Predicting the Rating of an App Beyond its Functionalities

Lega, Mathieu; Burnay, Corentin; Faulkner, Stephane

Published in:
Proceedings of the ICDSST 2022 International Conference on Decision Support System Technology

Publication date:
2022

Document Version
Peer reviewed version

Predicting the Rating of an App Beyond its Functionalities: Introducing the App Publication Strategy

Mathieu Lega✉[0000−0003−1682−4920], Corentin Burnay[0000−0002−0325−1732], and Stéphane Faulkner

University of Namur, Namur Digital Institute (NADI / PReCISE), Namur, Belgium
mathieu.lega@unamur.be

Abstract. Mobile applications (or apps) are present on every portable device and have become the center of tremendous attention from developers and software vendors. Some apps meet significant success with high profitability, but most of them tend to remain anonymous, with weak returns on investment. The risk incurred when launching a new app is therefore significant. In this article, we introduce the concept of Publication Strategy, resulting from the numerous decisions made by an app designer on all the variables which are publicly visible on the stores (screenshots, description, title, etc.). This paper studies the extent to which the success of an app may be predicted using such Publication Strategy. To do this, we use metadata about more than 40,000 apps from both the Google Play Store and the Apple App Store and adopt a machine learning research strategy by training and testing a number of classification models. We observe that in about 50% of the cases, it is possible to predict the rating of an app based solely on its Publication Strategy. The results are very similar between the 2 stores. These results bring us to the definition of a number of research avenues to further explore the notion of App Publication Strategy which can be used to support apps designers in their decisions.

Keywords: mobile applications, apps, Publication Strategy, empirical study, machine learning, Google Play Store, Apple App Store, decision support

1 Introduction

Nowadays, more than two thirds of the global population owns a smartphone [12]. On average, a smartphone is used 3 hours a day and contains 100 applications [12]. In 2018, more than $100 billions were spent by users on mobile applications (also called mobile apps or apps) via in-app purchases, paid downloads and subscription fees [7] and 194 billions of mobile apps were downloaded [20]. Some predictions state that the global mobile apps revenue could get near to $1000 billions by 2023 [20].

It is clear from these numbers that the generated revenue is growing faster each year and that this evolution is expected to continue in the next years. This in turn emphasizes the fact that mobile applications now represent a huge market and therefore a significant source of opportunities for businesses.

However, designing and selling an app is not trivial; some apps are downloaded/purchased much more frequently than others, and pushing an app on the market may therefore not always represent a good move for a company as it requires a lot of resources. In other words, developing a new app does not always come with financial success and companies should be cautious about it.

We also know that the rating of a mobile application has an impact on the ranking of the app in search results for the App Store [2]. Knowing that more or less 50% of the customers use these stores to discover apps, rating has a direct impact on the discovery of an app [3, 5]. Moreover, 80% of the customers check the rating of an app before downloading it [5]. If an app has a rating of 2 stars out of 5, only 15% of the customers state that they will consider downloading it [3, 5]. This percentage increases to 96% if the rating is 4 stars [3, 5]. We can thus say that the rating also has an impact on the conversion rate to download and thus the success of an app.

All these reasons brought us to look at the extent to which the rating (used as proxy for the success), and more precisely the number of complete stars, of a mobile application can be predicted using only what we call the "Publication Strategy" of the companies developing and commercializing the app (latter called "the companies"). This Publication Strategy includes the variables controlled by the companies and displayed when a customer arrives on the page of the app on an app store. Our objective in doing so is to discover if the concept of Publication Strategy has an impact on the success of an app. If the Publication Strategy truly plays a role in the success of an app, better understanding this concept would allow companies to take better decisions regarding their Publication Strategy (which is easier to change than the functionalities of the app) and thus to minimize the number of low rated (and thus riskier) apps launched on the market for a given quality/functionality level. We also try to discover the most important variables/features (the 2 words will be used interchangeably in this work) of a Publication Strategy. Our research questions can be stated as follows:

- To which extent is it possible to predict the rating of a mobile application using only its Publication Strategy?
- Which are the most important variables in the Publication Strategy of a mobile application?

