Journal of Artificial Intelligence and Metaheuristic

♦ ISSN: 2833-5597 Vol: 10, No. 01, 2025 ♦ PP. 28–36

Article DOI: https://doi.org/10.54216/JAIM.100104



FULL LENGTH ARTICLE

Network Requests Classification Using Advanced Metaheuristic Optimization for Enhanced Network Security Systems

Marwa M. Eid[1,2,*]

[1] Faculty of Artificial Intelligence, Delta University for Science and Technology, Mansoura 11152, Egypt

[2] Jadara Research Center, Jadara University, Irbid 21110, Jordan

Email: mmm@ieee.org

Received: March 28, 2025 Revised: June 18, 2025 Accepted: August 19, 2025 ⋆ Corresponding author

ABSTRACT

The importance of network security has greatly increased in the continuously changing modern digital environment. Network security is a multi-layered defense mechanism that protects networks, data, and systems from unauthorized access and other malicious activities. As cyber threats become more advanced, traditional protective measures may no longer be adequate, and intrusion-detection systems must rapidly evolve. This research proposes BBERSC (Balance Between Al-Biruni Earth Radius Optimization and Sine Cosine Algorithm), which combines the local-search capability of Al-Biruni Earth Radius Optimization with the probabilistic improvement of the Sine Cosine Algorithm. The approach tunes Random Forest hyperparameters, including the number of trees, maximum depth, and minimum samples, to improve pattern recognition in network traffic. ANOVA and Wilcoxon signed-rank tests validate the effectiveness of the models and algorithms and assess differences between them. The Random Forest model within the BBERSC framework achieves an accuracy of 0.9901719, exceeding the other evaluated machine-learning algorithms.

Keywords: Network requests ▪ Al-Biruni Earth Radius Optimization ▪ Sine Cosine Algorithm ▪ Network security ▪ Intrusion detection

1. INTRODUCTION

The environment of cyber systems has become a contemporary dynamic process where it is crucial to harden networks against an array of advanced threat actors that differ from one day to another [1]. More than simply introducing sturdy hardware and software solutions, this problem requires a skilled arrangement of complex rules, configurations, and adaptive strategies to maneuver through an emerging threat terrain. The rapid progression of digital developments has further elevated the importance of preventing network attacks and protecting critical information from an expanding range of complex breaches. To address this changing problem, the coexistence of artificial intelligence and optimization offers tools to strengthen the resilience and agility of digital infrastructure.

Metaheuristic algorithms are used to solve a wide range of problems because optimization and the search for an optimal solution from a large set of solutions are necessary in many models [2]. Search algorithms for optimization include exact, heuristic, and metaheuristic approaches [3]. Exact algorithms can be impractical for large-scale problems because they may require substantial time [4]. Heuristic algorithms can become trapped in local optima and may fail to obtain high-quality solutions for complex problems [5]. Metaheuristics can discover optimal or near-optimal solutions within an acceptable period [6, 7]. They generate solutions by considering two components: diversification, which explores the full problem

© 2025 The Author(s). Published by ASPG. This article is licensed under CC BY 4.0.

space by producing varied solutions [8], and intensification, which searches promising regions. The nature of a problem therefore influences which algorithm can produce more accurate results. Problems with discrete search spaces require algorithms capable of discrete search, which can use genetic and neighborhood operators [9].

Optimization is relevant to mechanics, economics, travel planning, and Internet routing, where limited resources must be maximized under constraints [10]. By applying metaheuristic optimization alongside advanced classification models, this research seeks to strengthen network security in a dynamic environment. The optimization problem is characterized by complexity associated with real-world nonlinearity and multimodality [11, 12].

The proposed BBERSC optimization approach combines Al-Biruni Earth Radius Optimization (BER) and the Sine Cosine Algorithm (SCA) to support classification performance. Drawing on collective intelligence and trigonometric functions, the method is intended to balance the two optimization strategies for the Network Requests dataset (Figure 1).

The study also compares optimization techniques for improving the Random Forest Classifier. Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO) are swarm-inspired methods, while Whale Optimization Algorithm (WOA) provides another population-based approach. These algorithms are considered as alternative ways of exploring potential solutions and improving classifier performance.

The evaluated classifiers include Logistic Regression, Decision Tree, Random Forest [13, 14], Support Vector Machine (SVM), k-Nearest Neighbors (KNN) [15, 16], Naive Bayes, and SGD Classifier [17, 18]. These models are applied to network requests for web and mobile applications [19], and are assessed using accuracy, sensitivity, specificity, P-value, Positive Predictive Value (PPV), Negative Predictive Value (NPV), and F-score [20].

The main contributions of this study include:

- **Development of BBERSC:** This research presents a new optimization method combining SCA and BER. The algorithm integrates the advantages of both methods to maximize the efficiency of classification models designed for network-security tasks.
- **Optimization for network security:** The algorithm applies two different optimization strategies to practical network-security implementations, seeking to improve the classification of safe and unsafe requests in network traffic.
- **Comprehensive machine-learning framework:** The paper defines a complete machine-learning infrastructure that combines optimization technologies with multiple baseline models.
- **Random Forest hyperparameter tuning:** The study investigates the number of trees, maximum depth, and minimum-sample parameters to attain effective Random Forest performance for network-request classification.

Together, these contributions form an integrated strategy for network security using optimization algorithms, machine-learning models, and comparative experimentation, with potential value for both research and cybersecurity practice.

