COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS ON NSL-KDD DATASET

Ms Linsha Judith Shaji [1]
[1] Student, Department of Computer Applications
St. Teresa's College(Autonomous)
Ernakulam, India
linshajudithshaji@gmail.com

Ms Janeena Shaju[2]
[2]Assistant Professor, Department of Computer Applications
St. Teresa's College(Autonomous)
Ernakulam,India
janeenashaju@gmail.com

***Abstract*— The security of modern communication networks cannot be sufficiently ensured without intrusion detection systems (IDS). The main objectives of these systems have been pattern recognition, signature analysis, and the detection of rule violations. Recent developments in machine learning (ML) and deep learning (DL) methodologies have shown potential as viable alternatives in the domain of network intrusion detection (NID). These methods can differentiate between typical and anomalous patterns. This study evaluates network intrusion detection systems (NIDS) using multiple ML algorithms, such as KNN, decision trees (DT), XGBOOST, and random forests (RF), with the NSL-KDD benchmark dataset. We analyze the precision, recall, accuracy, and F1 score of various ML techniques. The findings indicate that machine learning methods substantially enhance detection rates while minimizing false alarms compared to traditional approaches. This research demonstrates not only the feasibility of achieving a high detection rate of attacks but also the capability to make accurate predictions. These results clearly suggest that machine learning holds significant potential for the development of highly efficient NIDS systems.**

Keywords: NSL-KDD, Machine Learning, Random Forests(RF), Decision Trees(DT), Network Intrusion Detection (NID), K-Nearest Neighbour (KNN), and XGBoost(XGB).

I. INTRODUCTION

In today's digital landscape, networked systems play a vital role in everyday activities, including communication, information sharing, online transactions, and the operation of critical infrastructure. As the volume of network traffic continues to expand, network systems have become more vulnerable to a wide range of cyberattacks, with intrusion activities increasing over time. Intrusion detection plays a critical role in preventing unauthorised access and misuse of network data. Intrusion detection systems (IDS) detect intrusions by classifying network traffic and differentiating malicious behaviour from normal activity [1]. IDSs use two primary approaches for attack detection, which are either anomaly based or signature-based. An intrusion detection system operates by identifying either known attack signatures or deviations from established normal behaviour. Signature-based intrusion detection, also known as misuse or knowledge-based detection, relies on predefined attack patterns stored in a signature database and primarily reacts to known threats [2]. However, this approach is unable to detect novel or zero-day attacks, as it can only identify threats that already exist in its signature database. However, this approach is unable to detect novel or zero-day attacks, as it can only identify threats that already exist in its signature database. In addition, maintaining and updating a large signature repository requires significant computational resources, since incoming data packets must be continuously compared against stored attack patterns. In contrast, anomaly-based or behaviour-based intrusion detection systems construct a model representing normal system behaviour and detect intrusions by identifying activities that significantly deviate from this model [3]. The limitations of traditional intrusion detection techniques have motivated the adoption of machine learning approaches for network security. Due to the increasing volume and complexity of network traffic, rule-based and signature-dependent systems struggle to detect emerging and zero-day attacks. Machine learning-based intrusion detection systems address these challenges by automatically learning traffic patterns and distinguishing between normal and malicious behaviors. Machine learning techniques are widely used in intrusion detection systems due to their ability to process large volumes of network data, enabling more effective and adaptive mechanisms for protecting information systems [4]. The effectiveness of such systems heavily depends on the availability of reliable benchmark datasets for training and evaluation. The NSL-KDD dataset, an improved version of the KDD Cup 1999 dataset, is widely used in intrusion detection research as it reduces redundancy and provides a more balanced evaluation environment [5]. In this study, the NSL-KDD dataset is utilized to conduct a comparative analysis of Decision Tree, Random Forest, K-Nearest Neighbour, and XGBoost models, with performance assessed based on accuracy, computational efficiency, and handling of imbalanced data using SMOTE.

The main points of this study are as follows:

- A comparison of various machine learning algorithms for detecting network intrusions.
- Utilizing the NSL-KDD dataset to make benchmarking more reliable and less biased.
- Showing that Random Forest performs better than other tested methods in accuracy, precision, recall, and F1-score.
- Using SMOTE to improve the representation of minority attack types and boost classification accuracy.
- Providing insights that help in creating more scalable and intelligent intrusion detection systems.

