Cluster Computing
https://doi.org/10.1007/s10586-024-04673-3

PSO-ACO-based bi-phase lightweight intrusion detection system combined with GA optimized ensemble classifiers

Arpita Srivastava[1] · Ditipriya Sinha[1]

Received: 16 April 2024 / Revised: 28 June 2024 / Accepted: 6 July 2024

Abstract
Features within the dataset carry a significant role; however, resource utilization, prediction-time, and model weight are increased by utilizing high-dimensional data in intrusion-detection paradigm. This paper aims to design a novel lightweight intrusion detection system in two phases utilizing a swarm intelligence-based technique. In 1st-phase, essential features are selected using particle swarm optimization algorithm by considering imbalanced dataset. Ant colony optimization algorithm is utilized in 2nd-phase for extracting information-rich and uncorrelated features. Additionally, genetic algorithm is employed for fine-tuning each detection model. Proposed model's performance is evaluated on different base and ensemble classifiers, and it is observed that xgboost achieves best accuracy with 90.38%, 92.63%, and 97.87% on NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets, respectively. The proposed model also outperforms other traditional dimensionality reduction and state-of-the-art approaches with statistical validation. This paper also analyses objective function of each metaheuristic algorithm used in this paper, applying convergence graphs, box, and swarm plots.

Keywords Cyber-security · Particle swarm optimization (PSO) · Ant colony optimization (ACO) · Genetic algorithm (GA) · Lightweight intrusion detection system · Feature optimization

1 Introduction

In the modern era, the Internet plays a vital role in connecting individuals worldwide. With the vast growth of the Internet, the chance of cyber-attacks increases and harms individuals at the organization and personal levels. Cyber-attackers exhibit high proficiency in exploiting vulnerabilities and causing harm to individuals. This harm covers a wide range of consequences, which include data breaches, online harassment, financial losses, intellectual property theft, cyberbullying, and disruptions to essential services like healthcare. Additional resources are being utilized and assigned to defend against these cyber-attacks or abnormal behavior in the network [1]. For this reason, cyber-security is gaining popularity and is necessary to protect organizations and individuals from cyber-attacks. Various network security measures have been proposed to mitigate these cyber-attacks, including firewalls, antivirus software, and malware programs, which serve as an initial line of defense [2]. Still, these security measures cannot properly protect organizations and individuals, especially from the contemporary cyber-attacks on the network [1].

An intrusion detection system is a security system that monitors, analyses the network traffic, and compares it with predefined patterns. If the match (or mismatch) happens between the observed traffic and predefined patterns, an alert signal is generated (based on the matching or mismatching criteria) and sent to the network administrator to take appropriate action. Based on the detection methods, two types of intrusion detection systems have been developed: signature-based intrusion detection systems (SIDS) and anomaly-based intrusion detection systems (AIDS). SIDS compares and analyses the observed network traffic with the pre-defined signature of the malicious behavior stored in the database and triggers an alarm signal when malicious traffic is detected in the network. It depends on the signature of the attack traffic behavior and fails for a

✉ Arpita Srivastava
arpitas.ph21.cs@nitp.ac.in

Ditipriya Sinha
ditipriyasinha87@gmail.com

1 Department of Computer Science & Engineering, National Institute of Technology Patna, Patna Bihar, India

new type of attack or zero-day attacks. On the other hand, AIDS overcomes the issues of SIDS by creating a baseline model that depends on the behavior of normal network traffic. If the observed network traffic deviates from the usual baseline behavior of the model, an alarm is generated and sent to the network administrator. It is generally useful in detecting novel categories of attacks or zero-day attacks.

In this paper, an Intrusion detection system model is designed using an intelligence approach. Most of the traditional intrusion detection system datasets are highly imbalanced and contain many features that significantly degrade the attack detection performance of the system [3]. The learning models with these many features take a long time, resulting in worse classification results [4]. Moreover, the model's weight is one of the most crucial factors influencing the accuracy and efficiency of any intelligent model [5]. It depends upon the number of features present in the data, which is fed as an input to the model. Efficient resource (or power) consumption is a significant concern for the networks of the IoT, where numerous devices are interconnected to each other [6]. Due to the constraints of limited resources, it is necessary to utilize efficiently the available resources. To mitigate the aforementioned resource constraint problems, there is a significant necessity for designing of lightweight intrusion detection system that can detect anomalies with high accuracy by utilizing resources efficiently [7]. Most of the traditional IDS designs do not consider the weight of the IDS model and are heavily weighted in nature [8]. It is important to note that the model's weight is minimized by extracting the most appropriate features from the network data, which enhances the attack detection accuracy and reduces the detection model's prediction time and false alarm rate (FAR).

2 Research gap

State-of-the-arts commonly employ unsupervised techniques for selecting a substantial number of features in IDS design, which reduces the attack detection rate. Many studies [4, 9–15] also overlook the imbalanced nature of datasets when selecting relevant features by using accuracy and error rate in the fitness function. Furthermore, the existing models [10, 16–18], are evaluated by applying only one classifier, which may lead to biased performance. On the other hand, several works proposed by [4, 9–11, 14, 15, 19–23] neglect addressing the hyperparameter tuning of the detection models. As a result, the attack detection rate of the model is very low. Additionally, fine-tuning some existing detection models such as [5, 6, 18, 24, 25], often rely on grid or exhaustive search and random search techniques which have high computational cost and lack of interpretability.

In some state-of-the-art, it is revealed that the utilization of an exhaustive search for feature selection, which examines all possible feature combinations and selects the best result, is an extremely naïve method with very high time complexity. Most of them are NP-hard problems due to the high computational cost and processing time. Metaheuristic algorithms solve NP-hard problems and complicated optimization issues. Rather than obtaining exact solutions, these algorithms (which are suboptimal) find appropriate solutions in a reasonable amount of time [4]. Traditional intrusion detection systems (IDS) are designed using fuzzy algorithms, genetic algorithms, swarm intelligence algorithms, data mining, machine learning, and deep learning models. Most of them overlook the significance of the weight of the IDS model, which leads to computationally intensive and resource-heavy models.

The aforesaid problems motivate to design a Lightweight Intrusion Detection System, applying swarm intelligence-based techniques and genetic algorithms with machine intelligence approaches. The primary objectives of the proposed model are as follows: (i) Saving attack detection time with attack detection accuracy, (ii) Compressing data features, (iii) Reducing the curse of dimensionality by utilizing the combination of PSO and ACO for feature selection. It is a powerful and efficient swarm intelligent approach to addressing high-dimensional data and improving the performance of various intelligence models, and (iv) This paper employs the genetic algorithm for optimizing the hyperparameter of the detection model using the weighted f1-score as the fitness function. This approach identifies the optimal hyperparameters and evaluates the model using multi-class classification.

Figure 1 outlines the proposed model's block diagram, which is divided into four major modules: data preprocessing, Bi-Phase feature optimization, Hyperparameter tuning, and Classification.

2.1 The key contributions of the proposed approach

The main contributions of this paper are outlined as follows:

(i) Bi-phase swarm intelligence-based feature optimization: An improved Bi-Phase swarm intelligence-based feature optimization technique is proposed to reduce the number of features in the data with the objective of designing a lightweight IDS.

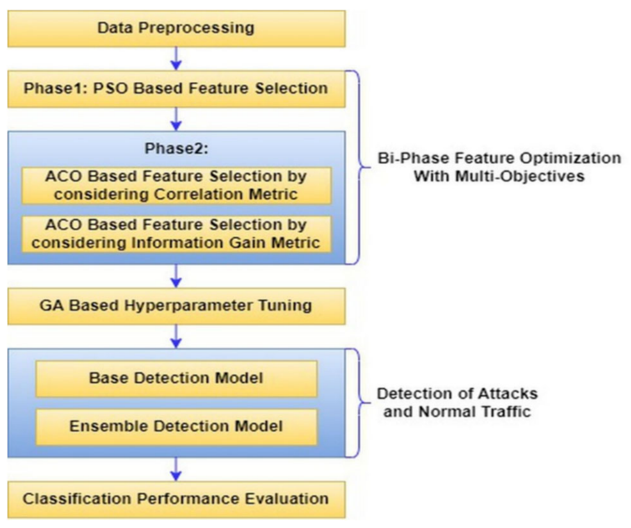

Data Preprocessing
Phase1: PSO Based Feature Selection
Phase2:
ACO Based Feature Selection by considering Correlation Metric
ACO Based Feature Selection by considering Information Gain Metric
Bi-Phase Feature Optimization With Multi-Objectives
GA Based Hyperparameter Tuning
Base Detection Model
Ensemble Detection Model
Detection of Attacks and Normal Traffic
Classification Performance Evaluation

Fig. 1 Block diagram of proposed model

(ii) GA-based optimization of detection models: To fine-tune various detection models (used in this paper), a nature-influenced genetic algorithm is applied to provide each model with the optimal hyperparameter values.

(iii) Classification module: In classifying different malicious and normal traffic, two categories of detection models are applied to assess the effectiveness of the proposed lightweight IDS. Base detection models (such as DT, KNN, SVM, logistic regression, DNN, and CNN) are utilized in the category-1 classification, while ensemble detection models (such as RF, xgboost, lightgbm, catboost, majority, and mean voting) are used in the category-2 classifications.

2.2 Paper structure

The rest of the paper is organized as follows: Sect. 3 analyses the related works, followed by the research objective of this paper in Sect. 4. After that, Sect. 5 broadly discusses the proposed methodology, followed by Sect. 6, which outlines the experimental results and discussion. Last but not least, Sect. 7 concludes the paper by discussing the limitations and future direction of the work.

3 Related works

This section analyses the state-of-the-art works in the field of intrusion detection systems. The paper mainly focuses on designing a lightweight IDS model with fewer features and aims for accurate classification among several malicious and normal network traffic. Therefore, only those state-of-the-art works are considered that emphasize the feature optimization of the data for lightweight IDS design. Additionally, this paper considers hyperparameter optimization of the detection models for accurate classification performance. In the existing works, feature optimization is performed utilizing various metaheuristic and non-metaheuristic algorithms. Especially for feature optimization, the related works in this paper are divided into two subsections: non-metaheuristic feature optimization algorithm-based and metaheuristic feature optimization algorithm-based. This paper surveys one separate subsection for the hyperparameter optimization of detection models. Related work is structured into a total of three main subsections according to the feature optimization approaches and hyperparameter-tuning techniques such as non-metaheuristic feature optimization algorithms-based, metaheuristic feature optimization algorithms-based, and hyperparameter tuning of detection model-based. This section concludes by demonstrating the superiority of the proposed model compared to the existing state-of-the-art.

3.1 Non-metaheuristic feature optimization algorithm-based

Several traditional methods are used to reduce the data's dimensionality, including filter, wrapper, embedded, and feature extraction techniques like PCA, ICA, LDA, t-SNE, and autoencoder. This subsection describes selecting information-rich features from the data in the IDS paradigm without utilizing metaheuristic algorithms.

Chebrolu et al. [26] have built a lightweight IDS that can efficiently and effectively detect intrusions in the classification process. Irrelevant and redundant features are eliminated from the data by implementing markov blanket and decision tree models. Following this, bayesian networks and CART algorithms and an ensemble of bayesian networks and CART algorithms are employed in constructing a lightweight IDS model. The bayesian network utilized here requires either an O(n^2) CI test in special or an O(n^4) CI test in general cases where n denotes the number of domain variables. The outcomes of the model with reduced features are 19.70s and 10.10s average training and testing time, respectively, with only 88.84% average accuracy. Li et al. [27] have used a wrapper-based feature selection approach to build lightweight IDS. Modified linear SVM and modified random mutation hill-climbing (*RMHC*) approaches are used in the proposed wrapper-based feature selection method. Decision tree algorithm is utilized for the classification where nodes of the decision tree contain linear SVMs. Best feature subsets are selected separately for each attack category in the KDD CUP 1999 dataset. Two methods are compared in terms of processing time in such a way that RHMC takes 1.5 h to process U2R attacks, whereas modified RHMC processes it in under 1 h. Mukherjee & Sharma [20] have proposed a vitality-based feature selection method on the NSL-KDD [28] dataset, and out of a total of 41 features, 24 best features are selected. The process of the models is that pre-defined accuracy, average TPR, and RMSE values are considered threshold values. Each feature in the original data is removed, and the performance is checked to see whether it increases, decreases, or remains constant. Here, a sequential search is performed, and the same steps are repeated for each feature (41 times). The importance of each feature is examined at a time based on the pre-defined threshold value. Thus, both time and space complexities of the proposed method take O(n). The accuracy obtained by the proposed method is 97.78%, and the time taken to build the model is 9.42. In this case, the primary limitation is the low true positive rate (TPR) value for the U2R attack class, as well as the lack of addressing hyperparameter tuning. Additionally, the model is evaluated on only one traditional dataset and within a very complex framework. The high complexity of the proposed framework results in high resource utilization.

Li et al. [29] have proposed a three-phase model that includes data preprocessing, feature selection, and anomaly detection. L2 regularization is employed in the preprocessing phase, and feature selection is performed using the random forest with affinity propagation clustering algorithm for the feature grouping. Auto-encoder is used at the anomaly detection phase, where average RMSE is utilized to measure the error rate of multiple auto-encoders. Despite the affinity propagation technique being advantageous in classifying massive amounts of data, it is very difficult with an O(n^2log$n$) complexity (in terms of time as well as space) where n is the number of instances in data. The detection time and recall rate achieved by the *AE-IDS* model are 2493.83s (which is very high) and 61.90%, respectively, on the brute force—web dataset. Kunhare et al. [30] have designed the IDS model, where features are selected by combining the wrapper method with the filter. Random forest algorithm is utilized for calculating the importance of each feature in the data. It is observed that the random forest algorithm reduces the number of features from 41 to 10, and the performance of the model is best utilizing the PSO-based classification algorithm. Based on the experimental results, it is found that the proposed framework has a 99.26% detection rate, 99.32% efficiency, and low computing complexity. The main limitation here is that the data imbalance problem is not considered, and hyperparameter optimization of the model is not explored. Gu & Lu [19] have carried out naïve bayes-based feature embedding in the initial step of the proposed model to transform the data. Subsequently, the transformed train data is used to train the SVM model, and finally, trained models are used to detect intrusions using new test data samples. An optimal accuracy achieved by the detection model is 93.75%, 98.92%, 99.35%, and 98.58% on UNSW-NB15, CICIDS2017, NSL-KDD, and Kyoto 2006 + datasets, respectively. However, the detection performance is evaluated here utilizing binary classification, which considers only three metrics. Moreover, the model's hyperparameter tuning has not been explored. Rao et al. [17] have proposed the two-stage hybrid IDS model. In the first stage, an unsupervised sparse auto-encoder is employed for the extraction of the features, and in the second stage, the deep neural network is utilized to classify different attacks. Auto-encoder is utilized in the first stage of the proposed model for the feature extraction, and classification among different attacks is performed using the deep neural network model. In the classification stage, only one deep neural network model is used. Here, the paper does not deal with the hyperparameter optimization of the deep neural network model. The optimal performance in terms of efficiency is achieved with ten features for the KDD-CUP99 and NSL-KDD datasets and 11 features for the UNSW-NB15 dataset. The accuracy and detection rate of the proposed model are 99.03% and 99.48%, respectively, on the KDD-CUP99 dataset. However, here, a standard technique for hyperparameter tuning is not discussed, and detection performance is evaluated on one detection model, which leads to biased results. Kunhare et al. [31] have explored the effectiveness of port scanning methods for obtaining the IP addresses of networked hosts that are vulnerable to attack. The attacker's initial step to launch a targeted cyber-attack is employing the port scanning technique. The snort IDS tool is also analyzed, including its architecture, installation process, file configuration, and detection approach. Furthermore, real-time network traffic implementation of the different variants of DoS attacks is demonstrated.

Li et al. [16] have introduced a *Hierarchical and Dynamic Feature Extraction Framework (HDFEF)* for designing network intrusion detection systems (NIDS). This approach considers multiple network flow packets to comprehensively define network activity. Here, the optimal performance is best achieved by combining HDFEF with the LSTM with focal loss instead of cross-entropy loss. The experiment is performed up to 40 epochs, which achieves an accuracy of 99.75%, and the time for one epoch taken by the model is 145.17s. Zhao et al. [21] have proposed a three-phase framework that includes data pre-processing, dimensionality reduction, and weighted stacking of classifiers with the aim of improving accuracy and efficiency. For dimensionality reduction, correlation-based feature selection with the deferential evolution algorithm (*CFS-*

DE) approach is utilized, and classification is performed based on weights given to the base classifiers. The twenty best features are selected by the proposed *CFS-DE* model, which achieves an accuracy rate of 87.34% and 168.93s on the KDDTest + dataset. Here, the model increases the computation time cost because of the extra overhead of the base model's weight calculation. Gupta et al. [32] have used an ensemble model for the detection of brain tumors and classification of the cancer stage, which is either pituitary, meningioma, or glioma cancer. The proposed model is majorly divided into three modules: data preprocessing, tumor detection, and classification. Preprocessing of the data is performed, which includes increasing the contrast of the MRI images, followed by image augmentation with the help of *CycleGAN*. For the detection of the brain tumor in the second module, modified inception ResNetV2 is employed, which gives a binary output of either yes or no. If the tumor is detected, then the tumor stage is classified in the third module. It is achieved by combining modified Inception ResNetV2 and random forest algorithms, which yield either of the three categories, including pituitary, meningioma, and glioma cancer, with an accuracy of 98%. Azimjonov & Kim [5] have developed a lightweight IDS capable of detecting various cyber-attacks while addressing the challenges posed by limited computational resources and high-dimensional data within the IoT environment. The number of features is reduced by utilizing four feature selection methods: importance coefficient, backward sequential, forward sequential, and correlation coefficient with ridge regressor model. A stochastic gradient descent-based classifier is used for the classification. Here, the worst-case time complexity of the method is O(n^2) with a 92.69% accuracy rate on average, where the data contain the 'n' number of features. The run time of the model with a reduced feature set utilizing the backward sequential algorithm is 2.5 ms on the N-BaIoT-2021 dataset. The backward sequential algorithm takes the worst time (7.21 h) to select the appropriate features on the N-BaIoT-2021 dataset. Here, exactly the six best features are selected by the method. Here, the main limitation is that grid-search-based hyperparameter optimization is performed, which increases the complexity of the model as a result, takes high resource utilization. Additionally, only traditional feature selection methods are explored. Dhanya & Chitra [33] have designed a framework for the IoMT environment to reduce resource utilization with comprehensive time. Auto-encoder is used to decode the features while the xgboost classifier detects the malware. The hyperparameters of the auto-encoder are tuned using the random search, and the hyperparameters of the xgboost classifier are tuned using the genetic algorithm. The adaptive mutation used in the genetic algorithm enhances the search space, hence making it complex in terms of space. Here, cohen's kappa metric is employed for statistical validation of the model, and it achieves an accuracy of 98.66%, while cohen's kappa is 96.37%.

3.2 Metaheuristics feature optimization algorithm-based

This subsection discusses several metaheuristic methods for feature selection within the data, including GA, tabu search, MFO, and RSA. It selects the crucial and information-rich features from the data in the IDS paradigm by applying metaheuristic algorithms.

Khammassi & Krichen [9] have proposed a genetic algorithm combined with a logistic algorithm-based wrapper approach for the feature selection. The model is divided into three stages: preprocessing, feature selection, and classification. The optimal subset of features is obtained from the feature selection stage, where the GA-LR-based approach is applied. The complexity of the proposed genetic algorithm depends upon the fitness function here. The aim is to maximize the fitness function within the genetic algorithm, which is a combination of accuracy (directly proportional to fitness function) and the number of features in the subset (inversely proportional to fitness function). The number of features and accuracy pair in this work is as follows: (18, 99.90%) and (20, 81.42%) on the KDD99 and UNSW-NB15 datasets respectively. Vijayanand et al. [10] have proposed a method that uses a genetic algorithm for the feature selection, and multiple support vector machines are used to detect multiple attacks and build IDS for wireless mess networks. Multiple SVM classifiers are arranged linearly, and each classifier is dedicated to each attack and normal class in the input dataset. Performance achieved in this paper is 95.7% accuracy, with 1.90% FPR, 0.5486s average training time, and 0.0023s average testing time on the WMN dataset. The time complexity of the proposed method is O(L*S), where L represents the length of the candidate solution, and S denotes the size of the population in the genetic algorithm. Mohammadi et al. [14] have designed an IDS model that combines filter and wrapper-based methods for feature grouping and feature selection. In the data pre-processing phase, transformation, discretization, and normalization-based techniques have been applied. A filter method-based linear correlation coefficient is used for feature grouping and is called *FGLCC*. Additionally, the authors have combined the cuttlefish algorithm (*CFA*) to improve the performance of the model. Here, the fitness function is a combination of detection rate and false positive rate. The main aim of this paper is to enhance the fitness score of the candidate solution as much as possible. To classify between intrusive activity and normal flow, decision tree classification algorithm is employed here. The efficiency of the proposed *FGLCC-CFA* with the ten best features in terms of detection rate, accuracy rate, FPR, fitness, model building, and testing time are as follows: 95.23%, 95.03%, 1.65%, 95.46%, 83.28s, and 43.50s respectively on KDD CUP99 dataset. Nguyen & Kim [22] have used the genetic algorithm along with KNN and the fuzzy c-means clustering algorithm for the optimal feature subset selection and feature improvement, respectively. After selecting the optimal feature subset, the optimal model is selected employing GA-CNN along with fivefold cross-validation. Only the training dataset is used in these two aforementioned steps. After that, the model is validated using a validation set, and deep features are extracted by the CNN model. Finally, classification models such as KNN, RF, BG, and BS, along with fivefold cross-validation, are utilized to evaluate the performance of the proposed NIDS. Here, the model achieves an accuracy of 98.2%, FPR of 0.5%, and TPR of 95.4% with the 33 best features on the KDDTest-21 dataset. The main limitation is that the data imbalanced issue is not addressed, has a very high computational time, and is evaluated on only one dataset.

Khammassi & Krichen [13] have used the combination of NSGA2 and logistic regression classifier for the feature selection in network intrusion detection. Two schemes are utilized to test the proposed feature selection approach, which includes multinomial logistic regression corresponding to multiple classes and binary logistic regression, which corresponds to each attack class separately in the dataset. Three different decision tree algorithms, such as the C4.5 decision tree, naïve bayes tree, and random forest, are applied to test the performance of the model. The main limitation of this work is that the proposed multi-objective function includes accuracy, which is not a better metric for evaluating the candidate feature subsets in case of an imbalanced dataset. The value of the weighted mean CPU time of the proposed NSGA2-BLR is approx. 20000s while NSGA2-MLR takes nearly 200000s on the CIC-IDS2017 dataset. Performance of the model in terms of accuracy, detection rate, FAR, and the number of features on the UNSW-NB15 dataset is as follows: 94.90%, 55.73%, 0.72, and 8 to 17 respectively, for the binary class, while 66%, 64.90%, 3.85%, and 11 respectively for multi-class. Nazir & Khan [11] have applied the tabu search algorithm to select an optimal subset of features, and the random forest is used to evaluate the performance of the model. In the fitness (cost) function of the tabu search, a combination of multiple objective functions have been used such as error rate, false positive rate, and number of features in the candidate solution. The main aim is to minimize the fitness function for each candidate solution as much as it can be. Here, the feature space is decreased by greater than 60% because tabu search is not hampered by the complexity of the search space, and the time complexity is decreased by up to 40% with a random forest classifier. The proposed method achieves 83.12% accuracy and 3.70% FPR, with 16 optimal features, resulting in a 12.18% cost for the UNSW-NB15 dataset. Halim et al. [15] have designed an IDS that performs feature selection by applying the genetic algorithm for the designing of IDS. The fitness function in the genetic algorithm uses the combination of the correlation metric and accuracy. The correlation metric employs the different combinations of feature sets in the original dataset for the specific candidate feature subset. Moreover, accuracy is not an appropriate metric for evaluating the candidate feature set in an imbalanced dataset, and features are selected in an unsupervised manner. The roulette wheel selection function is applied for the selection of the parent solution in the genetic algorithm. After applying the genetic algorithm, the number of features is reduced up to 10. Different machine learning classification algorithms, such as xgboost, SVM, and KNN, are used to detect intrusive and normal traffic. Time and space complexities of the proposed algorithm are $O(g(p * c^2))$ and O $(p * c^2)$, respectively, where g, p, and c represent the number of generations, population size, and length of chromosome within the genetic algorithm. The average accuracy of the model is reported as 98.11%. Ogundokun et al. [34] have applied the PSO algorithm to select the feature and design an IDS model. Subsequently, decision tree and KNN algorithms are utilized to evaluate the feature subset distilled by applying the PSO algorithm. This paper does not consider the imbalanced nature of the dataset. Furthermore, the objective function in the proposed PSO algorithm is not discussed, which is a crucial phase for selecting candidate feature subsets. The model achieves an accuracy rate of 98.6%, a detection rate of 89.6%, and an FPR of 1.1%. The time and space complexities of the model have not been discussed.

Aksu & Aydin [4] have used machine learning techniques to secure CAN Buses. A modified genetic algorithm is utilized to select the 'm' optimal feature according to the k-fold cross-validation. Furthermore, five different classification algorithms are employed as candidate classifiers: decision tree, SVM, KNN, logistic regression, and linear discriminant analysis classifier. The overall run time complexity of the proposed model in the worst case is $O(n^7)$. Here, the hyperparameter of the detection model is not addressed for better accuracy and detection rate. The main limitation is a complex structure with high computational time. Chohra et al. [35] have used the PSO algorithm for feature optimization in the anomaly detection domain. The fitness function employs the ensemble of different machine learning and deep learning classifiers where weighted f1-scores of the ensemble model are selected as the objective function. Subsequently, the selected features are used to filter out original datasets, and a deep learning-based autoencoder model is used for the anomaly detection task. Autoencoder uses the following hyperparameter settings: dropout rate of 0.5 and L2 regularization, categorical cross-entropy, and mean squared error loss functions, which are utilized in the anomaly detection phase. Here, the time complexity of the proposed method is O(k*n*m*log(m)), where n, m, and k represent the number of features, number of samples, and number of trees, respectively. The model is not hampered by space complexity due to the utilization of 128 GB RAM, and it reports an 89.523% accuracy and (28 min + 38s) training time on the UNSW-NB15 dataset. Here, the fine-tuning of only two and three hyperparameters is considered for random forest and xgboost, respectively. Additionally, features are not selected based on the correlation between feature-feature and class-feature pairs. Alazab et al. [36] have designed a network-based IDS (called *CossimMFO*) by utilizing the swarm optimization algorithm combined with the machine learning algorithm for classification. A modified moth-flame optimizer (MFO) algorithm (a wrapper method) is proposed for selecting the best feature subsets, and a decision tree algorithm is applied for the classification task. Only four of the best features are selected in the NSL-KDD and UNSW-NB15 datasets, and five features are selected in the KDD-CUP99 dataset. The model achieves an accuracy rate and TPR of 97.8% and 99.6%, respectively. Dahou et al. [37] have designed an IDS for IoT security by utilizing deep learning and metaheuristic algorithms. CNN is utilized to extract the relevant features from the IoT data, followed by an enhanced reptile search algorithm (RSA) to select the information-rich features. The fitness function of the RSA uses a combination of error rate and ratio of selected features. The error rate is computed by utilizing the KNN-based classification algorithm. The main limitation of the paper is that the convergence rate of the proposed RSA algorithm is very low. The time complexity of the model is O(n * (t * d + 1)), where n, t, and d indicate the number of candidate solutions, max. number of iterations and the dimension of each candidate solution, respectively. The model achieves a 92.04% accuracy rate for multi-classification on the KDD99 dataset.

