Journal of Big Data

RESEARCH Open Access

Shielding networks: enhancing intrusion detection with hybrid feature selection and stack ensemble learning

Ali Mohammed Alsaffar[1,2], Mostafa Nouri-Baygi[1*] and Hamed M. Zolbanin[3]

*Correspondence:
nouribaygi@um.ac.ir

[1] Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[2] Department of Computer Techniques Engineering, Imam Al-Kadhum College (IKC), Baghdad, Iraq
[3] School of Business Administration, University of Dayton, Dayton, OH, USA

Abstract

The frequent usage of computer networks and the Internet has made computer networks vulnerable to numerous attacks, highlighting the critical need to enhance the precision of security mechanisms. One of the most essential measures to safeguard networking resources and infrastructures is an intrusion detection system (IDS). IDSs are widely used to detect, identify, and track malicious threats. Although various machine learning algorithms have been used successfully in IDSs, they are still suffering from low prediction performances. One reason behind the low accuracy of IDSs is that existing network traffic datasets have high computational complexities that are mainly caused by redundant, incomplete, and irrelevant features. Furthermore, standalone classifiers exhibit restricted classification performance and typically fail to produce satisfactory outcomes when dealing with imbalanced, multi-category traffic data. To address these issues, we propose an efficient intrusion detection model, which is based on hybrid feature selection and stack ensemble learning. Our hybrid feature selection method, called MI-Boruta, combines mutual information (MI) as a filter method and the Boruta algorithm as a wrapper method to determine optimal features from our datasets. Then, we apply stacked ensemble learning by using random forest (RF), Catboost, and XGBoost algorithms as base learners with multilayer perceptron (MLP) as meta-learner. We test our intrusion detection model on two widely recognized benchmark datasets, namely UNSW-NB15 and CICIDS2017. We show that our proposed IDS outperforms existing IDSs in almost all performance criteria, including accuracy, recall, precision, F1-Score, false positive rate, true positive rate, and error rate.

Keywords: Intrusion detection system, Machine learning, Feature selection, Stacked ensemble

Introduction

The Internet has turned into an important part of our daily lives and an indispensable tool. The reliance of people on the Internet for accessing a range of electronic services, such as online education, banking, business, job search, and shopping, as part of their daily routines has been steadily increasing. Following the widespread growth in Internet usage, the cyberthreat landscape has evolved rapidly [1] and security breaches have now become a routine [2], making information systems, computer networks, and activities

associated with them prone to new and sophisticated attacks. Consequently, recent years have witnessed a surge in data and privacy breaches [2, 3] and the security of cyberspace has become the focus of much more scrutiny. For instance, many studies have focused on detecting cyberattacks on computer networks, such as the Internet of Things [4, 5], vehicle networks [6], and wireless networks [7], or on developing secure systems in certain business sectors, such as healthcare [8, 9] and energy [1]. Regardless of the application domain, what cybersecurity initiatives have in common is that they all need to develop a robust security system to fend off cyberattacks. Although different security tools have been used to secure networks, such as user authentication mechanisms, data encryption techniques, firewalls, and antivirus software, they cannot be effective against all threats and cyberattacks [10]. Detecting cyberattacks, which are constantly growing in numbers and sophistication, requires intelligent security mechanisms, such as intrusion detection systems (IDSs) to protect networks and information systems from an array of threats [11, 12]. IDS is a hardware or software system that can carry out a variety of tasks, such as analyzing and monitoring computer networks, examining abnormal activity patterns, monitoring and inspecting user and system activities, and identifying common attacks. Hence, the criticality of IDS in network security cannot be overstated as it plays a crucial role in detecting and preventing malicious activities [13] that have the potential to violate security policies and compromise the confidentiality, integrity, and availability (CIA) of information [14, 15].

Generally, IDS are classified into two types: host intrusion detection systems (HIDS) and network intrusion detection systems (NIDS) [16]. A HIDS is established on a single host to monitor all activities on the host, e.g., scanning for suspicious activities and violations of security policies. On the other hand, a NIDS is established to secure all devices and networks against potential security breaches. IDS can also be categorized into two types based on the detection mechanism they use: (1) signature-based IDS or misuse-based IDS, and (2) anomaly-based IDS [17]. Signature-based IDS identifies known attack patterns by utilizing a database of their signatures [18]. The main disadvantage of these systems is that they cannot detect zero-day attacks or more recent attacks if their databases are outdated. Anomaly-based IDS, on the other hand, are designed to monitor and analyze all actions across a network by learning normal and anomalous network behaviors; therefore, they can also detect novel attacks [19, 20].

Over the past two decades, the performance of IDSs has been enhanced through the application of various machine learning techniques; however, these techniques still face various challenges. First, since intrusion detection algorithms deal with large amounts of network traffic data, and these data contain redundant, noisy, and/or irrelevant features, reducing the feature space by selecting the most informative features still bears an important weight on the performance of IDS. Second, the use of single algorithm models may create statistical, representational, or computational issues that are driven by the numerous, multi-dimensional features of massive datasets. In the context of IDS, a single algorithm classifier may reduce the efficiency of an IDS and make it unable to detect multiclass attacks.

Our goal in this paper is to address these challenges. Initially, we opt for the most suitable subset of features to enhance the effectiveness of machine learning algorithms, which helps to reduce the amount of data, storage space, and processing cost required.

Next, we develop a stacked ensemble learning model to detect cyberattacks. Finally, we test our model using two benchmark datasets and compare our results with existing studies, most of which have developed IDS using single algorithm models. The experimental results show that our model performs well based on different evaluation metrics for binary and multi-class classification. Our study offers the following contributions:

1. Using a combination of filter and wrapper methods, it develops a hybrid feature selection method that combines the benefits of the two approaches and enables the development of a simpler, yet more accurate intrusion detection model.
2. It uses random forest (RF), Catboost, and XGBoost algorithms as base learners and multilayer perceptron (MLP) as the meta-learner to develop a stacked ensemble model that improves the multi-class classification performance of existing models in the literature.
3. Constructing and testing the model on two well-known benchmark datasets of different sizes illustrate that the proposed model is generalizable.

The subsequent sections of the paper are structured as follows. Sect. "Related work" presents the related work for IDS feature selection techniques and ensemble learning. In Sect. "Methodology", the proposed methodology is presented. Sect. "Experimental Results and Discussion" discusses the experimental analysis and evaluates the results. Finally, Sect. "Conclusion" concludes the paper.

Related work

Many studies have been carried out to implement an efficient IDS. From a broad perspective, these studies can be classified based on two criteria: the type of feature selection method and the type of classification algorithm they use. Feature selection methods can be further divided into the filter, wrapper, and embedded methods. Similarly, classification algorithms can be split into single and ensemble learning algorithms. In what follows, we present a summary of the literature on IDSs, focusing on feature selection and model building approaches used in those studies.

The main idea behind filter methods is to select the most relevant features from a dataset based on their intrinsic characteristics, without involving a machine learning algorithm to predict the target variable. In filter feature selection methods, the features are evaluated based on some statistical measure, such as correlation [21, 22], information gain[23– 27], chi-square test [28–30], or ANOVA F-test [26, 30], and then ranked according to their scores. The top-ranked features are then selected for further analysis or used as input for a machine learning algorithm. Some advantages of using filter feature selection methods are that they are computationally efficient, easy to implement, and can work well with high-dimensional datasets; however, they may not always capture complex relationships between features and the target variable, and may also suffer from redundancy issues [31, 32].

In contrast to filter methods, wrapper feature selection methods use a machine learning algorithm to evaluate the relevance of features and select the best subset for a given model. In wrapper methods, the feature selection process is integrated with the training process of a machine learning algorithm. The algorithm is used to evaluate the

performance of different subsets of features and select the one that results in the best model performance. This process is typically done using a cross-validation approach, where the dataset is divided into training and validation sets, and the algorithm is evaluated on multiple iterations. Wrapper feature selection methods can be more accurate than filter methods, as they take into account the interactions between features and the target variable. However, they can be computationally expensive and prone to overfitting if the number of features is large compared to the size of the dataset. Examples of wrapper methods that are used in prior studies include recursive feature elimination (RFE) [33, 34], Pigeon-inspired optimization [35, 36], Cuckoo Search Optimization [37], Particle Swarm Optimization [38, 39], Bat Optimization [40], Grey Wolf Optimization [39], and the Grasshopper optimization algorithms [41]. These methods are commonly used in machine learning applications where the number of features is high, and the goal is to find the optimal subset of features that leads to the best model performance [42, 43].

Embedded methods are techniques for feature selection that involve incorporating the feature selection process into the model training process itself. These methods aim to find the most informative subset of features to improve model performance and reduce overfitting. In embedded methods, feature selection is performed during the model training process to select the best subset of features that can be used to train the model. This can be achieved by adding a penalty term to the model's objective function that encourages the selection of only the most informative features. Some examples of embedded feature selection methods that are used in previous studies include Gradient Boosting [26, 44–46], and decision tree-based algorithms such as Random Forest [30, 47, 48], which select the most important features at each split. Embedded feature selection methods are often preferred over traditional filter and wrapper methods because they can lead to more accurate and efficient models by simultaneously selecting the most relevant features and optimizing model performance. However, they can be computationally expensive and require more resources compared to other feature selection methods [31, 49].

Regarding the classification algorithms used in previous studies, single algorithm classifiers that predict binary outcomes [24, 26, 35–40] comprise a significant proportion. In contrast, a smaller proportion of single algorithm efforts [23, 33, 47, 50] have tackled the more challenging problem of predicting multi-class network attacks. Some studies [17, 25, 41, 44, 46] have predicted both binary and multi-class outcomes. A similar pattern is observed among the ensemble learning models: studies that predict binary outcomes include [21, 27–29, 51, 52]; those that focus on multi-class prediction include [34], and studies that predict network attacks both as binary and multi-class outcomes include [22, 30, 48, 53]. Table 1 provides a summary of previous studies that have developed IDSs using machine learning techniques.

