Correlation Analysis of Applications' Features: A Case Study on Google Play

A. Mohammad Ebrahimi, M. Saber Gholami, Saeedeh Momtazi(✉), M. R. Meybodi, and A. Abdollahzadeh Barforoush

Department of Computer Engineering,
Amirkabir University of Tehran, Tehran, Iran
{amirebrahimi, sabergh, momtazi, mmeybodi, ahmadaku}@aut.ac.ir

Abstract. The presence of smartphones and their daily usages have changed several aspects of modern life. Android and IOS devices are widely used these days by the public. Besides, enormous number of mobile applications have been developed for the users. Google launched an online market which is known as Google Play for offering applications to end users as well as managing them in an integrated environment. Applications have many features that developers should clarify while they are uploading apps. These features have potential correlations which studying them could be useful in several tasks such as detecting malicious or miscategorized apps. Motivated by this, the purpose of this paper is to study these correlations through Machine Learning (ML) techniques. We apply various ML classification algorithms to distinguish these relations among key features of applications. Additionally, we perform many examinations to observe the relations between the size of the feature vector and the accuracy of the mentioned algorithms. Furthermore, we compare the algorithms to find the best choices for each part of our experiments. The results of our evaluation are promising. Also, in the majority of cases there are strong correlations between features.

Keywords: Mobile devices · Machine learning · Natural Language Processing · Data analysis · Feature engineering

1 Introduction

In recent years usage of smartphones has been increased. They evolved from simple devices to smart ones that enable users to do various tasks like emailing, navigation, communicating with others, browsing on the internet, taking photos, gaming, etc. These tasks can be done through applications. Furthermore, smartphones need an operating system (O.S) in order to manage all the mentioned tasks. There are several O. S for these devices i.e. IOS, Android, Blackberry and Symbian. Recent advances in the context of applications lead to tight competition among developers to build brand new products. Consequently, many applications have been developed, and they demand huge markets to be organized. For satisfying this requirement, several online markets have been founded. Apple store was the first online market that implied this demand. Afterward, Google introduced "Google Play" for Android users.

M. Bohlouli et al. (Eds.): CiDaS 2019, LNDECT 45, pp. 202–216, 2020.
https://doi.org/10.1007/978-3-030-37309-2_16

At March 2017 Google announced that Android has more than 2 billion monthly active devices. Google also has a lot of other services such as Gmail, YouTube, Google Maps, Chrome and Google Play with over one billion active users in a month [1]. Google Play is an online app store which launched in 2008. At the moment, it has more than 3.6 million available apps in diverse categories [2]. These magnificent number of applications provide a massive amount of worthwhile information, which revolutionize research in many data science areas i.e. security analysis, store ecosystem, release engineering, review analysis, API usage, prediction and feature analysis [3].

Feature Analysis is broken down into many subcategories like "Classification", "Clustering", "Lifecycles", "Recommendation" and "Verification" [3]. From Classification perspective one of the most critical concerns is selecting a set of appropriate features. Papers in this field extract features from multiple sources such as an application's information in Google Play page or binary files of a specific app, with the aim of feeding them into a classifier to distinguish categories and find miscategorized applications [4–8]. Additionally, checking app security, detecting malicious behaviors, and identifying usage of sensitive information (e.g. 'location' and 'contact') have been studied in these scopes [9–12].

There are several different attributes that can be used for feature selection. One way is selecting features from an application's page in Google Play. It has various informative data for each application i.e. Permissions, Description, User reviews, Rate and so on. These features can be used along with other features from other sources for training a classifier to predict a target. In comparison, if selected features contain raw text, converting text into understandable data for a classifier is a critical concern. "Feature Selection" algorithms are a promising solution for this matter. Also, there are too many machine learning classification methods that can be applied for predicting. So picking the suitable algorithms can affect classification performance.

Motivated by these facts, we launched a study to investigate the mentioned topics. Our ultimate objective is studying the main features from google play that can be predicted by other features and finding the correlation between features when predicting the target feature. The contribution of this work is three-fold:

- We gathered approximately 7311 applications from Google Play pages[1].
- We employed eight classification techniques to categorize and predict multiple targets with the purpose of finding the best method for predicting each target.
- We compared the performance of the classifiers for every possible combination of features in order to study the correlation between them.

The remainder of this paper is structured as follows: We start in Sect. 2 with a survey of previous relevant studies. Section 3 explores Google Play structure and features that are available online. Section 4 describes the methods we used and also defines different kind of features. Section 5 discusses feature engineering phase and preprocessing. In Sect. 6 we present experimental setup and evaluation results. We also discuss some practical usage of our findings. Finally, Sect. 7 discusses the results.

