Journal of Soft Computing Exploration
Homepage: htttps://shmpublisher.com/index.php/joscex
p-ISSN: 2746-7686
e-ISSN: 2746-0991

.

Improving intrusion detection performance using bayesian hyperparameter optimization for supervised network traffic classification

Dahlan[1], **Dadan Saepul Ramdan**[2]
[1, 2]Department of Informatics Engineering, Politeknik TEDC Bandung University, Indonesia

Article Info

Article history:

Received May 03, 2026
Revised May 16, 2026
Accepted May 25, 2026

Keywords:

Intrusion detection
Bayesian
Hyperparameter optimization
Network traffic
Classification

ABSTRACT

The rapid growth of networked systems has increased the complexity of network traffic and the risk of cyber-attacks, making intrusion detection more challenging. Machine learning approaches have been widely used to address this issue; however, their performance often depends on appropriate hyperparameter settings. This study examined the effect of Bayesian-based hyperparameter optimization on the performance of supervised machine learning models for network traffic classification. A publicly available dataset was used, consisting of various traffic-related features and labeled instances indicating normal or malicious activity. Several machine learning models, including Random Forest, Decision Tree, AdaBoost, Logistic Regression, Gradient Boosting, and Naïve Bayes, were evaluated. Each model was tested using default parameters and then optimized using Bayesian Optimization. The performance was assessed using accuracy, precision, recall, and F1-score. The results showed that ensemble-based models, particularly Gradient Boosting and Random Forest, achieved the best performance after optimization, with accuracy values above 89% and strong F1-scores. However, the findings also revealed a trade-off between precision and recall, where higher precision was often associated with lower detection of certain attack instances. In contrast, simpler models such as Logistic Regression showed lower performance, indicating their limitations in capturing complex patterns. Overall, the study demonstrated that Bayesian-based hyperparameter optimization contributed to improving model performance and provided a more reliable approach for network traffic classification.

Corresponding Author:

Dahlan,
Department of Informatics Engineering,
Politeknik TEDC Bandung University,
Pasantren KM 2, Cibabat, Cimahi, 40513, West Java, Indonesia
Email: dahlan@poltektedc.ac.id
https://doi.org/10.52465/joscex.v7i2.87

1. INTRODUCTION

Rapid growth of networked systems has brought not only convenience but also a noticeable increase in security-related risks. The widespread use of cloud services, IoT devices, and high-speed communication infrastructures has significantly expanded the attack surface of modern networks [1]–[3]. As network traffic

continues to grow in both volume and complexity, identifying malicious activities becomes more challenging, especially when attacks are designed to mimic normal behavior [3], [4]. Traditional intrusion detection systems, which largely depend on predefined rules or signature-based mechanisms, often struggle to keep up with these changes [3]. They tend to perform well for known attack patterns, but their effectiveness decreases when dealing with previously unseen or evolving threats [5]. This situation highlights the need for more adaptive and data-driven approaches that are capable of learning from traffic patterns and improving detection performance in more dynamic and realistic network environments.

Although machine learning (ML)-based approaches have been widely adopted in intrusion detection systems due to their ability to learn complex traffic patterns, several limitations still remain, particularly when models are used without proper optimization [6], [7]. In practice, many ML classifiers tend to rely on default parameter settings, which can lead to suboptimal performance and higher false alarm rates when dealing with diverse and high-dimensional network traffic. Recent studies have shown that baseline ML models often exhibit noticeable performance gaps before any tuning is applied, indicating their limited ability to generalize across different traffic scenarios [8]. In addition, the effectiveness of ML-based intrusion detection is often influenced by factors such as parameter sensitivity, data imbalance, and feature complexity, which are not adequately handled in standard configurations [9]. As highlighted in recent literature, building an effective ML-based intrusion detection system is still a non-trivial task, requiring careful adjustment of model parameters to achieve reliable detection performance in real-world environments [10]. These challenges suggest that relying solely on conventional ML models without optimization may leave critical gaps in detecting evolving or subtle network attacks.

Study by [11] proposed a machine learning-based intrusion detection system using the Random Forest classifier on the CICIDS2017 dataset. The study focused on improving detection performance through feature selection and data preprocessing, achieving a weighted F1-score close to 99.8%. These results indicate that tree-based models are highly effective for structured network traffic classification. However, the study did not place systematic hyperparameter optimization as its main experimental focus, leaving limited insight into how parameter configurations affect model adaptability and detection stability.

Study by [12] analyzed several machine learning models, including Decision Tree and Random Forest, for intrusion detection using the UNSW-NB15 and TON_IoT datasets. The study emphasized feature selection to improve classification performance and demonstrated that machine learning models can achieve competitive results in detecting multiple attack categories. Nevertheless, the model configurations were largely based on standard settings, and the role of structured hyperparameter tuning was not examined in depth. This limitation makes it difficult to determine whether the reported performance could be further improved through systematic optimization.

Study by [13] compared multiple machine learning and deep learning models for binary and multiclass intrusion detection. The findings showed that models such as Random Forest and XGBoost achieved strong classification performance across different evaluation scenarios. However, the study mainly concentrated on performance comparison among classifiers rather than analyzing how optimization affects each model individually. As a result, the influence of parameter tuning on model reliability and generalization remains less clearly addressed.

Study by [14] explored a machine learning and deep learning approach for anomaly-based network intrusion detection. The proposed framework demonstrated promising results, particularly through the use of model combination and improved learning capability for complex attack patterns. Even so, the performance gains were primarily associated with architectural and ensemble-oriented strategies rather than a dedicated investigation of Bayesian-based hyperparameter optimization. Therefore, the extent to which systematic tuning contributes to performance improvement is still not fully clarified.