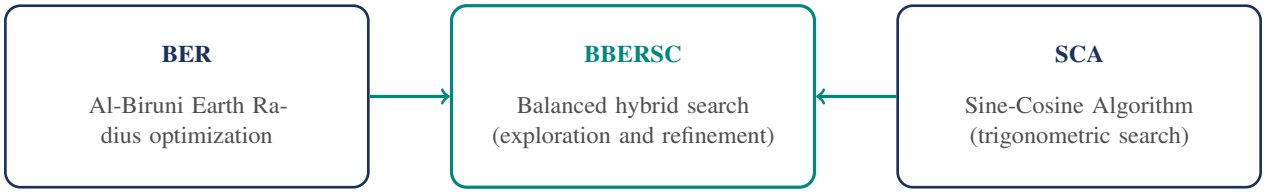
2. RELATED WORKS

Technological paradigm shifts underway in the technology landscape are poised to redefine programming methodologies and alter interactions with the world [21]. Cloud computing, mobile computing, and the Internet of Things (IoT) have had a profound impact, with IoT seeking to connect Internet-enabled devices to foster intelligent environments. Across these domains and their intersections, numerous computing paradigms have emerged, including Mobile Cloud Computing (MCC), cloudlet computing, mobile clouds, mobile IoT computing, IoT cloud computing, fog computing, Mobile Edge Computing (MEC), edge computing, the Web of Things (WoT), Semantic WoT (SWoT), Wisdom WoT (W2T), opportunistic sensing, participatory sensing, mobile crowdsensing, and mobile crowdsourcing. A lack of standardized definitions can create confusion: one term may denote multiple paradigms, or several terms may refer to a single paradigm. Research has therefore sought to distinguish these paradigms, clarify their positions within cloud computing, mobile computing, and IoT, trace their historical development, and describe research directions.

In IoT, analytics is an important means of deriving knowledge and supporting smart-home applications [22]. Connected appliances and devices generate data that can reveal consumer behavior and daily routines. IoT analytics can help tailor applications to individual homeowners while also meeting the needs of industries that use consumer profiles. One framework integrates fog nodes and cloud systems to provide data-driven services and address the complexity and resource demands of online and offline processing, storage, and classification. Its specifications and design components are described, and a case study uses data from a smart home in Vancouver, Canada. The reported experimental results support the platform's efficacy and practical viability.

Scientific workflows, characterized by many interdependent tasks, are an important class of complex scientific applications [23]. Serverless infrastructures such as Google Cloud Functions and AWS Lambda implement the Function-as-a-Service model. Although initially designed for background tasks in Web and IoT applications or event-driven stream processing, these infrastructures are being examined for computationally and data-intensive scientific workflows. Prototype workflow-executor functions have been developed with AWS Lambda, Google Cloud Functions, and the HyperFlow workflow engine. These functions execute workflow tasks on AWS and Google infrastructures, stage data to and from S3 or Google Cloud Storage, and run custom application binaries. The Montage astronomy workflow, a common evaluation benchmark, has been deployed and run, with initial performance results reported. The approach is considered user-friendly and promising, although preparing portable binaries for remote execution incurs costs. Future work includes serverless execution studies, cost analysis, and consideration of resource management in scientific applications.

Another study addresses service delivery for IoT applications with stringent real-time and predictable-latency requirements [24]. Since IoT devices have limited capabilities, they must be integrated with cloud and fog computing. Fog computing brings resources closer to the network edge and can reduce service latency. The study formulates IoT service-



Reconstructed from the method description in the supplied paper

Figure 1. The balance between Al-Biruni Earth Radius Optimization (BER) and Sine Cosine Algorithm (SCA) for the Network Requests dataset. The schematic is reconstructed from the method description because the source artwork was not embedded in the supplied PDF.

request scheduling as an optimization problem and uses integer programming to minimize latency. Recognizing the problem's NP-hard nature, it introduces a customized genetic algorithm (GA) as a scheduling heuristic. In simulations with dynamic conditions, the GA improves overall latency by 21.9% to 46.6% and increases the number of requests meeting their deadlines by up to 31% compared with other techniques.

Deep-learning methods can extract important network-traffic features for identifying malicious agents and intrusion detection, although challenges remain. Convolutional neural networks can be used for feature extraction. One study, titled "Anomaly-Based Intrusion Detection System in the Internet of Things Using a Convolutional Neural Network and Multi-Objective Enhanced Capuchin Search Algorithm," combines a CNN with a binary multi-objective Capuchin Search Algorithm (CSA) for feature selection. The model is trained and tested using TON-IoT and NSL-KDD datasets and reports acceptable accuracy in detecting malicious agents. However, deep learning can make the model time-consuming.

Optimization algorithms for feature selection are often combined with improvement operators or other algorithms to address their weaknesses. In "An Improved PIO Feature Selection Algorithm for IoT Network Intrusion Detection System Based on Ensemble Learning," an improved pigeon-inspired optimization (PIO) algorithm is combined with local search for intrusion-detection feature selection. An ensemble-learning mechanism based on multiple one-class classifiers is also used. The model is evaluated on NLS-KDD, BoT-IoT, KDDCUP99, and UNSW-NB15 datasets and compared with other algorithms using several criteria; the reported results indicate acceptable performance.