[1] Available online at: https://github.com/sabergh/Google_Play_Applications.

2 Related Works

Recent studies in app stores could be divided into seven categories i.e. security, store ecosystem, size and effort prediction, API usage, feature analysis, release engineering and reviews [3]. Two of these fields are related to our work which are "security" and "feature analysis". Researches on security domain try to identify potentially harmful behaviors which are malware detection and inappropriate usage of permissions. On the other hand, papers in feature analysis aim at extracting features out of different sources and use them for classification, recommendation systems, clustering, etc. Papers in these two categories will be discussed below.

Security Varma et al. [12] attempted to detect malicious apps based on the requested permissions. They performed and compared five machine learning algorithms to predict suspicious applications. For training the classifiers, they extracted permissions of applications out of the manifest file and used them as classifiers' features. Gorla et al. [13] proposed CHABADA framework which tries to distinguish trustable apps from dangerous ones by making a contrast between app's description and the usage of API. They performed an LDA topic modeling to find the number of appropriate categories and used k-means to categorize applications. In comparison, Ma et al. [14] performed a semi-supervised learning method to the same problem and achieved higher performance than CHABADA. Shabtai et al. [15] aimed to apply machine learning techniques on an application byte-code for classifying applications into two categories i.e. games and tools. They also suggest that this successful categorization could be used for detecting suspicious behaviors of an application.

Feature Analysis Liu et al. [9] aimed to decide whether an application is suitable for children or not with SVM classifier. They used a variety of features like app category, content rating, title, description and its readability, picture, and texts on it. After that, they generated a list of suitable apps for kids. Olabenjo [8] suggested an appropriate category for new applications. He mined more than 1 million applications and reduced this number to approximately 10000 by removing all applications that have been not developed by top-developers. Then used five features of each application contains app name, content rating, description, whether the application is free or not and whether it has in-app-purchase or not. After that, he performed Bernoulli and Multinomial Naïve Bayes. Berardi et al. [6] focused on presenting an automatic system for suggesting the category of an application which is based on the user's demands. They crawled approximately 6000 applications and extracted the main features of each. Then performed SVM classifier to predict the category of applications. They reached an accuracy of 0.89 which is highly dependent on the imbalance rate of data. In other words, 84.6% of mined applications were in the same category. In 2017 Surian et al. [5] introduced FRAC+: a framework for app categorization which aims to suggest an appropriate category for new applications and also detect the miscategorized ones. The framework consists of two main sections: (i) calculate the optimal number of categories (ii) running the topic model with the calculated number.

However, based on the above discussion, in this paper, we studied the classification of Google Play applications with various learning models and with every possible permutation of features which highly differs from prior studies.

3 Google Play and Dataset

Google Play launched in March 2012 and is an online market that Android developers use to offer their applications. People use applications to satisfy their needs like massaging, photography, playing, emailing, communicating with others on social media, etc. Each application in Google Play has various features. In this section, we discuss these features and clarify their distribution and scaling in our data set.

Every application has many attributes in Google Play. These attributes divided into two main types: (i) attributes that are available on application's page and filled by the developer like name, developer's name, suggested categories, number of downloads, user ratings, description, reviews, last update, size, current version, Android version, content rating and permissions list. (ii) Additionally, there is another type of features that could be extracted from applications byte-code or manifest file. In this paper, we concentrate on the first type.

We crawled 7311 applications from google play. The number is reduced to 6668 by removing non-English apps. The distributions of these applications which are divided into 48 classes are shown in Table 1. To reduce the number of classes, we merged similar categories based on their functionality. In Table 1 there is a number in parentheses in front of each category that illustrates the mapping to the new categories.

Table 1. Distribution of crawled applications categories

Category	Frq	Category	Frq	Category	Frq
Action (1)	252	Education (3)	434	Food & Drink (7)	33
Adventure (1)	203	Books & References (3)	75	Health & Fitness (7)	198
Racing (1)	203	News & Magazines (3)	112	Lifestyle (7)	90
Role playing (1)	172	Auto & Vehicles (4)	17	Medical (7)	7
Simulation (1)	262	House & Home (4)	28	Parenting (7)	46
Trivia (1)	76	Maps & Navigation (4)	59	Music & Audio (8)	178
Arcade (2)	319	Shopping (4)	71	Photography (8)	187
Board (2)	55	Travel & Local (4)	79	Vide players-Editors(8)	65
Card (2)	54	Weather (4)	63	Personalization (9)	236
Casino (2)	51	Comics (5)	32	Dating (10)	5
Casual (2)	463	Entertainment (5)	268	Communicating (10)	134
Educational (2)	400	Libraries & Demo (5)	16	Social (10)	115
Music (2)	80	Sports (5)	241	Art & Design (11)	66
Puzzle (2)	364	Business (6)	64	Events (11)	14
Strategy (2)	169	Finance (6)	76	Productivity (11)	194
Word (2)	90	Beauty (7)	10	Tools (11)	242

