Machine Learning with Applications 22 (2025) 100738

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Enhancing IDS performance through a comparative analysis of Random Forest, XGBoost, and Deep Neural Networks

Sow Thierno Hamidou *, Adda Mehdi

Department of Mathematics, Computer Science, and Engineering, University of Quebec at Rimouski, 300, allée des Ursulines, Rimouski, QC G5L 3A1, Québec, Canada

ARTICLE INFO

Keywords:
Cybersecurity
Intrusion Detection System (IDS)
Machine learning
Deep learning
NSL-KDD
Smote

ABSTRACT

Intrusion Detection Systems (IDS) face major challenges in network security, notably the need to combine a high detection rate with reliable performance. This reliability is often affected by class imbalances and inadequate hyperparameter optimization. This article addresses the issue of improving the detection rate of IDS by evaluating and comparing three machine learning algorithms: Random Forest (RF), XGBoost, and Deep Neural Networks (DNN), using the NSL-KDD dataset. In our methodology, we integrate SMOTE (Synthetic Minority Oversampling Technique) to tackle the unbalanced nature of the data, ensuring a more balanced representation of the different classes. This approach helps optimize model performance, reduce bias, and enhance robustness. Additionally, hyperparameter optimization is performed using Optuna, ensuring that each algorithm operates at its optimal level. The results show that our model, using the Random Forest algorithm, achieves an accuracy of 99.80%, surpassing the performance of XGBoost and Deep Neural Networks (DNN). This makes our approach a true asset for intrusion detection methods in computer networks.

1. Introduction

Intrusion represents a major threat in the field of cybersecurity, constituting a crucial challenge for the protection of computer systems. Indeed, a single malicious breach can compromise sensitive data, leading to its theft or destruction, and threatening critical infrastructures in a few seconds (Ahmad et al., 2018). In this context, the effectiveness of Intrusion Detection Systems (IDS) is essential to ensure the security of computer networks, as they play a crucial role in identifying cyber-attacks and protecting sensitive data. According to Catania and Garino (2012), IDS must be able to detect intrusions effectively and quickly to mitigate the risks associated with cyber threats.

However, these systems face many challenges, including the need to maintain a high detection rate while ensuring reliable performance. These aspects, particularly detection accuracy and performance reliability, represent fundamental issues. Indeed, a problem related to effectiveness in data classification lies in class imbalance, especially when one class is markedly less represented than the others, such as in the case of fraud detection (Chan et al., 1999) or medical diagnosis (Srinivas & Katarya, 2022). Attacks often represent a small portion of the recorded events in security applications, which explains the imbalance in the datasets. Thus, Raju et al. (2024), in their study, highlighted the issue of class imbalance in datasets, in the context of facial and vocal emotion recognition. This imbalance can make it difficult to learn the minority classes, leading to poor model performance that favors the majority classes. In a similar intrusion detection context, this imbalance complicates the learning of models, which tend to be biased toward the majority classes, potentially reducing their effectiveness in detecting anomalies. This limitation becomes problematic in security contexts where it is essential to detect the rare intrusion incidents.

Moreover, hyperparameter optimization is a key element in improving the performance of classification models, as it helps balance the complexity of the model and its ability to generalize. This process reduces overfitting and ensures more reliable predictions on new data. As Tran et al. (2020) highlight, the importance of an adequate optimization approach to maximize IDS performance while minimizing costs and the risks of overfitting, emphasizing the strategic impact that precise and appropriate hyperparameter tuning can have on the effectiveness of intrusion detection models. Srinivas and Katarya (2022) also demonstrate the importance of hyperparameter optimization to improve the effectiveness of the detection model, here applied to predicting heart disease, but with similar implications for IDS.

To address these challenges, this study evaluates three machine learning algorithms: Random Forest (RF), XGBoost, and Deep Neural Networks (DNN) on the NSL-KDD dataset. Indeed, these algorithms have achieved good results in previous studies, such as the one by Azam et al. (2023), which explored various algorithms and the challenges

* Corresponding author.
E-mail addresses: thiernohamidou.sow@uqar.ca (S.T. Hamidou), mehdi_adda@uqar.ca (A. Mehdi).

https://doi.org/10.1016/j.mlwa.2025.100738
Received 20 May 2025; Received in revised form 6 September 2025; Accepted 16 September 2025
Available online 27 September 2025

related to big data in order to overcome the limitations of traditional IDS in the face of current cyber threats. The authors reported a detection accuracy of 99.60% achieved by XGBoost, using a feature selection strategy based on a discernibility function. Similarly, the study by Pranto et al. (2022) achieved an accuracy of 99.50% with Random Forest on the KDD'99 dataset by applying a dimensionality reduction and hyperparameter optimization approach. Their approach included Min-Max data normalization as well as feature selection based on variance and correlation. The hyperparameters were optimized using cross-validation and grid search. Moreover, the DNN method was used in the study by Mohammed and Gbashi (2021) on the NSL-KDD dataset, achieving an accuracy of 94%. Their approach included Min-Max normalization, categorical attribute encoding, and feature selection through Recursive Feature Elimination (RFE) with Random Forest, along with dropout regularization applied to reduce overfitting. Another study by Kikissagbe et al. (2024) achieved an accuracy of 99.62% in detecting Denial of Service (DoS) attacks on the Edge IIoT dataset. This performance was achieved through six combinations of techniques, including data balancing with Synthetic Minority Oversampling Technique (SMOTE) and Random Under-Sampling (RUS), as well as feature selection using RF, DNN, PCA (Principal Component Analysis), and classifiers such as SVM (Support Vector Machine), RF, XGBoost, and DNN.

Therefore, we hypothesize that by using these algorithms, we will be able to improve the performance of IDS, making them more precise and robust. We have integrated the SMOTE technique, which generates synthetic samples for the minority class, thereby balancing the data and improving the model's ability to detect rare attacks. This also reduces training time and enhances the overall model performance by avoiding biases related to class imbalance (Jiang et al., 2020).

Indeed, for a reliable performance evaluation, we used stratified K-Fold cross-validation, which divides the data into several folds while preserving the class distribution. This approach allows the model to be trained and tested on different subsets, ensuring accurate evaluations by minimizing biases caused by unbalanced data splits. Mahesh et al. (2023). We also used Optuna for hyperparameter optimization, a method that dynamically defines the search space, allowing flexibility in adjusting parameters during trials. Unlike grid or random search, Optuna is faster, less computationally expensive, and suitable for complex models such as neural networks, thus improving the accuracy and efficiency of prediction models (Hanifi et al., 2024).

Our results demonstrate that the Random Forest algorithm achieves an accuracy of 99.80%, which indicates that our approach provides an effective alternative to the different approaches we have explored. This improves the robustness of IDS to address the growing challenges of cybersecurity.

This work makes the following contributions:

- We demonstrate that optimal hyperparameter tuning via Optuna, applied to IDS models on the NSL-KDD dataset, achieves detection accuracy surpassing the best reported values in the literature, positioning our approach as a reference for high precision in the IDS context.
- We propose a reproducible experimental protocol that uses identical hyperparameters to fairly compare configurations with and without SMOTE, for both grid search and Optuna optimization. This ensures that observed benefits stem directly from the applied techniques, providing a reliable evaluation rarely addressed with such precision in existing works.
- Our analysis shows that even on a relatively balanced dataset like NSL-KDD, SMOTE provides a modest but essential improvement for detecting minority classes. This improvement is crucial because training a model without addressing class imbalance can bias its performance, especially in the IDS context. As noted by Chawla et al. (2002), imbalanced datasets pose a significant challenge: most examples belong to the majority class, while minority class examples are rare, and misclassifying minority examples often incurs a higher cost. SMOTE mitigates this issue by generating synthetic minority samples, thereby improving model sensitivity to rare cases and enhancing overall reliability.
- Our detailed comparison of three models, considering both classification performance and computational costs, offers a clear and practical framework to select IDS models adapted to operational constraints. This guide aids practitioners in making informed decisions balancing efficiency and performance.

