Decision Making Advances

Journal homepage: www.dma-journal.org
ISSN: 2956-2384

Towards Robust Network Security: Evaluating Machine Learning Algorithms for Intrusion Detection

Hafiz Burhan Ul Haq[1*], Rabia Younis[1], Muhammad Shujat Ali[1]

[1] Department of Information Technology, Faculty of Computer Science, Lahore Garrison University, Lahore, Pakistan

ARTICLE INFO

Article history:
Received 22 October 2024
Received in revised form 09 November 2024
Accepted 09 November 2024
Available online 09 November 2024

Keywords:
Network Security; Machine Learning; Remove Machine; Intrusion Detection.

ABSTRACT

The constant growth of cyber threats has made network intrusion detection systems (NIDS) more crucial. Targeting anomalous behavior in a network is challenging because of the large number of features that exist. Consequently, the accuracy is affected, and there will be a greater chance of less reliability in the network. This study overcomes the limitations of traditional NIDS by using multiple machine learning (ML) algorithms to enhance intrusion detection capabilities. The efficacy of many ML algorithms in the context of NIDS is examined by paying special attention to their ability to detect intrusion based on features. Similarly, experimental analysis is conducted using a publicly available large dataset containing 41 features, whereby six algorithms were compared: AdaBoost, Gaussian Naive Bayes, Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors, and Multinomial Naive Bayes. Resultantly, SVM achieves the lowest accuracy at 53.15%. RF performs the best with a 99.78% accuracy rate. In addition, a comparative analysis is also performed, which is crucial for practitioners in the industry who want to implement effective NIDS.

1. Introduction

Today, network intrusion detection systems (NIDS) are of great importance in the protection of digital assets in an ever more interconnected world. Along with cyber threats, the strategies to detect and eradicate these threats are also developing with time. According to the World Economic Forum's Global Cybersecurity Outlook 2024, cyberattacks have evolved into more complex attacks, and in the cloud, the environment increasing intrusions and the emergence of malware-free attacks. Today, there are 4.7 million global cybersecurity professionals, a new reality of its grand importance in protecting organizations around the world. But while the number of breaches grew by 13%, as did financial losses, the incidence of ransomware, which is about 72% of all cyber-attacks in 2023, still presents a significant challenge to 72% of businesses [1-2].

[*] *Corresponding author.*
E-mail address: burhanhashmi64@lgu.edu.pk

https://doi.org/10.31181/dma31202559

These stats point out the huge gap between the adversary's capability and the organization's defensive capability. Although significant progress has been made in the field of network intrusion detection (NID), numerous existing persistent challenges hinder traditional and current techniques in actively detecting intrusion. However, classical NIDSs face high false positive rates for such nonstationary network environments, causing trouble in differentiating between legitimate and malicious traffic. The problem compounds itself when working with encrypted traffic or complex attack patterns that copy the normal behavior. As a result, the detection system becomes less efficient since an increased number of false alerts leads to alarm fatigue amongst security teams.

In addition, scalability is an important factor when NID is faced with large volumes of real-time network traffic. High-speed networks often overwhelm traditional systems, creating packet loss and lags in detection. When a model overfits, the model performs extremely well on training data but not at all on new, unseen data, which makes real-world application of the model difficult. Machine learning (ML) models can also be computationally intense and require large amounts of processing power and memory to train and test, which may not be practical for real-time deployment. With a large-scale dataset or complex algorithm, this problem gets worse as it becomes harder to locate the intrusion clues in time. Another problem is that network traffic often contains missing packets, resulting in incomplete data and the resulting loss of detection accuracy. However, such things are even worse in the case of imbalanced datasets with normal traffic being 100 times more than the malicious. Thus, the task of anomaly detection becomes very difficult [3]; i.e. it results in models that try to predict the normal class more preferentially than they wrongly ignore tiny attack patterns [4].

To address these issues, a multimodal approach is applied that is based on ensemble learning, which fuses several models in reducing false positive cases and increasing detection rates and handling of missing and unbalanced data using different techniques in ensemble learning. Such improvements, thus, in addressing a range of dynamic cyber threats, are significant in enhancing the dependability and robustness of NIDS.

In our work, several ML models are used, such as Random Forest (RF), Support Vector Machine (SVM), AdaBoost, K-Nearest Neighbour (KNN), Gaussian Naïve Bayes, and Multinomial Naïve Bayes, to enhance the actual detection feature of NIDS. For example, RF and AdaBoost can be used in the system since they can capture non-linear relationships between features and are less sensitive to overfitting, hence reducing the impact of minority class misclassification. Meanwhile, SVM and KNN assist in classifying the traffic correctly as either good or malicious traffic that is required to point at subtle shifts in the network intrusion. The Gaussian and multinomial Naive Bayes models also have computational advantages, which outperform relatively computationally expensive methods such as SVM in terms of identifying real-time. By integrating these models, the system achieves a trade-off between the accuracy and the cost of computation, making it possible to build a large-scale network strategy. Moreover, the experimental analysis is done on a publicly available NIDS dataset. The effectiveness of our approach is also determined by testing on many instances, using recall, accuracy, and F1-score; it is demonstrated that our approach is more effective as compared to state-of-the-art methods.