Study by [15] presented a systematic literature review of machine learning-based intrusion detection systems, highlighting that detection performance is strongly influenced by dataset characteristics, feature quality, and model configuration. The review also showed that many previous works tend to prioritize model selection, feature engineering, or hybrid architectures, while optimization strategies are not always treated as a central experimental component. This observation reinforces the need for further investigation into how structured hyperparameter optimization can improve classification reliability and provide a clearer understanding of model behavior in intrusion detection tasks.

Although previous studies have demonstrated the effectiveness of machine learning models for intrusion detection, many of them primarily emphasize model comparison, feature selection, or ensemble construction, while the systematic role of hyperparameter optimization across different supervised classifiers remains less explicitly examined. To address this gap, this study investigates Bayesian-based hyperparameter optimization as a central strategy for improving network traffic classification performance. The scientific contribution of this work lies in its comparative evaluation of multiple supervised learning models before and after optimization, allowing a clearer analysis of how parameter tuning influences model behavior, classification reliability, and precision–recall trade-offs. In addition, this study provides practical insight into

the suitability of different optimized models for intrusion detection scenarios, particularly when balancing the need to minimize false alarms and reduce undetected attack instances. Therefore, this work contributes not only by applying Bayesian Optimization, but also by offering a more interpretable and practically oriented assessment of its impact on supervised intrusion detection performance.

2. METHOD

Proposed Workflow

The overall workflow of the proposed approach is seen in Figure 1. The process begins with the collection of raw network traffic data obtained from a publicly available dataset. This data typically contains various features representing network flow characteristics, which are then prepared through a preprocessing stage. In this stage, irrelevant attributes are removed, missing values are handled, and the data is transformed into a suitable format for further analysis. After preprocessing, a label and attribute mapping process is performed to ensure that the data is properly structured for classification tasks. This step involves organizing the features and assigning appropriate labels to distinguish between normal and malicious traffic. Once the data is prepared, it is divided into two subsets, where 80% of the data is used for training and the remaining 20% is reserved for testing. The hold-out 80:20 split was adopted to provide a clear separation between model development and final testing on unseen data. This strategy allows the optimized models to be evaluated using an independent testing subset that is not involved in the training process. Although this approach is suitable for an initial comparative assessment of classifier performance, the use of repeated experiments or Stratified K-Fold Cross Validation may provide a more robust estimation of model generalization in future studies.

The training data is then used to build several supervised machine learning models, including Random Forest (RF), Decision Tree (DT), AdaBoost, Logistic Regression (LR), Gradient Boosting, and Naïve Bayes (NB). At this stage, Bayesian optimization is applied to tune the hyperparameters of each model. Instead of relying on default configurations, the optimization process iteratively searches for parameter combinations that can improve model performance. The optimized models are then evaluated based on their performance using the training process, and the best-performing model is selected for further validation. The selected model is subsequently tested using the unseen testing data to assess its generalization capability. Finally, the classification results are analyzed using a confusion matrix to provide a more detailed evaluation of model performance, including metrics such as accuracy, precision, recall, and F1-score. The overall process concludes with the identification of the most effective model for network traffic classification based on the applied optimization strategy.

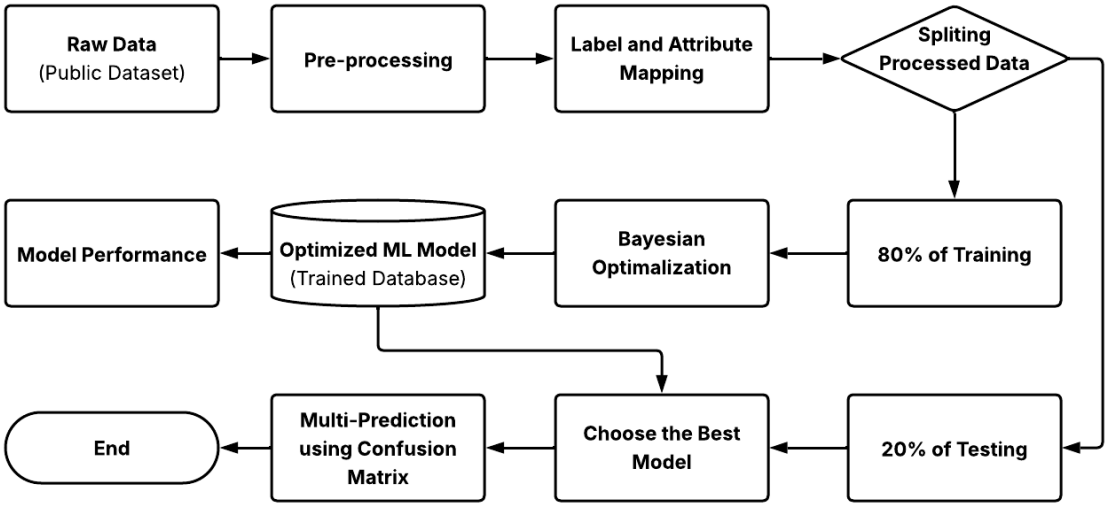

Figure 1. Proposed workflow

Data Collection

The dataset used in this study was obtained from a publicly available source on Kaggle [16]. It consists of 9,537 network traffic records that represent communication sessions involving both normal and potentially malicious activities. Each record describes a single session through several traffic-related and behavioral

attributes that can be used to identify suspicious access patterns. The target variable in this dataset is labeled as attack_detected, which indicates whether a network session belongs to the normal class or the attack class. The remaining variables are treated as attributes that describe different aspects of the network traffic. These attributes include both numerical and categorical features, allowing the models to capture a range of patterns from basic traffic statistics to more context-specific indicators such as protocol type and encryption usage. The dataset contains features such as session identifiers, packet size, protocol type, number of login attempts, session duration, encryption methods, IP reputation scores, and failed login counts.

