MASTER IN BUSINESS ENGINEERING PROFESSIONAL FOCUS IN DATA SCIENCE

Beyond Apps Functionalities

Could companies use the publication strategy of their app to predict its rating?

Lega, Mathieu

Award date:
2020

Awarding institution:
University of Namur

Link to publication

Beyond Apps Functionalities: Could companies use the publication strategy of their app to predict its rating?

Mathieu LEGA

Directeur: Prof. S. FAULKNER

Mémoire présenté
en vue de l'obtention du titre de
Master 120 en ingénieur de gestion, à finalité spécialisée
en data science

ANNEE ACADEMIQUE 2019-2020

Université de Namur, ASBL
Faculté des Sciences économiques, sociales et de gestion – Département des Sciences de gestion

Rempart de la Vierge 8, B-5000 Namur, Belgique, Tel. +32 [0]81 72 48 41/49 58, Fax +32 [0]81 72 48 40

Acknowledgements

First of all, I would like to thank my promoter Prof. Faulkner for his availability, his feedbacks and the support he provided during the realisation of this master's thesis.

Then, special thanks to Prof. Frenay for helping me with the machine learning part of this work through practical suggestions.

Finally, I very much appreciate the unwavering support of my family who never stops encouraging me.

Contents

Chapter 1

Introduction

Nowadays, more than two thirds of the global population owns a smartphone [1]. On average, a smartphone is used 3 hours a day and contains 100 applications [1]. 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 [2] and 194 billions of mobile apps were downloaded [3]. Some predictions state that the global mobile apps revenue could get near to $1000 billions by 2023 [3]. 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.

All these numbers highlight 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 something 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 others words, developing a new app does not always come with financial success, and companies should be cautious about this. 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 [4]. 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 [5, 6]. Moreover, 80% of the customers check the rating of an app before downloading it [6]. If an app has a rating of 2 stars out of 5, only 15% of the customers state that they will consider downloading it [5, 6]. This percentage increases to 96% if the rating is 4 stars [5, 6]. We can consequently say that the rating also has an impact on the conversion rate to download.

For all these reasons, we decided to look at the extent to which the rating, and more precisely the number of complete stars, of a mobile application can be predicted using only 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. Indeed, the intent here is to discover if it would be possible to better control the presentation of a new app (which is easier to change than the functionalities of the app) by testing several configurations and finding the one that is the most likely to be appreciated by the customers. This would allow companies to minimize the number of low rated (and thus riskier) apps launched on the market. Moreover, we will compare the results of applications available on the Google Play Store store and applications available on the Apple App Store. Our research question can be stated as follows:

- To which extent is it possible to predict the rating of a mobile application using only its publication strategy?

For this purpose, we will first create two databases containing metadata about applications. One database will contain metadata of apps of the Google Play Store and the other database will contain metadata of apps of the Apple App Store. Then, feature engineering will be used in order to extract the most important variables from the raw data. Finally, machine learning will be used and more precisely the following classification algorithms: decision tree, random forest, k-nearest neighbors, support vector machine and neural networks. The results of these algorithms will be compared and discussed.

Chapter 2

Literature review

2.1 Models of user adoption

The phenomenon of mobile applications has been addressed in several ways in the literature. Some authors analyze customers' feedback in order to test models to gain insights in this domain. Hongwei Yang (2013) 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 [7]. He tested his model with a web survey answered by American college students [7]. 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 [7]. 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 [7]. Hans van der Heijden (2004) got different results while investigating how user acceptance models differ based on the type of information system used: utilitarian or hedonic [8]. He used a cross-sectional survey answered by 1144 users of one hedonic information system for this purpose [8]. For the analysis, he used a TAM2 model, semantic differential scales, a factor analysis and a structural equation model [8]. He concluded that perceived usefulness had a smaller impact on intentions to use than perceived ease of use and perceived enjoyment for hedonic information systems [8]. As a consequence, the impact on user acceptance models for hedonic information systems is the following: ease of use and enjoyment replace perceived usefulness as dominant predictive factors [8]. The study of Sang Chon Kim, Doyle Yoon and Eun Kyoung Han (2014) had 2 goals [9]. First of all, they wanted to find factors that have an impact on smartphone app usage based on: behavioral theories, uses and gratification theory and previous empirical studies [9]. Then, they wanted to test a model predicting the way factors and usage of technology are linked [9]. In their study, they used the answers of 257 undergraduates coming from university [9]. They identified the following factors as significant to predict the use of an app: perceived informative and entertaining usefulness, perceived ease of use, and user review [9].

2.2 Models of App Success Factors

Another approach is to focus on the analysis of top grossing charts. Gunwoong Lee and Raghu (2014) decided to look at how the success in an app market is impacted by both app-related and seller-related characteristics [10]. 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 [10]. For this purpose, they used a generalized hierarchical modeling approach, a hazard model and a count regression model [10]. They concluded that, at a seller-level, it is very important to diversify across categories to achieve sales performance [10]. 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 [10]. Picoto, Duarte and Pinto (2019) also used the tops grossing for their study [11]. The aim of their study was to find the factors that could have an impact on the ranking and the success of an app [11]. 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 [11]. 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 [11]. 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 [11]. A surprising result is that the higher the user rating, the lower the probability for this app to be in the top 50 [11]. 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 [11].

