

# Network Traffic Anomaly Detection for Server Security Using a Hybrid CNN–BiLSTM and LightGBM Optimized by Whale Optimization Algorithm

Marliana Sari<sup>1</sup>, Nanang Sadikin<sup>2</sup>, Andi Supriadi Chan<sup>3</sup>

<sup>1,3</sup>Computer Engineering and Information, Politeknik Negeri Medan, Indonesia

<sup>2</sup>Computer Engineering and Information, Sekolah Tinggi Teknologi Informasi NIIT, Indonesia

## Article Info

### Article history:

Received Feb 21, 2026

Revised Apr 08, 2026

Accepted Apr 18, 2026

### Keywords:

IoT

DDoS

CNN–BiLSTM

Optimization

hyperparameters

## ABSTRACT

The rapid growth of server-based digital services (e.g., web services, databases, cloud platforms, and IoT gateways) expands the attack surface, while threats such as DDoS, brute-force attempts, port scanning, web attacks, bot activity, and infiltration are increasingly difficult to distinguish from legitimate traffic. This study proposes a hybrid anomaly detection pipeline that combines deep learning for representation learning with gradient-boosted decision trees for multiclass classification on structured traffic features. The pipeline applies Min–Max scaling to harmonize feature ranges and SMOTE to mitigate class imbalance, then uses a CNN–BiLSTM network to capture local patterns and temporal dependencies, followed by LightGBM as the final classifier. To enhance LightGBM performance, its hyperparameters are optimized using the Whale Optimization Algorithm (WOA). We evaluate the approach using accuracy, precision, recall, and F1-score in a multiclass setting. Experimental results show that the CNN–BiLSTM + LightGBM baseline achieves 0.92 accuracy with strong performance on most attack categories, while rare and ambiguous classes (notably Infiltration and Web Attack) remain challenging. With WOA-based optimization, overall accuracy increases to 0.93 and the macro-average F1-score improves from 0.88 to 0.89, with clearer gains for several time-dependent attack classes. While the proposed framework is designed with practical deployment in mind by leveraging a tree-based classifier on learned features, this study does not report system-level efficiency measurements (e.g., training time, inference latency, or memory footprint); these will be addressed in future work to substantiate computational feasibility in real server environments.

*This is an open access article under the [CC BY-SA](#) license.*



## Corresponding Author:

Marliana Sari,

Computer Engineering and Information

Politeknik Negeri Medan,

Almamater Road No.1, Medan, Indonesia.

Email: marlianasari@polmed.ac.id

## 1. INTRODUCTION

The rapid development of server based digital services such as web services, databases, cloud platforms, IoT gateways, and governmental or corporate systems has led to increased network traffic and operational complexity while simultaneously expanding the attack surface in network security [1], [2], [3]. Malicious activities such as Distributed Denial of Service attacks, brute force attempts, port scanning, web attacks, bot activity, and infiltration are becoming increasingly difficult to distinguish from normal traffic because attack patterns continuously evolve, data volumes are extremely large, and attack techniques can mimic legitimate user behavior [4], [5]. This condition creates an urgent need for anomaly detection approaches capable of accurately and

rapidly detecting new attacks, including previously unseen patterns, within server environments [6].

In practice, rule based or signature based Intrusion Detection Systems often face limitations when dealing with new attacks and diverse modern traffic. Therefore, machine learning and deep learning approaches have become widely studied solutions to enhance intrusion detection capabilities. A study conducted in a government environment using Random Forest with a combination of CIC ToN IoT whitelist and HoneyNet BSSN blacklist datasets demonstrated good performance with accuracy above ninety percent, precision ninety one percent, recall ninety percent, and F1 score ninety percent. However, classes with fewer samples tended to exhibit lower performance, indicating the strong impact of class imbalance in network attack detection [7].

Another study comparing several algorithms on the NSL KDD dataset reported that boosting methods achieved superior performance with an accuracy of ninety nine point thirty six percent and were more reliable in detecting minority classes, although they required longer training time. These findings confirm that ensemble methods can improve sensitivity to rare attacks but still involve trade offs in computational efficiency [8]. Furthermore, approaches that incorporate temporal representations have also proven effective. A Double Layer GRU model with feature fusion was reported to improve accuracy and reduce the false positive rate, achieving accuracy up to ninety eight point sixty percent, making it suitable for sequential network traffic data [9].

Research in network security monitoring also shows that deeper traffic analysis integration is becoming a growing trend. A study combining Deep Packet Inspection with an Intrusion Detection System for detecting DDoS attacks in large scale networks reported an increase in true positive rate along with a significant reduction in false positives compared to conventional IDS. This improvement occurs because DPI can analyze traffic components in more detail, including headers and payloads, allowing attack patterns that resemble normal traffic to be identified more accurately [10]. These findings reinforce that modern intrusion detection requires approaches capable of learning complex attack patterns, both in terms of feature relationships and sequential traffic dynamics. At the same time, such approaches must incorporate strategies for handling class imbalance to prevent bias toward dominant normal traffic, as well as optimization mechanisms to remain efficient and feasible for server environments that demand fast and stable detection processes.

Based on these gaps, this study proposes a hybrid approach to improve anomaly detection performance in server environments within the context of network security by combining deep learning as a representation learner and machine learning as an efficient classifier for structured data. This approach is designed to address key challenges commonly found in network security data, particularly class imbalance where normal traffic is far more dominant than attack traffic, the diversity and complexity of attack patterns that often appear as combinations of local characteristics and sequential patterns, and the need for models that are not only accurate but also efficient and practical for deployment in server environments requiring fast and stable detection.

In general, the research workflow follows integrated data processing and modeling stages. Initially, all numerical features in traffic data or server logs are normalized using Min Max Scaling so that values fall within a uniform range between zero and one [11]. This normalization is necessary because IDS or traffic data typically contain features with highly varied scales such as packet counts, connection duration, byte values, and ratios. Extreme scale differences can make models more sensitive to certain features and reduce training stability, particularly in deep learning components. With Min Max normalization, the learning process becomes more stable, convergence is faster, and feature representations become more consistent for subsequent stages.

