

Machine Learning-Based Intelligent Network Intrusion Detection System Using Random Forest Classification

Shravani Godase, *School of Computational Sciences, JSPM University, Pune, Maharashtra, India.*

Rushikesh Garkal, *School of Computational Sciences, JSPM University, Pune, Maharashtra, India.*

Dr. R.S. Deshpande, *School of Computational Sciences, JSPM University, Pune, Maharashtra, India.*

Dr. Nirupama Patodare, *School of Computational Sciences, JSPM University, Pune, Maharashtra, India.*

Manuscript Received: June 05, 2026; Revised: June 30, 2026; Published: July 07, 2026

Abstract: The fast-paced development of communication infrastructures and integrated cloud systems poses an increased risk of being exposed to any form of cyber threats. The traditional IDS that relies on signatures has certain limitations when it comes to detecting new or zero-day exploits. In order to overcome this crucial limitation, this work proposes a machine learning network security solution that involves the use of the Random Forest classifier. For testing purposes, the NSL-KDD data set is used to detect several types of attacks, namely, DoS, U2R, R2L, and probes. The entire methodological framework consists of various stages: data pre-processing, labeling, scaling, feature selection, and testing against other supervised machine learning approaches such as Naïve Bayes, Decision Tree, K-Nearest Neighbors, and Logistic Regression. It has been proven empirically that the proposed Random Forest model outperforms other algorithms, with its superior accuracy equalling 98.3%. Finally, the presented approach allows for developing a system that produces the fewest possible false positives, offers scalability, and can accurately track multiple classes of attacks.

Keywords: Network Intrusion Detection System, Machine Learning, Random Forest, Cybersecurity, NSL-KDD Dataset, Intrusion Detection, Supervised Learning.

1. Introduction

Rapidly evolving state-of-the-art telecommunication networks and utility cloud-based services have resulted in a parallel increase in various types of network security attacks [1]. Modern networking systems face continuous threats from malicious activities. In this regard, Network Intrusion Detection Systems play the role of critical safety mechanisms that analyze traffic and detect any unauthorized activity [2]. However, current methods rely heavily on predefined rule sets and/or signatures. For this reason, such detection systems fail when confronted with zero-day vulnerabilities due to the absence of relevant data in the preset database of attack scenarios. Additionally, conventional models lack inherent dynamism and require manual reconfiguration while also resulting in numerous false-positive alarms [3]. With an aim to overcome the shortcomings of conventional perimeter tools, there has been an advent of intelligent machine learning architecture that serves as a potent set of techniques to counteract dynamic cyberattacks [4]. The intelligent data models use advanced self-learning algorithms that can effectively distinguish between legitimate network flows and malicious traffic patterns. The intelligent framework used in this study relies on the standard NSL-KDD dataset as the main baseline dataset against which the intelligent machine learning NIDS is tested and optimized. As the improved version of the standard KDD Cup 1999 dataset, the NSL-KDD database effectively addresses critical issues of imbalanced data distribution and the presence of redundant samples, making it a fair ground for performance measurement [5], [6]. The NSL-KDD database contains numerous types of attacks among which DoS, R2L, U2R, and Probing attacks prevail.

The following supervised machine learning methods were used for a comprehensive analysis of the potential and classification capabilities of different types of statistical models: Naïve Bayes, Decision Trees, K-Nearest Neighbor (KNN), Logistic Regression, and Random Forest [7], [8]. The results indicate that the Random Forest model provided the most effective detection metrics, which confirms the reliability of Random Forest when addressing different

challenges associated with diverse cyber threats [9]. The main outcomes of the present research paper include the following points:

- Development of an intelligent platform designed for identifying various cyber threats.
- Comparative analysis of the effectiveness of well-known supervised learning algorithms.
- Systemic investigation of feature importance rankings in order to detect crucial connection variables.
- Performance testing using recognized diagnostic parameters.
- Creation of a highly efficient and scalable cyber threat management solution.

A substantial number of modern research papers concentrate on attaining optimal results in terms of abstract precision metrics while paying little attention to important aspects like extreme scarcity of classes, semantic interpretability, and constraints on computational speed. It is essential to note that very localized attack types, namely U2R and R2L attacks, prove to be particularly challenging to detect due to the insufficiency of training samples. In order to address this challenge, this paper proposes a reliable and easily scalable detection system which maintains the perfect balance of performance, resource usage, and semantic interpretability.

