Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 125 (2018) 346–356

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on Smart Computing and Communications, ICSCC 2017, 7-8 December 2017, Kurukshetra, India

Performance Evaluation of Filter-based Feature Selection Techniques in Classifying Portable Executable Files

Shiva Darshan S.L.* and Jaidhar C.D.

Department of Information Technology, National Institute of Technology Karnataka, Surathkal, Mangalore, India

Abstract

The dimensionality of the feature space exhibits a significant effect on the processing time and predictive performance of the Malware Detection Systems (MDS). Therefore, the selection of relevant features is crucial for the classification process. Feature Selection Technique (FST) is a prominent solution that effectively reduces the dimensionality of the feature space by identifying and neglecting noisy or irrelevant features from the original feature space. The significant features recommended by FST uplift the malware detection rate. This paper provides the performance analysis of four chosen filter-based FSTs and their impact on the classifier decision. FSTs such as Distinguishing Feature Selector (DFS), Mutual Information (MI), Categorical Proportional Difference (CPD), and Darmstadt Indexing Approach (DIA) have been used in this work and their efficiency has been evaluated using different datasets, various feature-length, classifiers, and success measures. The experimental results explicitly indicate that DFS and MI offer a competitive performance in terms of better detection accuracy and that the efficiency of the classifiers does not decline on both the balanced and unbalanced datasets.

Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications

Keywords: Feature Selection Technique; Malware; Malware Detection System; Machine Learning; Portable Executable Files;

1. Introduction

Malware is a computer program designed to harm the host system without the user consent. It can morph itself to gain control of the host system, whereby it can access the system level operations in multiple dimensions. It can easily evade the existing detection techniques using various modern obfuscation characteristics. It has grown drastically and has emerged as an insurmountable issue for many anti-malware defensive solutions. Therefore, there is an immediate need of an intricate MDS [9] to resist the attacks caused by such malware.

* Corresponding author.
E-mail address: it15f02.shivadarshan@nitk.edu.in

1877-0509
Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications
10.1016/j.procs.2017.12.046

Generally, traditional malware defensive solutions rely on signature-based detection technique and thus, are vulnerable to unknown malware, if the malware database has not been updated. It extracts static features from the executable file, including binary sequences, function calls, and any other information to determine whether the executable file is a malware. These techniques are said to be more resilient to the malevolent activity of malware, but are easily disrupted by the obfuscation characteristics [14].

The behavioural-based detection technique detects the malware by monitoring the behaviour of the executable file during its runtime [6]. It isolates the malware in an environment called the sandbox [18] and records behaviours such as API calls, system calls, or any other function-based calls triggered upon the operating systems. Thus, it provides a new perspective to analyze the unknown malware. However, it fails to balance between False Positive Rate (FPR) and malware detection rate.

The heuristic-based detection techniques employ the machine learning method to learn the behaviour of an executable file. To detect malware, it deliberately analyzes features such as system calls, API calls, Opcodes, and structural information like header information, etc. [5]. However, in a real scenario, it becomes tedious to examine all the recorded features to acquire the most predominant features for the purpose of the classification operation. Under such circumstances, FST plays a vital role in minimizing the dimensionality of the original feature space and boosts the predictive performance of the classifiers [13]. Information Gain [19], MI [12], Fisher Score [20], Chi-square [7], etc. are examples of FSTs.

In this paper, an MDS has been designed that detects malware based on the extracted information related to Portable Executable Optional Header Fields (PEOHF). Moreover, our prime focus is on the performance analysis of FSTs that are capable of selecting the most relevant features, which are crucial in discriminating between benign and malware PE files. We have employed Single-Stage-Feature-Selector that acquires significant features by adapting the filter-based FST. Further, we compute the score for those extracted PEOHF (features) and then, choose the topmost features as predominant features based on the highest score. From the experimental results, we observe that the features suggested by the DFS and MI were successful in attaining malware detection accuracy of 98.677% for the Balanced Dataset (BD) and 99.308% for the Unbalanced Dataset (UBD) under the 10-fold cross-validation test. Finally, the accuracy shown by the classifiers for both the BD and UBD was considered and the difference was computed to convey that the efficiency of the classifiers does not change much on the BD and the UBD.

The major contributions of this paper are summarized as follows:

- In this work, MDS is designed, implemented, and evaluated using real-world malware samples. The MDS is proficient in precisely distinguishing between malware and benign PE files based on the features recommended by the Single-Stage-Feature-Selector.
- We have employed different FSTs such as DFS, MI, CPD, and DIA to select a compact set of the most significant features to boost the efficiency of the classifier for better accuracy. Four filter-based FSTs were adopted to measure the comparison with the intention of identifying the better one. To evaluate the performance of the different FSTs, two sets of experiments were conducted on the BD and the UBD.
- The experimental results demonstrated that the features recommended by the DFS and MI were successful in achieving malware detection rate of 98.677% for the BD and 99.308% for the UBD under the 10-fold cross-validation test.
- Lastly, evaluation on the BD and the UBD was made to measure the accuracy variation between them. The results clearly indicated that the accuracy difference range of <1% was not of much affect on the efficiency of the classifiers.

The rest of this paper is organized as follows. In Section 2, we study the background of the Portable Executable files. In Section 3, we review earlier research work on filter-based FSTs used in classification. Section 4 elucidates the methodology to effectuate the performance analysis of the filter-based FSTs. Section 5 provides a brief description of the filter-based FSTs used in our experimental work. The obtained empirical results are presented in Section 6. Finally, the conclusion is summarized in Section 7.

