Optimizing Android Malware Detection Via Ensemble Learning

https://doi.org/10.3991/ijim.v14i09.11548

Abikoye Oluwakemi Christiana, Benjamin Aruwa Gyunka
University of Ilorin, Ilorin, Nigeria

Akande Noah Oluwatobi (✉)
Landmark University, Omu-Aran, Nigeria
akande.noah@lmu.edu.ng

Abstract—Android operating system has become very popular having the highest market share amongst all other mobile operating systems. However, the popularity of Android based mobile applications have opened it up to several attacks and malwares. Traditional signature-based malware detection techniques have been proven to be less effective in detecting new and unknown malware, therefore, machine learning techniques are taking the lead for timely zero-day anomaly detections. Therefore, this study presents an optimized android malware detection model using ensemble learning technique. Random Forest, Support Vector Machine, and k-Nearest Neighbours were used to develop three distinct base models and their predictive results were further combined using majority vote combination function to produce an ensemble model. Reverse engineering procedure was employed to extract static features from large repository of malware samples and benign applications. WEKA 3.8.2 data mining suite was used to perform all the learning experiments. The results obtained revealed that Random Forest had a better sensitivity of 97.9% and a classification accuracy of 98.00% among the other base classifiers connoting that it is a strong base model. However, the ensemble model achieved a sensitivity of 98.1% and a classification accuracy of 98.16%. The finding shows that, although the base learners had good detection results, the ensemble learner produced a better optimized detection model compared with the performances of those of the base learners.

Keywords—Android Malware Detection, Machine Learning Models, Base Learners, Ensemble Learner, Reverse Engineering

1 Introduction

Hand held devices have become very critical part of human lives and they play very crucial part in the economy. Table 1 shows that the most popular of these are the smart devices, in the form of Smartphones or Tablets, that runs the Android Operating System [1]. The rapid increase of the Android systems in the economy is also supported by the open source nature of the platform. This has drawn lots of attentions to the platform from both legitimate (innocent) and illegitimate (malicious) users. Malware developers

have especially seen this as an opportunity to exploit the users of these devices through cyber thefts and other devastating attacks [2]. These mischievous developers are releasing Android malware into the economy at an exponential rate [3] and research has shown that, in the second quarter of 2018, a new malicious Android application was introduced every 7 seconds into the wild [4]. The continuous growth rate of Android malware, both in overall volume and in number of existing variants, is so rapid that it has become very difficult to deploy signature-based detection systems to combat the new trend [5,6].

Table 1. Worldwide Smartphone OS Market Share Forecast [7]

Year	Android (%)	iOS (%)	Others (%)	Total (%)
2017	85.10	14.70	0.20	100.00
2018	85.10	14.90	0.00	100.00
2019	86.70	13.30	0.00	100.00
2020	86.60	13.40	0.00	100.00
2021	86.90	13.10	0.00	100.00
2022	87.00	13.00	0.00	100.00
2023	87.10	12.90	0.00	100.00

Malware as an application carries out functions completely contrary from its legitimate intent. This poses serious questions of trust on the developers of application software and also on the rigidity ability of applications against tampering by malicious attackers. Android malware can be propagated via different attack vectors and channels such as Bluetooth connections, memory cards, wireless networks, Universal Serial Bus (USB) connections, third party Apps providers, as well as Google Play Store [8]. Most of these media provide entry-level defense and detection mechanisms such as authentications and scans, but the USB channel does not provide any verification or authentication for apps or devices during installation and are thus found to be a critical infection vector especially in the case of cross platform malware [8, 9, 10]. Cloud based applications are not exempted from malware attacks. This was explored in [11]. Authors in [12] observed that most malwares are in portable executable files, therefore, a technique to scan these files using Hadoop and Boyer–Moore-Horspool Search algorithm was proposed.

The sophistication of techniques employed by these emerging Android malwares have continued to generate a growing challenge for most traditional detection and analysis systems which are unable to handle the subtlety and behavioural dynamism of these malware [5]. This challenge has created gap for the need of a detection system that does not employ the techniques and signatures-based methods of most existing detection systems. Predictive models that are based on single classifiers are not easily scalable from one context to another context [13] as there does not exist any single classifier that is considered dominant for all data distributions or can do discriminative well enough in cases where the number of classes is large [14, 15, 16]. Therefore, the fusing of many different classifiers to form a composite model, also known as ensemble learning shown in Fig. 1 has become very pertinent in tackling situations or challenges of

dealing with numerous complicated patterns [17]. The technique of classifiers combination helps to eradicate the tendencies of selecting the worse classifier and it has also shown to enhance the performance of the best individual base models for a wider range of classification problems [18,19,20]. Research has shown that ensemble learning is also an effective approach for resolving the problems of class imbalances, at the algorithm level [21] and it also reduces generalization error.

The effectiveness of ensemble methods is largely due to the observable fact that various types of classifiers have different inductive biases [22]. Ensemble methods can effectively make use of the inductive biases of the different classifiers to reduce the variance-error without increasing the bias-error [23, 24]. Authors in [25] noted that ensemble learning, which is based on the aggregation of the results from multiple models, is an approach that is more sophisticated for increasing the accuracy of models compared to the traditional practice of parameter tuning on a single model. Ensemble learning is able to obtain increased accuracy because of the simple but very powerful process of group averaging or majority vote which enables the reduction of base model variance, and to a lesser extent bias reduction [26].

