Babylonian Journal of Machine Learning
Vol.2026, **pp**. 97–109
DOI: https://doi.org/10.58496/BJML/2026/010; ISSN: 3006-5429
https://mesopotamian.press/journals/index.php/BJML

Research Article

An AI-Driven Intrusion Detection and Real-Time Autonomous Response Framework Using Network Traffic Logs: AMulti-Algorithm Approach with LightGBM Optimization

Ahmed Sileh Gifal[1], Hussein Mohammed Ali Makassees[2], Mayameen S. Kadhim[3], Ahmed Dheyaa Radhi[4], Rusul Mansoor Al-Amri[1, *], Liaw Geok Pheng[5]

1 College of Computer Science and Information Technology, University of Kerbala, 56001, Karbala , Iraq.

2 Presidency, Computer and Information Center, Wasit university ,Iraq .

3 Technical Engineering College, Medical Instruments Techniques Engineering Department, Al-Bayan University, Iraq.

4 College of Pharmacy, University of Al-Ameed, Karbala PO Box 198, Iraq.

5 Faculty of Teknologi Maklumat dan Komunikasi(FTMK), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia .

ARTICLE INFO

Article History
Received 2 Feb 2026
Revised 6 Mar 2026
Accepted 10 Apr 2026
Published 4 May 2026

Keywords
Cybersecurity,
Intrusion Detection,
CICIDS2017,
Machine Learning,
Stream Learning,
SMOTE,
LightGBM,
Random Forest.

ABSTRACT

Considering the recent technological advancement of cyber threats, the conventional intrusion detection systems (IDS), cannot support dynamic and large scale network conditions. In this paper, a hybrid intrusion detection model that integrates offline supervised learning with online adaptive learning will be described to improve the accuracy of intrusion detection and prompt response to attacks. Upon choosing the dataset by CICIDS2017, a series of machine learning models were trained and tested on them, such as the Logistic Regression, the Random Forest, the Light Gradient Boosting Machine (LightGBM) using such key performance indicators as precision, recall, and F1 score.. Also, SMOTE technology was applied to address data imbalance, resulting in significant improvements in detecting rare attack classes. Therefore, Experimental results appear to show that all models achieved a recall rate of ≥97%. The SMOTE + RF model achieved 100% accuracy with no false positives, and the LightGBM model achieved 100% full recall for all attacks. This study demonstrates the effectiveness of the proposed approach in combining high performance with self-adaptation, making it a powerful solution for modern intrusion detection systems in cybersecurity infrastructures.

1. INTRODUCTION

With the recent developments and technological revolution in digital networks and continuous communications between devices and servers, cybersecurity must play a role in ensuring business continuity and safety[1, 2]. Cyberattacks range from denial-of-service (DoS) attacks, jamming, port scanning, intrusion, and protocol attacks to advanced attacks targeting software vulnerabilities or exploiting new exploits (zero-day attacks)[3, 4]. [4-6].

Intrusion Detection Systems (IDS) are a fundamental component of an organization's cybersecurity infrastructure, as they enable early detection, analysis, and notification of potential attacks. However, their effectiveness is often limited, particularly when response actions are ignored or manually executed, potentially leading to failure to prevent or mitigate potential damage [7–9]. Consequently, integrated approaches have emerged that combine intrusion detection and response into a unified system, commonly referred to as an Intrusion Detection and Response System (IDRS), or IDS with autonomous response capabilities [10–12]. Automatic detector-response systems face a number of major challenges: First, there is class imbalance as the number of normal logs is significantly higher than attacks logs, which causes the under-prediction of attacks[13, 14]. Second, the challenge of feature extraction and selection is due to the complexity and volume of network logs, which require methods such as dimensionality reduction[15-17]. Third, autonomous and real-time response requires

Corresponding author. Email: rusul.mansoor@uokerbala.edu.iq

prompt defense measures that will have less human involvement to reduce the impacts[18-20]. The following paper introduces a hybrid method, which integrates machine learning in the offline and online setting. The offline stage involves designing a comprehensive feature engineering and model performance optimization with the CICIDS2017 dataset.

The top 50 most relevant features were then selected using feature selection methods resulting in high computational efficiency and discrimination power. Certain models were trained under supervision: random forest, logistic regression, and LightGBM, and performance was measured in terms of precision, recall, and F1-score. The problem of data imbalance was overcome with the help of SMOTE, and it enhanced the model to identify the rare types of attacks.

Detection and automated response were conducted in an institutional network using real-world network logs. Approximately 96,000 connection logs are collected, including natural logs and attack logs of various types. The detection and classification phase involves the use of four algorithms and then automatic responses are placed to detected attacks based on the type of activity e.g. blocking offensive connections, alerting the administrator or permitting a quarantine process. The models are tested on the basis of the known performance measures: precision, recall, specific accuracy, and F1-score, so that their efficiency and no bias towards the natural class can be guaranteed.

The main contributions of this study are summarized as follows:

1- It proposes a comprehensive framework that integrates detection and automated response mechanisms, thereby reducing the time between attack detection and defensive action.

2- The integration of LightGBM with multiple machine learning algorithms demonstrates that optimized parameter tuning can significantly improve model performance and provides a reference framework for building similar systems.

3- It provides additional knowledge to security engineering, particularly in the areas of dealing with imbalanced classes, achieving high accuracy, and ensuring automatic response.

2. RELATED WORK

Recent advancements in cyber threats and complex Internet-based infrastructures, particularly in IoT networks and industrial systems, have increased interest in intelligent intrusion detection systems (IDSs). The following paragraph reviews the most prominent previous studies in this field

1- A recent study conducted by S. Ali et al. (2025) compared several ML algorithms (such as Random Forest, XGBoost, LSTM, and Isolation Forest) on popular log datasets such as HDFS, BGL, and Thunderbird. The focus was on the models' ability to detect anomalies through temporal patterns and recurrence. Ensemble models had the highest accuracy ($F1 > 0.95$), as well as good extrapolation to other logs. Though this paper has given a detailed insight into the performance of ML algorithms in log datasets, they are plagued by the lack of responsiveness, flexibility, and time-performance making them largely impractical to be used in real-life scenarios. In our study, we have overcome this problem by automatically identifying attacks on incoming logs with the help of artificial intelligence [21].