After normalization, the training data is processed to address class imbalance commonly found in IDS datasets. In such conditions, normal traffic such as BENIGN often dominates, while certain attack classes such as infiltration, SQL injection, or heartbleed contain relatively few samples. To reduce bias toward the majority class, this study applies the Synthetic Minority Oversampling Technique. SMOTE generates synthetic samples for minority classes based on similarity with their nearest neighbors, resulting in a more representative class distribution [12]. This balancing process ensures that the model does not only pursue global accuracy but is also capable of recognizing

minority attack classes that are practically critical in network security, while reducing the risk of false negatives where attacks are incorrectly predicted as normal traffic.

The next stage involves representation learning using CNN and BiLSTM architectures to capture complex patterns in traffic data or server logs. Convolutional Neural Networks are used to extract local patterns and correlations among features that often represent statistical characteristics of attacks, such as feature combinations indicating scanning, flooding, or abnormal connection behavior [13]. CNN effectively forms high level representations from multivariate input. These representations are then processed by Bidirectional Long Short Term Memory to capture temporal dependencies and sequential dynamics in network traffic. Many attacks occur as sequences of activities, making sequential modeling important. BiLSTM learns context from both forward and backward directions, enabling it to represent traffic behavior changes over time more effectively than unidirectional models [14].

Feature vectors extracted by CNN and BiLSTM are then used as input for classification using LightGBM. LightGBM is selected because it provides high performance on structured data, is computationally efficient, and can handle nonlinear relationships through gradient boosting mechanisms [15]. In the context of multiclass IDS, LightGBM can utilize deep learning features to separate various attack classes more sharply while remaining lighter than large fully connected neural network layers.

To ensure that LightGBM operates with optimal configuration, this study applies the Whale Optimization Algorithm for hyperparameter optimization. WOA searches for the best combination of parameters such as learning rate, number of estimators or trees, tree depth, number of leaves, subsampling, and regularization parameters. These parameters significantly influence LightGBM performance, while manual tuning or grid search often requires higher computational cost. Through adaptive metaheuristic search, WOA is expected to find configurations that improve accuracy while maintaining model generalization, thereby reducing the risk of overfitting in specific classes [16].

Finally, model performance is comprehensively evaluated using metrics relevant to network security including accuracy, precision, recall, and F1 score, along with confusion matrix analysis for each attack class. This evaluation is important because in intrusion detection systems, high accuracy alone is insufficient. The model must minimize false positives that cause excessive alerts in normal traffic and false negatives which are far more dangerous because attacks may pass undetected. Through this series of stages, the study aims to produce an anomaly detection system that is more accurate, more robust to class imbalance, and more adaptive to attack variations within server and network ecosystems.

## 2. RESEARCH METHOD

The proposed architecture is a hybrid approach that integrates deep learning and machine learning techniques to improve anomaly detection performance in server environments within the context of network security. The process begins with a data processing stage to ensure feature quality and consistency, followed by handling class imbalance to obtain a more representative data distribution. The model then performs feature extraction using a combination of Convolutional Neural Network and Bidirectional Long Short Term Memory to capture local patterns and temporal dependencies in network traffic. The resulting feature representations are classified using Light Gradient Boosting Machine optimized with the Whale Optimization Algorithm to obtain the best parameter configuration. The final stage evaluates model performance using classification metrics to assess detection capability across various types of attacks.

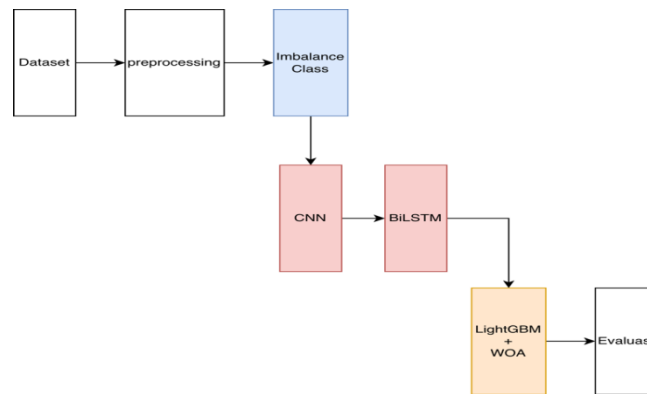


Figure 1. Methodology flow

## 2.1. Preprocessing

The preprocessing stage aims to prepare the data so that it has a uniform scale, allowing the model training process to become more stable [17]. Normalization is performed using the Min–Max Scaling method, which transforms each feature value into a range between zero and one. This transformation helps reduce the dominance of features with larger scales compared to others and accelerates the convergence process during model training [18]. The Min Max normalization equation is defined as follows:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Where  $x$  is the original feature value,  $x_{min}$  is the minimum value of the feature, and  $x_{max}$  is the maximum value of the feature. In this study, Min–Max normalization is applied to the numerical attributes of the network traffic dataset prior to model training to ensure that all input features contribute proportionally during optimization.

The network traffic records are taken from the CIC-IDS2017 dataset provided by the Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB), Canada, which is publicly available through the CIC dataset repository. From this source, 11,000 log entries are selected as the experimental sample for model development and evaluation. This dataset contains labeled benign and attack traffic, making it suitable for supervised anomaly detection experiments, and the selected subset is used consistently throughout the preprocessing, training, and testing pipeline.

## 2.2. Handling Class Imbalance Using SMOTE

Intrusion detection datasets generally have imbalanced class distributions where normal traffic is much more dominant than attack traffic. This condition can cause the model to be biased toward the majority class. To address this issue, the Synthetic Minority Oversampling Technique is used to generate synthetic samples for minority classes by interpolating data points based on their nearest neighbors [19]. This technique improves the representation of attack classes so that the model can learn patterns more evenly. The synthetic sample generation equation is defined as follows:

$$x_{new} = x_i + \lambda(x_{nn} - x_i) \quad (2)$$

Where  $x_i$  is a minority sample,  $x_{nn}$  is its nearest neighbor, and  $\lambda$  is a random value between zero and one.

Figure 2 is the initial data in this study.

