Malicious iOS Apps Detection Through Multi-Criteria Decision-Making Approach

Arpita Jadhav Bhatt[1*], Neetu Sardana[2]
Department of Computer Science & Engineering and Information Technology, Jaypee Institute of Information Technology, India
E-mail: arpitajadhav@gmail.com, neetu.sardana@jiit.ac.in
[*]Corresponding author

Keywords: multi-criteria decision making (MCDM), ensemble learning, analytic hierarchy process, iOS app, privacy leaks

Received: September 2, 2024

In today's era, smartphones are used in daily lives because they are ubiquitous and can be customized by installing third-party apps. As a result, the menaces because of these apps, which are potentially risky for user's privacy, have increased. Information on smartphones is perhaps, more personal than compared to data stored on desktops or computers, making it an easy target for intruders. After Android, the most prevalently used mobile operating system is Apple's iOS. Both Android and iOS follow permission-based access control to protect user's privacy. However, the users are unaware whether the app is breaching the user's privacy. To combat this problem, in the paper we propose a hybrid approach to detect malicious iOS apps based on its permissions. In the first phase, weights have been assigned to app permissions using multi-criteria decision-making (MCDM) approach namely Analytic Hierarchy Process (AHP), and in the second phase machine learning& ensemble learning techniques have been employed to train the classifiers for detecting malicious apps. To test the efficacy of the proposed method dataset comprising 1150 apps from 12 app categories has been used. The results demonstrate the proposed approach improves the efficacy of detecting malicious iOS apps for majority of categories.

Povzetek: Raziskava predlaga hibridni pristop za zaznavanje zlonamernih iOS aplikacij z uporabo analitičnega hierarhičnega procesa za dodelitev uteži dovoljenjem aplikacij in strojnega učenja za klasifikacijo, kar izboljša zaznavanje.

1 Introduction

Apple's iOS (iPhone Operating System) is one of the most widely used mobile operating systems after its counterfeit Android. Apple manufactures and distributes different types of iOS devices such as iPad, iPod touch, iPhone, Apple Watch, Apple TV, etc. Apple has a huge customer base because it provides a lot of unique features such as multitasking, multi-touch gestures, internal accelerometers, voice assistant Siri, etc.(Wikipedia). Apple also offers a platform for its developers to publish and distribute their applications also called apps through its online store 'App Store'. This online store is also a repository of billions of apps from which iOS users can download apps from different categories (Apple Inc.). With the presence of 1.96 million apps, the App Store is the world's second-largest online store than its counterpart Android which has 2.87 million apps on its official store Play Store. Such a fast-growing platform motivates developers, IT industries, marketing firms, and organizations to develop feature-rich apps. It also allures the customers or users to download these apps.

With the upsurge of smart devices, the number of apps, and the number of smartphone users, smartphones and smart devices have also become a device for storing a large amount of user's personal data (Erickson et al.). This includes personal information such as address book, photo gallery, email IDs, passwords, calendar events, etc. Additionally, a smartphone always generates contextual data via its sensors. Such crucial information of users is undoubtedly more personal when compared with the data, which is stored on personal computers because smartphones stay with individuals throughout the day and generate a lot of contextual data by sensors. These sensors are not present on personal computers; therefore, it can be inferred that smartphones contain a lot of user's personal data which makes them a valuable resource for the developers of malicious apps who might intend to develop privacy-infringing apps and access user's data.

To preserve the privacy of its users both Apple and Android follow a permission-based access control policy (Krupp). The policy ensures that the app will notify its users about all the permissions and resources the app will use during its run-time. Android follows an install-time and run-time permission policy in which the users are aware of the permissions the app will acquire during its usage during app installation. Whereas, Apple follows a run-time permission policy in which the users are informed about app permissions during app usage (Khan et al.). Additionally, Apple provides privacy controls through its inbuilt 'setting app', through which they can explicitly define the permission for every app which has been installed (Krupp). Additionally, Apple follows a strict code signing process in which the apps that have

been submitted by the developers on the App Store are critically examined before they are published. However, past attacks on iOS devices via privacy-infringing apps have demonstrated these methods adopted by Apple are inadequate to preserve the privacy of the users.

A lot of research work has been conducted to highlight the privacy breach by apps. Research conducted by Wired has identified that thousands of iOS and Android apps leak data from the cloud. The apps leaked lots of user's personal data such as medical information, phone number, device identifiers, passwords, etc.(WIRED). A similar kind of research conducted by Oxford researchers identified that a lot of third-party apps are sharing data with Facebook and Google. A total of 959,000 apps were analyzed. The study identified that 88% of these apps were transferring data to Alphabet which is Google's parent company, while Twitter, Facebook, Microsoft, and Amazon received 34%, 43%, and 18% of user's data respectively (Millman). To name a few the data collected by these companies included age, location, gender, etc. A report by Pt security highlighted the threats by the third-party mobile apps which included insecure data storage by apps, Escalated privileges taken by the app, side-loaded software, client-side vulnerabilities, etc.(Ptsecurity.com).

The issue of privacy leaks by apps also increases because of the issues in the current permission model adopted by Apple. The users do not have fine-grained access control over the data which is shared by the app in use. Smartphones do offer coarse-grained privacy as well as security controls where the users can either deny or allow permission for an explicit resource by an app. However, the problem with this approach is that once a user grants permission to the app he/she does not have any control over restricting the app from sharing the data. For example, once a user grants location permission to an app, the user cannot restrict the app from accessing a particular location. Likewise, once a user grants permission to access the address book, the user cannot restrict the app from accessing a specific contact. Since the users do not have the option to specify the accuracy of the sensors while accessing location, device accelerometer, and locally shared data, thus the apps must restrict their access. However, developers of malicious apps intentionally create over-privileged apps. Users are allured by the features of apps and thus grant permissions to the friendly-looking apps. However, the users never know if the app is using their data locally or sharing it with third-party domains without their consent. To eradicate this, problem we present a privacy detection model that uses app permissions during static analysis. The model first extracts user permission using the concept of reverse engineering and constructs a Boolean value permission matrix P_mxn where m represents the number of apps under analysis and n represents no. of permissions. Here a total of 1150 iOS apps from 12 app categories are tested for 10 different user permissions. Machine learning and ensemble-based techniques are employed to train the classifiers and determine malicious iOS apps. Apps are reverse-engineered from 12 different categories. In order to improve the precision of classifiers, the correlation of permissions for each category has been computed using Analytic Hierarchy Process (AHP). Later, the weighing factors obtained from AHP for each permission category-wise have been used to construct a weighted permission matrix P[mxn] to train the classifiers. The motivation for using a weighted permission matrix is that every permission has a different weight for a category. Apple provides a set of predefined app categories on the App Store which helps the developer to choose the best category before uploading the app. The category also defines the necessary features that the app provides the users during its usage. For example, the category navigation specifies that the apps belonging to the navigation category will fetch the user's location to guide them and navigate them. Likewise, an app belonging to the photo and video category will require access to the user's photo gallery and camera to serve its intended purpose. Apple provides a limited set of permissions, which require explicit approval during app usage. The problem of privacy leaks exists because it is very difficult to determine a benign app, an over-privileged app, or a malicious app with this limited permission set. Even the operating system cannot determine the intention of an app during its run-time. In other words, it is very difficult to identify malicious iOS apps as there exists a thin boundary line to identify how well a permission is correlated to a category. Using our approach, we identify the most significant permissions within a category using AHP approach that helps in identifying the malicious iOS apps. To address the challenges in the existing model for handling privacy breaches, we use the AHP technique to find the correlation of permission for a category to detect privacy violations by apps. For example, the permission of a user's location is an essential feature for an app belonging to the 'navigation' category because it functions after it receives the user's approval to access the GPS location. The same permission might have a different weight for the 'books' category as the prime purpose of this category is to provide stories, comics, graphic novels, and interactive content for which location permission may not be mandatory permission. Based on the aforementioned facts and guidelines provided by Apple we propose a heuristic approach to determine privacy leaks by iOS apps.