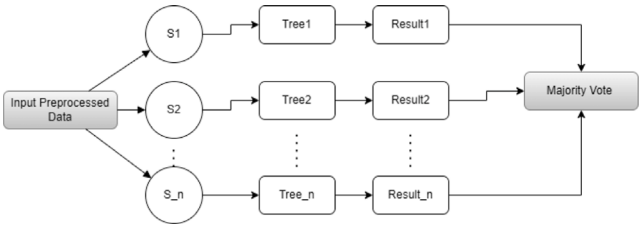

Fig. 1. Architecture of the RF.

Nevertheless, this study has limitations. The NSL-KDD dataset is widely used as a benchmark in IDS research, it presents certain limitations. Its relatively old design does not reflect the evolution of current cyber threats or the increasing complexity of real-world network traffic. This obsolescence may limit the generalizability of the results obtained to more recent production environments.

The structure of the paper is as follows: Section 2 presents the context of the topic, Section 3 addresses the review of the literature, Section 4 describes the proposed approach, and Section 5 details the experiments conducted. The results are analyzed in Section 6, ollowed by a discussion and future work in 7, while the conclusion is addressed in Section 8.

2. Background

The importance of IDS in the field of cybersecurity is paramount, especially as cyberattacks grow increasingly sophisticated and targeted. IDS play a crucial role in detecting various types of network threats, such as DoS attacks, port scans, malware, and advanced persistent threats (APT).

These systems are designed to continuously monitor data flows and identify malicious behaviors or anomalies that may indicate potential intrusions. Traditionally, IDS relied on predefined rules and signatures to detect known attacks (Khraisat et al., 2019). However, these approaches show their limitations in the face of emerging threats. This has led to the integration of machine learning techniques, which allow IDS to learn and adapt to new forms of attacks.

In this study, the NSL-KDD dataset used simulates real attacks, such as DoS, Remote to Local (R2L), Probe, and User to Root (U2R). It provides a comprehensive framework to evaluate the models' ability to accurately distinguish between normal and malicious behaviors. Each of the algorithms we applied has distinct characteristics that make them particularly effective for classification.

2.1. Random Forest

Random Forest is an ensemble machine learning algorithm used for classification and regression tasks. Its operation is based on the generation of multiple decision trees during the learning phase, with each tree contributing to the final prediction (Ahmad et al., 2018). In the context of classification, the final prediction is obtained through a majority vote of the predictions made by each decision tree in the forest.

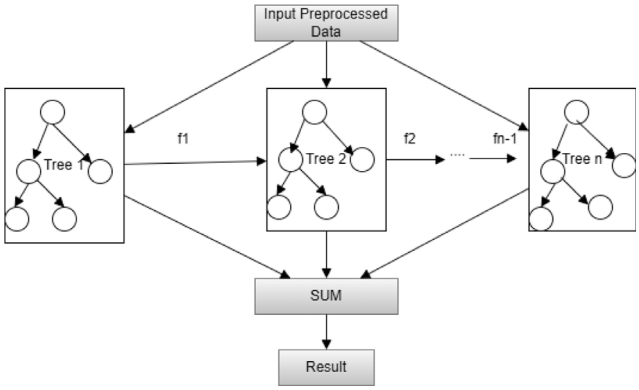

Fig. 2. Architecture of the XGBoost.

Fig. 1 presents the architecture of the Random Forest model used in an IDS. It describes the process in several steps:

The input data (samples) are preprocessed before being used in the model. Next, this data is divided into several subsets $S_1, S_2, \ldots, S_n$, each intended for a distinct decision tree. From these subsets, unique decision trees $Tree_1, Tree_2, \ldots, Tree_n$ are created, with each performing classification independently. The classification results obtained by each tree $Result_1, Result_2, \ldots, Result_n$ are then generated. Finally, to obtain a final decision, the results from the various trees are combined through a majority vote, allowing the dominant class to be selected.

2.2. XGBoost

XGBoost is a learning algorithm used for classification and regression tasks. It relies on the sequential construction of decision trees, where each tree improves the predictions of the previous one using gradient boosting. With XGBoost, the final prediction is obtained by summing the predictions of each individual tree in the model. Mathematically, this can be expressed as follows:

$$\hat{y} = \sum_{k=1}^{K} f_k(x)$$

where:

- $\hat{y}$ is the final prediction,
- K represents the number of trees,
- f_k is the prediction of tree k for the input x,
- x denotes the input features of the instance.

Each tree f_k contributes an additional prediction, and the overall result is the sum of all the predictions from the trees. This model was chosen for this study due to its ability to achieve high accuracy, which has already been demonstrated in previous works on intrusion detection (Dhaliwal et al., 2018).

Fig. 2 illustrates the architecture of the XGBoost model used in an IDS. It details the process in several steps:

The input data is first preprocessed before being used in the model. Unlike Random Forest, the decision trees in XGBoost are created sequentially. Each tree $Tree_1, Tree_2, \ldots, Tree_n$ is constructed by taking into account the errors of the previous trees. Each new tree adjusts its predictions to correct the errors of the earlier trees. At each step, a function $f_1, f_2, \ldots, f_{n-1}$ is applied to improve the accuracy of the model. The intermediate results provided by each tree are then summed to obtain a final score.

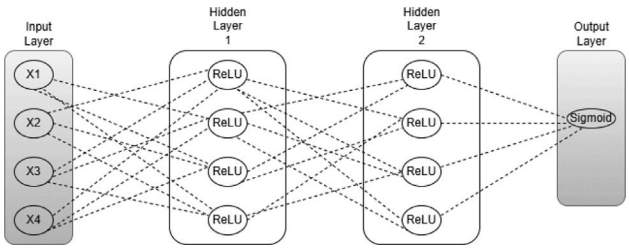

Fig. 3. Architecture of a DNN.

2.3. Deep Neural Network (DNN)

A Deep Neural Network (DNN) is a type of artificial neural network, structured in a sequence of layers called Multilayer Perceptrons (MLP). Each layer refines the representations acquired from the previous one, allowing the model to capture complex relationships in the data (Al-Maksousy et al., 2018). The output of a deep neural network (DNN) is evaluated based on the weights and inputs of a neuron using the following function:

$$y = \delta\left(\sum_{n=1}^{N} W_n x_n + b\right) = \delta(W^T X + b)$$

where:

- W is the weight vector.
- X is the input vector.
- b denotes the bias.
- δ represents the activation function (often a sigmoid or ReLU function).
- $W^T X$ is the dot product between the weight vector W and the input vector X. This product yields a value representing the weighted sum of the inputs.

Thus, the output of a DNN at each neuron is a weighted linear combination of the inputs, modified by a non-linear activation function to enhance the network's ability to capture relationships. This model was chosen for its ability to achieve high accuracy, as demonstrated by numerous studies.

Fig. 3 illustrates the architecture of a deep neural network (DNN) designed for an IDS. The process unfolds in several steps:

The input features $X_1, X_2, \ldots, X_n$ are fed into the first hidden layer. In this layer, each neuron, which is connected to all the input neurons, applies the ReLU activation function to learn complex relationships. The outputs of this first layer are then passed to a second hidden layer, also composed of neurons using the ReLU function, which refines the acquired representations.