Our review of prior research identifies several gaps in the literature. First, a large number of prior studies have used outdated datasets, such as KDD99 or NSL-KDD, that do not cover modern network attacks. Second, when building and testing an IDS using large datasets, such as CICIDS2017, researchers have typically tended to focus on a subset of commonly occurring network attacks while disregarding the infrequent ones. Therefore, their IDSs are unable to detect all possible cyber threats. Third, only a few attempts have been made to evaluate intrusion detection systems using multiple datasets that vary in

Table 1 A Summary of Prior Research on Intrusion Detection Systems

Author(s)	FS techniques	FS techniques type	Classification algorithm techniques	Classification algorithm techniques type	Type of classification	Datasets
[23]	IG	Filter	NB, BN, RF J48, RT	Single	Multi-class	CICIDS2017
[24]	IG	Filter	MLP	Single	Binary	UNSW-NB15
[44]	XGBoost	Embedded	DT, ANN, LR, KNN, SVM	Single	Binary & Multi-class	UNSW-NB15
[25]	IG, Gain Ratio	Filter	DT, NB	Single	Binary & Multi-class	UNSW-NB15
[17]	Genetic Algorithm	Wrapper	LSTM + RNN	Single	Binary & Multi-class	NSL-KDD
[35]	PIO	Wrapper	DT	Single	Binary	KDD99, NSL-KDD, UNSW-NB15
[38]	PSO	Wrapper	DT, KNN	Single	Binary	KDD99
[40]	BOA	Wrapper	SVM	Single	Binary	KDD99
[50]	TS–RF	Wrapper	RF	Single	Multi-class	UNSW-NB15
[39]	PSO, GWO	Wrapper	SVM, DT	Single	Binary	NSL-KDD, UNSW-NB15
[33]	RFE	Wrapper	DT, RF, KNN, NB	Single	Multi-class	NSL-KDD
[37]	CS	Wrapper	ANN	Single	Binary	NSL-KDD
[36]	PIO	Wrapper	CNN	Single	Binary	NSL-KDD
[45]	LightGBM	Embedded	AE, DAE, VAE	Single	Binary	NSL-KDD
[46]	LightGBM	Embedded	DNN	Single	Binary & Multi-class	KDD99, NSL-KDD, UNSW-NB15
[41]	Ensemble Feature Selection, Grasshopper Optimization	Wrapper	SVM	Single	Binary & Multi-class	KDD99, NSL-KDD
[47]	IGRF-RFE	Filter + Embedded + Wrapper	MLP	Single	Multi-class	UNSW-NB15
[26]	ANOVA, Pearson Correlation, Gradient Boosting, IG, Greedy Algorithms	Filter + Wrapper + Embedded	GRU, Bi-LSTM +, CNN, RF, DT	Single	Binary	NSL-KDD, UNSW-NB15
[28]	Chi-Square	Filter	XGBoost + K-means + BPNN + SVM	Ensemble	Binary	NSL-KDD
[29]	Chi-Square	Filter	NB + SVM + LR + MLP	Ensemble	Binary	NSL-KDD
[22]	Correlation	Filter	Adaboost + Bagging + REPTree + J48 + RF	Ensemble	Binary & Multi-class	KDD99, NSLKDD
[27]	IG	Filter	REPTree + RF + RT + DS + J48	Ensemble	Binary	NSL-KDD
[21]	Correlation- & Consistency-based methods	Filter	RF + BPNN + RBFN	Ensemble	Binary	KDD99, Honeynet environment
[51]	Variance threshold	Filter	XGBoost + RF + DT	Ensemble	Binary	NSL-KDD, UNSW-NB15
[34]	RFE	Wrapper	RF + SVM + DT	Ensemble	Multi-class	KDD99
[52]	SFS	Wrapper	XGBoost + ET + RF + Bagging + MLP	Ensemble	Binary	NSL-KDD
[48]	RF	Embedded	GBM + RF	Ensemble	Binary & Multi-class	NSL-KDD
[53]	Gradient Boosting Trees	Embedded	XGBoost	Ensemble	Binary & Multi-class	CICIDS2017

Table 1 (continued)

Author(s)	FS techniques	FS techniques type	Classification algorithm techniques	Classification algorithm techniques type	Type of classification	Datasets
[30]	Chi-Square, ANOVA, Linear SVM, RF	Filter + Embedded	DT + NB + LR	Ensemble	Binary & Multi-class	CICIDS2017

terms of the types of attacks they represent as well as their size. Fourth, few studies have focused on developing and testing IDSs for multi-class prediction using datasets that include all current types of network attacks. Lastly, almost all previous studies, especially those predicting multi-class attacks, have a fairly large false positive rate.

In order to address these deficiencies, we develop an IDS that improves the accuracy of predicting network attacks, while reducing the false positive rate. Our approach involves employing a hybrid feature selection method, which we call MI-Boruta, to optimize feature selection through the use of both mutual information (MI) as a filter method and the Boruta algorithm as a wrapper method. After selecting the most relevant features from our datasets, we construct a stacked ensemble model that combines several machine learning algorithms, including random forest (RF), Catboost, and XGBoost, as base learners, with a multilayer perceptron (MLP) serving as the meta-learner. We evaluate the effectiveness of our proposed method by conducting experiments on two widely recognized benchmark datasets.

Methodology

As we explained before, filter, wrapper, and embedded feature selection models have their advantages and disadvantages. In this study, we use a combination of filter and wrapper methods to reap the benefits of both feature selection techniques. [1] When combining these two methods, filter methods can be used initially to identify the most relevant features based on some criterion, such as correlation with the target variable, and these selected features can then be passed on to the wrapper method for further evaluation. Wrapper methods can then be used to fine-tune the subset of features, by considering all possible combinations and selecting the subset that maximizes the model performance. This combination approach can help to reduce the number of features and improve the accuracy and interpretability of the model while avoiding some of the drawbacks of either method used alone.

After implementing our hybrid feature selection method, we utilize a stacked ensemble learning classifier to enhance the detection performance of IDSs. The architecture and stages of our methodology are illustrated in Fig. 1, and we elaborate further on the individual steps of our approach in the subsequent sections.

Data description

This research uses two benchmark datasets: the UNSW-NB15 and the CICIDS2017 datasets. We describe these two datasets in the following subsections.

[1] We do not consider embedded methods for feature selection to keep the computational cost of our methodology as low as possible, given that we will be building and testing our IDS on a significantly large benchmark dataset.

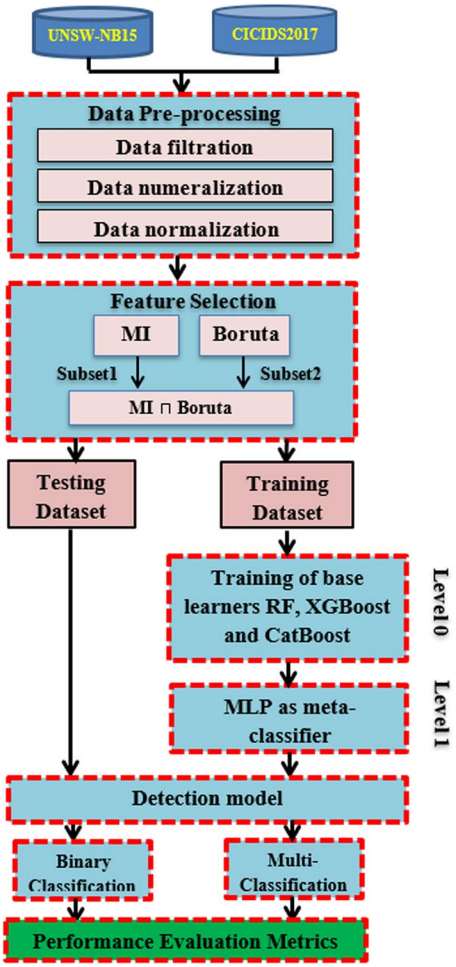


Fig. 1 Proposed Methodology

UNSW-NB15 dataset (D_1)

Created by the Australian cybersecurity center [54], the UNSW-NB15 dataset (D_1) comprises both training and testing partitions. The training set is composed of 175,341 records, whereas the test set includes 83,332 records. The records of D_1 can be broadly classified into normal network traffic and cyberattack, which itself has different types. The dataset has 44 features, two of which are class labels (i.e., *attack cat* and *label*). Features of the dataset are of binary, categorical, or numeric (float and integer) formats. D_1 consists of ten classes, wherein one class represents normal traffic, and the remaining nine classes represent different types of attacks: DoS, Worms, Backdoors, Fuzzers, Shellcode, Generic, Reconnaissance, Exploits, and Port scans [55]. The distribution of these ten classes is shown in Table 2. In the D_1 dataset, the majority class is "Normal" with 93,000 instances and the minority class is "Worms" with 174 instances. The imbalance ratio for each class is shown in Fig. 2. As we see in this figure, $\mathbf{D_1}$ is highly imbalanced, with the majority class being over 534 times more frequent than the minority class.

$$\text{Imbalance Ratio} = \frac{Number\ of\ instances\ in\ majority\ class}{Number\ of\ instance\ in\ minority\ class} \tag{1}$$

Table 2 The class distribution of the UNSW-NB15 dataset

Attack type	Training dataset	Testing dataset
Normal	56,000	37,000
Generic	40,000	18,871
Exploits	33,393	11,132
Fuzzers	18184	6062
Denial of service (DoS)	12,264	4089
Reconnaissance	10,491	3496
Analysis	2000	667
Backdoor	1746	583
Shellcode	1133	378
Worms	130	44
Total	175,341	82,332

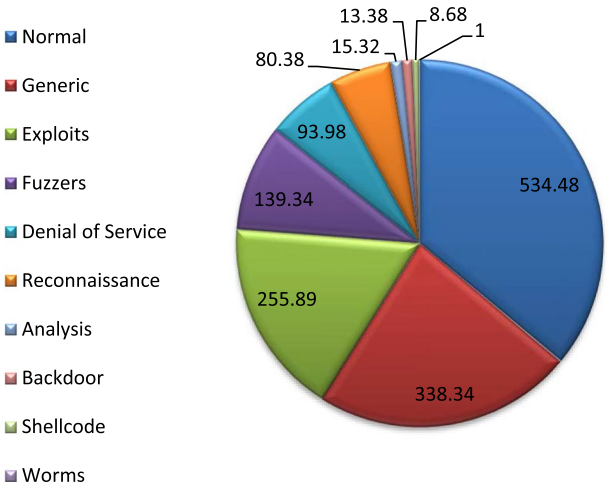


Fig. 2 Distribution of classes for D_1

CICIDS2017 dataset (D_2)

This dataset was created by the Canadian Institute for Cybersecurity (CIC) in 2017 [56] and includes 2,830,743 records that are spread across eight files. Each record comprises 78 features plus a label. D2 contains 15 classes, including benign network traffic and 14 attack types: DDoS, DoS Slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, Heartbleed, PortScan, Bot, FTP-Patator, SSH-Patator, Web Attack-Brute Force, Web Attack-XSS, Web Attack-SQL Injection, and Infiltration. Table 3 shows the class distribution of D_2. Before preprocessing this dataset, we combined DoS slowloris, DoS Slowhttptest, DoS Hulk, and DoS GoldenEye into a single class called DOS attack. Similarly, we aggregated Web Attack-XSS, Web Attack-Sql-Injection, and Web Attack-Brute Force to form a single class called the Web Attack class. In the D_2 dataset, the majority class is "Benign" which has 2,273,097 instances, and the minority class is "Heartbleed" with 11 instances. The imbalance ratio for each class is shown in Fig. 3. As we see in this figure, D_2 is highly imbalanced. The "Benign" class is over 206,649 times more frequent than the "Heartbleed" class.

Table 3 The class distribution of the CICIDS2017 dataset

Category	Type	Total
BENIGN	BENIGN	2,273,097
DoS	DDoS	128,027
	DoS slowloris	5796
	DoS Slowhttptest	5499
	DoS Hulk	231,073
	DoS GoldenEye	10,293
PortScan	PortScan	158,930
Bot	Bot	1966
Brute-force	FTP-Patator	7938
	SSH-Patator	5897
Web attack	Web Attack-Brute Force	1507
	Web Attack-XSS	652
	Web Attack-SQL Injection	21
Heartbleed	Heartbleed	11
Infiltration	Infiltration	36
Total Attacks		471,454
Total Records		2,830,743

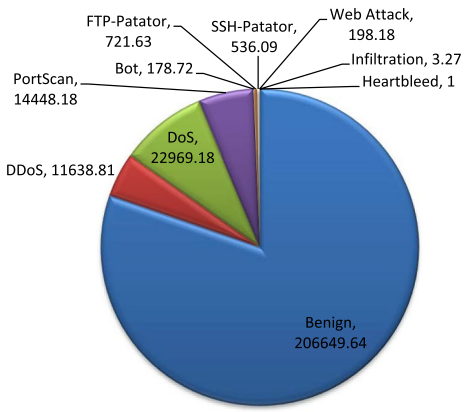


Fig. 3 Distribution of classes for D_2

Drawing from the methodologies employed by leading authorities in the fields of statistical analysis and deep learning [57, 58], we aggregated the training and testing datasets for both D_1 and D_2. Subsequently, we randomly partitioned the combined data into two non-overlapping subsets, with 80% of the data allocated for model training and 20% reserved for testing and validation. This approach is particularly suited for larger datasets as it circumvents the computationally intensive run-time required by k-fold cross-validation. Moreover, it affords the model with sufficient data to learn from, while still retaining an adequate amount of data for validation and testing purposes. Following the construction of our model using the training datasets, we will proceed to assess its binary and multiclass classification performance for both D_1 and D_2.

