IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 18, 2020, accepted April 17, 2020, date of publication April 21, 2020, date of current version May 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989304

WS-LSMR: Malicious WebShell Detection Algorithm Based on Ensemble Learning

ZHUANG AI[1], NURBOL LUKTARHAN[1], YUXIN ZHAO[2], AND CHAOFEI TANG[2]
[1]College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
[2]College of Software, Xinjiang University, Urumqi 830046, China

Corresponding author: Nurbol Luktarhan (nurbol@xju.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61433012, and in part by the Innovation Environment Construction Special Project of Xinjiang Uygur Autonomous Region under Grant PT1811.

ABSTRACT To solve the problem that the features produced by hidden means, such as code obfuscation and compression, in encrypted malicious WebShell files are not the same as those produced by non-encrypted files, a WebShell attack detection algorithm based on ensemble learning is proposed. First, this algorithm extracted the feature vocabulary of the unigrams and 4-grams based on opcode; subsequently, the 4-gram feature word weights were obtained according to the calculated Gini coefficient of the unigram feature words and used to select the features, which will be selected again based on the Gini coefficient of the 4-gram feature words. Consequently, a feature vocabulary that can detect encrypted and unencrypted WebShell files was constructed. Second, in order to improve the adaptability and accuracy of the detection method, an ensemble detection model called WS-LSMR, consisting of a Logistic Regression, Support Vector Machine, Multi-layer Perceptron and Random Forest, was constructed. The model uses a weighted voting method to determine the WebShell classification. This experiment demonstrated that compared with the traditional single WebShell detection algorithm, the recall rate and accuracy rate improved to 99.14% and 94.28%, respectively, which proves that this method has better detection performance.

INDEX TERMS Ensemble learning, information entropy, WebShell.

I. INTRODUCTION

With the rapid development of communication networks, Web-based applications have gradually become the main way for Internet companies to provide services to users. Meanwhile, the various types of network attacks against Web applications are also rapidly growing, which greatly threatens the security of the Internet. The 2018 *China Internet network security report* [1] issued by the *National Internet Emergency Center* (CNCERT) shows that in 2018, approximately 16,000 IP addresses at home and abroad implanted backdoors into approximately 24,000 websites in China. A network backdoor obtains system-level permissions through a WebShell, which can exist in many kinds of scripting languages. Generally, the website project root folder further extends the harm to the local area network and plants Trojans in the network to spread the virus. The attacker can use the WebShell file to access data information, such as the server database and files. For the security of a Web site, it is essential to detect the WebShell files on a server.

According to the scripting languages, WebShells can be classified as ASP, PHP and JSP scripting Trojans [2]. Because of the simple syntax and high development efficiency of the PHP language, it has become the first choice for developing all types of portals and Web applications [3]. Therefore, this paper mainly studies the detection method for PHP WebShells.

WebShells can be classified into three categories based on their functionality: Full Trojan, Mini Trojan and one-sentence Trojan. The Full Trojan, which is a general purpose WebShell, is malicious code with full functionality, which includes being interface-friendly and can be a file operation, command execution and graphical interface during database operations. The Mini Trojan contains only one function. This WebShell category can provide the file upload function using malicious database code. The one-sentence Trojan, which is a short and powerful malicious code that is difficult to detect, plays a powerful role in continuous intrusions and generally

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

takes the form of a command execution code, such as an ''eval()'' function [4]. The features of WebShell scripts are constantly changing, which makes them increasingly difficult to detect [5]. Among the various issues, the selection of the optimal feature subset has long been a concern of researchers.

In recent decades, ensemble learning algorithms have been shown to be able to efficiently solve many problems that cannot be solved by single machine learning algorithms [6]–[9]. These algorithms can address the problem of too much data or too little data. If the amount of data is too large, a single learner generally can only learn a small part [10]. Meanwhile, if the amount of data is too small, ensemble learning can sample the data set according to a certain strategy, and it consequently obtains a variety of different combinations of data to expand the data sample [11], [12]. Therefore, in the field of machine learning, researchers pay more attention to them. At first, the algorithms aimed to improve the accuracy of automatic decision-making systems, but at present, these methods can be applied to a variety of machine learning problems and can generate good results [13], [14]. Ensemble learning is a machine learning strategy that is independent of an algorithm. If a single classifier is compared to a decision maker, then multiple classifiers are equivalent to multiple decision makers making a decision together. To ensure that the ensemble classifier achieves a better classification effect than the single classifier, three principles need to be followed:

(1) Each sub-classifier of the ensemble classifier must use different classification methods and training methods,
(2) The errors produced by the sub-classifiers in the ensemble classifier must be different, and
(3) The accuracy of the sub-classifiers must be greater than 0.5.

Therefore, malicious WebShell detection based on ensemble learning encounters the following difficulties:

(1) How to select the optimal feature subset of malicious WebShell scripts,
(2) How to select basic learner in ensemble learning, and
(3) How to calculate the weight parameters of the base learner in ensemble learning.

II. RELATED WORK

At present, the WebShell detection methods can be roughly classified as static feature-based detection and dynamic feature-based detection methods.

A. DETECTION BASED ON STATIC FEATURES

Static feature detection mainly matches feature values, dangerous function names and other known conditions to find WebShells. The advantages of this approach are that it is simple to deploy, has a high rate of finding known WebShells, and can be applied using a simple script; meanwhile, the disadvantage of this method is that only known WebShells can be found. Furthermore, manual cooperation is needed to find and exclude some weak features of the files. Although a small web site can be quickly located, and files can be excluded with weak features, using a combination of static features and manual work, for large web sites, the total amount of human effort is too large at this time. Therefore, *Webshell detection techniques in web applications* [15] proposes a new method to identify WebShells based on the optimal threshold of malicious signatures, malicious function samples and the longest characters at the beginning and end of file labels. The malicious code in each file of the Web application is scanned and found, and then a list of suspect files and a detailed log analysis table for each suspect file are automatically provided by the administrator for further inspection. In *Training a multi-criteria decision system and application to the detection of PHP WebShells* [16], signatures, fuzzy hashes, whether dangerous processes are invoked, whether there are obfuscating codes, and entropy are used as features to detect and classify WebShells using an algorithm that trains a multi-criteria decision system. *CNN-Webshell: Malicious Web shell detection with convolutional neural network* [17] proposed a feature extraction method based on ''word2vec''. First, the word2vec tool was used to transform each word of an HTTP request into a vector, and then it converted the vectors into a fixed sized matrix. Finally, a detection method based on the CNN model was used for classification.

B. DETECTION BASED ON DYNAMIC FEATURES

Detection based on dynamic features is a method that detects the features of the WebShell execution process. This approach is good at detecting the features generated by operations, such as code annotation and code compression; however, feature extraction and feature dimensionality reduction still represent problems. For example, in *A Method of Detecting Webshell Based on Multi-layer Perception* [18], the sample source code is converted into bytecode by compiling tools, and then the sample bytecode is decomposed into bytecode sequences by Bi-Gram. Next, the feature matrix of the training set is set using the word frequency matrix calculated by the TF-IDF. Finally, a multilayer neural network is used to detect and classify WebShell files. *Webshell detection based on random forest–gradient boosting decision tree algorithm* [19] used the features of the opcode sequences extracted from PHP source files, then used the TF-IDF vector and hash vector for feature selection, and finally used the combination of the random forest classifier and GBDT classifier for classification. *Detecting webshell based on random forest with fasttext* [20] first used the VLD tool in PHP to obtain the opcode sequences of PHP files. Then, this method used the FastText algorithm to train the opcode sequence model and predict the corresponding feature values and combined the predicted feature values and static features as the features of samples. Finally, the random forest was used to realize binary classification.

C. WebShell ATTACK DETECTION BASED ON ENSEMBLE LEARNING

The above two detection methods have improved the detection effect of WebShell to a certain extent, but they still

TABLE 1. Difference between static and dynamic detection of WebShell attacks.

