A Method of Extracting Malware Features Based on Gini Impurity Increment and Improved TF-IDF

Shimiao Sun
School of Electrical and Information Engineering,
Beijing University of Civil Engineering and Architecture,
China
sunshimiao001@foxmail.com

Yashu Liu
School of Electrical and Information Engineering,
Beijing University of Civil Engineering and
Architecture, China
liuyashu@bucea.edu.cn

Abstract: *In recent years, the quantities and types of malwares have grown explosively, which bring many challenges to identify and detect them. In order to improve the identification efficiency of malicious code, a malicious code feature representation method based on feature dimension reduction is proposed. By fusing the Gini impurity increment and the Improved Term Frequency-Inverse Document Frequency algorithm (ITF-IDF), ΔGini-Improving Term frequency inverse document frequency (ΔGini-ITFIDF) method is presented, which can get more valuable assembly instruction features for family detection. ΔGini-ITFIDF standardizes the assembly instructions of the PE disassembly files, then, measures the two indicators of the expected error rate increment and weight of the malicious code assembly instruction features, and obtains more valuable features to identify malicious codes. The experimental results show that the classification accuracy of the ITF-IDF algorithm is significantly improved compared with the ITF-IDF algorithm. At the same time, ΔGini-ITFIDF can effectively improve the classification performance.*

Keywords: *Gini impurity increment, TF-IDF, feature selection, dimensionality reduction.*

Received April 30, 2022; accepted February 12, 2023
https://doi.org/10.34028/iajit/20/3/14

1. Introduction

The development of network technology has promoted the exchange of information among users, and has also created favorable conditions for the breeding and wreaking havoc of computer and mobile phone viruses and Trojan horses. In March 2022, Internet security threat report from CNCERT pointed out that the IP address of computers controlled by virus Trojans in China reached 5.32 million, an increase of 11.3% compared with February, on the Internet, more than 9080 websites were tampered with by malware, an increase of nearly 173.2% over the previous month, the number of websites implanted with backdoors in China was 2524, an increase of 40.5% over the previous month, there were more than 3823 incidents involving malicious programs and more than 4400 incidents involving network vulnerabilities [6]. The attack of malicious code on the computer brings harm to the user's information security. The leakage of personal privacy, and the loss and damage of data also have a serious impact on daily life and work. Effective interception and killing of malicious programs have become an urgent problem.

To intercept and deal with malicious code, we must effectively identify the category, source and other attributes of malicious code. To achieve the above objectives, effective and reasonable features contained in different malicious code samples must be extracted. In recent years, this research field has also made significant progress and research results. Santos *et al*. [21] analyzed the family and behavior purpose of unknown malicious samples by selecting the frequency of malicious code operation instructions. Tang *et al*. [23] extracted opcode features of different granularities through Multy-Granularity Opcode Droid (MGOPDroid), and converted opcodes into sequences to improve the anti-obfuscation of malicious code identification. By researching the sequence of malicious program Application Programming Interface (API) calls, Amer *et al*. [4] and Al-Hashmi *et al*. [2] run malicious code PE files in a sandbox and analyzed the behavior of malicious code runtimes by recording API calls. Nataraj *et al*. [17] converted binary executable files into gray-scale images in 2011. This method used image processing technology to analyze binary PE files of malware, which had a far-reaching impact on texture analysis and visual analysis of malware [7, 10, 14, 16].

Conventional methods always use static or dynamic behaviors such as binary bytecode, API call sequences, function call sequences and so on, which are got by feature extraction tools such as Interactive Disassembler Professional (IDA-pro) or running malicious programs in sandbox. These methods usually obtain all the sample features information, resulting in a large number of redundant features, which affect the accuracy of the detection results. Therefore, it is necessary to further purify the features, filter out redundant features, obtain more valuable features, to improve the accuracy of malicious code detection.

Research shows that effectively reducing the feature dimension, selecting high-quality features to filter out redundant and irrelevant features has been widely used in data mining fields, such as text feature extraction and classification [8]. The purpose of the paper is to propose a reasonable feature extraction method of malicious code, so as to identify and classify malicious code more efficiently. This paper takes the Portable Executable (PE) disassembly files as the research object, constructs the assembly instruction word vector, analyzes the impact of assembly instructions' weight in the sample family on classification. Our method improves the classification accuracy of malicious code by extracting features that can better represent specific categories and reducing the feature dimension.

This paper presents a method based on ΔGini-Improving Term Frequency-Inverse Document Frequency (ITFIDF) feature extraction method, which is based on Gini impure increment and improved Term Frequency-Inverse Document Frequency (TF-IDF) algorithm for feature screening, dimensionality reduction, and identifies malware through Random Forest (RF) and K-Nearest Neighbor (KNN) classification model. Different from the methods proposed by Santos *et al.* [21], Mohammed *et al.* [16] and other existing methods, the feature extraction method proposed in this paper fully considers the impact of the distribution of feature within and outside the family on classification. ΔGini-ITFIDF selects high-quality features and improves the classification accuracy of the classifier. The main work of this paper is as follows:

1. Decompiling PE files, we get the opcodes and operands of assembly instructions, then normalize them by some rules.
2. We improve the traditional TF-IDF algorithm to highlight the discrimination of TF-IDF to malware families.
3. By analyzing the two indicators of expected error rate increment and weight of features, we propose a feature extraction algorithm based on Gini impure increment and improved TF-IDF fusion, ΔGini-ITFIDF, to improve the classification performance.

