluation. This partitioning aids in detecting overfitting problems, where the model is excessively intricate and fits too tightly to the training data, resulting in inadequate performance on the testing data. In this research, we have adopted an 80:20 splitting ratio, with 80% of the dataset used for training the applied techniques and the remaining 20% utilized for assessing the effectiveness of the applied methods.

3.3. Applied machine and deep learning approaches

In recent years, network attacks have become more sophisticated and harder to detect, which has led to a need for advanced techniques to identify and prevent them. Machine learning and deep learning approaches have been increasingly used for network attack detection due to their ability to learn patterns and identify anomalies in large datasets. Advanced machine learning algorithms have been applied to network traffic data to identify attack traffic patterns. On the other hand, numerous deep learning methods have shown promising results in detecting complex attacks such as intrusion attacks and distributed DoS attacks.

3.3.1. Decision trees

Decision Trees (DT) are a machine learning approach frequently used for classification and regression problems [20]. The algorithm constructs a tree-like representation of decisions and their potential outcomes, allowing for easy visualization and interpretation of the decision-making process. Every internal node of the tree represents a decision based on a feature, while each leaf node represents a class label or numerical value. The goal is to create a tree that accurately predicts the target variable. Decision Trees are widely used in various fields, including finance, healthcare, and marketing, to make data-driven decisions [21].

A decision tree is a model of decisions and their outcomes that resembles a tree. It is made out of branches that indicate potential outcomes or their probability and nodes that reflect the choice or chance event. A collection of equations or inequalities that explain the decision rules and the resulting probability can be used to depict a decision tree mathematically [22].

A decision tree with n nodes and m branches is called T. With a collection of branches B_i that stand in for the potential outcomes or their probabilities, each node i represents a choice or a chance event, indicated by D_i. Each branch j of node i leads to a child node k and has a chance of p_i, j of occurring.

A set of equations or inequalities that explain the circumstances under which a choice is taken or a chance event takes place can serve as a representation of the decision rules of a decision tree. Let X, for

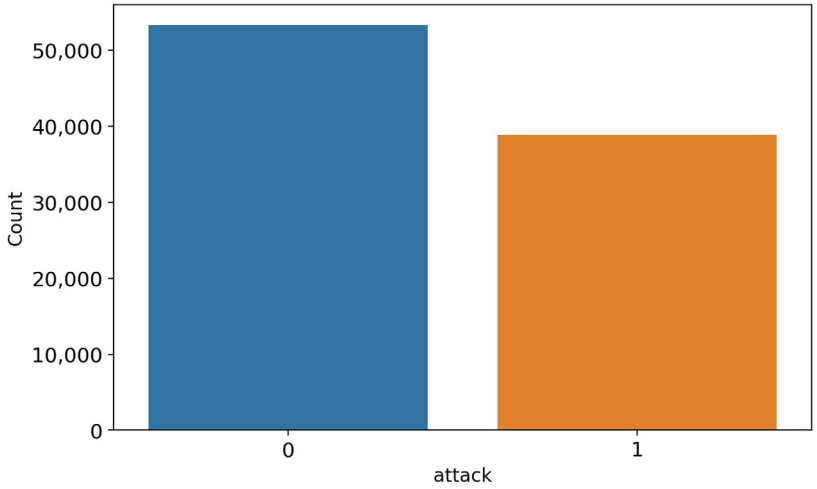

Fig. 2. The botnet attack target class distributions analysis.

instance, represent a characteristic or property of the data, and a serve as a fixed threshold. One way to represent the node i decision rule is as follows:

$$D_i = \begin{cases} 1 & \text{if } X < a\ 0 \\ \text{otherwise} \end{cases} \tag{1}$$

A set of conditional probabilities or joint probabilities that explain the likelihood of each result given the decision rules and prior outcomes can be used to represent the outcome probabilities of a decision tree. Let b be a constant value and let Y be a random variable that reflects the target variable or class label of the data. Given the decision rule D_i and the preceding outcome $Y_i - 1$, the probability of outcome j of node i may be written as follows:

$$p_{i,j} = P(Y_i = j | D_i, Y_{i-1} = b) \tag{2}$$

Using different methods like ID3, C4.5, or CART, the decision rules and outcome probabilities of a decision tree may be learnt from a training dataset. Once it has been trained, the decision tree may be used to forecast the future or categorize brand-new occurrences according to their feature values and the tree's decision-making rules.

3.3.2. Random forest

Random Forest (RF) is a well-known machine learning algorithm that leverages the concept of ensemble learning to enhance prediction accuracy [23]. The algorithm generates several decision trees, each trained on a random subset of the data and a random subset of features. Subsequently, it aggregates the results of all the decision trees to produce a final prediction. The RF is a popular method used for classification and regression tasks because it is relatively simple, less prone to overfitting than individual decision trees, and can handle high-dimensional datasets with many features [24].

Multiple decision trees are used in Random Forest, an ensemble learning technique, to increase the reliability and accuracy of classification or regression tasks. It comprises several decision trees that are aggregated to provide the final prediction after being trained on various subsets of the training data and the characteristics. The decision rules and outcome probabilities of the decision trees and the ensemble are described by a collection of equations or inequalities that serve as Random Forest's mathematical representation [25].

With n decision trees and m features, let F be a Random Forest. Each decision tree i is trained using a subset of the features F_i and training data D_i. With a set of branches B_i, j that stand in for the potential outcomes or their probabilities, each node j of the tree i represents a choice or a chance event, indicated by D_i, j. Each branch k of node j leads to a child node l and has a probability of occurrence p_i, j, k.

A set of conditional probabilities or joint probabilities that explain the likelihood of each result given the decision rules and prior outcomes can be used to represent the outcome probabilities of a decision tree. Let b be a constant value and let Y be a random variable that reflects the target variable or class label of the data. Given the decision rule D_i, j and the preceding outcome $Y_i, j - 1$, the probability of outcome k of node j of tree i may be written as follows:

$$p_{i,j,k} = P(Y_{i,j} = k | D_{i,j}, Y_{i,j-1} = b) \tag{3}$$

The probabilities of all decision trees may be combined to get the Random Forest's outcome probability. For classification tasks and regression tasks, the mean or median are the most often used aggregation methods. Let y_i represent the tree i's expected value for a certain instance. The predicted value of the Random Forest may be written as follows:

$$y_{RF} = \text{vote}(y_1, y_2, \ldots, y_n) \tag{4}$$

Various techniques, including bagging, random subspaces, and random patches, may be used to learn the decision criteria and result probabilities of a Random Forest using a training dataset. Using feature values and the ensemble's decision rules, the Random Forest may be trained to forecast the future or categorize brand-new data.

3.3.3. Adaboost

AdaBoost, Adaptive Boosting (AB), is a popular ensemble learning method used for regression and classification tasks in machine learning [26]. It is a meta-algorithm that combines multiple "weak" classifiers into a "strong" classifier. The idea is to sequentially train the weak classifiers on different subsets of the data, with each subset weighted based on the performance of the previous classifiers. This allows the algorithm to focus on the misclassified instances and improve the overall accuracy of the classifier. AdaBoost is effective in various applications, including image and speech recognition, text classification, and data mining [27].