Detection mode	Main detected contents	Advantages	Disadvantages
Static detection	Document statistical feature, feature code, syntax analysis	High accuracy of checking and killing known and partially deformed WebShell	Easy to false positives and can not detect new unknown WebShell
Dynamic detection	Detect behavior characteristics of executed Webshell files	Having a certain ability to detect the new variant script	Need to maintain a large behavior characteristics library

have shortcomings, which are summarized in the following Table 1.

To overcome the shortcomings of the above dynamic detection Methods and expand its advantages, this paper proposes using the ensemble learning of binary weighted voting to classify WebShells.

At present, ensemble learning can be roughly classified into three categories: Bagging, Boosting, and Stacking. Bagging is a method that extracts data from the original data set and conducts model training and prediction to obtain K models. Finally, this approach uses these models to predict the data. K predicted values can be obtained from each sample, and the final result can be obtained by voting [21]. Because the weights of the base learners in this algorithm are the same, the base learner selection in this algorithm will directly affect the results of the ensemble learning method. Boosting mainly assembles weak classifiers into a strong classifier. In this instance, AdaBoost will pay more attention to the mislearned samples when learning the algorithm, increase their respective weights, and then superimpose the models generated in each step to obtain the final model [22], [23]. Xgboost is a lifting tree model that can integrate many tree models together to form a strong classifier [24], [25]. However, the disadvantage of this algorithm is that it is easy to overfit due to noise. Stacking first trains the base learner using the initial data set, then combines the data generated by the base learner into a new data set, then inputs that new data set into the classification algorithm, and then finally obtains the final prediction result [26], [27]. The disadvantage of this algorithm is as follows: if the training set of the primary learner is directly used to generate the secondary training set, there is a great risk of overfitting.

A more important classification method is the weighted voting method, which combines the above methods to adaptively allocate weights. Given multiple base classifiers, high weights are assigned to the algorithms with high accuracy, which can better reflect the roles of excellent algorithms [28].

According to the above analysis, this paper will apply the weighted voting algorithm of the binary model to the detection of WebShell files. Compared with other ordinary single machine learning algorithms, this algorithm performs better in terms of its recall rate and accuracy.

D. TYPES OF GRAPHICS

In this paper, the feature vocabulary lists of unigrams and 4-grams are extracted from the sample set according to the opcode. Then, the 4-gram feature words are calculated based on the feature weight values calculated by the Gini coefficients of the unigram feature words, and the selected features are selected again according to the Gini coefficients of the 4-gram feature words. This approach will allow one to construct a feature vocabulary to detect encrypted and unencrypted WebShell files, consequently solving the problem that it is difficult to detect the script and select the optimal feature subset. Second, in order to improve the adaptability and accuracy of the detection method, this paper constructs a differentiated ensemble detection model WS-LSMR, which is composed of a Logistic Regression, Support Vector Machine, Multi-layer Perceptron and Random Forest. The advantages and disadvantages of each algorithm are shown in Table 2. In addition, this model uses the weighted voting method to determine the classification of WebShells.

The main contributions of this article are the following two aspects.

(1) Based on the original TF-IDF feature vectorization, the 4-gram feature words are weighted according to the feature weights calculated using the unigram feature words via the random forest, which will select features for the first time; and then the 4-gram feature words can be selected for the second time via the random forest. This algorithm can select the optimal feature subset that reduces the dimension and improves the efficiency of the algorithm.

(2) Based on the accuracy of the model training, the voting weights of each algorithm in the ensemble detection model are set to improve the detection effect.

III. MALICIOUS WebShell DETECTION ALGORITHM BASED ON ENSEMBLE LEARNING

A. SYSTEM ARCHITECTURE

This article uses an ensemble learning algorithm, which is called the WS-LSMR, and its structure is illustrated in Fig. 1 and Table 3. This research can be divided into three modules: preprocessing, feature selection, and model building and prediction.

Preprocessing: First, all files contained in the data set should be de-duplicated to prevent interference in the detection results. Opcode is used to preprocess the features that are obtained by splitting or encrypting dangerous functions. Second, the samples were divided into training samples and test samples at a ratio of 4:1. Finally, the training set is vectorized according to the unigrams and 4-grams, respectively, and the test set is vectorized according to the 4-grams.

Feature selection: First, the 4-gram feature word weights are obtained according to the calculated Gini coefficients of the unigram feature words to select the features. The random forest algorithm is applied to the training samples to select the important features. The training set is sampled using SMOTE

TABLE 2. Advantages and disadvantages of single classification algorithm.

#	Advantages	Disadvantages
LR	The processing of redundant features is relatively stable. The model has a great generalization ability.	It is difficult to deal with the data imbalance problem. It is more difficult to deal with non-linear data.
SVM	It can deal with high-dimensional data sets well, and it has a strong generalization ability. It is suitable for solving small samples and nonlinear problems.	It is sensitive to missing data and it is easy to produce overfitting in multiple classification problems.
MLP	It is more robust when dealing with a nonlinear model.	Many super parameters need to be adjusted, and it is sensitive to feature scaling.
RF	It is robust to missing data and unbalanced data, and it can tolerate noise and outliers well. It is suitable for both high-dimensional large and small samples.	The model is not easy to explain. If there are many divisions of a certain feature, then it has a greater impact on the decision-making of the random forest.

TABLE 3. Experimental structure.

Input: Data (the dataset), Base_classifier=[L1, L2, L3, L4],feature selection algorithm S.
Output: Test set prediction for each sample T = { T1, T2, ..., TN }
Step 1: Use algorithm 1 to conduct file processing.
Step 2: Divide the data_process data into the training set data_train and the test set data_test at a 4:1 ratio.
Step 3: Data_train_process is obtained by using TfidfVectorizer on the training set data_train.
Step 4: Data_test_process is obtained by using the TfidfVectorizer on the testing set data_test.
Step 5: Use algorithm 2 to select the important features.
Step 7: Use algorithm 3 to calculate the weight values.
Step 8: Calculate the weight value of the base classifier.

$$\lambda i = \frac{L\_score[i]}{(\sum_{i=1}^{4} L\_score[i])} + (L\_score - \frac{\sum_{i=1}^{4} L\_score[i]}{T} * n) \quad (1)$$

In this formula, T is the number of base classifiers, and n is the parameter that represents the gap between the good and bad algorithms.
Step 9: Algorithm 4 predicts the probability value of the test set.
Step 10: Test Set Selection Process
The prediction probability value prob of each sample in the test set is compared with the threshold value of 0.5 to determine the category T of the final test sample.

to prevent the prediction result from being one-sided because of the large positive and negative sample gap.

Model building and prediction: Four base classifiers (Logistic Regression, Support Vector Machine, Multi-layer Perception, and Random Forest) are used to train and verify the training samples, which will determine the accuracy of the corresponding base classifier and the prediction probability of each test sample based on the classifier. Then, the accuracy rate of the set is verified and used to obtain the weight value of the base classifier, and finally it and the prediction probability are used to obtain the final classification probability of the test set.