2. Literature Review

In the last decade, security has seen considerable progress, to the extent that digital security experts have turned to intelligent intrusion tracking systems equipped with machine learning and deep learning algorithms to detect any anomalies [10]. Modern research interest lies in the development of intelligent data-oriented security systems that are capable of developing along with new threats [11]. The recent literature shows the significance of data trait filters that help execute models effectively within IoT environments. For example, Li et al. designed an analytics architecture incorporating data selection and extraction algorithms to make classification procedures easier [12]. The results of the analysis proved that data feature filtration is essential to ensure minimal processing load alongside high levels of performance, thus providing an excellent solution for IoT systems [12]. On the other hand, Alomari et al. designed a light-weighted predictive system that was specifically designed for constrained hardware resources within IoT nodes [13]. Data trait filtration allowed decreasing the data dimensionality while still maintaining the performance metrics [13].

In pursuit of improved performance, the study conducted by Sharma and Verma compared different classifiers including Logistic Regression, KNN, Decision Trees, and Random Forest using the NSL-KDD dataset [14]. It was experimentally validated that ensemble classifier approaches, particularly Random Forest, offer much better classification outcomes along with lower false alarms than individual classical classifiers [7], [8]. Studies suggested by Ahmed and Khan considered the Decision Tree classifier approach as optimal for effective network threat recognition during its early stages [15]. Although demonstrating high efficiency in classifying attacks, their classifier proved to be highly prone to overfitting in the case of large data sets with a significant amount of network noise [15]. Furthermore, Vasquez investigated the possibility of using multiclass anomaly detection using traditional classifiers on the NSL-KDD dataset [16]. These findings showed that the use of the ensemble approach, i.e., Random Forest, significantly surpassed basic classifiers because of its better generalization capabilities [17].

Although intrusion detection research has already received much attention in cloud computing frameworks, edge computing devices, and industry, many open issues still exist. One of the major challenges is that state-of-the-art research work in this area concentrates mainly on achieving nominal accuracy levels while ignoring some basic architectural issues:

- Class imbalance in relation to rare attack vectors, like U2R and R2L [18].
- Black box vulnerabilities of the system and lack of explainability of features used [19].
- False positives that occur in very unstable network topologies [20].
- Extremely heavy computational demands of deep neural models.

Hence, it becomes necessary to design an active model where both high accuracy and low computation cost are ensured. The method described in this research paper directly deals with these constraints by employing a Network Intrusion Detection System based on Random Forest that considers attribute significance, multiclass classification, and comprehensive benchmarking across various supervised algorithms.

3. Proposed Methodology

The NIDS model used in this implementation makes use of supervised machine learning techniques in detecting and blocking malicious packets. The NIDS system architecture is developed with the objective of achieving high accuracy in threat categorization. Fig. 1 depicts the data flow and other operations involved in the said methodology.

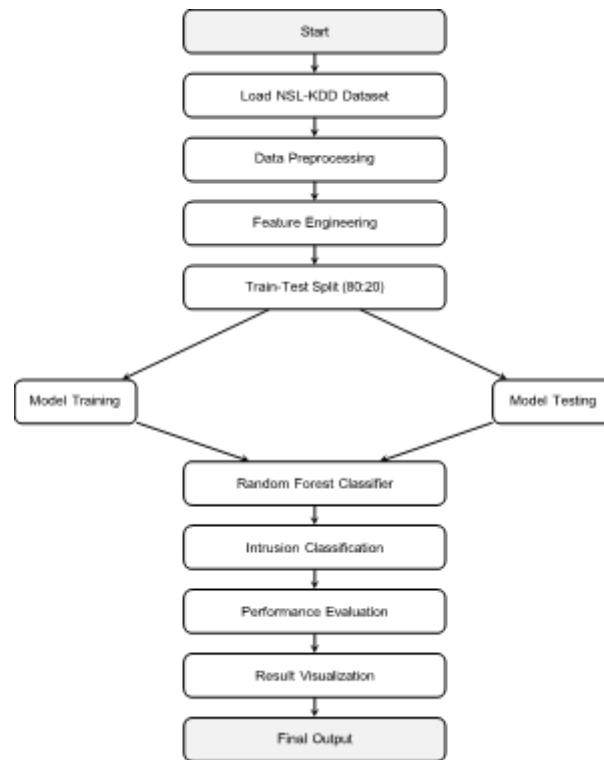


Fig. 1. Overall Workflow of Proposed Intrusion Detection System

3.1 Dataset Description

The NSL-KDD Data collection is a more advanced form of the KDDCUP'99 dataset designed for dealing with the common problem of instance duplication and class imbalance [5]. The NSL-KDD dataset incorporates both normal network interactions along with various forms of attacks. The log profile consists of 41 mathematical traffic features along with the corresponding category which indicates the status of the interaction. The interaction features considered include technical headers such as protocol type, service type, duration, source bytes, destination bytes, and connection flags.

