Fuzzy Systems and Soft Computing
ISSN : 1819-4362

ENHANCING PERFORMANCE OF INTRUSION DETECTION SYSTEM IN THE NSL-KDD DATASET USING META-HEURISTIC AND MACHINE LEARNING ALGORITHMS

K. Dinesh Research Scholar, Department of Computer science, Dr.SNS Rajalakshmi College of arts and science, Coimbatore-49. dineshcsc1990@gmail.com
D. Kalaivani Associate Professor& Head Department of Computer Technology, Dr.SNS Rajalakshmi College of Arts and Science, Coimbatore-49.

Abstract
As digital technologies become more integral to our lives, the risk of cyber-attacks and data breaches has grown significantly. In the realm of cyber security, the NSL-KDD dataset is a well-established benchmark for assessing the performance of Intrusion Detection Systems (IDS). These systems are crucial for identifying and responding to threats, thereby safeguarding organizational data and infrastructure. This research introduces an innovative approach to improving IDS performance on the NSL-KDD dataset through the use of meta-heuristic algorithms and machine learning techniques. By leveraging multiple meta-heuristic algorithms, the study aimed to optimize the hyperparameters of several machine learning models, including Random Forest (RF), Classification and Regression Trees (CART), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). The effectiveness of the IDS was measured using key evaluation metrics such as precision, recall, F1-score, and accuracy. The findings revealed that the proposed method achieved superior detection accuracy and robustness compared to existing techniques. This research underscores the value of meta-heuristic algorithms in enhancing IDS models and demonstrates the potential of machine learning-based solutions in tackling cyber security challenges.
Key words: Cyber-Attacks, Intrusion Detection Systems, Hyperparameters, Meta-Heuristic, Machine Learning.

1. Introduction
The significance of cyber security in the digital age has grown due to our increasing reliance on advanced technology in everyday activities. As digital transformation progresses, businesses, governments, and individuals depend more heavily on digital systems and networks to handle sensitive information. This shift has led to a rise in cyber-attacks, including data breaches, ransomware attacks, and phishing scams, which have become more prevalent and complex, causing significant financial losses and reputational harm.The COVID-19 pandemic has further accelerated the adoption of digital technologies, exposing vulnerabilities in remote work environments and making cyber security even more crucial. As a result, there is an urgent need for effective cyber security measures and strategies to protect against cyber threats and mitigate the potential impacts of these attacks.

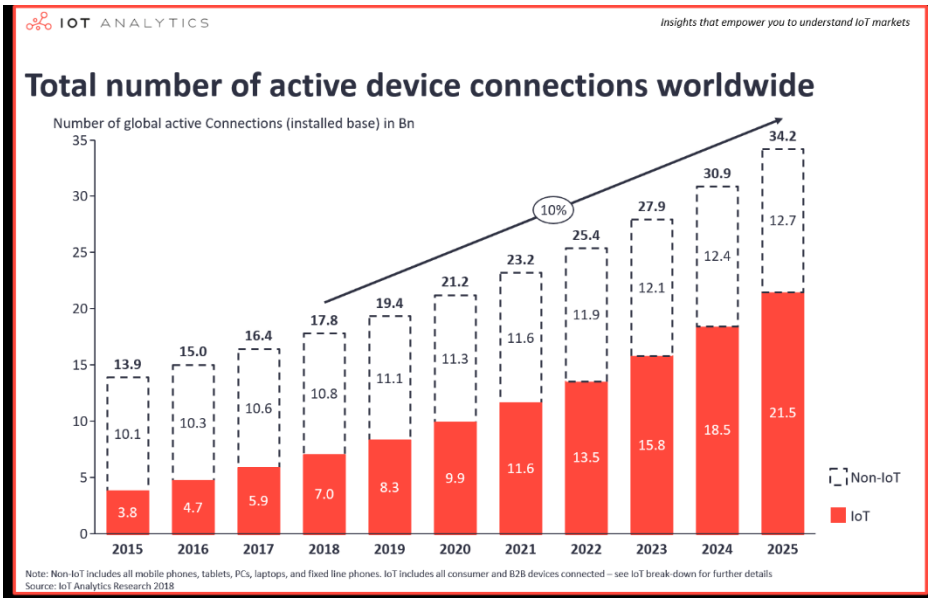


Figure.1. Number of connected devices in internet [2]

The number of connected devices globally has surpassed 17 billion, with IoT devices making up 7 billion of that total. This excludes smartphones, tablets, laptops, and fixed-line phones. IoT devices, used in consumer and enterprise/B2B applications, are a major driver of global connectivity growth. The number of active IoT devices is expected to reach 10 billion by 2020 and 22 billion by 2025, counting only active connections and not outdated devices.

Intrusion Detection Systems (IDS) are essential for detecting and preventing cyber-attacks by analyzing network traffic and host activity for signs of malicious behavior. IDS come in two types: network-based (NIDS) and host-based (HIDS). NIDS monitor network traffic for attacks, while HIDS monitor activity on individual host systems. IDS can alert security personnel to detected attacks and some can automatically respond by blocking traffic or isolating compromised systems. By quickly identifying and responding to threats, IDS help organizations minimize the impact of cyber-attacks and protect sensitive data.

The NSL-KDD dataset is a benchmark used to evaluate IDS performance. It is a refined version of the KDD Cup 1999 dataset, addressing issues like duplicate and irrelevant records and the absence of newer attack types. The NSL-KDD dataset includes 22 attack types, providing a more comprehensive and representative distribution. It contains around 4 million network connections categorized into four classes: normal, probe, DoS (denial of service), and U2R (user-to-root). Each connection is described by 41 features, such as protocol type, service, source and destination addresses, and flags. The dataset also includes class labels indicating whether a connection is normal or an attack, specifying the attack type if applicable.

The NSL-KDD dataset is crucial for evaluating IDS algorithms, offering a standardized benchmark for assessing their accuracy and effectiveness before real-world deployment. It has been widely used for developing and testing new IDS algorithms, providing a foundation for further network security research.