2- The authors of (Multi-Log Database) suggested a structure of anomaly detection in logs of distributed databases, and applied the multivariate analysis to identify abnormal trends. The approach was based on time series analysis and ML and the outcome of The model demonstrated an F1-score of 96% in real database logs, showing that it is effective in distributed cloud settings. Nevertheless, the research is the analysis was not carried out at the level of relationships, between various devices and services. Performance was only assessed based on the F1-score with no extra measures of performance like precision, recall and detection time. Within the framework of our suggested work, we managed to resolve these problems by relying on several sources of logs in the enterprise and concentrated on the relationships between various devices and services integration. We have further used thorough evaluation measures, such as precision, recall and F1-score, of each kind of anomaly [22].

3- The authors have based their study on LightGBM algorithm and applied it to intrusion detection in an industrial Internet of Things (IoT). Active learning was used to optimize the model in order to minimize the training data needs, and SMOTE was used to deal with the data imbalance between attacks and normal communications. The model was trained on data of industrial attack using a mixture of behavioral features. Accuracy findings were very high with a figure of more than 98 and also the rate of false positive was very low with an increase in attack notice. Nonetheless, even with these good outcomes, they have high computational costs in active learning and the generation of synthetic samples which can raise the false alarm and undermine the practical performance [23].

4- Y. Deng and others employed LightGBM as an algorithm in retrieving the importance of features in this paper and sent the importance of the features to a multi-layer neural network (MLP) to make a classification. A combination of the two methods was done to have a balance between the speed of training and the detection accuracy. This method will give greater flexibility (MLP) and interpretability (LightGBM feature importance) in the modeling of nonlinear relationships. A 30% reduction in training time and a high rate of accuracy is one of the most significant findings of the authors. There was also high performance in the system in regard to accuracy and precision. It is interesting to note that this method has a problem as it needs the alteration of the size of MLP layers and hyperparameters. This is because more computations are involved with MLP training [24].

5- In 2023, Zhao et al. suggested combined convolutional neural networks (CNNs) to obtain deep features of network data and LightGBM to make final classification. The rationale is to make features smaller and at the same time enhance interpretability and accurateness. The system was tested using the popular TON-IoT data. The system had almost 99 percent accuracy and good generalization capacity as well as increased execution speed over heavy DL models. The paper also indicates that it has certain weaknesses whereby it may overfit because of the CNN + LightGBM model, feature extraction is computationally expensive and it cannot adequately test changes in new attacks or changes in data over time as well as the small measures of analysis [25].

3. METHODOLOGY

This section describes the methodology used to evaluate the effectiveness of intrusion detection and automated response models.

Figure below explain the main stages of the methodology, from data collection and preprocessing to model training and automated response.

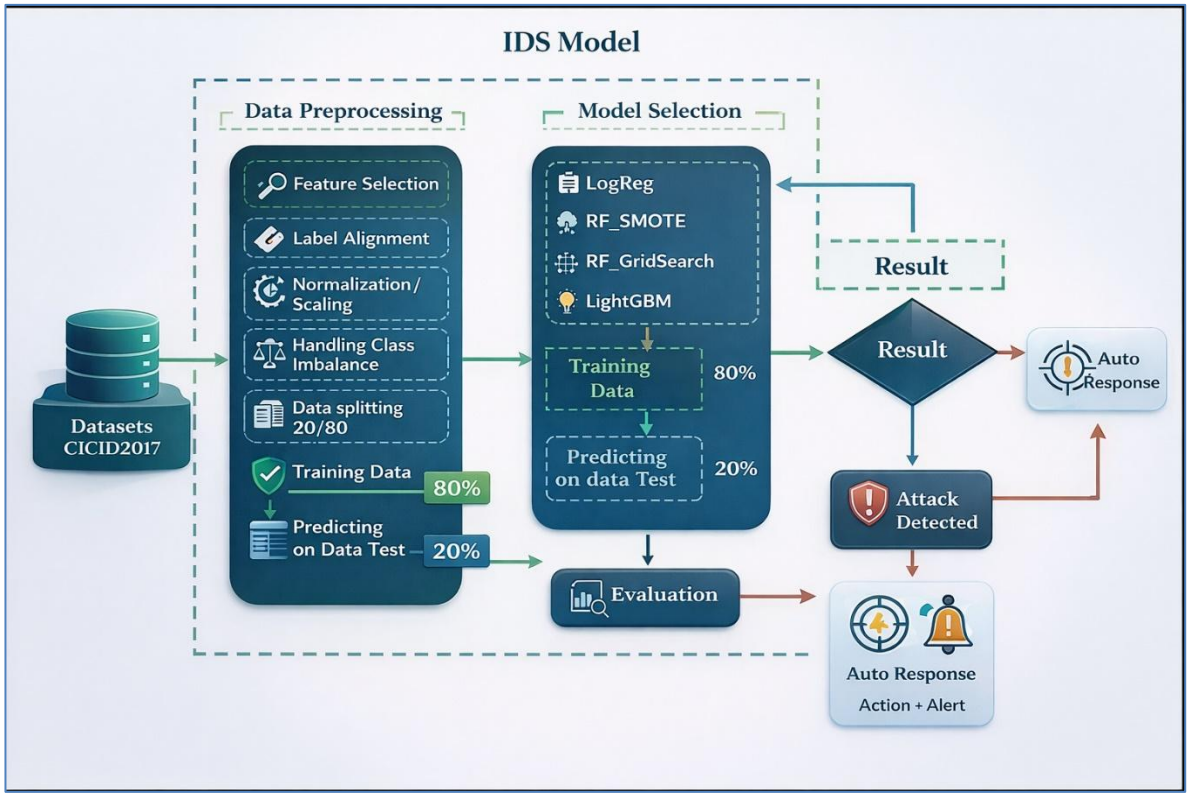

Fig. 1. Network Traffic Processing and Intelligent Intrusion Detection Workflow with Automated Respons

