EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article EAI.EU

An approach to reduce data dimension in building effective Network Intrusion Detection Systems

Hoang Ngoc Thanh[1,*], Tran Van Lang[2]

[1]Lac Hong University, Vietnam
[2]Institute of Applied Mechanics and Informatics, VAST, Vietnam

Abstract

The main function of the network Intrusion Detection System (IDS) is to protect the system, analyze and predict network access behavior of users. These behaviors are considered normal or an attack. Machine learning methods (ML) are used in IDSs because of the ability to learn from past attack patterns to recognize new attack patterns. These methods are effective but have relatively high computational costs. Meanwhile, the traffic of network data is growing rapidly, the computational cost issues need to be addressed. This paper addresses the use of algorithms combined with information metrics to reduce the features of the dataset to be analyzed. As the result, it helps to build IDSs with lower cost but higher performance suitable for large scale networks. The test results on the UNSW-NB15 dataset demonstrate: with the optimal set of features suitable for the attack type as well as the machine learning method, the quality of classification is improved with less training and testing time.

Received on 27 July 2019; accepted on 05 August 2019; published on 20 August 2019

Keywords: Intrusion Detection System, Machine learning, Feature selection, UNSW-NB15 dataset.

doi:10.4108/eai.13-7-2018.162633

1. Introduction

Due to recent technological advances, network-based services increasingly play an important role in modern society. Intruders are constantly looking for vulnerabilities on the computer system to gain unauthorized access to the kernel of the system. However, existing IDSs are still not flexible enough, scalability is not high, nor is it strong enough to deal with such attacks.

Previously, law-based methods were dominant. These methods find intrusion by comparing the characteristics of the data to be analyzed with known attack signs. As network traffic grows rapidly, updating the attack signs is becoming more and more difficult, tedious and time-consuming. Since then, machine learning methods have been introduced to solve intrusion detection problems. Machine learning refers to computer algorithms that are capable of learning from past attack patterns to recognize new attack patterns. Based on machine learning, IDSs have performed better in many reports as well as actual implementations [1]. However, the "no model" assets of such methods causes relatively high computational costs. Moreover, the traffic of network data is growing rapidly, computer cost issues need to be resolved [2].

One of the important solutions to reduce computational costs is to select the best features of the data to be analyzed. Reducing the number of features of a dataset will reduce training and testing time. At the same time, it improves the performance of classifiers in IDS. There are multiple feature selection methods, broadly categorized into Filter, Wrapper and Embedded methods based on their interaction with the predictor during the selection process. The Filter methods rank the variables as a preprocessing step, and feature selection is done before choosing the model. In the Wrapper approach, nested subsets of variables are tested to select the optimal subset that work best for the model during the learning process. The Embedded methods are those which incorporate variable selection in the training algorithm.

The content of this paper proposes a new method to reduce the features of a dataset based on Information Gain (IG), Gain Ratio (GR) and Correlation Attribute

*Corresponding author. Email: thanhhn@bvu.edu.vn

(CA) using the Wrapper approach. The comparative results show that the proposed method gives better performance than the other existing methods.

The remainder of this paper is organized as follows, Section 2 presents related works, Section 3 presents the challenges posed by the problem and proposed solution, Section 4 presents proposed feature selection method for improving classification efficiency with the UNSW-NB15 dataset, Section 5 presents the results obtained through experiments and Section 6 is conclusions and issues that need to be researched in the future.

2. Related works

The use of data dimension reduction techniques to enhance the effectiveness of IDS, has many different approaches presented by scholars, can be categorized into Bio-Inspired and Non-inspired Algorithms.

2.1. Bio-Inspired Algorithms

In Aslahi et al. [3] research a hybrid technique of Support Vector Machine (SVM) and GA was proposed for IDS. The proposed mix algorithm was utilized in decreasing the quantity of features from 41 to 10. Features sorted into three groups using GA algorithm from the most important feature group to less important feature groups. This is done such that 4 features are set in the highest significance, 4 included in the next, and 2 included in the third significance. The outcomes show that the proposed hybrid algorithm can achieve a true positive estimation of 0.973, while the false positive rate was 0.017.

Alternatively, a network intrusion detection method combining ant colony algorithm to select the features with a feature weighting SVM proposed by Xingzhu et al. [4]. First, the use of SVM classification accuracy and feature subset dimension construct a comprehensive fitness weighting index. Then use the ant colony algorithm for global optimization and multiple search capabilities to achieve optimal solutions feature subset search feature. And then selected the key feature of network data and calculated information gain access to various features weights and heavy weights to build SVM classifier based on the characteristics of network attacks right. At last, refine the final design of the local search methods to make the feature selection results without redundant features while improve the convergence resistance, and verify the dataset by KDD99 effectiveness of the algorithm. The results exhibited that the proposed approach can successfully reduce the dimension of features and have enhanced network intrusion detection accuracy to 95.75%

Rani et al. [5] proposed a new hybrid intrusion detection method combining multiple classifiers for classifying anomalous and normal activities on the computer network is presented. The misuse detection model is built based on the C5.0 Decision tree algorithm and using the information collected anomaly detection model is built which is implemented by one-class SVM. Integration of multiple algorithms helps to get better performance. The experimental results are performed on NSL-KDD dataset, and it is shown that overall performance of the proposed approach is improved in terms of detection rate and low false alarms rate in comparison to the existing techniques.

Ghanem et al. [6] anticipated a novel approach based on multi-objective artificial bee colony (ABC) for feature selection, particularly for intrusion-detection systems. The approach is divided into two stages: generating the feature subsets of the Pareto front of non-dominated solutions in the first stage and using the hybrid ABC and particle swarm optimization (PSO) with a feed-forward neural network (FFNN) as a classifier to evaluate feature subsets in the second stage. Thus, the proposed approach consists of two stages: (1) using a new feature selection technique called multi-objective ABC feature selection to reduce the number of features of network traffic data and (2) using a new classification technique called hybrid ABC–PSO optimized FFNN to classify the output data from the previous stage, determine an intruder packet, and detect known and unknown intruders. The proposed approach did not only provide a new approach for feature selection but also proposed a new fitness function for feature selection to diminish the number of features and achieve the minimum rate of classification errors and false alarms.

Acharya et al. [7] proposed an intelligent water drops (IWD) algorithm - based feature selection method. This method uses the IWD algorithm, a nature-inspired optimization algorithm for the feature subset selection along with SVM as a classifier for evaluation of the features selected. The experiments are conducted using KDD99 dataset. From 41 to 9, the model substantially reduced the features. Parameters found to have been better improved as presented in the new model with a proposed method are precision, false alarm rate accuracy and rate of detection. This outcome measured improved than other prevailing models. A precision rate of 99.4075% and an accuracy score of 99.0915% were recorded on the new model. While a low false rate of 1.405% and a precision rate of 99.108% were also recorded.

2.2. Non-inspired Algorithms

Ganapathy et al. [8] developed a new intelligent Conditional Random Field (CRF) based feature selection algorithm to optimize the number of features. In addition, an existing Layered Approach (LA) based algorithm is used to perform classification with these reduced features. The KDD99 dataset benchmark was used in

this research and the experimental results showed that the accuracy for different types of attack was (probe = 99.98, DOS = 97.62, R2L = 32.43, U2R = 86.91).
Ambusaidi et al. [9] proposed a mutual information based algorithm that analytically selects the optimal feature for classification. This algorithm can handle linearly and nonlinearly dependent data features. Its effectiveness is evaluated in the cases of network intrusion detection. Least Square Support Vector Machine based IDS (LSSVM-IDS), is built using the features selected by their feature selection algorithm. The performance of LSSVM-IDS is evaluated using three datasets: KDD99, NSL-KDD and Kyoto 2006. The evaluation results show that their feature selection algorithm contributes more critical features for LSSVM-IDS to achieve better accuracy and lower computational cost compared with other methods.
Madbouly et al. [10] proposed an upgraded model to choose an arrangement of the most applicable features to increase attack detection precision and enhance general system performance. In their approach, the KDD99 dataset was utilized, and the chosen important features containing just 12-beyond the 41-full features set. This now decreases the magnitude of the KDD99 workbench dataset by over 70%. They gauged the performance of the proposed model and confirmed its viability and attainability by comparing it with nine-unique models and with a model that utilized the 41-features dataset. The experimental outcomes demonstrated that, their improved prototype could productively accomplish increased detection rate, performance rate, low false alarm rate, and quick and consistent detection process.