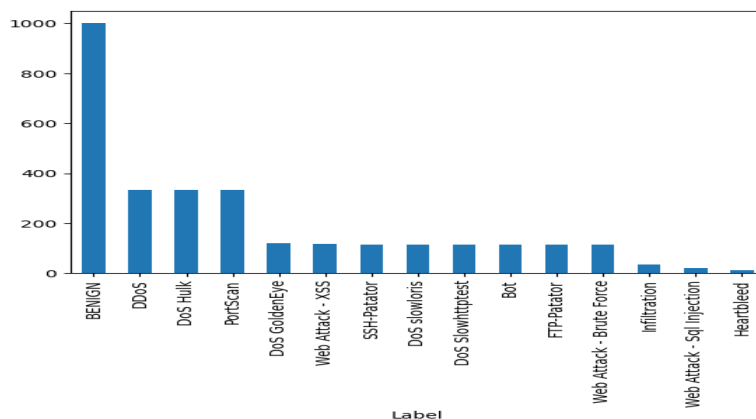


Figure 2. Class Distribution Before Data Balancing (Imbalanced Dataset)

Figure 2 shows the class distribution before data balancing, where a significant class imbalance can be observed. The BENIGN class contains a much larger number of samples than the other classes, while several classes such as Heartbleed, SQL Injection, and Infiltration have very limited samples. This imbalance may cause the model to become biased toward the majority class. Next is the result of class distribution after data balancing.

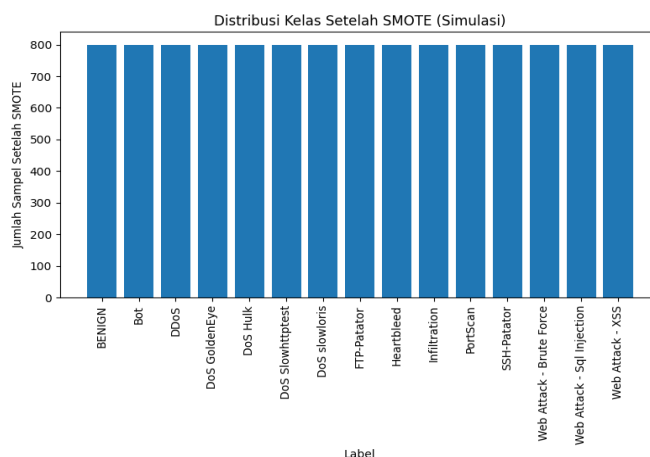


Figure 3. Class Distribution of the Dataset After Data Balancing Using SMOTE

Figure 3 shows the class distribution after applying SMOTE (Synthetic Minority Over-sampling Technique). It can be seen that the number of samples in each class becomes relatively balanced, with nearly equal distribution across all categories. This indicates that SMOTE successfully addresses the class imbalance problem, which is expected to improve the model's ability to recognize minority classes and produce fairer and more accurate classification performance.

### 2.3. Feature Extraction Using CNN

Before the feature learning process is performed, the dataset is divided into training and testing sets using an 80:20 split ratio. This partitioning strategy ensures that the majority of the data is used to train the model while reserving a separate subset for evaluating its generalization capability. Specifically, 80% of the dataset is allocated for model training, allowing the model to learn representative patterns from network traffic data, while the remaining 20% is used for testing to objectively measure model performance on unseen data. This approach helps prevent overfitting and provides a reliable assessment of the model's ability to detect anomalies in real-world scenarios.

Convolutional Neural Network is used to learn local patterns from network traffic data by performing convolution operations on input features. CNN is capable of extracting high level

representations that reflect complex relationships among features, helping to identify characteristics of attack activities [20]. The convolution operation can be expressed as:

$$h_j = f(\sum_i x_i * w_{ij} + b_j) \quad (3)$$

Where

- $h_j$ : Represents the output of neuron  $j$  after applying an activation function.
- $\sum_i x_i * w_{ij}$ : Computes the linear combination of input features  $x_i$  and their corresponding weights  $w_{ij}$ .
- $b_j$ : Adds an additional adjustable parameter to shift the output.
- Activation Function ( $f(\cdot)$ )
- Purpose: Transforms input features into meaningful representations for further processing in the network.

#### 2.4. Temporal Learning Using BiLSTM

After local features are obtained, the model uses Bidirectional Long Short-Term Memory to learn temporal dependencies from network traffic. BiLSTM processes sequence data in both forward and backward directions, enabling more comprehensive context capture. This is important because attack patterns often appear as sequences of related activities [21]. The basic LSTM equations are defined as follows:

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad (4)$$

Where

- forget gate ( $f_t$ ): The forget gate determines which information from the previous cell state ( $c_{t-1}$ ) should be retained or discarded. It uses a sigmoid function to produce values between 0 and 1. Values close to 0 indicate that the information is forgotten, while values close to 1 indicate that it is preserved.
- input gate ( $i_t$ ): The input gate controls how much new information from the current input ( $x_t$ ) will be added to the cell state. It uses a sigmoid function to select relevant information that should be updated in the memory.
- output gate ( $o_t$ ): The output gate determines which part of the cell state will be exposed as the hidden state ( $h_t$ ). It regulates how much information is passed to the next time step or to the next layer.
- cell state ( $c_t$ ): The cell state is updated by combining two components:
  - (1) the retained previous information through the forget gate ( $f_t \odot c_{t-1}$ ), and
  - (2) the new candidate information scaled by the input gate ( $i_t \odot \tanh(\dots)$ ).
 This mechanism allows LSTM to maintain long-term dependencies effectively.
- hidden state ( $h_t$ ): The hidden state represents the final output at time step  $t$ . It is computed by applying a tanh function to the updated cell state and scaling it with the output gate. This ensures controlled information flow to subsequent steps.

#### 2.5. Classification Using LightGBM

The extracted features are then classified using LightGBM, a gradient boosting algorithm based on decision trees. LightGBM can efficiently model nonlinear relationships and provides high performance on structured data [22]. The model works by iteratively building decision trees to minimize the loss function. The LightGBM objective function can be expressed as:

$$Obj = \sum_i L(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \quad (5)$$

Where

- $Obj$ : Represents the function to be minimized during training, combining prediction error and model complexity.
- $\sum_i L(y_i, \hat{y}_i)$ : Measures the difference between true labels and predicted values; lower values indicate better model accuracy.
- $\sum_k \Omega(f_k)$ : Penalizes model complexity to prevent overfitting and improve generalization.
- Overall Purpose: Balances accuracy and complexity to produce a model that performs well on both training and unseen data.

