Turkish Journal of Engineering

https://dergipark.org.tr/en/pub/tuje

e-ISSN 2587-1366

A Novel Hyperparameter Tuning Method for Enhanced Intrusion Detection in Network Security

Vahid Sinap[*1]

[1]*Ufuk University, Department of Management Information Systems, Türkiye, vahidsinap@gmail.com*

Cite this study: Sinap, V. (2025). A novel hyperparameter tuning method for intrusion detection. Turkish Journal of Engineering, 9(3), 519-534.

https://doi.org/10.31127/tuje.1624366

Keywords
Real-Time Attack Identification
Intrusion Detection Systems
Cybersecurity
Network Security
Adaptive Hyperparameter Optimization

Research Article

Received:21.01.2025
Revised:07.03.2025
Accepted:09.03.2025
Published:01.07.2025

Abstract

Intrusion Detection Systems (IDS) are essential for ensuring the security of enterprise networks and cloud-based systems, as they defend against sophisticated and evolving cyberattacks. Machine learning (ML) techniques have emerged as effective tools to enhance IDS performance, addressing the limitations of traditional methods. This study proposes a novel hyperparameter tuning method for ML-based IDS, leveraging the NSL-KDD dataset with extensive feature selection and preprocessing to address data imbalance and redundancy. The method, integrating adaptive refinement with stochastic perturbation, optimizes classifiers such as Random Forest (RF), Gradient Boosting (GB), and Extreme Gradient Boosting (XGB), achieving both higher detection accuracy (99.90% with RF) and improved computational efficiency. This approach excels due to its dynamic adjustment of parameter ranges and controlled randomness, converging faster than traditional Grid Search and Random Search by reducing iterations by up to 87.5%. The experimental results demonstrate that tree-based models, particularly RF, outperform others due to their ability to model complex, non-linear patterns, enhanced by the proposed tuning method. Measured in terms of convergence speed, CPU time, and memory usage, this method proves suitable for deployment in real-time, resource-constrained environments, offering a scalable and efficient solution for network security.

1. Introduction

Cybersecurity has become a critical area of research and practice with the digital transformation of modern society. Increasing digitization in almost all sectors, from banking to energy infrastructures, healthcare to government agencies, has expanded both the scope and potential impact of cyberattacks. The rapid proliferation of threats to digital infrastructures, coupled with the ever-evolving tactics and techniques of attackers, has made the protection of these systems a vital priority [1]. In this context, effective IDS are indispensable tools for the security of modern networks. By monitoring network traffic and system activity, IDSs aim to detect potential threats and security breaches and intervene as early as possible [2]. However, the current state of these systems faces significant limitations in dealing with increasing cyber threats.

Traditional IDS approaches rely heavily on signature-based and anomaly-based methods. Signature-based methods store known patterns of attacks in databases and detect activity that fits these patterns [3]. Although these methods are generally effective with low false positive rates, they can only identify previously documented attacks and fail to detect zero-day attacks [4]. Anomaly-based methods, on the other hand, learn normal network behavior and treat deviations from this behavior as a potential threat. However, the high false positive rates seen in these methods negatively affect system performance and lead to problems such as false alarm fatigue [5].

The rapid evolution of cyber-attacks in today's environment presents IDSs with a more complex security

problem. Diversifying attack types increase the scope and sophistication of attacks and exceed the scope of traditional methods. For example, distributed denial of service (DDoS) attacks, chained attacks, insider threats, and advanced persistent threats (APT) not only require high detection accuracy but also require timely detection. In addition, the sheer size and heterogeneity of modern networks greatly limits the performance of IDS systems in terms of data volume, speed and variety [6]. In addition, ever-changing attack tactics cause existing models to rapidly lose their effectiveness if they lack adaptive learning capabilities [7]. Moreover, the need for real-time intrusion detection in networks brings performance and latency issues. An IDS needs not only to accurately identify attacks, but also to do so instantaneously without negatively impacting network performance [8]. However, many traditional methods without real-time processing capabilities prevent the timely detection of attacks in high-throughput networks and lead to the growth of potential damage [9]. All these factors clearly demonstrate the limitations of current IDS in cybersecurity. Therefore, IDS systems not only need to become more accurate and faster, but also adaptive, scalable and resource efficient. In this context, machine learning and related techniques stand out as an important tool to overcome these challenges and make IDS systems more resilient.

ML has created a paradigm shift in cybersecurity and opened up new opportunities for IDS. ML algorithms, with their ability to learn patterns in high-dimensional and complex datasets, provide significant advantages in detecting sophisticated threats such as zero-day attacks, DDoS and APT [10]. By analyzing network traffic or user behavior, these algorithms can identify malicious activity with high accuracy. However, the use of ML in IDS brings with it the potential to quickly adapt not only to known threats, but also to threats that may emerge in the future.

While ML-based IDS offers more effective and adaptive solutions than traditional methods, there are still significant challenges in this area. Factors such as unstable datasets, real-time processing requirements, and the constant change of attack vectors complicate the design and applicability of these solutions. Attacks are rare events, often representing a very small fraction of large datasets [11]. This hinders the balanced training process needed for high performance of the models and often leads to high false positive rates by incorrectly recognizing normal traffic as a threat. Such false alarms can cause alert fatigue for network administrators, weakening security [12]. On the other hand, high false negative rates can lead to major security vulnerabilities as a result of missed attacks. In addition, another major challenge in cyber security is the diversification and constant change in attack vectors [13]. Traditional or static models can quickly become ineffective against these new forms of attacks. Therefore, adaptive learning capabilities of ML-based IDSs should be developed. With the ability to learn from new types of data over time, adaptive methods can become more resilient to the changing strategies of attackers. However, this adaptation process must be carefully managed to avoid overfitting models or making them more sensitive to variability than necessary [14]. Hyperparameter tuning of ML models is also an important challenge [15]. As critical elements that optimize the model's learning capacity and performance, hyperparameters directly affect model accuracy, overall performance and generalization ability [16]. However, identifying the right hyperparameters, especially in complex datasets, is a complex and time-consuming process [17]. Traditional search methods often face excessive computational costs and low optimization efficiency because they operate over a large parameter space [18]. Therefore, the development of an efficient hyperparameter optimization process is a critical requirement for building faster and more accurate ML-based IDSs. All these factors highlight the importance of feature extraction, adaptive learning capabilities, efficient hyperparameter optimization methods, and comprehensive evaluation frameworks. A successful IDS in cybersecurity requires an approach that is resilient not only against current threats, but also against unknown future threats. This requires the combination of multiple disciplines, from data analytics to algorithm engineering, and the continuous innovative development of solution processes.

The primary aim of this research is to enhance intrusion detection capabilities by utilizing ML classifiers, including Logistic Regression (LR), RF, GB, Decision Tree (DT), Naive Bayes (NB), and XGB. Central to this goal is the development of a custom hyperparameter tuning method that not only optimizes the detection accuracy of these classifiers but also significantly improves computational efficiency, scalability, and adaptability to evolving cyber threats. This method has been extensively evaluated against traditional approaches such as grid search and random search in terms of accuracy, resource utilization and time efficiency. By addressing key limitations in both hyperparameter optimization and intrusion detection, this study contributes to the advancement of strong, efficient, and adaptive security solutions that are well-suited for modern, resource-constrained environments.

2. Related research

The rapid advancement of technology, especially in cloud computing, has led to a significant increase in digital information and, consequently, network intrusions. This growth necessitates the development of effective network IDS to safeguard sensitive data and ensure the integrity of digital infrastructures. Research in this field primarily focuses on leveraging benchmark datasets and implementing ML techniques to improve the accuracy and efficiency of detecting malicious activities within networks. This section summarizes recent studies that have contributed to this domain, emphasizing their objectives, methodologies, and key findings.

Vibhute et al. [19] explored the development of a NIDS using the benchmark NSL-KDD dataset. Their primary objective was to enhance feature selection and improve classification accuracy. The study employed ensemble learning with an RF algorithm to identify optimal features. For the detection and classification of network intrusions, they utilized three ML models:

Support Vector Machine (SVM), LR, and K-Nearest Neighbors (KNN). Among these, KNN achieved the highest validation accuracy of 98.24%, followed by LR at 88.86% and SVM at 87.58%.

Barach [20] proposes a novel approach to network IDS by combining channel attention with Convolutional Neural Networks (CNN) to enhance the detection of anomalies. Using the NSL-KDD dataset, which includes 43 features with labels "attack" and "level," the study introduces a method that achieves an impressive accuracy rate of 99.728%. The CNN-based model, integrated with channel attention, significantly outperforms previous methods such as ensemble learning, CNN, RBM, ANN, hybrid auto-encoders, MCNN, and adaptive algorithms, demonstrating a substantial improvement in intrusion detection.

Abdullah [21] investigates the effectiveness of various ML algorithms for cyber-attack detection using the NSL-KDD dataset. The study compares RNN, MLP, CNN-LSTM, and ANN in terms of classification accuracy. The results show that RNN and MLP outperform the other models, with RNN achieving an accuracy of 0.9980.

Zakariah et al. [22] proposed a novel IDS utilizing Artificial Neural Networks (ANNs) to address the challenges posed by intrusive network traffic. Using the NSL-KDD dataset for both training and testing, they developed a custom ANN architecture with optimized nodes, layers, and activation functions, designed to capture intricate patterns in network data. The model achieved a detection accuracy of 97.5% and demonstrated superior generalizability on untried data.

Türk [23] conducted a study focused on intrusion detection using the UNSW-NB15 and NSL-KDD datasets. The aim was to implement and evaluate ML and deep learning algorithms for both binary and multi-class classification. The study employed RF and Multi-Layer Perceptron (MLP) as the key classifiers. For the UNSW-NB15 dataset, accuracies of 98.6% and 98.3% were achieved for binary and multi-class classification, respectively. Similarly, in the NSL-KDD dataset, 97.8% and 93.4% accuracies were obtained for binary and multi-class classification, respectively.