Meta-heuristic and machine learning (ML) algorithms are used to develop IDS and enhance network security. Meta-heuristic algorithms search for near-optimal solutions by exploring the search space, used for feature selection and parameter tuning in IDS. Machine learning algorithms learn from data to detect anomalies and classify network traffic as normal or malicious, enabling models to identify various attack types. Combining meta-heuristic and machine learning techniques significantly improves IDS accuracy and efficiency. Evaluating different algorithms helps researchers identify the most effective methods for improving network security.

2. Literature Review

Numerous research studies have focused on improving the performance of Intrusion Detection Systems (IDS) by employing various methods, including meta-heuristic and machine learning (ML) algorithms.

2.1 Data processing for IDSs

Recent studies have demonstrated that combining Min-Max normalization with one-hot encoding can enhance the performance of machine learning algorithms for intrusion detection on the NSL-KDD dataset. Yang et al. (2021) proposed a hybrid feature selection method that incorporated both techniques, achieving superior results compared to other methods. Similarly, Garg et al. (2020) found that Min-Max normalization outperformed other normalization techniques. Hu et al. (2021) also utilized Min-Max normalization and one-hot encoding in a deep neural network-based IDS. Collectively, these studies indicate that this combination of techniques can significantly improve IDS performance on the NSL-KDD dataset.

2.2 Machine leaning methods for IDSs

Several studies have explored various methods for enhancing Intrusion Detection Systems (IDS), including ensemble models, deep learning techniques, and hybrid approaches. Alazab and Venkatraman (2021) proposed an ensemble model that combines six machine learning algorithms, demonstrating its effectiveness on the UNSW-NB15 dataset. Chen et al. (2021) introduced a deep learning architecture called Neural Tree Network (NTN) for IDS, which showed superior performance on the CIC-IDS2017 dataset. Huang et al. (2021) presented a hybrid approach combining a ResNet-based deep neural network with the SVM algorithm, while Ahmad et al. (2020) developed an IDS that utilizes a deep convolutional neural network (CNN) for feature extraction and the SVM algorithm for classification. Additionally, Zeng et al. (2020) proposed a feature selection

method using mutual information and the grey wolf optimizer (GWO) algorithm, demonstrating its effectiveness on the NSL-KDD dataset. These studies suggest that various approaches can improve IDS performance, and their effectiveness can be evaluated using different benchmark datasets.

2.3 Meta-heuristic based Machine leaning methods for IDSs

Combining Support Vector Machine (SVM) with meta-heuristic algorithms like Firefly Algorithm and Artificial Bee Colony Algorithm has shown promise in enhancing the performance of Intrusion Detection Systems (IDS) on the NSL-KDD dataset, as evidenced by Aber et al. (2020) and Shalaginov et al. (2018). These hybrid approaches have demonstrated superiority over standalone SVM and other existing methods in terms of accuracy, precision, recall, and F1-score [13].

While SVM has been identified as performing better than other machine learning algorithms on the NSL-KDD dataset by Zaki et al. (2018), its performance can vary based on factors such as dataset quality, size, and hyperparameter selection. Therefore, it is advisable to assess different algorithms and hyperparameters across diverse datasets [14].

Kumar and Singh proposed two hybrid approaches that integrate Particle Swarm Optimization (PSO) with SVM, as well as Genetic Algorithm (GA) with SVM, for IDS on the NSL-KDD dataset. Both approaches demonstrated superior performance compared to standalone SVM and other existing methods [15].

In conclusion, leveraging meta-heuristic algorithms alongside machine learning techniques has the potential to enhance IDS performance across various datasets. Future research could explore the effectiveness of different meta-heuristic algorithms and their combinations with machine learning approaches in further detail.

2.4 Research gap and solution

Current machine learning-based IDS systems face several challenges including interpretability issues, limited ability to detect new or unknown attacks, resource-intensive operations, sensitivity to data quality and quantity, susceptibility to adversarial attacks, and difficulties in feature selection. Overcoming these challenges will require ongoing research and development aimed at creating more interpretable, robust, and resource-efficient machine learning models. Enhancements should focus on improving the systems' capacity to handle emerging threats, reducing vulnerability to adversarial manipulations, and refining feature selection processes. Furthermore, improving the quality and quantity of training data is essential for advancing the effectiveness of machine learning-based IDS.

3. Methodology

The study utilizes the NSL-KDD dataset and introduces a methodology comprising three key steps: pre-processing, feature extraction, and classification. Pre-processing is essential as it involves formatting initial data and normalizing values to optimize the performance of machine learning models. Feature extraction plays a critical role in eliminating irrelevant features that could otherwise diminish model accuracy and prolong training duration. The training phase involves fine-tuning parameters to improve model accuracy, thereby enhancing overall performance. The proposed model structure is illustrated in the figure below.

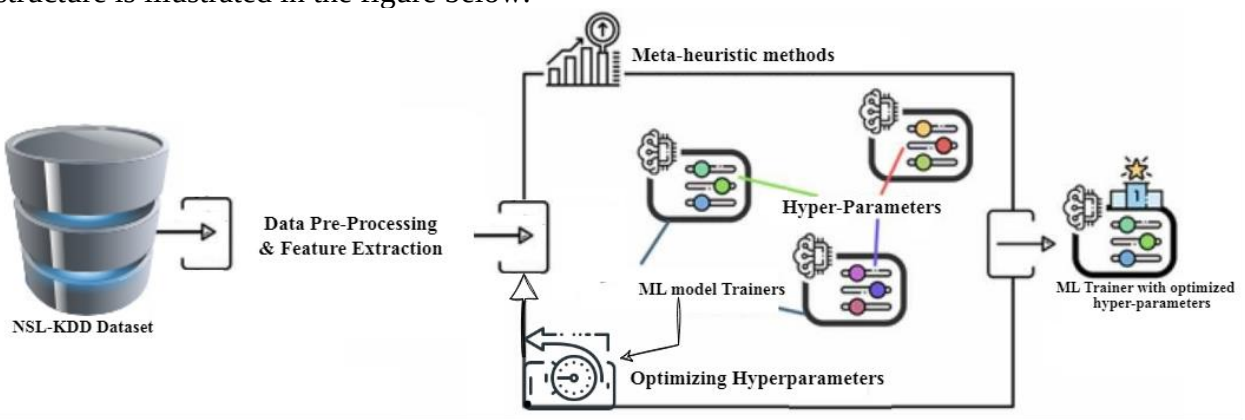


Figure.2. Developing a Meta-Heuristic based ML model: A step-by-step process

The proposed approach involves leveraging Meta-Heuristic Algorithms to optimize the hyperparameters of machine learning models for Intrusion Detection Systems (IDS). Meta-Heuristic Algorithms, inspired by natural phenomena, are capable of searching for optimal hyperparameters critical for enhancing model performance. Integrating Meta-Heuristic Algorithms with machine

