Received 16 June 2022, accepted 25 June 2022, date of publication 29 June 2022, date of current version 11 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186975

RESEARCH ARTICLE

A Hybrid Intrusion Detection System Based on Feature Selection and Weighted Stacking Classifier

RUIZHE ZHAO[1], YINGXUE MU[2,3], LONG ZOU[1], AND XIUMEI WEN[2,3]
[1]Department of Information Engineering, Hebei University of Architecture, Zhangjiakou 075000, China
[2]Department of Information Management, Hebei University of Architecture, Zhangjiakou 075000, China
[3]Big Data Technology Innovation Center of Zhangjiakou, Zhangjiakou 075000, China

Corresponding author: Xiumei Wen (xiumeiwen@163.com)

This work was supported in part by the Research Project of Fundamental Scientific Research Business Expenses of Provincial Colleges and Universities in Hebei Province under Grant 2021QNJS04, and in part by the Research Project on Basic Research Funds for Higher Education Institutions in Hebei Province under Grant 2021XSTD04.

ABSTRACT Cyber-attacks occur more frequently with the rapid growth in the Internet. Intrusion detection systems (IDS) have become an important part of protecting system security. There are still some challenges preventing IDS from further improving its classification performance. Firstly, the complexity of high-dimensional features challenges the speed and the performance of the classification for IDS. Secondly, the classification performance of traditional Stacking algorithm can be easily affected by the base classifiers. Tackling both challenges above, we propose a hybrid intrusion detection system based on a CFS-DE feature selection algorithm and a weighted Stacking classification algorithm. To limit the dimension of the features, we deployed the CFS-DE algorithm, which searches for the optimal feature subset. Afterwards, a weighted Stacking algorithm is proposed, which increases the weights of the base classifiers with good training results and drops those base classifiers with bad ones to improve the classification performance. As such, the model enhances the classification efficiency and yielding better accuracy. All experiments in this study were conducted on the NSL-KDD and CSE-CIC-IDS2018 data sets. The results based on KDDTest+ show that our proposed model has accuracy of 87.44%, precision of 89.09%, recall of 87.44% and F1-score of 88.25%. The results based on CSE-CIC-IDS2018 show that our proposed model has accuracy of 99.87%, precision of 99.88%, recall of 99.87% and F1-score of 99.88%. Compared with traditional machine learning models and models mentioned in other papers, out proposed CFS-DE-weighted-Stacking IDS has the best classification performance.

INDEX TERMS Intrusion detection system, feature selection, weighted Stacking, CFS-DE, cyber security.

I. INTRODUCTION

In recent years, with the continuous development of network communication technology, Internet of Things(IoT), cloud computing and other technologies, these network based technologies are deeply rooted in the modern society [1]–[3]. The security situation in cyberspace is becoming increasingly complex. The cyber security of enterprises, governments and individuals are constantly risked by various cyber security threats [4]. According to the "2021 Bad Bot Report" released by Imperva, in 2021, only 59.2% of the entire network traffic is human traffic. Among the traffic generated by machines, malicious traffic accounted for 25.6% of the entire network traffic [5]. In many network security incidents, malicious intrusions account for a large proportion, which imposes huge threats to both network users and companies. Intrusion detection systems (IDS) are currently an important part of protecting the security of hosts and systems [6].

Among the traditional intrusion detection systems, majority of them uses either a port-based approach or a Deep

The associate editor coordinating the review of this manuscript and approving it for publication was Alberto Cano.

Packet Inspection (DPI) method [7]. The port-based approach focuses on identifying traffic based on the ports registered by the Internet Assigned Numbers Authority(IANA) [8]. The DPI-based detection method is based on matching the packet payload and the packet's storage signature [9]. However, these two methods are unreliable when the application randomly generate ports or the packet contents are not allowed to be accessed. In recent years, researchers have been working on machine learning methods [10], which does not need to acquire the content of the packet, which can classify the instances based on the features of the data stream [11], [12].

Stacking, as one of the machine learning techniques, is widely used in intrusion detection [13]. Stacking algorithm can integrate the advantages of multiple base classifiers, which can improve the diversity and generalization ability of the model. However, the classification results of Stacking is easily affected by the base model. Moreover, another challenge for IDS is the high-dimensional features from the data set. These irrelevant and redundant features will have negative impact on accuracy and efficiency of the IDS [14].

In this study, we introduce a hybrid intrusion detection system which combines feature selection algorithm and weighted Stacking algorithm with the aim of improving the accuracy and efficiency of the classification. To address the problem of high-dimensional features, we proposed the CFS-DE feature selection algorithm, which uses the CFS to evaluate the feature subset, and uses the differential evolution algorithm to optimise the feature subsets. A weighted Stacking algorithm is also proposed to further improve the classification performance of Stacking, which uses the Kappa coefficient to weight the base classifier. This weighted Stacking algorithm will increase the weight of base classifiers with high accuracy and reduce the weight of base classifiers with low accuracy by weighting the results of the base classifiers to improve the classification performance of the algorithm. The major contributions of this paper are summarized as follows:

1) We propose a hybrid intrusion detection system which combines feature selection algorithms and weighted Stacking algorithms. The hybrid IDS improve the accuracy and efficiency of classification.
2) In the part of feature selection, we propose CFS-DE feature selection approach to reduce the dimension of the features and search for the best feature subset.
3) To further improve the accuracy and efficiency of classification on unbalanced data sets, we proposed weighted Stacking algorithm, which increases the weights of the base classifiers with good training results and drops those base classifiers with bad ones to improve the classification performance.
4) We made the experiment on NSL-KDD and CSE-CIC-IDS2018 data sets. Experimental result demonstrate that our proposed model improve the classification performance in terms of Accuracy, Precision, Recall and F1-score.

The rest of the paper is organized as follows. Section II introduced the research and improvement of other scholars in intrusion detection and Stacking methods. In Section III, we introduce the CFS-DE feature selection algorithm and weighted Stacking algorithm with the details. In Section IV, we build experiments on NSL-KDD and CSE-CIC-IDS2018 data sets. Finally, we draw the conclusions and future research directions in Section V.

II. RELATED WORK

The current research in the field of intrusion detection mainly includes two aspects: feature selection and cyber-attack classification [15]. In this section, we consider feature selection techniques, Stacking ensemble method and hybrid approaches.

A. FEATURE SELECTION TECHNIQUES

Feature selection is an important data pre-processing method that can eliminate certain redundant or unnecessary features and improve classifier efficiency and accuracy; therefore, scholars have paid much attention to it. [16]. Nimbalkar *et al.* [17]proposed a feature selection algorithm based on Information Gain (IG) and Gain Ratio (GR) for intrusion detection of Internet of Things (IoT). The algorithm obtains the final feature subset by taking the top 50% IG and GR features and merging them. The algorithm was experimented on the IoT-BoT and KDD CUP 1999 data sets and proved to have better performance than the other IDS. Li *et al.* [18]proposed an improved krill swarm algorithm based on linear nearest neighbour lasso step (LNNLS-KH). The algorithm introduce the number of features and classification accuracy into the fitness evaluation function to obtain the global optimal solution. Experiments on the NSL-KDD and CIC-IDS-2017 data sets have confirmed that the algorithm can reduce the number of features and improve the detection rate. The HW algorithm was proposed by Mohammad *et al.* [19]. They apply the algorithm for creating classification models on 10 UCI classification data sets and NSL-KDD data set. It was observed that this approach achieve better performance than Information Gain (IG) and chi-square.

