Multimedia Tools and Applications
https://doi.org/10.1007/s11042-021-10647-z

A new machine learning-based method for android malware detection on imbalanced dataset

Diyana Tehrany Dehkordy[1] · Abbas Rasoolzadegan[1]

Received: 31 December 2019 / Revised: 27 October 2020 / Accepted: 4 February 2021 /
Published online: 07 April 2021

Abstract
Nowadays, malware applications are dangerous threats to Android devices, users, developers, and application stores. Researchers are trying to discover new methods for malware detection because the complexity of malwares, their continuous changes, and damages caused by their attacks have increased. One of the most important challenges in detecting malware is to have a balanced dataset. In this paper, a detection method is proposed to identify malware to improve accuracy and reduce error rates by preprocessing the used dataset and balancing it. To attain these purposes, the static analysis is used to extract features of the applications. The ranking methods of features are used to preprocess the feature set and the low-effective features are removed. The proposed method also balances the dataset by using the techniques of undersampling, the Synthetic Minority Oversampling Technique (SMOTE), and a combination of both methods, which have not yet been studied among detection methods. Then, the classifiers of K-Nearest Neighbor (KNN), Support Vector Machine, and Iterative Dichotomiser 3 are used to create the detection model. The performance of KNN with SMOTE is better than the performance of the other classifiers. The obtained results indicate that the criteria of precision, recall, accuracy, F-measure, and Matthews Correlation Coefficient are over 97%. The proposed method is effective in detecting 99.49% of the malware's existing in the used dataset and new malware.

Keywords Malware detection · Android applications classification · Dataset balancing · SMOTE balancing

1 Introduction

Android has gained significant popularity among mobile operating systems (OS) since 2012 [34]. This popularity has led to an increase in the number of users of this OS. Android

✉ Abbas Rasoolzadegan
rasoolzadegan@um.ac.ir

Diyana Tehrany Dehkordy
d.tehrany@mail.um.ac.ir

[1] Computer Engineering Department, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

is an open-source OS and there are a lot of hidden malicious applications among benign ones published in application stores [6]. Thereby, the alarmingly increasing rate of malicious applications and their complexity has become a serious challenge [10]. Recent reports indicate that every ten seconds, one malware is released in app stores [26]. This has led to escalating attacks of malware applications among Android smartphones [5]. One of the main challenges in the process of malware detection is balancing the used dataset.

In recent years, the problem of data balancing has gained much significance for researchers [14, 23, 28, 29]. An imbalanced dataset is a dataset in which there is a large difference in the number of samples of its classes. Traditional classifiers like Naive Bayes, Logistic Regression, Support Vector Machine (SVM), and Random Forest are not flexible against imbalanced datasets and these are not designed to overcome imbalanced data problems. These classifiers are unable to solve the challenges of imbalanced datasets [44, 46, 49]. The performance of the traditional classifiers is affected by the imbalanced data. Thereby, classification is performed based on the majority class in imbalanced datasets [14, 46]. It still seems to be a challenge to propose an appropriate detection method that can accurately identify malware.

The most important phase of creating a malware detection method is analyzing applications and extracting features. These features are extracted using static analysis, dynamic analysis, or both [2, 34, 39, 45, 48]. The static analysis is a fast and lightweight method to extract features [3]. The advantages of static analysis are lightweight computation, low complexity, and low False Positive Rate (FPR) [3, 39]. In dynamic analysis, applications are run in a simulated or a controlled environment and features are extracted based on reports of analysis [44, 48]. The dynamic analysis requires interaction with applications because some of the destructive patterns expose themselves after doing some actions in a certain period of time [48]. Among the dynamic features, we can mention system call sequences, function calls, network traffic, API calls, and user interactions [2, 34, 48]. The advantage of this analysis is high accuracy and its disadvantages are high complexity, high computation load, high FPR, and time-consuming analysis [34, 48].

The purpose of the proposed method is to improve accuracy and reduce the error rate. In the proposed method, an attempt has been made to improve the process of detecting malware using methods of ranking features, data balancing techniques, and machine learning techniques. Initially, we try to identify effective features by ranking the extracted features. Removing the low-effective features reduces training time and complexity of the detection model, and improves accuracy. In most of the detection approaches, the extracted features are used without preprocessing on them [34, 39, 48]. After preprocessing the used dataset, we use three techniques of the Synthetic Minority Oversampling Technique (SMOTE), undersampling, and combination technique to balance the used dataset. So far, the problem of balancing datasets has not been considered in related studies [5, 10, 26, 29, 39, 46, 48]. It should be noted that the undersampling technique has also been improved as described in Section 3. Thereby, in order to build the malware detection model, we use the balanced dataset and the effective features and the classifiers of K-Nearest Neighbor (KNN), Iterative Dichotomiser 3 (ID3), and SVM.

The rest of the paper is organized as follows: Section 2 describes the related studies of malware detection using machine learning techniques. The data balancing techniques are then described. Section 3 describes the proposed method in four phases, namely collecting applications (phase I), analyzing applications and balancing the used dataset (phase II), creating and evaluating the detection model (phase III), and finally, detecting malware and classifying applications (phase IV). Section 4 presents the results of the evaluation of the proposed method and compares the obtained results with other related studies. Finally, Section 5 elaborates the conclusion and future works.

2 Literature review

Signature-based approaches were introduced in the mid-90s, which are one of the oldest detection approaches [3, 34]. These approaches use static analysis to detect malware [10]. The advantages of these approaches are high speed and high accuracy in detecting known malware apps. The disadvantages of this approach are failure against malware code changes, inability to detect new and unknown malware, unable to detect obfuscated malwares, time-consuming, failure against update the signatures databases, and expensive the process of signature extraction [3, 34, 39]. The rapid malware advancement overtook signature-based detection approaches and these methods were no longer able to identify new and unknown malware [2, 34, 39]. Therefore, the weak performance of signature-based approaches has motivated researchers to use learning-based methods to detect malware and overcome the disadvantages of signature-based approaches [42].

The main purpose of related studies is to detect malware and classify applications. The purpose of malware detection is to reduce malicious activities or terminate them, reduce damages of the devices, increase the security of user information, developers, and app stores [2, 34, 39, 42, 45, 48]. Therefore, the problem of malware detection is particularly important. So far, in most learning-based malware detection approaches, the number of benign applications is to be much larger than the number of malware ones [15, 25, 30, 33, 35, 41, 51, 54]. Therefore, it is necessary to make a balance between malware and benign applications. The proposed method uses machine learning-based techniques to detect malware and classify applications. Hence, in the following section, the machine learning-based studies are described. Then, the methods of balancing datasets are discussed.

2.1 Machine learning-based studies for malware detection

Malware detection approaches follow a hierarchy that includes applications analysis to extract features, feature selection, and malware detection [34, 42]. Li et al. proposed a malware detection approach called SIGPID [26]. They used static analysis to extract permissions features. In this approach, the significant permissions are analyzed and are identified to prevent the rise of malware. In SIGPID, instead of analyzing all permissions, preprocessing is performed on the set of extracted features, where three levels of pruning are done to identify significant permissions. Finally, applications are classified using SVM. Daniel et al. proposed the DREBIN approach [5]. Static analysis is used to extract features. The DREBIN method extracts features from the manifest file and analyzes codes to reduce the computational burden. In this approach, 545,000 features are used without preprocessing them. Also, SVM is used to detect malware and classify applications.

Yerima and Sezer proposed the DriodFusion approach [51]. Static analysis is used to extract features. DriodFusion uses weighting methods and ranking techniques to preprocess features. DriodFusion uses the ensemble learning method, which is a combination of the classifiers of J48, REPTree, Voted Perceptron, and Random Tree. Pektaş and Acarman proposed an approach in which features are extracted from applications using static analysis and flow graphs [35]. In this approach, deep learning methods are used alongside CNN and LSTM algorithms to detect malware. Garcia et al. proposed RevealDriod, which is capable of detecting malware apps and their families [15]. RevealDriod uses the features of API calls and function calls.