2.3 Enjoyment in Apps

June Lu, Chang Liu and June Wei (2016) decided to focus only on two factors: enjoyment and mobility [12]. In their study, they tried to understand the link between the perception of these factors and the intention to continue using an app [12]. They used a second-stage continuance model and data of 584 smartphone users collected with a survey [12]. 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 [12]. This confirms the results of Hans van der Heijden (2004) stating that perceived enjoyment has a big impact on intentions to use [8].

2.4 Prediction of ratings

A well-known problematic in this topic is the prediction of the user rating of an application. Dagmar Monett and Hermann Stolte (2016) have tackled this problem in their work [13]. They used a corpus of 1,760 annotated reviews about 130 mobile apps available on the Google Play Store [13]. Their goal was to predict the rating based on the polarity of subjective phrases found in the reviews [13]. For this purpose, several computational models have been used and evaluated [13]. They concluded that rating could be well predicted even using only phrase-level sentiment polarity [13]. Jingke Meng, Zibin Zheng, Guanhong Tao and Xuanzhe Liu (2016) implemented a weight base matrix factorization (WMF) capturing user-specific interests in order to predict an app rating for a specific user [14]. 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 [15]. In order to predict the rating of applications, Kevin Daimi and Noha Hazzazi (2019) decided to use the following algorithms: Linear Regression, Neural Networks, Support Vector Machines, Random Forest, M5 Rules, REP Tree and Random Tree [15]. 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 [15]. They found that Random Forest produced the best results when predicting the rating of an application from the Apple Store dataset [15]. Finally, Federica Sarro, Mark Harman, Yue Jia and Yuanyuan Zhang (2018) focused on predictions achieved by Case Based Reasoning (CBR) taking only the technical features of the apps into account [16]. 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 [16]. As a result, they discovered that, in 89% of the cases, the rating of an app could be perfectly predicted [16]. They also discovered that only 11-12 applications were sufficient to achieve the best prediction when using a case-based prediction system [16]. They thus concluded that it is possible to accurately predict the rating of an app taking only its features into account [16].

2.5 Positioning of this work

Our work differentiates from all of the above because we will not use feedbacks from users to test a model, neither focus on top charts. Instead, we will try to maximize our precision when predicting the rating given by the customers of an app. More precisely, we will try to predict the number of stars an app will receive using classification algorithms. We will thus use the well-known star rating used by several stores. What distinguishes this work from the other works about prediction is the fact that we will only use variables that are under the control of the companies and displayed on the store as predictors. This will be discussed with more details in chapter 4. Moreover we will 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 will thus be trained models of machine learning able to predict a rating (the number of stars) for an application. The differences between this work and the presented ones are summarized in table 2.1.

Title	Goal	Tools	Data
Bon Appétit for Apps: Young American Consumers' Acceptance of Mobile Applications (2013) [7]	- Predict use of mobile apps, attitudes of customers and intent to use.	- Theory of Planned Behavior. - Technology Acceptance Model. - Uses and Gratification Theory.	- Web survey answered by American college students.
User acceptance of hedonic information systems (2004) [8]	- Find the differences in user acceptance models based on the type of information systems: utilitarian or hedonic.	- TAM2 model. - Semantic differential scales. - Factor analysis. - Structural equation model.	- Cross-sectional survey answered by 1144 users.
Antecedents of mobile app usage among smartphone users (2014) [9]	- Find factors that have an impact on smartphone app usage.	- Behavioral theories. - Uses and gratification theory. - Previous empirical studies.	- Answers of 257 undergraduates coming from university.
Determinants of Mobile Apps Success: Evidence from App Store Market (2014) [10]	- Find how app-related and seller-related characteristics impact success in an app market.	- Generalized hierarchical modeling approach. - Hazard model. - Count regression model.	- Top 300 grossing charts of Apple App Store.
Uncovering top-ranking factors for mobile apps through a multimethod approach (2019) [11]	- Find factors that could have an impact on the ranking and the success of an app.	- Multivariate logistic regression. - Fuzzy Set Qualitative Comparative Analysis.	- 500 top grossing apps from the Apple App Store.
How Important Are Enjoyment and Mobility for Mobile Applications? (2016) [12]	- Understand the link between the perception of enjoyment and mobility and the intention to continue using an app.	- Second-stage continuance model.	- Data of 584 smartphone users collected with a survey.
Predicting Star Ratings based on Annotated Reviews of Mobile Apps (2016) [13]	- Predict the rating of an app based on annotated reviews.	- Several computational models.	- 1,760 annotated reviews about 130 mobile apps available on the Google Play Store.
User-Specific Rating Prediction for Mobile Applications via Weight-based Matrix Factorization (2016) [14]	- Predict the rating of a user for an app based on his usage of apps.	- Weight base matrix factorization (WMF).	- Logs of user's downloaded and uninstalled apps involving 5057 users and 4496 apps.
Using Apple Store Dataset to Predict User Rating of Mobile Applications (2019) [15]	- Predict the rating of an app based on mainly non-technical features.	- Linear Regression. - Neural Networks. - Support Vector Machines. - Random Forest. - M5 Rules. - REP Tree. - Random Tree.	- 7197 Apple Store apps.
Customer Rating Reactions Can Be Predicted Purely Using App Features (2018) [16]	- Predict the rating of an app based on technical features.	- Case Based Reasoning (CBR).	- 9588 apps from the BlackBerry App World store. - 1949 apps from the Samsung Android App store.
This work	- Predict the rating of an app based on its publication strategy.	- Decision tree. - Random Forest. - K-Nearest Neighbors. - Support Vector Machines. - Neural Networks.	- Metadata from 96,178 Android apps. - Metadata from 81,000 IOS apps.