2. Background

Portable Executable (PE) is represented as a common file format for all variants of the Windows operating system. Today, majority of the malware target PE files to perform illegitimate actions and acquire necessary information without the user's consent. Most of the existing anti-malware defensive solutions employ a variety of detection techniques and determine the type of file before they parse to detect the embedded malicious data. However, the malware can easily evade the anti-malware solution due to obfuscated characteristics. In order to overcome these issues, a real-time PE-based MDS is very desirable to identify the malware behaviours.

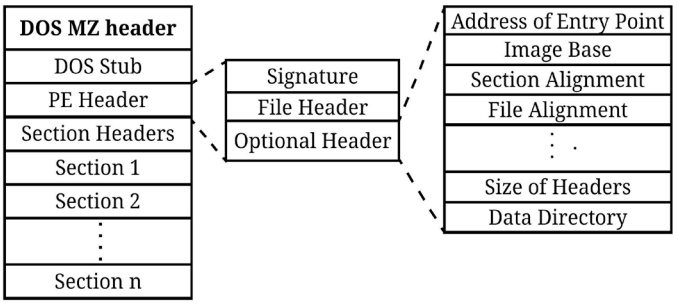

Fig. 1. A general layout of PE file depicting members of the PE Header and PE-Optional Header.

In order to understand the process of disassembling a PE file, it is necessary to perceive the structure of the PE file (Fig. 1). All PE files start with a Disk Operating System (DOS) `MZ Header` and its purpose is to verify whether the file is a valid executable or not when it is running under the DOS system. If the file runs under the DOS environment, then the `DOS stub` is a built-in executable used to display the error message. Next, to the DOS stub, there is a `PE Header`, which contains a necessary information to be used by the PE loader. There exists several sections such as `Section 1`, `Section 2` . . . `Section n` after the PE Header, which stores the data in terms of blocks, and each section data is organized based on common attributes. The PE Header format is an `IMAGE_NT_HEADERS` data structure, which consists of the PE-Signature, File Header, and the Optional Header. The Optional Header is composed of several fields as shown in Fig. 1. For more information, the reader can refer [16].

3. Related Work

Feature representation is essential for malware analysis so as to identify the malware as malware and benign as benign. If they are not sufficiently distinguished, it is difficult for any machine learning classifier to provide accurate prediction. The great difficulty in machine learning based technique is to identify a representative set of features in order to construct a classification model. The FST aims at recognizing a small subset of most supreme features and thereby, minimizes the dimensionality of the original feature space with the removal of noisy or irrelevant features. Several FSTs have been proposed in the literature and can be categorized as filter-based and wrapper-based FSTs. In a filter-based FST, for each feature, a score is computed and the feature is treated as significant based on its score. The wrapper-based FST makes use of the predictive model to generate a score to the features subsets and gives more discriminative power to that particular model. Typical examples of filter-based FST include Document Frequency (DF) [15], Information Gain (IG) [11][19][4], Fisher Score (FS) [15][20], Max-Relevance (MR) [26], MI [25][21][3], Odds Ratio [8][1], and Chi-square [4][7][2].

The malware analysis approach was proposed and was implemented by adapting Opcode as the file representation method [15]. The Opcode sequence was derived by disassembling an executable file, and then constructed using the Opcode N-grams as features for the classification process, but at the same time, they showed that the Document Frequency FST as accurate for most of the uppermost features. Meanwhile, the Fisher Score predominantly executed well when the number of features were few, i.e., for the topmost 50 and 100 features.

The detection and classification of the malware, which appears in wild, was addressed by employing renowned machine learning classifiers [11]. In their work, the experiments were conducted by gathering 1971 benign and 1651

malware executable files and by encoding each of them using the N-gram feature technique. The process resulted in generating more than 255 million predominant N-gram features and this led to high computational overhead. This issue was overcome using the IG FST that selected the topmost features based on the highest score and evaluated the performance of the different classifiers. The evaluation results suggested that 500 N-gram features with boosted J48 classifier produced an area under the curve of 0.996.

An approach to detect obfuscated malware using the frequency of Opcode sequences to construct a representation of executable files was proposed [17]. The processing of Opcode sequences was done by disassembling the executable files. Further, the authors built an Opcode profile comprising of Opcodes computing the relevance of each Opcode based on the frequency of the appearance of each of them in both the benign and malware classes by employing MI FST. The results showed that the density estimation works well for the Naive Bayes classifier.

The authors introduced a real-time PE MDS based on the interpretation of the PE-Optional Header information [7]. Their system used a combination of the Chi-square and the Phi coefficient FSTs to remove irrelevant features having Chi-square score lesser than 3.84. This allowed them to achieve better accuracy.

An Intelligent Malware Detection System based on the Windows API calls using association mining was proposed [26]. As not all extracted API calls contribute to malware detection, the Max relevance FST was applied to select the distinct features and to achieve better classifier accuracy. Accordingly, the authors successfully obtained an accuracy of 93.07%.

4. Methodology

An overview of the proposed MDS is shown in Fig. 2. It has two phases: 1) Training phase and 2) Prediction phase. The training phase is used to build a training file, which is needed to train the classifier. The prediction phase measures the detection ability of the trained classifier. However, the main objective of the proposed approach is to exhibit the performance analysis of the filter-based FSTs.