Saracino et al. presented a malware detection system called MADAM [40]. MADAM extracts the required features from levels of kernel, application, user, and package in order to detect malicious behaviors. This system uses static and dynamic analysis to extract features. Also, KNN is used to detect malware and classify applications. This system is able to detect malware with an accuracy of 96%. Aafer et al. proposed the DroidAPIMiner method to detect malware [1]. DroidAPIMiner uses the features of API Calls and permissions. Also, DroidAPIMiner uses the classifiers of KNN, SVM, ID3, and C4.5 to build the detection model. Finally, KNN has the best performance among these classifiers and is able to detect malware with an accuracy of 99% and FPR of 2.2%. Suarez-Tangil et al. proposed DriodSeive, which extracts features using static analysis [43]. This approach is fast and resilient against obfuscation techniques. Though DriodSeive is strong and has a low cost, it depends on features such as API calls, structures of code, permissions, and components set. In DriodSeive, features are given scores, and then, features are selected that have the highest score. DriodSeive uses the extra tree to detect malware and classify applications.

Similar to the related studies, static analysis is used to extract features in many detection approaches. Also, various learning techniques are used to detect malware and classify applications [25, 30, 33, 35, 41, 54]. In the following section, Table 1 compares the related studies based on the number of applications used. Due to a large number of related studies, some of them were reviewed as examples.

Table 1 Comparison of the number of applications used in the related studies

References	Year	Number of Applications		
		Malware	Benign	Total Number
[51]	2019	28870	34393	63263
[35]	2019	24650	25000	49650
[54]	2019	379	920	1299
[41]	2019	2973	5680	8653
[25]	2019	28848	14956	43804
[30]	2019	3427	5161	8588
[33]	2019	19273	58884	78157
[26]	2018	5494	305432	310926
[15]	2018	–	–	57882
[31]	2018	17400	107078	124478
[12]	2018	–	–	114000
[8]	2017	2794	0	2794
[43]	2017	17401	107078	124479
[4]	2017	1758	3723	5481
[52]	2016	1760	20000	21760
[19]	2016	–	–	3232
[50]	2015	202	633	835
[37]	2014	1400	4800	6200
[5]	2014	5560	123453	129013
[16]	2012	–	–	118000

The results of Table 1 indicate that in most of the detection approaches, there is no balance between applications used (except equal cases) [4, 5, 8, 25, 26, 30, 31, 33, 35, 37, 41, 43, 50–52, 54]. The number of benign applications is high in app stores and access to them is easy. Also, it is hard to access malware. Therefore, in most of the detection approaches, the number of benign applications is more than the number of malware applications. There should be a balance between malware and benign applications in order to avoid biased results. Balancing the dataset does not necessarily mean equalization. The dataset must be balanced according to the real-world data. Also, in some of the studies, the number of applications is not specified by type and some of them use only one type of application (benign or malware) [12, 15, 16, 19].

In order to create an appropriate detection method, it is necessary to use benign applications alongside malware ones, so that we can distinguish between their behaviors and improve the accuracy of the detection method. Thereby, malicious behaviors can be identified with higher accuracy and classification is performed more precisely. Investigation of machine learning techniques has indicated that many researchers use SVM to classify applications because SVM has a better performance than the other techniques over an imbalanced dataset [5, 8, 15, 26, 43, 51]. Also, in almost all the studies, accuracy is considered as the main criterion for evaluation, and other criteria are less considered [5, 15, 26, 30, 31, 35, 37, 50]. Accuracy is not reliable in imbalanced datasets. For example, if a dataset contains ten samples of the minority class and 90 ones of the majority class, the classifier labels all test data as major data that also has the best accuracy. But, this classifier is unable to detect the minority class correctly and the accuracy of the minority class is zero [14, 23, 29]. Therefore, the obtained results are not reliable because the imbalanced datasets lead to biased results. According to the obtained results, we try to balance the used dataset and improve the accuracy of the malware detection method. In the following section, we will review the methods of balancing the dataset.

2.2 Dataset balancing methods

In many real-world problems, datasets are mostly imbalanced, which makes classification processing more difficult and has a great impact on classifiers performance. Some of these problems are fault detection, anomaly detection, text classification, medical uses, fraud detection, intrusion detection, etc [14, 23, 24, 28, 29, 38, 49]. Considering the widespread usage of imbalanced datasets, it is necessary to manage the challenge of balancing datasets. Traditional classifiers assume the used dataset to be balanced. Therefore, the performance of classifiers is affected by majority classes when the dataset is not balanced. These classifiers have a low prediction about minority classes. Thereby, classification accuracy is not trustworthy [14, 44, 46].

If traditional classifiers are used to solve problems that have imbalanced datasets, these classifiers will face difficulties such as extraction of the biased model, overlapping, incorrect classification of minority classes, ignoring the data of the minority class, and identifying them as noise data [14, 24, 38, 46]. Therefore, using an imbalanced dataset will reduce the classification accuracy of the minority classes and reduce the detection model efficiency [24, 38]. Thus, it should be noted that the data of minority classes such as rare diseases and malware detection is very important [14, 46]. A review of the related studies has shown that the detection accuracy is high having the imbalanced dataset. But, the obtained accuracy belongs to the majority classes [5, 15, 26, 30, 31, 35, 37, 50]. So far, many methods are proposed to balance datasets and solve the classification of the imbalanced data which have promising results. These methods are divided into four categories:

- Algorithm Level Methods: The purposes of these methods are changing learning algorithms, leading the learning process to minority classes, and improving the performance of classifiers. Some of these methods develop independently and improve existing algorithms. The advantages of these methods include big data compatibility and needless of preprocess data. The complexity of computations is the disadvantage of these methods to assign weights [22, 38, 46].
- Data Level Methods: These methods remove the effects of imbalanced datasets before using classifiers. These balancing methods are standard and do not require formula and weight calculations [11, 13, 38, 46, 49].
- Cost-Sensitive Methods: These methods apply changes to the data level and algorithm level. Additionally, considering the wrong costing rate in each class, efforts have been made to minimize costs and to improve results. The disadvantage of this method is the wrong cost calculations in classifiers. There is typically no information about the costs dataset. These methods improve accuracy and precision. So far, attempts have been made to prevent the wrong classification of minority classes, but finding the best formula has complicated calculations for assigning weight to minority classes, which is one of the disadvantages of these methods [27, 38, 46, 53].
- Hybrid Methods: This method uses a combination of the above methods [38, 46].

According to the advantages and disadvantages of balancing methods, the proposed method uses data-level methods. There are three techniques to balance the dataset with data-level methods:

1. Oversampling Technique: This technique increases the samples of minority classes using techniques such as random oversampling, the Synthetic Minority Oversampling Technique (SMOTE), etc. These techniques are highly sensitive because the minority class is a high priority in imbalanced datasets. SMOTE is one of the well-known oversampling techniques. This technique increases samples of the minority class to avoid overfitting problems [11, 13, 38, 46, 49].
2. Undersampling Technique: This technique decreases samples of majority classes using techniques such as random undersampling. Removing data is a key action in these techniques, but important data may be lost. If data deletion does not perform correctly, it affects the classification performance. The disadvantages of this technique are to lose useful data and reduce the classification performance. However, if data is removed based on logical reasoning and appropriate purposes, the desirable results will be obtained [38, 46, 49].
3. Hybrid Technique: This technique uses a combination of oversampling and undersampling techniques [38, 46].

The proposed method uses all data-level techniques to balance the used dataset. Comparing the results of the performance of the used techniques, the best technique selects for the proposed method. In the following section, the proposed method is elaborated.

3 Proposed method

The proposed method consists of four phases which include collecting applications, analyzing the applications and balancing the dataset, creating and evaluating the malware detection model, and detecting malware. The activity diagram of the proposed method is illustrated in Fig. 1.