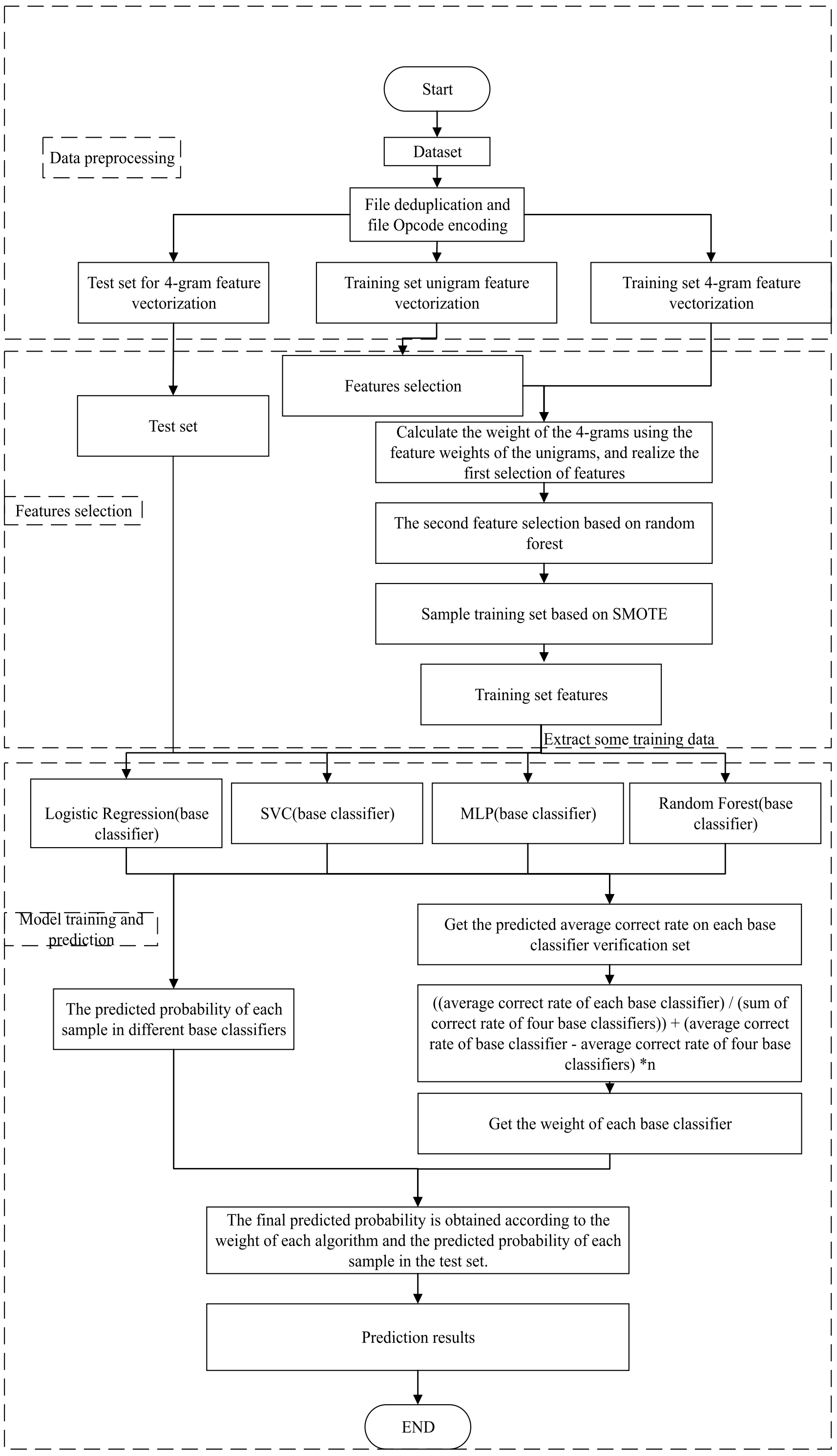
Suppose data= $(X_i, Y_i)_{i=1}^{N}$is a WebShell dataset, where N is the number of the samples and each training sample Xi has a corresponding target value Yi. Base_classifier= $[L1, L2, L3, L4]$ is used to represent the base classifier, $\lambda = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ represents the weight value of the base classifier and test_prob= $\{probi1, probi2, probi3, probi4\}_{i=1}^{N}$ represents the prediction probabilities of the ith sample for the four base classifiers.

Algorithm 1 File Processing
```
Input: sample data set : $dataSet$.
Output: opcode compiled file : $data\_process$.
1: The dataSet uses the MD5 function to de-duplicate the file and gets the intermediate result dataProcess.
2: dataProcess is compiled using the Zend engine to get the opcode compiler result data_process.
3: return $data\_process$.
```

Algorithm 2 Feature Selection
```
Input: feature selection algorithm : algorithm, unigrams Feature : one_gram_feature, 4-grams feature : four_grams_feature, unigrams threshold selection parameter: num1, 4-grams threshold selection parameter : num2.
Output: feature lists : feature_lists.
1: By feature selection algorithm "algorithm" and one_grams_feature, calculates the importance score of the unigram feature vocabulary: one_feature_score.
2: one_feature_score and four_grams_feature were used to calculate the total score value of each 4-gram Feature: four_score.
3: Obtaining four_score, the retained feature sequence, by implementing the first dimension reduction of the feature according to num1 and four_score.
4: By feature selection algorithm and four_score, calculates the importance score of the 4 - grams feature vocabulary: four_feature_score.
5: The second dimension reduction of the feature is implemented with num2 and four_feature_score to obtain the remaining feature sequence: feature_list.
6: Sample on feature_list dataset with SMOTE method to get the dataset that has been feature processed.
7: return $feature\_list$.
```

B. OPCODE

Opcode is an intermediate code in the PHP language for the Zend [29] engine to execute, and it is similar to the bytecode file in Java or pycodeobject in Python. The resulting Opcode bytecode file can be directly executed; therefore, the split dangerous functions or encrypted functions



Start
Data preprocessing
Dataset
File deduplication and file Opcode encoding
Test set for 4-gram feature vectorization
Training set unigram feature vectorization
Training set 4-gram feature vectorization
Features selection
Test set
Calculate the weight of the 4-grams using the feature weights of the unigrams, and realize the first selection of features
Features selection
The second feature selection based on random forest
Sample training set based on SMOTE
Training set features
Extract some training data
Logistic Regression(base classifier)
SVC(base classifier)
MLP(base classifier)
Random Forest(base classifier)
Model training and prediction
Get the predicted average correct rate on each base classifier verification set
The predicted probability of each sample in different base classifiers
((average correct rate of each base classifier) / (sum of correct rate of four base classifiers)) + (average correct rate of base classifier - average correct rate of four base classifiers) *n
Get the weight of each base classifier
The final predicted probability is obtained according to the weight of each algorithm and the predicted probability of each sample in the test set.
Prediction results
END

FIGURE 1. **Flow chart of ensemble learning algorithm.**

in the PHP file will also be reflected in the Opcode, and they will still appear the same Opcode statement code as the compilation result of the normal file when compiled.

At present, there are two kinds of malicious WebShells: encrypted files and non-encrypted files. The features that are directly extracted from the two types of files are different, but the features of the two PHP files after opcode encoding are completely consistent. Therefore, the first step for the data sample is to code the malicious files using opcode to obtain the feature set that can detect malicious WebShell files. In addition, you can identify WebShells using code

Algorithm 3 Calculation of Weight Value

```
Input: training set feature : x_train_data, training set target
    value : y_train_data, test set sample : x_test, base classifier list : Base_classifier, cross validation parameter :
    kfold_num, Sample m% from the training set : m.
Output: the accuracy of each base classifier : L_score, the
    predicted probability value of each test sample in the
    base_classifier : test_prob.
1: for $i \to lenth(Base\_classifier)$ do
2:     Random sampling of m% of data from X_train_data
       and y_ train_data yields : X_trains_list, y_trains_list.
3:     Cross verify X _train_list and y_train_list with kfold
       _ num fold to obtain the average score L_score [i] of
       Base_classifier[i].
4:     All samples X_test of the test set get the predicted
       probability value test_prob[i] on the Base_classifier[i].
5: end for
6: return $L\_score, test\_prob$.
```

Algorithm 4 Prediction Probability

```
Input: Base classifier weights : λ, The predicted probability
    value of each sample in the test set on the base classifier:test_prob.
Output: The final prediction probability value for each sample:probably.
1: Each sample calculates the final prediction probability
   value "probably" by the weight value "λ" of the base
   classifier and the corresponding prediction probability
   value "test_prob".
2: return $probably$
```

compression, code splitting, and code comments based on this feature.

VLD(Vulcan Logic Dumper) is an extension tool implemented as a hook in the Zend engine for outputting intermediate code (execution units) generated by PHP scripts, so this paper uses the VLD extension to analyse the following malicious PHP file

```
<? PHP call_user_func (create_function (null, 'assert
($_POST[C]);'));? >
```

and the annotated malicious PHP file

```
<? PHP call_user_func (create_function (null,
'assert/* annotation */ ($_POST[C]);'));? >
```

The resulting intermediate opcode is given in Table 4.

