Indonesian Journal of Electrical Engineering and Computer Science
Vol. 19, No. 1, July 2020, pp. 543~552
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v19.i1.pp543-552

Exploring permissions in android applications using ensemble-based extra tree feature selection

Howida Abubaker, Aida Ali, Siti Mariyam Shamsuddin, and Shafaatunnur Hassan
Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia (UTM), Malaysia

Article Info

Article history:

Received Nov 3, 2019
Revised Jan 19, 2020
Accepted Feb 1, 2020

Keywords:

Feature selection
Machine learning
Malware android classification
Permission-based analysis

ABSTRACT

The fast development of mobile apps and its usage has led to an increase the risk of exploiting user privacy. One method used in the Android security mechanism is permission control that restricts the access of apps to core facilities of devices. However, that permissions could be exploited by attackers when granting certain combinations of permissions. So, this paper aims to explore the pattern of malware apps based on analyzing permissions by proposing a framework utilizing feature selection based on ensemble extra tree classifier method and machine learning classifier. The used dataset had 25458 samples (8643 malware apps & 16815 benign apps) with 173 features. Three datasets with 25458 samples and 5, 10 and 20 features respectively were generated after using the proposed feature selection method. All the dataset was fed to machine learning. Support Vector Machine (SVM), K Neighbors Classifier, Decision Tree, Naïve Bayes and Multilayer Perceptron (MLP) classifiers were used. The classifiers models were evaluated using true negative rate (TNR), false positive rate (FNR) and accuracy metrics. The experimental results obtained showed that Support Vector Machine and KNeighbors Classifiers with 20 features achieved the highest accuracy with 94 % and TNR with a rate of 89 % using the KNeighbors Classifier. The FNR rate is dropped to 0.001 using 5 features with Support Vector Machine (SVM) and Multilayer Perceptrons (MLP) classifiers. The result indicated that reducing permission features improved the performance of classification and reduced the computational overhead.

Corresponding Author:

Howida Abubaker,
Faculty of Engineering, School of Computing,
Universiti Teknologi Malaysia (UTM), Johor, Malaysia.
Email: howida10@gmail.com

1. INTRODUCTION

With the growth of smartphones and the services, they provide such as online shopping, health monitoring system, money transaction and many more. The frequent use of mobile devices with that facilities encourage people to store and share their personal and critical information through using mobile devices, and the wide use of devices with Android system makes Android-based mobile devices a target for malicious application developers [1]. These malicious applications may leak the user's private information without their knowledge or consent. Since android operating system security model is based on application-oriented mandatory access control and sandboxing. Each application assigns a unique User ID and a set of permissions at the app installation time. Android developers must request permission to use these special features in a standard format that is parsed at installation time [2]. The permission model used in Android has many advantages and can be effective in preventing malware while also informing users what applications are capable of doing once installed [3]. When user download apps, that apps request some permissions to limit access to system resources. The applications can access resources on the condition that the permissions

are defined in manifeast file. Thereby, a security layer is created for the users. However, that permissions could be exploited by attackers This exploitation can be done via camera, SMS, call, audio, and image or location exploitation by attacking the system call, permission or API inside the Android smartphone [4-6]. Many studies employed machine learning in detecting attacks such as detecting phishing attack in the study of Jupin et al., [7]. In this paper, we will also utilize machine learning in exploring the requested permissions in android malware applications. Therefore, the contribution of this paper is to propose a framework to explore the permissions of android applications that are being requested at installation and run time by using feature selection method with the combination of using machine learning classifiers. The proposed feature selection method based on selecting the important features that contribute to class target using ensemble extra tree classifier. The proposed approach targets the binary class classification problem to classify android apps as malware or non-malware. The rest of this paper is organized as follows. Related work is introduced in section 2. In section 3, we provide an overview of the methodology. The experimental setup is explained in detail in section 4. Results and discussion are clarified in section 5. Section 6 concludes the paper and presents possible future work.

2. RELATED WORK

The increase usage of mobile devices [8] and their applications with android system in the market has been led to conduct an active research in analyzing android apps to investigate the pattern of the malicious apps [9]. The Android- Manifest.xml file is one component of the Application Package file (APK) that has an essential information about the application and in which permissions are stored [10]. It declares which permissions the application must have to access protected parts of the API and interact with other applications. In order to protect Android users, applications access to resources is restricted with permissions. An application must obtain permissions in order to use sensitive resources like the camera, microphone, or call log [11]. Investigating and studying requested permissions have been done by many researchers. X. Liu and J. Liu [12] proposed a framework that uses machine learning techniques to get high detection accuracy with the potential of detecting Android malware applications based on permissions [12].