The foremost contributions of the paper have been listed below.

- A novel hybrid approach that integrates MCDM approach, AHP with Machine learning & ensemble learning techniques has been proposed to detect malicious iOS apps.

Table 1: Summary of research works on detecting privacy breach

Authors	Objective	Technique/Method	Data Set
Abdirashid et al. (Sahal et al.)	Develop permission-based malware detection model	Utilize feature selection techniques	1000 samples of apk files
Jin et al. (Sun et al.)	Develop malware detection system SigPID	Identify the most significant app permission to classify malware apps	310926 benign Android apps, 5494 malicious Android apps
Wang et al. (Wang et al.).	Explore the permission-induced risk to classify Android apps	Ranking of permissions, identification of malicious apps using PCA and SFS	310,926 benign and 4868 malicious apps
Jing et al. (Jing et al.)	Develop a risk assessment framework	Computes a risk assessment baseline by monitoring the run time behaviour of apps	14 Android apps
Adhikari et al. (Adhikari et al.)	Analyze run time behaviour of health care apps	Computation of safe score and risk score of iOS apps during their usage	20 iOS apps
Kang et al.(Kang et al.)	Malware detection using static analysis	Permission-based analysis of malware	51,179 benign Android apps and 4,554 malware Android apps
Huang et al.(Huang et al.)	Detect malicious Android apps based on their permissions	Grouping of permissions to Boolean vector and then training in machine learning classifiers	124,769 benign Android apps and 480 Malicious Android apps

- The proposed approach has been evaluated on 1150 apps belonging to twelve category iOS apps. Each app possesses ten permissions.
- The proposed approach improves the malware detection accuracy best case value of 14%.

The rest of the paper is organized as follows. Section 2 describes the related work on machine learning and ensemble techniques used to detect malicious apps. The section also describes the techniques that have been used in this paper. Section 3 describes our proposed heuristic approach of multicriteria decision-making approach using AHP. Section 4 describes the experimental results and analysis. The paper is concluded in Section 5.

2 Machine learning, ensembling techniques to detect privacy leaks based on app permissions

Machine learning and ensembling learning techniques have been employed by many researchers to detect privacy violations by apps. Ping et al. have developed an ensemble classifier 'Enclamald' to identify the contrasting permission patterns to illustrate the important difference between malicious apps the benign ones based on permission usage(Xiong et al.). Liu et al. have proposed a two-layered permission-based detection model for Android apps. In their work, they considered both requested permission and used permission by apps to detect malicious apps using machine learning techniques (Liu and Liu). Congyi et al. used an implementation of ensemble learning- XGBoost method to detect malicious Android apps based on permission usage(Congyi and Guangshun). Alba et al. used feature selection and ensembling techniques to classify Android malware(Coronado-De-Alba et al.). Idrees et al. proposed a model PIndoid, permissions, and intent-based framework, to detect malicious Android apps. It uses a combination of permissions and intents integrated with the ensemble method to improve the malware detection accuracy(Idrees et al.). Abdirashid et al. proposed a model to detect unknown malware by using a permission-based approach to enhance the accuracy as well as efficacy(Sahal et al.). The authors have improved the feature selection technique by incorporating weighing method TF-IDFCF, based on the class frequency of the app features. Jin et al. developed a malware detection system SigPID capable of coping with malware and its variants(Sun et al.). The authors have used three levels of pruning to mine app permission and identify the most significant permission capable of distinguishing malware apps from benign ones. Wang et al. have applied different ranking algorithms to classify malicious Android apps based on different ranking techniques namely principal component analysis(PCA) and sequential forward selection (SFS) to detect risky app permissions along with their subsets(Wang et al.). The authors have used a data set of 310926 benign and 4868 malicious apps and employed several machine learning classifiers. Jing Y et al. developed an automated risk assessment framework RiskMon which utilizes machine learning models to rank and assess the risks by Android apps(Jing et al.). The highlight of the tool was that it continuously monitors the behaviour of Android apps by combining the expectations of app users with their run-time behaviour. Run time behaviour of 20 iOS mobile health care iOS apps was analyzed by Adhikari R et al.to determine their strengths and weaknesses by assigning a safe score and risk score to them based on their run time analysis(Adhikari et al.).

Table 1 details the summary of research works on detecting privacy breaches.

However, the limitation of the above approaches is the differentiation of benign apps from malicious apps if they all request a similar set of app permissions. Since in Android, a large set of app permissions (approximately 320+) is already available, hence application of feature selection techniques, machine learning techniques, and reverse engineering is easy. However, in the case of the iOS platform a limited set of 10-13 permissions is available which varies with the iOS version. Hence, distinguishing a benign app from a malicious app is very difficult because there exists a thin boundary between a malicious and a benign app.

As most of the work has been done for the Android platform, in this paper we propose the detection of malicious apps for the iOS platform using AHP and MCDM approach to detect malicious iOS with a minimal permission set. We have also employed several ensembling techniques. The machine learning classifiers that were used for evaluating the proposed method are listed below(Mesevage), (Abaker and Saeed; Chehal et al.), (Harahsheh and Chen).

(i) **Naïve Bayes (NB):** It is a Bayesian classification method and is based on Bayes' Theorem. The Bayesian classification method builds a probabilistic classifier that is based on modelling the underlying features for different classes. The classification technique predicts class member probability that a given sample/tuple belongs to a particular class. The advantages of using this technique are that it needs less training data, is highly scalable, and can be used for both multi-class as well as binary classification problems(Mesevage).

(ii) **Decision Tree (DT):** It constructs a tree structure in a top-down recursive manner based on the divide and conquer manner. The decision tree is a tree structure where the internal node represents a test on an attribute every branch represents the output of the test and the leaf nodes depict the class distribution and are easy to interpret(Chehal et al.).

(iii) **Random Forest (RF):** Random Forest generates many classification trees. Every tree gives a classification and the forest selects the classification that has the most votes [15].

(iv) **Neural Network (NN):** The basic unit of a neural network is neurons, which take inputs, perform mathematical computations with them, and generate output. Every input is multiplied by a weight, and then all weighted inputs are added together with a bias function, and then the sum is passed through an activation function(Zhou).

(v) **Support Vector Machine (SVM):** It is a fast machine learning algorithm used for solving multiclass classification problems for larger data sets. It can work with high-dimensional data comprising thousands of features and attributes. The algorithm can be used in text classification problems with high-dimensional spaces [15].

The ensembling approaches that have been employed are bagging using J48 (decision tree) and boosting.

(vi) **Bagging (Bg):** The technique is based on creating multiple subsets from the original dataset. The instances from the dataset are selected with replacements. Then a base model also called a weak model is created for each of the subsets. The models are run in parallel and they work independently of each other. The final predictions are computed by combining the predictions from all models(Idrees et al.).

