International Journal of Machine Learning and Cybernetics (2021) 12:1369–1411
https://doi.org/10.1007/s13042-020-01238-9

ORIGINAL ARTICLE

SemiDroid: a behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches

Arvind Mahindru[1,2] · **A. L. Sangal**[1]

Received: 22 October 2019 / Accepted: 5 November 2020 / Published online: 24 November 2020

Abstract
With the exponential growth in Android apps, Android based devices are becoming victims of target attackers in the "silent battle" of cybernetics. To protect Android based devices from malware has become more complex and crucial for academicians and researchers. The main vulnerability lies in the underlying permission model of Android apps. Android apps demand permission or permission sets at the time of their installation. In this study, we consider permission and API calls as features that help in developing a model for malware detection. To select appropriate features or feature sets from thirty different categories of Android apps, we implemented ten distinct feature selection approaches. With the help of selected feature sets we developed distinct models by using five different unsupervised machine learning algorithms. We conduct an experiment on 5,00,000 distinct Android apps which belongs to thirty distinct categories. Empirical results reveals that the model build by considering rough set analysis as a feature selection approach, and farthest first as a machine learning algorithm achieved the highest detection rate of 98.8% to detect malware from real-world apps.

Keywords Android apps · Permissions model · API calls · Unsupervised · Feature selection · Intrusion detection · Cyber security · Smartphone

1 Introduction

Detection of malware from smartphones has become a major concern for the research community. At the end of 2019, the number of Android users will be 3.3 billions throughout the world.[1] Android is based on the Linux kernel and provide useful services such as security configuration, process management and others. The primary reason for the growth of Android operating system is due to its open-nature and freely available apps. At the end of July 2019,[2] Android had 2.7 billion free and paid apps in its play store. There is an increase of 13%,[3] in downloading of apps from Google play store with respect to previous years. Android operating system is based on the principle of privilege-separated where each app has its own distinct system identity, i.e., group-ID and Linux user-ID. Each app run in a procedure sandbox and access permission to use the resources which are not present in its sandbox. Depending on the permission sensitivity, the system automatically grants permission or may prompt users to approve or reject requests for permission. Permissions granted by users include, access to the calendar, camera, body sensors, microphone, contacts, location, SMS, storage of the device. To defend Google official market[4] from malware-infected apps, Google introduced Google Bouncer in the year 2012, which scans new apps at the time of their launch. But, Google Bouncer has limitation, it can easily be fingerprint.[5] It is not very difficult for malware apps to bypass Google's security check and enter to Google official market[6] and ultimately to users' devices. By taking advantage of these permissions, cyber-criminals build malware

✉ Arvind Mahindru
er.arvindmahindru@gmail.com

[1] Department of Computer Science and Engineering, Dr. B.R. Ambedkar National Institute of Technology, Jalandhar 144011, India

[2] Department of Computer Science and Applications, D.A.V. University, Sarmastpur, Jalandhar 144012, India

[1] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[2] https://www.appbrain.com/stats.

[3] https://www.businessofapps.com/data/app-statistics/#1.

[4] https://en.wikipedia.org/wiki/Google_Play.

[5] http://blog.trendmicro.com/trendlabs-security-intelligence/a-look-at-google-bouncer/.

[6] https://play.google.com/store?hl=en.

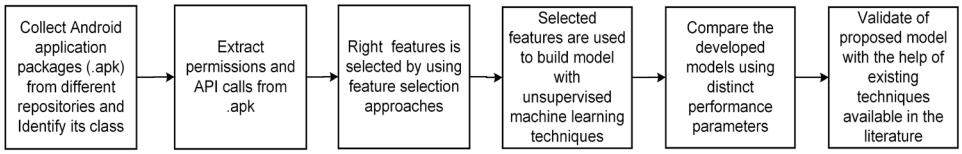


Fig. 1 Flow chart of the proposed work

apps on a daily basis and invite users to install these applications. More than two billion active Android devices are present in the market.[7] To overcome the drawback of the bouncer and to protect Android devices, Google introduced Google play protect in the market. Google play protect have the capability to protect data in real-time. However, according to a study in,[8] G-Data Security expert counted 4.18 millions malware applications until the end of the year 2019 and discovered over 7,50,000 new malware applications during the first quarter of 2020.

Android apps work on the permission-based model [15]. Android operating system provides protection at four levels, that categorize permissions as[9] "signature", "signature or system", "normal" and "dangerous". In our study, we do not consider "signature" and "signature or system" because they are system granted. We only consider "normal" and "dangerous" permissions which are granted by the user. Normal permissions does not pay any risk to the user's privacy. If the permission is listed in the manifest file, then system grants permission automatically. On the other hand, dangerous permission give access to the user's confidential data. However, it is purely dependent upon the user to give access or revoke the use of permission or set of permissions.

Selection of right feature or feature sets pay great effect on the performance of malware detection [45, 46, 49, 62]. Feature selection approach is based on the procedure to select appropriate features from total available features. Feature selection approaches are classified into two distinct groups i.e., one group contains feature ranking methods and second group contains feature subset selection methods [45, 46, 49, 62]. Feature ranking approach is based on ordering the feature on the basis of its scoring function [49, 62]. On the other hand, feature subset selection is to discover the optimal feature subset [46]. In our study, we implemented ten distinct feature selection approaches to select best features and hold only those feature sets which have excellent discriminatory power.

In the literature [45, 46, 49, 62], researchers and academicians had applied distinct machine learning algorithms that were based on classification, regression and clustering to develop Android malware detection model. The main flaw in their work is that they used labelled data set to develop malware detection model. So to overcome this issue, in this study, we consider five distinct unsupervised machine learning algorithms [i.e., K-mean, farthest first clustering , filtered clustering, density-based clustering and self-organizing map (SOM)] to develop a model for Android malware detection.

List of phases followed by us in developing malware detection model is demonstrated in Fig. 1. In the first stage, we collect Android application packages (.apk) files from different repositories and identify their classes. In the second stage of our experiment, we extract permissions and API calls from collected .apk files and consider them as features. Further in third stage, we select best features by using ten different feature selection approaches. Next, with the help of selected features we trained five different unsupervised machine learning algorithms and build models. We compare our developed models with the help of distinct performance parameters i.e., intra-cluster distance, inter-cluster distance, accuracy and F-measure. At the last stage, we validate our proposed model with the help of existing techniques available in the literature.

The novel and unique contribution of this research paper are:

- To the best of our knowledge, this is the first work in which 5,00,000 unique apps are collected which belongs to 30 different categories of Android apps. Extracted features are publicly available for researchers and academicians.[10] To build effective and efficient malware detection model we extract permissions, rating of an app, number of the user download the app, and API calls, as a feature and achieved a detection rate of 98.8% when compared to distinct anti-virus scanners.
- We proposed a new approach which works on the principle of unsupervised machine learning algorithm by selecting relevant features using feature selection. Our empirical result reveals that our suggested model is able to detect 98.4% unknown malware from real-world apps.

[7] https://source.android.com/security/reports/Google_Android_Security_2017_Report_Final.pdf.

[8] https://www.gdatasoftware.com/news/g-data-mobile-malware-report-2019-new-high-for-malicious-android-apps.

[9] https://developer.android.com/training/permissions/requesting.html.

[10] Mahindru, Arvind (2020), "Android permissions dataset, Android Malware and benign Application Data set (consist of permissions and API calls)", Mendeley Data, V3, doi: 10.17632/b4mxg7ydb7.3.

- Proposed framework is able to detect malware from Android apps by using 100% unlabelled data set.
- In this study, we applied *t* test analysis to investigate that features selected by feature selection approaches are having significant difference or not.
- Proposed malware detection approach is able to detect malware in less time when compared to distinct anti-virus scanners available in the market.

The rest of the paper is summarized as follows. In Sect. 2, we discuss about the work related to Android malware detection. In Sect. 3, we discuss about the Android permission model. In Sect. 4, we present the formulation of data set. Section 5 presents the features selection approaches implemented in this study. In Sect. 6, we discuss about the different machine learning algorithms. In Sect. 7, we present the different techniques which are used in the literature to detect malware from real-world apps. Section 8 represent the performance parameters and experimental setup is presented in Sect. 9. Section 10, contains the experimental results i.e., which model is best in detecting malware from real-world apps. At last in Sect. 11, we discuss about the threats to validity and conclusion of this empirical study is presented in Sect. 12.

2 Related work

Enck et al. [28] proposed Kirin framework which helps in detecting malware apps based on permissions requested by them during their installation time. Kirin is based on set of rules which helps to mitigate the effect of malware from Android apps. Suarez-Tangil et al. [65] examined out of cloud based detection or on-device detection method, which method is more power saving. They suggested a power model to compare both the methods with the help of machine learning algorithms. Empirical results reveals that cloud based detection method is more effective and better choice to detect malware. Cui et al. [26] proposed a malware detection model based on cloud computing by using network packets. They used the principles of data mining to reduce the branches of packets by gathering knowledge of packets whether it is useful for malware detection or not. They proposed SMMDS in their study which work on the principles of machine learning algorithms to detect malware. Chen et al. [23] proposed a solution which monitor the behavior of smartphones when they are sending user's private information to an external source. But the solution provided in their study is not effective, because it cannot support real-time detection. Narudin et al. [53] proposed STREAM which automatically installs and runs Android apps and extract features from them. Further, the extracted features are used to train with the help of machine learning classifiers to detect malware from Android apps. STREAM has a disadvantage, it takes a lot of system resources and time to load the data. Wei et al. [73] build a malware detection model based on anomaly behavior of Android apps. They developed a model by considering network information as a feature by using Naïve Bayes and Logistic machine learning algorithms and achieved higher accuracy rate. Ali et al. [11] suggested a malware detection model based on Gaussian mixture. They collected features based on hardware utilization such as CPU, memory, battery and so on and trained it with the help of Gaussian mixture. But the model proposed by them has a limitation, it needs a remote server for computation. Dixon et al. [27] developed a model by using the behaviors of battery life of smartphone when infected by malware. But, the model proposed by them is not able to detect some sophisticated malware.

Tong and Yan [67] proposed hybrid approach to detect malware from Android by using individual system call and sequential system calls related to accessing the files and networks. Their approach is able to detect the behavior of unknown app and achieved the detection rate of 91.76%. But, the presented approach has a limitation, it cannot support real-time detection. Quan et al. [59] used three different feature sets i.e., native code, system calls and API calls to detect malware from Android. The detection rate depends upon the predefined threshold value. Ng et al. [54] developed model by using Dendritic Cell Algorithm and considered system call as a feature. They selected best features by implementing statistical methods and achieved the higher detection rate. Sheen et al. [63] proposed Android malware detection system by considering API calls and permissions as features. They chose features by using Relief algorithm to train three different classifiers: J48, SVM and Naïve Bayes. Detection rate is good, but it also consumes number of resources and its computing burden is too high. Fung et al. [31] proposed a decision model RevMatch which work on the principle of malware detection history to make decision that Android app is infected with malware or not. This approach do not provide real-time detection. Babaagba and Adesanya [12] compared the performance of supervised and unsupervised machine learning algorithms, with and without using feature selection approaches. Empirical study were performed on 149 Android apps and result reveals that model developed with feature selection approach and supervised machine learning algorithm achieved higher detection rate when compared to the model developed using unsupervised machine learning algorithm. Yewale and Singh [78] proposed malware detection model based on opcode frequency. Experiments were performed on 100 distinct files and achieved 96.67% detection rate by using SVM as a machine learning algorithm.

Enck et al. [29] proposed TaintDroid which work on the principles of tracking information-flow in the network. TaintDroid track malicious behavior of apps communication through files, program variables and inter-sectional

messages. The process is too much time consuming to label that app is benign or malware. Abawajy and Kelarev [2] proposed ICFS which detect malware from Android by incorporating feature selection approaches and machine learning classifiers. Guo et al. [32] proposed smartphone network behavior using Naïve Bayes as machine learning algorithm. They build a pattern from benign and malware apps to discover malware from unlabeled apps. In recent study [68], Authors presented the mechanism that how an app breaching user privacy to gain user private data. They proposed a general and novel defence solution, to protect resources and data in Android based devices. Rahman and Saha [61] proposed StackDroid a multi-level architecture which is used to minimize the error rate. They detected malware at two different levels, in the first level they consider multi-;ayer perceptron, stochastic gradient descent, random forest and extremely randomized tree and in the second level they consider extreme gradient boosting as machine learning classifier to detect malware from Android. Barrera et al. [13] proposed a methodology which work on the principles of permission model by implementing self-organizing map (SOM) on collected data set of 1,100 Android apps. They analyzed the Android permission model which is used to investigate the malware apps from Android. SOM implemented by them, give a 2-dimensional visualization of high dimension data.

Alazab et al. [4] proposed an effective classification model that combines permission requests and API calls. Further, API calls were divided into three different groups i.e., ambiguous group, risky group, and disruptive group. Experiments were performed on 27,891 malware-infected Android apps and proposed model achieved an F-measure of 94.3%. Xiao et al. [75], proposed a novel detection approach based on the principle of deep learning. In their studies, authors consider semantic information in system call sequences as the natural language and construct a classifier based on the long short-term memory (LSTM) language model. Empirical result reveals that proposed approach achieved the detection rate of 96.6%. Yuxin and Siyi [79] proposed a malware detection model based on the principle of Deep Belief Network (DBN). In their study, they compare the performance of proposed model with three baseline malware detection models, which use support vector machines, decision trees, and the k-nearest neighbor algorithm as classifiers. Experimental results indicate that the autoencoder can effectively model the underlying structure of input data and significantly reduce the dimensions of feature vectors.

Vinayakumar et al. [69] proposed an effective zero-day malware detection model based on the principle of image processing technique with optimal parameters for machine learning algorithms (MLAs) and deep learning architectures. Experiments were performed on two distinct data sets and they achieved a detection rate of 96.2% and 98.8% with proposed detection model. Arora et al. [9] proposed a framework named as PermPair, that constructs and compares the graphs for malware and normal samples by extracting the permission pairs from the manifest file of an app. Empirical result reveals that proposed scheme is successful in detecting malicious samples with an accuracy of 95.44% when compared to mobile anti-malware apps. Lee et al. [42] proposed a malware detection model that learns the generalized correlation between obfuscated string patterns from an application's package name and the certificate owner name. Experimental results reveal that proposed model is robust in obfuscation and sufficient lightweight for Android devices.

Alzaylaee et al. [6] proposed DL-Droid, based on the principle of deep learning model. Experiments were performed on 30,000 distinct Android apps. Empirical result reveals that DL-Droid can achieve up to 97.8% detection rate (with dynamic features only) and 99.6% detection rate (with dynamic + static features) respectively. Ma et al. [44] proposed a malware detection approach based on the control flow graph of the app to obtain API information. On the basis of API information, they constructed three different data sets that is based on Boolean, frequency, and time-series. By using these three data sets three different malware detection models are constructed. Experiments were conducted on 10010 benign applications and 10683 malicious applications. The result shows that detection model achieves 98.98% detection precision. Jerbi et al. [36] proposed artificial malware-based detection (AMD) that was based on extracted malware patterns that were generated artificially. Experiments were performed on balanced and imbalanced data set and achieved an accuracy of 99.69% for balanced data set and 99.64% for imbalanced data set. Table 1 described the name of the framework or author name whose proposed the approach developed in the literature with its detection type, feature used, implemented algorithm, place of analysis, and major observations in their study.

Previous research work mentioned above has the following limitations: use of limited data set, higher detection rate with limited data set, high computation burden, implementation of limited feature selection approaches, implementation of limited classification algorithms using 100% labelled data set and unable to detect sophisticated malware. To overcome the first limitations, in this study, we collect 5,00,000 Android apps which belong to thirty different categories from different repositories mentioned in Sect. 4. Further, to overcome other limitations, we implement ten distinct feature selection approaches on extracted feature data set (i.e., permissions and API calls consider as feature in this study). Next, the selected features are considered as an input to develop a model by using unsupervised machine learning algorithms (means no labelled data is required to develop the models) so that a suitable model is build to identify malware from real-world apps.

Table 1 Dynamic analysis based smartphone detection presented in literature

Author/framework	Detection type	Feature used	Implemented algo.	Place of analysis	Observations
Yang et al. [77]	Signature based	Permissions	Pattern matching	On device	High memory consumption, not real-time and unable to protect privacy
Wei et al. [73]	Anomaly based	Network behavior	Naïve Bayes and logistic	Server	Better performance
Dixon et al. [27]	Anomaly based	Battery	Pattern matching	Server	Unable to detect sophisticated malware
Rahman [60]	Specification based	API	Markov logic network	Server	Low detection rate
Alam and Vuong [3]	Signature	Battery permissions and CPU	Random forest	Server	Data not collected on mobile devices
Amos et al. [7]	Anomaly based	Battery permissions and CPU	Random forest, J48 and logistic	Server	Detection is not real and performance is not so good
Shen et al. [64]	Signature	Topology	VF2 algorithm graphs	Server	Achieved low accuracy
Ng et al. [54]	Anomaly based	System calls	Dendritic cell algorithm	Server	Used three different classifiers with complex classification
Quan et al. [59]	Signature based	API and system calls	Pattern matching	Server	High detection accuracy and unable to list all sensitive behaviors
Sheen et al. [63]	Specification based	Permissions and API call	J48,NB and SVM	Server	Good performance but heavy computation burden
Caviglione et al. [21]	Specification based	Convert channel	NN and DT	Server	Feature not able to detect all malware
Almin and Chatterjee [5]	Signature based	Permissions	K-mean and NB	On-device and server	Not able to detect malware from unknown family
Holland et al. [34]	Signature based	API and permissions	Pattern matching	Server	Testing data-set is small and performance is not so good
Andriatsimandefitra and Tong [8]	Specification based	Information-flow	N/A	On-device and server	Data set is too small
Tong and Yan [67]	Anomaly based	System call	Pattern matching	Server	Not real and high power consumption
Martinelli et al. [51]	Signature	System calls	SVM	On-device	High detection but consume many local resouces
HinDroid [35]	Specification based	API call	Multi-kernel learning	Off-device	Limited data set used
DroidCat [19]	Specification based	Interprocess communication	Supervised learning	Off-device	34,343 apps were used
MalDozer [39]	Anomaly based	API calls	Neural network	Off-device	Limited data set were used
DroidDet [82]	Static	API calls and permissions	Random forest	Off-device	2130 Android apps were considered.
DeepDroid [45]	Anomaly based	API calls and permissions	Deep neural network	Off-device	1,20,000 Android apps were used
PerbDroid [49]	Anomaly based	System call and permissions	SVM, Deep neural network	Off-device	2,00,000 distinct Android apps utilized
GAdroid [48]	Anomaly based	System call and permissions	Deep neural network	Off-device	Consider 25,000 malware-infected apps

2.1 Research questions

To develop a malware detection model for Android with better detection rate and to cover the gaps that are present in the literature, we consider the following research questions in this research paper:

RQ1. Which malware detection model is most appropriate to detect malware from real-world apps?

This question helps in finding the most appropriate model which is suitable for malware detection in Android. In this work, we build 50 distinct models by considering ten distinct feature selection approaches and five different machine learning techniques. Further, to identify that which model is more appropriate for malware detection we consider four distinct performance parameters in our study.

RQ2. Whether the presented malware detection framework is effective or not to detect malware from Android devices?

The goal of this question is to investigate the performance of our malware detection approach. For this, we compare the performance of our developed model with some existing techniques available in the literature.

RQ3. Does a subset of feature perform better than all the extracted features for the task of detecting an app is malicious or not?

The aim of this question, is to evaluate the features and investigate their relationship among benign and malware apps. Distinct kinds of feature reduction approaches are being considered for finding subset of features which are able to detect either the app is malicious or not.

RQ4. Among different implemented feature ranking approaches which approach work best for the task to detect that either an Android app belong to benign class or malware class?

In feature ranking approach, efficiency of the machine learning algorithms is affected by the characteristics and nature of the malware data set. Distinct approaches are being implemented with various criterions to rank the collected feature sets. Four distinct performance criterions i.e., intra-cluster, inter-cluster, F-measure and accuracy are considered in this study, to compare distinct feature-ranking approaches.

RQ5. Among applied feature subset selection approaches, which approach performs foremost for the task of detecting malware from Android apps?

To determine the subset of features which are appropriate to detect either the Android app is benign or malware we consider feature subset selection approaches. In this work, we compare distinct approaches by using four performance criterions i.e., intra-cluster, inter-cluster, F-measure and accuracy.

RQ6. How do the feature subset selection approaches are compared with feature ranking approaches?

In this paper, pair-wise t test being used to determine either the feature subset selection approaches are appropriate than feature ranking approaches or both of them behave equally well.

RQ7. Do the feature selection approaches effect the outcome of the unsupervised machine learning approaches?

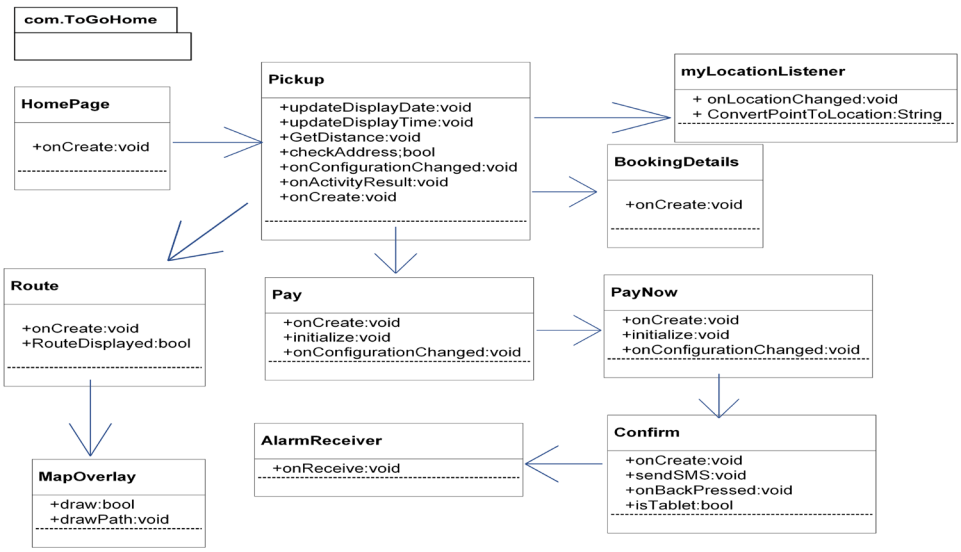
It is seen that number of feature selection approaches perform extremely well with specific unsupervised machine learning methods. Therefore, in this research work distinct feature selection approaches are evaluated using distinct unsupervised machine learning approaches to measure their performance. Further, it also emphasis on variation of performance of unsupervised machine learning approaches over distinct unsupervised machine learning approaches.

3 Android permission model

Android security is dependent upon the permission based model that access to functionality or features which could pay impact on the user's privacy. Android apps are written in java language and run in a Dalvik virtual machine. Android apps demand permissions during their installation and runtime from the users. Permissions such as send messages or making phone calls, access to vibrator or device screen on are asked by an app. User can grant or reject the request of permission made by an app. This facility of granting or revoking the permission is available from Android 6.0 version onwards. Currently in Android, there are more than 110 different features which are gaining access by using these permissions. Android is designed in such a way, that if any third-party app demand any new functionality, then it give privilege to the developer to define new permission under developer-defined permission or self-defined permission. By taking the advantage of this, cyber-criminals define new permissions on regular basis so that they can access the user's private data for their own benefits.

For an in-depth analysis of permission model, we build an app named as "ToGoHome".[11] Figure 2 represent the class diagram of Android app by using the structure showing its classes, attributes and methods involved in it. The package contains 10 distinct classes which contain the logic of cab reservation. In the very first step, when user click on the "HomePage" menu and searching for the cab it direct toward the class name "Pickup" that was collected from the current location of the user by using Global Positioning System (GPS) and call the class name "myLocationListener". Further "Pickup" class is divided into seven different attributes i.e., updateDisplayDate:void, `updateDisplayTime:void`, `GetDistance:void`,

[11] Testing were performed on local system.



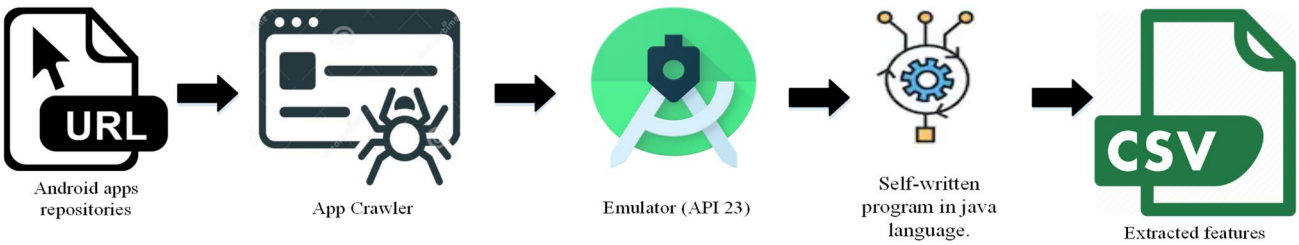


Fig. 2 Class diagram of com. ToGoHome Android app

Fig. 3 Extraction of features from .apk files

checkAddress;bool, onConfigurationChanged:void, onActivityResult:void and onCreate:void. If user confirm his pick up location through GPS then its booking will be confirmed by calling "BookingDetails". While travelling, user track his live location by calling "Route" class.[12] After completing the journey, our app call the "Pay" class and it will direct towards the payment website. Through out this process following Android apps permissions are used i.e., ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, ACCESS_NETWORK_STATE, INTERNET, SEND_SMS, and RECEIVE_SMS.