The arrangement is as follows. In Section 2, we outline the related literature. Section 3 gives the methodology. Section 4 gives results and analysis. Section 5 deals with conclusion remarks.

2. Literature Review

Susilo and Sari [5] proposed ML as well as deep learning (DL) methods using Scikit-learn, Tensorflow, and Seaborn to improve the security of Internet of Things (IoT) networks. Ashraf et al. [6] studied DL and ML techniques for intrusion detection systems (IDS) in IoT. This research focused

on highlighting the protocol of IoT architecture along with their technologies. Furthermore, they provided an overview of ML/DL methods for intrusion detection and discussed their pros and cons.

IDS using DL and ML models was studied by Liu et al. [7] to improve cybersecurity by having low false alarm rates and high detection rates. Liu et al. [7] used the difficult set sampling technique algorithm for tackling imbalanced data. Initially, ENN was utilized for dividing the data into difficult and easy sets. The compression was done on the majority of samples of difficulty using K-means. Next, zoom-in and zoom-out operations were done to increase minority samples in difficult sets. Finally, the integration of all these sets was performed. This research reduced the misbalancing issues and enhanced better classification values.

Sarker et al. [8] introduced an ML-based security model called the intrusion detection tree or "IntruDTree". They focused on security features based on their importance ranking and then developed a generalized intrusion detection model using a tree structure based on the important features that were chosen. This model was accurate in terms of accurate prediction for unseen test cases. The efficacy of the "IntruDTree" model was ultimately assessed by experimentation using cybersecurity datasets, with the precision, recall, F1-score, accuracy, and ROC values computed for evaluation. Dong et al. [9] reported an ML-based IDS combining multivariate correlation analysis (MCA) and long short-time memory (LSTM). The MCA-LSTM used the feature selection strategy of the information gain method. The test accuracy of MCA-LSTM was 82.15% for the experiment on the NSL KDD dataset on five-way classification. Besides, the accuracy was 77.74% for the 10-way classification job in the UNSW NB15.

Injadat et al. [10] proposed a multistage ML-based framework for the NIDS evaluation using the RF and KNN algorithms to classify the attacking types. Tree Parzen Estimator (TPE) was utilized to enhance hyperparameters. The research results showed that the TPE-optimized RF classifier was more accurate than alternative optimization techniques for detection. A hybrid data optimization technique was introduced, comprising two modules; i.e. selection of features and sampling of data. Similarly, an approach to network attack detection that combines flow calculations and DL was discussed by Zhang et al. [11].

Özalp and Albayrak [12] discussed that the selection of high-frequency features can increase the accuracy of intrusion detection by the adoption of ML algorithms. Talukder et al. [13] presented a hybrid approach that was a combination of ML and DL to enhance security. SMOTE was used for preprocessing to make balanced data, while XGBoost was considered for the feature selection. Besides, ML and DL models were utilized for intrusion detection; i.e. RF, Decision Tree (DT), KNN, Multilayer Perceptron (MLP), Convolution Neural Network (CNN), and Artificial Neural Network (ANN). Chen et al. [14] showed ensemble learning-based NIDS that made intelligent estimations by utilizing a variety of individual ML models. Although experiments using federated learning-based intrusion detection have been conducted [15], our focus is on centralized ML-based IDS.

3. Methodology

Figure 1 shows the proposed methodology. It contains several phases such as "data acquisition", "data preprocessing", and "ML model implementation".

3.1 Data Acquisition

This research starts with gathering a dataset that may be used for network intrusion detection. To benchmark intrusion detection systems, we use publicly accessible datasets, such as the Network Intrusion Detection dataset [16], which is well-known in the cybersecurity world. Numerous prope-

