Volume: 1, Issue: 1, Year: 2026

JASEM Journal Of Applied Science and Engineering Materials

Research Article

Received (First): 04.04.2026
Revision: 08.05.2026
Accepted: 21.06.2026

Network Intrusion Detection Using Machine Learning Algorithms on the NSL-KDD Dataset: A Comparative Experimental Study

Rasim Çekik[1]

[1] Department of Computer Engineering, Faculty of Engineering, Sirnak University, Sirnak, Türkiye

Abstract

The automatic detection of malicious activity in network traffic is one of the most critical components of today's cybersecurity infrastructure. This study presents a reproducible experimental comparison that evaluates the binary classification performance—specifically, the ability to distinguish between normal and attack traffic—of three different machine learning classifiers (Logistic Regression, Decision Tree, and Random Forest) on the NSL-KDD dataset, which is widely used in the intrusion detection literature. To this end, a common Python evaluation pipeline was established, consisting of steps such as converting categorical features (e.g., protocol type, service, flags) to numerical form, standardizing the features, performing stratified training/test splits, and evaluating the models using accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (ROC-AUC). The experimental results showed that the Random Forest model achieved the highest performance with 99.90% accuracy, a 99.89% F1-score, and an AUC of 1.00; this superiority was found to be statistically significant at $p < 0.001$ and was confirmed to be robust via 5-fold cross-validation. The results obtained quantitatively demonstrate the extent to which tree-based ensemble methods offer an advantage over linear models on NSL-KDD and show that the common evaluation protocol used can be easily extended to different datasets and classifiers.

Keywords: Intrusion Detection System, NSL-KDD, Machine Learning, Network Security, Random Forest, Comparative Evaluation

Citation: Çekik R. (2026). Network Intrusion Detection Using Machine Learning Algorithms on the NSL-KDD Dataset: A Comparative Experimental Study. *Journal Of Applied Science and Engineering Materials*, 1(1), 51–64. https://doi.org/10.71284/jasem202614

Corresponding Author:
Rasim Çekik
e-mail: rasimcekik@sirnak.edu.tr

JASEM Journal is a journal of the Academic Studies Group.
https://journals.academicianstudies.com/jasem

https://creativecommons.org/licenses/by/4.0/

Copyright & License

The copyright of works published in the journal is transferred by the authors to AXIS World: Journal of Applied Cross-Disciplinary Insights. All published works are made available as open access under the Creative Commons Attribution 4.0 International License (CC BY 4.0). This license permits the use, sharing, adaptation, distribution, and reproduction of the work in any medium or format, including for commercial purposes, provided that appropriate credit is given to the author(s) and the original publication, a link to the license is provided, and any changes made are clearly indicated.

1. Introduction

As the scale and complexity of computer networks have increased, the timely detection of malicious network activity—such as unauthorized access, denial-of-service attacks, and reconnaissance scanning—has become one of the fundamental requirements of enterprise security operations. The inadequacy of signature-based intrusion detection systems in detecting unknown attack types has led to a growing interest in machine learning-based approaches over the past two decades [1, 2]. A significant portion of the experimental work in this field is based on the KDD Cup 99 dataset, published in 1999, which was widely used in the attack detection literature for a long time [1]. However, a critical review by Tavallaee et al. [1] revealed that this dataset contained a high proportion of duplicate records in both the training and test sets, causing classifiers to become biased toward duplicate examples and, consequently, resulting in unreasonably high reported performance metrics. Developed to address these methodological issues, the NSL-KDD dataset has established itself in the literature as an improved version of KDD Cup 99 through the removal of duplicate records and the balancing of class distributions [1].

To date, a wide range of classical classifiers have been tested on NSL-KDD, ranging from linear models such as Logistic Regression [3] to individual tree-based methods such as Decision Trees [4] and ensemble approaches such as Random Forests [5, 6]. It has generally been reported that ensemble-based methods provide higher accuracy [7, 8]. However, in the majority of these studies, preprocessing steps—such as feature encoding, scaling, and training/test split—are applied in different ways, making it difficult to directly compare performance metrics reported in one study with those in another. Therefore, re-examining the same three classical classifiers under the same data preprocessing and evaluation steps still offers a valuable contribution toward bringing together these scattered findings in the literature on a consistent basis.

In this study, Logistic Regression, Decision Tree, and Random Forest classifiers were compared on the NSL-KDD dataset using a reproducible Python experimental workflow that applied the same preprocessing and evaluation steps. The findings were further analyzed using confusion matrices, ROC-AUC, and feature importance analyses to determine which network traffic features are discriminative in attack detection. The contributions of this study, which takes the form of a comparative evaluation, can be summarized as follows: (a) Preprocessing steps—including label binning, encoding of categorical features, standardization, and stratified training/testing splits—on the NSL-KDD dataset were combined into a single reproducible workflow; (b) Three existing classifiers were compared under the same conditions using five different metrics: accuracy, precision, sensitivity, F1-score, and AUC; (c) A feature importance analysis based on the Random Forest model is presented.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature. Section 3 introduces the NSL-KDD dataset. Section 4 explains the methods used in the comparison. Section 5 presents the experimental setup. Section 6 reports the experimental findings accompanied by tables and figures. Section 7 discusses the findings and concludes the study.

