ICIC Express Letters
Volume **10**, Number **7**, July **2016**

 ISSN 1881-803X
pp. **1605–1612**

STUDY ON MINIMAL FEATURE SELECTION FOR ANDROID MALWARE DETECTION

Huixiang Zhang[1], Chunlei Chen[2], Jinghua Li[1] and Yi Luo[1]

[1]School of Automation
Northwest Polytechnical University
No. 127, West Youyi Road, Xi'an 710072, P. R. China
zhanghuixiang@nwpu.edu.cn

[2]School of Computer Engineering
Weifang University
No. 5147, East Dongfeng Street, Weifang 261061, P. R. China
chunlei.chen@wfu.edu.cn

Received December 2015; accepted March 2016

Abstract. *Supervised machine learning algorithms are usually used to detect Android malware. The primary task is to select suitable features to train the classifiers. In this paper, the requested permissions and Android APIs are used as the classification features respectively. The permission feature sets are selected according to the difference value of declared ratio, the information gain and the permission category respectively. The Android API feature sets are selected based on the difference value of invoked ratio. These feature sets are evaluated using four commonly used supervised machine learning algorithms, i.e., Naïve Bayes, C4.5, KNN and Random Forest. As a result, a feature set of 40 permissions and a set of 60 Android APIs are determined to be the minimal feature set for Android malware detection.*
Keywords: Android malware detection, Feature selection, Permission, Android API

1. **Introduction.** Google's Android operation system employs a permission-based security model. Android requires that developers declare a list of permissions in a manifest file. In order to install an Android application into a mobile device, the user must allow all permissions requested by the application.

It is natural for researchers to detect Android malware according to the permissions requested by applications. Sanz et al. presented manifest analysis for malware detection in Android (MAMA) [1]. They extracted the permission and feature tags from the manifest file. These features were then used to build machine learning classifiers to detect malicious applications. [2] illustrated that a permission-based classifier could detect more than 81% of malicious samples. The permission-based mechanism can be used as a quick filter to identify malicious applications.

However, there are more than one hundred of build-in Android permissions and a large number of third-party permissions. Permission-based classifiers tend to be trained with high-dimensional feature vectors. For example, Huang et al. adopted 297 features for Android malware detection [2]. However, dealing with so many features can decrease the run-time efficiency. Moreover, it is also easy to degrade the classification performance if the redundant features exist. Thus, it is necessary to carry out the feature pruning to reach balance between the vector length and classification performance.

It should be noted that the requested permissions declared in the manifest file may not be the actual permissions required by an application. Felt et al. analyzed the source codes of Android applications to determine whether developers follow the least privilege principle with their permission requests [3]. They found that approximately one-third of evaluated applications are over privileged. Google does not provide a complete mapping

to the permissions that the API call may need, and it is not a trivial work to find the actually required permissions [4]. Therefore, we think it is better to directly use API as the classification features, rather than try obtaining the required permissions. However, there are a large number of APIs used in Android applications; we have to find a relatively small API feature set for Android malware detection.

In this paper, we investigate on minimal feature selection for Android malware detection. We extract requested permissions and Android APIs as the classification features. Several feature selection methods are presented to construct the feature sets. These feature sets are evaluated using four commonly used supervised machine learning algorithms. Finally, the minimal permission and Android API feature sets are determined for Android malware detection, respectively.

The remainder of the paper is organized as follows. In Section 2, the evaluation framework for minimal feature selection is introduced. The feature selection methods for requested permissions and Android APIs are discussed in Section 3 and Section 4, respectively. In Section 5, we compare the classification performance of feature sets constructed by different feature selection methods. Finally, we conclude the paper in Section 6.

2. **Approach Overview.** The evaluation framework for minimal feature selection is depicted in Figure 1.