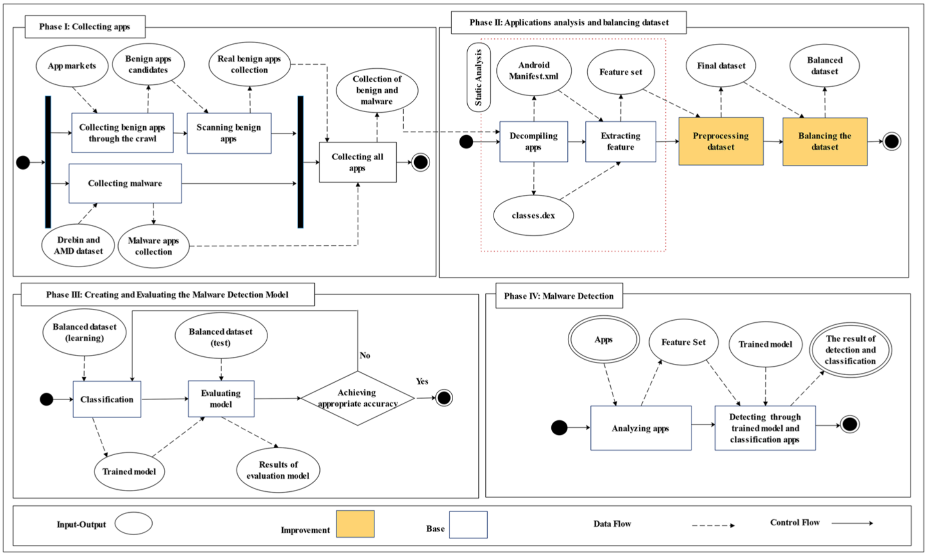


Fig. 1 Phases of the proposed method

3.1 Phase I: collecting applications

The first phase of the detection model is collecting the applications set. The activity diagram of collecting applications is illustrated in Fig. 2. Some conditions are considered to collect applications. Applying these conditions will practically affect the performance of classifiers and increase accuracy. In the first phase, the number of malware and benign applications is considered in a way that there is no balance between these applications. A crawler is used to download benign applications. According to a report published by Kaspersky, in 2018, the countries with the most android malware applications are Bangladesh, Nigeria, Iran, Tanzania, China, and India, respectively [20]. Therefore, the benign applications are collected from the play store and third-party markets of Iran, China, and Bangladesh [7, 18, 21]. Studies show malware producers have specific attention to popular applications [34]. Therefore, the applications are download based on conditions such as the number of downloads, most scores gained by users, multiple versions of updated applications, top applications, and most sales ones in recent years in the app stores. Currently, Google has 36 categories in play store, and all of them are considered in the proposed method.

To consider benign applications, a crystal clear verification is needed to identify real benign applications and prevent bias. Most researchers have considered benign applications without scanning them [2, 6, 10, 29, 34, 39]. Therefore, the candidate benign applications are scanned to identify the real benign ones using five known software including Microsoft, Kingsoft, Kaspersky, Avira, and Comodo. If an application is identified as malware using more than three softwares, this application is deleted. We collected 9223 applications for the detection method. In the proposed method, 6500 benign applications are collected in which 1000 applications belong to the Drebin dataset, and 5500 candidates benign ones belong to the mentioned third-party markets [5]. After scanning the candidate applications, 473 of them are removed. Thereby, 3000 benign applications are used to learn and initial test of the proposed method, and the remaining 3027 applications are used to retest the

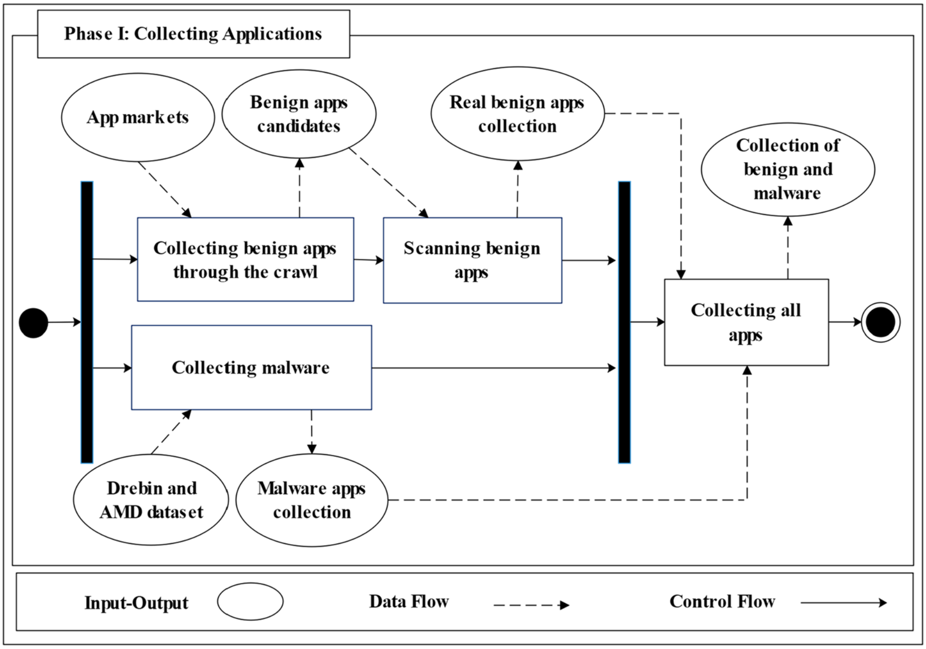


Fig. 2 Process of the collecting applications

model. Also, we collected 2723 malware applications from Drebin[1] and AMD[2] datasets in which 1000 applications are used to learn and 1723 ones are used to retest [5, 47]. These applications are collected from the families of FakeInstaller, Opfake, and FakeDoc. So far, a lot of malware of these families have been discovered [47]. Therefore, the purpose of the proposed method is to identify these types of malware. We have attempted to select the most effective features to identify these malware apps that help distinguish between malware and benign apps. Therefore, the removed features are less effective than the other features. The removed features only have a low effect on detecting this type of malware (families of FakeInstaller, Opfake, and FakeDoc). But it is likely that these features will be effective in other types of malware. Thereby, we used 3000 benign applications and 1000 malware applications to create the detection method (learning).

3.2 Phase II: analyzing applications and balancing dataset

In this phase, the applications collected in the first phase are reviewed and analyzed. Feature selection is effective in the process of detecting malware. The activity diagram of the application analysis phase is illustrated in Fig. 3.

The proposed method uses static analysis to extract features. In this phase, the collected applications decompile using tools of 7-zip and apk-decompiler. Then, static features are extracted from the manifest.xml and classes.dex files. The list of the extracted features is shown in Table 2.

[1] https://www.sec.cs.tu-bs.de/~danarp/drebin/index.html

[2] http://amd.arguslab.org

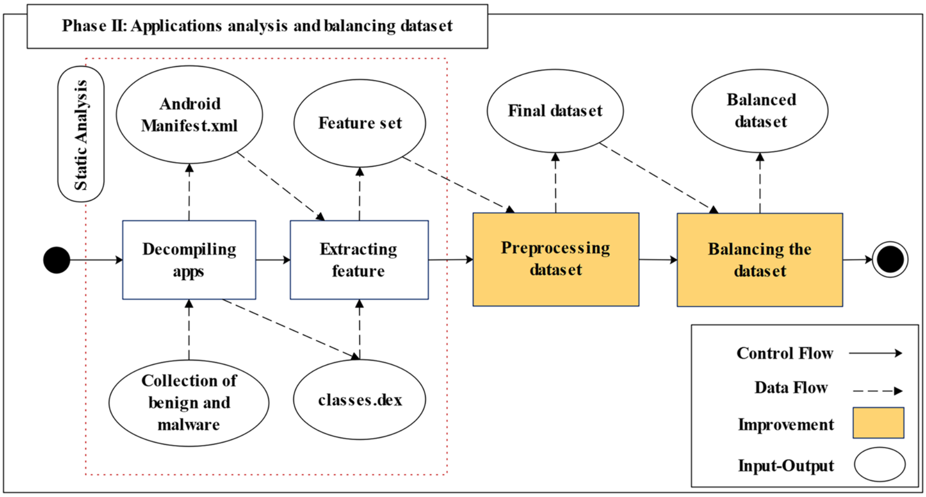


Fig. 3 Process of the analyzing applications and balancing dataset

As it can be seen in Table 2, the extracted features are divided into ten different categories. There are a total number of 1262 features where no preprocessing has been yet done on them. A high number of features without preprocessing led to an increase in computational load and complexity in the detection process [34]. Therefore, if an appropriate set of features is selected, the efficiency of malware detection is improved [2]. The proposed method has performed a preprocessing on the used dataset to remove the ineffective features. In the first step of the preprocessing, the existence of null data must be checked. In the obtained dataset, there is no null data. After this step, the ineffective features should be identified and removed.