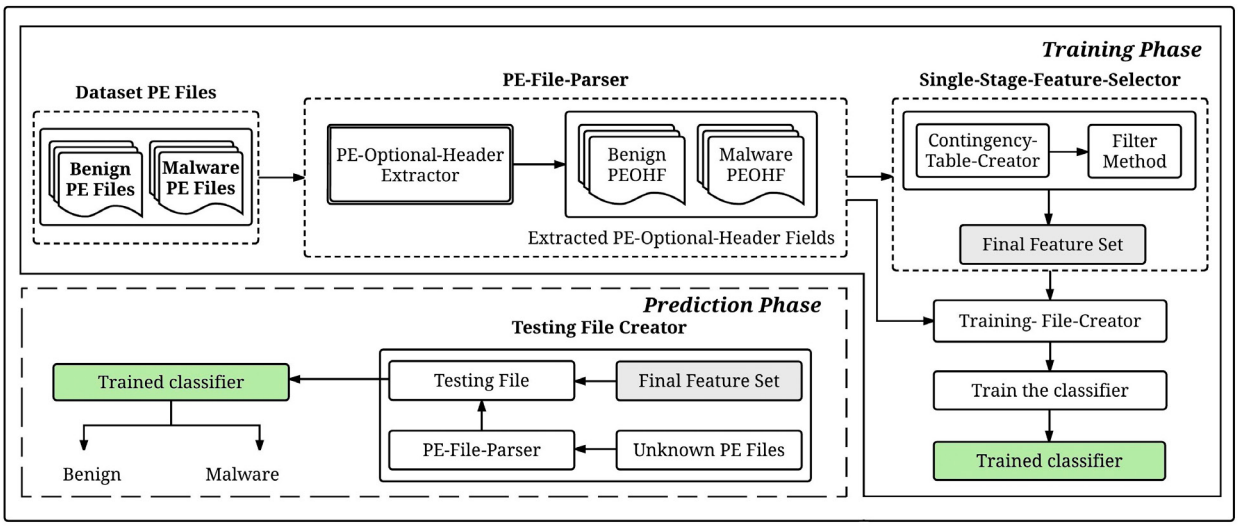

Fig. 2. Process architecture overview.

To train and evaluate the performance of the MDS, a dataset of PE files were collected from various sources (see Section 6) separately as BD and UBD for further processing. Both the datasets were used for the purpose of effective analysis of the filter-based FSTs. Each of these was supplied onto the PE-File-Parser individually to initiate the analysis task.

4.1. Training Phase

In the training phase, the MDS is provided with a training set of benign and malware PE files. Each PE file is parsed to extract the features related to the Optional Header fields. The representative features of the PE files were processed

using the FST to obtain the most informative features to prepare a training file required to train the classifiers. To accomplish this task, the training phase utilized the PE-File-Parser, Single-Stage-Feature-Selector, and Training-File-Creator as its essential components.

4.1.1. PE-File-Parser

The prime goal of the PE-File-Parser is to extract the PE Optional Header Fields names with their corresponding values as features from the dataset of PE files and output benign PEOHF and malware PEOHF files. This task was accomplished by employing its subcomponent PE-Optional-Header Extractor, which uses a python module called pefile[1] to derive the Optional Header features from the input PE files. However, the extracted set of features size was quite large due to the presence of noisy or irrelevant features. Therefore, the Single-Stage-Feature-Selector was utilized to identify the most crucial features required to prepare a training file to train the classifier and simultaneously reduce the dimensionality of the feature set.

4.1.2. Single-Stage-Feature-Selector

It plays a crucial role in preserving the informative features and to detect the unknown malware accurately from a number of other benign PE files. To perform this desired operation, it uses the Contingency-Table-Creator as one of its subcomponents.

In the Single-Stage-Feature-Selector, the task of the Contingency-Table-Creator is to create a Contingency Table (CT) that provides the frequency distribution of each feature such as the presence and absence in Benign PEOHF and Malware PEOHF files as an integer count. The FST utilizes the CT to generate a score for each feature, and based on the score, the features are selected as prominent features. In this regard, the four filter-based FSTs chosen in this work are DFS, MI, CPD, and DIA. Further, to demonstrate their efficiency, the four Final Feature Sets (FFSs) were constructed by selecting the topmost features of different thresholds. However, each FFS consists of features suggested by the individual FST such as the DFS, MI, CPD, and DIA. Finally, a training file was built using the FFS with the PEOHF files corresponding to the training samples. Lastly, the classifier was trained using the constructed training file.

4.1.3. Final Feature Set

FFS is a set that consists of distinct benign (malware) features of the benign (malware) PE files. These features are obtained after carrying out all the processing steps with no further elimination of features, and therefore, named as FFS. The features available in the FFS are used to prepare a training file as well as testing files crucial in measuring the efficiency of the classifiers.

4.1.4. Training-File-Creator

Training-File-Creator creates a training file essential to train the classifiers. It parses the training dataset of the benign and malware PEOHF files with FFS features in order to create a training file.

4.2. Prediction Phase

In the prediction phase, the Testing-File-Creator is used to create a testing file necessary to appraise the predictive performance of the trained classifiers. It makes use of FFS features and the output of the PE-File-Parser to deliver a testing file. The generated testing file is sent to the trained classifier to ascertain whether the test input file is benign or malware.

5. Filter-based Feature Selection Techniques

This section provides a description of the four filter-based FSTs used in this work for the purpose of performance analysis. Minimization of the immense dimensionality of the feature space is of greatest concern in malware classifi-

[1] https://code.google.com/p/pefile/

cation. The FST identifies features that have high classification potential and filters features that are noisy or irrelevant. This results in massive reduction of computational cost.