Data preprocessing

Data preprocessing is a necessary step in data mining that transforms the data into a suitable format for analysis and knowledge discovery. This phase contains three steps: data filtration, data numeralization, and data normalization.

Data filtration

Real-world data typically contain anomalous, incomplete, and inconsistent values that come from heterogeneous platforms and can degrade the classification performance of machine learning models. For instance, the feature 'Flow Packets/s' in D_2 includes abnormal values such as 'NaN' and 'Infinity' that may negatively influence the classification accuracy of predictive models. To deal with such instances, we applied basic data cleaning, including removing infinite data values and missing data samples from the datasets.

Data numeralization

This step converts the symbolic data (non-numeric features) into numerical values (numeric features) by using 1-N numeric coding, where N is the number of symbols. This paper has converted all the non-numeric features such as *proto*, *service*, and *state* in D_1 into numerical values.

Data normalization

Feature scaling is an important step in data preprocessing. In this study, we applied the *min–max normalization* to scale the features within the range of [0,1]. The formula for the *min–max normalization* is given below:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2}$$

where $X_{normalized}$ represents the value after normalization, X is the value before normalization, X_{min} is the minimum value of the feature, and X_{max} is the maximum value of the feature.

Feature selection

Feature selection is an important aspect of constructing an efficient IDS. Feature selection eliminates irrelevant and redundant features from a dataset to improve the efficiency of machine learning algorithms. As discussed earlier, there are three main feature selection methods: filter methods [32], wrapper methods [59], and embedded methods [60]. Filter methods select the most useful features without using any learning algorithm. They are based on the intrinsic characteristics of features, such as correlation, consistency, information distance, statistical measures, etc. These methods are fast and have low computational complexity. Wrapper methods employ a learning algorithm to determine a subset of features, which are subsequently evaluated by the classification algorithm. As a result, these methods can be computationally complex. Embedded methods are similar to wrapper methods, but the learning algorithm and search strategy are achieved simultaneously. In other words, feature selection in these methods is made during the learning process itself, which could make them computationally intensive for large datasets. In this section, we propose a hybrid features selection method

by combining mutual information as the filter method and the Boruta algorithm as the wrapper method to select the best features from each of the datasets.

Mutual information

Mutual information (MI) is one of the most used filter feature selection methods [61]. It measures the amount of mutual dependence between two random variables (features). More precisely, MI measures the quantity of information that is gained about one random variable by observing the other random variable. Mutual information has several advantages over other filter feature selection methods. First, mutual information is a non-parametric method, which means it does not make assumptions about the distribution of the data. This is particularly useful when dealing with complex or non-linear relationships between variables. Second, mutual information can capture both linear and non-linear [62] dependencies between variables. This is in contrast to some other methods, such as the Pearson correlation, which only capture linear relationships. Third, mutual information is robust to noise and outliers [63], which can be a problem for some other methods.

For these reasons, we utilize MI in the first step of our feature selection. MI can be calculated using the following equation:

$$MI = (X; Y) = e\,(X) - e\,(X|Y) \tag{3}$$

where $MI = (X; Y)$ is the value of mutual information for variables X and Y, $e(X)$ represents the marginal entropy of X, and $e(X|Y)$ represents the conditional entropy of X given Y.

Boruta algorithm

Boruta is a wrapper feature selection algorithm [64] that identifies the most important variables in a dataset. It works by creating shadow features that mimic the original features and comparing their importance to that of the original features using a random forest model [65]. The algorithm then assigns a tentative status to each feature based on its importance relative to the shadow features. Features that are more important than shadow features are given a confirmed status, while those that are less important are given a rejected status. The algorithm iteratively re-runs the random forest model until all features have been assigned a confirmed or rejected status, providing a robust feature selection process. The Boruta algorithm offers several advantages. First, it is a model-agnostic algorithm, meaning it can be used with any machine learning model and is not biased toward a specific algorithm. Second, it can handle different types of data, such as numerical, categorical, and mixed data. Third, the algorithm can identify complex relationships between features and can handle interactions between them. Finally, it is a robust algorithm that can handle noisy data and is less likely to overfit compared to other feature selection methods [64, 66].

The Boruta algorithm has the following steps:

1. Extend the dataset by creating shadow features that mimic the original features.
2. Shuffle the values of the added duplicate features to eliminate any correlation with the response variable.

3. Train a random forest classifier on the extended dataset (including both the original and shadow features) and calculate the *Zscore*, which is the mean of accuracy loss divided by the standard deviation of accuracy loss.
4. Find the maximum *Zscore* among the shadow attributes (*MZSA*), and then tentatively tag the features as 'important' if their *Z Score* is significantly greater than *MZSA* or as 'unimportant' otherwise.
5. Repeat steps 3 and 4 until all features are tagged as important or unimportant.
6. Remove the unimportant features and retain the important ones.
7. Remove all shadow attributes.
8. Optionally, rank the important features and select a subset of top-ranked features for model training.

A MI-boruta algorithm for feature selection

In this section, we propose an MI-Boruta algorithm to select the best subset of features from D_1 and D_2. At first, the MI method is used to select $X_{MI-best}$ from the set of all possible pairs of features in the feature space, with $X_{MI-best}$ being chosen based on the value of MI between the pairs. [2] Based on the MI algorithm, 31 and 51 features were respectively selected out of 42 and 78 features of D_1 and D_2. Table 4 shows features selected by the MI algorithm. In the next step, the Boruta algorithm uses the feature importances obtained from the random forest models it trains to select the best $Y_{Boruta-best}$ subset of features. Based on the Boruta algorithm, 33 and 59 features were respectively selected out of 42 and 78 features of D_1 and D_2. Table 5 shows features selected by Boruta. Lastly, the final subset of features, $Z_{MI-Boruta-best}$, is selected by finding the intersection between $X_{MI-best}$ and $Y_{Boruta-best}$. Algorithm 1 summarizes this process. Based on this algorithm, 27 and 37 features are respectively selected out of 42 and 78 features of D_1 and D_2. Table 6 depicts important selected features.

Table 4 Feature selected through the MI approach

Dataset	Features selected
D_1	Dur, proto, state, spkts, ct_state_ttl, ct_dst_ltm, dinpkt, sjit, djit, stcpb, smean, dloss, ackdat, ct_dst_sport_ltm, ct_dst_src_ltm, sinpkt, dload, sloss, ct_src_dport_ltm, ct_srv_dst, dttl,dbytes, dmean, sload, ct_srv_src, sttl, tcprtt, synack, rate, dpkts, sbytes
D_2	Destination Port, Total Length of Fwd Packets,, Fwd Packet Length Min, Fwd Packet Length Mean, Bwd Packet Length Min,,Flow Bytes/s, Flow Packets/s,Flow IAT Mean, Flow IAT Std, Flow IAT Max, Fwd IAT Total, Flow Duration, Total Backward Packets,Fwd IAT Mean, Total Length of Bwd Packets, Fwd Packet Length Max,Fwd IAT Std, Init_ Win_bytes_backward, min_seg_size_forward,Fwd IAT Max,, Fwd Packet Length Std, Bwd Packet Length Max, Bwd IAT Total, Bwd Packet Length Mean, Bwd Packet Length Std,Bwd IAT Mean, Subflow Bwd Bytes, Init_Win_bytes_forward,Bwd IAT Max, Bwd IAT Min, Fwd Header Length, Bwd Header Length, Fwd Packets/s, Bwd Packets/s, Min Packet Length, Max Packet Length, Packet Length Mean, Packet Length Std, Packet Length Variance, Average Packet Size, Idle Max, Idle Min.,Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Header Length.1,Subflow Fwd Bytes, Subflow Bwd Packets, Active Mean,Active Max, Active Min,Idle Mean,

[2] Based on a few rounds of experiments, we used MI ≥ 0.1 for this step.

Table 5 Features selected through the boruta approach

Dataset	Features selected
D_1	Dur, proto, spkts, dinpkt, service,sjit, djit, stcpb, tcprtt, dtcpb,synack, ct_dst_ltm, swin,ct_src_dport_ltm, ct_srv_dst, trans_depth, dpkts, sbytes, dbytes, sload, dload, sloss, ct_src_ltm, rate, sttl, ct_dst_sport_ltm, ct_dst_src_ltm, is_sm_ips_ports, ackdat, smean, dmean, ct_srv_src, ct_state_ttl
D_2	Destination Port, Total Backward Packets, URG Flag Count, CWE Flag Count,Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Avg Packets/Bulk, Bwd Packet Length Mean, Flow Packets/s, Bwd URG Flags, Fwd Header Length, Bwd Header Length, Min Packet Length, Max Packet Length,Bwd Avg Bulk Rate,Packet Length Std, Packet Length Variance,FIN Flag Count, SYN Flag Count, ECE Flag Count, Down/Up Ratio, Average Packet Size, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Header Length.1, Fwd Avg Bytes/Bulk, Subflow Fwd Packets, Init_Win_bytes_backward, act_data_pkt_fwd, min_seg_size_forward, Active Mean, Active Std, Active Max, Active Min,Idle Mean, Idle Std, Flow Duration, Total Fwd Packets, Flow IAT Mean, Flow IAT Std, Fwd URG Flags, Fwd Packet Length Mean, Fwd Packet Length Std, RST Flag Count, Bwd Packet Length Max, Bwd Packet Length Min, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Subflow Bwd Packets, Subflow Bwd Bytes, Subflow Fwd Bytes,Init_Win_bytes_forward, Bwd Packet Length Std,Flow Bytes/s, PSH Flag Count, ACK Flag Count

Table 6 Features selected through the MI-boruta approach

Dataset	Quantity	Features selected by the MI-Boruta approach
D_1	**27**	Dur, Proto, Dmean, Ct_srv_dst, Ackdat, Ct_dst_src_itm, Dpkts, ct_state_ttl, Stcpb, Ct_srv_src,Sload, Ct_dst_itm,Smean, Dinpkt, Sttl, Rate, Sloss, Ct_src_dport_itm, Sjit, Sbytes, Tcprtt, Dload, Synack, Spkts, Dbytes, Ct_dst_sport_itm, djit
D_2	**37**	Destination Port,Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Std, Flow Bytes/s, Flow Packets/s, Flow IAT Mean, Bwd Header Length, Min Packet Length, Packet Length Std, Packet Length Variance, Average Packet Size, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Header Length.1, Subflow Bwd Packets, Subflow Bwd Bytes, Active Min,Init_Win_bytes_forward, Init_Win_bytes_backward, min_seg_size_forward, Active Max, Active Mean, Idle Mean, Flow Duration, Total Backward Packets, Fwd Packet Length Mean, Fwd Packet Length Std, Flow IAT Std,Fwd Header Length, Max Packet Length, Packet Length Mean, Subflow Fwd Bytes

Algorithm 1 MI-Boruta Approach for Features Selection

Input : UNSW-NB15 (D_1), CICIDS2017(D_2)

Output: The best subset of features $Z_{MI-Boruta-best}$ for D_1 and D_2

Pre-processing dataset
// Data filtration
Step1:
$D_{pre-pro2}$= Remove 'NaN' and 'Infinity'from D_2.
// Data numeralization
Step2:
If (non-numeric features) then do:
$D_{pre-pro1}$= applied 1-N numeric coding for $D_{pre-pro1}$
End if
// Data normalization scaling
Step3:
$D_{pre-pro1}$, $D_{pre-pro2}$= compute Min-MaxScaling for ($D_{pre-pro1}$, $D_{pre-pro2}$)

Step4: $X_{MI-best}$= Remove the features whose MI score less than 0.1 for $D_{pre-pro1}$ and $D_{pre-pro2}$ based on the MI method.

