Chapter 3
Proposed Effective Feature Extraction and Selection for Malicious Software Classification

Cho Cho San and Mie Mie Su Thwin

3.1 Introduction

Due to the growing of malicious code in the information technology and cyber security, the knowledge and understanding of new unknown malicious code or program protection is an important trend in the suspicious software detection system using machine learning (ML) methods. There are normally two ways to carry out the suspicious software analysis, static and dynamic analysis for detecting and finding the new malware.

The process of analyzing the software or program without executing the program is referred to static analysis that can classify and detect the known and unknown malicious code [1]. It is the first approach for analyzing and detecting the malicious software that has been stated in [2]. The static malware analysis examines the assembly code of binary file to identify the retrieve flow of code and sequential instructions without actually executing the executable sample [7]. Reverse engineering is a common approach to extract static information from a binary. Disassembly and hexadecimal dumping of binary file are the two main techniques to pre-process and get static features from the sample [4]. A disassembler tool can be used to decompile Windows executable files, such as IDA Pro and OllyDbg, that display assembly instructions, provide information about the malware, and extract patterns to identify the attacker' desire.

In [5] static or code analysis is faster and simple than the other analysis. However, it cannot be effective for obfuscated and complex malware and might leave the significant malicious behaviors. Additionally, the obfuscation techniques, polymorphism, metamorphism, compression, encryption, and run-time packing,

C. C. San (✉) · M. M. S. Thwin
Cyber Security Research Lab, University of Computer Studies, Yangon, Myanmar
e-mail: chochosan@ucsy.edu.mm; drmiemiesuthwin@ucsy.edu.mm

G.R. Sinha (ed.), *Advances in Biometrics*,
https://doi.org/10.1007/978-3-030-30436-2_3

introduced by malware authors, lead static analysis complicated, time-consuming, and nearly unfeasible. Thus, malware analysts and researchers developed and performed dynamic method which is more effective than static to the obfuscation techniques.

The dynamic or run-time analysis method performs the running or executing the malware in a safe and controlled environment. It inspects the malware dynamic behavior to decide which function or system calls are intercepted sequentially, which is also called hooking method, by a malware to determine the nature of suspicious file behavior in a virtual machine environment [7].

Another way to analyze the malicious file is using sandbox. NIST defines the sandbox in [8]: it is a security model where applications/programs are run within a safe environment or a sandbox. Sandboxes record the changes of file system, registry keys, and network traffic and then generate standardized report format. There are common sandboxes that can leverage a quick analysis for malicious files. Sandboxes such as GFI Sandbox, Anubis, Joe Sandbox, ThreatExpert, and Cuckoo Sandbox can analyze malware for free. Cuckoo Sandbox has been used to discover the malware behavior patterns in our approach.

Figure 3.1 shows different types of static and dynamic features used for malware detection and classification in recent researches.

According to [4] dynamic features can be derived from host trace and network trace-based features. The activities of internal memory, files and file system, registry, hardware performance counters, and status of running processes of host are considered as host trace features. The proposed system used the API n-gram (where $n = 1,2,3$), from host trace of dynamic analysis. API is the most common used attributes or features in malware analysis. The proposed system used the Cuckoo Sandbox for analyzing the samples. The APISTATS from JSON report of sandbox

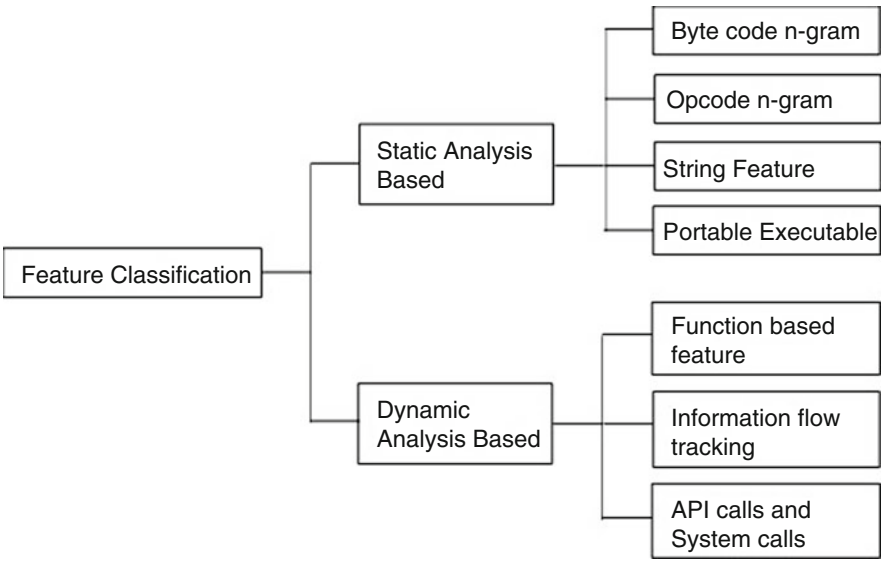

Fig. 3.1 Features based on types of analysis

are extracted as feature APIs for classification system. The number of extracted API from APISTATS is 306 for malware and benign. The malware samples contain six different categories such as Adware, Backdoor, Downloader, Trojan, Virus, and Worm.

The proposed system applies the n-gram technique as the extracted attributes or features depend on each other. However, bigram will only be applied because the system concerns the processing/run time of the classifier. And the experiment shows that these grams are enough to distinguish malware from benign. The proposed system applies two machine learning (ML) techniques and two feature selection (FS) approaches to differentiate the malware from benign. The sklearn ML library [6] has been used to apply these ML techniques and FS approaches. The next section will describe the related works that perform the classification, detection, and feature selection through dynamic analysis.

The content of the paper is structured into five sections. The recent related research work is described in the next section, and the proposed feature extraction, selection, and classification system are provided in Sect. 3.3. Section 3.4 supports the experiment and results discussion, respectively. Last section highlights the conclusion and the research plans for future.

3.2 Related Work

This section presents the current research works that have done through dynamic or run-time analysis. Researchers are now working by proposing the hybrid nature on both analysis and features such as hybrid analysis and hybrid features combination and feature fusion methodologies. And most of the attributes that use to detect and identify the malicious programs are API which is based on the number of occurrences of API (frequency), the order of API (sequence), and system calls.