learning models aims to improve accuracy, minimize false positives and negatives, and enhance anomaly detection capabilities. This approach offers efficiencies in exploring large hyperparameter spaces effectively, reducing manual effort. Overall, the proposed methodology holds promise for significantly enhancing IDS accuracy while mitigating false alarms.

3.1 NSL-KDD database description

The NSL-KDD dataset is a widely utilized benchmark in intrusion detection systems, derived from the KDD99 dataset. It addresses shortcomings of the KDD99 dataset, such as redundancy and duplicate records that could skew classifier outcomes. This freely available dataset is provided by the Canadian Institute of Cybersecurity and includes both KDDTrain+ and KDDTest+ sets. Notably, KDDTest+ incorporates seventeen additional attack types absent in KDDTrain+, necessitating the exclusion of instances associated with these categories for fair classification. For further details regarding the features of KDDTrain+ and KDDTest+ sets, refer to Table 1.

Table.1. Feature details of NSL-KDD dataset

duration	*destination bytes*	*num failed logins*	*num root*
is guest login	*rerror rate*	*dst host count*	*dst host srv diff host rate*
protocol type	*land*	*logged in*	*num file creations*
count	*srv rerror rate*	*dst host srv count cont*	*dst host serror rate*
service	*wrong fragment*	*num compromised*	*num shells*
srv count	*same srv rate*	*dst host same srv rate cont*	*dst host srv serror rate*
flag	*urgent*	*root shell*	*num access files*
serror rate	*diff srv rate*	*dst host diff srv rate*	*dst host rerror rate*
source bytes	*hot*	*su attempted*	*Num outbound cmds*
srv serror rate	*srv diff host rate*	*dst host same src port rate*	*dst-host srv rerror rate*
			is host login

3.2Data Preprocessing

The min-max normalization method employed in the NSL-KDD dataset involves adjusting each numerical feature x by subtracting its minimum value and then dividing by the range (difference between maximum and minimum values) of that feature. This process scales each feature to a range between 0 and 1, ensuring uniformity across the dataset. The formula for calculating the normalized value x_scaled is:

x_scaled = (x - min(x)) / (max(x) - min(x)).

It's crucial to apply this formula separately to each numerical feature in the dataset, utilizing the minimum and maximum values specific to each feature from the training dataset. This practice prevents the normalization process from being influenced by any information in the test dataset, thereby avoiding potential bias in the results [16].

3.3 One-hot-encoding

To perform one-hot encoding on a categorical feature with n unique values in the NSL-KDD dataset, we create n new binary columns, one for each unique value. The value of each binary column is 1 if the original feature value matches the corresponding unique value, and 0 otherwise.

For example, if we have a categorical feature "protocol_type" with three unique values: TCP, UDP, and ICMP, we would create three new binary columns: "protocol_type_TCP", "protocol_type_UDP", and "protocol_type_ICMP". To one-hot encode "protocol_type", we use the following mathematical equation:

- protocol_type_TCP = 1 if protocol_type = "TCP", 0 otherwise
- protocol_type_UDP = 1 if protocol_type = "UDP", 0 otherwise
- protocol_type_ICMP = 1 if protocol_type = "ICMP", 0 otherwise

Note that one-hot encoding can increase the dimensionality of the dataset and potentially slow down machine learning algorithms. Therefore, it's important to consider which categorical features to one-hot encode and which ones to leave as is [17].

3.4 Feature extraction

The processing module utilized in this approach extracts the most highly correlated features from the dataset. To accomplish this, the percentage of zeros is evaluated for each continuous feature in both the KDDTrain+ and KDDTest+ sets. The distribution of null values for each numeric variable in the KDDTrain+ set is depicted in Figure 2.

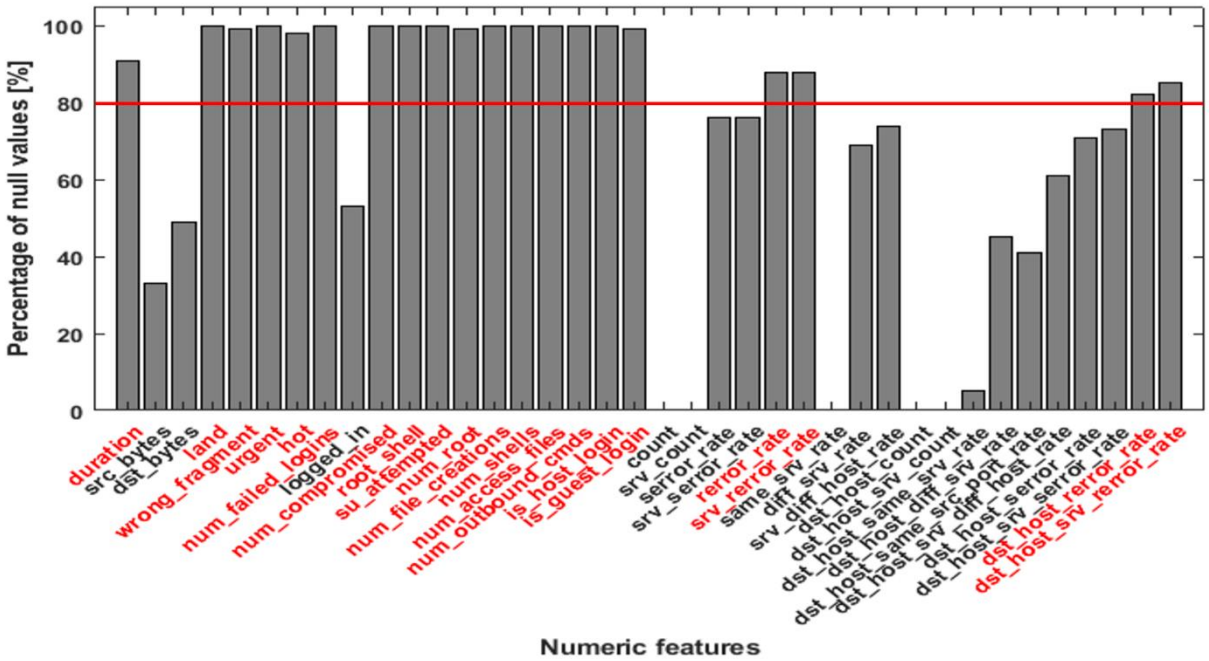


Figure.3. Graph of null values included in the 38 numeric variables of the KDDTrain+ set