4 Formulation of data set

Figure 3 demonstrates the phases which are followed in extracting features from Android apps. In the very first phase, we identify the URLs from which Android apps are to be collected (mentioned in Sect. 4.1). In the second phase, we take the help of an app crawler to download the apps from identified URLs. Our developed app crawler can download as many apps as possible and do not pay any impact on the android app repository. To perform dynamic analysis of the collected Android apps, we use Android studio as an emulator (mentioned in Sect. 4.2). Further, we write a program in java language and extract permissions and API calls from them and save into the .csv for developing Android malware detection model. Extracted features are publicly available for researchers and academicians.[13].

4.1 Collection of .apk files

Pervious studies mentioned in Table 1, used only limited data sets of Android apps to examine its associations with malware or benign class and in addition to that in the literature [72, 80] academicians and researchers were not

[12] Live location of user is seen on Google Maps. Google Maps are pre-installed on Android based devices.

[13] Mahindru, Arvind (2020), "Android permissions dataset, Android Malware and benign Application Data set (consist of permissions and API calls)", Mendeley Data, V3, doi: http://dx.doi.org/10.17632/b4mxg7ydb7.3

Table 2 Categories of .apk files belong to their respective classes (*.apk*)

ID	Category	Normal	Trojan	Backdoor	Worms	Botnet	Spyware
D1	Arcade and action (AA)	16291	1440	100	204	130	600
D2	Books and reference (BR)	15235	2000	250	56	150	150
D3	Brain and puzzle (BP)	14928	1820	54	28	50	50
D4	Business (BU)	18308	1520	150	150	22	22
D5	Cards and casino (CC)	12886	760	65	81	100	44
D6	Casual (CA)	12010	3210	69	46	150	140
D7	Comics (CO)	17667	650	95	35	3	0
D8	Communication (COM)	18414	2500	50	500	3	3
D9	Education (ED)	18744	5600	68	50	50	68
D10	Entertainment (EN)	14222	5000	500	500	100	42
D11	Finance (FI)	13999	500	200	99	65	92
D12	Health and fitness (HF)	18551	98	65	45	140	140
D13	Libraries and demo (LD)	15655	70	100	100	6	500
D14	Lifestyle (LS)	17650	155	200	100	193	192
D15	Media and video (MV)	18019	100	123	162	450	71
D16	Medical (ME)	16000	12	13	12	24	25
D17	Music and audio (MA)	27057	65	100	65	165	165
D18	News and magazines (NM)	18164	100	100	100	100	32
D19	Personalization (PE)	14334	500	42	500	200	22
D20	Photography (PH)	19133	100	120	50	96	500
D21	Productivity (PR)	19850	100	516	250	250	62
D22	Racing (RA)	17766	50	100	210	100	180
D23	Shopping (SH)	12673	100	100	120	150	50
D24	Social (SO)	26159	100	50	210	150	150
D25	Sports (SP)	22669	100	240	100	450	112
D26	Sports games (SG)	13889	100	145	145	650	198
D27	Tools (TO)	13346	120	500	550	475	563
D28	Transportation (TR)	13796	2	500	100	100	20
D29	Travel and local (TL)	23180	500	220	150	48	100
D30	Weather (WR)	12841	120	23	700	50	25

mentioned the categories of the app to which it belongs. Therefore, it is not able to draw generic conclusion relevant to all Android apps and its system. To overcome this gap, we collect apps of thirty different categories which are used to generalize and strengthen our outcomes. We collect the Android apps to build our data set from promise repository. We collect 5,50,000 of .apk files, from Google's play store,[14] hiapk,[15] appchina,[16] Android,[17] mumayi,[18] gfan,[19] slideme,[20] and pandaapp.[21] Among these 5,50,000 benign .apk files, 5,00,000 are distinct. Further, the features are extracted after deleting viruses infected apps, reported by VirusTotal[22] and Microsoft Windows Defender.[23] VirusTotal identify malware affected apps by using 70 different antivirus softwares simultaneously. A total of 55,000 malware samples, are collected from three different promised repositories. 1929, botnet samples were collected from [38], which further consist of 14 distinct botnet families. Android Malware Genome project [80] contains a data set of 1200 malware samples that cover the currently present Android malware families. We

[14] https://play.google.com/store?hl=en.

[15] http://apk.hiapk.com/.

[16] http://www.appchina.com/.

[17] http://android.d.cn/.

[18] http://www.mumayi.com/.

[19] http://apk.gfan.com/.

[20] http://slideme.org/.

[21] http://download.pandaapp.com/?app=soft&controller=android#.V-p3f4h97IU.

[22] https://www.virustotal.com/.

[23] https://www.microsoft.com/en-in/windows/comprehensive-security.

Table 3 Formulation of sets containing (permissions, API calls, number of users download an app, and rating of an app) as features

Set number	Description	Set number	Description
S1	Related to SYNCHRONIZATION _DATA	S2	Related to CONTACT_INFORMATION
S3	Related to PHONE_STATE and PHONE_CONNECTION	S4	Related to AUDIO and VIDEO
S5	Related to SYSTEM_SETTINGS	S6	Related to BROWSER_INFORMATION
S7	Related to BUNDLE	S8	Related to LOG_FILE
S9	Related to LOCATION_INFORMATION	S10	Related to WIDGET
S11	Related to CALENDAR_INFORMATION	S12	Related to ACCOUNT_SETTINGS
S13	Related to DATABASE_INFORMATION	S14	Related to IMAGE
S15	Related to UNIQUE_IDENTIFIER	S16	Related to FILE_INFORMATION
S17	Related to SMS_MMS	S18	Related to READ
S19	Related to ACCESS_ACTION	S20	Related to READ_AND_WRITE
S21	Related to YOUR_ACCOUNTS	S22	Related to STORAGE_FILE
S23	Related to SERVICES_THAT_COST_YOU_MONEY	S24	Related to PHONE_CALLS
S25	Related to SYSTEM_TOOLS	S26	Related to NETWORK_INFORMATION and BLUETOOTH_INFORMATION
S27	Related to HARDWARE_CONTROLS	S28	Related to Default group
S29	Contain info. Related to API calls	S30	Contain info. Related to rating and downloads

collected about 56,871 samples from AndroMalShare[24] along with their package names. After removing duplicate packages from the collected data set, we have 50,000 unique malware samples left in our study. Both benign and malware apps being collected from the above mentioned sources at the end of December 2018. Table 2 shows the number of .apk files belonging to different categories i.e., business, comics, communication, education and so on. To better differentiate between benign and malware apps we consider .apk files belonging to normal, trojan, backdoor, worms, botnet and spyware families[25] are mentioned in Table 2.

4.2 Extraction of features

After collecting a unique samples of .apk files from various sources mentioned in previous subsection, we extract permissions and API calls from each of the .apk file. Extraction of permissions and API calls have been performed with the help of an emulator (in our study we use Android studio). Emulator provides the same API level and execution environment as our smartphones provide to us. In our study, to extract permissions and API calls from Android apps we use Android system version 6.0 Marshmallow (i.e., API level 23) and form our data set for experiments. Previous developed frameworks or approaches used the previous version of Android to extract features from them. There are two reasons for selecting this Android version: first, it asks the user to revoke or grant the permission to use the resources of smartphones and second it covers 28.1% of Android devices which is higher than other versions present in the market.[26] To extract features from collected .apk files, we execute each of them in an emulator and extract permissions by using self-written code in java from "AndroidManifest.xml". These permissions are demanded by apps during their installation and run-time. By using the same process again and again, we extract permissions from 5,00,000 different Android apps and record them in the .csv file format. This extracted data set listing the name of the permissions is publicly available for the researchers.[27,28] Previous researchers used limited set of features to develop a model for malware detection. To overcome this gap, in this study we collect 1532 permissions and 310 API calls which helps in building an effective and efficient Android malware detection model. Hence, each of the collected app can be represented as a 1842-dimensional Boolean vector, where "1" implies that the app requires the permission and "0" implies that the permission is not required. It is very common that distinct apps may request the similar set of permissions for its execution. Permissions overview given by Google[29] is used to describe the behavior of a permission i.e., "dangerous" or "normal".

[24] http://202.117.54.231:8080/.

[25] Malware families are identified by VirusTotal.

[26] https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/.

[27] https://data.mendeley.com/datasets/9b45k4hkdf/1.

[28] https://github.com/ArvindMahindru66/Computer-and-security-dataset.

[29] https://developer.android.com/guide/topics/permissions/overview.

4.3 Formulation of feature sets

Several approaches had been developed for Android malware detection [1, 13, 45, 46, 49, 50, 74]. In this study, we divide the extracted API calls and permissions in to thirty different feature sets which helps in developing malware detection model. Table 3 displays the basic descriptions of the feature sets which are considered in our work.

5 Feature selection approaches

On the basis of previous studies, it is seen that in previous studies a number of authors applied different feature ranking approaches to detect malware from Android apps and achieved good detection rate. This indicates that the outcome of malware detection model rely on the features that are taken as an input to design a model. Selecting the suitable feature sets is essential for data preprocessing task in machine learning. In the field of malware detection, some researchers have used selection approaches to select appropriate set of features. In this paper, we implemented ten distinct types of feature selection approaches on a large collection of 1842 features (divided in to thirty distinct feature sets) to identify the best subset of features which assist us to detect malware detection with better detection rate and also minimize the figure of misclassification errors. Feature ranking approaches and Feature subset selection approaches can be defined in the following manner [45, 46, 49, 62]:

- *Feature ranking approaches* These approaches, use certain conclusive elements to rank the features. Further, on the basis of their ranks appropriate features can be selected to build the model [49, 62].
- *Feature subset selection approaches* These approaches aim to search subset of features which can have good detective capability [4, 46].

5.1 Feature ranking approaches

These approaches rank features separately without applying any training algorithm. Ranking of features depends upon their score. On the basis of our investigation of the previous studies, the majority of approaches are capable to calculate the grading of every feature. In this research, we employ six different ranking approaches to rank the features. Various feature ranking approaches are explained below:

5.1.1 Gain-ratio feature selection

In this selection approach, feature ranking work on the prediction of the gain-ratio in relation to the class [49, 55]. The "Z" known as the gain-ratio of feature is determined as:

$$\text{Gain-Ratio} = \frac{Gain(Z)}{SplitInfo_Z(X)}, \tag{1}$$

where Gain $(Z) = I(X) - E(Z)$ and X depicts the set including m numbers of instances with n different classes. The forthcoming statistics necessary to categorize a given sample is calculated by utilizing succeeding equation:

$$I(X) = -\sum_{i=1}^{m} P_i \log_2(p_i). \tag{2}$$

Here in this equation P_i is the chance that a random sample can be a member of class C_i and is measured by z_i/z.

The number of instances are given by z_{ij} of class C_i in subset N_j. The foreseen knowledge rely on the partition of subsets by F, and is presented by

$$E(Z) = -\sum_{i=1}^{M} I(X)\frac{n_{1i} + n_{2i} + \cdots + n_{mi}}{n}. \tag{3}$$

$SplitInfo_F(X)$ is measured by utilizing following equation:

$$SplitInfo_F(X) = -\sum_{i=1}^{t} \frac{|X_i|}{X} \log_2(\frac{|X_i|}{X}) \tag{4}$$

The value of $SplitInfo_F(X)$ show us the details achieved by dividing the data set of training X into t portions equivalent to t results of a test on the attribute Z.

5.1.2 Chi-squared test

This test is employed to examine the self-determination among two events [49, 57], and in our work, ranking of features is predicted by the significance of its statistic in relation to the class. Higher the calculated value implies the denial of the outliers and consequently these features can be analyzed as better relevance to detect malware from Android apps. Chi-squared attribute evaluation evaluates the worth of a feature by computing the value of the chi-squared statistic with respect to the class. The initial hypothesis H_0 is the assumption that the two features are unrelated, and it is tested by chi-squared formula:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{5}$$

where O_{ij} is an observed frequency and E_{ij} is an expected (theoretical) frequency, asserted by the null hypothesis. The greater the value of χ^2 evidence against the hypothesis H_0.

5.1.3 Information-gain feature selection

In Info-gain, features are selected on its relation with respect to the class [49, 55]. In information-gain feature selection approach, entropy is considered as a criterion of

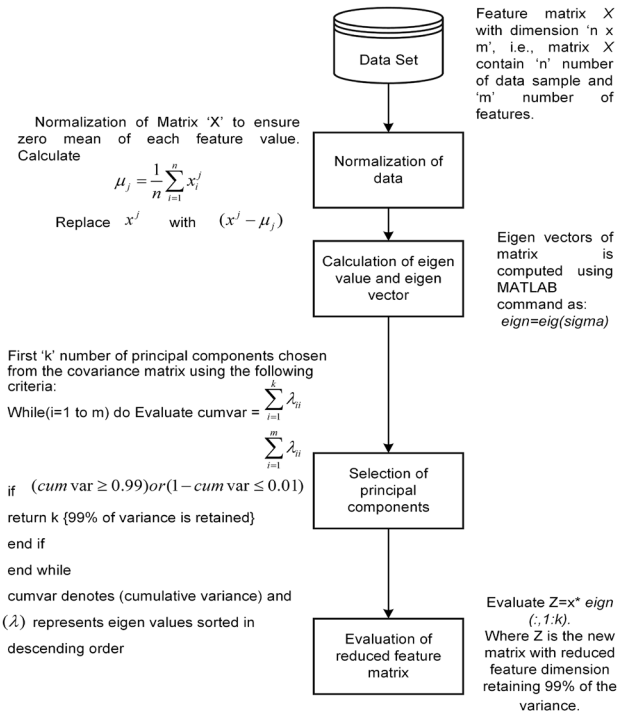


Fig. 4 Framework of PCA calculation

impurity in a training set S (data set), which can define a measure reflecting additional information about Y (random feature) provided by X (second random feature) that represents the amount by which the entropy of Y decreases. It is given by

$$IG = H(Y) - H(Y|X) = H(X) - H(X|Y) \tag{6}$$

The information gained about Y after observing X is equal to the information gained about X after observing Y. A weakness of the IG criterion is that it is biased in favor of features with more values even when they are not more informative [49, 55].

5.1.4 OneR feature selection

OneR feature selection approach is used for grading the features [49, 55]. To rank individual features it utilizes the classification mechanism. In it valuable features are considered as constant ones and divide the set of values into a few dissociate intervals made by straightforward method. In this study, we consider features with better classification rates.

5.1.5 Principal component analysis (PCA)

Reduction of attribute is accomplished by implementing PCA on our collected data set. PCA helps in transforming a high dimension data space into a low dimension data space. Features which are present in low dimension have extreme importance in detecting malware [49, 70]. Correlation among several features are high, so PCA is utilized to relocate these features that are not extremely correlated. The features obtained are named as principal component domain features. Further, to identify significant patterns in the data a small value of principal components is sufficient. The detailed phases of PCA are demonstrated in Fig. 4.

Feature data set is collected in the form of $m \times n$ matrix, that contains n number of data sample and m number of extracted features. In the second phase, normalization of the feature data set is performed by using equation

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_i^j$$

and replace x^j with $(x^j - \mu_j)$. Next, we calculate eigen value and eigen vector by using matlab environment. Next, to select first k number of principal components from the covariance matrix we performed following steps

while (i = 1 to m) do evaluate cumvar = $\sum_{i=1}^{k}\lambda_{ii}$

$$\sum_{i=1}^{m}\lambda_{ii}$$

if $(cumvar \geq 0.99)\, or\, (1 - cumvar \leq 0.01)$

return k 99% of variance is retained

end if

end while

cumvar denotes (cumulative variance) and (λ) represents eigen values sorted in descending order.

After evaluating this, reduced feature sets are selected for training purpose.

5.1.6 Logistic regression analysis

For feature ranking, univariate logistic regression (ULR) analysis is being considered to verify the degree of importance for every feature sets [25, 49]. In the current work, we consider two benchmarks of LR model which consider to discover the importance of every feature and also used to rank each feature sets. Parameters for Logistic regression analysis are as follows:

1. *Value of regression coefficient* The coefficient measure of features indicates the degree of correlation of every feature sets with malware.
2. *p value* p value i.e., level of significance shows the correlation significance.

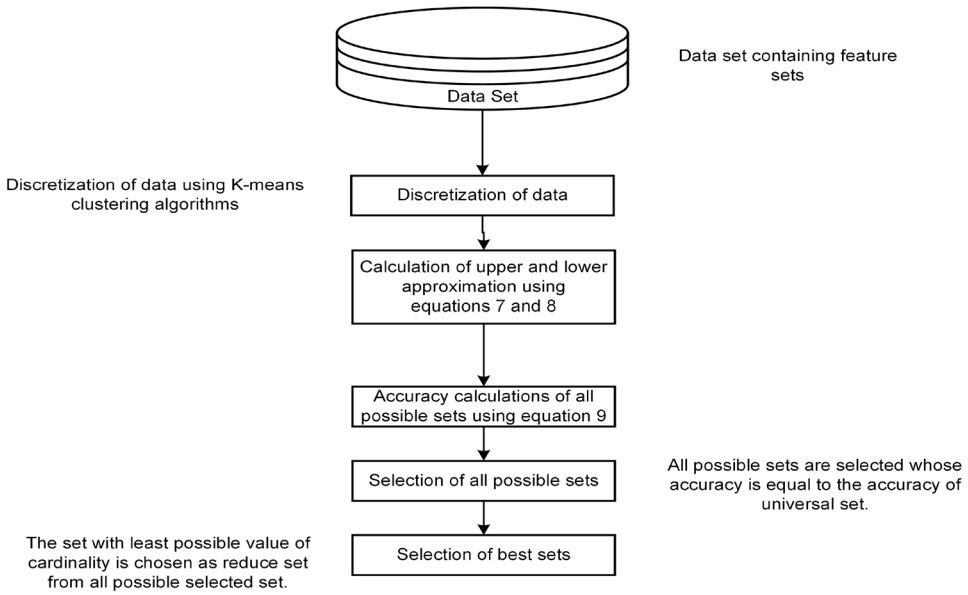


Fig. 5 Rough set theory framework

5.2 Feature subset selection approaches

These approaches are employed to detect appropriate subset of features which jointly have best detective capability. These are established on the hypothesis that developed model has better detection rate and lower value of misclassification errors when linked with few other features or when matched by itself. Several approaches are feasible to identify the right subset of features which helps in detecting malware. In this work, four distinct feature subset selection approaches are considered to calculate the score of feature. Implemented approaches are depicted below:

5.2.1 Correlation based feature selection

This approach is based on correlation approach which select a subset of features that are particularly related to the class (i.e., benign or malware). In this research paper, Pearson's correlation (r: Coefficient of correlation) has been used for searching the dependency among features. If the value of "r" is higher among the feature sets, it indicates a strong relation among these features. It implies that, there is a statistical reason to consider those classes which are having lower (or highest) feature value with that have lower (or highest) ranges of other highly correlated features [49].

5.2.2 Rough set analysis

This approach is an estimation of conventional set, in terms of a joins of feature sets which provide the upper and the lower estimation of the original data set [46, 56]. This formal estimation, depicts the upper and lower limits of the original data set. The application of this approach is in mining the data from imperfect data. This approach is used to select the reduced set of features from the extracted feature sets. RSA used three distinct notations such as approximations, reduced attributes and information system. The steps that are pursued to get reduced subset by utilizing RSA are mentioned-below and also demonstrated in Fig. 5.

(i) *Approximation* Let $A = (C, Z), X \subseteq Z$ and $Y \subseteq C$. $X-$ topmost (XY) and $X-$ lowermost $(< uline > X < /uline > Y)$ approximations of X are utilized to estimate Y. The topmost limit includes all the objects which maybe the part to the set and the lowermost approximation includes of all objects which certainly be a part of the set. The XY and $(< uline > X < /uline > Y)$ are computed by utilizing subsequent equations:

$$\bar{X}Y = \{y_i \in U \mid [y_i]_{Ind(B)} \cap Y \neq 0\} \tag{7}$$

$$\underline{X}Y = \{y_i \in U \mid [y_i]_{Ind(B) \cap Y}\}, \tag{8}$$

where $\mid [y_i]_{Ind(C)}$ belongs to the same class of y_i in connection $Ind(C)$.

(ii) *Reduced attributes* Correctness evaluation of the group Z $(Acc(Z))$ in $A \subseteq B$ is determined as:

$$\mu_B(A) = \frac{card(\underline{B}Z)}{card(\bar{B}Z)} \tag{9}$$

The number of features contained in the topmost or uppermost approximation of the set Z is called the cardinality of the set. Further, all possible feature sets are considered whose accuracy is equivalent to the accuracy of extracted feature sets.

(iii) *Information system* It is determined as $Z = (C, B)$, where C is a universe including non-empty set of confined

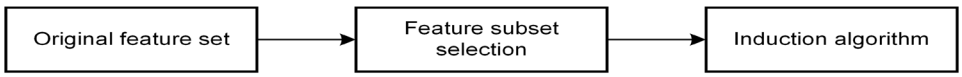


Fig. 6 Feature selection by utilizing filter approach

objects and B is the sets of attributes with a finite number of elements. For each $b \in B$, there exist a function $F_b : C \to V_b$, where V_b denotes the value of attribute b. For each $A \subset B$, there exists an equivalence relation known as B-indiscerbility relation is ($Ind(Z)$). $Ind(Z)$ can be defined as:

$$IND_A(Z) = \{(x, y) \in C^2 \mid \forall a \in Z, a(x) = a(y)\}. \quad (10)$$

5.2.3 Consistency subset evaluation approach

This technique provides the importance of subset of attributes by their level of consistency appearing in class values, when the training instances are applied on the subset of attributes. The consistency rate is calculated with the help of inconsistency rate, where two data elements can be considered as inconsistent if they belong to different class labels (i.e., benign or malware) but have same feature values. For this work, destination variable i.e., apps having two distinct characteristics (i.e., 0 for benign apps and 1 for malware apps). A group of feature (GF) is having Z amount of sample, there are z amount of instances in a manner that $Z = X_1 + X_2 + \cdots + X_z$. Instance X_i seems in entirely A samples from which A_0 numbers of samples are marked by 0 and A_1 number of samples are marked by 1, here $A = A_0 + A_1$. If A_1 is less than A_0, then the difference count for the instance X_i is $INC = A - A_0$. The inconsistency rate ($INCR$) of feature set is computed by utilizing succeeding equation:

$$INCR = \frac{\sum_{i=1}^{z} INC_i}{Z} \quad (11)$$

5.3 Filtered subset evaluation

Filtered subset evaluation is based on the principle to select random subset evaluator from data set that was gained by applying arbitrary filtering approach [40]. The filtering technique is not based on any induction algorithm. Filtered subset evaluation technique is scalable and fast. Figure 6 demonstrates the steps followed to find subset of feature by utilizing filter method.

6 Machine learning techniques

Various authors applied a number of unsupervised machine learning classifiers like K-mean [18, 52] and Self-Organizing Maps (SOM) [18, 20] to detect Android malware. But, they applied clustering algorithm on limited data set. To overcome this gap, in this study, we implement five different clustering algorithms on our extracted data set i.e., SOM, K-mean, farthest first clustering, filtered clustering and density-based clustering. The choice of clustering algorithms are based on its performance in the literature [18, 20, 52, 74].

6.1 Self-organizing maps (SOM)

SOM is a type of artificial neural network (ANN) that is trained with the help of unsupervised learning to produce a low-dimensional, discretized representation of the input space of the training samples called map and is therefore a method to do dimensionality reduction.[30] SOM consists of neurons, which have the same dimensionality as the input space and is arranged in a rectangular or a hexagonal grid. SOM neurons can be studied as pointers in the input space, in which more neurons point to regions with high concentration of inputs [13]. The training algorithm can be summarized in four basic steps:

1. Initialize neuron weights $W_i = [w_{i1}, w_{i2}, \ldots, wij]^T \in R^j$.
2. Input pattern $x = [x_1, x_2, \ldots, x_j]^T \in R^j$. Input pattern corresponding to an app in which permissions are expressed in the form of a bit string. Calculate the distance between pattern x and each neuron weight W_R, Therefore, identify the winning neuron or best matching neuron Z as follows

$$\|x - W_z\| = min_R\{\|x - W_R\|\} \quad (12)$$

   In our study, we employed Euclidean distance as the distance metric, normalized to the range [0, 1].
3. Adjust the weights of winning neuron Z and all neighbor units

$$w_R(t+1)w_R(t) + h_{Z_R}(t)[x(t) - W_R(t)], \quad (13)$$

   where R is the index of the neighbor and t is an integer, the discrete time co-ordinate. Neighborhood kernel $h_{ZR}(t)$ is a function of time and the distance between neighbor neuron r and winning neuron z. $h_{RZ}(t)$ defines the region of influence that the input has on the SOM and consists of two components it means that the neighborhood function $h(\|.\|, t)$ and the learning rate function $\alpha(t)$ is

[30] https://towardsdatascience.com/self-organizing-maps-ff5853a118d4.

$$h_{ZR}(t) = h(\|b_r - b_z\|;t)\alpha(t) \tag{14}$$

where b is the location of the neurons.

In our study, we used Gaussian neighborhood function and the first form of the neighborhood kernel with Gaussian function is

$$h_{RZ}(t) = exp\Big(\frac{\|b_r - b_z\|^2}{2\sigma^2(t)}\Big)\alpha(t), \tag{15}$$

where $\sigma(t)$ defines the width of the kernel.