B. STACKING ENSEMBLE METHOD

As a typical model in ensemble learning, Stacking is widely used in intrusion detection [20]. Nguyen *et al.* [21] designed a Stacking-based classification model in order to improve the accuracy of intrusion detection in surveillance and data collection systems. The model used Random Forest, LightGBM and XGBoost as base classifiers and MLP as secondary classifiers. Olasehinde *et al.* [22] focus on Meta classifiers in Stacking. By comparing Meta Decision Tree, Multi-Response Model Trees and Multi Response Linear Regression, the observed results showed that the best performance of MDT was the among three classifiers. However, choosing the meta classifier can not solve the problem caused by the base model. The base model with poor classification results will still affect the classification results of the model. Oriola *et al.* [23] used a combination of two secondary classifiers in order to improve the accuracy of the Stacking algorithm. They used grid search algorithm to optimise the combination of secondary classifiers. In order to identify zero-day attacks, Tommaso Zoppi *et al.* [24] proposed a Stacking algorithm, while building a two-layer hybrid Stacking algorithm. Supervised and unsupervised learning methods were both used as base classifiers in this algorithm. But they did not focus on the effect of poorly base classifiers on the classification results of the model. Gao *et al.* [25] designed an adaptive ensemble machine learning model for intrusion detection. Decision tree (DT), random forest (RF), K-nearest neighbors (KNN), deep neural network (DNN) were used as base classifiers. The algorithm reached the highest accuracy of 85.2%, which was verified on KDDTest+.

C. HYBRID APPROACHES

In recent years, many hybrid approaches combine with feature selection techniques and Stacking ensemble method to improve the performance of IDSs. Herrera-Semenets *et al.* [26] proposed an IDS based on multi-measure feature selection algorithm. They combine Information Gain (IG), chi square and ReliefF algorithms to construct multi-measure feature selection algorithm. Decision Tree was used to classify all records in data set. The results revealed that the IDS based on multi-measure feature selection algorithm is better than other related approaches. Krishnaveni *et al.* [27] proposed a novel intrusion detection system (IDS) on cloud computing. They use IG, Chi Square, Gain Ratio, Symmetric Uncertainty and Relief for feature selection. The IDS combine support vector machine(SVM), naive Bayesian, logistic regression and decision tree to build ensemble learning classifiers. Upadhyay *et al.* [28] proposed an Intrusion Detection System (IDS) of RFE-XGBoost feature selection algorithm and ensemble learning algorithms. The experimental results prove that the proposed frame-work fares well in terms of accuracy, detection rate, precision, and recall. Shunmugapriya *et al.* [29] use artificial bee colony algorithm to improve the performance of Stacking algorithm. Rajagopal *et al.* [30] proposed an algorithm using information gain based feature selection and Stacking techniques. They made experiments on UNSW-NB15 and UGR16 data sets. The observed results showed that the algorithm produces a significant improvement of accuracy.

III. METHODOLOGY

To further improve the classification ability of IDS, we propose a hybrid ML-based IDS using CFS-DE feature selection approach and weighted Stacking classifier. In this study, 5-fold cross validation is used to validate the classification performance of all models. All approaches of our proposed IDS are provided in this section.

The process of hybrid intrusion detection system shows in Fig. 1 mainly includes the following processes:

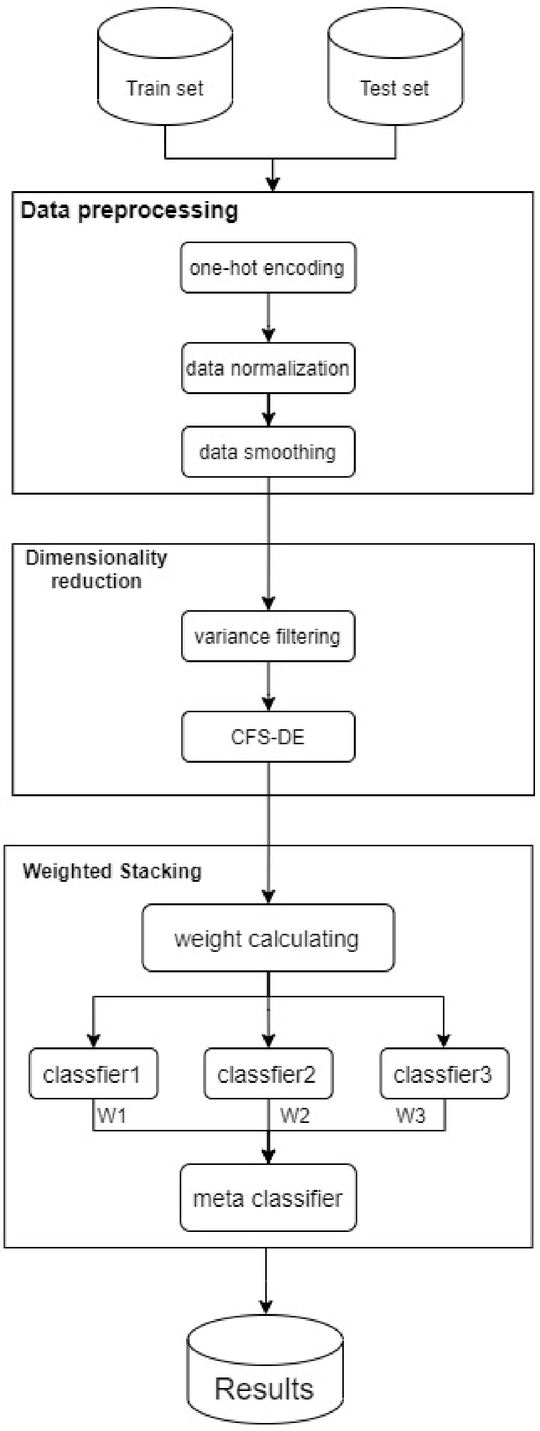


FIGURE 1. **The framework of the proposed feature selection-weighted Stacking model.**

1) Data loading. In this step, we input the train set and test set into the model.
2) Data pre-processing. One-hot encoding, variance filtering, data normalization and data smoothing are used in this step. One-hot encoding is used to expand the categorical features. Normalize the data set and use the *minmaxScaler* function to map all data to the interval

(0, 1). The data is smoothed using the $log1p$ function to make all data more in line with Gaussian distribution.

3) Dimension reduction: The variance of each feature is calculated and features with variance below a threshold are removed. Then we propose CFS-DE feature selection approach to reduce the dimension of the features and search for the best feature subset.
4) weighted Stacking: In order to further improve the accuracy of IDS classification, we propose the weighted Stacking algorithm, which weights the results of the base classifier. The algorithm increases the weights of base classifiers with high accuracy and decreases the weights of base classifiers with low accuracy to improve the accuracy of the model.

A. CORRELATION-BASED FEATURE SELECTION (CFS)

Correlation-based Feature Selection (CFS) is a classical filtering algorithm proposed by Hall in 1999 [31]. CFS algorithm can be calculated by the average correlation between each feature and label class and between each feature in each subset obtained. The greater the correlation between the feature subset and the label column, and the less redundancy with other features, the higher the evaluation value obtained. The evaluation method is shown as follows.

$$M_s = \frac{k r_{cf}}{\sqrt{k + k(k-1) r_{ff}}} \tag{1}$$

where: M_S represents an evaluation of a feature subset S containing k feature items; r_{cf} is the average correlation between different features and label classes, and r_{ff} is the average correlation between features and features.

According to the (1), CFS can accurately and efficiently evaluate the importance of each feature subset in the data set and can be used as an important basis for feature selection.

B. DIFFERENTIAL EVOLUTION ALGORITHM

Differential Evolution (DE) is an optimisation algorithm based on random multidimensional data [32]. The algorithm approximates towards the optimal solution by continuously retaining the good individuals and eliminating the poor ones. The differential evolution algorithm is divided into four main steps: individuals initialization, mutation, crossover, and individuals selection.

In the process of iteration, the population of each generation G contains N individuals, and this population can be represented by $X_{i,G}(i = 1, 2, \cdots, N)$.

The mutation operation means that the algorithm follows this strategy to generate new individuals from the parent population into the next generation. For individuals $X_{i,G} : i = 1, 2, \cdots, N$, a new individual can generate by (2).

$$V_{i,G+1} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G}) \tag{2}$$

The mutually exclusive integers $r1, r2, r3$ are randomly selected from the interval [1,n]. The value of F is from 0 to 2.

Crossover is to randomly select two individuals from the population and exchange some genes to generate new individual $V_{i,G+1}$. The purpose of crossover is to inherit the excellent genes and generate better individuals. The crossover is shown in (3) and (4).

$$U_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \cdots, u_{Di,G+1}) \tag{3}$$

$$u_{ji,G+1} = \begin{cases} V_{ji,G+1} & \text{if } randb(j) \le CR \text{ or } (j = mbr(i)) \\ X_{ji,G+1} & \text{otherwise} \end{cases} \tag{4}$$

$randb(j)$ is a uniformly distributed probability between 0 and 1. CR is the probability of crossover.