2. Related Work

Malicious code feature extraction is always divided into static feature extraction and dynamic feature extraction. Static analysis technology refers to extract the features of malicious code without executing it in a dedicated device such as a sandbox or a virtual machine, but it is often necessary to analyze the compiler information, import/export table, assembly code, resources, various string information (version information, URL, special identification, etc.,) and shell information are used for feature extraction. The method of actually running executable files in sandbox, virtual machine and other environments and getting key features is called dynamic feature extraction. Compared with the static analysis method, the dynamic analysis actually runs the PE file, and obtains malicious code features which more accurately describes the difference in the behavior of the malicious code, but there are also problems such as time-consuming and resource-consuming. Therefore, it is difficult to quickly process a large number of malicious code samples. This section reviews and summarizes the related research on malicious code feature extraction methods.

Abou-Assaleh *et al.* [1] used bytecode sequence-based N-Grams as static features for the first time. In terms of operating instructions for malicious code, Kang *et al.* [12] used the Long-Short Term Memory (LSTM) method in word2vec to extract the opcode of malicious code and the name of the API calling function for classification. Jeon and Moon [11] extracted opcode sequences from PE files of malicious code and constructed a convolutional recurrent neural network including opcode convolutional encoders. Pektaş and Acarman [19] detected malicious samples by analyzing the calling behavior and checking the execution path of the malicious code during the calling process through the instruction call graph, using the opcode sequence of the malicious code as a feature. O'Shaughnessy and Breitinge [18] constructed the letter-letter frequency-letter corresponding Huffman encoding as a composite feature to reduce the complexity of feature generation and feature analysis.

In terms of feature dimensionality reduction, Singh *et al.* [22] analyzed the implicit semantics of malware through Latent Semantic Indexing (LSI), and reduced the dimension of malware features by relying on singular values to improve the performance of classification. Şahin *et al.* [20] performed features dimensionality reduction by Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to solve the problem of limited system resources. Table 1 shows a comparison of different related works.

Table 1. Related work comparison.

Reference	**Feature**	**Malware detection Method**
Santos *et al.* [21]	Opcode	Opcode sequences frequency extraction
Al-Hashmi *et al.* [2]	API calls	deep learning+ EGBoosting
Nataraj *et al.* [17]	Binary	Gray Image + machine learning
Mohammed *et al.* [16]	Binary	Frequency image + CNN
Jeon and Moon [11]	Opcode	Opcode sequence + CRNN
Pektaş and Acarman [19]	Opcode	Opcode + FCG + LSTM and CNN
Singh *et al.* [22]	Opcode	Latent Semantic Indexing +RF
Our method	Opcode	Δ Gini-ITFIDF + RF, KNN

Most of the above methods were aimed at all features, and do not take into account the redundant information that may exist in the extracted features. Using LSI [22], LDA [20] or the TF-IDF method mentioned by Al-Hashmi *et al.* [2] for malware feature selection didn't consider the distribution of features within and outside the malware family. In addition, comparing with the Gini index mentioned by Al-Hashmi *et al.* [2], Gini impurity increment (ΔGini) value is proportional to expected classification accuracy, which is more directly reflects the classification effect. This paper improved the TF-IDF method to further filter the features by using the feature weight inside and outside the family, and use Δ Gini-ITFIDF as the indicator to measure feature importance and obtain more efficient classification performance.

3. Methodology

3.1. Gini Impurity

In the RF [9] algorithm composed of decision tree [24], several subdatasets are extracted from the dataset by bootstrap sampling method, and a Classified Regression Tree (CART) is constructed for each subdataset. When constructing CART, several features are randomly selected to form feature subsets to select node fields, and each node in each CART can have higher purity. Then, for CART tree, the classification and regression effects are determined by voting and mean value respectively. Gini impurity [13] is an important field selection indicator of CART in verifying the purity of each node, which comes from information entropy. Gini impurity measures the expected error rate of a feature in the data set, and its decline rate, i.e., increment, shows the impact of features on dependent variables. The calculation formula of Gini impurity is shown in Equation (1):

$$\begin{aligned} Gini(p_1, p_2, \dots p_J) &= \sum_{j=1}^{J}(1-p_j)p_j \\ &= \sum_{j=1}^{J}(p_j - p_j^2) = 1 - \sum_{j=1}^{J} p_j^2 \end{aligned} \tag{1}$$

Here, p_j represents the occurrence probability of the j-th item in the possible value of an event, which can also be expressed by empirical probability as:

$$Gini(D) = 1 - \sum_{j=1}^{J}\left(\frac{|D_j|}{|D|}\right)^2 \tag{2}$$

Among them, $|D|$ represents all samples in the event, and $|D_j|$ represents the number of occurrences of the type j sample in the dataset and the probability value p_j refers to the frequency represented, that is, the probability of class j samples. Therefore, for a characteristic A_i, the Gini purity is as shown in Equation (3).

$$\begin{aligned} Gini(A_i) &= P(A_i)[1-\sum_{j=1}^{J}P(D_j \mid A_i)^2] + P(\overline{A_i})[1-\sum_{j=1}^{J}P(D_j \mid \overline{A_i})^2] \\ &= (\frac{D_i}{D})[1-\sum_{j=1}^{J}(\frac{D_{ij}}{D_i})^2] + (1-\frac{D_i}{D})[1-\sum_{j=1}^{J}(\frac{D_i - D_{ij}}{D - D_i})^2] \end{aligned} \tag{3}$$

The decline rate of Gini impurity is expressed as Equation (4).

$$\Delta Gini(A_i) = Gini(D) - Gini(A_i) \tag{4}$$

According to the results of Equation (4), select the features with large decline rate of Gini impurity, that is the features that have a great impact on the classification results.