3. Preliminary

The above known researches still have problems that need to be solved:
(1) The dataset used in the experiments is KDD99 or NSL-KDD, these are datasets that are over 20 years old, so it does not contain modern normal activities and contemporary synthetic attack behaviors.
(2) The evaluation metrics used by the majority of authors is not suitable for the imbalanced datasets as in network intrusion detection problem.
(3) Known feature reduction algorithms have low accuracy and high false alarm rates.

4. Proposed method

This paper proposes to resolve the above issues by the following methods:
(1) Use the new UNSW-NB15 dataset [1] created by the Australian Network Security Center (ACSC) in 2015.
(2) Use evaluation metrics such as $F-Measure$, $G-Means$ especially suitable for imbalanced datasets.
(3) Use the IG, GR and CA combined with the Backward Feature Elimination (BFE) algorithm to effectively reduce the features.
To find the set of features best suited to the type of attack as well as the machine learning method. First, depending on the type of attack, features will be ordered (descending) based on IG, GR and CA. Then, BFE algorithm is applied to select the most appropriate features for each machine learning method.
The following section gives a brief overview of measures used for ranking features, machine learning models, evaluation metrics, as well as the algorithms to select features used in the experiments.

4.1. Measures used for ranking features

The proposed measures to rank features include: Information Gain, Gain Ratio and Correlation Attribute are defined as follows.
Let S be set consisting of s data instances with m distinct classes. The expected information needed to classify a given instance is given by

$$I(S) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

where p_i is the probability that an arbitrary instance belongs to class C_i and is estimated by s_i/s.
Let attribute A has v distinct values. Let s_{ij} be number of instances of class C_i in a subset S_j. S_j contains those instances in S that have value a_j of A. The entropy, or expected information based on the partitioning into subsets by A, is given by

$$Ent(A) = -\sum_{i=1}^{m} I(S) \frac{s_{1i} + s_{2i} + ... + s_{mi}}{s}$$

(1) The Information Gain on feature A is calculated as follows [11]:

$$Gain(A) = I(S) - Ent(A)$$

The value represents the information generated by splitting the training dataset S into v partitions corresponding to v outcomes of a test on the attribute A:

$$SplitInfo(A) = -\sum_{i=1}^{v} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

(2) The Gain Ratio is defined as [11]:

$$GainRatio(A) = Gain(A)/SplitInfo(A)$$

(3) Attribute Correlation specifies the degree of dependency between attributes, it represents a linear relationship between attributes [11]:

$$r_{ab} = \frac{\sum_{i=1}^{n} (a_i - \overline{A})(b_i - \overline{B})}{N \sigma_A \sigma_B}$$

Here, N is the number of instances, a_i and b_i are the corresponding values of A and B in the i_{th} instance, $\overline{A}$ and $\overline{B}$ are the mean values of A and B; σ_A and σ_B are the standard deviations of A and B.

4.2. Machine learning models

In the experiments of this paper, the machine learning models used include: K Nearest Neighbors (kNN), Support Vector Machine (SVM), Artificial Neural Network (ANN), Decision Tree (DT), Naive Bayes (NB) and Logistic Regression (LR).

4.3. Use the right evaluation metrics

Using $Accuracy$ to assess the quality of classifiers has been used by many scholars. However, with imbalanced data, using $Accuracy$ to evaluate the quality of the classifiers is not really effective. Therefore, more comprehensive metrics have been suggested for evaluation such as $F-Measure$, $G-Means$ [12], [13].

4.4. Algorithms using BFE combine with IG, GR and CA to reduce features

The BFE algorithm is implemented as follows: First, the features of the UNSW-NB15 dataset are calculated for IG, GR and CA. The features will then be sorted in descending order of IG, GR and CA. A classification is built on the training dataset and is evaluated on the testing dataset with all features. After that, the features will then be removed one by one starting with the least important feature (with the lowest values of IG, GR and CA), then the classifier will be trained and tested on the training and testing datasets with the set of features is reduced, if the quality of the classification is improved, the feature will be removed, otherwise the feature will be retained. The pseudo code of the BFE algorithm is as follows:

```
Algorithm start
  S=Set of features ranked by IG, GR or CA
  S1=S
  Build classifier with features in S1
  Calculate evaluation metrics
  i=Number of features on training dataset
  while i>0
    S2=S1 removed the ith feature
    Build classifier with features in S2
    Calculate evaluation metrics
    if the metrics of S2 is better than S1
      S1=S2
    endif
    i=i-1
  endwhile
  return S1
Algorithm end
```

5. EXPERIMENTS

Programs and algorithms in the experiment using the Java programming language, based on the library, Weka machine learning framework developed by Waikato University, New Zealand.
Part 5.1 presents the datasets used in the experiment.
Part 5.2 presents the evaluation metrics used in the experiment.
Part 5.3 presents the results obtained using BFE combined with IG, GR and CA to reduce features with attack classes such as Worms, Shellcode, Backdoor, Generic, ...

5.1. Datasets

About the datasets used in IDS: KDD99, NSL-KDD and UNSW-NB15 are the most popular datasets, according to statistics from 2015 to 2018, showed that the NSL-KDD dataset is used 38%, KDD99 is 23% and UNSW-NB15 is 12% [1]. Previously, machine learning and data dimensional reduction techniques with the KDD99, NSL-KDD datasets are also performed by the authors [14–16]. In this paper, the UNSW-NB15 dataset was used for experiment because of its outstanding advantages compared to the KDD99 and NSL-KDD datasets [17]. Furthermore, studies of IDS from 2015 to 2018 show the low classification efficiency of their methods with the UNSW-NB15 dataset [1]. So a problem is to improve the classification method so that the results are better on this dataset.
The UNSW-NB15 dataset was developed using the IXIA tool to extract modern and offensive behavior conducted by ACSC in 2015. This dataset includes 9 types of attacks, 49 features and 2,540,044 instances [17], Table 1 describes the order and names of the features. A part of this dataset is divided into training and testing datasets, which are used extensively in scholars' experiments, detailed information about the datasets is presented in Table 2, with 9 types of attacks in dataset include: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode and Worms. In the experiments, to avoid overfitting, we do not use k-fold cross validation but use training dataset (82.332 instances) and test dataset (175.341 instances) separately included in UNSW-NB15.
The UNSW-NB15 dataset has several advantages when compared to the NSL-KDD dataset. First, it contains modern normal behavior and contemporary general attack activities. Second, the probability distribution of training and test datasets is similar. Third, it includes a set of features from payload and header of packages to reflect effective network packets. Finally, the complexity of evaluating UNSW-NB15 for existing classification systems shows that this dataset has complex patterns. This means that the dataset can be used to evaluate current and new classification methods

Table 1. The features of UNSW-NB15 dataset.