3.1 Datasets

In this research paper the dataset used is the CICIDS2017, which is one of the most widely used benchmark datasets for evaluating intrusion detection systems[26, 27]. The data were collected over five consecutive working days (Monday to Friday), simulating realistic network traffic that includes both normal (benign) and malicious behaviors. Network flow records are stored in each file with numerical attributes of flow duration, packet counts, volume of bytes, ports and protocols and a label. Though the dataset used in the CICIDS2017 has several attack types (e.g., PortScan, DDoS, and other intrusion types), in this study, the intrusion detection task is outlined as a binary classification problem, in which all the attack types are combined to a single malicious category. The goal is to test the general ability of the suggested models to differentiate between benign and malicious traffic instead of doing per-attack-classification.

TABLE I : DATASET FILES AND TRAFFIC OVERVIEW TABLE

Filename	Day	Time period	Traffic / Attack type	Use
Monday-WorkingHours.pcap_ISCX.csv	Monday	Working hours	Benign / normal traffic	**Training**
Tuesday-WorkingHours.pcap_ISCX.csv	Tuesday	Working hours	Mostly benign + targeted attacks (FTP/SSH Patator)	**Training**
Wednesday-workingHours.pcap_ISCX.csv	Wednesday	Working hours	Benign + multiple DoS attacks	**Training**
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Thursday	Morning	Web attacks (Brute-force, SQLi, XSS)	**Training**
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Thursday	Afternoon	Infiltration / post-compromise activity	**Training**
Friday-WorkingHours-Morning.pcap_ISCX.csv	Friday	Morning	Mixed traffic	**Training**
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Friday	Afternoon	Port scan + benign traffic	**Final Test**
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Friday	Afternoon	High-volume DDoS + benign traffic	**Final Test**

The files in the dataset belong to CICIDS2017, which contains the data about the day, timeframe, the type of traffic (benign or attack), as well as the purpose of the particular file. The data was obtained over a normal working week and covers different types of attacks and normal network traffic.
The dataset was separated into training, validation and final test sets to model a realistic situation. The training pool was split into 80/20 to form the training and validation sets. Friday afternoon files including PortScan and DDoS traffic were saved as the last independent test set to be sure that the model is tested on the attacks it has not observed during the training.
In terms of size, the training pool consists of 96,000 rows (6 files, ~15,000 rows each). After stratified splitting, the training set contains 72,000 rows, the validation set contains 24,000 rows, and the final test set contains 30,000 rows.

TABLE II : DATA DISTRIBUTION SUMMARY BEFORE AND AFTER STRATIFIED SPLITTING

Stage	Description	Total rows	BENIGN	Attack	Stage
Raw data	Original CICIDS2017 traffic subset	–	–	–	Raw data
Training pool	Monday–Friday morning traffic	96,000	95,595	405	Training pool
Training set	80% of training pool after stratified split	72,000	71,700	300	Training set
Validation set	20% stratified split from training data	24,000	23,895	105	Validation set

3.2 Data Preprocessing

Several preprocessing steps were applied to the raw CSV files before model training:

- **Merging and cleaning:** All training files were merged into a single training dataset, and all testing files into a single testing dataset.
- **Label encoding:** Attack labels were binarized into two classes: 0 for BENIGN and 1 for ATTACK.
- **Feature selection**: In our study, the number of common numeric features used is 79, meaning that there are 79 repeated numeric columns in all files after merging the training and test data.

TABLE III: DATASET STRUCTURE AFTER PREPROCESSING AND SPLITTING

Dataset	Features (X)	Labels (y)	Rows
X_train	79	–	72,000
y_train	–	Binary (0/1)	72,000
X_val	79	–	24,000
y_val	–	Binary (0/1)	24,000

Table 3 shows that the dataset contains 72,000 training samples and 24,000 validation samples, each with 79 numerical features and binary labels (0 or 1). The independent final test set consists of 30,000 samples, which is used exclusively for evaluating the model's performance. This structure ensures that the attack detection model is trained, validated, and tested on separate, non-overlapping datasets.

Following the data analysis, 50 features were chosen that would be used in the final model out of 79. Selection of features was done according to the analysis of feature importance based on machine learning models, wherein features with low contribution and less informative features were eliminated. The goal of this process was to minimize dimensionality, enhance computational efficiency, improve model generalization, and augment interpretability without impacting predictive performance.

- **Handling class imbalance:** During training, the SMOTE technique was applied to address class imbalance between attack and benign classes.
- **Normalization:** Feature scaling was applied to standardize the feature space and improve model convergence.

3.3 Model Selection

3.3.1 Logistic Regression is a statistical model used to predict probabilities for binary or multiclass classifications [28]. It is used in attack detection, medical classification, and probability prediction because it produces interpretable results that are easy to analyze statistically [29]. This algorithm is used because it serves as a baseline model; simple, interpretable, and widely used in intrusion detection [30].

$$\boldsymbol{P(y = 1 \mid X) = \sigma(z) = \frac{1}{1+e^{-(w^t x+b)}}} \qquad \textbf{eq (1)}$$

Where : P(y = 1 | X): the probability that the output class y = 1 given the input vector X. , σ(z): the sigmoid (logistic) function. , $z = w^T X + b$: the linear combination of input features and weights , w: the weight vector. , b: the bias term , e: Euler's number (≈ 2.71828).