(vii) **Boosting (Bo):** In boosting a subset is created from the original dataset and instances are given equal weights initially. The base model is constructed on the previously created subset and is utilized to make predictions for the complete dataset. Errors are determined to employ real and anticipated values. Higher weights are allocated for the perceptions that are incorrectly anticipated. A strong learner is defined using the weighted mean of weak learners(Idrees et al.).

The following section describes the proposed approach to determine malicious iOS apps using a multi-criteria decision-making approach.

3 Classifying iOS Apps using proposed hybrid approach

Figure 1 shows the proposed framework to classify iOS apps using machine learning and ensemble techniques based on ranked permissions. Permissions are ranked using AHP. In the proposed method, the apps are installed from the AppStore, and their features are fetched (here features refer to the app permissions such as location, camera, photo gallery, etc.). We have considered ten features for twelve categories of iOS apps. Each app has a set of features in the form of a permission vector. Generally, permission for each feature is either present or absent corresponding to an app, and the features if present are considered to be equally important. In reality, the features of each category app have different weights. Based on this belief, we have ranked the permission set of each category of apps. On the basis of permission usage across the category, we have applied correlation coefficient and ranked permissions across the category. For example, an app belonging to the Social Networking category can have app permissions like photo, camera, location, internet, etc. whereas a simple flashlight app from the utilities category may require only camera permission. Thus, the ranking of permission for camera would be entirely different in social networking and utility category.

The proposed method has two phases. In the first phase, the app features are assigned weights based on the app category. In the second phase, the classification algorithms are applied for the identification of malicious iOS apps.

Assigning weights to app permissions

To rank the permissions of an app we are using the AHP, a MCDM approach. Step 1 is to identify the AHP Hierarchy. In this step, first, the goal is defined. In this work, the goal is to classify an iOS app as Malign or benign. Second, the criteria are identified. Here, the criterion is the apps category.

Figure 1: Framework to classify iOS App using machine learning and ensemble technique based on ranked permissions

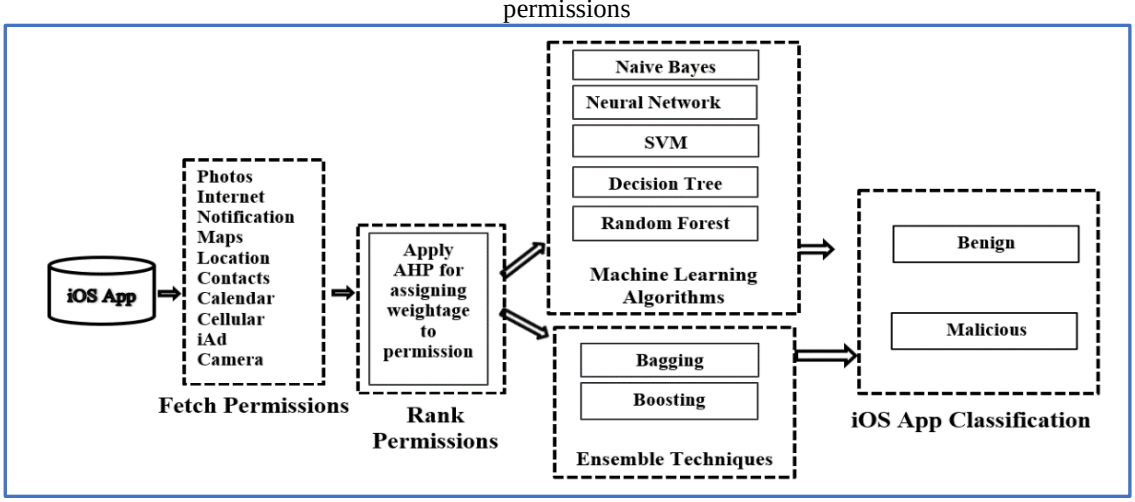
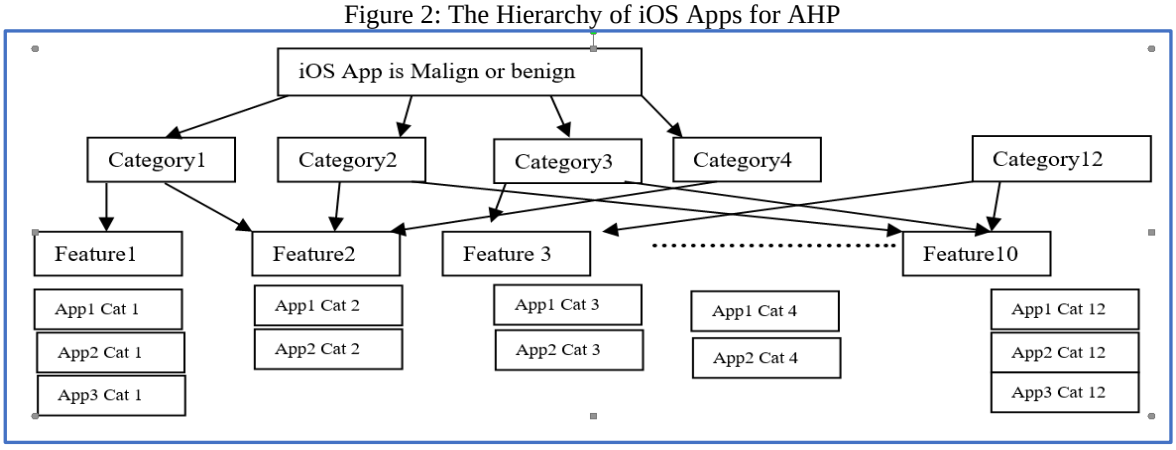


Figure 2: The Hierarchy of iOS Apps for AHP



Twelve mobile app categories have been considered. The categories are Books, Navigation, Games, Education, Health & Fitness, Lifestyle, Music, Utilities, Photo & Video, Sports, Social Networking, and Finance(Bhatt et al.). Third, the sub-criterion is defined if present. Here, the sub-criterion is the feature set of each category app for which permissions are present.AHP has been used to identify important features of each category app. The bottom level consists of the various alternatives. In our problem, 1150 apps belonging to twelve categories are the varied alternatives. The AHP hierarchy for ranking permissions of each category app is shown in Figure 1.

In step 2, pair-wise comparisons among features are performed to create a judgement matrix. We consider the ten feature sets of all apps belonging to each category. The features set consists of photos, internet, notification, map, location, contacts, calendar, cellular data, iAd, and camera. The Pearson correlation coefficient is utilized to find the feature importance for a particular category app. Then we used the Pearson correlation coefficient to perform a pair-wise comparison among features to find the relative importance of each pair and are given values in the range of 1 to 9. Table 2 has been used for the creation of a judgement matrix that provides the relative ranking that signifies the intensity of importance among the considered feature pair. The order of the judgement matrix depends upon the number of elements that the level of comparison. Since we are comparing the 10 feature pairs so for each category, the matrix of dimension 10*10 is formed. As the judgement matrices are formed, the eigenvectors and maximum eigenvalue (λmax) for each matrix are computed. Later consistency index (CI) and consistency ratio (CR) are calculated as shown in Algorithm 1. RI is a random consistency index given by Saaty for n varying from 1 to 10 (Refer Table 3). The acceptable value of CR is less than 0.1. If the CR value exceeds 0.1, it represents inconsistencies and the result is meaningless. Thus, the entire process requires re-evaluations(Saaty).