Finally, the data flows through an output layer, consisting of a single neuron utilizing a sigmoid activation function, intended for binary classification.

The performance of these models has been evaluated using several key indicators: accuracy, precision, recall, F1 score, and the ROC curve.

Accuracy represents the total percentage of correctly classified instances and is calculated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision, on the other hand, focuses on positive results and is calculated by the following ratio:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall is determined by dividing the number of true positives by the sum of true positives and false negatives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

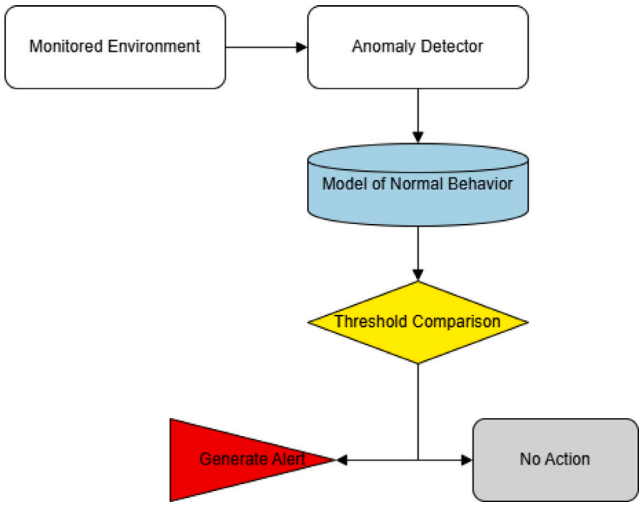

Fig. 4. IDS anomaly detection process.

The F1 score is defined by the formula:

$$\text{F1 Score} = \frac{2 \times TP}{2 \times TP + FN + FP}$$

A high F1 score indicates a good balance between recall and precision. Finally, the ROC curve represents the true positive rate against the false positive rate (FPR).

The results of this research aim to improve IDS by ensuring continuous monitoring and rapid response to threats. Fig. 4 illustrates the intrusion detection process through behavioral analysis, where suspicious actions are compared to a model of normal behavior to detect anomalies (Aldallal & Alisa, 2021).

3. Literature review

The necessity for efficient IDS has led to the integration of machine learning techniques to optimize detection accuracy in complex network environments. Numerous studies have proposed frameworks using various algorithms to enhance IDS performance. In this section, we will examine the different approaches developed for intrusion detection.

Ahmad et al. (2018) compared SVM, Random Forest, and Extreme Learning Machine (ELM) on the NSL-KDD dataset, revealing that ELM offered the best accuracy 99.67%, particularly for large datasets. SVM with an Radial Basis Function (RBF) kernel showed better performance on reduced subsets. However, the study is limited to metrics such as accuracy and F1-score without including the AUC-ROC, which is essential for evaluating the ability to differentiate classes in imbalanced data contexts, which would have allowed a more comprehensive assessment of the models for intrusion detection.

Similarly, Azam et al. (2023) evaluated the effectiveness of various machine learning algorithms applied to the KDD Cup 99 and NSL-KDD datasets. Their study focuses on methods such as C4.5, Naive Bayes, Multi-Layer Perceptron (MLP), NB-Tree, and Decision Tree for detecting attacks, including DoS and Distributed Denial of Service (DDoS). The Decision Tree, combined with feature selection via RFE, achieved an accuracy of 99%, while the XGBoost algorithm, optimized in a parallel computing environment, recorded a detection rate of 99.60%. Although the results are promising, the authors highlight that traditional machine learning methods have limitations compared to deep learning techniques, which are more suited for modeling complex data.

Moreover, Pranto et al. (2022) proposed a method using classifiers such as k-NN, Decision Tree, Naive Bayes, Logistic Regression, and Random Forest. Their approach, applied to the NSL-KDD dataset, showed that Random Forest with 1400 trees achieved an accuracy of 99.5% and an alarm rate of 0.6%. However, although performance metrics like the ROC curve and AUC were included, the study did not address potential risks of overfitting, especially with such a high number of trees. Stricter regularization or reducing the number of trees could have improved the model's generalization.

Similarly, Mohammed and Gbashi (2021) proposed a method combining (DNN) and Recurrent Neural Networks (RNN) with RFE. Their DNN model achieved 94% accuracy in binary classification, while the RNN model, used to classify five categories (Normal, DoS, Probe, R2L, U2R), demonstrated strong results, with 96% accuracy for the DoS class and 94% for U2R. However, although preprocessing and normalization techniques were applied, their method does not address the imbalance among attack classes, which may limit the model's generalization for less frequent attack types.

Kikissagbe et al. (2024) proposed an approach aimed at improving the DoS attacks in IoT systems based on the use of the Edge IIoT dataset. Their method integrates class balancing techniques, including SMOTE and RUS, to address class imbalance issues. At the same time, they applied feature selection techniques such as DNN, RF, and PCA and tested four classifiers (SVM, DNN, XGBoost, RF) across six combinations of techniques, where the SMOTE + DNN combination achieved the best results with an accuracy of 0.9962. However, the study is limited to DoS attacks. Other types of threats in IoT systems, such as adversarial attacks and Man-in-the-Middle attacks, could be further explored.

Similarly, Meena and Choudhary (2017) proposed an intrusion detection framework using the J48 Graft decision tree, validated on the NSL-KDD datasets. Their study highlighted J48's effectiveness in detecting attacks like DoS and SYN Flood, achieving 99.435% accuracy, outperforming Naive Bayes. However, the study did not address class imbalance in the dataset, where normal connections are predominant and some attacks are underrepresented, potentially skewing results in favor of majority classes.

In parallel, Chandra et al. (2017) used the MRFA (Modified Random Forest Algorithm), an enhanced version of Random Forest, for intrusion detection on the KDD-Cup 99 and NSL-KDD datasets. They achieved an accuracy of 97.9% on KDD-Cup 99 and 99.25% on NSL-KDD. Although the MRFA algorithm yielded promising results, the study did not address certain limitations, such as the computation time compared to simpler models, which could pose a problem for real-time applications. Additionally, the impact of class imbalance was not considered, a crucial factor that can affect the model's robustness in real-world scenarios where intrusions are generally rare.

Zhang et al. (2021) developed a deep learning model using a stacked autoencoder (SAE) for dimensionality reduction and a Random Forest classifier for intrusion detection, achieving 99.92% accuracy in binary classification and 99.90% in multiclass classification on the CICIDS2017 dataset. This approach effectively handles high-dimensional data. However, the study does not analyze the computational cost, which could limit its applicability for real-time intrusion detection. Additionally, while min–max normalization improves scaling, it does not necessarily ensure the model's effectiveness in the presence of outliers, which can degrade the model's performance.

Similarly, Xu et al. (2021) proposed a model using a five-layer autoencoder (AE) to detect network anomalies, aiming to enhance cybersecurity against cyberattacks. Their method includes preprocessing to remove outliers, thus reducing data bias. This model achieved 90.61% accuracy and an F1 score of 92.26% on the NSL-KDD dataset. However, in intrusion detection, outliers may correspond to abnormal or unusual behavior and should therefore be handled carefully, as they may represent either legitimate anomalies or errors that require specific handling.

Rawat et al. (2022) evaluated an IDS based on classical techniques and a DNN on the NSL-KDD dataset. Their method, combining PCA and a five-layer DNN, achieved an accuracy of 79.3% with all features, and 75.9% when limited to six features specific to SDN networks. However, although the model displayed good performance on the training set,

its accuracy decreased on the test data. This phenomenon limits the model's ability to generalize to other intrusion datasets. Moreover, using all the features did not considerably improve the results, suggesting that a more relevant feature selection could strengthen the model.