The dynamic analysis means the malicious behavior is monitored by tracing or inspecting the API calls from Windows and network connections by running the suspicious files in a safe environment. The extracted API function calls are used to detect malicious behavior through behavior or dynamic or run-time analysis method. The API calls from different categories such as process, registry, file, and network contain the function names, return values, and parameters of an executable. The dynamic analysis extracts distinct features to find the malicious software using the API sequence and frequency [5, 11]. The frequency on API might indicate how API calls play an important role for a malicious file, while API sequence shows the knowledge about how important consecutive behaviors of the malware are. Moreover, the researchers also utilized additional behaviors as features beside API calls which are dynamic link library (DLL), file opened/closed, and mutex that provide useful data about the suspicious files [31].

Most of the dynamic techniques focused on API calls [9–12] to represent malware behaviors. The authors used TEMU for dynamic analysis module which is based on QEMU. They collected the API calls and other essential information of running malware and then established the multilayer dependency chain [9].

The authors in [14] performed classification through run-time analysis using Cuckoo. The total number of samples, 42,068, was used for classification, 67% was used for training set, and 33% was used for the testing set. The authors extracted and used 151 API calls as main features; the first 200 API calls were used for sequence. They combined the features of 24 API FBs, modified sequence of first 40 different API, and 4 counters captured by modifying the Cuckoo Sandbox. In their approach, they employed a combination of features that achieved an average weighted AUC value of 0.98, TPR of 0.896, and FPR of 0.049 by applying RF classifier.

In [9], the authors proposed the variants of malware classification technique based on behavior profile. The authors used the TEMU to monitor the malware behaviors. They captured the API calls and other information and then established multilayer dependency chain by converting the function flow into multilayer behavior chain. To assess the validity and accuracy of the method, they downloaded 200 samples of 12 types from Anubis website. To identify the malware variants similarity, similarity comparison algorithm had been used in their work.

In [15], the authors experimented a 552 PE dataset with their corresponding API calls. These samples were executed in a Windows 7 virtual environment using Cuckoo Sandbox. Tf-idf (term frequency-inverse document frequency) had been used to extract relevant 4-gram API call features. The authors used four machine learning methods for training and testing the data. They got the accuracy between 92% and 96.4%. In [16], 2 malware datasets had been created such as 10 families and 10 different types. Then the authors extracted the features by using the memory access patterns recording technique from the sample. Then the authors performed n-grams size of 96. N-grams apply on the features of dynamic and static.

In [17], the authors extracted separately different features through run-time analysis, likewise API call, the usage of system library, and the operations. Four different classifiers and correlation-based feature selection method from WEKA tool had been applied in their work. Bigram API and API frequency approaches give the best performance by using the RF for four datasets. In [13], the authors conducted the detection and classification system using the calls of API sequences for four different families including normal group.

Masud et al. used information gain after the n-grams extracting to select the best 500 features. They experimented on two different datasets: first dataset contains 1435 executable files (597 cleanware and 838 malware), and the second dataset contains 2452 executables (1370 clean and 1082 malware). The information gain (IG) attribute selection method was used in [18–20] and their accuracies with 98%, 94.6%, and 97.7%, respectively. The accuracy of the hybrid model was 97.4 for both datasets $n = 6, 4$, respectively [21]. The chi-square feature selection was applied in [22, 23], and Tf-idf was applied in [24] to get the most relevant features.

The proposed system has also used the Chi2 χ^2 and PCA methods to support the high performance for classification by reducing the features size. The proposed approach provides the over 99% of accuracy on 300 features using χ^2 feature selection approach and 10 features with PCA approach on unigram. This section presents the existing works related to malware classification using machine learning algorithms and supports previous researches about feature extraction methods based

on dynamic malware analysis and classification. These research efforts use different malware modeling techniques using static or dynamic features obtained from malware samples.

3.3 Malicious Software Family Classification System

Malicious software classification is not a new topic but it is still needing attention and solution to be solved for cyber threats nowadays. Many researches have been carried out to analyze and classify the malicious files using the API function calls sequences to model malicious behavior. Thus, the malware behavioral patterns can be obtained by understanding the API Call Sequences (API-CS). Therefore, the proposed system also used the API-CS by proposing the API Feature Extraction Procedure and applying the n-gram method. Figure 3.2 shows the step-by-step process of malicious software analysis architecture for the proposed system.

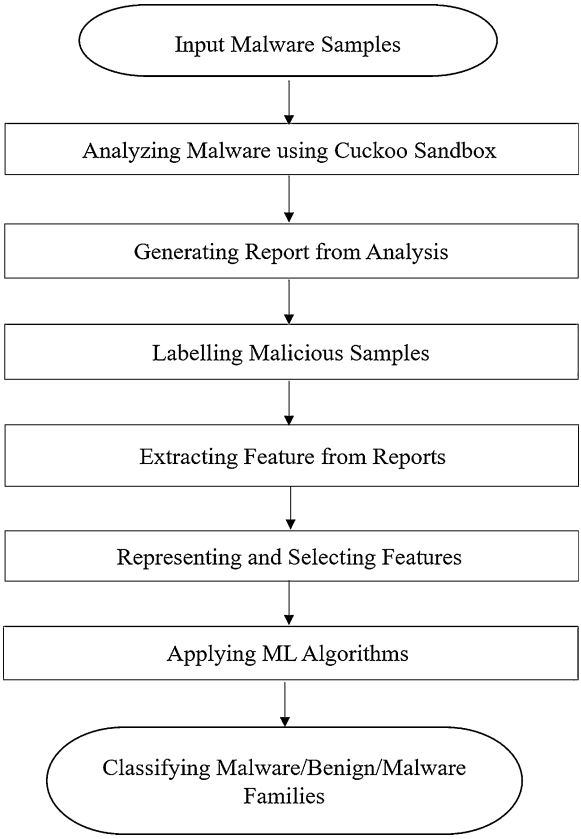

Fig. 3.2 Malicious software analysis and classification system

Malicious samples have been collected from virus share[1]; the proposed system experimented nearly 25,000 from 6 different families. However, the proposed system discards the samples based on the following conditions:

1. If the analysis report does not contain Virus Total (VT) label results
2. If the family does not have at least 1500 samples
3. If the extracted API features from report do not have at least 15 API features without duplicate ones

Therefore, the experiment provided a total of 20,809 samples from 6 malware families and cleanware in this research work. And Table 3.1 describes the number of samples for 14 different families and target label (Class) for each family. Table 3.2 describes the six different families for malware class.