Another study, "Research on Hybrid Intrusion Detection Based on Improved Harris Hawk Optimization Algorithm," proposes an improved Harris Hawk optimization algorithm to select features and reduce noisy features. A Singer chaotic map is used for initialization, multi-information fusion is used to select the best solution, and an additional strategy is used to obtain optimal features. The data are preprocessed using a deep denoising autoencoder and KNN to address network-traffic imbalance. A deep neural network performs classification using KDD CUP99, NSL-KDD, and UNSW-NB15. The results indicate acceptable intrusion-detection performance, although the method also has a relatively long running time.

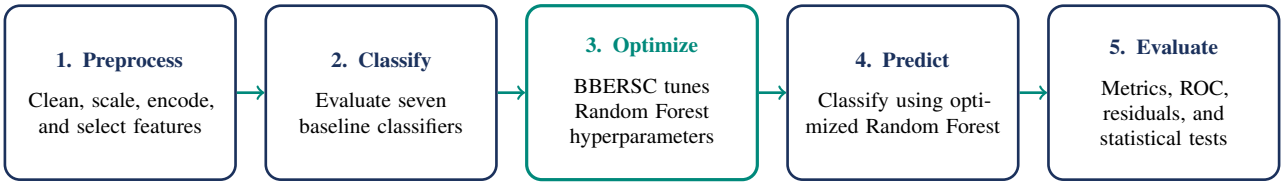
Cloud computing has a central role in data processing, but IoT devices generate increasing amounts of data [25]. Fog computing places storage and computation capabilities at the network edge, close to end users, helping mitigate network traffic and address limitations of cloud computing. Research surveys its taxonomy, differences from cloud and edge computing, applications, communication and storage technologies, and associated challenges. In Intelligent Transportation Systems, Vehicular Ad Hoc Networks (VANETs) are an important enabler [26]. The growth of vehicular applications and data increases demand for communication, computation, and storage resources under strict response-time and network-capacity requirements. Mobile Edge Computing (MEC) moves computation and storage from distant cloud resources to the network edge near vehicular users, reducing latency and bandwidth use. The integration of vehicular networking with MEC has motivated Vehicular Edge Computing (VEC); related work reviews its state, architecture, drivers, benefits, disadvantages, and potential real-world applications.

3. MATERIALS AND METHODS

The proposed methodology classifies network requests using an advanced Al-Biruni optimization algorithm. Figure 2 summarizes the five stages described in the study.

The proposed classification methodology consists of the following stages:

1. **Dataset preprocessing:**
    - **Data cleaning:** Identify and handle missing dataset values using imputation techniques or by removing instances with missing values.
    - **Feature scaling:** Normalize or standardize features so that each contributes equally to further processing.
    - **Encoding categorical variables:** Convert categorical variables into numerical representations using techniques such as one-hot encoding or label encoding.
    - **Feature selection:** Select features relevant to the classification task, potentially using feature importance or dimensionality-reduction methods such as Principal Component Analysis.
2. **Classification using machine-learning algorithms:**
    - **SGD Classifier:** A stochastic gradient descent classifier that optimizes the loss function iteratively.
    - **Logistic Regression:** A linear model used for binary classification.
    - **Decision Tree Classifier:** Constructs a decision tree from feature values and predicts the class label for an instance.



Reconstructed from the five methodological steps in the supplied paper

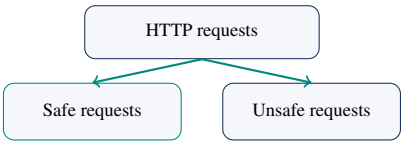
Figure 2. The steps of the proposed methodology. The flowchart is reconstructed from the five steps described in the supplied paper.

- **Gaussian Naive Bayes:** Assumes that features are independent and follow a Gaussian distribution.
- **K-Nearest Neighbors (KNN):** Classifies instances by the majority class among their k nearest neighbors.
- **Support Vector Machine (SVM):** Separates classes by finding a hyperplane that maximizes the margin between them.
- **Random Forest Classifier:** Constructs an ensemble of decision trees and aggregates their predictions to improve accuracy and reduce overfitting.

3. **Random Forest parameter optimization:** Apply the advanced Al-Biruni optimization algorithm to tune parameters such as the number of trees, tree depth, and minimum samples per leaf node.
4. **Classification with the optimized Random Forest:** Apply the classifier with optimized hyperparameters to the dataset so that it operates with the selected configuration.
5. **Performance measurement:**
   - **Accuracy:** The ratio of correctly classified instances to the total number of instances.
   - **ROC curve:** Plots the true-positive rate against the false-positive rate across classification thresholds.
   - **Residual analysis:** Examines differences between predicted and actual values to assess model performance.
   - **Homoscedasticity:** Tests whether error variance is constant across levels of the predictor variables.
   - **QQ plot:** Compares the distribution of residuals with a normal distribution.
   - **Heatmap:** Visualizes performance metrics across scenarios, such as different hyperparameters or feature combinations.
   - **ANOVA and Wilcoxon tests:** Assess the significance of differences between groups or paired samples.
   - **Regression analysis:** Examines relationships between predictor variables and the target using regression techniques.

3.1 Dataset

The data in this study were drawn from Kaggle [27], a source of machine-learning and data-science datasets. The dataset comprises HTTP requests [28], each with parameters capturing multiple associated aspects. Figure 3 was intended to visualize the distribution of safe and unsafe requests in the selected dataset. This graphical interpretation introduces the characteristics that provide background for the study.



Class counts not provided in source

Figure 3. Schematic classification of safe and unsafe requests; class counts were not provided in the source PDF.