Step5: $Y_{RFE-best}$= Using the Boruta algorithm to select the best subset for $D_{pre-pro1}$ and $D_{pre-pro2}$

Step6: $Z_{MI-Boruta-best}$= INTERSECTION ($X_{MI-best}$, $Y_{Boruta-best}$) for $D_{pre-pro1}$ and $D_{pre-pro2}$

Ensemble learning classifier

Ensemble learning classifiers combine two or more machine learning algorithms to attain better performance in contrast to using individual algorithms. The approach of ensemble learning can be classified into three categories: boosting, bagging, and stacking. The boosting method, which was first discussed by Schapire [67] in 1990, involves intensive training of models on misclassified data during the training phase. Eventually, the model with the highest accuracy is selected as the classifier for the test data. The bagging approach, developed in 1996 by Breiman [68], involves using homogeneous models to predict the class of test data. Initially, the predictions of the selected homogenous models are recorded. Then, the class that is predicted by the majority of models is assigned to the test data. Wolpert [69] introduced the stacking approach, also known as *stacked generalization*, in 1992. This method is considered an effective strategy as it provides a general framework to combine numerous ensemble algorithms. It involves two levels of learning, where the initial (base) learners are trained using a training dataset in level 0, and meta-learning takes place in level 1. The base learners generate a new dataset for the meta-learner, and this new dataset is used to train the meta-learner. The trained meta-learner is then used to classify the test set. Unlike boosting and bagging ensembles that typically use the same type of model, stacked ensemble can integrate different types of models. Therefore, selecting the optimal base learner is a crucial part of the stacking approach, and several base learners should be selected for the training dataset instead of a single one.

Overall, stacked ensemble learning provides several advantages over bagging and boosting methods:

- Improved predictive accuracy: Stacked ensembles can achieve higher predictive accuracy than individual models or other ensemble techniques, such as bagging and boosting. This is due to the combination of diverse base models in the stacked ensemble, which can capture different aspects of the data and reduce bias and variance [69].
- Better robustness: Stacked ensemble learning can reduce the risk of overfitting and increase the robustness of the model by incorporating a meta-learner that combines the outputs of the base models. The meta-learner can identify and correct errors made by the base models, leading to better generalization performance [70].
- Flexibility: Unlike bagging and boosting, which typically use the same type of model, stacked ensembles can integrate different types of models, including both linear and non-linear models. This allows for more flexibility in model selection and can improve performance on complex datasets [71].

To leverage the advantages of stacked ensembles, we will employ them in this paper by using random forest, CatBoost, and XGBoost algorithms as base learners and multilayer perceptron (MLP) as the meta-learner. As we discuss later in the paper, using stacked ensembles enables us to reduce the variance of prediction errors, as well as to improve robustness by minimizing prediction dispersion [72].

Random forest algorithm

Breiman's random forest (RF) [73] is an ensemble-based classification method that constructs multiple classification trees by drawing bootstrap samples from the original data. Each tree then casts a vote on how to categorize a new item. The final decision is made based on the majority vote. RF offers several advantages, such as the ability to predict important variables, low classification errors, the capability to handle imbalanced datasets, efficient handling of big data sets with thousands of variables, addressing overfitting, and maintaining accurate classification across various datasets. These advantages make RF a powerful and versatile classification method.

XGBoost algorithm

eXtreme Gradient Boosting (XGBoost) [74] is an ensemble learning method that is based on the gradient boosting decision tree (GBDT) algorithm. The Xgboost supports linear classifiers and uses traditional CART (Classification And Regression Trees) as the base classifier. It enhances prediction accuracy by utilizing Taylor expansion for the cost function and incorporating second derivatives. The XGBoost algorithm utilizes an additive training technique to optimize the objective function, where the optimization process of each subsequent step relies on the results of the previous step. The principles of the XGBoost algorithm are as follows:

1. XGBoost employs an additive training technique to optimize the objective function, which means that the optimization process of latter steps relies on the optimization result of the previous steps. The t^{th} objective function of the model is expressed by the following equation:

$$obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{t-1}\right) + f_t(x_i)) + \Omega(f_t) + C \tag{4}$$

where l represents the loss term of the t^{th} round, Ω is the regularization term, and C represents a constant term. The value of $\Omega(f_t)$ is shown in Eq. 5

$$\Omega(f_t) = \gamma.T_t + \lambda\frac{1}{2}\sum_{j=1}^{T} w_j^2 \tag{5}$$

where both γ and λ are customization parameters of XGBoost. In general, increasing the values of these two parameters simplifies the tree structure, effectively addressing the problem of overfitting.

2. Perform a second-order Taylor expansion on Eq. 4 This process is shown in Eq. 6where g is the first derivative, and h is the second derivative. g_i and h_i can be described as the following:

$$obj^{(t)} = \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{t-1}\right) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2 + \Omega(f_t) + C\right] \tag{6}$$

$$g_i = \partial_{\hat{y}_i^{t-1}} l\left(y_i, \hat{y}_i^{t-1}\right) \tag{7}$$

$$h_i = \partial^2_{\hat{y}_i^{t-1}} l\left(y_i, \hat{y}_i^{t-1}\right) \tag{8}$$

3. Substitute (5), (7), and (8) into (6) and take the derivative. Then, solutions can be obtained from (9) and (10) as follows:

$$w_j^* = -\frac{\sum g_i}{\sum h_i + \lambda} \tag{9}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T} \frac{\left(\sum g_i\right)^2}{\sum h_i + \lambda} + \gamma.\,\mathrm{T} \tag{10}$$

In these equations, w_j^* refers to the weights solution, and obj^* represents the score of the loss function. The smaller this score, the better the structure of the tree.

Categorical boosting (Catboost) algorithm

Catboost is a machine learning classifier based on the gradient boosting decision tree framework and is available as an open-source library [75]. This classifier offers several advantages, including (1) the ability to handle categorical data using statistical methods, (2) optimization of extensive input parameters to reduce overfitting, (3) reduction of the loss function in each iteration to improve model generalization, (4) faster performance than other boosting algorithms thanks to the implementation of symmetric trees, (5) only requiring a small number of parameters, and (6) suitability for large datasets with low latency requirements.

Multilayer perceptron algorithm

MLP is a feed-forward artificial neural network that is constructed with three or more layers. The first layer is the input layer, the last layer is the output layer, and the middle layer has one or more hidden layers. Each layer contains a set of neurons or nodes. MLP learns a function $f(x) : R^m \rightarrow R^o$ by training on a dataset using the backpropagation learning method. Here, m represents the number of input dimensions, and o represents the number of output dimensions. In an MLP, each layer can be characterized by the following equivalent equations:

$$y = \phi\left(\sum_{i=1}^{n} W_i X + b\right) \tag{11}$$

$$y = \phi\left(W^T X + b\right) \tag{12}$$

In the above equations, ϕ is the activation function, W is the weight vector, X is the input vector, and b is the bias. For this study, we used MLP as the meta-learner, consisting of one hidden layer with 200 neurons

Table 7 Experimental model parameters

Model	Parameters
RF	n_estimators = 200, criterion = "entropy", max_depth = 60
XGBoost	n_estimators = 600, max_depth = 31, objective = "binary:logistic" for binary classification and
Catboost	objective = "multi:softprob" for multi-classification
MLP	n_estimators = 500, loss_function = "Logloss" for binary classification and
	loss_function = "MultiClass" for multi-classification
	one hidden_layer_sizes with 200 neurons

Stacked ensemble classifier

This section describes a proposed stacking ensemble classifier consisting of base classifiers (level 0) and meta-classifiers (level 1). The input training dataset (D) for the base classifiers is composed of selected features from MI-Boruta. The predictions from level 0 serve as input for level 1. Overfitting issues during training are addressed by implementing a k-fold cross-validation method. The training data (D) is partitioned into k disjoint subsets of equal size $(D_1, D_2, D_3, \ldots, D_n)$, with one of the k-subsets used for testing and the remaining subsets $(k-1/k)$ used for training the classifiers. In this study, the base classifiers (level 0) include RF, XGB, and Catboost, and are trained using five-fold cross-validation. The meta-classifier (level 1) utilizes the MLP algorithm to combine the results of the three base classifiers and generate the final predicted output. Table 7 shows the parameters of the stacked ensemble model. Algorithm 2 outlines the general algorithm for stacking ensemble classifiers.

Algorithm 2 Stacked Ensemble Classifier

Input: Train data $D = \{x_i, y_i\}_{(i=1)}$; $(x_i \epsilon R^n,\ \ y_i \epsilon\ y)$

Output: Predictions from the stacked ensemble classifier (H)

Step 1: Apply five-fold cross-validation in preparing the training set for the classifier
Step 2: Randomly Split D into $K = 5$ equally sized subsets, $D = (D_1, D_2, D_3, D_4, D_5)$.
Step 3: **for** $k \leftarrow 1\ to\ K$ **do** where $K = 5$
  Employ base level classifiers (RF, XGB, and Catboost) (level 0)
  for $t \leftarrow 1\ to\ T$ **do**
    Learn a classifier h_{kt} from $D \backslash D_k$
  end for
  Construct a training set for meta classifier MLP
  for $x_i \in D_k$ **do**
    Get record $\{\acute{x}_i, y_i\}$, where $\acute{x}_i = \{h_{k1}(x_i), h_{k2}(x_i), \ldots, h_{kT}(x_i)\}$
  end for
 end for
Step 4: Learn meta-classifier MLP (level 1)
  Learn a new classifier $\acute{h}$ from the collection of $\{\acute{x}_i, y_i\}$
Step 5: Re-learn (level 0) classifiers
  for $t \leftarrow 1\ to\ T$ **do**
    Learn a classifier (h_t) based on D
  end for
Step 6: Return $H(x) = \acute{h}\big(h_1(x), h_2(x), \ldots,\ h_T(x)\big)$

Performance evaluation

In this section, we evaluate our proposed model using different performance metrics, such as accuracy, recall, precision, F_1 score, and the AUCROC curve. These metrics are derived from a confusion matrix, which is a two-dimensional table that compares the predicted and

actual classes and distinguishes the outcomes of the classification. The confusion matrix is based on the following four values:

- True Negative (TN): both the original data as well as the predicted data are false.
- True Positive (TP): both the original data as well as the predicted data are true.
- False Negative (FN): original data are true, but the predicted data are false.
- False Positive (FP): original data are false, but the predicted data are true.