For this purpose, we first create two databases containing metadata about applications, one for the Google Play Store and one for the Apple App Store. Then, feature engineering is used in order to extract the most important variables from the raw data. Finally, machine learning is applied to train prediction models. We use the following classification algorithms: decision tree, random forest, k-nearest neighbors, support vector machine and neural networks. The results of these algorithms are compared and discussed.

The remainder of this paper is structured as follows. In section 2, we present some related works in the field of mobile applications. In section 3, we explain with more details our methodology for the creation of the databases, the use of feature engineering and the training of the models. In section 4, we present our results. In section 5, we discuss our results, the limitations of our work and some research avenues. Finally, we conclude our work in section 6.

2 Related works

This section presents some related works about apps success factors and the prediction of the rating of mobile apps. We also detail our positioning compared to the presented articles.

2.1 Models of App Success Factors

The reasons behind the success of a mobile application has been studied in several ways in the literature. Lee and Raghu [11] decided to look at how the success in an app market is impacted by both app-related and seller-related characteristics. For this purpose, they tracked the presence of apps in the top 300 grossing charts of the Apple App Store and analyzed the factors allowing an app to stay in these charts or not [11]. For this purpose, they used a generalized hierarchical modeling approach, a hazard model and a count regression model [11]. They concluded that, at a seller-level, it is very important to diversify across categories to achieve sales performance [11]. At an app-level, the following factors may impact positively the way an app performs in terms of sales: being free, higher initial popularity, the fact of investing in less popular categories, continuously updating the features of the app and an higher number of user feedbacks [11].

Picoto, Duarte and Pinto [16] also used the tops grossing for their study. The aim of their study was to find the factors that could have an impact on the ranking and the success of an app [16]. For this purpose, they took 500 top grossing apps from the Apple App Store from which they extract the top 50 and bottom 50 for analysis [16]. Once they had identified potential antecedents for an app's ranking, they used a multivariate logistic regression and a Fuzzy Set Qualitative Comparative Analysis (fsQCA) to identify the factors that could determine the success of an app [16]. They found that the following factors make it more likely for an app to be in the top 50: category popularity, number of languages supported, package size, and app release date [16]. A surprising result is that the higher the user rating, the lower the probability for this app to be in the top 50 [16]. Finally, they found that the attributes, functionalities, and longevity of an app have a bigger impact on the success of an app than the user rating [16].

Yang [21] tried to predict the use of mobile apps, the attitudes of customers towards these apps and the intent to use with the Theory of Planned Behavior, the Technology Acceptance Model, and the Uses and Gratification Theory. The

author tested the proposed model with a web survey answered by American college students [21]. The results state that the following factors may be used to predict the attitude of consumers related to apps: perceived enjoyment, usefulness, subjective norm, and ease of use [21]. If the aim is to predict the use of applications, the significant factors are: perceived usefulness, mobile internet use, mobile apps intent, personal income, and gender [21].

Lu, Liu and Wei [13] decided to focus only on two factors: enjoyment and mobility. In their study, they tried to understand the link between the perception of these factors and the intention to continue using an app [13]. They used a second-stage continuance model and data of 584 smartphone users collected with a survey [13]. As a result, they discovered that "the salience of disconfirmation and beliefs in enjoyment and mobility serve as the primary driver of the changes in satisfaction and attitude toward continuance intentions" (Lu, Liu & Wei, 2016, p.1). Another result is that more than 60% of the variance related to the attitude after usage can be attributed to: perceived enjoyment, mobility and satisfaction [13].

2.2 Prediction of ratings

The prediction of the user rating of an application is a well-known problematic. Monett and Stolte [15] have tackled this problem in their work. They used a corpus of 1,760 annotated reviews about 130 mobile apps available on the Google Play Store [15]. Their goal was to predict the rating based on the polarity of subjective phrases found in the reviews [15]. For this purpose, several computational models have been used and evaluated [15]. They concluded that rating could be well predicted even using only phrase-level sentiment polarity [15].