3.1.1 Proposed BBERSC Algorithm

The BBERSC (Balance Between Al-Biruni Earth Radius Optimization and Sine Cosine Algorithm) algorithm is the basis of this study and aims to achieve a balance between BER and SCA. This combination uses both optimization methods and their respective characteristics to improve performance for a given dataset. The BBERSC algorithm is intended to provide a successful and robust optimization process tailored to the dataset and the complexity of the classification tasks. Its implementation and performance are discussed in the experimental results, including its ability to increase the accuracy of the Random Forest Classifier. The pseudocode is presented below.

Algorithm 1. The proposed BBERSC optimization algorithm

Initialize BER population $\mathbf{P}_i$ $(i = 1,2,\ldots,d)$ with size d, iterations Max_{iter}, fitness function F_n, $t = 1$, n_1, n_2, a, and $r_1, r_2, r_3, r_4, r_5, r_6, r_7$.
Calculate fitness function F_n for each $\mathbf{P}_i$.
Find the best solution as $\mathbf{P}^*$.
while $t \leq \text{Max}_{\text{iter}}$ **do**.
  if $R \leq 0$ **then**.
    for each solution in the exploration group **do**.
      Head toward the best solution.
      $r = h \dfrac{\cos(x)}{1-\cos(x)}$.
      $\mathbf{D} = r_1\left(\mathbf{P}(t) - 1\right)$.
      $\mathbf{P}(t+1) = \mathbf{P}(t) + \mathbf{D}(2r_2 - 1)$.
    end for.
    for each solution in the exploitation group **do**.
      Apply elitism to retain the best solution.

14: $\mathbf{D} = r_3\big(\mathbf{L}(t) - \mathbf{P}(t)\big)$.

15: $\mathbf{P}(t+1) = r_2\big(\mathbf{P}(t) + \mathbf{D}\big)$.

16: $k = 1 + \dfrac{2t^2}{\mathrm{Max}_{\mathrm{iter}}}$.

17: Investigate the area around the best solution:

18: $\mathbf{P}'(t+1) = r_1\big(\mathbf{P}^*(t) + k\big)$.

19: Compare $\mathbf{P}(t+1)$ and $\mathbf{P}'(t+1)$ and select the best $\mathbf{P}^*$.

20: **if** the best fitness has not changed in the last two iterations **then**.

21: Mutate the solution: $\mathbf{P}(t+1) = kz^2 - h\dfrac{\cos(x)}{1-\cos(x)}$.

22: **end if**.

23: **end for**.

24: Update F_n for each $\mathbf{P}(t)$ using BER.

25: **else**.

26: Update agents' positions using the sine-cosine optimization method.

27: **end if**.

28: Update BER and SCA parameters, and set $t = t+1$.

29: **end while**.

30: **Return** $\mathbf{P}^*$.

3.2 Hyperparameter Tuning

The search for optimal model performance includes Random Forest hyperparameter tuning. The chosen parameter set is an important determinant of the model's ability to identify patterns. The study investigates:

- `n_estimators`: The number of trees in the forest. A larger number can improve performance but also substantially increase computational complexity, so a balance is needed [29].
- `max_features`: The number of feature subsets considered at each split. A higher value can produce more varied trees but may also increase overfitting, so selecting the number of features per split is important [30].
- `max_depth`: The maximum number of levels in each decision tree. Greater depth can capture more complex training patterns but can also increase overfitting; controlling depth helps balance the model [31].
- `min_samples_split`: The minimum number of data points required at a node before splitting. An appropriate minimum helps prevent creation of nodes with insufficient data and supports generalization [32].
- `min_samples_leaf`: The minimum number of data points at a leaf node. A minimum leaf size limits tree coarseness and prevents excessive fitting to the training data [33].
- `bootstrap`: Specifies whether samples are drawn with replacement. This choice affects the diversity of trees in the forest [34].

The hyperparameter-tuning procedure searches several combinations and evaluates performance using a selected metric [35, 36]. The resulting configuration is then used to assess the predictive capabilities of the tuned model [37].

4. EXPERIMENTAL RESULTS

This section presents empirical evidence for the proposed framework, which integrates machine-learning models with optimization mechanisms to improve network security. It focuses on BBERSC and its role in improving the accuracy of network-request classification. The quality of Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine classifiers is evaluated.

4.1 Baseline Classifier Performance

Table 1 compares the performance of baseline classifiers on the test dataset. The principal indicators are sensitivity, specificity, PPV, and NPV, in addition to accuracy and F-score. Random Forest has the highest reported baseline accuracy, 0.810 (81%), compared with the other models. The table provides an overview of the relative strengths and weaknesses of the classifiers for network security.

Table 1. Results of the basic classification models on the tested dataset.

Classification	Acc.	Sens.	Spec.	PPV	NPV	F1
SGD Classifier	0.630	0.163	0.982	0.875	0.275	0.609
Logistic Regression	0.715	0.663	0.754	0.671	0.667	0.748
Decision Tree	0.735	0.756	0.719	0.670	0.710	0.796
Gaussian Naive Bayes	0.745	0.640	0.825	0.733	0.683	0.752
K-Nearest Neighbors	0.785	0.640	0.895	0.821	0.719	0.767
SVM	0.785	0.581	0.939	0.877	0.699	0.748
Random Forest	0.810	0.628	0.947	0.900	0.740	0.771

Figure 4 compares the performance metrics of the classification algorithms. The visualization is intended to help readers interpret accuracy, sensitivity, specificity, and other measures, identify trends and patterns, and compare the performance of the models for network security.