Table 2: Criteria for comparison

Intensity of Importance	**Definition**
1	Equal importance
3	Weak importance of C_i over C_j
5	Essential or strong importance
7	Demonstrated importance
9	Absolute importance
2,4,6,8	Intermediate
Reciprocals	If C_i has one of the above judgements assigned to it when compared with C_j has the reciprocal value when compared with C_i

Table 3: RI values

Size	1	2	3	4	5	6	7	8	9	10
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Step1: Define AHP hierarchy. (Refer Figure 2)

1. Goal: To identify an iOS app is benign or malicious
2. Criteria: Twelve iOS App Categories
3. SubCriteria: Feature vectors of each category app.
4. Alternatives: 1150 apps belonging to different categories.

Step 2: Pairwise comparison is performed to find relative ranking among features of each category apps. (Refer Table 1)

Step 3: Compute Judgement Matrix (M) of dimension 10*10 for each category apps. (Illustrated in Table 5)

Step 4: Calculate M_n, normalized judgement matrix which can be obtained by dividing each element with the column sum.

Step 5: Find the average of all the row elements of $M_n$to get eigenvectors W^T having dimension 10*1 that are considered as the weights of each feature if the matrix is consistent.

Step 6: Check the consistency of the matrix, M.

a. The maximum eigen value (λ_{max}) for each matrix is calculated.

$$\lambda max = \frac{1}{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{n}\cdot} \frac{\text{ith entry in } \mathrm{MW}^T}{\text{ith entry in } \mathrm{W}^{\mathrm{T}}}$$

b. The consistency index (CI) of each matrix of order 10 is calculated

$$CI = \frac{(\lambda max - n)}{(n-1)} \text{Here n =10 (number of app features)}$$

c. Compute consistency ratio (CR)

$$CR = \frac{CI}{RI}\ Here, RI = 1.49$$

Algorithm 1 : Ranking permission of varied category Apps using AHP

The judgement matrix is found to be consistent if the value of CR is less than 0.1. The weights of features are fetched.

Apply classification algorithms to identify the benign and malign iOS apps

The feature weights computed by applying AHP are used for each category of apps and are used to train the classifiers. Five machine learning and two ensemble learning techniques are applied for the identification of malign and benign apps. The machine learning techniques that are considered are Naïve Bayes, Neural Network, Random Forest, Decision Tree, and SVM.

The ensemble learning techniques are applied as the feature set in some of the categories is found to be skewed. The techniques used are bagging and boosting where J48 is used as a baseline technique.

4 Experimental results and analysis

This section presents the experimental setup and results of the proposed model on 12 categories of iOS 1150 apps.

4.1 Experimental setup

The proposed permission weighting approach is evaluated on twelve categories of iOS apps. The categories are Books (Bks), Education (Edu), Finance (Fin), Games (Gam), Health & Fitness(H&F), Lifestyle (LS), Music (Mus), Navigation (Nav), Photo&Video(P&V), Social Networking (SN), Sports (Sp) and Utilities (Util). There are a total of 1150 iOS apps that have been considered. The apps are fetched from the App Store. Table 5 shows the distribution of apps in each category that has been considered.

Table 4: Category wise apps distribution

Category	**Apps Count**
Books	126
Education	141
Finance	91
Games	201
Health & Fitness	81
Lifestyle	101
Music	91
Navigation	46
Photo & Video	61
Social Networking	91
Sports	46
Utilities	86
Total	**1150**

Each category app defines the important features that it serves to the users. The features set consists of photos, internet, notification, map, location, contacts, calendar, cellular, iAd, and camera. The permission log of these features has been extracted. Each app has its own set of feature vectors, for example, App1 of the Books category has feature vector fv= (1,0,0,1,0,1,0,0,0,0) which depict that this app has three features: photos, map, and contacts. The different features of apps are generally given equal importance. Here photos, map, and contacts are given equal importance. It has been noticed that certain features of apps are more important than other features. Based on this belief, this paper prioritizes the features of an app using the AHP, MCDM technique to classify the app as malicious or benign.

The proposed method undergoes two phases to detect malicious iOS apps: In the first phase the features of each category of iOS apps are ranked using AHP and in the second phase machine learning as well as ensemble learning methods are applied to ranked features. Finally, the performance of both approaches is evaluated using precision defined by equation 1. Here, precision is a metric used to evaluate the model's positive classifications which are actually positive. It is defined as a ratio of true positive (TP) predictions and total number of predicted positives which includes both false positives (FP) as well as true positives. Precision improves when false positives decrease.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

4.2 Experimental results

This section presents the results obtained from our experimental study. During the first phase, the features of varied category apps were ranked using AHP. The feature's importance has been computed using the correlation coefficient. The results of which are given in Table 4 and have been taken from our previous work on the detection of malicious iOS apps using static and dynamic analysis approaches (Bhatt et al.). The work is an extension of previous work by including ensembling and multi-criteria decision-making approach. The correlation values are used for pair-wise feature comparison. The judgement matrix has been computed for each category app using pairwise feature comparison. Figures 3a-3l represent the Judgement Matrix corresponding to each category of apps.

Table 5: Features Importance in each category apps

Bks	**Edu**	**Fin**	**Gam**
iAd	Notifi.	Notifi.	iAd
Photos	MapKit	Location	Notifi.
Internet	Cellular	MapKit	Cellular
Notifi.	Location	Camera	Calendar
Contacts	Camera	Contacts	Photos
MapKit	Calendar	Cellular	Location
Calendar	Contacts	Photos	Internet
Cellular	Internet	Calendar	Camera
Location	Photos	iAd	MapKit
Camera	iAd	Internet	Contacts
H&F	**LS**	**P&V**	**SN**
MapKit	Notifi.	Notifi.	Notifi.
Notifi.	MapKit	Camera	Cellular
Contacts	Cellular	iAd	Camera
iAd	Contacts	MapKit	Calendar
Calendar	Location	Calendar	Location
Camera	iAd	Cellular	iAd
Photos	Calendar	Internet	Contacts
Cellular	Photos	Location	MapKit
Location	Camera	Photos	Photos
Internet	Internet	Contacts	Internet

Spo	Util	Mus	Nav
Cellular	MapKit	Cellular	Cellular
Calendar	Cellular	Notifi.	Calendar
MapKit	Calendar	MapKit	Notifi.
Notifi.	Camera	Calendar	Internet
Camera	Location	Internet	Photos
Photos	Internet	Location	iAd
Contacts	Photos	Photos	MapKit
Location	Contacts	Camera	Camera
Internet	Notifi.	iAd	Contacts
iAd	iAd	Contacts	Location

Bks	N	iA	Ce	Cl	M	I	L	Ca	P	Co
N	1	3	2	2	3	4	5	6	7	8
iA	0.333	1	2	3	2	2	3	3	3	4
Ce	0.50	0.50	1	1	1	1	3	3	9	1
Cl	0.50	0.33	1.00	1	1	1	6	4	3	3
M	0.33	0.50	1.00	1.00	1	1	1	4	4	4
I	0.25	0.50	1.00	1.00	1.00	1	5	2	3	2
L	0.20	0.33	0.33	0.17	1.00	0.20	1	4	1	3
Ca	0.17	0.33	0.33	0.25	0.25	0.50	0.25	1	2	2
P	0.14	0.33	0.11	0.33	0.25	0.33	1.00	0.50	1	4
Co	0.13	0.25	1.00	0.33	0.25	0.50	0.33	0.50	0.25	1