Table 2. Distribution of new assigned categories

Category	Frq	Category	Frq	Category	Frq	Category	Frq
1	1168	2	2045	3	621	4	317
5	557	6	140	7	384	8	430
9	236	10	254	11	516		

Overall, we suggested 11 categories that could be found in Table 2 with distributions. As the data was fine-grain, we faced another problem for values and scaling of features such as "rating", "size" and "number of downloads". For example, users can rate an app between 0 to 5 stars so the average rating of an app could be a float number like $0.1, 0.2, \ldots, 4.9, 5.0$. We merged these to 0, +1, +2, +3 and +4. Since the size of applications is in Megabyte, we change them to a coarse-grained scaling by dividing them into categories like 0–10 Mb, 10–20 Mb, etc. Additionally, as 39% of applications do not have the size or the developer mentions: "varies with devices", we put a label "NaN" for them. Furthermore, we used the same solution to tackle with "number of downloads". Therefore all applications that have a close number of installations merged into the same category.

4 Classification Algorithms

In this section, we explore several machine learning algorithms which are used in our experiments. Machine learning is a field of research in artificial intelligence that uses statistical techniques to allow computer systems to learn from data and getting them to act without being explicitly programmed [17].

Generally, machine learning algorithms can be divided into categories based on their purpose or type of training data. From the training data perspective, there are three approaches: supervised learning, unsupervised learning, and semi-supervised learning. In supervised learning, each of the training examples in training dataset must be labeled, then the algorithm analyzes the training data and produces a model which can be used to label unseen examples [18]. Unsupervised algorithms learn from training data that has not been labeled, so the learning process is based on the similarity between the training examples [19]. Semi-supervised learning falls between supervised learning and unsupervised learning because it uses a mixture of labeled and unlabeled data. In comparison with supervised learning, this approach helps to reduce the cost and effort of labeling data. Also, the small proportion of labeled data used in this approach improves the classification accuracy compared to unsupervised learning [20].

In this paper, we used supervised learning for two reasons: first, our experiments are based on classifying different targets. Additionally, the selected data are properly labeled by developers, therefore we do not need to put any effort into annotation. Following the above discussion, the supervised classification algorithms, which are used in our experiments, will be introduced below.

Naïve Bayes (NBs) algorithms are a set of common supervised learning algorithms based on applying Bayes' theorem. One of the most important principles of NBs is

"naïve" assumption which considers every feature independent of others [21]. Despite being simple, they work quite well in real-world scenarios. They demand much less labeled data comparing to other learning algorithms. Furthermore, concerning runtime, they can be extremely fast compared to more sophisticated methods. In this paper, we experimented different versions of Naïve Bayes algorithm i.e. Bernoulli Naïve Bayes (BNB), Gaussian Naïve Bayes (GNB) and Multinomial Naïve Bayes (MNB) [22].

Support Vector Machine (SVM) algorithms are a type of supervised learning algorithms that can be employed for both classification and regression purposes. SVMs have been applied successfully in a variety of classification problems such as text classification, image classification and recognizing hand-written characters. SVMs are famous for their classification accuracy and the ability to deal with high dimensional data [23]. One of the most important aspects of SVMs is selecting a suitable kernel function. A kernel function takes data as input and transforms it into the required form. There are different kernel functions like linear, polynomial, Radial Basis Function (RBF) and sigmoid. In our work, we selected RBF because of low time complexity. Also, RBF works quite well in practice, and it is relatively easy to tune as opposed to other kernels.

Decision Tree (DT) algorithm belongs to the family of supervised learning algorithms. Similar to SVM algorithms DT can be applied for both regression and classification problems. It is based on building a model that can predict class or value of the target variable by learning decision rules inferred from training data. The main advantages of DT cause to be selected in our work are (1) its ability to discover nonlinear relationships and interactions (2) interpretability (3) its robustness of dealing with outliers and missing data [24].

Random Forest (RF) algorithm is an ensemble learning method for classification and regression tasks. Generally, this classifier builds several decision trees on randomly selected sub-samples of training data. It then merges the results from different decision trees to make a decision about the final class of the test example. The process of voting helps to reduce the risk of overfitting. As a result, it improves classification accuracy. According to the above discussion we selected RF in our experiments [25].