Removing ineffective features causes complexity reduction and training time of the model. In the proposed method, two methods are implemented to remove the ineffective features, which are described below. In the categories with a small number of features, the frequency of each category is calculated in malware and benign applications. Also, the frequency of each feature is gained based on the existence or nonexistence (0 or 1) of the

Table 2 Number of extracted features

Feature Set Name	Number of Extracted Features
Activity	133
Intent	112
Real Permission	48
Request Permission	133
Hardware Component	16
Service Reciever	230
Call	35
Api Call	123
Provider	217
Urls	214

features in all applications. For example, we extract features from 4000 applications (3000 benign apps and 1000 malware apps) to create the model. The frequency of the hardware component features is shown in Table 3.

A scatter plot is also used to analyze features in a better way. This plot allows us to have a clear recognition of the distribution of the frequency of features. For example, the scatter plot of the hardware components is illustrated in Fig. 4.

The numbers of 1 to 17 in Fig. 4 represent the number of features listed in Table 3. To remove the ineffective features, the following conditions are considered:

– The features which are just used in malware or benign applications.
– The features which are the most frequently that are used in both malware and benign applications will be removed. As it can be seen in Table 3, the "touchscreen" feature has the most frequency in both types of applications. So, this feature is present in all applications and cannot be used to differentiate between malware and benign applications. The features like "touchscreen" will mislead the detection model and will not help classification.
– The features which are rarely used in either malware or benign applications: For example, the frequency of the "camera.autofocus" feature is 178 in which 168 of them belong to benign applications, and 10 of them belong to malware. Therefore, the existence of features like "camera.autofocus" is not essential as it will not help classification and it has no considerable effect on the detection performance.

Thereby, the ineffective features are removed according to the explained conditions and the analysis of the plots. Therefore, in the category of hardware components, the features such as screen.portrait, location, wifi, location.network, location.gps, and telephony are considered as selected features and the rest of them are removed. In categories with a large

Table 3 Frequency of the hardware components in malware applications and benign ones

ID	Hardware Components	Frequency in Malware Apps	Frequency in Benign Apps	Frequency in all Apps	Frequency in none of Apps
1	touchscreen	955	3000	3955	45
2	screen.portrait	158	1125	1283	2717
3	location	268	770	1038	2962
4	location.network	198	618	816	3184
5	location.gps	229	568	797	3203
6	screen.landscape	28	423	451	3549
7	wifi	308	375	683	3317
8	telephony	823	286	1109	2891
9	camera	19	236	255	3745
10	camera.autofocus	10	168	178	3822
11	microphone	7	84	97	3989
12	live wallpaper	1	29	30	3970
13	multitouch	0	4	4	3996
14	bluetooth	0	3	3	3997
15	accelerometer	0	3	3	3997
16	glEsVersion	5	0	5	3995
17	camera.flash	4	0	4	3996

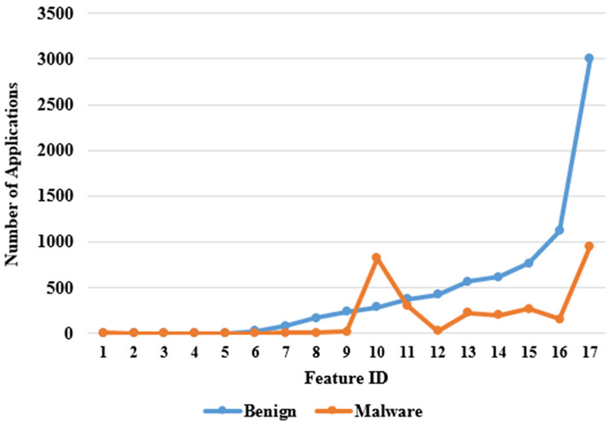


Fig. 4 Scatter plot of the hardware components category

number of features such as the request permissions, a data frame is initially created for each category which specifies the frequency of features. The ineffective features are removed by calculating the percentage of difference in obtained frequencies regarding the existence (Freq1) or nonexistence (Freq0) features. The difference between the gained values is in the range of 0 to 1. If the obtained difference is close to 0, then the distribution of values is the same. After analyzing the obtained values for each category, the obtained values are set (tune). According to the tuning value of each category, it can be said the features can be selected that differ less from the tuned value while the rest of features are recognized as the ineffective features (with a higher difference) and are removed. The tuned values are different for each category of features. For example, the set values of categories such as url, intent, and API Call are 0.8, 0.77, and 0.6, respectively. The part of calculations for removing the ineffective features from the request permissions are shown in Table 4.

As it can be seen in Table 4, the percentage difference of the features such as Call_phone, Set_wallpaper, Change_wifi_state, Mount_unmount_filesystem, and, Record_audio is in the range of 0.7 to 1. These features have been used in many applications or have been used in a few applications. Therefore, these features are not effective in detection and should be removed. Finally, in the request permissions category, the features with a percentage difference greater than 0.6 are removed. The selected features from 133 features of the request permissions are shown in Table 5.

After doing preprocessing, 78 features are selected and 1183 features are removed. Thereby, we obtained the preprocessed dataset. There is no balance between malware and benign applications in the obtained dataset. Hence, after preprocessing, we use the techniques of SMOTE, Random undersampling, and Hybrid to balance the dataset. In the random undersampling technique, some conditions have taken effect to remove the records of the dataset. The technique efforts to remove the records that their existence does not affect or low affect the detection process. At first, the records are selected which are rarely used in the dataset and they have a lot in common. The random undersampling technique is then applied to these records. Thereby, the changes prevent deletion of the effective records. As a result, we balance the initial dataset (1000 malware and 3000 benign applications) using these techniques. The mentioned techniques are used to balance the initial dataset. The results of the balancing dataset are shown in Table 6.

Table 4 Percentage of the difference of frequency in some of the request permissions

ID	Request Permissions Name	Freq0	Freq1	Percentage of Difference
1	ACCESS_NETWORK_STATE	567	637	0.05814
2	ACCESS_CHECKIN_PROPERTIES	143	1061	0.76246
3	ACCESS_DYNAMIC_REFRESH	1184	20	0.96678
4	INTERNET	1203	1	0.99834
5	WRITE_EXTERNAL_STORAGE	650	554	0.07973
6	READ_PHONE_STATE	592	612	0.01661
7	WAKE_LOCK	494	710	0.1794
8	ACCESS_FINE_LOCATION	857	347	0.42359
9	RECEIVE_BOOT_COMPLETED	957	247	0.5897
10	ACCESS_WIFI_STATE	977	227	0.62292
11	WRITE_SETTINGS	1070	134	0.77741
12	ACCESS_COARSE_LOCATION	1006	198	0.6711
13	VIBRATE	903	301	0.5
14	SEND_SMS	877	327	0.45681
15	READ_CONTACTS	1145	59	0.90199
16	GET_TASKS	1177	27	0.95515
17	CALL_PHONE	1071	133	0.77907
18	SET_WALLPAPER	1027	177	0.70598
19	CHANGE_WIFI_STATE	1145	59	0.90199
20	RECORD_AUDIO	1172	32	0.94684
21	WRITE_CONTACTS	1043	161	0.73256
22	MOUNT_UNMOUNT_FILESYSTEMS	1194	10	0.98339
23	GET_ACCOUNTS	1172	32	0.94684