4.2 Optimization Algorithm Performance

Table 2 compares optimization algorithms integrated with Random Forest for network-request classification. BBERSC obtains the highest reported accuracy (0.990), sensitivity (0.993), specificity (0.988), PPV (0.988), NPV (0.993), and F-score (0.990); each reported measure exceeds 98%. The individual components BER and SCA achieve accuracies of 0.966 and 0.954, respectively, indicating the reported improvement from their hybrid combination. Among the other methods, PSO achieves 0.940 accuracy, GWO 0.922, and WOA 0.911. The traditional Genetic Algorithm (GA) has the lowest reported accuracy, 0.882. The reported results attribute BBERSC's performance to balancing exploration and exploitation and present it as suitable for network-security classification.

Figure 5 presents the reported accuracy values for the optimization algorithms when combined with Random Forest. The comparison is intended to make trends apparent and show how the optimization algorithms affect classification accuracy

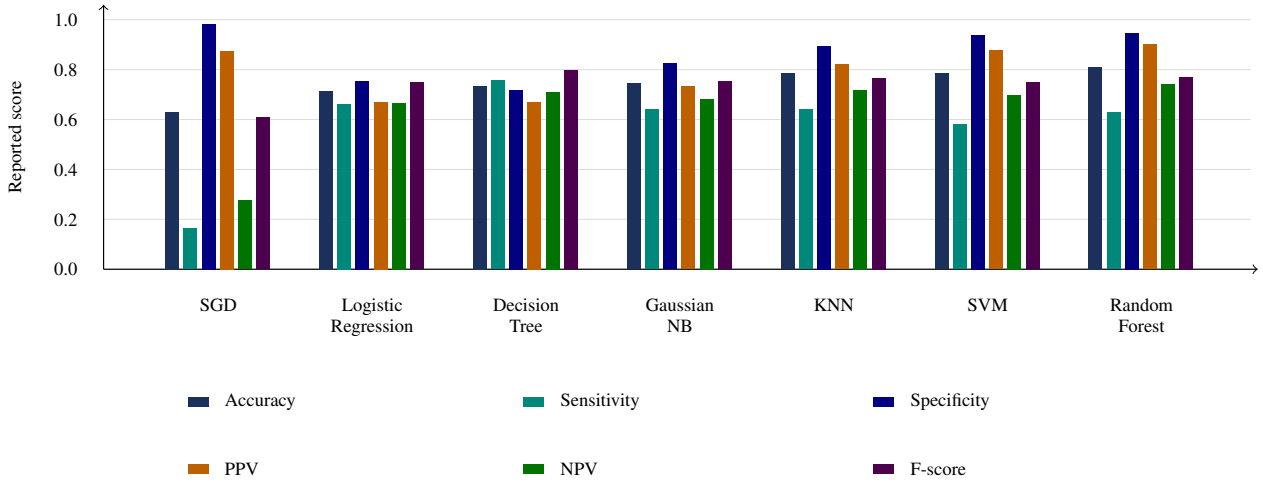


Figure 4. Accuracy investigation of various classification algorithms, reconstructed from the values in Table 1.

Table 2. Performance evaluation of various optimization algorithms with the Random Forest Classifier.

Model	Acc.	Sens.	Spec.	PPV	NPV	F1
BBERSC	0.990	0.993	0.988	0.988	0.993	0.990
BER	0.966	0.969	0.964	0.964	0.969	0.966
SCA	0.954	0.950	0.958	0.957	0.951	0.954
PSO	0.940	0.930	0.950	0.950	0.930	0.940
GWO	0.922	0.915	0.931	0.941	0.901	0.928
WOA	0.911	0.909	0.913	0.935	0.881	0.922
GA	0.882	0.879	0.887	0.924	0.825	0.901

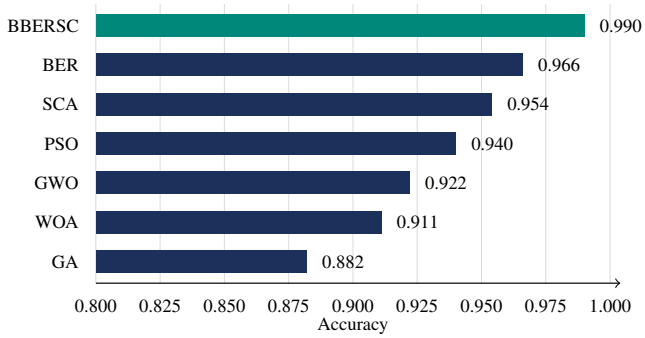


Figure 5. Accuracy investigation of various optimization algorithms, reconstructed from the values in Table 2.

and network security.

The source paper describes an accuracy histogram for the optimization algorithms (Figure 6), but its artwork and repeated-run measurements were not embedded in the supplied PDF. The histogram below is reconstructed from the seven accuracy values in Table 2, one reported value per algorithm; it does not represent run-to-run variability.



Figure 6. Histogram of the seven reported algorithm accuracies in Table 2; one value is available per algorithm.