Figure 3 a: Judgement Matrix for Apps from (Books (Bks) category

Edu	N	M	Ce	Co	I	L	Cl	P	iA	Ca
N	1	1	2	2	5	6	7	7	8	9
M	1	1	4	4	5	5	6	8	9	6
Ce	0.50	0.25	1	2	1	5	8	8	2	6
Co	0.50	0.25	0.50	1	1	6	6	9	2	6
I	0.20	0.20	1.00	1.00	1	1	5	7	8	4
L	0.17	0.20	0.20	0.17	1.00	1	1	7	1	1
Cl	0.14	0.17	0.13	0.17	0.20	1.00	1	1	1	4
P	0.14	0.13	0.13	0.11	0.14	0.14	1.00	1	1	2
iA	0.13	0.11	0.50	0.50	0.13	1.00	1.00	1.00	1	2
Ca	0.11	0.17	0.17	0.17	0.25	1.00	0.25	0.50	0.50	1

Figure 3 b: Judgement Matrix for Apps from Education (Edu) category

Gam	N	iA	Ce	P	Cl	L	I	Ca	M	Co
N	1	3	4	3	4	5	6	7	7	8
iA	1	1	4	5	4	5	5	5	5	7
Ce	0.25	0.25	1	6	4	7	8	8	7	9
P	0.33	0.20	0.17	1	4	4	4	4	5	5
Cl	0.25	0.25	0.25	0.25	1	3	4	4	4	5
L	0.20	0.20	0.14	0.25	0.33	1	2	2	1	1
I	0.17	0.20	0.13	0.25	0.25	0.50	1	1	1	1
Ca	0.14	0.20	0.13	0.25	0.25	0.50	1.00	1	2	2
M	0.14	0.20	0.14	0.20	0.25	1.00	1.00	0.50	1	1
Co	0.13	0.14	0.11	0.20	0.20	1.00	1.00	0.50	1.00	1

Figure 3 c: Judgement Matrix for Apps from Games (Gam) category

H&F	N	M	Ce	Co	I	L	Cl	P	iA	Ca
N	1	6	9	9	9	7	9	9	9	9
M	0.167	1	6	7	7	8	9	9	9	9
Ce	0.11	0.17	1	2	2	3	6	9	9	9
Co	0.11	0.14	0.50	1	2	3	2	2	3	8
I	0.11	0.14	0.50	0.50	1	4	3	2	2	2
L	0.14	0.13	0.33	0.33	0.25	1	2	1	2	2
Cl	0.11	0.11	0.17	0.50	0.33	0.50	1	1	3	2
P	0.11	0.11	0.11	0.50	0.50	1.00	1.00	1	1	1
iA	0.11	0.11	0.11	0.33	0.50	0.50	0.33	1.00	1	2
Ca	0.11	0.11	0.11	0.13	0.50	0.50	0.50	1.00	0.50	1

Figure 3 d: Judgement Matrix for Apps from Health & Fitness (H&F) category

Mus	N	Ce	L	M	Co	Cl	P	iA	Ca	I
N	1	7	3	3	6	7	9	9	9	9
Ce	1	1	7	5	5	7	6	7	4	9
L	0.14	0.14	1	5	3	6	4	5	8	5
M	0.33	0.20	0.20	1	1	1	1	2	4	3
Co	0.17	0.20	0.33	1.00	1	1	1	4	3	3
Cl	0.14	0.14	0.17	1.00	1.00	1	1	2	3	3
P	0.11	0.17	0.25	1.00	1.00	1.00	1	3	3	3
iA	0.11	0.14	0.20	0.50	0.25	0.50	0.33	1	1	3
Ca	0.11	0.25	0.13	0.25	0.33	0.33	0.33	1.00	1	1
I	0.11	0.11	0.20	0.33	0.33	0.33	0.33	0.33	1.00	1

Figure 3 e:Judgement Matrix for Apps from Music (Mus) category

Nav	Ca	Ce	Cl	Co	I	L	M	P	N	iA
Ca	1	8	5	5	9	6	4	7	9	9
Ce	0.125	1	6	5	5	9	7	7	9	8
Cl	0.20	0.17	1	4	4	4	4	7	7	7
Co	0.20	0.20	0.25	1	3	3	2	3	3	3
I	0.11	0.20	0.25	0.33	1	2	3	2	3	7
L	0.17	0.11	0.25	0.33	0.50	1	1	1	2	3
M	0.25	0.14	0.25	0.50	0.33	1.00	1	1	2	3
P	0.14	0.14	0.14	0.33	0.50	1.00	1.00	1	1	3
N	0.11	0.11	0.14	0.33	0.33	0.50	0.50	1.00	1	1
iA	0.11	0.13	0.14	0.33	0.14	0.33	0.33	0.33	1.00	1

Figure 3 f:Judgement Matrix for Apps from Navigation (Nav) category

SN	N	Ce	L	Cl	Ca	Co	iA	P	M	I
N	1	2	3	3	3	5	5	7	8	9
Ce	1	1	3	5	4	4	6	6	6	6
L	0.33	0.33	1	1	3	3	3	3	3	3
Cl	0.33	0.20	1.00	1	5	4	6	7	9	7
Ca	0.33	0.25	0.33	0.20	1	6	5	7	5	5
Co	0.20	0.25	0.33	0.25	0.17	1	2	1	2	2
iA	0.20	0.17	0.33	0.17	0.20	0.50	1	1	3	3
P	0.14	0.17	0.33	0.14	0.14	1.00	1.00	1	1	3
M	0.13	0.17	0.33	0.11	0.20	0.50	0.33	1.00	1	3
I	0.11	0.17	0.33	0.14	0.20	0.50	0.33	0.33	0.33	1

Figure 3 g: Judgement Matrix for Apps from Social Networking (SN) category

Fin	N	M	Ce	Co	I	L	Cl	P	iA	Ca
N	1	1	2	3	4	5	7	8	9	9
M	1	1	3	1	4	4	2	8	7	9
Ce	0.50	0.33	1	1	5	5	2	6	6	9
Co	0.33	1.00	1.00	1	2	4	2	3	2	8
I	0.25	0.25	0.20	0.50	1	2	4	5	5	7
L	0.20	0.25	0.20	0.25	0.50	1	5	4	5	3
Cl	0.14	0.50	0.50	0.50	0.25	0.20	1	2	2	4
P	0.13	0.13	0.17	0.33	0.20	0.25	0.50	1	2	4
iA	0.11	0.14	0.17	0.50	0.20	0.20	0.50	0.50	1	2
Ca	0.11	0.11	0.11	0.13	0.14	0.33	0.25	0.25	0.50	1

Figure 3 h: Judgement Matrix for Apps from Finance (Fin)category

P&V	I	N	Ca	iA	Cl	M	Ce	co	L	P
I	1	1	2	2	4	4	6	6	8	9
N	1	1	2	1	4	3	3	3	5	3
Ca	0.50	0.50	1	2	1	6	6	4	4	3
iA	0.50	1.00	0.50	1	3	3	3	3	1	3
Cl	0.25	0.25	1.00	0.33	1	1	4	4	3	1
M	0.25	0.33	0.17	0.33	1.00	1	2	5	4	2
Ce	0.17	0.33	0.17	0.33	0.25	0.50	1	1	4	2
co	0.17	0.33	0.25	0.33	0.25	0.20	1.00	1	2	2
L	0.13	0.20	0.25	1.00	0.33	0.25	0.25	0.50	1	2
P	0.11	0.33	0.33	0.33	1.00	0.50	0.50	0.50	0.50	1