The following is how Adaboost stated mathematically:

Let $X = x_1, x_2, \ldots, x_n$ be the training dataset, where each instance x_i is associated with a label y_i. Let $H = h_1, h_2, \ldots, h_T$ be a set of weak classifiers that take an instance x as input and output a binary prediction $h(x) \in -1, 1$. Let $D_t = w_{t,1}, w_{t,2}, \ldots, w_{t,n}$ be the weight distribution over the training data at iteration t, where $w_{t,i}$ is the weight assigned to instance i at iteration t [28]. The Adaboost algorithm can be described as follows:

1. Initialize the weight distribution D_1 to be uniform over the training data, i.e., $w_{1,i} = \frac{1}{n}$ for all i.
2. For each iteration $t = 1, 2, \ldots, T$:

- Train a weak classifier h_t on the training data X using the weight distribution D_t.
- Compute the weighted error ϵ_t of the weak classifier on the training data, i.e., $\epsilon_t = \sum_{i=1}^{n} w_{t,i} \mathbb{1}(h_t(x_i) \neq y_i)$, where $\mathbb{1}(\cdot)$ is the indicator function.
- Compute the weight α_t of the weak classifier in the final prediction, i.e., $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$.
- Update the weight distribution D_{t+1} by reweighting the misclassified instances, i.e., $w_{t+1,i} = \frac{w_{t,i}\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, where Z_t is the normalization factor.

3. Compute the final prediction $H(X) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(X))$, where $\text{sign}(\cdot)$ is the sign function.

Adaboost is a powerful classifier with minimal training error and high generalization performance that is guaranteed to converge. Adaboost's strength resides in its capacity to concentrate on the challenging examples in the training data and give them more weight, increasing the accuracy of the final prediction.

3.3.4. Gradient boosting

Gradient Boosting (GB) [29] is a machine-learning method that involves building a series of models, each of which attempts to rectify the errors made by the previous model in the series. This technique involves iteratively fitting a model to the data and then adjusting the weights of the data points based on how well the model predicted their values. The key idea behind Gradient Boosting is to use the gradient value of the loss function to guide the model updates. This technique is effective for various supervised learning problems, including classification, regression, and ranking [30].

The mathematical representation of Gradient Boosting can be expressed as follows:

1. Initialize the prediction $F_0(x) = \text{argmin}\gamma \sum i = 1^n L(y_i, \gamma)$, where $L(y_i, \gamma)$ is the loss function that measures the error of predicting γ for instance i [31].
2. For each iteration $t = 1, 2, \ldots, T$:
   - Compute the negative gradient of the loss function with respect to the current prediction, i.e., $r_{t,i} = -\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}$.
   - Train a weak learner h_t on the residuals $r_{t,i}$ using the training data X.
   - Compute the optimal step size γ_t that minimizes the loss function, i.e., $\gamma_t = \text{argmin}\gamma \sum i = 1^n L(y_i, F_{t-1}(x_i) + \gamma h_t(x_i))$.
   - Update the prediction $F_t(x) = F_{t-1}(x) + \gamma_t h_t(x)$.
3. Compute the final prediction $F_T(x)$ of the ensemble.

A robust and popular approach, gradient boosting may be used to solve a variety of machine learning issues, including regression and classification. Gradient Boosting excels at handling intricate, nonlinear interactions between features and the target variable by repeatedly improving the prediction utilizing the residuals of the prior weak learners.

3.3.5. Logistic regression

Logistic Regression (LR) is a statistical technique commonly employed for solving binary classification problems [32]. The method estimates the correlation between one or more independent variables and a binary dependent variable. LR uses a logistic function to model the probability of the dependent variable taking on a particular value. It is a popular and interpretable binary result prediction technique in data science, machine learning, and social sciences [33,34].

The mathematical representation of Logistic Regression can be expressed as follows:

Let $X = x_1, x_2, \ldots, x_n$ be the training dataset, where each instance x_i is associated with a binary output variable $y_i \in 0, 1$ and a set of input variables $x_{i1}, x_{i2}, \ldots, x_{ip}$. Let $\beta = \beta_0, \beta_1, \ldots, \beta_p$ be a set of coefficients that we want to estimate. The logistic regression algorithm can be described as follows:

1. Compute the weighted sum of the input variables and the coefficients, i.e., $z = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$.
2. Apply the logistic function to the weighted sum to obtain the probability of the output variable, i.e., $p(y = 1|x) = \frac{1}{1+e^{-z}}$.
3. Estimate the coefficients that maximize the likelihood of the training data, i.e., $\hat{\beta} = \text{argmax}\beta \prod i = 1^n p(y_i|x_i; \beta)$.
4. Use the estimated coefficients to make predictions on new instances, i.e., $\hat{y} = \text{argmax}_y p(y|x; \hat{\beta})$.

The logistic function is a sigmoid function, making it appropriate for binary classification issues since it transfers any real value to the range [0, 1]. The maximum likelihood estimation approach estimates the coefficients that maximize the likelihood function, which represents the likelihood of witnessing the training data given the coefficients. A variety of binary classification issues may be solved with the help of the straightforward and understandable approach known as logistic regression.

3.3.6. Ridge classifier

A Ridge Classifier (RC) is a machine-learning approach that is utilized for classification problems [35]. It is an extension of the logistic regression algorithm that uses the L2 regularization technique to prevent overfitting and improve the model's generalization performance. By leveraging the input features, the RC model is capable of predicting the likelihood of each class label and ultimately selects the class with the highest probability as its output. The Ridge Classifier algorithm is particularly useful for high-dimensional data containing many irrelevant features, as it can reduce the impact of these features and improve the model's accuracy. It has applications in various fields, such as image recognition, text classification, and bioinformatics [36].

A linear classifier called the Ridge Classifier employs L2 regularization to avoid overfitting the training set. The L2 norm is used to penalize the magnitude of the coefficients while the algorithm fits a linear model to the input data. Ridge Classifier's mathematical representation is best summed up as follows:

1. Compute the weighted sum of the input variables and the coefficients, i.e., $z = \sum_{j=1}^{p} w_j x_j$.
2. Apply the sign function to the weighted sum to obtain the predicted output variable, i.e., $\hat{y} = \text{sign}(z)$.
3. Estimate the weights that minimize the regularized sum of squares loss, i.e., $\hat{w} = \text{argmin}w \sum i = 1^n (y_i - \text{sign}(\sum_{j=1}^{p} w_j x_{ij}))^2 + \alpha \sum_{j=1}^{p} w_j^2$.
4. Use the estimated weights to make predictions on new instances, i.e., $\hat{y} = \text{sign}(\sum_{j=1}^{p} \hat{w}_j x_j)$.

The sign function is a step function that works well for binary classification issues since it converts every real number to either −1 or 1. The regularized sum of squares loss function uses the L2 norm to penalize the size of the weights and measures the sum of squares of the errors between the expected and actual output variables. The L2 regularization intensity is controlled by the hyperparameter $alpha$, which may be adjusted to enhance the algorithm's performance on the validation data. Ridge Classifier is a straightforward and efficient linear classifier that handles high-dimensional input data while avoiding overfitting.

3.3.7. SGD classifier

The Stochastic Gradient Descent (SGD) Classifier is a linear classification algorithm widely used in machine learning applications [37]. It is based on SGD optimization, a widely used optimization technique for training large-scale machine learning models. The algorithm works

by iteratively adjusting the weights of a linear model in the course of the loss function's negative gradient. The SGD Classifier is known for its efficiency, scalability, and ability to handle large datasets, making it a popular choice in many real-world applications [38].

The mathematical representation of the SGD Classifier can be expressed as follows:

Let $X = x_1, x_2, \ldots, x_n$ be the training dataset, where each instance x_i is associated with a binary output variable $y_i \in -1, 1$ and a set of input variables $x_{i1}, x_{i2}, \ldots, x_{ip}$. Let $w = w_1, w_2, \ldots, w_p$ be a set of weights that we want to estimate. The SGD Classifier algorithm can be described as follows:

1. Initialize the weights with small random values, i.e., $w \sim \mathcal{N}(0, \sigma^2)$.
2. For each iteration t, randomly select a subset of the training data $S_t \subseteq X$.
3. Compute the weighted sum of the input variables and the coefficients for the instances in S_t, i.e., $z_t = \sum_{i \in S_t} w_i x_i$.
4. Compute the gradient of the loss function with respect to the weights using the instances in S_t, i.e., $\nabla_w \mathcal{L}(w) = \frac{1}{|S_t|} \sum_{i \in S_t} y_i x_i (1 - \sigma(y_i z_i)) + 2\alpha w$.
5. Update the weights using the gradient descent rule, i.e., $w_{t+1} = w_t - \eta \nabla_w \mathcal{L}(w)$, where η is the learning rate.
6. Repeat steps 2–5 until convergence or a maximum number of iterations is reached.
7. Use the estimated weights to make predictions on new instances, i.e., $\hat{y} = \text{sign}(\sum_{j=1}^{p} \hat{w}_j x_j)$.