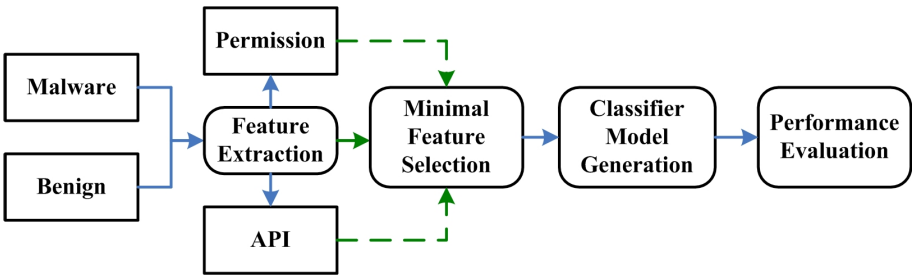

FIGURE 1. The evaluation framework for minimal feature selection

A total of 25255 Android samples are collected in our work. There are 22476 benign samples downloaded from various Android markets. Moreover, there are 2779 malicious samples collected from Android Malware Genome Project [5] and DroidAnalytics Project [6].

Apktool [7] is used to unpack the apk file of a sample into its constituent resources and the AndroidManifest.xml file, and then the Android application's classes.dex file is disassembled into a directory tree of smali files. The requested permissions are extracted from the AndroidManifest.xml file (under the uses-permission tag). The Android APIs are extracted from those smali files.

With these extracted features, several methods are carried out to select features into various feature sets. According to these feature sets, the binary feature vectors are built. If the apk sample contains a selected feature, the corresponding bit in the feature vector is set to 1, otherwise is set to 0. At last, a class label is appended at the end of a feature vector to show that the vector belongs to a benign or a malicious application.

The binary feature vector is evaluated using four supervised machine learning algorithms, i.e., Naïve Bayes, C4.5, KNN and Random Forest. In order to evaluate the impact of various feature vectors on classification performance, the following metrics are measured.

1) True positive ratio (TPR): the ratio of the number of malware samples correctly classified over the total number of malware samples.

2) True negative ratio (TNR): the ratio of the number of benign samples correctly classified over the total number of benign samples.

3) F-score: the harmonic mean of Recall and Precision.

In this paper, the 10-fold cross-validation is used to evaluate the performance of machine learning classifiers for the selected feature sets. The weka data mining software [8] is used to perform the evaluation.

3. **Minimal Feature Selection for Permission-Based Detection.** There are 882 permissions extracted from all the samples, including 147 Android permissions and 735 third-party defined permissions. The amount of third-party defined permissions is more than that of Android permissions, but only requested in few samples. Thus, only Android permissions are considered in this paper. The top 15 permissions requested in benign and malware samples are shown in Figure 2. The INTERNET, ACCESS_NETWORK_STATE, WRITE_EXTERNAL_STORAGE and READ_PHONE_STATE permissions are the most requested permissions in both categories. The malware samples prefer to request the permission related to short messages, such as READ_SMS, SEND_SMS, and WRITE_SMS.

In order to obtain the minimal feature vector, each of the requested permissions is pruned in accordance with the difference value of declared ratio, the information gain, and the permission category, respectively.

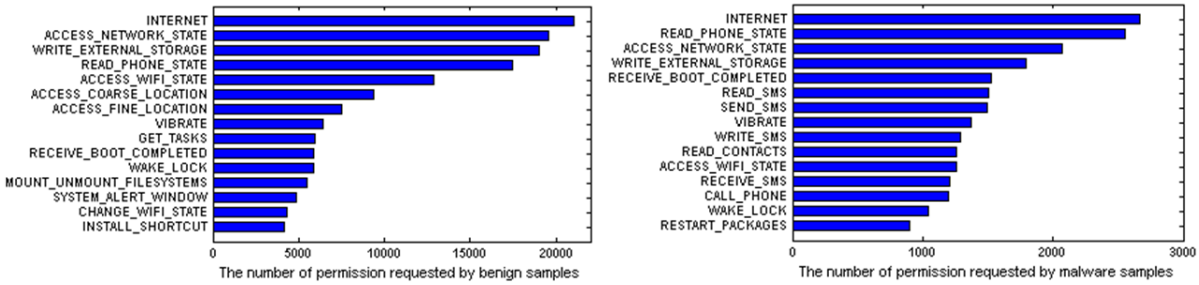

FIGURE 2. The top 15 requested permissions from the samples

3.1. **The difference value of declared ratio (D-value).** For a single Android permission, the declared ratio is defined as the ratio of the requested count of this permission to the requested count of all the 147 Android permissions. The declared ratio is calculated for benign and malware samples, respectively. The permission is selected if its declared ratio of malware samples is larger than that of benign samples. There are 68 Android permissions matching the condition. Moreover, there are 4 permissions declared only in malware samples, and they are GLOBAL_SEARCH_CONTROL, BRICK, DIAGNOSTIC and SET_ACTIVITY_WATCHER. These permissions are ranked in descending order according to the difference value of declared ratio. The top N permissions are used to construct the feature vector, where N is varied from 10 to 72. The classification performance is shown in Figure 3.