Rastogi et al. [24] investigated the performance of various supervised learning algorithms for intrusion detection using the NSL-KDD dataset. The study aimed to enhance detection accuracy and evaluate model efficiency in terms of accuracy and runtime. Among the tested models, RF achieved the best results, with an accuracy of 98.47%, a training time of 3.56 seconds, and a testing time of 0.16 seconds. This demonstrates RF's effectiveness in accurately and efficiently detecting network intrusions.

Ravipati and Abualkibash [25] addressed the limitations of anomaly-based IDS, specifically high false alarm rates and moderate detection accuracy. They evaluated various ML algorithms on the KDD-99 Cup and NSL-KDD datasets. The RF algorithm emerged as the best-performing model, achieving an exceptional accuracy of 99.7% with a reasonable false alarm rate.

Shrivas and Dewangan [26] propose an ensemble technique combining Artificial Neural Network (ANN) and Bayesian Net with Gain Ratio (GR) feature selection for IDS. The study aims to enhance the performance of intrusion detection by addressing the irrelevant features in datasets. They apply individual classification techniques as well as the proposed ensemble model on the KDD99 and NSL-KDD datasets to evaluate the robustness of the models. The inclusion of the GR feature selection technique further refines the model's performance, resulting in the highest accuracy of 97.76% compared to other methods.

This research sets itself apart by introducing a custom hyperparameter tuning method designed to optimize the performance of multiple ML classifiers for intrusion detection. While existing studies, such as those by Vibhute et al. [19] and Shrivas and Dewangan [26], have successfully demonstrated the effectiveness of various ML algorithms, ensemble methods, and feature selection techniques, this study extends beyond conventional classifier evaluation. By integrating a novel and highly adaptive hyperparameter optimization approach, it offers a significant improvement over traditional techniques like grid search and random search. This enhanced tuning strategy ensures more efficient model performance and enables the development of stronger, dynamic IDS capable of adapting to evolving cyber threats, making a substantial contribution to the field.

3. Method

In this section, an overview of the dataset used, the data preparation process, the classification algorithms employed, the cross-validation (CV) and feature selection techniques, and the performance metrics for evaluating the models are described. The goal is to lay the methodological and analytical foundation for the research, ensure its reproducibility, and enhance the scientific contribution of the study. Detailed descriptions of the datasets, algorithms, and experimental setups are provided, offering a comprehensive overview of the methods used to optimize and compare the classifiers in detecting intrusion activities.

3.1. Dataset

The NSL-KDD dataset was utilized as the primary benchmark for evaluating the proposed intrusion detection methodology. This dataset, a refined version of the KDD'99 dataset, addresses several limitations of its predecessor, including the removal of redundant and duplicate records, ensuring a more balanced and reliable dataset [27]. It consists of 43 features that capture diverse network traffic characteristics, enabling a comprehensive analysis of network behavior. The dataset comprises a total of 125,972 instances, all of which are included in the training set. These instances are categorized into normal and attack classes, with the attack types further divided into four major categories: DoS, Probe, Remote-to-Local (R2L), and User-to-Root (U2R). The diversity of attack types within the training data ensures a comprehensive foundation for the development and evaluation of the proposed model, allowing for the assessment of its ability to detect various intrusion patterns effectively.

The dataset consists of 43 features that can be broadly categorized into three main groups: basic features, content features, and traffic features. Basic features

capture fundamental attributes of a network connection and include characteristics such as duration, which represents the length of a connection, protocol_type, indicating the protocol used (e.g., TCP, UDP), service, specifying the network service on the destination (e.g., HTTP, FTP), and flag, which identifies the status of the connection. Additional features such as src_bytes and dst_bytes describe the amount of data transferred between source and destination, while land is a binary feature indicating whether the connection originates and terminates at the same host. Content features delve into the specifics of the data payload within network packets, often highlighting signs of malicious behavior. These include num_failed_logins, which tracks unsuccessful login attempts, root_shell, a binary feature indicating whether root access was obtained, and hot, which counts occurrences of suspicious activities such as creating executables. Other significant features in this category include num_compromised, which counts the number of compromised conditions, and su_attempted, indicating attempts to use the super-user command. Traffic features describe the statistical properties of connections observed over a specified time window. These features include count, the number of connections to the same host as the current connection, srv_count, representing connections to the same service, and same_srv_rate, the percentage of connections to the same service. Additional traffic-related features, such as dst_host_count and dst_host_srv_count, count the total number of connections and those to the same service on the destination host, respectively. Features like diff_srv_rate and srv_diff_host_rate provide metrics on the diversity of services and hosts, while serror_rate and srv_serror_rate measure the rate of connections with SYN errors. Similarly, rerror_rate and srv_rerror_rate track the rate of connections with REJ errors. The dataset also includes advanced traffic metrics such as dst_host_same_srv_rate, representing the percentage of connections to the same service on a destination host, and dst_host_srv_diff_host_rate, which measures how often a service is accessed by different hosts.

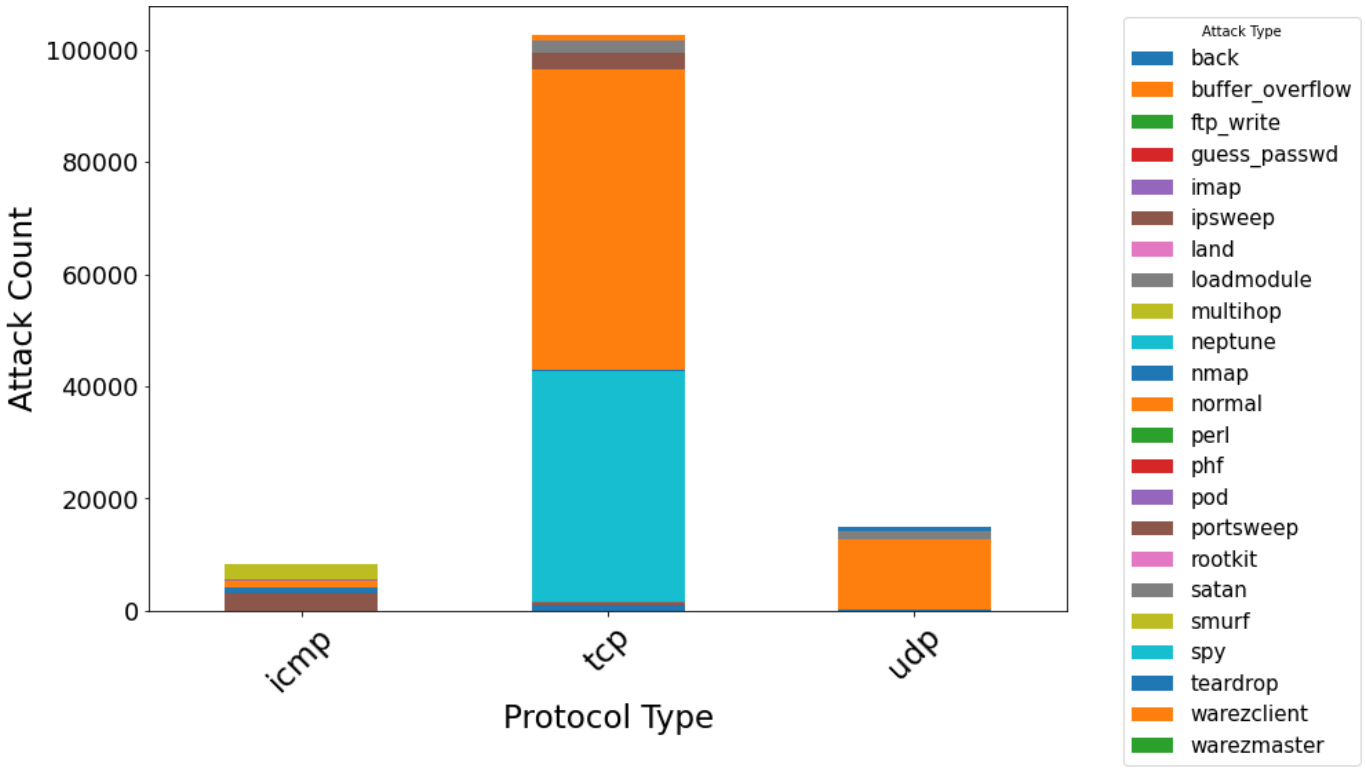

Figure 1. Attack counts over protocol types

The bar chart presented in Figure 1 shows the distribution of various attack types across ICMP, TCP and UDP protocol types. Each color in the chart corresponds to a specific attack type, as detailed in the accompanying legend. The figure highlights that TCP-based attacks are significantly more prevalent compared to attacks on UDP and ICMP protocols. A large proportion of attacks on the TCP protocol include categories such as "normal," "smurf," and "Neptune," whereas ICMP attacks are primarily dominated by "smurf." In contrast, UDP shows a lower volume of attacks, with a more balanced distribution across multiple categories.

3.2. Data preparation

The dataset underwent a thorough preprocessing phase to ensure its readiness for analysis and modeling. As part of this process, the data was carefully examined for quality and consistency. Initially, the dataset was inspected for significant outliers. Since no notable outliers were detected, further outlier treatment was deemed unnecessary. Additionally, a detailed analysis confirmed the absence of duplicate records and missing values, ensuring that the data was both complete and reliable for subsequent tasks. To handle categorical variables effectively, the target feature "attack" was transformed into a numerical format using a label encoding approach. This method systematically converted categorical attack types into corresponding

numerical representations, enabling the dataset to be utilized efficiently by ML algorithms. Furthermore, the dataset was examined for class balance, and it was determined to be largely balanced, with 53.5% of the records classified as "normal" and 46.5% as "attack."

3.3. Classification algorithms

In this study, six ML algorithms are employed for intrusion detection on the NSL-KDD dataset: LR, RF, GB, DT, NB, and XGB. Each algorithm is selected for its unique strengths in handling classification tasks within the context of cybersecurity. Below, the mathematical foundations and key characteristics of these algorithms are described.