Wang et al., [13] studied the requested permissions of android app by analyzing the risk of an individual permission and the risk of a group of collaborative permissions. They used feature ranking methods such as mutual information, correlation coefficient, and T-test to rank Android individual permissions with respect to their risk and they used the sequential forward selection as well as principal component analysis to identify risky permission subsets. Three machine learning classifiers used to evaluate their experiment (support vector machine, decision trees, and random forest). Their method achieved performance with a detection rate of 94.62% and a false positive rate of 0.6%. Jiao et al., [14] proposed a hybrid detection method based on permission. The applications are detected according to their permissions to benign and malicious applications. Then, the suspicious applications are run in order to collect the function calls related to sensitive permissions. Furthermore, suspicious applications are represented in a vector space model and their feature vectors are calculated by TF-IDF algorithm. Their method achieved a true positive rate at 91.2 % and a false positive rate at 2.1 %. Ju et al., [10] analyzed the permissions of malicious applications to investigate the ability of them in recognizing suspicious applications. They studied the system of monitoring requested permission in mobile applications to check the permission request history of each application and manage applications. Altaher [15] proposed a hybrid neuro-fuzzy classifier (EHNFC) for Android malware classification using permission-based features to improve detection accuracy. However, the set of permissions required by any Android app during installation time is considered as the feature set which are used in permission - based detection of Android malwares. Those high dimensional feature set should be reduced to minimize computational overhead by choosing an optimal sub - set of features.

There are many studies done on exploring that permissions; for example, Wang, et al., [13] used different methods of feature selection such as Sequential Forward Selection (SFS), and Principal Component Analysis (PCA). After selecting subset of features, they used SVM, Decision Tree and Random Forest, to detect suspicious apps based on the identified subsets of risky permissions. Verma et al., [16] used the information gain algorithm of feature selection to select the best features from the extracted features of android application package files. That method depends on the entropy of the attributes and selects the largest value of gain as the best feature. The study done by Altyeb Altahar [17] used two features selection algorithms, Information Gain (IG) and Pearson CorrCoef (PC) to rank the individual permissions and API's calls based on their importance. Kumar et al., [18], proposed a novel approach to distinguish between malware and benign applications based on permission ranking, similarity-based permission feature selection, and association rule for permission. However, most studies investigate requesting permissions based at installation time. In this study, we focus on exploring permissions as a feature for android apps at installation

and run time to find out the pattern of them in identifying the risky apps. We also employed an extra tree classifier which is known for fast performance to select the important features [19].

3. METHODOLOGY

The aim of our paper is to develop a framework to classify android app based on analyzing permissions to select the important subset of permissions features that are related to class target using ensemble extra tree classifier.

3.1. Framework components

The framework consists of the following phases:

a) Dataset: The dataset consists of the following information such as App's Package (the application's package name), Permissions (a list of permissions declared in the malware and non-malware apps). This phase is described as a Pre-processing dataset in which the dataset is cleaned as depicted below in Figure 1.
b) Features Selection: In this phase, a feature selection method based on an extra tree classifier applied to extract the important features as declared in Figure 1.
c) Classification and Evaluation: In this stage, several classification machine learning algorithms or classifiers are applied. The dataset generated from the second phase (after applying feature selection method) and cleaned dataset are fed to machine learning to build machine learning models and classify the android app as malware or non-malware as explained in Figure 1. We used Scikit Learn libraries [20] to apply feature selection techniques and implement machine learning algorithms.

3.1.1. Dataset

We used the dataset of Mahindra [21] to build the dataset of our study. They collected around 13,000 Android application packages (. apk) as normal apps from different resources and 6971 malicious applications from known sources such as Android Botnet data set [22], DroidKin data set [23], Android Malware Genome Project [24] and AndroMalShare [25]. They extracted the permissions at installation and run time after running the collected Android application packages (. apk) using emulator bluestack [26]. In this study, we used the new version of their dataset that contains 18,850 normal android application packages and 10,000 malware application packages. The dataset has 173 permissions features (99 static permissions and 74 dynamic permissions), where each feature represents the permission. The occurrence of permission is represented by one while the absence of permission is represented by zero. The static permissions collected at installation time are denoted by (S) while the dynamic permissions collected at run-time are indicated by (D). Those permissions were distributed among 30 categories of the apps [21]. The dataset is a publicly available from the website that is described in their paper [36]. After cleaning the dataset and deleting the duplicated samples, we got 25458 samples (8643 malware apps & 16815 benign apps) with 173 features, labeled as malware and non-malware.