In the study, feature vectors containing more than 80% zeros were omitted. Specifically, 20 variables were identified and excluded (highlighted in red in Figure 3). The remaining dataset included 18 continuous features, supplemented by 84 one-hot-encoded vectors, resulting in a final feature vector dimensionality of 102. This enhanced feature vector served as the input for the machine learning methods employed in the research.

3.5 Classification

Supervised machine learning algorithms have been widely utilized to evaluate the effectiveness of intrusion detection on the NSL-KDD dataset. The dataset comprises multiple features that signify network traffic and attacks. The objective is to classify the traffic into either normal or malicious categories. Several supervised ML algorithms such as RF, SVM, CART, and MLP have been applied to the NSL-KDD dataset.

i. CART:

The CART algorithm is utilized in building decision trees for classification tasks, including in IDS for categorizing network traffic as benign or malicious. The process involves several steps:

1. Begin with a labeled dataset of network traffic samples.
2. Calculate information gain or Gini impurity for each feature.
3. Select the feature with the highest gain or lowest impurity.
4. Split the dataset into two subsets based on the selected feature.
5. Recursively apply steps 2-4 to each subset until a stopping criterion is met.
6. Assign class labels to each leaf node based on the majority class in that node's subset.
7. The resulting decision tree can classify new network traffic samples by evaluating their features.

The CART algorithm generates a decision tree structure that can be represented as a series of if-then rules or binary splits on input features. The effectiveness of the model hinges on factors such as data quality and hyperparameter settings, such as maximum tree depth and minimum samples required to split a node [18].

ii. RF:

To implement the Random Forest (RF) method for IDS classification using the NSL-KDD dataset, the following steps are typically followed:

1. Randomly select a subset of samples and input features from the dataset.
2. Build a decision tree using the selected subset.
3. Repeat steps 1-2 to construct a forest of decision trees.
4. To classify a new network traffic sample, apply each tree in the forest to the sample and tally the number of trees that classify it as malicious or benign.
5. Assign the class label based on the majority vote (common vote) of the trees in the forest.

The RF algorithm produces a collection of decision trees that collectively classify new network traffic samples based on their features. The performance of the model is influenced by factors such as the number of trees in the forest, the quality of the data, and hyperparameters including maximum tree depth and feature subset size [19].

iii. SVM:

To construct an SVM system for IDS classification using the NSL-KDD dataset, the following steps are typically followed:

1. Preprocess the data by standardizing the features to have zero mean and unit variance.
2. Select an appropriate kernel function to map the data into a higher-dimensional feature space, if necessary.
3. Solve the optimization problem to find the hyperplane that maximizes the margin between support vectors of each class.
4. Classify new network traffic samples by mapping them into the same feature space as the training data and determining their position relative to the learned hyperplane.

The output of the SVM algorithm is a decision boundary that effectively separates different classes of network traffic samples. The performance of the SVM model is influenced by the selection of the kernel function and key hyperparameters such as the regularization parameter and kernel bandwidth [20].

iv. MLP:

To construct an MLP model for IDS classification using the NSL-KDD dataset, follow these steps:

1. Preprocess the input data by standardizing it to have zero mean and unit variance.
2. Define the structure of the MLP network, specifying the number of layers, neurons in each layer, and activation functions for each neuron.
3. Train the network using an optimization algorithm such as stochastic gradient descent to minimize the error between predicted and actual class labels.
4. Evaluate the performance of the trained network on a separate validation set to assess its accuracy.
5. Fine-tune hyperparameters, including the learning rate and regularization strength, to enhance the model's performance on the validation set.

The output of the MLP algorithm is a model capable of classifying new network traffic samples as either malicious or benign based on their features. The effectiveness of the MLP model relies on the chosen architecture, optimal hyperparameters, and the size and quality of the input data [21].

3.6 Hyper-Parameter Tuning (HT)

Machine learning parameters are settings learned during training that significantly influence model performance. These parameters encompass a wide range of values and configurations tailored to specific problems. Examples include learning rate, regularization parameters, number of hidden layers, activation functions, number of trees (in ensemble methods), kernel functions (for SVMs), and number of clusters (in clustering algorithms).

The selection of these parameters is crucial and typically determined through iterative testing or automated techniques such as grid search or Bayesian optimization. Each machine learning technique has its own distinct set of parameters that play a critical role in optimizing model performance for various tasks and datasets.

Table.2. Parameters for Applied ML techniques

Algorithms	Parameters
CART [18]	The maximum depth of the tree, the minimum number of samples needed to split a node, and the criterion used for splitting nodes
RF [19]	Number of trees, max tree depth, min samples to split a node, split criterion, and number of features for best split.
SVM [20]	Choice of kernel function, Kernel bandwidth
MLP [21]	Number of neurons per hidden layer, activation function for each neuron, and learning rate

HT is the process of finding the best hyperparameters for a ML algorithms. These are set by the practitioner and include values such as learning rate, regularization parameter, number of hidden layers, and activation function. Different methods can be used for hyperparameter tuning [22]:

- **Grid Search (GS):** This involves specifying a grid of possible hyperparameter values and testing every combination of hyperparameters to find the best set of values.
- **Gradient-Based Optimization (GBO):** This algorithm involves using gradient descent to optimize the hyperparameters. This method can be computationally expensive since it requires calculating gradients with respect to the hyperparameters, but it can be useful for small search spaces and differentiable objective functions [23].
- **Simulated Annealing (SA):** This algorithm is inspired by metallurgy's annealing process, which gradually cools a material to minimize defects. In the context of hyperparameter tuning, SA randomly selects a new set of hyperparameters and accepts the new solution with some probability based on a temperature parameter, which is gradually decreased over time to converge towards the optimal solution [24].
- **Genetic Algorithms (GA):** This method uses a population of hyperparameters and evolves the population using evolutionary principles, such as mutation and selection, to find the best set of hyperparameters.