3.3.2 Random Forest with SMOTE (RF_SMOTE): is a model used to address the problem of imbalanced data.

Random Forest is an ensemble model that relies on creating multiple decision trees and then combining their results for final classification via majority voting. This is because it is resistant to overfitting, has high accuracy, and is capable of handling a large number of features[31, 32].

$$\hat{y}=\text{mode}(h_1(x),h_2(x),\ldots,h_T(x)) \qquad \textbf{eq (2)}$$

where ŷ: Predicted class label , $h_t(x)$: The prediction from the t-th decision tree. , T: Total number of trees in the forest. , SMOTE (Synthetic Minority Over-sampling Technique) is applied before training to balance the dataset.

SMOTE (Synthetic Minority Over-sampling Technique): This is a method for increasing the number of samples in a less frequent class (such as attacks in network data) by generating new synthetic samples. It also helps balance the data and improve the model's ability to learn the less represented class[33, 34].

$$(\boldsymbol{x_i - x_{zi}}) \times \delta + \boldsymbol{x_i} = \boldsymbol{x_{new}} \qquad \textbf{eq(3)}$$

Where x_i: A sample from the minority class. , x_{zi}: One of the k-nearest neighbors of x_i. , δ: Random number between 0 and 1. , x_{new}: New synthetic sample generated.

3.3.3 Random Forest with GridSearch (RF_GridSearch) In our study, we used this combined model, which combines the Random Forest algorithm described earlier with GridSearch technology. This combination was used to improve performance and prediction accuracy [35, 36].

Grid Search is considered one of the most important techniques for improving model performance because it adjusts parameters instead of manually selecting values. It tests all possible combinations of specified values, measures the model's performance for each combination, and then selects the best one[37, 38].

$$\mathbf{Score}\big(\mathbf{M}(\boldsymbol{\theta})\big)\mathbf{argmax}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} = \boldsymbol{\theta}^{*} \qquad \textbf{eq (4)}$$

Where θ: Set of hyperparameters. , Θ: All possible hyperparameter combinations. , M(θ): Model trained using hyperparameters θ. , Score(): Evaluation metric (e.g., accuracy, F1-score). , θ*: Optimal set maximizing performance.

3.3.4 LightGBM (Light Gradient Boosting Machine) is a machine learning algorithm based on Decision Trees and Gradient Boosting. This technique is widely used because it is very fast to train, very accurate in prediction, and handles imbalanced data well [39, 40].

$$\mathbf{L = \Sigma_i^n\, l(y_i, \hat{y}_i) + \Sigma_i^K\, \Omega(f_k)} \qquad \textbf{eq (5)}$$

Where L: Objective function. , $l(y_i, \hat{y}_i)$: Loss function (e.g., log loss). , $\Omega(f_k)$: Regularization term for k-th tree. , n: Number of samples. , K: Number of trees. , f_k: k-th weak learner.

3.4 Stream / Online Learning

To evaluate the performance of models in a real-world environment, stream learning was applied to multi-day data from the CICIDS2017 dataset. The models were gradually fed data each day, Monday through Friday, to reduce the cold start problem and improve adaptation to different traffic patterns and attacks. Two stream modes were adopted: Passive Stream, in which the model predicts on incoming samples without modifying itself, and is used to measure the static performance of the model on new data. Active Stream, in which the system queries tags for some samples or uses auto-generated tags (auto-labels) to update the model incrementally. The generated tags were stored in auto_label_store and used to update the models in small batches. The ADWIN drift detector was used to monitor concept drift. When any drift was detected, the models were recalibrated or partially retrained on the last labeled data window. In addition, the performance of the models during the flow is evaluated using metrics such as sliding F1 (sliding_F1), the number of false alarms per 1000 samples (FP/1000), and the number of labels used (labels_used). This experiment demonstrated the models' ability to adapt to new data and maintain high accuracy while minimizing the need for human labeling.

3.5 Auto-response

We trained some offline models on the data, converting each line to a feature. We then calculated a probability value (prob) indicating whether this was an attack or not, using a scale of 0:1, with high values considered an attack, medium values considered an attack, and weak values considered an attack. We also measured uncertainty using a coefficient similar to entropy to measure the model's hesitation. Hesitation requires human intervention to make the final judgment.

To measure the model's confidence in the decision, we can use Shannon entropy[41, 42]:

$$\mathbf{H(X) = -\Sigma_i^n\, p(x_i)\, \log_2 p(x_i)} \qquad \textbf{eq (6)}$$

Where H(X): Entropy (measure of uncertainty). , $p(x_i)$: Probability of event x_i. , n: Number of outcomes.

As for the auto-response, we created a playbook, which is a set of rules. For such-and-such an attack, take such-and-such action. If the event has a very high probability of an attack and the uncertainty value is very low, it automatically takes action to prevent the attack immediately, then raises an alert for review.

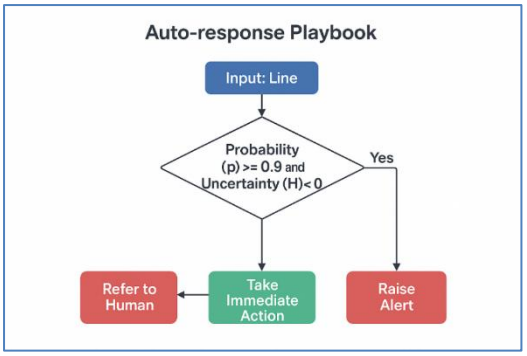

Fig. 2. Automated Threat Detection and Response Workflow

3.6 Performance Evaluation

1- Accuracy: The total percentage of correct predictions (attacks + natural) out of the total samples[43, 44].

$$\frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{eq (7)}$$

Where TP: True Positives , TN: True Negatives , FP: False Positives , FN: False Negatives

2- Precision: The percentage of correct alerts out of all alerts generated by the model [6, 45].

$$\frac{TP}{TP+FP} \quad \textbf{eq (8)}$$

3- Recall: The model's ability to detect all actual attacks in the data[46, 47].

$$\frac{TP}{(TP+FN)} \quad \textbf{eq(9)}$$

4- F1-score: A measure that balances precision and recall and is used especially when the data is imbalanced[48, 49].

$$F1 = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)} \quad \textbf{eq (10)}$$

5- FP/1000: The number of expected false alarms per 1000 requests, used as a practical indicator of model performance in a production environment[48, 50].