3.1.2. Features

The features used in this study are the app's permissions that are requested during installation and at run time. The focus of this paper is to find the optimal set of permissions, a set that gives high accuracy and is more related to class target, out of all the permissions provided by an Android operating system. To accomplish this, we used inbuilt class feature_importances of extra tree-based classifiers. Figure 1 shows a diagram of our proposed framework.

4. EXPERIMENTAL SETUP METHODOLOGY

Our experiment includes the following steps:

4.1. Implementation

We used Python; the programming language to conduct our experiment by utilizing Scikit- Learn [20]. Scikit- Learn is one of the communities that has implemented a machine learning algorithm.

4.1.1. Preprocessing

In this stage, we deleted the duplicated samples and organized them as malicious and benign by labelling them as malware and non-malware. The permissions of applications are extracted through installing and run time. The features are stored as a binary matrix of (0, 1) binary values. After that, we applied a feature selection method to build a dataset with different reduced features. As described in Figure 1.

4.1.2. Feature selection

The second phase of our framework is using the feature selection technique as shown in Figure 1. We used feature selection method based on ensemble extra Tree-based feature selection to select the important subset of features. Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of the model. We used the feature importance property of the model. Feature importance gives a score for each feature of data between zero and one. The higher the score is, the more important or relevant is the feature towards the output variable. This score helps in choosing the most important features and drop the least important ones for model building. Feature importance is an inbuilt class that comes with Tree Based Classifiers, we used Extra Tree Classifier [27, 28] which implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting to compute feature importance, which in turn can be used to discard irrelevant features. As depicted by Figure 1, the feature subset selection scheme starts with initializing the extra tree classifier. The classifier starts by building meta estimators. Each estimator represents the number of trees in the forest as explained in Figure 1. The attribute max_features search for the number of features to consider when looking for the best split of tree node where the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features. At each test node, each tree is provided with a random sample of n features from the feature-set from which each decision tree must select the best feature to split the data based on using the Gini Index. The function (“gini”) is used here by default to compute the Gini Importance of the feature by using the parameter feature_importances_ which is called also as the Gini Importance of the features. The output is ensembled using averaging to choose the important features. By averaging, the estimates of predictive ability over several randomized trees can help in reducing the variance of such an estimate and use it for feature selection. Every feature is ordered in descending order according to the Gini Importance of that feature. And to select the top n features, largest (n) function is used. For instance, to select the top 5, 10, and 20 features, (n) value is assigned to 5, 10, and 20 respectively as shown in Figure 1. After extracting the important permissions features, we plot the top 5, 10 and 20 features as demonstrated in the following figures.

The top 5 important features that we obtained after applying feature importance feature selection method are declared in Figure 2. As we can observe from Figure 2, the permission (Default: read phone state and identity (S)) represents the most important feature with a score of 0.12. This permission Allows only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any Phone Accounts registered on the device. This permission is considered dangerous as declared by the studies in [13, 15, 16] because that permission allows the application to call phone numbers without user intervention. As a result, malicious applications may cause unexpected calls on a user phone bill. The permission (write contact data (S)) under Contact pattern comes in the second rank after (Default: read phone state and identity (S)) permission. This permission allows an application to write the user's contacts data [30, 31]. This permission is dangerous because it involves the user's private information, as declared in [18, 31, 32].

The permissions (Your accounts: contacts data in Google accounts (D)) asked at run time and allowed Sites and apps to request different kinds of access to Google Account, including requests to:

a) See basic profile information: Many sites and apps only request access to basic info, including name, email address, and profile picture. When a user grants access to this info by choosing "Sign in with Google" on sites and apps that have this feature. Sharing this information makes it easier to create an account and helps the user avoid creating new passwords.
b) See some information in Google Account: In addition to basic profile information, some sites and apps might also ask for permission to see and make a copy of information in the user account. This may include information like Contacts, Photos, YouTube playlists, and more.
c) Edit, upload & create content in Google Account: In addition to seeing a basic profile and some information in the account, some sites or apps may ask for permission to do even more in Google Account. This may include editing, uploading, or creating content. For example, a film editing app may edit video and upload it to the YouTube channel, or an event planning app may create events on Google Calendar, with user permission [33].