3.3.1 Analyzing Malware Samples and Generating Reports

Cuckoo Sandbox [32] has been used to perform the dynamic analysis in this system. Windows 7 Operating System (OS) has been used for the virtual environment for analysis in Virtual Box and Ubuntu as host OS. It is widely used and open source for the researchers of academic and independent from a small to large business enterprises. It can analyze different types of malicious files, such as executables files, office document files, PDF files, emails, etc., and malicious websites. And it can also trace the API calls and the behavior of file, and dump and analyze network traffic, even encrypted with an SSL/TLS.

It generates the reports from analysis with multiple formats such as HTML, JSON, and PDF formats. But the proposed system used JSON format to extract the malicious behaviors. Figure 3.3 shows the lab setup environment of malware analysis. The lab setup environment has been described in the following figure. Ubuntu 18.4 (host) OS and Windows 7 (guest) have been used for analyzing the malicious samples on sandbox. The normal applications such as Office Documents, Adobe Reader, Browsers, etc. have been installed on virtual OS.

Table 3.1 Malware/benign dataset for experiment

Name	# of samples	Target label (Class)
Clean	5420	0
Malicious	15,389	1
Total number of samples	**20,809**	**2 (0/1)**

[1] http://tracker.virusshare.com:6969/

Table 3.2 Six different categories of malware

Categories	# of samples
Adware	2200
Backdoor	2076
Virus	1740
Trojan	5000
Worm	2397
Downloader	1976
Total	**15,389**

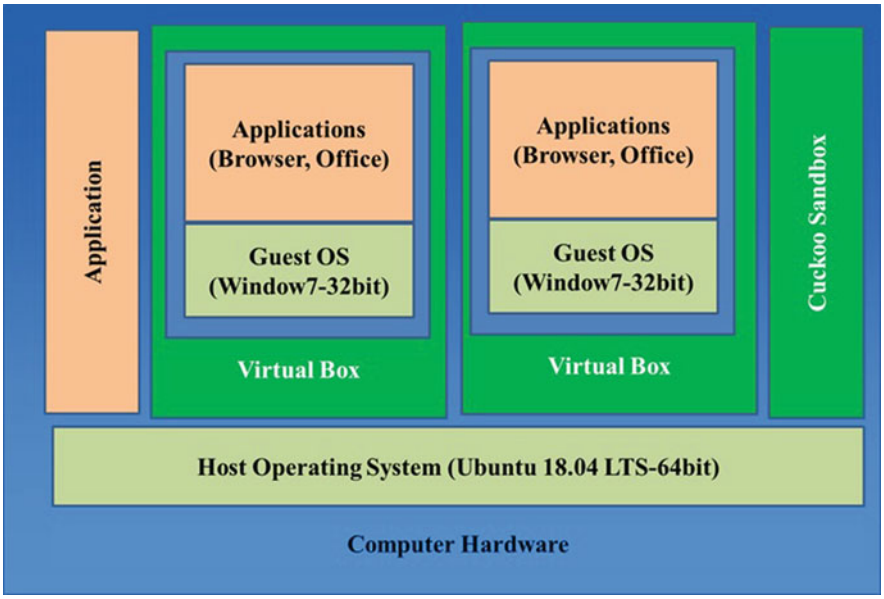

Fig. 3.3 Lab setup environment for malware analysis

3.3.2 Labeling Malicious Samples

The report of sandbox provides the VT label for each malicious sample if the analyzed sample exists on the VT database. Figure 3.4 describes the VT scans results with their anti-virus vendors, respectively.

The proposed system extracted these results using the regular expression (RE) theory. RE is a very powerful, useful, efficient, and flexible text processing language. And then count the occurrence of each results that extracted from RE, and choose the maximum value of word as label for each sample. Figure 3.5 shows the example of choosing the label for each malicious sample. These labels indicate the single malicious file. According to Fig. 3.5, this kind of malware sample will be labelled as Adware, and it is an Adware category or family.

```
"virustotal": {
    "scans": {
        "Bkav": {
            "detected": true,
            "version": "1.3.0.9899",
            "result": "W32.HfsAdware.88A5",
            "normalized": [
                "HfsAdware"
            ],
            "update": "20190130"
        },
        "MicroWorld-eScan": {
            "detected": true,
            "version": "14.0.297.0",
            "result": "Adware.Generic.1775594",
            "normalized": [],
            "update": "20190131"
        },
        "CMC": {
            "detected": false,
            "version": "1.1.0.977",
            "result": null,
            "normalized": [],
            "update": "20190131"
        },
        "CAT-QuickHeal": {
            "detected": true,
```


Fig. 3.4 VT label from analysis report

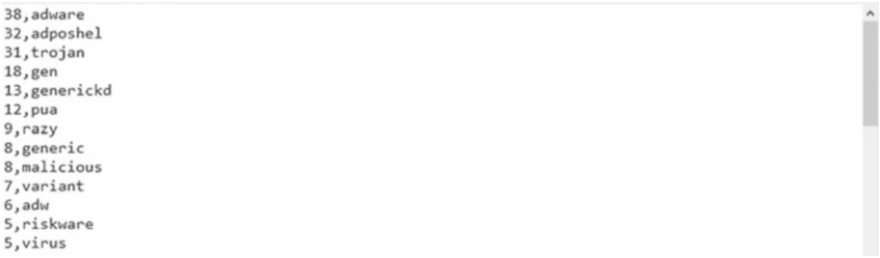

Fig. 3.5 Choosing label for each malicious sample

3.3.3 Extracting Malicious Features

After categorizing the malicious family, the process of extracting the malicious features has been performed in the proposed system. The APISTATS result has been extracted from the JSON for API features. And the procedure of extracting APISTATS process is described as followed:

```
APISTATS Feature Extraction Procedure
Input: JSON reports
Output: extracted API files fi
1: begin
2:          if (JSON ≤ JSONs)
3:           try
4:                  data = []
5:               data = json.load(JSON)
6:                 try
7:                  api = data['behavior'] ['apistats']
8:           print (api)
9:           except KeyError:
10:                 print ("APIStats KeyError")
11:         except ValueError:
12:              print ("JSONDecodeError")
13:         end if
14: end
```

The extracted raw APIs attributes are extremely large, and it is not able to handle the classification system. So, data cleaning processes have been provided in this phase of proposed system. The raw data cleansing processes are described as followed:

1. Remove empty line if the extracted API files have empty line
2. Remove noise data such as comma, colon, single code, double code, curly braces, and so on.
3. Remove duplicate API by keeping the order of API calls
4. Discard the extracted API files if the number of APIs does not have at least 15 API.