3.2. TF-IDF Algorithm and its Improvement

TF-IDF [5] is Composed of Term Frequency (TF) and Inverse Document Frequency (IDF). If the frequency of the entry in the sample is high and the entry appears less in other sample files, the entry has a better classification effect on the text than other entries. Expressed as the product of word occurrence frequency and inverse document frequency:

$$\text{TFIDF} = tf_j * idf_j \tag{5}$$

The definition of term frequency tf_j is as follows:

$$tf_j = \frac{n_j}{\sum_{i=1}^{k} n_k} \tag{6}$$

There are k entries in each type of document, n_j is the number of entries j in each document. The denominator is the sum of the number of entries in each document, that is the total number of entries. Inverse document frequency idf_j is:

$$idf_j = \log \frac{D}{D_j} \tag{7}$$

Here, D is the total number of documents, and D_j is the number of inverse text documents, that is the number of texts containing the entry in the corpus.

The TF indicates the importance of each term in the text. With the decrease of D_j, it shows that the frequency of this entry in the whole dataset decreases and the inverse file frequency IDF increases. Therefore, TF-IDF tends to retain entry features that are important to this document and not important to other texts.

The traditional TF-IDF algorithm can effectively judge whether the entry feature is representative for the text. If the frequency of an entry in a kind of text is very high, it can show that the entry occupies an important position, and the better representativeness in this kind of text. But there is also a problem, the increasement of the number of an entry in the same type of text also increases the number of documents D_j containing the entry in the whole corpus, which reduces the IDF value. Hence, it is necessary to further distinguish whether the text with frequent occurrence of this entry is the same type of text. If more entries appear in the same type of text, the impact on TF-IDF value should be positive. On the contrary, if they appear frequently in other types of text, the impact on TF-IDF value should be negative.

Facing the above problems, in order to better distinguish the category of word segmentation entries, Zhang *et al*. [25] made some improvements to the traditional TF-IDF algorithm to adapt to text classification, the Equation is:

$$idf_j = \log \frac{d_j}{D_j} D \tag{8}$$

Here, d_j is the number of documents containing entry j

in family *A* documents, D_j is the number of entry *j* in the document set, and *D* is the total number of documents.

It can be seen that the formula takes into account the distribution of features within and outside the family. However, the improved algorithm only takes into account the number of texts containing entry *j* in the document of family *A* and the number of entry *j* in the sample set, thus ignoring the number of entry *j* in other families. This will increase the proportion of entries in the texts of family *A* and others, which will lead to the increase of D_j. If the number of entries *j* in the samples of other families is too large, it will weaken the influence of D_j on idf_j. In addition, the large number of documents containing entry *j* in other documents does not mean that the proportion will be large. Therefore, this paper made further improvements on this basis, and calculated the frequency of an entry *j* in documents of the family *A* and others respectively. Based on this, the original IDF definition is modified as follows:

$$idf_j' = \log \frac{\frac{d_j}{d}}{\frac{d_j}{d} + \frac{D_j - d_j}{D - d}} D \tag{9}$$

Here d_j is the number of documents containing entry *j* in family *A*, *d* is the total number of documents contained in family *A*, *D* is the total number of documents, and D_j is the number of documents containing entry *j* in the document set. Taking the proportion of documents containing entry *j* in family *A* and other documents as variables can avoid the situation of large sample size but small proportion. In the family *A* documents where entry *j* is located, the value of idf_j' is positively correlated with the proportion of the number of documents containing entry *j*, and the stronger the ability of this entry to distinguish family *A*. In the documents of other families, the value of *idf'* is negatively correlated with the proportion of the number of documents containing entry *j*, and the weaker the ability of this entry to distinguish family *A*. Therefore, the value of *idf'* determines whether an entry has been distinguishability.

In Addition, the original TF algorithm only represents the weight of entries in a single sample, which makes the frequency of entries in a family complex, resulting in large fluctuation of TF-IDF value. Not only it can't represent the weight of entries in the family, but also lead to the possibility of erroneously screening important features in feature screening. Therefore, in order to more intuitively represent the weight of entries in the family, this paper further modifies the TF algorithm.

$$tf_j' = \frac{N_j}{\sum_{i=1}^{k} N_k} \tag{10}$$

Here, N_j is the number of entries in family *A* where entry *j* belongs, and the denominator represents the total number of all entries in family *A*. In this way, the frequency corresponding to the entry changes from the frequency of the entry in the document to the frequency in the whole family, so that each entry in the same family corresponds to a unique TF-IDF value. It can't only better reflect the degree of distinction between entries and families, but also simplify the screening process of entries.