3.2 Data Preprocessing

Systematic data cleaning tasks are extremely important in developing a robust intrusion detection system. Unstructured data collected from raw transmission logs can consist of missing entries, high levels of noise, inconsistencies, and categorical features that cannot be computed via ML algorithms. Consequently, several data cleaning techniques are applied to the dataset:

1. **Duplicate Entry Removal:** All duplicate records found in the database are deleted to prevent validation bias in the model training process.
2. **Addressing Incomplete Dataset Entries:** Inconsistencies such as blank indices in the dataset are detected and then solved using statistical imputations.
3. **Conversion of Non-Numeric Attributes:** There are several non-numeric categorical features in the NSL-KDD dataset. Label encoding is performed to translate categorical features to numeric variables.
4. **Normalization of Attributes:** The features representing network connectivity display highly contrasting scalar values and range spectrums. For standardization of these discrepancies, Min-Max normalization is carried out to normalize all ranges within the same scale. Min-Max Normalization can be defined as:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

5. **Division of the Data Set:** After the normalization process, the entire data set is divided into separate training and test sets in an 80:20 split ratio. The 80 percent data subset is used for training the algorithm while the 20 percent serves as a separate data subset for evaluating generalization performance.

3.3 Feature Engineering

Feature Engineering takes place in order to discover the best possible variables that affect classification performance. An isomorphic attribute reduction mechanism enhances the efficiency of computations in addition to reducing processing costs [21]. The use of a Random Forest Feature Importance method helps evaluate how much each variable affects threat classification. The following is the equation used for calculating feature contribution weights:

$$Importance(f_i) = \sum_{t=1} \Delta I_t(f_i) \quad (2)$$

3.4 Machine Learning Classifiers

Various machine learning techniques have been used and compared in terms of efficiency for classification of the network traffic vectors.

1) Logistic Regression: This is a basic probabilistic model where the binary or multi-class outputs are determined via the logistic response equation:

$$P(Y = 1|X) = \frac{1}{1 + e^{-z}} \quad (3)$$

2) K-Nearest Neighbors (KNN): The KNN model makes predictions through measuring the geometric closeness based on the Euclidean distance between the data vector and its closest neighbor:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (4)$$

3) Naïve Bayes: The Naïve Bayes algorithm operates according to Bayes' theorem based on the naive assumption that all variables within the sample are independent:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \quad (5)$$

4) Decision Tree: The Decision Tree algorithm splits datasets by creating hierarchical decision nodes in the form of a tree on the basis of reducing Gini Impurity:

$$Gini = 1 - \sum_{i=1}^n p_i^2 \quad (6)$$

with p_i standing for the probability allocation to class i .

5) Random Forest: Random Forest serves as a more sophisticated framework for ensemble learning through the use of decision trees to reduce prediction variance. The prediction is made as follows:

$$\hat{Y} = mode\{T_1(X), T_2(X), \dots, T_B(X)\} \quad (7)$$

4. System Architecture

The suggested architecture of NIDS is based on an organized multi-layer machine learning topology. The system involves a series of processes such as data collection, preprocessing, feature engineering, machine learning modeling, and analytical visualization stages to track threats on the network. Figure 2 shows the layered data flow architecture of the above-stated NIDS framework.