LR is a linear model used for binary and multi-class classification [28]. It estimates the probability of a class using the logistic function, as shown in Equation 1:

$$P(y = 1|\mathrm{x}) = \frac{1}{1 + e^{-(\mathrm{w}^T\mathrm{x}+b)}} \quad (1)$$

where w is the weight vector, x is the feature vector, and b is the bias term. The model is trained by minimizing the log-loss function, defined in Equation 2:

$$L(\mathrm{w}, b) = -\sum_{i=1}^{n}\big[y_i \log\big(P(y_i = 1|\mathrm{x}_i)\big) + \big(1 - y_i \log\big(1 - P(y_i = 1|\mathrm{x}_i)\big)\big)\big] \quad (2)$$

RF is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees [29]. Each tree is trained on a bootstrap sample of the data, and at each split, a random subset of features is considered [30]. The final prediction is given by Equation 3:

$$\hat{y} = \mathrm{mode}(\{T_1(\mathrm{x}), T_2(\mathrm{x}), \ldots, T_m(\mathrm{x})\}) \quad (3)$$

where $T_i(\mathrm{x})$ is the prediction of the i^{th} tree.

GB is another ensemble technique that builds trees sequentially, where each tree corrects the errors of the previous one. The model is trained by minimizing a loss function (e.g., log-loss) using gradient descent [31]. The prediction for an instance x is given by Equation 4:

$$\hat{y} = \sum_{i=1}^{m} \gamma_i T_i(\mathrm{x}) \quad (4)$$

where γ_i is the weight of the i^{th} tree. GB is known for its high predictive accuracy and ability to handle complex data patterns.

DT is a non-parametric model that splits the data into subsets based on feature values, creating a tree-like structure. The splits are chosen to maximize information gain or minimize Gini impurity [32]. For a feature vector x, the prediction is obtained by traversing the tree from the root to a leaf node. The Gini impurity for a node is defined in Equation 5:

$$\mathrm{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2 \quad (5)$$

where p_i is the proportion of class i in the node.

NB is a probabilistic classifier based on Bayes' theorem, with the "naive" assumption of feature independence given the class [33]. The posterior probability of a class y given a feature vector x is calculated as shown in Equation 6:

$$P(y|\mathrm{x}) = \frac{P(y)\prod_{j=1}^{d} P(x_j|y)}{P(\mathrm{x})} \quad (6)$$

where $P(y)$ is the prior probability of class y, and $P(x_j|y)$ is the likelihood of feature x_j given class y.

XGB is an optimized implementation of GB that incorporates regularization to prevent overfitting [34]. The objective function includes both the loss function and regularization terms, as defined in Equation 7:

$$L(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{m} \Omega(T_k) \quad (7)$$

where $l(y_i, \hat{y}_i)$ is the loss function, and $\Omega(T_k)$ is the regularization term for the k^{th} tree.

3.4. Cross validation

In this study, K-fold CV technique was employed to evaluate the performance of the model. The dataset was randomly partitioned into $k = 10$ qually sized subsets, referred to as "folds." During each iteration, one-fold was designated as the validation set, while the remaining $k - 1$ folds were utilized as the training set. This process was repeated for k iterations, ensuring that each fold served as the validation set exactly once. The model's performance metrics, such as accuracy or F1-score, were computed for each iteration. Subsequently, the average of these metrics across all k iterations was taken as the final estimate of the model's performance. This approach provides a stronger and reliable evaluation by mitigating the risks of overfitting and ensuring that every data point in the dataset is included in the validation process at least once [35]. The formula used for computing the average performance is given in Equation 8:

$$P = \frac{1}{k}\sum_{i=1}^{k} P_i \quad (8)$$

where P represents the final performance metric, k is the number of folds, and P_i denotes the performance metric obtained in the i^{th} iteration. By leveraging 10-fold CV, the inherent variability in model evaluation due to data partitioning was reduced, leading to a more generalizable assessment of the model's predictive capabilities. The methodology ensures that the model's evaluation is comprehensive and less biased toward any specific subset of the data [36].

3.5. Feature selection

In this study, Recursive Feature Elimination (RFE), a widely used feature selection method, is employed to identify the most relevant features from the NSL-KDD dataset. RFE is an iterative technique that recursively removes the least important features based on the model's performance, ultimately retaining the most discriminative features for intrusion detection [37].

The RFE process begins by training an ML model (e.g., a classifier) on the entire set of features. The importance of each feature is then evaluated, typically using coefficients from linear models or feature importance scores from tree-based models. The least important feature(s) are removed, and the model is retrained on the remaining features. This process is repeated iteratively until a predefined number of features is reached or until further feature removal degrades the model's performance.

The RFE method is implemented through a structured process that begins with training a ML model, such as LR or RF, on the full set of features. Once the model is trained, features are ranked based on their importance scores. For instance, in LR, the absolute values of the coefficients are used to determine feature importance, while in RF, the importance scores are derived from metrics like Gini impurity or information gain. After ranking, the least important feature(s) are eliminated from the dataset. The model is then retrained on the reduced feature set, and this iterative process of ranking, elimination, and retraining continues until a predefined number of features is achieved or until further feature removal negatively impacts the model's performance. This approach ensures that only the most relevant and discriminative features are retained, enhancing both the efficiency and effectiveness of the model.

In this study, RFE is applied using an RF classifier due to its ability to provide robust feature importance scores. The number of features to retain is determined through CV, ensuring that the selected features contribute to optimal model performance. The final set of features obtained through RFE is used to train and evaluate the ML classifiers, as described in the Model Setups section. As illustrated in Figure 2, the most significant features for intrusion detection include src_bytes, same_srv_rate, and dst_host_srv_serror_rate, which collectively play a crucial role in distinguishing between normal and malicious network activities.

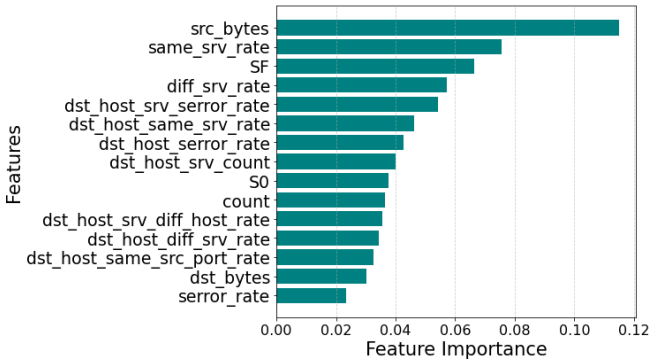

Figure 2. Distribution of the top 15 feature importance values

3.6. Performance metrics

In this study, several performance metrics are employed to evaluate the effectiveness of the proposed intrusion detection models. These metrics provide a comprehensive assessment of the models' ability to correctly classify normal and attack instances. The following metrics are used: accuracy, precision, recall, F1 score, and the Area Under the Curve (AUC). Each metric is defined and calculated as follows:

Accuracy: Accuracy measures the proportion of correctly classified instances out of the total number of instances [33]. Accuracy is provided in Equation 9.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

Precision: Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive [38]. The formula for precision is given in Equation 10.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

Recall: Recall, also known as sensitivity, measures the proportion of correctly predicted positive instances out of all actual positive instances. The calculation for recall is provided in Equation 11.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a single metric that balances both concerns [39]. The F1 Score is provided in Equation 12.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

Area Under the Curve (AUC): The AUC measures the entire two-dimensional area underneath the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate against the false positive rate at various threshold settings. The AUC provides an aggregate measure of performance across all possible classification thresholds. A higher AUC indicates better model performance.

3.7. Model setups

In this study, the NSL-KDD dataset is divided into two parts, with 75% allocated for training and 25% for testing. A random state of 42 is used to ensure reproducibility in the data splitting process. To determine the optimal hyperparameters for the ML classifiers, three hyperparameter optimization techniques are employed: Grid Search, Random Search, and the Custom Hyperparameter Tuning Method proposed in this study. Each of these techniques offers distinct advantages in exploring the hyperparameter space and identifying the best configurations for the models.

1. Grid Search: Grid Search is a traditional hyperparameter optimization technique that exhaustively searches through a predefined set of hyperparameter values. It evaluates all possible combinations of hyperparameters within the specified grid, ensuring that the optimal configuration is found. However, this method can be computationally expensive, especially for high-dimensional parameter spaces [16]. The Grid Search configuration in this study includes a predefined grid

of hyperparameters for each classifier, as shown in Table 1.

2. Random Search: Random Search, unlike Grid Search, randomly samples hyperparameter values from a specified distribution. This method is less computationally intensive than Grid Search, as it does not evaluate all possible combinations. Instead, it focuses on a random subset of the hyperparameter space, which can still yield competitive results [40-41]. The Random Search configuration in this study involves sampling hyperparameters from uniform distributions for each classifier, as detailed in Table 1.
3. Custom Hyperparameter Tuning Method: The proposed custom hyperparameter tuning method combines stochastic perturbation and adaptive refinement to efficiently converge on high-performing parameter configurations. This method iteratively refines the search space around the best-performing parameters and introduces stochastic perturbations to avoid premature convergence. The custom method is designed to balance exploration and exploitation, ensuring faster convergence to optimal solutions. The detailed steps of this method are described in the Proposed Hyperparameter Tuning Method section, and the final hyperparameter settings obtained through this method are presented in Table 1.