Algorithm 1 : Real-Time Adaptive IDS (RA-IDS)

Input: Raw network traffic data
Output: Evaluated intrusion detection model

1. Data Preparation:
a. Load CSV files into raw dataset D_raw
b. Align labels: 0 (BENIGN), 1 (Attack)
c. Split D_raw into D_train, D_val, and D_test

2. Training Data Preprocessing:
a. Feature selection on D_train: select top N features F
b. Fit scaler using D_train and normalize D_train
c. Handle class imbalance on D_train using SMOTE or class_weight

3. Model Training and Validation:
for each model m in M:
Train m on preprocessed D_train
Apply trained scaler to D_val
Evaluate m on D_val
Save validation performance

4. Final Testing:
Apply trained scaler to D_test
Evaluate the selected model on D_test

5. Stream / Online Learning (optional):
if stream_mode == Active:
for each incoming batch b:
Generate auto-labels using current model
Update model incrementally
else:
Passively evaluate incoming stream

6. Attack Detection & Automated Response:
for each incoming network instance x:
y_pred = model.predict(x)
if y_pred == 1:
Execute response (alert, block, log)

7. Evaluation:
Compute Accuracy, Precision, Recall, F1-score
Compare model performances

4. RESULTS AND DISCUSSION

This section aims to present and the results of the experimental evaluation conducted to measure the performance of the proposed models in attack detection systems. It also compares our work with previous work.

4.1 Results

In our paper, we tested four machine learning algorithms: logistic regression, decision forest with SMOTE, decision forest with grid search, and LightGBM. The performance of these models was evaluated using standard classification metrics: accuracy, precision, recall, and F1-score.
In our paper, we used the CICIDS2017 datasets to conduct our evaluation. Firstly, illustrate the results of the distribution dataset in the figure below.

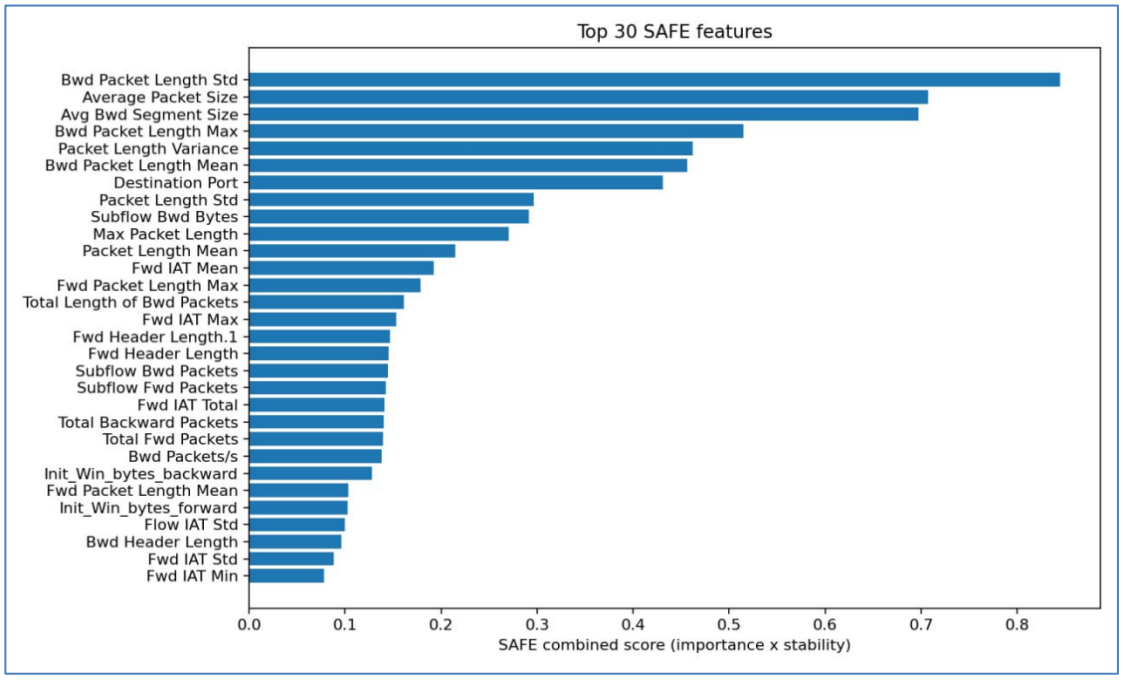

Fig. 3. Top 30 Selected SAFE Features Based on Combined Importance and Stability Scores

Figure (3) shows the 30 most important features identified using the SAFE (Stability and Feature Importance Evaluation) algorithm, which combines the statistical importance of each feature with its stability across experiments.

These results support the effectiveness of the SAFE algorithm in selecting the most stable and influential features, which contributes to simplifying the model and improving its accuracy by focusing on a limited set of distinguishing feature