Kunhare et al. [38] have proposed a model that is separated into four major modules, including data pre_processing, feature_selection, classification, and finally, optimization. For selecting the best subset of features from the NSL-KDD data (which originally contains 41 features), here genetic algorithm is applied (which reduces the number of features to 20). These reduced features are utilized to filter the dataset with only these feature sets. The filtered dataset is used in the classification module, which employs the hybrid method combining supervised and unsupervised classifiers such as decision trees and logistic regression. In the last module, several metaheuristic algorithms, including GWO, PSO, MVO, and BAT, are applied for optimization. It is observed that the GWO algorithm gives the best accuracy (99.44%), FPR (0.60%), and detection rate (99.36%). Here, the time complexity of the proposed GWO-based algorithm is O(n*logn). Chowdhury et al. [23] have built a network intrusion detection system to identify malicious traffic using the information-rich feature subset. Various combinations of the PSO algorithm, GA algorithm, and threshold correlation (TC) have been explored. PSO and GA algorithms are used to remove the redundant features, and threshold correlation is used to remove the correlated features by setting a certain threshold value. In phase 2 of the classification model, different ensemble models have been employed that best perform at phase 1, including majority voting, mean voting, and catboost. The performance of the model is 73.23% accuracy and 187.125s run time with the SVM classifier and 98.39% accuracy and 5.862s run time with the xgboost classifier for binary and multiclass classification, respectively. Kumar et al. [12] have used the grasshopper optimization-based algorithm to extract the most essential and relevant features from the datasets. *Deep residual convolutional neural networks* are applied to design an IDS for classification, which further optimizes utilizing the gazelle optimization-based algorithm. The aim is to minimize the fitness function for each candidate solution as much as possible. The time complexity of the model is O(Maximum_Iteration * m * (m*d)), where m represents the number of candidate solutions utilized, and d denotes the size of the problem. Here, the model achieves the following results: 99.17% accuracy, 0.87% FAR, 99.08% detection rate, 47s processing time, and 23.01s testing time. Here, the error rate in the fitness function is utilized, which is a very common approach.

3.3 Hyperparameter tuning of detection model-based

Several studies use traditional methods for optimizing the hyperparameter values, such as grid search, random search, and bayesian optimization methods, and some use metaheuristic methods like the firefly algorithm. This subsection outlines the selection of the optimal hyperparameters of the models in the IDS paradigm.

Wazirali [18] has proposed a method that is majorly separated into four phases: data pre-processing, feature selection, classification, and model validation. This paper mainly addresses the zero-day attack problem to reduce model building and model testing time. In this paper, the optimized hyperparameters are as follows: number of neighbors, distance function and weight, and data standardization. The main drawback of this paper is that the model is evaluated only on a single classifier, such as KNN. Furthermore, the hyperparameter of the model is optimized using the exhaustive search technique, which is a computationally extremely inefficient approach (takes exponential time O(n^k)), where n and k are no. of hyperparameter values and no. of hyperparameters respectively. The accuracy and f1-score of the presented framework are 98.49% and 98.43%, respectively. Kunang et al. [24] have separated the proposed architecture into three modules: a data preprocessing module, a deep learning module with hyperparameter optimization, and an attack detection module. Furthermore, hyperparameter optimization is used in the second module to determine the best model. The deep autoencoder is used as the model in the second module for feature extraction, which includes the encoding, decoding, and bottleneck layers. Achieved values of the accuracy rate, training, and run time of the proposed framework are 83.33%, 382.48s, and 0.968s, respectively, with multiclass classification on the NSL-KDD dataset. Batchu & Seetha [25] have used machine learning models to detect DDoS attacks. During data preparation and pre-processing, the data undergo five phases: exploratory analysis, sample balancing with techniques like SMOTE and Tomek, imputing missing/infinite/zero values with median values, feature normalization using a standard scaler, and label encoding for categorical features. Feature selection is performed using a combination of two traditional feature selection techniques, filter and embedded-based, and the model's hyperparameters are optimized by utilizing the grid search technique. The main drawback of the paper is that grid search-based hyperparameter tuning increases the time complexity of the model exponentially (O(n^k)). Here, a gradient boosting algorithm is utilized, which takes O($n * f * n_{trees}$), where f and n_{trees} are a number of features and a number of trees respectively. The performance of the proposed model in terms of accuracy and run time is 99.97% and 40.78s, respectively.

Jovanovic et al. [39] have designed a network intrusion detection system (called *XGBoost-TSFA*) using the improved firefly and xgboost algorithms. The six different hyperparameters of the xgboost algorithm (including eta, max_depth, gamma, colsamplel_bytree, min_child_weight, and subsample) are optimized using the improved firefly algorithm, which enhances the detection capabilities of the IDS. Evaluation of the proposed framework is performed on the UNSW-NB15 dataset. The experiments are performed using a population size of ten with fifteen iterations, and the model is evaluated by applying binary and multi-class classification with 97.49% and 86.96% accuracies, respectively. To enhance the detection abilities, reducing the false positives and false negatives ratio [40] have proposed a NIDS, which uses the xgboost algorithm to identify malicious traffic. To improve the performance, the hyperparameters of the xgboost algorithm are optimized using the modified sine–cosine metaheuristic algorithm. The performance of the proposed model is evaluated utilizing the NSL-KDD dataset and compared with another metaheuristic algorithm based on optimized xgboost and without optimized xgboost algorithm. Kalita et al. [41] have used a drift detection technique to measure the magnitude of the drift in the dynamic or non-stationary environment. Only the hyperparameters of the SVM classifier (C & γ) are discussed, and based on the magnitude of drift, one of three mechanisms is selected. The first mechanism is the introduction of the base optimization algorithm, i.e., the moth flame optimization algorithm (MFO), and random initialization of the algorithm is considered here. In the second mechanism, lightweight-MFO is introduced, which uses the knowledge base for the initialization of the algorithm. In the third mechanism, the knowledge base search space is utilized to achieve the optimal value of the SVM hyperparameters. The execution time at the 10th time instance with and without drift detection module are 17,473.99s and 28,321.6s, respectively, and the average accuracy obtained by the model is 97.5%.

Savanovi et al. [42] have developed an IDS model for the security of IoT devices for healthcare 4.0. Here, the machine learning classification algorithm is utilized along with the metaheuristic algorithms. The modified firefly algorithm is utilized to optimize the xgboost model's hyperparameters. To select the best feature within the dataset, the KNN algorithm is applied with the value of K = 5. As a result, out of 50 features, ten best features are selected. The proposed model is compared with the other eight metaheuristic algorithms such as FA [43], GA [44], PSO [45], ABC [46], ChOA [47], COLSHADE [48], and SASS [49]. The SHAP plot is utilized to analyze the selected features, and statistical validation of the observed results is performed using the p-values at significance levels 0.1 and 0.05. The accuracy and f1-score of the proposed framework are 99.69% and 99.69%, respectively. Six hyperparameters of only xgboost based detection model are tuned. Saheed & Misra [50] have designed an intrusion detection system for IoT security, which considers the average probability of a voting classifier. The dimensionality of the dataset is reduced with the hybrid approach utilizing information gain for feature selection and PCA for feature extraction. The voting classifier uses four machine and deep learning-based base classifiers such as random forest, KNN, decision tree, and multilayer perceptron. The hyperparameters of these base classifiers are optimized using the gray wolf optimizer. The class imbalance issues present in the IoT datasets (such as UNSW-NB15 and BoT-IoT) are handled with the help of SMOTE. The performance attained by the framework in terms of accuracy, detection rate, and FAR is 99.87%, 99.89%, and 1.20%, respectively. Azimjonov & Kim [6] have presented a framework with two main contributions based on implementation and methodology. In the implementation, the dataset preparation portion has been discussed, such as balancing imbalanced data, removing duplicate records, transforming categorical data into numerical data, dealing with missing values, and splitting the data into train and test sets. Four feature selection techniques have been applied in the methodology: forward sequential, backward sequential, importance coefficient, and correlation coefficient with linear SVM classifier. Moreover, the hyperparameters of the ridge regressor and LSVM classifier have been tuned utilizing the grid search-based hyperparameter tuning approach. However, the grid search-based hyperparameter tuning approach is inefficient in terms of computational cost and high-dimensional data. Grid search explores all combinations of the search space, and hence, it takes exponential time (e.g. O(n^k)). Although the model's accuracy is 94.64%, it takes an extremely long training time of 5394.409 ms. Table 1 summarizes the state-of-the-art work by discussing the five major components such as (i) Objective, (ii) Method, (iii) Result, (iv) Advantages, and (v) Limitations.

4 Research objective

This paper proposes a lightweight intrusion detection system to address the aforementioned challenges. The proposed model uses the swarm intelligence-based technique to select the most crucial features from the network traffic dataset. There are three main advantages of using swarm intelligence-based feature optimization techniques, which are as follows: (i) Capability to adjust to the dynamic environment, (ii) Resilience to individual failures, and (iii) Capability to effectively explore a broad solution space. Since the IDS datasets are imbalanced and contain many features, they extensively obstruct the accuracy of attack detection [3]. To account for the imbalanced nature of the dataset, the PSO-based feature selection algorithm incorporates the geometric mean of ensemble models into its fitness function. The metric "geometric mean" is said to be superior to accuracy in dealing with the imbalanced nature of the dataset [51]. Here, feature selection is performed using a supervised approach, which prioritizes the target class when determining the optimal feature subset by introducing the correlation metric and information gain metric into the fitness function in the ACO-based feature selection approach. On the other hand, this paper considers the hyperparameter tuning of the different detection models using the nature-influenced genetic algorithm-based technique, which gives optimized results even in complex and high-dimensional data scenarios. The genetic algorithm-based fine-tuning technique determines the best hyperparameter settings for each detection model after every generation. This search process continues iteratively till an optimized result can be achieved. Moreover, the potency of the proposed lightweight Intrusion Detection System is examined on twelve different detection models, each on three different datasets. A detailed description of the proposed methodology of this paper is given in the following section.

5 Proposed methodology

The Proposed Light-Weight IDS model is mainly developed employing four major modules, which are given as follows: (i) *Dataset Description & Data Preprocessing*, (ii) *Bi-Phase Swarm Intelligence-based Feature Optimization* (Phase1: PSO-based feature selection, and Phase2: ACO-based feature selection), (iii) *Hyperparameter Tuning* (genetic algorithm-based hyperparameter tuning), and (iv) *Classification* (by applying either Base or Ensemble detection models). Figure 2 depicts the overall flow of the proposed model. Algorithm 1 summarizes the complete step-by-step development of the Light-Weight IDS model employing all four major modules discussed above. Table 2 illustrates the abbreviations and their description used in the algorithm.

Table 1 A concise summary of state-of-the-art works

Paper	Objective	Method	Result	Advantages	Limitations
Chebrolu et al. [26]	Designing a lightweight IDS model	Bayesian Networks, CART algorithm, an ensemble of bayesian networks and CART algorithm	The proposed model achieves accuracy for Normal, Probe, DoS, U2R, and R2L attacks 100%, 100%, 100%, 84%, and 99.47% respectively	• Multiclass classification on the KDD Cup 99 dataset	• Real-time implementation is not explored • Only one dataset is used
Li et al. [27]	Developing lightweight IDS	Linear SVM, Random Mutation Hill Climbing, wrapper-based feature selection technique, Decision Tree	Performance is evaluated on the KDD CUP99 dataset; 18 and 8 s is the time consumed on all features and selected features, respectively	• Features are selected for each attack class separately • Multiclass classification is performed, • The ROC curve for each attack class is evaluated	• Wrapper based feature selection technique is computationally inefficient, • Only one dataset is used
Mukherjee & Sharmas [20]	Designing effective and efficient NIDS	Naïve Bayes; Correlation-Based, Information-Based, and Gain Ratio-based feature selection	The proposed *FVBRM* model achieves 97.78% accuracy, and the time taken to build the model is 9.42	• A simple approach is applied • Out of 41 features, 24 optimal features are selected	• One traditional dataset is used • Hyperparameter tuning is not addressed • Low TPR value for U2R attack class • Complex framework
Khammassi & Krichen [9]	Building NIDS with reduced features	Wrapper based with GA, Logistic regression, Decision Tree	Approximately 99.8% and 81.2% accuracies are reported against KDD99 and UNSW-NB15 datasets respectively; 18 and 20 features are selected in KDD99 and UNSW-NB15 datasets, respectively	• Three decision tree algorithms (C4.5, RF, NBTree) are used, • Multi-class classification is performed	• Hyperparameter tuning of the classification model is not addressed, • Weka software is used in the classification phase
Vijayanand et al. [10]	Developing wireless mesh network IDS	genetic algorithm, SVM	83.54% and 95.56% accuracies are achieved on the WMN dataset 512-bit and 1024-bit, respectively	• Multi-class classification is performed • Performance is evaluated on multiple datasets	• Only the SVM classification algorithm is used for evaluating the model, • Hyperparameter tuning is not addressed
Mohammadi et al. [14]	Aiming to design an IDS with reduced features and high accuracy	Filter & wrapper methods, linear correlation coefficient & cuttlefish algorithm, Decision Tree	95.03% accuracy is achieved on the KDD Cup 99 dataset	• Accuracy rate, detection rate, and false positive rate are improved for the proposed *FGLCC-CFA* model • Proposed model (*FGLCC-CFA*) performs better with less number of features (10) compared to *FGLCC*	• Performance is evaluated only on the KDD-Cup 99 dataset, • Not consider the imbalance nature of data • Fine tuning of the classifier is not performed

Table 1 (continued)

Paper	Objective	Method	Result	Advantages	Limitations
Kunhare et al. [30]	Designing IDS with reduced features	RF, PSO, KNN, SVM, LR, DT, Naïve bayes	99.32% efficiency and 99.26% attack detection rate are observed on the NSL-KDD dataset	• High attack detection performance • Ten best features are selected • Computationally efficient	• Hyperparameter optimization of the classifiers is not discussed • Data imbalanced issue is not considered
Li et al., 2020 [29]	Building *AE-IDS* to improve accuracy and decrease training time	Auto-Encoder, Affinity Propagation, Random Forest, Gaussian Mixture Model, K-Means, RMSE, L2 Regularization	Recall rates are 2.99%, 11.32%, 17.22%, and 23.37% for DoS attacks-Hulk, SQL Injection, Brute Force –XSS, and Infiltration datasets, respectively	• The CSE-CIC-IDS 2018 dataset is used in the overall experiment, • Binary classification is performed	• Performance is evaluated on only one dataset, • Recall rate is very low, detection time is high
Nguyen & Kim [22]	Designing NIDS with reduced features	CNN, GA, Fuzzy c-means clustering, Bagging classifier, KNN, and RF	98.2%, 0.5%, and 95.4% accuracy, FPR, and TPR, respectively are achieved on the NSL-KDD dataset	• Feature construction with three-layered architecture • Validation method • Multi-class classification is performed	• Only one dataset is used • Data imbalanced issue is not addressed • High time complexity • Hyperparameter tuning of the classification models is not addressed
Khammassi & Krichen [13]	Designing of NIDS, which reduces computational time	C4.5 DT, Random forest, Naïve bayes tree, Logistic regression	98.99%, 66.00%, and 95.16% accuracies are achieved on NSL-KDD, UNSW-NB15, and CIC-IDS2017 datasets, respectively	• Both multi-class and binary classification are performed • Evaluated on three benchmark datasets • Employing multiple objectives to select appropriate features	• The proposed multi-objective function does not consider the imbalanced nature of the dataset
(Wazirali, 2020) [18]	Designing an IDS for detecting Zero-day attacks with reduced data dimensionality	Semi-supervised approach, PCA, KNN, fivefold cross-validation, one hot encoding, standard scalar	98.87% accuracy is achieved by the proposed model on the NSL-KDD dataset	• Optimal values of number of neighbors (k), distance function, distance weight, data standardization, • Dimensionality of data reduced from 42 to 2	• Only the KNN classification algorithm is used, • Exhaustive search-based hyperparameter tuning is performed, which is computationally very expensive
Rao et al. [17]	A hybrid model is proposed for the intrusion detection	Sparse auto-encoder, DNN	99.98% accuracy and 99.99% detection rate are obtained on the UNSW-NB15 dataset	• The L1-regularization technique is utilized to create an optimized model, • KDDCup99, NSL-KDD, and UNSW-NB15 datasets are used	• No standard technique is considered for the hyperparameter tuning of the classification model; • Only one classification model is utilized

Table 1 (continued)

Paper	Objective	Method	Result	Advantages	Limitations
Nazir & Khan [11]	Designing NIDS with the aim of reducing features, error rate, FPR	Tabu Search, Random Forest	83.12%, and 3.7% accuracy and FPR, respectively are achieved	• 16 best features are selected	• Only one dataset is used for evaluating the performance, • The imbalanced dataset problem is not addressed, • Hyperparameter tuning of the classifier is not addressed
Halim et al. [15]	Designing IDS with a reduced feature set utilizing an unsupervised manner	GA, Xgboost, SVM, KNN	Performance of the proposed approach is 98.94%, 98.90%, and 96.48% on CIRA-CIC-DOHBrw-2020, Bot-IoT, and UNSW-NB15 datasets, respectively	• Multi-class classification is performed • Efficiency of the proposed model is evaluated on three different datasets • Performance with the reduced feature set increases	• Ensembing approach is not utilized • Unsupervised feature selection is performed • The fitness function uses accuracy, that is not a suitable metric for imbalanced data • Hyperparameter tuning of different classifiers is not discussed
Gu & Lu [19]	Building IDS model using SVM and Naïve bayes	Naïve bayes, SVM	93.75%, 98.92%, 99.35%, and 98.58% accuracies are achieved on UNSW-NB15, CICIDS2017, NSL-KDD, and Kyoto 2006 + datasets, respectively	• Performance of the proposed model is evaluated on four traditional datasets	• Binary classification is performed, • Only three metrics (accuracy, detection rate, and FAR) are considered, • Hyperparameter tuning of the model has not been carried out
Batchu & Seetha [25]	Designing a model to detect DDoS attacks with reduced features	Logistic regression, KNN, DT, SVM, Gradient boost, Grid search-based hyperparameter tuning, SMOTE + Tomek, standard scaler, label encoding	99.97% accuracy is achieved by the gradient boost model on the CICDDoS2019 dataset	• High accuracy is obtained, • Very less computation time (40.78 s), • Removal of unnecessary features results in 80 number of new features	• Filter + Embedded based feature selection technique is chosen, which falls under traditional feature selection technique, • Grid search-based hyperparameter optimization takes a very long time to find optimal hyperparameters • Only one dataset is used for evaluating the proposed model
Ogundokun et al. [34]	Designing NIDS for detecting network anomalies utilizing the semi-supervised technique	PSO + DT, and PSO + KNN	Detection accuracy for PSO + KNN is 96.2% and for PSO + DT is 89.6%	• PSO + KNN gives better performance than PSO + DT	• Binary classification is performed • Performance is evaluated on only one dataset (i.e. KDD-CUP 99), • Performance is not evaluated on any deep learning model

Table 1 (continued)

Paper	Objective	Method	Result	Advantages	Limitations
Kunang et al. [24]	Designing NIDS with optimized hyperparameters to classify the attacks	One-hot encoding, min–max scaling, Deep auto-encoder, Deep neural network, Stacked auto-encoder, and auto-encoder, grid and random search for hyperparameter tuning	83.33% overall accuracy is achieved on NSL-KDD testing data	• Multiclass classification is performed on two datasets such as NSL-KDD and CSE-CIC-IDS2018	• Hyperparameters are optimized using the automatic method, which merges two approaches, such as grid and random search, • Hyperparameter optimization using these techniques takes a comparatively very long time
Li et al. [16]	Designing network IDS with hierarchical and dynamic feature extraction structure (*HDFEF*)	CNN, RNN, LSTM, and GRU	F1-scores achieved by *HDFEF* are 99.84%, 99.24%, and 98.49% for CIC-IDS2017, UNSW-NB15, and CSE-CIC-IDS2018 datasets, respectively	• Three datasets are used such as CSE-CIC-IDS2018, CIC-IDS2017, and UNSW-NB15	• Statistical features from multiple network flow traffic are not taken • Performance is assessed using only a limited number of classification models • Complexity is increased
Zhao et al. [21]	Designing IDS with low dimensionality and weighted classifiers	One-hot-encoding, min–max scaler, CFS-DE, Random forest, Xgboost, KNN, and Logistic regression	87.44% and 99.87% accuracies are achieved on NSL-KDD and CSE-CIC-IDS2018 datasets, respectively	• Weights of the base classifier are calculated depending on their classification performance	• Hyperparameter tuning of the classifiers is not addressed
Chohra et al. [35]	Designing of network anomaly detection model with a reduced number of features	Autoencoder, PSO algorithm, Random forest, Xgboost, CNN, NN, Catboost, LightGBM	92.09%, 92.90%, and 97.30% f1-scores are achieved on NSL-KDD, UNSW-NB15, and IoT-Zeek datasets respectively	• Performance is evaluated on three datasets • The AUC value for the Zeek oversampled dataset is 0.990	• Fine-tuning of only two and three hyperparameters is considered for random forest and xgboost classification models, respectively • Correlation-based feature selection is not explored
Aksu & Aydin [4]	Designing IDS for the CAN buses' security	Modified genetic algorithm, SVM, DT, KNN, and Linear discriminant analysis	98%, 96.5%, and 99.3% accuracies are reported against HCRL-car hacking, UNSW-NB15, and CIC-IDS2017 datasets respectively	• Best performance is achieved on 5, 7, and 9 subsets of features, • Both multi-class and binary classification is performed	• Hyperparameter tuning of the classifiers is not addressed
Chowdhury et al. [23]	Bi-phase NIDS is designed with reduced features	PSO, GA, Threshold correlation algorithm, voting classifier, MLP, KNN, DT, RF, and CatBoost	Phase1 and Phase2 have achieved detection accuracies of 99.82% and 99.41% respectively	• Combinations of different algorithms are used for feature selection • Evaluation is performed in two phases • Real-time implementation is performed	• Fitness function used in the PSO and GA is not discussed • Fine-tuning of different classifiers is not explored

Table 1 (continued)

Paper	Objective	Method	Result	Advantages	Limitations
Kalita et al. [41]	Designing an IDS model for the non-stationary domain	SVM, Moth flame optimization algorithm, drift detection technique	97.5% average accuracy is achieved on the NSL-KDD dataset; Overall reduction in computation time	• The optimal hyperparameter values of the SVM classifier, such as C and γ are obtained	• Only hyperparameters of the SVM classifier are optimized, • Hyperparameter optimization of the other classifiers, such as KNN, ANN, DT, RF, etc., are not explored, • Performance is evaluated on only one dataset
Savanovi et al. [42]	Designing IDS for IoT devices for Healthcare 4.0	Modified Firefly algorithm, Xgboost, KNN, SHAP	99.17% accuracy is obtained on the ICU dataset	• Binary and multi-class-classification • SHAP analysis • Statistical validation • 10 best features are selected	• Only the xgboost model is analyzed for hyperparameter tuning • Few hyperparameters (six only) are considered for analysis
Kumar et al. [12]	Designing NIDS to improve the security of the network	One-hot-encoding, Deep residual CNN, Gazelle optimization algorithm, Grasshopper optimization algorithm	99.56%, 99.06%, and 99.12% accuracies are achieved on CIC-IDS2017, UNSW-NB15, and Cicddos2019 datasets, respectively	• Multi-class classification, • Optimizing the hyperparameters of the DRCNN	• Error rate is used in the fitness function, which is a very common approach
Azimjonov & Kim [6]	Designing an accurate and lightweight IDS for IoT networks	Ridge regressor, Linear SVM, Forward sequential, Backward sequential, Correlation coefficients, Importance coefficient	95.66%, 99.48%, and 99.81% maximum accuracies are achieved on KDD-CUP-1999, BotIoT-2018, and N-BaIoT-2021 datasets, respectively	• The feature sets are reduced up to 6 features out of 40, 15, and 115 features for KDD-Cup-1999, BotIoT-2018, and N-BaIoT-2021 datasets, respectively	• Grid search is used for the hyperparameter tuning of the ridge regressor; • LSVM is an inefficient approach in terms of computational cost and for a large number of hyperparameters
Dhanya & Chitra [33]	Building an intelligent system that reduces resource utilization and time in the IoMT environment	Auto-encoder, Xgboost, genetic algorithm, Hyperparameter tuning using random search	98.98% and 98.69% of accuracies are reported against dataset1 (Wustl, 2020) [66] and dataset2 (Shahane, 2021) [67] respectively	• Results are evaluated on two datasets, • Hyperparameter tuning of the models is addressed	• Optimal hyperparameter values of only the Xgboost classifier are given

Table 1 (continued)

Paper	Objective	Method	Result	Advantages	Limitations
Azimjonov & Kim [5]	Designing a lightweight Intrusion detection system for IoT devices	Stochastic gradient descent classifier, feature optimization using importance coefficient, backward sequential, forward sequential, and correlation coefficient, grid search for hyperparameter tuning	The best accuracy achieved by the IDS model is 96.61%, 94.76%, and 98.42% on the KDD-1999, BotIoT-2018, and N-BaIoT-2021 datasets, respectively	• Six best features are selected • Duplicate records are dropped • Three datasets are used such as KDD-CUP-1999, BotIoT-2018, N-BaIoT-2021	• Traditional feature selection approach is utilized • Grid search-based hyperparameter tuning increases the computational complexity • Real-time implementation of the data or implementation of the realistic data is not explored

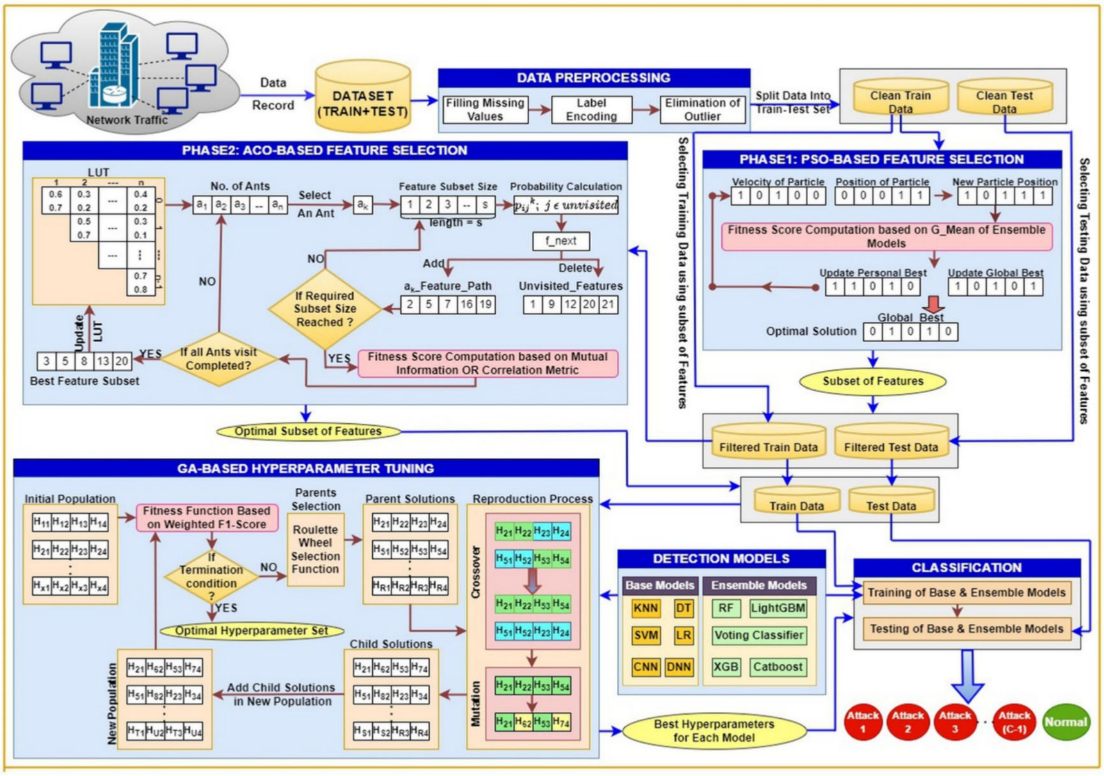


Fig. 2 Overall Working of Proposed Model

Table 2 Algorithm Parameters

Parameter	Description
F_P	A subset of features after applying PSO
F_A	An optimal subset of features after applying ACO
M_B	Base Model
M_E	Ensemble Model
H_B	Best_Hyperparameters
P_B	Classification Report for Base Model
P_E	Classification Report for Ensemble Model

Table 3 Description of datasets used in this paper

Dataset	NSL-KDD	UNSW-NB15	CSE-CIC-IDS2018
No. of Features (except target class)	42	44	79
No. of Training Samples	1,25,973	1,75,341	1,59,784
No. of Testing Samples	22,543	82,332	39,946

Algorithm 1 Proposed model

```
     Input: Original Dataset ($D_{O\ Train}$, $D_{O\ Test}$)
     Output: Lightweight IDS (LIDS) Model
1    procedure LIGHT-WEIGHT INTRUSION DETECTION MODEL
2        begin
3        $D_{C\ Train}$, $D_{C\ Test}$ ← Data Preprocessing ($D_{O\ Train}$, $D_{O\ Test}$)
4        $F_P$ ← Phase1for Feature Selection ($D_{C\ Train}$)
5        $D_{F\ Train}$, $D_{F\ Test}$ ←Filtering($D_{C\ Train}$, $D_{C\ Test}$, $F_P$)
6        $F_A$ ← Phase2 for Feature Selection ($D_{F\ Train}$)
7        $D_{Train}$, $D_{Test}$ ←Filtering($D_{F\ Train}$, $D_{F\ Test}$, $F_A$)
8        $H_B$ ← Hyperparameter Tuning($D_{Train}$, $D_{Test}$, $M_B$ or $M_E$)
9        $P_B$ or $P_E$ ← Classification ($D_{Train}$, $D_{Test}$, $H_B$, $M_B$ or $M_E$)
10       $LIDS_{BEST}$ ← Best Performing Model either $M_B$ or $M_E$ (based on $P_B$ or $P_E$ values)
11       return $LIDS_{BEST}$
12       end
13   end procedure
```

5.1 Dataset description & data preprocessing

The first module of the proposed framework discusses the dataset and the preprocessing steps applied in this paper.