II. LITERATURE REVIEW

Intrusion Detection Systems (IDS) play a crucial role in safeguarding modern network infrastructures against increasingly sophisticated cyber-attacks. Traditional security mechanisms such as firewalls and antivirus software are often insufficient in detecting novel and zero-day attacks, which has led researchers to explore machine learning and deep learning approaches for intrusion detection. The NSL-KDD dataset has emerged as one of the most widely used benchmark datasets for evaluating IDS performance due to its improved structure over the original KDD'99 dataset. Early studies primarily focused on analysing the characteristics of the NSL-KDD dataset and evaluating the effectiveness of classical classification algorithms. Dhanabal and Shantharajah conducted an extensive analysis of the NSL-KDD dataset using various classification techniques implemented in the WEKA tool. Their study highlighted the relationship between network protocols and attack types and demonstrated that machine learning classifiers can effectively distinguish between normal and anomalous traffic [6].

Subsequent research emphasised the comparative evaluation of supervised machine learning algorithms for intrusion detection. Belavagi and Muniyal evaluated Logistic Regression, Naïve Bayes, Support Vector Machine, and Random Forest classifiers on the NSL-KDD dataset. Their experimental results showed that Random Forest outperformed other classifiers in terms of detection accuracy, reinforcing the suitability of ensemble learning methods for IDS applications [7].

With the advancement of computational resources, researchers began exploring hybrid and ensemble approaches to improve detection performance. A comparative study of intrusion detection systems demonstrated that combining multiple machine learning models can enhance classification accuracy while reducing false alarm rates. The study emphasized that no single algorithm performs optimally across all attack categories, highlighting the importance of comparative analysis in IDS research [8].

Other studies focused on feature selection and performance optimization to reduce computational complexity. Research comparing different intrusion detection techniques demonstrated that selecting relevant features significantly improves detection accuracy and processing efficiency. These studies also highlighted the importance of preprocessing steps such as normalization and categorical feature encoding when working with the NSL-KDD dataset [9].

More recent work has emphasized the need for continuous evaluation and benchmarking of IDS models due to the dynamic nature of cyber threats. Studies published in engineering and technology journals demonstrated that while deep learning models achieve higher accuracy, they often require greater computational resources. This trade-off highlights the importance of selecting IDS models based on application requirements and resource constraints [10].

III. METHODOLOGY

A. *Dataset Description*

This study utilizes the NSL-KDD dataset, a widely recognized benchmark for intrusion detection research. NSL-KDD is an enhanced version of the original KDD'99 dataset, developed to address key limitations such as redundant records, biased class distributions, and extremely frequent instances. These improvements make NSL-KDD particularly suitable for evaluating machine learning based intrusion detection systems. The dataset consists of 125,972 instances, each with 41 input features and 1 target label. Input features include a mixture of continuous, discrete, and categorical attributes, describing various characteristics of network connections. Categorical features include protocol type Transmission Control Protocol(TCP), User Datagram Protocol- UDP, Internet Control Message Protocol-ICMP), service (HTTP, SMTP), and flag (SF, S0), while numerical features capture packet statistics, connection duration, and other traffic characteristics. The target label identifies the type of network activity and includes 22 classes, representing normal connections and multiple attack types such as Denial-of-Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) attacks. The dataset is divided into training and testing subsets, with KDDTrain+ used for training and KDDTest+ used for validation. Unlike KDD'99, NSL-KDD avoids duplicate entries and provides a more balanced class distribution, enabling machine learning models to generalize better. The comprehensive structure combined with its reduced redundancy and controlled class balance, makes NSL-KDD a standard benchmark for evaluating IDS models under realistic conditions.

B. *Dataset Loading*

The NSL-KDD dataset used in this study is stored in .txt format and was imported into the Python environment using the Pandas library, which enables efficient data manipulation and preprocessing. The raw text file was converted into a structured DataFrame using the read_csv() function. This ensured that each instance and its corresponding features were organized in a tabular format suitable for subsequent analysis and model training. After loading, the dataset dimensions were verified to confirm the total number of rows and columns. This step is crucial to ensure that the complete dataset was successfully imported and that all instances were available for preprocessing operations.