3.3.4 Applying N-gram

After processing the data cleansing steps that are described above, n-gram method has been applied to ensure the identifying of malicious files. It is a continuous sequence of *n*th items from a given sequence. It is very useful for characterizing the sequences in natural language processing and DNA sequencing areas [3]. It has been adopted to extract the sequence of features in malware classification for static and dynamic analysis. But the static is the one mostly used n-gram such as opcode n-gram, byte-code n-gram, and API call n-grams. The proposed system applies the n-gram technique, where $n = 1,2,3$, to identify the malicious families and benign. The total number of APIs after processing the data cleansing stage is 306 APIs. Therefore, the number of features for unigram (1G) is 306. Then, the number of features for bigrams (2G) is 10,796 g. And the total number of samples for classification is 20,809 instances. The explosive number of features will increase as long as the *n* number increases in dataset. Moreover, it could lead to an overfitting. So the proposed system used the unigram and bigrams for the classification. Both unigram and bigram provide a high accuracy to distinguish malware and benign.

3.3.5 Representing and Selecting Malicious Features

Attributes representation process has been conducted after applying n-grams on extracted APIs. The process of attributes representation has been performed based on the presence and absence of features in global feature database. The proposed system used the binary feature vector representation that is described in our previous research work [25] and described as follows:

$$\mathrm{API_i} = \begin{cases} 1, \text{ if API is in MBAPIDS File} \\ 0, \text{ otherwise} \end{cases}$$

The total global database MBAPIDB (Malware Benign API Database) contains all API features of malware and benign. If the extracted API contains in MBAPIDB, it is denoted as 1, and if not, it is denoted as 0. For example, the sample F1 is the single malware instance feature representation, and the last item 1 is the class label for malware family.

$$\mathrm{F1} = \{1, 1, 1, 0, 1, 0, 1, 0, 0, 1, \ldots, 1\}$$

The next step is the selection of feature for classification. The purposes of applying the selection approaches are to select the appropriate features to the target class and to minimize the processing time. Feature or attribute selection methods are used for reducing the size of a feature dataset. The key role of feature selection process is to improve the classification performance as well as improving the detection accuracy by choosing or transforming the feature set. Subsequently, the processing time for classification process can speed up and improve the evaluation results since the feature number is reduced.

Among the three FS methods such as filter, wrapper, and embedded methods, most researchers commonly used the filter-based approach in malicious classification and detection research areas. The filtering approach does not depend on any particular algorithm. It is very fast and computationally less expensive than the other two methods. It is easy to scale to very high-dimension datasets [30]. So, the proposed system applies the χ^2 method from filtering approach.

The proposed system used the two feature selection methods from sklearn. The efficiency of classification system can be improved by applying the attribute selection techniques such as chi-square (χ^2) and principal component analysis (PCA).

Chi-Square ($\boldsymbol{\chi^2}$) It is a statistic approach and very effective for feature selection process. The proposed system chooses $\boldsymbol{\chi^2}$ to select the feature because it can handle the multi-class data with an excellent performance. The proposed system used the implementation of $\boldsymbol{\chi^2}$ from sklearn ML library with python.

Principal Component Analysis (PCA) It is used to visualize and explore high-dimensional datasets. It reduces a set of possibly -correlated, high-dimensional variables to a lower-dimensional set of linearly uncorrelated synthetic variables called principal components. PCA reduces the dimensions of a dataset by projecting the data onto a lower-dimensional subspace [27].

3.3.6 Classifying Malware vs Benign Using Machine Learning

ML has powerful ability and capability to do many things for cybersecurity. It can be used to identify the advanced persistent threats (APTs) and zero-day attacks which are more complex than the normal malware or threats. And it can be used in many intrusion detection systems (IDS) because it can detect new and unknown attacks. It can be applied in many areas of information security such as spam and phishing email detection, phishing website detection, and virus detection. To classify the malicious and benign software, the proposed system used the two ML methods, random forest (RF) and K-nearest neighbor (KNN) from sklearn ML library.

Random Forest It combined the multiple decision trees, so it became an ensemble. It can handle the binary, categorical, continuous, and missing values, so it is suitable for high- dimensional data modeling. It can overcome the overfitting problems due to the nature of bootstrapping and ensemble scheme. Thus, it does no need to prune the trees. Besides high prediction accuracy, it is efficient, interpretable, and non-parametric for various types of datasets [28].

K-Nearest Neighbors (NN) It is an instance-based learning and also known as lazy learner. The lazy is called not because of its apparent simplicity, but because it doesn't learn a discriminative function from the training data but memorizes the training dataset instead. It is a sub-category of non-parametric approach [29].

The performance of ML classifiers has been evaluated using confusion matrix (CM), accuracy (ACC), precision recall (PR), and receiver operator characteristics area (ROC).

Confusion Matrix (CM) It is a popular way to describe a classification model. CM can be formed for binary and multi-class classification models. It has been created by comparing the predicted class label of a data point with its actual class label. After comparing the whole dataset repeatedly, the comparison results are formatted in a matrix form. This resultant matrix is the confusion matrix [26]. And Fig. 3.6 describes the typical structure of a CM.

Accuracy (ACC) It is a common evaluation method of a classifier performance. It is used to define as the percentage of overall accuracy of correct predictions. It can be calculated from the formula [26]:

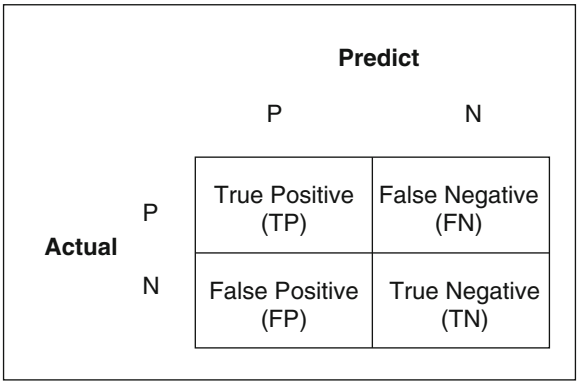

Fig. 3.6 Confusion matrix (CM) structure

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{3.1}$$

Precision (P) It is a positive predictive value that can be achieved from CM. It is defined as the number of predictions made that is actually correct or relevant out of all the predictions based on the positive class [26]. It can be calculated from the following formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.2}$$

Recall (R) It is also known as sensitivity, and it is used to identify the relevant data points with percentage. It is defined as the number of instances of the positive class that were correctly predicted [26]. It is also called as hit rate, coverage, or sensitivity. The value of recall can be computed as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.3}$$

Receiver Operating Characteristic (ROC) It can be used for both binary and multiclass classifiers. TP rate and the FP rate of a classifier are used to plot the ROC curve. TPR is also called recall or sensitivity, and it is the total number of correct positive results, predicted among all the positive samples in dataset. FPR is also defined as 1- specificity or false alarms, determining the total number of incorrect positive predictions among all negative samples in the dataset [26].