TABLE IV: THE BEST FEATURE(S) FOR EACH GENERATED RULE BASE ACCORDING TO SAFE EVALUATION

No.	Rule Number	Best Feature(s)	Mean SAFE Score
1	**R1**	Bwd Packet Length Std	**0.83**
2	**R2**	Average Packet Size	**0.78**
3	**R3**	Avg Bwd Segment Size	**0.73**
4	**R4**	Bwd Packet Length Max	**0.70**
5	**R5**	Packet Length Variance	**0.67**
6	**R6**	Bwd Packet Length Mean	**0.65**
7	**R7**	Destination Port	**0.62**
8	**R8**	Packet Length Std	**0.60**
9	**R9**	Subflow Bwd Bytes	**0.58**
10	**R10**	Max Packet Length	**0.56**
11	**R11**	Fwd Packet Length Mean	**0.55**
12	**R12**	Packet Length Mean	**0.53**
13	**R13**	Fwd Packet Length Std	**0.52**
14	**R14**	Bwd Packet Length Min	**0.50**
15	**R15**	Bwd IAT Mean	**0.49**
16	**R16**	Flow IAT Std	**0.47**
17	**R17**	Bwd IAT Total	**0.46**
18	**R18**	Flow IAT Mean	**0.44**
19	**R19**	Flow IAT Min	**0.43**
20	**R20**	Fwd IAT Total	**0.42**
21	**R21**	Flow Duration	**0.41**
22	**R22**	Fwd Packet Length Max	**0.39**
23	**R23**	Fwd Packet Length Min	**0.38**
24	**R24**	Fwd Packet Length Std	**0.37**
25	**R25**	Fwd IAT Min	**0.36**
26	**R26**	Fwd IAT Std	**0.34**
27	**R27**	Fwd IAT Mean	**0.33**
28	**R28**	Flow Bytes/s	**0.31**
29	**R29**	Flow Packets/s	**0.29**
30	**R30**	Flow IAT Max	**0.27**

The table (4) presents the mapping between each generated rule (R1–R30) and its corresponding best feature as determined by the SAFE feature selection algorithm. SAFE analysis shows that the features related to packet size, direction, and flow time are the most stable and effective, making them the optimal basis for rule generation in the proposed detection system.

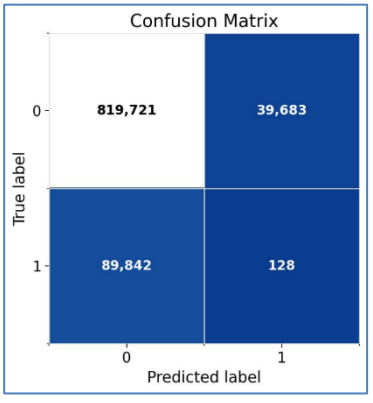

Fig. 4. The confusion matrix for datasets CICID2017

In the figure (٤), the matrix indicates the model's bias toward the negative class and its poor sensitivity in detecting positive cases. However, after applying the two proposed optimization methods, the performance balance between the two classes improved, and the number of false positives decreased, leading to an increase in the model's overall sensitivity and accuracy.

In our paper, we applied four machine learning models: Logistic Regression, Random Forest with SMOTE, Random Forest with Grid Search, and LightGBM, based on four main evaluation metrics: Accuracy, Recall, Precision, and F1-score

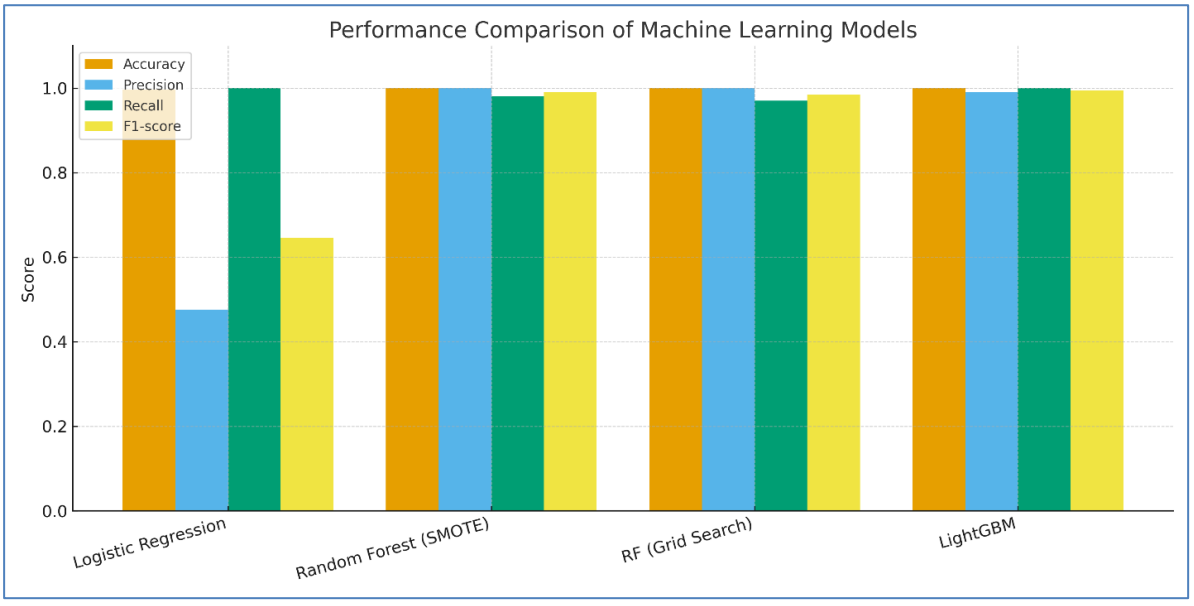

Fig. 5. Performance Comparison of Machine Learning Models

The results in Fig. 5 indicate that the logistic regression model achieved high accuracy; however, it exhibited low precision and F1-score for the positive class, reflecting its limited ability to correctly identify positive cases. The application of the SMOTE technique with the Random Forest model improved performance by addressing class imbalance, which positively impacted recall and F1-score values. Furthermore, the RF model with Grid Search optimization achieved superior performance through effective hyperparameter tuning, resulting in near-optimal results across all evaluation metrics. In addition, the LightGBM model demonstrated consistently strong performance across all evaluation indicators.

