Array 19 (2023) 100306

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

Ensuring network security with a robust intrusion detection system using ensemble-based machine learning☆

Md. Alamgir Hossain *, Md. Saiful Islam

Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), Palashi, Dhaka, 1205, Bangladesh

ARTICLE INFO

Keywords:
Intrusion detection system
Feature extraction for IDS
Ensemble-based approach
Machine learning for IDS
Computer network security
Cyber attacks detection
Etc

ABSTRACT

Intrusion detection is a critical aspect of network security to protect computer systems from unauthorized access and attacks. The capacity of traditional intrusion detection systems (IDS) to identify unknown sophisticated threats is constrained by their reliance on signature-based detection. Approaches based on machine learning have shown promising results in identifying unknown malicious attacks. No learning algorithm-based model, however, is able to accurately and consistently detect all different kinds of attacks. Besides that, the existing models are tested for a specific dataset. In this research, a novel ensemble-based machine-learning technique for intrusion detection is presented. Numerous public datasets and multiple ensemble strategies, including Random Forest, Gradient Boosting, Adaboost, Gradient XGBoost, Bagging, and Simple Stacking, will be employed to evaluate the performance of the proposed approach. The most relevant features for the detection of intrusion are selected using correlation analysis, mutual information, and principal component analysis. Our research using different ensemble methods demonstrates that the proposed approach using the Random Forest technique outperforms existing approaches in terms of accuracy and FPR, typically exceeding 99% with better evaluation metrics like Precision, Recall, F1-score, Balanced Accuracy, Cohen's Kappa, etc. This strategy may be a useful tool for strengthening the safety of computer systems and networks against emerging cyber threats.

1. Introduction

Network intrusion refers to unauthorized access or malicious activity on a computer network or system by an external attacker or an insider. A network intrusion occurs when an intruder attempts to obtain unauthorized access to system or network resources, steal sensitive information, interfere with network functionality, or install malware or backdoors for future attacks. Denial-of-service (DoS) attacks, brute-force attacks, port scanning, malware infection, distributed denial-of-service (DDoS) attacks, phishing, botnet attacks, social engineering, etc. are just a few examples of the many ways that networks may be infiltrated. Network intrusions can seriously harm businesses, including by resulting in monetary damages, reputational harm, legal liabilities, and the loss of confidential information [1–4].

Due to the rise in online threats and attacks, it is crucial to develop an intrusion detection system, also known as an IDS, for protecting computer networks and systems [5]. Cybercrime has become a significant concern for individuals, businesses, and governments worldwide. Hackers and cybercriminals are continually creating novel and sophisticated attack techniques to breach computer networks, steal confidential data, and interfere with enterprises' regular business activities [6,7]. Monitoring network traffic, identifying and warning administrators about possible security risks, and assisting in the prevention of security breaches are all functions of an IDS, a crucial security component. Organizations may be at risk from cyberattacks without an IDS, which might have serious consequences including financial losses, legal liability, the loss of sensitive data, and damage to reputation. IDSs provide several benefits to organizations and individuals in the cyber world, including proactive monitoring of network traffic, real-time detection of potential security threats, and quick response to potential security breaches. IDSs can also improve the accuracy and effectiveness of incident response by providing detailed information about security threats and attack patterns. Moreover, IDSs can help organizations comply with legal and regulatory requirements for information security. Many

☆ Dr. Md. Saiful Islam provided valuable insights and ideas, supervised the research process, and checked the implemented results for accuracy. He actively participated in the revision and improvement of the research paper, providing important feedback and suggestions to enhance its quality. His guidance and expertise were instrumental in ensuring the quality and rigor of the research.

* Corresponding author.
E-mail addresses: alamgir.cse14.just@gmail.com (Md.A. Hossain), mdsaifulislam@iict.buet.ac.bd (Md.S. Islam).

https://doi.org/10.1016/j.array.2023.100306
Received 9 May 2023; Received in revised form 24 June 2023; Accepted 24 June 2023
Available online 1 July 2023

industries, such as finance, healthcare, and government, have strict regulations and compliance requirements for protecting sensitive data. An IDS can help organizations meet these requirements by providing a means to monitor and detect potential security threats [8,9].

Developing a stable intrusion detection system (IDS) in digital cyberspace is challenging due to several factors. The ever-evolving nature of cyber threats and the complexity of modern computer networks make it difficult to create a system that can detect all types of attacks accurately and efficiently [10]. Additionally, the vast amount of network traffic generated in today's digital world makes it challenging to identify abnormal patterns and activities that could be indicative of an attack [11]. False positives and false negatives rates are key challenges in developing stable IDS. False positives happen when an IDS warns administrators of a potential attack but it turns out to be a false alarm, while false negatives happen when an IDS misses an actual attack. Both of these scenarios can have serious consequences, as false positives can lead to unnecessary system downtime and increased workload for security personnel, while false negatives can result in a security breach that can have devastating effects on an organization [12]. Furthermore, the development of stable IDS requires expertise in both cybersecurity and machine learning. It is challenging for security experts to keep current with the most recent attack methodologies and plans due to the complexity of contemporary computer networks and the constantly developing nature of cyber threats. Similarly, machine learning algorithms and techniques are constantly evolving, and developing stable IDS requires a deep understanding of these techniques and how to apply them effectively relating to network protection [13].

A variety of machine learning-based intrusion detection methods has been presented to researchers in recent years utilizing a variety of publicly accessible datasets. While some machine learning models might perform well on a particular dataset, they might have difficulty generalizing to new data from different sources [14]. For example, Jain et al. [15] designed a framework for hybrid intelligent security monitoring with CNN and LSTM methods. The accuracy for binary classification of their approach with the NSL-Botnet dataset is 92%, when they tested their model with another dataset UNSW-NB15, the accuracy was changed and that is 82%. In intrusion detection, where attack patterns are continually changing and the current models are not entirely reliable, this is an ongoing problem. Machine learning models might generate a lot of false alarms, which could result in a lot of false positives and unnecessary warnings [16]. The employment of some machine learning models in real-time detection systems for intrusions may also be constrained by their high computing resource requirements [17]. In this research, we provide a reliable technique for intrusion detection utilizing ensemble-based machine learning to address these issues. Below is a summary of our contributions to this research:

- Examine the most recent intrusion detection techniques and identify their benefits as well as their drawbacks.
- Evaluate the effectiveness of different ensemble-based machine-learning algorithms for detecting unknown intrusions/attacks
- Selecting the relevant features for detecting intrusions using correlation analysis, mutual information, and principal component analysis
- Developing a stable framework for combining multiple learning classifiers with the ensemble-based machine-learning algorithms
- Using more than 10 public datasets to compare the effectiveness of the proposed framework with current intrusion detection techniques

Our proposed approach has the potential to strengthen the safety of computer systems and networks against emerging cyber threats by providing a stable and effective method for intrusion detection. We implemented the model in Google Colaboratory with Python programming language. We checked the model with various ensemble methods after preprocessing and feature selection like Random Forest, Bagging, Adaboost, Gradient Boosting, Gradient XGBoost, and Simple Stacking. From the all ensemble methods, our proposed approach with the Random Forest ensemble method provides the best results for all the evaluation metrics for various public datasets like WSN-DS, UNSW-NB, UNR-IDD, UKM-IDS, SIMARGL, NSL-KDD, NF-UQ-NIDS, NF-ToN-IoT, KDDCUP, CICIDS, Cyber Clean Center (CCC), etc. The model accuracy is more than 99% for all the datasets. This proposed model addresses the limitations of existing intrusion detection systems and provides a framework tested with multiple datasets, making it more reliable and accurate in detecting intrusions.

A full summary of the relevant work in the field will be provided in the literature review that introduces the next portion of this paper. After the proposed architecture, the experimental findings demonstrating the effectiveness of the proposed strategy will be presented. In the conclusion, the article's main results will be summarized along with suggestions for further research in this area.

2. Literature review

This section presents a quick summary of earlier research on machine learning and detection systems for intrusions, emphasizing their advantages and disadvantages. This sets the context for the proposed ensemble-based machine learning approach, which is the focus of the research paper. Researchers make numerous machine learning-based model approaches at different times over the years. To better comprehend the significance of our proposed ensemble-based approach to detect intrusion and ensure network security, we discussed fairly current models with their strengths and drawbacks.

Papamartzivanos et al. developed a method for the detection of network intrusions in 2018. They introduced Dendron, a novel approach for creating Decision Tree (DT) classifiers using Genetic Algorithm (GA), in order to give detection criteria in the context of abuse systems for detection. They employed three distinct publicly available datasets to assess the effectiveness of their model. They consider the output label in every case as multiclass detection. The KDDCup'99 model has a 99% accuracy rate and an average accuracy rate of 89%. The identification accuracy was 97% and on average 90% when they tested their model against another public dataset called NSL-KDD. The NSL-KDD dataset is nearly identical to the KDDCup'99 dataset. Their model's accuracy was 84% when compared to another publicly available dataset, UNSW-NB15, whereas the average accuracy was only 52% [18]. As a result, the detection efficiency of various datasets varies significantly. From here, it is obvious that this kind of model can only be used with particular kinds of datasets or cyberattacks. When a new intrusion occurs, it will be unable to identify attacks.

In order to identify intrusion, Halimaa et al. [19] suggested a ML-based model using the Support Vector Machine (SVM) approach in 2019. The model's performance was evaluated using the NSL-KDD knowledge discovery dataset, and they were able to reach an accuracy of 93.95%, which is inadequate in the present networking context. As a result of the numerous types of attacks carried out by attackers every day, they advised creating a more effective model based on organized classifiers that is capable of classifying new attacks with better performance. This is because their single classifier-based model is unable to detect all of the attacks. For the same dataset and the detection of intrusion in the same year Yang et al. [20] proposed a deep belief network (DBN) based SVM model. They are about 97% accurate in their detection of intrusion. Their model showed good detection performance, but when the sample size of the network intrusion type was tiny, it was not substantially more efficient.