Figure 7 was described as showing residual values and a

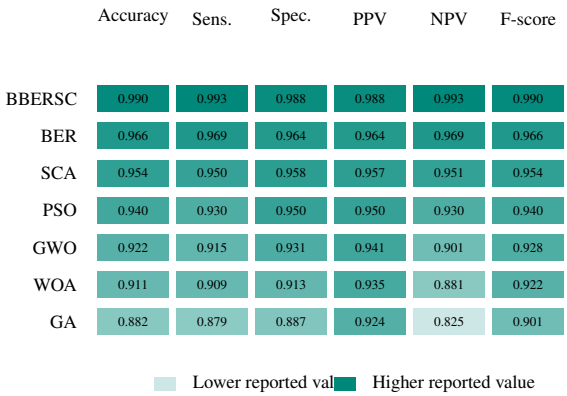


Figure 7. Heatmap of reported performance metrics, reconstructed from Table 2. Residual data were unavailable in the supplied PDF.

heatmap. Residual observations were not included in the supplied PDF, so the plot below instead visualizes the six reported performance metrics from Table 2; it does not represent residuals.

The paper describes the Receiver Operating Characteristic (ROC) curve for BBERSC as showing the trade-off between sensitivity and specificity across classification thresholds (Figure 8). The source provides one operating point through the reported sensitivity and specificity, but no threshold-level scores to reconstruct a full ROC curve.

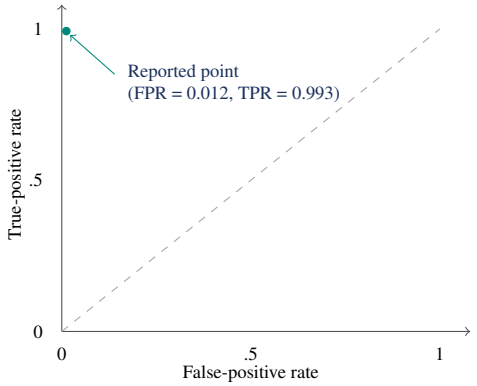


Figure 8. Available BBERSC ROC operating point from reported sensitivity and specificity; the full curve cannot be reconstructed without threshold-level data.

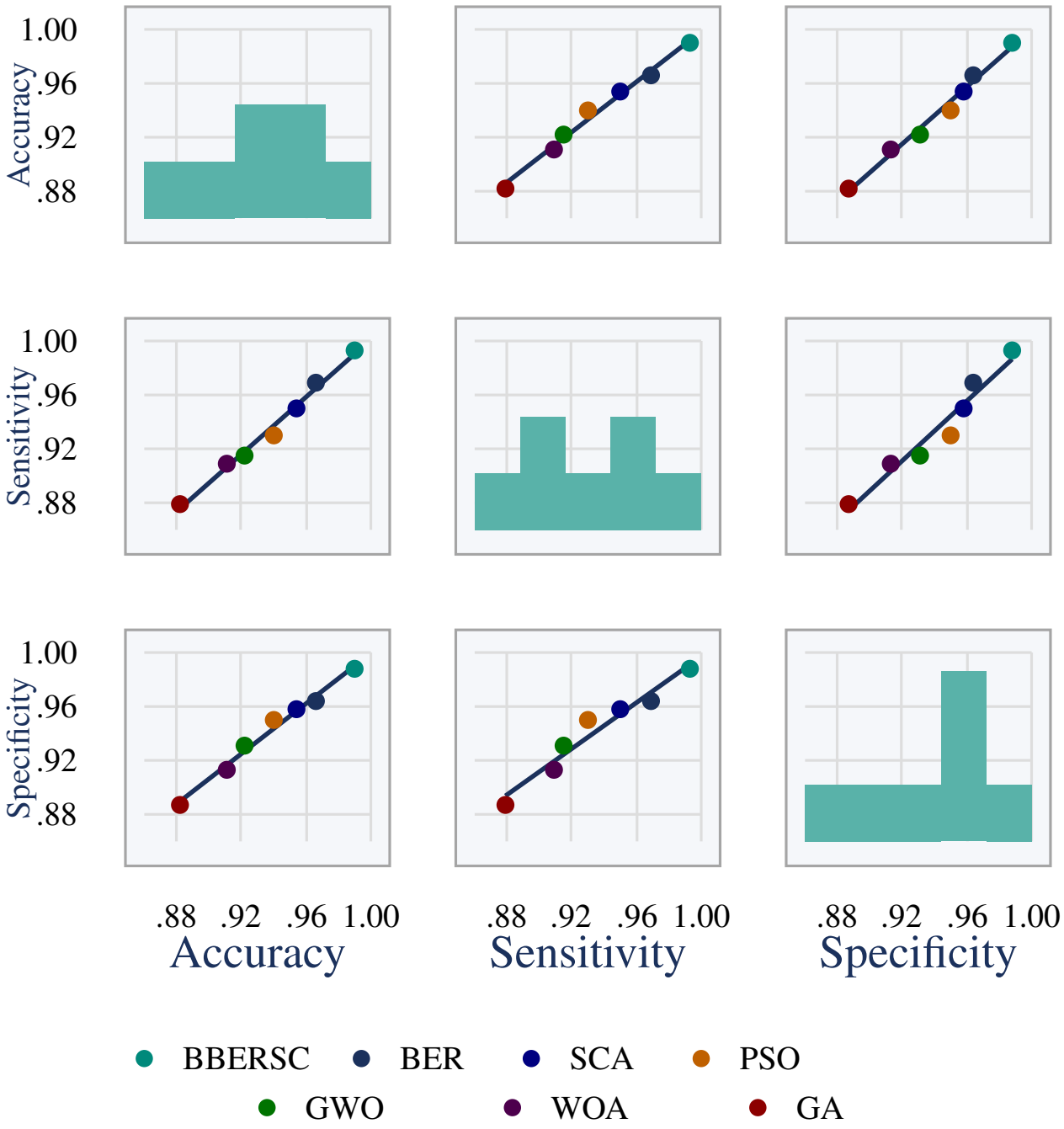


Figure 9. Pair plot of reported Accuracy, Sensitivity, and Specificity values for the seven optimization algorithms in Table 2. Reconstructed from aggregate scores; fitted lines are descriptive and do not represent repeated-run analyses.