Meng, Zheng, Tao and Liu [14] implemented a weight base matrix factorization (WMF) capturing user-specific interests in order to predict an app rating for a specific user. The used dataset containing logs of user's downloaded and uninstalled apps involved 5057 users and 4496 apps [14]. Their model got convincing results, performing better than some other prediction models [14].

A third approach can be found in [4]. In order to predict the rating of applications, Daimi and Hazzazi decided to use the following algorithms: Linear Regression, Neural Networks, Support Vector Machines, Random Forest, M5 Rules, REP Tree and Random Tree [4]. They used an Apple Store dataset composed of 7197 apps and the following attributes: user rating count for all version, user rating count for current version, average user rating for all version (the attribute to predict), average user rating for current version, number of supporting devices, number of screen shots showed for display and number of supported languages [4]. They found that Random Forest produced the best results when predicting the rating of an application from the Apple Store dataset [4].

Finally, Sarro, Harman, Jia and Zhang [18] focused on predictions achieved by Case Based Reasoning (CBR) taking only the technical features of the apps into account. They used a dataset dating from 2011 containing 9588 apps and 1256 extracted features from the BlackBerry App World store and 1949 apps and 620 extracted features from the Samsung Android App store [18]. As a result,

they discovered that, in 89% of the cases, the rating of an app could be perfectly predicted [18]. They also discovered that only 11-12 applications were sufficient to achieve the best prediction when using a case-based prediction system [18]. They thus concluded that it is possible to accurately predict the rating of an app taking only its features into account [18].

2.3 Positioning of this work

Our work differentiates from all the studies about apps success factors presented above because we do not use feedbacks from users to test a model, neither focus on top charts. Instead, we try to maximize our precision when predicting the rating (that acts as proxy for the success) given by the customers of an app. More precisely, we try to predict the number of stars an app will receive using classification algorithms. We thus use the well-known star rating used by several stores.

What distinguishes this work from the other works about the prediction of the rating of an app is the fact that we only use the Publication Strategy as predictor. Indeed, we investigate the impact of the Publication Strategy on the success, observed by the number of stars an app receives on the stores. We thus only focus on variables that may be leveraged by the companies and that are displayed on the stores because this is by definition what we consider to be the Publication Strategy.

Moreover we differentiate apps from the Apple App Store and apps coming from the Android Play Store. For this purpose, we gathered metadata about more than 80 000 applications for the former and 90 000 applications for the latter. Our final output is thus trained models of machine learning able to predict a rating (the number of stars) for an application.

3 Methodology

3.1 Creation of the databases

To answer our research questions, the first step is to collect data about mobile applications including the rating of each app and a maximum of variables controlled by the companies and displayed on the store. For this purpose, we use the API of 42matters (https://42matters.com/app-market-data). We chose this one for several reasons. First of all, it allows us to retrieve metadata from both Android (from the Google Play Store) and IOS (from the Apple App Store) apps. Then, all the fields available on the stores are included and more precisely the rating of each app and the variables about the Publication Strategy. There are also mechanisms to iterate over the applications using some criteria and filters allowing us to get a maximal amount of data. Finally, the results are returned under an easy to process format (this is explained later).

The risk while using data collected by another party is to get data of bad quality. However, 42matters extracts its data directly from the Apple App Store

and the Google Play Store. Moreover, we checked the retrieved data of several apps (selected randomly) by comparing with the actual stores and found no error. We also checked for aberrant values in the different retrieved fields and found none. We thus infer that the obtained data is reliable.

In order to retrieve data from this API, a query must be built. There are different parameters that allow to specify the criteria that the returned apps must fulfill (these parameters may be found on the website of 42matters). As we are interested in the rating, we use this latter as selection criterion to retrieve apps. We thus divide the loading of the data in different steps, specifying a range of 0.5 for the rating at each time. We first collect apps with a rating lower or equal to 0.5 and then increment this step by step until we collect apps with a rating between 4.5 and 5. The API returns a maximum of 10,000 apps for a given set of criteria and the apps are returned and sorted in descending order of number of ratings which allows us to get the apps with the highest numbers of ratings (ans thus the most reliable ratings) for each range.