In 2020, a model for network intrusion detection systems based on convolution neural networks (CNN) and bi-directional long short-term memories (BiLSTM) is provided in the research [21]. There are two datasets used to test this model. The model using NSL-KDD produced results of 83.58% accuracy and 84.49% recall. However, the model's performance using the UNSW-NB15 dataset provides values of 77.16% accuracy and 79.91% recall. When the dataset is altered, the results

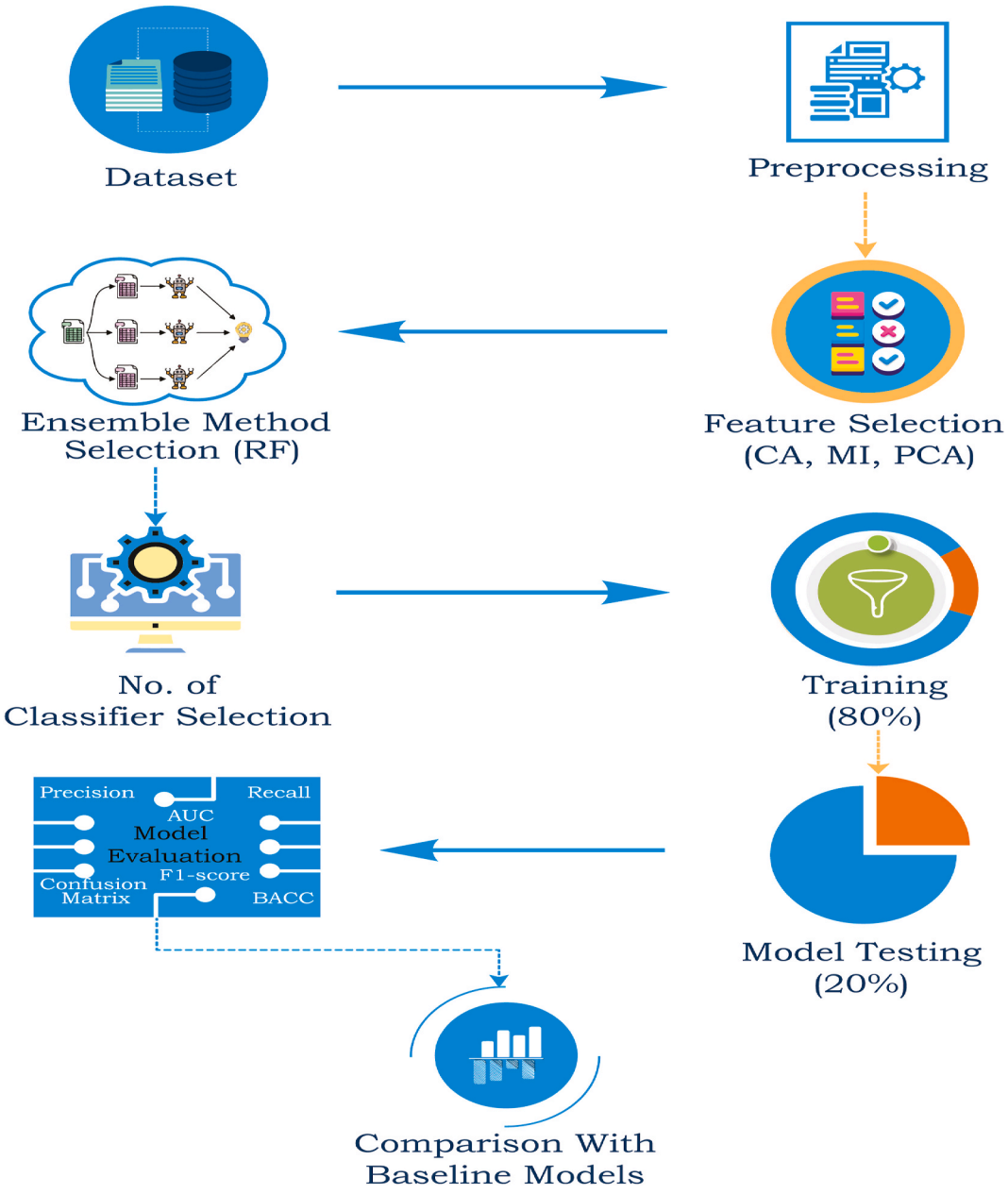

Fig. 1. Proposed model for intrusion detection system (IDS).

significantly differ. A different dataset indicates a different attack or intrusion pattern. In both situations, detection is not completely effective. For network intrusion detection, another model employed by Elmasrya et al. [22] is a dual Particle Swarm Optimization (PSO)-based approach to choose feature subsets and hyperparameters simultaneously. It is essential to consider that the proposed approach is only tested on two popular IDS datasets, and it could not be adapted well to other datasets. Using deep learning models and the double PSO-based method in the suggested technique may also necessitate utilizing substantial computer resources. Using the CNN technique, Turaiki et al. [23] designed Anomaly-Based Network Intrusion Detection and experimented with it on the NSL-KDD dataset. This model's 83% true positive rate is extremely low. Using the NSL-KDD and CICIDS2017 datasets, the model is evaluated. Even though both datasets have accuracy levels above 96%, the error rate is higher and the true positive rate is lower. Prasada et al. [24] proposed a novel Bayesian-Rough Set (BRS) feature selection-based model for intrusion detection with the CICIDS2017 dataset and attain an accuracy of 97%. But in respect of this dataset, the FPR is very high for this model. Using the same dataset, Panigrahia's et al. [25] suggested DTNB and MOEFS-based machine learning strategy achieved 96.80% accuracy and 97.40% precision. Just five features were employed in this research and they failed to identify minor class attacks.

For the IDS, Salam et al. [26] provides an adaptive method and a new dataset in 2021. The accuracy with the UKM-IDS20 dataset is 93%. UNSW-NB15 is 89%, whereas KDD-CUP'99 is 96%. The connection records that were employed to build the UKMIDS20 dataset in this model were manually labeled. Their proposed adaptive IDS establishes and analyses connection data on a 24-h basis at periodic intervals. For future investigation, they recommended various time frames and their impact on identifying malicious attempts. An ensemble-based approach for detecting network anomalies was proposed by Liao et al. The model's recall score, precision, and f1-score for the UNSW-NB15 dataset are 0.924, 0.979, and 0.949 respectively. The evaluation metrics values are dropping when the model is trained and evaluated on the CICIDS2017 [27] dataset, with accuracy being 0.838, recall being 0.840, and f1-score

being 0.835. The evaluation measure value differences in this model are likewise fairly large [28].

By combining the ARF and HAT approaches, Tabbaa and Ifzarne produced an online ensemble learning model in 2022 that successfully detected attacks in wireless sensor networks with an accuracy of 99.42%. Their model's precision, which is 96.84%, has to be improved [29]. Tahri et at [30]. also suggest an SVM-based model for the detection of intrusions. The accuracy with the UNSW-NB15 dataset is 97.77%. This single classifier-based model is not able to detect new intrusions. Ahmed et al. [31] suggested an approach for the identification of intrusions that is based on machine learning ensembles. For the UNSW-NB15 dataset, this model has a precision of 94.80% and an accuracy of 95.10%. To solve the issue of class imbalance, the Synthetic Minority Oversampling Method (SMOTE) is used. The appropriate feature is then chosen using Principal Component Analysis (PCA), and the prediction is made using the Random Forest (RF) machine learning technique. Yet, the precision would have to be higher. Moreover, the SMOTE might occasionally result in overfitting and produce noisy data. The Nearest Neighbor-based machine learning model proposed by Andrecut M [32]. is another single classifier. The accuracy of the model, which was trained and evaluated using the CSE–CIC–IDS2018 dataset, is 98.58%. To improve this model's accuracy, data must be normalized.

Wang et al. [33] proposed an intrusion detection system based on a Gaussian mixture model and a one-class SVM in 2023. In the present research, the one-class support vector machine (OCSVM) and the Gaussian mixture model (GMM), two semi-supervised detectors, are trained on the generated features after the autoencoder (AE) extracts representative features from normal data. Their accuracy on the IDS2018 dataset is 95.10%, and their FPR is a very high 5.772%. The higher PFR is one of the main limitations of this research. In order to identify botnet-based intrusion, Srinivasan et al. [34] uses stacking ensemble classification-based machine learning, which is another advancement in the provision of cyberspace security. The accuracy of their approach is 94%. Yet, the TPR is lower and the FPR is higher. On the other hand, the applied stacking ensemble technique's training and testing processes take a lot of time. Because the stacking ensemble uses a meta-classifier and when the training dataset is larger, it takes a huge amount of time. For IoT intrusion detection, Jemili et al. [35] suggested and evaluated a different model combination of Random Forest and XGBoost, which provides a total accuracy equivalent to 97% with the *N*-BaIoT [36] dataset.

The current intrusion detection model demonstrates that not all forms of intrusion detection are compatible with single-classifier-based machine learning models with the existing feature selection method. Since not all datasets or newly patterned intrusions can be tackled by all models, some offer much better solutions. So, it is required to develop an intrusion detection model that would work for all forms of intrusions, all publicly available datasets, and new intrusions. We introduce novel feature selection techniques in our proposed methodology. Then, we build the model using the random forest ensemble classifier. To increase the models accuracy and stability, the random forest ensemble-based machine learning technique combines many decision trees. The random forest method may more accurately detect a variety of intrusion types while decreasing the percentage of false positives and false negatives. The proposed method of applying a random forest ensemble-based machine learning methodology for intrusion detection may be able to offer a more complete and effective solution for ensuring network security against all varieties of cyberattacks or intrusions.

3. Proposed Approach Developing

This section provides a detailed description of the proposed model for detecting intrusions. Fig. 1 displays the architecture of the machine learning-based model development pipeline including the features selection.

3.1. Dataset description

Datasets are essential for machine learning-based models because they provide the data that the model needs to learn and make accurate predictions or decisions. Besides that, a high-quality dataset with accurate and relevant examples will lead to a more accurate and reliable model. The suggested model is trained and evaluated using a variety of publicly available datasets. The following is a short summary of these datasets:

3.1.1. UNR-IDD

The UNR-IDD, a NIDS dataset produced by Das et al. contains the majority of the network port data. In the proposed model, we used this dataset as a type of binary classification dataset. Binary categorization is used to distinguish between intrusions and normal working scenarios. This dataset has 34 features in total. In the Label feature, the terms "normal data" and "attack" denote different types of network functionality. Under the heading "attack," many forms of attacks are mentioned, including TCP-SYN, PortScan, Overflow, Blackhole, Diversion, etc [37].

3.1.2. SIMARGL2021

During a series of attacks in 2021, Mihailescu et al. [38] created the dataset from real traffic. This dataset has 50 features in total. More than 1.33 million samples are used to train and evaluate the model in this research. The "LABEL" column offers three different label types: normal flow, XMAS Scan, and NULL Scan. XMAS Scan and NULL Scan are taken into account as intrusions, whereas Normal flow is taken into account for typical network flow. This dataset serves as the basis for the multi-class classification issue in this research.

3.1.3. NF-UQ-NIDS

In 2021, Sarhan et al. [39] analyzed information and created the NF-UQ-NIDS dataset for network intrusion detection. The datset has 43 features. 11,994,893 entries comprise the NF-UQ-NIDS dataset, of which 2,786,845 (23.23%) are attacks and 9,208,048 (76.77%) are benign flows. In addition to the benign, normal flow Attacks are classified into the following categories: DoS, DDoS, Injection, Reconnaissance, Brute Force, Infiltration, Password, XSS, Ransomware, Exploits, Scanning, Backdoor, Fuzzers, Bot, Analysis, Theft, MITM, Shellcode, Worms, and Generic. In this research, the attacks category is regarded as an intrusion.

3.1.4. NF-ToN-IoT

As a NetFlow-based Internet of Things dataset for IoT intrusion detection, Sarhan developed NF-ToN-IoT in 2021. There are 1,379,274 data flows in total, of which 1,108,995 (80.4%) are attack samples and 270,279 (7.9%) are benign data flows [39]. In addition to the regular flow, intrusions include Backdoor, Injection DoS, MITM, DDoS, Password, Scanning, Ransomware, and XSS.

3.1.5. UKM-IDS20

In 2021, Al-Daweri et al. [26] presented a model dataset for an intrusion detection system called UKM-IDS20. This dataset has 46 characteristics that cover four different forms of attacks: DoS, scans, ARP poisoning, and exploits. These four attacks are regarded as intrusions into our analysis.