4. Repeat steps 2–3 until the convergence criterion is satisfied.

6.2 K-mean

K-mean adopt the method of vector quantization. K-mean clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.[31] The working of K-mean algorithm is as follows:

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the data set and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
   - Compute the sum of the squared distance between data points and all centroids.
   - Assign each data point to the closest cluster (centroid).
   - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

6.3 Farthest first

Farthest first is based on the principle of a bounded metric space in which first point is selected arbitrarily and each successive point is as far as possible from the set of previously-selected points.[32] The working of farthest first clustering is described below [41]:

For each $X_i = [x_i, 1, x_i, 2, ...x_i, m]$ in D that is described by m categorical attributes, we use $f(x_{i,j}|D)$ to denote the frequency count of attribute value $x_{i,j}$ in the data set. Then, a scoring function is designed for evaluating each point, which is defined as:

$$Score(X_i) = \sum_{j=1}^{m} f(x_{i,j}|D). \tag{16}$$

1. Farthest first traversal (D: data set, k: integer) {
2. randomly select first center;
3. select centers
4. for (I = 2,⋯,k) {
5. for (each remaining point) { calculate distance to the current center set; }
6. select the point with maximum distance as new center; }
7. assign remaining points
8. for (each remaining point) {
9. calculate the distance to each cluster center;
10. put it to the cluster with minimum distance; } }

6.4 Filtered cluster

Filtered cluster is a special subset of a partially ordered set, in which each cluster is labeled with an ID. If Cluster belong to the first class then it will be labeled with 1 otherwise it is labeled as 0. The working of filter clustering algorithm is described below [37]:

1. In only one scan of the data set derive the $\frac{l(l-1)}{2}$ contingency tables necessary for computing the previously presented adequacy indices
2. Run the Genetic Algorithm using the fitness function $fit(SA, SA_*)$[33]
   - a chromosome codes (corresponds to) a subset of SA;
   - each gene of the chromosome codes an attribute of SA (so, there are l genes);
   - each gene of a chromosome has a binary value: the gene value is 1 (resp. 0) if its associated attribute is present (resp. absent) in the subset of SA coded by the chromosome.
3. Select the best subspace found by the Genetic Algorithm.

6.5 Density-based cluster

Density-based cluster is based on the notion of density. It either grow clusters according to the density of

[31] https://en.wikipedia.org/wiki/K-means_clustering.

[32] https://en.wikipedia.org/wiki/Farthest-first_traversal.

[33] A data set DS composed by a set (O) of n objects described by a set (SA) of l attributes.

Table 4 Confusion matrix to classify a Android app is benign or malware (*.apk*)

	Benign	Malware
Benign	Benign-> Benign	Benign-> Malware
Malware	Malware-> Benign	Malware-> Malware

neighborhood objects or according to some density function. The working of density-based clustering is described below [24]:

1. initialize $t_c = 0$;
2. initialize an empty hash table *grid_list*[34];
3. **while** data stream is active **do**
4. read record $x = (x_1, x_2, upto, x_d)$;
5. determine the density grid g that contains x;
6. if(g not in *grid_list*) insert g to *grid_list*;
7. update the characteristic vector of g;
8. **if** $t_c == gap$ **then**
9. call *initial_clustering*(*grid_list*);
10. **end if**
11. **if** t_c mod $gap == 0$ **then**
12. detect and remove sporadic grids from *grid_list*;
13. call adjust clustering(*grid_list*);
14. **end if**
15. $t_c = t_c + 1$;
16. **end while**
17. **end procedure**

7 Comparison of proposed model with different existing techniques

To validate that our proposed framework is able to achieve higher detection rate or not, we compare the result of our proposed model with three different techniques mentioned below:

(a) *Comparison of results with previously used classifiers* To validate that our proposed model is feasible to detect malware as equivalent to previous used classifiers or not, we calculate two performance parameters like Accuracy and F-measure for new proposed model and existing models. In addition to that, in this study, we also compared our proposed framework with existing frameworks that are present in the literature.

(b) *Comparison of results with different anti-virus scanners* To compare the performance of our model for malware detection, we chose ten available distinct anti-virus scanners and compare their detection rate with the detection rate of the proposed model.

(c) *Detection of known and unknown malware families* Further, to evaluate how much our proposed model is reliable to detect known and unknown malware families, we test known and unknown malware families with our proposed model and calculate the accuracy to detect the malware.

8 Evaluation of performance parameters

In this section of the paper, we discuss the fundamental definitions of the performance parameters utilized by us while evaluating our proposed model for malware detection. Confusion matrix is used to calculate all these parameters. It consists of actual and detected classification information built by detection models. Table 4 demonstrates the confusion matrix for malware detection model. In the present work, four performance parameters namely, inter-cluster distance, intra-cluster distance, F-measure and accuracy are utilized for measuring the performance of malware detection approaches.

Inter-cluster distance For each of these techniques, we first calculate centroid and then centroid Euclidian distance. In case of N, d-dimensional data points in a cluster: $\{\mathbf{X}_i\}$ where $i = 1, 2, 3, \ldots N$, the centroid $\{\mathbf{C}_0\}$, as defined in[35] is given by

$$\mathbf{C_0} = \frac{\sum_{i=1}^{N} \mathbf{X_i}}{N}. \tag{17}$$

Next, we define centroid Euclidian distance between the centroid of two clusters. Given the centroid of two clusters C_{01} and C_{02}, the centroid Euclidian distance or inter-cluster distance between them is defined by

$$D_0 = ((C_{01} - C_{02})^2)^{\frac{1}{2}}. \tag{18}$$

Intra-cluster distance To calculate the intra-cluster distance, we find root-mean-square-total-sample standard deviation (RMSSTD). This is defined by

$$RMSSTD = \sqrt{\frac{\sum_{j=1}^{p} \bar{s_j}^2}{p}}, \tag{19}$$

where $\bar{s_j}$ denotes the standard deviation of the attributes and p is the number of features. The smaller the value, the more homogenous the observations are with respect to the variables and vice-versa. Since root-mean-square is scalable dependent, it should only be used to compare the

[34] *grid_list* consist of attributes.

[35] https://en.wikipedia.org/wiki/Euclidean_distance.

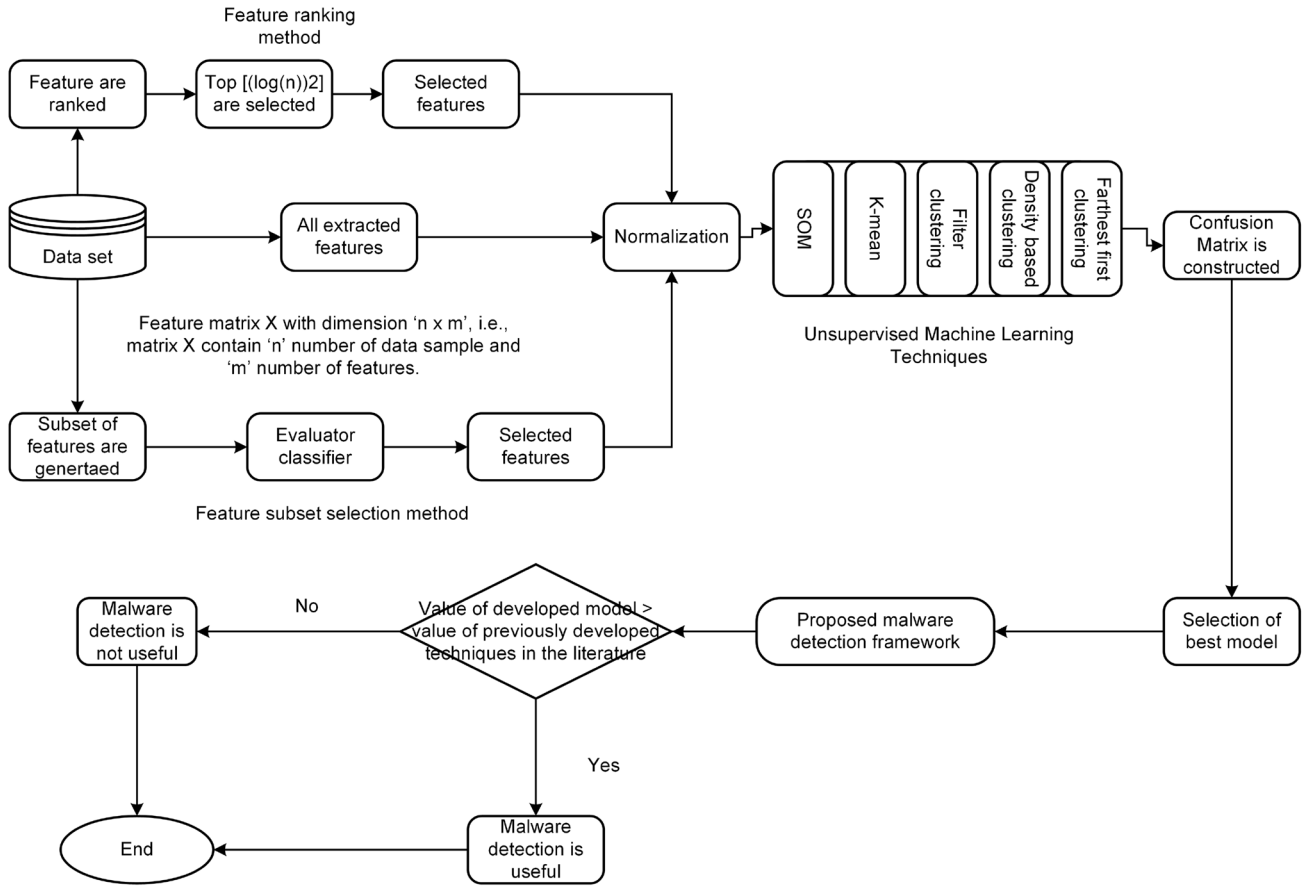


Fig. 7 Proposed framework i.e., SemiDroid

homogeneity of datasets whose variables are measured using similar scales.

F-measure The F-measure is harmonic mean of the precision and recall values used in information retrieval [16]. Precision shows how many applications are in the right cluster w.r.t. the cluster size. Recall shows that how many applications are in the right cluster w.r.t. the total applications. Let i denotes the class label and j denotes the cluster, then the Precision and recall for class i and cluster j are defined as:

$$Recall(i,j) = \frac{n_{i,j}}{n_j} \tag{20}$$

$$Precision(i,j) = \frac{n_{i,j}}{n_i} \tag{21}$$

where $n_{i,j}$ is the number of applications with class label i in cluster j, n_i is the number of applications with class label i and n_j denotes the number of applications in cluster j. The F-measure for class i and cluster j is given as:

$$F(i,j) = \frac{2 * Recall(i,j) * Precision(i,j)}{Recall(i,j) + Precision(i,j)}. \tag{22}$$

The total F-measure of clustering process is given by:

$$F = \sum \frac{n_i}{n} * maxF(i,j) \tag{23}$$

where n is the total number of applications.

Accuracy Let $n_{i,j}$ is the number of applications with class label i in cluster j, $n_{j,i}$ is the number of applications with class label j in cluster i, n_i is the number of applications with class label i and n_j denotes the number of applications in cluster j. Then Accuracy becomes:

$$Accuracy = \frac{x_{ij} + x_{ji}}{x_{ij} + x_{ji} + x_i + x_j}. \tag{24}$$

9 Experimental setup

In the present section, we introduce the experimental setup done to find the performance of our developed malware detection models. Five distinct unsupervised machine learning algorithms are implemented on thirty different categories of android apps mentioned in Table 2. All these

data sets have varying number of benign or malware apps which are adequate to perform our analysis. Figure 7 demonstrates the framework of our proposed model named as SemiDroid. In the very first step, feature ranking and feature subset selection approaches are applied on the extracted features data set. In the next step, we use the Min-max normalization approach to normalize the data. This approach is based on the principle of linear transformation, which bring each data point D_{q_i} of feature Q to a normalized value D_{q_i}, that lie in between $0-1$. Following equation is considered to find the normalized value of D_{q_i} :

$$Normalized(D_{q_i}) = \frac{D_{q_i} - min(Q)}{max(Q) - min(Q)},$$

where $min(Q)$ and $max(Q)$ are the minimum and maximum significance of attribute Q, respectively. In the third step, we trained significant features by implementing distinct machine learning techniques. In the next step, we construct a confusion matrix and calculate the performance parameters i.e., Accuracy and F-measure. Next, we compare the performance of the developed malware detection model and select the best malware detection model. At last, we compare the performance of our proposed malware detection model with existing techniques available in the literature and distinct anti-virus scanners. If the performance of our proposed malware detection model is better than existing techniques then it is useful and in reverse of it if the performance is not enhanced than the proposed malware model is not useful.

The subsequent measures are pursued at the time of selecting a subset of features to built the malware detection model that detects either an app is benign or malware. Feature selection approaches are implemented on 30 different data sets of Android apps. Hence, a total of 1650 ((1 selecting all extracted features + 10 feature selection approaches) × 30 data sets (subsets of different feature sets particular to data sets determined after conducting feature selection) × 5 detection methods) different detection models have been build in this work. Below we provide step by step details of our approach:

1. In the present work, four feature subset selection approaches and six feature ranking approaches are implemented on 30 different feature sets to select the right set of features for malware detection.
2. The subsets of features obtained from aforementioned procedure are given as an input to machine learning classifiers. To compare the developed models, we use 20 fold cross-validation method. Cross-validation is a statistical learning approach that is utilized to classify and match the models by dividing the data into two different portions [40]. One portion is utilized to train and the remaining portion of data is utilized to verify the build model, on the basis of training [40]. The data is initially separated into K same sized segments. K-1 folds are utilized to train the model and the rest one fold is utilized for testing intention. K-fold cross-validation is having important significance in utilizing the data set for the both testing and training. For this study, 20-fold cross-validation is utilized to analyze the models, i.e., data sets are segregated into 20 portions. The outcomes of all build malware detection models are matched with each other by employing two distinct performance measure parameters: F-measure and accuracy.
3. SemiDroid i.e., proposed model build by utilizing above two steps are validated with the existing techniques developed in the literature to review whether the build malware detection model is useful or not.

Table 5 Used naming convention in this study

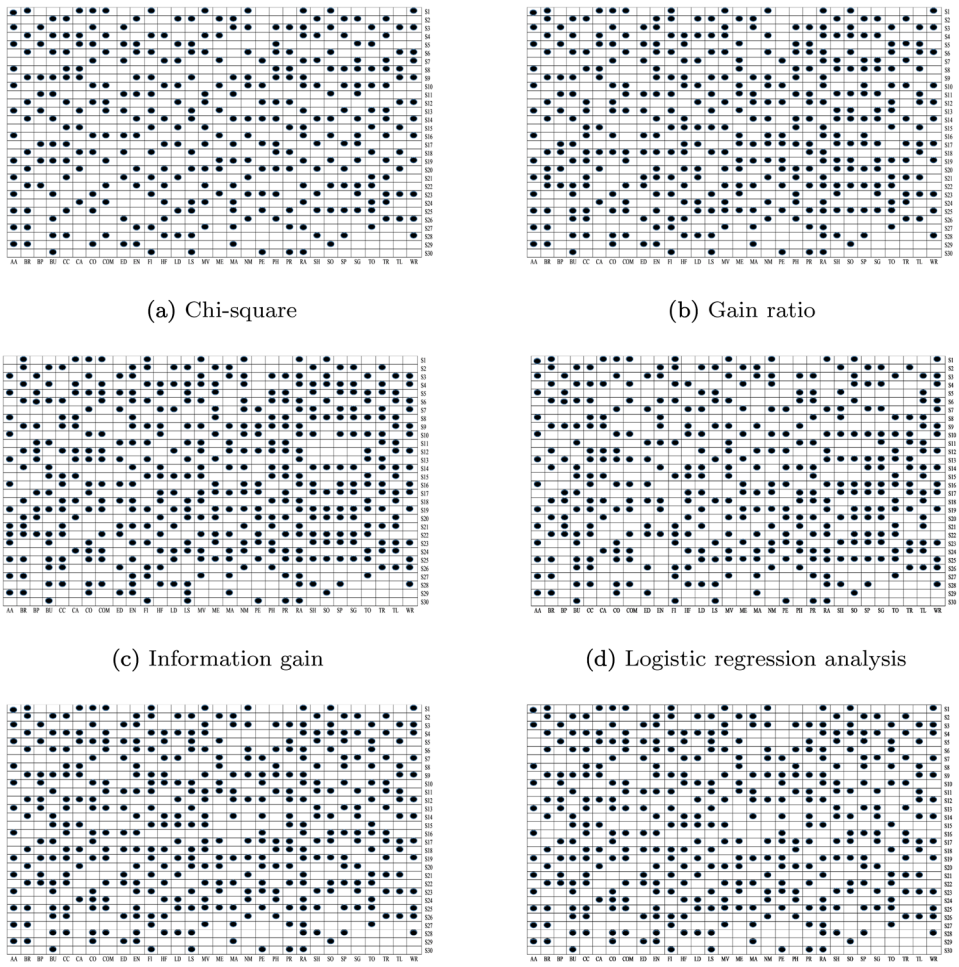
Abbreviation	Corresponding name
DS	Data set
FS1	Correlation best feature selection
FS2	Classifier subset evaluation
FS3	Filtered subset evaluation
FS4	Rough set analysis (RSA)
FR1	Chi squared test
FR2	Gain ratio feature evaluation
FR3	Filtered subset evaluation
FR4	Information gain feature evaluation
FR5	Logistic regression analysis
FR6	Principal component analysis (PCA)
AF	All extracted features

10 Results of performed experiment

In the current section of the paper, the relationship among different feature sets and malware detection at the class level is submitted. Set of features are used as an input and present the ratio of benign and malware apps within an experiment. intra-cluster distance, inter-cluster distance, F-measure and accuracy are used as performance assessment parameters to match the performance of malware detection model build by using five different unsupervised machine learning algorithms. To depict the experimental results we utilize the abbreviations as given in Table 5 corresponding to their actual names.

10.1 Feature ranking approaches

Six feature ranking approaches: gain-ratio feature evaluation, Chi-squared test, information gain feature evaluation, logistic regression analysis, information gain, oneR feature



(a) Chi-square (b) Gain ratio

(c) Information gain (d) Logistic regression analysis

(e) OneR (f) PCA

Fig. 8 Feature ranking approaches

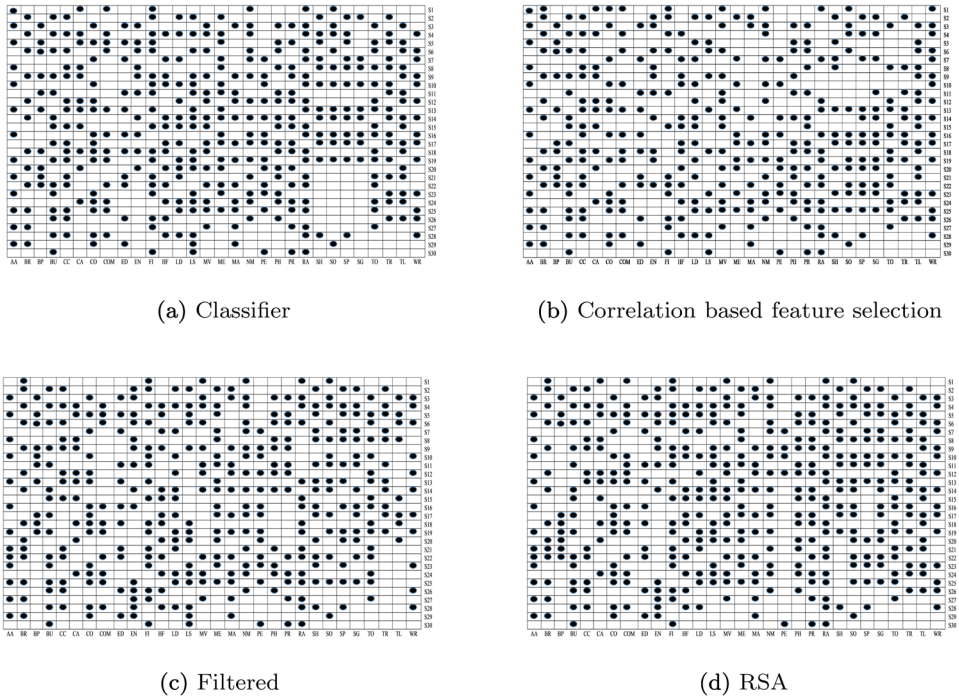
evaluation and principal component analysis are implemented on a distinct feature sets. Each approach utilizes distinct performance parameters to rank the feature. Moreover, top $\lceil \log_2 a \rceil$ set of features from "a" number of features being measured to build a model for detecting malware. For initial four feature ranking approaches (gain-ratio feature evaluation, Chi-squared test, OneR feature evaluation and Information gain), top $\lceil \log_2 a \rceil$ are selected as subset of features, where a is the number of features in the original data set (for this work $a = 20$). However, in the case of ULR, those features are selected which posses a positive value of regression co-efficient, i.e., p value measure is below 0.05, and in matter of PCA, only those features are selected which have eigenvalue greater than 1. Considered features using feature ranking approaches are demonstrated in Fig. 8.

10.2 Feature subset selection approaches

In the present work, four distinct kinds of feature subset selection approaches are implemented on thirty data sets of Android apps one after another. Feature subset selection approaches work on the principle of hypothesis which make models with better accuracy and make less amount of misclassified errors, while selecting the best features from available number of features. Searching principles are based on heuristic search [17] (for correlation based feature selection, rough set analysis), best first search [14] (for consistency subset evaluation and filtered subset evaluation). Later, these isolation subset of features has been selected as an input for building a model to detect either the app is benign or malware. Considered set of features after feature subset selection approaches are demonstrated in Fig. 9.

10.3 Machine learning techniques

Eleven subsets of features (1 considering all set of extracted features + 10 resulting by implemented feature selection approaches) are used as an input to build a model for malware detection. Hardware utilized to carry out this study is Core i7 processor having storage capacity of 1TB hard disk



(a) Classifier (b) Correlation based feature selection

(c) Filtered (d) RSA

Fig. 9 Feature subset selection approaches

and 8GB RAM. Detection models are build by using the MATLAB environment. Figure 6 demonstrates the implemented unsupervised machine learning algorithms on our collected data set. Red-cross represents the normal permissions and blue-cross represents the dangerous permissions. From Fig. 10, we analysis that clusters formed by using SOM, density-based clustering, K-mean, and filter clustering have overlapping of normal and dangerous permissions. Only farthest first clustering performed the cluster without overlapping of the permissions. Further, the performance of each detection model is measured by using four performance parameters: intra-cluster distance, inter-cluster distance, F-measure and accuracy.

Tables 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15, present the performance values obtained for distinct data sets by utilizing SOM, K-mean, farthest first clustering, filtered clustering and density based clustering. On the basis of Tables 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15, it may be concluded that:

- Bold values indicate the highest detection rate when compared to other values in a specific row.
- Value of inter-cluster and intra-cluster distance are calculated by using Eqs. 18 and 19.
- F-measure and accuracy are measured by using Eqs. 22 and 24.
- Models developed by considering features selected by feature selection approaches as an input is able to detect malware more effectively rather than model developed by using all extracted feature sets.
- Model constructed by considering FS4, as an input achieved higher detection rate when compared to other models developed by using different feature selection approaches.
- Model build by considering farthest first clustering by selecting FS4, as an input achieved higher detection rate when compared to other models developed by using different feature selection approaches.