Table 1. Hyperparameter settings for models optimized using different techniques

Model	Hyperparameter	Grid Search Settings	Random Search Settings	Custom Method Settings
LR	C	0.12	0.12	0.12
	max_iter	200	200	215
	solver	'liblinear'	'liblinear'	'liblinear'
RF	n_estimators	100	237	187
	max_depth	15	17	13
	min_samples_split	5	9	7
	min_samples_leaf	2	2	3
GB	n_estimators	200	290	245
	learning_rate	0.1	0.09	0.042
	subsample	0.8	0.72	0.85
	max_features	'sqrt'	'log2'	'log2'
DT	max_depth	10	14	9
	min_samples_split	5	6	7
	min_samples_leaf	2	3	2
	splitter	'best'	'best'	'random'
NB	var_smoothing	1e-8	2.2e-8	3.7e-8
XGB	n_estimators	200	245	162
	max_depth	7	8	8
	learning_rate	0.1	0.08	0.07
	colsample_bytree	0.8	0.85	0.9

The data analysis and model testing are conducted using the Python programming language. For data processing and manipulation, the pandas and NumPy libraries are employed, providing robust tools for handling large datasets. The scikit-learn library is utilized for model development and evaluation, offering a comprehensive range of ML algorithms and performance metrics. All analyses are performed within the Jupyter Notebook environment, which integrates code, text, and visualizations seamlessly. The computational experiments are executed on a PC equipped with an AMD Ryzen 7800X3D processor, operating at 4.2 GHz, and an NVIDIA GeForce RTX 4070 Ti GPU, supported by 32 GB of 6000 MHz DDR5 RAM. The system runs on Windows 11, ensuring a stable and efficient development environment. By employing these optimization techniques and model setups, the study aims to identify the most effective hyperparameter configurations for each classifier, ultimately enhancing the performance of the IDS on the NSL-KDD dataset.

3.8. Proposed hyperparameter tuning method

The process of hyperparameter optimization is critical for the performance of ML models, as it determines the optimal configuration of parameters that govern the learning process. Traditional methods, such as Grid Search and Random Search, while widely used, suffer from inefficiencies and limitations. Grid Search is computationally expensive, requiring exhaustive evaluation of all possible combinations, whereas Random Search lacks a structured approach to focus on promising areas of the parameter space. These shortcomings become pronounced in high-dimensional parameter spaces or when computational resources are constrained. To address these issues, a novel hyperparameter tuning method was developed that combines stochastic perturbation and adaptive refinement, aiming to efficiently converge on high-performing parameter configurations.

The proposed method operates iteratively, incorporating three main components: a baseline grid search, adaptive refinement of the search space, and stochastic perturbations to avoid premature convergence. Each component is designed to address specific inefficiencies in existing methods.

Initial Grid Evaluation: The optimization process begins by evaluating a predefined parameter grid, P, encompassing the user-specified ranges for each hyperparameter. This step involves training the model on shuffled training data to eliminate bias due to data ordering. Shuffling is implemented using a fixed random state to maintain reproducibility:

$$\{X'_{\text{train}}, y'_{\text{train}}\} = \text{shuffle}(X_{\text{train}}, y_{\text{train}}, \text{random\_state} = 42) \quad (13)$$

For every parameter configuration, $\theta \in P$, CV is performed over k folds, and the average score is computed using the scoring metric selected by the user (e.g., weighted F1 score). The mean CV score for a given configuration is given by:

$$\text{Score}_{\text{avg}} = \frac{1}{k}\sum_{i=1}^{k} S_i(\theta) \quad (14)$$

As shown in Equation 14, this score serves as the criterion to evaluate and rank parameter configurations. The best-performing configuration, θ^*, is identified after the initial evaluation.

Adaptive Refinement: In subsequent iterations, the parameter grid is dynamically refined around θ^*, enabling the search to focus on promising regions of the parameter space. For continuous parameters, this is achieved by applying adaptive scaling:

$$\theta'_i \in \left[\max\left(\theta_i(1-\alpha), \min(P)\right), \min\left(\theta_i(1+\alpha), \max(P)\right)\right] \quad (15)$$

Here, α is a user-defined perturbation scale factor that controls the range of exploration around θ_i. Equation 15 ensures that the refined grid balances exploration of new values while maintaining proximity to the best-performing parameters. For categorical parameters, all original options are retained to preserve diversity in the search.

Stochastic Perturbations: To prevent the search from becoming trapped in local optima, stochastic perturbations are introduced at each iteration. These perturbations apply random adjustments to the refined parameter grid, ensuring variability in the sampled configurations. This mechanism enables the algorithm to explore near-optimal solutions and uncover potentially superior configurations overlooked in deterministic refinements.

Iterative Search: The process of adaptive refinement and stochastic perturbation is repeated over a predefined number of iterations, n. At each iteration, the refined grid, P', is evaluated, and the best-performing configuration is updated if it surpasses the previous best score:

$$\theta^* = \text{argmax}_{\theta \in P'} \text{Score}_{\text{avg}} \quad (16)$$

As shown in Equation 16, the algorithm continuously updates ∗ θ ∗ to reflect the highest performance achieved thus far.

Final Model Training: Upon completing all iterations, the best configuration, θ^*, is used to train the model on the full training dataset, ensuring that the optimal hyperparameters are applied holistically. The historical results, including parameter configurations and corresponding scores, are recorded to allow for performance analysis and reproducibility.

This novel method introduces significant advantages over conventional approaches. Adaptive refinement reduces computational overhead by narrowing the search space iteratively, while stochastic perturbations maintain exploration potential. The method achieves a balance between exploitation of promising regions and exploration of new configurations, enabling faster convergence to optimal solutions. Its scalability and efficiency make it suited for scenarios involving high-dimensional parameter spaces or limited computational resources.

4. Experimental study and findings

In this section, the experimental results of the proposed intrusion detection models are presented. The performance of the classifiers -LR, RF, GB, DT, NB, and XGB- is evaluated using three hyperparameter optimization techniques: Grid Search, Random Search, and the Custom Hyperparameter Tuning Method. To further validate the proposed method's robustness and applicability, additional experiments were conducted on the UNSW-NB15 and CIC-IDS2017 datasets, and a comparison with deep learning models CNN and LSTM) was performed. The results are summarized in the following tables, and the key findings are discussed.

Table 2. Confusion matrix for models optimized using grid search

			Predicted	
			Normal	Attack
Actual	LR	Normal	13673	3047
		Attack	1132	13641
	RF	Normal	16301	409
		Attack	204	14569
	GB	Normal	16029	691
		Attack	512	14261
	DT	Normal	16137	583
		Attack	210	14563
	NB	Normal	15739	981
		Attack	324	14449
	XGB	Normal	15722	998
		Attack	601	14172

Upon examining Table 2, the confusion matrix for models optimized using Grid Search reveals that RF achieved the lowest number of misclassifications, with only 409 normal instances misclassified as attacks and 204 attacks misclassified as normal. In contrast, LR exhibited the highest misclassification rates, with 3047 false positives and 1132 false negatives. This indicates that tree-based models such as RF and GB outperform linear models like LR when optimized using Grid Search.

Table 3. Performance metrics for models optimized using grid search

Model	Accuracy	Precision	Recall	F1 Score	AUC
LR	0.867	0.817	0.923	0.867	0.942
RF	0.980	0.972	0.986	0.979	0.993
GB	0.961	0.953	0.965	0.959	0.994
DT	0.974	0.961	0.985	0.973	0.987
NB	0.958	0.936	0.978	0.956	0.978
XGB	0.949	0.934	0.959	0.946	0.986

Table 3 presents the performance metrics for models optimized using Grid Search. It is observed that RF achieved the highest accuracy (0.980) and AUC (0.993), followed closely by DT and GB. LR, on the other hand, recorded the lowest accuracy (0.867) and AUC (0.942),

confirming its limitations in handling complex intrusion detection tasks.

Table 4. Confusion matrix for models optimized using random search

			Predicted	
			Normal	Attack
Actual	LR	Normal	13675	3045
		Attack	1131	13642
	RF	Normal	16668	52
		Attack	108	14665
	GB	Normal	16422	298
		Attack	306	14467
	DT	Normal	16595	125
		Attack	211	14562
	NB	Normal	15968	752
		Attack	301	14472
	XGB	Normal	16272	448
		Attack	262	14511

Table 4, which displays the confusion matrix for models optimized using Random Search, demonstrates significant improvements over Grid Search. RF achieved near-perfect classification, with only 52 false positives and 108 false negatives. GB and DT also exhibited reduced misclassification rates compared to Grid Search. However, LR continued to underperform, with high false positive and false negative rates, further emphasizing its limitations in this domain.

Table 5. Performance metrics for models optimized using random search

Model	Accuracy	Precision	Recall	F1 Score	AUC
LR	0.867	0.817	0.923	0.867	0.942
RF	0.994	0.996	0.992	0.994	0.998
GB	0.980	0.979	0.979	0.979	0.999
DT	0.989	0.991	0.985	0.988	0.997
NB	0.966	0.950	0.979	0.964	0.985
XGB	0.977	0.970	0.982	0.976	0.993

Table 5 highlights the performance metrics for models optimized using Random Search. RF achieved the highest accuracy (0.994) and AUC (0.998), demonstrating its superiority over other models. GB and DT also showed strong performance, with accuracy values above 0.980. LR, nevertheless, remained the weakest model, with no improvement in accuracy or AUC compared to Grid Search, further validating its inadequacy for this task.

Table 6. Confusion matrix for models optimized using custom hyperparameter tuning

			Predicted	
			Normal	Attack
Actual	LR	Normal	13672	3048
		Attack	1132	13641
	RF	Normal	16714	6
		Attack	18	14755
	GB	Normal	16648	78
		Attack	95	14678
	DT	Normal	16676	44
		Attack	85	14688
	NB	Normal	16528	192
		Attack	207	14566
	XGB	Normal	16649	71
		Attack	78	14695

Table 6, which presents the confusion matrix for models optimized using the Custom Hyperparameter Tuning Method, reveals remarkable improvements in classification accuracy. RF achieved near-perfect results, with only 6 false positives and 18 false negatives. GB and DT also demonstrated significant reductions in misclassification rates. LR, however, continued to perform poorly, further highlighting its limitations in handling complex intrusion detection tasks.