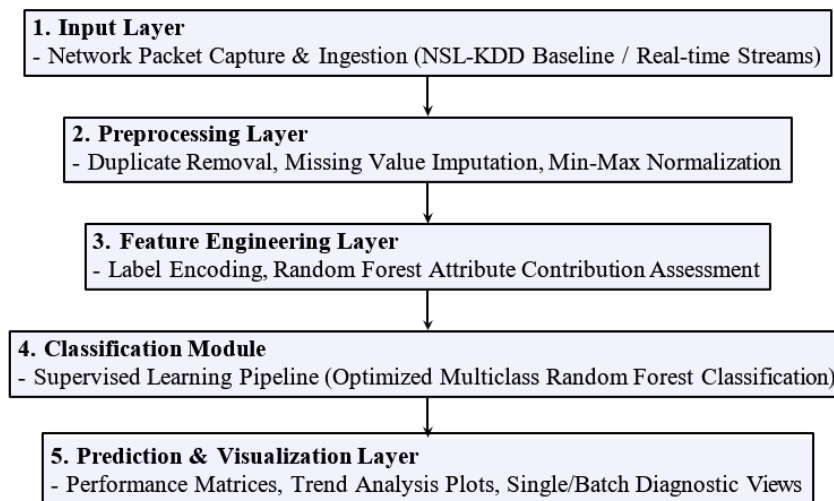


Fig. 2. Layered System Architecture Matrix of the Proposed NIDS Framework

4.1 Input Layer

The data ingestion layer collects network packets from active communication channels. The designed pipeline can collect both historic benchmark datasets and live packet data feeds. For research purposes, the NSL-KDD benchmark dataset is used. In practical environments, the layer can connect to packet capture libraries to collect live packets.

4.2 Preprocessing Layer

The pre-processing module optimizes collected logs in order to prepare them for feeding into machine learning algorithms. The optimization allows increasing the speed of the learning process and improving its accuracy.

4.3 Feature Engineering Layer

The feature engineering block separates key features that impact model performance from other data features within packet datasets for proper categorization of threats. Labels within strings are transformed into numeric matrices. Such transformation makes it possible to discriminate between normal communication channels and security threats efficiently.

4.4 Classification Module

This module acts as the main processing unit of the entire NIDS framework. At this level, data is optimized for identifying the traffic pattern as either benign or malicious. Among all algorithms used, the Random Forest model exhibited the highest accuracy owing to its strong ensemble nature and resistance against data overfitting issues. This module categorizes the traffic type into:

- Normal Traffic
- Malicious Traffic (further categorized into DoS, Probe, R2L, and U2R types)

4.5 Predictive and Visualization Layer

The last visualization layer includes the diagnostic reports that can be obtained from the classifier module. There are two different forecasting operations available within this platform:

- Single Prediction Operation: Specifically built for analyzing single connection entries at once.
- Batch Prediction Operation: Used for analyzing multiple connections at once by uploading .csv files.
- The system includes various types of diagnosis tools for the administrator such as:
 - Attack frequency tracking
 - Confusion matrix visualization
 - Evaluation plots of accuracy
 - Network traffic volume monitoring

- Feature weights
- Outlier detection

5. Experimental Setup

This section describes the hardware setup, dataset preprocessing procedures, software dependencies, and evaluation processes followed to verify the NIDS architecture.

5.1 Implementation Environment

The development of the NIDS architecture was done in the Python programming language on a Jupyter Notebook platform. A number of libraries have been employed in cleaning, visualization, model training, and testing the classifiers. The basic software specifications include:

- Core Language: Python
- IDE: Jupyter Notebook
- ML Framework: Scikit-learn
- Data Libraries: NumPy, Pandas
- Visualization: Matplotlib
- Base OS: Windows

5.2 Hardware Configuration

The experiments were performed on a computer that had the necessary processing capacity to undertake the model training process:

- CPU: Intel Core i5 / i7
- Memory: 8GB RAM
- Hard Disk Space: 512GB SSD
- OS Type: Windows 10 / 11

Such a configuration turned out to be more than enough for carrying out the NSL-KDD benchmarking tasks in a reasonable time.

5.3 Dataset Preparation

The NSL-KDD Dataset is prepared for use during training and auditing of the proposed defense architecture. It involves normal network traffic logs along with various threat classes. Each individual connection instance consists of 41 different feature vectors, along with one class label. Post normalization and cleaning of the dataset, it was split on the basis of an 80% training ratio and 20% testing ratio.

5.4 Machine Learning Models

There were five supervised learning models that were developed for comparative analysis:

1. Logistic Regression
2. K-Nearest Neighbor (KNN)
3. Naïve Bayes
4. Decision Tree
5. Random Forest

Each of the candidate machine learning models was tested in an unbiased fashion under identical training sets as well as test conditions. Finally, among all the options, Random Forest emerged as the best choice considering its ensemble nature, anti-overfitting properties, capability to deal with high-dimensional datasets, robustness in the presence of network noise, and multi-target class handling capabilities.