This allows us to build a database with several thousands of apps for each rating. This process is repeated for the Apple App Store and the Google Play Store. As the API returns Json files, we use MongoDB to store our data because it allows to work in a document-oriented way.

3.2 Feature engineering

The second step, realised with Python, is to extract the different variables that will be later used in the prediction of the rating. The main criterion to choose these variables is that they must be controlled by the companies before launching the app on the market and displayed on the store. Indeed, the final goal is to study the importance of the Publication Strategy of an app on its rating.

We also analyze and adapt the raw data in order to detect potential problems (such as missing values or duplicated rows) and give it the right shape for the different algorithms. Indeed, some of them have requirements for the data in order to run successfully.

Moreover, we use dimensionality reduction techniques in order to check if it increases the performances of our models. Principal Component Analysis (PCA) and Random Forest are used for this purpose [1, 19]. The former allows to create uncorrelated linear combinations of the existing features and the latter allows to extract the most important features of our dataset [1, 19].

3.3 Training of the models

Finally, classification algorithms are used because our goal is to predict the number of complete stars an app will receive and not the precise rating (this is thus a classification problem and not a regression one). This step is also performed using Python. We train the following models with the prepared data: (i) Decision Tree; (ii) Random Forest; (iii) K-Nearest Neighbors (KNN); (iv) Support Vector Machines (SVM); (v) Neural Networks and more specifically MultiLayer Perceptron (MLP).

Most of these models have been chosen using the flowchart presented in [8]. We just added the Decision Tree and the MLP that are not listed in the mentioned chart as the former is very simple and the latter is very flexible [6, 9]. Also, we do not use the linear SVC (Support Vector Classifier) model as we already chose SVM. Indeed, the latter may be used with a linear kernel and, even if it represents another implementation, the results are often similar with the linear SVC [10].

The training process is the following for each model. We first divide our data into train data (80%) and test data (20%). The train data is used to tune the hyperparameters and the test data allows us to measure the accuracy of each model on data never seen before. In order to find the optimal hyperparameters for each model, we proceed in different steps. First, we analyze the evolution of the accuracy depending on the values of some hyperparameters. Then, a grid search is used to compare different configurations based on cross-validation. Once this tuning is finished, we test the final model with the test data.

Three kinds of accuracies are calculated for each model: a training accuracy, a validation accuracy and a testing accuracy. The first is the accuracy obtained when predicting the labels of the data used to train the model. The second is the accuracy obtained with the best configuration while tuning the hyperparameters. The third accuracy is obtained by predicting the labels of the test data. The training and the validation accuracies are biaised because they are used to enhance the model.

Python is also used for this step and more precisely the functions from the scikit-learn package. The performances of the different algorithms are discussed and compared. The performances of a random prediction are also computed in order to have a reference point.

4 Results

4.1 Feature engineering

Selection of the variables. For the Android apps, we obtained a database containing 96,178 rows and 53 fields. The entire list of fields may be found on the website of 42matters. Obviously, a big part of these variables did not interest us. As a reminder, we only wanted to keep the variables controlled by the companies and displayed on the store. Moreover, we had to transform some fields in order to make them usable. The kept fields are presented in table 1. We did not take the list of countries where the app is available neither the languages supported by the app as features because we consider that these variables may be impacted by the success of the app after the commercialization.

The database of IOS apps counted 81,000 rows and 52 fields. It is important to note that some fields were available for Android but not for IOS and vice versa. This explains why different fields were used for IOS in comparison with Android. The used selection criteria was also that the variables must be controlled by the companies and displayed on the store. Table 2 presents the kept fields for IOS apps.

Table 1. Selected features for Android apps

Name	Description
Category	The category of the app.
Promo video	Whether the app contains a promotional video.
Price	The price of the app (in $).
Content rating	A rating of the content of the app.
Number of screenshots	The number of screenshots displayed on the store.
Size	The size of the app in bytes.
In-app purchases	Whether in-app purchases are available in the app.
Minimum in-app purchase	The minimum cost of in-app purchase.
Maximum in-app purchase	The maximum cost of in-app purchase.
Length of the description	The length of the description of the app.
Length of the short description	The length of the short description of the app.
Length of title	The length of the title of the app.
Ads	Whether the app contains ads.
Rating	The rating of the app.