To provide a clearer overview of the dataset structure, the main characteristics of the dataset are summarized in Table 1. As seen in Table 1, the dataset includes a diverse set of features that represent both traffic-level and behavior-level characteristics. This combination allows the classification models to capture not only statistical patterns but also contextual indicators that may be associated with anomalous or malicious activities. The presence of both numerical and categorical attributes also provides a more realistic representation of network traffic data, which is essential for evaluating the robustness of machine learning models in intrusion detection tasks.

Table 1. Raw dataset characteristics

No	Attribute Name	Data Type	Description
1	session_id	Categorical	Unique identifier for each network session
2	network_packet_size	Numerical	Size of transmitted packets within the session
3	protocol_type	Categorical	Type of communication protocol (e.g., TCP, UDP)
4	login_attempts	Numerical	Number of login attempts within a session
5	session_duration	Numerical	Duration of the network session
6	encryption_used	Categorical	Type of encryption applied (e.g., DES, AES)
7	ip_reputation_score	Numerical	Score indicating the trust level of the source IP
8	failed_logins	Numerical	Number of failed login attempts
9	browser_type	Categorical	Type of browser used in the session
10	unusual_time_access	Numerical	Indicator of access occurring at unusual times (binary: 0 or 1)

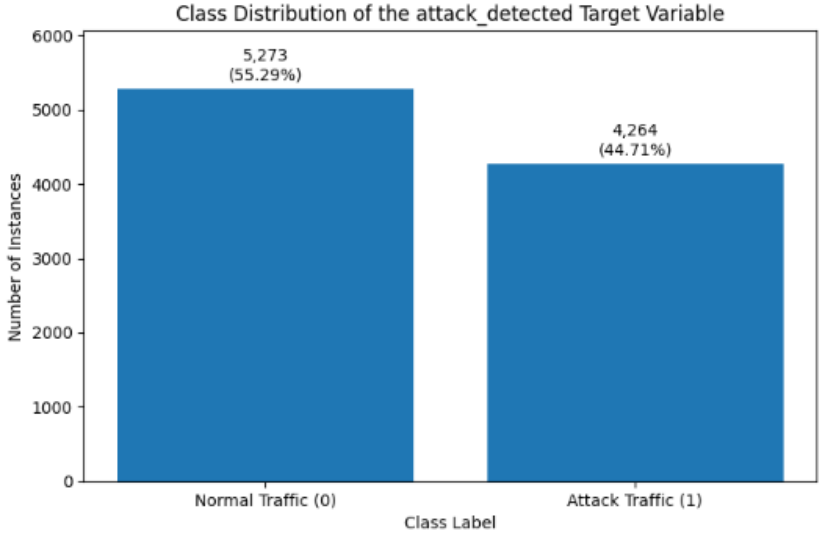

Figure 2. Class distribution

The distribution of the target variable was also examined to assess the balance between normal and attack traffic instances. The dataset contains 5,273 normal traffic records and 4,264 attack traffic records, corresponding to 55.29% and 44.71% of the total data, respectively. This results in an imbalance ratio of approximately 1.24:1, indicating that the dataset is relatively balanced and does not exhibit severe class dominance. Such a distribution is important because substantial imbalance may cause classification models to favor the majority class and produce misleading performance results. In this study, the relatively balanced class composition provides a more reliable basis for evaluating model performance using accuracy, precision, recall, and F1-score. For a more intuitive representation of the target composition, the class distribution of the attack_detected variable is illustrated in Figure 2. The visualization confirms that both normal and attack traffic classes are represented in comparable proportions, supporting the suitability of the dataset for binary intrusion detection classification.

Pre-processing Data

Before training the machine learning models, several preprocessing steps were applied to ensure that the dataset was suitable for classification tasks. This stage was conducted to reduce irrelevant information,

handle incomplete attributes, maintain data consistency, and transform the retained variables into a format that could be processed by the selected classifiers. The initial inspection of the dataset showed that most attributes were complete. However, the encryption_used feature contained 1,966 missing values, while the remaining attributes had no missing entries. In addition, a duplicate-record check confirmed that the dataset did not contain redundant instances. These observations were used as the basis for determining the appropriate preprocessing strategy.

Two attributes were removed during preprocessing. First, session_id was excluded because it only serves as a unique identifier for each network session and does not represent behavioral or traffic-related characteristics relevant to intrusion detection. Second, encryption_used was removed because it contained a substantial number of missing values. Rather than applying an imputation procedure that might introduce additional assumptions into the data, this attribute was excluded to maintain consistency and reduce the risk of bias during model training.

After the removal of these two attributes, the retained dataset consisted of numerical and categorical predictors along with the target variable attack_detected. The numerical features included network_packet_size, login_attempts, session_duration, ip_reputation_score, failed_logins, and unusual_time_access. Meanwhile, the remaining categorical variables were protocol_type and browser_type. Since machine learning classifiers require numerical input, the categorical attributes were converted into numerical representations through an encoding process before model development. This step ensured that the categorical information could be incorporated into the classification procedure without altering the role of the target variable. To provide a clearer summary of the dataset after preprocessing, the retained attributes, their data types, and their descriptions are presented in Table 2. This processed dataset represents the final input structure used for subsequent model training, hyperparameter optimization, and performance evaluation.