2. Related Works

A review of the literature on network intrusion detection indicates that the use of features extracted from raw network traffic in conjunction with classification algorithms achieves better performance in anomaly detection than traditional signature-based methods [9]. In line with this, the KDD Cup 99 dataset, which was developed and long regarded as the de facto standard of the field, has been extensively used in the literature. However, detailed analyses conducted on the dataset have shown that the excessive duplication rate, approximately 78% in the training set and around 75% in the test set, causes classifiers to become biased toward frequently observed records [1]. The NSL-KDD dataset removes these redundant records and performs subsampling according to difficulty levels, thus providing a more reliable and objective comparison ground for algorithm performance.

Revathi and Malathi [7] compared various algorithms such as SVM, Random Forest, k-NN, and decision trees on NSL-KDD and reported that ensemble-based methods generally provide higher accuracy compared to single classifiers. Furthermore, Dhanabal and Shantharajah [8] conducted a detailed analysis of the feature structure of NSL-KDD in terms of classification algorithms. Their study emphasized that proper encoding of categorical features such as protocol type, service, and flag is decisive on model performance. In addition, the Random Forest algorithm proposed by Breiman [5] is based on the principle of training numerous decision trees on bootstrap samples and combining them via majority voting, and it generally produces more robust results compared to linear models against overfitting in high-dimensional, noisy data. This characteristic of the algorithm makes it stand out in datasets such as NSL-KDD, which has 41 features and harbors complex interactions among classes.

In recent years, alongside classical machine learning algorithms, deep learning and large language model (LLM)-based approaches have been increasingly investigated on NSL-KDD and similar intrusion detection datasets. Kasula et al. [10] proposed a hybrid model combining Random Forest with LSTM networks to perform real-time anomaly detection in cloud environments, achieving a precision value of 94.78% on NSL-KDD and cloud log data. This study demonstrates that Random Forest continues to be used as a fundamental component even in complex hybrid architectures. Ali et al. [11], in their system called HuntGPT, combined a Random Forest classifier trained on the KDD99 dataset with explainable AI (XAI) techniques such as SHAP and LIME, along with a large language model-based chat interface, enabling security analysts to interpret detection results more easily. Ziems et al. [12], on the other hand, aimed to enhance the interpretability of decision tree-based models in network attack detection by converting the decision paths of decision tree models into natural language and generating explanations through large language models. These studies show that classical classifiers such as Logistic Regression, Decision Tree, and Random Forest still play a central role in the literature, both as standalone models and as components of more complex hybrid and LLM-supported architectures. This study aims to compare these three classical classifiers under a common and transparent evaluation protocol. To this end, the study presents a standardized preprocessing pipeline, an evaluation framework consisting of five different performance metrics, and a feature importance analysis, thereby providing a reproducible application infrastructure with a priority on interpretability and computational efficiency.

3. Dataset

In this study, the 'KDDTrain+' subset, which constitutes the training portion of the NSL-KDD dataset [13], was used for training network intrusion detection models. This dataset contains a total of 125,973 records, each consisting of 41 features, a class label, and a difficulty level score. The features in the dataset are grouped into four main categories based on their structural characteristics. The first of these, basic connection features, includes fundamental network parameters such as duration, protocol_type, service, flag, src_bytes, and dst_bytes. The second group, content features, encompasses attributes related to connection content and security, such as num_failed_logins, root_shell, and num_compromised. The third group, time-based traffic features, presents statistical data such as the number of connections within a specific time window (count), the number of connections to the same service (srv_count), and the SYN error rate (serror_rate), while the final group, host-based traffic features, describes host-oriented traffic patterns such as dst_host_count and dst_host_same_srv_rate.

Upon examining the data structure, it is observed that features such as protocol_type ('tcp', 'udp', 'icmp'), service, and flag take categorical values. To enable numerical-based classification algorithms to process these data effectively, these textual features need to be converted into appropriate numerical forms during the preprocessing stage. In the original dataset, the target variable consists of approximately 22 different classes representing various types of cyberattacks (e.g., neptune, smurf, satan, normal). However, in this study, in order to address the problem within a binary classification framework, the relevant labels were reduced to two fundamental classes: 0 for 'normal' records and 1 for 'attack'

records, created by consolidating all anomaly and attack types. The resulting class distribution, with 67,343 normal (53.5%) and 58,630 attack (46.5%) records, is shown in Figure 1.

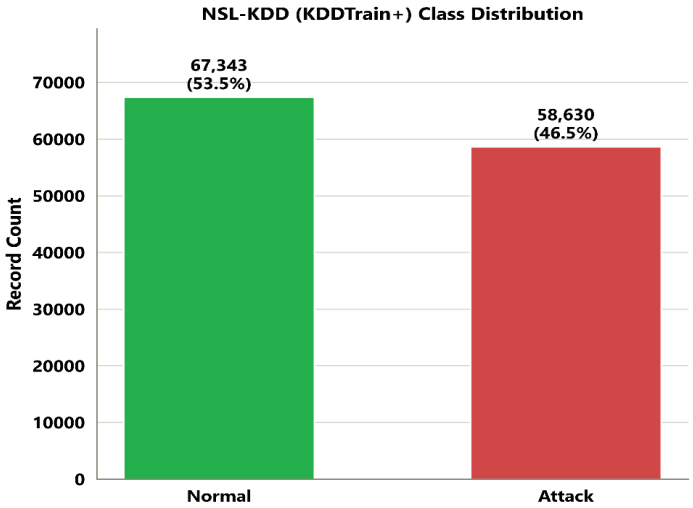

Figure 1. NSL-KDD (KDDTrain+) dataset class distribution

4. Materials and Methods

In this section, Logistic Regression, Decision Tree, and Random Forest algorithms are comparatively addressed on the NSL-KDD dataset under the same data preprocessing and evaluation steps. The preprocessing steps applied sequentially, the classification models included in the comparison, and the evaluation metrics used are described.