The weighted sum of the input variables and coefficients is mapped to the range [−1, 1] using the sigmoid function, which is appropriate for binary classification tasks. The loss function penalizes the size of the weights using the L2 norm and calculates the sum of squares of the errors between the anticipated output variable and the actual output variable. The L2 regularization strength and learning rate are controlled by the hyperparameters *alpha* and *eta*, respectively, and may be adjusted to enhance the algorithm's performance on the validation data. A versatile and effective linear classifier that can handle large-scale, high-dimensional datasets is the SGD Classifier.

3.3.8. Support vector classifier

The Support Vector Classifier (SVC) is a widely-used machine-learning algorithm that is specifically designed for classification tasks [39]. The algorithm operates by determining the hyperplane that optimally separates the classes in the dataset. This hyperplane is selected to maximize the distance between the two classes, resulting in an improvement in the model's ability to generalize. The SVC algorithm is flexible enough to work with both linear and non-linear data, and it employs diverse kernels such as polynomial and radial basis function kernels. This versatility has led to its adoption in a variety of fields, including image classification, text classification, and bioinformatics [40].

SVC is a linear classifier that divides the data points into the various classes with the greatest margin of error. The following is how SVC may be stated mathematically:

Let $X = x_1, x_2, \ldots, x_n$ be the training dataset, where each instance x_i is associated with a binary output variable $y_i \in -1, 1$ and a set of input variables $x_{i1}, x_{i2}, \ldots, x_{ip}$. The goal of SVC is to find a hyperplane defined by the equation $w^T x + b = 0$, where $w = w_1, w_2, \ldots, w_p$ is the weight vector and b is the bias term.

The hyperplane should satisfy the following conditions:

1. It should separate the data points of different classes with the largest margin.
2. It should minimize the classification error.

The optimization problem can be formulated as follows:

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n \tag{5}$$

the inequality constraints guarantee that the data points of various classes are separated by a distance of at least one, and $\|w\|^2$ is the weight vector's L2 norm. The Karush–Kuhn–Tucker criteria and the Lagrange dual formulation can be used to solve the optimization issue.

Once the weight vector and bias term are estimated, the predicted output variable for a new instance x can be computed as $\hat{y} = \text{sign}(w^T x + b)$. SVC is a versatile and effective linear classifier that uses several kernel functions, such as polynomial, radial basis function (RBF), and sigmoid kernels, to handle both linearly separable and non-linearly separable datasets. To improve the performance of the method on the validation data, the SVC's hyperparameters, such as the regularization parameter C, the kernel function, and the kernel parameters, can be adjusted.

3.3.9. K-Neighbors classifier

K-Neighbors Classifier (KNC) is a popular supervised learning algorithm utilized for classification tasks in machine learning [41]. It operates by locating the k nearest neighbors to a new data point and categorizing it based on the majority of those neighbors' classes. KNC is a simple and effective algorithm but can suffer from the curse of dimensionality and can be computationally expensive with large datasets. It is often used as a baseline model for comparison with more complex algorithms.

KNN is a non-parametric method that operates by locating a new instance's k-nearest neighbors and predicting its output variable based on the output variable that is shared by the majority of the k neighbors [42]. The following is how KNN may be stated mathematically:

Let $X = x_1, x_2, \ldots, x_n$ be the training dataset, where each instance x_i is associated with a categorical output variable $y_i \in 1, 2, \ldots, K$ and a set of input variables $x_{i1}, x_{i2}, \ldots, x_{ip}$. KNN aims to predict the output variable y for a new instance x based on its k-nearest neighbors.

The distance between two instances x_i and x_j can be measured using various metrics such as Euclidean distance, Manhattan distance, and Minkowski distance. The k-nearest neighbors of x can be identified by sorting the training instances in increasing order of their distances to x and selecting the k instances with the smallest distances [43].

Once the k neighbors are identified, a majority vote rule can compute the predicted output variable for x. Specifically, the predicted output variable for x is the class label k that appears most frequently among the k neighbors:

$$\hat{y} = \arg \max_{k \in 1,2,\ldots,K} \sum_{i \in N_k(x)} [y_i = k] \tag{6}$$

where $N_k(x)$ is the set of indices of the k-nearest neighbors of x and $[y_i = k]$ is an indicator function that takes the value 1 if $y_i = k$ and 0 otherwise. A random one is selected if there is a tie among the most common output variables.

KNN is an easy-to-understand technique that may be applied to regression and classification applications. The performance of the method on the validation data may be optimized by adjusting the KNN hyperparameters, such as the number of neighbors k and the distance measure. However, because KNN must calculate the distances between every pair of instances, it can be computationally costly for huge datasets and high-dimensional input spaces.

3.3.10. Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is a probabilistic approach for machine learning classification tasks [44]. Based on the Bayes theorem, it presupposes that the characteristics are independent and distributed normally. The algorithm is simple and fast, making it ideal for large datasets with high-dimensional features. However, the feature independence assumption can sometimes limit its accuracy. Gaussian Naive Bayes is frequently used in text classification and spam filtering.

A probabilistic technique called Gaussian Naive Bayes makes use of the Bayes theorem to forecast the class probabilities of a new instance depending on its input variables. Following are some examples of how to express Gaussian Naive Bayes mathematically:

Table 2
Applied RNN model layered architecture analysis.

Layer (Type)	Output shape	Parameters
RNN Layer	(None, 8)	80
Dropout Layer	(None, 8)	0
Output layer	(None, 1)	9
Total Parameter		89

Let $X = x_1, x_2, \ldots, x_n$ be the training dataset, where each instance x_i is associated with a categorical output variable $y_i \in 1, 2, \ldots, K$ and a set of input variables $x_{i1}, x_{i2}, \ldots, x_{ip}$. Gaussian Naive Bayes aims to predict the class probabilities $P(y = k|X = x)$ for a new instance x.

Gaussian Naive Bayes assumes that the input variables are independent and follow a Gaussian distribution within each class. Specifically, the probability density function of the jth input variable x_{ij} for class k can be expressed as follows:

$$P(x_{ij}|y = k) = \frac{1}{\sqrt{2\pi\sigma_{jk}^2}} \exp\left(-\frac{(x_{ij} - \mu_{jk})^2}{2\sigma_{jk}^2}\right) \tag{7}$$

where μ_{jk} and σ_{jk}^2 are the mean and variance of the jth input variable for class k, respectively. The parameters μ_{jk} and σ_{jk}^2 can be estimated from the training data using maximum likelihood estimation. Once the class-conditional probability densities are estimated, the class probabilities for a new instance x can be computed using Bayes' theorem:

$$P(y = k|X = x) = \frac{P(y = k)\prod_{j=1}^{p} P(x_j|y = k)}{\sum_{k'=1}^{K} P(y = k')\prod_{j=1}^{p} P(x_j|y = k')} \tag{8}$$

Where $P(y = k)$ is the prior probability of class k and can be estimated from the training data as the fraction of instances in class k. The denominator in the formula is the marginal likelihood of the data and serves as a normalization factor to ensure that the probabilities sum up to one.