C. *Data Preprocessing*

Effective data preprocessing is a critical step in preparing the NSL-KDD dataset for machine learning.

a) *Encoding Categorical Features*

Several features in NSL-KDD, such as protocol type, service, and connection flags, are categorical in nature. Since most machine learning algorithms require numerical input, these attributes were converted into numerical codes. Each unique category was assigned a distinct integer, preserving the distinctions between categories while making the data interpretable by the models. This step is essential for ensuring that all features contribute effectively to the learning process and prevent errors during model training.

b) *Partitioning the Dataset*

To accurately assess model performance on unseen data, the dataset was divided into training and testing subsets using an 80/20 split. Stratified sampling was applied to maintain the original distribution of classes in both subsets. This approach ensures that each subset represents the variety of attack types and normal traffic, allowing the models to learn from a

representative sample and providing a reliable estimate of their generalization ability on new network traffic.

c) Filtering Rare Classes

Very few instances represent certain attack types in NSL-KDD. These extremely rare classes can lead to instability during training and negatively affect evaluation metrics. Classes with fewer than ten samples were removed from the training data, ensuring that the models focus on classes with sufficient representation. This filtering improves the stability of learning and reduces the risk of overfitting to underrepresented attack types, resulting in more reliable detection performance.

d) Data Intergity and Preparation Workflow

The preprocessing workflow also emphasizes overall data integrity. Before training, the dataset was inspected for missing or inconsistent values, ensuring that all instances are complete and correctly formatted. The combination of encoding, stratified splitting, and rare class removal provides a clean, structured, and balanced dataset, forming a strong foundation for machine learning experiments. These preprocessing steps are designed to enhance learning efficiency, improve model accuracy, and support fair evaluation across all attack categories.

D. Handling Class Imbalance Using SMOTE

After the removal of rare attack classes, the training dataset remained imbalanced, with majority classes dominating the class distribution. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to the training data. SMOTE generates synthetic samples for minority classes by interpolating between existing data points in the feature space, thereby improving class balance without simply duplicating instances. The oversampling process was performed exclusively on the training set to avoid data leakage and to preserve the validity of the evaluation process. By balancing the class distribution, this step reduced the bias of machine learning models toward majority classes and improved their ability to learn decision boundaries for underrepresented attack categories.

E. Machine Learning Models

To evaluate the effectiveness of different classification techniques for intrusion detection, four supervised machine learning models were trained and compared. These models were selected to represent a mix of simple, ensemble-based, distance-based, and advanced boosting algorithms. All models were trained using the SMOTE-balanced training dataset to ensure fair comparison and to reduce bias toward majority classes.

a) Decision Tree Classifier

The Decision Tree classifier is a widely used supervised learning algorithm known for its simplicity and interpretability. A Decision Tree is a non-parametric supervised learning algorithm used for both classification and regression problems. It represents the decision-making process in a hierarchical tree structure composed of a root node, internal decision nodes, branches, and leaf nodes. Each internal node corresponds to a decision rule based on a feature, while the leaf nodes represent the final predicted outcomes [11]. This structure makes Decision Trees easy to interpret and understand, particularly for classification tasks It operates by recursively splitting the dataset based on feature values, forming a tree-like structure where internal nodes represent decision rules and leaf nodes represent class labels. The splitting process aims to maximise class separation at each step. Decision Trees can handle both numerical and categorical features and do not require feature normalisation or scaling. These properties make them well-suited for network intrusion datasets such as NSL-KDD. In this study, the Decision Tree classifier was trained on the SMOTE-balanced dataset and used as a baseline model for comparison with more complex algorithms.