The following are the different filter-based feature selection techniques utilized in this work:

(i) Distinguishing Feature Selector

DFS [24] evaluates the contribution of the features in a representative vector to the class discrimination in a probabilistic approach and computes the score for each feature as per Eq. 1,

$$DFS(f) = \sum_{i=1}^{N} \frac{P(c_i|f)}{P(\bar{f}|c_i) + P(f|\bar{c_i}) + 1} \tag{1}$$

Where, N is the total number of categories, $P(c_i|f)$ denotes the conditional probability of category c_i when the feature f is present, $P(\bar{f}|c_i)$ signifies the conditional probability of the category c_i when the feature f is absent, and $P(f|\bar{c_i})$ is the conditional probability of feature f given the category other than c_i.

(ii) Mutual Information

MI [23] calculates the mutual dependence of any two random variables. It measures the decline in uncertainty about one random variable as a function of the other. If the MI score between two random variables is zero, then the variables are independent, and those with the highest score will have a large reduction in uncertainty. The MI score for a feature and the category pair is computed as per Eq. 2.

$$MI(f, c_k) = \sum_{v_f \in \{1,0\}} \sum_{v_{c_k} \in \{1,0\}} P(f = v_f, c_k = v_{c_k}) \ln \frac{P(f = v_f, c_k = v_{c_k})}{P(f = v_f)P(c_k = v_{c_k})} \tag{2}$$

Where, f indicates the feature that takes the value $v_f = \{1, 0\}$. If the value of $v_f = 0$ then the document does not contain the feature f, and if $v_f = 1$, it indicates that the document contains the feature f. c_k is the category that takes the value one, i.e., $v_{c_k} = 1$, when the document is present in a category c_k, otherwise, the value is zero, i.e., $v_{c_k} = 0$ indicating the absence of the document in the category c_k.

(iii) Categorical Proportional Difference

CPD [22] calculates the degree to which a feature distinguishes a specific category from other categories. The attainable values for CPD are limited to the interval (-1, 1). The CPD score near to -1 denotes that the feature is present in most of the documents in all the categories. If the score is equal to one, it represents that the feature is present in the document of only one category. The CPD score for the feature f in the category c_k is formulated as per Eq. 3 and 4.

$$CPD(f, c_k) = \frac{N_{f,c_k} - N_{f,\overline{c_k}}}{N_f} \tag{3}$$

The CPD for the feature f is the ratio associated with the category c_k for which the value is highest.

$$CPD(f) = max_k\{CPD(f, c_k)\} \tag{4}$$

(iv) Darmstadt Indexing Approach

DIA [10] FST considers the properties of the features, categories, and pair-wise relationships as a dimension. The DIA score for the feature f is calculated as per Eq. 5.

$$DIA(f, c_k) = \frac{N_{f,c_k}}{N_f} \tag{5}$$

Where N_{f,c_k} are the documents containing feature f in the category c_k and N_f denotes the number of documents containing the feature f.

6. Experimental Results and Analysis

Our experimental data consists of two datasets, BD and UBD. BD consists of an equivalent number of benign and malware PE files. In UBD, the malware PE files are twice the count of the benign PE files as shown in Table 1. The benign PE files include the Windows system files collected from a freshly installed Windows XP virtual machine. The malware PE files were downloaded from the public source VirusShare[2]. To ensure that all the files in the dataset are correctly labelled, both the datasets were scanned with more than 40 anti-malware engines available on VirusTotal[3].

Table 1. Experiment Dataset details

	Benign PE files	Malware PE files
BD: Balanced Dataset	200	200 (trojan(100) + backdoor(100))
UBD: Unbalanced Dataset	200	400 (trojan(100) + backdoor(100) + rootkit(200))

As explained earlier (in Section 4), the PE-File-Parser receives both the benign and malware PE files to extract information related to the Optional Header Fields as features. Each extracted feature is indicated by its name and its corresponding value. The derived features are gathered to form an original feature space. Further, to attain the best features, FSTs are applied separately onto the original feature space to get the score for each feature separately, and the topmost K number of features is chosen based on the highest score. Four different FSTs such as DFS, MI, CPD, and DIA are used with the intention of identifying the best one. Experiments were conducted for different values of K such as 25, 50, 75, and 100. These best features were processed to prepare a training file as well as testing files, which were supplied to the classifiers in order to determine which classifier achieved the best malware detection rate with low FPR.

The main aim was to perform a comparative analysis of the four different FSTs and to identify the best FST with the potency to recommend the most significant features. The classifier predictive performance utterly depends on the features used in the training file. From that perspective, the FFS generated consists of features recommended by the Single-Stage-Feature-Selector. The FFS features are employed as final features since there is no further feature elimination, and these features are used to prepare the training as well as the testing files desirable to measure the efficiency of the classifiers. Six different classifiers such as the Sequential Minimal Optimization (SMO), Simple Logistic, Logistic, J48, Random Forest, and Random Tree available in WEKA were used to know which classifier outperformed for the derived FFS. The performance of each classifier was evaluated using evaluation metrics such as True Positive Rate (TPR), FPR, and accuracy.