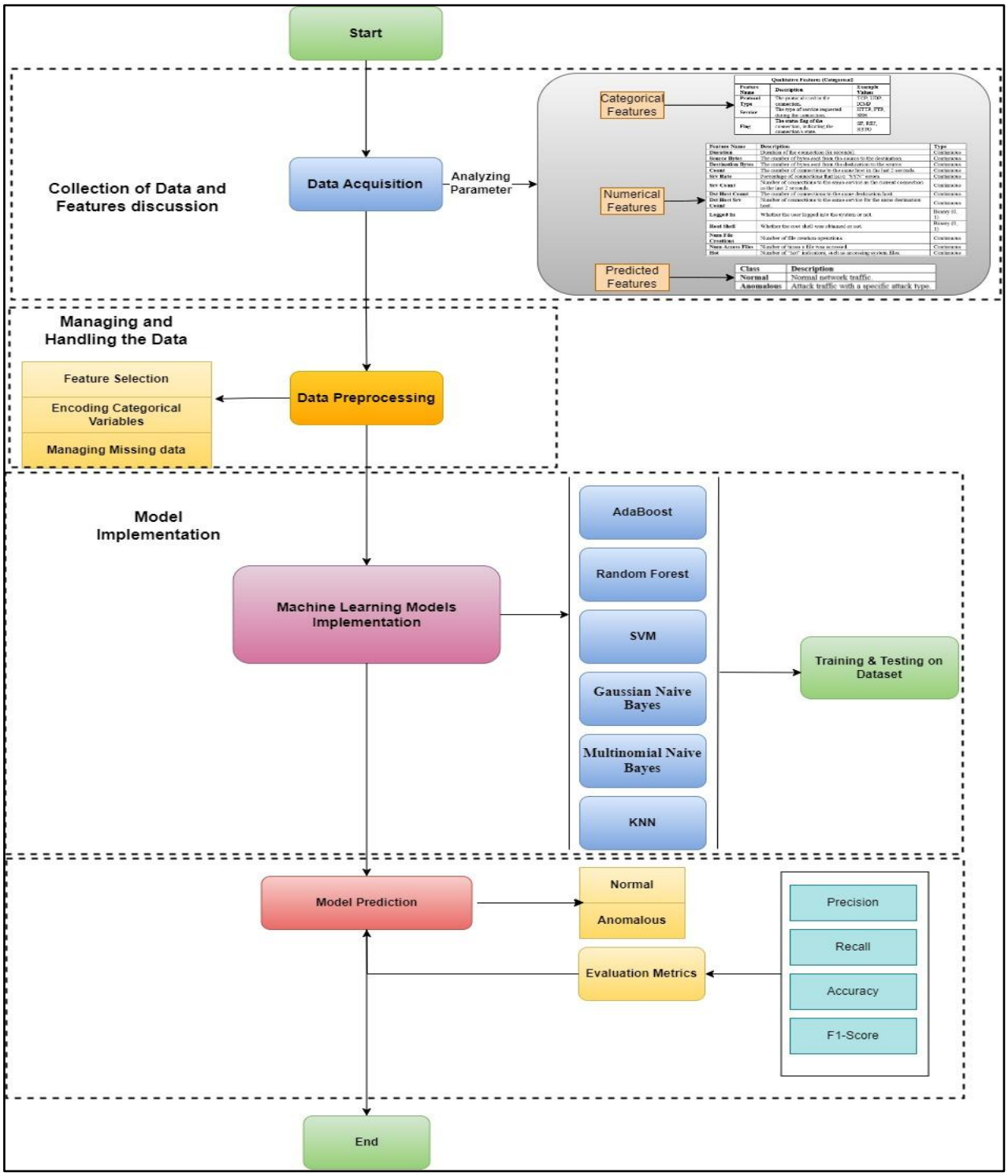

Fig. 1. Workflow diagram of the proposed methodology

rties listed in Figure 2 are included in this dataset, which offers an extensive collection of parameters for evaluation. The dataset for audit includes a wide variety of intrusions within the context of a military network environment. It was intended to monitor unprocessed TCP/IP information by emulating the U.S. Air Force LAN environment. The LAN was made to mimic the real-life environment and tested with one attack after another. In more detail, each connection in the network is described as a sequence of TCP packets, which during some time intervals are exchanged between the source IP and target IP under some protocol. These are the normal connections and the attacks where each

attack could be given the kind of label it belongs to. Individual connection records have an average size of roughly 100 bytes of the file data.

1. Qualitative Features (Categorical)

Feature Name	Description	Example Values
Protocol Type	The protocol used in the connection.	TCP, UDP, ICMP
Service	The type of service requested during the connection.	HTTP, FTP, SSH
Flag	The status flag of the connection, indicating the connection's state.	SF, REJ, RSTO

2. Quantitative Features (Numerical)

Feature Name	Description	Type
Duration	Duration of the connection (in seconds).	Continuous
Source Bytes	The number of bytes sent from the source to the destination.	Continuous
Destination Bytes	The number of bytes sent from the destination to the source.	Continuous
Count	The number of connections to the same host in the last 2 seconds.	Continuous
Srv Rate	Percentage of connections that have "SYN" errors.	Continuous
Srv Count	Number of connections to the same service as the current connection in the last 2 seconds.	Continuous
Dst Host Count	The number of connections to the same destination host.	Continuous
Dst Host Srv Count	Number of connections to the same service for the same destination host.	Continuous
Logged In	Whether the user logged into the system or not.	Binary (0, 1)
Root Shell	Whether the root shell was obtained or not.	Binary (0, 1)
Num File Creations	Number of file creation operations.	Continuous
Num Access Files	Number of times a file was accessed.	Continuous
Hot	Number of "hot" indicators, such as accessing system files.	Continuous

3. Class Variable

Class	Description
Normal	Normal network traffic.
Anomalous	Attack traffic with a specific attack type.

Fig. 2. Workflow diagram of the proposed methodology

In general, for every TCP/IP connection, 41 features including three qualitative and 38 quantitative features are extracted from both normal and anomalous traffic. The class variable is divided into two categories; i.e. normal and anomalous. The primary characteristics of interest are shown in Figure 3.

```
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
       'num_access_files', 'num_outbound_cmds', 'is_host_login',
       'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate', 'class'],
```

Fig. 3. Workflow diagram of the proposed methodology

Class labels, such as regular or anomalous, indicate the kind of traffic. Implementing robust ML models for intrusion detection is made possible by this extensive feature collection. However, the qualitative and quantitative features are very helpful for examining the legitimate and anomalous activities in network traffic. So, after applying the model by considering these features, a network traffic will be more secure, and different types of attacks (e.g. SYN flood attacks, Botnets, phishing attacks, Eavesdropping, etc.) can be identified in a better way. Figure 4 provides an explanation of the data distribution.