4.1. Preprocessing

The following preprocessing steps were applied sequentially on the raw dataset: (1) The label column representing the target variable was reduced in accordance with the binary classification architecture. (2) The categorical features in the dataset, namely protocol_type, service, and flag, were subjected to integer encoding to be converted into numerical format. (3) The transformed dataset was split into 70% training and 30% test sets using a stratified method to preserve the overall distribution ratio of the original classes. (4) All features were standardized using StandardScaler with the mean and standard deviation computed from the training set. The scaler was computed solely on the training data, and only the transformation was applied to the test data; this approach was preferred to prevent information leakage from the test set into the training process.

In the standardization step, a z-score transformation was applied for each feature value x, as shown in Equation (1):

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ and σ are the mean and standard deviation of the respective feature, computed solely on the training set [14].

4.2. Classification Models

In the classification phase, three different classifier families were compared: (i) Logistic Regression, based on linear decision boundaries and considered as the baseline model due to its high interpretability. (ii) Decision Tree, a single tree model that can learn non-linear decision boundaries by recursively partitioning the feature space. (iii) Random Forest, which has an ensemble learning-based architecture

where numerous decision trees are trained on bootstrap samples and combined via majority voting. The parameters used for running the methods are given in Table 1. All three models were trained and evaluated on the same training/test split and the same scaled feature set.

Table 1. Models used for comparison and the parameters employed in their implementation

Classifier	Parameter	Value	Description
Logistic Regression	max_iter	1000	Maximum number of iterations allowed for the optimization solver to converge
Decision Tree	criterion	gini (default)	Impurity measure used to assess split quality
Decision Tree	random_state	42	Fixing internal randomness sources of the tree (split selection)
Random Forest	n_estimators	100	Number of decision trees used in the ensemble
Random Forest	random_state	42	Fixing bootstrap sampling and feature subset selection
General (all models)	train/test split random_state	42	Fixed to ensure reproducibility of the data split

In the Logistic Regression model, given the weight vector w and the bias term b, the probability that a sample belongs to the attack class ($y = 1$) is calculated using the sigmoid function, as given in Equation (2):

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w \cdot x + b)}} \tag{2}$$

The model parameters were optimized on the training set by minimizing the binary cross-entropy (log-loss) cost function, defined in Equation (3) [15]:

$$J(w, b) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)] \tag{3}$$

Here, N denotes the number of training samples, yi represents the true label, and $\hat{p}_i$ denotes the predicted probability produced by the model for sample i.

In the Decision Tree model, each split is determined by selecting the feature and threshold value that maximize the reduction in class impurity at a node. For a node t, the Gini impurity is defined as in Equation (4):

$$\text{Gini}(t) = 1 - \sum_{k=1}^{K} p_k^2 \tag{4}$$

Here, K denotes the number of classes ($K = 2$ in this study), and pk represents the proportion of samples belonging to class k at node t. The tree is constructed recursively by selecting splits that maximize the information gain, defined as the weighted impurity reduction between a parent node and its child nodes [6].

The Random Forest model consists of an ensemble of B decision trees ($B = 100$ in this study), each trained independently on a bootstrap sample generated through random sampling with replacement from the training data. The final class prediction is determined by majority voting among the individual trees, as given in Equation (5):

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_B(x)\} \quad (5)$$

where $hb(x)$ denotes the prediction of the b-th decision tree for sample x. By aggregating the predictions of multiple trees, Random Forest reduces variance and improves generalization performance compared with a single decision tree [5].

4.3. Evaluation Metrics

The performance of the trained models was assessed using five complementary metrics: accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC). To prevent potentially misleading conclusions arising from the class imbalance commonly encountered in cybersecurity datasets, the evaluation was not based solely on overall accuracy. Instead, precision, recall, and F1-score values were also analyzed on a class-specific basis.

Furthermore, the confusion matrix of each model was examined to provide a detailed assessment of the rates of false positives (normal traffic incorrectly classified as attacks) and false negatives (attack traffic incorrectly classified as normal), the latter of which may lead to critical security vulnerabilities. Based on the values of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) obtained from the confusion matrix, the evaluation metrics were mathematically defined as in Equations (6)–(9):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (7)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (8)$$

$$F_1\text{-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (9)$$

The Receiver Operating Characteristic (ROC) curve, which reflects the discriminative capability of a classification model, graphically illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) across different decision thresholds. The Area Under the Curve (AUC) metric, representing the area beneath the ROC curve, provides a threshold-independent measure of classification performance by quantifying the probability that a randomly selected attack instance is assigned a higher attack score than a randomly selected normal traffic instance [16].