Two sets of experiments were carried out by us: In the first set of experiments, the BD was considered. The PE-File-Parser processed both the benign and malware PE files to extract the entire PE Optional Header Fields information as features. The extracted data was then stored into an appropriate output file and a separate output file was maintained for each individual PE file. Each extracted data was treated as an individual feature, which is a prerequisite task. Since 200 malware PEOHF and 200 benign PEOHF files produced by the PE-File-Parser were considered, these files were directly sent as input to the chosen FSTs separately. At first, the DFS FST was executed on distinct features of 1323 to generate the DFS score using Eq. 1. Since all these 1323 features cannot be used to train the classifier, the predominant features were identified and selected as crucial features based on their highest score. To evaluate the performance of each FST, the topmost K number of features were selected in increments of 25, i.e., K=25, K=50, K=75, and K=100. Accordingly, the corresponding FFSs were prepared. Similarly, the other three FSTs were applied on 1323 features separately to get the corresponding scores using Eq. 2, Eq. 4, and Eq. 5. Subsequently, the topmost K number of features was selected in increments of 25, i.e., 25, 50, 75, and 100 and the FFSs were constructed separately.

From the experimental results depicted in Table 2, we noticed that the SMO, Simple Logistic, and J48 classifiers outperformed by achieving maximum identical accuracy of 98.677% with 0.013 FPR for all the topmost K number of features such as 25, 50, 75, and 100 recommended by the DFS and MI FSTs. The performance was not much appreciable when the same training file and testing files were supplied to other classifiers such as Logistic, Random

[2] https://virusshare.com
[3] https://www.virustotal.com/

Forest, and Random Tree. The accuracy achieved by each of them is tabulated in Table 2. At the same time, we noticed that the features recommended by the CPD and DIA FSTs for the topmost K number of features (K=25, K=50, K=75, and K=100) underperformed by achieving drastically reduced accuracy.

The second sets of experiments were conducted with the UBD mentioned in Table 1. The operational steps performed in this set of experiments are similar to the first set of experiments, except for the number of files. In the second set of experiments, the PE-File-Parser was fed 200 benign and 400 malware PE files in order to extract all the PE Optional Header Fields using the PE-Optional-Header-Extractor. The PE-File-Parser produced benign (200) and malware (400) PEOHF files and these files were supplied as input to the FSTs chosen in this experimental work. The DFS FST was executed on 1877 features to compute a score for each feature using Eq. 1.

Furthermore, the same original features space of 1877 features was supplied as input to the other FSTs one after another to determine the score for the features using Eq. 2, Eq. 4, and Eq. 5. The uppermost K number of features was selected in increments of 25 up to 100 to prepare a separate FFS, and each of them was used to build a training file as well as testing files to measure the efficiency of the classifiers. The accuracy produced by the different classifiers is presented in Table 2. In the case of UBD, the SMO, and Simple Logistic classifiers outperformed by accomplishing highest equivalent accuracy of 99.308% with 0.014 FPR for all the topmost number of features (K=25, K=50, K=75, and K=100) recommended by the DFS and MI FSTs. The other classifiers such as J48, Logistic, Random Forest, and Random Tree underachieved for the same topmost number of features (K=25, K=50, K=75, and K=100) and the accuracy accomplished by each of them is also tabulated in Table 2. FSTs such as the CPD and DIA were found to be inefficient and consistently performed in a way similar to in the first set of experiments.

The two sets of experiments were computed substantially and analyzed thoroughly to decide the best FST based on the accuracy produced by the classifier. In this direction, the obtained and analyzed results proved that FST certainly provided the most significant features based on the computed score, but all the features may not contribute to detect the malware.

Table 2. Accuracy of different classifiers on BD and UBD

Feature Selection Technique	No. of features	Accuracy (%)																	
		SMO			Simple Logistic			Logistic			J48			Random Forest			Random Tree		
		BD	UBD	Diff	BD	UBD	Diff	BD	UBD	Diff	BD	UBD	Diff	BD	UBD	Diff	BD	UBD	Diff
DFS	25	98.677	99.308	0.631	98.677	99.308	0.631	96.296	98.270	1.974	98.677	99.135	0.458	98.148	99.308	1.160	98.148	99.135	1.160
MI	25	98.677	99.308	0.631	98.677	99.308	0.631	97.090	97.232	0.142	98.677	99.135	0.458	98.677	99.135	0.458	98.677	98.962	0.285
CPD	25	53.968	67.474	13.506	56.085	67.474	11.389	56.614	67.474	10.860	52.910	67.474	14.564	56.614	67.474	10.860	56.614	67.474	10.86
DIA	25	51.852	67.301	15.449	51.852	67.474	15.622	51.323	67.301	15.978	50.265	67.474	17.209	51.323	67.474	16.151	52.116	67.301	15.185
DFS	50	98.677	99.135	0.458	98.677	99.308	0.631	98.148	98.616	0.468	98.677	99.135	0.458	98.677	99.308	0.631	98.413	98.270	0.143
MI	50	98.677	99.308	0.631	98.677	99.308	0.631	96.032	98.789	2.757	98.677	99.135	0.458	98.413	99.135	0.722	96.296	99.135	2.839
CPD	50	61.905	67.820	5.915	62.169	67.474	5.305	63.492	67.820	4.328	56.878	67.474	10.596	63.757	67.820	4.063	63.228	67.820	4.592
DIA	50	53.439	67.647	14.208	53.439	67.474	14.035	53.704	67.647	13.943	51.587	67.474	15.887	53.439	67.820	14.381	53.704	67.647	13.943
DFS	75	98.677	99.308	0.631	98.677	99.308	0.631	98.413	98.962	0.549	98.677	99.135	0.458	98.148	99.135	0.987	98.413	98.616	0.203
MI	75	98.677	99.308	0.631	98.677	99.308	0.631	98.148	98.097	0.051	98.677	99.135	0.458	98.413	99.308	0.895	97.619	98.270	0.651
CPD	75	63.757	68.512	4.755	61.905	68.166	6.261	64.286	68.512	4.226	56.878	68.166	11.288	64.286	68.512	4.226	63.492	68.512	5.020
DIA	75	57.672	67.647	9.975	57.407	67.474	10.067	57.672	67.647	9.975	55.027	67.474	12.447	57.672	67.820	10.148	57.672	67.647	9.975
DFS	100	98.677	99.308	0.631	98.677	99.135	0.458	98.148	98.616	0.468	98.677	99.135	0.458	98.413	99.308	0.895	97.355	98.443	1.088
MI	100	98.677	99.308	0.631	98.677	99.308	0.631	97.884	98.270	0.386	98.677	99.135	0.458	98.148	99.135	0.987	97.619	98.097	0.478
CPD	100	69.550	69.841	0.291	66.402	69.031	2.629	70.370	69.550	0.820	62.434	68.858	6.424	70.106	69.550	0.556	69.312	69.550	0.238
DIA	100	63.492	67.474	3.982	63.227	67.474	4.247	64.286	67.647	3.361	60.318	67.474	7.156	64.286	67.820	3.534	64.021	67.647	3.626