ID	Feature	ID	Feature	ID	Feature
1	attack_cat	16	dloss	31	response_body_len
2	dur	17	sinpkt	32	ct_srv_src
3	proto	18	dinpkt	33	ct_state_ttl
4	service	19	sjit	34	ct_dst_ltm
5	state	20	djit	35	ct_src_dport_ltm
6	spkts	21	swin	36	ct_dst_sport_ltm
7	dpkts	22	stcpb	37	ct_dst_src_ltm
8	sbytes	23	dtrcpb	38	is_ftp_login
9	dbytes	24	dwin	39	ct_ftp_cmd
10	rate	25	tcprtt	40	ct_flw_http_mthd
11	sttl	26	synack	41	ct_src_ltm
12	dttl	27	ackdat	42	ct_srv_dst
13	sload	28	smean	43	is_sm_ips_ports
14	dload	29	dmean		
15	sloss	30	trans_depth		

Table 2. Information about UNSW-NB15 traning and testing datasets.

Types of attacks	Testing dataset		Training dataset	
Normal	56.000	31,94%	37.000	44,94%
Analysis	2.000	1,14%	677	0,82%
Backdoor	1.746	1,00%	583	0,71%
DoS	12.264	6,99%	4.089	4,97%
Exploits	33.393	19,04%	11.132	13,52%
Fuzzers	18.184	10,37%	6.062	7,36%
Generic	40.000	22,81%	18.871	22,92%
Reconnaissance	10.491	5,98%	3.496	4,25%
Shellcode	1.133	0,65%	378	0,46%
Worms	130	0,07%	44	0,05%
Total	175.341	100,00%	82.332	100,00%

effectively and reliably.
However, because the UNSW-NB15 dataset is still quite new, it has not been used by many scholars in their studies. Therefore, there are limitations when comparing results with other studies.

5.2. Evaluation metrics

We denote:
TP_i : the number of correctly classified instances for class c_i
FP_i : the number of instances that were incorrectly classified to the class c_i
TN_i : the number of correctly classified instances that do not belong to the class c_i
FN_i : the number of instances that were not classified as belonging to the class c_i
The performance evaluation of the classifiers is done by measuring and comparing metrics:
- $Accuracy_i = (TP_i + TN_i)/(TP_i + FP_i + TN_i + FN_i)$
- $Sensitivity_i = TPR_i = (TP_i)/(TP_i + FN_i)$
- $Specificity_i = TNR_i = TN_i/(TN_i + FP_i)$
- $Efficiency_i = (Sensitivity_i + Specificity_i)/2$
- $Precision_i = TP_i/(TP_i + FP_i)$
- $FNR_i = FN_i/(FN_i + TP_i)$
- $FPR_i = FP_i/(FP_i + TN_i)$
- Time for training and testing

The use of *Accuracy* to assess the quality of classification has been used by many scholars. However, the class distribution in most nonlinear classification problems is very imbalanced. Therefore, using *Accuracy* to evaluate the quality of classification of a model is not really effective. Therefore, the more comprehensive metrics recommended for the evaluation of $F-Measure$ and $G-Means$ are calculated as follows [12], [13]:

$$F-Measure_i = \frac{(1+\beta^2) \times Precision_i \times Recall_i}{\beta^2 \times Precision_i + Recall_i}$$

Here, β is the coefficient that adjusts the relationship between Precision and Recall and usually $\beta = 1$. $F-Measure$ shows the harmonious correlation between *Precision* and *Recall*. $F-measure$ values are high when both *Precision* and *Recall* are high. And the $G-Means$ indicator is calculated:

$$G-Means_i = \sqrt{Sensitivity_i \times Specificity_i}$$

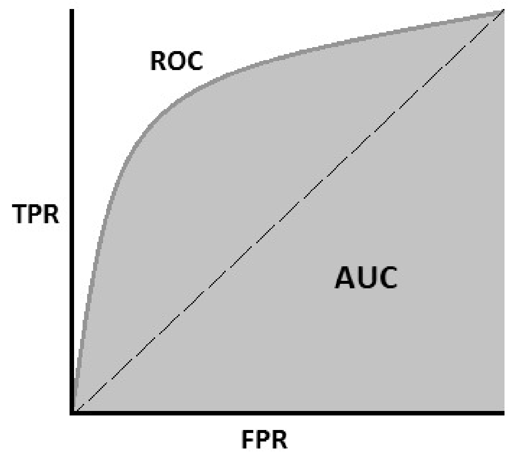

Figure 1. AUC - ROC Curve.

ROC (Receiver Operating Characteristics) is a method of calculating the performance of a classification model according to different classification thresholds. Assuming a binary classification problem (2 classes) using Logistic Regression, the selection of different classification thresholds [0..1] will affect the classification ability of the model and the level of influence of thresholds needs to be calculated. *ROC* is a probability and Area Under The Curve (AUC) representing the degree of classification of the model. Meaning of the interpretable AUC: It is the probability that a randomly sampled positive sample will be ranked higher than a randomized negative sample, $AUC = P((score(x^+) > score(x^-))$ The higher the AUC, the more accurate the model is in classifying classes. The *ROC* curve represents the pair of metrics (TPR, FPR) at each threshold with TPR is the vertical axis and FPR is the horizontal axis (Fig. 1). The evaluation metrics used in the experiments of this paper are: $Sensitivity$, $Specificity$, $Precision$, $F-Measure$, $G-Means$, AUC, Training time and Testing time.

5.3. Using BFE combined with IG, GR and CA to reduce features

Reduce features on the WORMS class. The results of using BFE in combination with IG, GR and CA on WORMS class with different machine learning algorithms are shown in the Table 3, Table 4 and Table 5. The machine learning techniques used for comparison include: DT, NB, LR, SVM, kNN and ANN. Accordingly, the best results are obtained when the feature reduction algorithm used is IG-BFE, the best machine learning algorithm used is DT, the number of remaining features after being reduced is 38. This is because as shown in Table 3, Table 4 and Table 5: although the $G-Means$ of NB, LR and kNN are higher, but they have a lower $Precision$, due to a high false alarm rate (normal access is misclassified as an attack); In contrast, SVM has a high $Precision$, but has too low $Sensitivity$ (attack access is misclassified as normal).

Table 3. Results of using IG-BFE algorithm on WORMS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.8385	0.8769	0.8615	0.1615	0.9077	0.8462
Specificity	0.9992	0.9681	0.9884	1.0000	0.9982	0.9977
Precision	0.6987	0.0600	0.1468	0.9545	0.5339	0.4622
F-Measure	0.7622	0.1124	0.2508	0.2763	0.6724	0.5978
G-Means	**0.9153**	0.9214	0.9228	0.4019	0.9519	0.9188
AUC	0.9489	0.9734	0.9726	0.5808	0.9519	0.9612
Train time	2.88 s	344 ms	64 s	32 s	0 ms	445 s
Test time	31 ms	1.98 s	156 ms	25 s	520 s	266 ms
Features	38	32	37	41	33	42

Table 4. Results of using GR-BFE algorithm on WORMS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.8385	0.8923	0.9231	0.1615	0.9231	0.8462
Specificity	0.9991	0.9672	0.9543	1.0000	0.9981	0.9977
Precision	0.6855	0.0593	0.0447	0.9545	0.5333	0.4622
F-Measure	0.7543	0.1113	0.0853	0.2763	0.6761	0.5978
G-Means	0.9153	0.9290	0.9385	0.4019	0.9599	0.9188
AUC	0.9489	0.9730	0.9510	0.5808	0.9521	0.9612
Train time	2.95 s	266 ms	616 s	43 s	6 ms	265 s
Test time	44 ms	1 s	125 ms	17 s	414 s	177 ms
Features	39	30	39	41	34	42

Table 6 compares the results of using IG-BFE with other feature reduction algorithms such as: Recursive Feature Elimination (RFE) algorithm combined with the ranking of the features using: Entropy, Gini index (Gini), Ridge, Random Forest and the average value of the above indexes (Avg). Thereby, IG-BFE gives better results when the features are not reduced. Although

Table 5. Results of using CA-BFE algorithm on WORMS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.8385	0.8615	0.9077	0.1615	0.9077	0.8462
Specificity	0.9991	0.9700	0.9723	1.0000	0.9981	0.9977
Precision	0.6855	0.0625	0.0707	0.9545	0.5291	0.4622
F-Measure	0.7543	0.1165	0.1311	0.2763	0.6686	0.5978
G-Means	0.9153	0.9142	0.9394	0.4019	0.9518	0.9188
AUC	0.9489	0.9742	0.9621	0.5808	0.9419	0.9612
Train time	2.25 s	367 ms	245 s	43 s	14 ms	266 s
Test time	86 ms	1.16 s	133 ms	16 s	528 s	170 ms
Features	39	31	36	41	35	42

$G-Means$ of other methods are better than IG-BFE, but the false alarm rate also increases, so IG-BFE is still selected.