Table 5 Selected features in the category of request permissions

ID	Request Permissions Name	Percentage of Difference
1	ACCESS_NETWORK_STATE	0.05814
5	WRITE_EXTERNAL_STORAGE	0.079734
6	READ_PHONE_STATE	0.016611
7	WAKE_LOCK	0.179402
8	ACCESS_FINE_LOCATION	0.423588
9	RECEIVE_BOOT_COMPLETED	0.589701
13	VIBRATE	0.5
14	SEND_SMS	0.456811
26	RECEIVE_SMS	0.019934
29	RESTART_PACKAGES	0.267442
31	READ_SMS	0.596346
35	ACCESS_NETWORK_STATE	0.480066

Table 6 Number of applications after balancing the initial dataset

Balancing Techniques	Number of Benign Apps	Number of Malware
SMOTE	3000	2958
Random Technique	974	1000
Combined Technique	2075	1942

3.3 Phase III: creating and evaluating the malware detection model

In phase II, the preprocessing is applied to remove the ineffective features. Then, the initial dataset is balanced using the techniques of SMOTE, Random undersampling, and Hybrid. The third phase of the proposed method is creating the detection model with the initial dataset. So far, Different classifiers are used to create a detection model. The results of the studies indicate that SVM and KNN have been used widely in malware detection methods. In this phase, the detection model is created using the most common classification algorithms such as SVM, ID3, and KNN [2, 4, 5, 8, 25, 26, 30, 31, 33, 35, 37, 39, 41–43, 45, 48, 50–52, 54]. Therefore, we examined these classifiers. The accuracy of KNN and SVM is high and comparable to each other. These classifiers are black-box models. A black-box model has great accuracy. But it is difficult to understand the internal functions and check the interaction between features. Hence, we used ID3 that it is a white-box model. White-box models make it easy to interpret and explain, significantly. We used ID3 to increase ensure the correct performance of the classifiers. We then reviewed the advantages of these classifiers. The advantages of KNN are ease of output interpretation, low computational time, simple to implement, high accuracy, and suitable for quantities dataset. The time complexity of KNN is O(n^2) and the space complexity is O(n). The advantages of SVM are high accuracy, more efficient in higher dimensions, and optimally in memory consumption. The time complexity of SVM is generally between O(n^2) and O(n^3). The advantage of ID3 is interpretability, low complexity, understandable prediction rules, build the fastest tree among decision tree algorithms, and build a short tree. The time complexity of ID3 is O(nm) [24, 42]. Therefore, the classifiers of KNN, ID3, and SVM classifiers were considered for the proposed method by studying the related studies and reviewing the classifiers used. Finally, the best classifier is selected to detect malware by comparing the performance of classifiers. The activity diagram of this phase is illustrated in Fig. 5.

The K-Fold cross-validation (k=10) method is used to train and test the detection model so that it prevents the possibility of overfitting and the obtained results are reliable. Therefore, all applications are learned once and are tested once. Finally, the best classifier is selected based on the obtained results and comparison the performance of the classifiers. The proposed method tries to improve accuracy and to reduce the error rate using preprocessing dataset and balancing the used dataset.

3.4 Phase IV: detecting malware

After achieving acceptable accuracy, the number of applications increases, and the phases one to three re-perform. After entering the applications and analyzing them, the selected features are extracted from the applications. This dataset is then balanced and

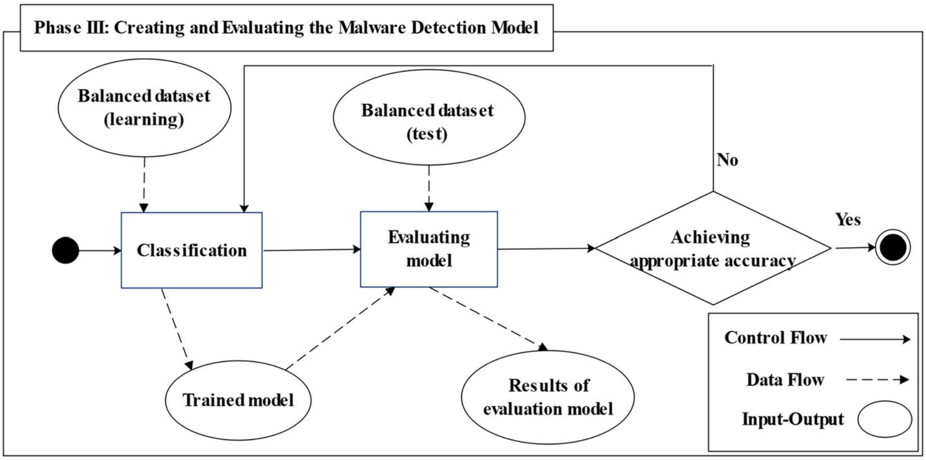

Phase III: Creating and Evaluating the Malware Detection Model
Balanced dataset (learning)
Balanced dataset (test)
Classification
Evaluating model
Achieving appropriate accuracy
No
Yes
Trained model
Results of evaluation model
Control Flow
Data Flow
Input-Output

Fig. 5 Process of the creating and evaluating malware detection model

considered as input to the detection model. In this phase, 3027 other benign applications and 1723 other malware applications are added to the dataset and the model is re-evaluated. The activity diagram of the malware detection phase is illustrated in Fig. 6.

After increasing applications in this phase, the final dataset is also balanced. Thereby, the malware detection process is completed in this phase. The obtained results are elaborated in Section 4. The results of the balancing dataset are shown in Table 7.

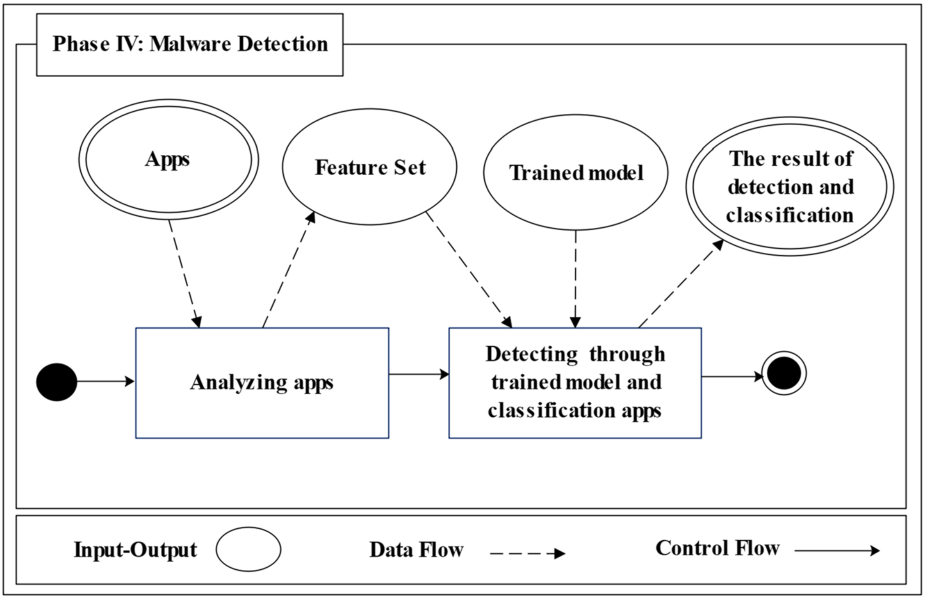

Phase IV: Malware Detection
Apps
Feature Set
Trained model
The result of detection and classification
Analyzing apps
Detecting through trained model and classification apps
Input-Output
Data Flow
Control Flow

Fig. 6 Process of the Malware detection

Table 7 Number of applications after balancing the final dataset

Balancing Techniques	Number of Benign Apps	Number of Malware
SMOTE	6027	5985
Random Technique	2674	2723
Combined Technique	5423	5288