Accuracy is the ratio of instances that have been correctly classified to the total number of instances. This can be defined as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

Recall, also referred to as the detection rate (DR), true positive rate (TPR), or sensitivity, is the percentage of total true positive (TP) cases divided by the total number of true positive (TP) and false positive (FN) cases. It is calculated as illustrated in the following formula:

$$Recall = DR = TPR = \frac{TP}{TP + FN} \tag{14}$$

Precision is the percentage of total true positive (TP) cases divided by the total number of true positive (TP) and false positive (FP) cases. The following formula represents precision:

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

F_1 score is the harmonic mean of recall and precision defined by the following equation:

$$F_1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{16}$$

The False Positive Rate (FPR) shows the percentage of normal traffic detected as an attack. The FPR is calculated as:

$$FPR = \frac{FP}{FP + TN} \tag{17}$$

The False Negative Rate (FNR) is the percentage of attacks detected as normal traffic. FNR is calculated as:

$$FNR = \frac{FN}{TP + FN} \tag{18}$$

Error Rate (ER) is calculated using the following equation:

$$\text{ER} = \frac{FP + FN}{TP + TN + FP + FN} \tag{19}$$

Finally, the Area Under the Curve (AUC) is a parameter used for evaluating the performance of a machine learning model at all categorization thresholds. It is characterized by the false positive and true positive rates on the X and Y axes, respectively. When the

curve is close to the top left corner, the model performs better. Conversely, the model performs poorly if the curve is closer to the baseline. AUC is obtained using the following formula:

$$AUC = \int_0^1 TPRd(FPR) \tag{20}$$

Experimental results and discussion

In this section, we evaluate the performance of our proposed stacked ensemble classifier with various individual classification algorithms, as well as with state-of-the-art methods. It deserves to mention that model development and evaluation were performed using Python on a machine with the following specifications:

- Operating system: Windows 11 (64-bit)
- CPU: Ryzen7 5800 h
- RAM: 32 GB
- HDD: 1 TB SSD
- VGA: RTX3070 8 GB

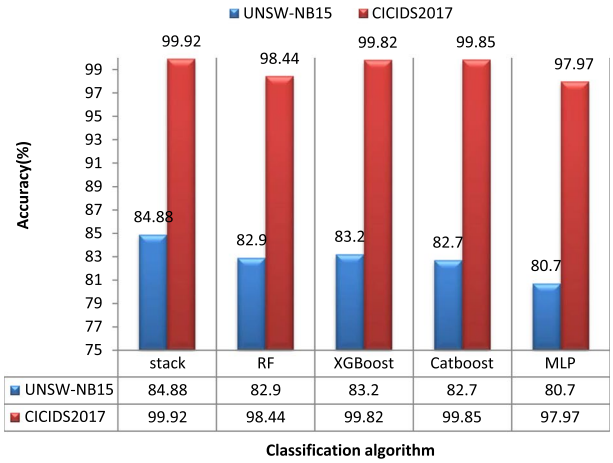
Comparison with individual classifiers

We assessed the performance of our stack ensemble classifier by comparing it to several individual classification algorithms (such as RF, XGBoost, Catboost, and MLP) using the same MI-Boruta feature selection technique. Our study employed a multi-classification approach and evaluated the accuracy and F_1 score using two confusion matrices. Figure 4 provides a summary of the model's performance relative to the individual classification algorithms. According to Fig. 4a, our stack ensemble classifier outperformed the individual algorithms on both the UNSW-BN15 and CICIDS2017 datasets, with an accuracy of 84.88 and 99.92% for D_1 and D_2, respectively. Similarly, as shown in Fig. 4b, our model achieved a higher F_1-score than the individual classification algorithms on both datasets, with scores of 84.15 and 99.92% for D_1 and D_2.

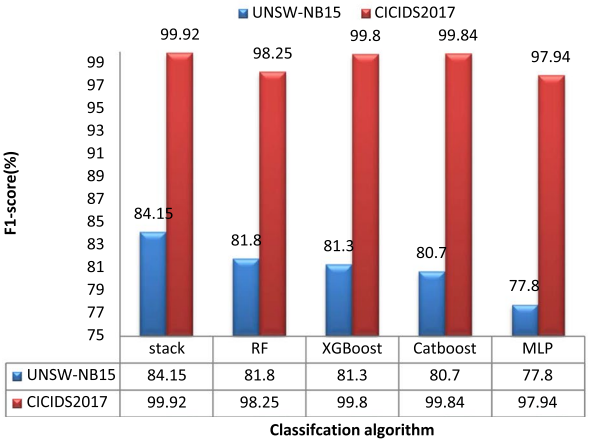
We recorded the computation time of each base classifier and the stacked ensemble for UNSWNB-15 and CICIDS2017 datasets, which have different sizes. The computation times of UNSWNB-15 were 0.14 s, 0.24 s, 0.28 s, 0.03 s, and 0.68 s for RF, Catboost, XGBoost, MLP, and stacked ensemble respectively. The computation times of CICIDS2017 were 1.21 s, 3.29 s, 2.44 s, 0.43 s, and 7.21 s for RF, Catboost, XGBoost, MLP, and stacked ensemble respectively. As expected, the stacked ensemble takes more time than individual classification models in both datasets.

Comparison with the state-of-the-art methods

In this section, we present a comparative analysis of our proposed model's performance with earlier research on binary and multi-class classification using D_1 and D_2. The evaluation metrics are presented in Tables 8, 9, 10. Table 8 shows the binary classification performance of our proposed model and other recent IDS studies for D_1. Similarly,



(a)



(b)

Fig. 4 Performance comparison of classifiers for the two datasets

Table 9 shows the performance of our model and recent IDS studies for binary classification using D_2. Table 10 compares the multi-class classification performance of our model with that of recent IDS studies for both datasets.

According to the information presented in these tables, our proposed model has a better overall performance than prior studies. For binary classification of D_1, our model's accuracy is 95.34%, recall is 94.64%, precision is 92.62%, F_1 score is 93.62%, FPR is 4.2%, and the AUC is 95.19% (Fig. 5a). For multi-class classification of D_1 using a subset of 27 features (which denotes a simpler model), the accuracy is 84.88%, recall is 84.88%, precision is 84.93%, and F_1 score is 84.15%. Table 11 presents the performance results of the proposed model for each class of D_1. For D_2, the accuracy for binary classification is 99.925%, recall is 99.926%, precision is 99.979%, F_1 score is 99.953%, FPR is 0.08%, and the AUC is 99.922% (Fig. 5b). For multi-class classification of D_2 using a subset of 37 features, the accuracy is 99.92%, recall is 99.92%, precision is 99.92%, and F_1 score is 99.92%. Table 12 presents the performance results of the proposed model for each class of D_2.The confusion matrices for binary and multi-class classification of both datasets are shown in Figs. 6 and 7.

Table 8 Comparison of evaluation metrics between our proposed model and recent IDS studies for binary classification on the UNSW-NB15 dataset (best values are denoted in bold)

Ref	Feature selection method	Classification method	Accuracy (%)	Recall (%)	Precision (%)	F_1 score (%)	FPR (%)	FNR (%)	ER (%)	AUC
[35]	PIO	DT	91.3	89.7	N/A	90.4	5.2	N/A	N/A	N/A
[24]	IG	MLP	76.96	77	79.8	N/A	20.6	N/A	23.95	N/A
[46]	LightGBM	DNN	88.34	87.54	N/A	88.14	12.46	N/A	N/A	N/A
[76]	GTO-BSA	KNN	71.01	81.54	N/A	N/A	N/A	N/A	N/A	N/A
[77]	UMAP	RF	92.6	91	N/A	90.5	N/A	N/A	N/A	85
[78]	N/A	Bagging-GBM	94.66	92.94	92.21	92.6	4.38	N/A	N/A	**99.1**
[51]	Selects *k* best features according to the highest scores	Stacking ensemble (DT, RF, XGoost)	94	94	N/A	N/A	6	N/A	N/A	96
[79]	N/A	Stack ensemble (DNN, GBM)	92.83	N/A	N/A	N/A	8.91	N/A	N/A	N/A
This study	**MI-Boruta**	**Stacked ensemble (RF, XGBoost, Catboost, MLP)**	**95.346**	**94.645**	**92.623**	**93.623**	**4.2**	**5.3**	**4.8**	95.1

Table 9 Comparison of evaluation metrics between our proposed model and recent IDS studies for binary classification on the CICIDS2017 dataset (best values are denoted in bold)

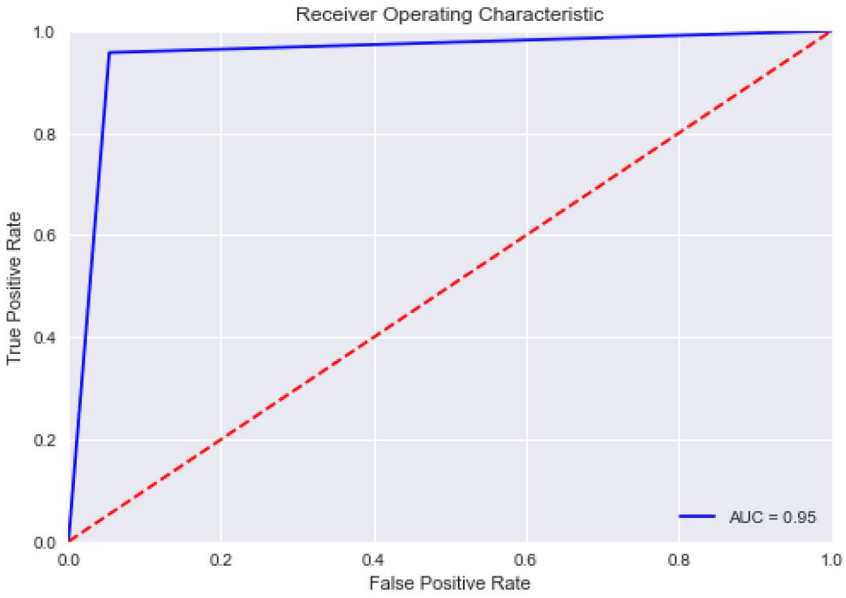
Ref	Feature selection method	Classification method	Accuracy (%)	Recall (%)	Precision (%)	F_1 Score (%)	FPR (%)	FNR (%)	ER (%)	AUC
[30]	RF	Hard voting (NB, LR, DT)	88.92	N/A	N/A	N/A	N/A	N/A	N/A	N/A
[76]	GTO-BSA	KNN	98.7915	97.2644	N/A	N/A	N/A	N/A	N/A	N/A
[53]	Gradient Boost Trees with meta-transformer	XGBoost	99.86	99.75	99.69	99.72	0.2	0.17	0.14	99.72
[80]	Feature fusion technique based on gradient importance enhancement	ResNet-18	99.78	99.79	99.82	99.80	N/A	N/A	N/A	N/A
[81]	Gain ratio and Pearson's correlation	RF	99.8302	99.8839	99.9044	99.8942	0.3872	N/A	N/A	N/A
[82]	N/A	Stacking (MLP, LSTM, DNN, CNN)	98.7	97.6	95.9	96.7	1	N/A	N/A	99.9
[79]	N/A	Stack (DNN, GBM)	99.65	N/A	N/A	N/A	1.67	N/A	N/A	N/A
This study	**MI-Boruta**	**Stacked ensemble (RF, XGboost, Catboost, MLP)**	**99.925**	**99.926**	**99.979**	**99.953**	**0.08**	**0.07**	**0.07**	**99.922**

Table 10 Comparison of evaluation metrics between our proposed model and recent IDS studies for multi-classification for both datasets (best values are denoted in bold)