3.1.6. CSE–CIC–IDS2018

The CSE–CIC–IDS2018 dataset was developed in 2018 as a result of a collaboration between The Communications Security Establishment and The Canadian Institute for Cybersecurity. It uses the notion of profiles to methodically compile cybersecurity datasets. When coupled with abstract distribution models for apps, standards, or entry-level network components, it provides a thorough explanation of attacks. Seven different attack scenarios are included in the dataset: botnet, brute force,

heartbleed, DoS, DDoS, network intrusion, and web attacks [40]. 80 characteristics and 1048575 data were used in our research to test the proposed model. Attacks are all regarded as intrusions.

3.1.7. WSN-DS

The WSN-DS dataset, a particular dataset for detecting intrusions in wireless networks of sensors, was developed in 2016 by Almomani et al. [41]. This dataset has 17 characteristics and 374661 records, covering DoS attacks including Flooding, Blackhole, Scheduling, and, Grayhole. This dataset was used to test our model, which treats all DoS attacks as intrusions while treating all other attacks as part of the regular flow.

3.1.8. UNSW-NB15

The NSW-NB 15 dataset was produced in 2015 using the IXIA PerfectStorm tool at the Cyber Range Lab at UNSW Canberra to develop a blend of actual modern day routine activities and fabricated modern day assault behaviors [42]. Reconnaissance, Fuzzers, Backdoors, Analysis, DoS, Generic, Shellcode, Exploits, and Worms are among the nine types of attacks in this dataset. In addition to the regular traffic flow, all attacks are considered as intrusions. This dataset is now widely used for testing intrusion detection models. In our experiment, 45 features from this dataset were employed.

3.1.9. CCC

We use the C08, C09, C10, and C13 datasets from the publicly available Cyber Clean Center (CCC) dataset [34,43]. IRC port 6667 and HTTP port 80 traffic packets are included in this dataset. A C&C server connection is necessary for the bot. In this dataset, there are 56 features available. In this research, the term "attack" is used to refer to an intrusion, whereas the term "normal" is used to refer to regular traffic flow.

3.1.10. NSL-KDD

The intrusion detection domain KDD-CUP99 [44] data set is the foundation for the NSL-KDD [45] data collection. The standard KDD-CUP99 data set's issues with redundant features and duplicate records are resolved. It has been widely used in the field of network detection for intrusions. The NSL-KDD data set categorizes each network connection as normal or abnormal. In this research, 43 characteristics were both available and utilized. We employed this dataset for multi-class classification in our research. Where back, buffer overflow, imap, ftp write, guess passwd, ipsweep, land, loadmodule, multihop, nmap, perl, phf, pod, portsweep, rootkit, satan, warezclient, smurf, spy, teardrop, and warezmaster are classified into several intrusion classes in addition to the non-attack class.

3.2. Experimental setup

Python and the Scikit-learn package for Python are employed throughout the entire experiment for this research. The experiment was carried out using Google Colaboratory, sometimes referred to as "Colab," a program created by Google Research. With Colab, anybody may develop and execute arbitrary Python code, which is particularly suited to machine learning, data analysis, and instructional purposes. Colab is a hosted Jupyter Notebook service, to be more specific. StandardScaler, LabelEncoder, and other preprocessing modules from the Scikit-Learn package are implemented. SelectKBest, mutual_info_classif, PCA, and other methods are used to choose the features. BaggingClassifier, AdaBoostClassifier, RandomForestClassifier, GradientBoostingClassifier, XGBClassifier, StackingClassifier, and more classifiers are utilized from the ensemble module. Additionally, the Scikit-learn (python library) infrastructure's ROC curve, ROC auc score, and Cohen kappa score as well as confusion matrix, accuracy score, precision, FPR, recall score, BCC, and f1 score are utilized to evaluate the model [46].

3.3. Data preprocessing

In order for machine learning models to utilize the data effectively, it must first be cleaned up and transformed, which is why data preliminary processing is a crucial stage in the entire procedure. The accuracy and effectiveness of the final model depend greatly on the quality of the input data, and data preprocessing makes it possible to guarantee the consistency, accuracy, and usefulness of the input data. It helps to remove inconsistencies and errors in the data, it can help to normalize the data, which can make it easier to compare and analyze, and it can help to make the data more manageable and useable for machine learning models [47].

In this model, the preprocessing steps include removing duplicates, replacing infinite and large values with NaNs, dropping rows containing NaNs, separating numerical and categorical columns, normalizing numerical columns, encoding categorical columns, and converting the target variable into a discrete variable. First, we check for duplicate rows in the dataframe using the "duplicated" function from pandas. If any duplicates are found, the function returns True for those rows. The "drop_duplicates" function of the panda's library is then used for eliminating the duplicate rows from the dataframe.

Next, we replaced infinite and large values with NaNs using the "replace" function from pandas and the numpy library. It also replaced values that do not match a specified pattern with NaNs using the "replace" function.

Then, we dropped any rows that contain NaNs using the "dropna" function from pandas. The remaining dataframe is separated into numerical and categorical columns. Numerical columns are those with dtype "float64" or "int64", and categorical columns are those with dtype "object".

After that, the code normalizes the numerical columns using the "StandardScaler" function from the sklearn library. This ensures that each feature has zero mean and unit variance. The categorical columns are encoded using the "LabelEncoder" function from the sklearn library. This converts categorical variables into numerical variables, with each unique value assigned a unique integer value.

Finally, we separated the dataset into features (X) and labels (y), where the "Label" column is used as the target variable. The target variable y is then converted into a discrete variable using the "cut" function from pandas. The target variable is divided into 10 equal bins, and each bin is assigned a unique integer value. The resulting y variable contains the integer values of the bins rather than the bin labels.

3.4. Feature selection

In the process of creating a machine-learning model to identify network intrusion, feature selection is an essential step. This is because not all features in the dataset may be relevant or contribute equally to the prediction of network intrusion [48]. In fact, overfitting, when the model is overly complicated and performs badly on unobserved data, can result from integrating unnecessary or duplicate characteristics. In this proposed method three different methods Correlation analysis (CA), Mutual information (MI), and Principal component analysis (PCA) are employed. A short description of these three methods is given below:

3.4.1. Correlation analysis (CA)

Correlation analysis is important for feature selection in ensemble methods because it helps identify highly correlated features that may cause overfitting or redundancies in the model. Ensemble methods combine multiple models to improve predictive accuracy, and using highly correlated features can result in multiple models making similar predictions, reducing the diversity of the ensemble. Correlation analysis may be used to find and eliminate highly correlated features, which will enhance the performance and stability of the ensemble approach [49]. In the implementation of our model, correlation analysis is used to select relevant features for prediction. We calculate the correlation matrix

"corr" of the features in X, and then select the features that have an absolute correlation coefficient greater than 0.5.

The correlation coefficient's definition is the covariance between two variables, X and Y, divided by the sum of their standard deviations. This can be expressed mathematically as the form of Equation (1).

$$\mathrm{corr}(X, Y) = \mathrm{cov}(X, Y) \,/\, (\mathrm{std}(X) * \mathrm{std}(Y)) \quad (1)$$

where std(X) is the standard deviation of X and std(Y) is the standard deviation of Y, and cov(X, Y) is the covariance between X and Y. In our code, the "corr_abs" variable is created by taking the absolute value of the correlation matrix "corr". This is because the absolute value of the correlation coefficient represents the strength of the linear relationship between two variables, regardless of whether the relationship is positive or negative. The "relevant_features_corr" variable in Equation (2) is then created by selecting the index of the columns (i.e., features) that have an absolute correlation coefficient greater than 0.5. This is done by using the ".index.tolist()" method on the subset of the correlation matrix that meets the threshold:

$$\mathrm{relevant\_features\_corr} = \mathrm{corr\_abs}[\mathrm{corr\_abs} > 0.5].\mathrm{index.tolist}() \quad (2)$$

The resulting relevant_features_corr list contains the names of the features that are highly correlated with one another, and thus may be redundant. These features can then be removed or combined to reduce the dimensionality of the dataset, which can help to improve the performance of our model.

3.4.2. Mutual information (MI)

A measure of the reliance between two random variables is mutual information. It evaluates how much information a feature contributes to the target variable in the context of feature selection. In our implementation, it is used to select the top k features that are most informative about the target variable y. The mutual information between a feature X_i and the target variable y is defined in the form of Equation (3).

$$I(X\_i, y) = H(X\_i) - H(X\_i|y) \quad (3)$$

where H(X_i) is the entropy of feature X_i, and H(X_i|y) is the conditional entropy of feature X_i given the target variable y.

The level of uncertainty or randomness in a random variable is measured by entropy. It is calculated using the following Equation (4):

$$H(X) = -\mathrm{sum}(p(x) * \log 2(p(x))) \quad (4)$$

where p(x) is the probability of observing the value x in the random variable X.

Given the value of another random variable Y, conditional entropy calculates the degree of uncertainty in a random variable X. It is calculated using the following Equation (5):

$$H(X|Y) = -\mathrm{sum}(p(x, y) * \log 2(p(x|y))) \quad (5)$$

where p(x,y) is the joint probability of observing the values x and y in the random variables X and Y, and p(x|y) is the conditional probability of observing the value x in X given the value y in Y [50].

The "mutual_info_classif" function from the "sklearn.feature_selection" module is used to calculate the mutual information between each feature and the target variable y. The "SelectKBest" function from the same module is then used to select the top k features with the highest mutual information scores. We selected the top 20 features with the highest mutual information scores by specifying the values of k as 20.

3.4.3. Principal component analysis (PCA)

By determining the most significant features that capture the greatest amount of variance in the data, Principal Component Analysis (PCA) is a crucial approach for feature selection since it enables us to lower the dimensionality of the input data [51]. The underlying equation for PCA is as follows:

Given a data matrix X with n samples and m features, PCA aims to find a set of k orthogonal vectors u1, u2, …, uk in the m-dimensional space, such that the projected data Y onto the subspace spanned by these vectors maximizes the variance of the data like as Equation (6):

$$Y = XU\_k \quad (6)$$

where U_k is a matrix of the top k eigenvectors of the covariance matrix of X. The principal components are the columns of the Y matrix, which represent the new features obtained by projecting the original data onto the subspace spanned by the eigenvectors. In our implementation, the "PCA" class from the "sklearn.decomposition" module is used to perform the PCA analysis on the input data X. The "pca.components_" attribute of the "PCA" object returns the matrix U_k, and the "argmax()" method is used to obtain the indices of the features that have the highest absolute values in each component. These indices are then mapped back to the original feature names using the "X.columns" attribute of the input data to obtain the most important features.

Finally, we combine the relevant features selected from three different methods - correlation analysis, mutual information, and principal component analysis (PCA) - into a single list.

The "set().union()" function is used to merge the three lists of relevant features while removing duplicates. The resulting set is then converted to a list using the "list()" function to create a final list of relevant features.

By combining relevant features from multiple feature selection methods, we can create a more robust and accurate set of features that capture the most important information in the data. As a result, the model's performance may be enhanced and new knowledge about the underlying connections between the features and the target variable may be obtained.