Gaussian Naive Bayes is a straightforward and effective approach that can handle both binary and multi-class classification issues. It is especially beneficial when other algorithms could experience the dimensionality curse in high-dimensional input areas. On the other hand, Gaussian Naive Bayes makes the unproven assumption that the input variables are independent and have a Gaussian distribution.

3.3.11. Recurrent neural network

A Recurrent Neural Network (RNN) is an artificial neural network that uses sequential data to process it [45]. Unlike traditional feed-forward neural networks that take a fixed input size and produce a fixed output size, RNNs can process input sequences of varying lengths and output sequences of varying lengths. RNNs work by maintaining a memory of previous inputs and using this information to inform future predictions. This makes them well-suited to tasks such as language modeling, speech recognition, and machine translation, where the output depends on previous inputs in a sequence. The layered architecture of the applied deep learning-based RNN model is analyzed in Table 2.

3.4. Hyperparameter optimization

Hyperparameter optimization involves selecting the optimal values of model hyperparameters that can significantly impact the model's performance [46]. To improve the performance of applied models, we optimized the hyperparameter by fine-tuning. Our study results demonstrate the importance of hyperparameter optimization in improving the performance results of machine learning methods for detecting botnet attacks. Our study best fit selected hyperparameter analysis is shown in Table 3.

Table 3
The fine-tuning hyperparameters optimization analysis.

Technique	Hyperparameters
DT	max_depth=25, criterion='entropy'
RF	n_estimators=100, max_depth=50, criterion='gini
AB	n_estimators=5
GB	learning_rate=0.1, n_estimators=100, max_depth=3
LR	C=1.0, penalty='l2', solver='liblinear'
RC	alpha=1.0
SGD	loss='hinge', penalty='l2', alpha=1e−3
SVC	kernel='linear', C=1.0, gamma='scale'
KNC	n_neighbors=3, weights='uniform'
GNB	var_smoothing=1e−09
RNN	activation='sigmoid', loss = 'binary_crossentropy', optimizer = 'adam', metrics='accuracy'

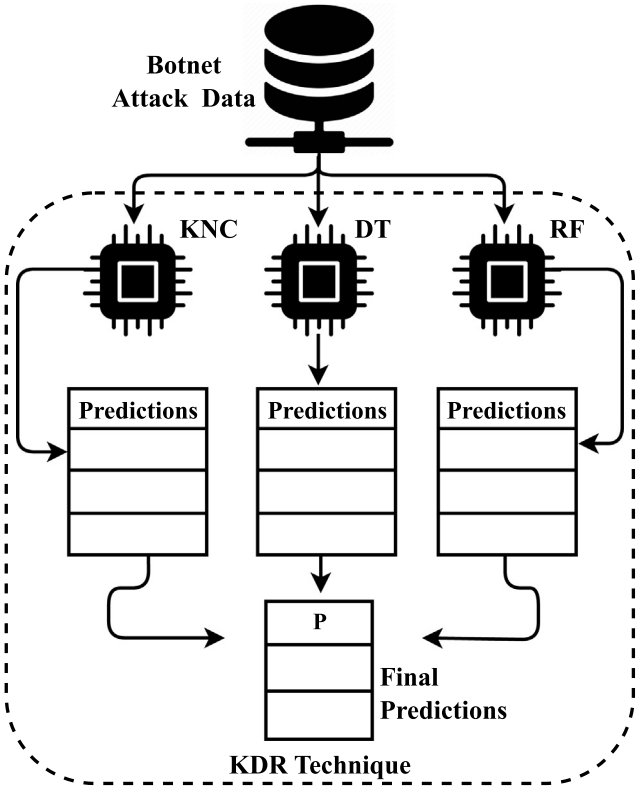

Fig. 3. The proposed ensemble model architecture analysis for botnet attack detection.

3.5. Novel proposed ensemble technique

The proposed novel ensemble learning-based KDR technique is proposed for efficient botnet attack detection, as illustrated in Fig. 3. The advanced machine learning-based KNC, DT, and RF methods are combined to create the proposed ensemble method. The botnet attack dataset is input to each combined model. Each model performs the predictions on the attack dataset. Then a majority voting mechanism is performed to select the final output from the combined model's outputs. Our study performance results show that the proposed ensemble approach achieved high-performance scores for botnet attack detection compared to the state-of-the-art approaches.

Ensemble models have demonstrated remarkable success in achieving high scores for network attack detection. This technique involves

the fusion of multiple models or algorithms to enhance the overall robustness and accuracy of the system [47]. Integrating signature-based, anomaly-based, and machine learning-based detection methods is possible with ensemble models, which expands the system's ability to detect various attack types while improving the overall accuracy. Additionally, ensemble models effectively reduce the likelihood of false positives and negatives by consolidating results from different models. This approach leads to a more dependable and resilient detection system critical for cybersecurity. In summary, the success of ensemble models in network attack detection stems from their ability to merge various detection techniques and improve the system's overall performance.

Algorithm 1 shows the step-by-step flow of the proposed ensemble model.

Algorithm 1 KDR Algorithm

Input: Botnet Attack Networks Dataset (CTU-13)
Output: Network Attack Prediction | Normal Traffic or Botnet Attack.

initiate;
1- $T_{knc} \longleftarrow KNC_{training}(TrS)$ // $TrS \ \epsilon \ CTU-13$, here TrS is training set belong to the original data.
2- $T_{dt} \longleftarrow DT_{training}(TrS)$
3- $T_{rf} \longleftarrow RF_{training}(TrS)$
4- **for** i in (TeS): **do** // $TeS \ \epsilon \ CTU-13$ which is a testing set
5- $KNC_p \longleftarrow T_{KNC}(i)$ // KNC_p is the KNC prediction against samples from TeS
6- $DT_p \longleftarrow T_{DT}(i)$ // DT_p is the DT prediction against samples from TeS
7- $RF_p \longleftarrow T_{RF}(i)$ // RF_p is the RF prediction against samples from TeS
8- $F_{Pred} \longleftarrow \sum_{majority\ vote}\{KNC_p, DT_p, RF_p\}$ // $F_{Pred} \ \epsilon$ Normal Traffic and Botnet Attack. which is final prediction

4. Results and discussion

This section presents a comparative evaluation of the results obtained from implementing machine learning and deep learning techniques. The effectiveness of each applied model in detecting botnet attacks is analyzed, highlighting their respective strengths and weaknesses. The performance metrics of each model are thoroughly discussed in this section to provide a comprehensive understanding of applied methods.

4.1. Experimental setup

The experimental setting used to construct the applied machine learning and deep learning techniques is analyzed in this section. The Python programming language 3.0 is utilized to build and evaluate the applied methods. The Pandas module is used for importing the network attack dataset. The Sklearn module is used for training and testing the machine learning models. The TensorFlow modules are used to train the deep learning model. The Google Colab with a GPU backend and 13 GB RAM with 90 GB of disk space hardware is used to conduct all experiments. The performance evaluation metrics used are the accuracy score, f1 score, precision score, and recall score.

4.2. Results with the machine and deep learning

The time series line graph analysis of the utilized deep learning technique is illustrated in Fig. 4. The time series line graph analysis is based on the performance metrics score comparisons during the training of the RNN model. The analysis shows that the loss score is high within the first three epochs, and the accuracy score is low. Also, the validation loss score is high, and the validation accuracy score is low. After the third epoch, the RNN model determined the optimal weights, gradually improving the performance score. At the last epoch, the loss score is low, and accuracy scores are high for network attack detection.

Table 4
Analysis of performance results from the deployed machine and deep learning methods.