To generate individuals of the $G+1$ generation, we compared the fitness of individuals of the new generation with those of the previous generation. The individual selection strategy can be represented as (5). The fitness of individual X is represented by $IEF(X)$.

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & \text{if } IEF(U_{i,G+1}) < IEF(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \tag{5}$$

C. CFS-DE ALGORITHM FOR FEATURE SELECTION

In this section, we proposed CFS-DE algorithm for feature selection. The CFS-DE algorithm uses the correlation measure of CFS as the fitness function to evaluate the new feature subset generated, and uses the differential evolution algorithm to search for the optimal feature subset.

For feature subset S with N features, CFS calculate the evaluation of feature subset according to (1). CFS can consider both the relationship between features and label classes and between features, reducing the impact of redundant features while retaining important features. To search for the optimal subset of features from the CFS evaluation results, we use a differential evolution algorithm. The differential evolution algorithm simulates the mechanism of biological evolution in nature by generating temporary individuals based on the degree of individual variation within a population, and then randomly recombining to achieve population evolution. This algorithm is well suited to solving optimisation problems due to its good global convergence and robustness.

The process of CFS-DE is presented in Algorithm 1. The main steps of the algorithm can be seen as follows:

1) Initialization (lines 1-2). In this step, the parameters of initial population are initialized here.
2) Mutation (line 5). The new individuals $V_{i,G+1}$ are generated parent populations according to (2).
3) Crossover (line 6). In order to increase the diversity of the new population, the CFS-DE algorithm introduces crossover operation. The new individuals $X_{i,G}$ and $V_{i,G+1}$ exchange gene fragments with each other according to (3) to generate new individuals $U_{i,G+1}$.

Algorithm 1: CFS-DE Algotirhm for Feature Selection

```
    Input: Train set and Test set
    Output: Selected Feature Subset $X_{best}$
 1  Initialize population of $n$ individuals
      $X_{i,G}(i = 1, 2, \cdots, N)$;
 2  Initialize the mutation rate $F$ and crossover rate $CR$;
 3  for $G = 1$ to $G_{max}$ do
 4      for $i = 1$ to $N$ do
 5          Generate new individuals $V_{i,G+1}$ ;
 6          Generate new individuals $U_{i,G+1}$ ;
 7          Calculate the fitness $IEF(S)$ of feature subset;
 8          Generate new generation $X_{i,G+1}$ ;
 9          if $(IEF(U_{i,G+1})<IEF(X_{i,G}))$ then
10              $X_{i,G+1} = U_{i,G+1}$
11          end
12          else
13              $X_{i,G+1} = X_{i,G}$
14          end
15      end
16      $G = G + 1$;
17  end
18  $X_{best} = X_{i,G+1}$;
```

4) Fitness calculation (line 7). In this step, we use $-M_s$ as the fitness of individuals according to (1).
5) Individuals selection (lines 8-14). Based on (5), we compare the fitness of individuals to generate the next generation individuals.

D. STACKING

Stacking is a well-known ensemble learning method proposed by Wolpert [33] in 1995. Its base classifier is usually composed of different algorithms. By combining different algorithms, Stacking decreases the risk of falling into local minima and overfitting. [34]. Many ensemble learning algorithms can be regarded as variants or special cases of Stacking. The general process of stacking is to train the classifiers of the next layer and combine the output of the classifiers to generate the final results. The specific process is shown in the Fig. 2:

In the process diagram, $C_1, C_2, \cdots, C_T$ represents T base classifiers. These T classifiers are located in the first layer of Stacking, which can be the same classifier or different. After the data set is trained by classifiers, $P_1, P_2, \cdots, P_T$ represent the prediction results of the corresponding classifiers. The prediction result becomes a new training set and is provided to the second layer of Stacking. The meta classifier in the flowchart is located in the second layer of Stacking. After the predictions of the meta classifiers, the algorithm outputs the final results.

Stacking can use different classifiers or even other ensemble classifiers in the first and second layers. The learning method of Stacking is based on the capacity of various learners for features, rather than the impact of

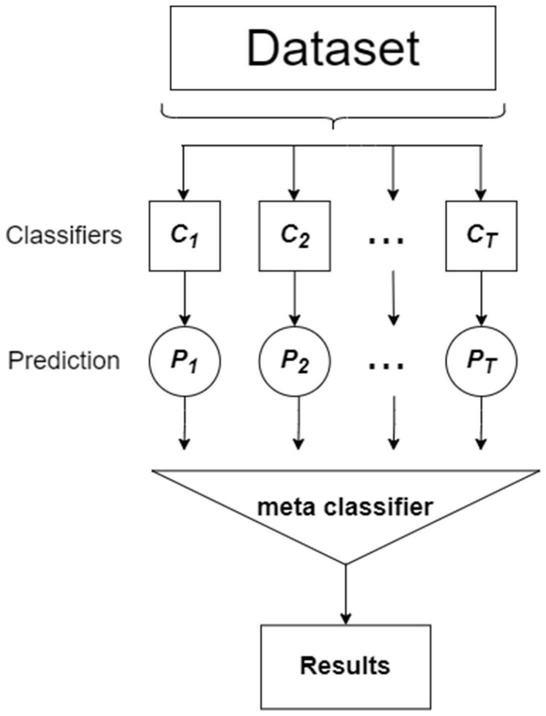


FIGURE 2. Stacking algorithm process.

multi-layer stacking. However, all base models in the Stacking algorithm have the same weights. A base model with a poor classification effect will easily affect the final classification result of the model.

E. KAPPA COEFFICIENT

The Kappa coefficient is an important parameter for testing the data consistency of experimental prediction results. The Kappa coefficient response the ratio of error reduction between classification and completely random classification. The Kappa coefficient takes into account both the accuracy and recall of the classifier on the final classification result. Suppose N is the total number of evaluated objects, n is the total number of evaluated objects, K is the number of evaluation levels, and n_{ij} is the number of levels divided by evaluation object j to evaluated object i. P_0 means the observed agreement, while P_e represent the agreement by chance. CK denotes the Kappa coefficient of the classifiers. The specific calculation formula is shown in (6) to (8).

$$CK = \frac{P_0 - P_e}{1 - P_e} \tag{6}$$

$$P_0 = \frac{1}{N}\sum_{i=1}^{N} P_i = \frac{1}{Nn(n-1)}(\sum_{i=1}^{N}\sum_{j=1}^{K} n_{ij}^2 - Nn) \tag{7}$$

$$P_e = \sum_{j=1}^{K} P_j^2 = \sum_{j=1}^{K}(\frac{1}{Nn}\sum_{i=1}^{N} n_{ij})^2 \tag{8}$$

The calculation results of Kappa coefficient are generally between -1 and 1, but usually falls between 0 and 1. It is generally divided into 5 groups to indicate the consistency of different levels:

(1) 0-0.20 (very low consistency)
(2) 0.21-0.40 (General consistency)
(3) 0.41-0.60 (medium consistency)
(4) 0.61-0.80 (high consistency)
(5) 0.81-1 (almost identical)

F. WEIGHTED STACKING MODEL

Stacking algorithm is easily affected by the classification effect of the base classifier. Since each classifier is independently trained, the classification accuracy is different, and the imbalance of the sample category will further affect the classification effect of the base classifier, thereby affecting the overall classification ability of the Stacking model. In order to solve the above problems, we proposes weighted stacking model. The main method is to evaluate the classification effects of different base models during the training phase of the base classifier. When the meta classifiers perform classification, the results of each primary classifier are weighted.

Commonly used model evaluation indicators such as accuracy rate and recall rate can only evaluate the classification effect of the model from one aspect, and cannot accurately describe the performance of the model under unbalanced data sets. Therefore, this paper uses the Kappa coefficient to evaluate the effect of model classification. The greater the Kappa coefficient value, the better the classifier's performance and the higher the agreement between predicted and actual results. Literature [35] proposed that the relationship between the weight of each independent classifier and the accuracy rate is shown in (9). W describe the weight of classifiers and P means the precision of classifiers.

$$W \propto \ln \frac{p}{1-p} \tag{9}$$

In our experiments, we replace p in (4) with the CK values corresponding to the model. We also rewrite (4) as (5) since the value of CK ranges (-1, 1) but the value of W ranges (0, 1). CK means the Kappa coefficient of classifiers. W_{ck} describes the weight of the classifiers.

$$W_{CK} = \ln \frac{1+CK}{1-CK} \tag{10}$$

According to (5), the higher the CK of the classifier, the greater the weight assigned, and the greater the impact on the final classification result. The relationship between CK value and weight is shown in Figure 3.