Table 2.1: Comparison of the presented works

Chapter 3

Theoretical review

3.1 Supervised learning

Supervised learning is the process of training a machine learning algorithm with pairs of input-output data in order to make predictions [17, 18]. The training dataset is composed of different features (the input) and a label (the output) [17, 18]. The goal of the training is to allow the algorithm to be able to predict the label of new unlabeled data based on the features [17, 18]. Two different tasks may be accomplished: classification and regression [17, 18]. In a classification task, the output to predict is discrete (classes, groups, categories, etc) [17]. In a regression task, the output is continuous [17]. The algorithms presented below (3.2 until 3.6) may all be used for classification.

One of the biggest risks while using supervised learning is overfitting. It is a phenomenon that may occur when a machine learning model is trained [19]. It means that the model learns the noise in the training data and not only the patterns of interest [19]. It thus decreases the generalisation performance of the model [19]. A symptom of overfitting is a low training error and a high testing error [20]. It often occurs when the model is too complex for the task [20]. In order to avoid this phenomenon, different hyperparameters have to be tuned depending on the model [20].

3.2 Decision Tree

This method creates a decision-making process based on a tree structure [21]. A tree is composed of internal nodes, edges and leaves [21]. An internal node represents a condition that will split the tree into edges (branches) [21]. When a branch stops splitting, its end is called a leaf [21]. It represents the decision [21]. Each split is based on a feature of the dataset [21].

This method has several advantages. First of all, the resulting decision-making process is easy to understand and to interpret [21, 22]. Then, it may be used with numerical and categorical data and with multi-output problems [21, 22]. Finally, it makes an implicit feature selection [21, 22].

There are also some disadvantages. Decision trees often overfit (some parameters as the maximal depth may be used to avoid it) and are quite unstable [21, 22]. Then, the optimal tree is very difficult to generate [21, 22]. Finally, a decision tree may be biased if some classes are dominant [21, 22].

3.3 Random Forest

This algorithm uses a bagging approach to combine the predictions of several models [23, 24]. Several decision trees are trained at the same time on sub-samples of the dataset [23, 24]. Then, the most frequent prediction (for classification) is used in order to combine the different predictions and improve the prediction performance [23]. Random forest may also be used for feature selection [25].

The biggest advantage of this technique in comparison with decision trees is that it is more stable [25]. However, the resulting decision-making process becomes harder to understand and to replicate, i.e., it is closer to a black box.

3.4 KNN

The idea behind the K-Nearest Neighbors (KNN) algorithm is that similar points are close to each other [26, 27]. It therefore calculates the distances between the different data points and use the k nearest to make a prediction for a given data point (using a majority vote) [26, 27].

Some advantages of this method are its simplicity and the low number of parameters to tune [26]. However, it may take a lot of time when the size of the dataset or the number of predictors (k) increases [26]. Moreover this algorithm only works with numerical columns [26].

3.5 SVM

Support Vector Machines (SVM) is an algorithm that tries to find the optimal hyperplane that separates the data into the different classes (in the case of classification) using what is known as the kernel trick [28]. This latter impacts the way of computing dot products [28].

This algorithm remains efficient even if there are more dimensions than samples or if there is a high number of dimensions [28, 29]. However, a lot of time may be required to process large datasets [28, 29]. Moreover, its performance may vary significantly depending on the separability of the classes [28].

3.6 MLP

Multi-Layer Perceptron (MLP) is a type of Artificial Neural Networks (ANNs) and, as such, is inspired by the human brain and central nervous system [30]. A MLP is composed of several connected layers (themselves composed of neurons) as illustrated in figure 3.1.

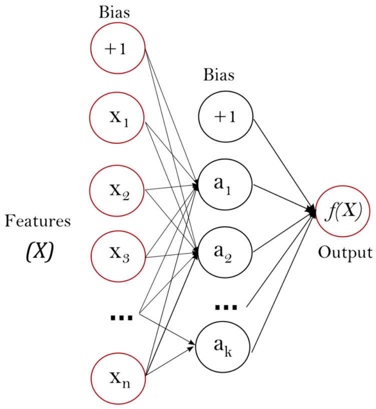


Figure 3.1: one hidden layer MLP [31]

The input layer holds the input data [30]. The hidden layers are in charge of the processing of the data [30]. Finally, the output layer contains the output of the whole process [30]. Each arc has a weight that will be modified during the process [30]. The interested reader is directed to [30] for more information.