TABLE 4. Opcode compiled files.

#	OPCODE
1	SEND_VAL
2	SEND_VAL
3	DO_FCALL
4	SEND_VAR_NO_REF
5	DO_FCALL
6	RETURN
7	NOP
8	RETURN
9	FETCH_CONSTANT
10	FETCH_R
11	FETCH_DIM_R
12	SEND_VAR
13	DO_FCALL
14	RETURN

C. FEATURE VECTORIZATION

The TF-IDF (term frequency-inverse document frequency) is a weighting technique that automatically extracts text keywords [30] and is widely used in text classification for feature vectorization [31]. The TF-IDF uses the IDF (inverse document frequency) to determine the word frequency. By calculating the frequencies of words and the inverse document, the algorithm can effectively identify those invalid words with high word frequency but no actual meaning. Therefore, this algorithm also improves the simple word frequency statistics method [32].

This paper uses the TF-IDF to carry out the feature vectorization of the PHP file, and the basic process is as follows. PHP files are called a sample after being encoded using opcode, and each sample is a segment after being split by the n-grams word bag. TF is the number of times that segment X appears in a given sample. The IDF reflects the frequency of this segment in all samples, and its calculation formula is as follows:

$$IDF(x) = log\frac{N}{N(x)} \tag{2}$$

N is the total number of texts in the corpus, and N(X) is the number of texts containing the corpus segment X. The TF_IDF (X), which measures the importance of the corpus segment X, can be obtained using the TF (word frequency) and IDF (reverse file frequency) according to formula 3.

$$TF\_IDF(x) = TF(x) * IDF(x) \tag{3}$$

D. FEATURE SELECTION

Because most of the opcode features can only use a small part of the words in the vocabulary, this property will lead to the sparseness of the word vector. Therefore, it is necessary to eliminate the unimportant features to prevent the computational efficiency from decreasing due to feature explosion. At present, the main algorithm that can be selected for calculating the importance of features is the decision tree model. In it, each feature calculates the incremental value of the Gini coefficient according to formula 4, and it finally selects the important features based on the incremental value order.

The Gini coefficient is a measure of the impurity of data:

$$GINI(D) = 1 - (\sum_{i=1}^{m} (p_i)^2) \tag{4}$$

M represents the number of type C entries in dataset D, and p_i represents the probability that any sample in D belongs to Ci.

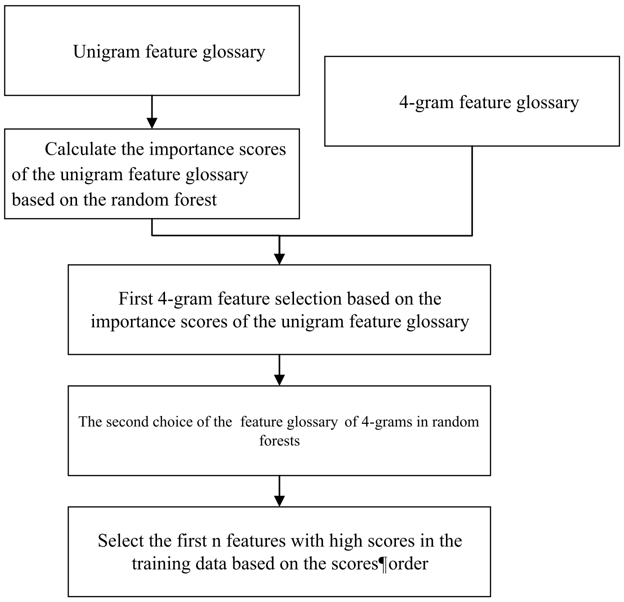

Unigram feature glossary
4-gram feature glossary
Calculate the importance scores of the unigram feature glossary based on the random forest
First 4-gram feature selection based on the importance scores of the unigram feature glossary
The second choice of the feature glossary of 4-grams in random forests
Select the first n features with high scores in the training data based on the scores¶order

FIGURE 2. Feature selection execution flow.

The Gini coefficient after splitting by attribute a is calculated as follows:

$$Gini_a = \frac{D_1}{D}Gini(D_1) + \frac{D_2}{D}Gini(D_2) \tag{5}$$

where D_1 is a non-void proper subset of D; D_2 is a complement of D_1 in D, that is, $D_1 + D_2 = $ D; and the minimum value is selected as the Gini coefficient of feature a.

The feature can be indicated much more important when its increment of the feature is larger and the ability to distinguish black and white lists is more stronger.

$$GINI(a) = GINI(D) - Gini_a(D) \tag{6}$$

The specific steps of feature selection are as follows:

(1) Calculate the importance score for the unigram feature words using the random forest;
(2) The 4-gram feature words are selected for the first time according to the score values;
(3) The feature importance score of the 4-grams is calculated using the features selected at the first time to carry out the second selection of the features; and
(4) Finally, select the important features that meet certain important conditions.

The features obtained above are the optimal feature subset, which improve the algorithmic efficiency while ensuring the accuracy and recall rate. The feature selection execution process is shown in Fig. 2, and Table 5 shows the top 10 most important features.

E. TRANSFORMATION OF THE WEIGHTS OF UNIGRAM AND 4-GRAMS FEATURE WORDS

In the process of TF-IDF vectorization, 4-grams features are combined based on unigram features such that each 4-grams feature can be linked to a corresponding single unigram feature. In Fig. 3, for the 4-grams feature "ABCD", the weight value of this feature word is the simple addition of the unigram weight values of feature A, feature B, feature C, and feature D.

TABLE 5. Ensemble algorithm for selecting important features.

#	Important 4-gram features
1	send_val, do_fcall, send_var_no_ref, do_fcall
2	begin_silence, fetch_r, init_fcall_by_name, send_val
3	assign, begin_silence, fetch_r, init_fcall_by_name
4	init_fcall_by_name, send_val, do_fcall_by_name, include_or_eval
5	do_fcall, send_var_no_ref, do_fcall, include_or_eval
6	send_val, do_fcall_by_name, include_or_eval, end_silence
7	fetch_r,init_fcall_by_name,send_val,do_fcall_by_name
8	do_fcall_by_name, include_or_eval, end_silence, return
9	assign, assign, assign, send_val
10	send_var_no_ref, do_fcall, include_or_eval, return

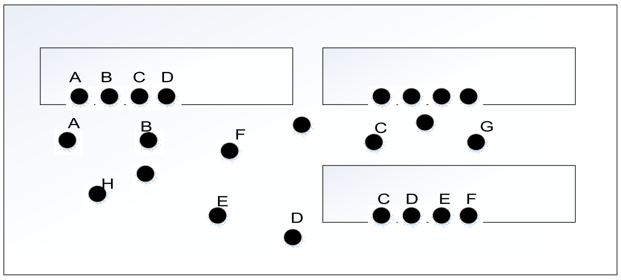


FIGURE 3. 4-grams calculation method.

IV. EXPERIMENT

A. EXPERIMENTAL CONDITIONS

This experiment is based on Python version 3.5.4, the experimental environment uses the win10 64-bit operating system, the processor is an Intel (R) Core (TM) i3-7130U CPU @ 2.70 GHz, and there is 8G of memory.

B. EXPERIMENTAL DATA

The malicious WebShell samples are mainly downloaded from GitHub public projects. Since this study only performs offensive detection for PHP files, the total number of malicious samples is 566. The normal PHP samples mainly come from common PHP frameworks, including PHPCMS, WordPress, Fenxiangyo, oa and yii2. There are a total of 5,379 samples. The data sources are shown in Table 6.

C. EVALUATION CRITERIA

This paper evaluates the WebShell detection method based on ensemble learning using the recall rate, accuracy rate and specificity. The model evaluation confusion matrix is shown in Table 7.

A true positive is an outcome where the model correctly predicts the positive class (True Positive = TP).

A false negative is an outcome where the model incorrectly predicts the negative class (False Negative = FN).