3.5. Ensemble method selection

Ensemble-based machine learning models are a promising approach for intrusion detection systems, offering improved detection rates and resilience to attacks. These models combine multiple individual models to improve the overall accuracy and robustness of the prediction, especially for the intrusion detection system (IDS). The diversity of individual classifiers in the ensemble ensures that the IDS can detect a wide range of attack types and patterns [52]. Various ensemble techniques are applied in the proposed model. A very short description of those is given below:

3.5.1. Random forest

A final classification determination is made by combining many decision trees using the ensemble-based machine learning method known as random forest. To lessen overfitting and boost generalization performance, the random forest constructs each decision tree individually using a randomly picked subset of the training data and characteristics. A majority vote of the various decision trees determines the random forest algorithm's output. The projected class of the input data point is voted on by each decision tree in the forest, and the class with the highest votes is the final prediction [53]. Working process in a dataset using Random Forest algorithm is given in Algorithm 1.

Algorithm 1. Working process Random Forest classifier in the Detection of Intrusion

i. Initialize a set of decision trees T
ii. For each decision tree t in T:
   a. Randomly select a subset of the training data D′ from the full training data D
   b. Randomly select a subset of the features F′ from the full feature set F
   c. Construct a decision tree using D′ and F′
   d. Add the decision tree t to the set T

iii. For a given input data point x, predict the class label y as follows:
   a. Let C be the set of possible class labels
   b. For each decision tree t in T, let y_t be the class label predicted by t for input x
   c. Compute the frequency of each class label in the set {y_t} for all t in T
   d. Assign the class label y to the class with the highest frequency

3.5.2. Bagging ensemble

Ensemble-based bagging is a machine-learning algorithm that combines multiple models to improve the overall performance and robustness of the classifier. The bagging algorithm involves training multiple models on different subsets of the training data and then aggregating their predictions to make a final classification decision [54]. The training process for bagging can be summarized as follows in Algorithm 2.

Algorithm 2. Training process of bagging ensemble classifier

i. For each model k in the ensemble:
   a. Randomly select a subset of the training data D_k of size N_k (N_k $<$ N)
   b. Train the model using D_k
ii. For a given input data point x, predict the class label y as follows:
   a. Let C be the set of possible class labels
   b. For each model k in the ensemble, let y_k be the class label predicted by k for input x
   c. Compute the frequency of each class label in the set {y_k} for all k in the ensemble
   d. Assign the class label y to the class with the highest frequency

The prediction process can be summarized mathematically as follows:

Ensemble-based Bagging Classifier:

$$y = \text{majority\_vote}(y\_1, y\_2, \ldots, y\_K) \quad (7)$$

where majority_vote is a function that returns the class label with the highest frequency among the set of predictions {y_1, y_2, …, y_K}.

Different basic classifiers, including decision trees, support vector machines, and neural networks, can be utilized with bagging. The bagging ensemble's individual classifiers are trained using various feature sets and parameters on a portion of the given data. The outputs of all individual classifiers are then combined to produce a final classification decision.

3.5.3. Adaboost

Adaboost is a machine learning technique based on ensembles that combines a number of weak classifiers to produce a powerful classifier. The weak classifiers in Adaboost are decision trees with only one split, also called decision stumps. Adaboost assigns a weight to each data point in the training set and adjusts the weights after each iteration to give more weight to misclassified data points, allowing the subsequent iteration to focus on these points [55]. The training process for Adaboost can be summarized as follows in Algorithm 3.

Algorithm 3. Training process for Adaboost ensemble classifier

i. Initialize the weights of all data points in the training set to 1/N, where N is the number of data points.
ii. For each iteration t from 1 to T, where T is the number of weak classifiers to be trained:
   a. Train a weak classifier h_t on the training data with weights assigned to each data point.
   b. Compute the weighted error ε_t of the classifier h_t as follows: ε_t = $\sum$ w_i * I(y_i ≠ h_t(x_i))/$\sum$ w_i, where w_i is the weight of data point i, y_i is the true class label of data point i, and I is the indicator function.
   c. Compute the weight α_t of the classifier h_t as follows: α_t = ln((1 - ε_t)/ε_t)
   d. Update the weights of the data points as follows: w_i = w_i * exp(α_t * I(y_i ≠ h_t(x_i)))
iii. Compute the final weighted sum of the weak classifiers to make the classification decision as follows: H(x) = sign($\sum$ α_t * h_t(x)), where sign is the sign function, and H(x) is the final predicted class label.

3.5.4. Gradient boosting

An ensemble-based machine learning approach called gradient boosting combines a number of weak learners to produce a strong learner. Unlike Adaboost, Gradient Boosting uses decision trees as weak learners and focuses on improving the performance of the model by minimizing the loss function at each iteration [56]. The training process for Gradient Boosting can be summarized as follows in Algorithm 4.

Algorithm 4. Training process for Gradient Boosting ensemble classifier

i. Initialize the model by fitting a weak learner to the data.
ii. For each iteration t from 1 to T, where T is the maximum number of iterations:
   a. Compute the negative gradient of the loss function for the current model with respect to the target variable: r_{it} = -[∂L(y_i, F_{t-1}(x_i))/∂F_{t-1}(x_i)], where L is the loss function, y_i is the true target value of data point i, and F_{t-1}(x_i) is the predicted value of data point i by the current model up to iteration t-1.
   b. Fit a weak learner h_t to the negative gradient values r_{it}, to obtain a model that minimizes the residual error.
   c. Compute the weight α_t for the new model h_t using a line search algorithm, which optimizes the value of the objective function to minimize the residual error.
   d. Update the model by adding α_t * h_t to the current model, to reduce the residual error.
iii. Compute the final prediction for a new data point x by summing up the predictions of all the weak learners: F(x) = $\sum$ α_t * h_t(x), where F(x) is the predicted value of the target variable for the new data point x.

3.5.5. Gradient XGBoost

XGBoost is an optimized implementation of Gradient Boosting that uses both tree-based and linear-based models to improve the model's performance. It includes a regularization term in the loss function to prevent overfitting. The algorithm computes the negative gradient of the regularized loss function, fits a weak learner to the negative gradient values, computes the weight for the new model, and updates the model by adding the new weak learner. The final prediction is computed by summing up the predictions of all weak learners [57].

3.5.6. Simple stacking

A machine learning procedure called stacking, which uses an ensemble approach, integrates many base models by using their outputs to build a meta-model. The original dataset is used to train the base models, and the predictions from the base models are used to train the meta-model [58]. The training process for Stacking is summarized as follows in Algorithm 5.

Algorithm 5. Training process for Stacking ensemble classifier

i. Split the training data into K-folds, where K is the number of base models.
ii. For each fold k from 1 to K:
   a. Train the kth base model on K-1 folds of the training data.
   b. Make predictions on the remaining fold of the training data.
iii. Concatenate the K sets of predictions from the base models to create a new dataset.
iv. Train the meta-model on the new dataset.
v. Repeat steps ii-iv for each fold of the training data, and compute the average performance of the model.

The final prediction for a new data point x is computed by feeding the data point into each base model, obtaining K predictions, and feeding these predictions into the trained meta-model. The meta-model computes the final prediction for the new data point. The meta-model can be any type of machine learning model, such as a logistic regression, linear regression, or neural network. In this research, the linear regression is used. The equation for the meta-model is: y = f(x_1, x_2, …, x_k), where y is the predicted value of the target variable for the new data point, x_1, x_2, …, x_k are the predictions of the base models for the new data point, and f is the function learned by the meta-model.

The hyperparameters used for Random Forest, Bagging, Adaboost, Gradient Boosting, Gradient XGBoost, and Stacking at the time of implementation are summarized below in Table 1.

3.6. Splitting dataset into training and testing

The whole dataset is divided into training and testing data using the scikit-learn (sklearn) package's "train_test_split" function. While 80% is used for training and the remaining 20% of the data is utilized for testing.

3.7. Evaluation metrics

The ultimate objective of machine learning is to create a model that can correctly forecast outcomes from data that is new. Evaluation metrics help us to determine how well the model is able to achieve this goal. By evaluating a model's performance using metrics, we can identify areas where the model is making mistakes and make improvements to increase its accuracy and effectiveness [59]. Using a variety of evaluation metrics, we confirmed that our proposed approach is effective at detecting intrusions. These measures are briefly described as follows:

Accuracy: It is the ratio of the number of correct predictions to the total number of predictions made by the model. The equation for accuracy is present in Equation (8):

$$\text{Accuracy} = (\text{TP} + \text{TN}) \,/\, (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \tag{8}$$

Precision: It is the ratio of the number of true positives to the total number of positive predictions made by the model. The equation for precision is:

$$\text{Precision} = \text{TP} \,/\, (\text{TP} + \text{FP}) \tag{9}$$

Recall: It is the ratio of the number of true positives to the total number of actual positive samples in the dataset. The equation for recall is:

$$\text{Recall} = \text{TP} \,/\, (\text{TP} + \text{FN}) \tag{10}$$

F1-score: It is the harmonic mean of precision and recall. The equation for F1-score is:

$$\text{F1} - \text{score} = 2 * (\text{Precision} * \text{Recall}) \,/\, (\text{Precision} + \text{Recall} \tag{11}$$

From Equations (8)–(10), TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

Balanced Accuracy (BACC): It is the arithmetic mean of sensitivity and specificity. The equation for BACC is:

$$\text{BACC} = (\text{TPR} + \text{TNR}) \,/\, 2 \tag{12}$$

where TPR is the true positive rate (sensitivity) and TNR is the true negative rate (specificity).

Cohen's Kappa: A statistic measures the agreement between the predicted labels and the actual labels. The equation for Cohen's Kappa is:

$$\text{Kappa} = (\text{observed accuracy} - \text{expected accuracy}) \,/\, (1 - \text{expected accuracy}) \tag{13}$$

where observed accuracy is the proportion of agreement between the predicted labels and the actual labels, and expected accuracy is the proportion of agreement expected by chance.

Area Under the ROC Curve (AUC-ROC): It is a measure of the trade-off between true positive rate (TPR) and false positive rate (FPR). The equation for AUC-ROC is:

$$\text{AUC-ROC} = \int \text{TPR}(\text{FPR})\, d\text{FPR} \tag{14}$$

Where TPR is the true positive rate and FPR is the false positive rate. The AUC-ROC value ranges from 0 to 1, where a value of 1 indicates a perfect classifier and a value of 0.5 indicates a random classifier.

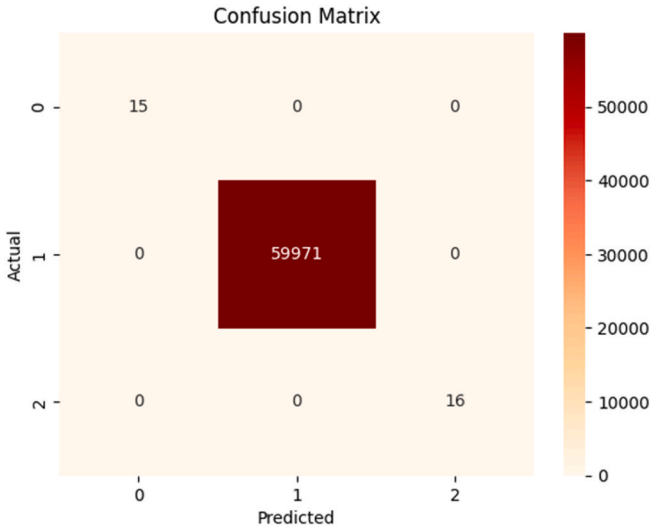

Fig. 2. Visualization of confusion matrix.