4 Evaluation of the proposed method

In this section, the proposed method is evaluated. A common challenge to evaluate the malware detection methods is selecting suitable criteria. This is particularly important when the used dataset is an imbalance. The quality of malware detection methods is determined by several criteria, including **1)** False Positive (FP): The number of samples that are incorrectly identified as malware, **2)** True Positive (TP): The number of samples that are correctly identified as malware, **3)** False negative (FN): The number of samples that are incorrectly identified as benign, and **4)** True Negative (TN): The number of samples that are correctly identified as benign. Accuracy is widely used for evaluating the performance of machine learning algorithms [5, 15, 26, 30, 31, 35, 37, 50]. However, accuracy is sensitive in imbalanced datasets. Also, precision, recall, or F-measure can be confusing and misleading over imbalanced datasets. MCC is less influenced by imbalanced datasets because it considers accuracy and the error rate in both classes [9, 17]. The results of the proposed method are evaluated with the following criteria:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FN) \times (FN + TN) \times (FN + TN) \times (TP + FN)}} \tag{5}$$

4.1 Evaluating results of the proposed method

As mentioned in Section 3, the malware collection was obtained from the datasets of Drebin and AMD [5, 47]. The k-fold cross-validation method has also been used to train and test the detection model. As such, the dataset once used to train and once used to test. Finally, the detection model was constructed using SVM, KNN, and ID3. At first, the results of the creation of the detection model are examined without preprocessing of the extracted features. The obtained results of the performance of the detection model are shown in Table 8.

The obtained results indicate that the performance of KNN has better than the performance of the other classifiers. According to the explanations at the beginning of this section, we considered MCC and F-measure to compare the performance of the classifiers. The

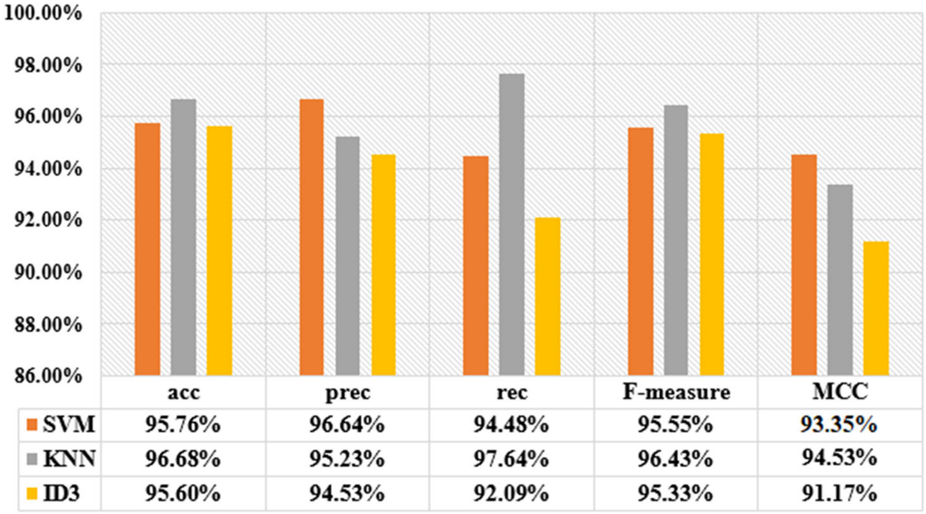
Table 8 The performance results of the proposed method before preprocessing the extracted features

Classifier	Accuracy	Precision	Recall	F-measure	MCC
SVM	92.77%	92.68%	92.20%	92.44%	85.55%
KNN	92.93%	93.87%	91.47%	92.65%	85.97%
ID3	92.24%	90.49%	90.49%	90.49%	83.94%

results of these criteria are more reliable than the other criteria. After preprocessing the extracted features, the used dataset is balanced. So, the results of the detection method are evaluated over the imbalanced dataset. The best classifier is then determined based on the results of the preprocessed features and the imbalanced dataset. The evaluation results are illustrated in Fig. 7.

The obtained results indicate that the accuracy of KNN is 96.68% in the initial imbalanced dataset (1000 malware and 3000 benign applications). The comparison of the different evaluation criteria indicates that the performance of KNN is better than the other classifiers. The comparison between the results presented in Table 8 and the results presented in Fig. 7 indicates that the performance of the proposed method has been improved by preprocessing the extracted features. After ensuring the obtained results, the dataset is balanced using the balancing techniques. The results of the performance classifiers on the trained and balanced dataset are shown in Table 9.

The obtained results indicate that the lowest accuracy belongs to SVM that uses the combined technique of balancing with an accuracy of 97.59%. Therefore, the performance of the proposed method is desirable in all the used classifiers and all balancing techniques. The results indicate that the performance of the classifiers with SMOTE is better than the performance of the classifiers with the other balancing techniques. Thereby, the performance of KNN is better than the other classifiers with an accuracy of 98.69% in the balanced dataset using SMOTE. After ensuring the performance of the trained detection model, we increase



	acc	prec	rec	F-measure	MCC
■ SVM	95.76%	96.64%	94.48%	95.55%	93.35%
■ KNN	96.68%	95.23%	97.64%	96.43%	94.53%
■ ID3	95.60%	94.53%	92.09%	95.33%	91.17%

Fig. 7 The performance results of the proposed method after preprocessing the extracted features over the imbalanced dataset

Table 9 Comparison between the results of the classifiers with balancing techniques on the final balanced dataset

Evaluation criteria	Balancing techniques	SMOTE	Random Technique	Combined Technique
SVM	Accuracy	97.83%	97.64%	97.59%
	Precision	95.73%	95.79%	95.66%
	Recall	99.96%	99.36%	99.64%
	F-measure	97.80%	97.54%	95.59%
	MCC	95.76%	95.33%	95.20%
KNN	Accuracy	**98.69%**	98.54%	98.43%
	Precision	**97.89%**	97.63%	97.39%
	Recall	**99.49%**	99.37%	99.55%
	F-measure	**98.69%**	98.50%	98.47%
	MCC	**97.39%**	97.10%	96.89%
ID3	Accuracy	98.24%	98.04%	97.88%
	Precision	97.06%	96.71%	96.24%
	Recall	97.07%	99.26%	99.65%
	F-measure	98.22%	97.97%	97.92%
	MCC	96.50%	96.11%	95.83%

The best obtained results are shown in bold

the number of applications and re-evaluate the performance of the proposed method (phase IV). In this phase, 1723 other malware applications and 3026 other benign ones add to the dataset. The results of increasing the number of applications are shown in Table 10.

As shown in Table 10, increasing the number of applications has improved the performance of SVM and ID3. After balancing the dataset, we can consider accuracy in order to evaluate and compare. The performance of KNN also remains constant. After increasing applications, the lowest accuracy is 97.44% (SVM with SMOTE). Comparing the obtained results of Tables 9 and 10 indicates that after balancing the dataset, the performance of the classifiers has improved. As a result, KNN with SMOTE has the best performance among the used classifiers with an accuracy of 98.69%. The error rate is 1.3% in KNN. In the following section, the proposed method is compared to the related studies.

The following conditions are considered to make a better comparison of the proposed method with the other studies: A method that uses the datasets of Drebin and/or AMD to collect applications, uses static analysis to extract features, and uses similar classifiers to the proposed method. The approaches presented in [4, 5, 15, 32, 36, 51] were selected for evaluation. The comparison results of the performance are shown in Table 11.