Hyperparameter tuning should always be performed on a separate validation set to avoid overfitting to the training data.

i. Proposed algorithm steps

Utilizing a metaheuristic algorithm for hyperparameter tuning in machine learning follows these steps:

- ***Step1: Define the search space:*** Let S be the search space, where each element s represents a candidate solution consisting of a set of hyperparameters to be optimized.
- ***Step2: Initialize the population:*** Let P be the population of candidate solutions, where each element p_i represents a solution in the search space. The population is initialized by randomly generating or selecting initial solutions from the search space.
- ***Step3: Evaluate the fitness:*** Let f(p_i) be the fitness function that evaluates the performance of each candidate solution p_i using a metric such as accuracy, AUC, or F1 score.
- ***Step4: Update the population:*** Apply the metaheuristic algorithm to generate new candidate solutions based on the current population and fitness values. Let P' be the new population, where each element p'_i is generated by applying variation operators such as mutation and crossover to the current population P.
- ***Step5: Evaluate the fitness of the new solutions:*** Evaluate the fitness of the new candidate solutions using the fitness function f(p'_i), and compare them to the previous best solutions to determine if there has been an improvement in performance.
- ***Step6: Repeat steps 4-5:*** Iterate the algorithm until some stopping criterion is met, such as a maximum number of iterations or convergence of the fitness values.
- ***Step7: Select the best hyperparameters:*** Once the algorithm has completed, select the hyperparameters that correspond to the best performing solution based on the fitness function.

These optimized hyperparameters are subsequently used to train the final machine learning model on the entire training dataset and evaluated on a separate test dataset to estimate its generalization performance.

6. Result and Discussion

The objective of this research is to optimize hyperparameters for Machine Learning (ML) algorithms to achieve optimal performance in Network Intrusion Detection using the NSL-KDD dataset. The study will be conducted on a custom-built computer running Windows 11, equipped with an Intel Core i5 CPU, 8GB RAM, and a 256GB SSD. Python will serve as the primary programming language, utilizing libraries such as Scikit-learn, Pandas, NumPy, Matplotlib, and Pickle, along with the Keras framework for data analysis, modeling, experimentation, and performance evaluation. The development and testing of models will be carried out using Jupyter Notebook.

Performance Metrics and Evaluation: The research employs ML methodologies to enhance the accuracy of network data classification. The datasets were partitioned as shown in the table below.

The NSL-KDD dataset, which includes a subset of selected records, facilitates efficient model evaluation. This dataset encompasses various types of attacks, summarized in the table along with their respective attack types [24]:

Table.3. Type of attacks

S.No	Attack Type	Attack
1	Denial of Service (DoS)	back, land, teardrop, neptune, pod, smurf
2	Remote to Local (R2L)	buffer_overflow, ftp_write, guess_passwd, imap, loadmodule, multihop, perl, phf, rootkit, spy, warezclient, warezmaster
3	Probe	ipsweep, nmap, portsweep, satan
4	User to Root (U2R)	buffer_overflow, httptuneel, rootkit,loadmodule, perl, xterm, ps, SQLattack

The proposed architecture was trained and tested using a dataset comprising 125,972 items in the training set and 22,544 items in the test set. This dataset is composed of 41 features categorized into four groups. The initial three features include protocol type, service, and flag [25].

Table.4. testing and training .of data sets

S.No	NSL-KDD Dataset	Total data	Normal	DoS	R2L	U2R	Probe
01	Training set	125,937	67,343	45,927	995	52	11,656
02	Testing set	22,544	9711	7458	2754	200	2421

The proposed architecture was tested on a dataset, and metrics were utilized to evaluate its performance. The table below displays the mathematical expressions for the applied metrics.

Table.5. Mathematical Equations for the Computation of Performance Measures

S.No	Metrics	Expression
01	Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
02	Recall	$\frac{\text{TP}}{\text{TP+FN}}$ x100
03	Precision	$\frac{TN}{TP + FP}$
04	F1-Score	$2.\frac{Precison * Recall}{Precision + Recall}$

TP is True Positive Values, TN is True Negative Values, FP is False Positive and FN is False Negative values [26].

Results and Findings: Here, the improvement of the proposed design as well as the impact of various existing models will be discussed. The effectiveness of various ML architectures is compared in the tables that follow.

Table.5. Performance analysis of ML algorithms

ML Algorithms	Accuracy	Precision	Recall	F1-Score
CART	85	87	81	84
RF	90	91	87	89
SVM	91	92	87	90
MLP	93	93	91	92

The table compares the performance of four machine learning algorithms: CART, RF, SVM, and MLP. CART has the lowest scores with 85% accuracy and an F1-Score of 84%. RF performs better with 90% accuracy and an 89% F1-Score. SVM slightly surpasses RF with 91% accuracy and a 90%

F1-Score. MLP leads with the highest performance, achieving 93% accuracy, 93% precision, 91% recall, and a 92% F1-Score, making it the best among the algorithms evaluated.