Similarly, Kasongo (2023) also used RNN for IDS, noting that RNN outperformed Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) methods. By applying these approaches to the UNSW-NB15 and NSL-KDD datasets, promising results were obtained, particularly with attacks such as Exploits, Fuzzers, DoS, Reconnaissance, Backdoor, and Shellcode. The LSTM-XGBoost and GRU-XGBoost combinations achieved accuracies of 85.93% and 85.65%, respectively. However, while RNN can capture temporal dependencies, they require intensive computations. Additionally, the study does not consider class imbalance, which could affect detection performance for some less frequent attack classes.

Moreover, Jia et al. (2019) proposed an intrusion detection method based on a DNN applied to the KDD99 and NSL-KDD datasets, targeting DoS, R2L, U2R, and Probe attacks. Their approach achieved an accuracy of over 98%, with an F1 score reaching 98.84%, highlighting the effectiveness of deep learning techniques for intrusion detection in complex network environments. However, this method may face challenges when dealing with imbalanced data or rare attacks, which could affect its overall performance.

Gurung et al. (2019) proposed a IDS based on a deep learning model, combining a sparse autoencoder and logistic regression, validated on the NSL-KDD dataset and achieving an overall accuracy of 87.2%. They argue that this approach enhances the ability to distinguish normal traffic from malicious traffic. However, the model uses the 115 features derived from one-hot encoding all categorical variables in the dataset, without any prior feature selection process. While this allows for a comprehensive representation of the data, the lack of relevant feature selection may introduce unnecessary noise, which could harm the model's accuracy due to the inclusion of superfluous variables.

Additionally, Khan and Mailewa (2023) proposed a hybrid method combining PCA with SVM and Deep Autoencoders (DAE), achieving a 95% accuracy for DoS attacks, thereby improving the effectiveness of combined approaches. However, this method presents an important limitation, namely the model achieves a low detection rate for U2R and L2R attacks, as these attacks are less represented in the training data, which is due to a lack of class balancing in the data.

4. Methodology

The approach that we have proposed for intrusion detection is structured in six key steps, each playing an essential role in the classification process and the evaluation of the performance of the models. The proposed model is illustrated in Fig. 5. The methodology steps are as follows: Section 4.1 presents the data collection process, while Section 4.2 addresses the data preprocessing steps, including handling categorical variables, feature selection, and data normalization, followed by Section 4.3, which deals with the stratified cross-validation used for splitting the data. The class balancing using SMOTE is addressed in Section 4.4, followed by the classification process in Section 4.5. Finally, the model's performance evaluation is detailed in Section 4.6.

4.1. Data collection

In the current literature, the NSL-KDD dataset (Gurung et al., 2019; Mohammed & Gbashi, 2021; Xu et al., 2021) is widely used as a benchmark standard for evaluating IDS, particularly in experimental studies involving various machine learning techniques. For this reason, we selected this dataset to conduct our own evaluations.

NSL-KDD is an improved version of the KDD Cup 99 dataset (Tavallaee et al., 2009), from which redundant records have been removed. It consists of simulated data representing computer network connections.

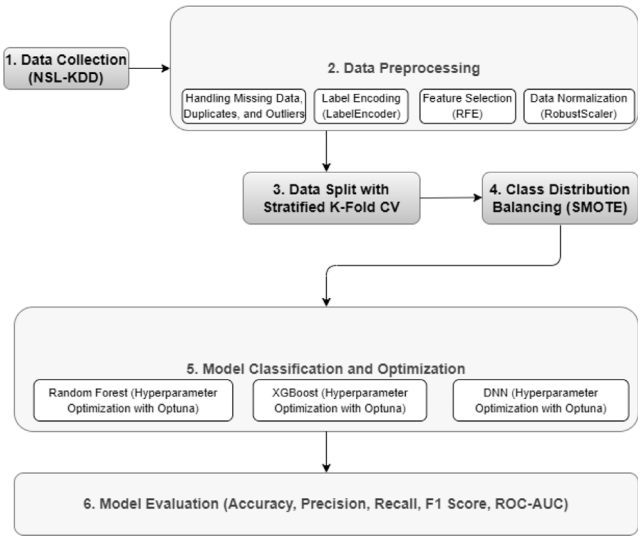

Fig. 5. Proposed approach for an intrusion detection system.

The dataset contains a total of 25,192 instances, described by 42 features. These features include protocol type, connection duration, network service type, and the number of failed connections, among which three are categorical in nature.

In addition, a target variable is provided to indicate whether a connection is considered normal or anomalous. Attacks are categorized into four types: DoS, Probe, R2L, and U2R, all grouped under a single anomalous class. This is illustrated in the first step: Data Collection (NSL-KDD) in Fig. 5.

4.2. Data preprocessing

In this study, we applied various preprocessing techniques to prepare the data for the training of models. This included addressing missing values, removing duplicates, handling outliers, and managing categorical variables. We also performed feature selection to ensure that only the most relevant features were used during model training. These steps are essential for enhancing the performance of machine learning models by reducing noise and optimizing the data for improved generalization. This process is represented in step 2: Data Preprocessing in Fig. 5.

4.2.1. Management of categorical variables

The classifier cannot efficiently process raw data due to certain qualitative variables. Thus, we applied the LabelEncoder to transform these categorical values into numerical representations. This approach allows us to retain categorical information while ensuring that machine learning algorithms can effectively process the data. This is illustrated in step 2: Preprocessing Label Encoding (LabelEncoder) in Fig. 5.

4.2.2. Feature selection

Feature selection is a crucial step in preprocessing as it helps identify and retain the most relevant features. This reduces model complexity while improving its accuracy and generalization ability. The feature selection method used in this research is RFE. RFE is a selection technique that iteratively eliminates the least relevant features while retaining those that contribute the most to the model's performance (Urmi et al., 2024). This approach is particularly effective for high-dimensional datasets as it reduces overfitting while enhancing the model's interpretability. This is represented in step 2: Preprocessing Feature Selection (RFE) in Fig. 5.

4.2.3. Data normalization

After selecting the relevant features, it is necessary to scale the data, an essential preprocessing step that normalizes the data to a specific range. This optimizes the calculation speed of the algorithms. We chose the RobustScaler because it is less sensitive to outliers compared to other scaling methods, making it suitable when the dataset contains outliers. The RobustScaler scales the data by removing the median and adjusting the values based on the interquartile range (IQR). This method is robust to outliers because it relies on statistics that are less sensitive to them (Rashmi & Shantala, 2024). This is represented in step 2: Data Normalization (RobustScaler) in Fig. 5.

4.3. Data splitting with stratified cross-validation

In our approach, we implemented K-fold cross-validation to split the dataset into k subsets of nearly equal sizes. In each iteration, one of the k subsets serves as the test set, while the others form the training set. The stratified version ensures a class distribution in each subset that reflects that of the complete dataset, which is particularly beneficial in cases of class imbalance (Mahesh et al., 2023). This is represented in step 3: Data Split with Stratified K-Fold CV in Fig. 5.

4.4. Class balancing with SMOTE

To correct class imbalances, we applied the SMOTE technique, which generates synthetic examples for the minority class. This improves model performance by preventing it from being biased toward the majority class, making the model more robust, especially when certain classes are under-represented (Jiang et al., 2020). This is represented in step 4: *Class Distribution Balancing (SMOTE)* in Fig. 5.