As shown in Table 11, the accuracy of the proposed method is more than the presented approaches in [4, 5, 15, 32, 36, 51]. In the detection methods proposed in [32, 36, 51] the number of malware and benign applications are considered equal. In these approaches, accuracy is reported for evaluation. Therefore, the other evaluation criteria are not described in Table 11. These approaches use SVM to create the detection model. Therefore, the results of SVM are also presented in this table. The obtained results indicate that the proposed method performs better than these approaches in detecting malware and classifying applications. In the following, the performance of the proposed method compares to the SIGPID

Table 10 Comparison between the results of the classifiers with balancing techniques on the final balanced dataset

Evaluation criteria	Balancing techniques	SMOTE	Random Technique	Combined Technique
SVM	Accuracy	97.44%	97.90%	97.66%
	Precision	94.76%	95.96%	95.85%
	Recall	100%	99.86%	99.59%
	F-measure	97.31%	97.87%	97.69%
	MCC	94.99%	95.87%	95.39%
KNN	Accuracy	**98.69%**	98.54%	98.43%
	Precision	**97.89%**	97.63%	97.39%
	Recall	**99.49%**	99.37%	99.55%
	F-measure	**98.69%**	98.50%	98.47%
	MCC	**97.39%**	97.10%	96.89%
ID3	Accuracy	98.27%	98.14%	97.88%
	Precision	97.13%	96.91%	96.24%
	Recall	99.41%	99.89%	99.65%
	F-measure	98.26%	98.38%	96.13%
	MCC	96.57%	96.91%	95.83%

The best obtained results are shown in bold

method [26]. SIGPID uses static analysis to extract features and uses the category of permissions. There is no balance in the dataset. But preprocessing is performed on the dataset. The results of comparing the proposed one with SIGPID are shown Table 12.

The obtained results indicate that the proposed method performs better than SIGPID. As a result, the comparison between the results of the proposed method and the results of the related studies indicates that the performance of the proposed method is better than the performance of the other methods [4, 5, 15, 26, 32, 36, 51]. The proposed method tries to improve accuracy and reduce the error rate by scanning benign applications, preprocessing to remove the ineffective features, and balancing the used dataset.

Table 11 Comparison of the accuracy between the proposed method with the related studies

Comparison Methods	Preprocessing Dataset	Balancing Dataset	Accuracy
Proposed Method (SVM)	✓	✓	97.44%
Proposed Method (KNN)	✓	✓	98.69%
[5]	×	×	93%
[4]	×	×	89.60%
[51]	×	✓	89.4%
[15]	✓	×	98%
[32]	×	✓	89.26%
[36]	×	✓	92.73%

Table 12 Comparison between the performance of the proposed method and the performance of SIGPID

Comparison Methods	Proposed Method (SVM)	Proposed Method (KNN)	SIGPID
Accuracy	97.44%	98.69%	93.62%
Precision	95.73%	97.89%	96.39%
Recall	99.69%	99.49%	85.78%
F-measure	97.80%	98.69%	90.77%
Balancing Dataset	Yes	Yes	No
Preprocessing Dataset	Yes	Yes	Yes

5 Conclusion and future works

In this paper, a method was proposed to detect malware and classify applications. Static analysis was used to extract features from applications. Knowing that the number of the extracted features is very large, we performed preprocessing to rank the features and eliminate the ineffective features. Preprocessing avoids the increased computational burden and complexity of the detection model. Many researchers did not perform preprocessing on the extracted features. Many related studies used features with the highest frequency to create a detection model using ranking algorithms. However, the results indicated that the highest frequency of features is not appropriate to be used in constructing a malware detection model. Therefore, the preprocessing of features is necessary to improve the accuracy of the malware detection process. We found out that the challenge of balancing the dataset is significant in detecting malware. Hence, we balanced the dataset using the techniques of SMOTE, random undersampling, and hybrid. Eventually, KNN was able to detect malware with an accuracy of 98.69% using SMOTE. In the proposed method, despite the high accuracy obtained, the percentage of benign applications that were detected as malware (FPR) was in the range of 2.09% and 4.77%. Therefore, reducing FPR is to be considered in future works. Family detection can also be an effective step in the process of malware detection. The proposed method did not address this issue. Family detection helps to act stronger against the threats of different types of malware and prevents further damage. Therefore, family detection is also to be considered in future works.

Other issues that can be addressed in future works. These topics include detection of different types of malware associated with multimedia social networks, detection of malicious activities of multimedia applications based on a mobile camera, security of multimedia digital data in the field of mobile, and identification of malware in multimedia video files, detection of malware in the internet of things (IoT) environments and investigation of mobile multimedia applications, detection of malware using dynamic analysis and monitoring multimedia applications, and finally, detection of malware using deep learning techniques, image visualization, and Image-based techniques. Also, we can extend and apply our idea to other multimedia applications, such as 3D data exchange, malicious media files detection, image segmentation, security threats detection in mobile multimedia applications, image detection, malware detection in the multimedia social network, multimedia content detection in file-sharing networks, and image dehazing.

References

1. Aafer Y, Du W, Yin H (2013) Droidapiminer: Mining api-level features for robust malware detection in android. In: International conference on security and privacy in communication systems, pp 86–103, Springer
2. Agrawal P, Trivedi B (2019) A survey on android malware and their detection techniques. In: 2019 IEEE International conference on electrical, computer and communication technologies (ICECCT), pp 1–6, IEEE. https://doi.org/10.1109/ICECCT.2019.8868951
3. Ahmadi M, Ulyanov D, Semenov S, Trofimov M, Giacinto G (2016) 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, pp 183–194, ACM. https://doi.org/10.1145/2857705.2857713
4. Alam S, Qu Z, Riley R, Chen Y, Rastogi V (2017) Droidnative: Automating and optimizing detection of android native code malware variants. Comput Secur 65:230–246. https://doi.org/10.1016/j.cose.2016.11.011
5. Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K, Siemens C (2014) Drebin: Effective and explainable detection of android malware in your pocket. In: Ndss, vol. 14, pp 23–26
6. Aung Z, Zaw W (2013) Permission-based android malware detection. Int J Sci Technol Res 2(3):228–234
7. Apkpure apps store(bangladesh) (2019). https://apkpure.com/developer/Apps%20for%20Bangladesh
8. Backes M, Nauman M (2017) Luna: quantifying and leveraging uncertainty in android malware analysis through bayesian machine learning. In: 2017 IEEE European symposium on security and privacy (euros&p), pp 204–217, IEEE. https://doi.org/10.1109/EuroSP.2017.24
9. Bekkar M, Djemaa HK, Alitouche TA (2013) Evaluation measures for models assessment over imbalanced data sets. J Inf Eng Appl 3(10):1,2, and 4
10. Canfora G, Di Sorbo A, Mercaldo F, Visaggio CA (2015) Obfuscation techniques against signature-based detection: a case study. In: 2015 Mobile systems technologies workshop (MST), pp 21–26, IEEE. https://doi.org/10.1109/MST.2015.8
11. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357. https://doi.org/10.1613/jair.953
12. Dong S, Li M, Diao W, Liu X, Liu J, Li Z, Xu F, Chen K, Wang X, Zhang K (2018) Understanding android obfuscation techniques: A large-scale investigation in the wild. In: International conference on security and privacy in communication systems, pp 172–192, Springer. https://doi.org/10.1007/978-3-030-01701-9_10
13. Fernández A, Garcia S, Herrera F, Chawla NV (2018) Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J Artif Intell Res 61:863–905. https://doi.org/10.1613/jair.1.11192
14. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F (2018) Imbalanced classification for big data. In: Learning from imbalanced data sets, pp 327–349, Springer. https://doi.org/10.1007/978-3-319-98074-4_13
15. Garcia J, Hammad M, Malek S (2018) Lightweight, obfuscation-resilient detection and family identification of android malware. ACM Trans Softw Eng Methodology (TOSEM) 26(3):11
16. Grace M, Zhou Y, Zhang Q, Zou S, Jiang X (2012) Riskranker: scalable and accurate zero-day android malware detection. In: Proceedings of the 10th international conference on mobile systems, applications, and services, pp 281–294, ACM. https://doi.org/10.1145/2307636.2307663
17. Halimu C, Kasem A, Newaz S (2019) Empirical comparison of area under roc curve (auc) and mathew correlation coefficient (mcc) for evaluating machine learning algorithms on imbalanced datasets for binary classification. In: Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, pp 1–6. ACM
18. Huawei apps store(china) (2019). https://appstore.huawei.com/
19. Hung SH, Tu CH, Yeh CW (2016) A cloud-assisted malware detection framework for mobile devices. In: 2016 International computer symposium (ICS), pp 537–54, IEEE. https://doi.org/10.1109/ICS.2016.0112
20. It threat evolution q3 (2018) statistics — securelist. https://securelist.com/itthreat-evolution-q3-2018-statistics/88689/. [Accessed: 22-Feb-2019]
21. Iranapps apps store (2019). https://iranapps.ir/
22. Krawczyk B (2016) Learning from imbalanced data: open challenges and future directions. Progr Artif Intell 5(4):221–232. https://doi.org/10.1007/s13748-016-0094-0
23. Kuncheva LI, Arnaiz-González Á, Díez-pastor JF, Gunn IA (2019) Instance selection improves geometric mean accuracy: a study on imbalanced data classification. Progr Artif Intell 8(2):215–228. https://doi.org/10.1007/s13748-019-00172-4

24. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N (2018) A survey on addressing high-class imbalance in big data. J Big Data 5(1):42. https://doi.org/10.1186/s40537-018-0151-6
25. Lei T, Qin Z, Wang Z, Li Q, Ye D (2019) Evedroid: Event-aware android malware detection against model degrading for iot devices. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2019.2909745
26. Li J, Sun L, Yan Q, Li Z, Srisa-an W, Ye H (2018) Significant permission identification for machine-learning-based android malware detection. IEEE Trans Industr Inform 14(7):3216–3225. https://doi.org/10.1109/TII.2017.2789219
27. Ling CX, Sheng VS, Yang Q (2006) Test strategies for cost-sensitive decision trees. IEEE Trans Knowl Data Eng 18(8):1055–1067. https://doi.org/10.1109/TKDE.2006.131
28. Liu J, Zio E (2019) Integration of feature vector selection and support vector machine for classification of imbalanced data. Appl Soft Comput 75:702–711. https://doi.org/10.1016/j.asoc.2018.11.045
29. Lopez-Garcia P, Masegosa AD, Osaba E, Onieva E, Perallos A (2019) Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. Appl Intell 49:1–16. https://doi.org/10.1007/s10489-019-01423-6
30. Lou S, Cheng S, Huang J, Jiang F (2019) Tfdroid: Android malware detection by topics and sensitive data flows using machine learning techniques. In: 2019 IEEE 2Nd international conference on information and computer technologies (ICICT), pp 30–36, IEEE. https://doi.org/10.1109/INFOCT.2019.8711179
31. Martinelli F, Mercaldo F, Nardone V, Santone A, Sangaiah AK, Cimitile A (2018) Evaluating model checking for cyber threats code obfuscation identification. J Parallel Distrib Comput 119:203–218. https://doi.org/10.1016/j.jpdc.2018.04.008
32. Martín A, Lara-Cabrera R, Camacho D (2019) Android malware detection through hybrid features fusion and ensemble classifiers: the andropytool framework and the omnidroid dataset. Inform Fusion 52:128–142
33. McGiff J, Hatcher WG, Nguyen J, Yu W, Blasch E, Lu C (2019) Towards multimodal learning for android malware detection. In: 2019 International conference on computing, networking and communications (ICNC), pp 432–436, IEEE. https://doi.org/10.1109/ICCNC.2019.8685502
34. Odusami M, Abayomi-Alli O, Misra S, Shobayo O, Damasevicius R, Maskeliunas R (2018) Android malware detection: a survey. In: International conference on applied informatics, pp 255–266, Springer
35. Pektaş A, Acarman T (2019) Learning to detect android malware via opcode sequences. Neurocomputing. https://doi.org/10.1016/j.neucom.2018.09.102
36. Pektaş A, Acarman T (2020) Learning to detect android malware via opcode sequences. Neurocomputing 396:599–608
37. Quan D, Zhai L, Yang F, Wang P (2014) Detection of android malicious apps based on the sensitive behaviors. In: 2014 IEEE 13Th international conference on trust, security and privacy in computing and communications, pp 877–883, IEEE. https://doi.org/10.1109/TrustCom.2014.115
38. Rout N, Mishra D, Mallick MK (2018) Handling imbalanced data: a survey. In: International proceedings on advances in soft computing, intelligent systems and applications, pp 431–443, Springer. https://doi.org/10.1007/978-981-10-5272-9_39
39. Samra AAA, Qunoo HN, Al-Rubaie F, El-Talli H (2019) A survey of static android malware detection techniques. In: 2019 IEEE 7Th palestinian international conference on electrical and computer engineering (PICECE), pp 1–6, IEEE. https://doi.org/10.1109/PICECE.2019.8747224
40. Saracino A, Sgandurra D, Dini G, Martinelli F (2016) Madam: Effective and efficient behavior-based android malware detection and prevention. IEEE Trans Dependable Secure Comput 15(1):83–97
41. Shrivastava G, Kumar P (2019) Sensdroid: analysis for malicious activity risk of android application. Multimed Tools Appl 78:1–19. https://doi.org/10.1007/s11042-019-07899-1
42. Siddiqui M, Wang MC, Lee J (2008) A survey of data mining techniques for malware detection using file features. In: Proceedings of the 46th annual southeast regional conference on xx, pp 509–510, ACM. https://doi.org/10.1145/1593105.1593239
43. Suarez-Tangil G, Dash SK, Ahmadi M, Kinder J, Giacinto G, Cavallaro L (2017) Droidsieve: Fast and accurate classification of obfuscated android malware. In: Proceedings of the Seventh ACM on conference on data and application security and privacy, pp 309–320 ACM. https://doi.org/10.1145/3029806.3029825
44. Tavallaee M, Stakhanova N, Ghorbani AA (2010) Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Trans Syst Man Cy Part C (App Rev 40(5):516–524. https://doi.org/10.1109/TSMCC.2010.2048428
45. Ucci D, Aniello L, Baldoni R (2018) Survey of machine learning techniques for malware analysis. Computers & Security. https://doi.org/10.1016/j.cose.2018.11.001

46. Wang S, Yao X (2009) Diversity analysis on imbalanced data sets by using ensemble models. In: 2009 IEEE Symposium on computational intelligence and data mining, pp 324–331, IEEE. https://doi.org/10.1109/CIDM.2009.4938667
47. Wei F., Li Y., Roy S., Ou X., Zhou W. (2017) Deep ground truth analysis of current android malware. In: International conference on detection of intrusions and malware, and vulnerability assessment (DIMVA'17), pp 252–276. Springer, Bonn, Germany. https://doi.org/10.1007/978-3-319-60876-1_12
48. Yan P, Yan Z (2018) A survey on dynamic mobile malware detection. Softw Qual J 26(3):891–919. https://doi.org/10.1007/s11219-017-9368-4
49. Yang Q (2006) Wu, x.: 10 challenging problems in data mining research. Int J Inform Technol Dec Making 5(04):597–604. https://doi.org/10.1142/S0219622006002258
50. Yang W, Xiao X, Andow B, Li S, Xie T, Enck W (2015) Appcontext: Differentiating malicious and benign mobile app behaviors using context. In: Proceedings of the 37th international conference on software engineering-volume 1, pp 303–313. IEEE Press
51. Yerima SY, Sezer S (2018) Droidfusion: a novel multilevel classifier fusion approach for android malware detection. IEEE Trans Cybern 49(2):453–466. https://doi.org/10.1109/TCYB.2017.2777960
52. Yuan Z, Lu Y, Xue Y (2016) Droiddetector: android malware characterization and detection using deep learning. Tsinghua Sci Technol 21(1):114–123. https://doi.org/10.1109/TST.2016.7399288
53. Zhao L, Shang Z, Qin A, Zhang T, Zhao L, Wei Y, Tang YY (2019) A cost-sensitive meta-learning classifier: Spfcnn-miner future generation computer systems. https://doi.org/10.1016/j.future.2019.05.080
54. Zhou Q, Feng F, Shen Z, Zhou R, Hsieh MY, Li KC (2019) A novel approach for mobile malware classification and detection in android systems. Multimed Tools Appl 78(3):3529–3552. https://doi.org/10.1007/s11042-018-6498-z

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.