5. Experimental Setup

The experiments were conducted in a Python 3.13 environment using the scikit-learn, pandas, NumPy, Matplotlib, and Seaborn libraries [14]. As the benchmark dataset, the training portion of the NSL-KDD dataset proposed by Tavallaee et al. [1] was utilized. The dataset file was obtained from the publicly accessible defcom17/NSL_KDD GitHub repository hosting the NSL-KDD dataset [13].

To ensure reproducibility, a fixed random seed was employed in all procedures involving randomness, including the training–test split and the internal randomization processes of the Decision Tree and Random Forest models. The random seed was fixed at 42 throughout all experiments. The hyperparameters of the models were intentionally restricted to commonly used default and baseline values reported in the

literature, without performing extensive optimization. The parameter settings used in the experiments are summarized in Table 1. This choice was motivated by the primary objective of the study, which is to reveal the relative differences among model families under a common preprocessing framework. Advanced hyperparameter optimization is considered as a direction for future research.

6. Experimental Results and Findings

This section presents the comparative performance results of the three classifiers evaluated on the NSL-KDD test set. The reported results include a performance table containing accuracy, precision, recall, F1-score, and AUC values, class-wise classification reports, a comparative model performance chart, confusion matrices, ROC curves, and a feature importance analysis based on the Random Forest model.

6.1. Overall Performance Comparison

Table 2 summarizes the accuracy, precision, recall, F1-score, AUC, and Average Precision (AP) values achieved by the three classifiers on the test set. Figure 2 provides a visual comparison of the models according to these key performance metrics.

According to the obtained results, the Random Forest model achieved the best overall performance among the three classifiers, with 99.90% accuracy, 99.89% F1-score, 1.00 AUC, and 1.00 AP. The Decision Tree model exhibited a performance level that was nearly indistinguishable from that of Random Forest, achieving 99.81% accuracy and 99.80% F1-score. In contrast, the Logistic Regression model, which relies on a linear decision boundary, lagged behind the two tree-based approaches, with 95.35% accuracy and 94.95% F1-score. Furthermore, the recall value of the Logistic Regression model (93.96%) was the lowest among the three models. This finding indicates that Logistic Regression was more prone to misclassifying a portion of attack instances as normal traffic compared with the Decision Tree and Random Forest models. Consequently, the tree-based methods demonstrated a superior capability for capturing the nonlinear relationships and complex patterns inherent in network intrusion detection data.

Table 2. Performance comparison of the models on the test set

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC	AP
Logistic Regression	95.3500	95.9700	93.9600	94.9500	0.9919	0.9904
Decision Tree	99.8100	99.8200	99.7700	99.8000	0.9981	0.9970
Random Forest	99.9000	99.9500	99.8400	99.8900	1.0000	1.0000

The superior performance of the two tree-based models, Decision Tree and Random Forest, compared with the linear model suggests that the relationships among the features in the NSL-KDD dataset are largely nonlinear in nature. By recursively partitioning the feature space and learning complex decision boundaries, tree-based methods are able to capture such interactions more effectively than the linear decision boundary employed by Logistic Regression. The fact that Random Forest achieved marginally better results than the single Decision Tree across all evaluation metrics further indicates that training multiple trees on bootstrap samples and aggregating their predictions through majority voting reduces the tendency of an individual tree to overfit, thereby decreasing model variance and improving generalization performance.

The observation that all models achieved AUC and AP values above 0.99 indicates that the binary classification problem (normal vs. attack traffic) is highly separable on the NSL-KDD test set, regardless of the selected decision threshold. However, this finding should be interpreted in conjunction with the dataset limitations discussed in Section 3.

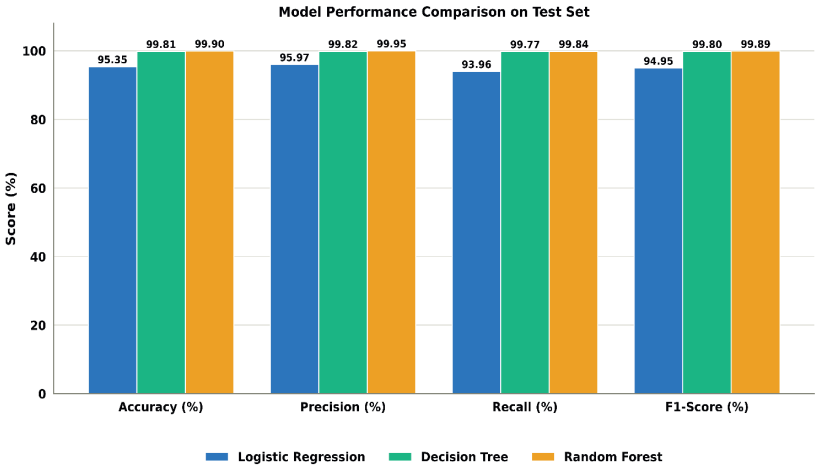

Figure 2. Comparison of the models based on accuracy, precision, recall, and F1-score

6.2. Class-Wise Performance

Table 3 presents the precision, recall, and F1-score values separately for the normal and attack classes for each classifier. Such a class-wise analysis is particularly important in intrusion detection applications because the recall value of the attack class has significant operational implications. A low recall value indicates that a portion of actual attack instances is incorrectly classified as normal traffic, resulting in false negatives. These errors are generally considered more critical than false positives because undetected attacks may lead to severe security breaches.