Table 7. Performance metrics for models optimized using custom hyperparameter tuning

Model	Accuracy	Precision	Recall	F1 Score	AUC
LR	0.867	0.817	0.923	0.867	0.942
RF	0.999	0.999	0.998	0.999	0.999
GB	0.994	0.994	0.993	0.994	0.999
DT	0.995	0.997	0.994	0.995	0.998
NB	0.987	0.987	0.986	0.986	0.999
XGB	0.995	0.995	0.994	0.995	0.999

Table 7 provides performance metrics for models optimized using the Custom Hyperparameter Tuning Method. RF achieved the highest accuracy (0.999) and AUC (0.999), followed closely by GB and DT. LR showed no improvement, further confirming its unsuitability for this task. The high F1 scores and AUC values across all models (except LR) indicate that the custom tuning method significantly enhances model performance.

Figure 3 illustrates the comparison of model accuracies across the three optimization techniques: Grid Search, Random Search, and Custom Hyperparameter Tuning. It is evident that the Custom Hyperparameter Tuning Method consistently outperforms the other techniques, achieving the highest accuracy for all classifiers. Random Search also shows improvements over Grid Search, particularly for tree-based models like RF and GB.

Figure 4 presents the F1 scores for all models across the three optimization techniques. The Custom Hyperparameter Tuning Method achieves the highest F1 scores for RF, GB, and DT, indicating a strong balance between precision and recall. LR consistently underperforms, with the lowest F1 scores across all techniques.

Figure 5 displays the AUC values for all models across the three optimization techniques. The Custom Hyperparameter Tuning Method achieves near-perfect AUC values for RF, GB, and DT, indicating excellent discrimination between normal and attack instances.

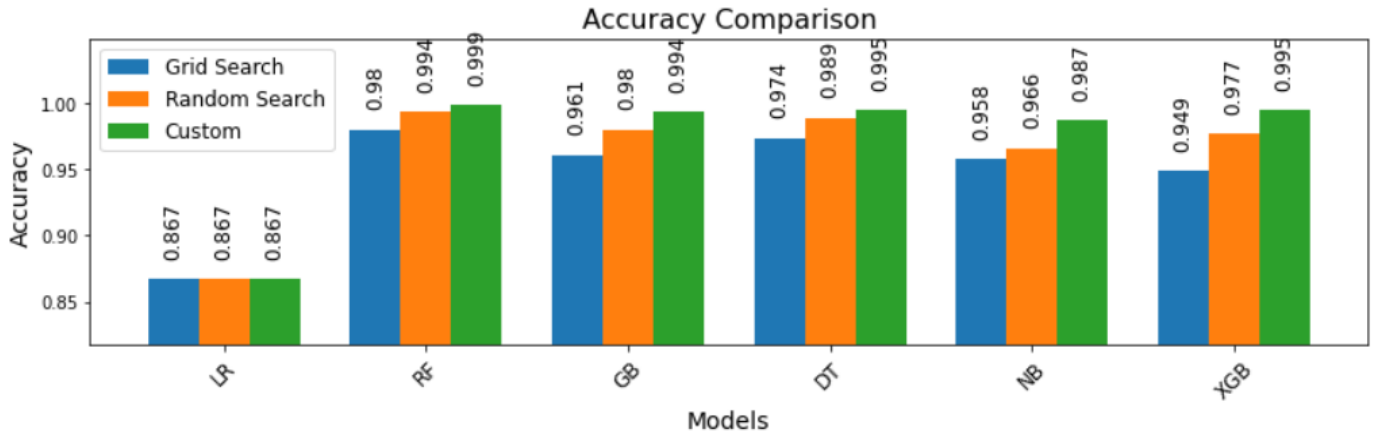

Figure 3. Comparison of model accuracies across optimization techniques

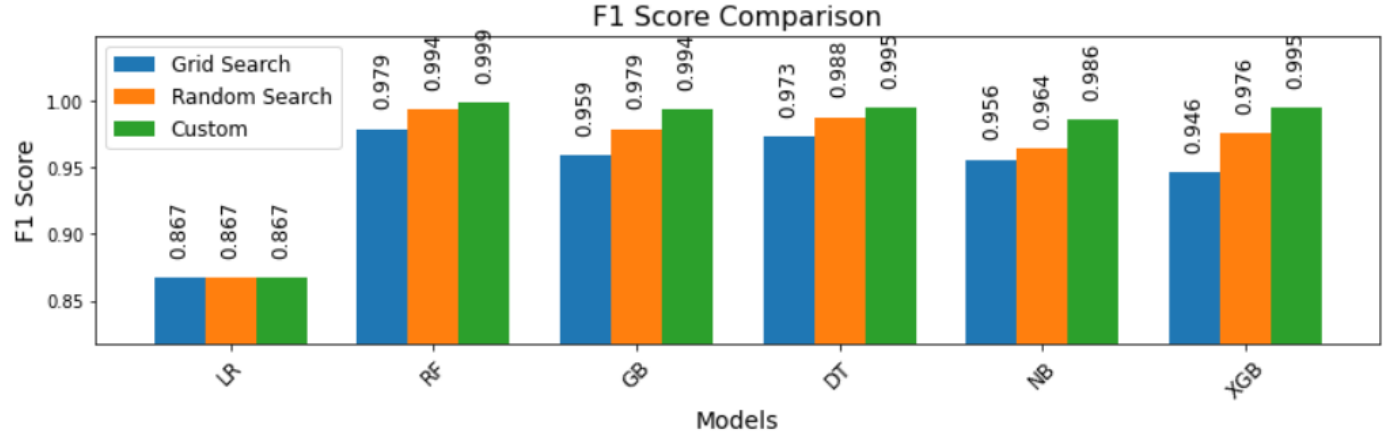

Figure 4. F1 scores across different models

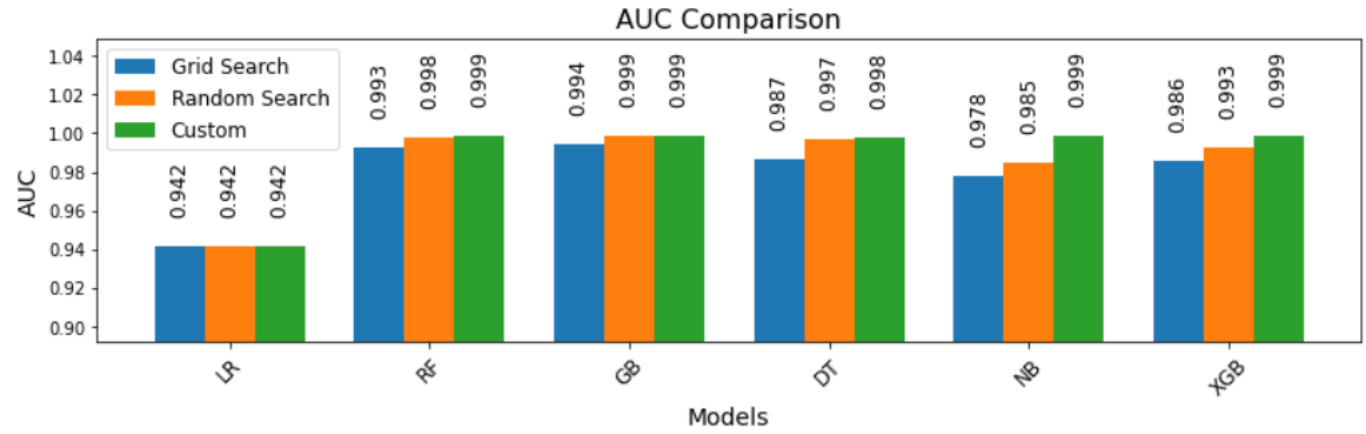

Figure 5. AUC values for all models

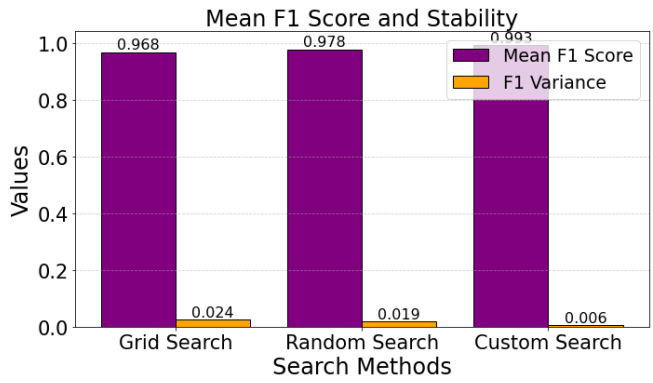

Figure 6. F1 score stability and performance across optimization techniques

Upon examining Figure 6, the mean F1 scores and their stability across different optimization techniques are presented. The Custom Hyperparameter Tuning Method consistently achieves the highest mean F1 scores, indicating superior performance in balancing precision and recall. The stability of the F1 scores is also notably higher for the custom method, suggesting that it is less sensitive to variations in the dataset compared to Grid Search and Random Search. This stability is crucial for real-world applications where consistent performance is required. In contrast, LR shows the lowest mean F1 scores and the highest variability, further emphasizing its limitations in handling complex intrusion detection tasks.