TABLE 6. Sample distribution.

Sample	Name	Source
Normal Samples	PHPCMS	https://github.com/johnshen/-PHPcms
	WordPress	https://github.com/WordPress
	Fenxiangyo	https://github.com/learnstartup/-4tweb
	yii2	https://github.com/yiisoft/yii2
	oa	https://github.com/rainrocka/-xinhu
Malicious samples		https://github.com/tennc/-WebShell
		https://github.com/tanjiti/-WebShell-Sample
		https://github.com/JohnTroony/-PHP-WebShells

TABLE 7. Confusion matrix.

Reality/Prediction	0	1
0	TN	FP
1	FN	TP

TABLE 8. Recall rate, accuracy and specificity comparison of the different *n*-grams.

N-grams	acc	Recall	Specificity
1	0.955425	0.853448	0.966449
2	0.967199	0.922414	0.972041
3	0.96804	0.965517	0.968313
4	**0.941968**	**0.991379**	0.936626
5	0.922624	0.982759	0.916123
6	0.893188	0.974138	0.884436

A false positive is an outcome where the model incorrectly predicts the positive class (False Positive = FP).

A true negative is an outcome where the model correctly predicts the negative class (True Negative = TN).

The Recall = TP/ (TP + FN), which represents the proportion of correct predictions and measures the recognition ability of a classifier for positive cases (WebShell) with respect to all the positive results.

The accuracy rate is calculated as (ACC) = (TP + TN)/ (TP + FN + FP + TN), which represents the proportion of samples that are correctly classified by the model's prediction.

The specificity = TN/ (TN + FP), which represents the proportion of negative samples that are correctly predicted by model and measures the recognition ability of the classifier for negative cases (normal PHP files) with respect to all negative results.

D. FEATURE SELECTION

An opcode sequence is more representative than a single opcode. It is necessary to find the best performance for the range from unigrams to 6-grams and, finally, use the feature vector to conduct filtering to reduce the feature dimension.

Fig. 4 and Table 8 show that during the transition from unigrams to 4-grams, the recall rate increases as the grams

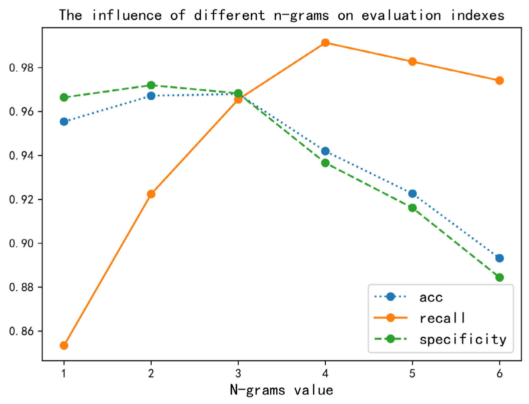


FIGURE 4. Recall rate, accuracy and specificity comparison of different n-grams.

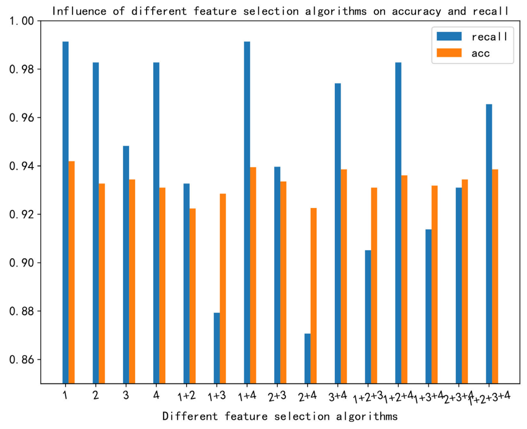


FIGURE 5. Comparison of different feature selection algorithms.

increase, and the model is better able to distinguish between black and white list samples. The recall rate decreases when n_grams > 4 because of fewer occurrences in the WebShell file, thereby creating an invalid presentation vector, and the recall rate reaches its maximum at 4-grams. Therefore, 4-grams are optimal for the final training and testing.

The experimental comparison of different feature selection algorithms is shown in Fig. 5. Random forests are more suitable for training and testing samples than other algorithms in terms of the recall and accuracy. In Fig. 5, "1" represents the "RandomForestClassifier" algorithm, "2" represents the "AdaBoostClassifier" algorithm, "3" represents the "GradientBoostingClassifier" algorithm, and "4" represents the "XGBClassifier" algorithm.

The experimental data of the thresholds selected for the first time are shown in Fig. 6 and Table 9. In Table 9, "len" represents the number of remaining features selected according to the threshold from the original 15,000 features. Figure 6 shows that when the threshold value is between 0.01 and 0.03, the three evaluation indexes linearly increase, which increase as the number 4-gram features increases. The three evaluation indexes were in a state of decline between 0.03 and 0.07. At this time, the value decreased as the number of 4-gram features increased. The main reason was that some

TABLE 9. Threshold selection in the first dimension reduction of features.

Threshold	len	acc	Recall	Specificity
0.01	14973	0.931876	0.905172	0.934762
0.02	14712	0.932717	0.922414	0.93383
0.03	13643	**0.942809**	**0.991379**	0.937558
0.04	12430	0.941127	0.982759	0.936626
0.05	11319	0.940286	0.982759	0.935694
0.06	9716	0.906644	0.974138	0.899348
0.07	7907	0.89487	0.862069	0.898416
0.08	6219	0.861228	0.965517	0.849953
0.09	4398	0.963835	0.801724	0.981361

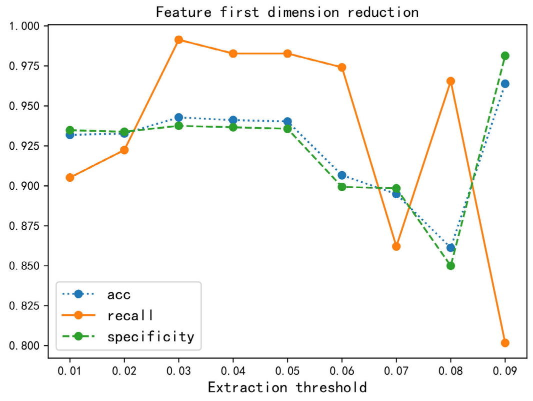


FIGURE 6. Threshold selection in the first dimension reduction of features.

impure features were added to the feature set, which affected the final detection effect. The recall rate, accuracy rate and specificity reach their respective peaks between 0.08 and 0.09. At this time, the increase of the number of features will increase the detection index in a single direction, while the overall detection index will decrease. Therefore, the threshold of the dimension reduction selection of the first feature is 0.03, which is the best detection standard.

E. BINARY WEIGHTED VOTING MODEL

Binary weighted voting is a more effective way to deal with classification problems, and it is a highly direct method, primarily because an algorithm with good classification performance receives a high weight. In addition, the voting results can often use the complementary between single classification models to reduce the error of a single classifier and improve the prediction performance and classification accuracy.

In this paper, to make the difference of single classifier more obvious, each classifier randomly extracts part of the data from the training set. Table 10 and Fig. 7 show that after the percentage of extracted training data reaches 80%, the recall rate and accuracy rate reach their peaks. This finding indicates that the data are saturated at this time. If the input of the training set continues to increase, the difference between the base classifiers will worsen, and the accuracy rate and recall rate will both be reduced. Therefore, the extraction of 80% of the training data is optimal for the final training effect.

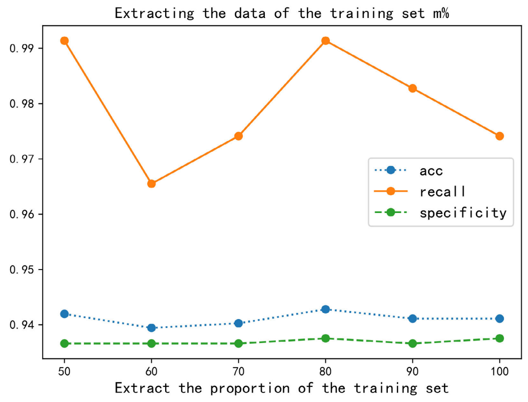


FIGURE 7. Effects of extracting m% data from the training set on recall rate, accuracy rate and specificity.