This model is rather flexible and allows to learn non-linear models [31, 32]. However, the accuracy of the model may vary based on the initialization of the weights [31]. Moreover, different hyperparameters (such as the number of layers and the number of neurons of each layer) have to be tuned and attention must be paid to feature scaling [31].

Model	Advantages	Disadvantages
Decision Tree	- Easy to understand and to interpret. - May be used with numerical and categorical data. - No need to scale the features. - Implicit feature selection.	- High risk of overfitting. - Unstable. - May be biaised if some classes are dominant.
Random Forest	- May be used with numerical and categorical data. - No need to scale the features. - Implicit feature selection.	- High risk of overfitting. - May be biaised if some classes are dominant. - The decision process is a black box.
KNN	- Low number of parameters to tune. - Simple.	- May take a lot of time when the size of the dataset or k increases. - Does not work with categorical features. - Sensitive to feature scaling.
SVM	- Remains efficient even if there are more dimensions than samples or if there is a high number of dimensions.	- May take a lot of time when the size of the dataset increases. - Its performance may vary significantly depending on the separability of the classes. - Does not work with categorical features. - Sensitive to feature scaling.
MLP	- Rather flexible and allows to learn non-linear models.	- Its performance may vary significantly depending on the initialization of the weights. - Different hyperparameters need to be tuned. - Does not work with categorical features. - Sensitive to feature scaling.

Table 3.1: Comparison of the different models

Chapter 4

Methodology

4.1 Creation of the databases

To answer our research question, 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 [33]. We chose this one for several reasons. First of all, it allows us to retrieve metadata from both Android and IOS apps. Then, a lot of fields are included and more precisely the rating of each app and several 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 in the next paragraph). 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 [33]. We can thus infer that the obtained data is reliable.

In order to retrieve data from this API, cURL is used and a query must be built. There are different parameters that allow to specify the criteria that the returned apps must fulfill. A table explaining the syntax of the queries is available at [34] for Android and at [35] for IOS. Each query returns a Json file containing apps corresponding to the specified criteria. A maximum of 50 apps may be returned for a single query but it is possible to use a parameter (the page number) to keep the same criteria and get the 50 following apps. However, it is not possible to iterate over more than 10,000 apps with the same criteria. As we want to get more apps and are interested in the rating, we use this latter as criterion to be able to retrieve a maximal amount of data. We divide the loading of the data in different steps, specifying a range of 0.5 for the rating at each time. We begin with apps with a rating lower or equal to 0.5 and increment this until apps with a rating between 4.5 and 5.

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.

4.2 Feature engineering

Then, we 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 predict the rating of apps before commercializing them.

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 [36, 37]. The former allows to create new uncorrelated features and the latter allows to extract the most important features of our dataset [36, 37].

This step is realised with python.

4.3 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 step is also performed using python. We train the following models with the prepared data:

- Decision Tree
- Random Forest
- KNN

- SVM
- MLP

Most of these models have been chosen using the flowchart presented in [38]. 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 [21, 31]. Also, we do not use the linear SVC 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 [29].

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.

Chapter 5

Results: Android applications

5.1 Feature engineering

5.1.1 Selection of the variables

We obtained a database containing 96,178 rows and 53 fields. The entire list of fields may be found at [39]. 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 and their possible transformations are presented in table 5.1. 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.

Name	Description	Type	Transformation	Value examples
category	The category of the application.	string	/	"Action", "Lifestyle"
promo_video	A link to a promotional video.	URL link	1 if there is a link and 0 if not.	0, 1
price_numeric	The price of the application (in $).	number	/	0.99, 3.49
content_rating	A rating of the content of the application.	string	/	"Everyone", "Teen"
screenshots	A list of URLs corresponding to screenshots of the application.	list of URLs	The length of the list is taken as the number of screenshots.	1, 8
size	The size of the application in bytes.	number	/	3460300, 4508876
iap	Whether in-app purchases are available in the application or not.	boolean	/	true, false
iap_min	The minimum in-app purchase (only present if iap is True).	number	0 when not present.	0, 0.99
iap_max	The maximum in-app purchase (only present if iap is True).	number	0 when not present.	1.99, 43.99
description	The description of the application.	string	The length is taken.	152, 2163
short_desc	A promotional text for the application.	string	The length is taken.	54, 55
title	The title of the application.	string	The length is taken.	12, 18
contains_ads	Whether the application contains ads.	boolean	/	true, false
rating	The rating of the application.	number	/	1.5, 1.9285714626312

Table 5.1: Selected features for Android apps

5.1.2 Analysis and preparation of the data

First of all, as we created our database 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 and arrived to a database of 94,581 rows.

Then, we managed the different categorical features (category, content_rating). 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 [40]. This method allows to transform a categorical column in several boolean columns (one for each category) [40].

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 [40].

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 dataset 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. An important consequence is that there was no application with a rating of 0 star any more, reducing the number of classes to 5. Our final dataset thus consisted of 64504 rows and 65 columns. Figure 5.1 and table 5.2 describe this latter.