Figure 3 i: Judgement Matrix for Apps from Photo & Video (P&V) category

Spo	Ce	Cl	N	M	Ca	P	I	Co	L	iA
Ce	1	1	4	4	4	4	4	4	3	3
Cl	1	1	2	2	5	4	4	4	3	3
N	0.25	0.50	1	1	3	3	4	4	4	3
M	0.25	0.50	1.00	1	1	4	3	3	4	3
Ca	0.25	0.20	0.33	1.00	1	1	3	4	4	3
P	0.25	0.25	0.33	0.25	1.00	1	1	4	3	3
I	0.25	0.25	0.25	0.33	0.33	1.00	1	1	4	3
Co	0.25	0.25	0.25	0.33	0.25	0.25	1.00	1	1	3
L	0.33	0.33	0.25	0.25	0.25	0.33	0.25	1.00	1	3
iA	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.50	2

Figure 3 j: Judgement Matrix for Apps from Sports (Spo)category

Util	N	iA	Ce	Cl	M	I	L	Ca	P	Co
N	1	1	3	3	4	4	4	5	7	7
iA	1	1	3	2	2	2	2	2	1	2
Ce	0.33	0.33	1	4	4	3	3	3	4	4
Cl	0.33	0.50	0.25	1	3	3	3	2	3	1
M	0.25	0.50	0.25	0.33	1	2	2	2	3	3
I	0.25	0.50	0.33	0.33	0.50	1	4	2	2	2
L	0.25	0.50	0.33	0.33	0.50	0.25	1	2	2	2
Ca	0.20	0.50	0.33	0.50	0.50	0.50	0.50	1	2	2
P	0.14	1.00	0.25	0.33	0.33	0.50	0.50	0.50	1	2
Co	0.14	0.50	0.25	1.00	0.33	0.50	0.50	0.50	0.50	1

Figure 3 k: Judgement Matrix for Apps from Utilities (Util)category

LS	N	Ce	L	Co	M	Cl	iA	P	Ca	I
N	1	2	3	3	5	5	7	7	9	9
Ce	0.5	1	6	3	6	3	3	1	4	7
L	0.33	0.17	1	4	4	1	1	6	6	7
Co	0.33	0.33	0.25	1	1	3	1	1	3	5
M	0.20	0.17	0.25	1.00	1	1	1	3	4	2
Cl	0.20	0.33	1.00	0.33	1.00	1	1	1	2	3
iA	0.14	0.33	1.00	1.00	1.00	1.00	1	1	2	4
P	0.14	1.00	0.17	1.00	0.33	1.00	1.00	1	3	3
Ca	0.11	0.25	0.17	0.33	0.25	0.50	0.50	0.33	1	1
I	0.11	0.14	0.14	0.20	0.50	0.33	0.25	0.33	1.00	1

Figure 3 l: Judgement Matrix for Apps from Lifestyle (LS) category

The procedure given in Algorithm 1 has been followed and the final weights of the ten features for all the twelve category Apps are shown in Table 6. Table 7 shows the Consistency Index (CI) and inconsistency ratio (CR) values obtained for each iOS category app. A CR value below 0.1 suggests that the comparisons made are consistent and that the judgments used in the AHP process are reliable. In our analysis, it can be clearly observed from the tables 6 & 7 that the CR values obtained were consistently below this threshold for all category iOS apps, which means that the pairwise comparisons of the permissions were logical and coherent.

Table 6: Final weights of all features for twelve category apps

apps**Features**	**Weights (Bks)**	**Features**	**Weights (Edu)**	**Features**	**Weights (Gam)**	**Features**	**Weights (H&F)**
N	0.260416	N	0.22869837	N	0.258617391	N	0.373378406
iA	0.1536197	M	0.25874097	iA	0.217896213	M	0.239753234
Ce	0.1118894	Ce	0.13251323	Ce	0.196327424	Ce	0.127111058
Cl	0.1130392	Co	0.11850109	P	0.108141669	Co	0.070634637
M	0.0972874	I	0.1023565	Cl	0.080053713	I	0.054553629
I	0.0941756	L	0.0464333	L	0.034984294	L	0.03462572
L	0.0570132	Cl	0.03384061	I	0.025581562	Cl	0.030962473
Ca	0.0387621	P	0.02371545	Ca	0.030363325	P	0.025841841
P	0.0395886	iA	0.03329195	M	0.025065278	iA	0.023556455
Co	0.0342088	Ca	0.02190853	Co	0.02296913	Ca	0.019582547
Features	**Weights (Mus)**	**Features**	**Weights (Nav)**	**Features**	**Weights (SN)**	**Features**	**Weights (Spo)**
N	0.3164822	Ca	0.32905485	N	0.243144501	Ce	0.226412065
Ce	0.2583536	Ce	0.2360261	Ce	0.230117625	Cl	0.193037193
L	0.1466667	Cl	0.14566162	L	0.098318076	N	0.130405534
M	0.0571527	Co	0.07414116	Cl	0.158169247	M	0.114890071
Co	0.0561844	I	0.06437649	Ca	0.110922948	Ca	0.089492707
Cl	0.0476753	L	0.03690112	Co	0.040755572	P	0.070530109
P	0.0505276	M	0.04092982	iA	0.03808108	I	0.058589306
iA	0.0271474	P	0.03279818	P	0.032758226	Co	0.043199994
Ca	0.0210871	N	0.0221352	M	0.028127397	L	0.039473318
I	0.018723	iA	0.01797546	I	0.019605329	iA	0.033969703
Features	**Weights (Fin)**	**Features**	**Weights (Util)**	**Features**	**Weights (P&V)**	**Features**	**Weights (LS)**
N	0.2430282	N	0.24600855	I	0.239110134	N	0.286591226
M	0.200416	iA	0.14601401	N	0.169841704	Ce	0.198462154
Ce	0.151552	Ce	0.15944265	Ca	0.154772531	L	0.141693949
Co	0.1195261	Cl	0.10371867	iA	0.119366316	Co	0.079178536
I	0.0901974	M	0.08086139	Cl	0.081263199	M	0.064995197
L	0.0727182	I	0.07471276	M	0.074714065	Cl	0.056637209
Cl	0.0504631	L	0.055601	Ce	0.047334716	iA	0.061643922
P	0.0312343	Ca	0.04967195	co	0.040725409	P	0.064021004
iA	0.0256206	P	0.04642525	L	0.037987177	Ca	0.025483038
Ca	0.015244	Co	0.03791838	P	0.03488475	I	0.021293766

Table 7: Consistency Index (CI) and inconsistency ratio (CR) values

Sno	Category	CI	CR
1	Books	0.14527	0.097497
2	Education	0.146455	0.098292
3	Finance	0.136704	0.091747
4	Games	0.145712	0.097794
5	Health & Fitness	0.135315	0.090816
6	Lifestyle	0.144084	0.096701
7	Music	0.14871	0.099806
8	Navigation	0.143824	0.096526
9	Photo & Video	0.1444	0.096913
10	Social Networking	0.142957	0.095944
11	Sports	0.135897	0.091206
12	Utilities	0.147348	0.098891