Two main elements that influence the performance of ensemble models are the diversity and strength of the base classifiers. This is because when the base classifiers have high level of diversity and more strength, the ensemble model will then have lower generalization error – that is, the combination of the outputs of multiple classifiers reduces the generalization error [27].

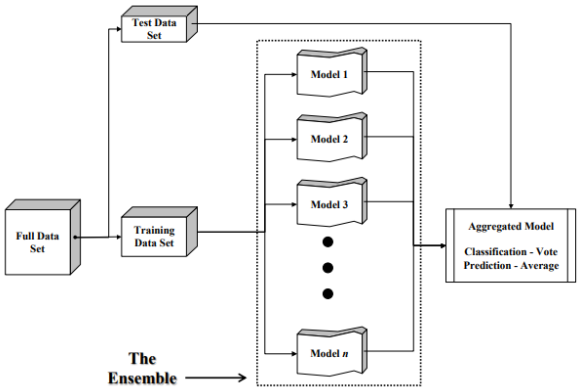

Fig. 1. Generalized Ensemble Method [28]

Other terms used to describe ensemble methods include hybrid methods, committee of classifiers, opinion of pool, cooperative agents and aggregation [29, 30]. The main goal of classifier combination study is to be able to identify the conditions under which an ensemble of classifiers will yield the greatest gain in performance compared to the individual classifiers [18, 31]. This study thus deployed machine learning techniques to develop an ensemble learning classification model, that is based on static permission features, for Android malware detection.

2 Classification Algorithms

Machine learning algorithms have one main goal which is to look for patterns, similarities, regularities, trends, and redundancies in a given data and any model produced using the given data sample has the ability to predict the properties of observation provided in the future from the same data source [32, 33]. A classification algorithm (also known as a method or classifier) is a systematic approach used for building classification models from data set given as an input [34, 35]. Three main classification algorithms deployed in this work are Random Forest, Support Vector Machine, and k-Nearest Neighbours.

2.1. Tree-based: Random Forest (RF)

This is a Tree-based classification algorithm that combines the power of random Decision Trees with that of Bagging in order to obtain very high accuracy in classification [36, 37]. This technique makes RF to work differently from the traditional Decision Tree by applying a test on a number of features that are randomly given focusing on the individual node of the tree and does not do any pruning. RF makes use of Bagging (bootstrap aggregation) in order to produce a diverse ensemble of classifiers by the introduction of randomness into the input of the learning algorithm [38].

2.2. Function-based: Support Vector Machine (SVM)

Support Vector Machines (SVM), also known as binary classifiers, are non-probabilistic supervised machine learning classification algorithm that are applied widely in the analysis and anomaly detection of malware [20, 39]. The algorithm makes use of a hypothesis space of linear functions in a high dimensional feature space, in which it strives to achieve linear separability. For it to do that, it deploys margin maximization and kernels [40]. Two main versions of SVM exist in WEKA and they are Sequential Minimal Optimization (SMO) (with Puk kernel) and LibSVM (with linear kernel). SVMs have the ability to deal with data that are highly-dimensional and sparse. It makes use of hyper-planes to separate various instances into their respective classes. For it to work with non-linearly separable data, SVM depends on kernel methods. By default, SVM are normally bundled with three kernel functions namely; polynomial, sigmoid, and radial basis function [39]. SMO, which is an optimization algorithm, was deployed in this work for the training of the SVM.

2.1 Lazy-based: k-Nearest Neighbours (k-NN)

KNN is known typically as lazy learner. It is so called not because it's obvious simplicity but for the reason that it does not learn a discriminative function from the data provided for training but it rather memorizes the given dataset for training, thus KNN does not have any training time. What it does is to just store the data meant for training and then wait until data for testing is provided and the classification is performed based

on the most related data in the training data that was stored [41]. It doesn't use the training data points to do any generalization (i.e., it keeps all the training data). The training phase for K-NN is extremely fast because it does not really have a training phase. For K-NN to make prediction, it searches for the nearest neighbours in the entire training dataset. In this work, K-NN was implemented in the WEKA environment using IBK classification filter on the given dataset. Unlike eager learners, the lazy learner takes less time in the training phase but more time in predicting.

3 Methodology

The research methodology employed in this study is discussed in this section.

3.1 Data collection

Data used in this study were extracted primarily from Android Application Packages files (APKs). The APKs are of two distinct types: benign and malicious. The Android malware application packages were downloaded from the Android Malware Genome Project [42, 43]. The Benign APK files were downloaded primarily from Google Play Store. Furthermore, Evozi APK downloader as shown in Figure and apkpure web tools were used to download the APKs and details such as package name, file size, QR Code, MD5 file hash, date last fetched and app version were retrieved from the downloaded APKs.

Fig. 2. Evozi APK Downloader Downloading First Bank MobileApp

Afterwards, Virus Total application which is an online malware scanner, was used to properly screen the downloaded APKs so as to ascertain their true state, either benign or malicious, before being put to use.