3.4 Results and Discussion

The total 25,000 malicious samples have been analyzed in this research work, but 20,809 benign samples have been used for family classification. The proposed system contributes the API feature extraction for malicious family classification.

After processing the raw data cleansing on extracted APIs, the remaining API features are 306 APIs. As the malicious behavioral patterns sometimes depend on the sequence of function calls and system calls, the proposed system applies the n-gram technique to ensure the right family classification. Therefore, the proposed system noted these APIs as unigram (1G). The total number of bigram (2G) API features is 10,796 g, and trigram (3G) features are 37,919 g. In this case, the trigram features are quite large, so the proposed system only considers to perform the unigram and bigram on our dataset since we concern the processing/run time. Table 3.3 shows the number of API grams for classification. After utilizing the n-grams on extracted APIs, the proposed system performs the attribute selection process before classifying families. The proposed system uses the two attribute selection methods, chi-square with SelectKBest and PCA, from sklearn. RF and kNN have been used to classify the malicious families and benign. The proposed system uses the 25% (5203 executables) for testing and the rest 75% (15,606 executables) for training. The proposed system uses accuracy, precision, recall, and ROC scores to assess the efficiency and effectiveness of extracted prominent APIs.

Table 3.4 describes the comparison of accuracy on unigram API (306 g) dataset using two FS and ML techniques. The proposed system compares the accuracies by selecting the five different numbers of features from unigram such as 10, 50, 100, 200, and 300 APIs. The RF classifier provides better accuracy 99% on selected 10 API using PCA and 300 API using χ^2 (Chi2). The kNN classifier provides 97% on χ^2 with 300 API and PCA with 10 API. The finding from the experiment is that the accuracy is increased when PCA chooses the small number of API. It is inversely proportional to the Chi2 approach. In χ^2, the accuracy has been increased as long as the selected API number is increased. RF classifier produces better accuracy on PCA with 10 features and χ^2 with 300 features than the kNN.

Figure 3.7 provides the confusion matrix results for RF on Chi2 χ^2 (300 API) and PCA (10 API). The correctly classified instances number of PCA on malware is slightly better than the Chi2's result.

Figures 3.8 and 3.9 describe the precision-recall (PR) curves and ROC curves of Chi2 for 300 API on RF and kNN classifiers.

Table 3.5 shows the comparison tables of accuracy on bigram API features. For bigram API selection, the proposed system used the different number of features

Table 3.3 Total number of grams after applying n-gram on extracted API dataset

N-grams	# of grams or APIs
Unigram	306
Bigrams	10,796

Table 3.4 Accuracy (%) comparison on selected unigram API

	RF					kNN				
	10	50	100	200	300	10	50	100	200	300
Chi2	86.2	97.2	97.9	98.5	**98.7**	82.9	95.9	96.4	96.4	**97**
PCA	**99**	98.7	98.7	98.7	98.7	**97.3**	97.2	97.2	97.2	97.1

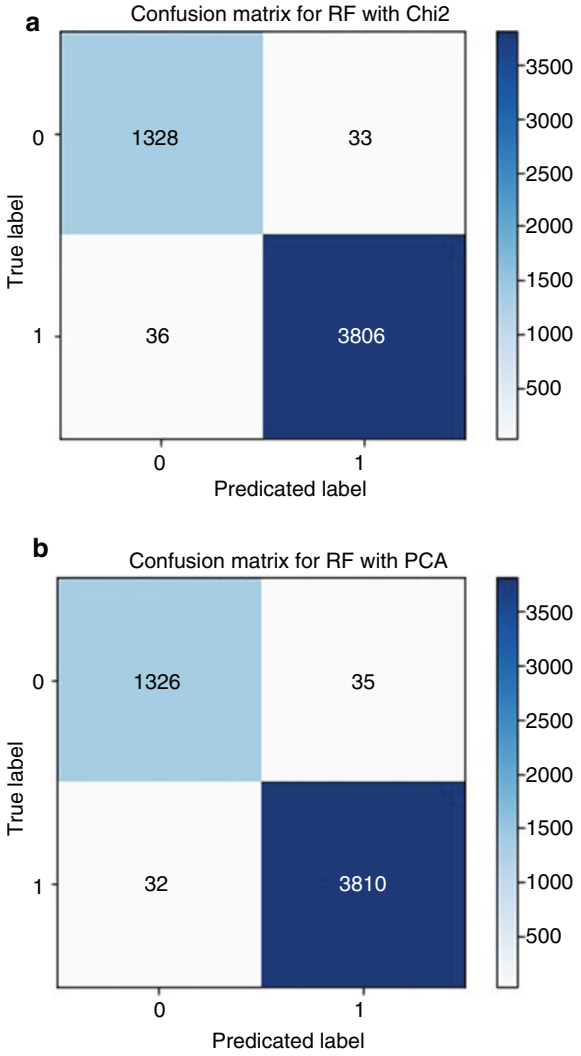

Fig. 3.7 Confusion matrix of RF classifier on unigram dataset. (**a**) CM for RF on selected 300 API using Chi2. (**b**) CM for RF on selected 10 API using PCA

such as 100, 200, 300, 400, and 500, unlike unigram. Unigram has been selected according to 10, 50, 100, 200, and 300.

The total number of bigram API is 10,796, and testing dataset is 5203 from 20,809 instances or samples. The training and testing dataset are split 75% and 25% of the dataset. The experiment shows that PCA increases the accuracy slightly better than the Chi2 on both classifiers. Figure 3.10 and 3.11 depict the ROC and PR curves for RF classifier using Chi2 and PCA for 500 g API.