4.5. Classification

This step focuses on training classification models with hyperparameter optimization. We compared the performance of classification algorithms such as Random Forest, XGBoost, and DNN, using Optuna to find the best hyperparameters. Optuna relies on an advanced technique, the Tree-structured Parzen Estimator (TPE), which focuses on the most promising hyperparameter combinations while discarding less relevant trials (Hanifi et al., 2024). This approach allows for a more efficient exploration of hyperparameter space regions where the likelihood of improving model performance is higher, thus reducing the time and resources required for optimization. The choice of hyperparameter values depends on the complexity of the data, the need to prevent overfitting, and the available computing resources. These models are represented in step 5: *Model Classification and Optimization* in Fig. 5.

4.6. Model evaluation

During model building, each model was trained k times, as we divided our data into k folds using cross-validation. Thus, performance is evaluated at each iteration, and an average is calculated to provide an overall performance assessment of the model. The models were evaluated using several performance metrics such as accuracy, precision, recall, F1-Score, and ROC-AUC curve. These metrics allow for comparison of model performance and identification of which is the most effective. This process is represented in the final step: *Model Evaluation* in Fig. 5.

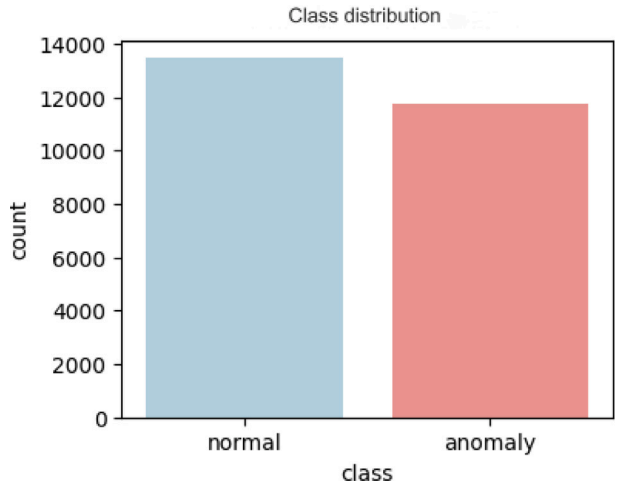

Fig. 6. Class distribution.

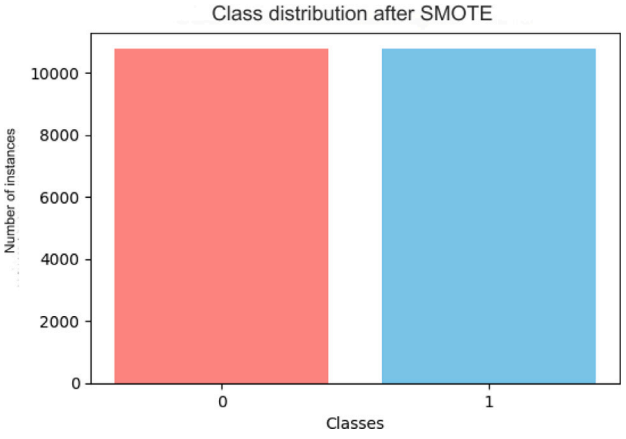

Fig. 7. Class distribution after SMOTE.

5. Experiments

To implement and evaluate the models, we used Python with the Scikit-learn and Keras libraries. Before training the models, the dataset was preprocessed, which reduced the data to 20 columns with 20,153 instances for training and 5039 for testing. The dataset contains no missing values or duplicates; however, some columns contain outlier values. The class distribution before and after the application of SMOTE is presented in Figs. 6 and 7.

For each model, the hyperparameters were optimized using Optuna, which employs the TPE method to find optimal combinations. After optimization, the best parameters obtained are as follows:

- **Random Forest**: *n_estimators = 85, max_depth = 23, min_samples_split = 5, min_samples_leaf = 1, max_features = 0.5866, bootstrap = False*
- **XGBoost**: *n_estimators = 140, max_depth = 10, learning_rate = 0.1184, gamma = 0.0041, colsample_bytree = 0.3235, subsample = 0.8982, reg_alpha = 0.00042, reg_lambda = 1.5426e−08*
- **DNN**: *n_layers = 2, n_units = 128, dropout_rate = 0.3303, activation = tanh, learning_rate = 0.00071*

To provide further insight into the model's decision-making process, we performed a detailed analysis of the features selected using RFE. This technique enabled us to identify the most influential features based on the best-performing model, Random Forest. Table 1 presents the

Table 1
Selected features by RFE with their importance scores (Random Forest).

Feature	Importance score
src_bytes	0.440841
dst_bytes	0.175580
flag	0.077083
same_srv_rate	0.055170
protocol_type	0.033939
dst_host_srv_count	0.031876
diff_srv_rate	0.031439
count	0.022363
dst_host_same_src_port_rate	0.020331
srv_count	0.018410
service	0.018236
hot	0.016697
logged_in	0.012228
dst_host_same_srv_rate	0.012056
dst_host_srv_diff_host_rate	0.009594
dst_host_diff_srv_rate	0.008571
dst_host_rerror_rate	0.005163
dst_host_count	0.004471
dst_host_serror_rate	0.003475
dst_host_srv_serror_rate	0.002477

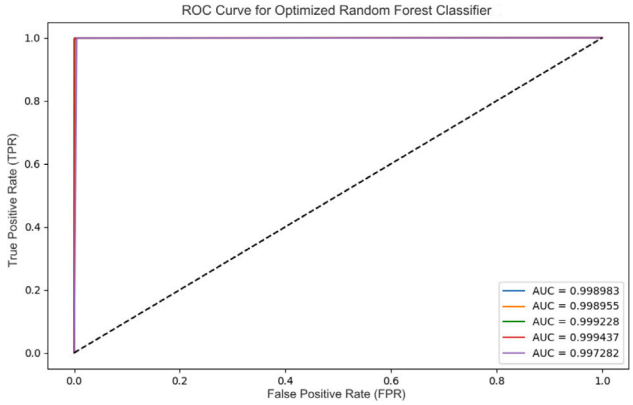

Fig. 8. ROC curve for Random Forest classifier.

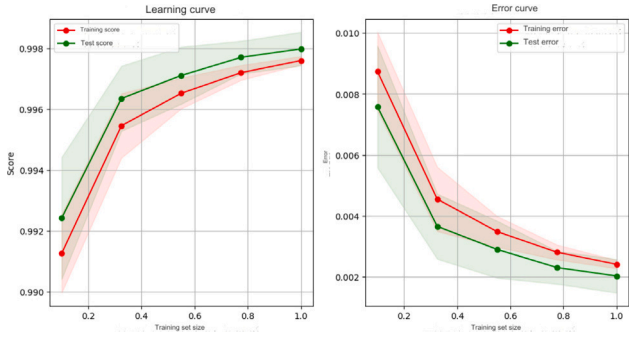

Fig. 9. Learning curve and error curve of the Random Forest classifier.

selected features along with their corresponding importance scores. This information can guide future feature engineering efforts in IDS.

6. Results and interpretation

The performances of the optimized models were evaluated on the test set (5039 instances). The metrics used include accuracy, precision, recall, F1-score, and the ROC-AUC curve. The experimental results are detailed in this section where we compare the models based on these metrics.

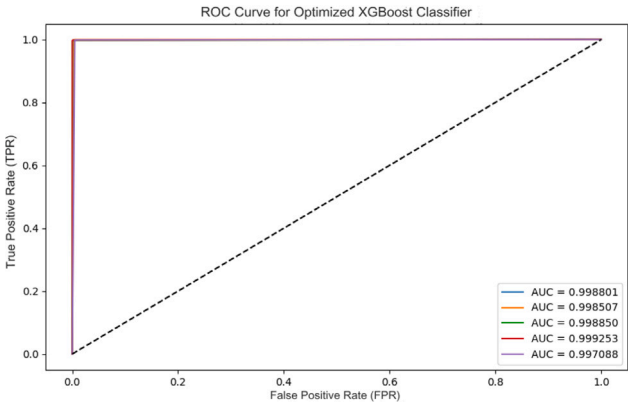

Fig. 10. ROC curve for XGBoost.