A detailed illustration of AHP steps has been omitted for the sake of the length of the paper. The results of this evaluation were analyzed to find whether the inclusion of AHP in prioritizing and determining the weights of features improves the accuracy of iOS app classification or not. We have used the cross-validation technique in the Weka toolkit to measure the efficiency of the models. Generally, whenever an inadequate amount of data instances is available, cross-validation method is preferred to accomplish an unbiased approximation of the model performance. In the k-fold cross-validation technique, the dataset is divided into k subsets, each of equal size. The model is constructed 'k' times, each time using (k−1) sets of data instances for training the classifier and leaving out one subset as a 'test set' for predictions. We considered five machine-learning techniques and two ensemble-based techniques for classification. The machine learning classifiers that are used for evaluating the proposed method are Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Neural Network (NN), and Support Vector Machine (SVM) and the considered ensemble approaches are bagging using J48(Bg) and Boosting (Bo). We compared the proposed AHP-based weighing approach with actual permission-based classification approaches.

The summary of the precision values for different classifiers has been depicted in Table 8 and Table 9. The tables also depict the comparison of precision values before/after applying the AHP technique for various classifiers. The results depicted in Table 8 and Table 9 demonstrate that the proposed AHP-based approach achieved an improved average accuracy in all the category apps. The improved average accuracy attained for classification algorithms Random Forest, Support Vector Machine, Naïve Bayes, Neural Network, and Decision tree is 77.83%, 78.61%, 77.99%, 75.21%, and 78.21% respectively. It has been observed that Random Forest and SVM-based AHP classifier (SVM_{AHP}), performs better in 8 categories out of 12 categories apps, and Naïve Bayes-based AHP (NB_{AHP}) and Neural Network-based AHP (NB_{AHP}) classifier performs better in 9 categories out of 12 categories apps. Integration of machine learning with AHP has shown the best performance for Health & fitness category apps as the improvement can be clearly observed in the case of three classifiers SVM_{AHP}, NB_{AHP}, and NN_{AHP} as 9.8%, 9.1%, and 8.3%. The proposed hybrid model has also shown good results for the apps belonging to the categories: Navigation and Photo & Video. The results reveal the improvement of 4.4%, 2.2%, and 2.9% in SVM_{AHP}, NB_{AHP}, and NN_{AHP} classifiers for the Navigation category and 1.9%, 3.3% and 4.9% in RFAHP, SVM_{AHP} and NB_{AHP} for Photo & Video category. The average accuracy attained in ensemble techniques, Boosting and Bagging using J48 is 80.01% and 77.22%. The ensemble learning technique, boosting integrated with AHP performed the best as it has shown better accuracy in all the 12 categories of apps. The highest improvement attained is 14% for Health and Fitness Apps. Figure 4 shows the improved precision scores for 12 iOS apps categories using the proposed hybrid approach.

Table 8: Summary of Results Precision Values (in Percentage)

Category	RF	RF_{AHP}	SVM	SVM_{AHP}	NB	NB_{AHP}	NN	NN_{AHP}	DT	DT_{AHP}
Books	76.1	76	75.3	76.3	76.8	75.9	71.8	75.3	79.9	79.9
Education	76.5	75.7	76.5	69	75.7	74.3	78.6	70.7	78.6	77.9
Finance	62.2	61.6	63.9	63.9	67.9	73.3	66	61.3	71	67.2
Games	85.1	86	86.2	85.7	84.1	84.6	84.2	84.5	83.5	82.5
Health & Fitness	83.7	82.5	74.9	84.7	71.7	80.8	77.9	86.2	84.1	78.5
Lifestyle	79.1	80	82.9	76	74.9	74.9	75.8	74.8	82.9	81.9
Music	97.8	98.9	98.9	98.9	95.6	95.6	95.6	95.6	98.6	98.6

Navigation	67.2	67.2	71.6	76	76	78.7	60.4	63.3	67.2	67.2
Photo & Video	75.9	77.8	68	71.3	67.3	72.2	77.4	77.4	74.1	74.1
Social Networking	79.7	81	82	81.9	75.2	74.2	76.5	76.5	78.5	81.3
Sports	66.1	67.9	74.7	74.7	69.9	69.9	58.4	58.4	70.8	74.8
Utilities	77.6	79.4	83.1	84.9	80.8	81.5	78.5	78.5	71.1	77.1
Average	77.25	**77.83**	78.17	**78.61**	76.33	**77.99**	75.09	**75.21**	78.36	**78.42**

Table9: Evaluation of weighing based approach using machine learning classification algorithm

S No	Category	Bo	BoAHP	Bg	BgAHP
1	Books(B)	78.5	79.3	79.2	76
2	Education(E)	70	73	76.4	78.6
3	Finance(F)	73.5	73.6	67.2	66.9
4	Games(G)	84	84	84.7	83.5
5	Health & Fitness (HF)	70	83.9	80	84.3
6	Lifestyle(L)	72.9	72.9	78.9	76.8
7	Music(M)	97.8	98.9	98.9	98.9
8	Navigation(N)	78.7	78.7	55.6	53.8
9	Photo & Video (PV)	77.8	79.6	70.4	70.4
10	Social Networking (SN)	79.6	79.7	82	82
11	Sports(S)	72.2	72.2	75.2	75.2
12	Utilities(U)	85.4	85.4	80.8	80.2
	Average	**78.37**	**80.1**	**77.44**	**77.22**

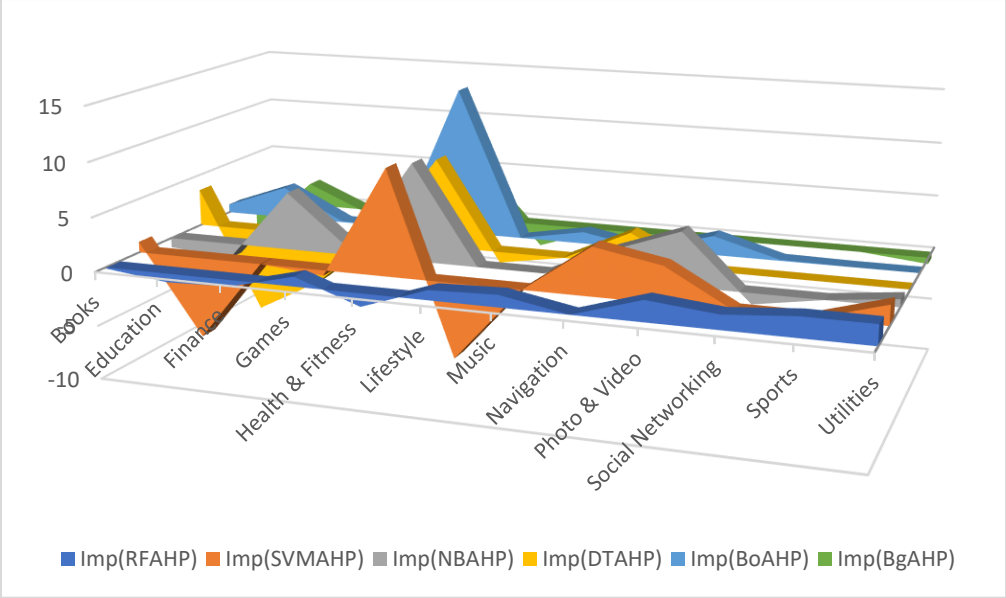


Figure 4: Improved precision using proposed hybrid approach for 12 iOS category apps

Based on the above results from Table 8 and Table 9 it can be concluded that the proposed approach of using a multi-criteria decision-making approach using AHP improves the detection rate of malicious apps. Up to 9.8% in machine learning techniques and 14% in ensemble learning techniques.