To sum up, this paper gets the final improved TF-IDF algorithm, defined as:

$$\text{ITF-IDF} = tf_j' * idf_j' = \frac{N_j}{\sum_{i=1}^{k} N_k} * \log \frac{\frac{d_j}{d}}{\frac{d_j}{d} + \frac{D_j - d_j}{D - d}} D \tag{11}$$

ITF-IDF algorithm avoids the influence of category on the weight calculation of malicious code, and can screen out the characteristic terms that highlight the characteristics of this category, and can also filter out the redundant term features that have little impact on the classification.

4. Feature Extraction

4.1. Preprocessing of Assembly Instructions

Firstly, this paper preprocesses the malicious samples, which can be transformed into code in the form of assembly language through reverse tools. This experiment uses IDA-Pro to disassemble these samples and convert it into assembly codes. And it uses the pefile in Conda to parse and generate assembly text.

Because the assembly instructions of malicious code are messy, rich in meaning and different in length, there are many redundant interference information. In order to solve the above problems, we first need to consider the normalization of the length of the operation code, and standardize the operands of the assembly code such as register, memory, immediate number, calling instruction, and the number of operations after the jump instruction. The specific methods are as follows:

1. Opcode length normalization.
The length range of opcodes in assembly instructions is about 2-6 characters, about 48% of opcodes are within 3 characters, 29% of opcodes are 4 characters, 18% of opcodes are 5 characters, and 5% of opcodes are 6 characters. However, more than 90% of the characters of the called opcode are less than three. Therefore, this paper uniformly selects the first three characters of the opcode, such as, call → cal, push → pus. If the character length is less than 3, the insufficient character bits shall be filled with spaces, such as, js→js("_"represents space).
2. Operand standardization and normalization

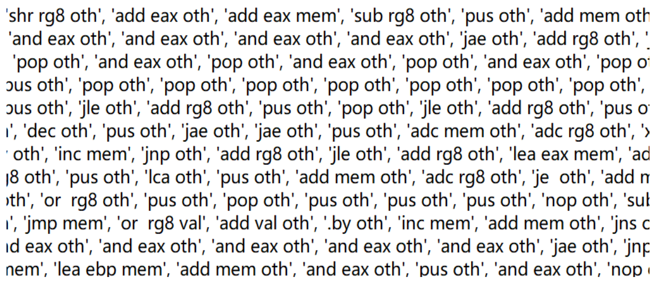
   a) Register. There are three kinds of registers commonly used, 8b, 16b and 32B. This paper standardizes three different types of registers as follows, al→rg8, ax→r16, eax, ebx, ecx, edx, edi, esi, ebp, esp.etc.,→r32(Since all 32B registers described in R32 is too rough, R32 standardization is not adopted in this paper).

b) Memory. [*eax+3Ch*], [*edi+4*]. etc., operating instructions representing memory are standardized into mem.

c) Immediate. 5A4Dh, 70h. etc., operation instructions representing immediate numbers are standardized into val.

d) Call instruction. Only the operands that call the internal instruction are processed. For example, the internal function "call sub_101C02D" after normalization is "call sub".

e) The operand after the jump instruction. Normalize "jnz short loc_1018887" to "jnz loc".



```
'shr rg8 oth', 'add eax oth', 'add eax mem', 'sub rg8 oth', 'pus oth', 'add mem oth
 'and eax oth', 'and eax oth', 'and eax oth', 'and eax oth', 'jae oth', 'add rg8 oth', '
  'pop oth', 'and eax oth', 'pop oth', 'and eax oth', 'pop oth', 'and eax oth', 'pop o
ous oth', 'pop oth', 'pop oth', 'pop oth', 'pop oth', 'pop oth', 'pop oth', 'pop oth',
ous oth', 'jle oth', 'add rg8 oth', 'pus oth', 'pop oth', 'jle oth', 'add rg8 oth', 'pus o
', 'dec oth', 'pus oth', 'jae oth', 'jae oth', 'pus oth', 'adc mem oth', 'adc rg8 oth', '
 oth', 'inc mem', 'jnp oth', 'add rg8 oth', 'jle oth', 'add rg8 oth', 'lea eax mem', 'ad
8 oth', 'pus oth', 'lca oth', 'pus oth', 'add mem oth', 'adc rg8 oth', 'je  oth', 'add m
th', 'or  rg8 oth', 'pus oth', 'pop oth', 'pus oth', 'pus oth', 'pus oth', 'nop oth', 'sul
', 'jmp mem', 'or  rg8 val', 'add val oth', '.by oth', 'inc mem', 'add mem oth', 'jns c
d eax oth', 'and eax oth', 'and eax oth', 'and eax oth', 'and eax oth', 'jae oth', 'jnp
nem', 'lea ebp mem', 'add mem oth', 'and eax oth', 'pus oth', 'and eax oth', 'nop
```

Figure 1. Examples of word segmentation results of some malicious samples.

The processed opcodes and operands show the corresponding types more intuitively, reduce the feature dimension and sample complexity, reduce the interference of complex features on the follow-up work. Figure 1 shows part of the vocabulary content extracted by the sample "0a152188e52dafab247c7c9120aeef0cFe092db7" after the assembly instruction is normalized.