Diff: **|BD – UBD|**

6.1. Evaluation on Balanced and Unbalanced Datasets

In order to measure accuracy variation between BD and UBD, we calculated the difference. Table 2 demonstrates that the difference in classifier accuracy is not of much significance.

We noticed that the SMO classifier produced an accuracy of 98.677% for BD and 99.308% for UBD with 25 features recommended by the DFS. The difference found was 0.631% (i.e., |98.677 − 99.308|). Further, the SMO classifier accuracy difference between BD and UBD was 0.458%, 0.631%, and 0.631%, respectively, for the other three FFSs of sizes 50, 75, and 100 features suggested by the DFS. This indicates that the classifier accuracy did not decline for a UBD and justified that the accuracy variation was in the range of 0.458% to 0.631%. Similarly, when the same 25, 50, 75, and 100 features were applied to the Simple Logistic, Logistic, J48, Random Forest, and Random Tree classifiers, their counselled range was 0.458% to 0.631%, 0.468% to 1.974%, 0% to 0.458%, 0.631% to 1.16%, and 0.143% to 1.16%, respectively. The accuracy difference in the range of 0% to 0.631% indicated better classification and specified that the efficiency of the classifiers did not minimize much and good enough for both the BD and UBD. Other FST such as MI was also efficient and guaranteed that the classifier performed well in the range of 0.458% to 0.631%.

On the other hand, the classifier performance was inefficient for the top features selected based on the highest score recommended by the CPD and DIA FSTs. The expected range to determine the better classifier accuracy was very high. Accordingly, as per the observations in Table 2, the SMO classifier gained an accuracy of 53.968% for BD and 67.474% for UBD with the highest difference of 13.506% for the top 25 features selected based on the CPD score. Subsequently, when the SMO classifier accuracy was checked with the foremost 100 features based on the CPD score, it attained 69.550% for BD and 69.841% for UBD with least difference of 0.291%. The other classifiers performance was not remarkable and achieved very less accuracy with maximum distinguishable range. The observed accuracy difference range for the Simple Logistic classifier was between 0.112% -15.622%, for the Logistic classifier it was between 0.051% - 15.978%, for the J48 classifier between 0.458% - 17.209%, for the Random Forest classifier between 0.556% - 16.151%, and lastly, for the Random Tree classifier between 0.143% - 15.185%. The evaluation on BD and UBD showed that the accuracy difference range (i.e., <1%) did not have much impact on the efficiency of the classifier and resulted in poor performance.

Table 3. Comparison of TPR achieved by different classifiers on different feature lengths of BD and UBD

No. of Features	Classifiers	DFS		MI		CPD		DIA	
		BD	UBD	BD	UBD	BD	UBD	BD	UBD
25	**SMO**	0.987	0.993	0.987	0.993	0.540	0.675	0.519	0.673
	Simple Logistic	0.987	0.993	0.987	0.993	0.561	0.675	0.519	0.675
	Logistic	0.963	0.983	0.971	0.972	0.566	0.675	0.513	0.673
	J48	0.987	0.991	0.987	0.991	0.529	0.675	0.503	0.675
	Random Forest	0.981	0.993	0.987	0.991	0.566	0.675	0.513	0.675
	Random Tree	0.981	0.993	0.987	0.990	0.566	0.675	0.521	0.673
50	**SMO**	0.987	0.991	0.987	0.993	0.632	0.678	0.534	0.676
	Simple Logistic	0.987	0.993	0.987	0.993	0.627	0.675	0.534	0.675
	Logistic	0.979	0.986	0.968	0.988	0.632	0.678	0.537	0.676
	J48	0.984	0.991	0.984	0.991	0.503	0.675	0.503	0.675
	Random Forest	0.987	0.993	0.987	0.991	0.635	0.678	0.534	0.678
	Random Tree	0.976	0.983	0.974	0.991	0.635	0.678	0.537	0.676
75	**SMO**	0.987	0.993	0.987	0.993	0.638	0.685	0.577	0.676
	Simple Logistic	0.987	0.993	0.987	0.988	0.619	0.682	0.574	0.675
	Logistic	0.984	0.990	0.981	0.981	0.643	0.685	0.577	0.676
	J48	0.987	0.991	0.987	0.991	0.569	0.682	0.550	0.675
	Random Forest	0.981	0.991	0.976	0.993	0.643	0.685	0.577	0.678
	Random Tree	0.984	0.986	0.984	0.983	0.635	0.685	0.577	0.676
100	**SMO**	0.987	0.991	0.987	0.995	0.698	0.696	0.619	0.675
	Simple Logistic	0.987	0.991	0.987	0.990	0.680	0.690	0.616	0.675
	Logistic	0.987	0.986	0.979	0.983	0.701	0.696	0.632	0.676
	J48	0.984	0.991	0.984	0.991	0.545	0.689	0.545	0.675
	Random Forest	0.984	0.993	0.981	0.991	0.701	0.696	0.632	0.678
	Random Tree	0.976	0.984	0.979	0.981	0.696	0.696	0.630	0.676