3.2 The ensemble model framework

The proposed detection model is shown in Fig. 4 which is a combination of three heterogeneous base-classifiers; Random Forest, Support vector machine, and k-Nearest Neighbours. The different stages include feature extraction, feature vectors matrix formulation, base-learners training and the ensemble model creation through a combination function and a metal combiner.

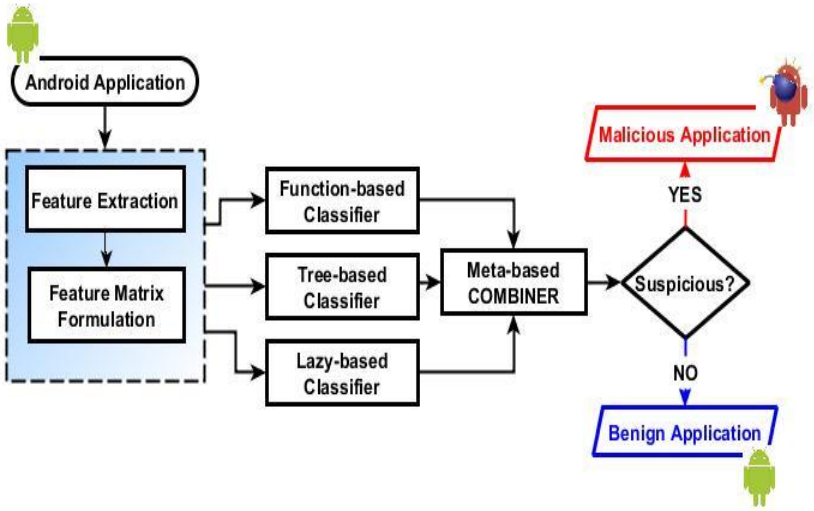

Fig. 3. The Ensemble Model Framework

3.3 Feature extraction

Permission features were extracted from 1904 applications, 952 benign and 952 malicious, through the process of reversed engineering. Reverse engineering is a process of studying and analyzing the underlying source codes of any application or software in order to understand its functionalities and behaviours and to come up with a new or an improved version [44]. According to [45], software or Applications reverse engineering integrates several arts: code breaking, puzzle solving, programming, and logical analysis. The two distinct tools used in this study for reverse engineering APKs are Androguard and Sublime Text 2. Fig. 4 shows each stage of the apk reverse engineering and features extraction processes.

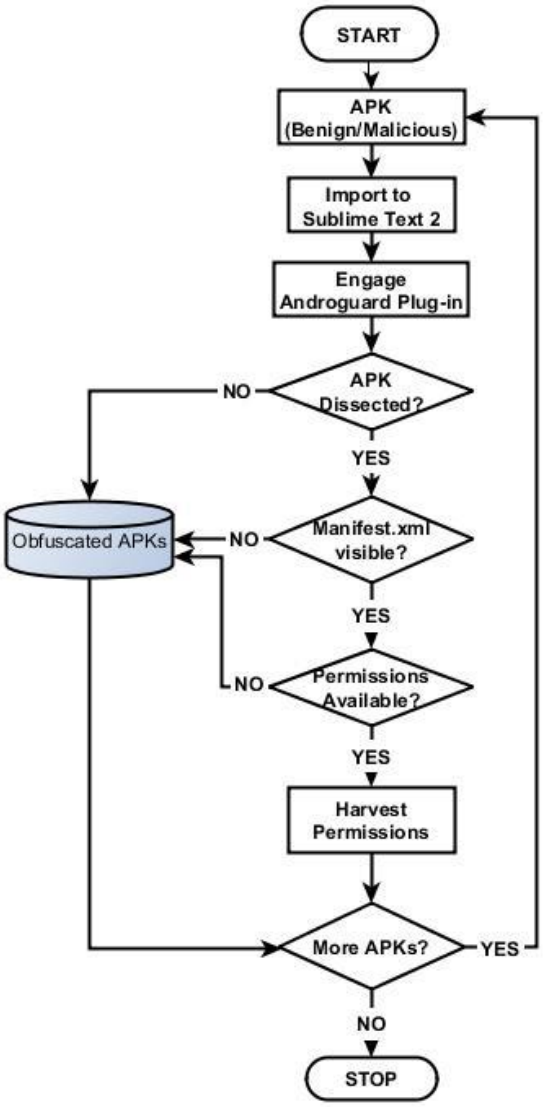

Fig. 4. Reverse Engineering Stages for APK Files

3.4 Base-classifiers training and classification phase

The three classification algorithms used are heterogeneous and each one of them was trained through WEKA using the same dataset to generate the first-level base prediction models (classifiers) as shown in Fig. 5.

```
STEP 1:  Input training data set i of size [1904X163] and nominal classes of size 2
STEP 2:  Input base algorithms, j = (RF, SVM, kNN)
STEP 3:  FOR all i do
STEP 4:      FOR j from 0 to 2 do
STEP 5:          Perform model training
STEP 6:          Generate base classifier
STEP 7:      END FOR
STEP 8:  END FOR
STEP 9:  Start Classification
STEP 10: FOR all base classifiers do
STEP 11:     Output classification results
STEP 12:     Perform comparative analysis of classification results
STEP 13: END FOR
STEP 14: Output best performing base classifier
STEP 15: END
```