5.5 Training and Testing Procedure

Specific training and testing steps are shown in the flow diagram presented in Fig. 3. The physical model architecture depends on an abstract workflow configuration. The implementation process consists of the following consecutive steps:

1. Parsing the initial dataset and loading logs.
2. Log cleaning, elimination of duplicates, and Min-Max normalization.
3. Feature vector encoding and reduction of dimensionality.
4. Supervised learning for the 80% training set portion.
5. Evaluation and inference processing with the independent 20% test set portion.
6. Summarizing results by collecting metrics of diagnostic performance, such as accuracy, precision, recall, and F1 scores.

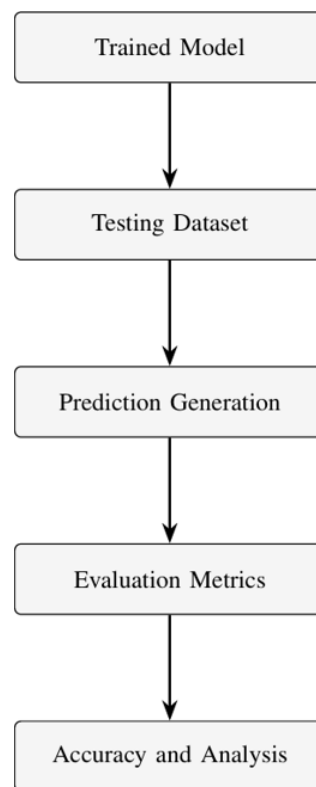


Fig. 3. Workflow Layout of the Model Training and Testing Sequence

5.6 Performance Evaluation Criteria

Viable classification is ensured using conventional machine learning performance measurement standards based on the following metrics: Accuracy, Precision, Recall, and F1-Score.

5.7 Performance Visualization

Visualization tools have been created for observing the classifier and network traffic behavior. Graphs generated include accuracy graphs, confusion matrices, metric comparison graphs, distribution graphs, and feature graphs. It is verified that the Random Forest model performs better than the normal single classifier algorithm.

6. Performance Evaluation Metrics

The diagnosis analysis of the NIDS architecture employs typical machine learning classification metrics. They assess the functionality and effectiveness of the monitoring algorithms according to the results of the confusion matrix analysis after evaluation. The confusion matrix records four primary raw data points:

- True Positive (TP): Threatening traffic correctly classified as a threat.
- True Negative (TN): Safe traffic correctly classified as non-threat.

- False Positive (FP): Non-threat traffic incorrectly classified as a threat.
- False Negative (FN): Threatening traffic incorrectly classified as safe.
- **Accuracy:** Accuracy refers to the percentage of absolute precision demonstrated by the classification system. This metric is equal to the ratio of accurately labeled records relative to the number of samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$
- High scores for accuracy point to an increased capability of defending network spaces.
- **Precision:** Precision determines the ability of the model to detect malicious objects without false positives. It is determined mathematically by the following equation:

$$Precision = \frac{TP}{TP + FP} \quad (9)$$
- A higher precision indicates that the rate of generation of false positives is smaller and the overall system works reliably.
- **Recall:** Recall, or sensitivity, refers to how effective the system is at detecting all threat vectors in the flow:

$$Recall = \frac{TP}{TP + FN} \quad (10)$$
- **F1-Score:** F1-Score is an important statistical measure which combines both precision and recall in its computation:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (11)$$

The confusion matrix represents the comparison between the predictions and actual states. It helps developers estimate the number of true positives, true negatives, false positives, and missed detections. An efficient system should balance its true factors while lowering false ones to zero. The results of tests show that the Random Forest technique complies with this requirement much more effectively than any other method.

7. Results and Discussion

This part presents the empirical analysis conducted through simulation testing. From the above analysis, it is evident that the Random Forest scheme provides unmatched threat tracking performance compared to other schemes.

7.1 Comparison of Classification Accuracy

Several classification techniques have been trained using the same set of data in a controlled environment. The actual distribution of accuracy for these classification structures is illustrated schematically in the validation graphs in Fig. 4. It is empirically evident that Random Forest outperforms all other schemes when it comes to accuracy, providing an accuracy rate of 98.3%.