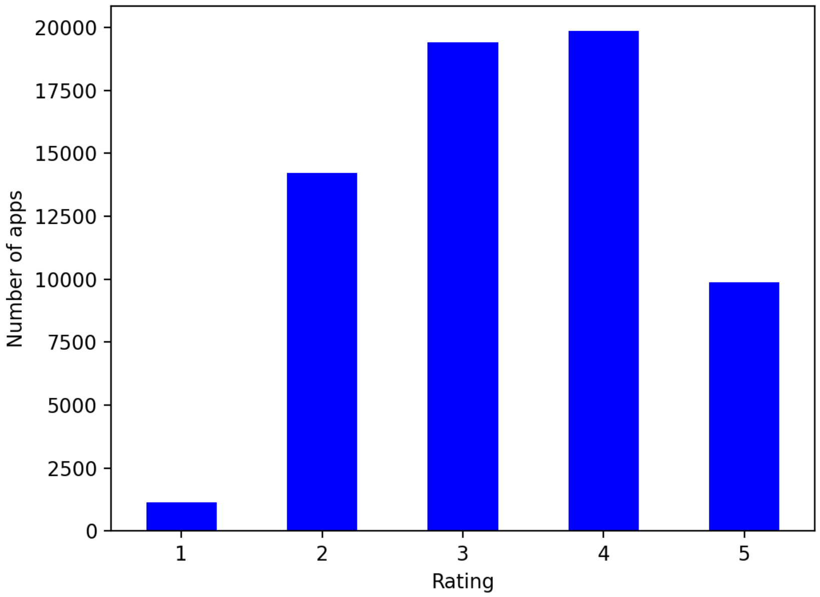


Figure 5.1: Distribution of the ratings for Android apps

Metric	**Value**
Number of rows	64504
Number of columns	65
Number of free apps	63556
Most represented categories	Tools (5550 apps) Entertainment (4659 apps) Finance (3158 apps)
Most represented content ratings	Everyone (51519 apps) Teen (8101 apps) 10+ (2454 apps)
Number of apps containing iap	18410
Number of apps containing ads	36464
Number of apps with a promo video	21231
Mean length of the description	1389 characters
Mean length of the title	22 characters
Mean size	27.7 megabytes
Mean number of screenshots	10

Table 5.2: Summary of the Android dataset

5.1.3 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. 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. The 9 remaining features based on this process are represented in figure 5.1.

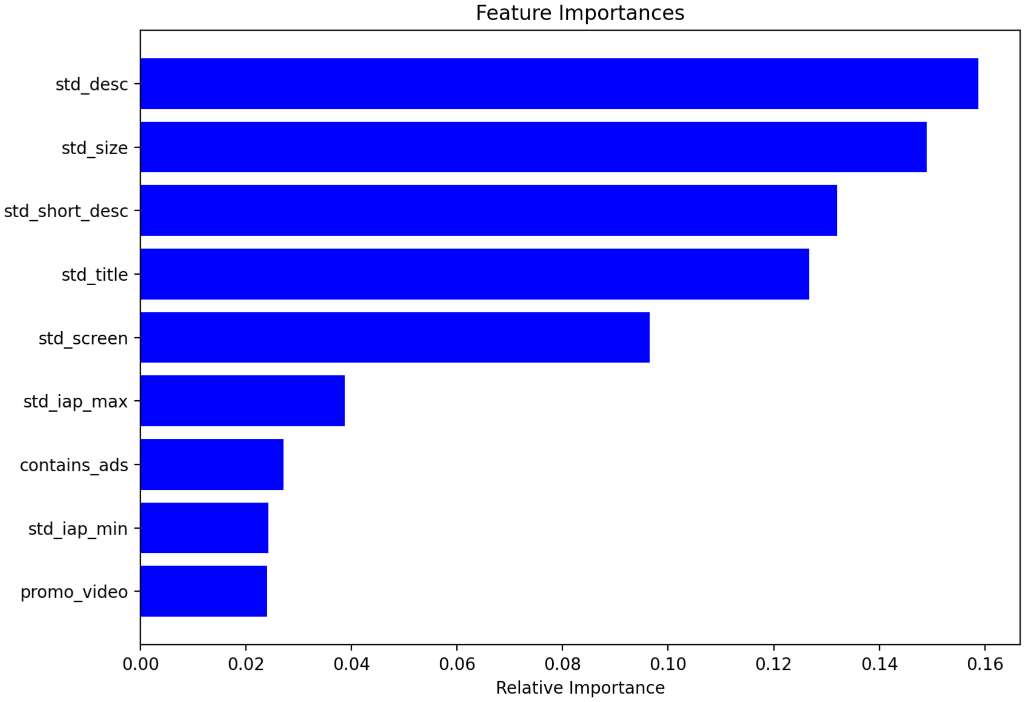


Figure 5.2: The most important features for Android apps

We can see that 5 features are significantly more important than the others: the length of the description, the size, the length of the short description, the length of the title and the number of screenshots.

5.2 Training of the models

The training of the different models for Android apps is summarized in table 5.3. A reference point is given with a random prediction (1/number of classes).