Technique	Accuracy score	Target	Precision	Recall	F1
DT	0.97	0	0.97	0.99	0.98
		1	0.99	0.95	0.97
		Avg.	0.98	0.98	0.98
RF	0.96	0	0.99	0.96	0.97
		1	0.94	0.99	0.96
		Avg.	0.97	0.97	0.97
AB	0.94	0	0.93	0.98	0.95
		1	0.97	0.89	0.93
		Avg.	0.95	0.94	0.94
GB	0.95	0	0.94	0.98	0.96
		1	0.97	0.91	0.94
		Avg.	0.95	0.95	0.95
LR	0.50	0	0.75	0.21	0.33
		1	0.45	0.90	0.60
		Avg.	0.63	0.50	0.45
RC	0.86	0	0.84	0.93	0.88
		1	0.89	0.75	0.81
		Avg.	0.86	0.86	0.85
SGD	0.46	0	0.71	0.12	0.20
		1	0.43	0.93	0.59
		Avg.	0.57	0.53	0.40
SVC	0.85	0	0.84	0.93	0.88
		1	0.88	0.75	0.81
		Avg.	0.85	0.85	0.85
KNC	0.97	0	0.97	0.98	0.98
		1	0.97	0.96	0.97
		Avg.	0.97	0.97	0.97
GNB	0.69	0	0.66	0.98	0.79
		1	0.91	0.29	0.43
		Avg.	0.76	0.69	0.64
RNN	0.58	0	0.58	1.00	0.74
		1	0.00	0.00	0.00
		Avg.	0.34	0.58	0.43

The performance comparison of the employed machine learning and deep learning methods is evaluated in this section. Table 4 compares each applied technique's accuracy, f1 score, precision score, and recall score. The target class-wise performance comparison is also performed. The analysis shows that the DT, RF, and KNC techniques achieved the maximal accuracy scores of 0.97, 0.96, and 0.97, respectively. These techniques also achieved high precision, recall, and f1 scores, indicating their effectiveness in accurately predicting the classification of the botnet attack data. The applied AB, GB, RC, and SVC techniques achieved acceptable scores. However, not the highest. On the other hand, LR, SGD, and RNN techniques performed poorly compared to the other techniques, with accuracy scores of 0.50, 0.46, and 0.58, respectively. These techniques also achieved low recall, precision, and f1 scores, indicating their ineffectiveness in accurately predicting the classification of the botnet attack data. This analysis shows that the performance of the applied machine and deep learning methods can vary significantly with botnet attack data.

The comparative performance analysis presented in Fig. 5 offers valuable insights into the effectiveness of different machine-learning techniques for detecting botnet attacks. The poor performance scores achieved by LR, SGD, GNB, and RNN suggest that these techniques may not be suitable for this particular task. On the other hand, the high-performance scores achieved by applied DT, RF, and KNC indicate their effectiveness in detecting botnet attacks. The findings of this analysis have important implications for developing machine learning-based botnet detection systems. The analysis suggests that future research

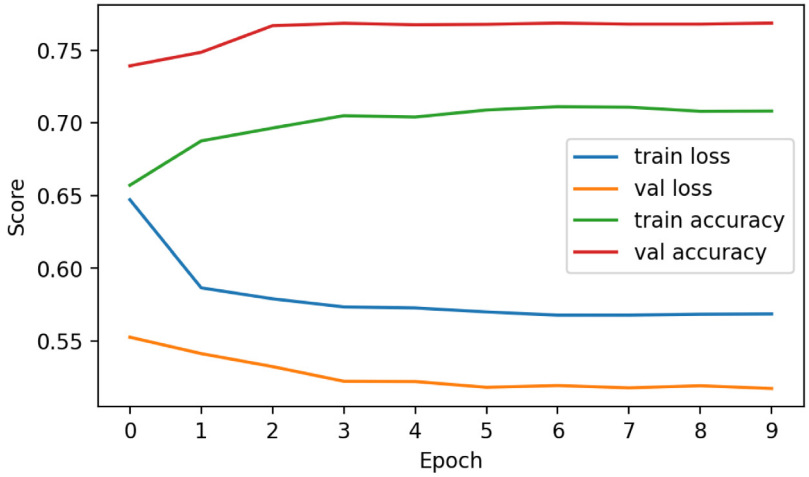

Fig. 4. The time series graph analysis of applied RNN model.

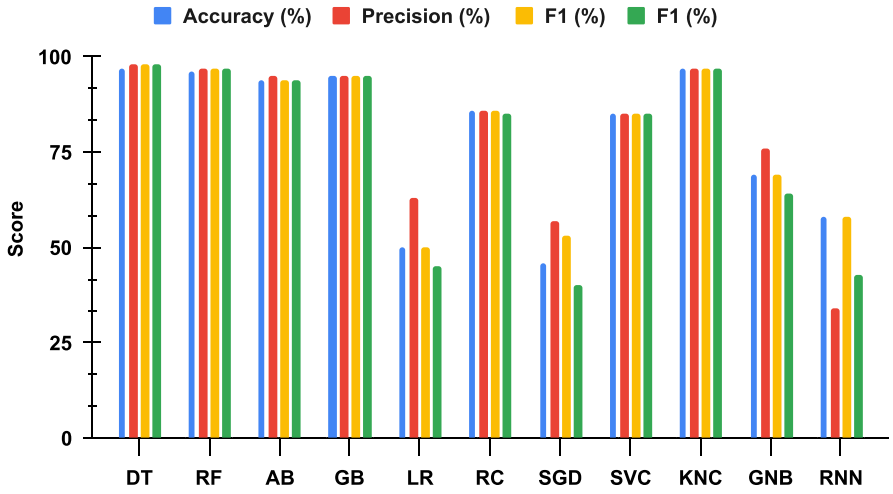

Fig. 5. Using a bar chart, the comparative performance examines applicable machine and deep learning approaches for unseeing test data.

should further refine and optimize the performance of applied DT, RF, and KNC for detecting botnet attacks.

4.3. K-fold cross validation performance analysis

The K-fold cross-validation approach is used to evaluate the performance of our employed machine and deep learning approaches. In Table 5, we have applied the K-fold cross-validation technique to analyze the machine's performance and deep learning models. The analysis demonstrates that the DT and RF classifiers achieved the maximal accuracy scores of 0.97 and 0.98, respectively, with relatively low standard deviations. The KNC method also achieved the same accuracy score as the DT classifier of 0.97, with a low standard deviation of 0.0013. The AB and GB model achieved an acceptable accuracy score of 0.94 and 0.95, respectively, with moderate standard deviation. On the other hand, the LR and SGD classifiers demonstrated relatively low accuracy scores of 0.52 and 0.55, respectively, with a high standard deviation. The SVM model achieved an accuracy score of 0.75, with a moderate standard deviation of 0.0762. In conclusion, the K-fold cross-validation analysis revealed that the DT, RF, and KNC methods performed the best. At the same time, the LR and SGD classifiers demonstrated poor performance in detecting the botnet attack.

4.4. Accuracy performance validation analysis

Fig. 6's validation analysis presents valuable insights into algorithmic accuracy performance, facilitating the selection of the best-suited techniques for specific problems by researchers and practitioners. It is important to note that LR, SGD, GNB, and RNN techniques exhibit subpar results in managing complicated datasets. Thus, their adequacy could be better for high-dimensional features and nonlinear relationships. Meanwhile, the success of DT, RF, AB, GB, and KNC techniques implies potential adaptability to various datasets, such as CTU-13.

4.5. Runtime computations complexity

Table 6 represents the computational complexity analysis of used machine learning and deep learning approaches based on their runtime computation in seconds. The analysis reveals that the RF and RC have achieved the runtime computation of 0.8483 s and 0.1700 s, respectively, suggesting their efficiency in processing large datasets. The KNC and GNB also achieved a minimum runtime computation of 0.0522 s and 0.1146 s, respectively, indicating their potential for fast data processing. In contrast, the deep learning-based RNN technique took a significantly longer computation time of 267.349 s, indicating its limitations in processing large datasets. Moreover, the techniques such as AB, GB, and SGD have also shown relatively higher computation time, indicating their potential inefficiency in processing large datasets. This analysis suggests that the RF, KNC, and RC techniques could be considered as promising approaches for fast processing of data, while the deep learning-based RNN technique may require further optimization for efficient processing.