## 2.6. Hyperparameter Optimizaation Using Whale Optimization Algorithm

To improve LightGBM performance, the Whale Optimization Algorithm is used as a hyperparameter optimization method. WOA mimics whale hunting behavior through global and local search mechanisms to find optimal solutions within the parameter space. This algorithm helps identify the best parameter combination that maximizes classification performance [23]. The WOA position update equation is defined as:

$$\begin{aligned} D &= |C \cdot X^* - X| \\ X(t+1) &= X^* - A \cdot D \end{aligned} \quad (6)$$

Where

- $D = |C \cdot X^* - X|$ : Represents the distance between the current solution ( $X$ ) and the best solution found so far ( $X^*$ ) scaled by coefficient  $C$ .
- $X(t+1) = X^* - A \cdot D$ : Updates the position of the current solution by moving it toward the best solution, controlled by coefficient  $A$ .
- Purpose: These equations guide the search process toward the optimal solution by adjusting positions based on the distance from the current best candidate.

## 2.7. Model Evaluation

The evaluation stage aims to measure the model's ability to accurately detect anomalies. Evaluation uses classification metrics including accuracy, precision, recall, and F1 score along with confusion matrix analysis to observe performance across attack classes. The use of multiple metrics is necessary because in intrusion detection systems, balancing detection capability and minimizing misclassification is crucial [24]. The evaluation metric equations are defined as follows:

$$\begin{aligned} Accuracy &= \frac{TP+TN}{TP+TN+FP+FN} \\ Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F1 &= \frac{2 \times Precision \times Recall}{Precision + Recall} \end{aligned} \quad (7)$$

- Accuracy is used to measure the overall accuracy of a model by comparing the number of correct predictions to the total data set. This value indicates how well the model classifies all existing classes.
- Precision indicates the accuracy of the positive predictions generated by the model. This metric measures the proportion of true positive data out of all data predicted as positive by the model.
- Recall measures the model's ability to identify all true positive data. A high recall value indicates that the model is able to detect a large proportion of positive data in the dataset.
- F1-score is the harmonic mean between precision and recall. This metric is used to provide a more balanced evaluation, especially in imbalanced datasets, because it considers both prediction accuracy and the model's ability to identify positive data.

### 3. RESULTS AND DISCUSSION

Class distribution is a crucial factor in anomaly detection research because it directly influences the model's ability to learn the characteristics of each network traffic category. In this study, the dataset was processed using the Synthetic Minority Oversampling Technique to produce a more balanced class distribution, allowing the model learning process to proceed more effectively. The distribution results show that each class contains a relatively similar number of samples, with approximately eight hundred instances per category. This condition provides a more uniform representation of various attack types as well as normal traffic, enabling the model to have equal opportunities to learn patterns from each class [25].

A balanced distribution is particularly important in the context of network security because anomaly detection models are not only required to achieve high accuracy but also to effectively recognize different types of attacks with diverse characteristics. With a relatively equal number of samples across classes, the learning process becomes more stable and the risk of bias toward certain classes can be minimized. This allows the model to construct more optimal decision boundaries for distinguishing between normal traffic and attack traffic. In addition, the use of SMOTE helps improve the representation of pattern variations within attack classes, allowing the model to capture attack characteristics more comprehensively. A more balanced data representation contributes positively to the feature extraction process during deep learning since the model can learn from a broader pattern distribution. This serves as an important foundation before training the CNN and BiLSTM models.

Overall, the data distribution after applying SMOTE indicates that the dataset has achieved a more balanced class composition, thereby supporting a more effective model learning process. This condition is expected to enhance the model's ability to detect various types of anomalies in server environments and reduce potential misclassification errors in subsequent modeling stages. Following the data balancing process, feature normalization is performed using the Min Max Scaling method to ensure that all features fall within a uniform value range. This normalization aims to reduce scale differences between features that may affect the stability of the model learning process. The normalization results show that all feature values have been successfully transformed into the range between zero and one, ensuring that no feature dominates others due to scale differences. This condition is especially important for deep learning models because large scale differences can slow convergence and potentially reduce model performance.

The normalization process also improves numerical stability during model training and accelerates learning because the distribution of feature values becomes more consistent. With uniform scaling, the model can learn relationships among features more effectively without being influenced by magnitude differences. Furthermore, Min Max normalization ensures that each feature contributes more evenly to the pattern extraction process in the CNN stage and the sequence learning process in BiLSTM. Overall, the Min Max normalization results indicate that the data are in an optimal condition for the modeling stage, which is expected to improve learning performance and help generate more stable and accurate feature representations in subsequent anomaly detection processes.

#### 3.1. Result

To ensure that the model training process runs optimally, this study defines the initial parameter configuration for each model component, including the CNN architecture, BiLSTM, LightGBM, and the optimization parameters of the Whale Optimization Algorithm. The determination of these initial parameters aims to provide a stable starting point for the learning process while allowing sufficient exploration space for the optimization mechanism so that the model can discover the best parameter configuration during training.

The parameters in the CNN are focused on the number of filters, kernel size, activation function, and pooling method to ensure that the feature extraction process can effectively capture local patterns and generate informative feature representations. In the BiLSTM component, the number of units and dropout value are set to maintain a balance between the model's ability to capture temporal dependencies and reduce the risk of overfitting during training.

Furthermore, LightGBM parameters such as learning rate, number of estimators, tree depth, and number of leaves are defined to control model complexity and ensure that the learning process remains efficient and stable. These parameters play an important role in shaping the optimal decision boundary during classification. Meanwhile, in the optimization stage, the Whale Optimization Algorithm uses population size and number of iterations to control the search process for the best solution within the parameter space. This parameter combination is expected to provide a balance between model performance and computational efficiency so that the training process can produce a model with good generalization capability.

The following table shows the initial parameter configuration used for each model component in this study.