4. Result and analysis

In this part, we evaluate the performance of our proposed intrusion detection model. At first, using a dataset named SIMARGL21 and the model with random forest ensemble technique, we visualize the model. In the SIMARGL21 dataset, there are 50 features available. The features description is given in the appendix section. Moreover, in our experiment, we used 1330692 initial samples. It is a multi-classification dataset because the label column contains three types of values, Normal flow, XMAS Scan, NULL Scan. The XMAS Scan and the NULL Scan entries are considered intrusions. The evaluation metrics are then presented along with the other datasets and ensemble methods, including simple stacking, adaboost, gradient boosting, gradient XGBoost, and bagging. The same processes, which are described in detail in the section titled "Proposed Approach Developing", are utilized to record the model's results when applied to various datasets. For the purpose of evaluating the model's effectiveness, just the datasets are altered. The efficiency of the model's intrusion detection is then evaluated by comparing it to other intrusion detection models.

Table 1
Hyperparameters of the model for the different ensemble methods.

Ensemble Method	Hyperparameters
Random Forest	RandomForestClassifier(n_estimators = 10, max_depth = None, criterion = 'gini', min_samples_split = 2, min_samples_leaf = 1, max_leaf_nodes = None, min_weight_fraction_leaf = 0.0, oob_score = False, max_features = 'auto, min_impurity_decrease = 0.0, bootstrap = True, n_jobs = None, random_state = 42, verbose = 0, class_weight = None, warm_start = False, ccp_alpha = 0.0, max_samples = None)
Bagging Ensemble	BaggingClassifier(base_estimator = None, n_estimators = 10, bootstrap_features = False, max_samples = 1.0, max_features = 1.0, bootstrap = True, oob_score = False, n_jobs = None, warm_start = False, random_state = 42, verbose = 0)
Ada Boosting	AdaBoostClassifier(estimator = dt, n_estimators = 50, learning_rate = 1.0, algorithm = 'SAMME')
Gradient Boosting	GradientBoostingClassifier(loss = 'deviance', n_estimators = 100, learning_rate = 0.1, max_depth = 3)
Gradient XGBoost	XGBClassifier(learning_rate = 0.1, max_depth = 3, n_estimators = 100)
Simple Stacking	SVC(kernel = 'linear', C = 0.1), DecisionTreeClassifier(), GaussianNB(), these three base classifiers are used here.

Table 2
Values of the confusion matrix for different classes.

Test Data (20%)	TP	FN	FP	TN
Class 0	15	0	0	59987
Class 1	59971	0	0	31
Class 2	16	0	0	59986

Table 3
Evaluation metrics-1 for the model with various ensemble techniques.

Accuracy	Precision	Recall	F1-score	FPR
Model with Random Forest Ensemble Method				
1.00000	1.00000	1.00000	1.00000	0.00000
Model with Bagging Ensemble Method				
0.99998	0.99998	0.99998	0.99998	0.00001
Model with Adaboost Ensemble Method				
0.99963	0.99938	0.99963	0.99950	0.21506
Model with Gradient Boosting Ensemble Method				
0.99960	0.99950	0.99960	0.99952	0.19358
Model with Gradient XGBoost Ensemble Method				
0.99963	0.99938	0.99963	0.99950	0.21506
Model with Stacking Ensemble Method				
0.99996	0.99996	0.99996	0.99996	0.00001

4.1. Confusion matrix visualization

The heatmap is given in Fig. 2 for the visualization of the confusion matrix. It has 3 rows and 3 columns, representing the 3 classes in the multi-class classification problem. The predicted class is shown by the horizontal axis, while the actual class is represented by the vertical axis. The heatmap's cells show the number of times the predicted class and the actual class correspond.

The values of the confusion matrix for the three classes in the test data are shown in Table 2. For each class, the values for true positive (TP), true negative (TN), false negative (FN), and false positive (FP), is presented. The model achieved a high number of true positives for classes 0 and 1, and a moderate number of true positives for class 2. The test data doesn't predict any false positives or false negatives for any of the classes. Considering these outcomes, it appears that the model performed well in accurately predicting the classes in the test data.

Table 4
Evaluation metrics-2 for the model with various ensemble techniques.

BACC	Error Rate	Training Accuracy	Testing Accuracy	AUC Score
Model with Random Forest Ensemble Method				
1.00000	0.00000	1.00000	1.00000	1.00000
Model with Bagging Ensemble Method				
0.99999	0.00002	1.00000	0.99998	1.00000
Model with Adaboost Ensemble Method				
0.67371	0.00037	0.99935	0.99963	0.99925
Model with Gradient Boosting Ensemble Method				
0.71291	0.00040	0.99953	0.99960	0.54649
Model with Gradient XGBoost Ensemble Method				
0.67371	0.00037	0.99935	0.99963	0.99925
Model with Stacking Ensemble Method				
0.99999	0.00002	1.00000	0.99998	1.00000

Table 5
Evaluation metrics-3 for the model with various ensemble techniques.

Cohen's Kappa	Observed Accuracy (Po)	Expected Accuracy (Pe)	Training Time (s)	Testing Time (s)
Model with Random Forest Ensemble Method				
1.00000	1.00000	0.99897	3.08307	0.05435
Model with Bagging Ensemble Method				
0.98412	0.99998	0.99895	16.11728	0.13870
Model with Adaboost Ensemble Method				
0.49988	0.99963	0.99927	44.48168	0.48840
Model with Gradient Boosting Ensemble Method				
0.51986	0.99960	0.99917	688.86739	0.24656
Model with Gradient XGBoost Ensemble Method				
0.99988	0.99963	0.99897	146.46553	0.13396
Model with Stacking Ensemble Method				
0.98412	0.99998	0.99895	Few Hours	Few Hours

4.2. Result of the evaluation metrics of the developed model for various ensemble techniques

Table 3 provides the performance metrics for six different ensemble methods used for the classification tasks with the SIMARGL21 dataset. The metrics reported are accuracy acore, recall, precision, F1-score, and FPR (False Positive Rate). The Model with Random Forest Ensemble Method has an accuracy of 1.00000, precision of 1.00000, recall of 1.00000, F1-score of 1.00000, and an FPR of 0.00000. According to these performance measures, the Random Forest ensemble method-based model is doing incredibly well and is able to accurately identify all the positive and negative samples in the dataset without producing any false positive or false negative mistakes. This suggests that this model is a strong contender for the detection of intrusion.

Table 4 provides the performance metrics for six different ensemble methods used for intrusion detection classification task. The metrics reported are balanced accuracy (BACC), error rate, training accuracy, testing accuracy, and AUC score. The Model with Random Forest Ensemble Method has a BACC of 1.00000, error rate of 0.00000, training accuracy of 1.00000, testing accuracy of 1.00000, and AUC score of 1.00000. These performance metrics indicate that the model with the Random Forest ensemble method is performing extremely well and is able to correctly classify all the positive and negative samples in the dataset without making any errors. The balanced accuracy (BACC) is a measure of accuracy that takes into account imbalanced class distributions. An error rate of 0.00000 means that the model has made no errors in classifying the samples. The model appears to have learned the training data correctly, and its testing accuracy of 1.00000 indicates that it is capable of generalizing effectively to data that has not yet been seen. Finally, the model's AUC score of 1.00000 shows that it can differentiate between positive and negative samples with perfect accuracy.

Table 5 provides the performance metrics like Cohen's Kappa, observed accuracy (Po), expected accuracy (Pe), training time, and testing time. The Model with Random Forest Ensemble Method has a

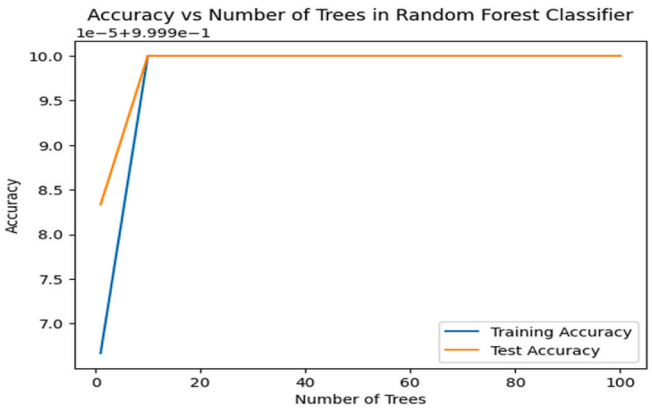

Fig. 3. Model accuracy on the increasing no of trees.

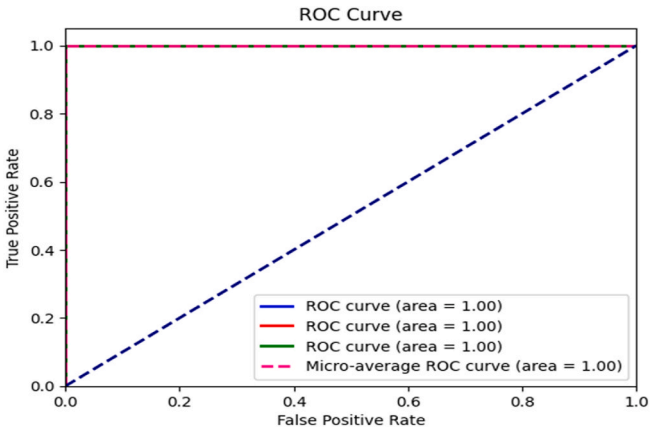

Fig. 4. ROC curve for TPR vs FPR.

Cohen's Kappa of 1.00000, observed accuracy (Po) of 1.00000, expected accuracy (Pe) of 0.99897, training time of 3.08307 s, and testing time of 0.05435 s. The Cohen's Kappa of 1.00000 suggests perfect agreement between the predicted and actual labels, while the observed accuracy of 1.00000 indicates that the model correctly classified all the samples in the dataset. The expected accuracy (Pe) of 0.99897 suggests that the model is able to classify the samples with a high degree of accuracy. In terms of performance metrics, the Model with Random Forest Ensemble Method outperforms all other models in table. It has a perfect Cohen's Kappa score, indicating perfect agreement between predicted and actual labels. Additionally, it has a perfect observed accuracy of 1.00000 and expected accuracy of 0.99897, suggesting that it can classify the samples with a high degree of accuracy. Finally, it has the lowest training and testing times, indicating that it is efficient and scalable.

So, considering all the evaluation metrics, the proposed approach with the Random Forest ensemble method can be used as a robust and accurate model for the detection of intrusion. The method works by combining multiple decision trees and aggregating their outputs to make the final prediction. This approach can lead to better generalization and improved performance compared to other models. Additionally, the short training and testing times make this model an attractive option for large-scale classification tasks.

4.3. Effectiveness of the described model with variable decision tree numbers in random forest

The model is trained using the appropriate features that have been chosen, and the model's correctness is determined using training and test sets for various numbers of trees. Fig. 3, illustrates about this. Following the selection of the n_trees bigger than 10, the model looks to perform incredibly well, obtaining a training accuracy score of 1.0 and a test accuracy score of 1.0 for all values of n_trees tested. Because the random forest ensemble classifier performs well on both the training and testing data for each example, it may generalize effectively to new, unexplored data without overfitting or underfitting.