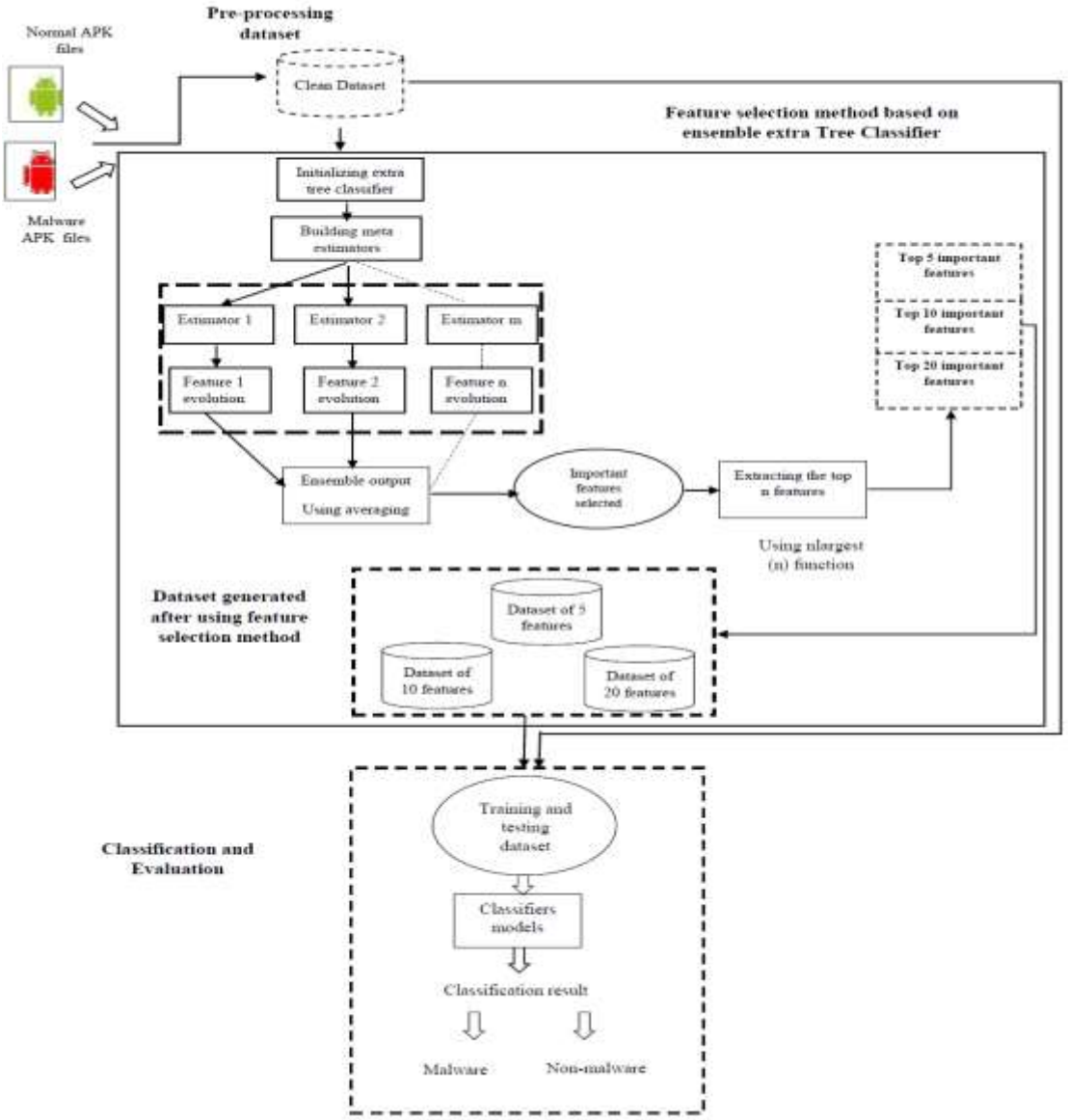

Figure 1. The proposed framework

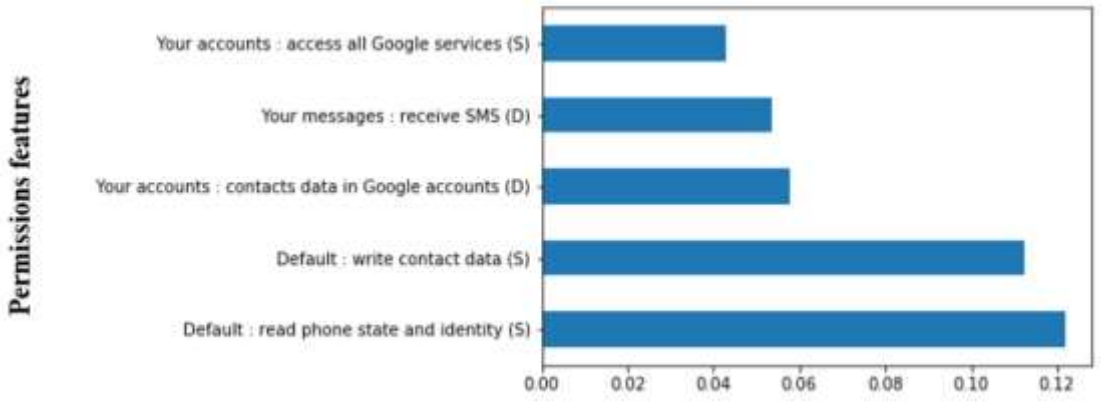

Figure 2. Top 5 important features with its feature importance score

The study done by Mahindru [21], categorized this permission as dangerous permission since asked by the most suspicious apps. Wang et al., [13] used mutual information, correlation coefficient, and T-test features selection methods to rank Android individual permissions with respect to their risk. Their method showed that (receive _SMS) permission has ranked the very top risk permission by the three ranking methods used. This type of SMS-related activities mainly contributes to the dominant threats [13]. The permission (access all Google services) grouped as one of the risky permission as identified by Mahindru [21]. The top 10 features are selected by assigning the value of n with (10) in largest (n) method, Figure 3 highlights the top 10 important features selected. As we notice from Figure 3, static permissions have more propagation than dynamic permissions with an average value of 0.6.

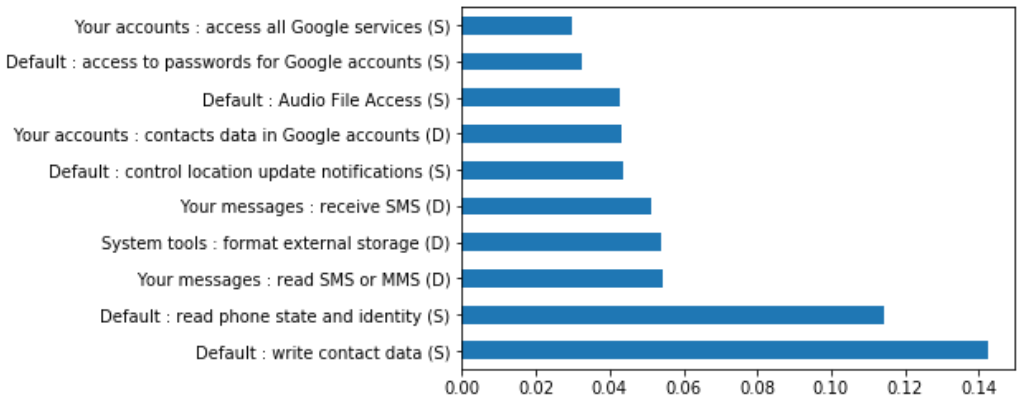

Figure 3. Top 10 important features with its feature importance score

When we extracted the top 10 important features, we found out that the top 5 features that extracted above in Figure 2 occur again in the list of top 10 features as shown in Figure 3. However, static features still have the most distribution with an average of 0.6. As we observe from the Figure 3 above, the new occurrence of dangerous permissions such as (format external storage (D), read SMS or MMS (D), control location update notification (S), Audio File Access (S), access to passwords for Google accounts (D). These permissions listed as risky permission as stated by the study in [21].

When we extracted the top 20 important, we find that most permissions percentage are dynamic permissions that occur during run time with an average of 0.55 as displayed in Figure 4. These dynamic permissions (read SMS or MMS (D), receive _ SMS (D), contacts data in Google accounts (D), format external storage (D), read calendar event (D), write contact data (D), add or modify calendar event and send email to guest (D), write browser's history and bookmarks (D), read contact data (D), find (GPS) location (D) and: read phone state and identity (D)) are listed as dangerous permissions as proved by the study in [21].