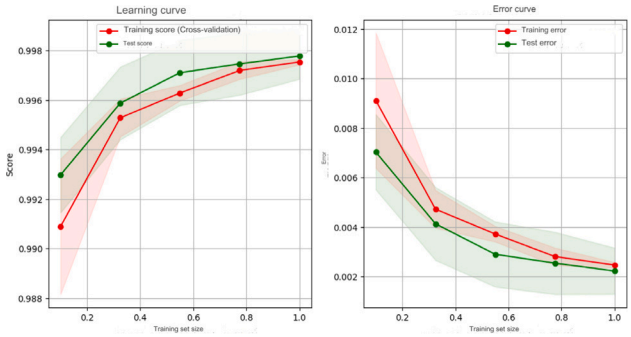

Fig. 11. Learning curve and error curve of the XGBoost classifier.

The Random Forest classifier achieved outstanding results, with an accuracy of 99.80%. Fig. 8 presents the ROC curve of the RF model, highlighting its ability to effectively distinguish between classes, with an average AUC of 0.9988 ± 0.0008, indicating high predictability. Regarding the learning curve, shown in Fig. 9, it demonstrates that the training and testing results converge with a consistent decrease in errors. This suggests that the model improves with more training data and reduces the gap between training and testing errors, indicating good generalization without overfitting.

The performance of the XGBoost classifier was also remarkable, with an accuracy of 99.79% and an average AUC of 0.9985 ± 0.0007, as shown in Fig. 10, indicating a high predictive capability. In Fig. 11, continuous improvement in performance can be observed as the data evolves, with decreasing errors, demonstrating good generalization of the model.

The accuracy of the DNN classifier reached 98.66%, reflecting a high level of precision. According to Fig. 12, the ROC curve of the DNN model shows an average AUC of 0.9872 ± 0.0007, which is slightly lower than those of Random Forest and XGBoost, but still performs well. The learning curve, represented in Fig. 13, illustrates effective generalization of the model, while the error curve reflects a continuous reduction in errors, indicating effective learning.

The comparison of ROC curves displayed in Fig. 14 shows that Random Forest and XGBoost offer very similar performances, with Random Forest slightly ahead, while the DNN model falls slightly below.

Table 2 shows that Random Forest and XGBoost display almost equivalent performances, with a slight advantage for Random Forest. Although the DNN shows good performance, it is slightly behind. The goal is to achieve the best possible performance, making Random Forest the preferred choice in this case, closely followed by XGBoost. The DNN may require additional adjustments to reach a comparable level of performance.

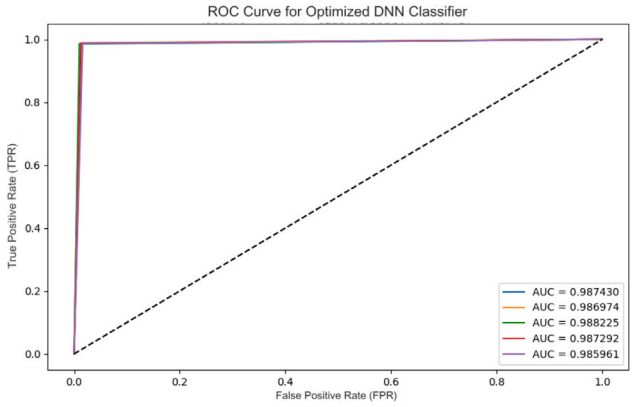

Fig. 12. ROC curve for DNN.

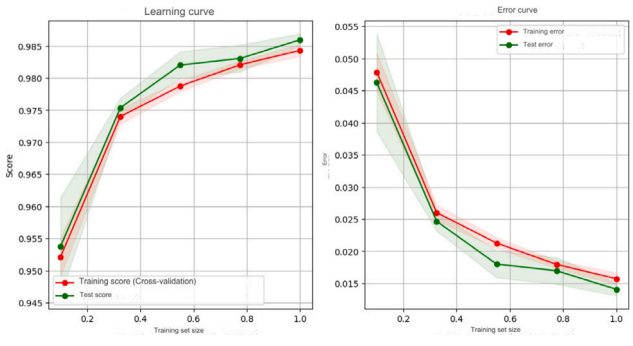

Fig. 13. Learning curve and error curve of the DNN classifier.

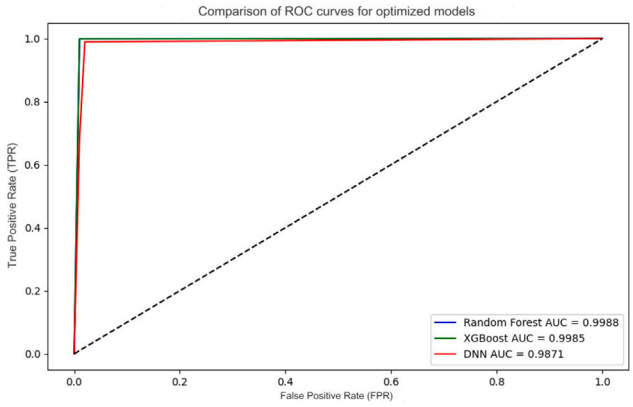

Fig. 14. Comparison of ROC curves.

Table 2
Comparison of model performance metrics.

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.9980	0.9973	0.9989	0.9981
XGBoost	0.9979	0.9971	0.9989	0.9980
DNN	0.9865	0.9878	0.9868	0.9873

Table 3
Comparison of model performance using the NSL-KDD dataset.

Ref.	Techniques	Accuracy
Ahmad et al. (2018)	SVM, RF, ELM	99.67%
Azam et al. (2023)	DT, XGB	99.60%
Pranto et al. (2022)	RF, k-NN, NB, DT	99.50%
Mohammed and Gbashi (2021)	DNN, RNN	94.00%
Meena and Choudhary (2017)	J48, NB	99.43%
Chandra et al. (2017)	MRFA	99.25%
Xu et al. (2021)	AE, RF	90.61%
Rawat et al. (2022)	PCA, DNN	79.30%
Kasongo (2023)	LSTM, GRU	85.93%
Jia et al. (2019)	NDNN	98%
Gurung et al. (2019)	Sparse AE	87.20%
Khan and Mailewa (2023)	PCA, DAE, SVM	95.00%
Our approach	RF, XGB, DNN	**99.80%**

Table 4
Comparison of RF performance based on optimization method and use of SMOTE.

Method	Accuracy	Precision	Recall	F1-score
GridSearch No SMOTE	0.9945	0.9927	0.9970	0.9948
GridSearch + SMOTE	0.9946	0.9928	0.9972	0.9950
Optuna No SMOTE	0.9977	0.9967	0.9987	0.9977
Optuna + SMOTE	0.9980	0.9973	0.9989	0.9981

Table 3 presents a comparison of intrusion detection techniques applied to the NSL-KDD dataset and shows the accuracies achieved by each approach. Our method, based on Random Forest (RF), stands out with an accuracy of 99.80%, surpassing all other techniques.

6.1. Ablative experiments: Respective impact of SMOTE and Optuna

To assess the individual effects of SMOTE and Optuna on our final model, we performed ablative experiments. The results comparing their impacts are shown in Table 4.