Table 2. Processed data

No	Attribute Name	Data Type	Description
1	network_packet_size	Numerical	Size of transmitted packets within the session
2	protocol_type	Categorical	Type of communication protocol (e.g., TCP, UDP)
3	login_attempts	Numerical	Number of login attempts within a session
4	session_duration	Numerical	Duration of the network session
5	ip_reputation_score	Numerical	Score indicating the trust level of the source IP
6	failed_logins	Numerical	Number of failed login attempts
7	browser_type	Categorical	Type of browser used in the session
8	unusual_time_access	Numerical	Indicator of access occurring at unusual times (binary: 0 or 1)
9	attack_detected	Numerical	Target label indicating normal (0) or attack (1)

The preprocessing stage produced a cleaner and more compact dataset by removing non-informative and incomplete attributes, confirming the absence of duplicate records, and preparing categorical features for computational analysis. These steps were intended to improve the reliability of the experimental pipeline while preserving the most relevant traffic and behavioral indicators for intrusion detection classification.

Bayesian Optimization

In this study, Bayesian Optimization is employed to improve the performance of the selected machine learning models by systematically tuning their hyperparameters [17], [18]. Unlike conventional approaches such as grid search or random search, Bayesian Optimization utilizes a probabilistic model to guide the search process, allowing it to explore the parameter space more efficiently. The main idea behind Bayesian Optimization is to build a surrogate model that approximates the relationship between hyperparameter configurations and model performance [19]. Based on this approximation, the method iteratively selects promising parameter combinations that are expected to yield better classification results. The complete hyperparameter search space used in this study is summarized in Table 3.

Table 3. Hyperparameter search space used in bayesian optimization

Model	Hyperparameter	Search Space
Random Forest	n_estimators	50–200
	max_depth	5–20
	min_samples_split	2–10
	min_samples_leaf	1–5

Decision Tree	max_features	sqrt, log2
	max_depth	3–20
	min_samples_split	2–10
	min_samples_leaf	1–5
	criterion	gini, entropy
Ada Boost	n_estimators	50–200
	learning_rate	0.01–1.00
Logistic Regression	C	0.1–5.0
	max_iter	100–500
	solver	lbfgs
Gradient Boosting	n_estimators	50–200
	learning_rate	0.01–0.30
	max_depth	2–6
	min_samples_split	2–10
Naïve Bayes	var_smoothing	1e-12–1e-7

The optimization process was implemented in Python using BayesSearchCV from the scikit-optimize library, which applies sequential model-based optimization. During the search, the Expected Improvement (EI) acquisition function was used to balance exploration and exploitation. Exploration enables the algorithm to investigate less-examined regions of the hyperparameter space, while exploitation directs the search toward configurations that have shown stronger performance in previous evaluations. This mechanism allows Bayesian Optimization to identify suitable hyperparameter combinations more efficiently without requiring an exhaustive evaluation of all possible settings.

For each classifier, a predefined hyperparameter search space was established based on parameters that strongly influence model complexity, learning behavior, and classification performance. The explored parameters included the number of estimators, maximum tree depth, minimum sample requirements, learning rate, regularization strength, iteration limits, and smoothing values, depending on the characteristics of each model.

The optimization process was conducted over 30 evaluation iterations for each model. At each iteration, a candidate hyperparameter configuration was selected, the corresponding model was trained using the training subset, and its performance was assessed through internal validation derived from the training data. The optimization objective was to identify the configuration that produced the best classification performance, with particular attention to accuracy and F1-score as the main indicators of overall predictive quality. The search process was terminated after the predefined iteration limit was reached, and the best-performing configuration was retained for each model.

Machine Learning Models

This study employs several supervised machine learning models to evaluate the effectiveness of Bayesian-based hyperparameter optimization in network traffic classification. The selected models include Random Forest, Decision Tree, AdaBoost, Logistic Regression, Gradient Boosting, and Naïve Bayes. These models were chosen due to their widespread use and relatively different learning characteristics, allowing a more comprehensive comparison. Each model is first trained using its default configuration and then further optimized using Bayesian Optimization. The optimized hyperparameters are expected to improve model performance by better adapting to the characteristics of the dataset.

Random Forest (RF) is an ensemble learning method that constructs multiple decision trees and aggregates their predictions [20]. In this study, the optimization process results in a configuration with n_estimators set to 150, max_depth of 12, min_samples_split of 4, and min_samples_leaf of 2, while max_features is set to sqrt. This configuration indicates that increasing the number of trees while controlling their depth helps the model capture important patterns without introducing excessive complexity.

Decision Tree. Decision Tree (DT) is a rule-based model that partitions data into branches based on feature values [21], [22]. The optimized parameters include a max_depth of 10, min_samples_split of 6, and min_samples_leaf of 3, using the gini criterion. By limiting the depth and increasing the minimum number of samples required for splitting, the model becomes more stable and less prone to overfitting.

AdaBoost is an ensemble method that combines multiple weak learners to improve classification performance [23], [24]. In this study, the optimized configuration uses n_estimators of 100 and a learning_rate of 0.8. This balance allows the model to progressively improve its predictions while avoiding overly aggressive updates that could reduce generalization ability.

Logistic Regression. Logistic Regression (LR) is a linear model widely used for binary classification tasks [25]. The optimization process results in a C value of 1.5, using the lbfgs solver, with a maximum of 200 iterations (max_iter). These settings help the model achieve better convergence while maintaining an appropriate level of regularization.

Gradient Boosting is a sequential ensemble technique where each model attempts to correct the errors of the previous one [26]. The optimized parameters include n_estimators set to 120, learning_rate of 0.1,

max_depth of 4, and min_samples_split of 5. This configuration allows the model to improve prediction accuracy while keeping the learning process stable and controlled.