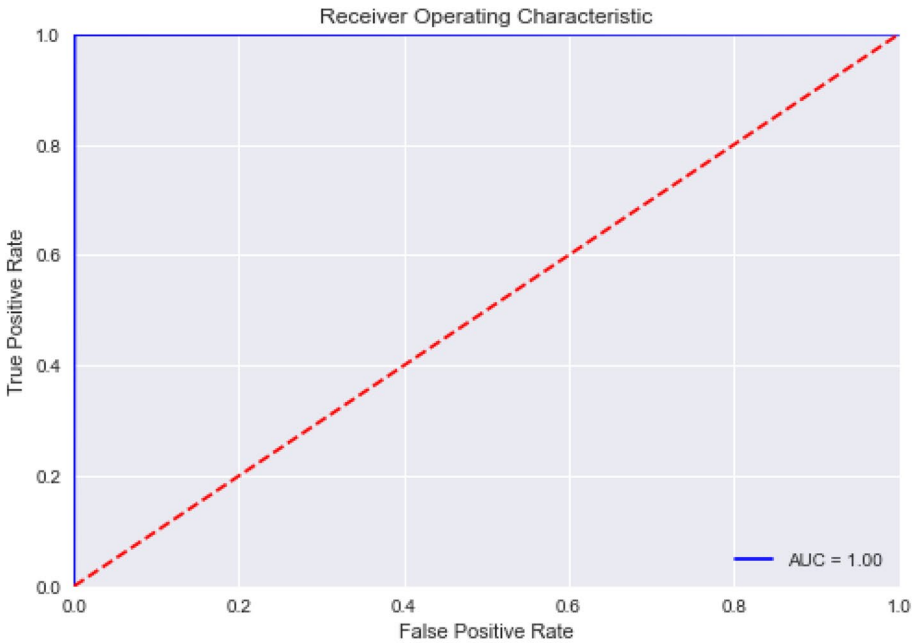
Ref	Dataset	Feature selection method	Classification method	Accuracy (%)	Recall (%)	Precision (%)	F_1 Score (%)
[50]	D_1	TS	RF	83.12	N/A	N/A	N/A
[47]	D_1	IGRF-RFE	MLP	84.24	0.8424	0.8360	0.8285
[83]	D_1	NRS-SSA	Adaptive ensemble (DT, KNN, RF, XGBoost)	81.54	81.54	82.7	79.7
[77]	D_1	UMAP	RF	81.6	80	N/A	80
[84]	D_1	N/A	Ensemble (Group convolution network snapshot ensmble)	79.03	80.01	81.21	80.61
This study	**D_1**	**MI-Boruta**	**Stacked ensemble (RF, XGboost, Catboost, MLP)**	**84.88**	**84.88**	**84.93**	**84.15**
[30]	D_2	RF	Hard voting (NB, LR, DT)	88.96	N/A	N/A	N/A
[85]	D_2	Boruta	ET	99.8	99.8	N/A	N/A
[53]	D_2	Gradient Boost Trees with meta-transformer	XGBoost	99.90	99.90	99.90	99.90
[86]	D_2	Chi-squared	Triple layered hybrid (weighted DNN, CNN + LSTM, XGBoost)	98.46	98.21	86.42	98.1
[82]	D_2	N/A	Stacking (MLP, LSTM, DNN, CNN)	98.7	98.7	98.7	98.6
This study	**D_2**	**MI-Boruta**	**Stacked ensemble (RF, XGboost, Catboost, MLP)**	**99.92**	**99.92**	**99.92**	**99.92**

Conclusion

With the widespread use of the Internet, network security has become a crucial aspect in distributed systems. While previous studies have used different machine learning algorithms to improve the performance of intrusion detection systems (IDS), they still face several challenges in achieving optimal performance. To address this, we presented an effective IDS model that utilizes a hybrid feature selection method and an ensemble learning technique on two well-known intrusion detection benchmark datasets. Our approach started by proposing an MI-Boruta algorithm to select the best features from the datasets. Then, we used a stacked ensemble classifier, comprising random forest, XGBoost, and Catboost as base learners and multilayer



(a) D_1



(b) D_2

Fig. 5 The Area under the ROC Curve for D_1 and D_2

perceptron as the meta-learner, for classification. Our experimental results indicated excellent performance in binary and multi-class classification for both datasets. Specifically, the accuracy of binary classification was 95.34%, with 94.64% recall, 92.62%

Table 11 Performance results of D_1 for multiclass classification

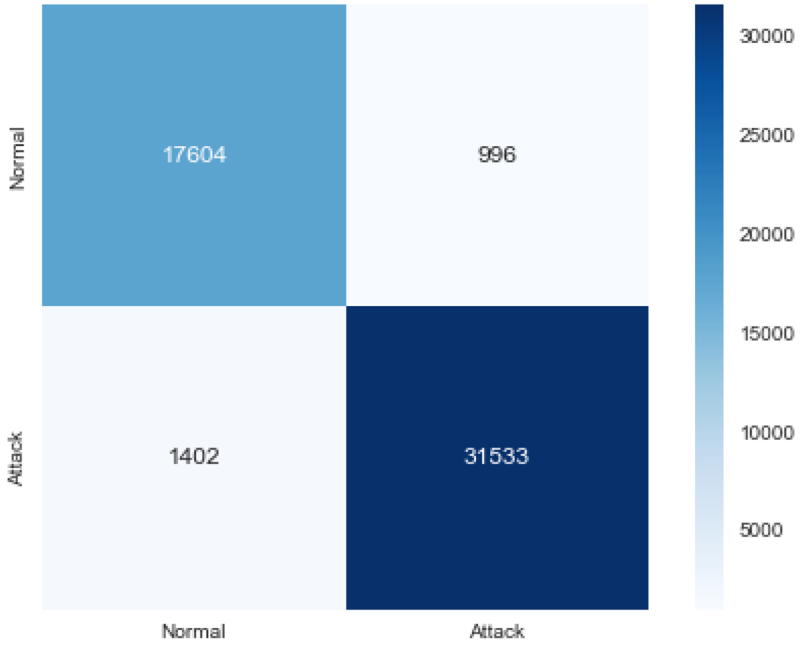
Attack type	Precision	Recall	F1 score
Normal	93.19	93.45	93.19
Reconnaissance	92.27	75.51	83.05
Fuzzers	72.26	67.82	69.97
Analysis	76.63	15.32	25.54
Exploits	66.74	88.27	76.01
Backdoor	49.6	26.60	24.63
DoS	56.12	32.34	41.03
Generic	99.42	98.42	98.92
Shellcode	66.56	73.84	70.01
Worms	71.87	65.71	68.65

Table 12 Performance results of D_2 for multiclass classification

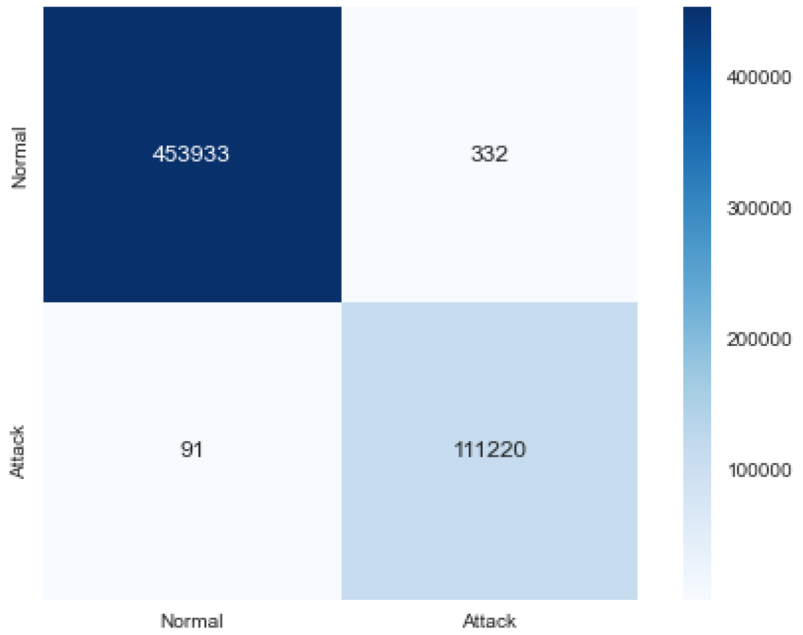
Attack type	Precision	Recall	F1 score
BENIGN	99.97	99.93	99.95
DDoS	99.97	99.96	99.97
FTP-Patator	100	99.93	99.96
DoS	99.83	99.96	99.89
Bot	98.63	73.91	84.5
Heartbleed	100	100	100
Infiltration	100	100	100
PortScan	99.34	99.98	99.66
SSH-Patator	100	100	100
Web Attack	99.06	97.47	98.26

precision, 93.62% F_1 score, and a 4.2% false positive rate. For multi-class classification, the accuracy was 84.88%, with 84.88% recall, 84.93% precision, and 84.15% F_1 score, using a subset of 27 features from the UNSW-NB15 dataset. In particular, our model achieved the highest accuracy for binary and multi-class classification on CIC-IDS2017, with a binary classification accuracy of 99.92%, 99.92% recall, 99.97% precision, 99.95% F_1 score, and 0.08% FPR. For multi-class classification, the accuracy was 99.92%, with 99.92% recall, 99.92% precision, and 99.92% F_1 score, using a subset of 37 features from D_2. Furthermore, our proposed stacked ensemble classifier outperformed individual classification methods in terms of accuracy and F_1 score. Finally, comparisons with state-of-the-art methods demonstrated that our model provides a competitive advantage in the intrusion detection domain.

Our study has several implications for future research in the field of intrusion detection. First, the proposed hybrid feature selection method and stacked ensemble classifier have demonstrated good results in detecting intrusions in widely used datasets