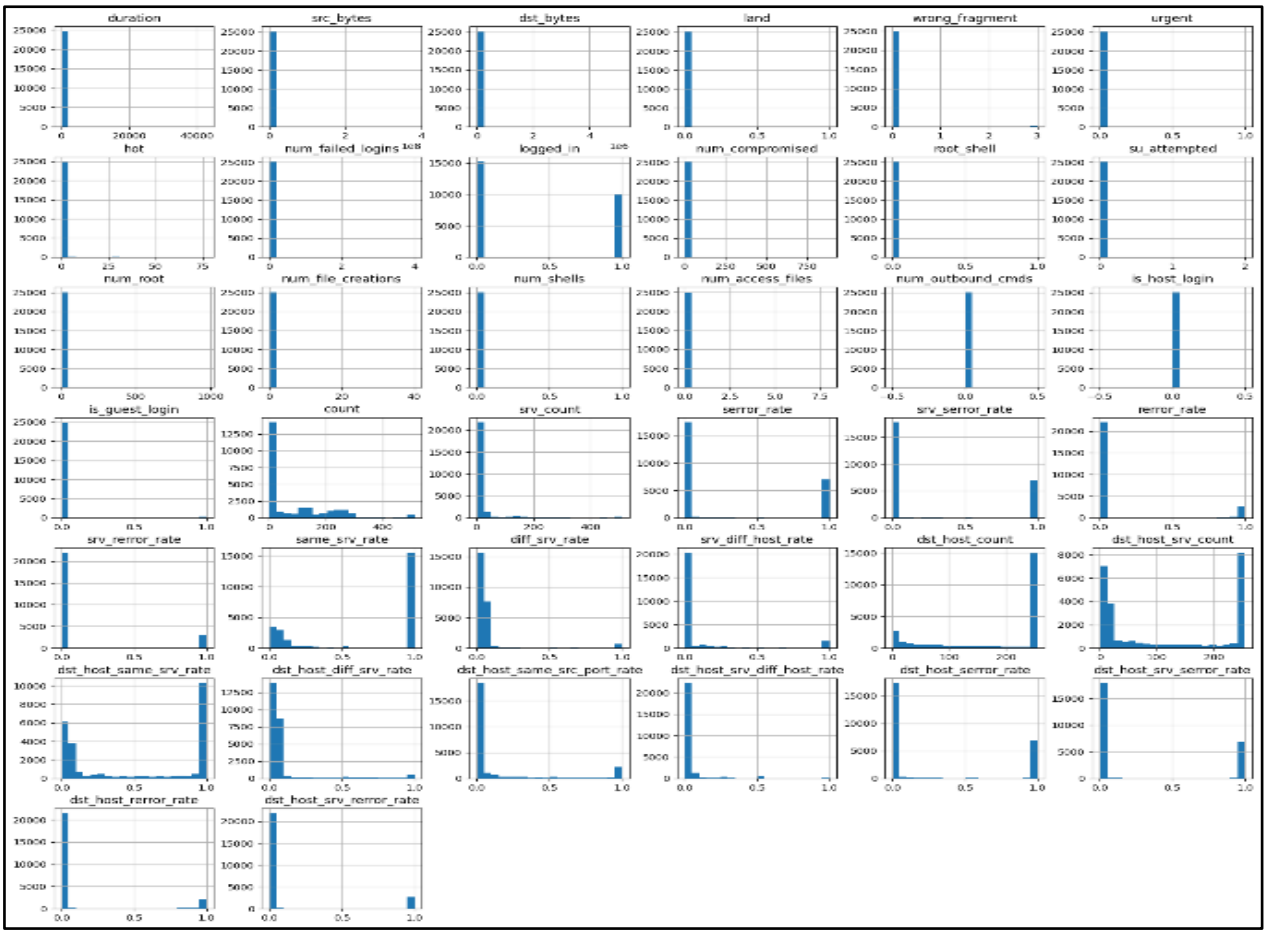

Fig. 4. Distribution of features plots

3.2 Data Exploration and Visualization

Exploratory data analysis (EDA) is implemented to show the distributions of classes and measure the dataset quality to perform preprocessing. Whereas Seaborn is used to draw plots to assess class imbalances that are required for preprocessing. The distribution classes are displayed in Figure 5.