Fig. 5. Pseudocode for Classification with Base-Algorithm

3.5 Ensemble learning training and classification phase

The combination of classifiers is a technique that has been extensively and widely used in data mining because it has been shown to immensely reduce the error rate in classification problems compared to single classifiers and it also enables the production of a system that has a robust performance against the difficulties that individual classifiers may be having on individual data set [46].

$$V\ (x) = \ argmax_n \sum_{j=1}^{t} w_j I(M_j(x) = n) \tag{1}$$

However, it is not a fixed rule for an ensemble method to always outperform the base-classifiers; factors like selection of classifiers and the combination scheme used in fusing together the predictions can affect the stack performance of the Ensemble system [47]. The results of the base learners were fused using fixed combination function shown in equation (1) and vote metal combiner as illustrated in Fig. 6.

```
STEP 1:  Input training data set of size [1904X163] and nominal classes of size 2
STEP 2:  Input base classifiers (RF, SVM, kNN)
STEP 3:  Engage Vote Meta Combiner for Multiple Classifiers combination
STEP 4:  Select Combination Rule to engage
STEP 23: Combine Classifiers using MAJORITY_VOTING_RULE
STEP 24:   IF argmax_malicious Σ_{m=1}^{3} p_m > argmax_benign Σ_{m=1}^{3} p_m  THEN
STEP 25:                    Classify as malicious App
STEP 26:   ELSE
STEP 27:                    Classify as benign App
STEP 28:   END IF
STEP 29: Output class with the highest vote as the ensemble prediction
STEP 30: END
```

Fig. 6. Vote Ensemble Learning Workflow

3.6 Model testing and performance evaluations

Performance evaluation is critical to any data mining task. [48,49] identified True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Negative Rate (FNR), Precision, F-Measure, Accuracy (ACC), and Error Rate (ERR) as standard data mining performance evaluation metrics. As listed in equations (2) - (9),

they were employed to evaluate the performance of the models. TPR measures the ratio of correctly classified malicious apps to the total number of malicious apps in the dataset while TNR measures the ratio of correctly classified benign apps to the total number of benign apps in the dataset. FPR measures the ratio of incorrectly classified benign apps to the total number of benign apps in the dataset while FNR determines the ratio of incorrectly classified malicious apps to the total number of malicious apps in the dataset. The total accuracy of the classifier is measured by the ACC while ERR measures the error rate. The ratio of the malicious application that have been correctly classified/ predicted to the total number of predictions as malicious was measured by precision which is also the positive predictive value. The predictive strength of the classifier is measured by the receiver operating characteristic curve. It is a graphical plot that shows the diagnostic ability of a classifier system as its discrimination threshold is varied.

$$TPR\ (Sensitivity) = {}^{TP}/_{TP + FN} \tag{2}$$

$$TNR\ (Specificity) = {}^{TN}/_{TN + FP} \tag{3}$$

$$FPR = {}^{FP}/_{TP + FP} \tag{4}$$

$$FNR = {}^{FN}/_{FN + TN} \tag{5}$$

$$Precision = {}^{TP}/_{TP + FP} \tag{6}$$

$$F - Score = \frac{2*Precision*Recall}{Precision*Recall} \tag{7}$$

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

$$ERR = 1 - ACC \tag{9}$$

4 Results and Discussion

The results obtained from the three base classifiers and the ensemble model is presented in this section. Their performance was measured using metrics such as: true positive, false positive and error rate, F-Measure, precision, ROC Area, modelling time, kappa statistics and classification accuracy.

4.1 Random forest results analysis

As presented in Table 2, Random Forest had a sensitivity of 97.9% with a F-Measure and ROC Area of 98.7% and 99.8% respectively. The model however recorded a very low false alarm rate of 1.9%. The F-Measure result shows that the model correctly classified 98.7% of test instances in 10-fold cross validation as malware.

Table 2. Prediction Accuracy Results for Random Forest Classifier

Class	Evaluation Metrics				
	TP Rate	***FP Rate***	***Precision***	***F Measure***	***ROC Area***
Benign	0.981	0.021	0.979	0.980	0.998
Malicious	0.979	0.019	0.981	0.980	0.998
Weighted Avg.	0.980	0.020	0.980	0.980	0.998

The Confusion Matrix, Table 3, shows that Random Forest was able to correctly classify 932 instances as malicious and 934 as benign. 18 instances are incorrectly classified as malicious while 20 are incorrectly classified as benign.

Table 3. Confusion Matrix Result for Random Forest Classifier

		Predicted class	
		Malicious	***Benign***
Actual Class	Malicious	$TP = 932$	$FN = 20$
	Benign	$FP = 18$	$TN = 934$

The time taken by Random Forest (RF) to build the model was 1.49 seconds as shown on Table 4. A commutative total of 1866 instances were correctly classified, that is 98.0042% of the total number. The Random Forest Model has Kappa statistics of 0.9601 (96.01%), indicating excellent detection performance.