AdaBoost (AB) classifier is another ensemble classifier that aims to build a robust classifier from a number of weak classifiers. Its process starts by building a model on the original dataset and then the subsequent classifiers attempts to correct the previous models [26].

Multilayer Perceptron (MLP) is a kind of neural network algorithms, and it is based on a network of perceptrons which organized in a feedforward topology. Basically, it consists of at least three layers: an input layer, a hidden layer, and an output layer. MLP belongs to the family of nonlinear classifier because it uses nonlinear activation functions in all layers except for the input layer [27]. We selected MLP because its nonlinear nature makes it suitable to learn and model complex relationships which are too complicated to be noticed by human or other learning algorithms.

5 Feature Engineering

The success of supervised machine learning algorithms strongly depends on how data is represented to them in terms of features. Feature engineering is the process of transforming raw data into features that make machine learning algorithms work better. In fact, providing an appropriate set of features is a fundamental issue of every learning algorithm. In this section, we will explore our features in general and our strategy for selecting suitable features [28].

Overall, we considered eight factors to extract feature out of them: (1) rate (2) number of votes (3) size (4) number of downloads (5) detailed permissions (6) general permissions (7) description (8) category. To identify the best set of features that results in the maximum classification accuracy and analyzing the role of each feature in predicting others, we accounted for every possible combination of features in our experiments. In order to calculate combinations for predicting a target variable *t*, all the features except *t* have been passed to a function which outputs every possible subset of features that could be constructed from the input features. That means for predicting target variable *t* if we pass N features to the function it will return all possible 2^N subsets of features. Furthermore, to use generated subsets in our experiments they must be converted into the vector space model. In the following, we will explain features in detail in terms of definition, idea and the process of converting to vectors.

Rate (R) users who install an application can score it from 1 to 5. The average rate is calculated by summing all the scores and then dividing it by the total number of participants. In order to avoid exceeding the number of possible values for this factor, we discriminated it by the procedure in Sect. 3.

Number of voters (RN) to distinguish between applications with a high number of voters and those with a low number of voters, we defined the multiplication of rate and number of voters as a single feature. This will help to reduce the effects of the rate for an application when few users rated it.

Size (S) we selected this factor as a feature because it might be related to the application category. For example, an application with high size might be a game rather than others. In order to avoid enormous number of possible values for this factor, we classified it by the process explained in Sect. 3.

Number of downloads (I) downloads count is another factor that could be correlated to others features. For example apps in popular categories could have a higher chance to be downloaded. So we used it as a feature in our final feature vector.

Description (D) Google Play allows developers to write about their apps. Regularly, this description talks about the features that users will get from the app. In the context of Natural Language Processing (NLP), the description might contain words which could represent the underlying problem better for our learning algorithms. In this matter, we selected description as a factor to extract feature out of it. To use the description as a feature, we converted preprocessed texts into vectors using x-square algorithm and TF-IDF as a weighting schema. Among those features, we selected the first top 300 features to build the vectors. The more features are included in the training phase, the more time consumes for training though. We also repeated our experiments

with the first top 100, 200 and 300 features in order to analyze the effects of description vector size on the final results of classifiers.

Permissions (P) every application gets various permissions on the host device. This feature might help effectively in the prediction of category or number of downloads. Thus, we decided to include this feature to the developed vector. Besides, every application has two kinds of permissions: General (GP) and Detailed (DP). For instance, an application could get four GPs: Location, Photos/Media/Files, Storage and Other. Every GP might have one or more DPs. For example, "Storage" might contain two detailed ones (1) Read the content of your USB storage, and (2) Modify or delete the contents of USB storage. We included both kinds of permissions in the vector. All crawled applications have 16 unique GPs and 199 unique DPs.

Category (C) each application's category is included in the vector. As it is shown in Table 2, we end up with 11 categories for all applications. So this number is involved in our vector as the last feature.

1	1	1	1	*	16	199	1
R	RN	S	I	D	GP	DP	C

Fig. 1. The features of the final vector and their size. **D stands for Description and the size varies with 100, 200 and 300.*

The final vector is shown in Fig. 1. The size of all features except D is 220. Thus, the total size will vary in the range of 320, 420 and 520.

6 Evaluation

6.1 Experimental Setup

To perform our experiments, we used Python which is a widely used open source programming language. As python has many different libraries to deal with machine learning and NLP problems, it could be used effectively for such processes. Nltk[2] and Sklearn[3] are two libraries that have been used in our work.