4.4. ROC curve for false positive rate and true positive rate

The ROC curves for each class and the micro-average ROC curve for the intrusion detection issue are displayed in Fig. 4. For various threshold settings, the ROC curve illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR). The micro-average ROC curve displays the classifier's overall performance. The legend shows the micro-average AUC as well as the area under the ROC curve (AUC) for each class. A higher AUC value denotes a classifier that performs better. Based on the ROC curve and AUC values, we can judge the model's performance for this problem. The higher AUC values for each class and the micro-average AUC suggest that the model is

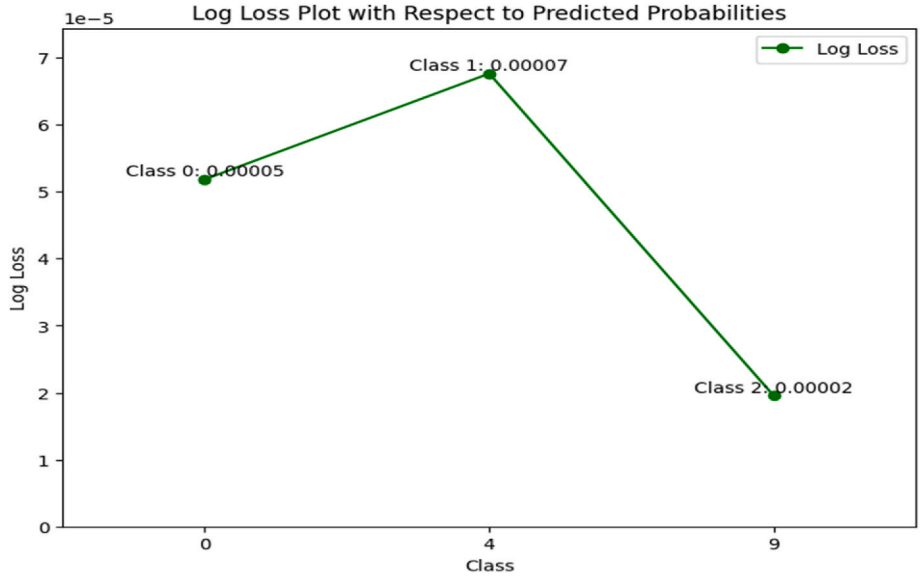

Fig. 5. Log-Loss vs Predicted Probability.

Table 6
Performance evaluation of the demonstrated approach against that of existing techniques on distinct datasets.

Dataset with Reference	Model with References	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
UNR-IDD [37]	Proposed Model	100.0	100.0	100.0	100.0
	RF, 2023 [37]	95.00	96.00	93.00	94.00
	FURIA, 2022 [60]	99.96	–	–	–
	Tab-SRU, 2022 [61]	99.00	98.00	97.00	97.00
SIMARGL2021 [38]	Proposed Model	100.0	100.0	100.0	100.0
	SFV and Domain Adaption, 2022 [62]	–	99.41	98.58	98.99
NF-UQ-NIDS [63]	Proposed Model	100.0	100.0	100.0	100.0
	RF, 2021 [64]	98.00	98.00	98.00	98.00
	Extra Trees ensemble, 2021 [63]	97.25	–	–	94.00
	2D-ACNN, 2023 [65]	95.20	–	–	–
NF-ToN-IoT [39]	Proposed Model	100.0	100.0	100.0	100.0
	SSW and XGBoost, 2022 [66]	98.80	98.80	98.80	98.80
	Extra Trees ensemble, 2021 [63]	99.66	–	–	100.00
	2D-ACNN, 2023 [65]	90.10	–	87.00	89.80
UKM-IDS20 [26]	Proposed Model	100.0	100.0	100.0	100.0
	HOE-DANN, 2021 [26]	96.66	–	–	–
CSE–CIC–IDS2018 [40]	Proposed Model	99.99	99.99	99.99	99.99
	RF, 2023 [37]	99.00	96.00	91.00	93.00
	CNN and LSTM, 2023 [67]	98.85	98.85	98.85	98.83
	AE, OCSVM and GMM, 2023 [33]	95.10	96.10	95.65	95.85
	Nearest Neighbor, 2022 [32]	98.58	96.67	97.15	96.21
WSN-DS [41]	Proposed Model	99.61	99.61	99.61	99.61
	RF, 2019 [68]	99.40	99.40	99.40	99.40
	ID-GOPA, 2021 [69]	96.00	96.00	96.00	96.00
UNSW-NB15 [42]	Proposed Model	100.0	100.0	100.0	100.0
	ARF and HAT, 2022 [29]	99.42	96.84	97.23	96.96
	Tab-SRU, 2022 [61]	99.23	99.04	99.64	99.34
	PCA and RF, 2022 [31]	95.10	94.80	95.70	95.10
	SVM, 2022 [30]	97.77	–	–	–
	ENAD, 2021 [28]	–	97.90	92.40	94.90
	HOE-DANN, 2021 [26]	94.08	–	–	–
	RF, 2019 [68]	90.30	98.80	86.70	92.40
	CNN-BiLSTM, 2020 [21]	77.16	82.63	79.91	81.25
Cyber Clean Center (CCC) [70]	Proposed Model	99.01	99.01	99.01	99.01
	Ensemble-based Stacking, 2023 [34]	94.08	71.42	86.50	78.24
	KNN, 2021 [71]	97.00	98.10	99.60	98.00
NSL-KDD [45]	Proposed Model	99.88	99.88	99.88	99.88
	RF, 2023 [37]	99.00	79.00	74.00	76.00
	DBN-SVM, 2019 [20]	97.45	97.48	97.78	98.62
	DBN, 2020 [22]	96.91	98.10	92.29	95.11
	CNN-BiLSTM, 2020 [21]	83.58	85.82	84.49	85.14
	CNN-IDS, 2020 [23]	83.00	85.00	83.00	83.00

The proposed machine-learning strategy evaluation metrics for various IDS datasets are displayed in Table 7. The evaluation metrics include the number of features, false positive rate (FPR), balanced accuracy, training accuracy, testing accuracy, and AUC score.

performing well and has a good discriminative ability.

4.5. Predicted probabilities with a log-loss plot

Fig. 5 shows the log-loss values for the random forest classifier on the SIMARGL21 dataset. The x-axis represents the three classes, where class 0, 1, and 2 correspond to the original class labels of 0, 4, and 9, respectively. The difference between the expected probability and the actual classes is shown on the y-axis by the log loss. The log-loss values are very small, indicating that the classifier performs well on the test set. The log-loss for class 0 is the smallest with a value of 0.00005, followed by class 2 with a value of 0.00002 and class 1 with a value of 0.00007.

These results indicate that the random forest classifier has successfully learned the underlying patterns in the data and can accurately predict the class labels for new data points.

4.6. The proposed models accuracy for different IDS datasets with existing models

Table 6 shows various intrusion detection datasets along with the proposed model and other existing models used, their accuracy, recall, precision, and F1-score. For the purpose of determining the evaluation measures in comparison to the currently available models, more than 10 well-known publicly accessible datasets are examined.

The table clearly shows that the suggested models had excellent accuracy for all datasets, ranging from 99.01% to 100%. In most cases, the proposed models outperformed the other models. The proposed model achieves perfect precision and recall on most datasets. This shows that the suggested model is quite good at identifying intrusions. On the majority of the datasets, the suggested model receives a flawless F1-score of 100%. This is a really impressive score, indicating that the

Table 7
Results of other evaluation metrics of different IDS datasets for the proposed model.

Dataset Name	Year	No. of Features	FPR	Balanced Accuracy (%)	Training Accuracy (%)	Testing Accuracy (%)	AUC Score (%)
UNR-IDD	2023	34	0.0000	100.0	100.0	100.0	100.0
NF-UQ-NIDS	2021	43	0.0000	100.0	100.0	100.0	100.0
NF-ToN-IoT	2021	43	0.0000	100.0	100.0	100.0	100.0
UKM-IDS20	2020	50	0.0000	100.0	100.0	100.0	100.0
CSE–CIC–IDS2018	2018	80	0.0000	99.99	100.0	99.99	99.99
WSN-DS	2016	17	0.0049	98.43	99.94	99.61	99.23
UNSW-NB15	2015	45	0.0000	100.0	99.99	100.0	100.0
CCC	2014	56	0.0126	98.73	100.0	98.86	99.62
NSL-KDD	2009	43	0.0006	92.61	99.99	99.88	99.68
KDDCup	1999	42	0.0012	93.54	99.99	99.93	99.92

suggested model is quite good at identifying network intrusions. The suggested models appear to be more effective in detecting all types of intrusions than the existing models, based on the high accuracy, recall score, precision, and F1-score values.

Overall, the proposed ensemble based machine-learning model achieved very high accuracy and AUC scores for most of the datasets, indicating its effectiveness in detecting intrusions. However, the false positive rate is very low for most of the cases. The results of the evaluation metrics are better for most of the cases and most of the existing models. Therefore, the model shows promising outcomes and has the potential as an intrusion detection tool.

5. Conclusion of the research

In the subject of network security, intrusion detection systems are essential considering the increasing number of network threats and advancements in technology. IDSs have attracted a lot of interest because of their ability to increase network security. The Random Forest-based ensemble model outperformed other strategies and attained excellent accuracy and detection rates, according to the results of our proposed IDS evaluation after that on various public datasets. The results we obtained show that the recommended approach is capable of identifying different attack types with accuracy and has the potential to be a helpful tool for enhancing the security of computer systems and networks against new cyber threats. Overall, the proposed approach has shown promising outcomes and has the potential to aid in the creation of intrusion detection systems for network security that are more effective.

Credit author statement

The research paper entitled "Ensuring Network Security with a Robust Intrusion Detection System Using Ensemble-based Machine Learning" was a collaborative effort between two authors, Md. Alamgir Hossain and Dr. Md. Saiful Islam; Md. Alamgir Hossain took the lead in this research by conceiving the idea, designing the model, implementing the system, and writing the paper. He played a crucial role in every aspect of the research, from the initial concept to the final manuscript.

Finding

The authors did not receive any funds for this research.

Declaration of competing interest

The authors declare that there are no conflicts of interest that could potentially influence the objectivity or integrity of the research findings. We have no financial, professional, or personal relationships with any individual, organization, or entity that could bias the interpretation or reporting of the research results. We affirm that the submitted manuscript is original, not previously published, and not under consideration elsewhere.

Data availability

All the datasets are citied in the paper.