As illustrated in Figure 3, the TPR reaches 76% and F-score reaches 84% with all the 72 permissions taken into consideration. In the process of gradually reducing the feature vector length to 40, the performance is almost not affected. However, the performance declines apparently when decreasing the length to 30. The TNR remains above 90% whatever the length of feature vector is. That means most of the benign samples are correctly classified. Thus, the minimal feature set can be reduced to the size of 40 using the method based on the difference value of declared ratio.

3.2. **Information gain (IG).** Information gain is used as the feature selection algorithms in this evaluation. Information gain is the method of determining the rank of appropriate feature through the entropy difference between the case of accurate classification through feature and the case otherwise. The 147 Android permissions are sorted

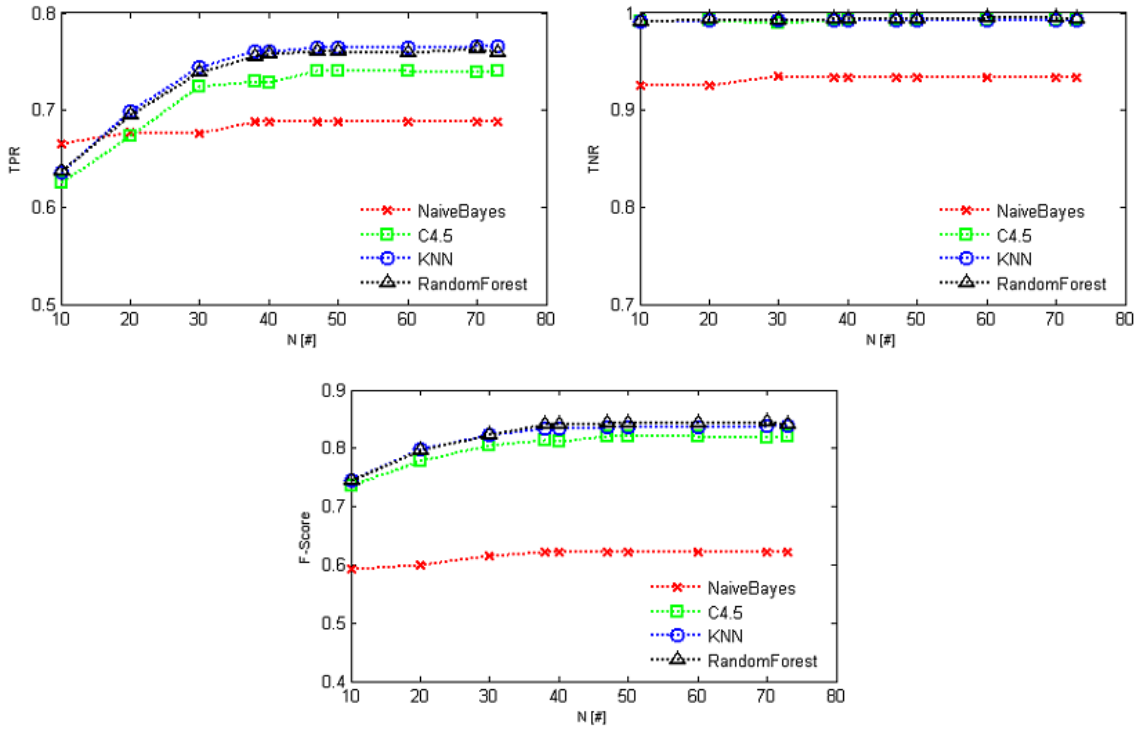

FIGURE 3. The classification performance with features based on the D-value method