Table 1. Parameter Model

| Component | Parameter            | Initial Value |
|-----------|----------------------|---------------|
| CNN       | Number of Filters    | 32            |
| CNN       | Kernel Size          | 3             |
| CNN       | Activation           | Relu          |
| CNN       | Pooling              | Max Pooling   |
| BiLSTM    | Units                | 64            |
| BiLSTM    | Dropout              | 0.2           |
| LightGBM  | Learning Rate        | 0.01          |
| LightGBM  | Number of Estimators | 100           |
| LightGBM  | Max Depth            | 10            |
| LightGBM  | Num Leaves           | 31            |
| WOA       | Population Size      | 30            |
| WOA       | Iterations           | 50            |

Based on the parameter configuration presented in Table 1, the input data are first represented as sequential feature vectors, where each sample is arranged as an ordered feature matrix so that the convolution operation can be applied along the feature sequence. In this representation, each row corresponds to a feature vector at a specific position in the sequence, allowing the CNN component to scan neighboring vectors using one-dimensional convolution. With 32 filters and a kernel size of 3, the CNN is designed to capture local interactions among adjacent feature elements, enabling the model to learn short-range patterns without significantly increasing computational complexity. The ReLU activation function is selected because it improves model nonlinearity and accelerates convergence during training [26]. Max Pooling is then applied to reduce feature dimensionality while preserving the most informative activations, resulting in more compact and robust feature representations [27].

The output of the CNN is subsequently forwarded to the BiLSTM layer, where the extracted local features are processed sequentially in both forward and backward directions. This arrangement allows the BiLSTM to capture temporal dependencies by modeling how local feature patterns evolve across the sequence. In the BiLSTM component, 64 units are used to provide sufficient memory capacity for learning sequential dependencies in network traffic data. A dropout value of 0.2 is applied as a regularization mechanism to reduce overfitting by randomly disabling some neurons during training, allowing the model to learn more general patterns [28].

The LightGBM parameters indicate a learning rate of 0.01 to ensure gradual and stable learning. A total of 100 estimators are used to build a sufficiently strong ensemble model capable of learning complex relationships among features. A tree depth of 10 and 31 leaves are used to control model complexity so that nonlinear patterns can be captured without causing overfitting. In the optimization stage, the Whale Optimization Algorithm uses a population size of 30 and 50 iterations to provide sufficient search space for finding the best parameter combination. This configuration allows the optimization process to run effectively without excessively increasing computational cost.

Overall, the parameter configuration demonstrates a balance between model capacity in learning complex patterns and computational efficiency, thereby supporting optimal anomaly detection model training. To evaluate model performance in classifying various types of network traffic, testing is conducted using CNN and BiLSTM as feature extractors combined with LightGBM as the classifier. Evaluation is performed using classification metrics including

precision, recall, and F1-score for each attack class, as well as overall accuracy. This analysis aims to assess the model's ability to recognize patterns in each traffic category and identify classes that still exhibit relatively high classification errors. The model performance evaluation results are presented in Table 2 in the form of a classification report.

Table 2. Classification Report CNN-BiLSTM+LightGBM

| label                      | precision        | recall        | f1-score        | support        |
|----------------------------|------------------|---------------|-----------------|----------------|
| BENIGN                     | 0.97             | 0.91          | 0.94            | 200            |
| Bot                        | 0.71             | 0.96          | 0.81            | 23             |
| DDoS                       | 0.99             | 1.00          | 0.99            | 67             |
| DoS GoldenEye              | 0.89             | 1.00          | 0.94            | 24             |
| DoS Hulk                   | 1.00             | 0.99          | 0.99            | 67             |
| DoS Slowhttptest           | 0.96             | 0.96          | 0.96            | 23             |
| DoS Slowloris              | 0.96             | 0.96          | 0.96            | 23             |
| FTP-Patator                | 1.00             | 1.00          | 1.00            | 23             |
| Heartbleed                 | 1.00             | 1.00          | 1.00            | 2              |
| Infiltration               | 0.62             | 0.71          | 0.67            | 7              |
| PortScan                   | 0.94             | 0.97          | 0.96            | 67             |
| SSH-Patator                | 1.00             | 1.00          | 1.00            | 23             |
| Web Attack - Brute Force   | 0.61             | 0.61          | 0.61            | 23             |
| Web Attack - Sql Injection | 1.00             | 0.75          | 0.86            | 4              |
| Web Attack - XSS           | 0.58             | 0.58          | 0.58            | 24             |
|                            | <b>precision</b> | <b>recall</b> | <b>f1-score</b> | <b>Support</b> |
| accuracy                   |                  |               | 0.92            | 600            |
| macro avg                  | 0.88             | 0.89          | 0.88            | 600            |
| Weighted avg               | 0.93             | 0.92          | 0.92            | 600            |

Based on the classification report in table 2, the CNN BiLSTM LightGBM model demonstrates good classification performance with an accuracy of 0.92. The macro average F1 score of 0.88 indicates that the model can classify multiple classes with relatively balanced performance, while the weighted average F1 score of 0.92 indicates stable performance when considering the proportion of data in each class.

A more detailed observation shows that most attack classes have very high precision and recall values. The DDoS class has a precision of 0.99 and recall of 1.00 with an F1 score of 0.99, indicating that the model is highly effective in detecting this type of attack. The DoS Hulk class also shows excellent performance with precision of 1.00 and recall of 0.99, resulting in an F1 score of 0.99, suggesting that traffic patterns in this attack class have clear characteristics that can be well learned by the model.

The FTP Patator and SSH Patator classes show perfect precision and recall values of 1.00, resulting in an F1 score of 1.00, indicating that the model can recognize brute force attack patterns very well due to their distinct characteristics compared to normal traffic. In addition, the PortScan class also shows high performance with an F1 score of 0.96, indicating that the model can effectively distinguish network scanning activities.

However, several classes show relatively lower performance compared to others. The Infiltration class has an F1 score of 0.67, indicating that the model still has difficulty recognizing this attack pattern, possibly due to traffic characteristics that are similar to normal traffic. The Web Attack Brute Force class shows an F1 score of 0.61, indicating significant misclassification, while the Web Attack XSS class has an F1 score of 0.58, suggesting that the model has not fully captured the attack pattern optimally.

The Web Attack SQL Injection class shows a precision of 1.00 but recall of 0.75, resulting in an F1 score of 0.86. This indicates that predictions for this class are highly accurate when detected, but some instances remain undetected, suggesting that SQL injection patterns have complex variations that cannot always be consistently recognized.

Overall, the evaluation results indicate that the CNN BiLSTM LightGBM model performs very well in detecting most types of network attacks, particularly those with clear traffic patterns.

However, performance on several attack classes with more complex characteristics still needs improvement, highlighting the importance of parameter optimization in the next stage.