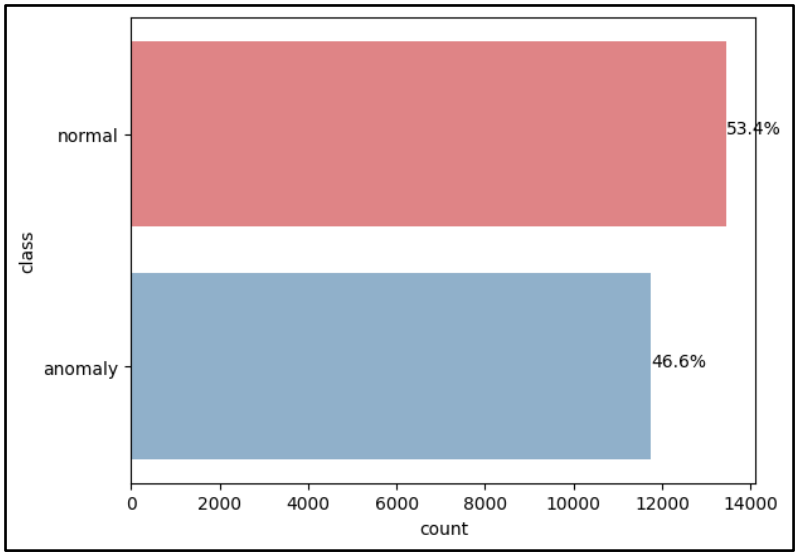

Fig. 5. Distribution of classes

3.3 Data Preprocessing

However, in this phase, preprocessing is done to make the data more essential so that the model can efficiently learn. It can be done in multiple steps:

i. *Feature selection* – The class labels that include normal or anomalous are expressed by the target variable (*y*), and the dataset features are represented by *X*.
ii. *Encoding categorical variables* – The conversion of categorical characteristics into numerical representations is done by using one-hot encoding. To show if the features exist, this process converts each category value to a new categorical column and assigns them a value (1/0), which is a binary value.
iii. *Managing missing data* – Missing data is managed by the simple imputer class imported from Scikit-learn. It replaces each column's mean for any missing numbers or values. This imputation maintains the dataset's size and integrity.

3.4 Model Training

After performing the preprocessing, the dataset is ready for training. So, in this method, a number of ML models are trained on the NIDS dataset. A brief discussion of these models is mentioned below.

3.4.1 Random Forest Classifier

This ensemble learning technique produces several decision trees and then combines their predictions. It also helps to reduce the noise level in the dataset, which is achieved by averaging the result from many trees to reduce the chances of getting overfitted [17].

RF, which averages several decision trees to prevent overfitting, can easily handle large datasets and provide accurate categorization. Each tree is built using a random collection of features and examples.

The final prediction $F(x)$ for a sample x is given by:

$$\boldsymbol{F(x) = \frac{1}{N}\sum_{i=1}^{N} h_i(x)}, \quad (1)$$

where N is the number of trees and $h_i(x)$ is the prediction from the i-th tree.

3.4.2 Support Vector Classifier

Support Vector Classifier (SVC) seeks the best hyperplane in a high dimensional space in order to separate several classes from each other. The idea behind the mathematical representation is trying to develop a better margin between support vectors [18].

SVM is efficient in high-dimensional domains employed to generate hyperplanes that divide several classes (normal and anomalous) inside the feature space.

For a linear SVM, the decision function can be expressed as:

$$\boldsymbol{F(x) = w^T x + b}, \quad (2)$$

where w is the weight vector and b is the bias term. The optimization problem involves minimizing as:

$$\boldsymbol{min}\frac{\mathbf{1}}{\mathbf{2}}\|\boldsymbol{w}\|^{\mathbf{2}}+\boldsymbol{b}$$

subject to: (3)

$$\boldsymbol{y_i}(\boldsymbol{w^t x_i}+\boldsymbol{b})\geq \mathbf{1}, \forall \boldsymbol{i}$$

3.4.3 AdaBoost Classifier

Using this boosting method, a powerful classifier is produced by combining the predictions of weaker ones. In order to reduce bias, each successive classifier concentrates on cases that the one before it misclassified [19].

AdaBoost is used in NIDS applications where it generates a strong classifier by merging many weak classifiers. For a sequence of classifiers, each of them is trained subsequently, proffering more focus on the misclassified examples during previous rounds.

The final classifier $H(x)$ is a weighted sum of the weak classifiers $h_t(x)$:

$$\boldsymbol{H(x)}=\sum_{\boldsymbol{t=1}}^{\boldsymbol{T}}\boldsymbol{\alpha_t h_t(x)}, \quad (4)$$

where α_t is the weight assigned to the h_t classifier based on its accuracy.

3.4.4 K-Nearest Neighbors

When employing Euclidean distance to determine the level of similarity, KNN is an instance-based learning algorithm that puts new instances into the class towards which their nearby neighbors lean [20]. KNN clusters are according to the majority label of the nearest neighbor within the feature space. An important advantage of this method is that the decision boundaries can be complex because the method used is non-parametric and does not depend on pre-specified assumptions.

The predicted class for an instance *x* is expressed by:

$$\hat{\mathrm{y}}=\boldsymbol{argmax_y}\sum_{\boldsymbol{i=1}}^{\boldsymbol{k}}\boldsymbol{I(y_i=y)}, \quad (5)$$

where *k* is the number of neighbors, y_i is the class of the *i*-th neighbor, and *I* is an indicator function.

3.4.5 Gaussian Naive Bayes

Naive Bayes uses the Bayes theorem for its application with the assumption that all features are independent of each other. The Gaussian variation gives good results with continuous data while the Multinomial variation is used on the discrete type of data [21–22]. Naive Bayes works well, particularly for a large number of instances with the assumption that the features it holds are independent of each other. In doing this probability of each class concerning the attributes is computed and then it determines which class is likely to belong to.