5.1.1 Dataset description

Three most popular Intrusion detection system traditional datasets (such as NSL-KDD [28], UNSW-NB15 [52], and CSE-CIC-IDS2018 [53]) are utilized in this paper. The NSL-KDD [28] dataset is an enhanced form of the KDD Cup'99 dataset [54]. A detailed description of the CSE-CIC-IDS2018 dataset is present at (https://www.unb.ca/cic/datasets/ids-2018.html) [55]. All these datasets are imbalanced. Here, complete training and testing sets of NSL-KDD and UNSW-NB15 are used. Since the CSE-CIC-IDS2018 dataset contains many records, only the samples belonging to a subset of Wednesday traffic are randomly selected in this paper. Table 3 shows a brief description of the datasets used in this paper.

5.1.2 Data preprocessing

Data preprocessing is required for cleaning the dataset as the first step of the proposed framework since the dataset contains null or redundant values, outliers, and categorical values. Here, the data preprocessing process is divided into 3 steps, which include (i) Filling Missing Values, (ii) Label Encoding, and (iii) Outlier Removal. The sequential flow of data preprocessing steps used in this paper is described as follows:

(i) Filling Missing Values: Some of the entries contain "Null" values, which give no information about the detection of attack. Therefore, in this paper, a particular row's 'null' values are filled with a top value of that column.

(ii) Label Encoding: In the datasets used in this paper, some of the features contain categorical values that need to be converted into numerical values. For this reason, the label encoding technique is applied here. By doing label encoding, the values of categorical features can be converted into their corresponding numerical values.

(iii) Outlier Removal: Outlier removal is the process of normalizing or scaling the features in the data to a specific range. In this paper, min–max scaling is applied, and its formula is given in Eq. (1) as follows:

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

Where x is the specific feature's original value, *x.min* is the minimum value of that specific feature, *x.max* is the maximum value of that specific feature, and *x.scaled* is the specific feature's scaled value within the range [0,1].

5.2 Bi-phase swarm intelligence-based feature optimization

To make data with the most efficient utilization for building the model, appropriate and scrupulous selection of the features in the data is one of the important parts that is being addressed in this module. In the context of machine learning or deep learning, the model's weight is highly dependent on the number of features in the data. The weight of the model is highly dependent on the number of features used in training the model. If the data contains high dimensions, the model weight becomes high. Many numbers of redundant and non-informative features in the training data make the model heavyweight. Therefore, the number of features plays a crucial role in designing a lightweight IDS model.

This module is divided into two phases; hence, it is called Bi-Phase Swarm Intelligence-based Feature Optimization. The term swarm intelligence [56] is used because it is influenced by how a group of simple agents works together to solve complicated or intricate problems in social organisms. These fields come under the nature-inspired metaheuristic class of algorithms. A modified version of two algorithms, particle swarm optimization, and ant colony optimization, is applied to the proposed approach. The reason behind introducing two-phase feature optimization is to build the model as light as possible, which can save time and reduce overfitting and overall cost.

However, the field of optimization has seen the emergence of several advanced algorithms such as the reptile search algorithm (RSA) [57], red fox optimization algorithm (RFO) [58], salp swarm algorithm (SSA) [59], butterfly optimization algorithm (BOA) [60], and many others. Despite these advancements, the hybrid use of PSO and ACO remains prevalent. The no free lunch (NFL) theorem [61] asserts that no single optimization algorithm outperforms all others across all possible problems. Its problem-specific nature implies that an algorithm effective for one type of problem may not work well for another. Consequently, there is no universally superior heuristic for all optimization tasks. This underscores the importance of carefully selecting and tuning optimization algorithms based on the unique characteristics of each problem rather than a generalized approach.

Considering the "no free lunch" (NFL) theorem, combining different algorithms or utilizing hybrid approaches can offer a way forward by harnessing the unique strengths of multiple algorithms. For instance, combining PSO and ACO for feature selection tasks offers several advantages over algorithms like RSA, RFO, SSA, and BOA due to their complementary strengths. RSA may suffer from slower convergence and lack of fine-tuning mechanisms, RFO may not possess a robust dual mechanism for balancing exploration and exploitation across diverse problems, and on the other hand, SSA may struggle with exploitation and fine-tuning, and BOA may lack the precise search refinement offered by the combination of PSO's global search and ACO's local search capabilities. PSO offers fast convergence and effective global search, while ACO provides robust exploration and fine-tuning through pheromone trails. PSO excels in quick global search and convergence, while ACO excels in refining the search locally through pheromone-based learning. The combination of these two methods improves exploration and exploitation processes, preserves diversity, as well as adjusts the search process dynamically. This leads to a more resilient and efficient method for feature selection tasks. The following sections, 5.2.1 and 5.2.2, describe in detail the phases of the proposed feature selection process.

5.2.1 Phase1: particle swarm optimization based feature selection

Since the dataset used in this paper is imbalanced, g_mean is considered a more acceptable metric than accuracy for the imbalanced dataset [51]. The main goal of this phase is to extract the most crucial features from the intrusion detection system data based on the imbalanced nature of the dataset. It reduces the complexity and false alarm rate, enhances the attack detection accuracy and interpretability, and yields a more efficient model. Several machine and deep learning-based intelligent detection models are used in ensemble learning techniques. The machine learning models applied in the ensemble technique are decision tree, random forest, xgboost, k-nearest neighbor, support vector machine, lightgbm, and catboost. The deep learning models of ensemble technique are dense neural networks and 1-dimensional convolutional neural networks. The geometric mean is employed in the fitness function, which preserves the imbalanced property of the dataset and gives the best attack detection performance.

Algorithm 2 Phase1 for feature selection

```
    Input: Clean Train Dataset ($D_{C\ Train}$)
    Output: Subset of features ($F_P$)
1   procedure PSO-BASED FEATURE SELECTION
2       Begin
3       c1 ← 1, c2 ← 1
4       r1, r2 ← random (0,1)
5       w ← 0.2
6       swarm_size ← 10
7       $p$← no. of Iterations
8       swarm ← random(particles, swarm_size)
9       velocity ← random(swarm_size)
10      for i in range(swarm_size) do
11          $fitness\_particle_i$ ← Compute_fitness($particle_i$, $D_{C\ Train}$)
12          global_fitness ← $fitness\_particle_i$
13          global_best ← $particle_i$
14          $personal\_best_i$ ← $particle_i$
15          if ($fitness\_particle_i$ > global_fitness) then
16              global_fitness ← $fitness\_particle_i$
17              global_best ← $particle_i$
18          end if
19      end for
20      for i in range ($p$) do
21          for $particle_k$ in swarm do
22              $velocity\_particle_{k+1}$ ($V_{k+1}$) ← Compute_velocity(c1, c2, r1, r2, w, global_best, $personal\_best_k$,
                $particle_k$, $velocity\_particle_k$)
23              $particle_{k+1}$← Update_position($velocity\_particle_{k+1}$, $particle_k$)
24              $fitness\_particle_k$ ← Compute_fitness($particle_k$, $D_{C\ Train}$)
25              if ($fitness\_particle_k$ > global_fitness) then
26                  global_fitness ← $fitness\_particle_k$
27                  global_best ← $particle_k$
28                  $personal\_best_k$← $particle_k$
29              end if
30          end for
31      end for
32      Subset of features ($F_P$) ← global_best
33      return Subset of features ($F_P$)
34      End
35  end procedure
```

Figure 3 shows the flowchart, and Algorithm 2 outlines the steps involved in the proposed phase 1 feature selection. Mainly six steps are involved in the construction of this algorithm which is given as follows: (i) *Initialize Particle and Velocity Position,* (ii) *Evaluation of Fitness Function,* (iii) *Update Personal and Global Best Position,* (iv) *Compute Velocity and Update Particle Position,* (v) *Termination Condition*. Each step of the proposed phase 1 feature selection algorithm is described in detail as follows:

(i) Initialize particle and velocity position(i) Initialize particle and velocity position

Nitialization of the particle position and velocity is the first step in the particle swarm optimization algorithm. A particle position within a swarm represents one of the acceptable solutions. The size of a particle and its corresponding velocity position are randomly initialized. Particle position vectors are allocated with binary values denoting the inclusion or exclusion of the feaures within that feature subset. The presence of 1's in the particle position denotes that the corresponding feature is included in the feature subset and 0's denotes the absence of the feature in the subset. The size of a particle is equivalent to the number of 1's present in that particle position vector.

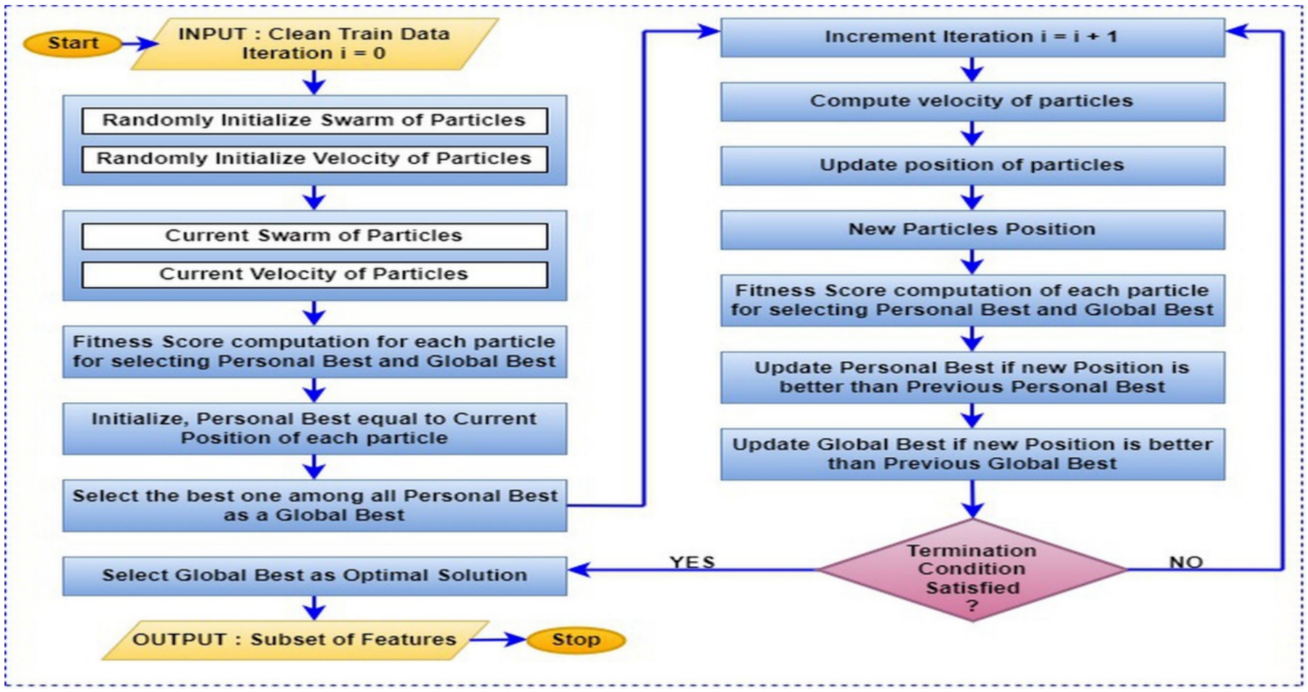

Start
INPUT : Clean Train Data
Iteration i = 0
Randomly Initialize Swarm of Particles
Randomly Initialize Velocity of Particles
Current Swarm of Particles
Current Velocity of Particles
Fitness Score computation for each particle for selecting Personal Best and Global Best
Initialize, Personal Best equal to Current Position of each particle
Select the best one among all Personal Best as a Global Best
Increment Iteration i = i + 1
Compute velocity of particles
Update position of particles
New Particles Position
Fitness Score computation of each particle for selecting Personal Best and Global Best
Update Personal Best if new Position is better than Previous Personal Best
Update Global Best if new Position is better than Previous Global Best
Termination Condition Satisfied ?
YES
NO
Select Global Best as Optimal Solution
OUTPUT : Subset of Features
Stop

Fig. 3 Flowchart of Phase 1 Feature Selection

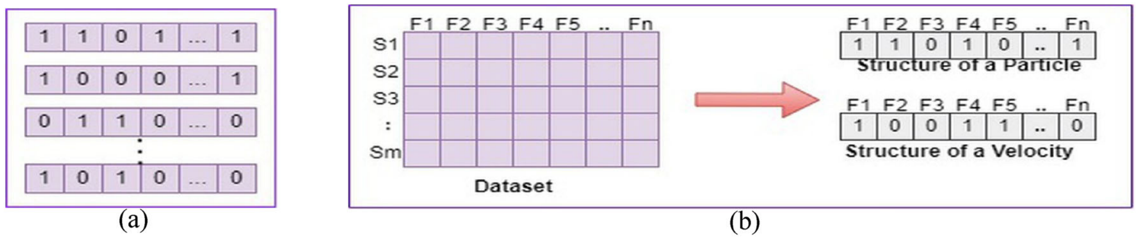


Fig. 4 In proposed framework **a** Arrangement of particles in a swarm, **b** Randomly initializing positions of a particle and a velocity

The maximum size of the particle is limited to the number of features available in the dataset. Figure 4a depicts the arrangement of particles in a swarm and Fig. 4b represents the structure of a particle position and velocity vector where, S1, S2, S3, …, Sm denote the set of samples and F1, F2, F3, …. Fn denote the set of features in the dataset.

(ii) Evaluation of fitness_function

Fitness function computation is a prominent step in the PSO algorithm. It produces the scores, and based on that score, the significance of a particular particle (candidate solution) is determined. This is the novelty of this paper because, here, the fitness function is computed by taking the G_mean of the ensemble model (described in Function 1). An ensemble model is employed, which combines multiple machine learning (in this case, 7) and deep learning models (in this case, 2) with a weight parameter (γ) given to each model (M) depending upon their importance. Common hyperparameter values given to both DNN and CNN models are as follows: 'Adam' as an optimizer, 'categorical cross-entropy' in the loss function, 'relu' & 'softmax' in the activation function, batch size = 32, and no. of epochs = 25. For the DNN model, the dropout rate is determined to be 0.2, and for the CNN model, kernel size and pool size are equal to 3 and 2, respectively. Equation (2) explains the fitness function used in the proposed phase 1 of the feature selection module.

$$\boldsymbol{Fitness\_Function} = \sum_{i=1}^{7} (\gamma * \boldsymbol{G_{Mean}}(\boldsymbol{M_i})) + \sum_{j=1}^{2} (\gamma * \boldsymbol{G_{Mean}}(\boldsymbol{M_j})) \tag{2}$$

where, $M_i \in \{DT, RF, Xgboost, Lightgbm, \quad Catboost, SVM, KNN\}, M_j \in \{DNN, CNN\}$

Several extensive experiments are performed by varying the value of γ from 0.1 to 1, and it is observed that γ = 1 offers optimal results in this paper. Thus, equal weights (γ)

are given to all the models, equal to 1, determined by the trial-and-error method.

The following Eq. (3) shows the formula for computing the geometric mean between specificity and sensitivity. Equations (4) and (5) provide the formula for computing specificity and sensitivity, respectively. Definitions of True Positives, False Negatives, True Negatives, and False Positives are given in Sect. 6.2 of this paper.

$$\mathbf{G_{Mean}} = \sqrt{(\mathbf{Specificity} * \mathbf{Sensitivity})} \tag{3}$$

where,

$$\boldsymbol{Specificity} = \frac{\boldsymbol{True\ Negatives}}{\boldsymbol{True\ Negatives} + \boldsymbol{False\ Positives}} \tag{4}$$

$$\boldsymbol{Sensitivity} = \frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives} + \boldsymbol{False\ Negatives}} \tag{5}$$

(iii) Update personal and global best position

After computing the fitness score of each particle in the swarm, the personal best position of each particle is updated based on fitness function. If a particle's fitness score performs better than its previous fitness score, change the current personal best position of that particular particle to the position with a large fitness score value; otherwise, the previous personal best position is considered the current one. The global best position is determined by the highest fitness score among all the personal best positions of the particles within a swarm. At the end of the iterations, the global best position is considered the best feature subset provided by this phase of the proposed model.

(iv) Compute velocity and update particle position

The velocity vector of the particle (described in Function 2) is computed using the following Eq. (6), and the position of the particle is updated (shown in Function 3) using the following Eq. (7).

$$\boldsymbol{V_i^{t+1}} = \boldsymbol{W}.\boldsymbol{V_i^t} + \boldsymbol{C_1}.\boldsymbol{r_1^t}\left(\boldsymbol{P_{b_1}^t} - \boldsymbol{P_i^t}\right) + \boldsymbol{C_2}.\boldsymbol{r_2^t}(\boldsymbol{g_b^t} - \boldsymbol{P_i^t}) \tag{6}$$

$$\boldsymbol{P_i^{t+1}} = \boldsymbol{P_i^t} + \boldsymbol{V_i^{t+1}} \tag{7}$$

where **W** is the inertia weight, $\boldsymbol{r}_1$, and $\boldsymbol{r}_2$ are the random numbers in the range [0,1], and they are randomly chosen, $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are the learning factors termed as cognitive behavior and social behavior coefficients, respectively. $\boldsymbol{P_i^t}$ and $\boldsymbol{V_i^t}$ denote the position and velocity vector of the ith particle, respectively, at time t. $\boldsymbol{P_{b_1}^t}$ denotes the personal best position of ith particle at time t, and $\boldsymbol{g_b^t}$ is the global best position (optimal feature subset) within the swarm at

Function 1 Compute_fitness(particle$_k$, D$_{C\_Train}$)

```
   Input: particle_k, D_C_Train
   Output: fitness_score_k
1  procedure FITNESS FUNCTION
2    Begin
3    $M_{ML} \leftarrow$ [DT, RF, Xgboost, Lightgbm, Catboost, SVM, KNN], $M_{DL} \leftarrow$ [DNN, CNN], $\gamma \leftarrow 1$
4    fitness_score_k $\leftarrow \sum_{i=1}^{7} \gamma * G_{Mean}(M_{ML_i}(particle_k,\ D_{C\_Train})) + \sum_{j=1}^{2} \gamma * G_{Mean}(M_{DL_j}(particle_k,$
     $D_{C\_Train}))$
5    return fitness_score_k
6    End
7  end procedure
```

Table 4 Parameter settings of the Phase 1 Feature Selection

Parameters	Values
Number of Iterations	58 (NSL-KDD), 95 (UNSW-NB15), 79 (CIC-IDS2018)
c1	1
c2	1
W	0.2
Γ	1
Swarm_size	10
Seed	42

Function 2 Compute_velocity(c1, c2, r1, r2, w, global_best, personal_best$_k$, particle$_k$, velocity_particle$_k$)

```
    Input: c1, c2, r1, r2, w, global_best, personal_best_k, particle_k, velocity_particle_k
    Output: velocity_particle_{k+1}
1   procedure COMPUTE VELOCITY
2       Begin
3       velocity_particle_{k+1} ← w * velocity_particle_k + c1 * r1 * (personal_best_k - particle_k) + c2 * r2 *
        (global_best - particle_k)
4       return velocity_particle_{k+1}
5       End
6   end procedure
```

time t. In this paper, several extensive experiments are performed on all three datasets to determine the best values of each parameter in phase 1 PSO-based feature selection. The optimal values of each parameter are provided in Table 4, which are determined by the trial-and-error method. Only those values are selected, which gives an optimal result in terms of fitness score at every iteration. The experiments are conducted by varying the swarm size from 10 to 50; it is observed that the swarm size is equal to 10, providing the optimal result for all the datasets.

Function 3 Update_position(velocity_particle$_{k+1}$, particle$_k$)

```
    Input: velocity_particle_{k+1}, particle_k
    Output: particle_{k+1}
1   procedure PARTICLE POSITION UPDATE
2       Begin
3       particle_{k+1} ← particle_k + velocity_particle_{k+1}
4       return particle_{k+1}
5       End
6   end procedure
```

(V) Termination condition

There are two ways to terminate the execution of the algorithm: (a) By defining a fixed number of iterations and (b) By constantly observing the progress of the graph between the best fitness score for each iteration and the number of iterations. In the second case, the execution of the algorithm will be terminated if the fitness score of the global best position (feature subset) of each iteration becomes stable. In this paper, a rigorous number of experiments are conducted to select the appropriate value for the number of iterations per dataset. Therefore, the number of iterations varies depending on the type of dataset. Figure 9a demonstrates the convergence graph between the best fitness score per iteration and the number of iterations in the algorithm. Table 4 indicates the number of iterations for each dataset as the convergence condition of the algorithm. After this phase, the number of features is reduced to 25, 29, and 48 for the NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively.

5.2.2 Phase2: ant colony optimization based feature selection

In this phase, the most informative features are selected based on the mutual information value and correlation value between feature-feature pairs and class-feature pairs. It is achieved by considering mutual information and correlation metrics in the fitness function of the ant colony optimization algorithm. Therefore, the feature subset obtained after this phase contains uncorrelated and information-rich features. By removing irrelevant features, this phase ensures computational efficiency, while the less dimensionality of the data ensures the lightweight of the model. Figure 5 shows the flowchart, and Algorithm 3 outlines the overall steps involved in the proposed phase 2 feature selection.

To determine the optimal values of each parameter in the ACO-based feature selection in phase 2, various comprehensive experiments are performed on each dataset considered in this paper. The optimal values of each parameter are provided in Table 5 by the trial-and-error method. Only those values are selected, which offers an optimal result in terms of the fitness function at each iteration and considering multi-class classification. The fitness function is observed to stop increasing its values after 50, 60, and 50 iterations for NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively demonstrated in Fig. 9a. The number of ants is initialized with the number

Algorithm 3 Phase 2 for feature selection

```
    Input: Filtered_Train_Data($D_{F\_Train}$), evaporation_rate, No. of Iterations, No. of Ants, No. of features in
    feature subset, alpha, beta
    Output: Optimal subset of features ($F_A$)
1   procedure ACO-BASED FEATURE SELECTION
2       Begin
3       Initialize, n ← len($D_{F\_Train}$.column) – 1, evaporation_rate ($\rho$) ← 0.1, m ← No. of Iterations, a ← No.
        of Ants, s ← No. of features in feature subset, alpha ($\alpha$)← 1, beta ($\beta$) ← 1, a ← n
4       Pheromone_LUT ← Initialize_Pheromone_LUT(n)
5       Heuristic_LUT ← Heuristic_LUT(n, Xgboost(), $D_{F\ Train}$)
6       for i in range(m) do
7           Initialize, score ← [ ], F1_Score ← [ ], feature_subset_list ← [ ]
8           for ia in range (a) do
9               feature_subset_ia ← antBuildSubset(ia, n, s, alpha, beta)
10              score_ia ← fitness_score (feature_subset_ia, $D_{F\ Train}$)
11              f1_score_ia ← f1_score(feature_subset_ia, Xgboost(), $D_{F\ Train}$)
12              feature_subset_list.append(feature_subset_ia)
13              score.append(score_ia)
14              F1_Score.append(f1_score_ia)
15          end for
16          index ← max(score).index
17          Best_features ← feature_subset_list[index]
18          Pheromone_LUT ← Update_Pheromone(n, $\rho$, Q, feature_subset_list, F1_Score,
            Pheromone_LUT)
19      end for
20      return Best_features
21      End
22  end procedure
```

of features obtained from the PSO-based feature selection phase corresponding to each dataset (discussed in Sect. 5.2.1). Moreover, the number of features in the subset is determined on the basis of the feature importance, which is obtained by the importance plot using Xgboost for each datasets.

The primary step of the ant colony optimization algorithm applied in this paper can be analyzed using the following points: (i) *Initialization of Look-up-table (LUT),* (ii) *Heuristic Function,* (iii) *Probability Function,* (iv) *Fitness Function,* (v) *Pheromone Update Rule*. Each step involved in the construction of this phase is described as follows:

(i) Initialization of look-up-table (LUT)

Initialization of the look-up table (LUT) is the first step in the ant colony optimization algorithm. LUT is a matrix in the form of either lower or upper triangular. The reason behind using only half (either upper or lower) part of the matrix is that it contains symmetric values on both sides of LUT. It is a matrix of dimension (n*n) where only one-half part of the matrix is utilized. It implies that only $(n*(n-1))/2$ entries contain unique values in the LUT. Therefore, only those entries in the LUT are filled with values. Here, 'n' represents the number of features obtained after the first phase of the feature selection module in the proposed model. It includes mainly two pieces of information associated with feature-feature pairs, i.e., (a) Pheromone value and (b) Heuristic value. Figure 6 shows an example of the LUT employed in this paper where both the values are combined in a single table. Functions 4 and 5 explain the initializing of the pheromone LUT and heuristic LUT, respectively.