Appendix

Features with the Description of the SIMARGL21 Dataset

Features	Description
OOORDER_IN_PKTS	Amount of inbound packets of data received out of sequence, possibly due to network congestion or packet manipulation.
DST_TO_SRC_SECOND_BYTES	The amount of data transmitted from the destination IP address to the source IP address within a specific time frame.
PROTOCOL_MAP	Mapping of protocol numbers to protocol names, which can help identify the type of traffic and potential intrusions.
RETRANSMITTED_OUT_PKTS	Number of outbound packets that were retransmitted, indicating potential network issues or malicious activity.
FLOW_END_SEC	Timestamp of when the flow ended, useful for correlating events and identifying suspicious behavior.
FIREWALL_EVENT	Indicator of whether the flow was permitted or denied by the firewall, important for identifying potential intrusions or misconfigurations.
PROTOCOL	Network protocol used in the flow, which can help identify the type of traffic and potential intrusions.
SAMPLING_INTERVAL	Time interval over which the data was sampled, important for understanding the context of the flow data.
FLOW_END_MILLISECONDS	Millisecond component of the flow end timestamp, useful for fine-grained analysis of events.
SRC_TO_DST_SECOND_BYTES	In the second portion of the flow, the number of bytes transported from origin to destination can be utilized to identify hidden channels or data espionage.
TOTAL_FLOWS_EXP	Total number of expected flows, useful for detecting anomalies or attacks.
RETRANSMITTED_IN_PKTS	Number of inbound packets that were retransmitted, indicating potential network issues or malicious activity.
FLOW_ID	Unique identifier for the flow, useful for correlating events and identifying suspicious behavior.
TCP_WIN_MAX_IN	This flow's optimum TCP window measurement can be utilized to identify possible TCP-based attacks.
TCP_WIN_SCALE_OUT	TCP window scale factor used in the outbound direction, which can help detect TCP-based attacks or performance issues.
IPV4_DST_ADDR	Destination IP address of the flow, which can help identify potential attacks or suspicious behavior.
FLOW_INACTIVE_TIMEOUT	Time interval after which the flow is considered inactive, important for network performance and security.
IN_BYTES	Number of inbound bytes in the flow, which can help identify potential attacks or suspicious behavior
TCP_WIN_MSS_IN	Maximum segment size received in the incoming direction, useful for identifying possible TCP-based cyberattacks.
TCP_WIN_MIN_IN	This flow's lowest TCP window dimension can be utilized to identify possible TCP-based cyberattacks.

(*continued on next page*)

(*continued*)

Features	Description
OUT_PKTS	The quantity of outgoing packets in the flow, which may be utilized to identify any potential malicious or network-related activities.
TCP_WIN_MSS_OUT	Maximum segment size sent in the outbound direction, which can be used to detect potential TCP-based attacks.
RETRANSMITTED_OUT_BYTES	Number of outbound bytes that were retransmitted, indicating potential network issues or malicious activity.
FLOW_START_MILLISECONDS	Millisecond component of the flow start timestamp, useful for fine-grained analysis of events.
OUT_BYTES	Number of outbound bytes in the flow, which can help identify potential attacks or suspicious behavior.
TCP_WIN_MIN_OUT	This flow's lowest TCP window dimension can be utilized to identify possible TCP-based cyberattacks.
BIFLOW_DIRECTION	Direction of the flow (unidirectional or bidirectional), important for network analysis and security.
FLOW_ACTIVE_TIMEOUT	Time interval after which the flow is considered active, important for network performance and security.
FLOW_DURATION_MICROSECONDS	Duration of the flow in microseconds, useful for understanding the context of the flow data at a more granular level.
L4_SRC_PORT	Source port number of the flow, which can help identify the type of traffic and potential attacks.
SRC_TOS	Type of Service (ToS) byte in the IP header of the packet, which can help identify the type of traffic and potential attacks.
IPV4_SRC_ADDR	Source IP address of the flow, which can help identify potential attacks or suspicious behavior.
FRAME_LENGTH	Length of the frame in bytes, which can help identify network performance issues or potential attacks.
FLOW_START_SEC	Timestamp of when the flow started, useful for correlating events and identifying suspicious behavior.
DIRECTION	Direction of the flow (inbound or outbound), important for network analysis and security.
MIN_IP_PKT_LEN	Minimum IP packet length in the flow, which can help identify potential attacks or suspicious behavior.
IN_PKTS	The quantity of incoming packets in the flow, which may be utilized to identify suspected malicious or network-related behavior.
LAST_SWITCHED	Timestamp of when the last packet of the flow was received, useful for identifying suspicious behavior or performance issues.
DST_TOS	Type of Service (ToS) byte in the IP header of the packet, which can help identify the type of traffic and potential attacks.
L7_PROTO_NAME	Application layer protocol used in the flow, which can help identify the type of traffic and potential attacks.
MAX_IP_PKT_LEN	Maximum IP packet length in the flow, which can help identify potential attacks or suspicious behavior.
L4_DST_PORT	Destination port number of the flow, which can help identify the type of traffic and potential attacks.
TCP_WIN_SCALE_IN	TCP window scale factor used in the inbound direction, which can help detect TCP-based attacks or performance issues.
TCP_WIN_MAX_OUT	This flow's optimum TCP window dimension can be utilized to identify possible TCP-based vulnerabilities.
FIRST_SWITCHED	Timestamp of when the first packet of the flow was received, useful for identifying suspicious behavior or performance issues.
FLOW_DURATION_MILLISECONDS	Duration of the flow in milliseconds, useful for understanding the context of the flow data.
OOORDER_OUT_PKTS	The total number of outbound packets that were transmitted outside the correct order, possibly due to network congestion or packet manipulation.
RETRANSMITTED_IN_BYTES	Number of inbound bytes that were retransmitted, indicating potential network issues or malicious activity.
TCP_FLAGS	TCP control flags in the TCP header of the packet, which can help identify the type of traffic and potential attacks.

References

[1] Mazhar T, et al. Analysis of cyber security attacks and its solutions for the smart grid using machine learning and blockchain methods. Future Internet Feb. 2023;15 (2):83. https://doi.org/10.3390/fi15020083.

[2] Venkatesh MM, V K R. Cyber security threats and countermeasures using machine and deep learning approaches: a survey. J Comput Sci Jan. 2023;19(1):20–56. https://doi.org/10.3844/jcssp.2023.20.56.

[3] Pallepati M. Network intrusion detection system using machine learning with data preprocessing and feature extraction. Int J Res Appl Sci Eng Technol Jun. 2022;10 (6):2360–5. https://doi.org/10.22214/ijraset.2022.44326.

[4] Perera S, Jin X, Maurushat A, Opoku D-GJ. Factors affecting reputational damage to organisations due to cyberattacks. Informatics Mar. 2022;9(1):28. https://doi.org/10.3390/informatics9010028.

[5] School of Computer Sc. and Engg.Chung-Ang University Korea, Abraham A, Grosan C. Cyber security and the evolution of intrusion detection systems. -Manag. J. Future Eng. Technol. Oct. 2005;1(1):74–82. https://doi.org/10.26634/jfet.1.1.968. Department of Computer Sc.Babes-Bolyai University, Cluj-Napoca 3400, Romania., Y. Chen, and School of Information Sc. and Engg.Jinan University, Jinan 250022 P.R.China,.

[6] Shinder L, Cross M. Facing the cybercrime problem head-on. In: Scene of the cybercrime. Elsevier; 2008. p. 1–39. https://doi.org/10.1016/B978-1-59749-276-8.00001-7.

[7] Bandakkanavar R. Krazy Tech, TECHNICAL PAPERS Causes of CyberCrime and Preventive Measures Jan. 2023. Accessed: Jan. 03, 2023. [Online]. Available: https://krazytech.com/technical-papers/cyber-crime.

[8] Sarker IH. Machine learning for intelligent data analysis and automation in cybersecurity: current and future prospects. Ann. Data Sci., Sep. 2022. https://doi.org/10.1007/s40745-022-00444-2.

[9] Jelen S. Intrusion detection systems: types, detection methods and challenges. securitytrails; Jul. 2020. https://securitytrails.com/blog/intrusion-detection-systems. [Accessed 1 January 2023].

[10] Network Security Concepts. Dangers, and defense best practical. Comput. Eng. Intell. Syst., Mar. 2023. https://doi.org/10.7176/CEIS/14-2-03.

[11] Jeong DH, Jeong B-K, Ji S-Y. Multi-resolution analysis with visualization to determine network attack patterns. Appl Sci Mar. 2023;13(6):3792. https://doi.org/10.3390/app13063792.

[12] Hachmi F, Boujenfa K, Limam M. Enhancing the accuracy of intrusion detection systems by reducing the rates of false positives and false negatives through multi-objective optimization. J Netw Syst Manag Jan. 2019;27(1):93–120. https://doi.org/10.1007/s10922-018-9459-y.

[13] Ahmed Md R, Islam salekul, Shatabda S, Islam AKMM, Robin Md TI. Intrusion Detection System in Software-Defined Networks Using Machine Learning and Deep Learning Techniques –A Comprehensive Survey Nov. 2022. https://doi.org/10.36227/techrxiv.17153213.v2. preprint.

[14] Musleh D, Alotaibi M, Alhaidari F, Rahman A, Mohammad RM. Intrusion detection system using feature extraction with machine learning algorithms in IoT. J Sens Actuator Netw Mar. 2023;12(2):29. https://doi.org/10.3390/jsan12020029.

[15] Jain S, Pawar PM, Muthalagu R. Hybrid intelligent intrusion detection system for internet of things. Telemat. Inform. Rep. Dec. 2022;8:100030. https://doi.org/10.1016/j.teler.2022.100030.

[16] Vijayakumar DS, Ganapathy S. Machine learning approach to combat false alarms in wireless intrusion detection system. Comput Inf Sci Jul. 2018;11(3):67. https://doi.org/10.5539/cis.v11n3p67.

[17] Mishra A, Thakur A. Study of machine learning classifiers for intrusion detection system. In: Sharma S, Peng S-L, Agrawal J, Shukla RK, Le D-N, editors. Data, engineering and applications. Lecture notes in electrical engineering, vol. 907. Singapore: Springer Nature Singapore; 2022. p. 213–24. https://doi.org/10.1007/978-981-19-4687-5_16.

[18] Papamartzivanos D, Gómez Mármol F, Kambourakis G. Dendron : Genetic trees driven rule induction for network intrusion detection systems. Future Generat Comput Syst Feb. 2018;79:558–74. https://doi.org/10.1016/j.future.2017.09.056.

[19] Halimaa A A, Sundarakantham K. Machine learning based intrusion detection system. In: 2019 3rd international conference on trends in electronics and informatics (ICOEI). Tirunelveli, India: IEEE; Apr. 2019. p. 916–20. https://doi.org/10.1109/ICOEI.2019.8862784.

[20] Yang H, Qin G, Ye L. Combined wireless network intrusion detection model based on deep learning. IEEE Access 2019;7:82624–32. https://doi.org/10.1109/ACCESS.2019.2923814.

[21] Jiang K, Wang W, Wang A, Wu H. Network intrusion detection combined hybrid sampling with deep hierarchical network. IEEE Access 2020;8:32464–76. https://doi.org/10.1109/ACCESS.2020.2973730.

[22] Elmasry W, Akbulut A, Zaim AH. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. Comput Network Feb. 2020; 168:107042. https://doi.org/10.1016/j.comnet.2019.107042.

[23] Al-Turaiki I, Altwaijry N, Agil A, Aljodhi H, Alharbi S, Lina A. Anomaly-based network intrusion detection using bidirectional long short term memory and convolutional neural network. ISC Intl J. Inf. Secur. Nov. 2020;12(3):37–44. https://doi.org/10.22042/isecure.2021.271076.624.

[24] Prasad M, Tripathi S, Dahal K. An efficient feature selection based Bayesian and Rough set approach for intrusion detection. Appl Soft Comput Feb. 2020;87: 105980. https://doi.org/10.1016/j.asoc.2019.105980.

[25] Panigrahi R, et al. Intrusion detection in cyber–physical environment using hybrid Naïve Bayes—decision table and multi-objective evolutionary feature selection. Comput Commun Apr. 2022;188:133–44. https://doi.org/10.1016/j.comcom.2022.03.009.