In this algorithm, there are a total of n classifiers, which can be the same or different classifiers. The algorithm is divided into four steps in total. The first step is to train the first-layer classifier according to the original data, and calculate the Kappa coefficient CK_i corresponding to each classifier and calculate the weight W_i according to (9) and (10). The second step is to train n models as in K-fold cross validation. The third step is to combine the outputs of each classifier

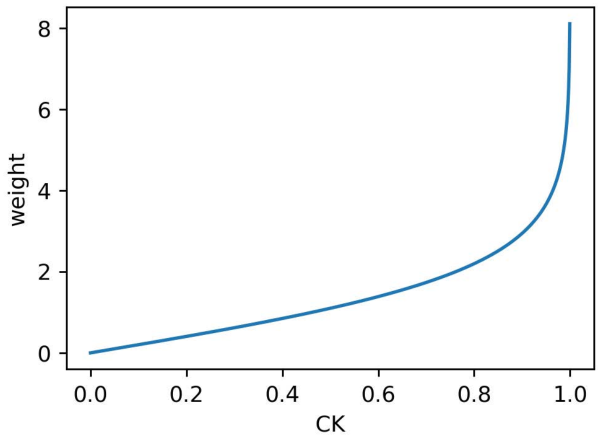


FIGURE 3. The relationship between CK and weight.

Algorithm 2: Weighted Stacking Model

```
Input: Train data: $train = \{X_i, Y_i\}_{i=1}^{m}$,Test data:
       $test = \{X_i\}_{i=1}^{m}$,Classifiers: $CLF = \{clf_i\}_{i=1}^{n}$
Output: $y_i = y_1, y_2, \cdots, y_m$
1  for $i = 1$ to $n$ do
2      fit the $clf_i$ use $train$;
3      calculate $CK_i$(Kappa coefficient) of $clf_i$;
4      calculate the $W_i$,$W_i = \ln \frac{1+CK_i}{1-CK_i}$;
5  end
6  split $train$ into k fold train=$T_1, T_2, \cdots, T_k$;
7  for $i = 1$ to $n$ do
8      for $j = 1$ to $k$ do
9          $train' = train\text{-}T_j$;
10         $clf_i$.fit($train'$);
11         $P_j = clf_i$.predict($T_j$);
12         $T_j = clf_i$.predict($test$);
13     end
14     $\bar{T}_i = average(T_j)$ $P_i' = (P_1, P_2, \cdots, P_k)^{\mathrm{T}}$
15 end
16 $Test_i' = W_1 * \bar{T}_1, W_2 * \bar{T}_2, \cdots, W_n * \bar{T}_n$;
17 $Train_i' = W_1 * P_1', W_2 * P_2', \cdots, W_n * P_n'$;
18 $meta\_classifier$.fit($Train_i'$);
19 $y_i = meta\_classifier$.predict($Test_i'$);
```

to construct new features. At the same time, the output results are weighted according to the weights calculated in the first step. The last step is to train the second-layer classifier *meta_classifier* based on the training set $Train'$ obtained in the third step, then use the trained classifier to predict the test set $Test'$, and the predicted result of the classifier is y_i. The weighted algorithm is described in Algorithm 2.

The weighted Stacking algorithm increases overall classification accuracy by applying varying weights to multiple classifiers. Despite the low performance of the base classifier, the algorithm has extremely high accuracy and produces great results in binary and multiple classification tasks.

IV. EXPERIMENT

As stated in Part III, this paper presents the CFS-DE feature selection algorithm and the weighted Stacking algorithm. In this section, to further improve the performance of the intrusion detection system classification, we propose a hybrid IDS. We use the CFS-DE algorithm to select a subset of features from high-dimensional features to improve the accuracy of classification. For the classification algorithm, we proposed the weighted Stacking algorithm. Random Forest, XGBoost and K-nearest neighbors (KNN) are selected as base classifiers and Logistic Regression (LR) is selected as meta classifier. To validate the classification effectiveness of the proposed model, we conducted experiments on the NSL-KDD, CSE-CIC-IDS2018 data set.

All the tasks are performed using the Python and Scikit-learn on Windows10 system. Matplotlib and seaborn are used to draw pictures. Cross validation is used to validate the classification performance of all models. The computer is equipped with Inter(R) CPU Core i7 9750H, Nvidia GPU GeForce GTX 1650 and 16GB RAM.

A. DATA SET

To evaluate the classification effectiveness of IDS, we chose the emulated data set NSL-KDD and the real-world packet captured data set CSE-CIC-IDS2018 for our experiments.

1) NSL-KDD

The NSL-KDD data set [36] improves the problems of the KDD-CUP [37] data set and removes duplicate instances in the train set and test set. The NSL-KDD data set increases the proportion of minority samples in the test set, which can better distinguish the classification performance of different intrusion detection models. Therefore, the NSL-KDD data set is used in this experiment to evaluate the performance of the model. The training set in NSL-KDD is called KDDTrain+ and the test set is called KDDTest+. Each record in the NSL-KDD data set has 42 attributes. 41 attributes represent the characteristic attributes of the data, and 1 attribute represents the type of the attack [38]. As is shown in Table 1, a total of 23 types of attacks are divided into five categories:

(1) Denial-Of-Service (DOS)
(2) Surveillance or probe ((Probe)
(3) User to Root (U2R)
(4) Remote to Local (R2L)

Fig. 4 shows the distribution of different types of data on the training set and test set of the NSL-KDD. As is shown in the Fig. 4, Normal and DoS samples account for the majority of the data set while the proportion of R2L and U2R samples in the data set is very low, especially in the training set. It is found that the distribution of the data is imbalanced, which may easily affect the accuracy of model classification, especially for minority samples such as R2L and U2R. However, R2L and U2R are always used by hackers, so it is significant to improve the classification accuracy of these records.

TABLE 1. Introduction of NSL-KDD.

Data types	Selected features	description
Normal	Normal	Normal record
DoS	Back,Teardrop,Smurf,Land Neptune,Pod	Denial of service attack e.g. ping of death
Probe	Nmap,Ipsweep,Portsweep, Satan,Ftp-write,Guess_passwd, Imap,Spy	Port monitoring or scanning, e.g.ping sweep
R2L	Warezclient,Warezmaster, Multihop,Phf	unauthorized access from a remote machine, e.g. guess password
U2R	Buffer_OVERFLOW,Perl, Rookit,Loadmodule	Illegal promotion of user access to root, e.g., buffer overflow attacks

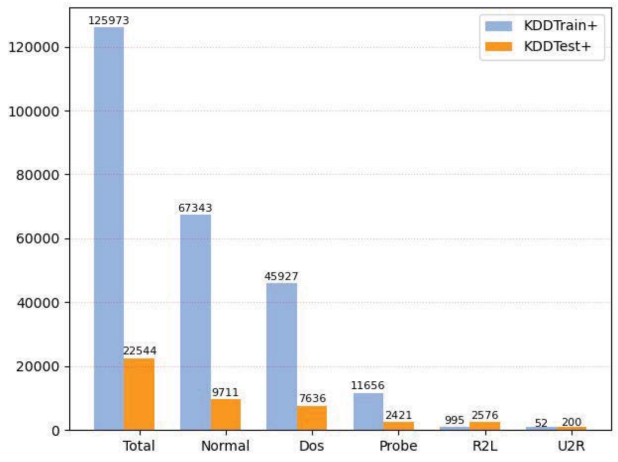


FIGURE 4. Proportional statistics of KDDTrain+ and KDDTest+.

TABLE 2. Statistics by category of NSL-KDD.