Table 2. Selected features for IOS apps

Name	Description
Length of the description	The length of the description of the app.
Number of iPhone screenshots	The number of iPhone screenshots of the app.
Number of iPad screenshots	The number of iPad screenshots of the app.
iPhone	Whether the app is available on iPhone.
iPad	Whether the app is available on iPad.
VPP distribution	Whether the app supports VPP distribution.
Content rating	A rating of the content of the app.
Size	The size of the app in bytes.
Game center	Whether the app supports Game Center.
Primary category	The primary category of the app.
Price	The price of the app.
Length of the title	The length of the title of the app.
Rating	The rating of the app.

Analysis and preparation of the data. First of all, as we created our databases in different steps, there were some duplicated rows. Indeed, the rating of an app may change from one week to another. We thus deleted them.

Then, we managed the different categorical features (the category for example). We first checked the distribution of the values of each column to see if some of them were under-represented. When several values appeared less than 1000 times in a column, we aggregated them into a new category "other" in order to keep only the relevant values. As explained before, some algorithms are not able to handle categorical input. We therefore decided to use one-hot-encoding [17]. This method allows to transform a categorical column in several boolean columns (one for each category) [17].

The next step was the scaling of the numerical columns. This was another requirement for some of our models. We decided to use standardization ($z = (x - \mu)/\sigma$) to decrease the impact of the outliers in the features [17]. In order to avoid data leakage, we scaled the data using only the training sets.

We also had to modify the rating in order to have the classes that we wanted. We used the following rounding rules: rating class = 0 if rating < 0.5; rating class = 1 if rating ≥ 0.5 and < 1.5; rating class = 2 if rating ≥ 1.5 and < 2.5; rating class = 3 if rating ≥ 2.5 and < 3.5; rating class = 4 if rating ≥ 3.5 and < 4.5; rating class = 5 if rating ≥ 4.5.

Finally, as the number of rows with missing values in the resulting datasets was very low (less than 100), we just deleted them. We also kept only the applications with at least 50 ratings from the customers. Indeed, for several apps, the average rating was not calculated on enough individual ratings to be significant.

For the Android apps, our final dataset consisted of 64504 rows and 65 columns. An important consequence of the treatment described above is that there was no application with a rating of 0 star any more, reducing the number of classes to 5.

The final IOS dataset had 41856 rows and 31 columns. As for Android, there was no app with a rating of 0 star.

Reduction of dimensionality. Two methods were used to decrease the number of features. First, we applied a Principal Component Analysis keeping 95% of the variance. This returned 24 principal components for the Android dataset and 15 principal components for the IOS dataset. Then, we used the built-in feature selection of the Random Forest to keep only the variables with a relative importance bigger than 0.01. Based on this process, 9 features remained for the Android dataset: the length of the description, the size, the length of the short description, the length of the title, the number of screenshots, the most expensive in-app purchase, the presence or absence of advertisements, the least expensive in-app purchase and the presence or absence of a promo video. Regarding the application of this second method to the IOS dataset, 10 features remained: the size, the length of the description, the length of the name, the number of screenshots, the number of screenshots on ipad, the fact that the app is a game or not, the fact that GameCenter is enabled or not, the fact that the

app is free or not, the fact that the content of the app is rated "4+" or not and the fact that the category of the app is "other" or not.

4.2 Training of the models

The results of the different models for Android apps is summarized in table 3 and table 4 summarizes the results of the different models for IOS apps. Each time, a reference point is given with a random prediction (1/number of classes). All these results have been achieved using the methodology presented in subsection 3.3. As a recall, the training accuracy is biaised because it is used during the optimisation of the parameters of the models. The validation accuracy is also biaised as it is used to tune the hyperparameters. The testing accuracy is thus the most significant approximation of the true accuracy of the model.