Naïve Bayes (NB) is a probabilistic classifier based on Bayes' theorem [27], [28]. In this study, the optimization mainly focuses on the var_smoothing parameter, which is set to 1e-9. Although simple, this adjustment helps improve numerical stability and ensures that the model can handle variations in feature distribution more effectively.

Evaluation Metrics

To evaluate the performance of the proposed models, several commonly used classification metrics are employed, including accuracy, precision, recall, and F1-score [29]–[31]. These metrics are calculated based on the confusion matrix, which summarizes the prediction results into four categories: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Accuracy measures the overall correctness of the model by comparing the number of correct predictions to the total number of instances. It is defined by equation (1). Precision reflects how many of the predicted positive instances are actually correct. This metric is important when false positives need to be minimized. It is calculated by equation (2). Recall, also known as sensitivity, measures the ability of the model to correctly identify actual positive instances. It is defined by equation (3). F1-score is the harmonic mean of precision and recall, providing a balanced evaluation when both metrics are important. It is given by equation (4). These evaluation metrics are used to provide a comprehensive assessment of model performance, particularly in distinguishing between normal and malicious network traffic. By considering multiple metrics rather than relying solely on accuracy, this study aims to obtain a more reliable understanding of how well the models perform under different conditions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

3. RESULTS AND DISCUSSIONS

Model Evaluation

The performance of each machine learning model was evaluated using accuracy, precision, recall, and F1-score on the testing dataset, which consisted of 20% of the total data and was not involved in the training process. These metrics were selected to provide a more comprehensive assessment of classification behavior, particularly because intrusion detection performance should not be interpreted only from overall accuracy. In practical security applications, precision reflects the reliability of attack predictions, while recall indicates the model's ability to detect actual attack instances. The complete evaluation results are summarized in Table 4.

Table 4. Model evaluation

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	89.2662	99.0555	77.1849	86.7632
Decision Tree	81.5514	78.7556	81.5087	80.1085
AdaBoost	88.1761	96.4245	76.9089	85.5681
Logistic Regression	74.1300	74.6331	65.5014	69.7697
Gradient Boosting	89.3920	100.0000	76.7249	86.8298
Naïve Bayes	82.7254	84.7580	75.7130	79.9806

As shown in Table 4, Gradient Boosting achieved the highest overall accuracy and F1-score, indicating the strongest general classification performance among the evaluated models. Its precision reached 100.0000, meaning that all traffic instances predicted as attacks were correctly identified. From a practical perspective, this behavior is valuable because it minimizes false alarms, which can reduce unnecessary investigation by security analysts. However, its recall of 76.7249 shows that a portion of actual attacks remained undetected. This suggests that Gradient Boosting adopts a relatively conservative decision pattern: it is highly reliable when declaring an attack, but less sensitive in capturing every malicious instance.

Random Forest produced a very similar performance profile, with accuracy and F1-score only slightly below Gradient Boosting. Its precision remained very high at 99.0555, while its recall reached 77.1849. Compared with Gradient Boosting, Random Forest provided a slightly better balance between attack detection capability and false-alarm control. This result indicates that ensemble tree-based approaches are well suited to the structure of the dataset, as they can capture non-linear relationships and interactions among traffic-related features more effectively than simpler classifiers. In intrusion detection scenarios, this balance may be beneficial when both prediction reliability and reasonable attack coverage are required.

AdaBoost also demonstrated strong performance, particularly in terms of precision. Its high precision value shows that the model was still effective in reducing false positive predictions. Nevertheless, its recall remained lower than that of Decision Tree and Random Forest, indicating that several attack instances were missed. This pattern suggests that boosting-based models in this experiment tended to prioritize confidence in positive predictions rather than maximizing sensitivity. Such behavior may be preferable in environments where excessive false alarms are costly, but it may be less ideal in systems where undetected attacks carry severe consequences.

A different pattern was observed in the Decision Tree model. Although its accuracy and F1-score were lower than those of the ensemble models, it achieved the highest recall among all classifiers. This means that Decision Tree was more sensitive in identifying actual attack traffic. However, its lower precision indicates that this sensitivity came at the cost of more false positive predictions. In practical intrusion detection systems, such a model may help reduce missed attacks, but it could also increase the workload of analysts due to a larger number of benign traffic instances being incorrectly flagged as malicious. This result illustrates the important trade-off between detection coverage and alert reliability.

Logistic Regression showed the weakest performance across all major metrics. Its lower accuracy, recall, and F1-score indicate that a linear decision boundary was not sufficient to effectively represent the complexity of the network traffic patterns in this dataset. The model's limited ability to capture non-linear relationships may explain why a larger proportion of attack traffic was not correctly recognized. This suggests that simpler linear classifiers may be less suitable when traffic behavior is influenced by multiple interacting features.

Meanwhile, Naïve Bayes achieved moderate results. Its performance was better than Logistic Regression but remained below that of the ensemble-based models. This outcome may be associated with the conditional independence assumption of Naïve Bayes, which is often difficult to satisfy in network traffic data where variables such as login behavior, packet characteristics, and access patterns may interact with one another. Even so, its relatively stable performance indicates that the model can still provide a computationally simple baseline for intrusion detection tasks.

Overall, the evaluation results show that ensemble-based models, especially Gradient Boosting and Random Forest, offered the strongest performance after Bayesian Optimization. However, the results also demonstrate that higher precision does not automatically imply superior detection capability in every operational context. Models with very high precision reduce false alarms, whereas models with higher recall are more effective at capturing a larger number of attacks. Therefore, the most appropriate model for intrusion detection should be selected according to the operational priority of the system, whether it emphasizes minimizing false positives or reducing false negatives.