b) Random Forest Classifier

Random Forest is an ensemble learning method that constructs multiple decision trees, where each tree is generated using a randomly sampled vector drawn independently from the same distribution. The final prediction is obtained by aggregating the outputs of all trees in the forest. As the number of trees increases, the generalization error of the model converges to a stable limit, improving predictive performance and robustness [12]. Each tree is built using a random subset of the training data and features, and the final prediction is obtained through majority voting among all trees. This ensemble strategy reduces overfitting and improves generalization compared to a single Decision Tree. Random Forest performs well on large and high-dimensional datasets and is robust to noise and outliers. Similar to Decision Trees, it does not require feature scaling. In this work, a Random Forest model consisting of 100 trees was trained on the SMOTE-balanced dataset to assess the performance of an ensemble-based classifier for intrusion detection.

c) K-Nearest Neighbours (KNN)

K-Nearest Neighbour is a distance-based classification algorithm that assigns a class label to a data instance based on the majority class of its nearest neighbours in the feature space. K-Nearest Neighbour (KNN) is a supervised machine learning algorithm commonly used for classification tasks. It classifies a data instance based on the majority class among its K closest neighbours in the feature space [13]. Unlike other classifiers, KNN does not build an explicit model during training. Instead, all computations occur during prediction, which can make it computationally expensive for large datasets. KNN is also sensitive to class imbalance, making the application of SMOTE essential before training. This model was included to analyze how a distance-based approach performs in comparison with tree-based classifiers on the NSL-KDD dataset.

d) XGBoost Classifer

XGBoost (Extreme Gradient Boosting) is a supervised machine learning algorithm based on gradient-boosted decision trees. It builds an ensemble of trees sequentially, where each new tree is trained to correct the errors made by previous trees by optimising a differentiable loss function. XGBoost incorporates regularisation, efficient handling of sparse data, and optimised system design to achieve high accuracy and scalability, making it suitable for large-scale classification problems [14]. It was included to compare the performance of a powerful boosting-based model with simpler classifiers such as Decision Trees, Random Forest, and KNN.

IV. IMPLEMENTATION

A. *EXPERIMENTAL SETUP*

All experiments in this study were conducted using Google Colab, which provides a cloud-based computational environment suitable for training and evaluating machine learning models. The NSL-KDD dataset was obtained from Kaggle and loaded into the Colab environment for processing and experimentation. The dataset was accessed directly within the working directory, ensuring seamless data handling without manual file transfers.

The experimental pipeline was implemented using the Python programming language. Standard machine learning libraries were employed, including Pandas and NumPy for data manipulation, Scikit-learn for model training and evaluation, Imbalanced-learn for handling class imbalance, and XGBoost for implementing the gradient boosting classifier. These libraries collectively enabled efficient preprocessing, model development, and performance analysis. Prior to model training, categorical features in the dataset were encoded into numerical representations to ensure compatibility with machine learning algorithms. The dataset was then divided into training and testing subsets using an 80:20 split with stratified sampling to preserve the original class distribution. To improve training stability, rare classes with very low sample counts were removed from the training set. Despite this filtering, the dataset remained imbalanced; therefore, the Synthetic Minority Over-sampling Technique (SMOTE) was applied exclusively to the training data to generate synthetic samples for minority classes and achieve a balanced class distribution. Four supervised machine learning classifiers were trained and evaluated: Decision Tree, Random Forest, K-Nearest Neighbours (KNN), and XGBoost. All models were trained on the SMOTE-balanced training dataset using fixed random seeds to ensure reproducibility. The classifiers were selected to represent diverse learning strategies, including tree-based, ensemble-based, distance-based, and boosting-based approaches. Model performance was evaluated using the unseen test dataset. Accuracy was used as the primary evaluation metric, while precision, recall, and F1-score were also analyzed to provide a detailed assessment of classification performance. The entire experimental setup was designed to ensure reproducibility, fair comparison among models, and reliable evaluation of intrusion detection performance.