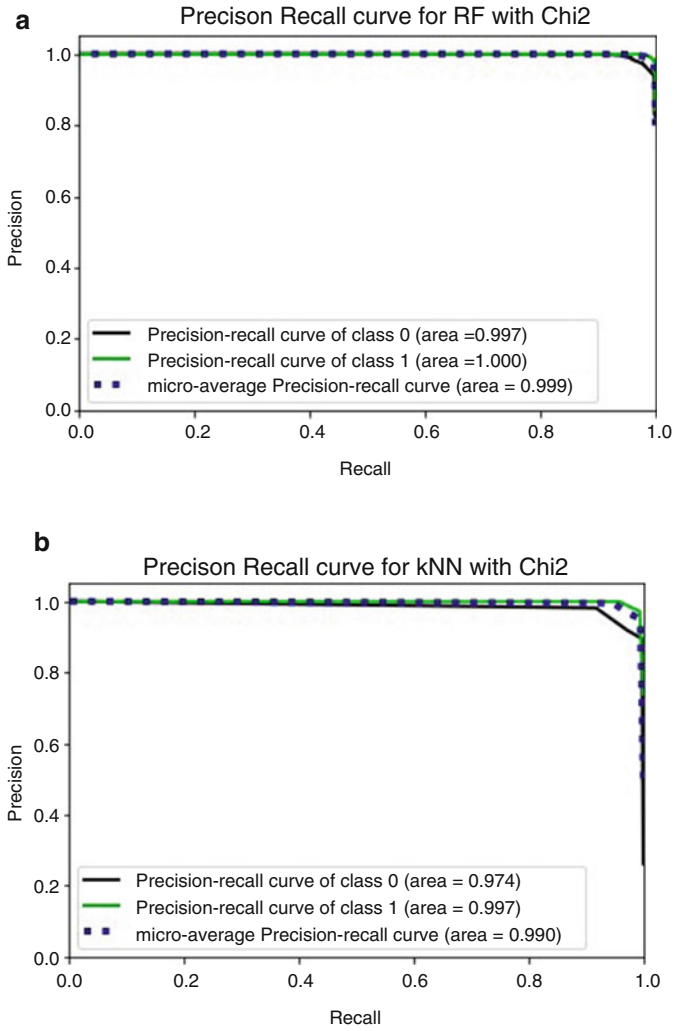

Fig. 3.8 PR curves for RF and kNN using Chi2 (χ^2). (**a**) PR curve of RF classifier on selected 300 APIs. (**b**) PR curve of kNN classifier on selected 300 APIs

Figure 3.12 shows the confusion matrix of RF classifier on selected 500 API bigram dataset using Chi2 (χ^2) and PCA. The confusion matrix results from PCA provide better than the Chi2 (χ^2) method, and the incorrectly classified instances are smaller than the Chi2 (χ^2).

Table 3.6 provides the accuracy comparison between our approach and other related works. Although the related work [22] is slightly better than our approach,

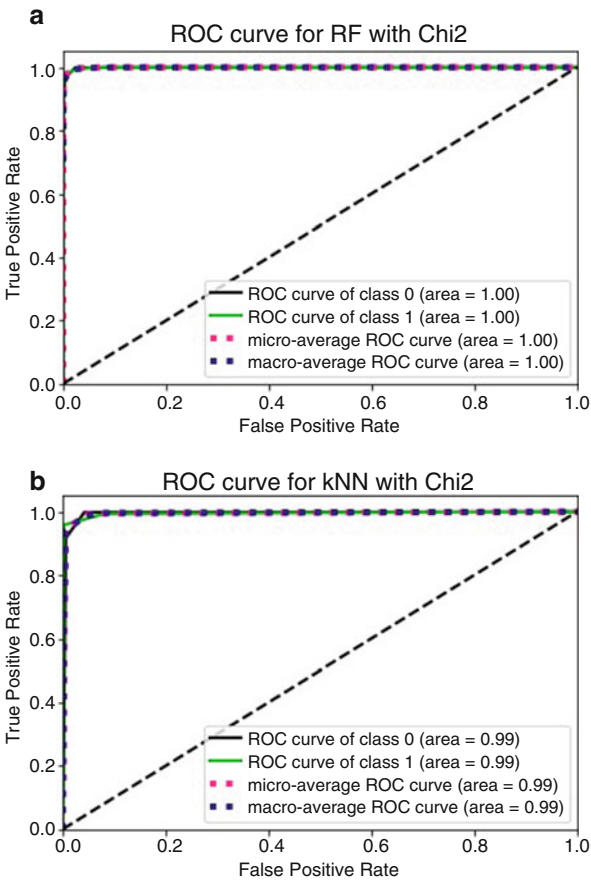

Fig. 3.9 ROC curves for RF and kNN using Chi2 (χ^2). (**a**) ROC curve of RF classifier on selected 300 APIs. (**b**) ROC curve of kNN classifier on selected 300 APIs

Table 3.5 Accuracy (%) comparison on selected bigram API

	RF					kNN				
	100	200	300	400	500	100	200	300	400	500
Chi2	95.5	97.2	97.9	98	**98.4**	94.5	95.8	96.1	96.7	**97.1**
PCA	98.8	99	99	99	**99**	97.9	98	98.1	98.1	**98.1**

the number of tested samples is quite small on both clean and malware. The original extracted API features provide the best accuracy of 99% on malware vs benign classification system. However, the proposed system applies the n-gram technique on extracted dataset since the proposed system concerns the malware that used the garbage code inserting techniques.

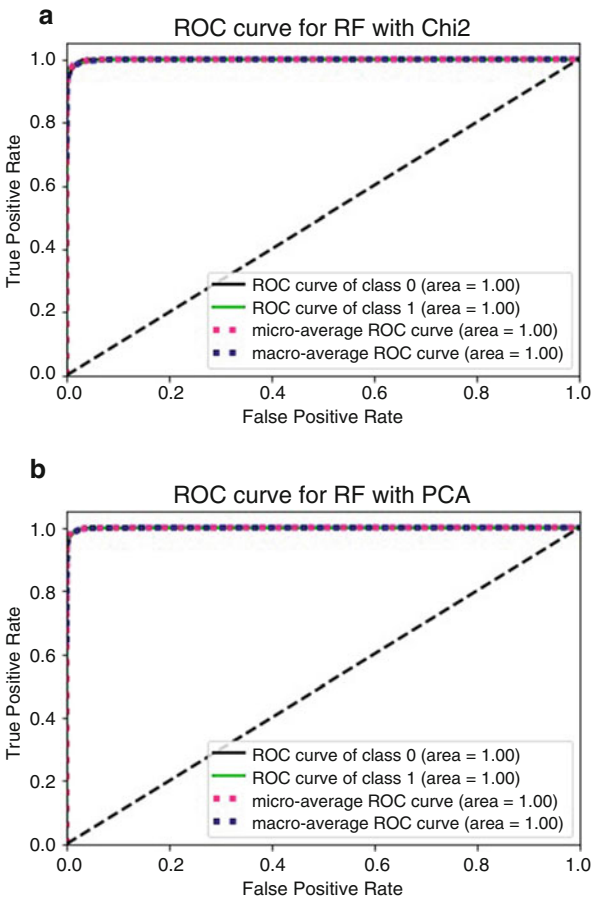

Fig. 3.10 ROC curves for RF classifier using Chi2 (χ^2) and PCA. (**a**) ROC curve on selected 500 g API using χ^2. (**b**) ROC curve on selected 500 g API using PCA