Table 6. Compare IG-BFE with other algorithms on WORMS.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.8231	0.8385	0.8769	0.8692	0.8846	0.9462
Specificity	0.9991	0.9992	0.9981	0.9983	0.9982	0.9659
Precision	0.6903	0.6987	0.5182	0.5407	0.5374	0.0605
F-Measure	0.7509	0.7622	0.6514	0.6667	0.6686	0.1137
G-Means	0.9068	**0.9153**	0.9356	0.9315	0.9397	0.9560
AUC	0.9448	0.9489	0.9339	0.9303	0.9558	0.9942
Train time	3.84 s	2.88 s	6 ms	6 ms	5 ms	475 s
Test time	48 ms	31 ms	621 s	561 s	581 s	297 ms
Features	42	38	38	40	17	17
ML	DT	DT	kNN	kNN	kNN	ANN

Reduce features on the other classes. Similarly, the results of using BFE on SHELLCODE class are shown in the Table 7, Table 8 and Table 9. Accordingly, the best results are obtained when the feature reduction algorithm used is CA-BFE, the best machine learning algorithm used is DT, the number of remaining features after being reduced is 32. Table 10 compares the results of using CA-BFE with other algorithms. Thereby, CA-BFE gives better results all of rest algorithms.

Table 7. Results of using IG-BFE algorithm on SHELLCODE.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9029	0.9894	0.9815	0	0.6920	0.9320
Specificity	0.9919	0.9185	0.9388	0	0.9985	0.9800
Precision	0.6926	0.1973	0.2449	0	0.9032	0.4853
F-Measure	0.7839	0.3289	0.3920	0	0.7836	0.6383
G-Means	0.9464	0.9533	0.9599	0	0.8312	0.9557
AUC	0.9556	0.9661	0.9753	0	0.8923	0.9813
Train time	2.2 s	297 ms	28 s	0	16 ms	361 s
Test time	47 ms	2 s	172 ms	0	508 s	203 ms
Features	33	27	39	0	30	32

The results of using BFE in combination with IG, GR and CA on BACKDOOR class with different machine learning algorithms are shown in the Table 11, Table 12 and Table 13. Accordingly, the best results are obtained when the feature reduction algorithm used is CA-BFE, the best machine learning algorithm used is DT, the number of remaining features after being reduced is 34. Table 14 compares the results of using CA-BFE with

Table 8. Results of using GR-BFE algorithm on SHELLCODE.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9091	0.9912	0.9885	0	0.9303	0.9056
Specificity	0.9929	0.9158	0.9060	0	0.9114	0.9803
Precision	0.7203	0.1924	0.1755	0	0.1752	0.4817
F-Measure	0.8037	0.3222	0.2981	0	0.2948	0.6289
G-Means	0.9501	0.9528	0.9464	0	0.9208	0.9422
AUC	0.9747	0.9650	0.9775	0	0.9497	0.9759
Train time	3.1 s	244 ms	32 s	0	5 ms	249 s
Test time	55 ms	896 ms	122 ms	0	464 s	159 ms
Features	34	26	40	0	33	37

Table 9. Results of using CA-BFE algorithm on SHELLCODE.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9276	0.9550	0.9956	0	0.9276	0.9267
Specificity	0.9925	0.9548	0.8480	0	0.9045	0.9747
Precision	0.7145	0.2996	0.1170	0	0.1642	0.4258
F-Measure	0.8072	0.4562	0.2094	0	0.2790	0.5835
G-Means	**0.9595**	0.9549	0.9188	0	0.9160	0.9504
AUC	0.9763	0.9837	0.9717	0	0.9415	0.9790
Train time	2.58 s	296 ms	35 s	0	6 ms	223 s
Test time	43 ms	1.17 s	123 ms	0	501 s	149 ms
Features	32	32	39	0	32	33

Table 10. Compare CA-BFE with other algorithms on SHELLCODE.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.8244	0.9276	0.9356	0.9356	0.9285	0.8782
Specificity	0.9942	0.9925	0.9089	0.9093	0.9336	0.9934
Precision	0.7407	0.7145	0.1720	0.1726	0.2204	0.7284
F-Measure	0.7803	0.8072	0.2906	0.2914	0.3562	0.7963
G-Means	0.9053	**0.9595**	0.9221	0.9223	0.9310	0.9340
AUC	0.9270	0.9763	0.9738	0.9770	0.9786	0.9600
Train time	3.97 s	2.58 s	320 ms	344 ms	383 ms	3.93 s
Test time	48 ms	43 ms	1.38 s	1.35 s	1.32 s	62 ms
Features	42	32	17	15	15	24
ML	DT	DT	NB	NB	NB	DT

Table 11. Results of using IG-BFE algorithm on BACKDOOR.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9651	0.8814	0.8952	0.3328	0.9152	0.8213
Specificity	0.9932	0.9134	0.9743	0.9993	0.9735	0.9763
Precision	0.8160	0.2409	0.5210	0.9356	0.5187	0.5192
F-Measure	0.8843	0.3784	0.6587	0.4909	0.6621	0.6362
G-Means	0.9790	0.8973	0.9339	0.5766	0.9439	0.8954
AUC	0.9796	0.9610	0.9825	0.6660	0.9563	0.9424
Train time	4.47 s	313 ms	5.78 s	116 s	16 ms	423 s
Test time	46 ms	1.45 s	109 ms	106 s	484 s	203 ms
Features	33	28	24	14	26	34

Table 12. Results of using GR-BFE algorithm on BACKDOOR.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9668	0.8969	0.9152	0.3408	0.9559	0.8786
Specificity	0.9931	0.8988	0.8952	0.9996	0.9895	0.9906
Precision	0.8127	0.2165	0.2141	0.9597	0.7401	0.7439
F-Measure	0.8831	0.3489	0.3470	0.5030	0.8343	0.8057
G-Means	0.9798	0.8979	0.9052	0.5836	0.9726	0.9329
AUC	0.9805	0.9551	0.9016	0.6702	0.9841	0.9775
Train time	3.26 s	233 ms	4.18 s	92 s	6 ms	213 s
Test time	53 ms	806 ms	76 ms	31 s	465 s	130 ms
Features	35	29	26	12	28	29

Table 13. Results of using CA-BFE algorithm on BACKDOOR.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9668	0.9049	0.8666	0.3402	0.8356	0.9078
Specificity	0.9931	0.8998	0.8099	0.9991	0.9954	0.9770
Precision	0.8139	0.2197	0.1245	0.9195	0.8497	0.5515
F-Measure	0.8838	0.3535	0.2177	0.4967	0.8426	0.6861
G-Means	**0.9799**	0.9024	0.8378	0.5830	0.9120	0.9418
AUC	0.9808	0.9588	0.8293	0.6696	0.9377	0.9754
Train time	3.07 s	265 ms	4.37 s	67 s	7 ms	199 s
Test time	53 ms	882 ms	85 ms	27 s	455 s	114 ms
Features	34	30	26	11	30	27

other algorithms. Thereby, CA-BFE gives better results all of rest algorithms.