B. *PERFORMANCE METRICS*

Performance metrics assess the ability of the models to correctly classify different types of network traffic and attacks present in the NSL-KDD dataset. Since intrusion detection datasets are often imbalanced, relying on a single metric may be misleading; therefore, multiple evaluation measures were considered.

a) Accuracy

Accuracy represents the proportion of correctly classified samples among the total number of samples. It provides a general indication of model performance; however, it may not fully reflect effectiveness when class distributions are imbalanced.

$$= \frac{+}{+ \quad + \quad +}$$

b) Precision

Precision measures the proportion of correctly predicted positive samples among all samples predicted as positive. High precision indicates a low false positive rate, which is important in intrusion detection systems to avoid misclassifying normal network traffic as malicious activity.

$$= \frac{}{+}$$

c) Recall

Recall, also known as sensitivity, measures the proportion of actual positive samples that are correctly identified by the model. A high recall value indicates a low false negative rate. In intrusion detection, recall is particularly important, as failing to detect an attack can lead to serious security consequences.

$$= \frac{}{+}$$

d) F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a balanced evaluation of a model's performance, especially when both false positives and false negatives must be considered. This metric is particularly suitable for imbalanced datasets such as NSL-KDD.

$$F1 - Score = 2 * \frac{(Precision\ .\ Recall)}{(\quad + \quad)}$$

e) Support

Support refers to the number of actual instances of each class present in the test dataset. It helps in understanding the class distribution and provides context for interpreting the reliability of the reported precision, recall, and F1-score values for each category.

V. RESULTS AND DISCUSSION

This section presents the experimental results obtained by evaluating four machine learning models—Decision Tree, Random Forest, K-Nearest Neighbour (KNN), and XGBoost—on the NSL-KDD dataset for network intrusion detection. The models were assessed using accuracy, precision, recall, and F1-score to provide a comprehensive comparison of their classification performance.

Among the models tested, the Random Forest classifier showed the best overall performance. It achieved an accuracy of 0.8987 and exhibited strong precision, recall, and F1-score values for the main types of attacks found in the NSL-KDD Dataset. Although a slight reduction in performance was observed for minority attack classes, Random Forest managed class imbalance more effectively than the other models. This is due to its ensemble nature, which combines

predictions from multiple decision trees, leading to better robustness and generalization. These characteristics make Random Forest the most effective and dependable model in this study for intrusion detection.

Both Random Forest and Decision Tree models achieved high accuracy values and showed strong performance for major attack categories. Their higher accuracy and weighted F1-scores indicate a better balance between correctly identifying attacks and reducing false alarms. Random Forest slightly outperformed the Decision Tree model, indicating the benefit of ensemble learning rather than a single-tree approach.

The XGBoost classifier showed moderate performance, achieving better results than KNN but falling short of the tree-based models. While XGBoost is known for its boosting techniques, its performance in this study may have been affected by dataset characteristics and parameter sensitivity.

KNN had the lowest accuracy among the models, mainly due to its sensitivity to high-dimensional data and class imbalance. Even with the use of SMOTE, KNN had difficulty accurately classifying less common attack types.

In conclusion, Random Forest is recognized as the most effective and reliable model for intrusion detection on the NSL-KDD dataset, as supported by the experimental results and comparative analysis. The use of weighted precision, recall, and F1-score ensures that the evaluation reflects real-world intrusion detection scenarios where attack and normal classes are unevenly distributed.

TABLE I: Performance Comparison of Machine Learning Models on the NSL-KDD Dataset

Model	Accuracy	Precision (weighted)	Recall (weighted)	F1-Score (weighted)
RF	0.8987	0.90	0.90	0.90
DT	0.8973	0.90	0.90	0.90
XGB	0.8592	0.87	0.86	0.86
KNN	0.8366	0.86	0.84	0.85

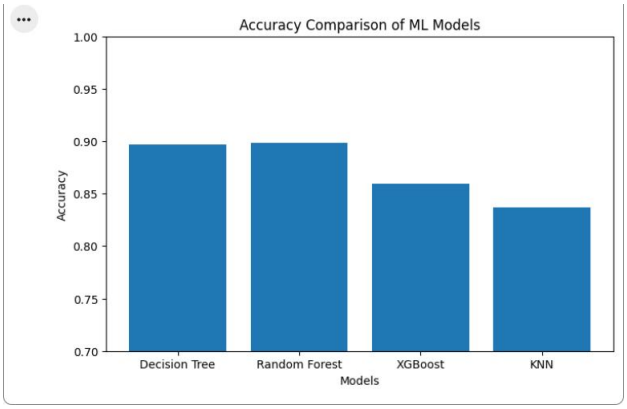

Fig.1: Accuracy Comparison of ML Models.