6.2 Results and Discussions

In this section, we report the experiment performance with respect to the feature sets. Our objective is to detect the correlation between features as well as detection of the best algorithm for predicting. For this purpose, we first tried to predict category of each application with the description of that app. Table 3 shows the result of this experiment. Note that P stands for precision, R for Recall and F for F-measure.

[2] Nltk.org.

[3] Scikit-learn.org.

Table 3. Prediction of category

	Algorithm	Features	D vector length	P	R	F
Predicting category	MLP	D	100	0.49	0.50	0.50
	SVM	D	100	0.51	0.46	0.48
	BNB	D	100	0.46	0.46	0.46
	MNB	D	100	0.46	0.46	0.46
	RF	D	100	0.44	0.45	0.44
	AB	D	100	0.43	0.43	0.43
	DT	D	100	0.39	0.39	0.39
	GNB	D	100	0.48	0.22	0.30

Based on the results, we can observe that GNB is not suitable for predicting this feature while MLP performs the best. In the next step, one variable has been changed which is the length of the description vector. This attribute changed to 200 and 300 in order to identify the influence of this element on the final results. The results are represented in Table 4 for the best classifiers from Table 3. The results show that increasing the size of the description vector leads to significant improvement in F-measure.

Table 4. Prediction of category with various sizes of description vector

	Algorithm	Features	D vector length	P	R	F
Predicting category	MLP	D	100	0.49	0.50	0.49
		D	200	0.59	0.59	0.59
		D	300	0.62	0.62	0.62
	SVM	D	100	0.51	0.46	0.48
		D	200	0.59	0.55	0.57
		D	300	0.60	0.56	0.58
	BNB	D	100	0.46	0.46	0.46
		D	200	0.55	0.54	0.54
		D	300	0.58	0.56	0.57
	MNB	D	100	0.46	0.46	0.46
		D	200	0.57	0.55	0.56
		D	300	0.60	0.58	0.59

The next step of our experiment is to predict category with various features. These features have been explained in Sect. 5. The results shown in Table 5 illustrate that the MLP algorithm performs better than any other algorithms in category predicting. Additionally, a simple comparison between two latter tables demonstrates that involving other features in the learning process leads to even better results. For example, using detailed permissions along with description improves the results in terms of f-measure by approximately 3%. This is probably because of some categories

that need special permissions, and this will help the classifier to distinguish between them. The results in Table 5 are selected among more than 700 experiments including all algorithms and features and indicate that other algorithms could not outperform MLP even with more features.

Table 5. Top ten results of predicting category with various algorithms and features

	Algorithm	Features	D vector length	P	R	F
Predicting category	MLP	D, P	300	0.64	0.64	0.64
	MLP	D, S, I, P	300	0.64	0.64	0.64
	MLP	D, R, S, P	300	0.64	0.64	0.64
	MLP	D, R, I, P	300	0.64	0.63	0.63
	MLP	D, R, P	300	0.64	0.63	0.63
	MLP	D, R, S, I, P	300	0.64	0.63	0.63
	MLP	D, I, P	300	0.63	0.63	0.63
	MLP	D, S, P	300	0.63	0.63	0.63
	MLP	D, R	300	0.62	0.62	0.62
	MLP	D, S	300	0.62	0.62	0.62

More precisely, after having done all these experiments for predicting categories, we expanded our study by predicting other features, namely Rate, Size and Install count. Table 6 reveals these results. Overall, RF and DT performed better than other algorithms in terms of predicting Size, Rate and Install count. For predicting the size of apps, the table shows that presence or absence of description in feature vector cannot affect the f-measure. This conclusion is based on the fifth row of the table which is similar to the other rows in terms of f-measure but has no description. Moreover, increasing the description vector length does not affect results. Comparing these results to other results shows that feature P can solely affect size prediction by 35%.

Additionally, in Rate and Install count prediction some experiments have been found without description which demonstrate the influence of other features. What's more, RN plays a significant role in predicting rate which is probably because of a strong correlation between these two factors. Besides, an interesting fact is that the reduction of feature vector size does not necessarily lead to f-measure plummet. For example in install count prediction, a tiny vector (R, RN, S, and C) with the size of 4 with RF, performed as good as huge vectors with DT algorithm.

We extended our experiments even more in order to analyze to what extent different general permissions have correlation with each other. As it is mentioned in Sect. 5, we found 16 different general permissions in our dataset. To perform correlation analysis, we first analyze the distribution of each general permission. Then we removed permissions which had an extremely unbalanced proportion. To achieve this, we ignored the permissions with less than 1000 sample of each label. Finally, we ended up with eight general permissions. Table 7 shows the distribution of these eight permissions which are considered to be involved in our experiments.