Function 4 Initialize_Pheromone_LUT(n)

```
    Input: n
    Output: pheromone_matrix
1   procedure INITIALIZATION OF PHEROMONE
2       Begin
3       value ← 0.8
4       pheromone_matrix ← full((n,n), value)
5       return pheromone_matrix
6       End
7   end procedure
```

Function 5 Heuristic_LUT(n, Xgboost(), D_{F_Train})

```
    Input: n, Xgboost(), $D_{F\_Train}$
    Output: heuristic_matrix
1   procedure COMPUTE HEURISTIC LUT
2       Begin
3       heuristic_matrix ← zeros((n,n))
4       for i in range(n) do
5           for j in range(n) do
6               if (i <= j):
7                   heuristic_value ← Heuristic_value(i, j, $D_{F\_Train}$, Xgboost())
8                   heuristic_matrix [i, j] ← heuristic_value
9                   heuristic_matrix [j, i] ← heuristic_value
10              end if
11          end for
12      end for
13      return heuristic_matrix
14      End
15  end procedure
```

Pheromone LUT is first initialized with a constant value (here, it is equal to 0.8) and is updated after each iteration, as indicated in Algorithm 3. Furthermore, heuristic LUT is initialized with the heuristic value (described in Function 6) corresponding to each entry in LUT, which is computed by applying the proposed heuristic function (shown in Eq. 8). Heuristic value depends upon 'true positive rate' and 'cosine similarity' corresponding to different feature-pairs in the LUT. Therefore, entries in heuristic LUT will remain constant throughout the execution of Algorithm 3. In Fig. 6, the upper section represents the heuristic value, whereas the bottom section denotes the pheromone value of each row in the LUT. For instance, the entries corresponding to feature '1' and feature '2' are 0.55 and 0.73, respectively, which indicate the heuristic and the pheromone values respectively.

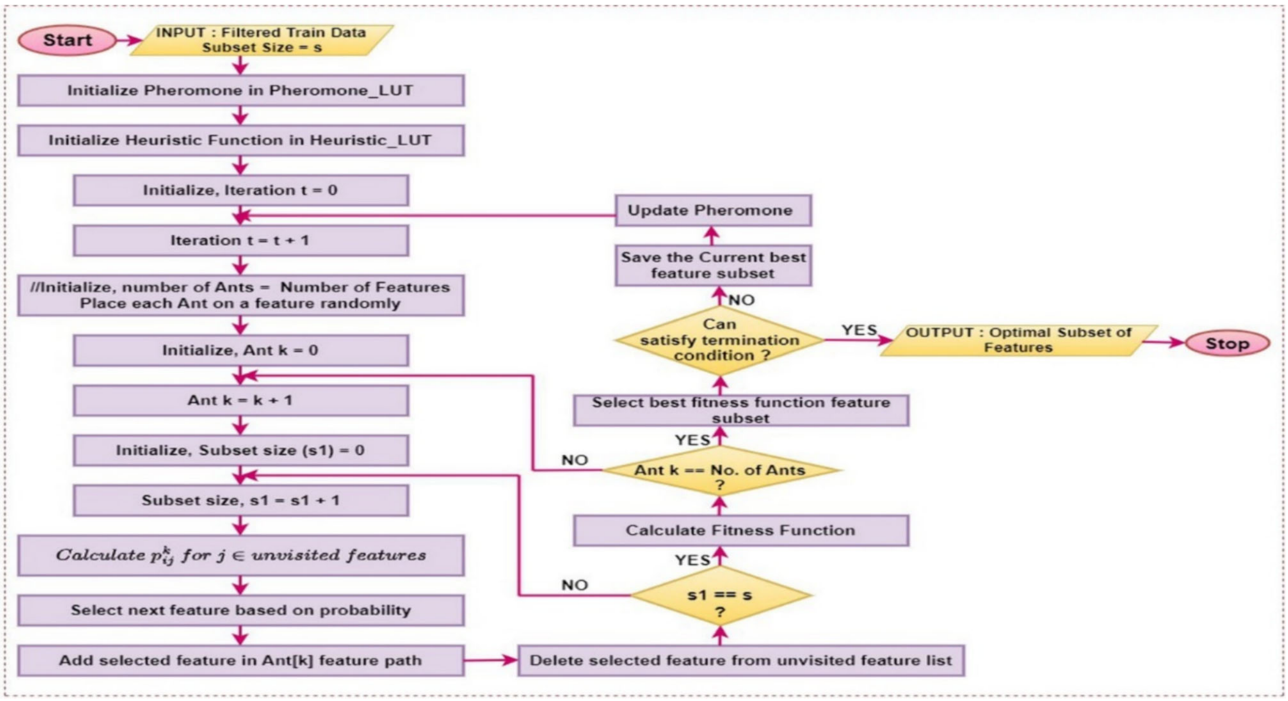

Start
INPUT : Filtered Train Data Subset Size = s
Initialize Pheromone in Pheromone_LUT
Initialize Heuristic Function in Heuristic_LUT
Initialize, Iteration t = 0
Iteration t = t + 1
//Initialize, number of Ants = Number of Features Place each Ant on a feature randomly
Initialize, Ant k = 0
Ant k = k + 1
Initialize, Subset size (s1) = 0
Subset size, s1 = s1 + 1
Calculate p_{ij}^k for $j \in$ unvisited features
Select next feature based on probability
Add selected feature in Ant[k] feature path
Delete selected feature from unvisited feature list
s1 == s ?
NO
YES
Calculate Fitness Function
Ant k == No. of Ants ?
NO
YES
Select best fitness function feature subset
Can satisfy termination condition ?
YES
OUTPUT : Optimal Subset of Features
Stop
NO
Save the Current best feature subset
Update Pheromone

Fig. 5 Flowchart of Phase 2 Feature Selection

Table 5 Parameter settings of Phase 2 Feature Selection

Parameters	Values
Number of Ants	25 (NSL-KDD), 29 (UNSW-NB15), 48 (CIC-IDS2018)
Number of Iterations	50 (NSL-KDD), 60 (UNSW-NB15), 50 (CIC-IDS2018)
Alpha (α)	1
Beta (β)	1
Evaporation_rate (ρ)	0.1
Number of Features in subset	15 (NSL-KDD), 20 (UNSW-NB15), 20 (CIC-IDS2018)
Constant (Q)	1
Initial Pheromone	0.8
Seed	42

(ii) Heuristic function

A novel heuristic function is utilized to evaluate the heuristic values in the heuristic LUT. Function 6 represents the overall step-by-step procedure for evaluating the heuristic value for different feature-feature pairs in the LUT. The heuristic value corresponding to feature 'A' and feature 'B' ($\boldsymbol{\eta_{AB}}$) is computed using the following Eq. (8). It is measured by simply dividing the 'True Positive Rate' by the 'Cosine_Similarity' between feature-feature pairs.

$$\boldsymbol{\eta_{AB}} = \frac{\boldsymbol{Weighted\_TPR}}{\boldsymbol{Cosine\_Similarity(A, B)}} \tag{8}$$

TPR for a specific class 'i' can be assessed using the given Eq. (9), while the weighted TPR can be measured using the corresponding Eq. (10). For measuring the value of Weighted_TPR, several classification algorithms (such as DT, RF, Xgboost, and KNN) are analyzed, and it is observed that the 'Xgboost' classifier offers the optimal performance.

$$\boldsymbol{TPR_i} = \frac{\boldsymbol{True\ Positive_i}}{\boldsymbol{True\ Positive_i + False\ Negative_i}} \tag{9}$$

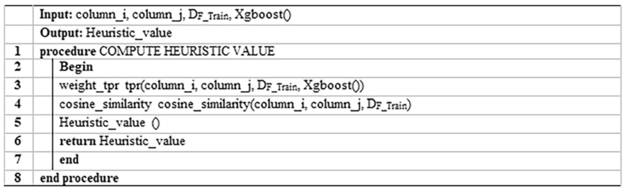
$$\boldsymbol{Weighted\_TPR} = \frac{\sum_{i=1}^{C} (\boldsymbol{S_i * TPR_i})}{\boldsymbol{S}} \tag{10}$$

where 'C' and 'S' are the total number of classes and the total number of samples, respectively, in the data.S_i represents the number of instances of a particular class 'i' present in the data. The Cosine_Similarity between two features, 'A' and 'B', is determined using the following Eq. (11). The actual meaning of True Positive and False Negative are provided in Sect. 6.2.

$$\boldsymbol{Cosine\_Similarity(A, B)} = \left| \frac{\sum_{i=1}^{S} (\boldsymbol{A_i * B_i})}{\left(\sqrt{\sum_{i=1}^{S} \boldsymbol{A_i}^2}\right) * \left(\sqrt{\sum_{i=1}^{S} \boldsymbol{B_i}^2}\right)} \right| \tag{11}$$

where A_i & B_i are the specific instances of feature vectors 'A' and 'B', respectively.

Function 6 Heuristic_value(column_i, column_j, D_{F_Train}, Xgboost())



```
Input: column_i, column_j, $D_{F\_Train}$, Xgboost()
Output: Heuristic_value
procedure COMPUTE HEURISTIC VALUE
    Begin
    weight_tpr  tpr(column_i, column_j, $D_{F\_Train}$, Xgboost())
    cosine_similarity  cosine_similarity(column_i, column_j, $D_{F\_Train}$)
    Heuristic_value  ()
    return Heuristic_value
    end
end procedure
```

(iii) Probability function

Initially, the number of ants is equivalent to the number of features in the subset obtained after phase 1 feature selection. Each ant is standing at every feature provided that no two ant stands at the same feature. Now each ant explores their path by visiting the other features (nodes) with the aim of achieving the best feature subset. This is attained by choosing the routes with the highest probability value. Equation (12) signifies the function for assessing probability values related to each feature in the unvisited feature list, which helps the ants select the next feature appropriately in their path. Function 7 explains the systematic approach to calculate the probability values between two given features and it gives the best feature subset selected by specific ant 'k'.

$$\boldsymbol{P_{AB}^{k}} = \frac{\boldsymbol{\tau_{AB}^{\alpha} * \eta_{AB}^{\beta}}}{\sum_{m} \boldsymbol{\tau_{AB}^{\alpha} * \eta_{AB}^{\beta}}} \tag{12}$$

where P_{AB}^{k} is the probability of selecting the next feature as 'B' by ant 'k' if the ant 'k' is standing at feature 'A'. 'η_{AB}' is the heuristic value between feature 'A' and feature 'B' present in the heuristic LUT. 'τ_{AB}' is the pheromone value between feature 'A' and feature 'B' present in the pheromone LUT. 'm' is the total number of unvisited features. The parameter 'α' adjusts the impact of 'τ_{AB}', while the parameter 'β' governs the effect of 'η_{AB}'.

Function 7 antBuildSubset(ia, n, s, alpha, beta)

```
Input: ia, n, s, alpha, beta
Output: feature_subset_ia
procedure
    begin
    initialize, unvisited_features ← [0,1,2,3,…,(n-1)], s1 ← 1
    indexes ← where(in1d(unvisited_features, ants[ia].feature_path))[0]
    ua ← unvisited_features[indexes]
    unvisited_features ← unvisited_features.delete(ua)
    while (s1 <= s) do
        values ← [0] * len(unvisited_features) , prob_value ← [0] * len(unvisited_features)
        for index_uf in range(len(unvisited_features)) do
            uf ← unvisited_features[index_uf]
            eta ← heuristic_matrix[ua, uf]
            tau ← pheromone_matrix[ua, uf]
            values [index_uf] ← (tau**alpha) * (eta**beta)
        end for
        total_sum ← sum(values)
        for index_uf in range(len(unvisited_features)) do
            prob_value[index_uf] ← values[index_uf] / total_sum
        end for
        max_index ← argmax(prob_value)
        next_features ← unvisited_features[max_index]
        if (s1 == s - 1) do
            feature_subset ← ants[ia].feature_path
            feature_subset ← feature_subset.append(next_features)
        end if
        ants[ia].feature_path.append(next_features)
        unvisited_features.delete(next_features)
        s1 ← s1 + 1
        ua ← next_features
    end while
    return feature_subset
    end
end procedure
```

(iv) Fitness function

In the ant colony optimization-based feature selection algorithm, the fitness function plays a very crucial role in precisely determining the best feature subset. Therefore, setting the fitness function appropriately is a primary key in the ant colony optimization-based feature selection algorithm. During this phase, two different fitness functions are utilized to analyze various combinations in subset selection. The aim is to identify the most crucial features subset for designing of a light-weighted IDS model. Description of both these fitness functions, along with the algorithmic details, are provided in the following subsections (a) and (b):

a. Mutual information-based

It selects the most crucial features based on the mutual information theory concept. It observes the mutual information between the feature subset selected by a specific ant and its corresponding target class feature. It calculates the entropy related to the subset and conditional entropy related to the subset given class feature. Functions 8.a and 8.a1 explain the systematic way to execute the mutual information as a fitness function in the proposed model. Equation (13) shows the mutual information $(I(F, C))$ by selecting a subset of features (F) and the class label (C).

Function 8.a. fitness_score (feature_subset_ia, D_{F_Train})

```
    Input: feature_subset_ia, $D_{F\_Train}$
    Output: score_ia
1   procedure MUTUAL INFORMATION BASED
2     begin
3     columns ← $D_{F\_Train}$.shape[1]
4     X ← $D_{F\_Train}$[: , feature_subset_ia], Y ← $D_{F\_Train}$[columns:]
5     features ← X.shape[1]
6     Mutual_Information ← [ ]
7     for feature in range (features) do
8         $X_f$ ← X[ : , feature]
9         mutual_info ← Mutual_Information($X_f$, Y)
10        Mutual_Information.append(mutual_info)
11        score_ia ← sum(Mutual_Information)
12    end for
13    return score_ia
14    end
15  end procedure
```

Function 8.a1. Mutual_Information(Xf, Y)

```
    Input: $X_f$, Y
    Output: mutual_info
1   procedure MUTUAL INFORMATION
2     begin
3     H_F ← Entropy($X_f$)
4     H_F_C ← Conditional_Entropy($X_f$, Y)
5     mutual_info ← (H_F - H_F_C)
6     return mutual_info
7     end
8   end procedure
```

$$\boldsymbol{I(F,C) = H(F) - H(F|C)} \tag{13}$$

Entropy 'H' of feature subset 'F' is estimated using the following Eq. (14).

$$\boldsymbol{H(F) = -\sum_{f_i \in F} P(f_i) * \log_2(P(f_i))} \tag{14}$$

Entropy 'H' for feature subset 'F' after analyzing class 'C' can be measured using the following Eq. (15).

$$\boldsymbol{H(F|C) = -\sum_{c_k \in C} P(c_k) * \sum_{f_i \in F} (P(f_i|c_k) * \log_2 P(f_i|c_k))} \tag{15}$$

where F and C represent the feature subset and target class, respectively. $I(F,C)$, $H(F)$, and $H(F|C)$ describe the mutual information between F and C, the entropy of F, and the conditional entropy of F provided C, respectively.$P(f_i|c_k)$ presents probability of a feature having a value f_i and target class being c_k, while $P(f_i)$, and $P(c_k)$ represent the probability of a feature having a value f_i and the probability of target class being c_k respectively.

b. Correlation-based

The second type of fitness function used here is the correlation-based. Equation (16) determines the potential of a particular feature subset 'F' selected by the ant colony optimization-based feature selection algorithm. This function selects features based on the feature-feature correlation and the class-feature correlation. Therefore, this fitness function gives importance to both correlation values. It is important to observe that for selecting the best feature subset, the correlation between the class-feature pair should be as high as possible, and the correlation between the feature-feature pair should be as low as possible. Function 8.b, along with functions 8.b1 and 8.b2, explains this fitness function.

$$\boldsymbol{Fitness\ Function = \frac{s * \overline{r_{cf}}}{\sqrt{(s + s * (s-1)\overline{r_{ff}})}}} \tag{16}$$

where 's' is the number of features in the feature subset selected by a particular ant and r_{cf} indicates the correlation between categorical and numerical features. It is measured using kendall's rank correlation coefficients, and its average value is taken here. The systematic procedure for computing the value of $\overline{r_{cf}}$ is outlined in function 8.b1. r_{ff} is the pearson correlation coefficient between two numerical features, and it is computed between two features 'A' and feature 'B' using the following Eq. (17) and its average value are taken. Where 'S' represents the total number of samples while A_i and B_i indicate the feature value of the 'ith' sample of features 'A' and 'B', respectively. The step-by-step process for computing the average value of $\overline{r_{ff}}$ is provided in function 8.b2.

Function 8.b. fitness_score (feature_subset_ia, D_{F_Train})

```
    Input: feature_subset_ia, $D_{F\_Train}$
    Output: score_ia
1   procedure CORRELATION BASED
2     begin
3     columns ← $D_{F\_Train}$.shape[1]
4     X ← $D_{F\_Train}$[ : ,feature_subset_ia] , Y ← $D_{F\_Train}$[columns : ]
5     r_cf ← compute_rcf(X,Y)
6     r_ff ← compute_rff(X)
7     s ← len(feature_subset_ia)
8     score_ia ← $(s * r_{cf})/(\sqrt{s + s * (s-1) * r_{ff}}\ )$
9     return score_ia
10    end
11  end procedure
```

$$\boldsymbol{r_{AB} = \frac{S\left(\sum_{i=1}^{S} A_i B_i\right) - \left(\sum_{i=1}^{S} A_i\right) * \left(\sum_{i=1}^{S} B_i\right)}{\sqrt{\left[S\sum_{i=1}^{S} A_i^2 - \left(\sum_{i=1}^{S} A_i\right)^2\right] * \left[S\sum_{i=1}^{S} B_i^2 - \left(\sum_{i=1}^{S} B_i\right)^2\right]}}} \tag{17}$$

Function 8.b1. compute_rcf(X,Y)

```
     Input: X,Y
     Output: r_cf
1    procedure
2       begin
3       r_cf ← [ ]
4       for i in range(X.shape[1]) do
5           tau ← kendalltau (X[: , i], Y)
6           r_cf.append(tau)
7       end for
8       return mean(r_cf)
9       end
10   end procedure
```

(v) Pheromone update rule

Pheromone updation (summarized in Function 9) is the last step in the ant colony optimization-based feature selection algorithm, and its value is updated in the pheromone LUT. It has been studied that ants secrete a kind of chemical (called pheromone) in the path where they walk. In searching for the best feature subset, they secrete a chemical to help other ants follow the suitable route attracted by the amount of chemicals in the route. If the concentration of pheromones in a route is higher, the probability of an ant selecting the route with the highest pheromone concentration increases. Hence, the pheromone LUT is updated appropriately corresponding to the feature-feature pair.

Function 8.b2. compute_rff(X)

```
     Input: X
     Output: r_ff
1    procedure
2       begin
3       r_ff ← [ ]
4       for i in range(X.shape[1]) do
5           for j in range(i+1, X.shape[1]) do
6               tau ← pearsonr (X[: , i], X[: , j])
7               r_ff.append(tau)
8           end for
9       end for
10      return mean(r_ff)
11      end
12   end procedure
```

The pheromone value is decremented for all feature–feature pairs. Moreover, Eqs. (18) and (19) describe the formula for updating the values in the pheromone LUT.

$$\tau_{ij}^{(t+1)} = (1-\rho) * \tau_{ij}^{(t)} + \sum_{k=1}^{|No.of Ants|} sum\_delta[k] \tag{18}$$

$$sum\_delta[k] = \frac{Q}{(1 - F1\_Score(ant[k].feature\_path)) * 100} \tag{19}$$

where $\tau_{ij}^{(t+1)}$ represents the amount of pheromones between feature (i) and feature (j) pair at the (t + 1)th iteration. ρ is the evaporation rate, and by the trial-and-error method, its value is determined as 0.1, and Q is a constant number (here, it is 1). To find the F1-Score value, various machine learning models, such as KNN, DT, RF, and xgboost, are applied. It is observed that xgboost performs better than others.

5.3 GA-based hyperparameter tuning

The best set of hyperparameters improves the detection models' performance. For this reason, this paper introduces a genetic algorithm-based hyperparameter tuning module for retrieving the best set of hyperparameters per detection model. Genetic algorithms (GAs) offer a powerful and flexible approach to optimizing hyperparameters in machine learning models. They excel in global search capability, adaptability, parallelism, robustness to noise, and a balanced exploration–exploitation trade-off. These attributes make GAs highly effective for navigating complex and high-dimensional hyperparameter spaces, often surpassing other optimization methods such as grid search, random search, and bayesian optimization. Compared to grid search, GAs efficiently explore the search space without an exhaustive search. They can leverage historical information to guide the search more effectively than random search. In contrast to bayesian optimization, GAs are less computationally intensive per iteration and are better equipped to handle larger search spaces. GAs also excel in handling mixed types of hyperparameters and maintaining diversity in the population, thereby reducing the risk of premature convergence. Algorithm 4 outlines the proposed GA-based hyperparameter tuning module. As shown in Algorithm 4, this module is executed using training and testing datasets.

Table 6 shows the parameter settings for executing the genetic algorithm in the search for the best hyperparameter values. To determine the optimal values of each parameter in the GA-based hyperparameter optimization in the third module, this paper performs several rigorous experiments by varying the population size, selection rate, and mutation

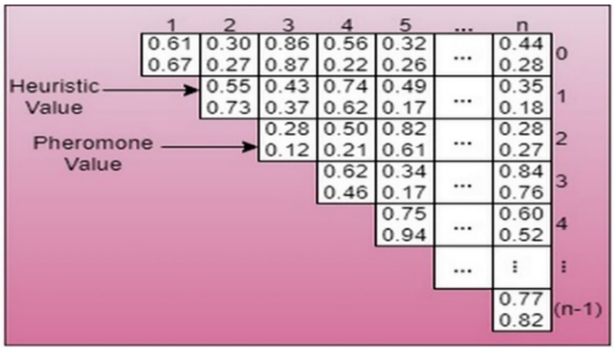


Fig. 6 Look-Up Table

rate on each dataset. The optimal values of each parameter are given in Table 6 by the trial-and-error method. Only those values that offer an optimal result in terms of the fitness function at each iteration are selected by focusing on multi-class classification. The fitness function gives maximum values at 128, 64, and 64 generations for NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively as depicted in Fig. 9b, the convergence graph between fitness score and no. of iterations (or generations).

(i) Initialization of population

Initialization of the population is the first step in implementing the GA-based hyperparameter tuning module. Each detection model is executed separately in the search of the best hyperparameters. So, this module is executed as many times as the number of detection models provided as input to this module. Hyperparameters and their approximate ranges are pre-specified in this module, as the populations are initialized based on the hyperparameters' ranges. Figure 7 describes the structure of a population, chromosome, and gene for xgboost classifier in the proposed genetic algorithm. For instance, executing this module for the xgboost classifier, the genes in the chromosome are 'max_depth', 'booster', 'n_estimators', 'min_child_weight', 'gamma', and 'subsample'.

Table 6 Parameter settings of GA algorithm in this work

Parameters	Values
No. of Generations	128 (NSL-KDD), 64 (UNSW-NB15), 64 (CIC-IDS2018)
Population Size	50
Selection Rate	0.5
Mutation Rate	0.5
Seed	42

(ii) Fitness score computation

Each candidate chromosome is evaluated using the fitness function. Since multi-class (multiple attacks and normal class) classification is being performed here, the weighted F1-Score of the detection model is utilized for the fitness score computation of the chromosomes. The formula for computing the f1-score, precision, and recall for a

Function 9 Update_Pheromone(n, ρ, Q, feature_subset_list, F1_Score, Pheromone_LUT)

```
Input: n, ρ, Q, feature_subset_list, F1_Score, Pheromone_LUT
Output: Pheromone_matrix
procedure UPDATE PHEROMONE LUT
    begin
    for i in range(n) do
        for j in range(n) do
            Pheromone_matrix [i][j] ← (1- ρ) * Pheromone_matrix [i][j]
        end for
    end for
    num_ants ← len(feature_subset_list)
    for f1_score, feature_subset in (F1_Score, feature_subset_list) do
        for i in range (len(feature_subset) - 1) do
            current_feature ← feature_subset[i]
            next_feature ← feature_subset[i+1]
            Pheromone_matrix[current_feature][next_feature] ←
            Pheromone_matrix[current_feature][next_feature] + ($Q/(1-f1\_score)$)
        end for
    end for
    return Pheromone_matrix
    end
end procedure
```

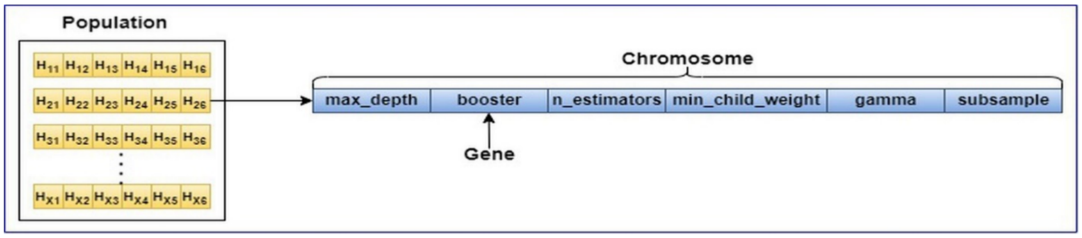


Fig. 7 Structure of a population, chromosome, and gene

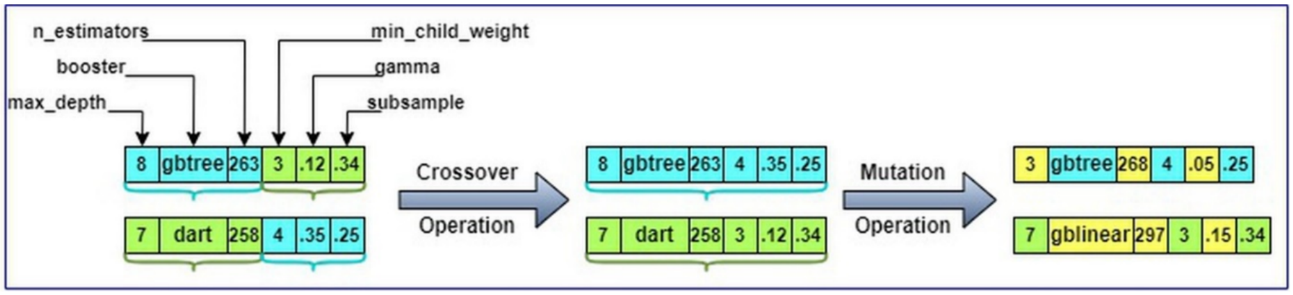


Fig. 8 Single point crossover and random resetting mutation operations in the proposed GA-based hyperparameter optimization

particular class 'i' are specified in Eqs. (20–22), respectively. The formula for computing the weighted f1-score is provided in Eq. (23).

$$F1 - Score_i = \frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i} \quad (20)$$

$$Precision_i = \frac{True\,Positive_i}{True\,Positive_i + FalsePositive_i} \quad (21)$$

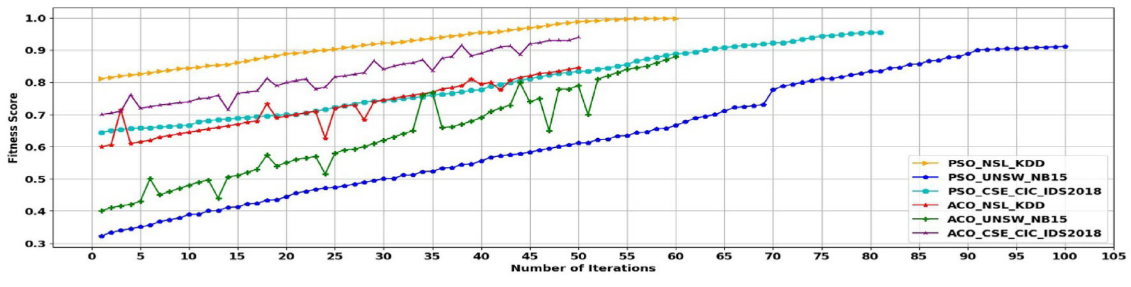
$$Recall_i = \frac{True\,Positive_i}{True\,Positive_i + Fals\,eNegative_i} \quad (22)$$

$$WeightedF1 - Score = \frac{\sum_{i=1}^{|c|} |S_i| F1 - Score_i}{N} \quad (23)$$

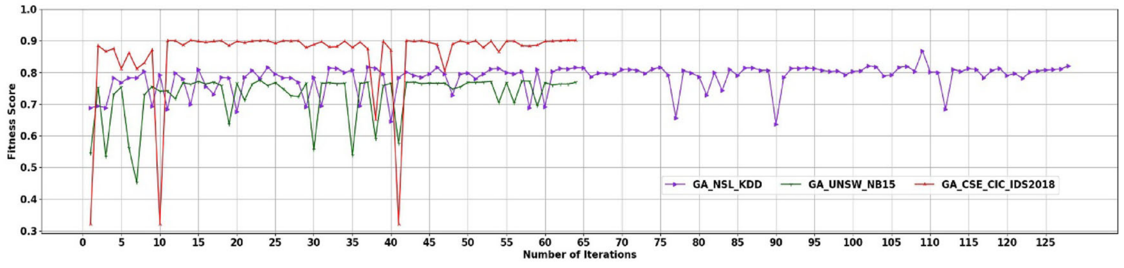
where N and $|c|$ represent the total no. of instances and total no. of classes, respectively, while $|S_i|$ denotes the total no. of instances of the 'i^th' class in the dataset.