Class Label	Number	Volume(%)
Benign	1566756	57.944
DDOS	687742	25.435
FTP-BruteForce	193354	7.151
SSH-Bruteforce	187589	6.938
Infilteration	68462	2.532

2) CSE-CIC-IDS2018

The CSE-CIC-IDS2018 intrusion detection data set is a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cyber Security (CIC) [39]. The CSE-CIC-IDS2018 data set was generated by simulating the attack from 50 computers and contains a total of 16233002 records and each record includes 80 features. The data set is larger than the previous CIC-IDS2017, however, it contains many redundant features that have little relevance to intrusion detection and may affect the overall performance of the intrusion detection system.

In this study, the Wednesday-Traffic set has been chosen through 5 fold cross-validation method. This data set includes 2703903 instances in total belonging to 5 categories. The specific information of the data set is shown in Table 2

TABLE 3. Confusion matrix.

Data		Forecast class	
		yes	no
Real class	yes	TP	FN
	no	FP	TN

B. EVALUATION

In this study, confusion matrix is used for evaluating the performance of the models [40]. The matrix is shown in Table 3.

(1) True Positive (TP) - The model classifies the instances of normal traffic correctly.
(2) False Positive (FP) - The model classifies the instances of attack traffic incorrectly.
(3) True Negative (TN)- The model correctly classifies the instances of attack traffic.
(4) False Negative (FN)- The model correctly classifies the instances of normal traffic as the instances of attack traffic.

In this experiment, accuracy, precision, recall, and F1-score are used to evaluate the effect of the model on attack classification.

The accuracy is defined as the percentage of correctly predicted samples to the total sample.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

The precision represents the correct proportion of model classification.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall is the percentage of the number of samples that are correctly predicted to the total number of samples of this type.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F-score (also known as F-measure) is a commonly used indicator to evaluate the accuracy of model predictions. At present, precision and recall are often mentioned in classification-related algorithms. F-score can consider these two values at the same time to reflect the accuracy of this algorithm in a balanced manner. When the precision rate and recall rate are the same weight, F-score is called F1-score.

$$F_1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{14}$$

C. DATA PREPROCESSING

Data preprocessing can make better use of the data and improve the performance of the model [41]. Among all features of the data set, the values of some features are text, which cannot be directly used by machine learning algorithms. So one-hot encoding was used to process those features. For example, the protocol_type is expanded to three columns,protocol_TCP,protocol_UDP and protocol_ICMP. If protocol type of the record is TCP, then the value of the protocolTCP column of the record is 1, and the values of protocol_UDP and protocol_ICMP are 0. It is very common for data sets to have some features with very low variance. Generally, this part of features can not provide sufficient information for classification. Therefore, variance filtering can be used to filter out the features with variance less than the threshold, achieving the purpose of feature selection. After calculating the variance of all the features, we compared the variance with the threshold value.

Data normalization is a standard procedure before training. Different features in the data set generally have different values or ranges of value, which will affect the prediction results of the final model. In order to eliminate this influence, it is necessary to perform data normalization processing. The magnitude of the normalized data is the same, which is suitable for overall correlation analysis and evaluation. This subject intends to adopt min-max normalization, also known as dispersion normalization, which is a linear transformation of the original data and maps the result value to [0,1]. The conversion formula is as follows, where: *max* is the maximum value of the sample data, and *min* is the minimum value of the sample data.

$$x = \frac{x - min}{max - min} \tag{15}$$

The $log1p$ function is used to transform the data with larger skewness to make it more obey the Gaussian distribution [42]. At the same time when the value of x is very close to 0, the validity of the data is guaranteed.

$$f(x) = \ln(1 + x) \tag{16}$$

TABLE 4. Classification results based on all features (41 features).

Model	Accuracy	Precision	Recall	F1-score	Time(s)
RF	85.18%	86.35%	85.18%	85.76%	15.24
KNN	82.43%	82.93%	82.43%	82.69%	251.668
LR	75.55%	63.23%	75.55%	68.84%	19.53
XGBoost	84.82%	86.22%	84.82%	85.51%	11.78
AdaBoost	55.27%	32.61%	55.27%	41.02%	12.16
SGD	37.00%	45.97%	37.00%	41.00%	2.82
Stacking	86.23%	85.38%	86.23%	85.80%	287.75
our model	86.31%	88.61%	86.31%	87.44%	299.90

D. RESULTS ON NSL-KDD

In this part, we used KDDTrain+ as train set and KDDTest+ as test set. All records are divided into five categories according to different attack types.

Table 4 and Tabel 5 summarizes the classification results based on KDDTest+ data set. In comparison to the results shown in Table 4, the proposed CFS-DE feature selection algorithm got better results. The CFS-DE algorithm searches for the best subset of features from all features, reducing the number of features from 42 to 20. The CFS-DE algorithm improves accuracy, precision, recall and F1-score of models and reduces the time spent on the models. In detail, our CFS-DE-weighted-Stacking model exhibits the highest accuracy of 87.34%, precision of 89.09%, recall of 87.34%,

TABLE 5. Classification results based on selected features using CFS-DE (20 features).

Model	Accuracy	Precision	Recall	F1-score	Time(s)
RF	86.51%	88.61%	86.51%	87.55%	9.06
KNN	85.70%	80.82%	85.70%	83.19%	130.42
LR	81.53%	72.67%	81.53%	76.84%	11.95
XGBoost	86.53%	87.37%	86.53%	86.95%	5.83
AdaBoost	78.98%	80.48%	78.98%	79.72%	6.81
SGD	48.82%	55.59%	48.82%	47.35%	1.52
Stacking	86.77%	88.50%	86.77%	87.63%	160.79
our model	87.34%	89.09%	87.34%	88.21%	168.93

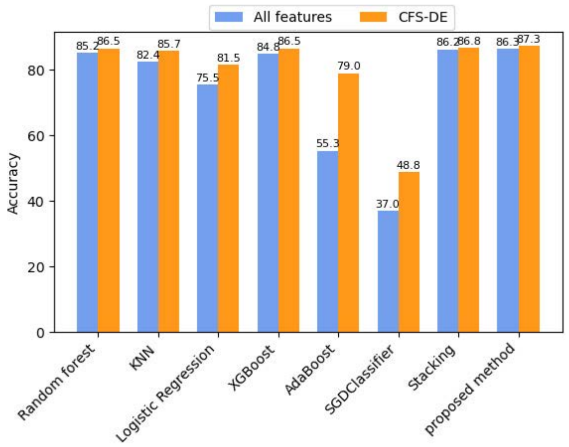


FIGURE 5. Accuracy of different models based on the KDDtest+.

F1-score of 88.21%. Furthermore, the CFS-DE algorithm searches for the best subset of features from all features, reducing the number of features from 42 to 20. Table 4 and Tabel 5 also show the time consumed by models. For KDDTest+ data set, the CFS-DE feature selection algorithm reduce over 40% of original time. We noticed that our method does have a marginal increase in the computational time when compare to other studies. However, this time increase is primarily due to the extra step of calculating the weights of base classifiers. We notice that the time cost on this extra step is quite capped and will not increase significantly with large datasets, as is shown in Table 4 and Table 5. These results fully prove that the CFS-DE feature selection algorithm improves the performance of model classification while reducing the feature dimensionality.

Fig. 5 presents the accuracy of the models and Fig. 6 presents the F1-score of the models. As illustrated in Fig. 5, the accuracy of the Random Forest, Stacking model and our proposed model is above 85% but the SGD has the lowest accuracy of all the models. Compared with other models, our proposed method achieves the highest accuracy of 87.34% among 8 models. Likewise, the findings presented in Fig. 6 illustrated that CFS-DE weighted Stacking model has the highest F1-score of 88.21%. However, SGD, AdaBoost and Logistic Regression appeared to have recorded the worst performance of F1-score. As deliberated above, it can be concluded that our proposed weighted Stacking algorithm further improves the performance by weighting the results of the base classifier.