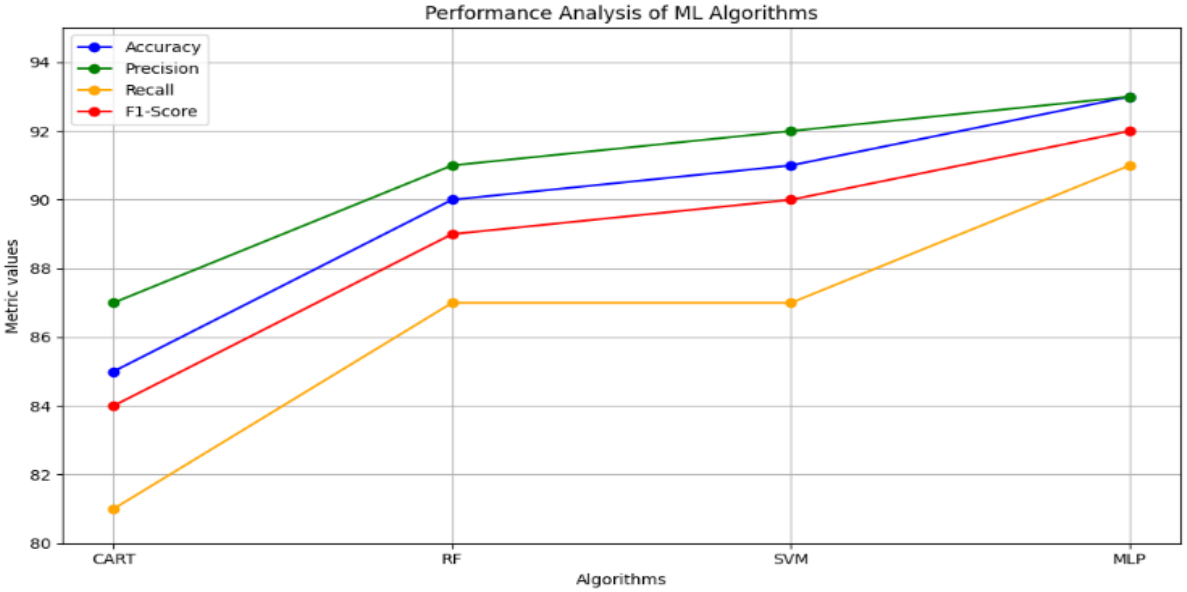
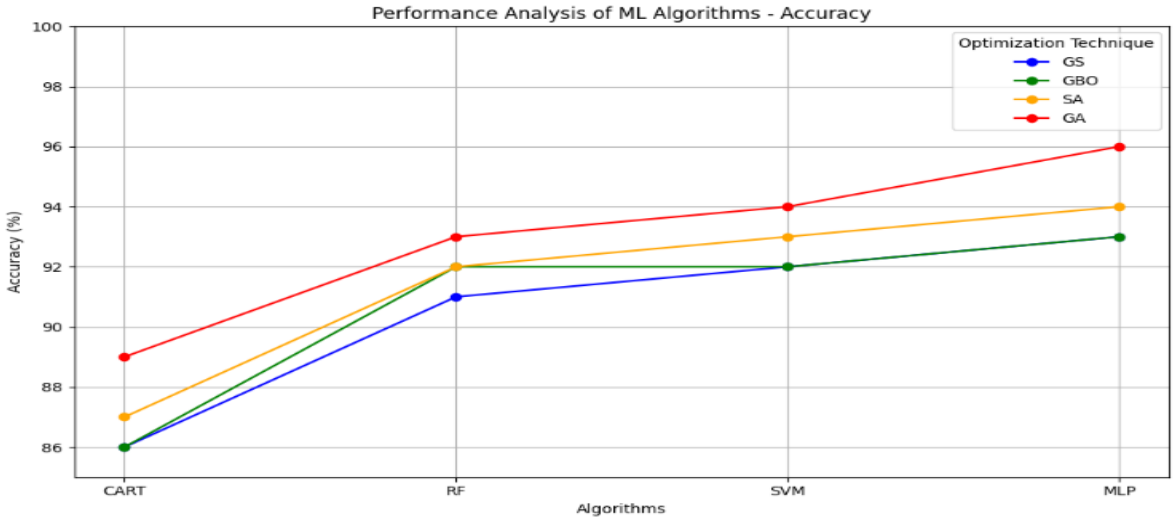


Figure.4. Performance Comparison of ML Algorithms

Table.6. Performance analysis of Optimized ML algorithms of Accuracy

	GS	GBO	SA	GA
Accuracy (%)				
CART	86	86	87	89
RF	91	92	92	93
SVM	92	92	93	94
MLP	93	93	94	96



The table compares accuracy (%) of CART (86-89%), RF (91-93%), SVM (92-94%), and MLP (93-96%) using GS, GBO, SA, and GA. Results highlight MLP's highest accuracy with GA (96%), while RF consistently performs well across methods, and CART and SVM show incremental improvements with advanced optimization techniques.

Table.7. Performance analysis of Optimized ML algorithms of Precision

	GS	GBO	SA	GA
Precision (%)				
CART	88	88	89	91
RF	93	94	94	95
SVM	94	94	95	96
MLP	95	95	96	97