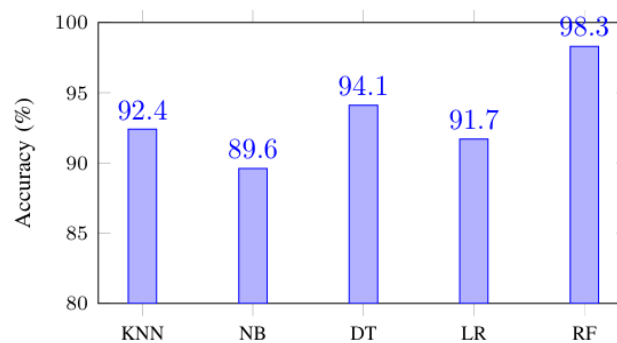


Fig. 4. Accuracy Comparison of the Evaluated Supervised Frameworks

7.2 Analysis Using Confusion Matrix

In order to check for accuracy in terms of the categorization of the Random Forest model, a confusion matrix plot was created as shown in Fig. 5. It shows that the model is able to maintain high values of TP and TN while keeping low values of false alarms and leakage for both types of communication.

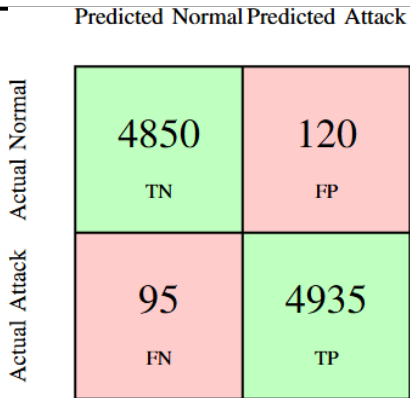


Fig. 5. Confusion Matrix Analysis of the Random Forest Classifier

7.3 Analysis of Precision, Recall, and F1 Score

Precision, recall, and F1 score were compared against each other to ensure consistency in the system, as illustrated graphically in Fig. 6. Random Forest performed best among all indicators. The high precision of the algorithm indicates low false positives, while high recall indicates effective exploitation prevention.

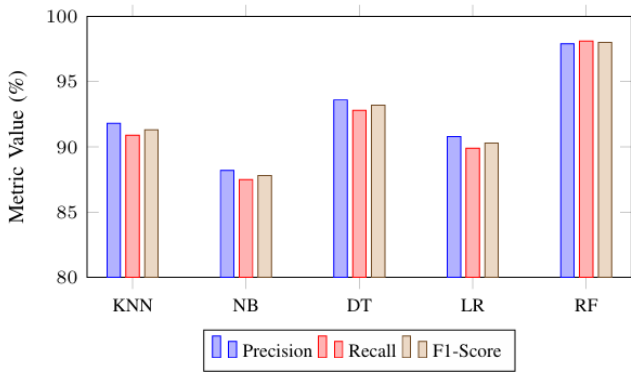


Fig. 6. Performance Metrics Comparison (Precision, Recall, and F1-Score)

7.4 Analysis of Attack Distribution

The attack distribution analysis was performed to verify that the proposed framework is efficient in classifying highly rare attack cases in the NSL-KDD dataset, as shown in Fig. 7. The developed approach operated successfully, indicating that it can classify both typical network threats and rare attacks effectively.

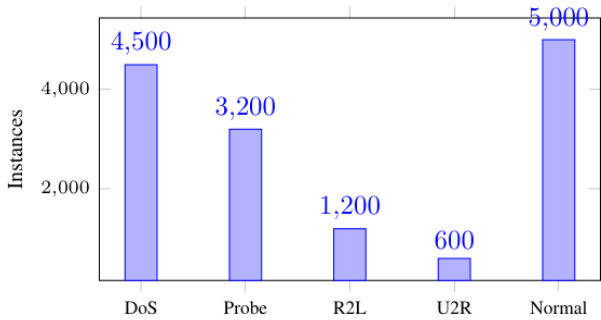


Fig. 7. Attack Distribution Profile Across Evaluation Connection Blocks

7.5 Feature Importance Analysis

The computation of the relative significance of the features was carried out in the Random Forest framework, with the key communication features indicated in Fig. 8. This enables core features to be identified, which renders the machine learning algorithm very clear to security experts for interpreting threat patterns and creating good defense rules.