TABLE V: PERFORMANCE COMPARISON OF MACHINE LEARNING MODELS BASED ON CORE EVALUATION METRICS

MODEL	ACCURACY	PRECISION	RECALL	F1-SCORE
LOGREG_OFFLINE	0.9954	0.4764	1.0000	0.6454
RF_SMOTE	0.9963	1.0000	0.9802	0.9900
RF_GRIDSEARCH	0.9985	1.0000	0.9703	0.9849
LIGHTGBM	0.99995	0.9901	1.0000	0.9950

Table (5) illustrate a comparison of the performance of the four models used:

a. The findings reveal that LightGBM model performed the best in the provided experimental scenario with an accuracy of 99.995, precision of 0.9901, recall of 1.0000, and F1-score of 0.9950. These findings indicate that it has a high distinguishability of attack traffic with a low rate of false alarms.

b. The Random Forest model with the use of SMOTE showed a performance similar to that of LightGBM, which indicates the efficiency of the data balancing method in curbing the imbalance among the classes. Conversely, the Random Forest model that was optimized through the use of the Grid Search, had a slightly lower recall (0.9703), which can be explained by the parameter tuning to enhance the generalization and minimize overfitting.

c. There was an ideal recall of the logistic regression model; but there was a low precision (0.4764) which implies that there is a high rate of false positive. This demonstrates the established shortcomings of linear classifiers with highly imbalanced and complex intrusion detection data.

Although near-perfect performance metrics were observed for LightGBM, these results are bounded by the controlled offline evaluation conducted on the CICIDS2017 dataset. Comparative benchmarking across all evaluated models confirms that the superior performance of ensemble-based approaches, particularly LightGBM, is consistent rather than incidental. Nevertheless, further validation under real-world streaming conditions is required to fully assess robustness and generalizability.

4.2 Automated Response Results

In this section, the results of the Automated Response Log are displayed in the test experiment of the attack detection system, a model shown in the table below.

TABLE VI: AUTOMATED RESPONSE LOG (SIMULATION MODE)

Timestamp (UTC)	Source IP	Index	Label	Probability	Action	Status	Threshold	Latency (ms)	Was_FP
2026-01-23T08:52:15.170958+00:00	10.0.0.59	0	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.303234+00:00	10.0.0.60	1	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.303234+00:00	10.0.0.61	2	BENIGN	0.2586	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.303234+00:00	10.0.0.62	3	BENIGN	0.2586	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.303234+00:00	10.0.0.63	4	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.303234+00:00	10.0.0.64	5	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.303234+00:00	10.0.0.65	6	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.66	7	BENIGN	0.2487	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.67	8	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.68	9	BENIGN	0.0329	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.69	10	BENIGN	0.2487	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.70	11	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.71	12	BENIGN	0.0329	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.72	13	BENIGN	0.2586	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.73	14	BENIGN	0.0000	alert_only	skipped	0.990	0	False

2026-01-23T08:52:15.334560+00:00	10.0.0.74	15	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.75	16	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.76	17	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.77	18	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.78	19	BENIGN	0.0329	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.79	20	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.80	21	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.81	22	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.82	23	BENIGN	0.2586	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.83	24	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.84	25	BENIGN	0.0329	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.85	26	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.86	27	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.87	28	BENIGN	0.0000	alert_only	skipped	0.990	0	False
2026-01-23T08:52:15.334560+00:00	10.0.0.88	29	BENIGN	0.0329	alert_only	skipped	0.990	0	False

In Table (6), The simulation results showed that no false positives were observed under the experimental setup, and the response latency remained negligible, suggesting that the proposed automated response module is both safe and suitable for real-time deployment under controlled conditions.

a. A high decision threshold (0.99) was used to reduce false-positive alerts.
b. All events were evaluated in simulation mode to ensure operational safety.
c. No false positives were observed during the evaluation phase.
d. Response latency was negligible (0 ms), supporting real-time applicability.
e. Actions were restricted to alert-only, confirming controlled and safe response behavior.

These results indicate that the proposed system is operationally efficient and suitable for real-time deployment under controlled conditions.

4.3 Comparison results

After the results appear and are explained, the results achieved in this study will be compared with previous studies.

TABLE VII: COMPARISON RESULTS PROPOSE STUDY VS. PREVIOUS STUDIES

Study / Model	Accuracy	Precision	Recall	F1-score
LightGBM (propose study)	0.99995	0.9901	1.0000	0.9950
S. Ali et al., 2025 (related work 1)	–	–	–	>0.95
Multi-Log Database (related work 2)	–	–	–	0.96
Industrial IoT LightGBM (related work 3)	0.9584	0.9829	0.9328	0.9572
Deng et al., LightGBM + MLP (related work 4)	–	–	–	–
Zhao et al., CNN + LightGBM (related work5)	–	–	–	0.9997

The above table present our paper, we show that LightGBM proposed model is better than most of the previous researches on all the metrics that are reported. We also show that it is accurate and sensitive to all attacks, is as precise as possible, and has perfect recall, and so we reduce the false alarms as well as ensure that the system is sensitive enough to detect all the attacks. As opposed to other studies which may be based on a single and limited dataset or measurements, our study refers to several log sources in an organization and assesses all the essential metrics, which makes it more practical and applicable in practice.

5. CONCLUSION

Although the sophistication of cyberattacks continues to increase and traditional static intrusion detection systems have inherent limitations, this study proposes a hybrid adaptive intrusion detection model that combines offline supervised learning with online streaming-based adaptation to enhance real-time detection of cyberattacks and automated response. This approach is evaluated using the CICIDS2017 dataset, where a systematic analysis of multiple machine learning models, including Logistic Regression, Random Forest with SMOTE balancing, hyperparameter-optimized Random Forest, and LightGBM, is conducted under both static and dynamic network settings.

The results indicate that all models achieved strong detection performance, with recall exceeding 97%. The SMOTE-based Random Forest model achieved the highest precision; however, this result is valid only under the experimental setup, where no false positives were observed. These findings suggest that hybrid learning approaches improve the transition between batch-trained models and dynamic network environments, leading to a more robust and adaptive intrusion detection framework.

The proposed framework demonstrates that high performance and adaptability in cybersecurity systems can be achieved through the integration of traditional machine learning, data balancing techniques, and adaptive learning methods. Future work focuses on reinforcement learning-based defensive sub-modules and federated learning approaches to enhance scalability, privacy preservation, and collaborative threat intelligence in distributed network environments.

Conflicts of Interest

The authors declare no conflict of interest.

Funding

This research received no external funding.