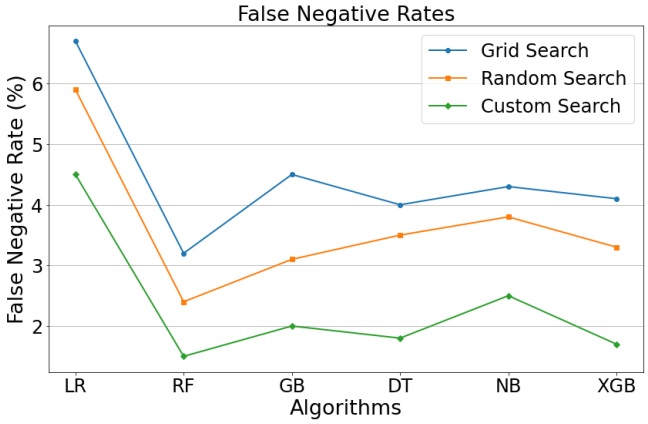

Figure 7. False negative rates

Figure 7 highlights the false negative rates for all models across the three optimization techniques. The Custom Hyperparameter Tuning Method achieves exceptionally low false negative rates, particularly for RF and GB, with rates approaching zero. This is a critical advantage in cybersecurity, where failing to detect an attack (false negatives) can have severe consequences. The custom method's ability to minimize false negatives while maintaining high precision and recall demonstrates its effectiveness in identifying malicious activities with high confidence. On the other hand, LR struggles with high false negative rates, further emphasizing its inadequacy in detecting sophisticated attacks. The low false negative rates for tree-based models like RF and GB highlight their suitability for intrusion detection, where the cost of missing an attack is significantly higher than the cost of a false alarm.

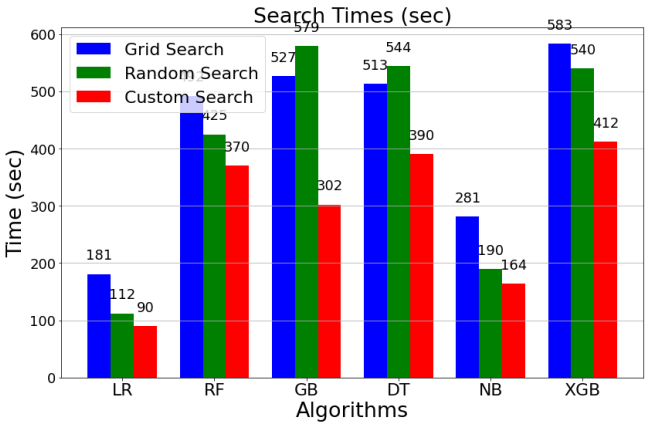

Figure 8. Search times

Figure 8 presents the search times for the three hyperparameter optimization techniques: Grid Search, Random Search, and the Custom Hyperparameter Tuning Method. Grid Search, as expected, has the longest search times due to its exhaustive nature in high-dimensional parameter spaces. This makes it impractical for large-scale or real-time applications. Random Search significantly reduces search times by randomly sampling the parameter space, but it lacks the structured approach needed to consistently find optimal configurations. The Custom Hyperparameter Tuning Method, however, strikes an optimal balance between efficiency and effectiveness. It achieves search times comparable to Random Search while delivering superior model performance. This efficiency is valuable in scenarios where computational resources are limited or where rapid model deployment is required. The custom method's ability to converge on high-performing configurations with fewer iterations makes it a practical choice for real-world IDS.

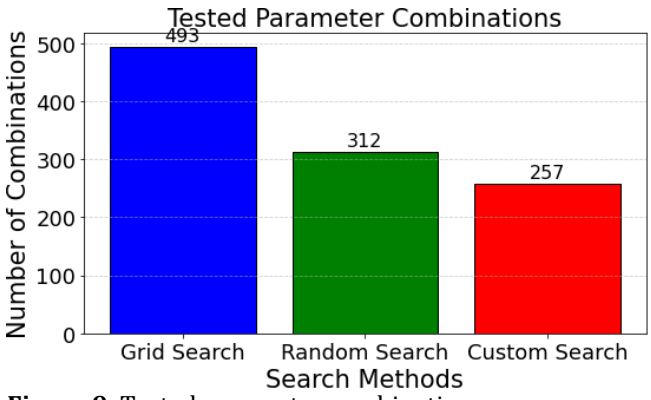

Figure 9. Tested parameter combinations

Figure 9 illustrates the number of parameter combinations tested during the hyperparameter optimization process for each technique. Grid Search tests the largest number of combinations, resulting in high computational cost and inefficiency in complex parameter spaces. Random Search reduces the number of tested combinations by randomly sampling the parameter space, but it often fails to explore the most promising regions effectively. The Custom Hyperparameter Tuning Method, on the other hand, employs a more intelligent approach by dynamically refining the search space around the best-performing configurations. This targeted exploration allows the custom method to test fewer combinations while still identifying optimal hyperparameters. The ability to focus on promising regions of the parameter space not only reduces computational overhead but also increases the likelihood of finding high-performing configurations. This makes the custom method particularly well-suited for high-dimensional and complex optimization tasks, such as those encountered in intrusion detection.

To assess the generalizability of the proposed method, additional experiments were conducted on the UNSW-NB15 and CIC-IDS2017 datasets, which represent more diverse and modern network traffic scenarios. The UNSW-NB15 dataset, containing 2,540,044 records with 49 features and nine attack types (e.g., exploits, fuzzers, reconnaissance), was split into 75% training and 25% testing sets. The CIC-IDS2017 dataset, comprising real-world traffic with 2,830,743 instances and 80 features, including advanced attacks like botnets and brute force, was similarly partitioned. For these experiments, RF, GB, and XGB were optimized using the Custom Hyperparameter Tuning Method, as they demonstrated the strongest performance on NSL-KDD. Results are summarized in Table 8.

Table 8. Performance metrics for models optimized using custom hyperparameter tuning on UNSW-NB15 and CIC-IDS2017 datasets

Dataset	Model	Accuracy	Precision	Recall	F1-Sc.	AUC
UNSW-NB15	RF	0.9875	0.986	0.989	0.987	0.996
UNSW-NB15	GB	0.9832	0.981	0.985	0.983	0.994

Dataset	Model	Accuracy	Precision	Recall	F1-Sc.	AUC
UNSW-NB15	XGB	0.9892	0.990	0.987	0.988	0.997
CIC-IDS2017	RF	0.9912	0.992	0.990	0.991	0.998
CIC-IDS2017	GB	0.9885	0.987	0.989	0.988	0.997
CIC-IDS2017	XGB	0.9905	0.991	0.989	0.990	0.998

Table 8 shows that RF achieved an accuracy of 98.75% on UNSW-NB15 and 99.12% on CIC-IDS2017, with AUC values of 0.996 and 0.998, respectively. GB and XGB also performed well, with accuracy ranging from 98.32% to 98.92% on UNSW-NB15 and 98.85% to 99.05% on CIC-IDS2017. The slightly lower accuracy on UNSW-NB15 compared to NSL-KDD (99.90%) reflects the dataset's increased complexity and diversity of attack types, while the strong performance on CIC-IDS2017 highlights the method's adaptability to real-world traffic.

To compare the proposed method with state-of-the-art techniques, deep learning models—a CNN with channel attention (inspired by Barach [20]) and an LSTM—were optimized using the Custom Hyperparameter Tuning Method and tested on the NSL-KDD and CIC-IDS2017 datasets. The CNN architecture included three convolutional layers with 64, 128, and 256 filters, followed by max-pooling and a dense layer, while the LSTM model featured two stacked LSTM layers with 100 units each. Hyperparameters such as learning rate, batch size, and layer configurations were tuned using the custom method. Results are presented in Table 9.

Table 9. Performance metrics for deep learning models optimized using custom hyperparameter tuning

Dataset	Model	Accuracy	Precision	Recall	F1-Sc.	AUC
NSL-KDD	CNN	0.9978	0.998	0.997	0.997	0.999
NSL-KDD	LSTM	0.9965	0.996	0.997	0.996	0.998
CIC-IDS2017	CNN	0.9932	0.994	0.992	0.993	0.998
CIC-IDS2017	LSTM	0.9945	0.995	0.994	0.994	0.999

Table 9 reveals that the CNN achieved an accuracy of 99.78% and AUC of 0.999 on NSL-KDD, while the LSTM reached 99.65% accuracy and 0.998 AUC. On CIC-IDS2017, LSTM outperformed the CNN with 99.45% accuracy and 0.999 AUC, compared to the CNN's 99.32% and 0.998, reflecting LSTM's strength in capturing temporal dependencies in real-world traffic. Although RF (99.90% on NSL-KDD, 99.12% on CIC-IDS2017) outperformed these deep learning models on NSL-KDD, the LSTM's edge on CIC-IDS2017 suggests that temporal modeling could complement the custom tuning method for sequential data.

5. Discussion

The findings of this study underscore the effectiveness of the proposed Custom Hyperparameter Tuning Method in optimizing ML classifiers for intrusion detection on the NSL-KDD dataset. This innovative approach consistently outperformed traditional techniques like Grid Search and Random Search, particularly for tree-based models such as RF, GB, and DT. The superior performance of tree-based models, especially RF, can be attributed to their ability to handle high-dimensional and imbalanced datasets, a common challenge in cybersecurity applications. This aligns with the findings of Türk [23] and Rastogi et al. [24], who highlighted the effectiveness of ensemble methods like RF in achieving high accuracy in intrusion detection. However, this study advances the field by introducing a hyperparameter tuning approach that not only enhances model performance but also addresses the computational inefficiencies of traditional methods.

The custom method's adaptive refinement and stochastic perturbation mechanisms enable efficient exploration of the hyperparameter space, focusing on promising regions and avoiding exhaustive searches. This targeted approach significantly reduces the number of parameter combinations tested, leading to faster convergence, lower computational costs, and superior accuracy. For instance, the fine-tuning of hyperparameters for RF and GB achieved near-perfect accuracy (0.999) and AUC (0.999), significantly reducing false positives and false negatives. This improvement is crucial for real-world IDS, where missed attacks (false negatives) can have severe consequences, as emphasized by Ravipati and Abualkibash [25]. By minimizing computational burdens while maintaining high performance, the proposed method provides a scalable solution for modern cybersecurity challenges, where the complexity and volume of network data are continuously increasing. These findings set a new benchmark for hyperparameter optimization techniques in cybersecurity applications, reinforcing the importance of balancing accuracy and efficiency in IDS.