5 Conclusion

The paper proposes an AHP-based weighting approach integrated with machine learning and ensemble learning techniques to detect iOS malicious apps. The proposed method initially extracts the app permissions using static analysis for 12 categories to compute a permission matrix comprising the number of apps and presence/absence of features. Then correlation of permissions for every category is computed using Pearson Correlation. Later, the AHP technique is applied to determine the weights of all permissions based on their correlation with respect to the category and in order to compute a weighted permission matrix. The proposed method has been compared with traditional permission-based classification methods. Empirical results depict that the proposed approach

improves the detection rate for all 12 categories of iOS apps. In the future, we plan to conduct a sensitivity analysis to test the robustness of the AHP-derived weights. We will also explore different privacy settings for iOS apps namely track, link, and not-link, and investigate which privacy settings are better predictors for determining malicious or benign apps based on app permissions.

References

[1] Abaker, Ali A., and Fakhreldeen A. Saeed. "A Comparative Analysis of Machine Learning Algorithms to Build a Predictive Model for Detecting Diabetes Complications." Informatica (Slovenia), 45(1), 117–25, 2021. doi:10.31449/inf.v45i1.3111.

[2] Adhikari, Rajindra, et al. "Security and Privacy Issues Related to the Use of Mobile Health Apps." 25th Australasian Conference on Information Systems (ACIS 2014), 2014.

[3] Apple Inc. App Store Downloads on iTunes. https://apps.apple.com/in/genre/ios/id36. Accessed 6 Apr. 2021.

[4] Bhatt, Arpita Jadhav, et al. "iABC: Towards a Hybrid Framework for Analyzing and Classifying Behaviour of iOS Applications Using Static and Dynamic Analysis." Journal of Information Security and Applications, 41, 144–58, 2018. doi:10.1016/j.jisa.2018.07.005.

[5] Chehal, Dimple, et al. "Predicting the Usefulness of E-Commerce Products' Reviews Using Machine Learning Techniques." Informatica (Slovenia),47(2),275–84, 2023. doi:10.31449/inf.v47i2.4155.

[6] Congyi, Deng, and Shi Guangshun. "Method for Detecting Android Malware Based on Ensemble Learning." ACM International Conference Proceeding Series, 8–31, 2020. doi:10.1145/3409073.3409084.

[7] Coronado-De-Alba, Lilian D., et al. "Feature Selection and Ensemble of Classifiers for Android Malware Detection." 2016 8th IEEE Latin-American Conference on Communications, LATINCOM 2016, 128, 2–7, 2016. doi:10.1109/LATINCOM.2016.7811605.

[8] Erickson, Jeremy, et al. AndroidLeaks: Detecting Privacy Leaks In Android Applications., 1–17, 2011. http://www.cs.ucdavis.edu/research/tech-reports/2011/CSE-2011-10.pdf.

[9] Harahsheh, Khawlah, and Chung Hao Chen. "A Survey of Using Machine Learning in IoT Security and the Challenges Faced by Researchers." Informatica (Slovenia), 47(6), 1–54, 2023. doi:10.31449/inf.v47i6.4635.

[10] Huang, Chun Ying, et al. "Performance Evaluation on Permission-Based Detection for Android Malware." Advances in Intelligent Systems and Applications,2, 111–20, 2013. doi:10.1007/978-3-642-35473-1_12.

[11] Idrees, Fauzia, et al. "PIndroid: A Novel Android Malware Detection System Using Ensemble Learning Methods." Computers and Security, 68, 36–46, 2017. doi:10.1016/j.cose.2017.03.011.

[12] Jing, Yiming, et al. "RiskMon : Continuous and Automated Risk Assessment of Mobile Applications." Proceedings of the 4th ACM Conference on Data and Application Security and Privacy - CODASPY '14, 99–110, 2014. doi:10.1145/2557547.2557549.

[13] Kang, Hyunjae, et al. "Detecting and Classifying Android Malware Using Static Analysis along with Creator Information." International Journal of Distributed Sensor Networks, 11(6), Hindawi Publishing Corporation, 479174,2015.

[14] Khan, Jalaluddin, et al. "Survey on Mobile User's Data Privacy Threats and Defense Mechanisms." Procedia Computer Science, 56(1), Elsevier Masson SAS, 376–83,2015. doi:10.1016/j.procs.2015.07.223.

[15] Krupp, Brian. "Enhancing Security And Privacy For Mobile Systems." Doctoral Dissertation, Department of Electrical and Computer Engineering, Cleveland State University, 148, 2015.

[16] Liu, Xing, and Jiqiang Liu. "A Two-Layered Permission-Based Android Malware Detection Scheme." Proceedings - 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2014, 128, IEEE, 142–48, 2014. doi:10.1109/MobileCloud.2014.22.

[17] Mesevage, Tobias Geisler. Machine Learning Classifiers - The Algorithms & How They Work. https://monkeylearn.com/blog/what-is-a-classifier/. Accessed 28 May 2021.

[18] Millman, Rene. Oxford Researchers Expose Personal Data Harvesting in Third-Party Facebook and Google Apps. https://www.itpro.co.uk/privacy/32190/oxford-researchers-expose-personal-data-harvesting-in-third-party-facebook-and-google. Accessed 7 Apr. 2021.

[19] Ptsecurity.com. "Vulnerability and Threats in Mobile Applications." Ptsecurity.Com, 2019.

[20] Saaty, Thomas L. "Decision Making with the Analytic Hierarchy Process." Int. J. Services Sciences, 1(1), 2008.

[21] Sahal, Abdirashid Ahmed, et al. "Mining and Detection of Android Malware Based on Permissions." 3rd International Conference on Computer Science and Engineering (UBMK 2018), IEEE, 264–68, 2018. doi:10.1109/UBMK.2018.8566510.

[22] Sun, Lichao, et al. "Significant Permission Identification for Machine-Learning-Based Android Malware Detection." IEEE Transactions on Industrial Informatics, 14(7), 3216–25, 2018. doi:10.1109/tii.2017.2789219.

[23] Wang, Wei, et al. "Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection." IEEE Transactions on Information Forensics and Security, 9(11),1869–82, 2014. doi:10.1109/TIFS.2014.2353996.

[24] Wikipedia. iOS. https://en.wikipedia.org/wiki/IOS. Accessed 6 Apr. 2021.
[25] WIRED. Thousands of Android and iOS Apps Leak Data From the Cloud. https://www.wired.com/story/ios-android-leaky-apps-cloud/. Accessed 7 Apr. 2021.
[26] Xiong, Ping, et al. "Android Malware Detection with Contrasting Permission Patterns." China Communications, 11(8), China Institute of Communications, 1–14, 2014. doi:10.1109/CC.2014.6911083.
[27] Zhou, Victor. Machine Learning for Beginners: An Introduction to Neural Networks . https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9. Accessed 28 May 2021.