Table 14. Compare CA-BFE with other algorithms on BACKDOOR.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.9416	0.9668	0.9548	0.9548	0.9416	0.9611
Specificity	0.9894	0.9931	0.9892	0.9892	0.9894	0.9844
Precision	0.7339	0.8139	0.7340	0.7340	0.7339	0.6580
F-Measure	0.8249	0.8838	0.8300	0.8300	0.8249	0.7812
G-Means	0.9652	**0.9799**	0.9718	0.9718	0.9652	0.9727
AUC	0.9611	0.9808	0.9689	0.9689	0.9611	0.9717
Train time	8.73 s	3.07 s	6.97 s	7.12 s	7.34 s	7.31 s
Test time	78 ms	53 ms	61 ms	67 ms	75 ms	62 ms
Features	42	34	30	31	41	28
ML	DT	DT	DT	DT	DT	DT

Similarly, the results of using BFE on ANALYSIS class are shown in the Table 15, Table 16 and Table 17. Accordingly, the best results are obtained when the feature reduction algorithm used is IG-BFE, the best machine learning algorithm used is kNN, the number of remaining features after being reduced is 23. Table 18 compares the results of using IG-BFE with other algorithms. Thereby, IG-BFE gives better results all of rest algorithms.

Table 15. Results of using IG-BFE algorithm on ANALYSIS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.8590	0.7570	0.7515	0.2460	0.9055	0.6985
Specificity	0.9748	0.9603	0.8190	0.9991	0.9805	0.9939
Precision	0.5492	0.4050	0.1292	0.9044	0.6234	0.8033
F-Measure	0.6700	0.5277	0.2204	0.3868	0.7384	0.7473
G-Means	0.9151	0.8526	0.7845	0.4958	**0.9422**	0.8332
AUC	0.9169	0.9502	0.8394	0.6225	0.9338	0.8019
Train time	5.80 s	219 ms	34 s	52 s	0 ms	435 s
Test time	47 ms	1.34 s	141 ms	45 s	408 s	203 ms
Features	41	20	39	16	23	36

The results of using BFE on RECONNAISSANCE class are shown in the Table 19, Table 20 and Table 21. The best feature reduction algorithm used is CA-BFE, the best machine learning algorithm used is DT, the number of remaining features after being reduced is 35. Table 22 compares the results of using CA-BFE with other algorithms.

The results of using BFE on DOS class are shown in the Table 23, Table 24 and Table 25. The best feature

Table 16. Results of using GR-BFE algorithm on ANALYSIS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.8590	0.7440	0.9040	0.2715	0.8620	0.6420
Specificity	0.9748	0.9759	0.8099	0.9970	0.9778	0.9994
Precision	0.5492	0.5239	0.1452	0.7648	0.5809	0.9764
F-Measure	0.6700	0.6149	0.2502	0.4007	0.6940	0.7747
G-Means	0.9151	0.8521	0.8557	0.5203	0.9181	0.8010
AUC	0.9169	0.9631	0.8421	0.6343	0.9008	0.9078
Train time	2.59 s	193 ms	17.5 s	32 s	7 ms	255 s
Test time	41 ms	612 ms	101 ms	17.6 s	458 s	152 ms
Features	41	18	34	10	28	38

Table 17. Results of using CA-BFE algorithm on ANALYSIS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.8590	0.7325	0.8660	0.2500	0.8385	0.6480
Specificity	0.9748	0.9868	0.8146	0.9996	0.9778	0.9987
Precision	0.5492	0.6653	0.1430	0.9597	0.5739	0.9453
F-Measure	0.6700	0.6973	0.2454	0.3967	0.6814	0.7689
G-Means	0.9151	0.8502	0.8399	0.4999	0.9055	0.8044
AUC	0.9169	0.9647	0.8411	0.6248	0.8916	0.8639
Train time	2.86 s	191 ms	22 s	67.6 s	9 ms	254 s
Test time	45 ms	673 ms	105 ms	27.1 s	487 s	164 ms
Features	41	21	31	16	32	38

Table 18. Compare IG-BFE with other algorithms on ANALYSIS.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.8570	0.9055	0.8585	0.8590	0.8570	0.8605
Specificity	0.9743	0.9805	0.9757	0.9748	0.9743	0.9762
Precision	0.5440	0.6234	0.5580	0.5492	0.5440	0.5633
F-Measure	0.6655	0.7384	0.6764	0.6700	0.6655	0.6809
G-Means	0.9138	**0.9422**	0.9152	0.9151	0.9138	0.9165
AUC	0.8670	0.9338	0.8650	0.9169	0.8670	0.9146
Train time	6.3 s	0 ms	4.65 s	5.17 s	5.56 s	6.36 s
Test time	102 ms	408.2 s	54 ms	60 ms	64 ms	47 ms
Features	42	23	17	36	40	10
ML	DT	kNN	DT	DT	DT	DT

Table 19. Results of using IG-BFE on RECONNAISSANCE.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9529	0.9736	0.9852	0.9552	0.9774	0.9604
Specificity	0.9931	0.8634	0.9441	0.9310	0.8935	0.9841
Precision	0.9629	0.5718	0.7675	0.7218	0.6322	0.9186
F-Measure	0.9579	0.7205	0.8628	0.8222	0.7678	0.9390
G-Means	0.9728	0.9169	0.9644	0.9430	0.9345	0.9722
AUC	0.9867	0.9507	0.9908	0.9431	0.9419	0.9891
Train time	5.28 s	500 ms	16.3 s	205.8 s	0 ms	441.4 s
Test time	62 ms	2.30 s	141 ms	285.5 s	672.8 s	250 ms
Features	33	31	32	17	32	33

Table 20. Results of using GR-BFE on RECONNAISSANCE.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9705	0.9776	0.9672	0.9358	0.8809	0.9637
Specificity	0.9900	0.8709	0.9442	0.9404	0.9942	0.9803
Precision	0.9477	0.5866	0.7646	0.7463	0.9662	0.9018
F-Measure	0.9589	0.7332	0.8541	0.8303	0.9216	0.9317
G-Means	0.9802	0.9227	0.9556	0.9381	0.9359	0.9720
AUC	0.9871	0.9510	0.9893	0.9381	0.9643	0.9900
Train time	3.49 s	249 ms	14.93 s	391 s	21 ms	273 s
Test time	52 ms	846 ms	112 ms	453 s	687 s	206 ms
Features	36	22	36	21	32	35

reduction algorithm used is GR-BFE, the best machine learning algorithm used is DT, the number of remaining

Table 21. Results of using CA-BFE on RECONNAISSANCE.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9705	0.9748	0.9710	0.9343	0.9750	0.9617
Specificity	0.9900	0.8642	0.9577	0.9467	0.8993	0.9869
Precision	0.9477	0.5736	0.8113	0.7666	0.6446	0.9321
F-Measure	0.9589	0.7222	0.8840	0.8422	0.7761	0.9467
G-Means	**0.9802**	0.9179	0.9643	0.9405	0.9364	0.9742
AUC	0.9871	0.9508	0.9904	0.9405	0.9515	0.9910
Train time	4.26 s	417 ms	8.67 s	415.1 s	16 ms	285.1 s
Test time	41 ms	1.21 s	110 ms	441 s	709 s	220 ms
Features	35	30	30	21	33	37

Table 22. Compare CA-BFE with other algorithms on RECONNAISSANCE.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.9172	0.9705	0.9435	0.9435	0.9562	0.9720
Specificity	0.9933	0.9900	0.9863	0.9863	0.9698	0.9738
Precision	0.9623	0.9477	0.9280	0.9280	0.8555	0.8741
F-Measure	0.9392	0.9589	0.9357	0.9357	0.9031	0.9204
G-Means	0.9545	**0.9802**	0.9646	0.9646	0.9630	0.9729
AUC	0.9583	0.9871	0.9915	0.9915	0.9843	0.9903
Train time	7.65 s	4.26 s	314 s	315 s	293 s	567 s
Test time	61 ms	41 ms	198 ms	209 ms	197 ms	250 ms
Features	42	35	37	37	25	26
ML	DT	DT	ANN	ANN	ANN	ANN

features after being reduced is 30.
Table 26 compares the results of using GR-BFE with other algorithms. Thereby, GR-BFE gives the best results.