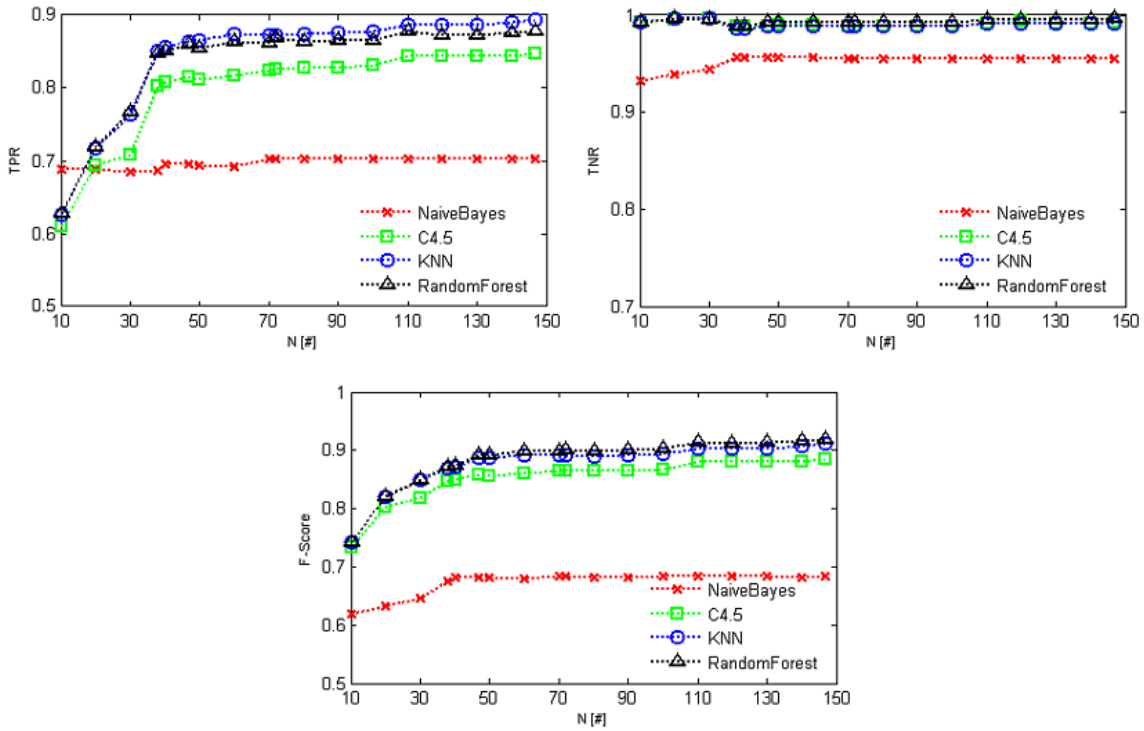

FIGURE 4. The classification performance with features based on the IG method

by their information gain value in descending order. The top N permissions are selected to construct the feature vector, where N varies from 10 to 147. The classification performance is shown in Figure 4.

As shown in Figure 4, adopting all 147 Android permissions leads to the best performance. TPR and F-score reach approximately 88% and 91%, respectively. When the length of feature vector is reduced to 50, there is little impact on the performance. If only the top 40 permissions are used, the performance declines slightly. The performance declines apparently if the length of feature vector is further decreased to 10. The TNR remains above 90% whatever the length of feature vector is. Thus, the minimal feature set can be reduced to the size of 40 using the method based on information gain.

3.3. **Permission category (PC).** The Android permissions are classified into four categories: normal, dangerous, signature, and signatureOrSystem [9]. The category of dangerous permissions means high risk, because giving this kind of permission can result in the following two consequences. First, access to private user data is approved. Second, the requesting applications obtain control over the device and become a threat to the user's privacy. There are 47 dangerous permissions in all samples. All the 47 dangerous permissions are selected to construct the feature vector. Meanwhile, we selected the top 47 permissions using the above two methods respectively. The evaluation results of the three methods are listed in Table 1.

Table 1. The evaluation results

Method	TPR (%)				TNR (%)				F-Score (%)			
	NB	C4.5	KNN	RF	NB	C4.5	KNN	RF	NB	C4.5	KNN	RF
D-value	68.8	74.1	76.5	76.1	93.4	99.2	99.2	99.4	62.3	82.0	83.7	84.2
IG	68.8	81.5	86.4	86.0	96.5	98.9	98.9	99.2	65.6	85.9	88.7	89.2
PC	70.1	70.9	74.0	73.4	95.3	99.1	99.2	99.3	67.6	79.6	82.2	82.1

The D-value method and the permission category method exhibit similar performances. When features are selected by the information-gain-based method, the feature set consists of more permission requested by benign samples. Therefore, the overall performance of information gain method outperforms the other methods. However, it should be noticed that the permissions declared in the manifest file are the requested permissions, other than the required permissions. If the malicious developers declared more permission usually in benign samples, the malware application may be classified into benign application incorrectly. The risk can be mitigated by selecting the feature set using the method of D-value of declared ratio. Overall consideration, the 40 permissions feature set selected by the method of D-value of declared ratio is considered as the better feature set for Android malware detection.