After obtaining baseline performance in Figure 2, the next stage applies hyperparameter optimization to the LightGBM component using the Whale Optimization Algorithm. This stage aims to refine parameter configuration so that class separation becomes more optimal, especially for attack classes that previously showed relatively low precision, recall, and F1 score. Evaluation continues to use precision, recall, F1 score, and overall accuracy to observe performance changes after optimization. The classification report of the optimized CNN BiLSTM LightGBM model is presented in table 3.

Table 3. Classification Report CNN-BiLSTM+LightGBM+WOA

| label                      | precision        | recall        | f1-score        | support        |
|----------------------------|------------------|---------------|-----------------|----------------|
| BENIGN                     | 0.97             | 0.96          | 0.96            | 200            |
| Bot                        | 0.71             | 0.96          | 0.81            | 23             |
| DDoS                       | 1.00             | 1.00          | 1.00            | 67             |
| DoS GoldenEye              | 1.00             | 1.00          | 1.00            | 24             |
| DoS Hulk                   | 0.99             | 1.00          | 0.99            | 67             |
| DoS Slowhttptest           | 1.00             | 0.96          | 0.98            | 23             |
| DoS Slowloris              | 0.96             | 1.00          | 0.98            | 23             |
| FTP-Patator                | 1.00             | 1.00          | 1.00            | 23             |
| Heartbleed                 | 1.00             | 1.00          | 1.00            | 2              |
| Infiltration               | 0.56             | 0.71          | 0.62            | 7              |
| PortScan                   | 0.96             | 0.99          | 0.97            | 67             |
| SSH-Patator                | 1.00             | 1.00          | 1.00            | 23             |
| Web Attack - Brute Force   | 0.59             | 0.57          | 0.58            | 23             |
| Web Attack - Sql Injection | 1.00             | 0.75          | 0.86            | 4              |
| Web Attack - XSS           | 0.56             | 0.58          | 0.57            | 24             |
|                            | <b>precision</b> | <b>recall</b> | <b>f1-score</b> | <b>support</b> |
| accuracy                   |                  |               | 0.93            | 600            |
| macro avg                  | 0.89             | 0.89          | 0.89            | 600            |
| weighted avg               | 0.93             | 0.93          | 0.93            | 600            |

Based on the classification report in Figure 3, the optimized model shows improved performance with an accuracy of 0.93. The macro average F1 score of 0.89 indicates better performance balance across classes compared to before optimization, while the weighted average F1 score of 0.93 indicates stable overall performance. Compared to the pre optimization results, several classes show significant improvements. The DDoS and DoS GoldenEye classes achieve perfect precision and recall values of 1.00 with an F1 score of 1.00, indicating improved recognition of strong traffic patterns. The DoS Slowloris and DoS Slowhttptest classes show recall improvements to 1.00 with F1 scores of 0.98, indicating increased sensitivity to time-based attack patterns.

The PortScan class shows improved performance with an F1 score of 0.97, indicating improved differentiation of scanning activities. FTP Patator, SSH Patator, and Heartbleed remain at perfect performance with F1 scores of 1.00, indicating that optimization does not degrade performance on already strong classes. However, some classes still show relatively low performance. The Infiltration class has an F1 score of 0.62, indicating persistent difficulty distinguishing this attack from normal traffic. The Web Attack Brute Force and Web Attack XSS classes have F1 scores of 0.58 and 0.57 respectively, suggesting that their traffic characteristics remain challenging to separate.

Overall, these results indicate that the Whale Optimization Algorithm successfully improves model performance, especially for classes with complex patterns, while enhancing performance balance across classes. Optimization helps the model find more suitable parameter configurations, improving generalization capability and reducing classification errors.

Comparing Figures 2 and 3 shows that applying the Whale Optimization Algorithm improves overall model performance. Accuracy increases from 0.92 to 0.93, while macro average F1 score increases from 0.88 to 0.89, indicating improved performance balance. Several classes such as DoS GoldenEye, DoS Slowloris, and DoS Slowhttptest show clear improvements, while classes such as Infiltration and Web Attack remain challenging due to their complex and similar traffic patterns. Overall, the comparison confirms that hyperparameter optimization plays an important role in improving generalization capability in hybrid deep learning and machine learning based anomaly detection systems.

### 3.2. Discussion

The results indicate that the hybrid approach combining deep learning-based feature extraction with gradient boosting classification provides strong performance in detecting various types of anomalies in server environments. In this study, each network traffic record is represented as a structured numerical feature vector, where every feature corresponds to a statistical or behavioral attribute extracted from the traffic flow. These feature vectors are arranged as one-dimensional feature sequences, allowing the CNN layer to perform 1D convolution across neighboring feature positions. Through this arrangement, the convolution kernels slide along the feature vector and capture local relationships among adjacent traffic attributes, such as packet statistics, flow duration, and protocol-related metrics. The integration of CNN and BiLSTM therefore enables the model to capture both local feature interactions and sequential dependencies in the traffic representation more comprehensively, resulting in more informative feature representations for the classification process. This is reflected in the high precision and recall values observed across most attack classes, particularly in categories with clear traffic patterns such as DDoS, DoS Hulk, FTP Patator, and SSH Patator.

The application of data balancing using SMOTE plays an important role in improving the model's ability to learn patterns from minority classes. A more balanced data distribution allows the model to construct more representative decision boundaries, thereby reducing bias toward the majority class. This condition is especially important in network security contexts because the ability to detect rare attacks has significant practical value. The balanced class distribution also contributes to training stability during the deep learning stage, as the model can learn a wider range of pattern representations from each traffic category.

The normalization process using Min–Max Scaling is shown to improve numerical stability during model training. With a uniform feature value range, convergence becomes faster and the model can learn relationships among features more effectively without being influenced by extreme scale differences. This supports the feature extraction process in CNN and sequence learning in BiLSTM, resulting in more consistent and stable representations.

Evaluation results from the baseline model indicate that the combination of CNN, BiLSTM, and LightGBM achieves high accuracy with balanced classification performance across most classes. However, some classes such as Infiltration and Web Attack show relatively lower F1 scores, suggesting that traffic patterns in these categories are either similar to normal traffic or possess more complex characteristics. This indicates that although the feature representation is sufficiently informative, the model parameter configuration can still be improved to achieve more optimal performance.