The class probabilities are computed as:

$$\boldsymbol{P(y|X)}=\frac{\boldsymbol{P(X/y)P(y)}}{\boldsymbol{P(X)}}, \quad (6)$$

where *X* is the feature vector and *y* is the class.

3.4.6 Multinomial Naive Bayes

Multinomial Naive Bayes is a particularly effective method in terms of the categorization of NID. It is used to feature vectors that represent the counts among other things like the frequency of a specific kind of characteristic in a network traffic. Regarding its mathematical representation, for a class *y* and feature vector *X* we have:

$$\boldsymbol{P}(\boldsymbol{y}|\boldsymbol{X}) \propto \boldsymbol{P}(\boldsymbol{y}) \prod_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{P}(\boldsymbol{x_i}|\boldsymbol{y}), \tag{7}$$

where x_i are the features.

3.5 Algorithm: Intrusion Detection

Input:
- *Dataset train_data with features X and target labels y*
- *Models: Random Forest, SVM, AdaBoost, KNN, Gaussian Naive Bayes, Multinomial Naive Bayes*

Output:
- *Accuracy, confusion matrix, and classification report for each model*

Begin*:*
1. Load the dataset:
X = drop 'class' column from train_data
y = select 'class' column from train_data
2. Identify categorical and numerical columns:
if *categorical columns exist:*
Encode categorical columns using one-hot encoding (pd.get_dummies)
else*:*
Proceed with X as is
end if
3. Split the dataset into training and test sets:
Split X and y into X_train, X_test, y_train, y_test (use train_test_split)
4. Handle missing values:
if *missing values in numerical columns:*
Impute missing values in X_train and X_test using mean strategy
else*:*
Proceed with original data
end if
5. Define models:
models = {RandomForest, SVM, AdaBoost, KNN, GaussianNB, MultinomialNB}
6. Train and evaluate models:
for *each model in models:*
Train model using X_train and y_train
Predict y_test using trained model
Evaluate the model:
Calculate accuracy
Generate confusion matrix
Generate classification report
Store results
end for
7. Display results:
for *each model in models:*
Print model name
Print accuracy
Print confusion matrix
Print classification report
end for
End

3.6 Model Evaluation

Understanding the models' effectiveness in categorizing network traffic depends on their evaluation. To facilitate comparisons in accuracy and other performance measures, each model is assessed according to how well it performs on the test set. Important measurements consist of:

i. *Precision* – The ratio of true positive predictions to the total predicted positives:

$$\boldsymbol{Precision}\ (\boldsymbol{P}) = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}. \tag{8}$$

ii. *Recall* – The ratio of true positive predictions to the total actual positives:

$$\boldsymbol{Recall}\ (\boldsymbol{R}) = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}. \tag{9}$$

iii. *Accuracy* – The ratio of correctly predicted instances to the total instances:

$$\boldsymbol{Accuracy}\ = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}}. \tag{10}$$

iv. *F1-score* – The harmonic mean of precision and recall, providing a balance between the two:

$$\boldsymbol{F1\text{-}score} = \frac{\boldsymbol{2\times P\times R}}{\boldsymbol{P+R}}. \tag{11}$$

4. Results and Discussion

After implementing the ML models the following result was achieved as mentioned in Table 1 and Figure 5.

Table 1
Achieved accuracy of the machine learning models

Models	Accuracy
Random Forest	**0.9978**
SVM	**0.5314**
AdaBoost	**0.9908**
KNN	**0.9849**
Gaussian Naive Bayes	**0.5572**
Multinomial Naive Bayes	**0.5514**

A comparative study of model accuracy finds notable variations between the different algorithms used in NIDS. With an accuracy of 99.78%, the RF model is the most accurate, proving its usefulness in managing extensive datasets and identifying complex patterns related to network intrusions. On the other hand, SVM shows a significantly lower accuracy of 53.14%, which may be related to its difficulties with noisy data and high-dimensional feature spaces, which are frequent in NIDS scenarios. With an accuracy of 98.49%, the AdaBoost model trails closely behind, demonstrating its ability to improve the performance of weak classifiers by concentrating on cases that were inaccurately classified. By achieving an accuracy of 98.49%, the KNN model is considered to have

mediocre performance. Similarly, the accuracies of multinomial Naive Bayes and Gaussian models are 55.72% and 55.14%, respectively.