6.2. Analysis of Evaluation Metrics

The performance of the MDS was analyzed in terms of popular evaluation metrics such as the TPR and FPR. The TPR (FPR) indicated that the malware instances were correctly (incorrectly) classified. For any MDS, it is desired that the TPR should be high and the FPR should be as low as possible. Table 3 and Table 4 summarize the TPR and FPR for different topmost K number of features such as 25, 50, 75, and 100 recommended by different FSTs such as the DFS, MI, CPD, and DIA.

From Table 3 statistics, we can easily infer that MDS achieved high TPR of 0.987 for features of different thresholds in terms of 25, 50, 75, and 100 recommended by the DFS and MI on the BD with SMO and Simple Logistic classifiers. Similarly, maximum TPR of 0.993 was accomplished by the SMO and Simple Logistic classifiers for all the foremost features of various thresholds (25, 50, 75, and 100) suggested by the DFS and MI. On the other side, the features suggested by the CPD and DIA attained very low TPR.

Lower FPR value represents the best FST. However, the classifiers were evaluated with the features recommended by the DFS, MI, CPD, and DIA. Table 4 shows the FPR achieved by the classifiers. The SMO and Simple Logistic classifiers attained the lowest FPR of 0.013 for all the topmost numbers of features (25, 50, 75, and 100) recommended by the DFS and MI on the BD. Correspondingly, the same classifiers were successful in obtaining minimum FPR of 0.014 for the features suggested by the DFS and MI for the UBD. Moreover, the same classifiers achieved slightly high FPR for all the features of different thresholds (25, 50, 75, and 100) suggested by the CPD and DIA FSTs. In comparison, it is evident that FSTs such as the DFS and MI were able to influence the classifiers to attain highest TPR with lowest FPR.

Table 4. Comparison of FPR achieved by different classifiers on different feature lengths of BD and UBD

No. of Features	Classifiers	DFS		MI		CPD		DIA	
		BD	UBD	BD	UBD	BD	UBD	BD	UBD
25	**SMO**	0.013	0.014	0.013	0.014	0.455	0.675	0.477	0.676
	Simple Logistic	0.013	0.014	0.013	0.014	0.435	0.675	0.476	0.675
	Logistic	0.037	0.025	0.029	0.033	0.429	0.675	0.483	0.676
	J48	0.013	0.015	0.013	0.015	0.466	0.675	0.493	0.675
	Random Forest	0.018	0.012	0.013	0.012	0.429	0.675	0.488	0.675
	Random Tree	0.018	0.012	0.013	0.016	0.429	0.675	0.474	0.676
50	**SMO**	0.013	0.014	0.013	0.014	0.364	0.668	0.462	0.668
	Simple Logistic	0.013	0.014	0.013	0.014	0.369	0.675	0.461	0.675
	Logistic	0.021	0.018	0.032	0.022	0.364	0.668	0.458	0.668
	J48	0.013	0.015	0.013	0.015	0.503	0.675	0.503	0.675
	Random Forest	0.013	0.014	0.013	0.015	0.361	0.668	0.463	0.668
	Random Tree	0.024	0.019	0.026	0.012	0.361	0.668	0.458	0.668
75	**SMO**	0.013	0.014	0.013	0.014	0.359	0.653	0.419	0.668
	Simple Logistic	0.013	0.014	0.013	0.014	0.377	0.660	0.421	0.675
	Logistic	0.016	0.013	0.018	0.023	0.353	0.653	0.419	0.668
	J48	0.013	0.015	0.013	0.015	0.427	0.660	0.445	0.675
	Random Forest	0.018	0.015	0.024	0.014	0.353	0.653	0.419	0.668
	Random Tree	0.016	0.018	0.016	0.025	0.361	0.653	0.419	0.668
100	**SMO**	0.013	0.014	0.013	0.014	0.298	0.632	0.378	0.672
	Simple Logistic	0.013	0.014	0.013	0.014	0.317	0.642	0.380	0.675
	Logistic	0.016	0.015	0.021	0.017	0.296	0.632	0.364	0.668
	J48	0.013	0.015	0.013	0.015	0.451	0.646	0.451	0.675
	Random Forest	0.016	0.014	0.019	0.015	0.296	0.632	0.364	0.668
	Random Tree	0.024	0.021	0.021	0.020	0.301	0.632	0.366	0.668

7. Conclusion

The MDS is proficient in distinguishing malware and benign PE files precisely based on the features recommended by the Single-Stage-Feature-Selector. The prime task of this work was to investigate the effectiveness of filter-based FSTs such as the DFS, MI, CPD, and DIA in classifying the PE files as benign or malware. The experiments carried out were evaluated using different classifiers available in the WEKA tool. From the experimental observation, it was

found that the best FSTs were DFS and MI, since the features suggested by them resulted in obtaining better classifier accuracy. The classifiers substantially performed well on both the BD and UBD for different feature lengths of 25, 50, 75, and 100. The accuracy difference calculation manifested that the range specification of <1% did not affect the efficiency of the classifiers.