Algorithm 4 Hyperparameter Tuning.

```
    Input: $D_{Train}$, $D_{Test}$, $M_B$ or $M_E$
    Output: Best_Hyperparameters($H_B$)
1   procedure GA-BASED HYPERPARAMETER TUNING
2       begin
3       n' ← no. of gen, size ← pop_size, s' ← selection_rate, m' ← mutation_rate, best_hyperparameters ←
        [ ]
4       hyperparameters_range ← Define hyperparameters_range($M_B$ or $M_E$)
5       population ← initialize_hyperparameters(size, hyperparameters_range, $M_B$ or $M_E$)
6       for i in range(n') do
7           scores, pop_after_fit ← fitness_score (population, $D_{Train}$, $D_{Test}$, $M_B$ or $M_E$)
8           pop_after_sel ← selection(pop_after_fit, s', size)
9           pop_after_cross ← crossover(pop_after_sel, s', size)
10          population ← mutation(pop_after_cross, m', hyperparameters_range)
11          best_hyperparameters.append(pop_after_fit[0])
12      end for
13      return best_hyperparameters
14      end
15  end procedure
```



(a)



(b)

Fig. 9 Convergence graph between fitness score and number of iterations for **a** PSO, and ACO algorithms, **b** GA

(iii) Parent selection

Parent chromosomes are selected using the roulette wheel selection function [62]. The parent selection rate is determined as 0.5 by trial and error. This means that 50% of chromosomes from the old population are selected as in the new population, and the remaining 50% of new child chromosomes are built through the reproduction process described below.

(iv) Reproduction operation

Reproduction operation is performed to generate new child chromosomes to balance the number of chromosomes in the population. This phase contains two operations: crossover and mutation. Figure 8 describes the overall reproduction operation used in this paper for the xgboost classifier.

a. Crossover: In this phase, single-point crossover [63] operation is applied, as shown in Fig. 8. From the two parent chromosomes, two new child chromosomes are created. The overall steps involved in the crossover operation are given as follows:
   a Initially, two chromosomes are selected from the population as the parent chromosomes for the crossover operation
   b The first half part of the first parent chromosome and the second half part of the second parent chromosomes are merged to make the first child chromosome
   c Similarly, the first half part of the second parent chromosome and the second half part of the first chromosome are merged to make the second child chromosome
b. Mutation: In this phase, a random resetting mutation operation is applied to make the variation in the population. This operation is performed on each chromosome in the population. Here, the mutation rate is determined as 0.5 by trial and error method. This means that 50% of genes in each chromosome in the population have changed their values to new random values from their pre-defined ranges. Figure 8 depicts the mutation operation where, out of 6 genes in the chromosome, randomly, 3 genes ('yellow' in color after mutation operation) changed their values to a new value from the appropriate pre-defined ranges for these genes.

(V) Termination condition

The termination condition is decided based on the fitness score value in this phase. If the fitness function stops enhancing its values, this is the termination condition for the genetic algorithm-based hyperparameter tuning. The trial-and-error method demonstrates that each detection model takes different generations to stop increasing their values (some detection model takes 64, and some take 128). Therefore, the maximum number of generations is fixed for each detection model, equal to 128 (in the case of NSL-KDD) and 64 (in the case of UNSW-NB15 and CIC-IDS2018). Table 6 shows the maximum number of generations the detection model takes to achieve optimal performance on each dataset. Choosing these values as the

Table 7 For Correlation-based Fitness function

Detection Model	Best Hyperparameter values for each detection model		
	NSL-KDD	UNSW-NB15	CSE-CIC-IDS2018
DT	• splitter: best	• splitter: random	• splitter: best
	• criterion: gini	• criterion: gini	• criterion: entropy
	• max_depth: 14	• max_depth: 13	• max_depth: 10
	• min_samples_split: 15	• min_samples_split: 16	• min_samples_split: 27
	• min_samples_leaf: 7	• min_samples_leaf: 15	• min_samples_leaf: 14
	• min_weight_fraction_leaf: 1.4e-07	• min_weight_fraction_leaf: 0.00265	• min_weight_fraction_leaf: 4.8e-08
	• min_impurity_decrease: 1.3e-05	• min_impurity_decrease: 0.00029	• min_impurity_decrease: 6.3e-05
	• max_leaf_nodes: 10	• max_leaf_nodes: 22	• max_leaf_nodes: 32
LR	• C: 689.89754	• C: 432.71826	• C: 0.10842
	• solver: lbfgs	• solver: lbfgs	• solver: lbfgs
	• penalty: l2	• penalty: l2	• penalty: l2
	• tol: 0.00798	• tol: 0.00128	• tol: 0.000326
	• max_iter: 587	• max_iter: 288	• max_iter: 525
KNN	• algorithm: auto	• algorithm: kd_tree	• algorithm: ball_tree
	• k_n_neighbors: 10	• k_n_neighbors: 10	• k_n_neighbors: 5
	• weights: distance	• weights: uniform	• weights: distance
	• p: 2	• p: 1	• p: 1
	• leaf_size: 6	• leaf_size: 10	• leaf_size: 10
SVM	• C: 30.72994	• C: 10.74238	• C: 88.06638
	• kernel: poly	• kernel: poly	• kernel: poly
DNN	• num_layers: 4	• num_layers: 2	• num_layers: 3
	• units_0: 118	• units_0: 93	• units_0: 98
	• units_1: 108	• units_1: 91	• units_1: 90
	• units_2: 86	• dropout_rate: 0.41787	• units_2: 66
	• units_3: 68	• learning_rate: 0.00603	• dropout_rate: 0.19406
	• dropout_rate: 0.15575	• batch_size: 32	• learning_rate: 0.00184
	• learning_rate: 0.00128		• batch_size: 32
	• batch_size: 16		
CNN	• filters: 25	• filters: 55	• filters: 49
	• kernel_size: 4	• kernel_size: 4	• kernel_size: 5
	• num_dense_units: 97	• num_dense_units: 119	• num_dense_units: 117
	• dropout_rate: 0.44467	• dropout_rate: 0.47851	• dropout_rate: 0.11797
	• learning_rate: 0.00327	• learning_rate: 0.00143	• learning_rate: 0.00365
	• batch_size: 16	• batch_size: 64	• batch_size: 16
RF	• n_estimators: 96	• n_estimators: 90	• n_estimators: 8
	• max_depth: 14	• max_depth: 16	• max_depth: 11
	• min_samples_split: 30	• min_samples_split: 22	• min_samples_split: 26
	• min_samples_leaf: 10	• min_samples_leaf: 11	• min_samples_leaf: 12

Table 7 (continued)

Detection Model	Best Hyperparameter values for each detection model		
	NSL-KDD	UNSW-NB15	CSE-CIC-IDS2018
Xgboost	• booster: gbtree	• booster: dart	• booster: gbtree
	• lambda: 7.07e-05	• lambda: 0.03457	• lambda: 0.00048
	• alpha: 0.00219	• alpha: 0.19941	• alpha: 0.00119
	• subsample: 0.25869	• subsample: 0.47035	• subsample: 0.59606
	• colsample_bytree: 0.96169	• colsample_bytree: 0.93334	• colsample_bytree: 0.97709
	• early_stopping_rounds: 9	• early_stopping_rounds: 20	• early_stopping_rounds: 22
	• n_estimators: 32	• n_estimators: 16	• n_estimators: 64
	• max_depth: 5	• max_depth: 9	• max_depth: 9
	• min_child_weight: 9	• min_child_weight: 8	• min_child_weight: 5
	• eta: 0.38903	• eta: 0.08403	• eta: 3.78013e-07
	• gamma: 0.00134	• gamma: 3.2e-05	• gamma: 7.99526e-07
	• grow_policy: depthwise	• grow_policy: depthwise	• grow_policy: depthwise
LightGBM	• learning_rate: 0.00223	• learning_rate: 0.01233	• learning_rate: 0.01144
	• n_estimators: 843	• n_estimators: 928	• n_estimators: 541
	• num_leaves: 35	• num_leaves: 31	• num_leaves: 27
	• max_depth: 5	• max_depth: 4	• max_depth: 10
	• min_data_in_leaf: 22	• min_data_in_leaf: 27	• min_data_in_leaf: 38
Catboost	• learning_rate: 0.04165	• learning_rate: 0.04226	• learning_rate: 0.05712
	• iterations: 890	• iterations: 715	• iterations: 683
	• depth: 5	• depth: 5	• depth: 6
	• l2_leaf_reg: 0.03197	• l2_leaf_reg: 0.02854	• l2_leaf_reg: 1.16369

Table 8 For Mutual Information-based Fitness function

Detection Model	Best Hyperparameter values for each model		
	NSL-KDD	UNSW-NB15	CSE-CIC-IDS2018
DT	• splitter: random	• splitter: best	• splitter: best
	• criterion: gini	• criterion: entropy	• criterion: gini
	• max_depth: 15	• max_depth: 7	• max_depth: 9
	• min_samples_split: 24	• min_samples_split: 31	• min_samples_split: 11
	• min_samples_leaf: 4	• min_samples_leaf: 16	• min_samples_leaf: 7
	• min_weight_fraction_leaf: 0.00033	• min_weight_fraction_leaf: 0.01069	• min_weight_fraction_leaf: 0.00016
	• min_impurity_decrease: 0.00053	• min_impurity_decrease: 0.00019	• min_impurity_decrease: 3.6e-08
	• max_leaf_nodes: 14	• max_leaf_nodes: 19	• max_leaf_nodes: 29
LR	• C: 0.00794	• C: 26.72402	• C: 0.09749
	• solver: lbfgs	• solver: lbfgs	• solver: lbfgs
	• penalty: l2	• penalty: l2	• penalty: l2
	• tol: 0.00066	• tol: 0.00015	• tol: 0.00909
	• max_iter: 903	• max_iter: 285	• max_iter: 1000
KNN	• algorithm: auto	• algorithm: brute	• algorithm: brute
	• n_neighbors: 6	• k_n_neighbors: 4	• k_n_neighbors: 3
	• weights: uniform	• weights: uniform	• weights: distance
	• p: 2	• p: 1	• p: 2
	• leaf_size: 6	• leaf_size: 6	• leaf_size: 2

Table 8 (continued)

Detection Model	Best Hyperparameter values for each model		
	NSL-KDD	UNSW-NB15	CSE-CIC-IDS2018
SVM	• C: 9.35784	• C: 14.33070	• C: 1.58833
	• kernel: rbf	• kernel: poly	• kernel: rbf
DNN	• num_layers: 4	• num_layers: 2	• num_layers: 2
	• units_0: 116	• units_0: 42	• units_0: 100
	• units_1: 65	• units_1: 78	• units_1: 112
	• units_2: 51	• dropout_rate: 0.27976	• dropout_rate: 0.22691
	• units_3: 70	• learning_rate: 0.00290	• learning_rate: 0.00261
	• dropout_rate: 0.30623	• batch_size: 64	• batch_size: 16
	• learning_rate: 0.00016		
	• batch_size: 16		
CNN	• filters: 48	• filters: 28	• filters: 31
	• kernel_size: 4	• kernel_size: 3	• kernel_size: 5
	• num_dense_units: 119	• num_dense_units: 125	• num_dense_units: 72
	• dropout_rate: 0.15852	• dropout_rate: 0.28289	• dropout_rate: 0.28022
	• learning_rate: 0.00019	• learning_rate: 0.00142	• learning_rate: 0.00177
	• batch_size: 16	• batch_size: 16	• batch_size: 32
RF	• n_estimators: 22	• n_estimators: 63	• n_estimators: 27
	• max_depth: 12	• max_depth: 14	• max_depth: 14
	• min_samples_split: 20	• min_samples_split: 5	• min_samples_split: 14
	• min_samples_leaf: 11	• min_samples_leaf: 5	• min_samples_leaf: 3
Xgboost	• booster: gbtree	• booster: dart	• booster: dart
	• lambda: 0.00028	• lambda: 0.34347	• lambda: 0.00027
	• alpha: 1.1e-07	• alpha: 0.00066	• alpha: 2.9e-08
	• subsample: 0.96331	• subsample: 0.98447	• subsample: 0.89111
	• colsample_bytree: 0.97232	• colsample_bytree: 0.76967	• colsample_bytree: 0.93290
	• early_stopping_rounds: 30	• early_stopping_rounds:14	• early_stopping_rounds: 26
	• n_estimators: 96	• n_estimators: 96	• n_estimators: 16
	• max_depth: 9	• max_depth: 5	• max_depth: 9
	• min_child_weight: 8	• min_child_weight: 7	• min_child_weight: 3
	• eta: 1.0e-08	• eta: 0.87030	• eta: 0.32162
	• gamma: 1.7e-07	• gamma: 0.00894	• gamma: 3.2e-06
	• grow_policy: depthwise	• grow_policy: depthwise	• grow_policy: lossguide
LightGBM	• learning_rate: 0.02191	• learning_rate:0.00241	• learning_rate: 0.07838
	• n_estimators: 604	• n_estimators: 215	• n_estimators: 416
	• num_leaves: 20	• num_leaves: 40	• num_leaves: 21
	• max_depth: 4	• max_depth: 8	• max_depth: 8
	• min_data_in_leaf: 48	• min_data_in_leaf: 44	• min_data_in_leaf: 47
Catboost	• learning_rate: 0.08224	• learning_rate:0.00617	• learning_rate: 0.28400
	• iterations: 936	• iterations: 962	• iterations: 865
	• depth: 7	• depth: 10	• depth: 9
	• l2_leaf_reg: 0.33265	• l2_leaf_reg: 0.00208	• l2_leaf_reg: 0.13820

number generation in this module gives a better result for each detection model. Figure 9b shows the convergence graph between the fitness score and no. of generations (or iterations) for the xgboost model. The graph shows stable behavior from 60 to 64 iterations for the UNSW-NB15 and CIC-IDS2018 datasets and from 123 to 128 iterations for the NSL-KDD dataset. Tables 7 and 8 outline the best hyperparameter values for correlation-based and mutual information-based fitness functions, respectively, corresponding to each detection model.

Algorithm 5 Classification

```
    Input: $D_{Train}$, $D_{Test}$, $H_T$, $M_B$ or $M_E$
    Output: Classification Report for Base or Ensemble Model ($P_B$ or $P_E$)
1   procedure MULTI-CLASS CLASSIFICATION
2       begin
3       columns ← $D_{Train}$.shape[1]
4       x_train ← $D_{Train}$[:, :columns-1], y_train ← $D_{Train}$[:, -1:], x_test ← $D_{Test}$[:, :columns-1], y_test ←
        $D_{Test}$[:, -1:]
5       LM ← classifier(x_train, y_train, $H_T$)          // classifier ∈ {$M_B$ or $M_E$}
6       y_pred← LM.predict(x_test, $H_T$)
7       $P_B$ or $P_E$ ← classification_report(y_test, y_pred)
8       return $P_B$ or $P_E$
9       end
10  end procedure
```

5.4 Classification

In this module, two different categories of detection models, such as base and ensemble models, are employed to identify distinct attacks and normal network traffic behavior. Twelve different detection models are used, six of which are included in the base model category and the remaining six in the ensemble model category. Algorithm 5 describes the classification modeling module of the proposed model.

Base Models = {Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Dense Neural Network (DNN), 1D-Convolution Neural Network (1D-CNN)}.

Ensemble Models = {Random Forest (RF), Xgboost, LightGBM, Catboost, Majority Voting, Mean Voting}.

The majority and mean voting classifiers are an ensemble of four detection models: RF, LR, KNN, and Xgboost.

5.5 Applications of the proposed method

By implementing the proposed technique, the IDS significantly enhances its efficiency and accuracy, transforming it into a viable solution for real-time attack detection in dynamic and resource-limited environments. The proposed method offers a range of applications, including:

(i) *Performance enhancement:* When feature selection is combined with hyperparameter tuning, it results in models that are not only lightweight but also highly accurate. This approach assures that the IDS can operate effectively, even in environments with limited computational resources. Specially in IoT-based organizations where resource constraints are one of the challenges, this lightweight IDS is highly applicable.

(ii) *Scalability:* A lightweight IDS can scale seamlessly across large networks or multiple devices without substantially increasing computational load.

(iii) *Real-time capabilities:* Enhancing real-time detection capabilities by reducing latency introduced by high-dimensional data is essential for promptly identifying and mitigating security threats.

(iv) *Cost efficiency:* Optimizing the IDS's computational efficiency reduces hardware and energy consumption costs. This approach is ideal for organizations seeking to implement highly cost-effective security solutions.

6 Experimental results and discussion

This section discusses the experimental setup and different performance metrics used in this paper to evaluate the proposed model's effectiveness. Based on the classification metrics, performance is analyzed for each detection model on all three datasets one by one. Furthermore, the

Table 9 Performance Metrics

Metrics	Formula
Accuracy	$\frac{\sum_{i=1}^{c} \text{True Positive}_i}{S}$
Weighted Precision	$\frac{\sum_{i=1}^{c} (S_i * \text{Precision}_i)}{S}$
Weighted Recall	$\frac{\sum_{i=1}^{c} (S_i * \text{Recall}_i)}{S}$
Weighted F1-Score	$\frac{\sum_{i=1}^{c} (S_i * \text{F1-Score}_i)}{S}$
False Alarm Rate	$\frac{\text{False Positive}_i}{\text{False Positive}_i + \text{True Negative}_i}$
Prediction Time	Stop_time − Start_time

Table 10 Classification performance after pre-processing module

Dataset	Metrics	Base Classifier						Ensemble Classifier					
		DT	LR	KNN	SVM	DNN	CNN	RF	Xgboost	LightGBM	Catboost	Majority Voting	Mean Voting
NSL-KDD	Accuracy(%)	69.49	72.57	74.80	71.86	72.68	72.45	72.22	33.08	73.57	**75.29**	71.96	73.20
	Precision(%)	65.21	72.16	74.14	76.21	66.93	67.22	79.19	10.95	72.63	**81.16**	67.38	67.24
	Recall(%)	69.49	72.57	74.80	71.86	72.68	72.45	72.22	33.08	73.57	**75.29**	71.96	73.20
	F1-Score(%)	64.20	67.68	70.17	66.96	67.74	67.52	67.56	16.45	69.85	**71.24**	67.03	68.19
	FAR(%)	10.44	9.20	8.50	9.62	9.39	9.41	9.50	20.00	8.77	**8.47**	9.66	9.17
	Prediction Time(sec)	0.017	0.023	16.87	16.78	1.502	2.666	0.255	0.031	4.997	**0.216**	17.08	16.96
UNSW-NB15	Accuracy(%)	58.94	66.14	66.68	63.91	71.61	70.58	**76.18**	58.89	65.92	63.66	69.20	71.91
	Precision(%)	70.36	70.64	81.83	74.62	76.75	80.40	**82.89**	76.02	70.99	62.27	82.77	83.55
	Recall(%)	58.94	66.14	66.68	63.91	71.61	70.58	**76.18**	58.89	65.92	63.66	69.20	71.91
	F1-Score(%)	61.31	67.12	67.78	65.78	71.69	71.43	**76.53**	61.28	66.96	60.40	69.59	72.77
	FAR(%)	4.73	4.12	3.57	4.28	3.38	3.31	**2.70**	4.18	4.28	3.90	3.31	3.12
	Prediction Time(s)	0.018	0.012	26.56	401.94	10.46	10.39	**0.92**	1.424	26.34	0.518	26.61	22.31
CIC-IDS2018	Accuracy(%)	89.71	89.18	89.96	89.43	89.42	89.51	89.88	90.11	**90.56**	90.40	90.21	90.50
	Precision(%)	93.70	92.71	90.18	92.74	93.67	93.23	91.59	93.14	**92.03**	91.23	93.46	91.37
	Recall(%)	89.70	89.17	89.96	89.43	89.41	89.50	89.87	90.11	**90.55**	90.39	90.20	90.49
	F1-Score(%)	90.09	89.57	90.05	89.84	89.81	89.90	90.22	90.49	**90.86**	90.63	90.58	90.73
	FAR(%)	2.01	2.14	2.18	2.10	2.10	2.06	2.08	1.97	**1.94**	2.02	1.93	2.00
	Prediction Time(s)	9.775	0.799	65.27	137.26	10.463	5.318	1.055	1.993	**4.010**	3.209	69.38	60.022

performance of the proposed model is also compared with different traditional dimensionality reduction techniques and similar state-of-the-art works.

6.1 Experimental setup

This paper performs all the experiments using Python programming with sklearn, matplotlib, keras, and tensorflow-based libraries. The experiments are performed on the local server by launching jupyter notebook on the anaconda. Here, the advantage of Google Colab has been taken for its high speed and GPU. The system specification of the machine where all the experiments are performed is as follows: RAM- 12.0 GB, Processor- Intel(R) Core(TM) i5-7200U CPU @2.50 GHz 2.70 GHz, and system type is the 64-bit operating system, $\times$64-based processor.

6.2 Performance metrics

Table 9 shows the performance metrics used in this paper to evaluate the effectiveness of the proposed model. Here, six different classification metrics are utilized: accuracy, weighted precision, weighted recall, weighted f1-score, false alarm rate, and prediction time. Where S and S_i represent the total no. of samples and the total no. of samples of the ith class, respectively, while C represents the total no. of classes in the dataset. The formulas for computing $\mathrm{F1-Score_i}$, $\mathrm{Precision_i}$, and $\mathrm{Recall_i}$ are discussed (Sect. 5.3) in Eqs. (20–22), respectively. Where True Positive denotes correctly identified samples in the attack class, True Negative represents correctly identified samples in the normal class, False Positive represents misclassified non-attacks as attacks, and False Negative outlines misclassified attacks as non-attacks.

6.3 Performance analysis

In this section, the proposed model's performance is analyzed sequentially. This means that after applying each module of the proposed framework one by one, the effectiveness of the proposed model is evaluated.

6.3.1 Detection model's performance after pre-processing module of the proposed model

Table 10 shows the performance of all the detection models (discussed in Sect. 5.4) in terms of classification metrics (discussed in Sect. 6.2) on all three datasets after applying the preprocessing module of the proposed framework (discussed in Sect. 5.1.2). Table 10 shows that performance on the CIC-IDS2018 dataset is better compared to the NSL-KDD and UNSW-NB15 datasets. It is also noticed that Catboost in the case of NSL-KDD, RF in the case of UNSW-NB15, and LightGBM in the case of CIC-IDS2018 datasets offer better performance than other detection models. The best results by the specific detection models are indicated by the numbers highlighted in bold in the Tables 10, 11, 12, 13, and 15.

6.3.2 Detection model's performance after applying phase 1 feature selection module of the proposed model

Table 11 shows the detection model's performance after applying the PSO-based feature selection module (discussed in Sect. 5.2.1) on all three datasets. The detection models' performance is significantly improved compared to the previous module on the NSL-KDD and UNSW-NB15 datasets. Here, the Xgboost model shows a significant improvement.

6.3.3 Detection model's performance after applying phase 2 feature selection module of the proposed model

In this phase, the performance is evaluated utilizing two different metrics in the fitness function of ACO-based feature selection (discussed in Sect. 5.2.2) such as (a) Mutual Information, (b) Correlation Metric.

(a) Fitness function: mutual information based

Table 12 shows the classification performance after stacking PSO with the ACO-based feature selection on all three datasets utilizing mutual information in the ACO's fitness function. The table shows a remarkable improvement after applying this module in the proposed framework. DNN in the case of NSL-KDD, RF in the case of UNSW-NB15, and LightGBM, as well as majority voting in the case of CIC-IDS2018, outperform other detection models.

(b) Fitness function: correlation based

Table 13 shows the classification performance after stacking PSO with the ACO-based feature selection on three datasets utilizing a correlation metric in the ACO's fitness function. From Table 13, it is observed that DNN and Catboost in the case of NSL-KDD, Xgboost, and Catboost in the case of UNSW-NB15, and LightGBM and mean voting in the case of CIC-IDS2018 give impressive detection performance. The reason behind improved performance compared to the previous phase is that here, noisy and correlated features are removed, which shows a high correlation with other features in the data, while only those features are retained, which shows a high correlation with the target class. Thus, the noisy features from the data are removed, hence, performance improvement after training the model on filtered with crucial features of data.

6.3.4 Detection model's performance after applying the hyperparameter tuning module of the proposed model

Tables 14 and 15 demonstrate the classification performance after utilizing all four major modules of the proposed framework in the case of mutual information-based and correlation-based, respectively, on three datasets. The tables show the result after stacking PSO + ACO + GA on each detection model.

GA-based hyperparameter tuning iteratively searches for the best hyperparameters corresponding to each detection model. Table 14 shows that xgboost performs better than others on the UNSW-NB15 and CIC-IDS2018 datasets, while catboost outperforms others on the NSL-KDD dataset. Moreover, from Table 15, it is scrutinized that xgboost and catboost give the best results on all three datasets. Both Tables 14 and 15 summarize that stacking of PSO + ACO + GA in the proposed framework gives the best results. The performance is significantly better in the case of correlation metric than mutual information on all the datasets for almost each detection model. The proposed model demonstrates notably superior performance on the NSL-KDD and UNSW-NB15 datasets while achieving comparatively stronger results on the CIC-IDS2018 dataset. From the results of the detection model, it is obvious that the application of PSO-ACO stacked with GA demonstrates a highly improved performance.

6.4 Results and discussion

After the preprocessing, the data is split into training and testing sets with the number of features 42, 44, and 84 for the NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively. At this point, it is observed in Table 10 that Catboost, RF, and LightGBM give better performance than other detection models as 75.29%, 76.18%, and 90.56%, respectively, on NSL-KDD, UNSW-NB15, and CIC-IDS2018 data. The performance is very low at this stage.