Table 23. Results of using IG-BFE algorithm on DOS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9689	0.8277	0.9336	0.8841	0.9561	0.9362
Specificity	0.9890	0.9287	0.8977	0.9652	0.9822	0.9892
Precision	0.9508	0.7176	0.6666	0.8478	0.9217	0.9498
F-Measure	0.9598	0.7688	0.7778	0.8655	0.9386	0.9430
G-Means	0.9789	0.8767	0.9155	0.9238	0.9691	0.9623
AUC	0.9836	0.9045	0.9650	0.9246	0.9805	0.9887
Train time	7.3 s	420 ms	9.97 s	216 s	0 ms	535 s
Test time	62 ms	1.69 s	141 ms	300 s	602 s	266 ms
Features	33	31	31	22	28	35

Table 24. Results of using GR-BFE algorithm on DOS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9690	0.8851	0.8166	0.8881	0.9641	0.9643
Specificity	0.9903	0.8928	0.9823	0.9652	0.9817	0.9556
Precision	0.9562	0.6439	0.9101	0.8484	0.9202	0.8261
F-Measure	0.9626	0.7455	0.8608	0.8678	0.9416	0.8899
G-Means	**0.9796**	0.8890	0.8957	0.9259	0.9729	0.9599
AUC	0.9799	0.9340	0.9820	0.9267	0.9843	0.9861
Train time	4.06 s	253 ms	6.35 s	305 s	10 ms	276 s
Test time	53 ms	825 ms	206 ms	185 s	629 s	165 ms
Features	30	26	29	24	28	34

The results of using BFE on FUZZERS class are shown in the Table 27, Table 28 and Table 29. The best feature reduction algorithm used is GR-BFE, the best machine learning algorithm used is SVM, the number of remaining features after being reduced is 15. Table 30 compares the results of using GR-BFE with other

Table 25. Results of using CA-BFE algorithm on DOS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9724	0.8651	0.8541	0.8780	0.9641	0.9510
Specificity	0.9874	0.9112	0.9802	0.9678	0.9643	0.9606
Precision	0.9441	0.6809	0.9043	0.8567	0.8553	0.8411
F-Measure	0.9580	0.7620	0.8785	0.8672	0.9064	0.8927
G-Means	0.9798	0.8878	0.9150	0.9218	0.9642	0.9558
AUC	0.9828	0.9259	0.9769	0.9229	0.9810	0.9838
Train time	6.27 s	304 ms	5.67 s	363 s	17 ms	283 s
Test time	67 ms	917 ms	105 ms	419 s	641 s	180 ms
Features	34	28	30	31	28	35

Table 26. Compare GR-BFE with other algorithms on DOS.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.9644	0.9690	0.9649	0.9649	0.9649	0.9644
Specificity	0.9892	0.9903	0.9891	0.9891	0.9891	0.9892
Precision	0.9512	0.9562	0.9511	0.9511	0.9511	0.9513
F-Measure	0.9577	0.9626	0.9579	0.9579	0.9579	0.9578
G-Means	0.9767	**0.9796**	0.9769	0.9769	0.9769	0.9767
AUC	0.9771	0.9799	0.9774	0.9774	0.9774	0.9773
Train time	11.05 s	4.06 s	8.41 s	9.03 s	10.62 s	11.19 s
Test time	68 ms	53 ms	73 ms	83 ms	93 ms	109 ms
Features	42	30	42	42	42	41
ML	DT	DT	DT	DT	DT	DT

algorithms. It is easy to see that GR-BFE has the best $G-Means$, but the $Sensitivity$ is not equal to other algorithms, because other algorithms have a high $Sensitivity$ but their $Specificity$ is very low, resulting in a high false alarm rate, so GR-FBE is still selected.

Table 27. Results of using IG-BFE algorithm on FUZZERS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.7377	0.9835	0.9962	0.3430	0.7988	0.8817
Specificity	0.8729	0.8032	0.8160	0.9516	0.8552	0.8398
Precision	0.6534	0.6187	0.6374	0.6971	0.6417	0.6412
F-Measure	0.6930	0.7596	0.7774	0.4598	0.7117	0.7424
G-Means	0.8025	0.8888	0.9016	0.5714	0.8265	0.8605
AUC	0.8287	0.8954	0.8997	0.6473	0.8467	0.9197
Train time	4.09 s	224 ms	20 s	157 s	30 ms	272 s
Test time	71 ms	961 ms	153 ms	149.8 s	456 s	197 ms
Features	35	25	38	12	24	37

Table 28. Results of using GR-BFE algorithm on FUZZERS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.7689	0.9691	0.9960	0.7069	0.8529	0.9148
Specificity	0.8648	0.8079	0.8160	0.9010	0.8513	0.8311
Precision	0.6487	0.6210	0.6373	0.6987	0.6507	0.6376
F-Measure	0.7037	0.7569	0.7773	0.7028	0.7382	0.7514
G-Means	0.8154	0.8849	0.9015	**0.7981**	0.8521	0.8720
AUC	0.8315	0.9026	0.9006	0.8040	0.8932	0.9200
Train time	5.55 s	254 ms	10.5 s	297.3 s	19 ms	243.7 s
Test time	80 ms	972 ms	142 ms	306.7 s	682.4 s	222 ms
Features	30	24	38	15	24	34

Similarly, the results of using BFE on EXPLOITS class are shown in the Table 31, Table 32 and Table 33. The best feature reduction algorithm used is CA-BFE, the best machine learning algorithm used is DT, the number of features after being reduced is 37. Table 34 compares

Table 29. Results of using CA-BFE algorithm on FUZZERS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.7622	0.9817	0.9962	0.4243	0.8489	0.8624
Specificity	0.8684	0.8046	0.8159	0.9539	0.8424	0.8406
Precision	0.6528	0.6200	0.6372	0.7493	0.6363	0.6373
F-Measure	0.7033	0.7600	0.7773	0.5418	0.7274	0.7329
G-Means	0.8136	0.8888	0.9015	0.6362	0.8457	0.8514
AUC	0.8421	0.9089	0.9011	0.6891	0.8993	0.9169
Train time	6.53 s	224 ms	9.99 s	212.9 s	19 ms	238.4 s
Test time	82 ms	938 ms	165 ms	183.8 s	606.2 s	199 ms
Features	31	20	38	10	22	33

Table 30. Compare GR-BFE with other algorithms on FUZZERS attacks.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.6927	0.7069	0.9959	0.9654	0.9959	0.9671
Specificity	0.8737	0.9010	0.8142	0.8154	0.8159	0.8152
Precision	0.6404	0.6987	0.6351	0.6294	0.6372	0.6295
F-Measure	0.6655	0.7028	0.7756	0.7620	0.7771	0.7626
G-Means	0.7780	**0.7981**	0.9005	0.8872	0.9014	0.8879
AUC	0.8060	0.8040	0.8977	0.9093	0.8993	0.9075
Train time	8.4 s	297 s	12.6 s	478 ms	14.55 s	515 ms
Test time	82 ms	307 s	161 ms	1.69 s	161 ms	2.81 s
Features	42	15	40	25	33	29
ML	DT	SVM	LR	NB	LR	NB