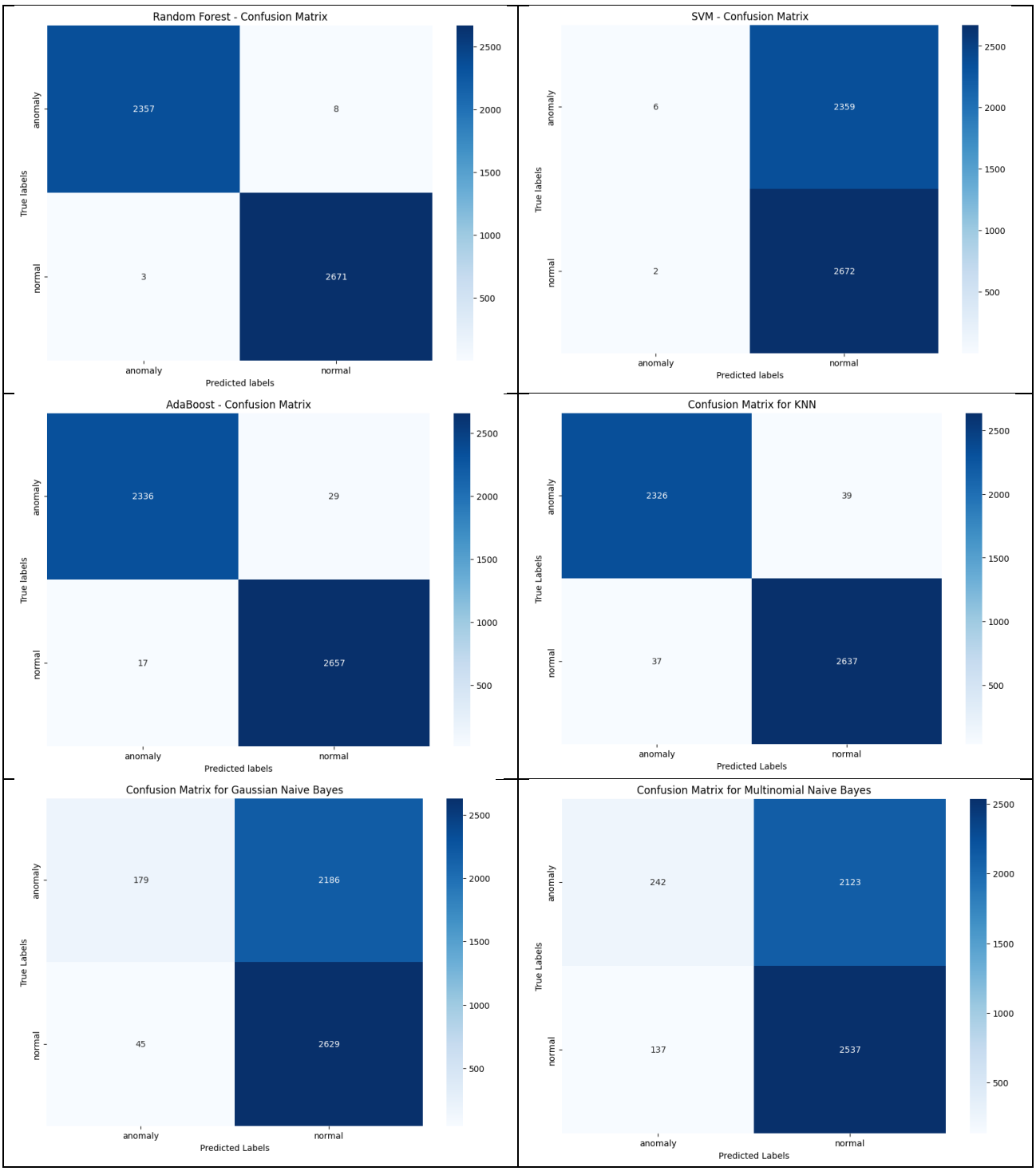

Fig. 6. Confusion of machine learning models

Figure 7 shows the accuracy distribution. Resultantly, it is shown that RF and AdaBoost are emerging as the best choices because of their better intrusion detection rate.

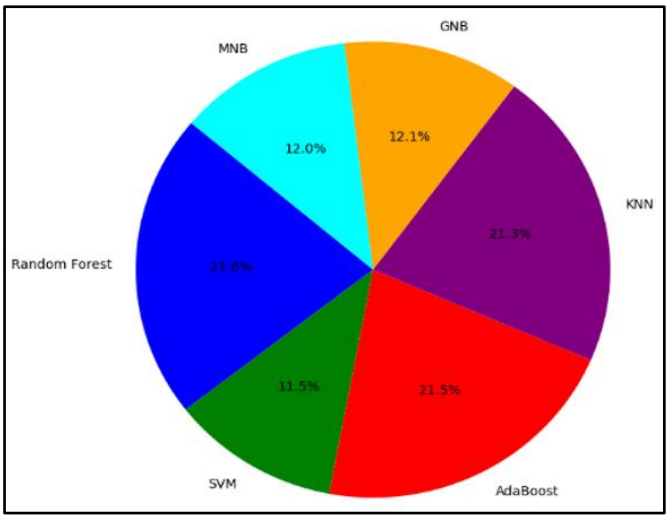

Fig. 7. Accuracy distribution

5. Conclusion

In this research, an intrusion detection method is proposed using ML models to make the network strong and reliable. The detection was done by considering 41 features such as duration, flag, wrong fragment, logged_in, etc. These features were useful to recognize whether the behavior was normal or anomalous. Through rigorous preprocessing and model training, we examined the effectiveness of RF, SVM, AdaBoost, KNN, Gaussian Naive Bayes, and Multinomial Naive Bayes by using the NIDS dataset.

There was a vast difference in the performance of models that were compared. The results of the experiments showed that RF reached a maximum accuracy of 99.78%. It can be stated that it was the best model for intrusion detection in our case. SVM had a far lower accuracy of 53.15%. This showed that the SVM algorithm struggled to cope with the challenges of the network data. Algorithms such as KNN, Gaussian Naive Bayes, and Multinomial Naive Bayes performed poorly to the level of ideal accuracy. It implies that the above models are not suitable for the NIDS task.