Table 4. Statistics Result for Random Forest

Time taken to build model	1.49 seconds
Correctly Classified Instances	1866 (98.0042%)
Incorrectly Classified Instances	38 (1.9958%)
Kappa statistic	0.9601

4.2 Support Vector Machine (SVM) results analysis

SVM recorded a total of 96.3% detection rate for malicious application as indicated on Table 5. An F-Measure and a ROC Area of 97.6%. It has very good classification strength with these metrics with a low false detection alarm rate of 1.2%. The classification was done using Pearson VII kernel function (PUK) for optimal performance of the SVM algorithm.

Table 5. Prediction Accuracy Results for SVM Classifier

Evaluation Metrics					
Class	***TP Rate***	***FP Rate***	***Precision***	***F-Measure***	***ROC Area***
Benign	0.988	0.037	0.964	0.976	0.976
Malicious	0.963	0.012	0.988	0.976	0.976
Weighted Avg.	0.976	0.024	0.976	0.976	0.976

The SVM Confusion Matrix, Table 6, showed that the classifier correctly classified 917 instances as malicious and 941 instances as benign, while 11 instances are incorrectly classified as malicious and 35 instances incorrectly classified as benign. The model was able to detect more instances as benign with very low false positive.

Table 6. Confusion Matrix Result for Support Vector Machine

		Predicted class	
		Malicious	***Benign***
Actual Class	Malicious	TP = 917	FN = 35
	Benign	FP = 11	TN = 941

Cumulatively, the Classifier correctly predicted 1858 (97.584%) instances while 46 (2.416%) were incorrectly classified. The model took 4.36 seconds to build and a Kappa statistic of 95.17% was obtained as shown in Table 7.

Table 7. Statistics Result for Support Vector Machine

Time taken to build model	4.36 seconds
Correctly Classified Instances	1858 (97.584%)
Incorrectly Classified Instances	46 (2.416%)
Kappa statistic	0.9517

4.3 k-nearest neighbours (k-NN) results analysis

kNN has a detection rate of 97% as presented in Table 8. The values of the F-Measure, Precision and ROC Area for k-NN are 98%, 99.1% and 98.8% respectively. However, the model recorded one of the lowest false positive detection rate of 0.8%.

Table 8. Prediction Accuracy Results for k-NN Classifier

	Evaluation Metrics				
	TP Rate	***FP Rate***	***Precision***	***F-Measure***	ROC Area
Class					
Benign	0.992	0.030	0.970	0.981	0.988
Malicious	0.970	0.008	0.991	0.980	0.988
Weighted Avg.	0.981	0.019	0.981	0.981	0.988

Table 9 shows that k-NN was able to correctly classify 923 instances as malicious and 944 instances are classified correctly as benign, while 8 instances are incorrectly classified as malicious and 29 incorrectly classified as benign.

Table 9. Confusion Matrix Result for k-Nearest Neighbours (k-NN)

		Predicted class	
		Malicious	***Benign***
Actual class	Malicious	TP= 923	FN=29
	Benign	FP=8	TN = 944

k-Nearest Neighbours built the model in 0 seconds, implying that the algorithm does not really learn on the given training set but stores them for application during testing. A total cumulative of 1867 instances were correctly classified; 98.0567% percent of the total instances. 37 (1.9433%) instances on the other hand were wrongly classified. The Classifier has a Kappa statistic of 96.11% as shown on Table 10.

Table 10. Statistics Result for k-Nearest Neighbours (k-NN)

Time taken to build model	0 seconds
Correctly Classified Instances	1867 (97.0567%)
Incorrectly Classified Instances	37 (1.9433%)
Kappa statistic	0.9611

4.4 Comparisons of base classifiers performance results

Table 11 summarizes the results for each individual classifier. The tabular comparison of the results show that Random Forest turns out as the best performing base-learner given that it has a very low false positive rate of 1.9% and an error rate of 0.2% and has the highest ROC Area of 99.8%. k-NN recorded a very strong competitive performance with Random Forest. k-NN has the lowest FPR of 0.8% and a highest precision of 99.1%. SVM also performed relatively well with very strong TPR, Accuracy, and error rates.

Table 11. Comparison of the Base-Classifiers Results

Metrics	**Base Classifiers (Models)**		
	RF	***SVM***	***k-NN***
TP Rate	0.979	0.963	0.970
FP Rate	0.019	0.012	0.008
Precision	0.981	0.988	0.991
Accuracy	0.980	0.976	0.981
F-Measure	0.980	0.976	0.980
Error Rate	0.02	0.024	0.019
ROC Area	0.998	0.976	0.988
Correctly classified Instances	1866 (98%)	1858 (97.58%)	1867 (98.06%)
Kappa Statistic	0.960	0.952	0.961
Model time	1.49	4.36	0

All the six different heterogeneous classifiers maintain very minimal time for model building thus having generally low computational overheads when performing classification on new instances.

4.5 Ensemble learning results analysis

To form the composite ensemble model, Vote (a meta learner) was used to combine all the base learners in a parallel form using Majority vote fixed combination function.

Ensemble models are slower to build compared to the single models, due to the reason that more time is needed to combine all the classifiers to post-process the results.