Confusion Matrix Prediction

The confusion matrix analysis provides a more detailed interpretation of model behavior by showing how each classifier distributes its predictions into true positive, true negative, false positive, and false negative outcomes. This evaluation is important in intrusion detection because overall accuracy alone cannot fully describe the practical consequences of classification errors. In security systems, false positives may generate unnecessary alerts and increase analyst workload, while false negatives are more critical because actual attacks may pass through the detection system without being identified.

The prediction results of all evaluated models are illustrated in Figure 3. Among the compared classifiers, Gradient Boosting and Random Forest demonstrated the strongest overall classification behavior. Gradient Boosting produced very few, or nearly no, false positive predictions, which is consistent with its perfect precision score. This indicates that the model was highly reliable when labeling traffic as malicious. However, the presence of false negatives confirms that some attack instances were still classified as normal traffic. From an operational perspective, this means that the model is effective in reducing unnecessary alerts, but its sensitivity to all attack cases remains limited.

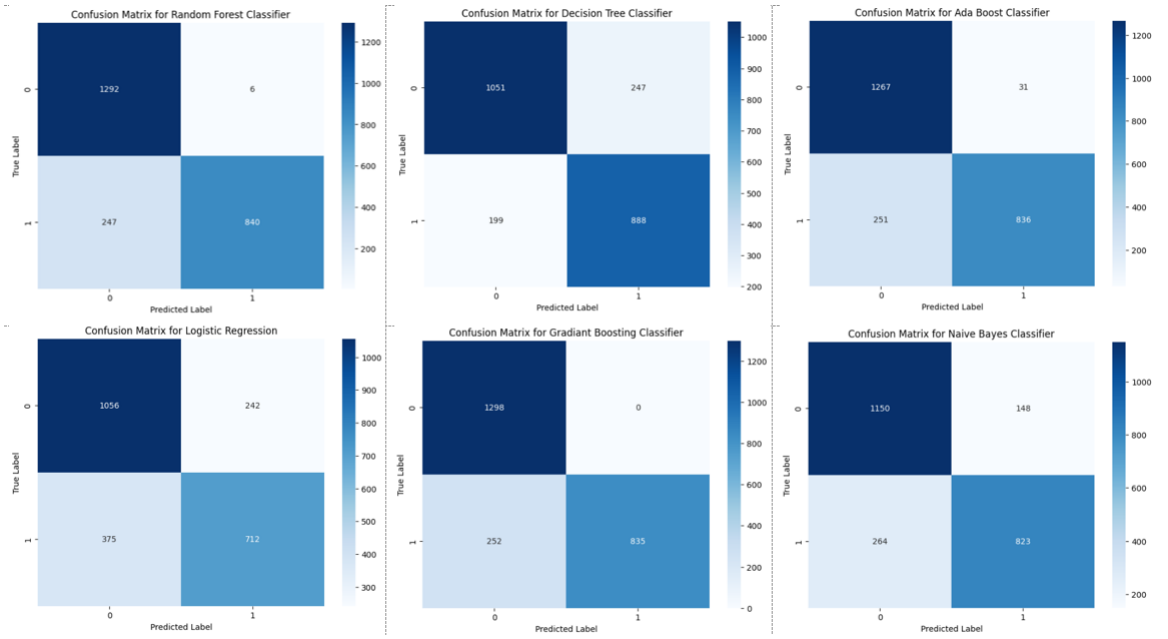

Figure 3. Confusion matrix results for each evaluated machine learning model

Random Forest showed a closely related prediction pattern, but with a slightly different trade-off. Compared with Gradient Boosting, Random Forest retained very high precision while achieving slightly better recall. This suggests that Random Forest was marginally more capable of capturing attack instances, although it introduced a small number of additional false positive predictions. Such behavior may be advantageous in practical intrusion detection environments that require a balance between reliable alerts and broader attack coverage.

The confusion matrix of AdaBoost further supports the observation that boosting-based classifiers tend to prioritize prediction confidence. The model maintained a relatively low number of false positives, yet still produced a noticeable number of false negatives. This indicates that AdaBoost was conservative in identifying malicious traffic, favoring stronger certainty before assigning an attack label. While this can reduce false alarms, it may also limit the model's usefulness in scenarios where missing attacks is considered more harmful than over-reporting suspicious traffic.

In contrast, Decision Tree displayed a more detection-oriented pattern. Its confusion matrix indicates a stronger ability to recognize attack instances, which aligns with its higher recall score. However, this came with an increase in false positive predictions compared with the ensemble-based models. This confirms that Decision Tree was more sensitive but less selective in its classification behavior. For intrusion detection applications, this characteristic may be useful when the primary goal is to minimize undetected attacks, although it may require additional filtering or analyst verification to manage excessive alert generation.

The weaker performance of Logistic Regression is also evident from its confusion matrix. The model produced a relatively high number of false negatives, indicating difficulty in identifying malicious traffic patterns. This supports the earlier interpretation that a linear classifier is less capable of capturing the complex feature interactions present in the dataset. Meanwhile, Naïve Bayes generated a more moderate prediction distribution, but still showed misclassifications in both classes. Its behavior suggests limited adaptability when feature dependencies are present, which is common in network traffic analysis.

Overall, the confusion matrix confirms that the models differ not only in numerical performance scores but also in the practical consequences of their prediction errors. Gradient Boosting and Random Forest are more suitable when reliable attack alerts and lower false positive rates are prioritized. On the other hand, Decision Tree may be considered when a higher attack detection rate is more important, even if it leads to additional false alarms. These findings reinforce the importance of selecting intrusion detection models based on operational security priorities rather than relying on a single aggregate metric.