3.5 Conclusion

The usage of ML techniques in cyber security is becoming increasingly than ever before. The proposed system used the two ML methods to classify the malware vs benign for classification system. The proposed system contributes the malicious feature extraction for API features with n-gram and classification through dynamic analysis. The system extracts the API by using the APISTATS keyword from JSON report format. The proposed system has performed the raw data cleansing process after extracting the API from JOSN. Malicious JSON reports contain six different types of malware categories like Adware, Downloader, Trojan, Backdoor, Worm, and Virus. Two feature selection approaches, Chi2 χ^2 and PCA, have been conducted to reduce the size of features especially for bigram APIs as the size of n-gram feature is large to handle the classification. The results from the experiment

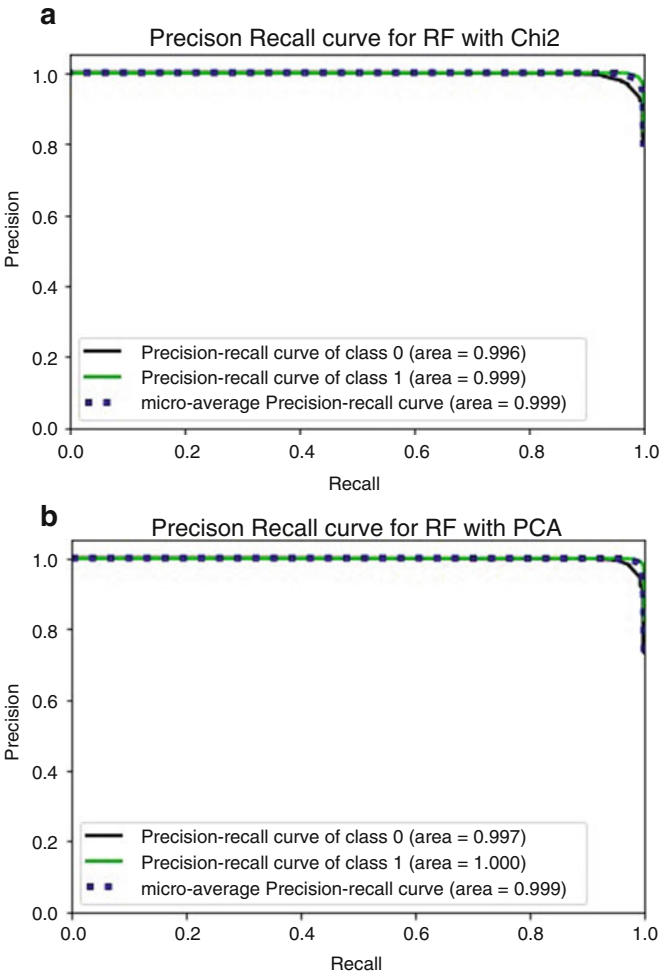

Fig. 3.11 PR curves for RF classifier on selected APIs using Chi2 (χ^2) and PCA. (**a**) PR curve of RF classifier on selected 500 APIs using χ^2. (**b**) PR curve of RF classifier on selected 500 APIs using PCA

can be noted that PCA provides better accuracy than the Chi2 χ^2 on unigram and bigram dataset.

In unigram, the accuracies of PCA are remained stable on different number of selected features. However, it is inversely proportional to the Chi2 approach. In Chi2 χ^2, the accuracy has been increased as long as the selected API number is increased. In bigram dataset, the accuracies of PCA also remain stable on both RF and kNN classifiers, while the accuracies of χ^2 vary on both RF and kNN classifiers. The proposed prominent feature extraction procedure provides a high accuracy with 99% and low FP and FN rates. The proposed system evaluates the performance of

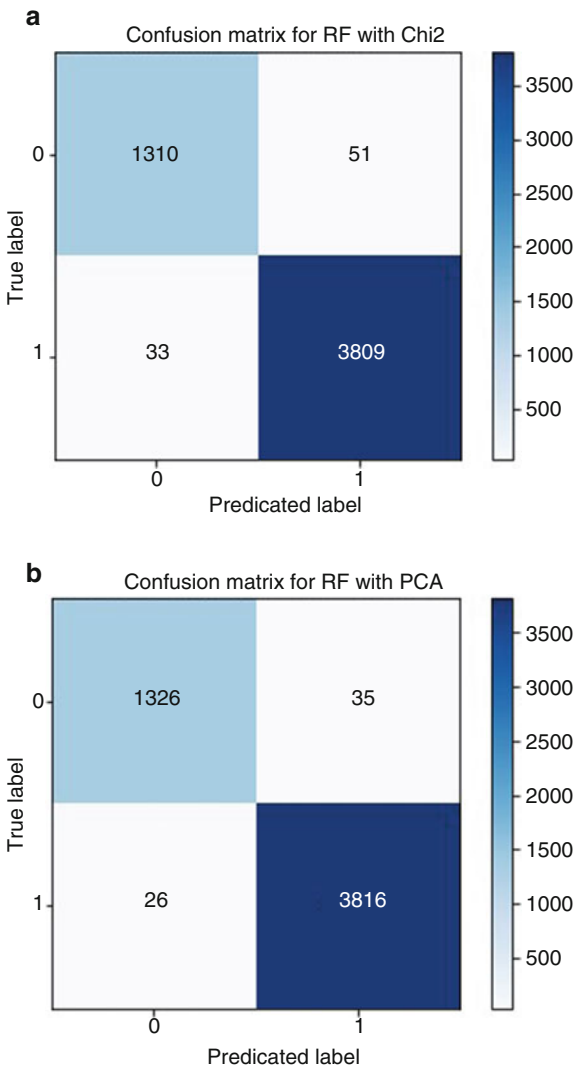

Fig. 3.12 CM for RF classifier on 500 API. (**a**) CM for RF classifier on selected 500 API using Chi2 (χ^2). (**b**) CM for RF classifier on selected 500 API using PCA

the classifier by using Accuracy, Precision-recall, and ROC scores. Moreover, the system also provides the low FP and FN rates on malware and benign classification.