Table 3. Training of the models for Android apps

Model	Selected hyperparameters and final values	Training accuracy	Validation accuracy	Testing accuracy
Random	/	20%	20%	20%
Decision Tree	- Maximal depth = 10 - Maximal number of leaf nodes = 90	47.5%	46.4%	46.3%
Random Forest	- Maximal depth of the trees = 20 - Number of estimators = 75	78.9%	49.5%	50.1%
KNN	- Number of neighbors = 30	51.4%	46.9%	47.5% 44.3%[1] 46%[2]
SVM	- Regularization parameter (C) = 7	54.9%	45.2%[2]	48.8% 45.5%[1] 46.9%[2]
MLP	- Number of layers = 1 - Number of neurons = 20	50.3%	48.7%	49% 46.1%[1] 47.4%%[2]

1: using only the most important features
2: using PCA

Looking at the results for the Android dataset, we can see that the performances of all the models are rather close during the testing phase. It varies between 44.3% and 50.1% with Random Forest achieving the best score. This last result is more than twice the result of a pure random prediction. Another conclusion is that both feature selection methods fail to improve the results. Using the built-in feature importance evaluation of Random Forest, we can assess the relative importance of the different features. The most important features for Android are thus: the length of the description, the size, the length of the short description, the length of the title and the number of screenshots.

Analyzing the results for the IOS dataset, table 4 shows that all the models have close testing performances. It varies between 45.2% and 49.6% and the best score is achieved by Random Forest. We can see that the random prediction is completely outperformed. This time, PCA seems to enhance slightly the results for KNN and SVM. Using the built-in feature importance evaluation of Random

Table 4. Training of the models for IOS apps

Model	Selected hyperparameters and final values	Training accuracy	Validation accuracy	Testing accuracy
Random	/	20%	20%	20%
Decision Tree	- Maximal depth = 10 - Maximal number of leaf nodes = 200	49.1%	45.8%	46.4%
Random Forest	- Maximal depth of the trees = 15 - Number of estimators = 125	72%	48.1%	49.6%
KNN	- Number of neighbors = 30	49%	43.7%	45.2% 45.2%[1] 45.6%[2]
SVM	- Regularization parameter (C) = 2	48.7%	45.35%[2]	47.3% 47%[1] 47.5%[2]
MLP	- Number of layers = 1 - Number of neurons = 16	47.7%	47.2%	48% 47.3%[1] 47.7%[2]

1: using only the most important features
2: using PCA

Forest, the most important features for IOS are: the size, the length of the description, the length of the title and the number of iPhone screenshots.

5 Discussion

5.1 Comparison of the performances

We can see that the models achieve rather close results when predicting the rating of applications from both the Apple App Store and the Google Play Store. The best score is achieved by the Random Forest model for both stores and is around 50%. This is more than twice the result of a random prediction (20%). This means that, in 50% of the cases, the number of stars that a mobile application will get may be predicted using only its Publication Strategy.

What these results suggest is that the concept of Publication Strategy defined earlier in this paper seems to be in practice a real and strong predictor of the rating of an app, and therefore of its potential success on a platform. Although we do no study a particular Publication Strategy in this paper, we find evidences that it actually matters, and that it could be leveraged by apps designers when making decisions about a new app.

Another conclusion is that the most important features are the same for both stores: the length of the description, the size, the length of the title and the number of screenshots.

5.2 Limitations and future works

First of all, the number of features extracted from the raw data could be increased to enhance the precision of the models. Indeed, the variables presented in this work are quite basic. In order to find more of them, a first idea would be to

use text mining on the description and the title. These two fields seem to be very important in the prediction of the rating while taking only the length into account. More information could be extracted from these two fields.

Then, we only looked at the definition of the concept "Publication Strategy" and its importance in the prediction of the rating of an app. We did not study a particular Publication Strategy nor the different types of Publication Strategies. A future study could focus on the identification of patterns of Publication Strategies that apps designers could use to enhance the performance of their apps.

Another possibility would be to study a decision support system guiding the companies in the decisions about the Publication Strategy of their apps. This could reduce the uncertainty of their choices and reduce the number of low rated apps due to bad presentation.

Finally, it would also be interesting to analyze the impact of the Publication Strategy on the number of downloads for apps that have the same rating. It would allow to see if the presentation of an application makes a difference in terms of popularity for apps of the same quality.