4.2. ΔGini-ITFIDF Feature Extraction Method

After the malicious code sample is standardized in the section 4.1. Method to obtain the parsed text, the opcodes of each assembly instruction are taken as a feature entry. Then, a malicious sample feature extraction algorithm based on Gini impure increment and ITF-IDF fusion is constructed (ΔGini-ITFIDF).

1. Screening phase of Gini impurity increment. Calculate the Gini impure increment for the entries in various samples in the dataset, and sort the filtered Gini impure increment values, observe the proportion of Gini indicators in different ranges, and reasonably set the threshold according to the proportion to filter out the entries with low Gini impure increment. Get the optimized feature subset to construct a new feature set.
2. Screening phase of ITF-IDF. Calculate the ITF-IDF weight X of each feature in this class over the filtered feature subset, set the threshold according to the situation of different file sets, filter out the features with the weight in the last a% in various samples, and form a new feature subset of each type of samples, $U_i=\{x_1, x_2, ..., x_{ki}\}$.

Here, i refers to the number of sample categories, i=1, 2,..., n, and k refers to the number of sample features, k=1,2,...,m. Because the important features of different categories of samples are different, it is necessary to obtain the union set of these new feature subsets to construct a new feature set $U = U_1 \cup U_2 \cup ... \cup U_n$.

3. Classifier construction. Take the filtered entries as features, put the new feature subset into the classifier model and adjust the classification parameters. The training set and test set are divided by 10 cross-validation, the classification performance of our model is evaluated by accuracy. Feature extraction process of ΔGini-ITFIDF is shown in Figure 2.

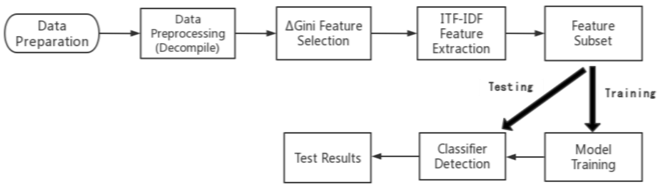


Figure 2. Process of feature selection algorithm.

5. Experimental Evaluation and Result

In the experiment, our dataset and Microsoft Kaggle BIG2015 dataset [15] are used in this paper. Our dataset consists of 10 families and 15000 samples and the detailed distribution of samples in each family is shown in Table 2. BIG2015 dataset consists of 9 families and 10868 samples and the detailed distribution of samples in each family is shown in Table 3. According to the method in section 4.1., our dataset obtains the standardized analytical text containing 345 feature terms. The BIG2015 dataset obtains the standardized analytical text containing 334 characteristic terms. The following is the first example ΔGini-ITFIDF feature extraction and analysis process on our dataset, then gives the comparison of experimental results on BIG2015 dataset.

5.1. Gini Impurity Increment Experiment

Calculate the Gini impurity increment of the entry in our dataset, conduct further correlation analysis on the entry, and arrange the Gini impurity increment of each feature in the original feature set in descending order, and its distribution is shown in Figure 3.

It can be seen from Figure3 that the Gini impurity increment of about 60 features is close to 0, while the features whose Gini impurity increment lower than 0.005 have almost no impact on the classification effect, which can be regarded as redundant features. Therefore, this paper sets the threshold as 0.05 to filter out features that have little impact on the classification effect. The experiment shows that the Gini impurity increment value of 57 features is lower than 0.005.

The 57 features are filtered out to obtain the optimized feature subset, named S1.

Table 2. Sample distribution of our dataset.

Family name	Number of samples
Agent	1500
SoftPulse	1500
Allaple	1500
Mentiger	1500
AdLoad	1500
Nimnul	1500
LMN	1500
Virut	1500
WBNA	1500
AGeneric	1500

Table 3. Sample distribution of BIG2015 dataset.

Family name	Number of samples
Ramnit	1547
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013
Gatak	1013

5.2. ITF- Agorithm Experiment

After the feature subset S1 is obtained through the feature screening of Gini impure increment, in order to select a reasonable threshold for ITF-IDF, 100 samples are randomly selected in each family of the feature subset to filter out the features with the ITF-IDF value of the last 5%, 10%, 15%, 20% and 25% in each family and sample. At the same time, in order to verify the improvement effect of TF-IDF algorithm in this paper. The ITF-IDF algorithm in this paper is compared with the improved algorithm in [25], and the feature subsets extracted by merging are put into the RF model for classification. The results are shown in Figure 4.

It can be seen from Figure 4 that the classification accuracy of feature subset extracted by ITF-IDF significantly improved compared with reference [25], which also shows that the improvement of TF-IDF algorithm in this paper highlights the weight of entries in the family, so that the ITF-IDF value can better represent the importance of entries in the family, which is also an advantage that the algorithm in [25] does not have. In addition, it can be seen from the figure that when 20% of the features are filtered out, the classification accuracy reaches the maximum value, and as the number of filtered features increases, the classification accuracy decreases. This is due to the fact that too few features lead to a reduction in the complexity of the tree in RF, resulting in a reduction in the strength of the tree, which affects the accuracy.