From this perspective, the Logistic Regression model achieved an attack-class recall of 93.96%, which is noticeably lower than the values obtained by the other two models. This finding suggests that the linear decision boundary employed by Logistic Regression fails to adequately separate some attack instances from normal traffic. In contrast, the Decision Tree and Random Forest models achieved attack-class recall values of 99.77% and 99.84%, respectively, thereby minimizing the number of missed attacks.

Table 3. Class-wise precision, recall, and F1-score results

Model	Class	Precision (%)	Recall (%)	F1-Score (%)	Support
Logistic Regression	Normal	94.83	96.56	95.69	20,203
Logistic Regression	Attack	95.97	93.96	94.95	17,589
Decision Tree	Normal	99.80	99.84	99.82	20,203
Decision Tree	Attack	99.82	99.77	99.80	17,589
Random Forest	Normal	99.86	99.96	99.91	20,203
Random Forest	Attack	99.95	99.84	99.89	17,589

The confusion matrices presented in Figure 3 enable a comparison of the three models in terms of their false positive and false negative rates. Regarding false negatives, out of the 17,589 attack records, Logistic Regression incorrectly classified 1,063 instances as normal traffic, while Decision Tree and Random Forest misclassified only 40 and 28 attack instances, respectively. In terms of false positives, among the 20,203 normal records, Logistic Regression incorrectly labeled 694 instances as attacks, whereas Decision Tree and Random Forest produced only 32 and 9 false alarms, respectively.

These findings demonstrate that the tree-based models maintain substantially lower false-negative rates

than the linear model, which is particularly important from an operational cybersecurity perspective. Since false negatives correspond to undetected attacks, minimizing such errors is crucial for maintaining the effectiveness of an intrusion detection system. Among the evaluated classifiers, Random Forest outperformed not only Logistic Regression but also the single Decision Tree model by achieving the lowest numbers of both false negatives and false positives. Consequently, Random Forest emerged as the most reliable model, simultaneously minimizing missed attacks and false alarms while providing the highest overall classification performance.

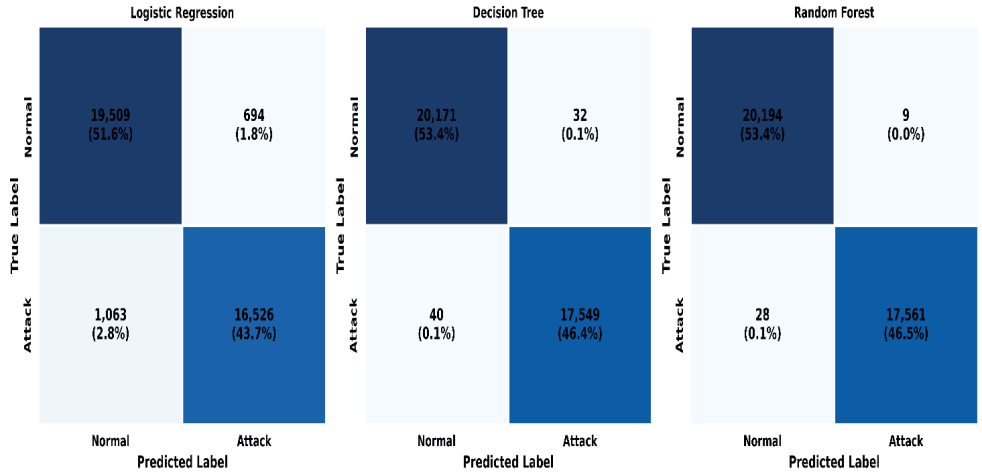

Figure 3. Confusion matrices of the evaluated models

Figure 4 presents the ROC curves and corresponding AUC values for the three classification models. The closer the ROC curve is to the upper-left corner of the plot and the closer its AUC value is to 1, the greater the model's ability to discriminate between classes. The ROC curves of the Decision Tree and Random Forest models rise almost vertically along the y-axis and nearly overlap throughout the graph, achieving AUC values of 0.9981 and 1.0000, respectively. These results indicate that both models possess an excellent capability to distinguish between normal and attack traffic.