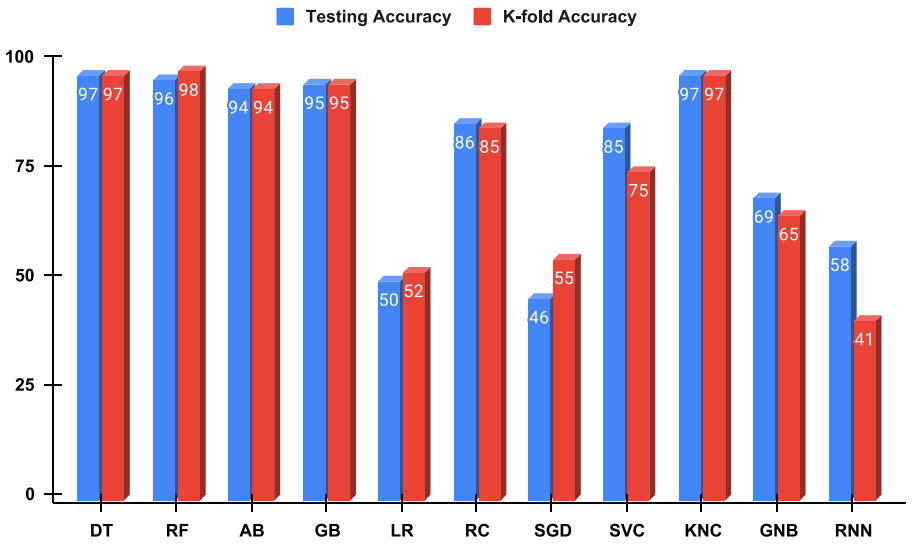

Fig. 6. The bar chart-based accuracy performance validation analysis.

Table 5
Performance analysis of applied machine and deep learning approaches using K-fold cross-validation.

Technique	k-fold	Accuracy score	Standard deviation (+/−)
DT	10	0.97	0.0013
RF	10	0.98	0.0056
AB	10	0.94	0.0044
GB	10	0.95	0.0019
LR	10	0.52	0.0187
RC	10	0.85	0.0028
SGD	10	0.55	0.0985
SVC	10	0.75	0.0762
KNC	10	0.97	0.0013
GNB	10	0.65	0.0049
RNN	10	0.41	0.3154

Table 6
Analysis of the computational complexity of the applicable machine and deep learning algorithms.

Technique	Runtime computation (seconds)
DT	1.3090
RF	0.8483
AB	4.1522
GB	4.7227
LR	1.7952
RC	0.1700
SGD	8.3257
SVC	0.2534
KNC	0.0522
GNB	0.1146
RNN	267.349

4.6. Results of proposed ensemble learning technique

Table 7 provides the results of our proposed ensemble learning technique, where the performance analysis has been carried out based on various evaluation metrics. The performance results are evaluated based on testing accuracy, 10-fold accuracy, standard deviation (+/-), precision, recall, and f1 Scores. The analysis reveals that the proposed KDR technique achieved high accuracy scores of 0.997 for testing and 10-fold cross-validation. The standard deviation score of +/- 0.0004 also suggests that the proposed method results are consistent and reliable. Moreover, the proposed KDR technique achieved 100% scores for precision, recall, and f1 for target classes 0 and 1. The class-wise results show that the weighted average scores are 100% for precision, recall, and f1, indicating that the ensemble learning technique has performed exceptionally well overall. The analysis results suggest that the proposed ensemble learning technique is a promising approach for detecting botnet attacks.

Fig. 7 analysis of the confusion matrix indicates that the KDR approach has a high TPR and a low FPR, thereby correctly identifying botnet attacks while reducing false alarms. This feature is especially critical in real-world situations where false alarms can lead to severe financial and reputational damage. Furthermore, the KDR approach achieves a high TNR, accurately identifying non-attack traffic and minimizing the risk of legitimate traffic being misclassified as botnet traffic, which could result in unnecessary disruptions to normal operations. Overall, the confusion matrix analysis results provide compelling evidence of the efficacy of the KDR approach in botnet attack detection, suggesting that it could be deployed in real-world environments with high accuracy and reliability.

Our proposed model has demonstrated exceptional performance through extensive confusion matrix analysis, exhibiting a mere 53 erroneous predictions amidst an impressive 18,390 correct predictions during rigorous testing. These remarkable results unequivocally underscore our model's high accuracy and robustness in effectively detecting botnet attacks. The significance of such accuracy cannot be overstated, as it establishes a solid foundation for bolstering cybersecurity measures and proactively safeguarding critical systems against potential threats.

4.7. Comparisons with state-of-the-art studies

The comparative results analysis with state-of-the-art studies using the CTU-13 dataset is analyzed in Table 8. The state-of-the-art studies from the year 2017 to 2023 are taken for comparative analysis. The lowest performance accuracy score is 0.86, achieved by state-of-the-art techniques, which is low. The analysis demonstrates that our proposed ensemble approach outperformed the state-of-the-art approaches with the highest performance score of 0.997.

4.8. Discussions

The botnet attack detection in the network using an optimized ensemble learning approach is proposed in this study. Ten advanced machine learning and one deep learning-based method are applied in comparisons to evaluate the performance. The publicly available benchmark CTU-13 dataset is utilized to evaluate performance scores of all used machine learning and deep learning approaches. We have used

Table 7
Performance analysis of proposed ensemble learning technique.

Technique	Accuracy score	10-fold accuracy score	Standard deviation (+/−)	Target attack	Precision score	Recall score	F1 score
KDR	0.997	0.997	0.0004	0	1.00	1.00	1.00
				1	1.00	1.00	1.00
				Avg.	1.00	1.00	1.00

Table 8
The performance comparisons analysis with state-of-the-art studies using the CTU-13 dataset.

Ref.	Year	Technique	Performance accuracy
[9]	2021	Multilayer framework	0.92
[48]	2020	BotEye	0.98
[49]	2017	Random forest	0.93
[50]	2019	X-Means	0.95
[51]	2021	ESVNN	0.86
[52]	2022	DT	0.98
[53]	2023	DT	0.92
[54]	2022	KNN	0.96
[55]	2023	Multilayer perceptron	0.98
Our study	**2023**	KDR	**0.997**

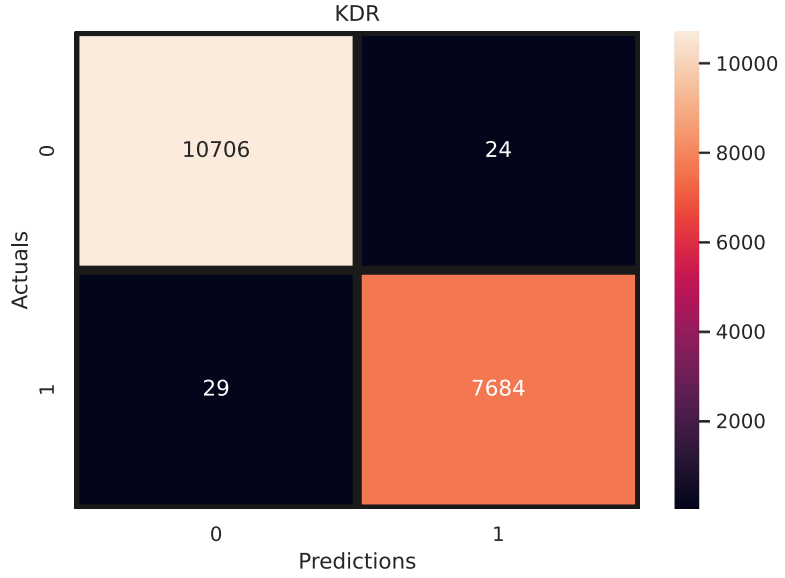

Fig. 7. The confusion matrix analysis of the proposed ensemble learning model for botnet attack detection.

the Python programming language 3.0 to build the applied methods. The study results show that the proposed KDR approach achieved the highest performance score of 99.7% in comparison.