Table 6. Top 10 results of predicting size, rate and install count with various algorithms and features

	Algorithm	Features	D vector length	P	R	F
Predicting size	RF	D, R, I, P, C	200	0.39	0.44	0.41
	RF	D, R RN, P, C	300	0.39	0.44	0.41
	RF	D, I, P, C	100	0.39	0.43	0.40
	RF	D, R, P, C	100	0.38	0.44	0.40
	RF	R, RN, I, P, C	–	0.38	0.43	0.40
	RF	D, P	100	0.38	0.43	0.40
	RF	D, I, P	100	0.38	0.43	0.40
	RF	D, P, C	100	0.38	0.43	0.40
	RF	D, R, RN, P	100	0.38	0.43	0.40
	RF	D, RN, P, C, S	100	0.38	0.43	0.40
Predicting rate	RF	RN, S, I, P, C	–	0.56	0.58	0.57
	RF	D, RN, S, I	200	0.55	0.58	0.56
	RF	D, RN, I, P	100	0.55	0.57	0.56
	RF	D, RN, S, I, P	100	0.55	0.57	0.56
	RF	D, RN, I, P, C	100	0.55	0.57	0.56
	RF	RN, S, I, P	–	0.55	0.56	0.55
	RF	D, RN, I, C	100	0.54	0.56	0.55
	RF	D, RN, S, I, C	100	0.54	0.56	0.55
	DT	D, RN, I, P	200	0.54	0.54	0.54
	DT	D, RN, S, I, P, C	100	0.54	0.54	0.54
Predicting install count	RF	R, RN, S, C	–	0.65	0.64	0.64
	DT	D, R, RN, P, C	100	0.64	0.64	0.64
	RF	D, R, RN, C	100	0.64	0.63	0.63
	DT	D, R, RN, P	200	0.64	0.63	0.63
	DT	D, R, RN, P, C	300	0.64	0.63	0.63
	DT	D, R, RN, P, C	200	0.64	0.63	0.63
	DT	D, R, RN, S, P, C	200	0.63	0.63	0.63
	DT	D, R, RN, P	300	0.63	0.63	0.63
	DT	D, R, RN, S, P, C	100	0.63	0.63	0.63
	DT	D, R, RN, S, P	200	0.63	0.63	0.63

In order to determine the correlations between selected general permissions, we apply the same approach as we discussed earlier in this section. However, in the feature selection part we considered all the general permissions except the target one in the feature vector; e.g., if GP1 is the target for prediction then other seven general permissions (i.e. GP2, GP3, GP4, GP5, GP6, GP7, and GP8) will make the feature vector. For classification, we used the same algorithms as before. Table 8 shows our results in predicting different general permissions with eight classification algorithms in terms of precision, recall, and F-measure.

Table 8 reveals several interesting points. First of all, despite the unbalanced distribution of classes, the classification results are substantially promising. In contrast to the Table 5, where there was a significant difference between the results of one special classification algorithm compare to the other algorithms, the results of Table 8 demonstrate that several classification algorithms could achieve higher results in comparison with the rest in terms of P, R, and F.

For predicting GP1, all the classification algorithms have the same performance (99%) based on all evaluation metrics. Performance of the classification algorithms for GP2 is relatively similar except for AdaBoost Algorithm, where there is a minor

Table 7. Different permissions and their distribution

Id	Target permission	1	0
GP1	Device & App History	3426 (53%)	3242 (47%)
GP2	Contact	4696 (70%)	1972 (30%)
GP3	Photos | Media | Files	1024 (15%)	5163 (85%)
GP4	Calendar	1631 (24%)	5037 (76%)
GP5	Cellular data settings	1857 (28%)	4811 (72%)
GP6	Wearable sensors | Activity data	1529 (23%)	5139 (77%)
GP7	Storage	1918 (29%)	4750 (71%)
GP8	Microphone	4695 (70%)	1973 (30%)

increase in precision (1%) as opposed to the other algorithms. The best results in GP3 classification are same as the best results in GP2. However, there are more than one algorithms which have reached the best performance. Results in predicting GP4 show that four algorithms have performed better in comparison to others.

F-measure scores in predicting GP5 are quite similar (99%) in most algorithms. However, there are three algorithms that have done slightly better with 1% percent increase in P and R. Performance of the algorithms in classifying apps when GP6 is the target are the highest among all the classifications results in Table 8 with 100% for all evaluation metrics. That means, there is a strong correlation between GP6 and other selected general permissions. Best results in predicting GP7 are completely equal with the best in GP1 and GP5. Comparing to other results in Table 8, the performance of predicting GP8 has significantly decreased by almost 15% but still promising.