In this research paper, five distinct unsupervised machine learning algorithms and ten distinct feature selection approaches are considered to select features which helps in identify Android malware more effectively. To find out which developed model is more capable to detect malware, we construct box-plot diagrams of the individual model. Box-plot diagrams helps in identify which model is best suitable for malware detection on the basis of few number of outliers and better value of median. Figures 11 and 12 demonstrate the box-plot diagrams for F-measure and accuracy for every developed model. The *x*-axis of the diagrams presents the feature selection techniques. Figures include eleven box-plot diagrams: one box-plot diagram consists of all extracted feature sets, four box-plot consist of feature subset selection approaches and six box-plot consist of feature ranking approaches. On the basis of the box-plot diagram, we find following observations:

- Model constructed by considering five distinct unsupervised machine learning algorithms and FS4 achieved

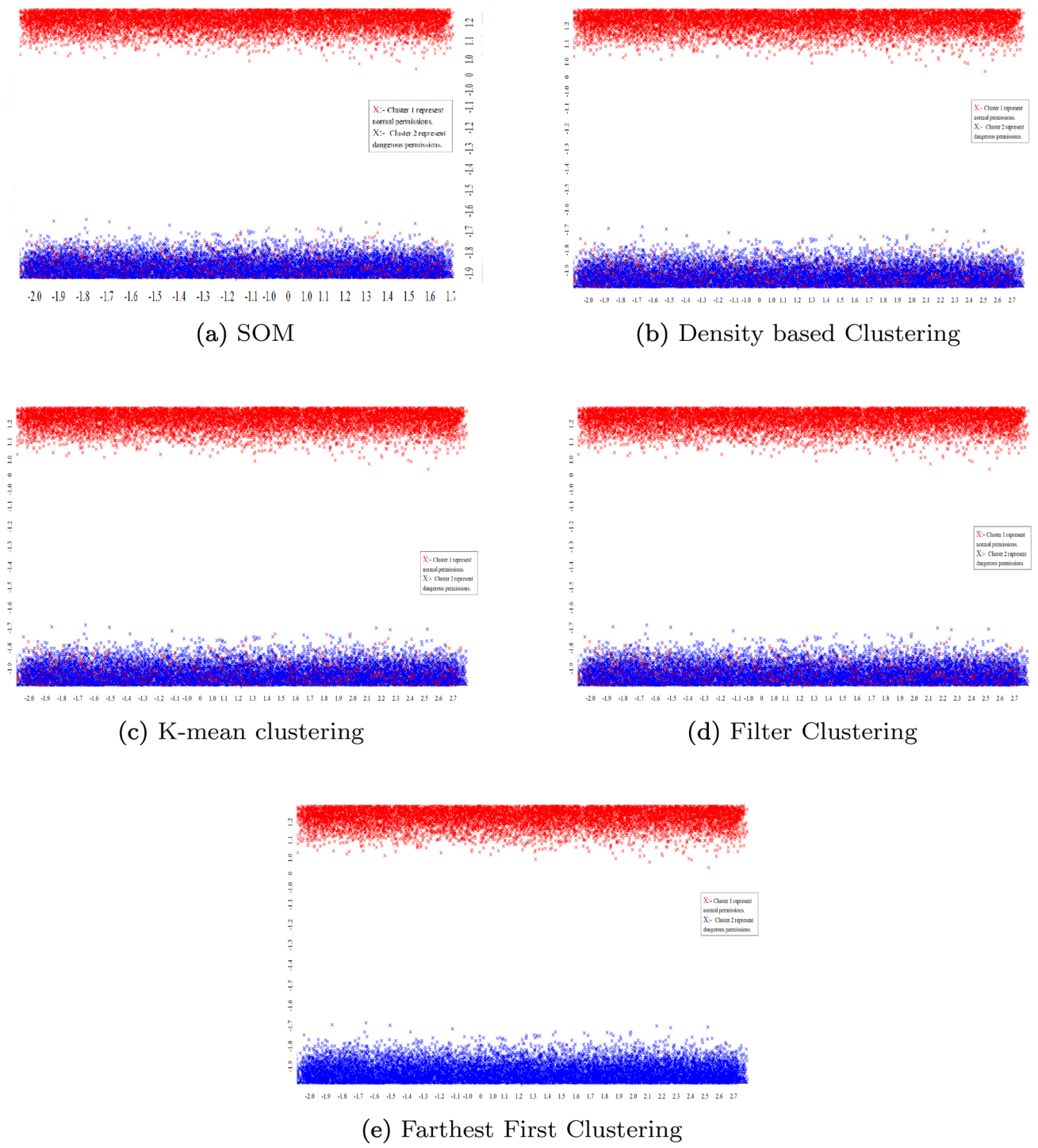


Fig. 10 Unsupervised machine learning algorithms

higher median value in addition to few outliers. On the basis of box-plot diagrams demonstrated in Figs. 11 and 12, model developed by considering FS4 as feature selection approach gives better detection rate when compared to other developed approaches.

– From box-plot diagrams, we observed that model build by considering farthest first machine learning algorithm and FS4, is having few outliers and higher median value. It means that model developed by using RSA for detecting malware and benign apps achieved better results when compared to others.

Table 6 Intra and inter cluster distance using SOM

Inter												Intra										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	0.268	0.381	0.388	0.40	0.42	0.42	0.43	0.45	0.47	0.477	**0.48**	− 1.1	− 2.0	− 2.8	− 2.1	− 3.1	− 3.1	− 3.8	− 3.2	− 3.0	− 3.8	− **3.9**
D2	0.28	0.38	0.39	0.40	0.43	0.44	0.43	0.46	**0.49**	0.48	0.48	− 1.8	− 2.7	− 2.9	− 2.0	− 2.8	− 2.1	− 2.7	− 2.2	− **3.0**	− 2.8	2.7
D3	0.28	0.31	0.328	0.34	0.41	0.34	0.40	0.40	0.42	0.42	**0.45**	− 1.1	− 2.0	− 2.8	− 2.1	− 2.1	− 2.1	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D4	0.20	0.22	0.23	0.34	0.34	0.31	0.33	0.35	0.37	0.37	**0.38**	− 1.1	− 1.0	− 1.8	− 2.1	− 2.1	− 2.1	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D5	0.22	0.32	0.38	0.34	0.32	0.32	0.34	0.35	0.37	0.37	**0.39**	− 1.1	− 2.8	− 2.8	− 2.1	− 2.5	− 2.2	− 2.8	− 2.2	− 3.0	− 3.8	− **3.9**
D6	0.299	0.381	0.388	0.41	0.40	0.45	0.43	0.45	0.45	0.46	**0.47**	− 1.0	− 1.3	− 1.8	− 1.6	− 1.8	− 1.6	− 1.8	− 1.2	− 1.0	− 1.8	− **1.9**
D7	0.21	0.31	0.36	0.34	0.32	0.33	0.35	0.35	0.37	0.37	**0.38**	− 1.6	− 2.2	− 2.8	− 2.1	− 2.1	− 2.1	− 2.8	− 2.2	− 3.0	− 2.8	− **3.1**
D8	0.25	0.37	0.38	0.42	0.44	0.41	0.40	0.46	0.48	0.47	**0.49**	− 1.3	− 1.7	− 2.0	− 2.1	− 2.9	− 2.5	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D9	0.22	0.30	0.35	0.46	0.47	0.48	0.48	0.46	0.46	0.45	**0.49**	− 2.9	− 3.8	− 3.8	− 3.1	− 3.3	− 3.8	− 3.8	− 3.2	− 3.0	− 3.8	− **3.9**
D10	0.21	0.29	0.28	0.30	0.32	0.32	0.33	0.35	0.37	0.36	**0.38**	− 1.1	− 2.0	− 2.8	− 2.1	− 2.1	− 2.1	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D11	0.25	0.31	0.36	0.40	0.39	0.41	0.42	0.42	0.43	0.44	**0.45**	− 3.1	− 3.0	− 3.8	− 3.1	− 4.1	− 4.1	− 4.8	− 4.2	− 4.0	− 4.8	− **4.9**
D12	0.28	0.38	0.39	0.43	0.45	0.45	0.41	0.43	0.45	0.46	**0.47**	− 1.1	− 2.0	− 2.8	− 2.1	− 2.1	− 2.2	− 2.8	− 2.2	− 3.0	− 3.1	− **3.2**
D13	0.18	0.21	0.23	0.31	0.32	0.32	0.33	0.35	0.37	0.37	**0.38**	− 2.1	− 2.0	− 2.8	− 3.1	− 3.1	− 3.1	− 3.8	− 3.2	− 3.5	− 3.8	− **3.9**
D14	0.28	0.31	0.35	0.41	0.43	0.44	0.45	0.46	0.48	0.49	**0.50**	− 3.1	− 3.9	− 3.8	− 3.8	− 4.1	− 4.1	− 4.8	− 4.2	− 4.2	− 4.2	− **4.3**
D15	0.21	0.29	0.30	0.32	0.32	0.33	0.34	0.35	0.37	0.36	**0.38**	− 1.1	− 3.0	− 3.8	− 3.8	− 3.7	− 3.8	− 3.8	− 4.2	− 4.0	− 4.8	− **4.9**
D16	0.19	0.27	0.28	0.30	0.32	0.33	0.35	0.36	0.37	0.38	**0.4**	− 2.1	− 3.0	− 3.8	− 3.9	− 3.8	− 3.6	− 3.8	− 4.0	− 4.0	− 4.1	− **4.2**
D17	0.18	0.81	0.88	0.89	0.87	0.86	0.83	0.85	0.87	0.87	**0.88**	− 1.1	− 2.0	− 2.8	− 2.1	− 2.1	− 2.3	− 2.8	− 2.5	− 2.6	− 2.8	− **2.9**
D18	0.20	0.28	0.29	0.30	0.32	0.33	0.38	0.39	0.32	0.33	**0.38**	− 1.1	− 2.0	− 2.1	− 2.1	− 2.2	− 2.3	− 2.3	− 2.4	− 2.5	− 2.8	− **2.9**
D19	0.28	0.33	0.38	0.34	0.37	0.38	0.41	0.42	0.45	0.45	**0.46**	− 1.8	− 2.8	− 2.6	− 2.6	− 2.8	− 2.0	− 2.7	− 2.2	− 2.3	− 2.8	− **2.9**
D20	0.18	0.28	0.28	0.24	0.26	0.27	0.28	0.26	0.27	0.21	**0.29**	− 0.1	− 1.2	− 1.8	− 1.1	− 1.1	− 1.1	− 1.8	− 1.2	− 1.0	− 1.8	− **1.9**
D21	0.008	0.071	0.088	0.087	0.089	0.0.091	0.021	0.35	0.17	0.27	**0.28**	− 1.1	− 2.0	− 2.8	− 2.5	− 2.5	− 2.7	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D22	0.20	0.26	0.28	0.27	0.29	0.21	0.27	0.27	0.28	0.29	**0.30**	− 1.1	− 2.2	− 2.3	− 2.5	− 2.6	− 2.7	− 2.5	− 2.2	− 2.0	− 2.7	− **2.8**
D23	0.002	0.060	0.020	0.057	0.079	0.081	0.087	0.087	0.088	0.092	**0.099**	− 0.9	− 1.2	− 1.3	− 1.5	− 1.6	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	− **1.8**
D24	0.12	0.18	0.20	0.22	0.26	0.27	0.28	0.28	**0.29**	0.22	0.22	− 1.0	− 1.5	− 1.3	− 1.5	− 1.6	− 1.7	− 1.5	− 1.2	− **1.8**	− 1.7	− 1.3
D25	0.17	0.20	0.22	0.23	0.23	0.23	0.25	0.21	0.21	0.27	**0.28**	− 0.9	− 1.2	− 1.3	− 1.2	− 1.3	− 1.4	− 1.3	− 1.3	− 1.4	− 1.7	− **1.8**
D26	0.19	0.26	0.25	0.24	0.27	0.28	0.29	0.28	0.25	0.24	**0.29**	− 0.3	− 1.2	− 1.0	− 0.9	− 0.9	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	− **1.8**
D27	0.17	0.23	0.24	0.26	0.24	0.25	0.23	0.23	0.25	0.27	**0.28**	− 0.9	− 1.2	− 1.3	− 1.5	− 1.2	− 1.4	− 1.5	− 1.2	− 1.0	− 1.7	− **1.8**
D28	0.19	0.23	0.27	0.24	0.25	0.26	0.23	0.23	0.24	0.25	**0.27**	− 0.1	− 0.8	− 0.8	− 0.5	− 0.6	− 0.7	− 0.5	− 0.7	− 0.7	− 0.7	− **0.8**
D29	0.11	0.20	0.22	0.24	0.22	0.20	0.26	0.25	0.25	0.25	**0.29**	− 0.8	− 1.2	− 1.3	− 1.5	− 1.6	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	− **1.9**
D30	0.13	0.16	0.18	0.17	0.19	0.20	0.23	0.24	0.25	0.23	**0.29**	− 0.9	− 1.6	− 1.3	− 1.5	− 1.6	− 1.7	− 2.0	− 2.0	− 2.1	− 2.3	− **2.4**

Table 7 Intra and inter cluster distance using K-mean

Inter												Intra										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	0.29	0.31	0.38	0.34	0.34	0.40	0.42	0.41	0.42	0.44	**0.45**	− 1.2	− 2.2	− 2.3	− 2.2	− 3.0	− 3.0	− 3.1	− 3.0	− 3.0	− 3.3	− **3.4**
D2	0.23	0.28	0.29	0.30	0.33	0.34	0.33	0.36	**0.39**	0.38	0.38	− 1.1	− 1.7	− 1.9	− 2.0	− 2.3	− 2.3	− 2.4	− 2.4	− **2.8**	− 2.2	2.3
D3	0.28	0.31	0.32	0.34	0.41	0.34	0.40	0.40	0.42	0.42	**0.44**	− 1.1	− 2.0	− 2.8	− 2.1	− 2.1	− 2.1	− 2.4	− 2.2	− 2.0	− 2.3	− **2.9**
D4	0.20	0.25	0.24	0.29	0.30	0.31	0.32	0.32	0.33	0.33	**0.35**	− 1.5	− 1.7	− 1.9	− 2.2	− 2.1	− 2.1	− 2.1	− 2.2	− 2.0	− 2.1	− **2.3**
D5	0.28	0.35	0.38	0.36	0.37	0.38	0.34	0.38	0.37	0.39	**0.40**	− 1.7	− 2.2	− 2.5	− 2.3	− 2.5	− 2.2	− 2.5	− 2.6	− 2.9	− 2.8	− **3.0**
D6	0.30	0.38	0.39	0.42	0.43	0.44	0.45	0.46	0.47	0.46	**0.48**	− 1.5	− 1.8	− 1.9	− 1.9	− 2.0	− 2.6	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D7	0.27	0.33	0.33	0.38	0.33	0.33	0.37	0.35	0.38	0.39	**0.40**	− 1.8	− 2.8	− 2.7	− 2.6	− 2.5	− 2.5	− 2.7	− 2.6	− 3.1	− 3.2	− **3.3**
D8	0.27	0.38	0.39	0.45	0.42	0.46	0.45	0.47	0.44	0.43	**0.48**	− 1.1	− 1.7	− 2.2	− 2.3	− 2.9	− 2.7	− 2.9	− 2.5	− 2.6	− 2.5	− **3.0**
D9	0.22	0.30	0.35	0.36	0.37	0.38	0.38	0.36	0.36	0.35	**0.39**	− 1.9	− 2.8	− 2.8	− 2.1	− 2.3	− 2.8	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D10	0.21	0.29	0.28	0.38	0.39	0.38	0.38	0.37	0.39	0.38	**0.45**	− 1.3	− 2.8	− 2.8	− 2.6	− 2.6	− 2.7	− 2.9	− 2.9	− 2.8	− 2.6	− **3.4**
D11	0.28	0.33	0.38	0.43	**0.49**	0.42	0.43	0.44	0.42	0.41	0.45	− 2.8	− 3.0	− 3.6	− 3.5	− 4.2	− **4.9**	− 4.2	− 4.1	− 4.0	− 4.0	− 4.0
D12	0.28	0.38	0.39	0.43	0.45	0.45	0.43	0.46	0.47	0.48	**0.49**	− 1.6	− 2.4	− 2.8	− 2.6	− 2.6	− 2.7	− 2.8	− 2.6	− 3.3	− 3.3	− **3.6**
D13	0.18	0.29	0.27	0.35	0.36	0.37	0.38	0.39	0.39	0.39	**0.40**	− 1.9	− 2.2	− 2.9	− 3.5	− 3.3	− 3.4	− 3.9	− 3.7	− 3.7	− 3.9	− **4.0**
D14	0.20	0.30	0.35	0.41	0.43	0.44	0.45	0.46	0.48	0.49	**0.49**	− 3.1	− 3.9	− 3.8	− 3.8	− 4.1	− 4.1	− 4.8	− 4.2	− 4.2	− 4.2	− **4.3**
D15	0.23	0.31	0.38	0.36	0.35	0.37	0.35	0.37	0.36	0.37	**0.40**	− 2.1	− 3.0	− 3.8	− 3.8	− 3.7	− 3.8	− 3.8	− 4.2	− 4.0	− 4.8	− **4.9**
D16	0.23	0.29	0.32	0.33	0.36	0.36	0.37	0.37	0.38	0.39	**0.42**	− 2.3	− 3.2	− 3.8	− 3.9	− 3.8	− 3.6	− 3.8	− 4.0	− 4.0	− 4.1	− **4.3**
D17	0.22	0.67	0.78	0.80	0.82	0.85	0.83	0.85	0.87	0.87	**0.88**	− 1.7	− 2.7	− 2.8	− 2.7	− 2.7	− 2.9	− 2.8	− 2.5	− 2.6	− 2.8	− **3.0**
D18	0.27	0.30	0.31	0.32	0.33	0.34	0.38	0.39	0.32	0.33	**0.40**	− 1.7	− 2.2	− 2.1	− 2.1	− 2.2	− 2.5	− 2.6	− 2.5	− 2.6	− 2.7	− **2.8**
D19	0.25	0.30	0.33	0.37	0.37	0.39	0.43	0.42	0.46	0.46	**0.47**	− 1.9	− 2.8	− 2.6	− 2.6	− 2.8	− 2.0	− 2.7	− 2.9	− 2.9	− 2.8	− **3.0**
D20	0.28	0.38	0.38	0.34	0.36	0.37	0.38	0.36	0.37	0.31	**0.39**	− 0.7	− 1.9	− 1.9	− 2.1	− 2.1	− 2.1	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D21	0.81	0.73	0.88	0.87	0.89	0.0.91	0.81	0.85	0.87	0.87	**0.89**	− 1.3	− 2.7	− 2.9	− 2.7	− 2.7	− 2.9	− 2.9	− 2.8	− 2.8	− 2.9	− **3.0**
D22	0.20	0.26	0.28	0.27	0.29	0.26	0.27	0.27	0.28	0.29	**0.30**	− 1.1	− 2.2	− 2.3	− 2.5	− 2.6	− 2.7	− 2.5	− 2.2	− 2.0	− 2.7	− **2.8**
D23	0.28	0.60	0.79	0.77	0.79	0.81	0.87	0.87	0.88	0.89	**0.90**	− 0.7	− 1.0	− 1.2	− 1.5	− 1.3	− 1.6	− 1.7	− 1.8	− 1.6	− 1.7	− **1.9**
D24	0.32	0.38	0.36	0.38	0.38	0.39	0.40	0.40	**0.49**	0.32	0.32	− 1.0	− 1.8	− 1.9	− 1.7	− 1.7	− 1.8	− 1.9	− 1.6	− **2.8**	− 2.7	− 2.3
D25	0.47	0.48	0.49	0.48	0.51	0.40	0.49	0.47	0.46	0.47	**0.52**	− 0.7	− 1.0	− 1.3	− 1.2	− 1.3	− 1.4	− 1.3	− 1.3	− 1.4	− 1.6	− **1.7**
D26	0.19	0.26	0.25	0.24	0.27	0.28	0.29	0.28	0.25	0.24	**0.32**	− 0.7	− 1.2	− 1.0	− 0.9	− 0.9	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	− **1.9**
D27	0.17	0.23	0.24	0.26	0.28	0.25	0.23	0.23	0.25	0.29	**0.31**	− 0.9	− 1.7	− 1.3	− 1.5	− 1.2	− 1.4	− 1.5	− 1.2	− 1.0	− 1.7	− **1.8**
D28	0.29	0.33	0.37	0.34	0.35	0.36	0.33	0.33	0.44	0.45	**0.47**	− 0.1	− 0.8	− 0.8	− 0.5	− 0.6	− 0.7	− 0.5	− 0.7	− 0.7	− 0.7	− **0.9**
D29	0.21	0.32	0.33	0.34	0.33	0.33	0.36	0.35	0.35	0.35	**0.39**	− 0.6	− 1.8	− 1.7	− 1.6	− 1.9	− 1.7	− 1.9	− 1.6	− 1.7	− 1.8	− **1.9**
D30	0.23	0.26	0.28	0.27	0.29	0.30	0.33	0.34	0.35	0.33	**0.39**	− 0.8	− 1.8	− 1.9	− 1.9	− 1.9	− 1.9	− 2.7	− 2.7	− 2.8	− 2.8	− **2.9**

Table 8 Intra and inter cluster distance using filtered clustering

Inter												Intra										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	0.27	0.32	0.38	0.33	0.35	0.37	0.38	0.38	0.39	0.39	**0.40**	− 1.2	− 2.0	− 2.2	− 2.3	− 2.7	− 2.6	− 2.7	− 2.5	− 2.5	− 2.6	**− 2.8**
D2	0.25	0.28	0.30	0.32	0.33	0.34	0.33	0.38	**0.40**	0.39	0.39	− 1.3	− 1.8	− 1.9	− 2.0	− 2.3	− 2.3	− 2.4	− 2.8	**− 2.9**	− 2.7	2.6
D3	0.26	0.33	0.37	0.34	0.43	0.40	0.42	0.43	0.42	0.43	**0.45**	− 1.7	− 2.2	− 2.7	− 2.2	− 2.3	− 2.3	− 2.4	− 2.4	− 2.5	− 2.3	**− 2.8**
D4	0.22	0.27	0.28	0.27	0.31	0.36	0.35	0.35	0.36	0.36	**0.39**	− 1.7	− 2.2	− 2.1	− 2.2	− 2.3	− 2.3	− 2.3	− 2.3	− 2.5	− 2.3	**− 2.7**
D5	0.30	0.37	0.39	0.38	0.37	0.39	0.37	0.39	0.38	0.39	**0.42**	− 1.8	− 2.2	− 2.6	− 2.8	− 2.7	− 2.5	− 2.6	− 2.7	− 2.9	− 2.8	**− 3.2**
D6	0.32	0.39	0.39	0.42	0.42	0.43	0.47	0.48	0.47	0.46	**0.49**	− 1.7	− 1.9	− 2.2	− 2.3	− 2.5	− 2.7	− 2.8	− 2.2	− 2.0	− 2.9	**− 3.0**
D7	0.26	0.33	0.35	0.39	0.37	0.38	0.38	0.38	0.39	0.39	**0.42**	− 1.9	− 2.9	− 2.7	− 2.6	− 2.8	− 2.8	− 2.7	− 2.8	− 3.2	− 3.3	**− 3.5**
D8	0.29	0.38	0.39	0.46	0.45	0.46	0.47	0.48	0.44	0.46	**0.49**	− 1.3	− 1.6	− 2.5	− 2.6	− 3.0	− 2.9	− 3.0	− 3.0	− 3.0	− 3.1	**− 3.2**
D9	0.25	0.32	0.33	0.36	0.37	0.38	0.38	0.36	0.36	0.38	**0.40**	− 1.7	− 2.7	− 2.7	− 2.6	− 2.7	− 2.8	− 2.8	− 2.2	− 2.0	− 2.9	**− 3.0**
D10	0.21	0.30	0.38	0.38	0.39	0.37	0.38	0.37	0.39	0.38	**0.41**	− 1.7	− 2.8	− 2.8	− 2.8	− 2.8	− 2.9	− 2.9	− 2.9	− 2.8	− 2.6	**− 3.1**
D11	0.28	0.33	0.38	0.43	**0.45**	0.42	0.43	0.44	0.42	0.41	0.45	− 2.8	− 3.0	− 3.6	− 3.5	− 4.2	**− 4.9**	− 4.2	− 4.1	− 4.0	− 4.0	− 4.0
D12	0.28	0.38	0.39	0.43	0.45	0.45	0.43	0.46	0.47	0.47	**0.48**	− 1.9	− 2.6	− 2.8	− 2.6	− 2.6	− 2.7	− 2.8	− 2.6	− 3.3	− 3.3	**− 3.4**
D13	0.22	0.29	0.27	0.38	0.36	0.37	0.38	0.39	0.39	0.39	**0.42**	− 1.9	− 2.2	− 2.9	− 3.5	− 3.3	− 3.4	− 3.9	− 3.7	− 3.6	− 3.5	**− 4.1**
D14	0.25	0.30	0.38	0.42	0.43	0.41	0.42	0.41	0.44	0.42	**0.45**	− 3.1	− 3.9	− 3.8	− 3.8	− 3.1	− 3.1	− 3.8	− 3.2	− 3.2	− 3.2	**− 4.0**
D15	0.30	0.33	0.39	0.38	0.37	0.37	0.35	0.37	0.36	0.37	**0.40**	− 2.5	− 3.0	− 3.8	− 3.8	− 3.9	− 3.8	− 3.8	− 4.5	− 4.3	− 4.7	**− 4.8**
D16	0.28	0.39	0.32	0.36	0.37	0.38	0.39	0.36	0.38	0.39	**0.40**	− 2.6	− 3.6	− 3.8	− 3.9	− 3.8	− 3.6	− 3.8	− 4.1	− 4.0	− 4.2	**− 4.3**
D17	0.29	0.62	0.73	0.82	0.82	0.81	0.82	0.84	0.86	0.87	**0.89**	− 1.3	− 2.1	− 2.8	− 2.7	− 2.7	− 2.9	− 2.8	− 2.5	− 2.6	− 2.8	**− 3.3**
D18	0.29	0.32	0.31	0.32	0.33	0.34	0.38	0.39	0.37	0.38	**0.40**	− 1.9	− 2.5	− 2.7	− 2.1	− 2.9	− 2.5	− 2.6	− 2.5	− 2.6	− 2.9	**− 3.2**
D19	0.29	0.32	0.36	0.39	0.39	0.39	0.43	0.42	0.4	0.43	**0.48**	− 1.9	− 2.8	− 2.6	− 2.6	− 2.8	− 2.2	− 2.7	− 2.9	− 2.9	− 2.9	**− 3.1**
D20	0.28	0.38	0.38	0.34	0.36	0.37	0.38	0.36	0.37	0.38	**0.40**	− 0.9	− 1.9	− 1.9	− 2.1	− 2.6	− 2.6	− 2.8	− 2.7	− 2.6	− 2.9	**− 3.2**
D21	0.81	0.73	0.80	0.81	0.89	0.88	0.87	0.85	0.88	0.88	**0.89**	− 1.3	− 2.8	− 2.9	− 2.9	− 2.3	− 2.9	− 2.9	− 2.8	− 2.8	− 2.9	**− 3.1**
D22	0.27	0.36	0.38	0.37	0.39	0.36	0.37	0.37	0.38	0.39	**0.40**	− 1.6	− 2.3	− 2.6	− 2.7	− 2.6	− 2.7	− 2.8	− 2.2	− 2.0	− 2.7	**− 2.9**
D23	0.28	0.60	0.29	0.57	0.79	0.81	0.87	0.87	0.88	0.86	**0.89**	− 0.20	− 1.0	− 1.2	− 1.5	− 1.8	− 1.6	− 1.7	− 1.8	− 1.6	− 1.7	**− 1.9**
D24	0.32	0.38	0.36	0.38	0.38	0.39	0.40	0.40	**0.49**	0.32	0.32	− 1.0	− 1.8	− 1.9	− 1.7	− 1.7	− 1.8	− 1.9	− 1.6	**− 2.8**	− 2.7	− 2.3
D25	0.47	0.48	0.49	0.47	0.50	0.40	0.49	0.47	0.46	0.47	**0.52**	− 0.7	− 1.0	− 1.3	− 1.2	− 1.3	− 1.4	− 1.3	− 1.3	− 1.4	− 1.6	**− 1.9**
D26	0.19	0.26	0.25	0.24	0.28	0.28	0.29	0.28	0.25	0.24	**0.30**	− 0.9	− 1.8	− 1.7	− 0.9	− 0.9	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	**− 1.9**
D27	0.17	0.23	0.24	0.26	0.28	0.25	0.23	0.23	0.25	0.29	**0.33**	− 0.9	− 1.7	− 1.3	− 1.5	− 1.2	− 1.4	− 1.5	− 1.2	− 1.0	− 1.7	**− 1.8**
D28	0.29	0.33	0.37	0.34	0.35	0.36	0.33	0.33	0.41	0.40	**0.42**	− 0.1	− 0.8	− 0.8	− 0.5	− 0.6	− 0.7	− 0.5	− 0.7	− 0.7	− 0.7	**− 0.9**
D29	0.21	0.32	0.33	0.34	0.33	0.33	0.36	0.35	0.32	0.33	**0.39**	− 0.6	− 1.8	− 1.7	− 1.6	− 1.9	− 1.7	− 1.9	− 1.6	− 1.7	− 1.8	**− 2.0**
D30	0.23	0.26	0.28	0.27	0.29	0.30	0.33	0.34	0.35	0.33	**0.40**	− 0.8	− 1.8	− 1.9	− 1.9	− 1.9	− 1.9	− 2.7	− 2.8	− 2.8	− 2.9	**− 3.0**