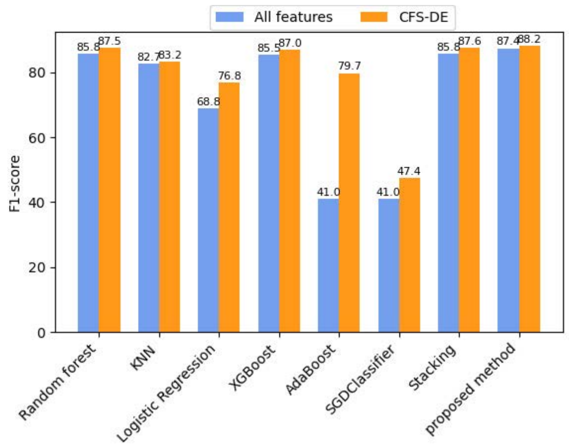
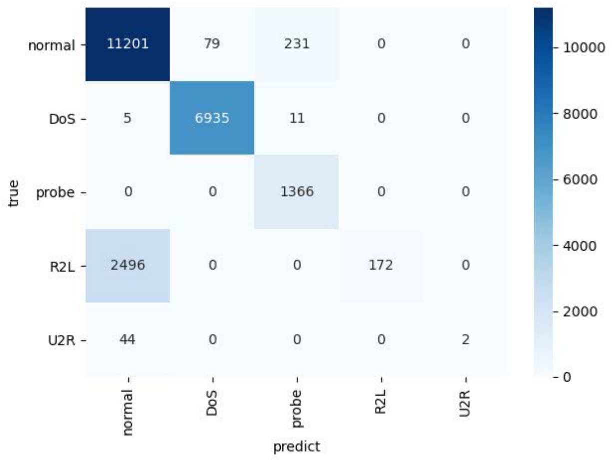


FIGURE 6. F1-score of different models based on the KDDtest+.

TABLE 6. Classification results of our model in five categories.

Type	Accuracy	Precision	Recall	F1-score
normal	97.34%	81.48%	97.34%	88.71%
DoS	99.67%	98.79%	99.67%	99.23%
probe	99.99%	85.28%	99.99%	92.06%
R2L	6.12%	99.99%	6.12%	11.66%
U2R	4.35%	99.99%	4.35%	8.25%



true \ predict	normal	DoS	probe	R2L	U2R
normal	11201	79	231	0	0
DoS	5	6935	11	0	0
probe	0	0	1366	0	0
R2L	2496	0	0	172	0
U2R	44	0	0	0	2

FIGURE 7. Confusion matrix for classification.

The experimental results are given in Table 6 and Fig. 7. In the Normal, DoS, and probe, our proposed model has a high accuracy of over 97%, covering the vast majority of records. The recall of the model is poor in the R2L and U2R. The NSL-KDD is an unbalanced dataset with far fewer R2L and U2R records than other types. The features of R2L and U2R are difficult to learn by the models. As a result, the model has a lower recall for those records. A large proportion of records incorrectly labeled as normal by the model.

We calculated the Kappa coefficients and the corresponding weights for all base classifiers according to (10). The

TABLE 7. The parameters of three base classifiers based on NSL-KDD.

Model	Accuracy	Kappa Coefficient	Weight
Random Forest	86.51%	0.7812	1.6773
XGBoost	86.53%	0.7390	1.5196
KNN	85.70%	0.6896	1.3638

TABLE 8. Comparision of the accuracy of our model with other methods.

Author	Accuracy	Precision	Recall	F1-score
Gao *et al.*[25]	85.20%	86.50%	85.20%	84.90%
Vinayakumar *et al.*[43]	85.50%	86.07%	85.50%	85.78%
Latah *et al.*[44]	84.29%	94.18%	77.18%	84.83%
Chowdhury *et al.*[45]	83.29%	86.00%	83.00%	83.00%
Mishra *et al.*[46]	82.16%	95.00%	71.00%	81.20%
Imrana *et al.*[47]	87.26%	88.81%	87.26%	88.03%
our model	**87.44%**	**89.09%**	**87.44%**	**88.25%**

TABLE 9. classification results based on all features (80 features).

Model	Accuracy	Precision	Recall	F1-score	Time(s)
RF	97.85%	97.01%	97.85%	97.42%	162.18
KNN	97.07%	97.07%	97.07%	97.07%	741.40
LR	74.86%	81.94%	74.86%	78.24%	148.77
XGBoost	98.38%	98.23%	98.38%	98.30%	126.44
AdaBoost	56.85%	48.04%	59.75%	53.26%	116.32
SGD	67.98%	76.99%	67.98%	72.20%	67.80
Stacking	98.68%	98.69%	98.68%	98.68%	757.90
W-Stacking	98.96%	98.96%	98.96%	98.96%	776.74

TABLE 10. Classification results based on the selected features using CFS-DE (15 features).

Model	Accuracy	Precision	Recall	F1-score	Time(s)
RF	98.01%	96.61%	98.01%	97.30%	29.18
KNN	98.90%	98.91%	98.90%	98.91%	134.27
LR	83.86%	88.22%	83.86%	85.99%	23.54
XGBoost	99.05%	97.36%	99.05%	98.20%	23.92
AdaBoost	71.28%	55.09%	71.28%	62.15%	32.49
SGD	75.56%	76.87%	75.56%	76.21%	11.28
Stacking	99.12%	99.22%	99.12%	99.17%	138.36
W-Stacking	99.88%	99.88%	99.88%	99.88%	144.50

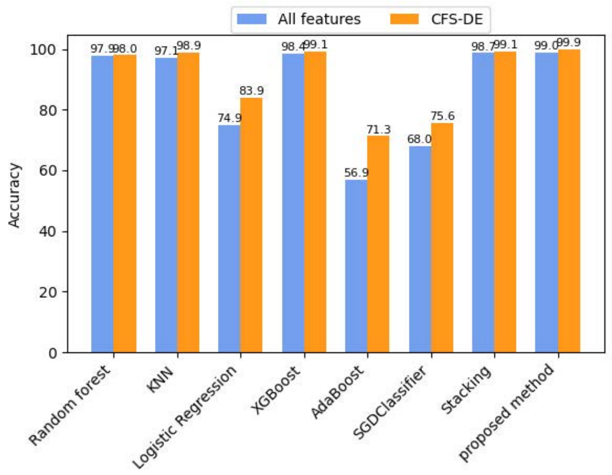


FIGURE 8. Accuracy of different models based on the CSE-CIC-IDS2018.

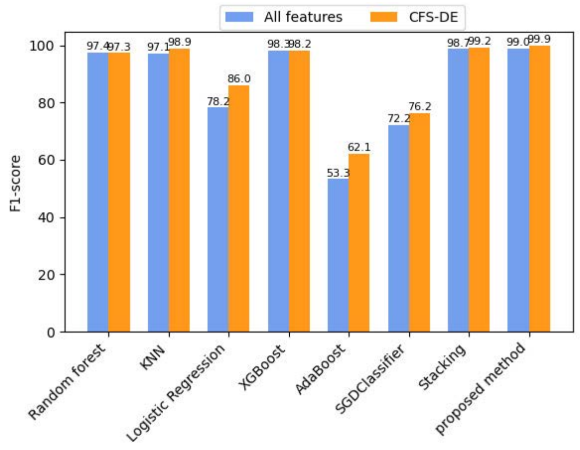


FIGURE 9. F1-score of different models based on the CSE-CIC-IDS2018.

accuracy, Kappa coefficients and weights of base classifiers are shown in Table 7.

In order to further study the classification effect of our proposed algorithm, we compare the experimental results with other studies based on KDDTest+. The comparative results and other information are shown in Table 8. It is not difficult to come to the conclusion that our model is effective for intrusion detection. Compared with other models, our CFS-DE-weighted-Stacking model gives the best results on the KDDTest+.

E. RESULTS ON CSE-CIC-IDS2018

The results of all algorithms based on CSE-CIC-IDS2018 are shown in this section. Due to the limitation of experimental conditions, we used the data of Wednesday in the CSE-CIC-IDS2018 data set to conduct experiments in this papaer. All records are divided into 5 types of attacks.

Similarly, the result on the CSE-CIC-IDS2018 data set are shown in Table 9 and Table 10. As we can see from the results, CFS-DE feature selection algorithm improves the accuracy of all models, except for the precision and F1-score of Random Forest. AdaBoost improve the accuracy about 15% and SGD improve the accuracy about 8%. Among all experimental models, the CFS-DE-weighted-Stacking approach achieves the highest accuracy rate of 99.88%, precision rate of 99.88%, recall rate of 98.88% and F1-score of 99.88%. In addition to the classification results, Table 9 and Table 10 also show the time used for classification. It is not difficult to come to the conclusion that the model after feature selection using the CFS-DE algorithm take less time compared to those using all features. Based on the classification results shown in Table 9 and Table 10, it is evident that the CFS-DE feature selection algorithm improves the performance of model while reducing the time spent by the model.