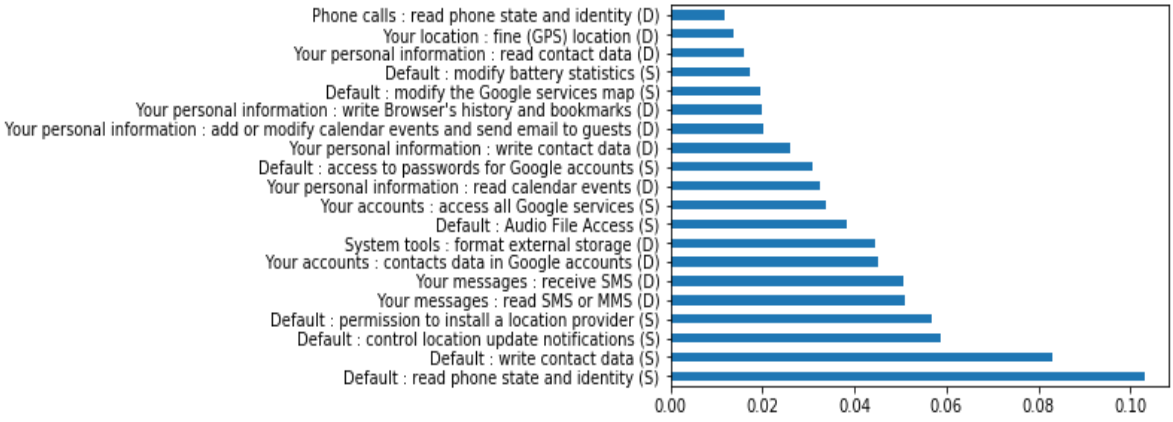

Figure 4. Top 20 important features with its feature importance score

4.1.3. Classification and evaluation

In this phase, we built classifier models using five machine learning classifiers Support Vector, K Neighbors Classifier, Decision Tree, Naive Bayes and Multilayer Perceptrons (MLP). We evaluated the classifiers by splitting the dataset to 17820 samples in the training set and 7638 samples in the testing set. We used four datasets with 25458 samples and 173, 5, 10, and 20 permissions features respectively.

4.1.4. Evaluation metrics

We used the following confusion matrices to evaluate our classifiers:

Specificity: Specificity also called the true negative rate that measures the proportion of actual negatives that are correctly identified as such (e.g., the percentage of malicious apps that are correctly identified as malicious as described below:

$$TNR = (TN / TN+FP) \quad (1)$$

The true positive (TP) in our study is the number of benign applications correctly recognized. And the false positive (FP) represents the number of malware applications wrongly identified as benign. The true negative (TN) indicates the number of malware applications correctly recognized. And the false negative (FN) represents the number of benign applications that are wrongly identified as malicious.

False Negative Rate (FNR): FNR measures the proportion of all benign apps that will be identified wrongly as malicious apps as shown below:

$$FNR= (FN / FN+TP) \quad (2)$$

Overall Accuracy (ACC): ACC measures the percentage of correctly identified applications:

$$ACC= (TP+TN / TP+TN+FP+FN) \quad (3)$$

5. RESULTS AND DISCUSSION

The result and discussion are explained in section 5.1 and section.5.2.

5.1. Results

The outcome of our experiment is represented in this section, we conducted 4 experiments using 4 datasets and 5 classifier algorithms. The first dataset consists of 25458 samples with 173 features. After using a feature selection method, we got three datasets with 25458 samples and 5, 10, 20 features respectively. The experiment measured the classifier's TNR, FNR, and prediction accuracy. We used 70% of the data set as a training dataset and 30% as a testing dataset. So, there are 17820 samples in the training set and 7638 samples in the testing set. Table. 1 depicts the evaluation metrics of each classifier algorithm with four dataset.

The results show that using less features (20 features) improves the accuracy. For instance, using Support Vector and KNeighbors Classifier with 20 features achieved the highest accuracy with 94 %. However, Naïve Bayes achieved the best result of the accuracy of 90 % with 10 features while using 173 features, the result obtained is 88 % which is the lowest rate. The best accuracy obtained was 95 % using Multilayer Perceptrons (MLP) with 20 and 173 features. In terms of FNR (the number of benign apps misclassified as malware), Support Vector and Multilayer Perceptrons (MLP) achieved the best result of 0.001 with 5 features. Also, using Naïve Bayes, the result dropped from 0.1 with 173 features to 0.03 with 5 features. KNeighbors Classifier produced the best performance in terms of TNR (number of malware classified as malware) with an 89 % rate using 20 features. However, Decision Tree produced the best accuracy of 94 % when we used all features (173 features) as shown in Table 1.

5.2. Discussion

As we can see from our previous result that support vector machine (SVM) and KNeighbors Classifiers achieved good accuracy when we minimized the number of features from 173 to 20. Since SVM and KNeighbors classifiers are computationally expensive due to the implementation of quadratic programming and require more time to execute classification [34, 35], so reducing features helps in reducing the computation process and improving accuracy as well [37]. However, when we used probabilistic classifiers such as Naïve Bayes, the performance of accuracy is also improved with a 90.4 % rate using 10 features. Moreover, since most 20 features that were selected as important features are dynamic permissions

features that asked at run time, we can conclude that choosing that permissions helps in improving the accuracy. Also, a different number of permissions requested by different kinds of apps gives evidence that permissions can be used as effective features to discriminate malware apps from benign apps since all that 20 permissions are considered as risky and represent the target class as well [21]. In addition, from our result obtained it is observed that using less attributes in the feature space resulted in poor performance. The primary reason is that these features do not represent the target class. Therefore, in respect of the accuracy, adding features (20 features) in the feature space captures the salient information variability in the feature vectors of instances belonging to the class and thus improving the classification. Multilayer Perceptrons (MLP) yields the best performance that is the accuracy is increased from 0.85 to 0.95 with 20 features, and all features. This gives evidence that using all features is not a guarantee that the best result would be found for a classifier. Furthermore, processing more features is computationally and timely expensive. In addition, we found that FNR decreased when we used the less features (5 features) to 0.001. Comparing our result with the result of study [17], we concluded that our method (feature selection based on extra tree classifier) achieved better accuracy with 94 % and 95 %, while study [17] achieved accuracy with 89 using Information Gain (IG) and Pearson CorrCoef (PC) features ranking algorithms. From our findings, we can conclude that our proposed method based on Extra-Tree classifier helps in increasing accuracy and reducing computational burdens [38].