Table 9 Intra and inter cluster distance using density based cluster

Inter												Intra									
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	0.29	0.31	0.38	0.34	0.34	0.40	0.42	0.41	0.42	0.44	**0.45**	− 1.2	− 2.2	− 2.3	− 2.2	− 3.0	− 3.0	− 3.1	− 3.0	− 3.0	− 3.3	− **3.4**
D2	0.23	0.28	0.29	0.30	0.33	0.34	0.33	0.36	**0.39**	0.38	0.38	− 1.1	− 1.7	− 1.9	− 2.0	− 2.3	− 2.3	− 2.4	− 2.4	− **2.8**	− 2.2	2.3
D3	0.28	0.31	0.32	0.34	0.41	0.34	0.40	0.40	0.42	0.42	**0.44**	− 1.1	− 2.0	− 2.8	− 2.1	− 2.1	− 2.1	− 2.4	− 2.2	− 2.0	− 2.3	− **2.9**
D4	0.20	0.25	0.24	0.29	0.30	0.31	0.32	0.32	0.33	0.33	**0.35**	− 1.5	− 1.7	− 1.9	− 2.2	− 2.1	− 2.1	− 2.1	− 2.2	− 2.0	− 2.1	− **2.3**
D5	0.28	0.35	0.38	0.36	0.37	0.38	0.34	0.38	0.37	0.39	**0.40**	− 1.7	− 2.2	− 2.5	− 2.3	− 2.5	− 2.2	− 2.5	− 2.6	− 2.9	− 2.8	− **3.0**
D6	0.30	0.38	0.39	0.42	0.43	0.44	0.45	0.46	0.47	0.46	**0.48**	− 1.5	− 1.8	− 1.9	− 1.9	− 2.0	− 2.6	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D7	0.27	0.33	0.33	0.38	0.33	0.33	0.37	0.35	0.38	0.39	**0.40**	− 1.8	− 2.8	− 2.7	− 2.6	− 2.5	− 2.5	− 2.7	− 2.6	− 3.1	− 3.2	− **3.3**
D8	0.27	0.38	0.39	0.45	0.42	0.46	0.45	0.47	0.44	0.43	**0.48**	− 1.1	− 1.7	− 2.2	− 2.3	− 2.9	− 2.7	− 2.9	− 2.5	− 2.6	− 2.5	− **3.0**
D9	0.22	0.30	0.35	0.36	0.37	0.38	0.38	0.36	0.36	0.35	**0.39**	− 1.9	− 2.8	− 2.8	− 2.1	− 2.3	− 2.8	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D10	0.21	0.29	0.28	0.38	0.39	0.38	0.38	0.37	0.39	0.38	**0.45**	− 1.3	− 2.8	− 2.8	− 2.6	− 2.6	− 2.7	− 2.9	− 2.9	− 2.8	− 2.6	− **3.4**
D11	0.28	0.33	0.38	0.43	**0.49**	0.42	0.43	0.44	0.42	0.41	0.45	− 2.8	− 3.0	− 3.6	− 3.5	− 4.2	− **4.9**	− 4.2	− 4.1	− 4.0	− 4.0	− 4.0
D12	0.28	0.38	0.39	0.43	0.45	0.45	0.43	0.46	0.47	0.48	**0.49**	− 1.6	− 2.4	− 2.8	− 2.6	− 2.6	− 2.7	− 2.8	− 2.6	− 3.3	− 3.3	− **3.6**
D13	0.18	0.29	0.27	0.35	0.36	0.37	0.38	0.39	0.39	0.39	**0.40**	− 1.9	− 2.2	− 2.9	− 3.5	− 3.3	− 3.4	− 3.9	− 3.7	− 3.7	− 3.9	− **4.0**
D14	0.20	0.30	0.35	0.41	0.43	0.44	0.45	0.46	0.48	0.49	**0.49**	− 3.1	− 3.9	− 3.8	− 3.8	− 4.1	− 4.1	− 4.8	− 4.2	− 4.2	− 4.2	− **4.3**
D15	0.23	0.31	0.38	0.36	0.35	0.37	0.35	0.37	0.36	0.37	**0.40**	− 2.1	− 3.0	− 3.8	− 3.8	− 3.7	− 3.8	− 3.8	− 4.2	− 4.0	− 4.8	− **4.9**
D16	0.23	0.29	0.32	0.33	0.36	0.36	0.37	0.37	0.38	0.39	**0.42**	− 2.3	− 3.2	− 3.8	− 3.9	− 3.8	− 3.6	− 3.8	− 4.0	− 4.0	− 4.1	− **4.3**
D17	0.22	0.67	0.78	0.80	0.82	0.85	0.83	0.85	0.87	0.87	**0.88**	− 1.7	− 2.7	− 2.8	− 2.7	− 2.7	− 2.9	− 2.8	− 2.5	− 2.6	− 2.8	− **3.0**
D18	0.27	0.30	0.31	0.32	0.33	0.34	0.38	0.39	0.32	0.33	**0.40**	− 1.7	− 2.2	− 2.1	− 2.1	− 2.2	− 2.5	− 2.6	− 2.5	− 2.6	− 2.7	− **2.8**
D19	0.25	0.30	0.33	0.37	0.37	0.39	0.43	0.42	0.46	0.46	**0.47**	− 1.9	− 2.8	− 2.6	− 2.6	− 2.8	− 2.0	− 2.7	− 2.9	− 2.9	− 2.8	− **3.0**
D20	0.28	0.38	0.38	0.34	0.36	0.37	0.38	0.36	0.37	0.31	**0.39**	− 0.7	− 1.9	− 1.9	− 2.1	− 2.1	− 2.1	− 2.8	− 2.2	− 2.0	− 2.8	− **2.9**
D21	0.81	0.73	0.88	0.87	0.89	0.0.91	0.81	0.85	0.87	0.87	**0.89**	− 1.3	− 2.7	− 2.9	− 2.7	− 2.7	− 2.9	− 2.9	− 2.8	− 2.8	− 2.9	− **3.0**
D22	0.20	0.26	0.28	0.27	0.29	0.26	0.27	0.27	0.28	0.29	**0.30**	− 1.1	− 2.2	− 2.3	− 2.5	− 2.6	− 2.7	− 2.5	− 2.2	− 2.0	− 2.7	− **2.8**
D23	0.28	0.60	0.29	0.057	0.79	0.81	0.87	0.87	0.88	0.89	**0.90**	− 0.7	− 1.0	− 1.2	− 1.5	− 1.3	− 1.6	− 1.7	− 1.8	− 1.6	− 1.7	− **1.9**
D24	0.32	0.38	0.36	0.38	0.38	0.39	0.40	0.40	**0.49**	0.32	0.32	− 1.0	− 1.8	− 1.9	− 1.7	− 1.7	− 1.8	− 1.9	− 1.6	− **2.8**	− 2.7	− 2.3
D25	0.47	0.48	0.49	0.48	0.51	0.40	0.49	0.47	0.46	0.47	**0.52**	− 0.7	− 1.0	− 1.3	− 1.2	− 1.3	− 1.4	− 1.3	− 1.3	− 1.4	− 1.6	− **1.7**
D26	0.19	0.26	0.25	0.24	0.27	0.28	0.29	0.28	0.25	0.24	**0.32**	− 0.7	− 1.2	− 1.0	− 0.9	− 0.9	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	− **1.9**
D27	0.17	0.23	0.24	0.26	0.28	0.25	0.23	0.23	0.25	0.29	**0.31**	− 0.9	− 1.7	− 1.3	− 1.5	− 1.2	− 1.4	− 1.5	− 1.2	− 1.0	− 1.7	− **1.8**
D28	0.29	0.33	0.37	0.34	0.35	0.36	0.33	0.33	0.44	0.45	**0.47**	− 0.1	− 0.8	− 0.8	− 0.5	− 0.6	− 0.7	− 0.5	− 0.7	− 0.7	− 0.7	− **0.9**
D29	0.21	0.32	0.33	0.34	0.33	0.33	0.36	0.35	0.35	0.35	**0.39**	− 0.6	− 1.8	− 1.7	− 1.6	− 1.9	− 1.7	− 1.9	− 1.6	− 1.7	− 1.8	− **1.9**
D30	0.23	0.26	0.28	0.27	0.29	0.30	0.33	0.34	0.35	0.33	**0.39**	− 0.8	− 1.8	− 1.9	− 1.9	− 1.9	− 1.9	− 2.7	− 2.7	− 2.8	− 2.8	− **2.9**

Table 10 Intra and inter cluster distance using farthest first cluster

Inter												Intra										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	0.30	0.38	0.48	0.44	0.44	0.40	0.42	0.41	0.42	0.44	**0.48**	− 1.8	− 2.8	− 2.9	− 2.8	− 3.3	− 3.3	− 3.5	− 3.7	− 3.5	− 3.3	− **3.9**
D2	0.33	0.38	0.39	0.39	0.38	0.39	0.43	0.46	**0.49**	0.48	0.48	− 1.7	− 1.9	− 2.5	− 2.7	− 2.8	− 2.8	− 2.9	− 2.9	− **3.0**	− 2.9	2.7
D3	0.28	0.41	0.42	0.44	0.48	0.44	0.40	0.45	0.46	0.46	**0.47**	− 1.9	− 2.5	− 2.9	− 2.7	− 2.7	− 2.7	− 2.7	− 2.8	− 2.7	− 2.6	− **3.1**
D4	0.27	0.35	0.34	0.39	0.40	0.41	0.42	0.42	0.43	0.43	**0.45**	− 1.8	− 2.0	− 2.3	− 2.5	− 2.5	− 2.7	− 2.7	− 2.8	− 2.9	− 2.9	− **3.3**
D5	0.38	0.45	0.48	0.46	0.47	0.48	0.44	0.48	0.47	0.49	**0.50**	− 1.8	− 2.6	− 2.8	− 2.7	− 2.7	− 2.8	− 2.5	− 2.6	− 2.9	− 2.9	− **3.2**
D6	0.30	0.38	0.39	0.42	0.43	0.45	0.45	0.46	0.48	0.47	**0.49**	− 1.8	− 2.8	− 2.9	− 2.9	− 2.8	− 2.6	− 2.8	− 2.8	− 2.7	− 2.9	− **3.1**
D7	0.37	0.43	0.43	0.48	0.43	0.43	0.47	0.45	0.48	0.49	**0.50**	− 1.9	− 2.9	− 2.7	− 2.6	− 2.5	− 2.5	− 2.7	− 2.6	− 3.1	− 3.4	− **3.5**
D8	0.27	0.38	0.39	0.45	0.42	0.46	0.45	0.48	0.44	0.49	**0.51**	− 1.7	− 2.9	− 2.8	− 2.7	− 2.9	− 2.7	− 2.9	− 2.5	− 2.9	− 2.5	− **3.3**
D9	0.32	0.40	0.45	0.46	0.47	0.48	0.48	0.46	0.46	0.45	**0.49**	− 2.0	− 3.8	− 3.8	− 3.1	− 3.3	− 3.8	− 3.8	− 3.2	− 3.0	− 3.8	− **3.9**
D10	0.31	0.39	0.38	0.38	0.39	0.40	0.48	0.47	0.49	0.48	**0.50**	− 1.9	− 2.9	− 2.8	− 2.6	− 2.6	− 2.7	− 3.1	− 2.9	− 2.8	− 2.6	− **3.6**
D11	0.30	0.43	0.48	0.43	**0.51**	0.42	0.43	0.44	0.47	0.46	0.47	− 2.9	− 3.3	− 3.7	− 3.8	− 4.2	− **4.9**	− 4.7	− 4.2	− 4.0	− 4.1	− 4.1
D12	0.38	0.48	0.49	0.47	0.48	0.45	0.43	0.46	0.47	0.48	**0.53**	− 2.6	− 2.7	− 2.9	− 2.6	− 2.6	− 2.9	− 3.1	− 3.6	− 3.3	− 3.7	− **3.8**
D13	0.28	0.39	0.37	0.45	0.46	0.47	0.48	0.49	0.49	0.49	**0.51**	− 2.9	− 3.2	− 3.9	− 3.5	− 3.3	− 3.8	− 3.9	− 3.9	− 3.9	− 4.2	− **4.5**
D14	0.30	0.38	0.45	0.44	0.45	0.46	0.48	0.48	0.48	0.49	**0.56**	− 3.1	− 3.9	− 4.8	− 4.8	− 4.7	− 4.7	− 4.8	− 4.7	− 4.7	− 4.2	− **5.1**
D15	0.33	0.38	0.39	0.46	0.45	0.47	0.45	0.47	0.46	0.47	**0.49**	− 2.8	− 3.3	− 3.9	− 4.8	− 4.7	− 4.8	− 4.8	− 4.8	− 4.8	− 4.7	− **5.1**
D16	0.33	0.39	0.38	0.39	0.39	0.39	0.38	0.39	0.41	0.42	**0.47**	− 2.9	− 3.7	− 3.9	− 4.1	− 4.3	− 4.2	− 3.8	− 4.0	− 4.0	− 4.1	− **4.7**
D17	0.52	0.67	0.80	0.80	0.82	0.85	0.83	0.85	0.87	0.87	**0.89**	− 2.0	− 2.7	− 2.8	− 2.7	− 2.7	− 2.9	− 2.8	− 2.5	− 2.6	− 2.8	− **3.4**
D18	0.40	0.42	0.43	0.44	0.45	0.43	0.47	0.46	0.47	0.48	**0.49**	− 1.9	− 2.3	− 2.4	− 2.5	− 2.8	− 2.5	− 2.6	− 2.7	− 2.8	− 2.9	− **3.0**
D19	0.27	0.31	0.38	0.35	0.36	0.39	0.46	0.47	0.48	0.48	**0.49**	− 1.7	− 2.9	− 2.8	− 2.8	− 2.9	− 2.6	− 2.8	− 2.9	− 2.9	− 2.8	− **3.1**
D20	0.30	0.39	0.39	0.36	0.38	0.39	0.39	0.38	0.37	0.36	**0.41**	− 0.9	− 1.9	− 2.3	− 2.4	− 2.3	− 2.1	− 2.8	− 2.8	− 2.0	− 2.8	− **3.0**
D21	0.81	0.73	0.88	0.87	0.89	0.0.85	0.81	0.85	0.87	0.87	**0.89**	− 1.8	− 2.9	− 2.9	− 2.7	− 2.7	− 2.9	− 2.9	− 2.8	− 2.8	− 2.9	− **3.2**
D22	0.31	0.36	0.38	0.37	0.39	0.36	0.37	0.37	0.38	0.39	**0.43**	− 1.3	− 2.7	− 2.8	− 2.7	− 2.8	− 2.7	− 2.5	− 2.6	− 2.6	− 2.3	− **2.9**
D23	0.30	0.60	0.38	0.57	0.79	0.81	0.87	0.80	0.88	0.82	**0.90**	− 0.9	− 1.1	− 1.3	− 1.7	− 1.9	− 1.9	− 1.8	− 1.8	− 1.9	− 1.7	− **2.0**
D24	0.32	0.39	0.36	0.38	0.38	0.39	0.40	0.40	**0.47**	0.40	0.42	− 1.7	− 2.8	− 2.9	− 2.7	− 2.7	− 2.8	− 2.9	− 2.6	− **3.1**	− 3.0	− 3.0
D25	0.37	0.48	0.49	0.48	0.51	0.40	0.49	0.47	0.46	0.47	**0.50**	− 0.7	− 1.0	− 1.3	− 1.2	− 1.3	− 1.4	− 1.3	− 1.3	− 1.4	− 1.6	− **1.9**
D26	0.22	0.36	0.35	0.34	0.37	0.38	0.39	0.38	0.35	0.34	**0.41**	− 0.9	− 1.7	− 1.2	− 1.9	− 1.9	− 1.7	− 1.5	− 1.2	− 1.0	− 1.7	− **2.0**
D27	0.27	0.30	0.34	0.36	0.38	0.35	0.33	0.33	0.35	0.39	**0.41**	− 0.9	− 1.9	− 1.8	− 1.7	− 1.8	− 1.7	− 1.8	− 1.9	− 1.6	− 1.8	− **1.9**
D28	0.27	0.33	0.37	0.34	0.35	0.36	0.33	0.33	0.43	0.42	**0.46**	− 0.1	− 0.8	− 0.8	− 0.5	− 0.6	− 0.7	− 0.5	− 0.7	− 0.7	− 0.7	− **1.3**
D29	0.21	0.32	0.33	0.34	0.33	0.33	0.36	0.35	0.35	0.35	**0.39**	− 0.6	− 1.8	− 1.7	− 1.6	− 1.9	− 1.7	− 1.9	− 1.6	− 1.7	− 1.8	− **1.9**
D30	0.23	0.26	0.28	0.27	0.29	0.30	0.33	0.34	0.35	0.33	**0.41**	− 0.9	− 1.8	− 1.9	− 1.9	− 1.9	− 1.9	− 2.7	− 2.7	− 2.8	− 2.8	− **2.9**

Table 11 Accuracy and F-measure using SOM

Accuracy												F-measure										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	62.33	73.33	75.0	77.37	77.66	**78.4**	77	**78.4**	76.6	77.6	**78.4**	0.68	**0.79**	0.77	0.76	**0.79**	0.72	0.71	**0.79**	**0.79**	**0.79**	**0.79**
D2	68.18	70	72.08	76.27	72.66	78	76	73	76.6	77.6	**83**	0.63	0.78	0.72	0.74	0.73	0.72	0.76	0.77	0.76	0.76	**0.82**
D3	70.8	82	82.8	83.67	81.06	84	83	82	82.6	81.6	**89.7**	0.72	0.81	0.82	0.82	0.83	0.84	0.83	0.82	0.84	0.83	**0.88**
D4	62.8	72	71.08	72.27	71.60	76	75	72	75.6	79.6	**82**	0.68	0.73	0.70	0.72	0.71	0.70	0.72	0.79	0.78	0.77	**0.80**
D5	70	76	81	82	86	82	85	81	86	87	**89**	0.67	0.80	0.84	0.85	0.83	0.86	0.84	0.82	0.84	0.81	**0.89**
D6	68.8	72	72.08	76.27	75.66	78	76	81	86.6	85.6	**88**	0.71	0.78	0.72	0.74	0.73	0.72	0.76	0.80	0.82	0.83	**0.89**
D7	67.8	71.8	80	82	86	87	87	85	80	88	**89**	0.71	0.80	0.80	0.84	0.83	0.84	0.82	0.86	0.88	0.89	**0.90**
D8	67	72	78	79.7	81	83	84	85	88	87	**90**	0.71	0.81	0.80	0.82	0.81	0.82	0.82	0.83	0.83	0.84	**0.87**
D9	78	82	87	89	89	90	90.8	**93**	91	91	92	0.78	0.83	0.86	0.84	0.91	0.90	0.92	**0.95**	0.90	0.90	0.94
D10	66.8	75	80	88	89	87	86	85	89	88	**90**	0.68	0.72	0.78	0.76	0.73	0.77	0.75	0.82	0.84	0.83	**0.89**
D11	69	78	80	84	84.7	88.7	88	81	85	87	**89**	0.68	0.77	0.76	0.80	0.81	0.84	0.83	0.83	0.84	0.83	**0.90**
D12	70.8	81	86	85	88	87	85	88	82	85	**89**	0.71	0.84	0.83	0.86	0.85	0.84	0.83	0.86	0.86	0.88	**0.89**
D13	71	79.9	84.6	87.3	88.7	86.7	87	88	88.2	88.1	**89.8**	0.67	0.78	0.82	0.84	0.83	0.81	0.85	0.83	0.84	0.83	**0.89**
D14	77.3	83.8	86.8	86.3	82	85	87	86	87.6	87.3	**89.7**	0.71	0.80	0.86	0.84	0.83	0.88	0.83	0.84	0.83	0.86	**0.88**
D15	70	88	84	88	89	89	89.9	89.8	91.8	90.7	**92.4**	0.73	0.84	0.85	0.86	0.83	0.84	0.85	0.88	0.88	0.89	**0.91**
D16	67	75	85	86	85	83	82	80	84	85	**88**	0.70	0.78	0.84	0.83	0.83	0.86	0.84	0.85	0.84	0.83	**0.88**
D17	79	86	88	86	89	87.7	88	88.4	87	86.8	**89.9**	0.76	0.82	0.80	0.85	0.83	0.85	0.86	0.88	0.86	0.88	**0.89**
D18	79	82	87	89	89.7	88	87	88	86	89	**89.9**	0.69	0.80	0.82	0.85	0.84	0.83	0.85	0.86	0.86	0.86	**0.88**
D19	68.9	73	76	80	81	82	83	84.8	85.8	86.9	**88**	0.72	0.85	0.82	0.85	0.86	0.84	0.81	0.82	0.80	0.82	**0.87**
D20	68	76	77	78.6	79.8	80	81.9	88.3	**89.6**	81	89	0.68	0.72	0.76	0.73	0.80	0.78	0.79	0.80	**0.81**	0.80	0.79
D21	67	79	78	80	80.9	87.8	88.7	89.8	90	89.7	**90.7**	0.74	0.82	0.81	0.82	0.83	0.84	0.84	0.83	0.81	0.80	**0.85**
D22	70	78	82	83	84.7	86	81	82	84	85	**88**	0.70	0.80	0.81	0.83	0.84	0.83	0.84	0.83	0.84	0.85	**0.89**
D23	68.9	78	81	80.88	83.76	88.78	83.71	83.9	**89**	84	88	0.66	0.72	0.79	0.82	0.80	0.81	**0.87**	0.83	0.82	0.80	0.81
D24	65	78	80	80.8	82	88	88.2	**89.3**	89	87.7	82	0.70	0.78	0.80	0.81	0.83	0.84	0.81	**0.87**	0.82	0.85	0.85
D25	78	80	82	84	85	88	89	87	89	84	**89.8**	0.71	0.80	0.84	0.86	0.87	0.85	0.83	0.82	0.81	0.80	**0.88**
D26	66	74	78.7	76.9	79.6	80.8	82.1	84	85	85	**88**	0.72	0.78	0.80	0.82	0.83	0.85	0.84	0.85	0.86	0.86	**0.87**
D27	67	80	83	84	81	88	81.9	89.6	87	86.9	**89**	0.67	0.78	0.76	0.80	0.82	0.81	0.82	0.83	0.84	0.82	**0.88**
D28	67	84	85	86	87	89	88	**89.8**	86	81	88	0.71	0.78	0.80	0.83	0.84	0.86	0.84	0.81	0.86	0.85	**0.88**
D29	67	78	80	82	83	81	87	**88.9**	85	84	81.9	0.67	0.78	0.80	0.83	0.85	0.87	0.86	0.87	**0.88**	0.81	0.82
D30	60	76	82	84	83	84	89	89	88	81	**89.8**	0.67	0.72	0.80	0.81	0.83	0.84	0.85	0.86	0.83	0.81	**0.88**