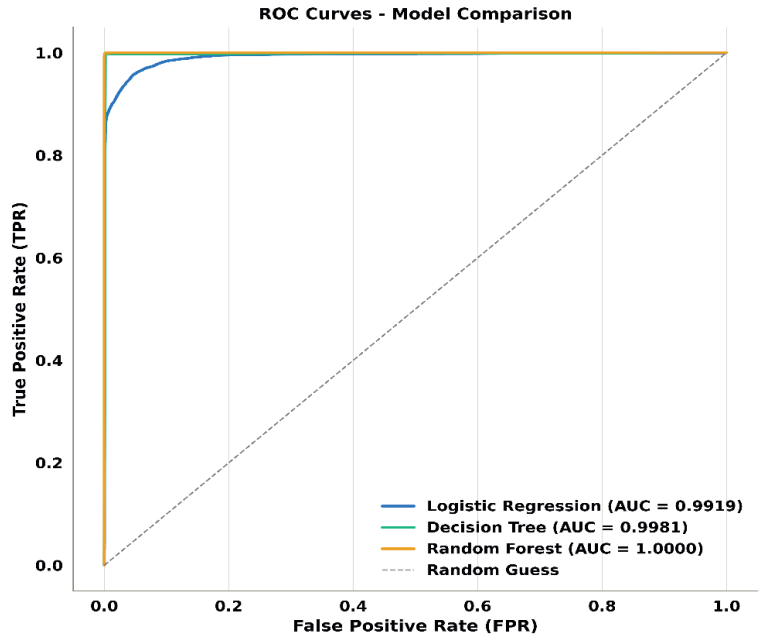

Figure 4. ROC curves and AUC comparison

In contrast, the ROC curve of the Logistic Regression model, with an AUC value of 0.9919, exhibits a noticeably flatter trajectory, particularly at low false-positive rates. This behavior suggests that the linear model produces more classification errors than the tree-based models when the decision threshold is set to a lower value. Although Logistic Regression still demonstrates strong discriminative performance overall, its lower AUC value reflects a reduced ability to capture the complex nonlinear relationships present in the NSL-KDD dataset compared with Decision Tree and Random Forest approaches.

Figure 5 illustrates the precision–recall (PR) curves and the corresponding Average Precision (AP) values for the three classification models. Since the class distribution in the dataset is moderately imbalanced, PR curves provide a complementary perspective to ROC curves by emphasizing the trade-off between precision and recall, particularly for the positive (attack) class.

The Random Forest model, with an AP value of 1.0000, and the Decision Tree model, with an AP value of 0.9970, maintain consistently high precision across nearly the entire recall range. This behavior indicates that both models are capable of identifying attack instances with very few false positives, even as recall approaches its maximum value.

In contrast, the Logistic Regression model achieved an AP value of 0.9904. Its precision–recall curve begins to decline noticeably once recall exceeds approximately 0.85, indicating a rapid decrease in precision at higher recall levels. Consequently, when the decision threshold is relaxed to capture a greater proportion of attack instances, the false-positive rate of Logistic Regression increases more rapidly than that of the tree-based models. This finding further highlights the superior ability of Decision Tree and Random Forest models to preserve a favorable balance between precision and recall across a wide range of operating conditions.

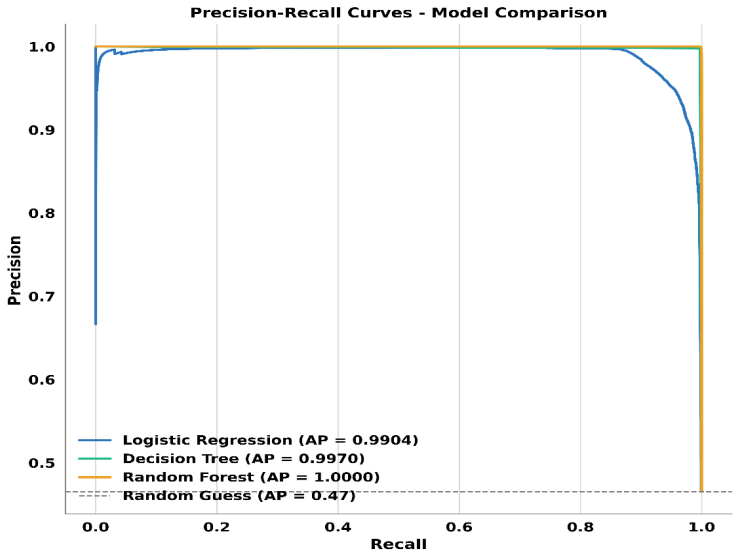

Figure 5. Precision–recall curves of the evaluated models

To assess the extent to which the results obtained from a single train–test split were affected by random variation, 5-fold cross-validation was performed. According to the results summarized in Table 4 and visualized through box plots in Figure 6, the Random Forest model demonstrated the highest average performance and the lowest variance, achieving a mean accuracy of 99.86% with a standard deviation of 0.01%, thereby confirming the model's stability and robustness. In contrast, the Logistic Regression model exhibited both lower average performance and relatively higher variability, with a mean accuracy of 95.46% and a standard deviation of 0.14%.

These findings support the conclusion that the performance differences reported in Table 2 are not merely the result of random sampling effects associated with a particular train–test split, but rather reflect systematic differences in predictive capability among the evaluated models. The consistently high mean scores and low standard deviations observed for the tree-based methods indicate that their superior performance remains stable across different data partitions.