The application of the Whale Optimization Algorithm for hyperparameter optimization demonstrates an overall performance improvement. The increase in accuracy and macro-average F1 score indicates that parameter optimization helps the model find configurations that better match the characteristics of the data. WOA performs both global and local search processes, allowing the model to obtain parameter combinations that improve class separability, particularly for time-based attacks such as DoS Slowloris and DoS Slowhttptest. The increased sensitivity in these classes indicates that optimization successfully enhances the model's ability to capture more complex temporal patterns.

Nevertheless, some classes such as Infiltration and Web Attack still exhibit relatively lower performance despite optimization. This may be due to traffic characteristics that closely resemble normal traffic, making decision boundaries more difficult to distinguish. In addition, variations in

attack patterns within these classes may not be fully represented by the available features, causing the model to face challenges in achieving optimal generalization. This condition suggests that future research could consider incorporating additional feature extraction techniques or attention-based approaches to improve sensitivity toward more subtle attack patterns.

Overall, the findings demonstrate that the proposed hybrid approach effectively improves anomaly detection performance in server environments with stable and accurate results. The integration of deep learning for feature extraction, LightGBM as the classifier, and hyperparameter optimization using the Whale Optimization Algorithm provides an effective combination for handling the complexity of network security data. This approach shows strong potential for application in modern intrusion detection systems that require the ability to analyze complex patterns while maintaining high computational efficiency.

#### 4. CONCLUSION

This study proposed a hybrid framework for anomaly detection in server environments within the context of network security by integrating Min–Max normalization, SMOTE-based class balancing, CNN–BiLSTM feature extraction, and LightGBM classification optimized using the Whale Optimization Algorithm. Experimental results indicate that the CNN–BiLSTM + LightGBM baseline provides strong overall performance with 0.92 accuracy and high precision/recall for attack categories that exhibit distinct traffic characteristics, such as DDoS, DoS Hulk, FTP-Patator, and SSH-Patator. Applying WOA for hyperparameter optimization improves the overall performance to 0.93 accuracy and increases the macro-average F1-score from 0.88 to 0.89, reflecting better balance across classes and improved sensitivity for several sequential or time-dependent attacks. However, classes whose traffic patterns closely resemble benign behavior, such as Infiltration and certain Web Attack categories, still show relatively lower performance, suggesting the need for additional strategies to better capture subtle attack signatures. Overall, combining deep learning for feature representation, LightGBM for efficient structured-data classification, and WOA for adaptive hyperparameter search provides an effective and practical approach for server-side network anomaly detection and intrusion monitoring.

#### ACKNOWLEDGEMENTS

The authors would like to express their sincere gratitude to Politeknik Negeri Medan for their support and collaboration in this research. The institution's contribution and academic environment have greatly facilitated the completion of this study. The authors also appreciate the valuable assistance and support provided during the research process.

#### REFERENCES

- [1] R. Taheri, H. Ahmed, and E. Arslan, "Deep learning for the security of software-defined networks: a review," *Cluster Comput.*, vol. 26, no. 5, pp. 3089–3112, 2023, doi: 10.1007/s10586-023-04069-9.
- [2] A. Diro, S. Kaisar, A. V. Vasilakos, A. Anwar, A. Nasirian, and G. Olani, "Anomaly detection for space information networks: A survey of challenges, techniques, and future directions," *Comput. Secur.*, vol. 139, pp. 1–32, Apr. 2024, doi: 10.1016/j.cose.2024.103705.
- [3] A. S. Chan, H. A. Dwisastri, and F. Fachrizal, "Prediction of Poultry Farmer Performance Recapitulation Values in Broiler Chicken Farming Using Random Forest Regression," 2025, pp. 406–414. doi: 10.2991/978-94-6463-926-1\_46.
- [4] A. S. Chan, M. Husna, and P. P. Putra, "A Performance Enhancement Strategy for Sentiment Classification Models On Political Social Media Using Hyperparameter Tuning And Boosting," *International Journal of Advances in Data and Information Systems*, vol. 6, no. 3, pp. 763–772, Dec. 2025, doi: 10.59395/ijadis.v6i3.1455.
- [5] B. Harjito, K. N. Aini, A. Wijayanto, and B. Murtiyasa, "Comparison of Multinomial Naïve Bayes with K-Nearest Neighbors, Support Vector Machine and Random Forest for Classification of 'Network Attacks' Document," in *Fourth International Conference on*