The system will extend by adding the other malicious behavior features such as system library, process, and file opened/closed besides API in the future work. Moreover, the malicious samples from different families such as Zbot, Swizzor, Startpage, etc. will also be used to classify their families.

Table 3.6 Accuracy comparison for different FS approaches

Study	Features	FS	Dataset	Accuracy
[19] 2011	API string	IG	1368 malware 456 benign	97.4% for MFC 94.6% for M vs C
[31] 2012	API from 6 DLLs, input arguments	ReliefF	826 malicious 385 benign	98.4%
[24] 2015	n-gram (Static + Dynamic)	Tf-Idf	4288 samples from 9 families	~96%
[23] 2016	API, TPF	χ^2	338 malwares 214 benign 12 categories	98%
[22] 2016	Static + Dynamic	χ^2	7630 malware 1818 benign	99.60%
Our approach	API	χ^2 (unigram) PCA (bigram)	15,389 malicious 5420 benign 6 categories	99%

References

1. G. Tahan, L. Rokach, Y. Shahar, Automatic malware detection using common segment analysis and meta-features. J. Mach. Learn. Res. **13**, 949–979 (2012)
2. R. Lo, K. Levitt, R. Olsson, Mcf: A malicious code filter. Comput. Secur. **14**, 541–566 (1995)
3. M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto, Novel feature extraction, selection and fusion for effective malware family classification. In Proceedings of the sixth ACM conference on data and application security and privacy, ACM, March, 2016, pp. 183–194
4. A. Kumar, A framework for malware detection with static features using machine learning algorithms, Doctoral dissertation, Department of Computer Science, Pondicherry University, 2017
5. A. Pektaş, Behavior based malware classification using online machine learning, Doctoral dissertation, Grenoble Alpes, 2015
6. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, J. Vanderplas, Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
7. J.Y.C. Cheng, T.S. Tsai, C.S. Yang, An information retrieval approach for malware classification based on windows API calls. 2013 International Conference on Machine Learning and Cybernetics, vol. 4, IEEE, 2013, July, pp. 1678–1683
8. M. Souppaya, K. Scarfone, *Guide to Malware Incident Prevention and Handling for Desktops and Laptops* (National Institute of Standards and Technology, Gaithersburg, 2013)
9. G. Liang, J. Pang, C. Dai, A behavior-based malware variant classification technique. Int. J. Inf. Educ. Technol. **6**, 291–295 (2016)
10. H.S. Galal, Y.B. Mahdy, M.A. Atiea, Behavior-based features model for malware detection. J. Comput. Virol. Hacking Tech. **12**(2), 59–67 (2016)
11. Y. Ki, E. Kim, H.K. Kim, A novel approach to detect malware based on API call sequence analysis. Int. J. Distrib. Sens. Netw. **2015**(6: 659101), 1–9 (2015)
12. C.-I. Fan, H.-W. Hsiao, C.-H. Chou, Y.-F. Tseng, Malware detection systems based on API log data mining. 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015, pp. 255–260

13. M.A. Jerlin, K. Marimuthu, A new malware detection system using machine learning techniques for API call sequences. J. Appl. Secur. Res. **13**(1), 45–62 (2018)
14. R.S. Pirscoveanu, S.S. Hansen, T.M. Larsen, M. Stevanovic, J.M. Pedersen, A. Czech, Analysis of malware behavior: Type classification using machine learning. 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), IEEE, 2015, June, pp. 1–7
15. A. Ninyesiga, J. Ngubiri, Malware classification using API system calls. Int. J. Technol. Manag. **3**(2), 9–9 (2018)
16. S. Banin, G.O. Dyrkolbotn, Multinomial malware classification via low-level features. Digit. Investig. **26**, S107–S117 (2018)
17. A.G. Kakisim, M. Nar, N. Carkaci, I. Sogukpinar, Analysis and evaluation of dynamic feature-based malware detection methods, in *International Conference on Security for Information Technology and Communications*, (Springer, Cham, 2018), pp. 247–258
18. D. Komashinskiy, I. Kotenko, Malware detection by data mining techniques based on positionally dependent features. 2010 18th Euromicro conference on parallel, distributed and network-based processing, IEEE, 2010, pp. 617–623
19. V. Moonsamy, R. Tian, L. Batten, Feature reduction to speed up malware classification, in *Nordic Conference on Secure IT Systems*, (Springer, Berlin, Heidelberg, 2011), pp. 176–188
20. M. Mays, N. Drabinsky, S. Brandle, Feature selection for malware classification, In MAICS, 2017, pp. 165–170
21. M.M. Masud, L. Khan, B. Thuraisingham, A hybrid model to detect malicious executables. 2007 IEEE International Conference on Communications, IEEE, 2007, pp. 1443–1448
22. C. Cepeda, D.L.C. Tien, P. Ordóñez, Feature selection and improving classification performance for malware detection. 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), IEEE, 2016, pp. 560–566
23. M. Belaoued, S. Mazouzi, A chi-square-based decision for real-time malware detection using PE-file features. J. Inf. Process. Syst. **12**(4) (2016)
24. C.T. Lin, N.J. Wang, H. Xiao, C. Eckert, Feature selection and extraction for malware classification. J. Inf. Sci. Eng. **31**(3), 965–992 (2015)
25. C.C. San, M.M.S. Thwin, N.L. Htun, Malicious software family classification using machine learning multi-class classifiers, in *Computational Science and Technology: 5th ICCST 2018, Lecture Notes in Electrical Engineering*, vol. 481, (Springer, Singapore, 2018), pp. 423–433
26. D. Sarkar, R. Bali, T. Sharma, *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*, 1st edn. (Apress, Berkely, 2017)
27. G. Hackeling, *Mastering Machine Learning with Scikit-Learn* (Packt Publishing Ltd, Birmingham B3 2PB, UK, 2017)
28. Y. Qi, Random forest for bioinformatics, http://www.cs.cmu.edu/
29. S. Raschka, *Python Machine Learning* (Packt Publishing Ltd, Birmingham B3 2PB, UK, 2015)
30. M. Khan, S.M.K. Quadri, Effects of using filter based feature selection on the performance of machine learners using different datasets. Bharati vidyapeeth's institute of computer applications and management's International Journal of Information Technology **5** (2013)
31. Z. Salehi, M. Ghiasi, A. Sami, A miner for malware detection based on API function calls and their arguments. The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), IEEE, 2012, pp. 563–568
32. C. Guarnieri, A. Tanasi, J. Bremer, M. Schloesser, The Cuckoo Sandbox, 2012