Table 12 Accuracy and F-measure using K-mean

Accuracy												F-measure										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	72.33	84.33	85.66	87.57	81.66	**88.4**	81	**88.4**	80.6	80.6	**88.4**	0.71	**0.89**	0.87	0.86	**0.89**	0.82	0.81	**0.89**	**0.89**	0.79	**0.89**
D2	69.18	78.8	78.08	79.27	79.66	80	81	83	86.6	87	**88**	0.69	0.75	0.78	0.79	0.83	0.82	0.86	0.87	0.86	0.86	**0.89**
D3	72.8	82	84.8	87.67	85.06	87	82	84	83.6	84.6	**86.7**	0.68	0.77	0.80	0.81	0.84	0.85	0.86	0.84	0.85	0.82	**0.85**
D4	72.8	76	77.08	78.27	76.60	78	80	81	79.6	80.6	**81.9**	0.62	0.70	0.73	0.76	0.77	0.78	0.79	0.80	0.79	0.76	**0.81**
D5	76	79	83	86	88	85	83	84	83	84	**88.9**	0.69	0.79	0.82	0.86	0.81	0.84	0.82	0.80	0.81	0.84	**0.85**
D6	71.8	78	78.08	79.27	79.66	79	80	82	85.5	82.9	**87**	0.67	0.78	0.75	0.74	0.73	0.71	0.76	0.80	0.82	0.82	**0.84**
D7	67.8	72.8	78.9	82	83	84	85	86	88	89	**89.8**	0.77	0.82	0.81	0.83	0.84	0.85	0.81	0.86	0.87	0.80	**0.89**
D8	67	72	78	79.7	82	85.8	84	85	88	87	**89**	0.71	0.81	0.80	0.82	0.81	0.82	0.82	0.83	0.83	0.84	**0.87**
D9	78	82	87.7	89	89	90	90.8	**93.7**	91	91	93	0.78	0.83	0.86	0.84	0.91	0.90	0.92	**0.95**	0.90	0.90	0.94
D10	66.8	75	80	88	89.9	87.8	86	85	89	88	**89**	0.65	0.71	0.76	0.76	0.73	0.77	0.75	0.82	0.84	0.83	**0.85**
D11	69	78	80	84	85.7	86.1	82	81	85	87	**88.7**	0.68	0.72	0.74	0.80	0.81	0.84	0.83	0.81	0.82	0.83	**0.86**
D12	70.8	80.8	86.8	85.6	88.6	87.4	85.1	88.5	82.3	85.1	**88.9**	0.69	0.74	0.83	0.80	0.85	0.81	0.82	0.85	0.83	0.87	**0.88**
D13	75	79.9	82.6	87	86.7	86.7	87	81	82.2	87.1	**87.8**	0.69	0.76	0.81	0.83	0.85	0.80	0.85	0.81	0.83	0.80	**0.86**
D14	77.3	83.8	86.8	86.3	82	85	87	86	84.6	85.3	**86.7**	0.71	0.82	0.84	0.84	0.83	0.88	0.83	0.84	0.83	0.86	**0.87**
D15	78	85	86	88.8	89.5	88.7	88.9	88.8	81.8	80.7	**89.8**	0.70	0.81	0.86	0.85	0.86	0.85	0.80	0.84	0.88	0.87	**0.89**
D16	67	75	85	86	85	83	82	80	84	85	**88**	0.70	0.78	0.84	0.83	0.83	0.86	0.84	0.85	0.84	0.83	**0.88**
D17	79	86	88	86	82	86.7	88	86.4	87	82.8	**89**	0.76	0.82	0.80	0.85	0.83	0.85	0.86	0.88	0.86	0.88	**0.89**
D18	79	80	81	86	82.7	83	85	82	86	81	**86.9**	0.69	0.80	0.82	0.85	0.84	0.80	0.81	0.83	0.82	0.84	**0.85**
D19	68.9	73	76	82	83	85	84	85.8	86.8	87.9	**88.8**	0.70	0.81	0.82	0.83	0.85	0.82	0.80	0.82	0.80	0.82	**0.86**
D20	68	76	77	78.6	79.8	80.8	81.9	84.3	**86.6**	81	89	0.68	0.72	0.74	0.72	0.78	0.76	0.77	0.79	**0.80**	0.78	0.79
D21	67	79	78	80	82.9	87.8	87.7	86.8	87.8	83.7	**89.7**	0.72	0.81	0.83	0.85	0.87	0.85	0.87	0.86	0.87	0.82	**0.88**
D22	70	78	82	83	84.7	85	81	82	84	85	**86**	0.70	0.80	0.81	0.83	0.84	0.83	0.84	0.83	0.84	0.85	**0.89**
D23	68.9	78	81	80.88	83.76	88	83.71	83.9	**89.2**	84	88	0.66	0.72	0.79	0.82	0.80	0.81	**0.87**	0.83	0.82	0.80	0.81
D24	65	78	80	80.8	82	86	88.2	**88.6**	89	85.7	81	0.70	0.78	0.80	0.81	0.83	0.81	0.83	**0.86**	0.82	0.84	0.83
D25	78	80	82	84	85	82	84	83	87	84	**87.8**	0.71	0.81	0.82	0.84	0.85	0.86	0.85	0.83	0.80	0.82	**0.87**
D26	66	74	78.7	77.9	79.6	80.8	82.1	84	85	82	**87**	0.72	0.78	0.81	0.82	0.82	0.84	0.84	0.85	0.81	0.82	**0.85**
D27	67	80	83	84	81	82	80.9	89.2	81	85.9	**86**	0.67	0.78	0.76	0.80	0.82	0.81	0.82	0.81	0.84	0.82	**0.86**
D28	67	84	85	86	84	83	85	87.8	**89**	81	88	0.71	0.78	0.80	0.83	0.81	0.84	0.85	0.81	0.80	0.84	**0.86**
D29	67	78	80	82	83	81	87	**87.9**	85	84	83.7	0.67	0.78	0.82	0.81	0.83	0.85	0.81	0.83	**0.87**	0.83	0.84
D30	60	76	81	82	83	84.7	89.6	89.2	88	84	**89.8**	0.67	0.72	0.80	0.81	0.83	0.84	0.85	0.82	0.81	0.84	**0.89**

Table 13 Accuracy and F-measure using filtered clustering

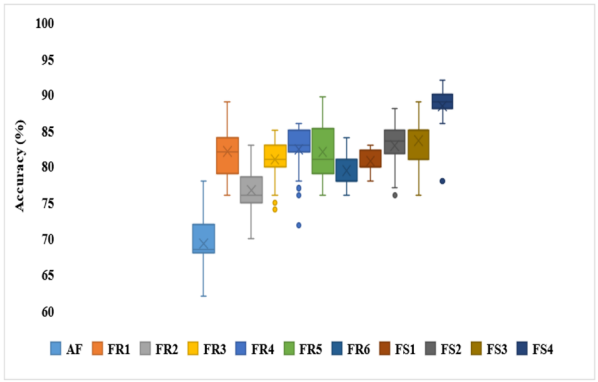
Accuracy												F-measure										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	78.33	82	84	86	86	82	83	88	87	84	**89**	0.72	0.80	0.83	0.84	0.85	0.86	0.87	0.82	0.88	0.81	**0.89**
D2	72.9	81.88	83	85	84	83	86	87	89	89	**90**	0.72	0.80	0.85	0.83	0.82	0.80	0.84	0.80	0.83	0.85	**0.86**
D3	67	81	84	87	89	86	82	84	85	88.9	**89.9**	0.78	0.87	0.86	0.85	0.83	0.85	0.86	0.85	0.84	0.85	**0.88**
D4	72.8	88	86	84	85	83	88	85	87	89	**90.9**	0.70	0.82	0.87	0.85	0.86	0.86	0.86	0.88	0.81	0.88	**0.89**
D5	73	81	83	80	81	86	88	82	83	84	**88.7**	0.67	0.80	0.81	0.82	0.83	0.83	0.84	0.85	0.86	0.82	**0.89**
D6	67	85	87	86	85	84	88	89	90.8	90	**91**	0.69	0.88	0.85	0.86	0.87	0.87	0.85	0.88	0.89	0.89	**0.90**
D7	72	81	85	88	89	89.6	88.7	86	86.8	89.7	**90.8**	0.70	0.86	0.85	0.84	0.87	0.89	0.86	0.87	0.88	0.81	**0.89**
D8	65	77	78	78	82	84	85	86	87	87.9	**90.9**	0.67	0.81	0.81	0.88	0.85	0.84	0.83	0.84	0.84	0.87	**0.88**
D9	68	84	87	90	88	83	84	**89**	**89**	87	**89**	0.78	0.89	0.82	0.84	0.83	0.82	0.89	**0.90**	0.81	0.82	0.88
D10	66.8	78	86	86	88	82	89	89	87.8	86.7	**89.9**	0.70	0.82	0.81	0.88	0.82	0.85	0.88	0.81	0.82	0.88	**0.89**
D11	79	88	88	86	86	89	89	80	86	88	**91**	0.72	0.87	0.86	0.87	0.86	0.87	0.85	0.84	0.82	0.85	**0.89**
D12	66.8	82	88	82	86	81	83	87	88	89.1	**89.8**	0.73	0.81	0.80	0.82	0.81	0.83	0.82	0.86	0.86	0.84	**0.88**
D13	70.1	78	81	80	82	82	87	82	86	88	**89.6**	0.60	0.78	0.80	0.81	0.80	0.79	0.81	0.83	0.85	0.85	**0.89**
D14	67	81	80	86	82	85	82	81	86	86	**87.9**	0.67	0.82	0.86	0.81	0.80	0.81	0.83	0.83	0.84	0.83	**0.86**
D15	69.7	88	88	86	86	88	88	89.8	88.7	82.9	**88.9**	0.69	0.86	0.78	0.80	0.86	0.85	0.83	0.81	0.82	0.84	**0.87**
D16	67	71	80	87	86	82	83.8	81	86.9	87.8	**88.1**	0.72	0.82	0.81	0.81	0.80	0.82	0.83	0.80	0.80	0.81	**0.84**
D17	80	86	88	86	89	89	88	83	86	81	**89.7**	0.67	0.86	0.82	0.85	0.82	0.78	0.72	0.80	0.78	0.88	**0.89**
D18	72	84	86	81	84	82.8	85	82.9	87	86	**89**	0.74	0.81	0.85	0.85	0.87	0.82	0.85	0.86	0.84	0.88	**0.9**
D19	78	83	86	88	88	89	87	86	87	89	**89.7**	0.72	0.81	0.81	0.84	0.84	0.88	0.81	0.82	0.80	0.82	**0.88**
D20	68	88	86	86	88	89	85	**88**	85	81	82	0.78	0.81	0.82	0.81	0.84	0.83	0.85	0.86	**0.87**	0.82	0.83
D21	62	78	78	80	81	80	81.7	82.7	83.4	84.2	**88.7**	0.67	0.82	0.85	0.84	0.85	0.87	0.80	0.81	0.87	0.86	**0.88**
D22	69.8	83	82	85	86	86	83	81	82	85	**86.9**	0.68	0.80	0.83	0.85	0.81	0.80	0.82	0.82	0.85	0.85	**0.86**
D23	68.9	80	81	80.88	82.76	81.78	85.71	86.9	**89.9**	81	80.1	0.68	0.72	0.80	0.84	0.80	0.82	**0.85**	0.83	0.82	0.80	0.82
D24	65	80	81	81	82	83	86.4	**88.3**	85	86.7	82	0.67	0.72	0.80	0.81	0.82	0.83	0.80	**0.84**	0.81	0.82	0.83
D25	69.9	86	84	84	84	87	85	83	86	87	**88.1**	0.77	0.88	0.81	0.86	0.88	0.88	0.82	0.85	0.86	0.88	**0.89**
D26	69.9	81	82.7	83.9	84.6	84.8	86.1	84	87	81	**87.1**	0.73	0.82	0.83	0.81	0.82	0.83	0.81	0.82	0.86	0.85	**0.89**
D27	67	82	82	84	80	80.1	82.9	83.6	86.8	85.8	**87.9**	0.71	0.82	0.82	0.84	0.84	0.85	0.87	0.85	0.86	0.87	**0.88**
D28	63	82	86	89	86	82.8	81.1	**86.5**	82	81.9	81.9	0.78	0.78	0.80	0.82	0.81	0.84	0.83	**0.85**	0.80	0.81	0.83
D29	67	78	82	85	86	82	87	**88.9**	85	84	88.4	0.77	0.80	0.81	0.84	0.86	0.88	0.87	0.86	**0.87**	0.85	0.86
D30	60	76	82	84	87	89	81	82	85	81	**89.6**	0.67	0.72	0.71	0.82	0.85	0.86	0.87	0.85	0.87	0.87	**0.88**

Table 14 Accuracy and F-measure using density-based clustering

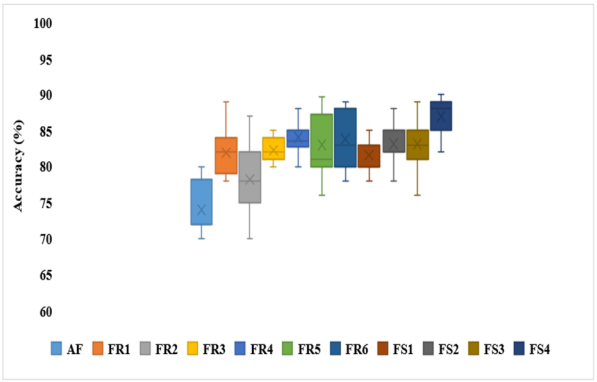
Accuracy												F-measure										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	72.33	84.33	85.66	87.57	81.66	**88.4**	81	**88.4**	80.6	80.6	**88.4**	0.71	**0.89**	0.87	0.86	**0.89**	0.82	0.81	**0.89**	**0.89**	0.79	**0.89**
D2	69.18	78.8	78.08	79.27	79.66	80	81	83	86.6	87	**88**	0.69	0.75	0.78	0.79	0.83	0.82	0.86	0.87	0.86	0.86	**0.89**
D3	72.8	82	84.8	87.67	85.06	87	82	84	83.6	84.6	**86.7**	0.68	0.77	0.80	0.81	0.84	0.85	0.86	0.84	0.85	0.82	**0.85**
D4	72.8	76	77.08	78.27	76.60	78	80	81	79.6	80.6	**81.9**	0.62	0.70	0.73	0.76	0.77	0.78	0.79	0.80	0.79	0.76	**0.81**
D5	76	79	83	86	88	85	83	84	83	84	**88.9**	0.69	0.79	0.82	0.86	0.81	0.84	0.82	0.80	0.81	0.84	**0.85**
D6	71.8	78	78.08	79.27	79.66	79	80	82	85.5	82.9	**87**	0.67	0.78	0.75	0.74	0.73	0.71	0.76	0.80	0.82	0.82	**0.84**
D7	67.8	72.8	78.9	82	83	84	85	86	88	89	**89.8**	0.77	0.82	0.81	0.83	0.84	0.85	0.81	0.86	0.87	0.80	**0.89**
D8	67	72	78	79.7	82	85.8	84	85	88	87	**89**	0.71	0.81	0.80	0.82	0.81	0.82	0.82	0.83	0.83	0.84	**0.87**
D9	78	82	87.7	89	89	90	90.8	**93.7**	91	91	93	0.78	0.83	0.86	0.84	0.91	0.90	0.92	**0.95**	0.90	0.90	0.94
D10	66.8	75	80	88	89.9	87.8	86	85	89	88	**89**	0.65	0.71	0.76	0.76	0.73	0.77	0.75	0.82	0.84	0.83	**0.85**
D11	69	78	80	84	85.7	86.1	82	81	85	87	**88.7**	0.68	0.72	0.74	0.80	0.81	0.84	0.83	0.81	0.82	0.83	**0.86**
D12	70.8	80.8	86.8	85.6	88.6	87.4	85.1	88.5	82.3	85.1	**88.9**	0.69	0.74	0.83	0.80	0.85	0.81	0.82	0.85	0.83	0.87	**0.88**
D13	75	79.9	82.6	87	86.7	86.7	87	81	82.2	87.1	**87.8**	0.69	0.76	0.81	0.83	0.85	0.80	0.85	0.81	0.83	0.80	**0.86**
D14	77.3	83.8	86.8	86.3	82	85	87	86	84.6	85.3	**86.7**	0.71	0.82	0.84	0.84	0.83	0.88	0.83	0.84	0.83	0.86	**0.87**
D15	78	85	86	88.8	89.5	88.7	88.9	88.8	81.8	80.7	**89.8**	0.70	0.81	0.86	0.85	0.86	0.85	0.80	0.84	0.88	0.87	**0.89**
D16	67	75	85	86	85	83	82	80	84	85	**88**	0.70	0.78	0.84	0.83	0.83	0.86	0.84	0.85	0.84	0.83	**0.88**
D17	79	86	88	86	82	86.7	88	86.4	87	82.8	**89**	0.76	0.82	0.80	0.85	0.83	0.85	0.86	0.88	0.86	0.88	**0.89**
D18	79	80	81	86	82.7	83	85	82	86	81	**86.9**	0.69	0.80	0.82	0.85	0.84	0.80	0.81	0.83	0.82	0.84	**0.85**
D19	68.9	73	76	82	83	85	84	85.8	86.8	87.9	**88.8**	0.70	0.81	0.82	0.83	0.85	0.82	0.80	0.82	0.80	0.82	**0.86**
D20	68	76	77	78.6	79.8	80.8	81.9	84.3	**86.6**	81	89	0.68	0.72	0.74	0.72	0.78	0.76	0.77	0.79	**0.80**	0.78	0.79
D21	67	79	78	80	82.9	87.8	87.7	86.8	87.8	83.7	**89.7**	0.72	0.81	0.83	0.85	0.87	0.85	0.87	0.86	0.87	0.82	**0.88**
D22	70	78	82	83	84.7	85	81	82	84	85	**86**	0.70	0.80	0.81	0.83	0.84	0.83	0.84	0.83	0.84	0.85	**0.89**
D23	68.9	78	81	80.88	83.76	88	83.71	83.9	**89.2**	84	88	0.66	0.72	0.79	0.82	0.80	0.81	**0.87**	0.83	0.82	0.80	0.81
D24	65	78	80	80.8	82	86	88.2	**88.6**	89	85.7	81	0.70	0.78	0.80	0.81	0.83	0.81	0.83	**0.86**	0.82	0.84	0.83
D25	78	80	82	84	85	82	84	83	87	84	**87.8**	0.71	0.81	0.82	0.84	0.85	0.86	0.85	0.83	0.80	0.82	**0.87**
D26	66	74	78.7	77.9	79.6	80.8	82.1	84	85	82	**87**	0.72	0.78	0.81	0.82	0.82	0.84	0.84	0.85	0.81	0.82	**0.85**
D27	67	80	83	84	81	82	80.9	89.2	81	85.9	**86**	0.67	0.78	0.76	0.80	0.82	0.81	0.82	0.81	0.84	0.82	**0.86**
D28	67	84	85	86	84	83	85	87.8	**89**	81	88	0.71	0.78	0.80	0.83	0.81	0.84	0.85	0.81	0.80	0.84	**0.86**
D29	67	78	80	82	83	81	87	**87.9**	85	84	83.7	0.67	0.78	0.82	0.81	0.83	0.85	0.81	0.83	**0.87**	0.83	0.84
D30	60	76	81	82	83	84.7	89.6	89.2	88	84	**89.8**	0.67	0.72	0.80	0.81	0.83	0.84	0.85	0.82	0.81	0.84	**0.89**

Table 15 Accuracy and F-measure using farthest first clustering

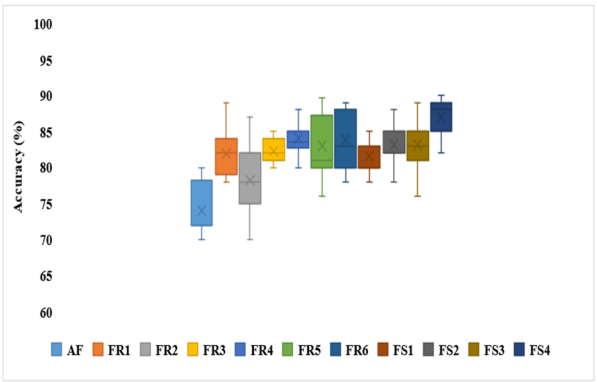
Accuracy												F-measure										
ID	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
D1	68.33	81	84	86	86	82	83	85	86	83	**89.8**	0.79	0.81	0.85	0.83	0.81	0.83	0.85	0.82	0.87	0.81	**0.89**
D2	65	80.88	84	87	86	85	82	85	86	89	**91.8**	0.75	0.82	0.86	0.85	0.84	0.81	0.85	0.83	0.85	0.81	**0.87**
D3	67	81	84	87	82	85	83	81	84	89	**90.8**	0.78	0.87	0.86	0.85	0.83	0.85	0.86	0.85	0.84	0.87	**0.89**
D4	62.8	78	81	89	86	83	89	85	87	89	**90.7**	0.72	0.80	0.88	0.84	0.87	0.86	0.86	0.87	0.81	0.86	**0.88**
D5	68.8	81	83	80	81	86	87	82	83	85	**89.8**	0.67	0.80	0.81	0.82	0.83	0.83	0.84	0.85	0.86	0.87	**0.90**
D6	67.9	85	87	86	85	84	88	89	92	94	**96.7**	0.69	0.88	0.85	0.86	0.87	0.87	0.85	0.88	0.87	0.88	**0.90**
D7	78	81	85	88	89	89.6	88.7	86	86.8	89.7	**93.8**	0.70	0.86	0.85	0.84	0.87	0.89	0.86	0.87	0.82	0.81	**0.89**
D8	65	78	75	78	82	84	85	86	87	88	**91**	0.67	0.81	0.81	0.88	0.85	0.84	0.83	0.84	0.84	0.88	**0.89**
D9	68	84	87	92	91	83	84	**96**	95	93	86	0.78	0.89	0.92	0.94	0.93	0.92	0.96	**0.99**	0.91	0.92	0.91
D10	66.8	78	86	88	89	82	89	89	89.8	89.7	**97**	0.70	0.82	0.81	0.88	0.86	0.87	0.85	0.88	0.82	0.88	**0.96**
D11	79	88	88	86	86	89	89	80	86	88	**98**	0.72	0.87	0.86	0.86	0.85	0.87	0.85	0.84	0.82	0.85	**0.93**
D12	66.8	82	88	82	86	81	83	88	87	89	**90**	0.75	0.80	0.81	0.82	0.81	0.81	0.82	0.86	0.86	0.84	**0.89**
D13	69.1	78	81	81	82	82	87	82	86	88	**89.8**	0.60	0.78	0.80	0.81	0.80	0.79	0.81	0.82	0.80	0.81	**0.88**
D14	67	81	80	86	82	85	82	81	86	86	**90.9**	0.67	0.82	0.86	0.81	0.80	0.88	0.83	0.85	0.81	0.82	**0.89**
D15	70.7	88	88	86	86	88	88	89.8	91	92	**97**	0.69	0.86	0.88	0.83	0.86	0.85	0.83	0.84	0.86	0.86	**0.94**
D16	67	71	80	87	86	82	83	81	86	87	**88**	0.72	0.82	0.81	0.80	0.80	0.81	0.83	0.80	0.80	0.81	**0.83**
D17	80	86	88	86	89	90	92	93	96	91	**98**	0.67	0.86	0.82	0.85	0.87	0.88	0.82	0.85	0.88	0.98	**1**
D18	72	87	89	91	90	92.8	91	92.9	95	96	**98.9**	0.70	0.89	0.85	0.85	0.87	0.84	0.85	0.86	0.88	0.90	**0.93**
D19	77	86	87	89	91	92	93	92	95	96	**97**	0.72	0.89	0.86	0.87	0.88	0.92	0.91	0.92	0.90	0.88	**0.95**
D20	68	88	86	86	89	90	92	93	**95**	91	92	0.78	0.82	0.86	0.83	0.82	0.85	0.87	0.88	**0.89**	0.84	0.85
D21	62	78	78	80	81	80	80.7	82	83	84	**85.7**	0.67	0.82	0.85	0.84	0.85	0.87	0.88	0.88	0.87	0.89	**0.9**
D22	69.8	83	85	87	88	88	90	92	96	95	**98**	0.68	0.82	0.88	0.87	0.85	0.82	0.85	0.85	0.88	0.89	**0.91**
D23	68.9	80	81	80.88	83.76	87.78	87.71	87.9	**91**	90	90.1	0.68	0.72	0.809	0.846	0.80	0.82	**0.85**	0.83	0.82	0.80	0.82
D24	65	81	80	82	83	89	89.2	**91.3**	90	89.7	88	0.67	0.80	0.81	0.82	0.86	0.85	0.82	**0.89**	0.82	0.85	0.85
D25	69.9	86	88	86	89	97	92	93	96	97	**98.8**	0.77	0.88	0.81	0.86	0.88	0.88	0.89	0.89	0.89	0.88	**1**
D26	69.9	84	88.7	86.9	89.6	94.8	96.1	94	97	95	**97.9**	0.73	0.82	0.83	0.84	0.85	0.86	0.86	0.85	0.88	0.86	**0.92**
D27	67	82	85	86	81	90.1	91.9	93.6	97	95.8	**97.9**	0.71	0.82	0.86	0.88	0.89	0.89	0.87	0.85	0.86	0.87	**0.91**
D28	63	82	86	89	86	92	91	**98**	86	81	91	0.78	0.78	0.82	0.85	0.82	0.85	0.86	**0.89**	0.88	0.85	0.88
D29	67	78	82	85	86	82	87	**92**	85	84	89	0.77	0.80	0.81	0.84	0.86	0.88	0.87	0.86	**0.87**	0.85	0.88
D30	60	76	82	84	87	89	91	92	95	91	**98**	0.67	0.72	0.71	0.82	0.85	0.86	0.87	0.85	0.87	0.87	**1**



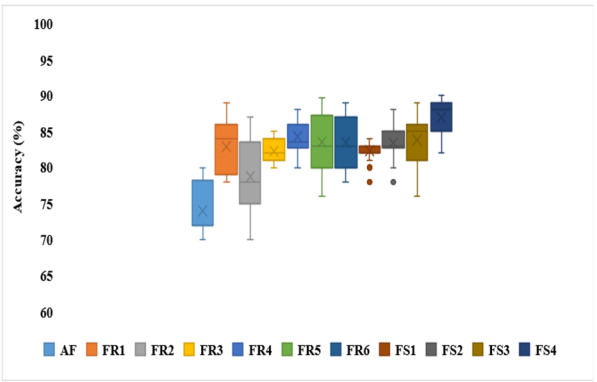
(a) SOM



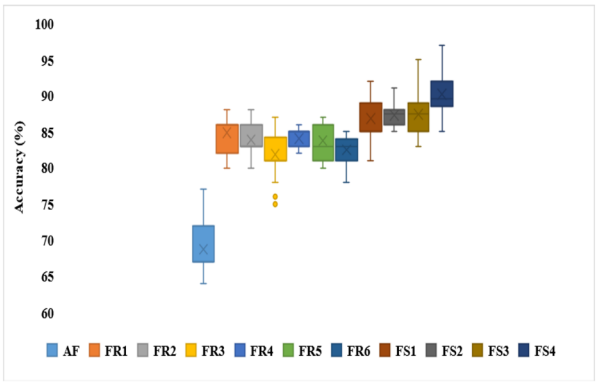
(b) Density based Clustering



(c) K-mean clustering



(d) Filter Clustering



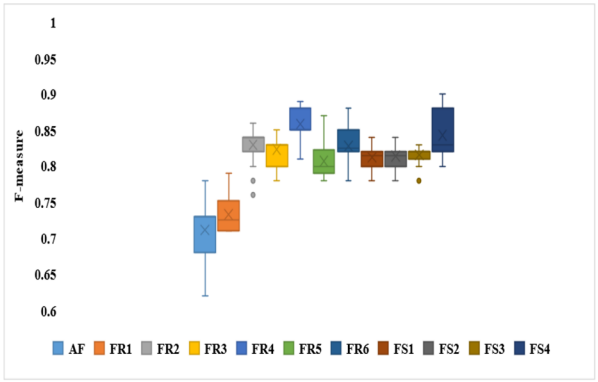
(e) Farthest First Clustering

Fig. 11 Box-plot diagram of accuracy

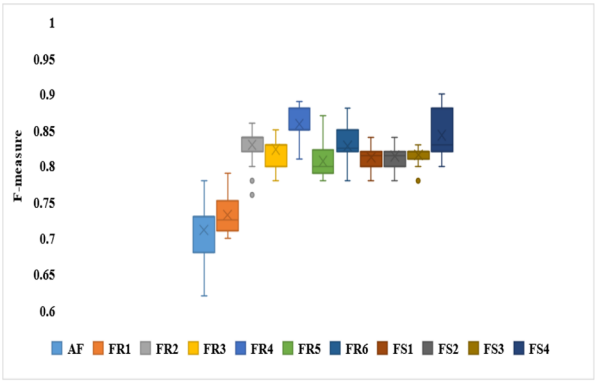
10.4 Comparison of results

To identify that out of implemented feature selection approaches and machine learning algorithms which technique work well or all of the techniques perform equally well, we employed pair-wise *t* test in our study.