Model	Used function	Selected hyperparameters and final values	Training accuracy	Validation accuracy	Testing accuracy
Random	/	/	20%	20%	20%
Decision Tree	tree.DecisionTreeClassifier()	- maximal depth = 10 - maximal number of leaf nodes = 90	47.5%	46.4%	46.3%
Random Forest	RandomForestClassifier()	- maximal depth of the trees = 20 - number of estimators = 75	78.9%	49.5%	50.1%
KNN	KNeighborsClassifier()	- number of neighbors = 30	51.4%	46.9%	47.5% 44.3%[1] 46%[2]
SVM	svm.SVC()	- Regularization parameter (C) = 7	54.9%	45.2%[2]	48.8% 45.5%[1] 46.9%[2]
MLP	MLPClassifier()	- 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

Table 5.3: Training of the models for Android apps

All these results have been achieved using the methodology presented in section 4.3. The confusion matrices corresponding to the testing results are available in appendix A. 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.

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 confusion matrices, we can also see that the rating 1 star is often neglected by the models as there are too few samples. Moreover, the models are less accurate when predicting a rating of 5 stars. A reason could be the lower amount of data for this rating.

Chapter 6

Results: IOS applications

6.1 Feature engineering

6.1.1 Selection of the variables

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 6.1 presents the kept fields for IOS apps.

Name	Description	Type	Transformation	Value examples
description	The description of the application.	string	The length is taken.	183, 491
screenshotUrls	A list of URLs corresponding to iPhone screenshots of the application.	list of URLs	The length of the list is taken.	2, 4
ipadScreenshotUrls	A list of URLs corresponding to iPad screenshots of the application.	list of URLs	The length of the list is taken.	0, 2
iPhone	Whether the application is available on iPhone.	boolean	/	true, false
iPad	Whether the application is available on iPad.	boolean	/	true, false
isVppDeviceBasedLicensingEnabled	Whether the application supports VPP distribution.	boolean	/	true, false
contentAdvisoryRating	A rating of the content of the app.	string	/	"4+", "9+"
fileSizeBytesNumeric	The size of the application in bytes.	number	/	171,753, 21,710,848
isGameCenterEnabled	Whether the application supports Game Center.	boolean	/	true, false
primaryGenreName	The primary category of the application.	string	/	"Utilities", "Education"
price	The price of the application (multiple currencies).	number	true if above 0, false otherwise.	true, false
trackCensoredName	The name of the application.	string	The length is taken.	12, 20
averageUserRating	The rating of the application.	number	/	1.0, 1.5

Table 6.1: Selected features for IOS apps

6.1.2 Analysis and preparation of the data

The same processing steps as in section 5.1.2 have been used to prepare the data.

1. Deletion of the duplicated rows.
2. Management of the categorical features (contentAdvisoryRating, primaryGenreName) through one-hot encoding and creation of "other" categories.
3. Scaling of the numerical columns through standardization.
4. Creation of the rating classes using the same rules as in section 5.1.2.
5. Deletion of the rows with missing values.
6. Filtering of the apps, keeping only applications with at least 50 individual ratings.

The final dataset had 41856 rows and 31 columns. Again, there was no app with a rating of 0 star. Figure 6.1 and table 6.2 give a description of the dataset.

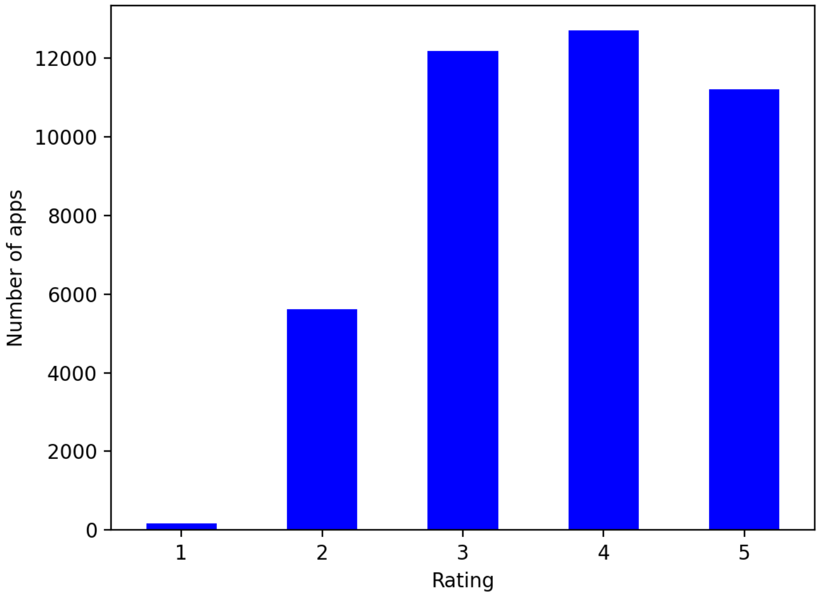


Figure 6.1: Distribution of the ratings for IOS apps

Metric	**Value**
Number of rows	41856
Number of columns	31
Number of free apps	37912
Number of apps available on iPhone	40650
Number of apps available on iPad	41697
Number of apps supporting VPP distribution	41849
Number of apps supporting Game Center	36545
Number of categories	15
Most represented categories	Games (13298 apps) Other (5086 apps) Entertainment (3807 apps)
Number of content ratings	5
Most represented content ratings	4+ (28476 apps) 12+ (5715 apps) 17+ (5037 apps)
Mean length of the description	1384 characters
Mean length of the title	20 characters
Mean size	145.7 megabytes
Mean number of iPhone screenshots	5
Mean number of iPad screenshots	1

Table 6.2: Summary of the IOS dataset

6.1.3 Reduction of dimensionality

in order to reduce the number of features, we also used PCA and Random Forest (see section 5.1.3). A 95% PCA gave 15 principal components. Figure 6.2 represents the 10 features with a relative importance larger than 0.01 for the Random Forest.