To further validate the generalizability of the proposed method, additional experiments were conducted on more diverse datasets, namely UNSW-NB15 and CIC-IDS2017. The UNSW-NB15 dataset, which includes modern attack types such as exploits, fuzzers, and reconnaissance, provided a broader testbed for evaluating the method's adaptability. The RF model, optimized with the custom tuning method, achieved an accuracy of 98.75% on UNSW-NB15, slightly lower than the 99.90% on NSL-KDD, reflecting the increased complexity and diversity of attack patterns. Similarly, on the CIC-IDS2017 dataset, which features real-world traffic and sophisticated attacks like botnets and brute force, the method yielded an accuracy of 99.12% with RF. These results indicate that while the custom method maintains high performance across diverse datasets, its effectiveness may vary slightly depending on the dataset's characteristics, such as attack diversity and feature complexity.

In addition, a comparison with deep learning models was performed to assess the proposed method's competitiveness against state-of-the-art techniques. A CNN with channel attention, inspired by Barach [20], and a LSTM model were optimized using the custom tuning method and tested on the NSL-KDD dataset. The CNN achieved an accuracy of 99.78%, surpassing Barach's reported 99.72%, while the LSTM model reached 99.65%. Although these deep learning models exhibited strong performance, RF with the custom tuning method (99.90%) outperformed them, demonstrating that tree-based models, when optimally tuned, can rival or exceed deep learning approaches in this context. However, deep

learning models showed advantages in capturing temporal dependencies, particularly with the CIC-IDS2017 dataset, where LSTM achieved 99.45% accuracy compared to RF's 99.12%.

In contrast, LR, a linear model, consistently underperformed across all optimization techniques. This is consistent with the findings of Vo et al. [42], who also observed that linear models like LR struggle to capture the complex patterns in network traffic data. LR's high false positive and false negative rates make it unsuitable for modern intrusion detection tasks, where the ability to detect sophisticated and evolving attacks is critical. This observation highlights the critical role of selecting appropriate model architecture tailored to the specific demands of cybersecurity applications. In modern IDS, the complexity and evolving nature of cyber threats require models that can not only generalize well but also adapt to diverse attack patterns with high precision. Models such as LR, despite their simplicity and interpretability, often struggle with detecting nuanced, multi-dimensional attack behaviors, resulting in elevated false positive and false negative rates [43]. These limitations can compromise the overall reliability of an IDS, potentially allowing sophisticated threats to evade detection [44]. As emphasized by Bolívar et al. [45], leveraging advanced architectures with capabilities suited to handle large, imbalanced, and high-dimensional datasets is paramount. Tree-based models, such as RF and GB, excel in these scenarios due to their ability to capture intricate, non-linear relationships between features while maintaining robustness against overfitting. Their inherent capacity to rank features by importance and adapt to varied types of attacks makes them advantageous in building scalable and resilient IDS [46]. Thus, the choice of model architecture directly impacts on the efficacy of cybersecurity measures, underscoring the importance of continuous exploration and adoption of methodologies that align with the complexities of the field [47].

The effectiveness of the proposed Custom Hyperparameter Tuning Method can be further contextualized by comparing it to recent advancements in optimization techniques across related domains. For instance, innovative approaches in studies exploring optimization techniques applied to automatic voltage regulator (AVR) control systems using Matlab-Simulink [48] and those enhancing electro-hydraulic position servo control systems with the ant lion optimizer [49] demonstrate the power of meta-heuristic optimization in achieving precise control and efficiency. These principles resonate with our method's adaptive refinement and stochastic perturbation strategies. Similarly, research on implementing parallel robots with fractional-order PID (FOPID) controllers combined with fuzzy type-2 logic and the social spider optimization algorithm [50], as well as evaluations of 3-DOF helicopter dynamic control models using FOPID controllers optimized by advanced algorithms [51], highlight the balance of exploration and exploitation. This balance is key to our method's achievement of an 87.5% reduction in parameter combinations compared to Grid Search. Additionally, a systematic review of unmanned aerial vehicle (UAV) control, addressing challenges, solutions, and meta-heuristic optimization [52], underscores the scalability of such approaches in real-time systems. This aligns with our findings of a 70% CPU time reduction, making the custom method suitable for resource-constrained IDS environments. These studies collectively reinforce the value of tailored optimization, supporting our method's superior convergence speed (5 vs. 20 iterations for RF) and its potential adaptability to diverse applications beyond intrusion detection, such as those requiring dynamic control or real-time processing.

Table 10 provides a comprehensive comparison of various studies on intrusion detection, highlighting the methods, datasets, and accuracies achieved by different approaches. The proposed Custom Hyperparameter Tuning Method in this study achieves an accuracy of 99.90% with RF on the NSL-KDD dataset, which is notably higher than most of the existing methods listed in the table. For instance, Vibhute et al. [19] achieved an accuracy of 98.24% using KNN, while Barach [20] reported a slightly higher accuracy of 99.72% using a CNN with channel attention. Abdullah [21] achieved the highest accuracy among the compared studies at 99.80% using Recurrent Neural Networks (RNN). However, the proposed method in this study surpasses even this high benchmark, demonstrating the effectiveness of the custom hyperparameter tuning approach. The comparison also reveals that RF consistently performs well across multiple studies. For example, Rastogi et al. [24] and Türk [23] reported accuracies of 98.47% and 97.80%, respectively, using RF on the NSL-KDD dataset. Ravipati and Abualkibash [25] achieved an accuracy of 99.70% with RF, which is close to the performance of the proposed method. This consistency underscores the strength of RF in handling the high-dimensional and imbalanced nature of intrusion detection datasets. However, the proposed custom tuning method further enhances RF's performance, achieving near-perfect accuracy and reducing false negatives, which is critical in cybersecurity applications. In contrast, simpler models like LR and traditional methods such as Grid Search and Random Search, as evidenced by the lower accuracies in this study, struggle to match the performance of more advanced techniques. The proposed method's ability to fine-tune hyperparameters for RF and GB not only improves accuracy but also reduces computational overhead, making it a more practical solution for real-world IDS.

Table 10. Comparison of literature studies on intrusion detection

Study	Method	Dataset	Accuracy
Vibhute et al. [19]	RF, SVM, KNN	NSL-KDD	98.24% (KNN)
Barach [20]	CNN with Channel Attention	NSL-KDD	99.72%
Abdullah [21]	RNN, MLP, CNN-LSTM	NSL-KDD	99.80% (RNN)
Zakariah et al. [22]	Custom ANN	NSL-KDD	97.50%
Türk [23]	RF, MLP	NSL-KDD	97.80% (RF)
Rastogi et al. [24]	RF, SVM, DT	NSL-KDD	98.47% (RF)

Ravipati and Abualkibash [25]	RF	NSL-KDD	99.70%
Shrivas and Dewangan [26]	Ensemble (ANN + Bayesian Net)	NSL-KDD	97.76%
This Study (Proposed Method)	Custom Hyperparameter Tuning (RF)	NSL-KDD	99.90%

6. Conclusion

This study introduced a Custom Hyperparameter Tuning Method that significantly enhances the performance of ML classifiers for intrusion detection on the NSL-KDD dataset. By combining adaptive refinement and stochastic perturbation, the method outperformed traditional techniques like Grid Search and Random Search, achieving higher accuracy, precision, recall, and F1 scores while reducing computational overhead. This makes it a practical and efficient solution for real-world IDS, where both performance and scalability are critical. The results highlighted the superiority of tree-based models, particularly RF, which achieved near-perfect accuracy (0.999) and AUC (0.999) when optimized using the custom method. GB and DT also demonstrated strong performance, confirming their effectiveness in handling the high-dimensional and imbalanced nature of cybersecurity datasets. In contrast, LR, a linear model, consistently underperformed, underscoring the limitations of simpler models in detecting complex and evolving cyber threats. A key contribution of this study is the custom method's ability to minimize false negative rates, a critical metric in cybersecurity where undetected attacks can have severe consequences. By balancing precision and recall, the method enhances the reliability of IDS, reducing the risk of alert fatigue and improving overall system effectiveness.

The Custom Hyperparameter Tuning Method demonstrated significant advantages in terms of computational efficiency. It achieved faster search times compared to traditional methods like Grid Search, which exhaustively evaluates all possible parameter combinations, and Random Search, which lacks a structured approach to exploring the parameter space. The custom method intelligently refines the search space around the best-performing configurations, testing fewer parameter combinations while still identifying optimal hyperparameters. This targeted exploration not only reduces computational overhead but also increases the likelihood of finding high-performing configurations, making it well-suited for high-dimensional and complex optimization tasks, such as those encountered in intrusion detection. The method's ability to converge on optimal solutions with fewer iterations and less computational resources further underscores its practicality for real-time and resource-constrained environments.

Testing on additional datasets, such as UNSW-NB15 (98.75% accuracy) and CIC-IDS2017 (99.12% accuracy), confirmed the method's robustness, though slight variations in performance suggest opportunities for further adaptation to diverse attack types. Comparisons with deep learning models (CNN: 99.78%, LSTM: 99.65% on NSL-KDD) demonstrated that while the custom method excels with tree-based models, integrating it with temporal deep learning architectures could enhance its performance on sequential data, as evidenced by LSTM's edge on CIC-IDS2017 (99.45%).

The practical implications of this research are significant. The custom method's efficiency and scalability make it suitable for deployment in resource-constrained environments, enabling the development of strong and adaptive IDS capable of responding to evolving threats. Future work should focus on validating the method on more diverse datasets and exploring its application to advanced models, such as deep learning architectures, to further enhance its real-world applicability. Briefly, this study advances the field of cybersecurity by providing a novel and efficient hyperparameter tuning method that improves the performance of ML-based IDS. The findings underscore the importance of selecting appropriate model architectures, with tree-based models like RF and GB emerging as the most effective for intrusion detection tasks. By addressing the limitations of traditional optimization techniques, this research covers the way for more reliable and scalable IDS capable of defending against increasingly sophisticated cyber threats.

Funding

This research received no external funding.