Comparison Results without Bayesian Optimization

This section presents a comparison of model performance without applying Bayesian Optimization. The comparison is conducted to highlight how the use of default hyperparameter settings affects the classification results. Several previous studies are included as references to provide a broader perspective on baseline performance using Random Forest. It should be noted that this comparison is limited to the Random Forest model, rather than including different classifiers from previous studies. This is because many of the referenced works employ different types of models, which may have distinct learning mechanisms and characteristics. Directly comparing different classifiers could lead to an unfair evaluation, as performance differences may be influenced by model architecture rather than parameter configuration. To ensure a fair comparison, this study adopts the parameter settings reported in previous studies [11]–[14] and evaluates them using the same dataset and experimental setup.

Table 5. Comparison results without bayesian optimization (random forest classifier)

References	Accuracy	Precision	Recall	F1-Score
Study by [11]	85.4321	92.1543	74.2810	82.2517
Study by [12]	87.1098	95.8762	76.5432	85.1024
Study by [13]	86.5543	93.6678	75.4321	83.5826
Study by [14]	86.5543	93.6678	75.4321	83.5826
Our Study	89.2662	99.0555	77.1849	86.7632

As seen in Table 5, the performance of Random Forest across different studies shows a relatively consistent pattern, with accuracy values generally above 85%, indicating that the model remains robust even under default configurations. However, noticeable variations can still be observed in precision, recall, and F1-score, suggesting that differences in dataset characteristics and feature distributions play a significant role in influencing model behavior. In this context, our study achieves the highest performance across all evaluation metrics, which may indicate a better alignment between the selected features and the underlying data patterns, even before applying any optimization strategy. A closer observation reveals that Study by [13] and Study by [14] produce identical results across all evaluation metrics. This similarity strongly suggests that both studies rely on the same default hyperparameter configuration of the Random Forest model, leading to nearly identical decision boundaries and classification behavior. While such configurations provide a stable baseline, they may limit the model's ability to adapt to subtle variations in the data, particularly in distinguishing between complex or overlapping traffic patterns. This further highlight that relying solely on default settings may result in comparable but not necessarily optimal performance, reinforcing the importance of systematic hyperparameter optimization as explored in this study.

4. CONCLUSION

This study evaluated the impact of Bayesian-based hyperparameter optimization on the performance of several supervised machine learning models for network traffic classification. The results show that optimization contributes to improved model performance, particularly for ensemble-based methods such as Gradient Boosting and Random Forest, which achieved the highest accuracy and F1-score. While these models demonstrate strong precision and overall reliability, the analysis also reveals a trade-off with recall, indicating that some attack instances remain undetected. In contrast, simpler models such as Logistic Regression show lower performance, highlighting their limitations in handling complex traffic patterns. Overall, the findings suggest that hyperparameter optimization plays an important role in enhancing model effectiveness, although model selection remains dependent on specific requirements, such as prioritizing detection rate or minimizing false alarms. Future work may extend this study by comparing Bayesian Optimization with other hyperparameter tuning strategies, such as Grid Search and Random Search, to provide a broader evaluation of optimization efficiency and classification performance. In addition, repeated experiments or Stratified K-Fold Cross Validation may be considered to further strengthen the robustness and generalization assessment of the proposed approach.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Dahlan: Original draft, Conceptualization, Methodology, Software, Review, Editing, Project administration, Funding. **Dadan Saepul Ramdan:** Original draft, Conceptualization, Methodology, Software, Review, Editing, Validation, Supervision.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no conflict interest.

DATA AVAILABILITY

Data will be made available on request.

REFERENCES

[1] E. Halboosa, "A systematic review: security information for agent approaches in networks - models and methods," *Przegląd Elektrotechniczny*, vol. 1, no. 5, pp. 262–271, May 2023.

[2] W. Zhang and J. P. Lazaro, "A Survey on Network Security Traffic Analysis and Anomaly Detection Techniques," *Int. J. Emerg. Technol. Adv. Appl.*, vol. 1, no. 4, pp. 8–16, May 2024.

[3] F. Hu, S. Zhang, X. Lin, L. Wu, N. Liao, and Y. Song, "Network traffic classification model based on attention mechanism and spatiotemporal features," *EURASIP J. Inf. Secur.*, vol. 2023, no. 1, p. 6, Jul. 2023.

[4] W. Wang, S. Jian, Y. Tan, Q. Wu, and C. Huang, "Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions," *Comput. \& Secur.*, vol. 112, p. 102537, Jan. 2022.

[5] P. A. D. S. N. Wijesekara and S. Gunawardena, "A Comprehensive Survey on Knowledge-Defined Networking," *Telecom*, vol. 4, no. 3, pp. 477–596, Aug. 2023.

[6] G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges," *Soft Comput.*, vol. 25, no. 15, pp. 9731–9763, Aug. 2021.

[7] K. Huang, Q. Zhang, C. Zhou, N. Xiong, and Y. Qin, "An Efficient Intrusion Detection Approach for Visual Sensor Networks Based on Traffic Pattern Learning," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 47, no. 10, pp. 2704–2713, Oct. 2017.

[8] S. S. Tripathy and B. Behera, "Hyperparameter Tuning-Based Optimized Performance Analysis of Machine Learning Algorithms for Network Intrusion Detection," *Int. J. Netw. Secur. \& Its Appl.*, vol. 17, no. 6, pp. 1–26, Nov. 2025.