4. **Minimal Feature Selection for API-based Detection.** In this section, we extract Android API from smali files disassembled by apktools. There are more than 200 million API records extracted from all the samples. The APIs which are not declared in the Android API level 21 documents [10] are removed. After that, there are 39303 Android APIs left. As shown in Figure 5, there are no significant differences in the top 15 invoked Android APIs in benign and malware samples. There are 9 API in common in the top 15 API list.

Here we used the difference value of invoked ratio to select the feature API. For a single Android API, the invoked ratio is defined as the ratio of the invoked count of this API to the invoked count of all the 39303 Android APIs. The API is selected if its invoked ratio of malware samples is more than that of benign samples. There are 8616 Android APIs that match the condition. Moreover, there are 1302 APIs invoked only in malware samples. Because the total number of API features is too large, we only focus on the 260 APIs with the difference value larger than 0.15. By increasing the difference value, we constructed the feature vectors with APIs from top 260 to top 10.

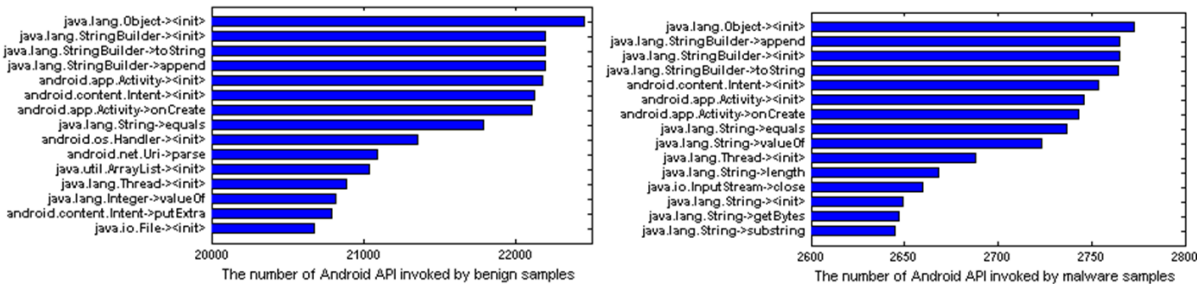

FIGURE 5. The top 15 requested Android APIs from the samples

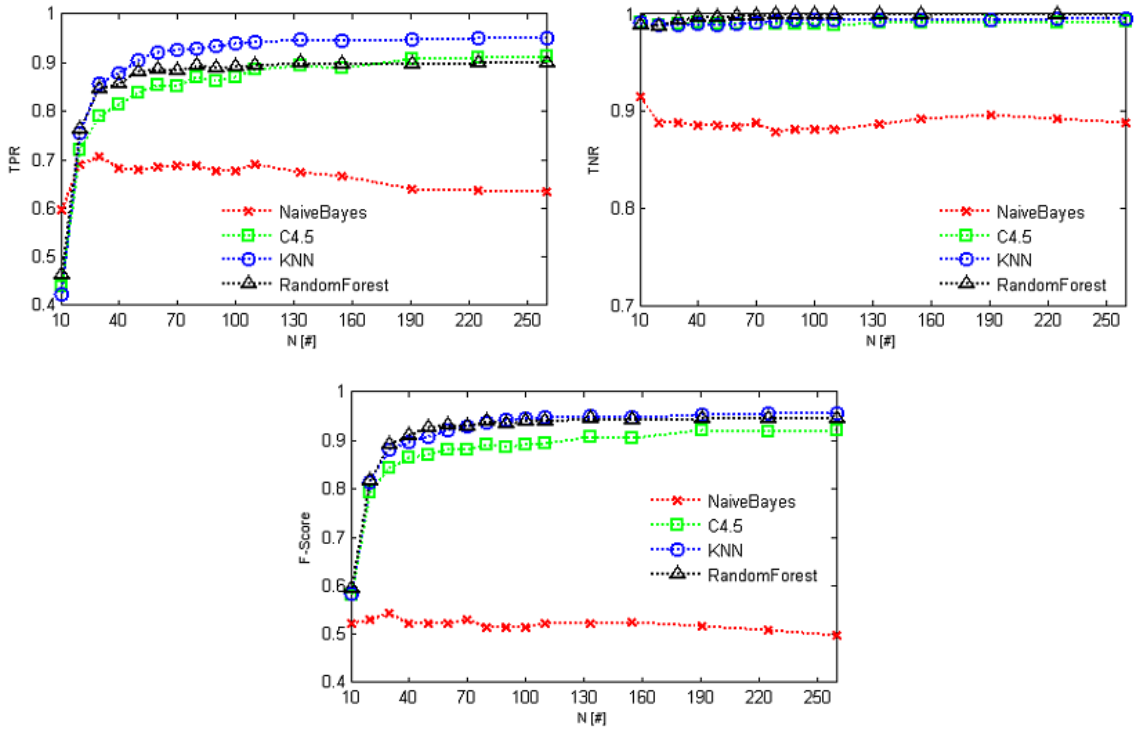

FIGURE 6. The classification performance with API feature sets

As shown in Figure 6, when only the top 60 APIs are used, the performance declines slightly. TPR, TNR and F-score reach 88%, 99%, 90%, respectively. The performance declines apparently if the length of feature vector further drops to less than 60. Thus, the minimal API feature set can be reduced to the length of 60 using the method based on the difference value of invoked ratio.

5. **Performance Comparison.** In this section, we compared the classification performance of feature sets constructed by different feature selection methods. These feature sets include the 40 permissions set, the 147 permissions set, the 60 Android APIs set and the 260 Android APIs set.

As illustrated in Figure 7, the performance of Android API set is higher than that of permission set. With the KNN classification algorithm, the TPR gets to 95%, TNR gets to 99%, and F-score gets to above 90%. However, the Android APIs are collected from a large number of disassembled smali files. The extracting process is more complicated and time consuming than the process of using permission features which are declared in a single manifest file. The KNN classifier always outperforms the other three classification algorithms on the four feature sets. The performance of Naïve Bayes classifier is the