All applied techniques are fully hyperparameter optimized and k-fold cross-validated in this study. The mathematical algorithm notation is also expressed to understand the architecture of the proposed approach. The runtime computations complexity analysis shows that our applied technique has the potential to detect network attacks with minimal time. Finally, the comparisons with state-of-the-art studies show the superiority of our proposed approach.

The related literature analysis identified that the previously published studies primarily used classical machine learning and deep learning approaches for network attack detection. Advanced ensemble learning-based approaches must need to implement for effective network attack detection. The performance accuracy scores for network attack detection in previously published studies are also low.

5. Conclusions

The ensemble learning approach for network botnet attack detection is proposed in this study. We use the benchmark CTU-13 dataset to build the employed machine learning and deep learning approaches in comparison. We proposed a novel ensemble technique KDR for botnet attack detection to achieve high performance. The advanced machine learning-based KNC, DT, and RF methods are combined to create the proposed ensemble method. Our study results show that the proposed KDR achieved 99.7% accuracy in 12.99 s of computations. The hyperparameter tuning and k-fold cross-validation are applied to validate the performance. Study results show that the proposed approach achieved high-performance scores for botnet attack detection compared to the state-of-the-art studies.

In the future, data analysis and machine learning algorithms will continue to evolve, leading to further improvements in accuracy and performance. One key focus area will be developing more advanced data balancing and feature selection techniques to improve the quality of input data used for machine learning. These techniques will involve analyzing and processing large datasets to identify imbalances or inconsistencies and then using statistical methods to correct these issues.

Another key development area will be using transfer learning-based advanced neural networks. These networks will be designed to learn from existing knowledge and adapt it to new tasks and scenarios. This will enable machines to effectively process and analyze complex data,

such as images and natural language, and make more accurate predictions and decisions based on this information. These advanced neural networks will be built using state-of-the-art hardware and software tools, allowing for faster and more efficient data processing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Ethical approval:

This article does not contain any studies with human participants or animals performed by any of the authors.

Funding

Not Applicable

References

[1] I. Debicha, B. Cochez, T. Kenaza, T. Debatty, J.-M. Dricot, W. Mees, Adv-bot: Realistic adversarial botnet attacks against network intrusion detection systems, Comput. Secur. (2023) 103176.

[2] F. Gul, I. Mir, L. Abualigah, P. Sumari, A. Forestiero, A consolidated review of path planning and optimization techniques: Technical perspectives and future directions, Electronics 10 (18) (2021) 2250.

[3] L. Abualigah, D. Falcone, A. Forestiero, et al., Swarm intelligence to face IoT challenges, Comput. Intell. Neurosci. 2023 (2023).

[4] P. Kumari, A.K. Jain, A comprehensive study of ddos attacks over IoT network and their countermeasures, Comput. Secur. (2023) 103096.

[5] J. Fang, Security evaluation method of distance education network nodes based on machine learning, in: International Conference on Machine Learning for Cyber Security, Springer, 2023, pp. 281–295.

[6] K. He, D.D. Kim, M.R. Asghar, Adversarial machine learning for network intrusion detection systems: A comprehensive survey, IEEE Commun. Surv. Tutor. (2023).

[7] A. Raza, H.U.R. Siddiqui, K. Munir, M. Almutairi, F. Rustam, I. Ashraf, Ensemble learning-based feature engineering to analyze maternal health during pregnancy and health risk prediction, PLoS One 17 (11) (2022) e0276525.

[8] A. Raza, K. Munir, M. Almutairi, F. Younas, M.M.S. Fareed, Predicting employee attrition using machine learning approaches, Appl. Sci. 12 (13) (2022) http://dx.doi.org/10.3390/app12136424, URL https://www.mdpi.com/2076-3417/12/13/6424.

[9] W.N.H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R. González Crespo, E. Herrera-Viedma, H. Fujita, Multilayer framework for botnet detection using machine learning algorithms, IEEE Access 9 (2021) 48753–48768, http://dx.doi.org/10.1109/ACCESS.2021.3060778.

[10] R.F.M. Dollah, M. Faizal, F. Arif, M.Z. Mas'ud, L.K. Xin, Machine learning for HTTP botnet detection using classifier algorithms, J. Telecommun. Electron. Comput. Eng. (JTEC) 10 (1–7) (2018) 27–30.

[11] S. Lee, A. Abdullah, N. Jhanjhi, S. Kok, Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning, PeerJ Comput. Sci. 7 (2021) e350.

[12] R.U. Khan, X. Zhang, R. Kumar, A. Sharif, N.A. Golilarz, M. Alazab, An adaptive multi-layer botnet detection technique using machine learning classifiers, Appl. Sci. 9 (11) (2019) http://dx.doi.org/10.3390/app9112375, URL https://www.mdpi.com/2076-3417/9/11/2375.

[13] H. Alkahtani, T.H. Aldhyani, Botnet attack detection by using CNN-LSTM model for Internet of Things applications, Secur. Commun. Netw. 2021 (2021) 1–23.

[14] K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, S. Sakib, et al., Botnet attack detection in IoT using machine learning, Comput. Intell. Neurosci. 2022 (2022).

[15] C.A. Fadhilla, M.D. Alfikri, R. Kaliski, Lightweight meta-learning BotNet attack detection, IEEE Internet Things J. (2022) 1, http://dx.doi.org/10.1109/JIOT.2022.3229463.

[16] O. Habibi, M. Chemmakha, M. Lazaar, Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT botnet attacks detection, Eng. Appl. Artif. Intell. 118 (2023) 105669, http://dx.doi.org/10.1016/j.engappai.2022.105669, URL https://www.sciencedirect.com/science/article/pii/S0952197622006595.

[17] F. Rustam, A. Raza, I. Ashraf, A.D. Jurcut, Deep ensemble-based efficient framework for network attack detection, in: 2023 21st Mediterranean Communication and Computer Networking Conference, MedComNet, 2023, pp. 1–10, http://dx.doi.org/10.1109/MedComNet58619.2023.10168864.

[18] B. Bojarajulu, S. Tanwar, T.P. Singh, Intelligent IoT-BOTNET attack detection model with optimized hybrid classification model, Comput. Secur. 126 (2023) 103064, http://dx.doi.org/10.1016/j.cose.2022.103064, URL https://www.sciencedirect.com/science/article/pii/S0167404822004564.

[19] S. García, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Comput. Secur. 45 (2014) 100–123, http://dx.doi.org/10.1016/j.cose.2014.05.011, URL https://www.sciencedirect.com/science/article/pii/S0167404814000923.

[20] G. Lucky, F. Jjunju, A. Marshall, A lightweight decision-tree algorithm for detecting ddos flooding attacks, in: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion, QRS-C, IEEE, 2020, pp. 382–389.

[21] A.H. Gandomi, F. Chen, L. Abualigah, Machine learning technologies for big data analytics, Electronics 11 (3) (2022) 421.

[22] S.B. Kotsiantis, Decision trees: a recent overview, Artif. Intell. Rev. 39 (2013) 261–283.

[23] M.G. Karthik, M.M. Krishnan, Hybrid random forest and synthetic minority over sampling technique for detecting internet of things attacks, J. Ambient Intell. Humaniz. Comput. (2021) 1–11.

[24] H. Al-Manaseer, L. Abualigah, A.R. Alsoud, R.A. Zitar, A.E. Ezugwu, H. Jia, A novel big data classification technique for healthcare application using support vector machine, random forest and J48, in: Classification Applications with Deep Learning and Machine Learning Technologies, Springer, 2022, pp. 205–215.