In the algorithm's weighted voting process, suppose the ensemble learning contains T learners $\{h_1, h_2, \ldots, h_i, .., h_T\}$, where the output of h_i on example x is $h_i(x)$. The learner h_i will predict a tag from the class tag set $\{c_1, c_2, c_3, c_4, \ldots, c_N\}$,where the predicted output of h_i on sample x is expressed as an n-dimensional vector $(h_i^1(x), h_i^2(x), h_i^3(x), \ldots, h_i^j(x), \ldots, h_i^N(x))$, where $h_i^j(x)$ represents the output of the classifier h_i on the category marker c_j.

Plurality voting

$$H(x) = c_{argjmax}\left(\sum_{i=1}^{T} h_i^j(x)\right) \tag{7}$$

In other words, the mark with the most votes is the predicted one, and if multiple marks receive the most votes at the same time, one of them is randomly selected.

Weighted voting (WV)

If each classifier also has a weight value w_i,

$$H(x) = c_{argjmax}\left(\sum_{i=1}^{T} w_i h_i^j(x)\right) \tag{8}$$

The specific steps of ensemble binary weighted voting are as follows.

(1) Calculate the average accuracy P_avg_i of each classifier using 5-fold cross-validation and use this model to predict the prediction probability of the test set P_test_i, where i represents the classifier.

(2) Calculate the weight value of each classifier. The formula is as follows:

$$\lambda i = \frac{P_{avg_i}}{\sum_{i=1}^{4} P_{avg_i}} + (P_{avg_i} - \frac{\sum_{i=1}^{4} P_{avg_i}}{T}) * n \tag{9}$$

where i represents the number of classifiers, T represents the number of classifiers, and n represents the gap between the good and bad algorithms for detecting WebShells.

(3) Calculate the probability of each sample in the test set.

$$p\_test = \sum_{i=1}^{T} \lambda i * p\_test\_i \tag{10}$$

TABLE 10. Effects of extracting m% data from the training set on recall rate, accuracy rate and specificity.

m%	acc	Recall	Specificity
50	0.941968	0.991379	0.936626
60	0.939445	0.965517	0.936626
70	0.940286	0.974138	0.936626
80	**0.942809**	**0.991379**	0.937558
90	0.941127	0.982759	0.936626
100	0.941127	0.974138	0.937558

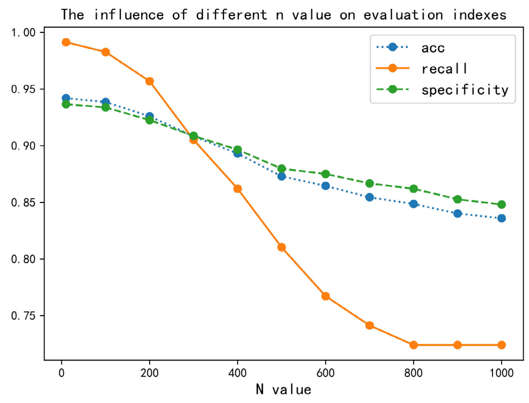


FIGURE 8. The influence of different values of n on the evaluation indexes.

TABLE 11. The influence of different values of *n* on the evaluation indexes.

N	Recall	acc	Specificity
10	**0.991379**	**0.941968**	0.936626
100	**0.982759**	**0.938604**	0.93383
200	**0.956897**	**0.925988**	0.922647
300	0.905172	0.908326	0.908667
400	0.862069	0.893188	0.896552
500	0.810345	0.873003	0.879776
600	0.767241	0.864592	0.875116
700	0.741379	0.8545	0.866729
800	0.724138	0.848612	0.862069
900	0.724138	0.840202	0.852749
1000	0.724138	0.835997	0.848089

(4) The final classification result is obtained by comparing the probability p_test with the threshold value 0.5.

In step 2, the range of n that increases the difference between base classifiers is generally not large. If the range is too large, the prediction probability of the final sample is likely to be greater than 1. Therefore, this experiment is conducted where n ranges from 10-1000. The experiment is divided into three steps to select the most appropriate n that increases the difference between the base classifiers.

(1) Different values of n are tested separately. In the first step of the experiment, n ranges from 0 to 1000. The specific values are 10, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. The experimental results are shown in Table 11 and Fig. 8.

(2) It is found that the best range for the value of n in Figure 8 is between 0-200. In the second step of the experiment, n ranges from 0 to 200. The specific values are 10, 20, 40, 60, 80, 100, 120, 140, 160, 180 and 200.

TABLE 12. The influence of different values of *n* on the evaluation indexes.

N	Recall	acc	Specificity
10	0.991379	0.941968	0.936626
20	0.991379	0.941968	0.936626
40	0.982759	0.941968	0.937558
60	0.982759	0.939445	0.934762
80	0.982759	0.938604	0.93383
100	0.982759	0.938604	0.93383
120	0.965517	0.936922	0.93383
140	0.965517	0.931876	0.928239
160	0.965517	0.930193	0.926375
180	0.965517	0.928511	0.924511
200	0.956897	0.925988	0.922647

TABLE 13. The influence of different values of *n* on the evaluation indexes.

N	Recall	acc	Specificity
10	0.991379	0.941968	0.936626
20	0.991379	0.941968	0.936626
30	**0.991379**	**0.942809**	0.937558
40	0.982759	0.941968	0.937558
50	0.982759	0.941127	0.936626
60	0.982759	0.939445	0.934762
70	0.982759	0.939445	0.934762
80	0.982759	0.938604	0.93383
90	0.982759	0.938604	0.93383

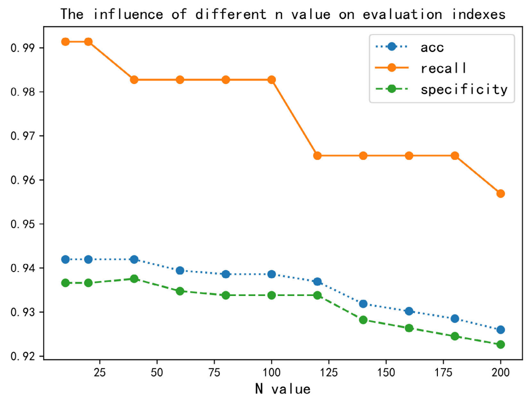


FIGURE 9. The influence of different values of n on the evaluation indexes.

The experimental results are shown in Table 12 and Fig. 9.

(3) It is found that the best range of n in Figure 9 is 10-90, and the third step is carried out using a range from 10-90. The specific values of n are 10, 20, 30, 40, 50, 60, 70, 80 and 90. The experimental results are shown in Table 13 and Fig. 10.

After testing the values of n that increase the difference between base classifiers, it is finally determined that $n = 30$ is the most appropriate. As shown in Fig. 11, when the value of n that increases the difference between base classifiers is between 0 and 30, the three evaluation indexes of the recall rate, accuracy rate and specificity increase. The main reason is that the difference between base classifiers increases as the value of n increases. When the value of n that increases the difference between base classifiers is greater than 30, the accuracy tends to decline at this time. The main reason is that

TABLE 14. Effect comparison with testing tools.

#	Version	URL	acc	Recall
D Shield	v2.1.5.4	http://www.d99net.net	**0.9805**	**0.9324**
WEBDIR+	v2019-0326-0800	https://scanner.baidu.com	0.9717	**0.6756**
SHELLPUB	v1.6.0	https://www.shellpub.com	0.9482	**0.5135**
WS-LSMR			**0.9428**	**0.9914**

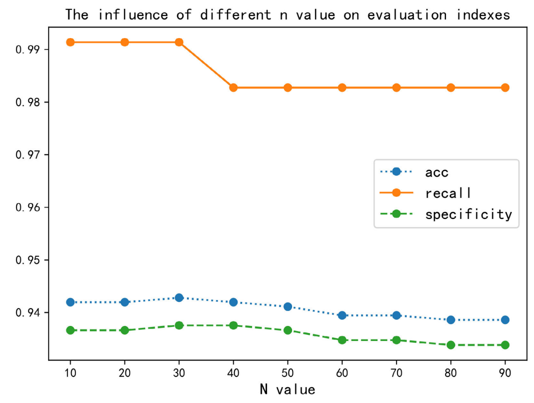


FIGURE 10. The influence of different values of n on the evaluation indexes.