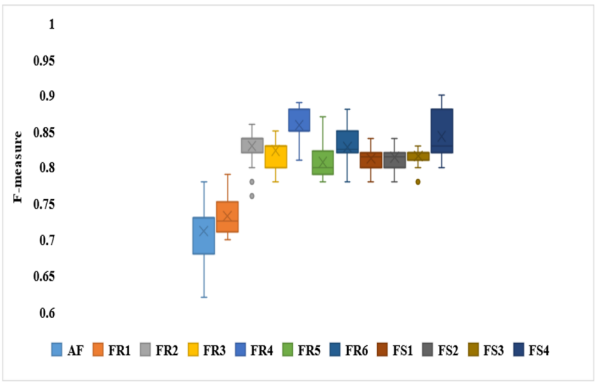
1. *Feature selection approaches* In this study, for each of the feature selection approaches two sets are formed, each of feature selection approach have 150 distinct data



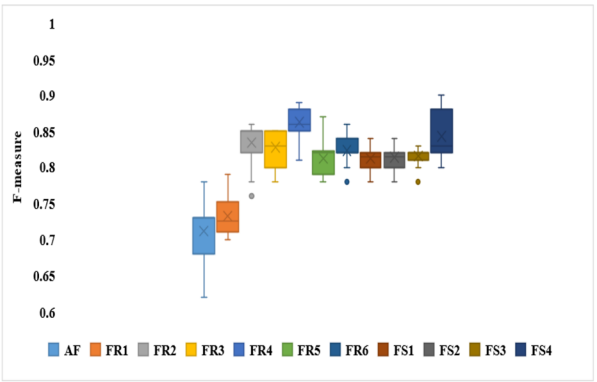
(a) SOM



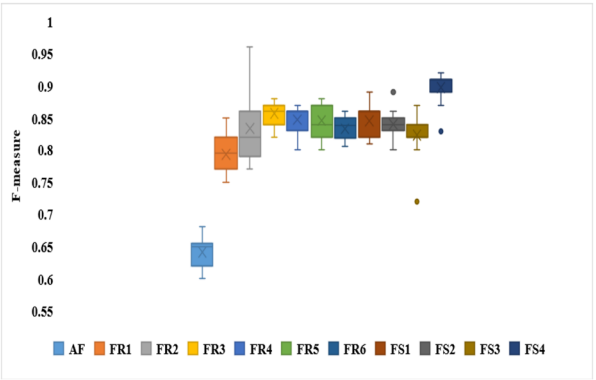
(b) Density based Clustering



(c) K-mean clustering



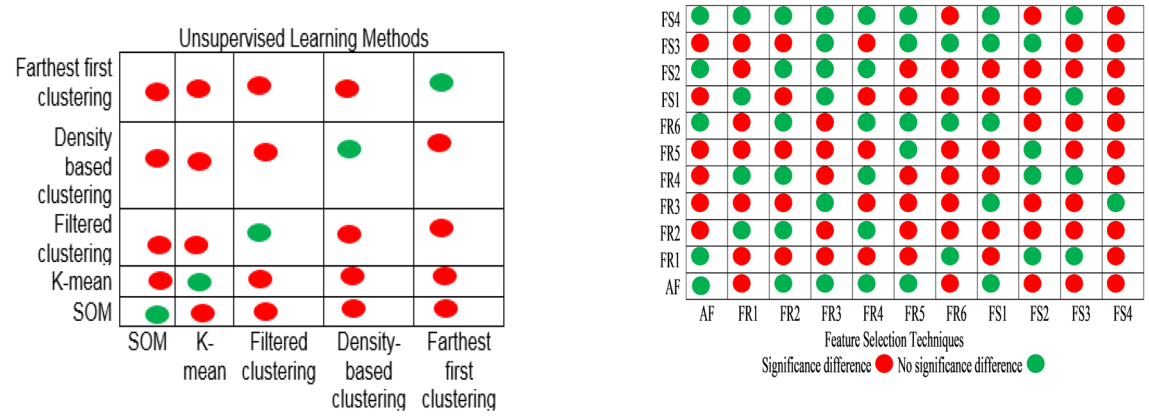
(d) Filter Clustering



(e) Farthest First Clustering

Fig. 12 Box-plot diagram of F-measure

points (5 machine learning techniques × 30 data set). *t* test is performed on distinct feature selection approaches and the respective p value to measure its statistical significance is compared. The outcome of *t* test study is demonstrated in Fig. 13b. In the figure, we used two different symbols to represent the p value i.e., circle filled with green color have p value > 0.05 (having no relevance difference) and circle filled with red color have p value ≤ 0.05 (relevance difference). After observing the Fig. 13b it is clear that, majority of the cells are filled with green color circle. This means that there is no relevance difference among the employed feature selection approaches. Further,



(a) Classification Methods (b) Feature Selection Techniques

Fig. 13 *t* test analysis (p value)

Table 16 Performance of distinct feature selection approaches after calculate its mean difference

Accuracy	AF	FR1	FR2	FR3	FR4	FR5	FR6	FS1	FS2	FS3	FS4
AF	0	− 1.9	− 0.96	− 0.78	− 1.91	− 1.90	− 4.89	− 1.77	− 1.80	− 0.87	− 5.8
FR1	1.8	0	0.77	0.87	− 0.78	− 0.80	− 3.8	0.07	0.32	0.80	− 3.89
FR2	0.87	− 0.78	0	0.5	− 2.0	− 2.0	− 3.89	− 0.9	− 0.32	0.20	− 4.54
FR3	0.67	− 0.68	− 0.2	0	− 1.32	− 1.32	− 4.08	− 0.8	− 0.45	0.07	− 4.88
FR4	1.88	0.77	1.22	1.36	0	0	− 2.99	0.77	0.8	1.7	− 3.66
FR5	1.88	0.77	1.22	1.36	0	0	− 2.99	0.75	0.8	1.7	− 3.66
FR6	4.5	3.88	3.22	4.09	2.88	2.88	0	3.55	3.67	4.19	− 0.50
FS1	1.65	− 0.09	0.61	0.77	− 0.80	− 0.81	− 3.88	0	0.21	0.80	− 3.98
FS2	1.09	− 0.29	0.39	0.51	− 0.9	− 0.9	− 3.81	− 0.22	0	0.8	− 4.21
FS3	0.87	− 0.88	− 0.21	− 0.09	− 1.8	− 1.8	− 3.8	− 0.08	− 0.7	0	− 4.89
FS4	**6.0**	**3.9**	**4.89**	**4.88**	**3.88**	**3.88**	**0.48**	**3.88**	**4.16**	**4.77**	**0**

by determining the measure of mean difference given in Table 16, we have observed that feature sets obtained by considering FS4 give best outcomes when examined with other implemented feature selection approaches.

In the present work, we also compare the developed model on the basis of cost-benefit analysis. For every feature selection approach, cost-benefit analysis is computed by employing following equation:

$$Cost\text{-}Benefit = (Based_{cost} + Benefit_{cost})/2. \tag{25}$$

Here, $Based_{cost}$ is dependent on the correlation among the selected features set and error in the class. $Based_{cost}$ can be calculated from the following equation:

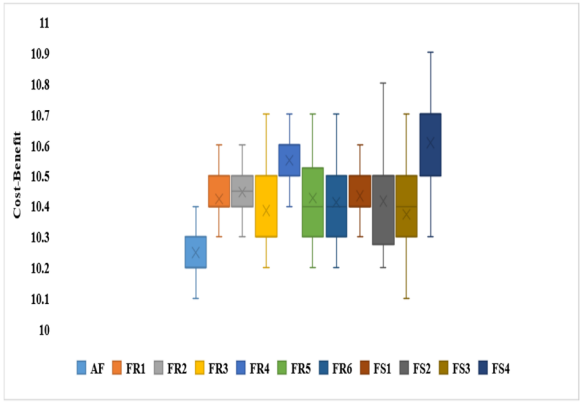
$$Based_{cost} = Accuracy\ (SM) \times \rho_{SM.fault}. \tag{26}$$

Here, *Accuracy* (*SM*) is the classification accuracy to build a malware detection model by utilizing selected features set, $\rho_{SM.fault}$ is a multiple correlation coefficient among selected features set and error. The proposed model produces higher accuracy and as it have high multiple correlation coefficient so it will achieve a high $Based_{cost}$. NAM is considered as feature sets and NSM is considered as the number of selected features after implementing features selection approaches. $Based_{cost}$ can be calculated from the following equation:

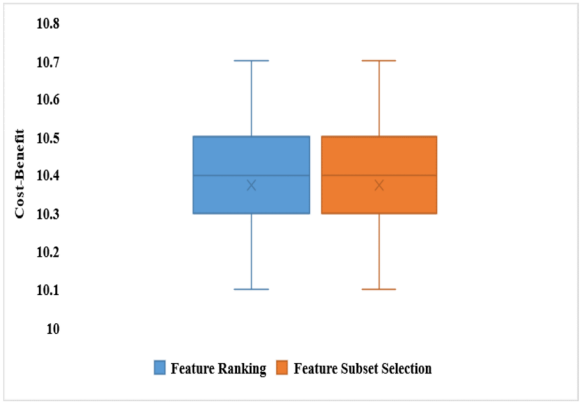
$$Based_{cost} = NAM - NSM/NAM \tag{27}$$

The feature selection approach which achieve higher value of cost-benefit is an foremost feature selection approach as proposed in [22]. Figure 14a, b demonstrates cost-benefit of distinct feature selection approaches. On the basis of Fig 14a, b we observed that FS4 achieved higher median Cost-benefit measure when matched with other approaches.

2. *Machine learning techniques* In our study, we implemented eleven different features subsets (i.e., 1 considering all features + 10 feature selection approaches) on thirty



(a) Feature Selection Approaches



(b) Feature subset selection and Feature ranking approaches

Fig. 14 Cost-benefit value

Table 17 Mean difference between performance of different Unsupervised methods

Accuracy	SOM	K-mean	Filter clustering	Density based clustering	Farthest first clustering
SOM	0	− 2.2	− 3.98	− 3.88	− 4.01
K-mean	1.86	0	− 2.81	− 3.10	− 4.89
Filter clustering	2.88	2.51	0	− 2.88	− 3.10
Density based clustering	1.86	2.88	2.01	0	− 4.89
Farthest first clustering	**5.77**	**3.78**	**2.99**	**2.77**	**0**

different Android app data set by examining four performance parameters i.e., intra- cluster, inter-cluster, F-measure and accuracy, all with 330 data points [(1 considering all set of features + 10 feature selection method) × 30 data sets)]. Figure 13a demonstrates the outcomes of *t* test analysis. On the basis of Fig. 13a, it is noticeable that, there is no relevance difference among these techniques because p value is smaller than 0.05. On the other hand, by determining the difference in their mean value as given in Table 17, Farthest first clustering gives best outcome when compared to other machine learning techniques.

3. *Feature subset selection and feature ranking approaches* For this study, pair-wise *t* test is used to identify which feature selection approach work better. For both of the implemented approaches (i.e., feature subset selection and feature ranking) sample pairs of performance evaluation are studied. The performance of averaged feature subset selection and feature ranking techniques outcomes of *t* test analysis are briefed in Table 18. In this research paper, five distinct kinds of machine learning algorithms are applied on thirty different Android categories by selecting Accuracy and F-measure as performance parameters, in accordance with each feature selection approaches an aggregate number of two sets are utilized. Feature subset selection with 360 distinct points (which means 4 feature subset selection approaches × 3 machine learning techniques × 30 data sets) and feature ranking with 540 distinct data points (3 machine learning techniques × 6 feature ranking approaches × 30 data sets). On the basis of Table 18, it is seen that, there isn't a relevant variation among two implemented approaches, because p value come out to be greater than 0.05. On the other side by calculating the mean difference value of feature subset selection approaches give best results when compare to feature ranking approaches. On the basis of *Cost-Benefit* analysis as demonstrated in Fig. 14, we can say that both feature subset selection and feature ranking have nearly similar *Cost-Benefit* value. It proves that the averaged cost and benefit of model build by considering selected set of features with feature subset selection approaches and feature ranking have nearly same value.

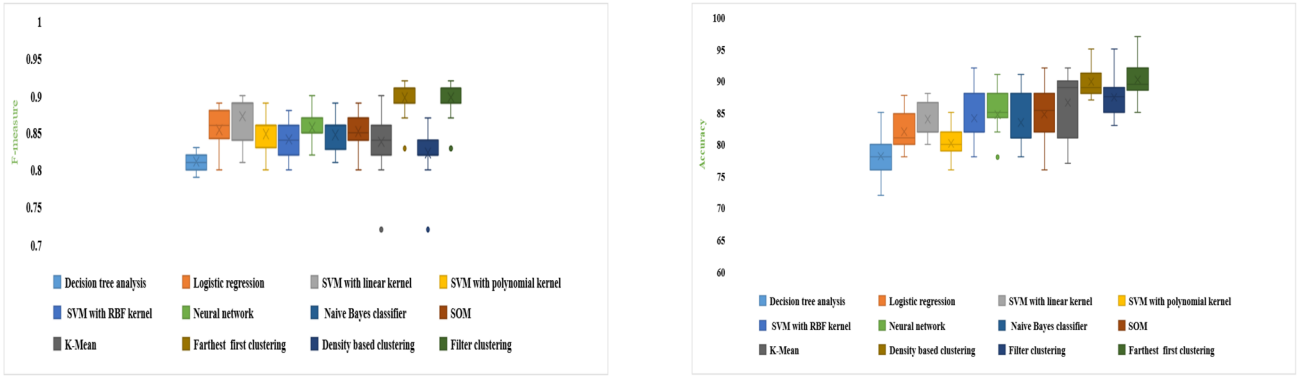


Fig. 15 Diagram of box-plot showing performance of different classifiers

Table 18 *t* test analysis among feature subset selection approaches and feature ranking approaches

Mean (FR-FS)	p value	t value
Accuracy		
− 0.1908	0.899	− 0.3211
F-measure		
− 0.0078	0.599	− 0.5251

10.5 Evaluation of proposed framework i.e., SemiDroid

10.5.1 Comparison of results with previously used classifiers

In addition to the study done in finding the best approach to build a malware detection model accurately, this study also makes the comparison with different most often used supervised machine learning approaches present in literature such as SVM with three distinct kernels i.e., linear, polynomial and RBF, decision tree analysis, logistic regression, neural network and Naïve Bayes classifier. Figure 15 demonstrates the box-plot diagrams for F-measure and accuracy of commonly utilized classifiers and five distinct machine learning algorithms implemented in this paper. On the basis of Fig. 15, we observed that farthest first clustering have higher median value along with some number of outliers.

Pair-wise *t* test is also implemented to decide which machine learning approach yield best performance. The outcomes of *t* test study for distinct machine learning approaches are demonstrated in Fig. 16. On the basis of Fig. 16 it is seen that, in number of the cases there is a relevance difference among these machine learning techniques because p value is smaller than 0.05. On the other hand by noticing the mean difference value in Table 19 it can be

Fig. 16 *t* test analysis (p value)

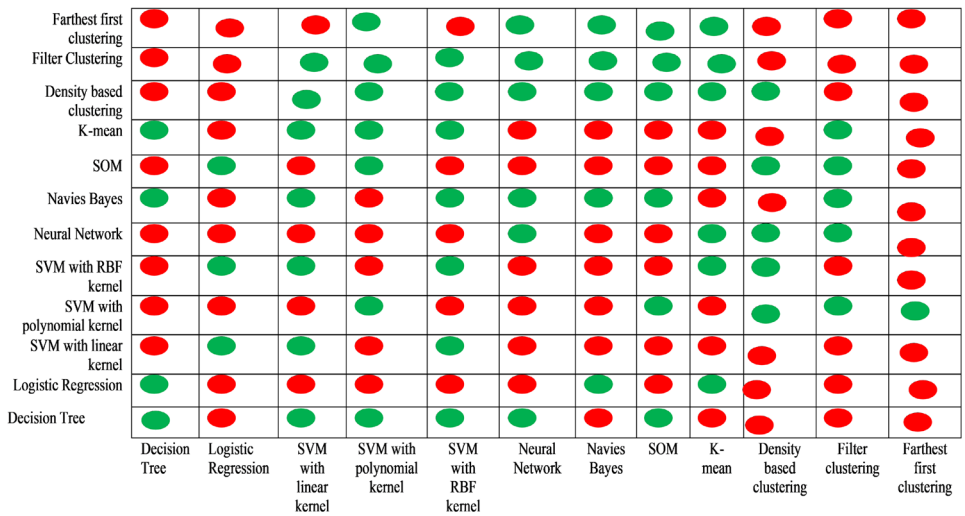


Table 19 Mean difference between performance of different supervised machine learning technique

Accuracy	Decision tree analysis	Logistic regression	SVM with linear kernel	SVM with polynomial kernel	SVM with RBF kernel	Neural network	Naïve Bayes classifier	SOM	K-mean	Density clustering	Filter clustering	Farthest clustering
Decision tree analysis	0	6.77	6.41	9.81	5.87	5.99	10.21	2.10	− 1.11	− 2.88	− 5.77	− 11.77
Logistic regression	− 6.88	0	− 0.88	− 4.88	2.88	− 1.89	3.88	− 4.99	− 5.11	− 5.88	− 8.11	− 9.88
SVM with linear kernel	− 6.55	0.77	0	− 4.81	3.87	− 0.89	3.21	− 4.10	− 5.77	− 6.17	− 7.8	− 9.41
SVM with polynomial kernel	− 2.88	4.71	4.41	0	7.27	3.29	8.01	− 0.19	− 2.77	− 3.14	− 3.41	− 5.28
SVM with RBF kernel	− 9.8	− 2.77	− 3.41	− 7.81	0	− 4.89	0.29	− 7.10	− 7.99	− 10.77	− 11.01	− 12.78
Neural network	− 5.88	1.77	0.41	− 4.81	4.17	0	4.21	− 3.10	− 5.11	− 6.10	− 7.11	− 8.88
Naïve Bayes classifier	− 10.78	− 3.77	− 3.41	− 8.10	− 0.87	− 4.99	0	− 8.10	− 8.88	− 10.1	− 11.1	− 12.80
SOM	− 2.10	4.97	4.10	0.81	7.87	3.99	8.21	0	− 2.97	− 3− 19	− 3.91	− 5.02
K-mean	1.99	8.17	7.17	3.81	11.71	7.09	12.1	3.03	0	− 1.88	− 2.11	− 3.11
Density based clustering	2.99	3.17	5.17	4.81	7.71	6.09	10.1	6.03	5.77	0	− 2.11	− 3.11
Filter clustering	2.88	5.17	6.17	7.81	7.71	6.09	10.1	1.03	1.77	0.77	0	− 2.0
Farthest first clustering	**4.99**	**9.77**	**9.41**	**5.81**	**12.87**	**8.99**	**13.21**	**4.90**	**2.99**	**1.99**	**0.99**	**0**

Table 20 Comparison with previously developed frameworks/approaches

Framework/approach	Goal	Methodology	Deployment	Data set	Detection rate	Labelled data set used
Andromaly [62]	Detection	Dynamic and profile-based	Distributed	Very-limited	High	100%
AndroSimilar [30]	Detection	Static	Off-device	Limited	Moderate	100%
Andrubis [43]	Analysis and detection	Static, dynamic, profile-based and behavioural	Off-device	Higher	Moderate	100%
Aurasium [76]	Detection	Dynamic and behavioural	Off-device	Limited	High	100%
CopperDroid [66]	Analysis and detection	Dynamic, system/API and VMI	Off-device	Limited	Moderate	100%
Crowdroid [18]	Detection	Dynamic, system call/API and behavioural	Distributed	Very-limited	High	100%
Paranoid Android [58]	Detection	Dynamic and behavioural	Off-device	Limited	–	100%
TaintDroid [29]	Detection	Dynamic system call/API and behavioural	Off-device	Very-limited	Moderate	100%
HinDroid [35]	Detection	Dynamic and API	Off-device	Limited	Moderate	100%
Mahindru and Singh [50]	Detection	Dynamic	Off-device	Limited	Moderate	100%
MalDozer [39]	Detection	Dynamic	Off-device	Limited	Moderate	100%
HEMD [81]	Detection	Dynamic and permissions	Off-device	Limited	Moderate	100%
DroidDet [82]	Detection	Static	Off-device	Limited	Moderate	100%
Wei Wang [71]	Detection	Dynamic	Off-device	Limited	Moderate	100%
MalInsight [33]	Detection	Dynamic	Off-device	Limited	High	100%
DeepDroid [45]	Detection	Dynamic	Off-device	Limited	Moderate	100%
PerbDroid [49]	Detection	Dynamic	Off-device	Limited	High	100%
Mahindru and Sangal [47]	Detection	Dynamic	Off-device	Limited	High	100%
SemiDroid (our proposed framework)	Detection	Dynamic,permissions, API calls, user-rating and Number of user download app	Off-device	Unlimited	Higher	No labelled data set is used

Detection rate of our proposed malware detection model (i.e., SemiDroid) is higher when compared to distinct frameworks/approaches available in the literature. Frameworks/approaches proposed in the literature developed and tested with limited data set. Experiments were performed on Drebin data set [10] and empirical result reveals that our proposed framework has achieved 2% higher detection rate when compared to distinct frameworks available in the literature with unlabelled data set

seen that farthest first clustering achieved better results when compared to other supervised machine learning techniques.

In addition to that, in our study we compare our proposed malware detection model (i.e., SemiDroid) with existing frameworks or approaches that were developed in the literature. Table 20 shows the name, goal, methodology, deployment, data set and detection rate of suggested approaches or frameworks.