4.6 Majority vote results analysis

Table 12 displays the results for majority vote combination rule recorded a TPR for malicious instances as 98.1% and a false positive rate of 1.8%. The Precision and F-Measure for the malicious class are 98.2% and 98.1% which all show great strength in the prediction capacity of the Ensemble model. The F-Measure result indicated that the Majority Voting rule enabled the classifiers to correctly classify 98.1% of test instances in 10-fold cross validation.

Table 12. Prediction Results for Majority Voting Combination Rule

	Evaluation Metrics				
Class	***TP Rate***	***FP Rate***	***Precision***	***F-Measure***	***ROC Area***
Benign	0.982	0.019	0.981	0.982	0.982
Malicious	0.981	0.018	0.982	0.981	0.982
Weighted Avg.	0.982	0.018	0.982	0.982	0.982

The confusion matrix, shown in Table 13, provides the parameter values for calculating the actual performance of the Majority Voting combination rule. It shows that the actual instances classified by the algorithms as malicious are 934 while 935 instances were classified as Benign. 17 instances were misclassified as malicious while 18 instances were misclassified as benign.

Table 13. Confusion Matrix for Majority Voting

		Predicted Class	
		Malicious	***Benign***
Actual class	Malicious	TP = 934	FN = 18
	Benign	FP = 17	TN= 935

Table 14 shows that the time taken to build the model was 9.23 seconds and the Kappa statistic was 96.32% which indicate prediction strength of the Model. 1869 (98.1618%) instances were correctly classified while 35 (1.8382%) were misclassified.

Table 14. Statistics Result for Majority Voting

Time taken to build model	9.23 seconds
Correctly Classified Instances	1869 (98.1618%)
Incorrectly Classified Instances	35 (1.8382%)
Kappa statistic	0.9632

4.7 Comparison of best base model and best ensemble model

The Accuracy and Error Rate are great measures that provide sound identifications on the performance result of a classification model. However, a very high detection accuracy value does not indicate that a model is doing well when the false positive detection rate of the model is also high. Table 15 shows the comparison between the best base-model and the ensemble model. The Majority Vote combination rule produced a true positive detection rate of 98.1% which is relatively an improvement in the detection accuracy compared to 97.9% detection rate obtained by the best single classifier, Random Forest.

Table 15. Comparison of Best Base Model and Best Ensemble Model

S/N	Metrics	Majority Voting	Random Forest
1.	TP Rate	0.981	0.979
2.	FP Rate	0.018	0.019
3.	Precision	0.982	0.981
4.	Accuracy	0.982	0.980
5.	F-Measure	0.982	0.980
6.	Error Rate	0.018	0.02
7.	ROC Area	0.982	0.998
8.	Correctly classified Instances	1869 (98.16%)	1866 (98%)
9.	Kappa Statistic	0.9632	0.960
10.	Model time	9.23	1.49

The Ensemble model obtained best performance results in false positive, accuracy, error rate and Kappa Statistics as 1.8%, 98.2%, 1.8% and 96.32%. The base model only has best performance in ROC Area as 99.8% against 99.7% for the Ensemble model.

5 Conclusion

Random Forest produced the best base detection model, having a true positive detection rate of 97.9%, false positive detection rate of 0.19%, accuracy of 98%, and a detection error rate of 0.2%. The Majority Vote combination rule produced an ensemble model with a true positive malware detection rate of 98.1%, false positive detection rate of 0.18%, a detection accuracy of 98.2%, and a detection error rate of 0.18%. The ensemble Model outperformed the single model with a relative difference of 0.2% on the true positive detection rate. The ensemble model has a very low false alarm rate of 0.18% and the lowest error rate of 0.18%. The study therefore concludes that a supervised ensemble model is an effective approach for the anomaly detection of Android malware.

6 References

[1] Statista. (2018). Smartphone OS global market share 2009-2018 | Statistic. Retrieved June 26, 2018, from https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[2] Narudin, F. A., Feizollah, A., Anuar, N. B., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. Soft Computing, 20(1), 343–357.https://doi.org/10.1007/s00500-014-1511-6

[3] Feng, Y., Anand, S., Dillig, I., & Aiken, A. (2014). Apposcopy: Semantics-Based Detection of Android Malware Through Static Analysis. In Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'14) (pp. 16–22).https://doi.org/10.1145/2635868.2635869

[4] Lueg, C. (2018). Malware figures for Android rise rapidly. Retrieved August 24, 2018, from https://www.gdatasoftware.com/blog/2018/07/30937-malware-figures-for-android-rise-rapidly

[5] Richter, L. (2015). Common Weaknesses of Android Malware Analysis Frameworks. In IT Security Conference, University of Erlangen-Nuremberg during summer term 2015 (pp. 1–10). Erlangen.

[6] Vidas, T., Tan, J., Nahata, J., Tan, C. L., Christin, N., & Tague, P. (2014). A5: Automated Analysis of Adversarial Android Applications. SPSM '14: Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, 39–50. https://doi.org/10.1145/2666620.2666630

[7] IDC. (2019). Smartphone Market Share. Retrieved June 18, 2019, from https://www.idc.com/promo/smartphone-market-share/os

[8] Aquilina, N. (2015). Cross-Platform Malware Contamination Cross - Platform Malware Contamination (Master's Thesis). Royal Holloway, University of London, London.