Table 31. Results of using IG-BFE algorithm on EXPLOITS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9677	0.9448	0.7694	0.9103	0.9549	0.8750
Specificity	0.9781	0.7718	0.9423	0.9438	0.9826	0.9855
Precision	0.9634	0.7117	0.8883	0.9061	0.9703	0.9729
F-Measure	0.9656	0.8119	0.8246	0.9082	0.9626	0.9214
G-Means	0.9729	0.8539	0.8515	0.9269	0.9687	0.9286
AUC	0.9828	0.8917	0.9474	0.9270	0.9833	0.9795
Train time	8.55 s	361 ms	9.14 s	180.8 s	16 ms	524.8 s
Test time	78 ms	1.98 s	228 ms	310.9 s	778.3 s	344 ms
Features	36	25	30	12	29	36

the results of using CA-BFE with other algorithms. CA-FBE was chosen because CA-BFE has the $G-Means$ close to other algorithms (0.9730 compared to 0.9734), but it has the higher $Specificity$ (0.9851 compared to 0.9769). That means the false alarm rate will be lower.

Table 32. Results of using GR-BFE algorithm on EXPLOITS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9661	0.9521	0.8155	0.8741	0.9562	0.8633
Specificity	0.9784	0.7002	0.9558	0.9502	0.9815	0.9829
Precision	0.9639	0.6544	0.9167	0.9128	0.9686	0.9678
F-Measure	0.9650	0.7757	0.8631	0.8931	0.9623	0.9126
G-Means	0.9722	0.8165	0.8829	0.9114	0.9688	0.9211
AUC	0.9810	0.8929	0.9564	0.9122	0.9803	0.9816
Train time	5.94 s	238 ms	4.79 s	222.3 s	13 ms	276.4 s
Test time	88 ms	928 ms	126 ms	184.6 s	716.2 s	209 ms
Features	35	24	30	13	29	33

The results of using BFE on GENERIC class are shown in the Table 35, Table 36 and Table 37. The best feature reduction algorithm used is IG-BFE, the best machine learning algorithm used is DT, the number of remaining features after being reduced is 34. Table 38 compares the results of using IG-BFE with other algorithms.

Table 33. Results of using CA-BFE algorithm on EXPLOITS.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9610	0.6752	0.8889	0.8853	0.9562	0.8329
Specificity	0.9851	0.9219	0.8365	0.9443	0.9815	0.9878
Precision	0.9746	0.8375	0.7643	0.9045	0.9686	0.9760
F-Measure	0.9678	0.7476	0.8219	0.8948	0.9623	0.8988
G-Means	**0.9730**	0.7889	0.8623	0.9143	0.9688	0.9071
AUC	0.9822	0.8836	0.9369	0.9148	0.9803	0.9716
Train time	6.63 s	236 ms	3.29 s	267.9 s	17 ms	279.2 s
Test time	92 ms	873 ms	122 ms	257.1 s	687.2 s	218 ms
features	37	21	24	12	30	35

Table 34. Compare CA-BFE with other algorithms on EXPLOITS.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.9580	0.9610	0.9699	0.9699	0.9593	0.9645
Specificity	0.9795	0.9851	0.9769	0.9769	0.9789	0.9776
Precision	0.9653	0.9746	0.9616	0.9616	0.9644	0.9626
F-Measure	0.9616	0.9678	0.9657	0.9657	0.9618	0.9635
G-Means	0.9687	**0.9730**	0.9734	0.9734	0.9690	0.9711
AUC	0.9687	0.9822	0.9809	0.9809	0.9734	0.9766
Train time	11.29 s	6.63 s	9.94 s	8.07 s	9.89 s	11.78 s
Test time	116 ms	92 ms	111 ms	111 ms	117 ms	281 ms
Features	42	37	31	31	37	33
ML	DT	DT	DT	DT	DT	DT

Thereby, IG-BFE gives the best results.

Table 35. Results of using IG-BFE algorithm on GENERIC.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9963	0.9763	0.9878	0.9853	0.9944	0.9901
Specificity	0.9933	0.9966	0.9891	0.9951	0.9934	0.9971
Precision	0.9906	0.9952	0.9848	0.9931	0.9908	0.9959
F-Measure	0.9934	0.9857	0.9863	0.9892	0.9926	0.9930
G-Means	**0.9948**	0.9864	0.9884	0.9902	0.9939	0.9936
AUC	0.9971	0.9922	0.9959	0.9902	0.9962	0.9984
Train time	4.14 s	224 ms	7.12 s	429.7 s	17 ms	356.1 s
Test time	66 ms	830 ms	138 ms	175.7 s	1246 s	283 ms
Features	34	17	30	36	33	35

Table 36. Results of using GR-BFE algorithm on GENERIC.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9965	0.9766	0.9878	0.9831	0.9954	0.9885
Specificity	0.9929	0.9854	0.9914	0.9991	0.9938	0.9981
Precision	0.9902	0.9795	0.9880	0.9987	0.9913	0.9973
F-Measure	0.9933	0.9780	0.9879	0.9908	0.9933	0.9928
G-Means	0.9947	0.9810	0.9896	0.9910	0.9946	0.9932
AUC	0.9974	0.9891	0.9960	0.9911	0.9967	0.9977
Train time	4.57 s	303 ms	8.06 s	445.7 s	19 ms	357.6 s
Test time	83 ms	1.21 s	130 ms	259.8 s	1136 s	290 ms
Features	36	25	29	37	29	37

Table 39 represents the effectiveness of the proposed method compared to the classification using full of features in the metrics: $G-Means$ (G-Mea), training time (TrTime) and testing time (TeTime).

Table 40 shows the selected features and machine learning algorithms for each attack type.

Table 37. Results of using CA-BFE algorithm on GENERIC.

Metrics	DT	NB	LR	SVM	kNN	ANN
Sensitivity	0.9963	0.9815	0.9879	0.9832	0.9950	0.9908
Specificity	0.9923	0.9948	0.9916	0.9991	0.9929	0.9970
Precision	0.9892	0.9927	0.9883	0.9987	0.9901	0.9957
F-Measure	0.9927	0.9871	0.9881	0.9909	0.9926	0.9932
G-Means	0.9942	0.9882	0.9898	0.9911	0.9940	0.9939
AUC	0.9973	0.9910	0.9961	0.9911	0.9963	0.9982
Train time	5.96 s	269 ms	5.57 s	375 s	15 ms	347.8 s
Test time	71 ms	1.07 s	124 ms	198.9 s	1140 s	265 ms
Features	37	21	27	34	30	36

Table 38. Compare IG-BFE with other algorithms on GENERIC.

	Origin	BFE	Entropy	Gini	Ridge	Avg
Sensitivity	0.9963	0.9963	0.9963	0.9922	0.9963	0.9929
Specificity	0.9894	0.9933	0.9896	0.9942	0.9900	0.9913
Precision	0.9854	0.9906	0.9856	0.9919	0.9862	0.9879
F-Measure	0.9908	0.9934	0.9909	0.9920	0.9912	0.9904
G-Means	0.9929	**0.9948**	0.9929	0.9932	0.9931	0.9921
AUC	0.9967	0.9971	0.9968	0.9984	0.9970	0.9970
Train time	8.76 s	4.14 s	8.2 s	395 s	10.4 s	16 ms
Test time	82 ms	66 ms	114 ms	305 ms	207 ms	1945 s
Features	42	34	42	17	41	12
ML	DT	DT	DT	ANN	DT	kNN

Table 39. The effect of feature reducing with attacks using BFE.