In the future, this work will be extended to target more features and also the use of hybrid technologies to make the method more robust and flexible in terms of selection parameters.

Conflicts of Interest

The authors declare no potential conflicts of interest in relation to this study.

References

[1] World Economic Forum. (2024). Global Cybersecurity Outlook 2024. Available at: https://www.weforum.org.

[2] National University. (2024). 101 Cybersecurity Statistics and Trends for 2024. Available at: https://www.nu.edu.

[3] Barreñada, L., Dhiman, P., Timmerman, D., Boulesteix, A. L., & Van Calster, B. (2024). Understanding overfitting in random forest for probability estimation: a visualization and simulation study. *Diagnostic and Prognostic Research, 8*(1), 14. https://doi.org/10.1186/s41512-024-00177-1.

[4] Rustam, F., & Jurcut, A. D. (2024). Malicious traffic detection in multi-environment networks using novel S-DATE and PSO-D-SEM approaches. *Computers & Security, 136*, 103564. https://doi.org/10.1016/j.cose.2023.103564.

[5] Susilo, B., & Sari, R. F. (2020). Intrusion detection in IoT networks using deep learning algorithm. *Information, 11*, 279. https://doi.org/10.3390/info11050279.

[6] Asharf, J., Moustafa, N., Khurshid, H., Debie, E., Haider, W., & Wahab, A. (2020). A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions. *Electronics, 9*, 1177. https://doi.org/10.3390/electronics9071177.

[7] Liu, L., Wang, P., Lin, J., & Liu, L. (2020). Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access, 9*, 7550-7563. https://doi.org/10.1109/ACCESS.2020.3048198.

[8] Sarker, H., Abushark, Y. B., Alsolami, F., & Khan, A. I. (2020). Intrudtree: A machine learning based cybersecurity intrusion detection model. *Symmetry, 12*, 754. https://doi.org/10.3390/sym12050754.

[9] Dong, R. H., Li, X. Y., Zhang, Q. Y., & Yuan, H. (2020). Network intrusion detection model based on multivariate correlation analysis-long short-time memory network. *IET Information Security, 14*(2), 166-174. https://doi.org/10.1049/iet-ifs.2019.0294.

[10] Injadat, M., Moubayed, A., Nassif, A. B., & Shami, A. (2020). Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Transactions on Network and Service Management, 18*(2), 1803-1816. https://doi.org/10.1109/TNSM.2020.3014929.

[11] Zhang, H., Li, Y., Lv, Z., Sangaiah, A. K., & Huang, T. (2020). A real-time and ubiquitous network attack detection based on deep belief network and support vector machine. *IEEE/CAA Journal of Automatica Sinica, 7*(3), 790-799. https://doi.org/10.1109/JAS.2020.1003099.

[12] Özalp, A. N., & Albayrak, Z. (2022). Detecting cyber attacks with high-frequency features using machine learning algorithms. *Acta Polytechnica Hungarica, 19*(7), 213-233.

[13] Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., Alharbi, F., & Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications, 72*, 103405. https://doi.org/10.1016/j.jisa.2022.103405.

[14] Chen, Z., Simsek, M., Kantarci, B., & Djukic, P. (2021). All predict wisest decides: A novel ensemble method to detect intrusive traffic in IoT networks. In *2021 IEEE Global Communications Conference*, GLOBECOM (pp. 01–06). IEEE. https://doi.org/10.1109/GLOBECOM46510.2021.9685318.

[15] Friha, O., Ferrag, M. A., Benbouzid, M., Berghout, T., Kantarci, B., & Choo, K.-K. R. (2023). 2DF IDS: Decentralized and differentially private federated learning-based intrusion detection system for industrial IoT. *Computers & Security, 127*, 103097. https://doi.org/10.1016/j.cose.2023.103097.

[16] Kaggle. Network Intrusion Detection. Available at: https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection/.

[17] Chandrashekar, G., & Sahin, F. (2020). A survey on feature selection methods. *Computers & Electrical Engineering, 40*(1), 16-28. https://doi.org/10.1016/j.compeleceng.2013.11.024.

[18] Hsu, C. W., & Lin, C. J. (2019). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks and Learning Systems, 30*(7), 1945-1958. https://doi.org/10.1109/72.991427.

[19] Scikit-learn. AdaBoostClassifier. Available at: https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[20] Sun, B., & Chen, H. (2021). A survey of k nearest neighbor algorithms for solving the class imbalanced problem. *Wireless Communications and Mobile Computing, 2021*, 5520990. https://doi.org/10.1155/2021/5520990.

[21] Rish, I. (2001). An empirical study of the Naive Bayes classifier. In *Handbook of Statistical Analysis and Data Mining Applications* (pp. 305-315). Academic Press.

[22] Abbas, M., Memon, K. A., Jamali, A. A., Memon, S., & Ahmed, A. (2019). Multinomial Naive Bayes classification model for sentiment analysis. *IJCSNS International Journal of Computer Science and Network Security, 19*(3), 62-67.