We conducted experiments using the same hyperparameters for configurations with and without SMOTE, both for GridSearch and Optuna optimization, to ensure a reliable comparison.

Impact of SMOTE:

The addition of SMOTE results in a slight improvement in recall, increasing from 0.9970 to 0.9972 with GridSearch, and from 0.9987 to 0.9989 with Optuna. Similarly, the F1-score shows a minor increase, from 0.9948 to 0.9950 with GridSearch, and from 0.9977 to 0.9981 with Optuna. This modest improvement is mainly due to the NSL-KDD dataset being relatively balanced, which limits the overall effect of SMOTE on performance.

Although these improvements in recall and F1-score appear subtle, they have a crucial impact in the context of IDS, as even a marginal increase in recall can lead to correct detection and more attacks, thereby reducing false negatives that could critically compromise network security. In effect, SMOTE enhances the model's ability to detect underrepresented minority attack classes, directly improving detection reliability and operational performance.

Thus, these findings highlight the important role of SMOTE in enhancing IDS effectiveness, representing incremental yet essential gains in attack detection.

Impact of Optuna compared to GridSearch:

When comparing models without SMOTE, Optuna optimization notably improves overall performance, with accuracy rising from 99.45% to 99.77%, and precision increasing from 0.9927 to 0.9967. These improvements are also observed in models using SMOTE, where Optuna

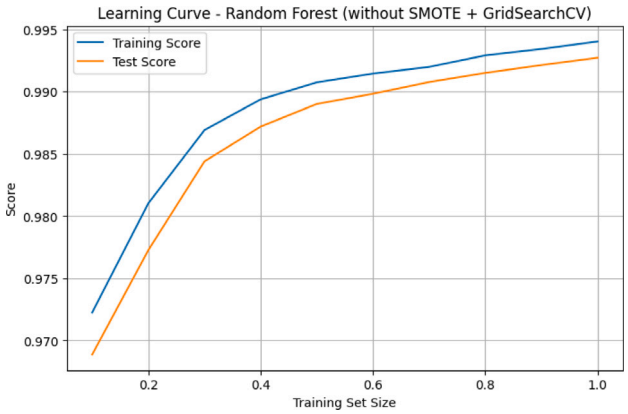

Fig. 15. Learning curve of the Random Forest model without SMOTE + Grid search.

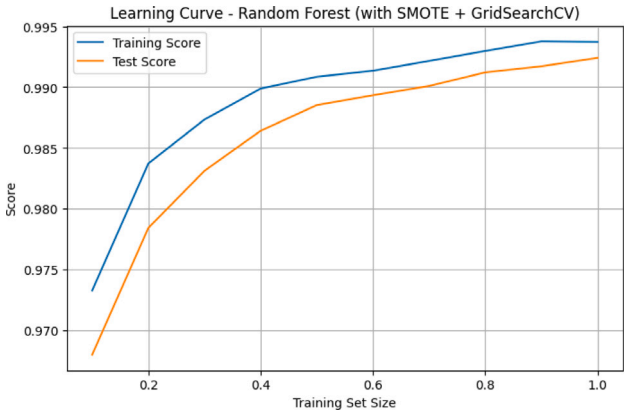

Fig. 16. Learning curve of the Random Forest model with SMOTE and Grid search.

achieves the best performance in terms of accuracy (from 99.46% to 99.80%) and precision (from 0.9928 to 0.9973).

While SMOTE specifically strengthens the detection of minority classes, as evidenced by the improved recall and F1-score, these results clearly demonstrate that Optuna enables more effective hyperparameter tuning than GridSearch, substantially enhancing the model's predictive capability. The gains in precision translate into more reliable network traffic classification. In the context of an IDS, greater precision reduces false alarms, improves the detection of actual attacks, and directly reinforces the overall robustness and effectiveness of the security system (see Figs. 15–17).

6.2. Computational cost analysis of the evaluated models

This subsection presents a comparative analysis of the training, inference, and optimization times required by each evaluated model. These three stages respectively reflect how long it takes to fit the model, make predictions, and perform hyperparameter tuning. Table 5 provides an overview that helps assess the computational cost of each model and highlights the trade-offs between predictive accuracy and processing efficiency.

As shown in Table 5, the trade-offs between performance and computational efficiency among the three evaluated models are highlighted. XGBoost stands out for its very fast training and inference

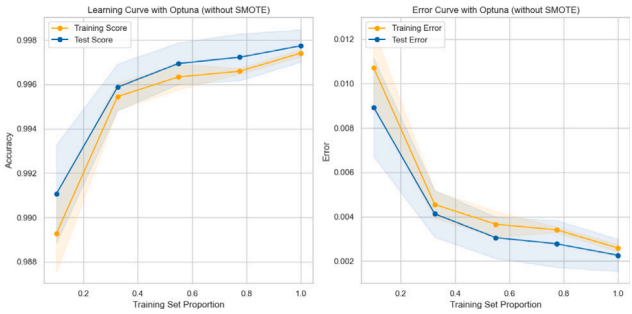

Fig. 17. Learning curve of the RF model with Optuna (without SMOTE).

Table 5
Comparison of training, inference, and hyperparameter optimization times (in seconds) for each model.

Model	Train (s)	Inference (s)	Optimization (s)
RF	21.37	0.0297	683.32
XGBoost	3.04	0.0163	541.94
DNN	27.75	0.2274	512.46

times, making it particularly suitable for scenarios requiring rapid deployment or real-time detection. The DNN, although slower especially during inference benefits from faster hyperparameter optimization compared to the RF.

RF offers the highest accuracy, but at the cost of longer optimization time due to the complexity of its ensemble architecture and hyperparameter tuning.

Thus, RF ensures optimal detection performance, while XGBoost provides a compelling alternative for applications where computational resources or latency are major constraints. The DNN may require further tuning to match the effectiveness of tree-based methods.

7. Discussion and future work

The experiments conducted in this research project focused specifically on the NSL-KDD dataset, which limits the applicability of the results to other datasets and real-world scenarios. Testing these models on diverse and real-world data would provide a more accurate assessment of their performance in practical situations. Future work could also consider using more recent and comprehensive datasets, such as CICIDS2017 or CSE-CIC-IDS2018, as well as evaluating the models on real network traffic when such data become available.

Furthermore, despite demonstrating good performance, machine learning models for intrusion detection face a significant limitation due to their vulnerability to adversarial attacks. These attacks involve malicious inputs crafted to deceive machine learning models while maintaining a legitimate appearance (Kurakin et al., 2016). Two scenarios can be distinguished: white-box attacks, where the attacker knows the model's architecture and parameters, and black-box attacks, where only the outputs are observed to infer the model's behavior. In both cases, an attacker can modify characteristics of malicious traffic (e.g., by masking certain patterns or adding controlled noise) to cause it to be misclassified as benign.

To make IDS more robust against these attacks, several research directions should be explored. Training models on augmented datasets containing adversarial examples appears particularly promising. Another approach involves modeling attacks and designing appropriate countermeasures to better protect IDS against intelligent adversaries

within the framework of adversarial machine learning (Huang et al., 2011).

Finally, adapting to unknown attacks remains a significant challenge. A hybrid approach that combines supervised and unsupervised learning particularly through clustering techniques prior to classification could enhance the detection of emerging threats.

8. Conclusion

The implementation of an effective IDS model remains a significant challenge in the field of cybersecurity. The objective of this study was to develop machine learning models for intrusion detection namely Random Forest, XGBoost, and DNN and to evaluate their performance using the NSL-KDD dataset.