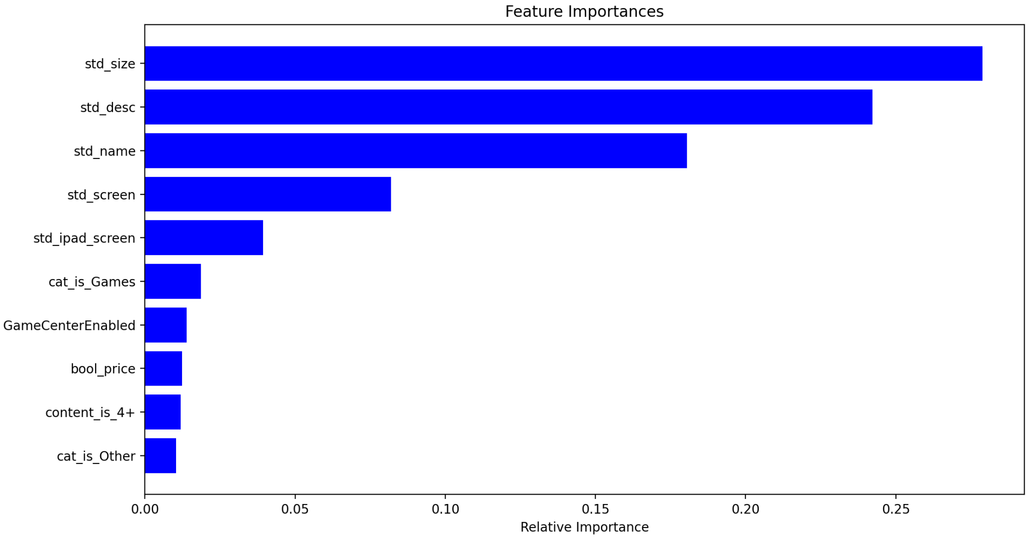


Figure 6.2: The most important features for Android apps

We can see that 3 features are significantly more important than the others: the size, the length of the description and the length of the name. The number of screenshots is also rather important.

6.2 Training of the models

Table 6.3 summarizes the training of the different models for IOS apps.

Model	Used function	Selected hyperparameters and final values	Training accuracy	Validation accuracy	Testing accuracy
Random	/	/	20%	20%	20%
Decision Tree	tree.DecisionTreeClassifier()	- maximal depth = 10 - maximal number of leaf nodes = 200	49.1%	45.8%	46.4%
Random Forest	RandomForestClassifier()	- maximal depth of the trees = 15 - number of estimators = 125	72%	48.1%	49.6%
KNN	KNeighborsClassifier()	- number of neighbors = 30	49%	43.7%	45.2% 45.2%[1] 45.6%[2]
SVM	svm.SVC()	- Regularization parameter (C) = 2	48.7%	45.35%[2]	47.3% 47%[1] 47.5%[2]
MLP	MLPClassifier()	- 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

Table 6.3: Training of the models for IOS apps

The methodology presented in section 4.3 has also been used to get these results. The confusion matrices corresponding to the testing results may be found in appendix B. Table 6.3 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. The confusion matrices show that the models neglect the rating 1 star most of the time due to the small amount of data for this rating. The accuracy of the models when predicting the rating 2 stars is also significantly smaller than for the other ratings. The reason could be the smaller amount of samples for this rating.

Chapter 7

Discussion

7.1 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. 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. These are all variables that the customer will encounter before using the application. This means that the rating of an app is influenced by elements available before the use of this application. Looking at the Pearson correlation coefficients between these variables and the rating (see appendix C), the number of screenshots, the size and the length of the description are significantly and positively correlated with the rating for both stores (between 0.13 and 0.35 for the correlation coefficients).

7.2 Recommendations to apps designers

We think that a system of rating prediction for applications including the publication strategy would enhance the ability of apps designers to produce highly rated apps. Indeed, this would allow them to test several configurations of features and select the one that has the highest probability to get a high number of stars. Based on our results, such a system is conceivable. Indeed, we achieved an accuracy of 50% using only basic variables. We think that this score could be enhanced by increasing the number of variables taken into account.

We also identified different variables that have an impact on the rating. First of all, an higher number of screenshots helps to increase the rating of an application. Then, the rating is likely to be higher when the description is long. Finally, bigger apps (in terms of size) tend to have an higher rating.