- Informatics and Computing (ICIC)*, IEEE Xplore, 2020, pp. 1–6. doi: 10.1109/ICIC47613.2019.8985919.
- [6] E. Alagöz, Y. M. Sahin, K. Özdem, A. O. Gedik, and M. A. Akcayol, “Log Anomaly Detection in Application Servers Using Deep Learning,” in *International Conference on Artificial Intelligence and Applied Mathematics in Engineering*, D. J. Hemanth, U. Kose, B. Patrut, and M. Ersoy, Eds., 2024, pp. 258–268. doi: 10.1007/978-3-031-56310-2\_20.
- [7] K. Inayah and K. Ramli, “Analisis Kinerja Intrusion Detection System Berbasis Algoritma Random Forest Menggunakan Dataset Unbalanced Honeynet BSSN,” *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK)*, vol. 4, no. 11, pp. 867–876, Aug. 2024, doi: 10.25126/jtiik1148911.
- [8] R. P. Putra and Amarudin, “A Comparative Study of Machine Learning Algorithms for Intrusion Detection Systems using the NSL-KDD Dataset,” *Sistemasi: Jurnal Sistem Informasi*, vol. 14, no. 4, pp. 1654–1664, 2025, doi: 10.20895/inista.v6i2.1389.
- [9] M. R. Wijaya, “Inovasi Model Intrusion Detection System (IDS) menggunakan Double Layer Gated Recurrent Unit (GRU) dengan Fitur Berbasis Fusion,” *Jurnal Ilmiah Edutic : Pendidikan dan Informatika*, vol. 12, no. 1, pp. 10–21, Feb. 2025, doi: 10.21107/edutic.v12i1.28822.
- [10] A. R. Syujak, K. Diantoro, V. Yuni T, A. Soderi, and P. A. Sucipto, “Integrasi Deep Packet Inspection dengan Intrusion Detection System (IDS) untuk Identifikasi Serangan DDoS dalam Jaringan Skala Besar,” *Jurnal Minfo Polgan*, vol. 13, no. 2, pp. 1971–1975, Dec. 2024, doi: 10.33395/jmp.v13i2.14324.
- [11] R. Sovia, M. K. Anam, I. A. Wisky, R. Permana, N. A. Rahmi, and R. H. Zain, “Robust Predictive Model for Heart Disease Diagnosis Using Advanced Machine Learning Techniques,” *Journal of Applied Data Sciences*, vol. 7, no. 1, pp. 541–553, Jan. 2026, doi: 10.47738/jads.v7i1.1092.
- [12] M. K. Anam *et al.*, “Enhancing the Performance of Machine Learning Algorithm for Intent Sentiment Analysis on Village Fund Topic,” *Journal of Applied Data Sciences*, vol. 6, no. 2, pp. 1102–1115, 2025, doi: 10.47738/jads.v6i2.637.
- [13] M. K. Anam, S. Defit, Havaluddin, L. Efrizoni, and M. B. Firdaus, “Early Stopping on CNN-LSTM Development to Improve Classification Performance,” *Journal of Applied Data Sciences*, vol. 5, no. 3, pp. 1175–1188, 2024, doi: 10.47738/jads.v5i3.312.
- [14] M. K. Anam *et al.*, “Improved Performance of Hybrid GRU-BiLSTM for Detection Emotion on Twitter Dataset,” *Journal of Applied Data Sciences*, vol. 6, no. 1, pp. 354–365, Jan. 2025, doi: 10.47738/jads.v6i1.459.
- [15] N. F. Fahrudin, K. R. Putra, S. Umaroh, and G. B. Lautan, “Influence of Data Scaling and Train/Test Split Ratios on LightGBM Efficacy for Obesity Rate Prediction,” *MIND Journal*, vol. 9, no. 2, pp. 220–234, Dec. 2024, doi: 10.26760/mindjournal.v9i2.220-234.
- [16] M. H. Nadimi-Shahraki, H. Zamani, Z. Asghari Varzaneh, and S. Mirjalili, “A Systematic Review of the Whale Optimization Algorithm: Theoretical Foundation, Improvements, and Hybridizations,” *Archives of Computational Methods in Engineering*, vol. 30, no. 7, pp. 4113–4159, Sep. 2023, doi: 10.1007/s11831-023-09928-7.
- [17] M. K. Anam, T. P. Lestari, H. Yenni, T. Nasution, and M. B. Firdaus, “Enhancement of Machine Learning Algorithm in Fine-grained Sentiment Analysis Using the Ensemble,” *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 19, no. 2, pp. 159–167, Mar. 2025, doi: 10.37936/ecti-cit.2025192.257815.
- [18] K. Alemerien, S. Alsarayreh, and E. Altarawneh, “Diagnosing Cardiovascular Diseases using Optimized Machine Learning Algorithms with GridSearchCV,” *Journal of Applied Data Sciences*, vol. 5, no. 4, pp. 1539–1552, Dec. 2024, doi: 10.47738/jads.v5i4.280.
- [19] S. Chopannejad, A. Roshanpoor, and F. Sadoughi, “Attention-assisted hybrid CNN-BiLSTM-BiGRU model with SMOTE-Tomek method to detect cardiac arrhythmia based on 12-lead electrocardiogram signals,” *Digit. Health*, vol. 10, pp. 1–20, Jan. 2024, doi: 10.1177/20552076241234624.

- [20] M. K. Anam *et al.*, “Sara Detection on Social Media Using Deep Learning Algorithm Development,” *Journal of Applied Engineering and Technological Science*, vol. 6, no. 1, pp. 225–237, Dec. 2024, doi: 10.37385/jaets.v6i1.5390.
- [21] X. Li, Y. Lei, and S. Ji, “BERT- and BiLSTM-Based Sentiment Analysis of Online Chinese Buzzwords,” *Future Internet*, vol. 14, no. 11, pp. 1–15, Nov. 2022, doi: 10.3390/fi14110332.
- [22] J. M. Ahn, J. Kim, and K. Kim, “Ensemble Machine Learning of Gradient Boosting (XGBoost, LightGBM, CatBoost) and Attention-Based CNN-LSTM for Harmful Algal Blooms Forecasting,” *Toxins (Basel)*, vol. 15, no. 10, pp. 1–15, Oct. 2023, doi: 10.3390/toxins15100608.
- [23] A. Brodzicki, M. Piekarski, and J. Jaworek-Korjakowska, “The whale optimization algorithm approach for deep neural networks,” *Sensors*, vol. 21, no. 23, pp. 1–16, Dec. 2021, doi: 10.3390/s21238003.
- [24] M. K. Anam, T. P. Lestari, L. Efrizoni, N. S. Handayani, and I. Andhika, “Sentiment Analysis Optimization Using Ensemble of Multiple SVM Kernel Functions,” *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 9, no. 4, pp. 905–914, Aug. 2025, doi: 10.29207/resti.v9i4.6708.
- [25] P. P. Putra, M. K. Anam, A. S. Chan, A. Hadi, N. Hendri, and A. Masnur, “Optimizing Sentiment Analysis on Imbalanced Hotel Review Data Using SMOTE and Ensemble Machine Learning Techniques,” *Journal of Applied Data Sciences*, vol. 6, no. 2, pp. 936–951, May 2025, doi: 10.47738/jads.v6i2.618.
- [26] Mesran, S. R. Yahya, F. Nugroho, and A. P. Windarto, “Investigating the Impact of ReLU and Sigmoid Activation Functions on Animal Classification Using CNN Models,” *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 8, no. 1, pp. 111–118, Feb. 2024, doi: 10.29207/resti.v8i1.5367.
- [27] U. I. Arsyah, M. Pratiwi, H. Fryonanda, M. K. Anam, and Munawir, “ACLM Model: A CNN-LSTM and Machine Learning Approach for Analyzing Tourist Satisfaction to Improve Priority Tourism Services,” *Journal of Applied Data Sciences*, vol. 6, no. 4, pp. 2891–2906, Sep. 2025, doi: 10.47738/jads.v6i4.974.
- [28] M. R. P. Putra and E. Utami, “Comparative Analysis of Hybrid Model Performance Using Stacking and Blending Techniques for Student Drop-Out Prediction in MOOC,” *J. RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 8, no. 3, pp. 346–354, 2024, doi: 10.29207/resti.v8i3.5760.