Table 1. The result of classification using different dataset of features

Classifier algorithms	Number of Features	TNR	FNR	Accuracy
Support Vector machine	5 features	0.58	0.001	0.85
	10 features	0.78	0.09	0.91
	20features	0.85	0.01	0.94
	173 features	0.90	0.03	0.91
KNeighborsClassifier	5 features	0.70	0.08	0.85
	10 features	0.83	0.02	0.91
	20 features	0.89	0.02	0.94
	173 features	0.81	0.02	0.92
Decision Tree	5 features	0.78	0.17	0.81
	10 features	0.87	0.1	0.89
	20 features	0.90	0.07	0.92
	173 features	0.93	0.05	0.94
Naïve Bayes	5 features	0.68	0.03	0.87
	10 features	0.79	0.04	0.90
	20 features	0.82	0.07	0.89
	173 features	0.83	0.1	0.88
Multilayer Perceptrons (MLP)	5 features	0.58	0.001	0.85
	10 features	0.78	0.01	0.92
	20 features	0.78	0.01	0.95
	173 features	0.89	0.02	0.95

6. CONCLUSION

Permission is one of the most important features for analyzing Android apps. Our proposed permission-based framework uses machine learning algorithms to classify the android app as malware or benign apps based on using Extra-Tree classifier feature selection method. Feature selection method based on ensemble extra tree classifier that has inbuilt class feature importance to assign a score for each feature of a dataset and select the important feature that closes to target class has been used. Four datasets were used with 5, 10, 20 and 173 features respectively and five classifiers algorithms were used (Support Vector Machine (SVM), K Neighbors Classifier, Decision Tree, Naïve Bayes and Multilayer Perceptrons (MLP)). The classifiers models are evaluated using true negative rate (TNR), false-negative rate (FNR) and accuracy metrics. The experimental results show that Support Vector and KNeighbors Classifier with 20 features achieved the highest accuracy with 94 % and the highest TNR rate with 89 %. And we concluded that most dangerous permissions are requested during the run time and have more contribution in enhancing model performance as obtained by using Support Vector and KNeighbors Classifier with 20 features.

Moreover, we concluded that our proposed feature selection method based on Extra-Tree classifier improves the classification accuracy and reduces computational loads. To further improve classification, in the future we will use different feature selection methods and investigate other fitness criteria to improve the efficiency of permission-based Android malware analysis.

ACKNOWLEDGEMENTS

The authors would like to thank the Universiti Teknologi Malaysia (UTM) for their support in Research and Development and the Soft Computing Research Group (SCRG) for the inspiration in making this study a success. This work is supported by Ministry of Higher Education (MOHE) under Fundamental Research Grant Scheme (R. J130000.7828.4F989).