Feature selection is implemented in the next module on the preprocessed data, and it gives an outstanding result as it removes all the correlated, redundant, and irrelevant features. It keeps only information-rich features through the proposed bi-phase feature selection technique in the proposed framework. In the first phase, PSO-based feature selection reduces the feature space by considering the imbalanced nature of the data. The value of the weight parameter $\gamma = 1$ gives the optimal performance, and the geometric mean of all nine detection models (including DT, RF, Xgboost, LightGBM, Catboost, SVM, KNN, DNN, and CNN) is summed. Here, the activation function of the DNN and CNN model considers 'relu' (at the hidden layer) and 'softmax' (at the last layer since multi-class

Table 11 Classification Performance after applying PSO-Based feature selection Module

Dataset	Metrics	Base Classifier						Ensemble Classifier					
		DT	LR	KNN	SVM	DNN	CNN	RF	Xgboost	LightGBM	Catboost	Majority Voting	Mean Voting
NSL-KDD	Accuracy (%)	72.79	74.19	74.71	74.08	72.38	72.52	72.24	74.21	75.21	**76.28**	72.57	73.24
	Precision (%)	73.34	67.72	74.15	76.13	66.67	75.75	79.65	77.05	81.24	**81.62**	80.03	80.11
	Recall (%)	72.79	74.19	74.71	74.08	72.38	72.52	72.24	74.21	75.21	**76.28**	72.57	73.24
	F1-Score (%)	68.74	69.17	70.18	69.22	67.38	67.42	67.49	70.20	70.97	**72.54**	67.79	68.47
	FAR (%)	9.17	8.74	8.51	8.83	9.22	9.34	9.52	8.79	8.49	**8.12**	9.44	9.13
	Prediction Time(s)	0.009	0.012	6.99	14.14	1.077	1.066	0.062	0.009	0.140	**0.039**	7.500	6.467
UNSW-NB15	Accuracy (%)	63.11	75.20	72.28	64.53	77.11	76.84	**83.62**	63.19	77.54	77.59	74.65	72.99
	Precision (%)	65.56	80.86	77.13	81.75	84.80	79.23	**84.48**	57.98	83.42	83.96	81.64	79.07
	Recall (%)	63.11	75.20	72.28	64.53	77.11	76.84	**83.61**	63.19	77.54	77.59	74.65	72.99
	F1-Score (%)	59.73	74.06	73.76	57.99	75.96	75.56	**83.45**	59.46	78.23	77.68	74.46	73.45
	(%)	4.11	2.77	3.27	3.98	2.53	2.58	**2.17**	3.96	2.55	2.58	2.97	3.01
	Prediction Time(s)	0.007	0.011	20.93	194.18	2.744	2.718	**0.475**	0.051	7.372	0.147	21.23	19.07
CICIDS2018	Accuracy (%)	89.83	89.23	89.98	89.44	89.44	89.57	90.34	90.25	**90.56**	90.37	90.34	90.51
	Precision (%)	93.61	92.85	90.17	92.88	92.65	93.61	92.47	92.35	**93.09**	92.67	93.26	91.54
	Recall (%)	89.83	89.22	89.98	89.43	89.43	89.56	90.33	90.25	**90.56**	90.37	90.33	90.51
	F1-Score (%)	90.22	89.61	90.06	89.83	89.84	89.96	90.69	90.60	**90.92**	90.73	90.71	90.77
	FAR (%)	1.99	2.13	2.18	2.09	2.06	2.04	1.95	1.96	**1.88**	1.94	1.91	1.98
	Prediction Time (s)	0.414	0.437	10.75	137.01	2.671	3.199	0.559	0.627	**2.228**	0.861	11.94	10.86

Table 12 Classification Performance after stacking of PSO + ACO in Case of Mutual Information

Dataset	Metrics	Base Classifiers						Ensemble Classifier					
		DT	LR	KNN	SVM	DNN	CNN	RF	Xgboost	LightGBM	Catboost	Majority Voting	Mean Voting
NSL-KDD	Accuracy(%)	76.27	78.71	77.67	79.09	**79.80**	72.79	76.37	78.88	75.23	79.19	77.09	79.15
	Precision(%)	75.93	81.27	80.53	81.04	**80.54**	67.29	81.02	81.87	77.33	81.84	81.00	82.03
	Recall(%)	76.27	78.71	77.67	79.09	**79.80**	72.79	76.37	78.88	75.23	79.19	77.09	79.15
	F1-Score(%)	74.30	76.62	74.59	76.05	**77.05**	67.68	73.86	77.00	73.02	77.78	74.40	76.85
	FAR(%)	7.26	6.43	7.18	6.56	**6.17**	9.32	7.90	6.80	8.00	6.72	7.57	6.72
	Prediction Time(s)	0.005	0.008	6.00	12.40	**1.03**	1.00	0.033	0.005	0.128	0.027	6.32	5.14
UNSW-NB15	Accuracy(%)	71.97	78.84	79.61	73.35	81.11	79.29	**85.41**	78.96	80.10	82.09	75.67	74.38
	Precision(%)	76.43	81.14	80.01	82.29	85.65	82.04	**86.33**	77.83	81.62	84.36	82.61	80.75
	Recall(%)	71.97	78.84	79.60	73.35	81.10	79.29	**85.41**	78.96	80.10	82.09	75.67	74.38
	F1-Score(%)	72.41	79.97	79.80	77.56	83.31	80.64	**85.86**	78.32	78.39	83.20	75.44	75.47
	FAR(%)	3.29	1.22	2.41	2.32	2.24	2.12	**1.98**	2.97	2.48	1.61	2.84	2.93
	Prediction Time(s)	0.006	0.010	18.09	180.26	2.271	2.66	**0.332**	0.020	5.652	0.131	19.39	17.215
CICIDS2018	Accuracy(%)	90.30	90.92	91.61	89.43	89.87	90.44	90.46	91.21	**91.55**	90.43	**91.93**	91.54
	Precision(%)	90.31	92.98	92.96	92.98	93.55	93.73	90.79	92.67	**93.60**	92.91	**93.86**	91.83
	Recall(%)	90.30	90.91	91.60	89.43	89.86	90.44	90.46	91.20	**91.54**	90.43	**91.92**	91.53
	F1-Score(%)	90.31	91.93	92.27	89.82	91.66	92.05	90.58	91.92	**92.55**	90.87	**92.48**	91.67
	FAR(%)	1.73	2.05	2.15	2.04	1.99	1.99	1.89	1.84	**1.81**	1.82	**1.85**	1.89
	Prediction Time(s)	0.365	0.388	9.72	129.33	2.565	3.178	0.547	0.601	**1.846**	0.748	**11.15**	10.44

Table 13 Classification Performance after stacking of PSO + ACO in case of Correlation Metric

Dataset	Metrics	Base Classifiers						Ensemble Classifier					
		DT	LR	KNN	SVM	DNN	CNN	RF	Xgboost	LightGBM	Catboost	Majority Voting	Mean Voting
NSL-KDD	Accuracy(%)	76.64	76.80	80.22	80.56	**81.49**	78.48	77.18	79.26	77.15	**81.38**	78.65	80.32
	Precision(%)	76.92	76.54	79.62	79.53	**79.96**	76.79	79.58	80.80	81.41	**82.60**	80.25	80.74
	Recall(%)	76.64	76.80	80.22	80.56	**81.49**	78.48	77.18	79.26	77.15	**81.38**	78.65	80.32
	F1-Score(%)	74.89	74.12	78.25	78.91	**79.92**	77.62	75.11	77.49	79.22	**80.15**	76.68	78.39
	FAR(%)	7.33	6.93	6.16	6.02	**5.46**	7.32	7.51	6.64	8.32	**5.96**	6.95	6.26
	Prediction Time(s)	0.004	0.005	5.933	11.07	**1.002**	1.058	0.053	0.007	0.129	**0.031**	5.998	5.197
UNSW-NB15	Accuracy(%)	86.64	79.44	79.98	78.69	82.60	79.98	87.69	**88.31**	78.01	**88.81**	87.90	87.58
	Precision(%)	88.38	80.97	84.11	83.47	84.99	80.33	89.01	**89.66**	86.38	**89.52**	89.29	87.55
	Recall(%)	86.64	79.44	79.98	78.69	82.60	79.98	87.69	**88.31**	78.01	**88.81**	87.90	87.58
	F1-Score(%)	87.05	77.99	80.91	77.08	83.77	80.15	87.25	**87.83**	81.98	**87.76**	86.67	86.92
	FAR(%)	1.45	2.32	2.16	2.41	1.94	2.14	1.34	**1.28**	2.01	**1.25**	1.34	1.37
	Prediction Time(s)	0.004	0.006	17.00	176.17	2.25	10.40	0.351	**0.043**	6.343	**0.139**	17.43	16.94
CICIDS2018	Accuracy(%)	90.26	90.17	90.66	91.41	92.41	92.30	93.53	93.29	**94.55**	94.30	92.42	**94.51**
	Precision(%)	90.27	93.51	91.00	93.00	93.35	94.34	93.89	94.73	**95.62**	95.07	91.09	**95.33**
	Recall(%)	90.25	90.16	90.65	91.40	92.40	92.30	93.53	93.28	**94.54**	94.29	92.41	**94.51**
	F1-Score(%)	90.26	91.80	90.82	92.19	92.87	93.30	93.70	93.99	**95.07**	94.67	91.75	**94.92**
	FAR(%)	1.14	2.05	1.87	1.91	1.78	1.37	1.14	1.21	**1.15**	1.38	1.03	**1.10**
	Prediction Time(s)	0.384	0.360	9.48	125.62	2.28	2.87	0.474	0.522	**1.760**	0.801	10.240	**9.88**

Table 14 Classification Performance after stacking of PSO + ACO + GA in case of Mutual Information

Dataset	Metrics	Base Classifiers						Ensemble Classifier					
		DT	LR	KNN	SVM	DNN	CNN	RF	Xgboost	LightGBM	Catboost	Majority Voting	Mean Voting
NSL-KDD	Accuracy(%)	80.18	82.15	79.71	82.76	82.36	78.69	79.96	83.21	80.74	84.65	81.59	82.95
	Precision(%)	79.68	84.90	80.98	83.22	84.46	80.08	83.81	84.43	82.19	85.80	84.32	84.62
	Recall(%)	80.18	82.15	79.71	82.76	82.36	78.69	79.96	83.21	80.74	84.65	81.59	82.95
	F1-Score(%)	79.92	79.32	80.33	82.98	83.39	74.37	81.83	83.81	81.45	85.22	82.92	83.77
	FAR(%)	6.98	5.50	6.39	5.40	5.95	6.82	6.78	6.13	7.26	5.95	6.53	5.51
	Prediction Time(s)	0.004	0.006	5.77	5.98	1.007	0.916	0.029	0.004	0.106	0.014	5.79	5.56
UNSW-NB15	Accuracy(%)	86.31	81.70	84.86	83.33	85.38	85.75	87.59	88.63	86.56	87.93	87.63	87.60
	Precision(%)	86.47	83.70	84.02	81.69	86.58	84.39	87.28	88.72	88.02	87.85	87.27	87.42
	Recall(%)	86.31	81.70	84.86	83.33	85.38	85.75	87.59	88.63	86.56	87.93	87.63	87.60
	F1-Score(%)	86.38	82.68	84.04	80.37	85.97	83.02	87.37	88.07	87.28	87.30	86.26	86.94
	FAR(%)	1.52	0.021	1.73	1.95	1.72	1.68	1.39	1.31	1.14	1.39	1.45	1.42
	Prediction Time(s)	0.006	0.008	15.70	142.29	1.55	1.612	0.275	0.015	2.91	0.107	15.86	15.44
CIC-IDS2018	Accuracy(%)	92.35	92.14	93.24	91.42	91.45	92.42	93.53	94.71	92.82	93.66	93.68	93.81
	Precision(%)	94.38	95.14	94.55	93.36	94.28	95.99	94.73	95.26	94.52	95.33	94.01	94.48
	Recall(%)	92.34	92.14	93.24	91.41	91.45	92.41	93.52	94.70	92.81	93.65	93.68	93.80
	F1-Score(%)	93.34	93.61	93.89	92.37	92.84	94.16	94.12	94.97	93.65	94.48	93.84	94.13
	FAR(%)	1.18	1.37	2.11	1.78	1.27	1.19	1.13	1.06	1.45	1.76	1.38	1.22
	Prediction Time(s)	0.286	0.262	7.84	114.89	1.803	2.879	0.471	0.524	1.35	0.571	9.37	8.51

Table 15 Classification Performance after stacking of PSO + ACO + GA in case of Correlation Metric

Dataset	Metrics	Base Classifiers						Ensemble Classifier					
		DT	LR	KNN	SVM	DNN	CNN	RF	Xgboost	LightGBM	Catboost	Majority Voting	Mean Voting
NSL-KDD	Accuracy(%)	88.27	89.35	82.31	82.55	85.46	82.16	89.06	**90.38**	85.43	**90.56**	81.07	81.14
	Precision(%)	89.71	91.46	80.64	80.57	81.86	80.65	90.76	**92.33**	87.85	**91.47**	81.86	81.23
	Recall(%)	88.27	89.35	82.31	82.55	85.46	82.16	89.06	**90.38**	85.43	**90.56**	81.07	81.14
	F1-Score(%)	88.15	90.39	81.43	81.54	83.62	80.18	89.90	**91.34**	86.62	**91.01**	79.41	79.34
	FAR(%)	5.25	6.77	6.11	5.98	5.17	5.32	6.87	**5.21**	7.00	**4.89**	5.14	5.01
	Prediction Time(s)	0.002	0.004	5.137	2.569	0.998	0.981	0.043	**0.005**	0.107	**0.022**	5.277	4.749
UNSW-NB15	Accuracy(%)	91.59	91.71	84.86	83.33	85.25	85.32	90.31	**92.63**	82.52	89.93	90.63	91.60
	Precision(%)	93.28	92.70	84.12	81.69	85.42	84.07	92.47	**93.72**	85.72	90.85	92.27	90.42
	Recall(%)	91.59	91.70	84.86	83.33	85.25	85.32	90.30	**92.63**	82.52	89.93	90.63	91.60
	F1-Score(%)	92.42	92.19	84.04	80.37	82.38	82.59	91.37	**93.17**	84.08	90.38	91.44	91.00
	FAR(%)	1.02	1.63	1.73	1.95	1.73	1.73	1.09	**0.95**	1.42	1.17	1.02	1.12
	Prediction Time(s)	0.002	0.005	12.82	118.74	2.005	9.33	0.226	**0.029**	1.19	0.136	13.81	11.62
CICIDS2018	Accuracy(%)	96.39	92.47	94.02	93.79	95.58	95.21	96.37	**97.87**	97.96	**97.90**	96.58	97.55
	Precision(%)	96.37	94.22	94.48	94.50	96.28	96.00	96.41	**98.11**	98.19	**98.15**	96.58	97.75
	Recall(%)	96.38	92.47	94.01	93.79	95.58	95.21	96.36	**97.87**	97.95	**97.90**	96.57	97.54
	F1-Score(%)	96.37	93.33	94.13	93.94	95.69	95.33	96.38	**97.90**	97.99	**97.93**	96.58	97.58
	FAR(%)	0.80	1.49	1.25	1.28	0.88	0.95	0.78	**0.41**	0.39	**0.40**	0.74	0.49
	Prediction Time(s)	0.267	0.282	8.49	103.63	1.889	1.979	0.339	**0.434**	1.210	**0.710**	9.94	8.31

classification is performed), 0.2 as the dropout rate, and executes these models up to 25 epochs. After completing this phase, 25, 29, and 48 features are obtained on NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively. By trial and error methods, the values of these parameters are determined. In phase 1, Catboost, RF, and LightGBM again give better performance than other detection models as 76.28%, 83.62%, and 90.56%, respectively, on NSL-KDD, UNSW-NB15, and CIC-IDS2018 data, respectively, as depicted in Table 11. It shows improved performance compared to the previous results on preprocessed data, achieved by effectively addressing imbalanced data issues and removing irrelevant features. This phase leverages all nine models trained on the training set and efficiently selects features.

Now, in the second phase, the model is analyzed from two perspectives: correlation-based feature analysis and mutual information-based feature analysis. Correlation-based feature selection considers both feature-feature importance and class-feature. Based on these measures, the most crucial features are selected utilizing the ACO-based feature selection technique. In phase 2, (DNN & Catboost), (Xgboost & Catboost), and (LightGBM & Mean voting) give better performance than other detection models as (81.49% & 81.38%), (88.31% & 88.81%), and (94.55% & 94.51%) respectively on NSL-KDD, UNSW-NB15, and CIC-IDS2018 data respectively, as shown in Table 13. Mutual information measures the information gained by taking a feature subset, more the mutual-information value of the feature subset, most likely of that candidate feature subset selection by the ACO algorithm. Considering the mutual-information metric in the fitness function of the ACO algorithm offers the following performance as DNN (79.80%), RF (85.41%), and LightGBM & Majority voting (91.55% & 91.93%, respectively) on NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively, as shown in Table 12. After executing this phase, 15, 20, and 20 features are obtained on NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively. Since only a few features are selected for training the final detection model, it ensures a lightweight implementation. Better detection performance is utilizing correlation metrics against mutual information metrics, and as a result, correlation metrics offer a linear relationship, which is the more interpretable, computationally efficient, and redundancy-handling approach. It selects and evaluates features based on feature-feature and class-feature pairs. For this reason, it has a high impact on the overall performance of the model, while the mutual information metric in the fitness function selects features that might not always result in the best combination of features.

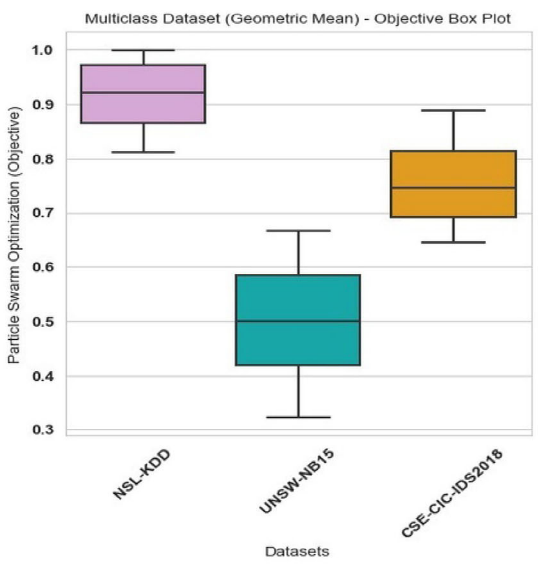
In the final module, the hyperparameters of each detection model are optimized using a nature-influenced genetic algorithm. The hyperparameter values of these detection models are shown in Tables 7 and 8. It is observed that 64, 64, and 128 generations give the optimal result against other values on the UNSW-NB15, CIC-IDS2018, and NSL-KDD datasets, respectively. Here, the population size is fixed, equal to 50 for all datasets. The performance after completing this module using the mutual information metric is as follows: Catboost achieved 84.65% accuracy, Xgboost achieved 88.63% accuracy, and Xgboost model achieved 94.71% accuracy on NSL-KDD, UNSW-NB15, and CIC-IDS2018 data, respectively, as shown in Table 14. The performance using the correlation metric is 90.38% and 90.56% accuracies for Xgboost and Catboost, 92.63% accuracy for Xgboost, and 97.87% and 97.90% accuracies for Xgboost and Catboost models on the same datasets as shown in Table 15. Xgboost outperforms other models across all three datasets due to its efficient handling of large datasets, capturing complex patterns through boosting, reducing overfitting with regularization, and quick data processing because of its parallelized implementation. The results show that the application of PSO + ACO + GA in the proposed framework improves performance. Furthermore, performance significantly improves when using the correlation metric compared to mutual information across all datasets for almost every detection model. Overall, the proposed model shows superior performance on the NSL-KDD and UNSW-NB15 datasets and achieves relatively strong results on the CIC-IDS2018 dataset.

6.5 Objective function analysis

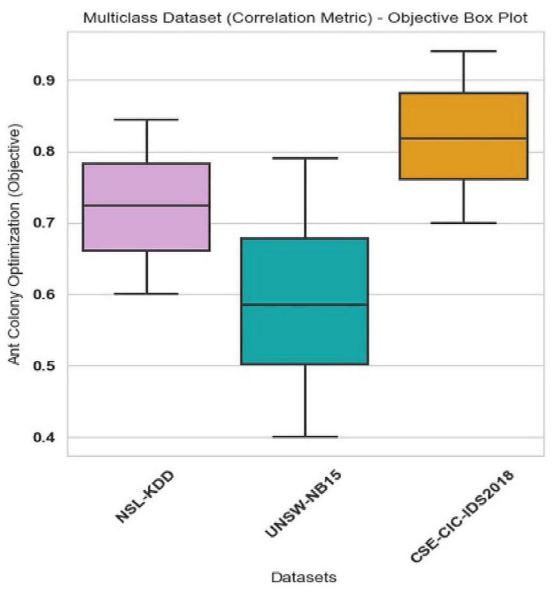
In this section, the objective function of each metaheuristic algorithm used in this paper is analyzed using convergence diagrams, box plots, and swarm plots. Finally, the outcomes of the objective function are analyzed in terms of best, worst, mean, median, standard deviation, and variance.

6.5.1 Convergence diagram

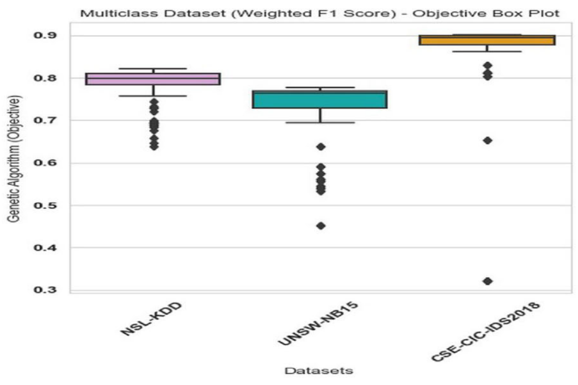
Figure 9 shows the convergence diagram of proposed metaheuristic algorithms such as PSO, ACO, and GA. Module 2 of the proposed model which discusses the feature selection in bi-phase utilizing PSO and ACO. Thus, the convergence diagram of PSO and ACO is combined in a single diagram, as shown in Fig. 9a. Here, the GA-based hyperparameter tuning module is applied, whose convergence diagram is shown in Fig. 9b. As a random algorithm, the converge history comparison on a single run is unfair. Thus, to reduce the effect of randomness, this paper fixes the seed value = 42 for each algorithm and performs the experiment with several runs or iterations (depending on



(a)



(b)



(c)

Fig. 10 Objective function box plots **a** PSO, **b** ACO, **c** GA

the selected algorithm), and holds the best fitness score for each optimization algorithm at every iteration.

In the case of the PSO algorithm, the objective function is the summation of the geometric mean of the nine detection models, so the result of the objective function lies within the range of [0,9]. The fitness score is normalized to lie within the range [0,10], especially for phase 1 of feature selection in the proposed framework. As observed from Fig. 9a, after 58, 95, and 79 iterations, the best fitness score value stopped increasing for NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively. Hence, this is the termination condition of the algorithms. Similarly, in the case of the ACO algorithm, two objective functions are considered for selecting the optimal feature subset, such as (i) correlation-based and (ii) mutual-information-based. Since the result based on the correlation function is better than the mutual information function, the correlation-based objective function in phase 2 of feature selection is analyzed here. The convergence diagram of Fig. 9a shows that the optimal fitness score value is obtained at 50, 60, and 50 iterations for NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively. Since the multi-class classification is explored, the fitness function in the genetic algorithm uses the weighted f1-score value. The best fitness score at each generation (or iteration) of the genetic algorithm is encountered for each detection model. Figure 9b analyses the fitness function of the xgboost algorithm within the correlation metric in the hyperparameter tuning module. The reason behind selecting the xgboost algorithm for the analysis is that it offers optimal results for all the datasets. It is obvious from the convergence diagram Fig. 9b that 128, 64, and 64 generations provide the best results for NSL-KDD, UNSW-NB15, and CIC-IDS2018 datasets, respectively.

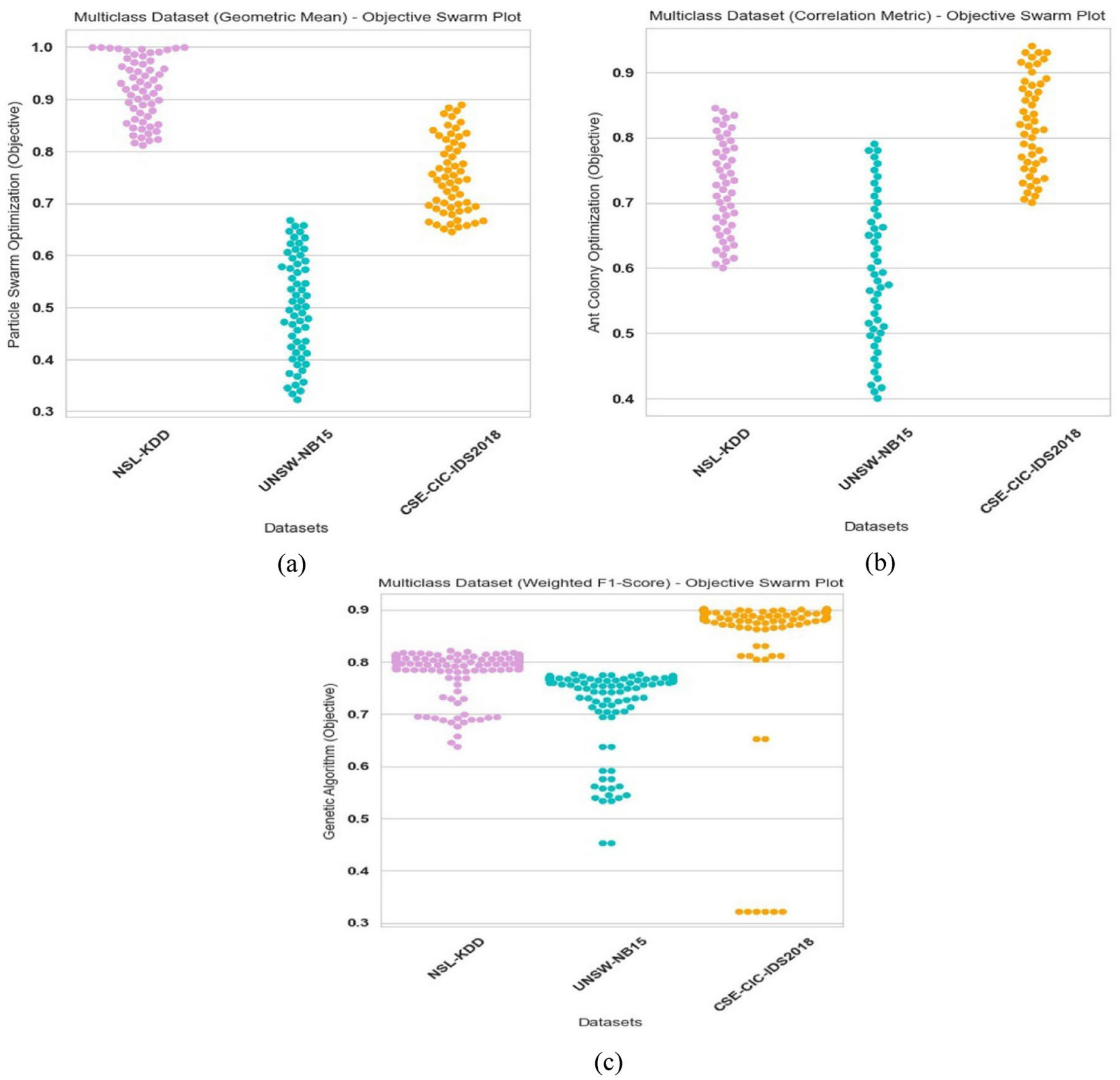


Fig. 11 Data diversity of objective function through swarm plots **a** PSO, **b** ACO, **c** GA

6.5.2 Box plot, and swarm plot

The paper analyses the objective function data distribution of each metaheuristic algorithm (used here) with the help of box plots. The box plots are analyzed with different runs for different algorithms in such a way that the PSO, ACO, and GA algorithms are analyzed with 95, 60, and 128 runs respectively. Figure 10a–c show the box plots of the objective function data for PSO, ACO, and GA respectively.