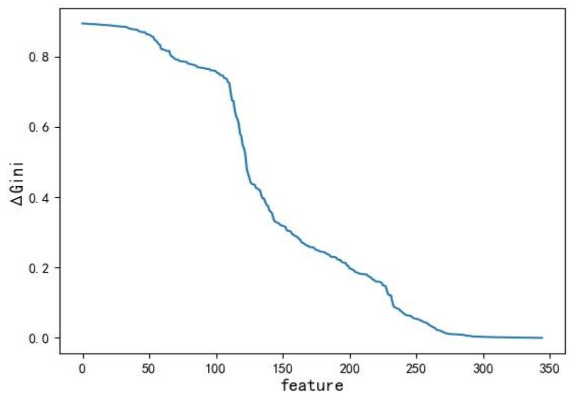


Figure 3. Gini impurity increments with different characteristics.

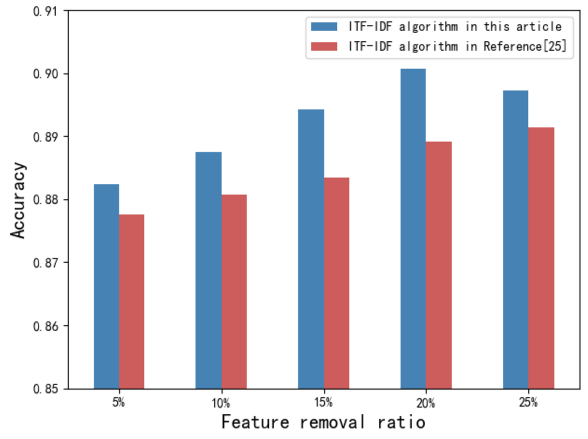


Figure 4. The classification results of different ITF-IDF methods after screening different ratio features.

5.3. Feature Extaction Algorithm Effective-Ness Experiment

In the experiment, each feature set is tested by 10 fold cross-validation. The classification results (est=20) in three different feature sets are shown in Figure 5. It is customary to use the horizontal axis to represent the number of cross-validation, the vertical axis to represent the classification accuracy, and three different dashed lines represent the classification results of different feature sets.

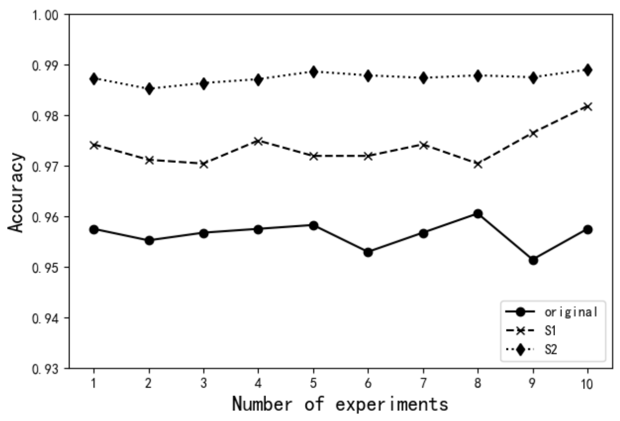


Figure 5. Classification results of each feature subset of our dataset (est=20).

As can be seen in Figure 5, the RF model has significantly improved the classification accuracy after the fusion of Gini impure increment and ITF-IDF feature extraction algorithm. It is preliminarily verified that the feature extraction algorithm proposed in this paper has the significance of improving the classification performance.

5.4. Compaison and Analysis of Microsoft BIG 20015 Experimental

Because the sample distribution of our dataset is relatively balanced, in order to verify the applicability of the algorithm in other unbalanced datasets, this paper further uses Microsoft Kaggle BIG2015 dataset for comparison. The feature extraction method of BIG2015 is the same as our dataset.

Compared with our dataset, BIG2015 dataset is unevenly distributed in each category, which makes the classification of samples more difficult. This paper also standardizes the assembly instructions of BIG2015 dataset, filters the features of the dataset through the fused feature extraction algorithm, and generates two feature subsets S1 and S2. The classification results of the three datasets under the random forest algorithm are shown in Figure 6.

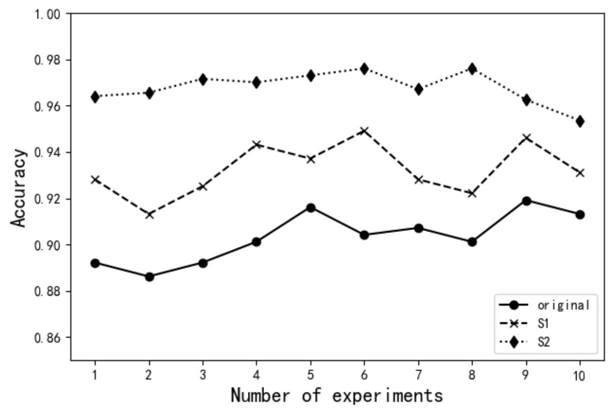


Figure 6. Classification results of each feature subset of BIG2015 (est=20).

As can be seen from Figure 6, under the BIG2015 dataset, the accuracy of the filtered feature subset in the random forest classifier is significantly improved compared with the original feature set. Then, this paper calculates the average value of the classification accuracy under the RF classifier with different parameters for each feature subset of our dataset and BIG2015 dataset respectively. The parameters of RF are selected as 10, 15, 20, and 25 respecti-vely. The classification results are shown in Table 4.

The results in Table 4 show the classification performance of our and BIG2015 datasets on RF model. It shows that the classification accuracy has been improved by 3%-5% respectively after twice feature extraction through Gini impurity increment and fusion ITF-IDF feature extraction algorithm.