Attack	Full of features			After using BFE		
	G-Mea	TrTime	TeTime	G-Mea	TrTime	TeTime
Worms	0.9068	3.84 s	48 ms	0.9153	2.88 s	31 ms
Shellcode	0.9053	3.97 s	48 ms	0.9595	2.58 s	43 ms
Backdoor	0.9652	8.73 s	78 ms	0.9799	3.07 s	53 ms
Analysis	0.9138	6.31 s	102 ms	0.9422	0 ms	408 s
Reconnai	0.9545	7.65 s	61 ms	0.9802	4.26 s	41 ms
DoS	0.9767	11.1 s	68 ms	0.9796	4.06 s	53 ms
Fuzzers	0.7780	8.44 s	82 ms	0.7981	297 s	307 s
Exploits	0.9687	11.3 s	116 ms	0.9730	6.63 s	92 ms
Generic	0.9929	8.76 s	82 ms	0.9948	4.14 s	66 ms

Table 40. Results of feature reducing with each attack type using BFE algorithm.

Attack	ID of features selected	ML
Worms	2,3,4,5,6,7,8,9,10,11,12,13,15,16,17,18,19,20,21,22, 23,24,25,26,27,28,30,31,33,34,36,37,38,39,40,41,42, 43	DT
Shellcode	4,5,6,7,8,9,11,12,13,14,15,16,17,18,19,21,24,27,28, 29,30,31,32,33,35,36,37,38,39,40,42,43	DT
Backdoor	2,3,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,21,23,24, 25,26,27,28,29,31,33,34,37,38,39,40,42,43	DT
Analysis	2,3,5,6,7,9,11,12,14,15,16,17,18,19,20,21,24,31,33, 38,39,40,43	kNN
Reconna-issance	2,3,4,5,6,7,8,10,11,12,13,16,18,19,20,21,22,23,24,25, 26,27,28,29,31,33,34,35,36,37,38,39,40,41,43	DT
DoS	2,3,4,8,9,12,14,15,16,17,19,20,21,23,24,25,26,28,29, 30,32,33,34,35,36,39,40,41,42,43	DT
Fuzzers	2,4,5,7,9,10,11,13,14,19,24,28,37,38,39	SVM
Exploits	2,3,4,5,6,7,9,10,11,13,14,15,16,17,19,20,21,22,23,24, 25,27,28,29,30,32,33,34,35,37,38,39,40,41,42,43	DT
Generic	2,3,4,7,8,9,10,11,12,13,14,15,16,18,20,21,22,24,25, 28,29,30,32,33,34,35,36,37,38,39,40,41,42,43	DT

6. CONCLUSION

From the experimental results, some conclusions are drawn as follows:

(1) Class distribution in the network intrusion detection system is very imbalanced, so using

Accuracy metric to evaluate the classification quality of a model is incorrect. Therefore, the use of metrics such as: $F-Measure$ and $G-Means$ proved it.

(2) The evaluation results on the UNSW-NB15 dataset show that this dataset has many complex patterns, especially at the attack classes such as WORMS, ANALYSIS and FUZZERS.

(3) Dimensional reduction of data not only reduces computational cost but also improves classification quality in Intrusion Detection Systems.

(4) The use of the IG-BFE, GR-BFE and CA-BFE algorithms to reduce data dimensions is better than the other known algorithms.

(5) For each different type of attack, different features and machine learning algorithm will be chosen so that the performance of the intrusion detection system is improved at the best.

At the same time, the experimental results also set out issues that need to be further studied, especially the contents:

(1) Research using ensemble methods such as boosting, bagging, voting, stacking, and etc, can help improve classification quality.

(2) The UNSW-NB15 dataset is still quite new, it has not been used by many scholars in their studies. Therefore, there are limitations when comparing results with other studies.

References

[1] SH Kok, Azween Abdullah, NZJhanjhi, and Mahadevan Supramaniam. A review of intrusion detection system using machine learning approach. *International Journal of Engineering Research and Technology, ISBN 0974-3154*, 12(1):8–15, 2019.

[2] Al-Jarrah O. Y., Siddiqui A., et al. Machine-learning-based feature selection techniques for large-scale network intrusion detection. In *Distributed Computing Systems Workshops, 2014 IEEE 34th International Conference on*, pages 177–181, 2014.

[3] B.M. Aslahi-Shahri, R. Rahmani, M. Chizari, A. Maralani, M. Eslami, M.J. Golkar, and A. Ebrahimi. A hybrid method consisting of ga and svm for intrusion detection system. *Neural Comput. Appl.*, 27:1669—-1676, 2016.

[4] W. Xingzhu. Aco and svm selection feature weighting of network intrusion detection method. *Analysis*, 9:129—-270, 2015.

[5] M.S. Rani and S.B. Xavier. A hybrid intrusion detection system based on c5.0 decision tree and one-class svm. *Int. J. Curr. Eng. Technol.*, 5:2001—-2007, 2015.

[6] W.A.H.M. Ghanem and A. Jantan. Novel multi-objective artificial bee colony optimization for wrapper based feature selection in intrusion detection. *Int. J. Adv. Soft Comput. Appl.*, 8:70—-81, 2016.

[7] N. Acharya and S. Singh. An iwd-based feature selection method for intrusion detection system. *Soft Comput.*, 22:1—-10, 2017.

[8] S. Ganapathy, P. Vijayakumar, P. Yogesh, and A. Kannan. An intelligent crf based feature selection for effective intrusion detection. *Int. Arab J. Inf. Technol.*, 13:44—-50, 2016.

[9] M.A. Ambusaidi, X. He, P. Nanda, and Z. Tan. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Comput.*, 65:2986—-2998, 2016.

[10] A.I. Madbouly and T.M. Barakat. Enhanced relevant feature selection model for intrusion detection systems. *Int. J. Intell. Eng. Inform.*, 4:21—-45, 2016.

[11] J. W. Han, M. Kamber, and J. Pei. *Data Mining Concepts and Techniques*. 3rd Edition, Morgan Kaufmann Publishers, Waltham, 2012.

[12] R. P. Espíndola and N. F. F. Ebecken. On extending f-measure and g-mean metrics to multi-class problems. In *WIT Transactions on Information and Communication Technologies*, volume 35. WIT Press, 2005.

[13] Yuk Ying Chung and Noorhaniza Wahid. A hybrid network intrusion detection system using simplified swarm optimization (sso). In *Applied Soft Computing 12*, pages 3014–3022. Elsevier, 2012.

[14] Hoang Ngoc Thanh, Tran Van Lang, and Hoang Tung. A machine learning approach to classify types of attacks in network intrusion detection system. In *Proceedings of the 9th National Conference on Fundamental and Applied IT Research (FAIR'9)*, pages 502–508, Vietnam, 2016.

[15] Hoang Ngoc Thanh and Tran Van Lang. Feature selection based on information gain to improve performance of network intrusion detection systems. In *Proceedings of the 10th National Conference on Fundamental and Applied IT Research (FAIR'10)*, pages 823–831, Vietnam, 2017.

[16] Hoang Ngoc Thanh and Tran Van Lang. Creating rules for firewall use of decision tree based ensemble techniques. In *Proceedings of the 11th National Conference on Fundamental and Applied IT Research (FAIR'11)*, pages 489–496, Vietnam, 2018.

[17] N. Moustafa and J. Slay. Unsw-nb15: A comprehensive dataset for network intrusion detection. In *Paper presented at the Military Communications and Information Systems Conference*, 2015.