10.5.2 Comparison of results with different anti-virus scanners

Although farthest first clustering gives a better performance as compared to the machine learning technique used in the literature, in the end it must be comparable with the common anti-virus products available in practice for Android malware detection. For this experiment, we select 10 different anti-viruses which are available in the market and applied them on our collected data set. The performance of proposed framework is comparatively better than many of the anti-viruses available in the experiment. Table 21 shows the results of the experiment with anti-virus scanners. The detection rate of the anti-viruses scanners varies considerably. Also the best anti-virus scanners detected 96.2% of the Android malwares and certain scanners identified only 82% of the malicious samples, likely do not being specialized in detecting Android malware. By using 1000 Android apps, our proposed framework i.e., SemiDroid gives the detection

Table 21 Comparison of proposed framework i.e., SemiDroid with distinct anti-virus scanners

Name of the anti-virus	Averaged detection rate (in %)	Speed to detect malware in sec
Cyren	82	60
Ikarus	82.68	62
VIPRE	89	40
McAfee	89	30
AVG	90	32
AVware	92.8	30
ESET NOD32	92.9	20
CAT QuickHeal	95.8	32
AegisLab	96.1	30
NANO Antivirus	96.2	20
SemiDroid (our proposed approach)	98.8	12

Detection speed calculated on Android apps whose size is less or equivalent to 50MB. To compare the performance of SemiDroid we consider freely available anti-virus in the market. Speed is measured for a particular Android app taken from real-world. To see the effect of obfuscation and polymorphism techniques on the malware detection process. We consider 1000 distinct Android apps collected from Google play store and third party app store having the same package name

rate of 98.8% and outperforms 1 out of 10 anti-virus scanners. From this, we can say that our proposed framework is more efficient in detecting malware when compared to distinct anti-virus scanners.

10.5.3 Detection of known and unknown malware families

Detection of known malware families In this section, we check that our proposed framework is capable to detect malware of known family or not. For this experiment, we select 20 sample of each families (in our study, we consider sample of 81 different families shown in Table 22.) and train it with our selected model. Farthest first clustering is capable to detect average 98.8% malware apps. The name of families and the samples used for each family can be found in Table 22 and the detection rate of our proposed framework for each family is illustrated in Fig. 17a, b.

Detection of unknown malware families To check whether the farthest first clustering is capable to detect unknown malware families or not, we trained, our proposed framework with the random selection of 10 different families obtained by principle of counting and test is applied on the rest of the remaining 71 families present in the data set. Table 23 shows the result of farthest first clustering when we train it with 10 selected families. From Table 23, we can say that if we train farthest first clustering with few number of known families samples which are necessary to generalize the behavior of most malware families it gives better detection rate.

In summary, our proposed framework is capable to detect Android malware more effectively when compared with several anti-virus scanners which regularly update their signature definition. In addition, our proposed framework is capable to identify Android malware more efficiently whenever we trained with limited number of malware families.

10.5.4 Experimental findings

The comprehensive conclusion of our experimental work is presented in this section. The empirical study was conducted for thirty different categories of Android apps by considering five different unsupervised machine learning techniques i.e., SOM, K-mean, filter clustering, density based clustering and farthest first clustering. On the basis of the experimental results, this research paper is able to answer the questions mentioned in Sect. 2.

RQ1. In this paper, we applied five distinct machine learning algorithms to build a model which help us to detect whether an app is benign or malware. On the basis of Tables 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15, it can be implicit that model build by employing farthest first clustering by using selected set of features obtained as a result of FS4 as an input gives better outcome when compared to others.

RQ2. To respond the RQ2, Fig. 17 and Tables 20 and 21 were analyzed. Here, it is found that model build by utilizing farthest first clustering is capable to detect malware from real-world apps.

RQ3. In the present paper, four distinct kind of feature subset selection approaches and six distinct kind of feature ranking approaches are used to identify the smaller subset of features. By utilizing these approaches, we considered best possible subsets of the features which help us to build a model to identify that either an app is benign or malware. On the basis of the Tables 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15, in number of cases there occurs a reduced subset of features which are best for building a detection model when compared to all the extracted features.

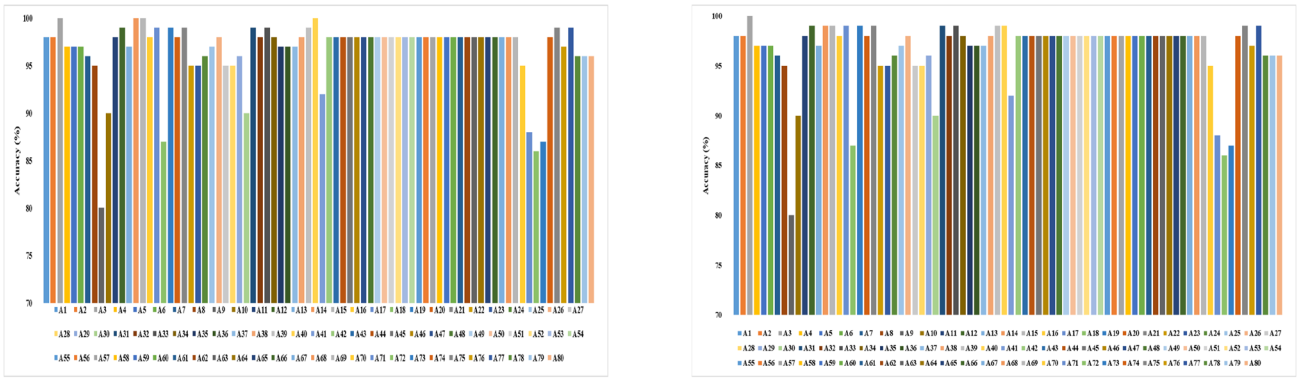
RQ4. In the present paper, six distinct variants of feature ranking approaches are used to discover the reduced subset of features. On the basis of *t* test study, it is seen that feature selection by implementing PCA approach gives the better outcomes when matched to others approaches.

RQ5, For this paper, four distinct kind of feature subset selection approaches are used to find the reduced subset of features. On the basis of *t* test study, it is seen that feature selection by utilizing FS4 gives the outcomes which are persuasively better when compared to other approaches.

RQ6. For this work, pair-wise *t* test being utilized to identify whether feature subset selection approaches perform better than feature ranking approaches or both of them carried out equally well. On the basis of *t* test outcomes it is seen that, there is a relevance difference among feature

Table 22 Top malware families used in our data set

ID	Family	# of samples	ID	Family	# of samples	ID	Family	# of samples
A1	Airpush	150	A2	AndroRAT	140	A3	Andup	300
A4	Aples	120	A5	BankBot	100	A6	Bankun	133
A7	Boqx	130	A8	Boxer	122	A9	Cova	100
A10	Dowgin	100	A11	DroidKungFu	100	A12	Erop	120
A13	FakeAngry	110	A14	FakeAV	120	A15	FakeDoc	120
A16	FakeInst	110	A17	FakePlayer	120	A18	FakeTimer	120
A19	FakeUpdates	120	A20	Finspy	111	A21	Fjcon	123
A22	Fobus	102	A23	Fusob	181	A24	GingerMaster	192
A25	GoldDream	20	A26	Gorpo	120	A27	Gumen	20
A28	Jisut	62	A29	Kemoge	720	A30	Koler	200
A31	Ksapp	290	A32	Kuguo	100	A33	Kyview	500
A34	Leech	300	A35	Lnk	100	A36	Lotoor	20
A37	Mecor	29	A38	Minimob	330	A39	Mmarketpay	200
A40	MobileTX	500	A41	Mseg	230	A42	Mtk	200
A43	Nandrobox	100	A44	Obad	100	A45	Opfake	120
A46	Penetho	120	A47	Ramnit	120	A48	Roop	120
A49	RuMMS	100	A50	SimpleLocker	110	A51	SlemBunk	120
A52	SmsKey	120	A53	SMsZombie	110	A54	Spambot	115
A55	SpyBubble	120	A56	Stealer	300	A57	Steek	230
A58	Svpeng	20	A59	Tesbo	21	A60	Triada	200
A61	Univert	210	A62	UpdtKiller	100	A63	Utchi	300
A64	Vidro	92	A65	VikingHorde	230	A66	Vmvol	533
A67	Winge	190	A68	Youmi	689	A69	Zitmo	230
A70	Ztorg	1000	A71	Imlog	50	A72	SMSreg	50
A73	Gappusin	50	A74	Adrd	50	A75	Geinimi	100
A76	Kmin	157	A77	Plankton	125	A78	GingerMaster	100
A79	Iconosys	100	A80	SendPay	18	A81	GoldDream	200



(a) (b)

Fig. 17 Detection rate of proposed framework farthest first clustering

subset selection and feature ranking approach. Moreover, the value of mean difference shows that feature subset selection approaches gives better results than the feature ranking.

RQ7. On the basis of Sect. 9, we can observe that the performance of the feature selection approaches vary by using the distinct machine learning techniques. Further, it also observed that selection of machine learning algorithm to build a malware detection model which detect either the app is malware or not is based on the feature selection approaches.

Table 23 Detection of SemiDroid to detect unknown malware families

Combination of Android malware families to trained the model	Detection rate when trained farthest first clustering
{**A1**, A2, A3, A4, A5, A6, A7, A8, A9, A10}	66%
{**A1**, A3, A4, A5, A6, A7, A8, A8, A10, A11}	70%
:	:
:	:
{**A2**, A3, A4, A5, A6, A7, A8, A9, A10, A11}	59%
:	:
:	:
:	:
:	:
:	:
{**A7**, A13, A42, A55, A67, A37, A68, A79, A22, A51}	98.4%
:	:
:	:
:	:
:	:
:	:

11 Threat to validity

In this section, threats to validity which are experienced at the time of performing the experiment are presented. Below we discuss them:

(i) *Construct validity* In this work, presented models for malware detection only detect either an app is benign or malware, but does not state that how many number of possible permissions and API calls are required to detect malware.

(ii) *External validity* Cyber-criminals develops malware on daily basis to misuse the user information. In this work, we considered 81 different malware families to train the model and our proposed model is capable to detect malware from known and unknown families. Further, research can be extended to train model with more malware families and which is capable to detect more malware apps from real-world.

(iii) *Internal validity* The threat lies in the consistency of the data used in this study. We collected data from different sources mentioned in Sect. 4. Any error in the information not mentioned in the sources was not considered in this work. Also, we can not claim that the data considered for the experiment is 100% accurate, we believed that it has been collected consistently.

12 Conclusion

This work emphasizes on designing a malware detection framework by using selected set of features which help us to identify that an Android app belongs to malware class or benign class. The execution process was performed by taking assistance of thirty different categories of Android apps.

Empirical results indicate that, it is feasible to determine a small subset of features. The malware detection model build by considering this determined set of features is able to detect malware and benign apps with inferior value of misclassified errors and better accuracy. Further, it is also seen that the results of malware detection model, is influenced by the feature selection approaches.

After performing in depth analysis, we found that AA, BU, LS, PE, RA, TO set of features are relevance detectors for malware detection by utilizing feature selection approaches. Further, on the basis of mean difference, it is seen that model build by considering selected set of features as an input gives better detection rate when compared to model build by considering all set of extracted features. Moreover, the model build by utilizing Farthest first clustering gives better outcomes when compared to other techniques.

At last, on the basis of *Cost-benefit* analysis, we implicit that the selected features by utilizing FS4, achieved high median *Cost-Benefit* value when compared to other

approaches and it is also seen that model build by utilizing Farthest first clustering is capable to detect 98.8% known and unknown malware from real-world apps.

In this work, proposed models for malware detection only detect that either the app is malware or benign. Further, work can be extended to develop a model for malware detection which predict whether a particular feature is capable to detect malware or not. Moreover, this study can be replicated over other Android apps repository which utilized soft computing models to attain better detection rate for malware.

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, Springer, pp 86–103
2. Abawajy J, Kelarev A (2017) Iterative classifier fusion system for the detection of android malware. IEEE Transactions on Big Data
3. Alam MS, Vuong ST (2013) Random forest classification for detecting android malware. In: 2013 IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing, IEEE, pp 663–669
4. Alazab M, Alazab M, Shalaginov A, Mesleh A, Awajan A (2020) Intelligent mobile malware detection using permission requests and API calls. Future Gener Comput Syst 107:509–521
5. Almin SB, Chatterjee M (2015) A novel approach to detect android malware. Procedia Comput Sci 45:407–417
6. Alzaylaee MK, Yerima SY, Sezer S (2020) DL-droid: deep learning based android malware detection using real devices. Comput Secur 89:101663
7. Amos B, Turner H, White J (2013) Applying machine learning classifiers to dynamic android malware detection at scale. In: 2013 9th international wireless communications and mobile computing conference (IWCMC), IEEE, pp 1666–1671
8. Andriatsimandefitra R, Tong VVT (2015) Detection and identification of android malware based on information flow monitoring. In: 2015 IEEE 2nd international conference on cyber security and cloud computing, IEEE, pp 200–203
9. Arora A, Peddoju SK, Conti M (2019) Permpair: Android malware detection using permission pairs. IEEE Trans Inf Forensics Secur 15:1968–1982
10. Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K, Siemens C (2014) Drebin: effective and explainable detection of android malware in your pocket. NDSS 14:23–26
11. Attar AE, Khatoun R, Lemercier M (2014) A gaussian mixture model for dynamic detection of abnormal behavior in smartphone applications. In: 2014 global information infrastructure and networking symposium (GIIS), IEEE, pp 1–6
12. Babaagba KO, Adesanya SO (2019) A study on the effect of feature selection on malware analysis using machine learning. In: Proceedings of the 2019 8th international conference on educational and information technology, pp 51–55
13. Barrera D, Kayacik HG, Oorschot PCV, Somayaji A (2010) A methodology for empirical analysis of permission-based security models and its application to android. In: Proceedings of the 17th ACM conference on computer and communications security, pp 73–84
14. Bibi KF, Banu MN (2015) Feature subset selection based on filter technique. In: 2015 international conference on computing and communications technologies (ICCCT), IEEE, pp 1–6
15. Birendra C (2016) Android permission model. arXiv preprint arXiv:160704256
16. Blair DC (1979) Information retrieval, 2nd ed. C. J. van Rijsbergen. J Am Soc Inf Sci 30(6):374–375. https://doi.org/10.1002/asi.4630300621. https://ideas.repec.org/a/bla/jamest/v30y1979i6p374-375.html
17. Blessie EC, Karthikeyan E (2012) Sigmis: a feature selection algorithm using correlation based method. J Algorithms Comput Technol 6(3):385–394
18. Burguera I, Zurutuza U, Nadjm-Tehrani S (2011) Crowdroid: behavior-based malware detection system for android. In: Proceedings of the 1st ACM workshop on security and privacy in smartphones and mobile devices, pp 15–26
19. Cai H, Meng N, Ryder B, Yao D (2018) Droidcat: effective android malware detection and categorization via app-level profiling. IEEE Trans Inf Forensics Secur 14(6):1455–1470
20. Canbek G, Baykal N, Sagiroglu S (2017) Clustering and visualization of mobile application permissions for end users and malware analysts. In: 2017 5th international symposium on digital forensic and security (ISDFS), IEEE, pp 1–10
21. Caviglione L, Gaggero M, Lalande JF, Mazurczyk W, Urbański M (2015) Seeing the unseen: revealing mobile malware hidden communications via energy consumption and artificial intelligence. IEEE Trans Inf Forensics Secur 11(4):799–810
22. Chaikla N, Qi Y (1999) Genetic algorithms in feature selection. In: IEEE SMC'99 conference proceedings. 1999 IEEE international conference on systems, man, and cybernetics (Cat. No. 99CH37028), IEEE, vol 5, pp 538–540
23. Chen PS, Lin SC, Sun CH (2015) Simple and effective method for detecting abnormal internet behaviors of mobile devices. Inf Sci 321:193–204
24. Chen Y, Tu L (2007) Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, pp 133–142
25. Cruz AEC, Ochimizu K (2009) Towards logistic regression models for predicting fault-prone code across software projects. In: 2009 3rd international symposium on empirical software engineering and measurement, IEEE, pp 460–463
26. Cui B, Jin H, Carullo G, Liu Z (2015) Service-oriented mobile malware detection system based on mining strategies. Pervas Mobile Comput 24:101–116
27. Dixon B, Mishra S (2013) Power based malicious code detection techniques for smartphones. In: 2013 12th IEEE international conference on trust, security and privacy in computing and communications, IEEE, pp 142–149
28. Enck W, Ongtang M, McDaniel P (2009) On lightweight mobile phone application certification. In: Proceedings of the 16th ACM conference on computer and communications security, pp 235–245
29. Enck W, Gilbert P, Han S, Tendulkar V, Chun BG, Cox LP, Jung J, McDaniel P, Sheth AN (2014) Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans Comput Syst (TOCS) 32(2):1–29
30. Faruki P, Ganmoor V, Laxmi V, Gaur MS, Bharmal A (2013) Androsimilar: robust statistical feature signature for android malware detection. In: Proceedings of the 6th international conference on security of information and networks, pp 152–159
31. Fung CJ, Lam DY, Boutaba R (2014) Revmatch: An efficient and robust decision model for collaborative malware detection. In: 2014 IEEE network operations and management symposium (NOMS), IEEE, pp 1–9

32. Guo DF, Sui AF, Shi YJ, Hu JJ, Lin GZ, Guo T (2014) Behavior classification based self-learning mobile malware detection. JCP 9(4):851–858
33. Han W, Xue J, Wang Y, Liu Z, Kong Z (2019) Malinsight: a systematic profiling based malware detection framework. J Netw Comput Appl 125:236–250
34. Holland B, Deering T, Kothari S, Mathews J, Ranade N (2015) Security toolbox for detecting novel and sophisticated android malware. In: 2015 IEEE/ACM 37th IEEE international conference on software engineering, IEEE, vol 2, pp 733–736
35. Hou S, Ye Y, Song Y, Abdulhayoglu M (2017) Hindroid: an intelligent android malware detection system based on structured heterogeneous information network. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1507–1515
36. Jerbi M, Dagdia ZC, Bechikh S, Said LB (2020) On the use of artificial malicious patterns for android malware detection. Comput Secur 92:101743
37. Jouve PE, Nicoloyannis N (2005) A filter feature selection method for clustering. In: International symposium on methodologies for intelligent systems, Springer, pp 583–593
38. Kadir AFA, Stakhanova N, Ghorbani AA (2015) Android botnets: What URLs are telling us. In: International conference on network and system security, Springer, pp 78–91
39. Karbab EB, Debbabi M, Derhab A, Mouheb D (2018) Maldozer: automatic framework for android malware detection using deep learning. Digit Investig 24:S48–S59
40. Kohavi R, John GH et al (1997) Wrappers for feature subset selection. Artificial intelligence 97(1–2):273–324
41. Kumar M, et al. (2013) An optimized farthest first clustering algorithm. In: 2013 Nirma University international conference on engineering (NUiCONE), IEEE, pp 1–5
42. Lee WY, Saxe J, Harang R (2019) Seqdroid: obfuscated android malware detection using stacked convolutional and recurrent neural networks. In: Deep Learning applications for cyber security, Springer, pp 197–210
43. Lindorfer M, Neugschwandtner M, Weichselbaum L, Fratantonio Y, Veen VVD, Platzer C (2014) Andrubis–1,000,000 apps later: a view on current android malware behaviors. In: 2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS), IEEE, pp 3–17
44. Ma Z, Ge H, Liu Y, Zhao M, Ma J (2019) A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE Access 7:21235–21245
45. Mahindru A, Sangal A (2019) Deepdroid: feature selection approach to detect android malware using deep learning. In: 2019 IEEE 10th international conference on software engineering and service science (ICSESS), IEEE, pp 16–19
46. Mahindru A, Sangal A (2020a) Feature-based semi-supervised learning to detect malware from android. Automated software engineering: a deep learning-based approach. Springer, Berlin, pp 93–118
47. Mahindru A, Sangal A (2020b) Feature-based semi-supervised learning to detect malware from android. Automated software engineering: a deep learning-based approach. Springer, Berlin, pp 93–118
48. Mahindru A, Sangal A (2020a) Gadroid: a framework for malware detection from android by using genetic algorithm as feature selection approach. Int J Adv Sci Technol 29(5):5532–5543
49. Mahindru A, Sangal A (2020b) Perbdroid: effective malware detection model developed using machine learning classification techniques. A journey towards bio-inspired techniques in software engineering. Springer, Berlin, pp 103–139
50. Mahindru A, Singh P (2017) Dynamic permissions based android malware detection using machine learning techniques. In: Proceedings of the 10th innovations in software engineering conference, pp 202–210
51. Martinelli F, Mercaldo F, Saracino A (2017) Bridemaid: an hybrid tool for accurate detection of android malware. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security, pp 899–901
52. Milosevic N, Dehghantanha A, Choo KKR (2017) Machine learning aided android malware classification. Comput Electr Eng 61:266–274
53. Narudin FA, Feizollah A, Anuar NB, Gani A (2016) Evaluation of machine learning classifiers for mobile malware detection. Soft Comput 20(1):343–357
54. Ng DV, Hwang JIG (2014) Android malware detection using the dendritic cell algorithm. In: 2014 international conference on machine learning and cybernetics, IEEE, vol 1, pp 257–262
55. Novakovic J (2010) The impact of feature selection on the accuracy of naïve bayes classifier. In: 18th telecommunications forum TELFOR, vol 2, pp 1113–1116
56. Pawlak Z (1982) Rough sets. Int J Comput Inf Sci 11(5):341–356
57. Plackett RL (1983) Karl pearson and the chi-squared test. International Statistical Review/Revue Internationale de Statistique 59–72
58. Portokalidis G, Homburg P, Anagnostakis K, Bos H (2010) Paranoid android: versatile protection for smartphones. In: Proceedings of the 26th annual computer security applications conference, pp 347–356
59. 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, IEEE, pp 877–883
60. Rahman M (2013) Droidmln: a markov logic network approach to detect android malware. In: 2013 12th international conference on machine learning and applications, IEEE, vol 2, pp 166–169
61. Rahman SSMM, Saha SK (2018) Stackdroid: evaluation of a multi-level approach for detecting the malware on android using stacked generalization. In: International conference on recent trends in image processing and pattern recognition, Springer, pp 611–623
62. Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y (2012) "Andromaly": a behavioral malware detection framework for android devices. J Intell Inf Syst 38(1):161–190
63. Sheen S, Anitha R, Natarajan V (2015) Android based malware detection using a multifeature collaborative decision fusion approach. Neurocomputing 151:905–912
64. Shen T, Zhongyang Y, Xin Z, Mao B, Huang H (2014) Detect android malware variants using component based topology graph. In: 2014 IEEE 13th international conference on trust, security and privacy in computing and communications, IEEE, pp 406–413
65. Suarez-Tangil G, Tapiador JE, Peris-Lopez P, Pastrana S (2015) Power-aware anomaly detection in smartphones: an analysis of on-platform versus externalized operation. Pervas Mobile Comput 18:137–151
66. Tam K, Khan SJ, Fattori A, Cavallaro L (2015) Copperdroid: automatic reconstruction of android malware behaviors. In: Ndss
67. Tong F, Yan Z (2017) A hybrid approach of mobile malware detection in android. J Parallel Distrib Comput 103:22–31
68. Tramontana E, Verga G (2019) Mitigating privacy-related risks for android users. In: 2019 IEEE 28th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE), IEEE, pp 243–248
69. Vinayakumar R, Alazab M, Soman K, Poornachandran P, Venkatraman S (2019) Robust intelligent malware detection using deep learning. IEEE Access 7:46717–46738
70. Wang W, Wang X, Feng D, Liu J, Han Z, Zhang X (2014) Exploring permission-induced risk in android applications for

malicious application detection. IEEE Trans Inf Forensics Secur 9(11):1869–1882
71. Wang W, Zhao M, Wang J (2019) Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. J Ambient Intell Humaniz Comput 10(8):3035–3043
72. 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, Springer, pp 252–276
73. Wei TE, Mao CH, Jeng AB, Lee HM, Wang HT, Wu DJ (2012) Android malware detection via a latent network behavior analysis. In: 2012 IEEE 11th international conference on trust, security and privacy in computing and communications, IEEE, pp 1251–1258
74. Wu DJ, Mao CH, Wei TE, Lee HM, Wu KP (2012) Droidmat: Android malware detection through manifest and API calls tracing. In: 2012 seventh Asia joint conference on information security, IEEE, pp 62–69
75. Xiao X, Zhang S, Mercaldo F, Hu G, Sangaiah AK (2019) Android malware detection based on system call sequences and LSTM. Multimed Tools Appl 78(4):3979–3999
76. Xu R, Saïdi H, Anderson R (2012) Aurasium: practical policy enforcement for android applications. In: Presented as part of the 21st {USENIX} security symposium ({USENIX} Security 12), pp 539–552
77. Yang L, Ganapathy V, Iftode L (2011) Enhancing mobile malware detection with social collaboration. In: 2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing, IEEE, pp 572–576
78. Yewale A, Singh M (2016) Malware detection based on opcode frequency. In: 2016 international conference on advanced communication control and computing technologies (ICACCCT), IEEE, pp 646–649
79. Yuxin D, Siyi Z (2019) Malware detection based on deep learning algorithm. Neural Comput Appl 31(2):461–472
80. Zhou Y, Jiang X (2012) Dissecting android malware: characterization and evolution. In: 2012 IEEE symposium on security and privacy, IEEE, pp 95–109
81. Zhu HJ, Jiang TH, Ma B, You ZH, Shi WL, Cheng L (2018) Hemd: a highly efficient random forest-based malware detection framework for android. Neural Comput Appl 30(11):3353–3361
82. Zhu HJ, You ZH, Zhu ZX, Shi WL, Chen X, Cheng L (2018b) Droiddet: effective and robust detection of android malware using static analysis along with rotation forest model. Neurocomputing 272:638–646