(a) D_1



(b) D_2

Fig. 6 Confusion matrices for binary classification of D_1 and D_2

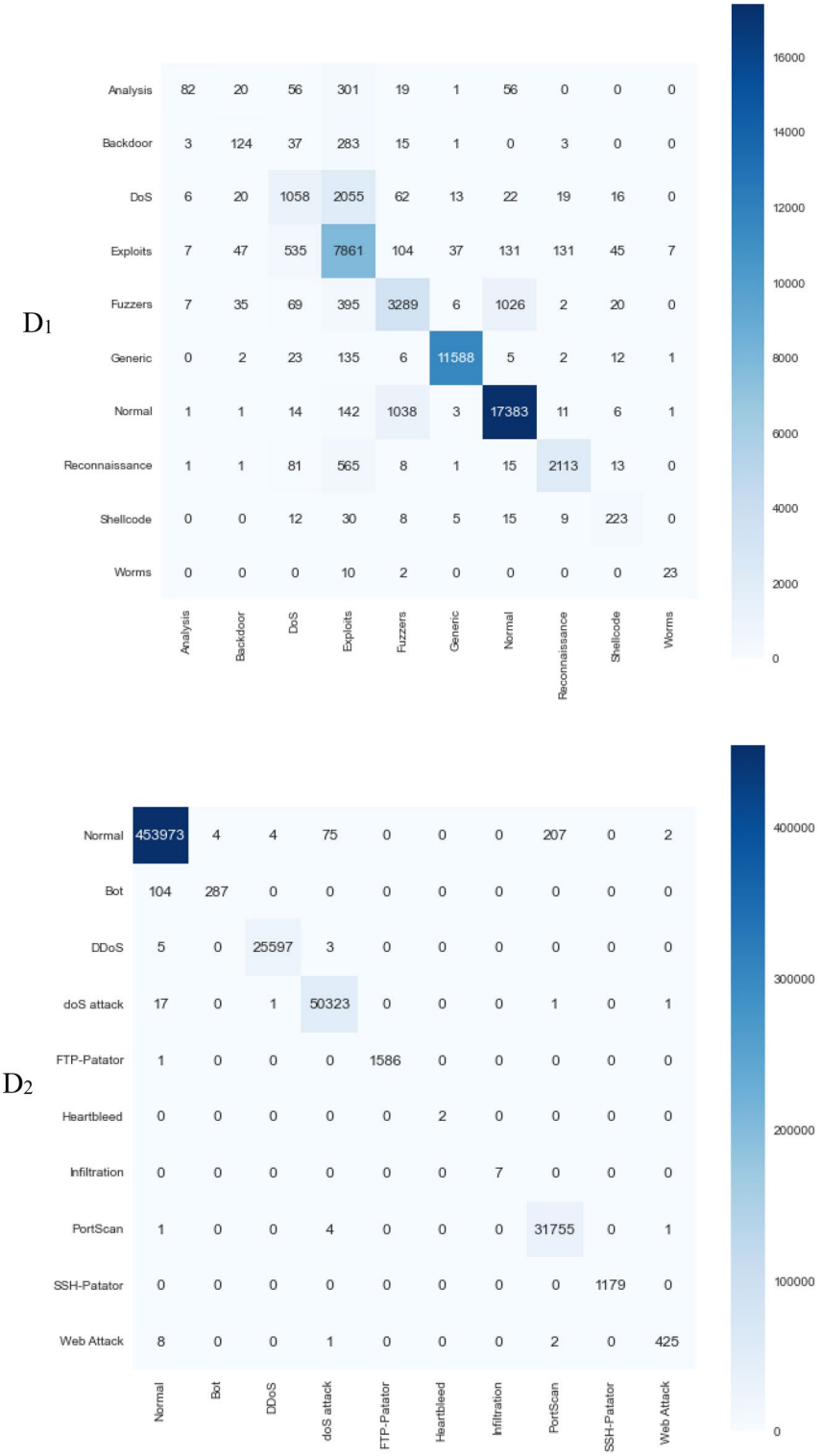


Fig. 7 Confusion matrix for multi-class classification of D_1 and D_2

such as UNSW-NB15 and CICIDS017. Therefore, further research could explore the effectiveness of these techniques in other datasets to further evaluate their generalizability. Second, while the proposed model outperformed state-of-the-art methods in terms of accuracy and F_1 score, there is still room for improvement in terms of reducing false positives and false negatives. Future research could investigate how to further improve the model's performance by fine-tuning its parameters or developing new algorithms to address these issues. Third, the proposed model used a combination of RF, XGBoost, and Catboost as base learners with MLP as a meta-learner. Future research could explore the effectiveness of other machine learning algorithms as base or meta-learners to enhance the model's performance. Finally, while the proposed model achieved good results in both binary and multi-classification, it would be interesting to investigate how to improve the performance in specific types of attacks or in more complex scenarios such as detecting attacks in real-time or in highly dynamic environments.

One limitation of our study is its lack of explainability. The model's decisions and predictions cannot be easily interpreted and understood, making it challenging to identify the root causes of errors or biases. This may limit the paper's practical applications in sensitive domains where interpretability is essential, such as healthcare or finance.

Abbreviations

RF	Random forest
BN	Bayes net
RT	Random tree
DT	Decision tree
LR	Logistic regression
KNN	K-nearest neighbor
SVM	Support vector machine
ELM	Extreme learning machine
XGBoost	EXtreme gradient boosting
Adaboost	Adaptive boosting
GBM	Gradient boosting machine
LightGBM	Light gradient boosting machine
RT	Random tree
DS	Decision stump
ET	Extra tree
IG	Information gain
PIO	Pigeon-inspired optimizer
PSO	Particle swarm optimization
BOA	Bat optimization algorithm
GWO	Grey wolf optimization
RFE	Recursive feature elimination
CS	Cuckoo search
SFS	Sequential forward selection
GTO	Gorilla troops optimizer
BSA	Bird swarms algorithm
UMAP	Uniform manifold approximation and projection
NRS	Neighborhood rough set
SSA	Salp swarm algorithm
TS	Tabu search
RNN	Recurrent neural network
LSTM	Long short-term memory
MLP	Multilayer perceptron
ANN	Artificial neural network
CNN	Convolutional neural network
DNN	Deep neural network
AE	Autoencoder
DAE	Denoising autoencoder
VAE	Variational autoencoder
DNN	Deep neural network

GRU	Gated recurrent units
Bi-LSTM	Bidirectional long-short term memory
BPNN	Back propagation neural network
RBFN	Radial basis function network
GBM	Gradient boosting machine

Author contributions
A.M.A. and H.M.Z wrote the manuscript. A.M.A. was the lead in data preprocessing, model building, and evaluation. M.N.B. supervised the study and was a major contributor in model building and evaluation. All authors read and approved the final manuscript.

Data availability
Not applicable.

Declarations

Competing interests
The authors declare no competing interests.

Received: 24 August 2023 Accepted: 27 August 2024
Published online: 18 September 2024

References

1. Leszczyna R, Wallis T, Wróbel MR. Developing novel solutions to realise the European energy—information sharing & analysis centre. Decis Support Syst. 2018;122:2019. https://doi.org/10.1016/j.dss.2019.05.007.
2. Zhang H, Chari K, Agrawal M. Decision support for the optimal allocation of security controls. Decis Support Syst. 2018;115:92–104. https://doi.org/10.1016/j.dss.2018.10.001.
3. Zadeh A, Jeyaraj A. A multistate modeling approach for organizational cybersecurity exploration and exploitation. Decis Support Syst. 2022;162(August):113849. https://doi.org/10.1016/j.dss.2022.113849.
4. Li S, Iqbal M, Saxena N. Future industry internet of things with zero-trust security. Inf Syst Front. 2022. https://doi.org/10.1007/s10796-021-10199-5.
5. Ge M, Syed NF, Fu X, Baig Z, Robles-Kelly A. Towards a deep learning-driven intrusion detection approach for internet of things. Comput Netw. 2021. https://doi.org/10.1016/j.comnet.2020.107784.
6. Derhab A, Belaoued M, Mohiuddin I, Kurniawan F, Khan MK. Histogram-based intrusion detection and filtering framework for secure and safe in-vehicle networks. IEEE Trans Intell Transp Syst. 2022;23(3):2366–79. https://doi.org/10.1109/TITS.2021.3088998.
7. Safaldin M, Otair M, Abualigah L. Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. J Ambient Intell Humaniz Comput. 2021;12(2):1559–76. https://doi.org/10.1007/s12652-020-02228-z.
8. Kumar P, Dwivedi YK, Anand A. Responsible artificial intelligence (AI) for value formation and market performance in healthcare: the mediating role of patient ' s cognitive engagement. Inf Syst Front. 2021. https://doi.org/10.1007/s10796-021-10136-6.
9. McLeod A, Dolezel D. Cyber-analytics: modeling factors associated with healthcare data breaches. Decis Support Syst. 2018;108:57–68. https://doi.org/10.1016/j.dss.2018.02.007.
10. Khammassi C, Krichen S. A GA-LR wrapper approach for feature selection in network intrusion detection. Comput Secur. 2017;70:255–77. https://doi.org/10.1016/j.cose.2017.06.005.
11. Elhag S, Fernández A, Bawakid A, Alshomrani S, Herrera F. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. Expert Syst Appl. 2015;42(1):193–202. https://doi.org/10.1016/j.eswa.2014.08.002.
12. Liang W, Xiao L, Zhang K, Tang M, He D, Li KC. Data fusion approach for collaborative anomaly intrusion detection in blockchain-based systems. IEEE Internet Things J. 2022;9(16):14741–51. https://doi.org/10.1109/JIOT.2021.3053842.
13. Jiang K, Wang W, Wang A, Wu H. Network intrusion detection combined hybrid sampling with deep hierarchical network. IEEE Access. 2020;8(3):32464–76. https://doi.org/10.1109/ACCESS.2020.2973730.
14. Mukhopadhyay A, Chatterjee S, Bagchi KK, Kirs PJ, Shukla GK. Cyber risk assessment and mitigation (CRAM) framework using logit and probit models for cyber insurance. Inf Syst Front. 2019;21(5):997–1018. https://doi.org/10.1007/s10796-017-9808-5.
15. Tchernykh A, Schwiegelsohn U, Ghazali Talbi E, Babenko M. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. J Comput Sci. 2019. https://doi.org/10.1016/j.jocs.2016.11.011.
16. Stampar M, Fertalj K. "Artificial intelligence in network intrusion detection. 2015 38th Int Conv Inf Commun Technol Electron Microelectron. 2015. https://doi.org/10.1109/MIPRO.2015.7160479.
17. Muhuri PS, Chatterjee P, Yuan X, Roy K, Esterline A. Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks. Inf. 2020;11(5):1–21. https://doi.org/10.3390/INFO11050243.
18. Wan J, et al. An efficient impersonation attack detectionmethod in fog computing. Comput Mater Contin. 2021;68(1):268–81. https://doi.org/10.32604/cmc.2021.016260.
19. Pranto MB, Ratul MHA, Rahman MM, Diya IJ, Bin Zahir Z. Performance of machine learning techniques in anomaly detection with basic feature selection strategy-a network intrusion detection system. J Adv Inf Technol. 2022;13(1):36–44. https://doi.org/10.1272/jait.13.1.36-44.