Finally, based on the classifications results and the above discussions we believe there are strong correlations between all the selected general permissions which leads to high-performance results in predicting each general permissions with the rest.

6.3 Practical Usage

The practical usage of our analysis would be in several domains. In case of predicting category, there are apps which have been miscategorized by developers in the Google play store [5]. Therefore, concerning the supervised learning algorithms as solutions for the problem, the type of features which are selected in training phase would be crucial.

Table 8. Prediction results for eight different GPs.

		P	R	F		P	R	F		P	R	F		P	R	F
BNB	GP1	**0.99**	**0.99**	**0.99**	GP2	0.97	0.97	0.97	GP3	**0.98**	**0.97**	**0.97**	GP4	0.95	0.91	0.93
GNB		**0.99**	**0.99**	**0.99**		0.97	0.97	0.97		0.97	0.97	0.97		0.95	0.94	0.95
MNB		**0.99**	**0.99**	**0.99**		0.97	0.97	0.97		0.87	0.87	0.85		0.94	0.97	0.96
DT		**0.99**	**0.99**	**0.99**		0.97	0.97	0.97		**0.98**	**0.97**	**0.97**		**0.95**	**0.97**	**0.96**
RF		**0.99**	**0.99**	**0.99**		0.97	0.97	0.97		**0.98**	**0.97**	**0.97**		**0.95**	**0.97**	**0.96**
MLP		**0.99**	**0.99**	**0.99**		0.97	0.97	0.97		**0.98**	**0.97**	**0.97**		**0.95**	**0.97**	**0.96**
AB		**0.99**	**0.99**	**0.99**		**0.98**	**0.97**	**0.97**		**0.98**	**0.97**	**0.97**		**0.95**	**0.97**	**0.96**
SVM		**0.99**	**0.99**	**0.99**		0.97	0.97	0.97		**0.98**	**0.97**	**0.97**		0.94	0.97	0.96
BNB	GP5	**0.99**	**0.99**	**0.99**	GP6	1.0	0.99	0.99	GP7	**0.99**	**0.99**	**0.99**	GP8	0.84	0.82	0.83
GNB		0.99	0.93	0.95		1.0	0.86	0.92		**0.99**	**0.99**	**0.99**		0.83	0.87	0.84
MNB		0.98	0.99	0.99		**1.0**	**1.0**	**1.0**		0.96	0.96	0.96		0.79	0.89	0.84
DT		**0.99**	**0.99**	**0.99**		**1.0**	**1.0**	**1.0**		**0.99**	**0.99**	**0.99**		0.83	0.89	0.84
RF		**0.99**	**0.99**	**0.99**		**1.0**	**1.0**	**1.0**		**0.99**	**0.99**	**0.99**		**0.86**	**0.89**	**0.84**
MLP		0.98	0.99	0.99		**1.0**	**1.0**	**1.0**		**0.99**	**0.99**	**0.99**		0.84	0.89	0.84
AB		0.98	0.99	0.99		**1.0**	**1.0**	**1.0**		**0.99**	**0.99**	**0.99**		0.83	0.89	0.84
SVM		0.98	0.99	0.99		**1.0**	**1.0**	**1.0**		**0.99**	**0.99**	**0.99**		0.79	0.89	0.84

For instance, in our analysis we figured out that description and permission are more important for predicting app's category. Additionally, another solution is to use clustering techniques [16] or the concepts of graph theory (community detection). In both cases, description and permission could be applied as a feature vector since they are highly correlated with app's category based on our analysis.

Predicting size is a relatively small and new research area [3]. However, our analysis has shown us there are not strong correlations among other features and the app's size. Therefore, we do not claim that this analysis can certainly be helpful in this task.

In contrast, as far as the practical usage of finding correlations among general permissions are concerned, there are several methods that can be used to identify hazardous applications. To this aim, a clustering algorithm, like k-means, can find the applications with *suspicious* permissions; which are the applications that get a special permission which is not common among their similar apps. Furthermore, based on our results in predicting each general permission, we expect to obtain acceptable accuracy in identifying dangerous applications with unusual behavior.

7 Conclusion

This paper presents an immense study on classifying Google Play applications with various algorithms and multiple features. The first part of our experimental results on more than 7000 applications demonstrates that there is a significant difference between algorithms for predicting each feature. More precisely, the MLP algorithm outperforms the rest in term of category prediction, while Decision Tree and Random Forest are the best algorithms to predict other features: i.e. rate, size, and the number of installations. Also, our results show that increasing the feature vector size does not necessarily leads to better accuracy and it is possible to achieve the same f-measure with small vectors.