In section 5.3., the effectiveness of the feature extraction algorithm proposed in this paper has been verified.

In order to further verify the improvement of the classification performance of them, this paper also uses the other classification algorithm for experiment. Compared with Support Vector Machine (SVM), Adaboost and Naïve Bayes, KNN has the advantages of no parameter estimation, insensitivity to outliers, and has higher accuracy on multi-classification problems. Therefore, it is widely used in text feature classification [3]. Considering above factor, this paper uses KNN algorithm for comparison experiment.

The number of nearest neighbors k is 1-10 respectively. For each number of nearest neighbors and each feature subset, 10 times of cross validation and average value are adopted. The classification results are shown in Table 5.

The results in Table 5 show that KNN classifier can obtain the highest accuracy of 98.16% on our dataset (*k=6*) and 93.32% on BIG2015 dataset (*k=2*). Compared with the original features, after filtering the features by the fusion feature extraction algorithm, the classification accuracy is improved by 5%-7%.

Comparing the results in Tables 4 and 5, we can see that whether RF or KNN model is adopted, the accuracy of the features extracted by ΔGini-ITFIDF method has been significantly improved, which shows the effectiveness of our method. In addition, for datasets with unbalanced sample distribution such as BIG2015, our method can still improve its classification accuracy, which shows that our method has more robustness.

Table 4. Classification accuracy of different datasets in the RF model.

	Our dataset			**BIG2015**		
	Original	**S1**	**S2**	**Original**	**S1**	**S2**
est=10	95.5%	96.96%	98.45%	90.24%	92.75%	95.54%
est=15	95.63%	97.2%	98.69%	90.59%	92.72%	95.86%
est=20	95.64%	97.38%	98.74%	90.12%	93.38%	96.1%
est=25	95.84%	97.36%	98.77%	90.33%	93.23%	95.81%

Table 5. Classification accuracy of different datasets in the KNN model.

		Our dataset			**BIG2015**		
		Original	**S1**	**S2**	**Original**	**S1**	**S2**
k	*1*	93.73%	96.04%	98.14%	86.4%	90.5%	92.96%
	2	93.78%	95.99%	97.98%	86.64%	90.27%	93.32%
	3	93.8%	96.16%	98.32%	86.7%	90.08%	92.23%
	4	93.25%	95.92%	98.23%	86.52%	89.19%	91.34%
	5	93.23%	95.87%	98.09%	85.83%	89.22%	91.69%
	6	93.14%	95.94%	98.16%	85.66%	88.83%	91.24%
	7	93.02%	95.7%	98.02%	86.13%	88.71%	91.1%
	8	93.22%	95.51%	97.93%	84.76%	87.32%	90.79%
	9	93.01%	95.62%	97.96%	85.17%	88.56%	90.58%
	10	92.78%	95.43%	97.91%	84.73%	87.33%	90.41%

The above experiments show that the enhanced data sets obtained after feature selection by the ΔGini-ITFIDF algorithm for two different data sets have significantly improved the classification accuracy of classification frameworks such as RF and KNN. It not only reflects the positive effect of ΔGini-ITFIDF on improving the classification of malicious codes, but highlighting the research significance of improving the feature selection algorithm in this paper.

At the end of this paper, the results obtained by other feature extraction methods on our dataset are compared with the method in this paper, and the results are shown in Table 6. The comparative experiments are: the experiment of converting the frequency of malicious code opcodes [21], the experiment of executing sequence by binary extraction of opcodes [11], the experiment of feature extraction of grayscale images by frequency conversion of adjacent bytes [16]. It can be seen from Table 6 that our method has a better classification effect than other malicious code feature extraction methods.

Table 6. Comparison of different references methods.

Method	Accuracy
Santos *et al*. [21]	93.8%
Jeon and Moon [11]	95.91%
Mohammed *et al*. [16]	98.3%
Our Method	98.74%（*est=20*）

6. Conclusions

According to Gini impurity increment and TF-IDF, this paper presents a feature extraction algorithm, ΔGini-ITFIDF, which combines Gini impure increment and ITF-IDF, and applied to malware classification. ΔGini-ITFIDF can extract more representative features and improve the classification accuracy of many traditional classification models. At the same time, the applicability of the feature selection model in unbalanced dataset is verified. Compared with other methods, ΔGini-ITFIDF uses Gini impurity increment and feature distribution weight inside and outside the family as feature selection indicators for the first time. Experimental results show that our method has more practical significance in improving classification accuracy.

Although ΔGini-ITFIDF can improve classification accuracy of unbalanced datasets. However, compared with balanced datasets, there is still room for further improvement. How to improve the fluctuation of classification performance caused by dataset balance is the main research direction in the future.

References

[1] Abou-Assaleh T., Cercone N., Keselj V., and Sweidan R., "N-gram-based Detection of New Malicious Code," *in Proceedings of the 28th Annual International Computer Software and Applications Conference,* Hong Kong, pp. 41-42, 2004.

[2] Al-Hashmi A., Ghaleb F., Al-Marghilani A., Yahya A., Ebad S., Saqib M., and Darem A., "Deep-Ensemble and Multifaceted Behavioral Malware Variant Detection Model," *IEEE Access*, vol. 10, pp. 42762-42777, 2022.