[9] Dunham, K., Hartman, S., Morales, J. A., Quintans, M., & Strazzere, T. (2014). Android malware and analysis (1st ed.). New York: Auerbach Publications. https://doi.org/10.1201/b17598

[10] Zhauniarovich, Y. (2014). Android TM Security (and Not) Internals (ASANI Book) (1.01). Trento: asani.

[11] Mahmoud M. El-Khouly, M. Samir Abou El-Seoud (2017). Malware Detection in Cloud Environment (MDCE). International Journal of Interactive Mobile Technologies (iJIM), 11(2), 139-145. https://doi.org/10.3991/ijim.v11i2.6575

[12] Eman Ahmed, Amin Sorrour, Mohammed Sobh, Ayman Bahaa-Eldin (2017). A Cloud-based Malware Detection Framework. International Journal of Interactive Mobile Technologies (iJIM), 11(2), 113-127. https://doi.org/10.3991/ijim.v11i2.6577

[13] Pandey, M., & Taruna, S. (2016). Towards the integration of multiple classifier pertaining to the Student's performance prediction. Perspectives in Science, 8, 364–366. https://doi.org/10.1016/j.pisc.2016.04.076

[14] Amancio, D. R., Comin, C. H., Casanova, D., Travieso, G., Bruno, O. M., Rodrigues, F. A., & Da Fontoura Costa, L. (2014). A systematic comparison of supervised classifiers. PLoS ONE, 9(4). https://doi.org/10.1371/journal.pone.0094137

[15] Kuncheva, L. I. (2004). Combining Pattern Classifiers: Methods and Algorithms (2nd ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

[16] Pappa, G. L., & Freitas, A. (2010). Automating the Design of Data Mining Algorithms: an evolutionary computation approach. Verlag Berlin Heidelberg: Springer Science & Business Media.

[17] Benmokhtar, R., & Huet, B. (2006). Classifier Fusion: Combination Methods For Semantic Indexing in Video Content. In International conference on artificial neural networks (pp. 65–74). Berlin, Heidelberg: Springer. https://doi.org/10.1007/11840930_7

[18] Bilmes, J. A., & Kirchhoff, K. (2003). Generalized rules for combination and joint training of classifiers. Pattern Analysis and Applications, 6(3), 201–211. https://doi.org/10.1007/s10044-002-0188-0

[19] Shahzad, R. K. (Blekinge I. of T., & Lavesson, N. (Blekinge I. of T. (2012). Comparative Analysis of Voting Schemes for Ensemble-based Malware Detection. Proc. of the 7th International Conference on Availability, Reliability, and Security, (August), 98–117.

[20] Milosevic, N., Dehghantanha, A., & Choo, K. K. R. (2017). Machine learning aided Android malware classification. Computers and Electrical Engineering, 61, 266–274. https://doi.org/10.1016/j.compeleceng.2017.02.013

[21] Phung, S. L., Bouzerdoum, A., & Nguyen, G. H. (2009). Learning Pattern Classification Tasks with Imbalanced Data Sets. In Pattern Recognition (pp. 193–208). https://doi.org/10.5772/7544

[22] Mitchell, T. M. (1997). Machine Learning (1st ed.). New York: McGraw-Hill, Inc.

[23] Ali, K. M., & Pazzani, M. J. (1996). Error Reduction through Learning Multiple Descriptions. Machine Learning, 24(3), 173–202.https://doi.org/10.1007/bf00058611

[24] Bartlett, P., & Shawe-Taylor, J. (1998). Generalization performance of support vector machines and other pattern classifiers. Cambridge: MIT Press Cambridge, USA.

[25] Seni, G., & Elder, J. F. (2010). Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. (R. G. (University of Illinois), Ed.) (1st ed.). San Rafael: Morgan and Claypool.

[26] Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. Neural Computation, 4(1), 1–58. https://doi.org/10.1162/neco.1992.4.1.1

[27] Wei, H., Lin, X., Xu, X., Li, L., Zhang, W., & Wang, X. (2014). A novel ensemble classifier based on multiple diverse classification methods. In 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2014 (pp. 301–305). Xiamen, China: IEEE. https://doi.org/10.1109/fskd.2014.6980850

[28] King, M. A. (2015). Ensemble Learning Techniques for Structured and Unstructured Data (PhD Thesis). Virginia Polytechnic Institute and State University.

[29] Duin, R. P. W., & Tax, D. M. J. (2000). Experiments with Classifier Combining Rules. In MCS 2000 Proceedings of the First International Workshop on Multiple Classifier Systems (Vol. 31, pp. 16–29). Cagliari, Italy. https://doi.org/10.1007/3-540-45014-9_2

[30] Stefanowski, J. (2009). Multiple classifiers. Retrieved June 24, 2018, from http://www.cs.put.poznan.pl/jstefanowski/aed/DMmultipleclassifiers.pdf

[31] Vidhya, A. (2017). How to handle Imbalanced Classification Problems in machine learning? Retrieved August 29, 2018, from https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/

[32] Gama, J. M. P. da. (1999). Combining Classification Algorithms (Doctoral Thesis). Faculdade de Ci^encias da Universidade do Porto.