[26] Al-Daweri MS, Abdullah S, Zainol Ariffin KA. An adaptive method and a new dataset, UKM-IDS20, for the network intrusion detection system. Comput Commun Dec. 2021;180:57–76. https://doi.org/10.1016/j.comcom.2021.09.007.

[27] Sharafaldin I, Habibi Lashkari A, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th

international conference on information systems security and privacy, funchal. Madeira, Portugal: SCITEPRESS - Science and Technology Publications; 2018. p. 108–16. https://doi.org/10.5220/0006639801080116.
[28] Liao J, Teo SG, Pratim Kundu P, Truong-Huu T. ENAD: an ensemble framework for unsupervised network anomaly detection. In: 2021 IEEE international conference on cyber security and resilience (CSR). Rhodes, Greece: IEEE; Jul. 2021. p. 81–8. https://doi.org/10.1109/CSR51186.2021.9527982.
[29] Tabbaa H, Ifzarne S, Hafidi I. An online ensemble learning model for detecting attacks in wireless sensor networks. arXiv, Apr. 28. 2022. Accessed: Apr. 11, 2023. [Online]. Available: http://arxiv.org/abs/2204.13814.
[30] Tahri R, Balouki Y, Jarrar A, Lasbahani A. Intrusion detection system using machine learning algorithms. ITM Web Conf 2022;46:02003. https://doi.org/10.1051/itmconf/20224602003.
[31] Ahmed HA, Hameed A, Bawany NZ. Network intrusion detection using oversampling technique and machine learning algorithms. PeerJ Comput. Sci. Jan. 2022;8:e820. https://doi.org/10.7717/peerj-cs.820.
[32] Andrecut M. Attack vs benign network intrusion traffic classification. 2022. https://doi.org/10.48550/ARXIV.2205.07323.
[33] Wang C, Sun Y, Lv S, Wang C, Liu H, Wang B. Intrusion detection system based on one-class support vector machine and Gaussian mixture model. Electronics Feb. 2023;12(4):930. https://doi.org/10.3390/electronics12040930.
[34] Srinivasan S, D P. Enhancing the security in cyber-world by detecting the botnets using ensemble classification based machine learning. Meas. Sens. Feb. 2023;25:100624. https://doi.org/10.1016/j.measen.2022.100624.
[35] Jemili F, Meddeb R, Korbaa O. Intrusion detection based on ensemble learning for big data classification. In: Review. preprint; Feb. 2023. https://doi.org/10.21203/rs.3.rs-2596433/v1.
[36] Meidan Y, et al. N-BaIoT—network-Based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput Jul. 2018;17(3):12–22. https://doi.org/10.1109/MPRV.2018.03367731.
[37] Das T, Hamdan OA, Shukla RM, Sengupta S, Arslan E. UNR-IDD: intrusion detection dataset using network port statistics. In: 2023 IEEE 20th consumer Communications & networking conference (CCNC). Las Vegas, NV, USA: IEEE; Jan. 2023. p. 497–500. https://doi.org/10.1109/CCNC51644.2023.10059640.
[38] Mihailescu M-E, et al. The proposition and evaluation of the RoEduNet-SIMARGL2021 network intrusion detection dataset. Sensors Jun. 2021;21(13):4319. https://doi.org/10.3390/s21134319.
[39] Sarhan M, Layeghy S, Portmann M. Towards a standard feature set for network intrusion detection system datasets. Mobile Network Appl Feb. 2022;27(1):357–70. https://doi.org/10.1007/s11036-021-01843-0.
[40] A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018).” Accessed: Jan. 02, 2023. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018.
[41] Almomani I, Al-Kasasbeh B, Al-Akhras M, Wsn DS. A dataset for intrusion detection systems in wireless sensor networks. J Sens 2016;2016:1–16. https://doi.org/10.1155/2016/4731953.
[42] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 military Communications and information systems conference (MilCIS). Canberra, Australia: IEEE; Nov. 2015. p. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942.
[43] Yukonhiatou C, Kittitornkun S, Kikuchi H, Sisaat K, Terada M, Ishii H. Clustering Top-10 malware/bots based on download behavior. In: 2013 international conference on information technology and electrical engineering (ICITEE). Yogyakarta, Indonesia: IEEE; Oct. 2013. p. 62–7. https://doi.org/10.1109/ICITEED.2013.6676212.
[44] Dua D, Graff C. KDD cup 1999 data data set. Univ. Calif. Irvine Sch. Inf. Comput. Sci.; 2019. Accessed: Jan. 25, 2023. [Online]. Available: http://archive.ics.uci.edu/ml.
[45] NSL-KDD dataset. Accessed: Feb. 05, 2023, https://www.unb.ca/cic/datasets/nsl.html.
[46] P, et al. Scikit-learn: machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[47] Subasi A. Data preprocessing. In: Practical machine learning for data analysis using Python. Elsevier; 2020. p. 27–89. https://doi.org/10.1016/B978-0-12-821379-7.00002-3.
[48] Di Mauro M, Galatro G, Fortino G, Liotta A. Supervised feature selection techniques in network intrusion detection: a critical review. Eng Appl Artif Intell May 2021;101:104216. https://doi.org/10.1016/j.engappai.2021.104216.
[49] Duangsoithong R, Windeatt T. Correlation-based and causal feature selection analysis for ensemble classifiers. In: Schwenker F, El Gayar N, editors. Artificial neural networks in pattern recognition. Lecture notes in computer science, vol. 5998. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. p. 25–36. https://doi.org/10.1007/978-3-642-12159-3_3.
[50] Macedo F, Valadas R, Carrasquinha E, Oliveira MR, Pacheco A. Feature selection using decomposed mutual information maximization. Neurocomputing Nov. 2022;513:215–32. https://doi.org/10.1016/j.neucom.2022.09.101.
[51] Odhiambo Omuya E, Onyango Okeyo G, Waema Kimwele M. Feature selection for classification using principal component analysis and information gain. Expert Syst Appl Jul. 2021;174:114765. https://doi.org/10.1016/j.eswa.2021.114765.
[52] Martindale N, Ismail M, Talbert DA. Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data. Information Jun. 2020;11(6):315. https://doi.org/10.3390/info11060315.
[53] Chauhan NS. Random Forest® — a powerful ensemble learning algorithm. KDnuggets Jan. 2020. Accessed: Feb. 27, 2023. [Online]. Available: https://www.kdnuggets.com/2020/01/random-forest-powerful-ensemble-learning-algorithm.html.
[54] Ghojogh B, Crowley M. The theory behind overfitting, cross validation, regularization, bagging, and boosting. Tutorial 2019. https://doi.org/10.48550/ARXIV.1905.12787.
[55] Rehman Javed A, Jalil Z, Atif Moqurrab S, Abbas S, Liu X. Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles. Trans. Emerg. Telecommun. Technol. Oct. 2022;33(10). https://doi.org/10.1002/ett.4088.
[56] Brownlee J. How to develop a gradient boosting machine ensemble in Python,” machine learning mastery. Accessed: May 03, 2023, https://machinelearningmastery.com/gradient-boosting-machine-ensemble-in-python/.
[57] Brownlee J. Extreme gradient boosting (XGBoost) ensemble in Python,” machine learning mastery. Accessed: Oct. 02, 2023, https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/.
[58] Brownlee J. Stacking ensemble machine learning with Python,” machine learning mastery. Apr. 2021. Accessed: Jan. 03, 2023. [Online]. Available: https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/.
[59] H M, S MN. A review on evaluation metrics for data classification evaluations. Int. J. Data Min. Knowl. Manag. Process Mar. 2015;5(2):1–11. https://doi.org/10.5121/ijdkp.2015.5201.
[60] Pawar K, Mohite B, Kshirsagar P. Analysis of feature selection methods for UKM-IDS20 dataset. In: Iyer B, Crick T, Peng S-L, editors. Applied computational technologies. Smart innovation, systems and technologies, vol. 303. Singapore: Springer Nature Singapore; 2022. p. 461–7. https://doi.org/10.1007/978-981-19-2719-5_43.
[61] Chen Y, Li J, Guo N. Efficient and interpretable SRU combined with TabNet for network intrusion detection in the big data environment. Int J Inf Secur Dec. 2022. https://doi.org/10.1007/s10207-022-00656-w.
[62] Komisarek M, Kozik R, Pawlicki M, Choraś M. Towards zero-shot flow-based cyber-security anomaly detection framework. Appl Sci Sep. 2022;12(19):9636. https://doi.org/10.3390/app12199636.
[63] Sarhan M, Layeghy S, Moustafa N, Portmann M. NetFlow datasets for machine learning-based network intrusion detection systems. In: Deze Z, Huang H, Hou R, Rho S, Chilamkurti N, editors. Big data technologies and applications. Lecture notes of the Institute for computer sciences, social informatics and telecommunications engineering, vol. 371. Cham: Springer International Publishing; 2021. p. 117–35. https://doi.org/10.1007/978-3-030-72802-1_9.
[64] Komisarek M, Pawlicki M, Kozik R, Hołubowicz W, Choraś M. How to effectively collect and process network data for intrusion detection? Entropy Nov. 2021;23(11):1532. https://doi.org/10.3390/e23111532.
[65] Nizamudeen SMT. Intelligent intrusion detection framework for multi-clouds – iot environment using swarm-based deep learning classifier. In: Review, preprint; Jan. 2023. https://doi.org/10.21203/rs.3.rs-2409418/v1.
[66] Karanfilovska M, Kochovska T, Todorov Z, Cholakoska A, Jakimovski G, Efnusheva D. Analysis and modelling of a ML-based NIDS for IoT networks. Procedia Comput Sci 2022;204:187–95. https://doi.org/10.1016/j.procs.2022.08.023.
[67] Wang Y-C, Houng Y-C, Chen H-X, Tseng S-M. Network anomaly intrusion detection based on deep learning approach. Sensors Feb. 2023;23(4):2171. https://doi.org/10.3390/s23042171.
[68] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access 2019;7:41525–50. https://doi.org/10.1109/ACCESS.2019.2895334.
[69] Ifzarne S, Tabbaa H, Hafidi I, Lamghari N. Anomaly detection using machine learning techniques in wireless sensor networks. J. Phys. Conf. Ser. Jan. 2021;1743(1):012021. https://doi.org/10.1088/1742-6596/1743/1/012021.
[70] Feng Y, Akiyama H, Lu L, Sakurai K. Feature selection for machine learning-based early detection of distributed cyber attacks. In: 2018 IEEE 16th intl conf on dependable, autonomic and secure computing, 16th intl conf on pervasive intelligence and computing, 4th intl conf on big data intelligence and computing and cyber science and technology congress(DASC/PiCom/DataCom/CyberSciTech). Athens: IEEE; Aug. 2018. p. 173–80. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00040.
[71] Joshi C, Bharti V, Ranjan RK. Botnet detection using machine learning algorithms. In: Dave M, Garg R, Dua M, Hussien J, editors. Proceedings of the international conference on paradigms of computing, communication and data sciences. In algorithms for intelligent systems. Singapore: Springer Singapore; 2021. p. 717–27. https://doi.org/10.1007/978-981-15-7533-4_56.