REFERENCES

[1] F. Idrees, and M. Rajarajan, "Investigating the android intents and permissions for malware detection," *Int. Conf. Wirel. Mob. Comput. Netw. Commun*, pp. 354-358, 2014.
[2] S. Kumar, A. Viinikainen, and T. Hamalainen, "A network-based framework for mobile threat detection," *1st Int. Conf. Data Intell. Secur.*, pp. 227-233, 2018.
[3] S. Bhandari, et al., "Android inter-app communication threats and detection techniques," *Comput. Secur*, vol. 70, pp. 392-421, 2017.
[4] Anwar, Z. and Khan, W.A., "Guess who is listening in to the board meeting: on the use of mobile device applications as roving spy bugs, " *Security and Communication Networks*, vol. 8, no. 16, pp. 2813-2825, 2015.
[5] M. Alenezi, and I. Almomani, "Abusing Android permissions: A security perspective," *IEEE Jordan Conf. Appl. Electr. Eng. Comput. Technol*, pp. 1-6, 2018.
[6] M. M. Saudi, et al., "A new mobile malware classification for camera exploitation based on system call and permission," *World Congr. Eng. Comput. Sci.*, vol. 1, 2017.
[7] J. A. Jupin, et al., "Review of the machine learning methods in the classification of phishing attack", *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1545-1555, 2019.
[8] D. Wahyono, et al., "Preliminary study of wireless balloon network using adaptive position tracking technology for post disaster event, " *TELKOMNIKA* , vol. 17, no. 4, pp. 1767-1773, 2019
[9] D, Lynkova.'25+ Awesome Mobile Marketing Statistics for 2019', 2019. [Online]. Available: https://techjury.net/stats-about/mobile-marketing/.
[10] S. Ju, H. Seo, and J. Kwak, "Research on android malware permission pattern using permission monitoring system," *Multimed. Tools Appl.*, vol. 75, no. 22. Pp. 14807-14807, 2016.
[11] developer.android.com. ' Permissions overview '. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview.
[12] X. Liu, and J. Liu, "A two-layered permission-based android malware detection scheme," *Proc. - 2nd IEEE Int. Conf. Mob. Cloud Comput. Serv. Eng*, pp. 142-148, 2014.
[13] W. Wang, et al., "Exploring permission-induced risk in android applications for malicious application detection, " *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 11, pp. 1869-1882, 2014.
[14] H. Jiao, et al., "Hybrid detection using permission analysis for android malware," *International Conference on Security and Privacy in Communication Networks*, vol. 152, pp. 541-545, 2015.
[15] A. Altaher, "An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 4147-4157, 2017.
[16] S. Verma, and S. K. Muttoo, "An android malware detection framework based on permissions and intents," *Def. Sci. J.*, vol. 66, no. 6, pp. 618-623, 2016.
[17] A. Altaher, and O. Mohammed, "Intelligent hybrid approach for android malware detection based on permissions and API calls," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, pp. 60-67, 2017.
[18] R. Kumar, et al., " Research on data mining of permission-induced risk for android IoT devices, " *Appl. Sci.*, vol. 9, no. 2, pp. 277, 2019.
[19] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3-42, 2006.
[20] scikit-learn.org.'scikit-learn Machine Learning in Python '. [Online]. Available:https://scikit-learn.org/stable/
[21] A. Mahindru, and P. Singh, "Dynamic permissions based android malware detection using machine learning techniques," pp. 202-210, 2017.
[22] Kadir, A.F. A., Stakhanova, N. and Ghorbani, A. A., "Android botnets: What urls are telling us," *International Conference on Network and System Security*, pp. 78-91, 2015.
[23] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "DroidKin: Lightweight detection of android apps similarity, " pp. 436-453, 2014.
[24] Y. Zhou and X. Jiang. "Android Malware Genome Project ," 2012. [Online]. Available: http://www.malgenomeproject.org/.
[25] 'SandDroid - An automatic Android application analysis system '. See User's Manual. [Online]. Available: http://sanddroid.xjtu.edu.cn:8080/.
[26] 'Bluestacks '. [Online]. Available: http://www.bluestacks.com/.
[27] Sklearn. Ensemble. 'ExtraTreesClassifier '. [Online]. Available: https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier.
[28] 'Tree-based feature selection¶'. [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html#tree-based-feature-selection.
[29] 'Python, Pandas DataFrame. nlargest ()'. [Online]. Available: https://www.geeksforgeeks.org/python-pandas-dataframe-nlargest/.

[30] W. Li, J. Ge, and G. Dai, "Detecting malware for android platform: An SVM-based approach, " *IEEE 2nd Int. Conf. Cyber Secur. Cloud Comput.* pp. 464-469, 2015.
[31] developer.android.com. 'WRITE_CONTACTS'. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission.html#WRITE_CONTACTS.
[32] L. Sun, et al., "Significant permission identification for machine-learning-based android malware detection, " *IEEE Trans. Ind. Informatics*, vol. 14, no. 7, pp. 3216-3225, 2018.
[33] 'Third-party sites & apps with access to your account'. Available: https://support.google.com/accounts/answer/3466521?hl=en
[34] C.-C. Chang and C.-J. Lin, "Libsvm, " *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1-27, 2011.
[35] M. V. Varsha, P. Vinod, and K. A. Dhanya, "Identification of malicious android app using manifest and opcode features, " *J. Comput. Virol. Hacking Tech.*, vol. 13, no. 2, pp. 125-138, 2017.
[36] A. Mahindru, "Android Malware and Normal permissions dataset," 2018. [Online]. Available: https://data.mendeley.com/datasets/958wvr38gy/5.
[37] A. Zakrani, M. Hain, and A. Idri, "Improving software development effort estimation using support vector regression and feature selection, " *International Journal of Artificial Intelligence*, vol. 8, no. 4, pp. 399-410, 2019.
[38] What is the extra trees algorithm in machine learning? [Online]. Available: https://www.quora.com/What-is-the-extra-trees-algorithm-in-machine-learning.