6 Conclusion

We presented the concept of Publication Strategy of a mobile application and studied its importance by looking at the extent to which the rating of an app can be predicted solely based on its Publication Strategy. The following classification algorithms were used to predict the rating: decision tree, random forest, KNN, SVM, MLP. We used metadata about 64504 Android apps and 41856 IOS apps. The performances of the algorithms were compared and discussed. We discovered that, for both stores, 50% of the ratings could be predicted using only variables controlled by the companies before the commercialization and displayed on the store, i.e. the Publication Strategy. We thus concluded that this Publication Strategy actually matters for the success of an app and that more research is needed to support apps designers in their decisions regarding the Publication Strategy. Moreover, the most important variables for the predictions were the length of the description, the size, the length of the title and the number of screenshots. The limitations were discussed and some avenues for future research were presented.

References

1. Agarwal, R.: The 5 feature selection algorithms every data scientist should know (2019), https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2
2. Apple: Ratings, reviews, and responses (2020), https://developer.apple.com/app-store/ratings-and-reviews/
3. Colgan, M.: How important are mobile app ratings & reviews? (2019), https://tapadoo.com/mobile-app-ratings-reviews/

4. Daimi, K., Hazzazi, N.: Using apple store dataset to predict user rating of mobile applications. In: 2019 International Conference on Data Science. pp. 28–33 (2019)
5. Gordon, G.: User ratings & reviews: How they impact aso – ultimate guide (2018), https://thetool.io/2018/user-ratings-reviews-aso-guide#How_does_User_Feedback_Impact_ASO
6. Gupta, P.: Decision trees in machine learning (2017), https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
7. Handley, L.: Nearly three quarters of the world will use just their smartphones to accessthe internet by 2025 (2019), https://www.cnbc.com/2019/01/24/smartphones-72percent-of-people-will-use-only-mobile-for-internet-by-2025.html
8. scikit learn: Choosing the right estimator (2019), https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
9. scikit learn: Neural network models (supervised) (2019), https://scikit-learn.org/stable/modules/neural_networks_supervised.html
10. scikit learn: Support vector machines (2019), https://scikit-learn.org/stable/modules/svm.html
11. Lee, G., Raghu, T.S.: Determinants of mobile apps' success: Evidence from the app store market. Journal of Management Information Systems **31**(2), 133–170 (2014)
12. Legal'Easy: Les chiffres des utilisateurs d'applications (2019), https://www.my-business-plan.fr/chiffres-application
13. Lu, J., Liu, C., Wei, J.: How important are enjoyment and mobility for mobile applications? Journal of Computer Information Systems **57**(1), 1–12 (2017)
14. Meng, J., Zheng, Z., Tao, G., Liu, X.: User-specific rating prediction for mobile applications via weight-based matrix factorization. In: 2016 IEEE International Conference on Web Services (ICWS). pp. 728–731. IEEE (2016)
15. Monett, D., Stolte, H.: Predicting star ratings based on annotated reviews of mobile apps. In: 2016 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 421–428. IEEE (2016)
16. Picoto, W.N., Duarte, R., Pinto, I.: Uncovering top-ranking factors for mobile apps through a multimethod approach. Journal of Business Research **101**, 668–674 (2019)
17. Rençberoğlu, E.: Fundamental techniques of feature engineering for machine learning (2019), https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114
18. Sarro, F., Harman, M., Jia, Y., Zhang, Y.: Customer rating reactions can be predicted purely using app features. In: 2018 IEEE 26th International Requirements Engineering Conference (RE). pp. 76–87. IEEE (2018)
19. Science, E.D.: Dimensionality reduction algorithms: Strengths and weaknesses (2019), https://elitedatascience.com/dimensionality-reduction-algorithms#feature-selection
20. statista: Mobile app usage - statistics & facts (2019), https://www.statista.com/topics/1002/mobile-app-usage/
21. Yang, H.: Bon appétit for apps: young american consumers' acceptance of mobile applications. Journal of Computer Information Systems **53**(3), 85–96 (2013)