20. Ozkan-Okay M, Samet R, Aslan O, Gupta D. A comprehensive systematic literature review on intrusion detection systems. IEEE Access. 2021;9:157727–60. https://doi.org/10.1109/ACCESS.2021.3129336.
21. Wu C, Li W. Enhancing intrusion detection with feature selection and neural network. Int J Intell Syst. 2021;36(7):3087–105. https://doi.org/10.1002/int.22397.
22. Iwendi C, Khan S, Anajemba JH, Mittal M, Alenezi M, Alazab M. The use of ensemble models for multiple class and binary class classification for improving intrusion detection systems. Sensors. 2020;20(9):1–37. https://doi.org/10.3390/s20092559.
23. Kurniabudi DS, Darmawijoyo MY, Idris BB, Bamhdi AM, Budiarto R. CICIDS-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access. 2020;8:132911–21. https://doi.org/10.1109/ACCESS.2020.3009843.
24. Mebawondu JO, Alowolodu OD, Mebawondu JO, Adetunmbi AO. Network intrusion detection system using supervised learning paradigm. Sci Afr. 2020. https://doi.org/10.1016/j.sciaf.2020.e00497.
25. Mebawondu OJ, Popoola OS, Ayogu II, Ugwu CC, Adetunmbi AO. Network intrusion detection models based on naives bayes and c4.5 algorithms. Proc 2022 IEEE Niger 4th Int Conf Disruptive Technol Sustain Dev NIGERCON 2022. 2022. https://doi.org/10.1109/NIGERCON54645.2022.9803086.
26. Ahsan M, Gomes R, Chowdhury MM, Nygard KE. Enhancing machine learning prediction in cybersecurity using dynamic feature selector. J Cybersecurity Priv. 2021;1(1):199–218. https://doi.org/10.3390/jcp1010011.
27. Rashid MM, Kamruzzaman J, Ahmed M, Islam N, Wibowo S, Gordon S. Performance enhancement of intrusion detection system using bagging ensemble technique with feature selection. 2020 IEEE Asia-Pac Conf Comput Sci Data Eng. 2020. https://doi.org/10.1109/CSDE50874.2020.9411608.
28. Zheng X, Wang Y, Jia L, Xiong D, Qiang J. Network intrusion detection model based on Chi-square test and stacking approach. 2020 7th Int Conf Inf Sci Control Eng. 2020. https://doi.org/10.1109/ICISCE50968.2020.00185.
29. Oriola O. A stacked generalization ensemble approach for improved intrusion detection. Int J Comput Sci Inf Secur. 2020;18(5):62–7.
30. Abbas A, Khan MA, Latif S, Ajaz M, Shah AA, Ahmad J. A new ensemble-based intrusion detection system for internet of things. Arab J Sci Eng. 2021;47(2):1805–19. https://doi.org/10.1007/s13369-021-06086-5.
31. Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res. 2003;3:1157–82. https://doi.org/10.1016/j.aca.2011.07.027.
32. Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng. 2014;40(1):16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024.
33. Mirlashari M, Rizvi SAM. Feature selection technique-based network intrusion system using machine learning. 2020 IEEE World Conf Appl Intell Comput. 2022. https://doi.org/10.1109/AIC55036.2022.9848861.
34. Sharma NV, Yadav NS. An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers. Microprocess Microsyst. 2021;85:104293. https://doi.org/10.1016/j.micpro.2021.104293.
35. Alazzam H, Sharieh A, Sabri KE. A feature selection algorithm for intrusion detection system based on Pigeon inspired optimizer. Expert Syst Appl. 2020. https://doi.org/10.1016/j.eswa.2020.113249.
36. Zhang L, Xu C. A intrusion detection model based on convolutional neural network and feature selection. 2022 5th Int Conf Artif Intell Big Data. 2022. https://doi.org/10.1109/ICAIBD55127.2022.9820384.
37. Imran M, Khan S, Hlavacs H, Khan FA, Anwar S. Intrusion detection in networks using cuckoo search optimization. Soft Comput. 2022;26(20):10651–63. https://doi.org/10.1007/s00500-022-06798-2.
38. Ogundokun RO, Awotunde JB, Sadiku P, Adeniyi EA, Abiodun M, Dauda OI. An enhanced intrusion detection system using particle swarm optimization feature extraction technique. Procedia Comput Sci. 2021;193:504–12. https://doi.org/10.1016/j.procs.2021.10.052.
39. Alzubi QM, Anbar M, Sanjalawe Y, Al-Betar MA, Abdullah R. Intrusion detection system based on hybridizing a modified binary grey wolf optimization and particle swarm optimization. Expert Syst Appl. 2022;204:117597. https://doi.org/10.1016/j.eswa.2022.117597.
40. Narayanasami S, et al. Biological feature selection and classification techniques for intrusion detection on BAT. Wirel Pers Commun. 2021. https://doi.org/10.1007/s11277-021-08721-8.
41. Dwivedi S, Vardhan M, Tripathi S. Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection. Cluster Comput. 2021;24(3):1881–900. https://doi.org/10.1007/s10586-020-03229-5.
42. Kohavi R, John GH. Wrappers for feature subset selection. Artif Intell. 1997. https://doi.org/10.1007/978-3-642-39038-8_27.
43. Guyon I, Barnhill JWS. Gene selection for cancer classification using support vector machines. Lect Notes Comput Sci. 2008;5139:62–72. https://doi.org/10.1007/978-3-540-88192-6_8.
44. Kasongo SM, Sun Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. J Big Data. 2020. https://doi.org/10.1186/s40537-020-00379-6.
45. Tang C, Luktarhan N, Zhao Y. An efficient intrusion detection method based on LightGBM and autoencoder. Symmetry. 2020;12(9):1–16. https://doi.org/10.3390/sym12091458.
46. Wang Z, Liu J, Sun L. EFS-DNN: an ensemble feature selection-based deep learning approach to network intrusion detection system. Secur Commun Netw. 2022. https://doi.org/10.1155/2022/2693948.
47. Yin Y, et al. IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. J Big Data. 2023. https://doi.org/10.1186/s40537-023-00694-8.
48. Rajadurai H, Gandhi UD. A stacked ensemble learning model for intrusion detection in wireless network. Neural Comput Appl. 2020;34(18):15387–95. https://doi.org/10.1007/s00521-020-04986-5.
49. Liu H, Yu L. Toward integrating feature selection algorithms for classification and clustering. IEEE Trans Knowl Data Eng. 2005;17(4):491–502. https://doi.org/10.1109/TKDE.2005.66.
50. Nazir A, Khan RA. A novel combinatorial optimization based feature selection method for network intrusion detection. Comput Secur. 2021;102:102164. https://doi.org/10.1016/j.cose.2020.102164.

51. Rashid M, Kamruzzaman J, Imam T, Wibowo S, Gordon S. A tree-based stacking ensemble technique with feature selection for network intrusion detection. Appl Intell. 2022;52(9):9768–81. https://doi.org/10.1007/s10489-021-02968-1.
52. Mushtaq E, Zameer A, Khan A. A two-stage stacked ensemble intrusion detection system using five base classifiers and MLP with optimal feature selection. Microprocess Microsyst. 2022;94:104660. https://doi.org/10.1016/j.micpro.2022.104660.
53. Mokbal F, Dan W, Osman M, Ping Y, Alsamhi S. An efficient intrusion detection framework based on embedding feature selection and ensemble learning technique. Int Arab J Inf Technol. 2022;19(2):237–48. https://doi.org/10.34028/iajit/19/2/11.
54. Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). Mil Commun Inf Syst Conf. 2015. https://doi.org/10.1109/MilCIS.2015.7348942.
55. Moustafa N, Slay J. The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf Secur J. 2016;25(1–3):18–31. https://doi.org/10.1080/19393555.2015.1125974.
56. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. Conf Inf Syst Secur Priv. 2018. https://doi.org/10.5220/0006639801080116.
57. T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction," *Springer Sci. Bus. Media*, 2009.
58. Yoshua B, Ian G, Aaron C. Deep learning. Cambridge: MIT Press; 2015.
59. Ganapathy S, Kulothungan K, Muthurajkumar S, Vijayalakshmi M, Yogesh L, Kannan A. Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. Eurasip J Wirel Commun Netw. 2013;271(1):1–16. https://doi.org/10.1186/1687-1499-2013-271.
60. Liu H, Zhou M, Liu Q. An embedded feature selection method for imbalanced data classification. IEEE/CAA J Autom Sin. 2019;6(3):703–15. https://doi.org/10.1109/JAS.2019.1911447.
61. Battiti R. Using mutual information for selecting features in supervised neural net learning. IEEE Trans Neural Networks. 1994;5(4):537–50. https://doi.org/10.1109/72.298224.
62. Vergara JR, Estévez PA. A review of feature selection methods based on mutual information. Neural Comput Appl. 2014;24(1):175–86. https://doi.org/10.1007/s00521-013-1368-0.
63. W. Li, "Mutual Information Functions Versus Correlation Functions in Binary Sequences," vol. 60, pp. 249–252, 1989, https://doi.org/10.1007/978-1-4757-0623-9_35.
64. Kursa MB, Rudnicki WR. Feature selection with the boruta package. J Stat Softw. 2010;36(11):1–13. https://doi.org/10.18637/jss.v036.i11.
65. Anand N, Sehgal R, Anand S, Kaushik A. Feature selection on educational data using Boruta algorithm. Int J Comput Intell Stud. 2021;10(1):27. https://doi.org/10.1504/ijcistudies.2021.113826.
66. Kursa MB, Jankowski A, Rudnicki WR. Boruta—a system for feature selection. Fundam Inform. 2010;101(4):271–85. https://doi.org/10.3233/FI-2010-288.
67. Schapire RE. The strength of weak learnability. Mach Learn. 1990;5(2):197–227. https://doi.org/10.1023/A:1022648800760.
68. Bbeiman L. Bagging predictors LEO. Kluwer Acad Publ Boston Manuf Netherlands Bagging. 1996;24:123–40. https://doi.org/10.3390/risks8030083.
69. Wolpert DH. Stacked generalization. Neural Netw. 1992;5:241–55.
70. Sagi O, Rokach L. Ensemble learning: a survey. Wiley Interdiscip Rev Data Min Knowl Discov. 2018;8(4):1–18. https://doi.org/10.1002/widm.1249.
71. Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern. 2012;42(4):463–84.
72. Džeroski S, Ženko B. Is combining classifiers with stacking better than selecting the best one? Mach Learn. 2004;54(3):255–73. https://doi.org/10.1023/B:MACH.0000015881.36452.6e.
73. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
74. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. Proc 22nd ACM SIGKDD Int Conf Knowl Discov Data Min. 2016. https://doi.org/10.1145/2939672.2939785.
75. L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," *Adv. Neural Inf. Process. Syst.*, pp. 6638–6648, 2018.
76. Kareem SS, Mostafa RR, Hashim FA, El-Bakry HM. An effective feature selection model using hybrid metaheuristic algorithms for iot intrusion detection. Sensors. 2022;22(4):1–23. https://doi.org/10.3390/s22041396.
77. Du X, Cheng C, Wang Y, Han Z. Research on network attack traffic detection hybridalgorithm based on UMAP-RF. Algorithms. 2022;15(7):1–17. https://doi.org/10.3390/a15070238.
78. Louk MHL, Tama BA. Dual-IDS: a bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. Expert Syst Appl. 2023;213(PB):119030. https://doi.org/10.1016/j.eswa.2022.119030.
79. Nkenyereye L, Tama BA, Lim S. A stacking-based deep neural network approach for effective network anomaly detection. Comput Mater Contin. 2020;66(2):2217–27. https://doi.org/10.32604/cmc.2020.012432.
80. Juan Fu J, Lan Zhang X. Gradient importance enhancement based feature fusion intrusion detection technique. Comput Netw. 2022;214:109180. https://doi.org/10.1016/j.comnet.2022.109180.
81. Tayde MV, Adhao RB, Pachghare V. Ensemble based feature selection technique for flow based intrusion detection system. 2022 IEEE 7th Int Conf Converg Technol. 2022. https://doi.org/10.1109/I2CT54291.2022.9824425.
82. Lazzarini R, Tianfield H, Charissis V. A stacking ensemble of deep learning models for IoT intrusion detection. Know-Based Syst. 2023;279:110941. https://doi.org/10.1016/j.knosys.2023.110941.
83. Yang Z, Liu Z, Zong X, Wang G. An optimized adaptive ensemble model with feature selection for network intrusion detection. Concurr Comput Pract Exp. 2022. https://doi.org/10.1002/cpe.7529.

84. Wang A, Wang W, Zhou H, Zhang J. Network intrusion detection algorithm combined with group convolution network and snapshot ensemble. Symmetry. 2021. https://doi.org/10.3390/sym13101814.
85. He H, Huang G, Zhang B, Qin L. Research on boruta-ET-based anomalous traffic detection model. Secur Commun Networks. 2022;2022:8. https://doi.org/10.1155/2022/9169266.
86. Harini R, Maheswari N, Ganapathy S, Sivagami M. An effective technique for detecting minority attacks in NIDS using deep learning and sampling approach. Alexandria Eng J. 2023;78(June):469–82. https://doi.org/10.1016/j.aej.2023.07.063.

Publisher's Note