The second part of our experiments reveals that there are strong correlations among general permissions. Moreover, the performance of different algorithms in the second part were fairly same and considerably high. Future works would include correlation analysis between other general permissions that have not been covered here on a larger and more balanced dataset. Finally, the findings of the second part of our experiments could be used to propose more sophisticated methods in predicting apps that get suspicious permissions.

References

1. Google announces over 2 billion monthly active devices on Android. https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users. Accessed 12 Aug 2018
2. Google Play Store: number of apps 2018—Statistic. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. Accessed 12 Aug 2018
3. Martin, W., Sarro, F., Jia, Y., Zhang, Y., Harman, M.: A survey of app store analysis for software engineering. IEEE Trans. Softw. Eng. **43**(9), 817–847 (2017)
4. Radosavljevic, V., et al.: Smartphone app categorization for interest targeting in advertising marketplace. In: Proceedings of the 25th International Conference Companion on World Wide Web - WWW 2016 Companion, pp. 93–94 (2016)
5. Surian, D., Seneviratne, S., Seneviratne, A., Chawla, S.: App miscategorization detection: a case study on Google Play. IEEE Trans. Knowl. Data Eng. **29**(8), 1591–1604 (2017)
6. Berardi, G., Esuli, A., Fagni, T., Sebastiani, F.: Multi-store metadata-based supervised mobile app classification. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC 2015, pp. 585–588 (2015)
7. Cunha, A., Cunha, E., Peres, E., Trigueiros, P.: Helping older people: is there an app for that? Procedia Comput. Sci. **100**, 118–127 (2016)
8. Olabenjo, B.: Applying Naive Bayes Classification to Google Play Apps Categorization, August 2016
9. Liu, M., Wang, H., Guo, Y., Hong, J.: Identifying and analyzing the privacy of apps for kids. In: Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications - HotMobile 2016, pp. 105–110 (2016)
10. Wang, H., Li, Y., Guo, Y., Agarwal, Y., Hong, J.I.: Understanding the purpose of permission use in mobile apps. ACM Trans. Inf. Syst. **35**(4), 1–40 (2017)
11. Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., Wu, K.-P.: DroidMat: android malware detection through manifest and API calls tracing. In: 2012 Seventh Asia Joint Conference on Information Security, pp. 62–69 (2012)
12. Varma, P.R.K., Raj, K.P., Raju, K.V.S.: Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms. In: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 294–299 (2017)
13. Gorla, A., Tavecchia, I., Gross, F., Zeller, A.: Checking app behavior against app descriptions. In: Proceedings of the 36th International Conference on Software Engineering - ICSE 2014, pp. 1025–1035 (2014)
14. Ma, S., Wang, S., Lo, D., Deng, R.H., Sun, C.: Active semi-supervised approach for checking app behavior against its description. In: 2015 IEEE 39th Annual Computer Software and Applications Conference, pp. 179–184 (2015)

15. Shabtai, A., Fledel, Y., Elovici, Y.: Automated static code analysis for classifying android applications using machine learning. In: 2010 International Conference on Computational Intelligence and Security, pp. 329–333 (2010)
16. Al-Subaihin, A.A., et al.: Clustering mobile apps based on mined textual features. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 2016, pp. 1–10 (2016)
17. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
18. Kotsiantis, S.: Supervised machine learning: a review of classification techniques. In: Emerging Artificial Intelligence Applications in Computer Engineering, pp. 3–24 (2007)
19. Kotsiantis, S., Panayiotis, P.: Recent advances in clustering: a brief survey. WSEAS Trans. Inf. Sci. Appl. **1**(1), 73–81 (2004)
20. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
21. Lewis, D.D.: Naive (Bayes) at forty: the independence assumption in information retrieval, pp. 4–15. Springer, Heidelberg (1998)
22. McCallum, A., Nigam, K.: A comparison of event models for Naive Bayes text classification. In: AAAI 1998 Workshop on Learning for Text Categorization, vol. 752, no. 1, pp. 41–48 (1998)
23. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning, pp. 137–142. Springer (1998)
24. Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers—a survey. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **35**(4), 476–487 (2005)
25. Svetnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R.P., Feuston, B.P.: Random forest: a classification and regression tool for compound classification and QSAR modeling. J. Chem. Inf. Comput. Sci. **43**(6), 1947–1958 (2003)
26. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. J.-Jpn. Soc. Artif. Intell. **14**(771–780), 1612 (1999)
27. Gardner, M.W., Dorling, S.R.: Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos. Environ. **32**(14–15), 2627–2636 (1998)
28. Dong, G., Liu, H.: Feature Engineering for Machine Learning and Data Analytics. CRC Press, Boca Raton (2018)