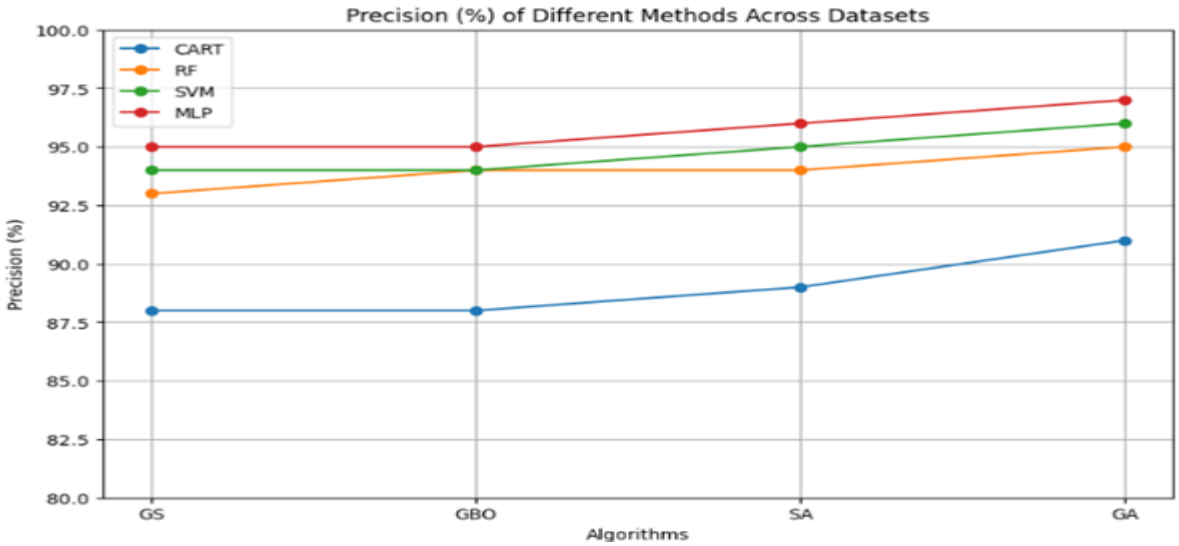


Table 7 compares precision (%) of CART (88-91%), RF (93-95%), SVM (94-96%), and MLP (95-97%) using GS, GBO, SA, and GA. Results demonstrate MLP's highest precision with GA (97%), while RF and SVM also show strong performance across methods, and CART exhibits incremental improvements with advanced optimization techniques.

Table.8. Performance analysis of Optimized ML algorithms of Recall

	GS	GBO	SA	GA
Recall (%)				
CART	82	82	83	85
RF	87	88	88	89
SVM	88	88	89	90
MLP	89	89	90	92

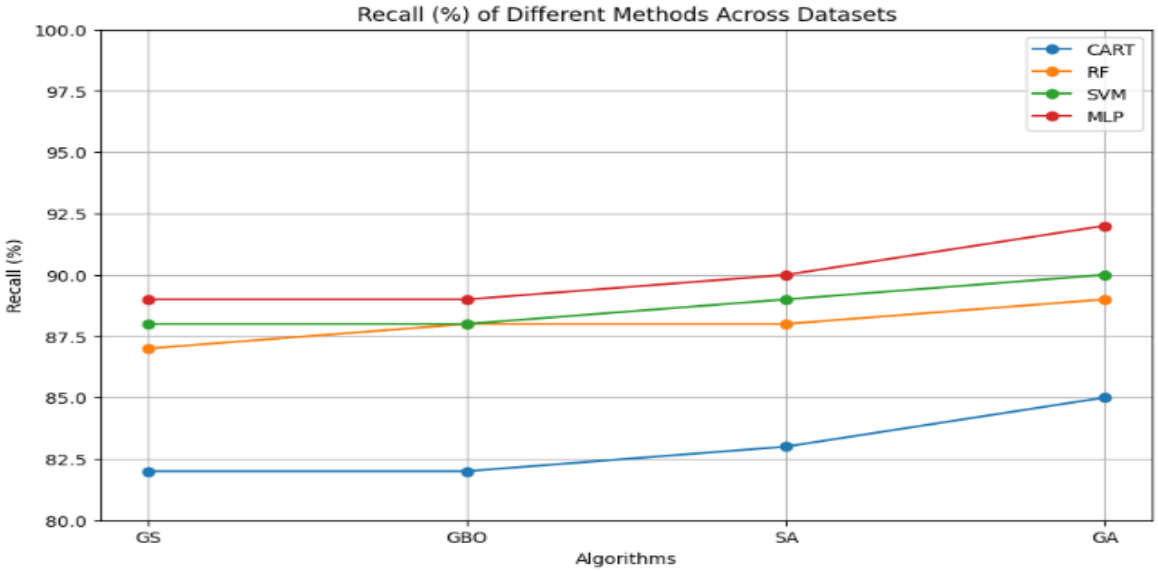


Table 8 compares recall (%) of CART (82-85%), RF (87-89%), SVM (88-90%), and MLP (89-92%) using GS, GBO, SA, and GA. MLP consistently achieves the highest recall, with GA providing the highest scores across all algorithms, indicating effective optimization for enhancing recall performance.

Table.9. Performance analysis of Optimized ML algorithms of F1-Score

	GS	GBO	SA	GA
F1-Score (%)				
CART	85	86	87	89
RF	90	92	92	93
SVM	91	92	93	94
MLP	92	93	94	96

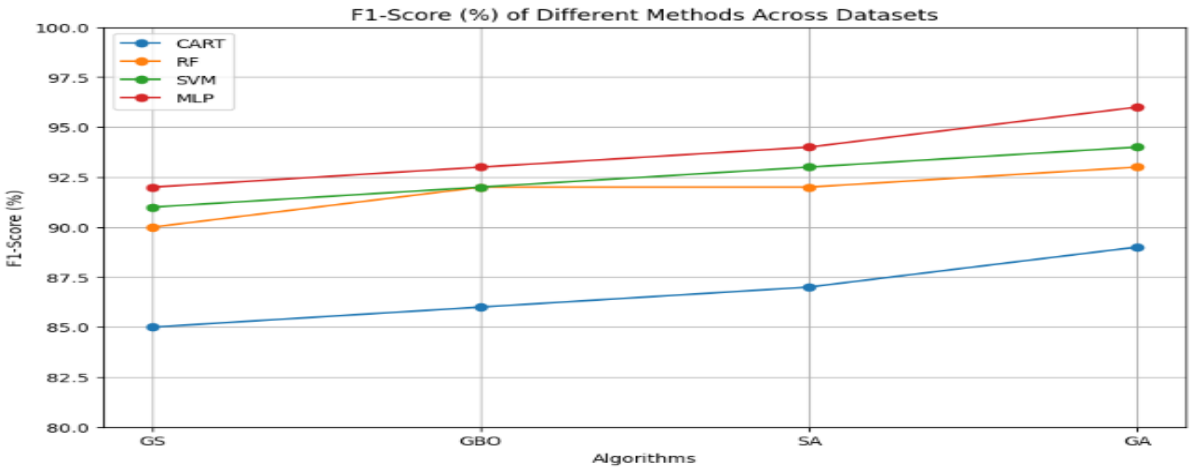


Table 9 compares F1-score (%) of CART (85-89%), RF (90-93%), SVM (91-94%), and MLP (92-96%) using GS, GBO, SA, and GA. MLP consistently achieves the highest F1-scores, with GA yielding the highest scores across all algorithms, demonstrating effective optimization for enhancing classification performance.