The results obtained in this study demonstrate the effectiveness of the adopted methodology and the selected techniques for intrusion detection, particularly considering the issue of data imbalance. Random Forest achieved an accuracy of 99.80% and an AUC of 0.9988, outperforming XGBoost (99.79%, AUC of 0.9985) and DNN (98.66%, AUC of 0.9872). This highlights the superior capability of Random Forest to accurately distinguish between normal connections and intrusions compared to the other models.

In comparison with related work, approaches based on XGBoost (Azam et al., 2023) and Random Forest (Pranto et al., 2022) have reported accuracies of around 99.60% and 99.50%, respectively, while DNN (Mohammed & Gbashi, 2021) achieved an accuracy of 94%. It is also worth noting that other models, such as SVM and ELM (Ahmad et al., 2018), achieved an accuracy of 99.67%, whereas AE (Xu et al., 2021) reported a lower accuracy of 90.61%. Furthermore, some hybrid models like LSTM or GRU (Kasongo, 2023), although effective, did not exceed 90% accuracy.

This work proposes a comparative framework to evaluate three machine learning algorithms Random Forest, XGBoost, and DNN under consistent conditions, including ablation studies with and without SMOTE, as well as hyperparameter optimization using GridSearch and Optuna. This approach enables a reproducible and balanced assessment of the effects of data balancing and parameter tuning techniques. The detailed comparison of classification performance alongside computational costs offers practical insights to guide the selection of IDS models tailored to operational constraints.

By applying this methodology to the NSL-KDD dataset, the study identifies the most effective algorithm for intrusion detection, contributing to the broader literature on machine learning in IDS. The results also show an improvement in detection accuracy compared to related works, highlighting the relevance of the chosen approach.

CRediT authorship contribution statement

Sow Thierno Hamidou: Conceptualization, Methodology, Writing – review & editing. **Adda Mehdi:** Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Ahmad, I., Basheri, M., Iqbal, M. J., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access*, *6*, 33789–33795. http://dx.doi.org/10.1109/ACCESS.2018.2841987.

Al-Maksousy, H. H., Weigle, M. C., & Wang, C. (2018). NIDS: Neural network based intrusion detection system. In *2018 IEEE international symposium on technologies for homeland security* (pp. 1–6). IEEE.

Aldallal, A., & Alisa, F. (2021). Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry*, *13*(12), 2306.

Azam, Z., Islam, M. M., & Huda, M. N. (2023). Comparative analysis of intrusion detection systems and machine learning-based model analysis through decision tree. *IEEE Access*, *11*, 80348–80391.

Catania, C. A., & Garino, C. G. (2012). Automatic network intrusion detection: Current techniques and open issues. *Computers & Electrical Engineering*, *38*(5), 1062–1072.

Chan, P. K., Fan, W., Prodromidis, A. L., & Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and their Applications*, *14*(6), 67–74.

Chandra, P., Lilhore, U. K., & Agrawal, N. (2017). Network intrusion detection system based on modified random forest classifiers for KDD Cup-99 and NSL-KDD dataset. *International Research Journal of Engineering and Technology*, *4*, 786–791.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Dhaliwal, S. S., Nahid, A. A., & Abbas, R. (2018). Effective intrusion detection system using XGBoost. *Information*, *9*(7), 149.

Gurung, S., Ghose, M. K., & Subedi, A. (2019). Deep learning approach on network intrusion detection system using NSL-KDD dataset. *International Journal of Computer Network and Information Security*, *11*(3), 8–14.

Hanifi, S., Cammarono, A., & Zare-Behtash, H. (2024). Advanced hyperparameter optimization of deep learning models for wind power prediction. *Renewable Energy*, *221*, Article 119700.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., & Tygar, J. D. (2011). Adversarial machine learning. In *Proceedings of the 4th ACM workshop on security and artificial intelligence* (pp. 43–58).

Jia, Y., Wang, M., & Wang, Y. (2019). Network intrusion detection algorithm based on deep neural network. *IET Information Security*, *13*(1), 48–53. http://dx.doi.org/10.1049/iet-ifs.2018.5176.

Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access*, *8*, 32464–32476.

Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Computer Communications*, *199*, 113–125. http://dx.doi.org/10.1016/j.comcom.2023.03.017.

Khan, S. S., & Mailewa, A. B. (2023). Detecting network transmission anomalies using autoencoders-SVM neural network on multi-class NSL-KDD dataset. In *2023 IEEE 13th annual computing and communication workshop and conference* (pp. 0835–0843). IEEE.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*, 20.

Kikissagbe, B. R., Adda, M., Célicourt, P., Haman, I. T., & Najjar, A. (2024). Machine learning for DoS attack detection in IoT systems. *Procedia Computer Science*, *241*, 195–202.

Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236.

Mahesh, T. R., Geman, O., Margala, M., & Guduri, M. (2023). The stratified K-folds cross-validation and class-balancing methods with high-performance ensemble classifiers for breast cancer classification. *Healthcare Analytics*, *4*, Article 100247.

Meena, G., & Choudhary, R. R. (2017). A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In *2017 international conference on computer, communications and electronics* (pp. 553–558). IEEE.

Mohammed, B., & Gbashi, E. K. (2021). Intrusion detection system for NSL-KDD dataset based on deep learning and recursive feature elimination. *Engineering and Technology Journal*, *39*(7), 1069–1079.

Pranto, M. B., Ratul, M. H. A., Rahman, M. M., Diya, I. J., & Zahir, Z. B. (2022). Performance of machine learning techniques in anomaly detection with basic feature selection strategy–a network intrusion detection system. *Journal of Advances in Information Technology*, *13*(1).

Raju, V. N., Saravanakumar, R., Yusuf, N., Pradhan, R., Hamdi, H., Saravanan, K. A., & Askar, M. A. (2024). Enhancing emotion prediction using deep learning and distributed federated systems with SMOTE oversampling technique. *Alexandria Engineering Journal*, *108*, 498–508.

Rashmi, C. R., & Shantala, C. P. (2024). Evaluating deep learning with different feature scaling techniques for EEG-based music entrainment brain computer interface. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, *7*, Article 100448.

Rawat, S., Srinivasan, A., Ravi, V., & Ghosh, U. (2022). Intrusion detection systems using classical machine learning techniques vs integrated unsupervised feature learning and deep neural network. *Internet Technology Letters*, *5*(1), Article e232. http://dx.doi.org/10.1002/itl2.232.

Srinivas, P., & Katarya, R. (2022). hyOPTXg: OPTUNA hyper-parameter optimization framework for predicting cardiovascular disease using XGBoost. *Biomedical Signal Processing and Control*, *73*, Article 103456.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications* (pp. 1–6). IEEE.

Tran, N., Schneider, J. G., Weber, I., & Qin, A. K. (2020). Hyper-parameter optimization in classification: To-do or not-to-do. *Pattern Recognition*, *103*, Article 107245.

Urmi, W. F., Uddin, M. N., Uddin, M. A., Talukder, M. A., Hasan, M. R., Paul, S., & Imran, F. (2024). A stacked ensemble approach to detect cyber attacks based on feature selection techniques. *International Journal of Cognitive Computing in Engineering*, *5*, 316–331.

Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., & Sabrina, F. (2021). Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. *IEEE Access*, *9*, 140136–140146. http://dx.doi.org/10.1109/ACCESS.2021.3118587.

Zhang, C., Chen, Y., Meng, Y., Ruan, F., Chen, R., Li, Y., & Yang, Y. (2021). A novel framework design of network intrusion detection based on machine learning techniques. *Security and Communication Networks*, *2021*(1), Article 6610675. http://dx.doi.org/10.1155/2021/6610675.