The source paper includes a pair plot with regression lines for BBERSC (Figure 9), but the original artwork and run-level observations were not embedded in the supplied PDF. To ensure Figure 9 is visible without inventing unavailable values, the reconstruction plots Accuracy, Sensitivity, and Specificity from the seven aggregate optimizer scores in Table 2. The fitted lines are descriptive summaries across those algorithms, not repeated-run statistical results.

4.3 Statistical Analysis

Table 3 reports the ANOVA results for BBERSC and the compared algorithms. The table includes sums of squares, degrees of freedom, mean squares, the F-statistic, and the P-value for treatment and residual effects [38]. These statistics quantify differences among algorithms and are presented as evidence for evaluating the proposed method. The reported treatment effect has $F(6, 133) = 764.3$ and $P < 0.0001$.

Table 3. ANOVA statistical results based on BBERSC and the compared algorithms.

Source	SS	df	MS	$F(6,133)$	P
Treatment	0.148	6	0.02467	764.3	< 0.0001
Residual	0.004293	133	3.23×10^{-5}		
Total	0.1523	139			

Table 4 presents the Wilcoxon signed-rank test results for BBERSC and the other optimization algorithms. The reported P-values and significance indicators are used to assess differences and compare the stability and accuracy of BBERSC with the other models.

Table 4. Wilcoxon signed-rank test statistical results for BBERSC and other algorithms.

Algorithm	P	Test	Signif. ($\alpha = .05$)	Accuracy
BBERSC	.002	Exact	Significant	0.9902
BER	.002	Exact	Significant	0.9665
SCA	.002	Exact	Significant	0.9539
PSO	.002	Exact	Significant	0.9401
GWO	.002	Exact	Significant	0.9222
WOA	.002	Exact	Significant	0.9109
GA	.002	Exact	Significant	0.8824

5. CONCLUSION AND FUTURE DIRECTIONS

This study integrates BBERSC with several machine-learning models as an approach to strengthening network security. The proposed framework includes Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors, Naive Bayes, and SGD Classifier. The Random Forest model has a reported accuracy of 0.99017198. Statistical validation using ANOVA and the Wilcoxon signed-rank test supports the reported comparisons.

Future work includes addressing real-world implementation and scalability so that the framework can adapt to changing environments. Dynamic datasets that represent real-world cases may improve adaptation to new cyber threats. Further research should investigate ensemble techniques that allow different models to cooperate in robust intrusion-detection systems. Continued refinement of optimization algorithms may also improve the framework. This study provides a basis for future research into adaptive and intelligent intrusion detection. Because cyber threats continue to change, network-security methods require ongoing development; the findings are intended to contribute both theoretical and practical approaches to proactive protection against cyberattacks.

REFERENCES

[1] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[2] D. S. Khafaga *et al.*, "An al-biruni earth radius optimization-based deep convolutional neural network for classifying monkeypox disease," *Diagnostics*, vol. 12, no. 11, p. 2892, 2022.

[3] D. R. Nayak, R. Dash, B. Majhi, and S. Wang, "Combining extreme learning machine with modified sine cosine algorithm for detection of pathological brain," *Computers & Electrical Engineering*, vol. 68, pp. 366–380, 2018.

[4] V. Jackins, S. Vimal, M. Kaliappan, and M. Y. Lee, "Ai-based smart prediction of clinical disease using random forest classifier and naive bayes," *The Journal of Supercomputing*, vol. 77, no. 5, pp. 5198–5219, 2021.

[5] C. M. Yesilkanat, "Spatio-temporal estimation of the daily cases of covid-19 in worldwide using random forest machine learning algorithm," *Chaos, Solitons & Fractals*, vol. 140, p. 110210, 2020.

[6] T. Cuong-Le, T. Nghia-Nguyen, S. Khatir, P. Trong-Nguyen, S. Mirjalili, and K. D. Nguyen, "An efficient approach for damage identification based on improved machine learning using pso-svm," *Engineering with Computers*, vol. 38, no. 4, pp. 3069–3084, 2022.

[7] R. Alkanhel *et al.*, "Enhancing wireless sensor network efficiency through al-biruni earth radius optimization," *Computers, Materials & Continua*, vol. 79, no. 3, pp. 3549–3568, 2024.

[8] Y. Tikhamarine, D. Souag-Gamane, A. N. Ahmed, O. Kisi, and A. El-Shafie, "Improving artificial intelligence models accuracy for monthly streamflow forecasting using grey wolf optimization (gwo) algorithm," *Journal of Hydrology*, vol. 582, p. 124435, 2020.

[9] E. M. Hassib, A. I. El-Desouky, L. M. Labib, and E.-S. M. El-kenawy, "Woa + brnn: An imbalanced big data classification framework using whale optimization and deep neural network," *Soft Computing*, vol. 24, no. 8, pp. 5573–5592, 2020.

[10] H. Nguyen, X.-N. Bui, Y. Choi, C. W. Lee, and D. J. Armaghani, "A novel combination of whale optimization algorithm and support vector machine with different kernel functions for prediction of blasting-induced fly-rock in quarry mines," *Natural Resources Research*, vol. 30, no. 1, pp. 191–207, 2021.