Similarly, the paper also analyses the data distribution of each metaheuristic algorithm's objective function (such as PSO, ACO, and GA) using swarm plots. It shows the diversity in the data. Here, the swarm plots are analyzed with different runs for different algorithms in such a way that the PSO, ACO, and GA algorithms are analyzed with 95, 60, and 128 runs, respectively. Figure 11 (a), (b), and (c) show the swarm plots of the objective function data for PSO, ACO, and GA, respectively.

6.5.3 Outcomes of objective function in terms of best, worst, mean, median, std, and var

Figure 12a–f shows the results of the metaheuristics objective function in terms of mean, best, worst, median, standard deviation (std), and variance (var). If a meta-heuristic algorithm's objective function is denoted as Z, then its best, worst, mean, median, standard deviation and variance are calculated utilizing the formula provided in Table 16. The symbols μ and $|I|$ indicate the mean and the number of iterations performed by the algorithm respectively.

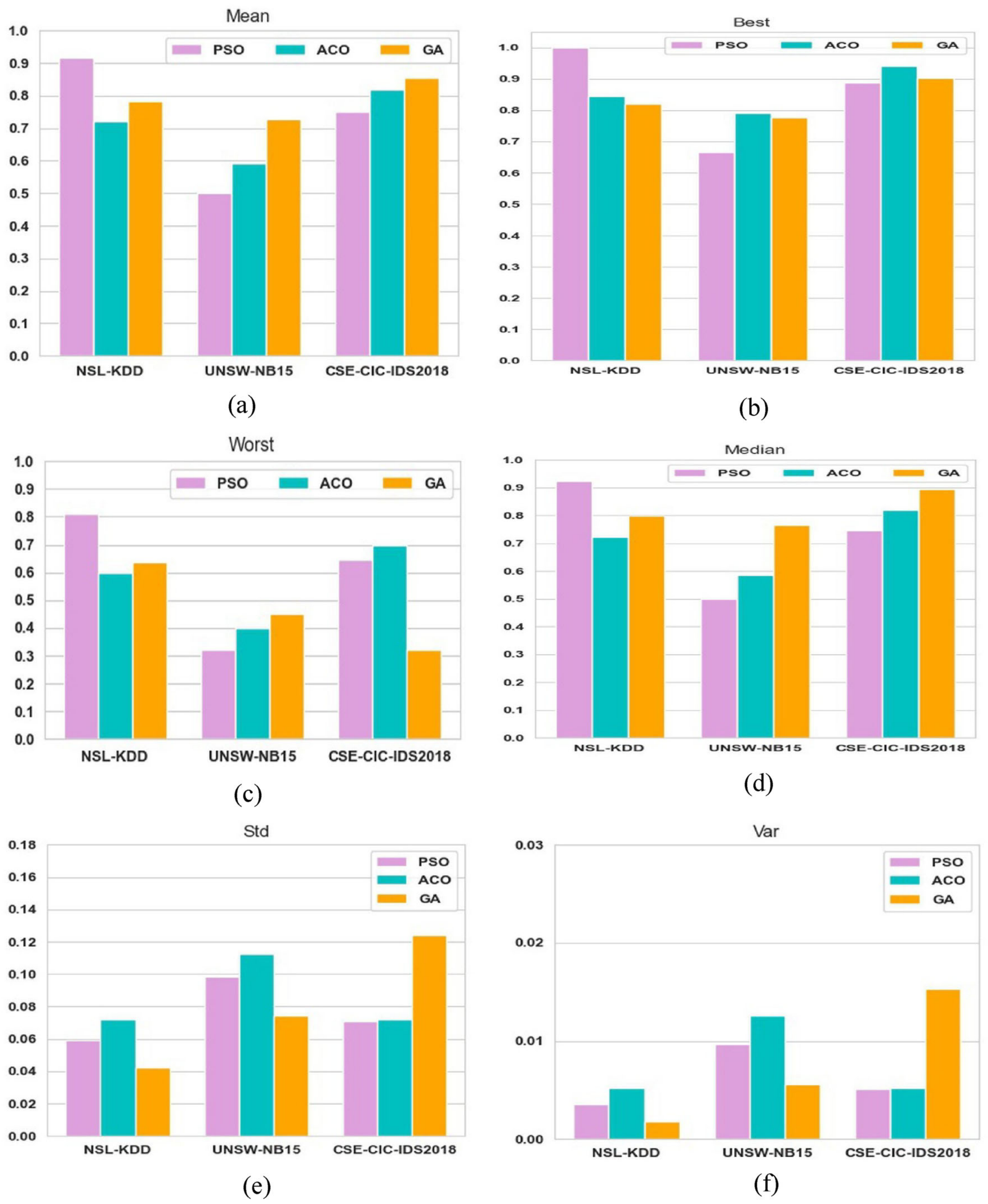


Fig. 12 Objective function in terms of **a** Mean, **b** Best, **c** Worst, **d** Median, **e** Standard deviation (Std), and **f** Variance (Var)

6.6 Comparative analysis

This section compares the proposed model's performance with that of other traditional dimensionality reduction techniques and state-of-the-art techniques.

6.6.1 Comparative analysis of the proposed method with other traditional dimensionality reduction techniques

Different dimensionality reduction techniques are utilized here for comparative purposes, such as principal component analysis (PCA), linear discriminant analysis (LDA), autoencoder (AE), information gain (IG), and Pearson

Table 16 Formula for objective function evaluation

Objective Function	Formula
Best	$Z_{Best} = Max_{i=1}^{\lvert I \rvert}(Z_i)$
Worst	$Z_{Worst} = Min_{i=1}^{\lvert I \rvert}(Z_i)$
Mean	$Z_{Mean} = \frac{1}{\lvert I \rvert}\sum_{i=1}^{\lvert I \rvert}(Z_i)$
Median	$Z_{Median} = \left(\frac{\lvert I \rvert+1}{2}\right)^{th} term$ or,$Z_{Median} = \frac{\left(\frac{\lvert I \rvert}{2}\right)^{th} term + \left(\frac{\lvert I \rvert+1}{2}\right)^{th} term}{2}$
Standard Deviation (Std)	$Z_{Std} = \sqrt{\frac{\sum_{i=1}^{\lvert I \rvert}(Z_i-\mu)^2}{\lvert I \rvert}}$
Variance (Var)	$Z_{Var} = \frac{\sum_{i=1}^{\lvert I \rvert}(Z_i-\mu)^2}{\lvert I \rvert}$

correlation (P.Corr.). Tables 17, 18, and 19 demonstrate the results after applying different dimensionality reduction techniques on the preprocessed form of NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets, respectively. It is observed from these tables that proposed framework outperforms other traditional dimensionality reduction techniques in terms of accuracy, precision, recall, f1-score, FAR, and prediction time. The reason behind the improved performance of the proposed method is that here, the most crucial and important features from the dataset are selected through two phases. It filters the features based on the imbalanced nature of the data using geometric mean in the objective function of the PSO algorithm. The best features subset is selected through this objective function. Now, these filtered data are provided as input to the second phase of the feature selection module, where the features are selected not only based on feature-feature correlation but also on the class-feature correlation. The feature subset selected has the highest correlation with target classes and the lowest correlation with the other features in the subset. This way, the proposed model selects the most important, and relevant features from the data and performs better than other dimensionality reduction methods.

6.6.2 Statistical validation

To demonstrate the enhanced performance of the proposed framework, a statistical analysis utilizing the two-tailed t-test hypothesis testing is conducted in this paper. Assumptions are made that the baseline method's f1-score is statistically identical to the value proposed in this work (null hypothesis H_0) and that the baseline method's f1-score differs statistically from the reported value (alternative hypothesis H_A). Ten times execution of the xgboost (for NSL-KDD and UNSW-NB15) and the catboost models (for CSE-CIC-IDS2018) are performed on each dataset to obtain the different outcomes. The significance threshold of the two-tailed t-test is selected as 5% (0.05) in this paper. Table 20 displays the p-value at the 0.05 significance level. Each entry in the table is less than 0.05. Thus, it shows the rejection of the null hypothesis in each case. The proposed model performance is not statistically identical to the respective baseline models. The performance of the proposed framework is statistically significant and does not happen by chance, as can be seen from the table where the p-value is significantly below the significance level.

6.6.3 Result interpretation of best model through SHAP analysis

The method of SHAP analysis has been acknowledged as a way to enhance transparency in evaluating model performance [42]. Figure 13a–c compute the feature importance using the xgboost algorithm for NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets. Through the experiment, it is observed that xgboost performs best in all three datasets. Thus, xgboost is considered for analyzing the feature importance and SHAP. It is noticed from Fig. 13a that 'level', 'dst_host_srv_count', 'dst_host_diff_srv_rate', and 'dst_host_count' are top-performing features followed by 'count', 'dst_bytes', and the remaining features are comparatively less important for the NSL-KDD dataset. Similarly, it is remarked from Fig. 13b that 'smean' is the top-most performing feature, followed by 'synack', 'sinpkt', 'dinpkt', and 'sjit' for the UNSW-NB15 dataset. In the same way, Fig. 13c shows the feature importance for the CSE-CIC-IDS2018 dataset. It is perceived from the fig that 'Fwd IAT Min' is the top performing feature, followed by 'Dst Port', and 'Flow Duration'. It is scrutinized that feature importance for the UNSW-NB15 dataset is comparatively higher than that of the NSL-KDD and CSE-CIC-IDS2018 datasets.

For analyzing the SHAP, features obtained after the proposed bi-phase feature selection method are utilized to determine the feature impact for each class in the dataset. Figure 14 (a), (b), and (c) show the feature impact of multi-class on the NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets, respectively. It is scrutinized from Fig. 14a that 'level' feature on the NSL-KDD dataset contributes approximately equal to all the five classes. Similarly, Fig. 14b suggests that 'smean' feature offers a high impact compared to others on the UNSW-NB15 dataset. However, it contributes equally to every class in the dataset. In the same way, the feature impact for the CSE-CIC-IDS2018 data is shown in Fig. 14c. It is noticed from Fig. 14c that feature 'day' provides a higher impact than other features in the dataset while it does not contribute equally to every class.

Table 17 Comparative analysis of different traditional dimensionality reduction techniques with the proposed approach on NSL-KDD

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
DT	PCA	77.41	80.10	77.41	74.84	7.18	0.003
	LDA	74.45	71.79	74.44	71.01	7.69	0.003
	AE	79.38	81.56	79.37	76.98	6.48	0.003
	IG	72.88	68.57	72.88	69.00	8.74	0.004
	P.Corr	78.33	80.69	78.33	75.77	6.75	0.007
	Proposed	88.27	89.71	88.27	88.15	5.25	0.002
RF	PCA	77.35	81.30	77.35	74.12	7.45	0.255
	LDA	74.68	72.88	74.68	71.12	7.66	0.341
	AE	78.21	81.63	78.21	74.24	7.24	0.257
	IG	74.70	77.98	74.69	69.90	8.69	0.278
	P.Corr	76.07	79.25	76.06	71.55	7.93	0.365
	Proposed	89.06	90.76	89.06	89.90	6.87	0.043
KNN	PCA	77.28	79.51	77.28	74.08	7.43	7.985
	LDA	75.56	73.21	75.56	72.01	7.44	10.670
	AE	78.17	79.44	78.17	74.86	7.15	9.628
	IG	77.31	78.33	77.31	73.96	7.63	29.88
	P.Corr	77.10	77.82	77.09	71.98	7.49	20.40
	Proposed	82.31	80.64	82.31	81.43	6.11	5.137
LR	PCA	79.04	80.84	79.04	76.02	7.48	0.004
	LDA	73.35	66.37	73.34	68.47	8.26	0.006
	AE	75.44	74.78	75.44	70.68	7.83	0.005
	IG	68.62	71.86	68.62	63.90	10.33	0.007
	P.Corr	79.76	81.74	79.76	75.32	7.16	0.017
	Proposed	89.35	91.46	89.35	90.39	6.77	0.004
SVM	PCA	78.06	78.28	78.05	74.58	6.96	15.29
	LDA	73.30	66.14	73.29	68.41	8.34	13.52
	AE	78.52	75.78	78.52	74.47	6.97	7.647
	IG	71.16	66.14	71.16	66.25	9.64	36.18
	P.Corr	76.28	77.33	76.28	71.33	7.51	10.51
	Proposed	82.55	80.57	82.55	81.54	5.98	2.569
XGB	PCA	79.86	81.54	79.85	76.70	6.42	0.216
	LDA	75.71	75.34	75.70	72.38	7.38	0.212
	AE	80.17	82.79	80.16	76.40	6.50	0.243
	IG	74.63	73.83	74.63	69.87	8.70	0.226
	P.Corr	78.16	81.37	78.16	74.87	7.14	0.206
	Proposed	90.38	92.33	90.38	91.34	5.21	0.005
LightGBM	PCA	77.37	77.85	77.37	74.53	7.24	0.921
	LDA	74.68	72.79	74.68	71.27	7.65	0.583
	AE	78.20	77.86	78.20	75.31	7.97	0.997
	IG	72.66	70.24	72.65	68.17	9.21	0.867
	P.Corr	74.91	77.44	74.91	74.22	7.66	0.881
	Proposed	85.43	87.85	85.43	86.62	7.00	0.107
Catboost	PCA	80.09	82.50	80.09	77.06	6.33	0.201
	LDA	76.58	75.89	76.57	73.04	7.19	0.237
	AE	78.60	81.43	78.60	74.72	7.015	0.124
	IG	74.76	76.25	74.75	69.99	8.65	0.169
	P.Corr	79.20	81.89	79.20	76.00	6.78	0.109
	Proposed	90.56	91.47	90.56	91.01	4.89	0.022

Table 17 (continued)

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
Majority Voting	PCA	77.78	81.33	77.78	74.41	7.30	8.484
	LDA	74.76	73.25	74.76	70.96	7.84	6.868
	AE	78.48	79.90	78.47	74.64	7.16	8.483
	IG	73.02	75.07	73.02	68.27	9.29	20.30
	P.Corr	77.09	79.58	77.09	72.56	7.56	19.79
	Proposed	81.07	81.86	81.07	79.41	5.14	5.277
Mean Voting	PCA	79.31	78.02	79.31	76.05	6.69	6.086
	LDA	75.13	73.58	75.13	71.20	7.59	5.579
	AE	79.85	79.49	79.84	76.08	6.63	6.651
	IG	74.38	76.14	74.38	69.62	8.78	36.44
	P.Corr	79.18	79.81	79.18	74.67	6.72	18.63
	Proposed	81.14	81.23	81.14	79.34	5.01	4.749
DNN	PCA	81.50	80.20	81.50	78.21	5.75	1.426
	LDA	73.35	66.35	73.35	68.43	8.38	2.724
	AE	80.06	77.79	80.06	75.81	6.45	1.350
	IG	69.34	63.18	69.34	64.57	9.86	1.368
	P.Corr	76.26	75.30	76.26	71.61	7.29	1.48
	Proposed	85.46	81.86	85.46	83.62	5.17	0.998
CNN	PCA	80.57	79.42	80.57	77.43	6.12	2.732
	LDA	75.29	65.01	75.28	69.39	7.84	1.379
	AE	80.39	79.36	80.38	75.42	6.34	2.682
	IG	69.52	63.32	69.52	64.65	10.13	2.727
	P.Corr	77.83	80.14	77.83	74.00	6.69	2.695
	Proposed	82.16	80.65	82.16	80.18	5.32	0.981

6.6.4 Comparative analysis of the proposed method with other state-of-the-art approaches

In this section, the proposed model is compared with different state-of-the-art techniques on the NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets, which are shown in Tables 21, 22, 23, and 24.

Table 21 compares the prediction time (in second) of various detection models in the proposed framework with the [23] on the CIC-IDS2018 dataset. Observing from the table, it is summarized that the proposed framework takes comparatively less prediction time than that of [23]. The reduced prediction time compared to [23] is attributed to the introduction of two phases in the feature selection module, enabling the selection of only essential, relevant, information-rich, uncorrelated with other features, and correlated with the target class. Consequently, this results in a significant reduction in the size of the data, leading to a substantial decrease in both model building and prediction time. As a result, the proposed model becomes lightweight.

Table 22 compares the logistic regression-based detection models of the proposed framework with that of [64] on two datasets. The table shows that the proposed model gives better results than [64] in terms of accuracy, precision, recall, and f1-score. Moreover, the values of recall and f1-score in the proposed framework are significantly better than that of [64] on both datasets, while the accuracy of the proposed model has not significantly deteriorated, particularly on the UNSW-NB15 dataset. The proposed model clearly outperforms the one mentioned in [64]. This is due to the fact that the cost matrix in [64] is determined using a random forest classifier to evaluate feature importance. The proposed model integrates feature selection from two phases, concentrating on the mutual information metric and correlation metric within the ACO objective function. Another key factor contributing to the superior performance of the proposed model is the optimization of the logistic regression-based detection model using the genetic algorithm, which significantly boosts performance.

Table 23 compares the proposed model with [21] on the NSL-KDD dataset. The table shows that the proposed model gives impressive results for the RF, LR, and xgboost-based detection models; however, performance is not significantly degraded for the KNN-based detection

Table 18 Comparative analysis of different traditional dimensionality reduction techniques with the proposed approach on UNSW-NB15

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
DT	PCA	59.54	68.50	59.54	55.02	4.82	0.008
	LDA	41.88	54.66	41.88	41.29	7.85	0.020
	AE	57.37	71.93	57.37	54.16	5.10	0.014
	IG	62.64	69.56	62.64	59.48	4.16	0.037
	P.Corr	38.68	41.70	38.68	32.87	9.05	0.023
	Proposed	91.59	93.28	91.59	92.42	1.02	0.002
RF	PCA	61.90	56.32	61.90	56.64	4.53	1.570
	LDA	44.52	56.18	44.52	44.56	7.55	1.841
	AE	70.52	81.20	70.51	72.00	3.16	1.340
	IG	80.30	80.11	80.29	78.85	2.86	1.041
	P.Corr	43.19	58.35	43.19	40.08	8.59	1.455
	Proposed	90.31	92.47	90.30	91.37	1.09	0.226
KNN	PCA	63.85	81.74	63.85	63.93	3.87	13.17
	LDA	41.73	55.41	41.72	40.66	7.91	4.448
	AE	69.49	80.87	69.49	71.38	3.26	13.16
	IG	61.01	80.25	61.01	59.69	4.17	14.31
	P.Corr	41.10	56.65	41.10	36.39	8.34	91.29
	Proposed	84.86	84.12	84.86	84.04	1.73	12.82
LR	PCA	70.67	79.12	70.66	68.91	3.27	0.015
	LDA	45.34	52.98	45.34	41.61	7.93	0.010
	AE	71.39	78.36	71.38	69.68	3.21	0.022
	IG	73.75	81.78	73.75	72.22	2.94	0.017
	P.Corr	32.83	22.05	32.82	25.58	10.77	0.026
	Proposed	91.71	92.70	91.70	92.19	1.63	0.005
SVM	PCA	63.58	56.97	63.58	56.44	4.08	257.78
	LDA	38.29	42.60	38.28	32.87	8.73	460.44
	AE	65.26	76.98	65.26	60.36	3.93	441.29
	IG	63.79	54.69	63.79	57.41	3.96	455.53
	P.Corr	38.57	29.42	38.56	31.64	9.22	599.40
	Proposed	83.33	81.69	83.33	80.37	1.95	118.74
XGB	PCA	61.79	59.48	61.78	58.80	4.06	1.471
	LDA	42.15	55.20	42.14	40.80	7.99	1.433
	AE	67.41	79.09	67.41	67.95	3.52	1.578
	IG	61.92	59.46	61.91	58.75	4.09	1.300
	P.Corr	39.51	35.14	39.50	34.51	8.78	1.547
	Proposed	92.63	93.72	92.63	93.17	0.95	0.029
LightGBM	PCA	66.60	70.74	66.59	64.86	4.63	19.29
	LDA	42.40	56.58	42.39	43.62	7.59	7.617
	AE	63.47	75.91	63.46	63.64	4.04	8.577
	IG	46.60	56.44	46.59	49.35	7.01	7.266
	P.Corr	42.07	55.19	42.07	38.32	8.71	7.678
	Proposed	82.52	85.72	82.52	84.08	1.42	1.19
Catboost	PCA	61.62	60.95	61.62	58.65	4.12	0.926
	LDA	43.30	55.25	43.30	42.68	7.80	0.566
	AE	70.37	81.53	70.36	71.04	3.21	0.567
	IG	63.65	63.39	63.65	60.32	3.90	0.648
	P.Corr	40.36	36.73	40.35	35.83	8.46	0.560
	Proposed	89.93	90.85	89.93	90.38	1.17	0.136

Table 18 (continued)

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
Majority Voting	PCA	64.45	83.44	64.45	63.48	3.82	16.66
	LDA	43.84	56.91	43.84	43.95	7.54	17.827
	AE	64.24	80.76	64.23	63.56	3.85	19.462
	IG	65.51	82.07	65.51	64.31	3.71	16.64
	P.Corr	39.03	56.78	39.03	33.87	9.02	95.60
	Proposed	90.63	92.27	90.63	91.44	1.02	13.81
Mean Voting	PCA	65.01	83.79	65.01	64.67	3.75	12.41
	LDA	43.21	55.42	43.21	42.09	7.85	13.765
	AE	64.92	80.78	64.92	64.68	3.77	16.015
	IG	66.22	82.04	66.21	65.08	3.63	13.61
	P.Corr	39.30	33.36	39.29	33.64	9.06	88.49
	Proposed	91.60	90.42	91.60	91.00	1.12	11.62
DNN	PCA	69.43	77.79	69.43	67.19	3.43	5.327
	LDA	37.76	31.31	37.76	32.25	8.80	10.35
	AE	66.52	76.58	66.51	63.19	3.76	3.089
	IG	76.40	84.20	76.40	75.42	2.61	10.40
	P.Corr	36.19	24.47	36.19	29.13	9.73	10.42
	Proposed	85.25	85.42	85.25	82.38	1.73	2.005
CNN	PCA	73.04	75.92	73.04	70.85	3.06	16.187
	LDA	51.33	71.85	51.32	54.88	5.51	10.48
	AE	61.81	76.64	61.81	54.99	4.28	13.691
	IG	45.74	51.57	45.74	45.15	7.75	16.027
	P.Corr	34.34	22.76	34.33	27.37	10.11	10.213
	Proposed	85.32	84.07	85.32	82.59	1.73	9.33

Table 19 Comparative analysis of different traditional dimensionality reduction techniques with the proposed approach on CSE-CIC-IDS2018

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
DT	PCA	94.68	94.67	94.68	94.67	1.17	0.308
	LDA	90.44	90.47	90.43	90.45	2.09	0.558
	AE	94.07	94.06	94.06	94.06	1.30	0.469
	IG	90.11	90.15	90.10	90.13	2.17	0.814
	P.Corr	94.36	94.35	94.36	94.35	0.99	0.338
	Proposed	96.39	96.37	96.38	96.37	0.80	0.267
RF	PCA	94.24	94.35	94.23	94.27	0.90	1.640
	LDA	91.24	91.36	91.23	91.28	1.91	2.818
	AE	95.13	95.20	95.12	95.15	1.05	1.646
	IG	90.63	91.32	90.63	90.84	1.98	1.184
	P.Corr	94.96	94.06	94.96	94.98	0.94	1.001
	Proposed	96.37	96.41	96.36	96.38	0.78	0.339
KNN	PCA	92.45	92.92	92.44	92.67	2.15	39.96
	LDA	90.49	90.63	90.49	90.55	2.07	12.534
	AE	93.79	93.09	93.79	93.43	2.09	40.86
	IG	89.43	89.83	89.42	89.58	2.28	71.47
	P.Corr	92.49	92.92	92.48	92.69	1.92	47.41
	Proposed	94.02	94.48	94.01	94.13	1.25	8.49
LR	PCA	88.45	89.00	88.45	88.65	2.48	0.407

Table 19 (continued)

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
	LDA	86.43	85.11	86.42	85.56	3.08	0.307
	AE	88.45	88.84	88.45	88.60	2.49	0.513
	IG	89.00	91.97	88.99	89.40	2.21	0.413
	P.Corr	88.29	88.70	88.28	88.44	2.53	0.423
	Proposed	92.47	94.22	92.47	93.33	1.49	0.282
SVM	PCA	90.35	89.07	90.35	89.70	1.95	188.36
	LDA	87.20	86.86	87.20	87.00	2.84	215.61
	AE	92.61	91.03	92.61	91.81	2.57	120.92
	IG	89.34	88.24	89.33	88.78	2.09	113.25
	P.Corr	91.12	84.88	91.12	87.88	3.80	119.89
	Proposed	93.79	94.50	93.79	93.94	1.28	103.63
XGB	PCA	91.16	89.41	91.16	90.27	0.79	0.591
	LDA	90.78	91.08	90.77	90.89	1.99	0.738
	AE	94.83	95.09	94.82	94.90	1.09	0.692
	IG	90.66	92.51	90.65	90.98	1.90	0.585
	P.Corr	93.81	88.06	93.81	90.84	1.42	0.549
	Proposed	97.87	98.11	97.87	97.90	0.41	0.434
LightGBM	PCA	85.91	86.24	85.90	86.06	0.84	2.057
	LDA	90.69	91.11	90.68	90.83	1.99	1.724
	AE	94.69	95.04	94.68	94.77	1.11	1.680
	IG	90.79	92.98	90.79	91.12	1.85	1.588
	P.Corr	87.86	88.12	87.86	87.90	1.41	1.400
	Proposed	97.96	98.19	97.95	97.99	0.39	1.210
Catboost	PCA	86.31	86.59	86.30	86.36	2.76	1.474
	LDA	90.69	90.95	90.69	90.79	2.01	2.336
	AE	94.77	95.00	94.76	94.83	1.11	2.424
	IG	90.81	92.91	90.81	91.14	1.85	1.211
	P.Corr	91.85	89.11	91.84	90.45	1.41	2.330
	Proposed	97.90	98.15	97.90	97.93	0.40	0.710
Majority Voting	PCA	86.02	86.09	86.01	86.04	2.85	42.72
	LDA	90.59	90.35	90.59	90.42	2.11	3.314
	AE	94.96	95.00	94.96	94.98	1.09	37.28
	IG	90.57	91.52	90.57	90.82	1.98	39.55
	P.Corr	87.08	87.18	87.08	87.10	2.61	48.52
	Proposed	96.58	96.58	96.57	96.58	0.74	9.94
Mean Voting	PCA	86.32	86.61	86.31	86.37	1.75	37.28
	LDA	91.19	91.31	91.18	91.24	1.92	1.318
	AE	85.31	85.57	85.30	85.37	2.98	34.51
	IG	90.71	91.79	90.70	90.97	1.94	38.13
	P.Corr	87.61	87.84	87.60	87.64	1.47	50.62
	Proposed	97.55	97.75	97.54	97.58	0.49	8.31
DNN	PCA	86.14	86.86	86.13	86.23	1.75	2.981
	LDA	88.52	92.34	88.51	88.91	2.27	2.261
	AE	83.42	85.29	83.41	84.33	1.29	2.803
	IG	89.31	93.48	89.31	89.71	2.09	3.947
	P.Corr	85.78	86.44	85.78	86.10	1.84	5.283
	Proposed	95.58	96.28	95.58	95.69	0.88	1.889

Table 19 (continued)