[9] M. Hasnain, N. Javaid, A. K. J. Saudagar, and N. Kumar, "An intelligent and explainable intrusion detection framework for Internet of Sensor Things using generalizable optimized active Machine Learning," *J. Netw. Comput. Appl.*, vol. 245, p. 104358, Jan. 2026.

[10] H. Liu, X. Wang, F. He, and Z. Zheng, "Automated Network Defense: A Systematic Survey and Analysis of AutoML Paradigms for Network Intrusion Detection," *Appl. Sci.*, vol. 15, no. 19, p. 10389, Sep. 2025.

[11] M. T. et al. Abdelaziz, "Enhancing Network Threat Detection with Random Forest-Based NIDS and Permutation Feature Importance," *J. Netw. Syst. Manag.*, vol. 33, no. 1, p. 2, Jan. 2025.

[12] R. A. Disha and S. Waheed, "Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique," *Cybersecurity*, vol. 5, no. 1, p. 1, Dec. 2022.

[13] A. Alharthi, M. Alaryani, and S. Kaddoura, "A comparative study of machine learning and deep learning models in binary and multiclass classification for intrusion detection systems," *Array*, vol. 26, p. 100406, Jul. 2025.

[14] R. Almuhanna and S. Dardouri, "A deep learning/machine learning approach for anomaly based network intrusion detection," *Front. Artif. Intell.*, vol. 8, Sep. 2025.

[15] H. M. R. U. et al. Rehman, "A systematic literature study of machine learning techniques based intrusion detection: datasets, models, challenges, and future directions," *J. Big Data*, vol. 12, no. 1, p. 264, Nov. 2025.

[16] D. N. K. Samudrala, "Cybersecurity Intrusion Detection Dataset." 2026.

[17] S. Sani, H. Xia, J. Milisavljevic-Syed, and K. Salonitis, "Supply Chain 4.0: A Machine Learning-Based Bayesian-Optimized LightGBM Model for Predicting Supply Chain Risk," *Machines*, vol. 11, no. 9, p. 888, Sep. 2023.

[18] R. M. Moraes, J. A. Ferreira, and L. S. Machado, "A New Bayesian Network Based on Gaussian Naive Bayes with Fuzzy Parameters for Training Assessment in Virtual Simulators," *Int. J. Fuzzy Syst.*, vol. 23, no. 3, pp. 849–861, Apr. 2021.

[19] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using Bayesian optimization," *Evol. Syst.*, vol. 12, no. 1, pp. 217–223, Mar. 2021.

[20] C. Umam, L. B. Handoko, and F. O. Isinkaye, "Performance Analysis of Support Vector Classification and Random Forest in Phishing Email Classification," *Sci. J. Informatics*, vol. 11, no. 2, pp. 367–374, May 2024.

[21] A. J. Albert, R. Murugan, and T. Sripriya, "Diagnosis of heart disease using oversampling methods and decision tree classifier in cardiology," *Res. Biomed. Eng.*, vol. 39, no. 1, pp. 99–113, Dec. 2022.

[22] A. S. Jaddoa, "Heart disease prediction system using (SMOTE technique) balanced dataset and decision tree classifier," 2023, p. 50006.

[23] Y. Ding, H. Zhu, R. Chen, and R. Li, "An Efficient AdaBoost Algorithm with the Multiple Thresholds Classification," *Appl. Sci.*, vol. 12, no. 12, p. 5872, Jun. 2022.

[24] S. Demir and E. K. Sahin, "An investigation of feature selection methods for soil liquefaction prediction based on tree-based ensemble algorithms using AdaBoost, gradient boosting, and XGBoost," *Neural Comput. Appl.*, vol. 35, no. 4, pp. 3173–3190, Feb. 2023.

[25] H. Šinkovec, G. Heinze, R. Blagus, and A. Geroldinger, "To tune or not to tune, a case study of ridge logistic regression in small or sparse datasets," *BMC Med. Res. Methodol.*, vol. 21, no. 1, p. 199, Dec. 2021.

[26] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1937–1967, Mar. 2021.

[27] P. Subarkah, W. R. Damayanti, and R. A. Permana, "Comparison of Correlated Algorithm Accuracy Naive Bayes Classifier and Naive Bayes Classifier for Classification of heart failure," *Ilk. J. Ilm.*, vol. 14, no. 2, pp. 120–125, Aug. 2022.

[28] O. Somantri, R. H. Maharrani, and S. Purwaningrum, "Coastal Sentiment Review Using Naïve Bayes with Feature Selection Genetic Algorithm," *Sci. J. Informatics*, vol. 10, no. 3, pp. 229–238, Jun. 2023.

[29] N. R. D. Cahyo and M. M. I. Al-Ghiffary, "An Image Processing Study: Image Enhancement, Image Segmentation, and Image Classification using Milkfish Freshness Images," *Int. J. Eng. Comput. Adv. Res.*, vol. 1, no. 1, pp. 11–22, 2024.

[30] M. M. I. Al-Ghiffary, N. R. D. Cahyo, E. H. Rachmawanto, C. Irawan, and N. Hendriyanto, "Adaptive deep learning based on FaceNet convolutional neural network for facial expression recognition," *J. Soft Comput.*, vol. 5, no. 3, pp. 271–280, 2024.

[31] N. R. D. Cahyo, C. A. Sari, E. H. Rachmawanto, C. Jatmoko, R. R. A. Al-Jawry, and M. A. Alkhafaji, "A Comparison of Multi Class Support Vector Machine vs Deep Convolutional Neural Network for Brain Tumor Classification," in *2023 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 2023, pp. 358–363.

.