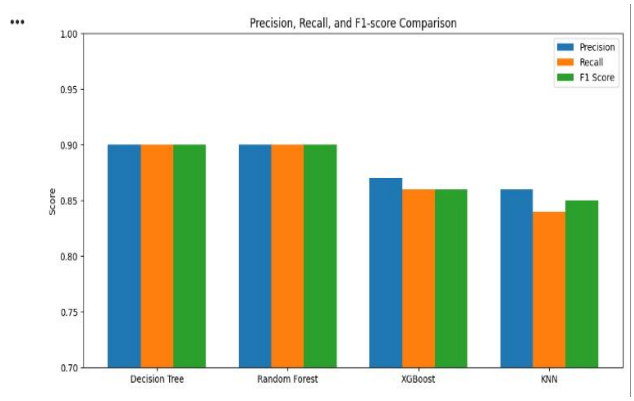

Fig.2: Metric Comparison.

VI. CONCLUSION

This study can be extended in several meaningful ways to improve the effectiveness of the intrusion detection system. Future work can explore deep learning models such as CNNs, RNNs, or LSTMs, which may capture more complex attack patterns than traditional machine learning methods. Feature selection and dimensionality reduction techniques like PCA or mutual information can also be applied to optimize performance and reduce training time. Additionally, the current work is based on offline data, so developing a real-time IDS capable of analyzing live network traffic would make the system more practical for real-world deployment. As new cyberattacks continue to emerge, implementing adaptive or online learning models can help the IDS update itself automatically. The system can also be enhanced by deploying it on cloud or edge platforms for better scalability, and by integrating it with existing security tools such as firewalls. Finally, although SMOTE helped address class imbalance, more advanced resampling approaches could be explored to further improve the detection of rare attack types.

REFERENCES

[1] Roy, D.B., Chaki, R.: State of the art analysis of network traffic anomaly detection. In: Applications and Innovations in Mobile Computing (AIMoC), IEEE, pp. 186–192 (2014)

[2] Masdari, M., Khezri, H.: A survey and taxonomy of the fuzzy signature-based intrusion detection systems. Appl. Soft Comput. 92, 106301 (2020).

[3] Masdari, M., Khezri, H.: Towards fuzzy anomaly detection-based security: a comprehensive review. Fuzzy Optim. Decis. Mak. 20(1), 1–49 (2021).

[4] Alam, S., Shuaib, M., & Samad, A. (2019). A Collaborative Study of Intrusion Detection and Prevention Techniques in Cloud Computing. In Lecture Notes in Networks and Systems (Vol. 55, pp. 231–240).

[5] Ravipati, R. D., & Abualkibash, M. (2019). Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets-a review paper. International Journal of Computer Science & Information Technology (IJCSIT) Vol, 11.

[6] Dhanabal, L., and S. P. Shantharajah. "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms." International journal of advanced research in computer and communication engineering 4.6 (2015): 446-452.

[7] Belavagi, Manjula C., and Balachandra Muniyal. "Performance evaluation of supervised machine learning algorithms for intrusion detection." *Procedia Computer Science* 89 (2016): 117-123.

[8] Eshak Magdy, Mina, et al. "A Comparative study of intrusion detection systems applied to NSL-KDD Dataset." The Egyptian International Journal of Engineering Sciences and Technology 43.2 (2023): 88-98.

[9] Revathi, Sathyanarayanan, and A. Malathi. "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection." International Journal of Engineering Research & Technology (IJERT) 2.12 (2013): 1848-1853.

[10] Kasongo, Sydney Mambwe. "A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework." *Computer Communications* 199 (2023): 113-125.

[11] IBM, "What is a decision tree?" IBM Think, https://www.ibm.com/think/topics/decision-trees accessed Jan 12 2026.

[12] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

[13] Suyal, Manish, and Parul Goyal. "A review on analysis of k-nearest neighbor classification machine learning algorithms based on supervised learning." International Journal of Engineering Trends and Technology 70.7 (2022): 43-48.

[14] Chen, Tianqi. "XGBoost: A Scalable Tree Boosting System." *Cornell University* (2016).