Model		Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Prediction Time (s)
CNN	PCA	83.69	84.67	83.69	84.17	1.28	5.309
	LDA	88.13	90.14	88.13	88.52	2.44	2.349
	AE	91.73	91.69	91.72	91.71	1.82	2.389
	IG	89.27	93.55	89.27	89.67	2.09	5.289
	P.Corr	85.81	86.37	85.81	86.08	2.84	5.311
	Proposed	95.21	96.00	95.21	95.33	0.95	1.979

Table 20 Statistical validation of the proposed model with baseline method using p-value on two-tailed t-test at 0.05 significance level

Dataset	p-value				
	PCA	LDA	AE	IG	P.Corr
NSL-KDD (Xgboost)	5.47e-07	1.33e-10	5.24e-09	2.95e-12	3.37e-09
UNSW-NB15 (Xgboost)	5.37e-09	2.34e-08	0.000103	1.14e-06	1.78e-08
CSE-CIC-IDS2018 (Catboost)	5.60e-05	0.021061	0.030745	0.021049	0.019673

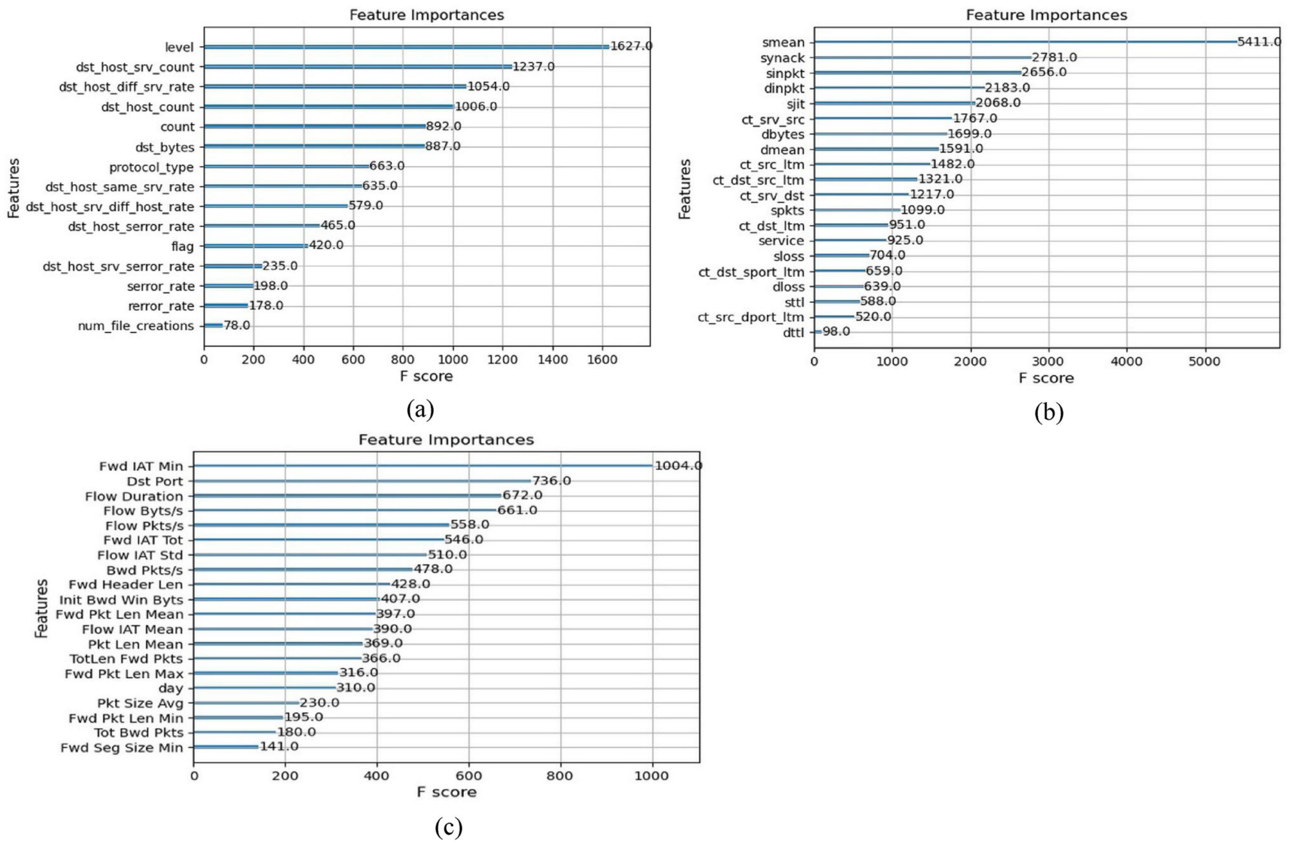


Fig. 13 Feature importance for xgboost on **a** NSL-KDD, **b** UNSW-NB15, **c** CSE-CIC-IDS2018

model. In existing research [21], the feature-selection method utilizing *CFS-DE* is examined, however, it does not address the optimization of the classifiers. The improved performance of the proposed model can be attributed to the implementation of a bi-phase feature optimization technique for data feature selection, in

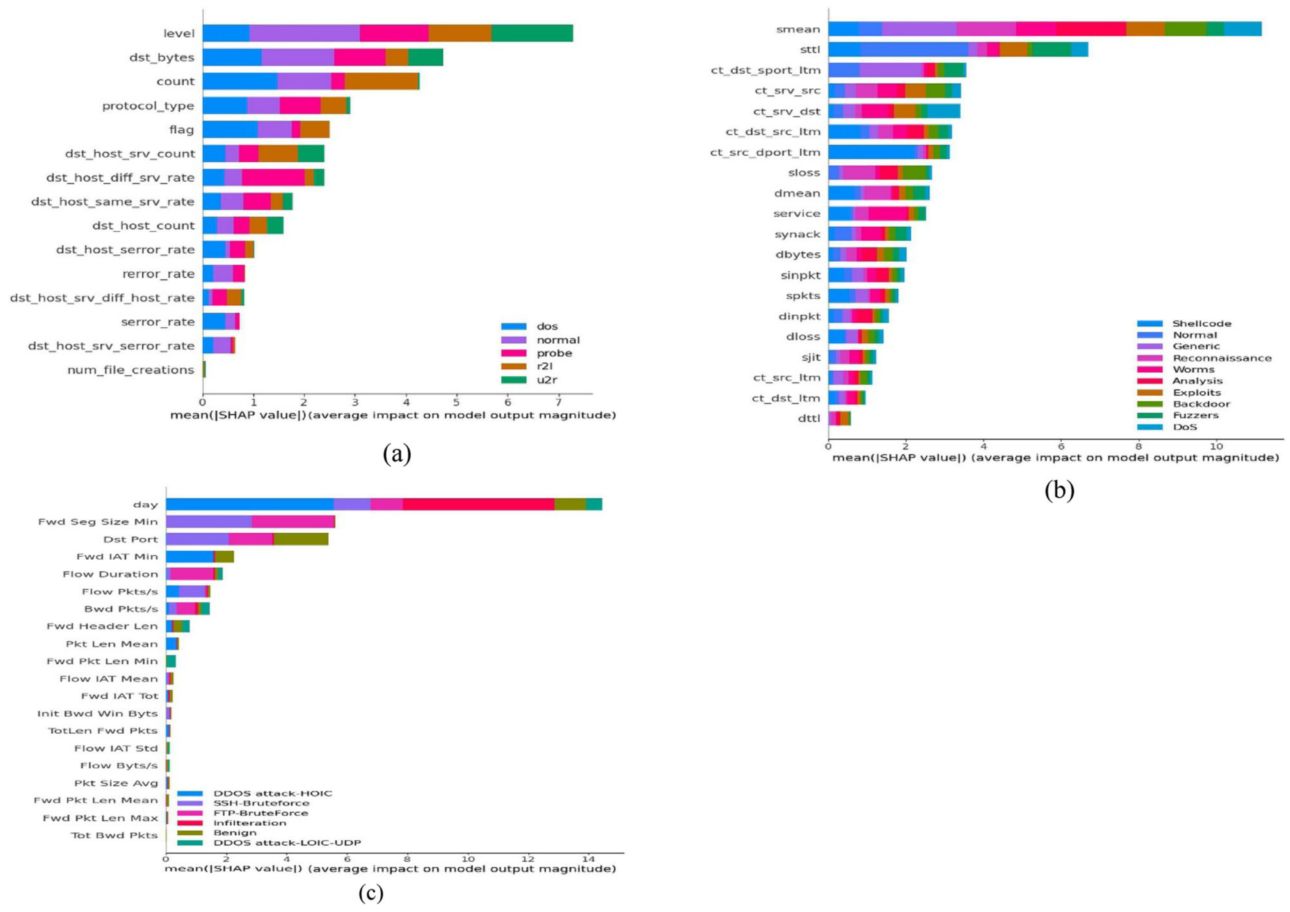


Fig. 14 SHAP analysis for xgboost on **a** NSL-KDD, **b** UNSW-NB15, **c** CSE-CIC-IDS2018

Table 21 Comparison of Prediction time (in seconds) of the Proposed Model and Chowdhury et al. [23] on the CIC-IDS2018 Dataset

Method	DT	KNN	DNN	RF	Catboost	Majority Voting	Mean Voting
Chowdhury et al. [23]	1.247	1030.725	109.213	23.398	2.140	640.041	524.656
Proposed	0.267	8.49	1.889	0.339	0.710	9.94	8.31

Table 22 Comparison of classification report of the Proposed Model with state-of-the-art Pramilarani & Kumari [64]

Dataset	Pramilarani & Kumari [64]				Proposed			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
NSL-KDD	84.25	89.78	83.43	86.27	89.35	91.46	89.35	90.39
UNSW-NB15	92.45	91.78	88.43	90.27	91.71	92.70	91.70	92.19

conjunction with the optimization of each detection model to yield precise results through fine-tuned hyperparameters.

Table 24 shows the results of different state-of-the-art works [9, 11, 35], and [65] and compares them with the proposed framework on NSL-KDD and UNSW-NB15 datasets. From Table 24, it is observed that the proposed model offers significant results in terms of accuracy, FAR, training, and testing time. The reason behind this is that the proposed model selects features based on the imbalanced nature of the data, the correlation between the feature-

Table 23 Comparison of the Proposed Model with state-of-the-art Zhao et al. [21] on NSL-KDD dataset

Detection Model	Zhao et al. [21]					Proposed				
	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Training Time (s)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Training Time (s)
RF	86.51	88.61	86.51	87.55	9.06	89.06	90.76	89.06	89.90	8.44
LR	81.53	72.67	81.53	76.84	11.95	89.35	91.46	89.35	90.39	9.37
Xgboost	86.53	87.37	86.53	86.95	5.83	90.38	92.33	90.38	91.34	1.097
KNN	85.70	80.82	85.70	83.19	130.42	82.31	80.64	82.31	81.43	29.309

Table 24 Comparison of Classification Report of the Proposed Model with different state-of-the-art works

Dataset	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FAR (%)	Training Time (s)	Testing Time (s)
NSL-KDD	Thakkar et al. [65]	82.22	92.01	75.30	82.82	8.62	NA	NA
	Chohra et al. [35]	90.71	89.35	95.00	92.09	NA	373	NA
	Proposed	90.56	91.47	90.56	91.01	4.89	97.127	0.022
UNSW-NB15	Thakkar et al. [65]	76.28	71.58	94.39	81.41	45.92	NA	NA
	Chohra et al. [35]	89.52	90.00	96.00	92.90	NA	1718	NA
	Khammassi & Krichen 2017 [9]	81.42	NA	NA	NA	6.39	NA	NA
	Nazir & Khan [11]	83.12	NA	NA	NA	3.7	NA	NA
	Proposed	92.63	93.72	92.63	93.17	0.95	401.442	0.029

feature pairs and class-feature pairs, and, last but not least, information-rich features. Moreover, all the detection models offer their performances on the best hyperparameter values.

7 Conclusion and future works

For securing the network from different kinds of cyber-attacks, the Intrusion Detection System is proven to be an effective and efficient technique. The main aim of this paper is to design a lightweight Intrusion detection system that uses resources efficiently. To achieve the aforementioned goal, features in the network traffic in the intrusion detection system dataset are reduced with the stacking of the particle swarm optimization and ant colony optimization algorithms. Here, the main objective is to select the most effective features from the dataset to make the IDS lightweight. The particle swarm optimization algorithm determines the features based on the imbalanced nature of the dataset by utilizing the geometric mean in the fitness function, and the ant colony optimization algorithm is utilized here to address issues related to correlated and uninformative features in the obtained feature subset. These issues can be mitigated by utilizing the correlation metric and information gain metric in the fitness function of the ant colony optimization-based feature selection algorithm. The selected features through proposed bi-phase technique is analysed using the feature importance and SHAP (discussed in Sect. 6.6.3). Several base and ensemble-based detection models (such as DT, SVM, KNN, RF, Xgboost, LightGBM, Catboost, LR, Majority Voting, Mean Voting, DNN, and 1D-CNN) are introduced in this paper to evaluate the effectiveness of the proposed model. A nature-influenced genetic algorithm is applied (individually for each model) by utilizing a weighted f1-score in the fitness function to optimize the hyperparameters of these detection models. The objective function of each metaheuristic algorithms are analysed using convergence graphs, box plots, swarm plots, and in terms of best, worst, mean, median, standard deviation, and variance (demonstrated in Sect. 6.5). Several extensive experiments are performed on three traditional datasets such as NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018, and it is compared with several traditional dimensionality reduction techniques such as PCA, LDA, Pearson Correlation, Information Gain, and Auto-encoder. Statistical validation of the proposed model is also performed to examine the effectivenss of the proposed approach with the baseline methods (shown in Table 20). The accuracy and FAR of the proposed model is as follows: (90.38% and 5.21%), (92.63% and 0.95%), and (97.87% and 0.41%) on NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets respectively. It is observed that proposed method outperforms other traditional dimensionality reduction techniques and the existing state-of-the-art works [9, 11, 21, 23, 35, 64, 65].

7.1 Limitations and future directions of the proposed research:

The limitations of the proposed research are highlighted as follows: (i) The current proposed model is limited to detecting only known attacks in the network traffic but can not detect unknown or new cyber-attacks. (ii) Several extensive experiments are performed only on traditional datasets, while implementation of the proposed model is unexplored on the real-time test bed. In the future, other enhanced feature selection techniques will be utilized to enhance the model's performance. Future research may consider the imbalanced nature of the IDS dataset by generating more realistic samples by adding generative AI, and real-time datasets can also be generated to test the detection performance. Furthermore, a more enhanced hyperparameter tuning module will be implemented to enhance the model's performance.

Author contributions Methodology: [Arpita Srivastava]; Implementation: [Arpita Srivastava]; Validation: [Arpita Srivastava]; Conceptualization: [Arpita Srivastava]; Data Curation: [Arpita Srivastava]; Writing- Original Draft: [Arpita Srivastava]; Conceptualization: [Ditipriya Sinha]; Methodology: [Ditipriya Sinha]; Validation: [Ditipriya Sinha]; Supervision: [Ditipriya Sinha]; Writing- Original Draft: [Ditipriya Sinha]; Writing- Review & Editing: [Ditipriya Sinha].

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability Data will be made available on request.

Declarations

Competing interests The authors declare that they have no conflict of interest.

References

1. Injadat, M., Moubayed, A., Nassif, A.B., Shami, A.: Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans. Netw. Serv. Manage. **18**(2), 1803–1816 (2020)
2. Salem, M.B., Hershkop, S., Stolfo, S.J.: A survey of insider attack detection research. In: Insider Attack and Cyber Security: Beyond the Hacker, pp. 69–90. Springer, Cham (2008)
3. Papamartzivanos, D., Mármol, F.G., Kambourakis, G.: Dendron: genetic trees driven rule induction for network intrusion detection systems. Futur. Gener. Comput. Syst. **79**, 558–574 (2018)
4. Aksu, D., Aydin, M.A.: MGA-IDS: optimal feature subset selection for anomaly detection framework on in-vehicle networks-CAN bus based on genetic algorithm and intrusion detection approach. Comput. Secur. **118**, 102717 (2022)
5. Azimjonov, J., Kim, T.: Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets. Expert Syst. Appl. **237**, 121493 (2024)
6. Azimjonov, J., Kim, T.: Designing accurate lightweight intrusion detection systems for IoT networks using fine-tuned linear SVM and feature selectors. Comput. Secur. **137**, 103598 (2024)
7. Wang, Z., Li, Z., He, D., Chan, S.: A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning. Expert Syst. Appl. **206**, 117671 (2022)
8. Sohn, I.: Deep belief network based intrusion detection techniques: a survey. Expert Syst. Appl. **167**, 114170 (2021)
9. Khammassi, C., Krichen, S.: A GA-LR wrapper approach for feature selection in network intrusion detection. Comput. Secur. **70**, 255–277 (2017)
10. Vijayanand, R., Devaraj, D., Kannapiran, B.: Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. Comput. Secur. **77**, 304–314 (2018)
11. Nazir, A., Khan, R.A.: A novel combinatorial optimization based feature selection method for network intrusion detection. Comput. Secur. **102**, 102164 (2021)
12. Kumar, G.S.C., Kumar, R.K., Kumar, K.P.V., Sai, N.R., Brahmaiah, M.: Deep residual convolutional neural Network: an efficient technique for intrusion detection system. Expert Syst. Appl. **238**, 121912 (2024)
13. Khammassi, C., Krichen, S.: A NSGA2-LR wrapper approach for feature selection in network intrusion detection. Comput. Netw. **172**, 107183 (2020)
14. Mohammadi, S., Mirvaziri, H., Ghazizadeh-Ahsaee, M., Karimipour, H.: Cyber intrusion detection by combined feature selection algorithm. J. Inform. Secur. Appl. **44**, 80–88 (2019)
15. Halim, Z., Yousaf, M.N., Waqas, M., Sulaiman, M., Abbas, G., Hussain, M., Hanif, M.: An effective genetic algorithm-based feature selection method for intrusion detection systems. Comput. Secur. **110**, 102448 (2021)
16. Li, Y., Qin, T., Huang, Y., Lan, J., Liang, Z., Geng, T.: HDFEF: a hierarchical and dynamic feature extraction framework for intrusion detection systems. Comput. Secur. **121**, 102842 (2022)
17. Rao, K.N., Rao, K.V., Prasad Reddy, P.V.G.D.: A hybrid intrusion detection system based on sparse autoencoder and deep neural network. Comput. Commun. **180**, 77–88 (2021)
18. Wazirali, R.: An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation. Arab. J. Sci. Eng. **45**(12), 10859–10873 (2020)
19. Gu, J., Lu, S.: An effective intrusion detection approach using SVM with naïve Bayes feature embedding. Comput. Secur. **103**, 102158 (2021)
20. Mukherjee, S., Sharma, N.: Intrusion detection using naive Bayes classifier with feature reduction. Procedia Technol. **4**, 119–128 (2012)
21. Zhao, R., Mu, Y., Zou, L., Wen, X.: A hybrid intrusion detection system based on feature selection and weighted stacking classifier. IEEE Access **10**, 71414–71426 (2022)

22. Nguyen, M.T., Kim, K.: Genetic convolutional neural network for intrusion detection systems. Futur. Gener. Comput. Syst. **113**, 418–427 (2020)
23. Chowdhury, R., Sen, S., Goswami, A., Purkait, S., Saha, B.: An implementation of bi-phase network intrusion detection system by using real-time traffic analysis. Expert Syst. Appl. **224**, 119831 (2023)
24. Kunang, Y.N., Nurmaini, S., Stiawan, D., Suprapto, B.Y.: Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. J. Inform. Secur. Appl. **58**, 102804 (2021)
25. Batchu, R.K., Seetha, H.: A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning. Comput. Netw. **200**, 108498 (2021)
26. Chebrolu, S., Abraham, A., Thomas, J.P.: Feature deduction and ensemble design of intrusion detection systems. Comput. Secur. **24**(4), 295–307 (2005)
27. Li, Y., Wang, J.L., Tian, Z.H., Lu, T.B., Young, C.: Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. Comput. Secur. **28**(6), 466–475 (2009)
28. Revathi, S., Malathi, A.: A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. Int. J. Eng. Res. Technol. (IJERT) **2**(12), 1848–1853 (2013)
29. Li, X., Chen, W., Zhang, Q., Wu, L.: Building auto-encoder intrusion detection system based on random forest feature selection. Comput. Secur. **95**, 101851 (2020)
30. Kunhare, N., Tiwari, R., Dhar, J.: Particle swarm optimization and feature selection for intrusion detection system. Sādhanā **45**, 1–14 (2020)
31. Kunhare, N., Tiwari, R., & Dhar, J.: Network packet analysis in real time traffic and study of snort IDS during the variants of DoS attacks. In *Hybrid Intelligent Systems: 19th International Conference on Hybrid Intelligent Systems (HIS 2019) held in Bhopal, India, December 10–12, 2019 19* (pp. 362–375). Springer International Publishing. (2021)
32. Gupta, R.K., Bharti, S., Kunhare, N., Sahu, Y., Pathik, N.: Brain tumor detection and classification using cycle generative adversarial networks. Interdisc. Sci.: Comput. Life Sci. **14**(2), 485–502 (2022)
33. Dhanya, L., Chitra, R.: A novel autoencoder based feature independent GA optimised XGBoost classifier for IoMT malware detection. Expert Syst. Appl. **237**, 121618 (2024)
34. Ogundokun, R.O., Awotunde, J.B., Sadiku, P., Adeniyi, E.A., Abiodun, M., Dauda, O.I.: An enhanced intrusion detection system using particle swarm optimization feature extraction technique. Procedia Comput. Sci. **193**, 504–512 (2021)
35. Chohra, A., Shirani, P., Karbab, E.B., Debbabi, M.: Chameleon: Optimized feature selection using particle swarm optimization and ensemble methods for network anomaly detection. Comput. Secur. **117**, 102684 (2022)
36. Alazab, M., Khurma, R.A., Awajan, A., Camacho, D.: A new intrusion detection system based on moth-flame optimizer algorithm. Expert Syst. Appl. **210**, 118439 (2022)
37. Dahou, A., Abd Elaziz, M., Chelloug, S.A., Awadallah, M.A., Al-Betar, M.A., Al-Qaness, M.A., Forestiero, A.: Intrusion detection system for IoT based on deep learning and modified reptile search algorithm. Comput. Intell. Neurosci. **2022**(1), 6473507 (2022)
38. Kunhare, N., Tiwari, R., Dhar, J.: Intrusion detection system using hybrid classifiers with meta-heuristic algorithms for the optimization and feature selection by genetic algorithm. Comput. Electr. Eng. **103**, 108383 (2022)
39. Jovanovic, Luka, et al.: The xgboost tuning by improved firefly algorithm for network intrusion detection. 2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). IEEE, 2022.
40. AlHosni, N., Jovanovic, L., Antonijevic, M., Bukumira, M., Zivkovic, M., Strumberger, I., Bacanin, N.: The xgboost model for network intrusion detection boosted by enhanced sine cosine algorithm. In International Conference on Image Processing and Capsule Networks (pp. 213–228). Cham: Springer International Publishing. (2022)
41. Kalita, D.J., Singh, V.P., Kumar, V.: A novel adaptive optimization framework for SVM hyper-parameters tuning in non-stationary environment: a case study on intrusion detection system. Exp. Syst. Appl. **213**, 119189 (2023)
42. Savanović, N., Toskovic, A., Petrovic, A., Zivkovic, M., Damaševičius, R., Jovanovic, L., Nikolic, B.: Intrusion detection in healthcare 4.0 internet of things systems via metaheuristics optimized machine learning. Sustainability **15**(16), 12563 (2023)
43. Yang, X. S.: Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms (pp. 169–178). Berlin, Heidelberg: Springer Berlin Heidelberg. (2009)
44. Mirjalili, S., Mirjalili, S.: Genetic algorithm. Evolut. Algorithm. Neural Netw.: Theory Appl. **780**, 43–55 (2019)
45. Kennedy, J., & Eberhart, R.: Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942–1948). ieee. (1995)
46. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. **8**(1), 687–697 (2008)
47. Khishe, M., Mosavi, M.R.: Chimp optimization algorithm. Expert Syst. Appl. **149**, 113338 (2020)
48. Gurrola-Ramos, J., Hernàndez-Aguirre, A., & Dalmau-Cedeño, O.: COLSHADE for real-world single-objective constrained optimization problems. In 2020 IEEE congress on evolutionary computation (CEC) (pp. 1–8). IEEE. (2020)
49. Zhao, J., Zhang, B., Guo, X., Qi, L., Li, Z.: Self-adapting spherical search algorithm with differential evolution for global optimization. Mathematics **10**(23), 4519 (2022)
50. Saheed, Y.K., Misra, S.: A voting gray wolf optimizer-based ensemble learning models for intrusion detection in the internet of things. Int. J. Inform. Secur. (2024). https://doi.org/10.1007/s10207-023-00803-x
51. Tharwat, A.: Classification assessment methods. Appl. Comput. Inform. **17**(1), 168–192 (2020)
52. Moustafa, N., & Slay, J. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1–6). IEEE.
53. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
54. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A.: A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1–6). Ieee. (2009)
55. https://www.unb.ca/cic/datasets/ids-2018.html
56. Chakraborty, A., Kar, A.K.: Swarm intelligence: a review of algorithms. Nat. Inspired Comput. Optim.: Theory Appl. (2017). https://doi.org/10.1007/978-3-319-50920-4_19
57. Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z.W., Gandomi, A.H.: Reptile search algorithm (RSA): a nature-inspired meta-heuristic optimizer. Expert Syst. Appl. **191**, 116158 (2022)
58. Połap, D., Woźniak, M.: Red fox optimization algorithm. Expert Syst. Appl. **166**, 114107 (2021)
59. Abualigah, L., Shehab, M., Alshinwan, M., Alabool, H.: Salp swarm algorithm: a comprehensive survey. Neural Comput. Appl. **32**(15), 11195–11215 (2020)

60. Arora, S., Singh, S.: Butterfly optimization algorithm: a novel approach for global optimization. Soft. Comput. **23**, 715–734 (2019)
61. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
62. Lipowski, A., Lipowska, D.: Roulette-wheel selection via stochastic acceptance. Physica A **391**(6), 2193–2196 (2012)
63. Hasançebi, O., Erbatur, F.: Evaluation of crossover techniques in genetic algorithm based optimum structural design. Comput. Struct. **78**(1–3), 435–448 (2000)
64. Pramilarani, K., Kumari, P.V.: Cost based random forest classifier for intrusion detection system in internet of things. Appl. Soft Comput. **151**, 111125 (2024)
65. Thakkar, A., Kikani, N., Geddam, R.: Fusion of linear and non-linear dimensionality reduction techniques for feature reduction in LSTM-based intrusion detection system. Appl. Soft Comput. (2024). https://doi.org/10.1016/j.asoc.2024.111378
66. WUSTL, E. (2020). Dataset for internet of medical things (IoMT) Cybersecurity Research.
67. https://www.kaggle.com/datasets/saurabhshahane/classification-of-malwares

Arpita Srivastava is pursuing Ph.D. degree in the Department of Computer Science and Engineering from the National Institute of Technology Patna, Bihar, India. She has received her Master of Technology degree in the Department of Computer Science and Engineering from Kamla Nehru Institute of Technology, Sultanpur, Uttar Pradesh, India. Her research interest includes Intrusion Detection System, Machine Learning, and Deep Learning.

Ditipriya Sinha has received Ph.D. degree in the Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology (IIEST), Shibpur and Master of Technology from West Bengal University of Technology in the department of Software Engineering. She is the Silver Medallist during MTech. She is presently serving as an Assistant Professor in the department of Computer Science and Engineering, National Institute of Technology Patna. She was an Assistant Professor in the department of Computer Science and Engineering, Birla Institute of Technology, Mesra. Her area of research is Cyber Security, Blockchain, Machine and Deep Learning and Wireless Sensor Network.