Table 4. Results of 5-fold cross-validation

Model	Mean Accuracy (%)	Accuracy Std. (%)	Mean F1-Score (%)	F1-Score Std. (%)	Mean AUC	AUC Std.
Logistic Regression	95.4600	0.1400	95.0700	0.1500	0.9917	0.0002
Decision Tree	99.7500	0.0500	99.7400	0.0600	0.9976	0.0005
Random Forest	99.8600	0.0100	99.8500	0.0100	1.0000	0.0000

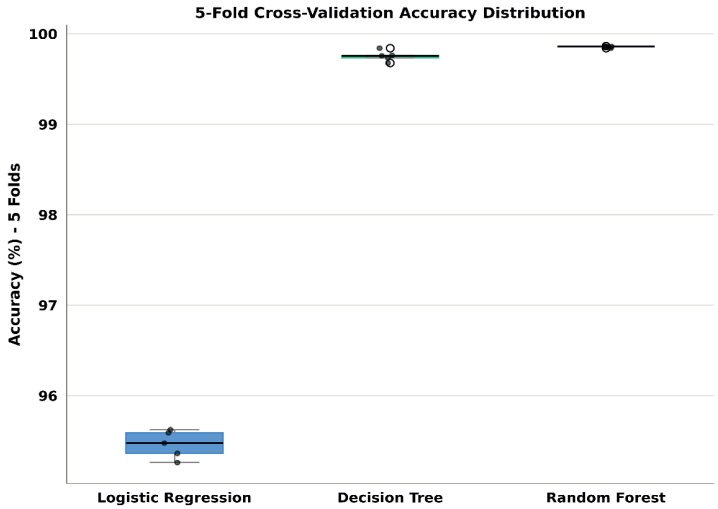

Figure 6. Cross-validation accuracy distributions

6.3. Statistical Significance Analysis

To determine whether the observed performance differences among the evaluated models were statistically significant, the McNemar test, a widely used method for pairwise comparison of binary classifiers, was applied. The results are presented in Table 5. The calculated p-values for all model pairs were found to be well below the significance threshold of 0.05, specifically at the $p < 0.001$ level, indicating that the observed differences in classification performance are not attributable to random chance but are statistically significant.

In particular, the differences between Logistic Regression and the tree-based models (Decision Tree and Random Forest) exhibited extremely strong statistical significance, with $\chi 2$ values exceeding 1600 and corresponding $p < 0.001$. Furthermore, the performance difference between Decision Tree and Random Forest was also found to be statistically significant, yielding $\chi 2 = 18.35$ and $p = 1.8 \times 10{-}5$. This result confirms that the performance gain achieved through the ensemble approach of Random Forest is genuine and cannot be attributed merely to measurement error or random variation.

Table 5. McNemar test results

Model A	Model B	Correct Only by A	Correct Only by B	Chi-Square (χ^2)	p-value	Significant ($p < 0.05$)
Logistic Regression	Decision Tree	39	1,724	1608.54	< 0.001	Yes
Logistic Regression	Random Forest	11	1,731	1696.30	< 0.001	Yes
Decision Tree	Random Forest	14	49	18.35	1.8×10^{-5}	Yes

6.4. Feature Importance Analysis

Figure 7 presents the fifteen most important features according to the Random Forest model. This analysis identifies the network traffic attributes that contribute most strongly to distinguishing between attack and normal traffic. Consequently, it not only enhances model interpretability but also provides valuable guidance for future feature selection studies.

The features with the highest importance scores are src_bytes and dst_bytes, followed by dst_host_srv_count, same_srv_rate, dst_host_same_srv_rate, and flag. These findings suggest that attack traffic is primarily differentiated from normal traffic through characteristics related to the volume of data transferred per connection and patterns of service access directed toward target hosts.

This observation is consistent with the nature of many cyberattacks. In particular, probing (probe) and denial-of-service (DoS) attacks typically generate either unusually high or unusually low byte-transfer volumes, as well as abnormally frequent and repetitive service requests directed at the same target host. As a result, traffic-volume-related features such as src_bytes and dst_bytes, together with service-similarity indicators such as same_srv_rate and dst_host_same_srv_rate, emerge as highly discriminative attributes for intrusion detection. The prominence of these features demonstrates that traffic behavior characteristics play a crucial role in distinguishing malicious activities from legitimate network operations.