References

[1] Ajay Kumara, M, A., Jaidhar, C, D., 2017a. Automated multi-level malware detection system based on reconstructed semantic view of executables using machine learning techniques at vmm. Future Generation Computer Systems .

[2] Ajay Kumara, M, A., Jaidhar, C, D., 2017b. Leveraging virtual machine introspection with memory forensics to detect and characterize unknown malware using machine learning techniques at hypervisor. Digital Investigation .

[3] Ambusaidi, M.A., He, X., Nanda, P., Tan, Z., 2016. Building an intrusion detection system using a filter-based feature selection algorithm. IEEE transactions on computers 65, 2986–2998.

[4] Bai, J., Wang, J., Zou, G., 2014. A malware detection scheme based on mining format information. The Scientific World Journal .

[5] Bazrafshan, Z., Hashemi, H., Fard, S.M.H., Hamzeh, A., 2013. A survey on heuristic malware detection techniques, in: Information and Knowledge Technology (IKT), 2013 5th Conference on, IEEE. pp. 113–120.

[6] Belaoued, M., Mazouzi, S., 2014. Statistical study of imported apis by pe type malware, in: Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on, IEEE. pp. 82–86.

[7] Belaoued, M., Mazouzi, S., 2015. A real-time pe-malware detection system based on chi-square test and pe-file features, in: IFIP International Conference on Computer Science and its Applications_x000D_, Springer. pp. 416–425.

[8] Chen, J., Huang, H., Tian, S., Qu, Y., 2009. Feature selection for text classification with naïve bayes. Expert Systems with Applications 36, 5432–5435.

[9] Egele, M., Scholte, T., Kirda, E., Kruegel, C., 2012. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR) 44, p. 6.

[10] Jin, W., Srihari, R.K., 2007. Graph-based text representation and knowledge discovery, in: Proceedings of the 2007 ACM symposium on Applied computing, ACM. pp. 807–811.

[11] Kolter, J.Z., Maloof, M.A., 2006. Learning to detect and classify malicious executables in the wild. Journal of Machine Learning Research 7, pp. 2721–2744.

[12] Li, S., Xia, R., Zong, C., Huang, C.R., 2009. A framework of feature selection methods for text categorization, in: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, Association for Computational Linguistics. pp. 692–700.

[13] Li, Y., Li, T., Liu, H., 2017. Recent advances in feature selection and its applications. Knowledge and Information Systems , pp. 1–27.

[14] Moser, A., Kruegel, C., Kirda, E., 2007. Limits of static analysis for malware detection, in: Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual, IEEE. pp. 421–430.

[15] Moskovitch, R., Feher, C., Tzachar, N., Berger, E., Gitelman, M., Dolev, S., Elovici, Y., 2008. Unknown malcode detection using opcode representation. Intelligence and Security Informatics , pp. 204–215.

[16] Peikari, C., Chuvakin, A., 2004. Security Warrior: Know Your Enemy. " O'Reilly Media, Inc.".

[17] Peng, H., Long, F., Ding, C., 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on pattern analysis and machine intelligence 27, 1226–1238.

[18] Qiao, Y., Yang, Y., He, J., Tang, C., Liu, Z., 2014. Cbm: free, automatic malware analysis framework using api call sequences, in: Knowledge Engineering and Management. Springer, pp. 225–236.

[19] Reddy, D.K.S., Pujari, A.K., 2006. N-gram analysis for computer virus detection. Journal in Computer Virology 2, pp. 231–239.

[20] Sami, A., Yadegari, B., Rahimi, H., Peiravian, N., Hashemi, S., Hamze, A., 2010. Malware detection based on mining api calls, in: Proceedings of the 2010 ACM symposium on applied computing, ACM. pp. 1020–1025.

[21] Santos, I., Brezo, F., Nieves, J., Penya, Y.K., Sanz, B., Laorden, C., Bringas, P.G., 2010. Idea: Opcode-sequence-based malware detection, in: International Symposium on Engineering Secure Software and Systems, Springer. pp. 35–43.

[22] Simeon, M., Hilderman, R., 2008. Categorical proportional difference: A feature selection method for text categorization, in: Proceedings of the 7th Australasian Data Mining Conference-Volume 87, Australian Computer Society, Inc.. pp. 201–208.

[23] Singh, B., Kushwaha, N., Vyas, O.P., et al., 2014. A feature subset selection technique for high dimensional data using symmetric uncertainty. Journal of Data Analysis and Information Processing 2, p. 95.

[24] Uysal, A.K., Gunal, S., 2012. A novel probabilistic feature selection method for text classification. Knowledge-Based Systems 36, pp. 226–235.

[25] Yang, Y., Pedersen, J.O., 1997. A comparative study on feature selection in text categorization, in: Icml, pp. 412–420.

[26] Ye, Y., Wang, D., Li, T., Ye, D., Jiang, Q., 2008. An intelligent pe-malware detection system based on association mining. Journal in computer virology 4, pp. 323–334.