[3] Alhutaish R. and Omar N., "Arabic Text Classification Using K-Nearest Neighbour Algorithm," *The International Arab Journal of Information Technology,* vol. 12, no. 2, pp. 190-195, 2015.

[4] Amer E., Zelinka I., and El-Sappagh S., "A Multi-perspective Malware Detection Approach through Behavioral Fusion of API Call Sequence," *Computers and Security,* vol. 110, pp. 102449, 2021.

[5] Arivarasan A. and Karthikeyan M., "Data Mining K-Means Document Clustering Using TFIDF and Word Frequency Count," *International Journal of Recent Technology and Engineering,* vol. 8, no. 2, pp. 2542-2549, 2019.

[6] CNCERT., "CNCERT Internet Security Threat Report - March 2022," Retrieved from: https://www.cert.org.cn/, Last Visited, 2022.

[7] Dai Y., Li H., Qian Y., and Lu X., "A Malware Classification Method Based on Memory Dump Grayscale Image," *Digital Investigation,* vol. 27, pp. 30-37, 2018.

[8] El-Hajj W. and Hajj H., "An Optimal Approach for Text Feature Selection," *Computer Speech and Language*, vol. 74, pp. 103164, 2022.

[9] Genuer R., Poggi J., Tuleau-Malot C., and Villa-Vialaneix N., "Random Forests for Big Data," *Big Data Research*, vol. 9, pp. 28-46, 2017.

[10] Hsiao S., Kao D., Liu Z., and Tso R., "Malware Image Classification Using One-shot Learning with Siamese Networks," *Procedia Computer Science*, vol. 159, pp. 1863-1871, 2019.

[11] Jeon S. and Moon J., "Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences," *Information Sciences*, vol. 535, pp. 1-15, 2020.

[12] Kang J., Jang S., Li S., Jeong Y., and Sung Y., "Long Short-term Memory-based Malware Classification Method for Information Security," *Computers and Electrical Engineering*, vol. 77, pp. 366-375, 2019.

[13] Laber E. and Murtinho L., "Minimization of Gini Impurity: NP-completeness and Approximation Algorithm via Connections with the k-Means Problem," *Electronic Notes in Theoretical Computer Science*, vol. 346, pp. 567-576, 2019.

[14] Li L., Ding Y., Li B., Qiao M., and Ye B., "Malware Classification Based on Double Byte Feature Encoding," *Alexandria Engineering Journal*, vol. 61, no. 1, pp. 91-99, 2022.

[15] Microsoft., "Microsoft. Kaggle Dataset," retrieved from, https://www.kaggle.com/c/malware-classification. Last Visited, 2022.

[16] Mohammed T., Nataraj L., Chikkagoudar S., Chandrasekaran S., and Manjunath B., "Malware Detection Using Frequency Domain-based Image Visualization and Deep Learningm," *in Proceedings of the 54th Hawaii International Conference on System Sciences,* Hawaii, pp. 7132, 2021.

[17] Nataraj L., Karthikeyan S., Jacob G., and Manjunath B., "Malware Images: Visualization and Automatic Classification," *in Proceedings of the 8th International Symposium on Visualization for Cyber Security*, Pittsburgh Pennsylvania, pp. 1-7, 2011.

[18] O'Shaughnessy S. and Breitinger F., "Malware Family Classification via Efficient Huffman Features," *Forensic Science International: Digital Investigation*, vol. 37, pp. 301192, 2021.

[19] Pektaş A. and Acarman T., "Learning to Detect Android Malware via Opcode Sequences," *Neurocomputing*, vol. 396, pp. 599-608, 2020.

[20] Şahin D., Kural O., Akleylek S., Kılıç E., "Permission-based Android Malware Analysis by Using Dimension Reduction with PCA and LDA," *Journal of Information Security and Applications*, vol. 63, pp. 102995, 2021.

[21] Santos I., Brezo F., Ugarte-Pedrero X., and Bringas P., "Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection," *Information Sciences*, vol. 231, pp. 64-82, 2013.

[22] Singh A., Wadhwa G., Ahuja M., Soni K., and Sharma K., "Android Malware Detection Using LSI-based Reduced Opcode Feature Vector," *Procedia Computer Science*, vol. 173, pp. 291-298, 2020.

[23] Tang J., Li R., Jiang Y., Gu X., and Li Y., "Android Malware Obfuscation Variants Detection Method Based on Multi-granularity Opcode Features," *Future Generation Computer Systems*, vol. 129, pp. 141-151, 2022.

[24] Trabelsi A., Elouedi Z., and Lefevre E., "Decision Tree Classifiers for Evidential Attribute Values and Class Labels," *Fuzzy Sets and Systems*, vol. 366, pp. 46-62, 2019.

[25] Yufang Z., Shiming P., and Jia L., "Improvement and Application of TFIDF Method Based on Text Classification," *Computer Engineering*, vol. 32, no. 19, pp. 76-78, 2006.

Shimiao Sun is currently pursuing the master's degree with the School of Electrical and Information Engineering, Beijing University of Civil Engineering and Architecture. His research interests mainly include data mining, malware detecting and information security.

Yashu Liu received the Ph.D degrees in Beijing Jiaotong University, China. She is currently an assistant professor at Beijing University of Civil Engineering and Architecture. Her research interests include data mining and network security. Now, she majors in detection and classification of malware.