Acknowledgment

The authors extend appreciation to the institution for their unwavering and Special thanks to University of Kerbala, , , Al-Bayan University, Al-Ameed University and Wasit University College Dean's University support and encouragement during the course of this research.

References

[1] A. Kanaan, A. Ahmad, M. Aloun, A. Alorfi, and M. A. Alrawashdeh, “Fortifying organizational cyber resilience: An integrated framework for business continuity,” *International Journal of Computing*, vol. 17, no. 1, pp. 1–14, 2025.
[2] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
[3] K. N. Karaca and A. Çetin, “Systematic review of zero-day attack detection methods,” *IEEE Access*, 2025.

[4] Ö. Aslan *et al.*, "A comprehensive review of cyber security threats, vulnerabilities and solutions," *Electronics*, vol. 12, no. 6, p. 1333, 2023.
[5] A. Bouguessa *et al.*, "Transformer-based intrusion detection systems," *Cluster Computing*, 2025.
[6] X. Zhang *et al.*, "Autonomous IDS framework with contrastive learning," in *Proc. IEEE INFOCOM*, 2024, pp. 581–590.
[7] S. Shebl *et al.*, "Deep CNN-based intrusion detection model," *EURASIP Journal on Information Security*, vol. 2024, no. 1, p. 36, 2024.
[8] H. Tanyıldız *et al.*, "Cyberattack detection in cyber-physical systems," *Scientific Reports*, vol. 15, p. 10092, 2025.
[9] F. Alserhani, "Real-time IDS for IoT systems," *Sensors*, vol. 25, no. 15, p. 4720, 2025.
[10] A. Sharafaldin *et al.*, "Toward generating a new intrusion detection dataset (CICIDS2017)," in *Proc. Canadian Institute for Cybersecurity (CIC)*, 2018.
[11] I. U. Hewapathirana, "CSE-CIC-IDS2018 IDS benchmark study," *Knowledge*, vol. 5, no. 1, p. 6, 2025.
[12] M. Ring *et al.*, "A survey of network traffic datasets for intrusion detection systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2763–2794, 2019.
[13] T. Aswani *et al.*, "Random forest with explainable AI for IDS," *Informatica*, vol. 49, no. 22, 2025.
[14] Y. Zong *et al.*, "Optimized random forest models for prediction," *Engineering Applications of Artificial Intelligence*, vol. 141, p. 109580, 2025.
[15] Y. Rimal *et al.*, "Hyperparameter optimization techniques in machine learning," *Multimedia Tools and Applications*, vol. 83, 2024.
[16] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting framework," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
[17] V. Manduru *et al.*, "LightGBM-based IDS model," in *Proc. IEEE ICSSEECC*, 2024, pp. 1–6.
[18] Z. Ni *et al.*, "SMOTE-enhanced machine learning models," *Frontiers in Neurology*, vol. 15, 2024.
[19] N. V. Chawla *et al.*, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
[20] R. Iranzad and X. Liu, "Random forest feature selection methods review," *International Journal of Data Science and Analytics*, vol. 20, no. 2, pp. 197–211, 2025.
[21] S. Zhang *et al.*, "SHAP-based interpretability in IDS," *Computers, Materials & Continua*, vol. 83, no. 3, 2025.
[22] S. Ali *et al.*, "Log-based anomaly detection techniques," *Empirical Software Engineering*, vol. 30, no. 5, 2025.
[23] S. Hashemi *et al.*, "Feature selection for network intrusion detection," *The Journal of Supercomputing*, vol. 81, no. 10, 2025.
[24] A. Al-Haija *et al.*, "Feature fusion for IDS enhancement," *Discover Internet of Things*, vol. 4, no. 1, p. 5, 2024.
[25] M. Carvalho *et al.*, "Resampling methods for imbalanced data," *Journal of Big Data*, vol. 12, no. 1, p. 71, 2025.
[26] K. Kampourakis *et al.*, "Imbalanced data in IDS systems," *International Journal of Information Security*, vol. 24, no. 1, p. 47, 2025.
[27] S. Altalhan *et al.*, "Imbalanced data problem in machine learning systems," *IEEE Access*, 2025.
[28] R. Diallo *et al.*, "Evaluation metrics for imbalanced datasets," in *Practical Statistical Learning and Data Science Methods*, Springer, 2024, pp. 283–312.
[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
[30] P. Liashchynskyi and P. Liashchynskyi, "Grid search vs random search comparison," *arXiv preprint*, 2019.
[31] Y. Zong *et al.*, "Bayesian optimization with random forest," *Engineering Applications of Artificial Intelligence*, 2025.
[32] M. Shams *et al.*, "Grid search-based machine learning optimization," *Multimedia Tools and Applications*, 2024.
[33] S. Naveed and F. Akhtar, "AI-based cyber defense systems," 2025.
[34] M. Alnfiai, "Reinforcement learning for cybersecurity," *EURASIP Journal on Wireless Communications and Networking*, 2025.
[35] M. Umar *et al.*, "Deep learning IDS for edge computing," *Journal of Cloud Computing*, 2025.
[36] A. Shebl *et al.*, "CNN-based IDS models," *EURASIP Journal on Information Security*, 2024.
[37] K. Backhaus *et al.*, "Logistic regression," in *Multivariate Analysis*, Springer, 2025, pp. 269–358.
[38] R. Huang *et al.*, "Statistical analysis in machine learning," *Scientific Reports*, vol. 14, 2024.
[39] S. Singh *et al.*, "Entropy-based feature evaluation," *Vision*, 2024.
[40] A. Ajagbe *et al.*, "Machine learning models evaluation in IDS," *SN Computer Science*, vol. 5, no. 8, 2024.
[41] A. Ali *et al.*, "Machine learning in IDS survey," *Frontiers in Computer Science*, vol. 6, 2024.
[42] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[43] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[44] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.
[45] D. C. Montgomery, G. C. Runger, and N. F. Hubele, *Applied Statistics and Probability for Engineers*. Wiley, 2018.