Fig. 8 indicates the accuracy of all 8 models on CSE-CIC-IDS2018. RF, KNN, XGBoost and Stacking exhibit accuracy of 97% while other models achieve accuracy of less than 85%. Our proposed model further improves the accuracy by weighting the three base models on base

TABLE 11. The parameters of three base classifiers based on CSE-CIC-IDS2018.

Model	Accuracy	Kappa Coefficient	Weight
Random Forest	98.00%	0.9743	3.6844
XGBoost	99.05%	0.9997	8.5479
KNN	98.89%	0.9848	4.1989

TABLE 12. Classification results of our model in five types.

Type	Accuracy	Precision	Recall	F1-score
Benign	99.85%	99.91%	99.85%	99.88%
FTP-BruteForce	99.99%	99.98%	99.99%	99.99%
SSH-BruteForce	99.88%	99.75%	99.88%	99.82%
DDOS	99.99%	99.94%	99.99%	99.97%
Infilteration	99.42%	99.31%	99.42%	99.36%

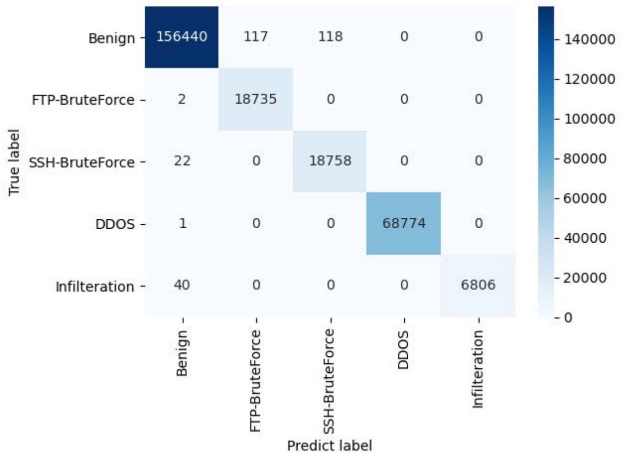


FIGURE 10. Confusion matrix based on CSE-CIC-IDS2018.

of Stacking, achieving the accuracy of 99.88%. Fig. 9 presents the F1-score of models on CSE-CIC-IDS2018. It is observed that KNN, XGBoost, Stacking and our proposed weighted Stacking outperform other models in terms of F1-score (98.91%, 98.20%, 99.17% and 99.88%). Nevertheless, AdaBoost performs the lowest f1-score value of 62.15%. These results fully prove that the weighted Stacking model improves the performance of Stacking by weighting the base model classification results, which has the best performance among 8 models.

Table 11 shows the accuracy, Kappa coefficient and weights of three base classifiers. After setting the weights of base classifiers, the meta classifier is used to classify the weighted results of base classifiers.

Table 12 and Fig. 10 represents the results of our CFS-DE-weighted-Stacking model on CSE-CIC-IDS2018. As shown by the results, the accuracy, precision, recall, and F1 score of our proposed model in the classification of 5 types of attacks are all higher than 99%. Among all types, the accuracy of Infilteration is the lowest. A small portion of Infilteration instances were misclassified as Benign.

To extend the benchmark, we compared CFS-DE-weighted-Stacking model with previous studies on CSE-CIC-IDS2018. As demonstrated by the results in

TABLE 13. Comparison results with other methods based on CSE-CIC-IDS2018.

Authors	Accuracy	Presicion	Recall	F1-score
Catillo *et al.*[48]	99.20%	95.00%	98.90%	96.91%
D'hooge *et al.*[49]	96.00%	99.00%	79.00%	87.88%
Fitni *et al.*[50]	98.80%	98.80%	97.10%	97.94%
Gamage *et al.*[51]	98.40%	97.79%	98.27%	98.03%
Hua *et al.*[52]	98.37%	98.14%	98.37%	98.25%
Lin *et al.*[53]	96.20%	96.00%	96.00%	96.00%
Zhao *et al.*[54]	97.90%	98.00%	98.00%	98.00%
our model	**99.87%**	**99.88%**	**99.87%**	**99.88%**

Table 13, the accuracy rate for all articles was above 96%. The paper [48] exhibits a high accuracy value of 99.20%, however, our model achieves higher accuracy by 0.67%. Among all models, our CFS-DE-weighted-Stacking model reaches the best performance on CSE-CIC-IDS2018.

V. CONCLUSION

With the development of the Internet technologies, intrusion detection has become increasingly important. To reduce high-dimensional features and further improve the performance of classification, we propose a hybrid framework with CFS-DE feature selection algorithm and weighted Stacking classification algorithm. First, we propose the CFS-DE feature selection algorithm to search for the best feature subset. Afterwards, the weighted Stacking classification algorithm is introduced to further improve the classification performance of IDS. In this study, we chose Random Forest, XGBoost and KNN algorithms as the base classifiers, and chose Logistic Regression as meta classifier. Finally, our proposed IDS is conducted on the NSL-KDD and CIC-IDS2018 data sets.

The results of our model for KDDTest+ provide accuracy rate of 87.44%, accuracy rate of 89.09%, recall rate of 87.44%, and F1-score of 88.25% with a subset of 20 features. It is remarkable that our model achieves the highest accuracy rate of 99.87%, precision rate of 99.88%, recall rate of 99.87%, and F1 score of 99.88% on the subset of 15 features for the CSE-CIC-IDS2018 data set. Meanwhile, the comparison with all features data set shows that our proposed CFS-DE-weighted-Stacking IDS reduces the time from 299.90s to 168.93s on KDDTest+ and from 776.74s to 144.50s on CSE-CIC-IDS2018. For future studies, the IDS can be implemented on the Hadoop platform, and MapReduce can be used for parallel detection.

REFERENCES

[1] N. Eddermoug, A. Mansour, M. Sadik, E. Sabir, and M. Azmi, "KLM-PPSA: KLM-based profiling and preventing security attacks for cloud environments: Invited paper," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2019, pp. 1–7.

[2] N. Eddermoug, A. Mansour, M. Sadik, E. Sabir, and M. Azmi, "KLM-based profiling and preventing security attacks for cloud computing: A comparative study," in *Proc. 28th Int. Conf. Telecommun. (ICT)*, Jun. 2021, pp. 1–6.

[3] A. Abusitta, M. Bellaiche, and M. Dagenais, "A trust-based game theoretical model for cooperative intrusion detection in multi-cloud environments," in *Proc. 21st Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2018, pp. 1–8.

[4] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–36, May 2019.

[5] Hasson and Timothy. *Bad BOT Report 2021*. Accessed: Apr. 13, 2021. [Online]. Available: https://www.imperva.com/blog/bad-bot-report-2021-the-pandemic-of-the-internet/

[6] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, Mar. 2017.

[7] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988–2014, Nov. 2019.

[8] I. A. N. Authority. *Protocol Assignments*. Accessed: Jun. 24, 2020. [Online]. Available: https://www.iana.org/protocols

[9] F. Erlacher and F. Dressler, "On high-speed flow-based intrusion detection using snort-compatible signatures," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 495–506, Jan. 2022.

[10] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, p. e4150, Jan. 2021.

[11] N. Krishnan and A. Salim, "Machine learning based intrusion detection for virtualized infrastructures," in *Proc. Int. CET Conf. Control, Commun., Comput. (IC)*, Jul. 2018, pp. 366–371.

[12] S. Sharma, P. Zavarsky, and S. Butakov, "Machine learning based intrusion detection system for web-based attacks," in *Proc. IEEE IEEE 6th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2020, pp. 227–230.

[13] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020.

[14] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3369–3388, 4th Quart., 2018.

[15] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: A survey," *EURASIP J. Wireless Commun. Netw.*, vol. 2013, no. 1, pp. 1–16, 2013.

[16] 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.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.

[17] P. Nimbalkar and D. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," *ICT Exp.*, vol. 7, no. 2, pp. 177–181, Jun. 2021.