[25] J.L. Speiser, M.E. Miller, J. Tooze, E. Ip, A comparison of random forest variable selection methods for classification prediction modeling, Expert Syst. Appl. 134 (2019) 93–101.

[26] T.T. Khoei, S. Ismail, N. Kaabouch, Boosting-based models with tree-structured parzen estimator optimization to detect intrusion attacks on smart grid, in: 2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, UEMCON, IEEE, 2021, pp. 0165–0170.

[27] A.I. Alaiad, E.A. Mugdadi, I.I. Hmeidi, N. Obeidat, L. Abualigah, Predicting the severity of COVID-19 from lung CT images using novel deep learning, J. Med. Biol. Eng. (2023) 1–12.

[28] J. Heng, C. Wang, X. Zhao, L. Xiao, Research and application based on adaptive boosting strategy and modified CGFPA algorithm: a case study for wind speed forecasting, Sustainability 8 (3) (2016) 235.

[29] A. Raza, F. Rustam, H.U.R. Siddiqui, I.d.l.T. Diez, B. Garcia-Zapirain, E. Lee, I. Ashraf, Predicting genetic disorder and types of disorder using chain classifier approach, Genes 14 (1) (2023) 71.

[30] L. Abualigah, Classification Applications with Deep Learning and Machine Learning Technologies, Vol. 1071, Springer Nature, 2022.

[31] C. Bentéjac, A. Csörgő, G. Martínez-Muñoz, A comparative analysis of gradient boosting algorithms, Artif. Intell. Rev. 54 (2021) 1937–1967.

[32] E. Besharati, M. Naderan, E. Namjoo, LR-HIDS: logistic regression host-based intrusion detection system for cloud environments, J. Ambient Intell. Humaniz. Comput. 10 (2019) 3669–3692.

[33] M. Jamei, M. Karbasi, M. Mosharaf-Dehkordi, I.A. Olumegbon, L. Abualigah, Z. Said, A. Asadi, Estimating the density of hybrid nanofluids for thermal energy application: Application of non-parametric and evolutionary polynomial regression data-intelligent techniques, Measurement 189 (2022) 110524.

[34] C.-Y.J. Peng, K.L. Lee, G.M. Ingersoll, An introduction to logistic regression analysis and reporting, J. Educ. Comput. Res. 96 (1) (2002) 3–14.

[35] A. Ahmed, V.V. Krishnan, S.A. Foroutan, M. Touhiduzzaman, C. Rublein, A. Srivastava, Y. Wu, A. Hahn, S. Suresh, Cyber physical security analytics for anomalies in transmission protection systems, IEEE Trans. Ind. Appl. 55 (6) (2019) 6313–6323.

[36] C. Peng, Q. Cheng, Discriminative ridge machine: A classifier for high-dimensional data or imbalanced data, IEEE Trans. Neural Netw. Learn. Syst. 32 (6) (2020) 2595–2609.

[37] N. Peppes, E. Daskalakis, T. Alexakis, E. Adamopoulou, K. Demestichas, Performance of machine learning-based multi-model voting ensemble methods for network threat detection in agriculture 4.0, Sensors 21 (22) (2021) 7475.

[38] L. Bottou, Stochastic gradient descent tricks, in: Neural Networks: Tricks of the Trade, second ed., Springer, 2012, pp. 421–436.

[39] O.D. Adeniji, D.B. Adekeye, S.A. Ajagbe, A.O. Adesina, Y.J. Oguns, M.A. Oladipupo, Development of DDoS attack detection approach in software defined network using support vector machine classifier, in: Pervasive Computing and Social Networking: Proceedings of ICPCSN 2022, Springer, 2022, pp. 319–331.

[40] K. Lau, Q. Wu, Online training of support vector classifier, Pattern Recognit. 36 (8) (2003) 1913–1920.

[41] R.S. Moorthy, P. Pabitha, Optimal detection of phising attack using SCA based K-NN, Procedia Comput. Sci. 171 (2020) 1716–1725.

[42] Y. Liao, V.R. Vemuri, Use of k-nearest neighbor classifier for intrusion detection, Comput. Secur. 21 (5) (2002) 439–448.

[43] A.H. Jahromi, M. Taheri, A non-parametric mixture of Gaussian naive Bayes classifiers based on local independent features, in: 2017 Artificial Intelligence and Signal Processing Conference, AISP, IEEE, 2017, pp. 209–212.

[44] R. Islam, M.K. Devnath, M.D. Samad, S.M.J. Al Kadry, GGNB: Graph-based Gaussian naive Bayes intrusion detection system for CAN bus, Veh. Commun. 33 (2022) 100442.

[45] A. Raza, K. Munir, M. Almutairi, F. Younas, M.M.S. Fareed, G. Ahmed, A novel approach to classify telescopic sensors data using bidirectional-gated recurrent neural networks, Appl. Sci. 12 (20) (2022) 10268.

[46] Y.N. Kunang, S. Nurmaini, D. Stiawan, B.Y. Suprapto, Attack classification of an intrusion detection system using deep learning and hyperparameter optimization, J. Inf. Secur. Appl. 58 (2021) 102804.

[47] A. Raza, F. Rustam, H.U. Siddiqui, I.d. Diez, I. Ashraf, Predicting microbe organisms using data of living micro forms of life and hybrid microbes classifier, PLoS One 18 (4) (2023) http://dx.doi.org/10.1371/journal.pone.0284522.

[48] J. Yadav, J. Thakur, BotEye: Botnet detection technique via traffic flow analysis using machine learning classifiers, in: 2020 Sixth International Conference on Parallel, Distributed and Grid Computing, PDGC, 2020, pp. 154–159, http://dx.doi.org/10.1109/PDGC50313.2020.9315792.

[49] R. Chen, W. Niu, X. Zhang, Z. Zhuo, F. Lv, An effective conversation-based botnet detection method, Math. Probl. Eng. 2017 (2017).

[50] P. Amini, R. Azmi, M.A. Araghizadeh, Analysis of network traffic flows for centralized botnet detection, J. Telecommun. Electron. Comput. Eng. (JTEC) 11 (2) (2019) 7–17.

[51] S. Jagadeesan, B. Amutha, An efficient botnet detection with the enhanced support vector neural network, Measurement 176 (2021) 109140, http://dx.doi.org/10.1016/j.measurement.2021.109140, URL https://www.sciencedirect.com/science/article/pii/S0263224121001664.

[52] W.A. Safitri, T. Ahmad, D.P. Hostiadi, Analyzing machine learning-based feature selection for botnet detection, in: 2022 1st International Conference on Information System & Information Technology, ICISIT, 2022, pp. 386–391, http://dx.doi.org/10.1109/ICISIT54091.2022.9872812.

[53] R.S.S. Moorthy, N. Nathiya, Botnet detection using artificial intelligence, Procedia Comput. Sci. 218 (2023) 1405–1413, http://dx.doi.org/10.1016/j.procs.2023.01.119, International Conference on Machine Learning and Data Engineering, URL https://www.sciencedirect.com/science/article/pii/S1877050923001199.

[54] D. Gong, Y. Liu, A mechine learning approach for botnet detection using lightgbm, in: 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications, CVIDL & ICCEA, 2022, pp. 829–833, http://dx.doi.org/10.1109/CVIDLICCEA56201.2022.9824033.

[55] S. Ahmed, Z.A. Khan, S.M. Mohsin, S. Latif, S. Aslam, H. Mujlid, M. Adil, Z. Najam, Effective and efficient DDoS attack detection using deep learning algorithm, multi-layer perceptron, Future Internet 15 (2) (2023) http://dx.doi.org/10.3390/fi15020076, URL https://www.mdpi.com/1999-5903/15/2/76.