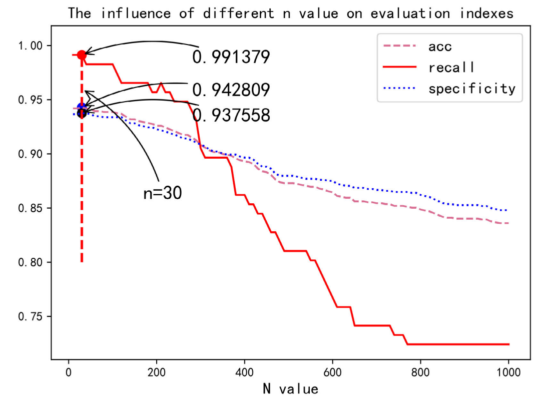


FIGURE 11. The influence of different values of n on the evaluation indexes.

the weight n between them is too large, making the difference between base classifiers weak.

After this weighting, the superiority of the best algorithm for the data can be better reflected. The higher the accuracy of the algorithm is, the greater the weight value will be, and the final prediction effect will be higher than the correctness of any algorithm.

F. COMPARISON OF THE WL-LSMR ALGORITHM AND OTHER CLASSIFICATION ALGORITHMS

The ensemble algorithm uses four base classifiers (logistic regression (LR), support vector machine (SVC), multilayer perceptron (MLP), and random forest (RF)). The algorithm adopts the 5-fold cross validation method to select the following parameters. The reciprocal of the logistic regression's regularization coefficient λ is the reciprocal of 100, and the penalty is the "L2" penalty term. The support vector

TABLE 15. Effect comparison with single machine learning algorithm.

Algorithm	acc	Recall	Time (s)
KNN	0.5828	0.9741	15.9444
MLP	0.9226	0.958	0.0064
DecisionTree	0.9074	0.8879	0.0065
RandomForest	0.9184	0.9051	0.0070
WS-LSMR	**0.9428**	**0.9914**	15.9368

machine's regularization coefficient λ is the reciprocal of 100, and the kernel is a linear kernel function. The activation function of the multilayer perceptron is "tanh", the learning rate is 0.001, and solver is the stochastic gradient descent. The Gini coefficient is used to assess the random forest, 300 decision trees are used, and the minimum sample size is 20.

To verify the performance of the ensemble learning algorithm, We downloaded several popular superior WebShell detectors from the Internet, which are respectively D Shield, WEBDIR + and SHELLPUB. Using the same test set as the WS-LSMR model for scanning detection.It is true that some of the detectors showed good performance on the scanning accuracy, such like D Shield. The experimental results are shown in Table 14. This paper compares it with other classical classifiers: the k-nearest neighbours (denoted as KNN), the multi-layer Perception (denoted as MLP), the Decision Tree and the Random Forest. The experimental comparison of all these classifiers for PCA dimension reduction is shown in Table 15. This paper compares it with several popular ensemble algorithms, RandomForest, Adaboost, and Xgboost use the default parameters provided in scikit-learn. Stacking uses the parameters of logistic regression, support vector machine, multilayer perceptron, and random forest to keep the same parameters as the four algorithm parameters in this article, and Meta-Learner is set to logistic regression. The experimental results are shown in Table 16 on the basis that the feature processing method is consistent with that in this paper. Compared with other methods, although the detection time of WS-LSMR is longer than other methods, the method in this paper obtained a good feature subset by optimizing and selecting features, and weighted optimization was performed based on each base classifier to achieve the best performance. our model can greatly improve the recall rate while ensuring the accuracy.

G. EXPERIMENTAL EXPANSION

The feature selection, model training and prediction methods proposed in this paper can also achieve good results in scripting languages JSP and ASP. The data of JSP and ASP come from the open source project downloaded from

TABLE 16. Effect comparison of different ensemble learning algorithms.

Types	Typical algorithms	acc	Recall	Time (s)
Bagging	RandomForest	0.9217	0.96551	0.0361
boost	adaboost	0.876366	0.982758	1.4875
	Xgboost	0.86291	0.982758	0.3606
Combination strategy	Stacking	0.926829	0.982758	**21.0488**
	WS-LSMR	**0.9428**	**0.9914**	**15.9368**

TABLE 17. Detection of other script languages by the algorithm in this paper.

Types	acc	Recall
asp	0.846154	0.947368
jsp	0.97753	0.925926

TABLE 18. Sample distribution of other scripting languages.

Sample type	Name	Source
asp	eshow	https://github.com/bangqu/eshow
	javashop	https://github.com/zrqgood/javashop
jsp	anyeip	https://www.oschina.net/p/anyeip
	moaspenginer	https://www.oschina.net/p/moaspenginer
	AspFramework	https://sourceforge.net/projects/aspframework/files/latest/download

the following website, and the blacklist is still downloaded using the blacklist website of this paper. The experimental results are shown in the Table 17. The data set is shown in the Table 18.

V. CONCLUSION

This paper proposes a feature selection algorithm based on GINI coefficient and a weighted voting method based on WS-LSMR. The experimental results show that this method can handle unbalanced WebShell data, and has a very high recall rate while ensuring the accuracy. The experimental procedures of feature selection, model training and prediction proposed in this paper are also applicable to the detection of other scripting languages, and can achieve better detection results, experiment shows that the proposed methodology can be extended to the detection of other scripting languages. Although the detection time of WS-LSMR is longer than other methods, the accuracy and the recall of the method proposed in this paper performs much better than other methods. In order to further ensure the security of the website, in the future, we will continue to classify and detect different types of offensive files in terms of time performance, so as to further improve the adaptability of the detection algorithm.

REFERENCES

[1] (2018). *National Internet Emergency Response Center: Overview of China's Internet Network Security Situation in 2018*. [Online]. Available: http://www.cac.gov.cn/2019-04/17/c_1124379080.htm

[2] X. Sun, X. Lu, and H. Dai, "A matrix decomposition based Webshell detection method," in *Proc. Int. Conf. Cryptogr., Secur. Privacy*, New York, NY, USA, 2017, pp. 66–70.

[3] Y. Wei, J. Q. Huang, and X. Zhou, "PHP technology and its application," *Comput. Mod.*, vol. 5, pp. 86–89, May 2000.

[4] H. Zhang, H. Guan, H. Yan, W. Li, Y. Yu, H. Zhou, and X. Zeng, "Webshell traffic detection with character-level features based on deep learning," *IEEE Access*, vol. 6, pp. 75268–75277, 2018.

[5] X. Mingkun, C. Xi, and H. Yan, "Design of software to search ASP Web shell," *Procedia Eng.*, vol. 29, pp. 123–127, 2012, doi: 10.1016/j.proeng.2011.12.680.

[6] R. Li, L. Zhou, S. Zhang, H. Liu, X. Huang, and Z. Sun, "Software defect prediction based on ensemble learning," in *Proc. 2nd Int. Conf. Data Sci. Inf. Technol. (DSIT)*, New York, NY, USA, Jul. 2019, pp. 1–6, doi: 10.1145/3352411.3352412.

[7] J. M. Reddy and C. Hota, "P2P traffic classification using ensemble learning," in *Proc. 5th IBM Collaborative Academia Res. Exchange Workshop (I-CARE)*, New York, NY, USA, 2013, pp. 1–4, doi: 10.1145/2528228.2528243.

[8] J. Vanerio and P. Casas, "Ensemble-learning approaches for network security and anomaly detection," in *Proc. Workshop Big Data Analytics Mach. Learn. Data Commun. Netw. (Big-DAMA)*, New York, NY, USA, 2017, pp. 1–6, doi: 10.1145/3098593.3098594.