[18] X. Li, P. Yi, W. Wei, Y. Jiang, and L. Tian, "LNNLS-KH: A feature selection method for network intrusion detection," *Secur. Commun. Netw.*, vol. 2021, pp. 1–22, Jan. 2021.

[19] R. M. A. Mohammad and M. K. Alsmadi, "Intrusion detection using highest wins feature selection algorithm," *Neural Comput. Appl.*, vol. 33, no. 16, pp. 9805–9816, Aug. 2021.

[20] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining Knowl. Discovery*, vol. 8, no. 4, p. e1249, Jul. 2018.

[21] D.-D. Nguyen, M.-T. Le, and T.-L. Cung, "Improving intrusion detection in SCADA systems using stacking ensemble of tree-based models," *Bull. Electr. Eng. Informat.*, vol. 11, no. 1, pp. 119–127, Feb. 2022.

[22] O. O. Olasehinde, O. V. Johnson, and O. C. Olayemi, "Evaluation of selected meta learning algorithms for the prediction improvement of network intrusion detection system," in *Proc. Int. Conf. Math., Comput. Eng. Comput. Sci. (ICMCECS)*, Mar. 2020, pp. 1–7.

[23] O. Oriola, "A stacked generalization ensemble approach for improved intrusion detection," *Int. J. Comput. Sci. Inf. Secur.*, vol. 18, no. 5, pp. 62–67, 2020.

[24] T. Zoppi and A. Ceccarelli, "Prepare for trouble and make it double! Supervised–unsupervised stacking for anomaly-based intrusion detection," *J. Netw. Comput. Appl.*, vol. 189, Sep. 2021, Art. no. 103106.

[25] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019.

[26] V. Herrera-Semenets, L. Bustio-Martínez, R. Hernández-León, and J. van den Berg, "A multi-measure feature selection algorithm for efficacious intrusion detection," *Knowl.-Based Syst.*, vol. 227, Sep. 2021, Art. no. 107264.

[27] S. Krishnaveni, S. Sivamohan, S. S. Sridhar, and S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing," *Cluster Comput.*, vol. 24, no. 3, pp. 1761–1779, Sep. 2021.

[28] D. Upadhyay, J. Manero, M. Zaman, and S. Sampalli, "Intrusion detection in SCADA based power grids: Recursive feature elimination model with majority vote ensemble algorithm," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2559–2574, 2021.

[29] P. Shunmugapriya and S. Kanmani, "Optimization of stacking ensemble configurations through artificial bee colony algorithm," *Swarm Evol. Comput.*, vol. 12, no. 12, pp. 24–32, Oct. 2013.

[30] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A stacking ensemble for network intrusion detection using heterogeneous datasets," *Secur. Commun. Netw.*, vol. 2020, pp. 1–9, Jan. 2020.

[31] M. A. Hall *et al.*, "Correlation-based feature selection for machine learning," Ph.D. thesis, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 1999.

[32] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[33] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[34] M. P. Sesmero, A. I. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using stacked generalization," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 5, no. 1, pp. 21–34, Jan. 2015.

[35] E. Menahem, L. Rokach, and Y. Elovici, "Troika—An improved stacking schema for classification tasks," *Inf. Sci.*, vol. 179, no. 24, pp. 4097–4122, Dec. 2009.

[36] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, 2013.

[37] *KDD Cup 1999*. Accessed: Apr. 2015. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[38] G. Meena and R. R. Choudhary, "A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA," in *Proc. Int. Conf. Comput., Commun. Electron. (Comptelix)*, Jul. 2017, pp. 553–558.

[39] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, vol. 1, Jan. 2018, pp. 108–116.

[40] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognit.*, vol. 91, pp. 216–231, Oct. 2019.

[41] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data preprocessing for supervised leaning," *Int. J. Comput. Sci.*, vol. 1, no. 2, pp. 111–117, Jun. 2006.

[42] H. Zhuang, X. Wang, M. Bendersky, and M. Najork, "Feature transformation for neural ranking models," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1649–1652.

[43] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1282–1289.

[44] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," *CCF Trans. Netw.*, vol. 3, nos. 3–4, pp. 261–271, Dec. 2020.

[45] R. Chowdhury, A. Roy, B. Saha, and S. K. Bandyopadhyay, "A step forward to revolutionize intrusion detection system using deep convolutional neural network," in *Data Driven Approach Towards Disruptive Technologies*. Cham, Switzerland: Springer, 2021, pp. 337–352.

[46] N. Mishra and S. Pandya, "Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access*, vol. 9, pp. 59353–59377, 2021.

[47] Y. Imrana, Y. Xiang, L. Ali, Z. Abdul-Rauf, Y.-C. Hu, S. Kadry, and S. Lim, "χ^2-BidLSTM: A feature driven intrusion detection system based on χ^2 statistical model and bidirectional LSTM," *Sensors*, vol. 22, no. 5, p. 2018, Mar. 2022.

[48] M. Catillo, M. Rak, and U. Villano, "2L-ZED-IDS: A two-level anomaly detector for multiple attack classes," in *Proc. Workshops Int. Conf. Adv. Inf. Netw. Appl.* Cham, Switzerland: Springer, 2020, pp. 687–696.

[49] L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, "Inter-dataset generalization strength of supervised machine learning methods for intrusion detection," *J. Inf. Secur. Appl.*, vol. 54, Oct. 2020, Art. no. 102564.

[50] Q. R. S. Fitni and K. Ramli, "Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems," in *Proc. IEEE Int. Conf. Ind. 4.0, Artif. Intell., Commun. Technol. (IAICT)*, Jul. 2020, pp. 118–124.

[51] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102767.

[52] Y. Hua, "An efficient traffic classification scheme using embedded feature selection and LightGBM," in *Proc. Inf. Commun. Technol. Conf. (ICTC)*, May 2020, pp. 125–130.

[53] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Proc. Int. Conf. Cloud Comput.* Cham, Switzerland: Springer, 2019, pp. 161–176.

[54] F. Zhao, H. Zhang, J. Peng, X. Zhuang, and S.-G. Na, "A semi-self-taught network intrusion detection system," *Neural Comput. Appl.*, vol. 32, no. 23, pp. 17169–17179, Dec. 2020.

RUIZHE ZHAO was born in Ningbo, Zhejiang, in 1998. He received the B.S. degree in computer science from the Zhejiang University of Science and Technology, Zhejiang, China, in 2020. He is currently pursuing the M.S. degree with the Department of Information Engineering, Hebei University of Architecture. He has published four relevant papers and obtained two software copyrights. His research interests include data mining and intrusion detection.

YINGXUE MU was born in Xingtai, Hebei, in 1989. She received the B.S. degree in computer science and technology from the Xi'an University of Posts and Telecommunications, Xi'an, China, in 2013, and the M.S. degree in computer science and technology from the University of Science and Technology Beijing, Beijing, China, in 2017. Since 2017, she has been teaching with the Hebei University of Architecture. Her research interests include big data technology and computer application. In the above fields, she participated in the preparation of one textbook, published eight relevant papers, and obtained four software copyrights.

LONG ZOU was born in Loudi, Hunan, in 1998. She received the B.S. degree in electronic commerce from Hunan International Economic University. She is currently pursuing the degree with the Hebei University of Architecture. Her research interests include network security and identification of important nodes in complex networks.

XIUMEI WEN was born in Zhangjiakou, China, in 1972. She received the B.S. degree in computer application technology from the Northwest Institute of Textile Technology, Xi'an, China, in 1995, and the M.S. degree in computer application technology from the Hebei University of Technology, Tianjin, China, in 2003. Since 2012, she has been a Professor with the Hebei University of Architecture, Zhangjiakou, China. Since 2018, she has been the Director of the Big Data Technology Innovation Center of Zhangjiakou. She has published over 30 papers in refereed journals and conferences in the above areas, authored/coauthored 12 books, and holds 20 software copyrights. Her research interests include big data technology and information processing. Her awards and honors include the Top ten teachers (Hebei University of Architecture), the Ten Outstanding Young Teachers (Zhangjiakou), Third Class Merit Award (Zhangjiakou), and Second Class Merit Award (Hebei University of Architecture).