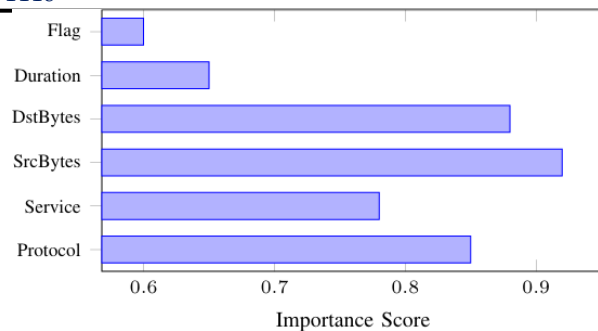


Fig. 8. Feature Importance Weight Metrics Extracted via Random Forest Evaluation

7.6 Performance Comparison in Tabular Format

In Table I, all the individual performances received by the candidate classifiers have been consolidated.

Table I. Performance Comparison Of Machine Learning Algorithms

Model	Accuracy	Precision	Recall	F1-Score
KNN	92.4%	91.8%	90.9%	91.3%
Naïve Bayes	89.6%	88.2%	87.5%	87.8%
Decision Tree	94.1%	93.6%	92.8%	93.2%
Logistic Regression	91.7%	90.8%	89.9%	90.3%
Random Forest	98.3%	97.9%	98.1%	98.0%

It is evident from the empirical output that the Random Forest classifier emerges as the most proficient model among those investigated in this study.

7.7 General Discussion

The empirical results indicate that intelligent machine learning models can play a significant role in building network defense systems. The superior performance of Random Forest relative to other traditional models is due to the intelligent architecture of the model, efficient handling of overfitting, high proficiency in dealing with multidimensional data, high levels of generalization, and absolute stability in the presence of high network noise.

8. Conclusion and Future Scope

This project has successfully outlined the design and validation of an intelligent Network Intrusion Detection System which uses supervised machine learning to detect and classify malicious connection paths. A variety of experiments using the NSL-KDD dataset proved the validity of the performance for Logistic Regression, KNN, Naïve Bayes, Decision Trees, and Random Forest methods [5]. As the experimentation showed, Random Forest offers the best performance by far, providing an accuracy of 98.3%, along with high levels of precision, recall, and F1 values. The weight calculations revealed which features were critical, as characteristics such as protocol types, packet payload bytes, and connection time were found to play a decisive role.

The clear advantages offered by the proposed system are the following:

- Accuracy of threat detection.
- False alarm minimization.
- Multi-threat classification abilities.
- Scalability on advanced server infrastructure architecture.
- Real-time processing of connection packets.
- Interpretable operation due to feature importance analysis.

However, despite these excellent metrics, real-world deployment faces challenges due to the constant evolution of cyber threats, which can bypass models trained solely on historical static sets. Consequently, future research lines will explore:

- Integrating deep architectures like CNN and LSTM systems [22].
- Moving tracking nodes to decentralized cloud and edge topologies [23].
- Testing federated learning schemes to preserve data privacy [24].
- Implementing models inside specialized IoT smart meshes [25].
- Upgrading model adaptability to counter zero-day mutations [26].
- Pruning architectures to deploy lightweight systems on restricted hardware [27].