[9] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surv.*, vol. 50, no. 2, Jun. 2017, Art. no. 23, doi: 10.1145/3054925.

[10] C. Y. Wang, L. L. Hu, M. Z. Guo, X. Y. Liu, and Q. Zou, "ImDC: An ensemble learning method for imbalanced classification with miRNA data," *Genet. Mol. Res.*, vol. 14, no. 1, pp. 123–133, 2015.

[11] D. Zhang, J. Ma, J. Yi, X. Niu, and X. Xu, "An ensemble method for unbalanced sentiment classification," in *Proc. 11th Int. Conf. Natural Comput. (ICNC)*, Zhangjiajie, China, Aug. 2015, pp. 440–445.

[12] Y. Liu, "Balancing ensemble learning through error Shif," in *Proc. 4th Int. Workshop Adv. Comput. Intell.*, Wuhan, China, 2011, pp. 349–356.

[13] J. Duan, K. Ma, and R. Sun, "Unbalanced data sentiment classification method based on ensemble learning," in *Proc. 2nd Int. Conf. Big Data Technol. (ICBDT)*, New York, NY, USA, 2019, pp. 34–38, doi: 10.1145/3358528.3358597.

[14] N. V. Chawla and J. Sylvester, "Exploiting diversity in ensembles: Improving the performance on unbalanced datasets," in *Proc. Int. Workshop Multiple Classifier Syst.* Berlin, Germany: Springer, 2007, pp. 397–406.

[15] T. Dinh Tu, C. Guang, G. Xiaojun, and P. Wubin, "Webshell detection techniques in Web applications," in *Proc. 5th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Hefei, China, Jul. 2014, pp. 1–7, doi: 10.1109/ICCCNT.2014.6963152.

[16] A. Croix, T. Debatty, and W. Mees, "Training a multi-criteria decision system and application to the detection of PHP webshells," in *Proc. Int. Conf. Mil. Commun. Inf. Syst. (ICMCIS)*, Budva, Montenegro, May 2019, pp. 1–8, doi: 10.1109/ICMCIS.2019.8842705.

[17] Y. Tian, J. Wang, Z. Zhou, and S. Zhou, "CNN-webshell: Malicious Web shell detection with convolutional neural network," in *Proc. 6th Int. Conf. Netw., Commun. Comput. (ICNCC)*, New York, NY, USA, 2017, pp. 75–79.

[18] Z. Wang, J. Yang, M. Dai, R. Xu, and X. Liang, "A method of detecting Webshell based on multi-layer perception," *Acad. J. Comput. Inf. Sci.*, vol. 2, no. 1, pp. 81–91, 2019, doi: 10.25236/AJCIS.010021.

[19] H. Cui, D. Huang, Y. Fang, L. Liu, and C. Huang, "Webshell detection based on random forest–gradient boosting decision tree algorithm," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Piscataway, PA, USA, Jun. 2018, pp. 153–160.

[20] Y. Fang, Y. Qiu, L. Liu, and C. Huang, "Detecting Webshell based on random forest with FastText," in *Proc. Int. Conf. Comput. Artif. Intell. (ICCAI)*, New York, NY, USA, 2018, pp. 52–56.

[21] A. Tartar and A. Akan, "Malignant-benign classification of pulmonary nodules by bagging-decision trees," in *Proc. Med. Technol. Nat. Conf. (TIPTEKNO)*, Oct. 2015, pp. 1–4, doi: 10.1109/TIPTEKNO.2015.7374622.

[22] Y. Zhang and P. He, "A revised AdaBoost algorithm: FM-AdaBoost," in *Proc. Int. Conf. Comput. Appl. Syst. Model. (ICCASM)*, Taiyuan, China, Oct. 2010, pp. V11-277–V11-281, doi: 10.1109/ICCASM.2010.5623209.

[23] T.-K. An and M.-H. Kim, "A new diverse AdaBoost classifier," in *Proc. Int. Conf. Artif. Intell. Comput. Intell.*, Sanya, China, Oct. 2010, pp. 359–363, doi: 10.1109/AICI.2010.82.

[24] M. Gumus and M. S. Kiran, "Crude oil price forecasting using XGBoost," in *Proc. Int. Conf. Comput. Sci. Eng. (UBMK)*, Antalya, Turkey, Oct. 2017, pp. 1100–1103, doi: 10.1109/UBMK.2017.8093500.

[25] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, ''XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud,'' in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Shanghai, China, Jan. 2018, pp. 251–256, doi: 10.1109/BigComp.2018.00044.

[26] B. Pavlyshenko, ''Using stacking approaches for machine learning models,'' in *Proc. IEEE 2nd Int. Conf. Data Stream Mining Process. (DSMP)*, Aug. 2018, pp. 255–258, doi: 10.1109/DSMP.2018.8478522.

[27] Y. Chen and M. L. Wong, ''Optimizing stacking ensemble by an ant colony optimization approach,'' in *Proc. 13th Annu. Conf. Companion Genetic Evol. Comput. (GECCO)*, 2011, pp. 7–8.

[28] R. Li and X. Wang, ''Self-adaptive weighted majority vote algorithm based on entropy,'' in *Proc. 2nd Asia–Pacific Conf. Intell. Robot Syst. (ACIRS)*, Wuhan, China, Jun. 2017, pp. 73–77, doi: 10.1109/ACIRS.2017.7986068.

[29] *GONNSAI.PHP Kennel Exploration: Opcode in PHP*. Accessed: Apr. 2011. [Online]. Available: http://www.nowamagic.net/librarys/veda/detail/1325

[30] K. Shen, L. Ke, J. Ma, K. Zhang, Y. Lu, and X. Chen, ''A blended feature selection method in text classification,'' in *Proc. Int. Conf. Cyberspace Technol. (CCT)*, 2013, pp. 573–576.

[31] Z. Zhai, H. Xu, B. Kang, and P. Jia, ''Exploiting effective features for Chinese sentiment classification,'' *Expert Syst. Appl.*, vol. 38, no. 8, pp. 9139–9146, Aug. 2011.

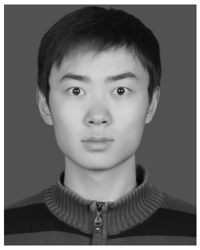
[32] Y. Yang, ''Research and realization of Internet public opinion analysis based on improved TF–IDF algorithm,'' in *Proc. 16th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci. (DCABES)*, Anyang, China, Oct. 2017, pp. 80–83, doi: 10.1109/DCABES.2017.24.

ZHUANG AI was born in Hanchuan, Hubei, China, in 1996. He received the B.S. degree in computer science and technology from the College of Engineering and Technology, Hubei University of Technology, Hubei, China, in 2018. He is currently pursuing the master's degree with Xinjiang University.

He is a student member of the China Computer Federation. His research interests include machine learning, artificial intelligence, and network security.

NURBOL LUKTARHAN was born in Xinjiang, China, in 1981. He received the B.S., M.S., and Ph.D. degrees in computer science from Jilin University, Changchun, China, in 2005, 2008, and 2010, respectively.

From May 2015 to July 2016, he was a Visiting Scholar with Tsinghua University. He is currently an Associate Professor with Xinjiang University and the Deputy Director of the Network and Information Technology Center, Xinjiang University. His research interests include network security and data mining.



YUXIN ZHAO was born in Linyi, Shandong, China, in 1993. He received the B.S. degree in computer science and technology from the College of Information Science and Technology, Linyi University, Shandong, in 2017. He is currently pursuing the M.Eng. degree with Xinjiang University.

He is a student member of the China Computer Federation. His research interests include machine learning, artificial intelligence, and network security.

CHAOFEI TANG was born in Xuzhou, Jiangsu, China, in 1993. He received the B.S. degree in civil engineering from the Pujiang College, Nanjing University of Technology, China, in 2017. He is currently pursuing the master's degree with Xinjiang University.

He is a student member of the China Computer Federation. His research interests include machine learning, artificial intelligence, and network security.