[11] F. B. Banadkooki *et al.*, "Enhancement of groundwater-level prediction using an integrated machine learning model optimized by whale algorithm," *Natural Resources Research*, vol. 29, no. 5, pp. 3233–3252, 2020.

[12] A. M. Anter, M. Abd Elaziz, and Z. Zhang, "Real-time epileptic seizure recognition using bayesian genetic whale optimizer and adaptive machine learning," *Future Generation Computer Systems*, vol. 127, pp. 426–434, 2022.

[13] S. Nusinovici *et al.*, "Logistic regression was as good as machine learning for predicting major chronic diseases," *Journal of Clinical Epidemiology*, vol. 122, pp. 56–69, 2020.

[14] H. Lu and X. Ma, "Hybrid decision tree-based machine learning models for short-term water quality prediction," *Chemosphere*, vol. 249, p. 126169, 2020.

[15] Y. Ao, H. Li, L. Zhu, S. Ali, and Z. Yang, "The linear random forest algorithm and its advantages in machine learning assisted logging regression modeling," *Journal of Petroleum Science and Engineering*, vol. 174, pp. 776–789, 2019.

[16] D. A. Pisner and D. M. Schnyer, "Support vector machine," in *Machine Learning*. Elsevier, 2020, pp. 101–121.

[17] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of k-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning," *Decision Analytics Journal*, vol. 3, p. 100071, 2022.

[18] A. Asadikia, A. Rajabifard, and M. Kalantari, "Systematic prioritisation of sdgs: Machine learning approach," *World Development*, vol. 140, p. 105269, 2021.

[19] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, "Evaluating the quality of machine learning explanations: A survey on methods and metrics," *Electronics*, vol. 10, no. 5, p. 593, 2021.

[20] H. Elazhary, "Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *Journal of Network and Computer Applications*, vol. 128, pp. 105–140, 2019.

[21] A. Yassine, S. Singh, M. S. Hossain, and G. Muhammad, "Iot big data analytics for smart homes with fog and cloud computing," *Future Generation Computer Systems*, vol. 91, pp. 563–573, 2019.

[22] M. Malawski, A. Gajek, A. Zima, B. Balis, and K. Figiela, "Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions," *Future Generation Computer Systems*, vol. 110, pp. 502–514, 2020.

[23] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling internet of things requests to minimize latency in hybrid fog-cloud computing," *Future Generation Computer Systems*, vol. 111, pp. 539–551, 2020.

[24] Y. Lu and X. Xu, "Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 92–102, 2019.

[25] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1145–1168, 2021.

[26] L. J. Muhammad, E. A. Algehyne, S. S. Usman, A. Ahmad, C. Chakraborty, and I. A. Mohammed, "Supervised machine learning models for prediction of covid-19 infection using epidemiology dataset," *SN Computer Science*, vol. 2, no. 1, p. 11, 2021.

[27] I. Sreeram and V. P. K. Vuppala, "Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied Computing and Informatics*, vol. 15, no. 1, pp. 59–66, 2019.

[28] M. A. Hassan *et al.*, "Evaluation of energy extraction of pv systems affected by environmental factors under real outdoor conditions," *Theoretical and Applied Climatology*, vol. 150, no. 1–2, pp. 715–729, 2022.

[29] W. Tong, Q. Wei, H.-Y. Yan, M.-G. Zhang, and X.-M. Zhu, "Accelerating inverse crystal structure prediction by machine learning: A case study of carbon allotropes," *Frontiers of Physics*, vol. 15, no. 6, p. 63501, 2020.

[30] H. Li *et al.*, "Machine learning assisted predicting and engineering specific surface area and total pore volume of biochar," *Bioresource Technology*, vol. 369, p. 128417, 2023.

[31] G. Li, Y. Sun, and C. Qi, "Machine learning-based constitutive models for cement-grouted coal specimens under shearing," *International Journal of Mining Science and Technology*, vol. 31, no. 5, pp. 813–823, 2021.

[32] K. Sandunil, Z. Bennour, H. Ben Mahmud, and A. Giwelli, "Effects of tuning hyperparameters in random forest regression on reservoir's porosity prediction: Case study, volve oil field, north sea," in *Proceedings of the ARMA US Rock Mechanics/Geomechanics Symposium*, Atlanta, Georgia, USA, 2023, aRMA-2023-0660.

[33] M. H. D. M. Ribeiro, R. G. Da Silva, S. R. Moreno, V. C. Mariani, and L. D. S. Coelho, "Efficient bootstrap stacking ensemble learning model applied to wind power generation forecasting," *International Journal of Electrical Power & Energy Systems*, vol. 136, p. 107712, 2022.

[34] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," *Informatics*, vol. 8, no. 4, p. 79, 2021.

[35] P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning, "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data," *Ecological Modelling*, vol. 406, pp. 109–120, 2019.

[36] H. A. Fayed and A. F. Atiya, "Speed up grid-search for parameter selection of support vector machines," *Applied Soft Computing*, vol. 80, pp. 202–210, 2019.

[37] R. Turner *et al.*, "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020," in *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, 2021, pp. 3–26.

[38] A. Djaafari *et al.*, "Hourly predictions of direct normal irradiation using an innovative hybrid lstm model for concentrating solar power projects in hyper-arid regions," *Energy Reports*, vol. 8, pp. 15 548–15 562, 2022.