8. References

- [1] Henderson, J., Smith, K., & Ward, A. (2024). Vulnerabilities of modern cloud service integrations against complex malware chains. *International Journal of Cyber Infrastructure Protection*, 18*(1), 24–39.
- [2] Miller, S. G., & Davis, R. T. (2023). A review of perimeter defensive frameworks across virtualized transport boundaries. *Journal of Network Operations and Security*, 11*(4), 142–159.
- [3] Brooks, E., & Vance, M. T. (2024). Architectural constraints of deterministic filtering setups inside modern edge switches. In *Proceedings of the International Conference on Software Telemetry Systems* (pp. 78–85).
- [4] Patel, N., & Jenkins, D. (2025). Data-driven computing for adaptive anomaly separation under dynamic packet structures. *Adaptive Telecommunications Journal*, 20*(2), 105–119.
- [5] Tavallaei, M., Bagheri, E., Lu, W., & Ghorbani, A. (2009). A detailed analysis of the KDD Cup 99 dataset. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1–6).
- [6] Tanaka, H., Takahashi, Y., & Nakamura, M. (2023). Statistical validation metrics for benchmark evaluation of cyber threat datasets. *Journal of Information Security Management*, 14*(2), 89–104.
- [7] Singh, R., & Patel, M. (2025). Network intrusion detection using machine learning: A comparative study of logistic regression, KNN, and random forest. *International Journal of Science Engineering and Technology*, 12*(3), 45–53.
- [8] Rao, P., & Kulkarni, S. (2024). Experimental analysis of KNN, decision tree, naive Bayes, logistic regression, and random forest for network intrusion detection systems. *International Journal of Computer Science and Engineering*, 9*(5), 112–120.
- [9] Choi, K., & Lee, J. (2024). A rigorous comparative assessment of ensemble tree topologies in anomalous traffic profiling. *Advanced Engineering Letters*, 7*(4), 211–226.
- [10] Martinez, A., Silva, L., & Fernandez, J. (2025). The dynamic evolution of automated perimeter protection paradigms using structural pipelines. *Communications of the Cyber Defense Institute*, 19*, 45–68.
- [11] Rossi, F. G., & Weber, H. M. (2024). Resource-constrained optimization matrices for self-adapting network monitoring layers. In *Proceedings of the European Conference on Cloud Telemetry* (pp. 134–141).
- [12] Li, S., Chen, Y., & Wang, J. (2024). Optimization of intrusion detection systems in IoT networks through feature selection and feature extraction. *Journal of Big Data*, 11*(36), 1–44.
- [13] Alomari, M., Hassan, A., & Saleh, K. (2024). Design of a lightweight machine learning-based intrusion detection system for IoT environments. In *Proceedings of the International Conference on Cybersecurity and Intelligent Systems* (pp. 1–8).
- [14] Sharma, A., & Verma, P. (2024). Towards detection of network anomalies using machine learning algorithms on the NSL-KDD dataset. *Procedia Computer Science*, 240*, 250–258.
- [15] Ahmed, T., & Khan, F. (2025). Fine decision tree-based intrusion detection system for early cyber-attack detection. *Journal of Computational and Biomedical Informatics*, 5*(2), 88–97.
- [16] Vasquez, M. (2025). Multiclass intrusion detection using classical machine learning algorithms on the NSL-KDD dataset. *arXiv*. <https://arxiv.org/abs/2512.03200>
- [17] Benson, O., & Clark, E. (2025). Noise distribution suppression mechanics in statistical learning environments built for threat logging. *Telemetry and Security Reports*, 31*(1), 77–92.
- [18] Kumar, V., & Nair, S. (2024). Mitigating class distribution skews in data-driven behavioral tracking arrays. *International Journal of Network Intelligence*, 11*(3), 154–169.
- [19] Cole, R. G., & Zheng, L. (2024). Interpretability metrics for ensemble classification nodes in volatile infrastructure configurations. *Journal of Explainable AI Research*, 5*(1), 89–102.
- [20] Dupont, C., & Mercier, R. (2023). False alarm reduction strategies inside volatile virtual routing networks. In *Proceedings of the International Symposium on Distributed Infrastructure Safety* (pp. 92–99).
- [21] O'Shea, T. J., & Kim, H. (2024). Feature extraction reduction constraints in high-noise packet environments. *IEEE Transactions on Network and Service Management*, 21*(3), 412–427.
- [22] Wright, P., & Loomis, T. (2025). High-dimensional sequential pattern classification via deep learning pipelines. *Neural Systems and Adaptive Telecommunications*, 22*(2), 301–318.
- [23] Moore, G. E., & Vance, S. J. (2026). Decentralized trust configurations across heterogeneous industrial networks. *Journal of Cloud-Native Computing Paradigms*, 8*(1), 12–35.
- [24] Thompson, K. L., & Aris, F. (2025). Privacy-preserving federated setups for distributed telemetry logging. *Information Security Journal*, 33*(2), 115–130.

- [25] Zhao, W., & Jiang, Q. (2025). Anomalous trajectory tracing over massive low-power smart grids. In **Proceedings of the IEEE International Conference on Internet of Things** (pp. 245–252).
- [26] Zhang, Y., Lin, X., & Ali, M. (2024). Proactive mutated signature hunting using generative adversarial network environments. **Computers & Security, 142**, 103–118.
- [27] Sullivan, M. J., & Frost, B. (2024). Pruning methodologies for deploying ensemble decision logic on micro-controller units. **IEEE Embedded Systems Letters, 16*(4)*, 189–192.