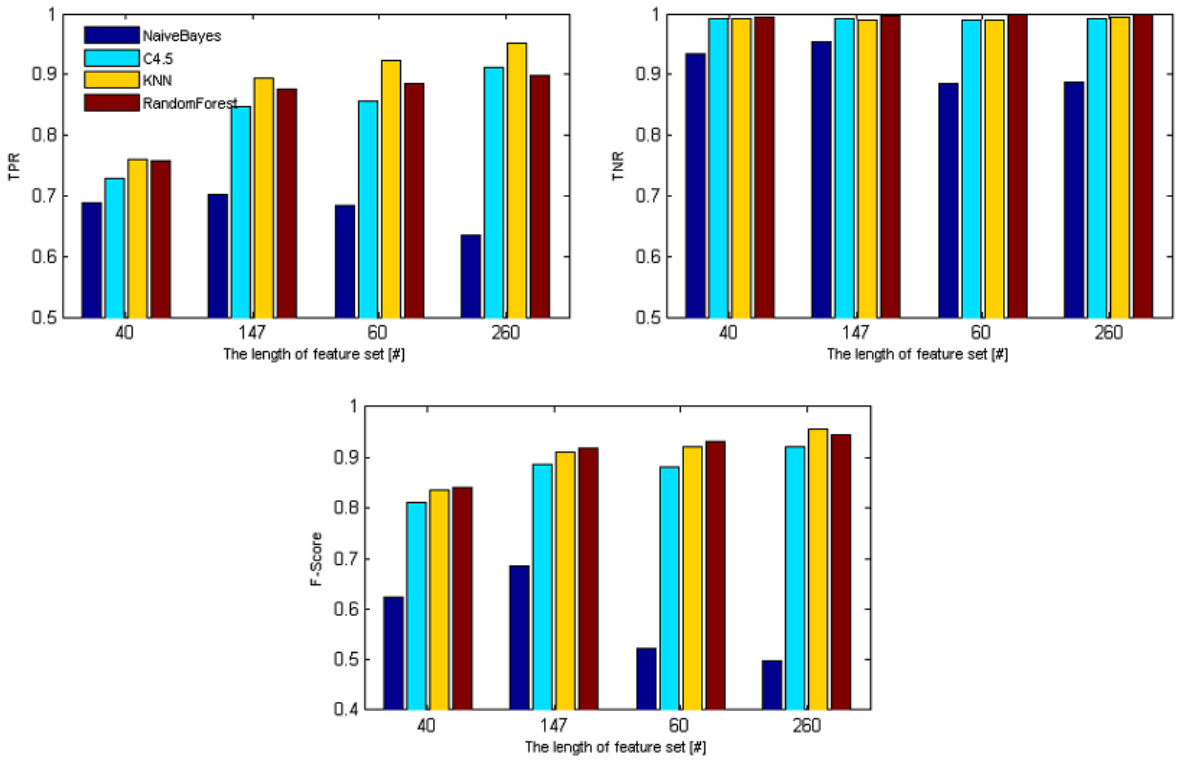

FIGURE 7. The evaluation results with the four feature sets

worst. However, as seen in Figure 3, Figure 4 and Figure 6, the performance of Naïve Bayes classifier is not affected by the size of feature sets.

Moreover, it should be noticed that there are 2% samples whose feature vectors are all zero by using the 147 permissions set. That means the requested permissions of these samples are not covered by the 147 permissions set. If the set of 40 permissions is used, there are more than 10% samples whose feature vectors are all zero. In such situation, we classified these samples into benign applications. However, there are only 1.3% samples with all zero feature vectors by using the 60 Android APIs set. Therefore, the set of Android APIs is more efficient to detect the malware.

6. **Conclusions.** In this paper, we investigated on the minimal feature selection for Android malware detection. We focused on the requested permission feature and Android API feature. For permission feature, three methods are used to select the permission feature set, including the method of difference value of declared ratio, the method of information gain and the method of permission category. For Android API feature, the method of difference value of invoked ratio is used to select the feature set. By comparing the performance of various classifiers trained with various feature sets, we obtain the minimal permission set with 40 permissions and the minimal Android API set with 60 APIs. The extraction of Android API feature is more complicated than that of the permissions, but the performance of the Android API set is better than that of the permission set.

As mentioned in Section 5, the size of feature sets has slight influence on the performance of Naïve Bayes classifier. However, the performance metrics of Naïve Bayes classifier is lower than those of the other three classifiers. Thus, it is possible to improve the Naïve Bayes classifier to achieve comparable performance with fewer features. This is left as future work.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61303224. The authors also gratefully

acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] B. Sanz, I. Santos, C. Laorden, X. U. Pedrero, J. Nieves, P. G. Bringas and G. A. Marañón, MAMA: Manifest analysis for malware detection in Android, *Cybernetics and Systems: An International Journal*, vol.44, nos.6-7, pp.469-488, 2013.

[2] C. Y. Huang, Y. T. Tsai and C. H. Hsu, Performance evaluation on permission-based detection for Android malware, *Advances in Intelligent Systems and Applications*, vol.2, pp.111-120, 2013.

[3] A. P. Felt, E. Chin, S. Hanna, D. Song and D. Wagner, Android permissions demystified, *Proc. of the 18th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, pp.627-638, 2011.

[4] K. W. Y. Au, Y. F. Zhou, Z. Huang and D. Lie, PScout: Analyzing the Android permission specification, *Proc. of the 19th ACM Conference on Computer and Communications Security*, Raleigh, NC, USA, pp.217-228, 2012.

[5] Y. Zhou and X. Jiang, Dissecting Android malware: Characterization and evolution, *Proc. of the 33rd IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp.95-109, 2012.

[6] M. Zheng, M. Sun and J. C. S. Lui, DroidAnalytics: A signature based analytic system to collect, extract, analyze and associate Android malware, *Proc. of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Melbourne, VIC, Australia, pp.163-171, 2013.

[7] *Apktool*, A tool for reengineering Android apk files, http://code.google.com/p/Android-apktool/.

[8] *Weka 3*, Data mining with open source machine learning software in Java, http://www.cs.waikato.ac.nz/ml/weka/, 2014.

[9] <permission>, *Android Developer – API Guides – Android Manifest*, http://developer.Android.com/guide/topics/manifest/permission-element.html.

[10] *Android APIs (Level 21)*, http://developer.Android.com/reference/packages.html.