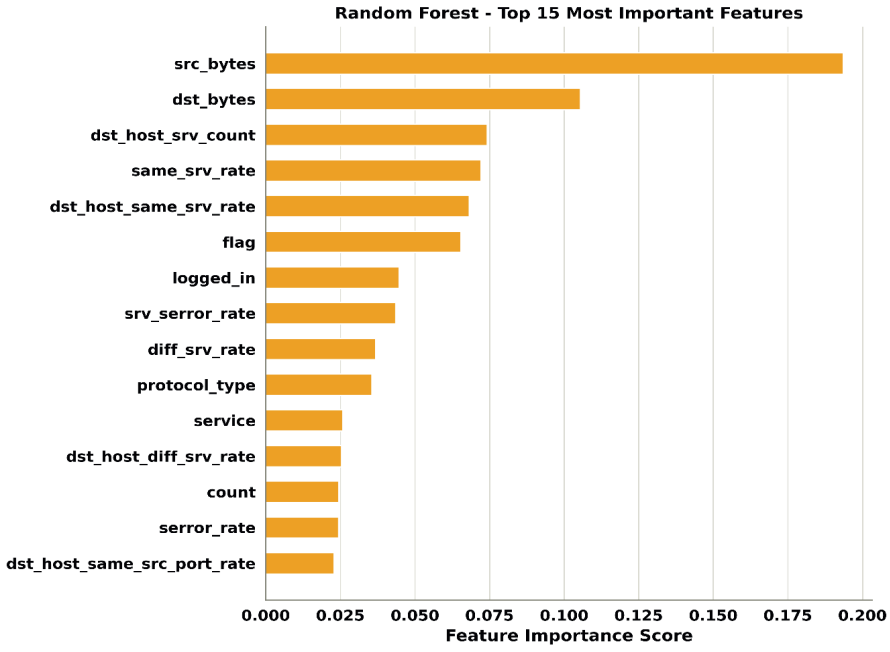

Figure 7. Random Forest feature importance ranking

7. Discussion and Conclusions

When the findings are considered collectively, the superiority of tree-based models over the linear model does not appear to be coincidental. This advantage is consistently observed across summary metrics such as accuracy and F1-score. The error distribution reflected in the confusion matrices also supports the same conclusion. The McNemar test confirms that this difference is statistically significant, while cross-validation demonstrates that it is not attributable to sampling randomness. This consistency indicates that the attack/normal classification problem in the NSL-KDD dataset contains nonlinear interactions among features that cannot be captured by the linear decision boundary of Logistic Regression but can be readily learned by tree-based models through recursive partitioning.

The nature of the differences among the models is as important as their magnitude. Even when two models achieve high accuracy, the distribution of error types may lead to different operational outcomes. In an intrusion detection system, a missed attack generally carries a higher cost than a false alarm. The false-negative rate of Logistic Regression is considerably higher than those of the tree-based models. This suggests that, despite its high overall accuracy, the model may not be considered sufficient as a standalone solution for operational deployment. Random Forest, on the other hand, maintains both false-negative and false-positive rates at low levels. In this regard, it emerges as a more balanced candidate from a security perspective, as it keeps both types of errors under control.

It is noteworthy that features related to the volume of data transferred per connection emerged as the most important variables in the feature importance analysis. This finding indicates that the model learned the distinction based not on random or superficial patterns, but on signals that can be associated with attack behavior. Therefore, the high performance achieved can be considered to be largely driven by meaningful learning.

On the other hand, the fact that all models achieved AUC and AP values above 0.99 should also be interpreted as a cautionary signal. Such high and closely clustered values may suggest that the NSL-KDD dataset is relatively separable compared with real-world network traffic. Consequently, whether the performance ranking observed in this study would be preserved to the same extent on noisier and more imbalanced real-world datasets remains an open question. In this context, it is recommended that: (i) additional validation be performed on the separate KDDTest+ dataset to provide a more robust assessment of model generalization capability; (ii) systematic hyperparameter optimization and the comparison of deep learning-based architectures be conducted under the same experimental protocol; (iii) classification be extended beyond the binary setting to perform a multiclass analysis of DoS, Probe, R2L, and U2R attack categories; and (iv) model complexity be reduced through feature selection based on the obtained feature importance rankings.

References

[1] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications; 2009 Jul 8-10; Ottawa, Canada. p. 1-6. doi:10.1109/CISDA.2009.5356528

[2] Lee W, Stolfo SJ. Data mining approaches for intrusion detection. In: Proceedings of the 7th USENIX Security Symposium; 1998 Jan 26-29; San Antonio, TX, USA.

[3] Cox DR. The regression analysis of binary sequences. *J R Stat Soc Series B Stat Methodol*. 1958;20(2):215-232. doi:10.1111/j.2517-6161.1958.tb00292.x

[4] Quinlan JR. Induction of decision trees. *Mach Learn*. 1986;1(1):81-106. doi:10.1007/BF00116251

[5] Breiman L. Random forests. *Mach Learn*. 2001;45(1):5-32. doi:10.1023/A:1010933404324

[6] Breiman L, Friedman J, Olshen RA, Stone CJ. Classification and Regression Trees. Boca Raton: Chapman and Hall/CRC; 2017. doi:10.1201/9781315139470

[7] Revathi S, Malathi A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *Int J Eng Res Technol*. 2013;2(12):1848-1853.

[8] Dhanabal L, Shantharajah SP. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int J Adv Res Comput Commun Eng*. 2015;4(6):446-452.

[9] Lee W, Stolfo SJ. A framework for constructing features and models for intrusion detection systems. ACM Trans Inf Syst Secur. 2000;3(4):227-261. doi:10.1145/382912.382914

[10] Kasula VK, Yadulla AR, Konda B, Yenugula M. Fortifying cloud environments against data breaches: a novel AI-driven security framework. *World J Adv Res Rev*. 2024;24:1613-1626.

[11] Ali T, Kostakos P, Sheikhi S. HuntGPT: integrating machine learning-based anomaly detection and explainable AI with large language models (LLMs). *Telecom*. 2026;7(3):73.

[12] Ziems N, Liu G, Flanagan J, Jiang M. Explaining tree model decisions in natural language for network intrusion detection. arXiv [Preprint]. 2023. arXiv:2310.19658. doi:10.48550/arXiv.2310.19658

[13] Canadian Institute for Cybersecurity. NSL-KDD dataset [Internet]. Fredericton: University of New Brunswick; 2009. Available from: https://www.unb.ca/cic/datasets/nsl.html

[14] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. *J Mach Learn Res*. 2011;12:2825-2830.

[15] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York: Springer; 2009. doi:10.1007/978-0-387-84858-7

[16] Fawcett T. An introduction to ROC analysis. *Pattern Recognit Lett*. 2006;27(8):861-874. doi:10.1016/j.patrec.2005.10.010