7. Conclusion

In today's digital landscape, intrusion detection systems (IDS) are vital for protecting computer networks against a variety of cyber-attacks. This study focused on evaluating IDS performance using meta-heuristic and ML algorithms with the NSL-KDD dataset. The results highlighted that combining these algorithms yielded superior outcomes in accuracy, precision, recall, and F1-score compared to using them individually. Specifically, the MLP classifier optimized with Genetic Algorithm (GA) achieved the highest accuracy of 96%.

Future research can enhance intrusion detection systems by developing datasets that accurately reflect current cyber-attack trends. Additionally, exploring the integration of ML and DL algorithms could further improve IDS effectiveness. Evaluating the proposed
algorithm's performance across different datasets will be essential to validate its robustness and applicability in diverse network security scenarios.

References:

1. M. Alizadeh, S. E. Mousavi, M. T. H. Beheshti and A. Ostadi, "Combination of Feature Selection and Hybrid Classifier as to Network Intrusion Detection System Adopting FA, GWO, and BAT Optimizers," IEEE, 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS), Tehran, Iran, Islamic Republic of, 2021, pp. 1-7.
2. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/
3. Vanin, P.; Newe, T.; Dhirani, L.L.; O'Connell, E.; O'Shea, D.; Lee, B.; Rao, M. A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning. Appl. Sci. 2022, 12, 11752.
4. Khraisat, A., Gondal, I., Vamplew, P. et al. Survey of intrusion detection systems: techniques, datasets and challenges. Springer, Cybersecur 2, 20 (2019).
5. ElDahshan, K.A.; AlHabshy, A.A.; Hameed, B.I. Meta-Heuristic Optimization Algorithm-Based Hierarchical Intrusion Detection System. Computers 2022, 11, 170.
6. Yang, C., Wang, X., & Zhao, S. (2021). Hybrid feature selection and normalization for intrusion detection system. IEEE Access, 9, 36401-36409.
7. Garg, S., Kumar, V., & Kumar, V. (2020). Comparative analysis of normalization techniques for network intrusion detection. International Journal of Electrical and Computer Engineering, 10(6), 6106-6115.
8. Hu, X., Wang, Z., Chen, Y., & Zhang, J. (2021). Deep learning-based intrusion detection for network security using NSL-KDD dataset. PeerJ Computer Science, 7, e503.
9. Alazab, M., & Venkatraman, S. (2021). An Ensemble Model for Intrusion Detection System Using Machine Learning Techniques. In Proceedings of the International Conference on Machine Learning and Data Engineering (pp. 9-19). Springer.

10. Chen, Y., Zheng, Y., Liu, Z., & Yang, Y. (2021). Neural Tree Network based Deep Learning Model for Intrusion Detection. IEEE Access, 9, 44402-44411.
11. Huang, T., Chen, L., & Zhang, H. (2021). A Hybrid Deep Learning and Machine Learning Method for Network Intrusion Detection. IEEE Access, 9, 30247-30256.
12. Ahmad, W., Bilal, M., & Han, K. (2020). Deep learning for network intrusion detection: a review. Neurocomputing, 417, 321-345.
13. Zeng, Z., Zhang, Y., Li, L., & Huang, Z. (2020). Feature Selection Based on Mutual Information and Grey Wolf Optimizer for Intrusion Detection. IEEE Access, 8, 115586-115594.
14. Aber, M., Ali, M., & Atta, A. (2020). Hybridization of SVM and Firefly algorithm for enhancing the performance of intrusion detection system. Journal of Ambient Intelligence and Humanized Computing, 11(10), 4147-4159.
15. Shalaginov, A., Osin, A., & Sukhov, A. (2018). Hybrid Intrusion Detection System based on Artificial Bee Colony algorithm and SVM. Journal of Information Security and Applications, 42, 67-79.
16. Zaki, A., El-Bahnasawy, A., & El-Kassas, S. (2018). A comparative study of machine learning algorithms for intrusion detection system. International Journal of Advanced Computer Science and Applications, 9(3), 119-124.
17. Ghosh, S., & Das, S. (2019). Hybrid Approach of Random Forest and AdaBoost Algorithm for Intrusion Detection System. International Journal of Advanced Computer Science and Applications, 10(2), 126-132.
18. Kumar, S., & Singh, K. (2019). Hybrid PSO-SVM Based Intrusion Detection System Using NSL-KDD Dataset. International Journal of Advanced Research in Computer Science, 10(4), 279-284.
19. Kumar, S., & Singh, K. (2018). A hybrid approach of support vector machine and genetic algorithm for intrusion detection system. International Journal of Computer Science and Information Security, 16(6), 9-15.
20. R. -F. Hong, S. -C. Horng and S. -S. Lin, "Machine Learning in Cyber Security Analytics using NSL-KDD Dataset," 2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), Taichung, Taiwan, 2021, pp. 260-265.
21. M. Srikanth Yadav. and R. Kalpana., "Data Preprocessing for Intrusion Detection System Using Encoding and Normalization Approaches," IEEE, 11th International Conference on Advanced Computing (ICoAC), Chennai, India, 2019, pp. 265-269.
22. M. Choubisa, R. Doshi, N. Khatri and K. Kant Hiran, "A Simple and Robust Approach of Random Forest for Intrusion Detection System in Cyber Security," 2022 International Conference on IoT and Blockchain Technology (ICIBT), Ranchi, India, 2022, pp. 1-5.
23. M. Khodaskar, D. Medhane, R. Ingle, A. Buchade and A. Khodaskar, "Feature-based Intrusion Detection System with Support Vector Machine," 2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), Pune, India, 2022, pp. 1-7.
24. M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), Dhaka, Bangladesh, 2014, pp. 1-6.
25. A. S. Ahanger, S. M. Khan and F. Masoodi, "An Effective Intrusion Detection System using Supervised Machine Learning Techniques," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1639-1644.
26. Kaveh M, Mesgari MS. Application of Meta-Heuristic Algorithms for Training Neural Networks and Deep Learning Architectures: A Comprehensive Review. Neural Process Lett. 2022 Oct 31:1-104.