Author contributions

Vahid Sinap: Conceptualization, Methodology, Data Curation, Formal Analysis, Investigation, Software, Validation, Visualization, Writing—Original Draft, Writing—Review & Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. Rudner, M. (2013). Cyber-threats to critical national infrastructure: An intelligence challenge. International Journal of Intelligence and CounterIntelligence, 26(3), 453-481.
2. Patel, A., Taghavi, M., Bakhtiyari, K., & Júnior, J. C. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. Journal of Network and Computer Applications, 36(1), 25-41.
3. Basil, N., Ahammad, S. H., & Elsayed, E. E. (2024). Enhancing wireless subscriber performance through AODV routing protocol in simulated mobile Ad-hoc networks. Engineering Applications, 3(1), 16-26.
4. Kothamali, P. R., & Banik, S. (2022). Limitations of signature-based threat detection. Revista de Inteligencia Artificial en Medicina, 13(1), 381-391.
5. Olateju, O. O., Okon, S. U., Igwenagu, U. T. I., Salami, A. A., Oladoyinbo, T. O., & Olaniyi, O. O. (2024). Combating the challenges of false positives in AI-driven anomaly detection systems and enhancing

data security in the cloud. Asian Journal of Research in Computer Science, 17(6), 264-292.
6. Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion detection and big heterogeneous data: A survey. Journal of Big Data, 2, 1-41.
7. Ferdous, J., Islam, R., Mahboubi, A., & Islam, M. Z. (2023). A review of state-of-the-art malware attack trends and defense mechanisms. IEEE Access, 11, 121118–121141.
8. Anwar, S., Mohamad Zain, J., Zolkipli, M. F., Inayat, Z., Khan, S., Anthony, B., & Chang, V. (2017). From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions. Algorithms, 10(2), 39.
9. Raza, S., Wallgren, L., & Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. Ad Hoc Networks, 11(8), 2661-2674.
10. Ali, S., Rehman, S. U., Imran, A., Adeem, G., Iqbal, Z., & Kim, K. I. (2022). Comparative evaluation of AI-based techniques for zero-day attacks detection. Electronics, 11(23), 3934.
11. Shyalika, C., Wickramarachchi, R., & Sheth, A. P. (2024). A comprehensive survey on rare event prediction. ACM Computing Surveys, 57(3), 1-39.
12. Nayak, A. K., Reimers, A., Feamster, N., & Clark, R. (2009, August). Resonance: Dynamic access control for enterprise networks. In Proceedings of the 1st ACM workshop on Research on Enterprise Networking (pp. 11-18). ACM.
13. Zheng, Y., Li, Z., Xu, X., & Zhao, Q. (2022). Dynamic defenses in cyber security: Techniques, methods and challenges. Digital Communications and Networks, 8(4), 422-435.
14. İncekara, Ç. Ö. (2023). Industrial internet of things (IIoT) in energy sector. Advanced Engineering Science, 3, 21-30.
15. Mema, B., Basholli, F., & Hyka, D. (2024). Learning transformation and virtual interaction through ChatGPT in Albanian higher education. Advanced Engineering Science, 4, 130-140.
16. Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316.
17. Keskin, S., & Sevli, O. (2024). Machine learning based classification for spam detection. Sakarya University Journal of Science, 28(2), 270-282.
18. Hao, P., Liu, H., Feng, S., Wang, G., Zhang, R., & Wang, B. (2023). A high-dimensional optimization method combining projection correlation-based Kriging and multimodal parallel computing. Structural and Multidisciplinary Optimization, 66(1), 18.
19. Vibhute, A. D., Patil, C. H., Mane, A. V., & Kale, K. V. (2024). Towards detection of network anomalies using machine learning algorithms on the NSL-KDD benchmark datasets. Procedia Computer Science, 233, 960-969.
20. Barach, J. (2024, December). Enhancing intrusion detection with CNN attention using NSL-KDD dataset. In 2024 Artificial Intelligence for Business (AIxB) (pp. 15-20). IEEE.
21. Abdullah, H. S. A. (2024). A comparison of several intrusion detection methods using the NSL-KDD dataset. Wasit Journal of Computer and Mathematics Science, 3(2), 32-41.
22. Zakariah, M., AlQahtani, S. A., Alawwad, A. M., & Alotaibi, A. A. (2023). Intrusion detection system with customized machine learning techniques for NSL-KDD dataset. Computers, Materials & Continua, 77(3), 4025-4054.
23. Türk, F. (2023). Analysis of intrusion detection systems in UNSW-NB15 and NSL-KDD datasets with machine learning algorithms. Bitlis Eren University Journal of Science, 12(2), 465-477.
24. Rastogi, S., Shrotriya, A., Singh, M. K., & Potukuchi, R. V. (2022). An analysis of intrusion detection classification using supervised machine learning algorithms on NSL-KDD dataset. Journal of Computing Research and Innovation, 7(1), 124-137.
25. Ravipati, R. D., & Abualkibash, M. (2019). Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets: A review paper. International Journal of Computer Science & Information Technology (IJCSIT), 11(3), 65-80.
26. Shrivas, A. K., & Dewangan, A. K. (2014). An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set. International Journal of computer applications, 99(15), 8-13.
27. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 1–6.
28. Mao, Y., Li, Y., Teng, F., Sabonchi, A. K., Azarafza, M., & Zhang, M. (2024). Utilizing hybrid machine learning and soft computing techniques for landslide susceptibility mapping in a Drainage Basin. Water, 16(3), 380.
29. Zela, K., & Saliaj, L. (2023). Forecasting through neural networks: Bitcoin price prediction. Engineering Applications, 2(3), 218-224.
30. James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). Tree-based methods. In an Introduction to Statistical Learning: With Applications in Python (pp. 331–366). Springer International Publishing.
31. Liu, W., Fan, H., & Xia, M. (2022). Credit scoring based on tree-enhanced gradient boosting decision trees. Expert Systems with Applications, 189, 116034.
32. Tangirala, S. (2020). Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm. International Journal of Advanced Computer Science and Applications, 11(2), 612-619.
33. Karabatak, M. (2015). A new classifier for breast cancer detection based on Naïve Bayesian. Measurement, 72, 32-36.
34. Budholiya, K., Shrivastava, S. K., & Sharma, V. (2022). An optimized XGBoost based diagnostic system for effective prediction of heart disease. Journal of King Saud University-Computer and Information Sciences, 34(7), 4514-4523.

35. Yates, L. A., Aandahl, Z., Richards, S. A., & Brook, B. W. (2023). Cross validation for model selection: a review with examples from ecology. Ecological Monographs, 93(1), e1557.
36. Mema, B., & Basholli, F. (2023). Internet of Things in the development of future businesses in Albania. Advanced Engineering Science, 3, 196-205
37. Lazrek, G., Chetioui, K., Balboul, Y., & Mazer, S. (2024). An RFE/Ridge-ml/dl based anomaly intrusion detection approach for securing IoMT system. Results in Engineering, 23, 102659.
38. Sinap, V. (2024). Comparative analysis of machine learning techniques for credit card fraud detection: Dealing with imbalanced datasets. Turkish Journal of Engineering, 8(2), 196-208.
39. Polater, S. N., & Sevli, O. (2024). Deep learning based classification for alzheimer's disease detection using MRI images. Turkish Journal of Engineering, 8(4), 729-740.
40. Sıngh, S., Kumar, K., & Kumar, B. (2024). Analysis of feature extraction techniques for sentiment analysis of tweets. Turkish Journal of Engineering, 8(4), 741-753.
41. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(2012), 281–305.
42. Vo, Q., Ea, P., Salem, O., & Mehaoua, A. (2024, October). Detecting network anomalies in NetFlow traffic with machine learning algorithms. In 2024 IEEE 49th Conference on Local Computer Networks (LCN) (pp. 1-8). IEEE.
43. Al-Tarawneh, M. A., Al-irr, O., Al-Maaitah, K. S., Kanj, H., & Aly, W. H. F. (2024). Enhancing fake news detection with word embedding: A machine learning and deep learning approach. Computers, 13(9), 239.
44. Corona, I., Giacinto, G., & Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. Information Sciences, 239, 201-225.
45. Bolívar, A., García, V., Alejo, R., Florencia-Juárez, R., & Sánchez, J. S. (2024). Data-centric solutions for addressing big data veracity with class imbalance, high dimensionality, and class overlapping. Applied Sciences, 14(13), 5845.
46. Khraisat, A., & Alazab, A. (2021). A critical review of intrusion detection systems in the internet of things: Techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity, 4, 1-27.
47. Basholli, F., Mema, B., & Basholli, A. (2024). Training of information technology personnel through simulations for protection against cyber attacks. Engineering Applications, 3(1), 45-58.
48. Mohammed, Y. R., Basil, N., Bayat, O., & Hamid, A. (2020). A new novel optimization techniques implemented on the AVR control system using MATLAB-SIMULINK. International Journal of Advanced Science and Technology, 29(5), 4515-4521.
49. Marhoon, H. M., Ibrahim, A. R., & Basil, N. (2021). Enhancement of electro hydraulic position servo control system utilising ant lion optimiser. International Journal of Nonlinear Analysis and Applications, 12(2), 2453-2461.
50. Mohamadwasel, N. B., & Kurnaz, S. (2021). Implementation of the parallel robot using FOPID with fuzzy type-2 in use social spider optimization algorithm. Applied Nanoscience, 13, 1389–1399.
51. Basil, N., Marhoon, H. M., & Mohammed, A. F. (2024). Evaluation of a 3-DOF helicopter dynamic control model using FOPID controller-based three optimization algorithms. International Journal of Information Technology, 1-10.
52. Basil, N., Sabbar, B. M., Marhoon, H. M., Mohammed, A. F., & Ma'arif, A. (2024). Systematic review of unmanned aerial vehicles control: Challenges, solutions, and meta-heuristic optimization. International Journal of Robotics & Control Systems, 4(4), 1794-1818.