[33] Komal, A. (2016). A Survey on malicious detection technique using data mining and analyzing in web security. 2016 IJEDR, 4(2), 319–322.

[34] Duch, W., & Grudzinski, K. (2001). Meta-learning: searching in the model space. In In Proceedings of the International Conference on Neural Information Processing (ICONIP) (pp. 235–240). Shanghai, China.

[35] Tan, P.-N., Steinbach, M., & Kumar, V. (2006). Classification : Basic Concepts , Decision Trees , and Model Evaluation Classification. Introduction to Data Mining, 1, 145–205.

[36] Breiman, L. (2001). RANDOM FORESTS. Machine Learning, 45(1), 5–32.

[37] Yerima, S. Y., & Sezer, S. (2018). DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. In IEEE Transactions on Cybernetics (pp. 1–14). IEEE. https://doi.org/10.1109/tcyb.2017.2777960
[38] Witten, I. H., & Frank, E. (2011). Data Mining: Practical Machine Learning Tools and Techniques (2nd ed.). AMSTERDAM: Elsevier.
[39] Ucci, D., Aniello, L., & Baldoni, R. (2018). Survey on the Usage of Machine Learning Techniques for Malware Analysis. Computers and Security, 1(1), 1–67. https://doi.org/10.1109/tcyb.2017.2777960
[40] Masud, M., Khan, L., & Thuraisingham, B. (2011). Data Mining Tools for Malware Detection (1st ed.). Boston: Auerbach Publications.
[41] Asiri, S. (2018). Machine Learning Classifiers. Retrieved January 4, 2019, from https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623
[42] Yajin, Z., & Xuxian, J. (2015). Android Malware Genome Project. Retrieved March 8, 2017, from http://www.malgenomeproject.org/policy.html
[43] Contagio, M. (2016). Pokemon GO with Droidjack - Android sample. Retrieved from http://contagiominidump.blogspot.com.ng/
[44] Whelan, R. J., Leek, T. R., Hodosh, J. E., Hulin, P. A., & Dolan-gavitt, B. (2016). Repeatable Reverse Engineering with the Platform for Architecture-Neutral Dynamic Analysis. MIT Lincoln Laboratory Lexington, 22(1), 90–99.
[45] Eilam, E. (2005). REVERSING Secret of Reverse Engineering (1st ed.). Indianapolis: Wiley Publishing, Inc.
[46] Moreno-Seco, F., Iñesta, J., León, P. De, & Micó, L. (2006). Comparison of classifier fusion methods for classification in pattern recognition tasks. Lecture Notes in Computer Science, 4109, 705–713. https://doi.org/10.1007/11815921_77
[47] Dˇzeroski, S., & Zenko, B. (2004). Is Combining Classifiers Better than Selecting the Best One? Machine Learning, 54(3), 255–273. https://doi.org/10.1023/b:mach.0000015881.36452.6e
[48] Fawcett, T. (2006). An introduction to ROC analysis. Journal of Pattern Recognition Letters, 27(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010
[49] Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure to Roc, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies, 2(1), 37–63.

7 Authors

Oluwakemi Christiana Abikoye received a National Diploma (ND) in Computer Science from Kwara State Polytechnic, Ilorin in 1996, a B.Sc degree in Computer Science from University of Ilorin in 2001, M.Sc degree in Computer Science from University of Ibadan, Ibadan in 2006 and Ph.D. degree in Computer Science in 2013.. She began her academic career at University of Ilorin, Department of Computer Science in 2004 as a Graduate Assistant and rose through the ranks. She is presently a Senior Lecturer. Oluwakemi is known for her innovative work in Computer/Communication Network Security, particularly on issues involving Security in computer, networks and Cash dispenser machines and transaction authentication system. Her research interests include Cryptography, Biometrics, Human-Computer Interaction, and Cybersecurity.

She is also involved in the supervision of postgraduate (Masters/Ph.D.) students' research work in specialized areas of Computer (Information Security). She has several publications in Local, national and international journals.

Benjamin Aruwa Gyunka graduated with a Doctor of Philosophy degree (Ph.D.) in Computer Science from the University of Ilorin, Nigeria. He also holds a Bachelor of Science degree in Mathematics from the University of Jos and a Master of Science degree in Information Systems Security from Sheffield Hallam University, United Kingdom. Prior to this time, he had extensive experience as a Network Administrator while working with the National Open University of Nigeria (NOUN). His research interest lies mostly in Information Security, Cybersecurity, data mining, Android security, digital forensics, and machine learning.

Akande Noah Oluwatobi had B. Sc. and M. Sc. degrees in Computer Science from Ladoke Akintola University of Technology. He presently lectures in the Department of Computer Science, Landmark University, Omu-Aran, Nigeria. He is a member of Computer Professional (Registration Council) of Nigeria (MCPN), Member, Nigeria Computer Society (MNCS), and IAENG Society of Computer Science. His research areas include Data and Information Security and Pattern Recognition (Medical Image Analysis). Email: akande.noah@lmu.edu.ng

Article submitted 2019-08-20. Resubmitted 2019-11-07. Final acceptance 2019-11-07. Final version published as submitted by the authors.