7.3 Limitations and future works

First of all, the number of features extracted from the raw data could be increased. 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 publication strategy while we know that the content of the app will play a big role in determining the rating. The intent was to look at the importance of the presentation of the app for the rating. Therefore, we did not take technical features or information about the content of the app into account (except the category and the content rating). It would now be interesting to analyze the impact of the publication strategy on the success (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 success for apps of the same quality.

Finally, a low number of apps rated 1 or 5 stars for Android and 1 or 2 stars for IOS remained after data processing. This means that our models were not able to predict these classes well. It would be interesting to build a database containing an equal and sufficient amount of apps for each rating with at least 50 individual ratings for each application.

Chapter 8

Conclusion

In this work, we looked at the extent to which the rating of an application can be predicted using only its publication strategy. The goal was to determine the importance of this latter in the way a customer perceives an application. The following classification algorithms were used to predict the rating: decision tree, random forest, KNN, SVM, MLP. We used metadata about 96,178 Android apps and 81,000 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. 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. Some recommendations were made to apps designers and future works were discussed.

References

[1] Legal'Easy. (2019). *Les chiffres des utilisateurs d'applications*. Consulted the 08/11/2019 at `https://www.my-business-plan.fr/chiffres-application`

[2] Lucy Handley. (2019). *Nearly three quarters of the world will use just their smartphones to access the internet by 2025*. Consulted the 08/11/2019 at `https://www.cnbc.com/2019/01/24/smartphones-72percent-of-people-will-use-only-mobile-for-internet-by-2025.html`

[3] statista. (2019). *Mobile app usage - Statistics & Facts*. Consulted the 08/11/2019 at `https://www.statista.com/topics/1002/mobile-app-usage/`

[4] Apple. (2020). *Ratings, Reviews, and Responses*. Consulted the 12/05/2020 at `https://developer.apple.com/app-store/ratings-and-reviews/`

[5] Colgan, M. (2019). *HOW IMPORTANT ARE MOBILE APP RATINGS & REVIEWS?*. Consulted the 12/05/2020 at `https://tapadoo.com/mobile-app-ratings-reviews/`

[6] Gordon, G. (2018). *User Ratings & Reviews: How They Impact ASO – Ultimate Guide*. Consulted the 12/05/2020 at `https://thetool.io/2018/user-ratings-reviews-aso-guide#How_does_User_Feedback_Impact_ASO`

[7] Yang, H. (2013). Bon Appétit for Apps: Young American Consumers' Acceptance of Mobile Applications. *Journal of Computer Information Systems*, pp. 85-96.

[8] Van der Heijden, H. (2004). User acceptance of hedonic information systems. *MIS quarterly*, pp. 695-704.

[9] Kim, S. C., Yoon, D., & Han, E. K. (2014). Antecedents of mobile app usage among smartphone users. *Journal of Marketing Communications*, pp. 653-670.

[10] Lee, G., & Raghu, T. S. (2014). Determinants of Mobile Apps Success: Evidence from App Store Market. *Journal of Management Information Systems*, pp.

133-170.

[11] Picoto, W. N., Duarte, R., & Pinto, I. (2019). Uncovering top-ranking factors for mobile apps through a multimethod approach. *Journal of Business Research*, pp. 668-674.

[12] Lu, J., Liu, C., & Wei, J. (2016). How Important Are Enjoyment and Mobility for Mobile Applications? *Journal of Computer Information Systems*, pp. 1-12.

[13] Monett, D., & Stolte, H. (2016). Predicting Star Ratings based on Annotated Reviews of Mobile Apps. *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 421-428.

[14] Meng, J., Zheng, Z., Tao, G., & Liu, X. (2016). User-Specific Rating Prediction for Mobile Applications via Weight-based Matrix Factorization. *2016 IEEE International Conference on Web Services (ICWS)*, pp. 728-731.

[15] Daimi, K., Hazzazi, N. (2019). Using Apple Store Dataset to Predict User Rating of Mobile Applications. *2019 International Conference on Data Science*, pp. 28-33.

[16] Sarro, F., Harman, M., Jia, Y., Zhang, Y. (2018). Customer Rating Reactions Can Be Predicted Purely Using App Features. *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pp. 76-87.

[17] Shobha, G., Rangaswamy, S. (2018). *Handbook of Statistics : Volume 38.* Elsevier.

[18] Talabis, M. R. M., McPherson, R., Miyamoto, I., Martin, J. L., Kaye, D. (2014). *Information Security Analytics : Finding Security Insights, Patterns and Anomalies in Big Data.* Elsevier.

[19] Data Analytics Post. (2020). *OVERFITTING.* Consulted the 14/05/2020 at `https://dataanalyticspost.com/Lexique/overfitting/`

[20] Elite Data Science. (2020). *Overfitting in Machine Learning: What It Is and How to Prevent It.* Consulted the 14/05/2020 at `https://elitedatascience.com/overfitting-in-machine-learning#how-to-prevent`

[21] Gupta, P. (2017). *Decision Trees in Machine Learning.* Consulted the 14/05/2020 at `https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052`

[22] scikit-learn. (2019). *Decision Trees.* Consulted the 14/05/2020 at `https://scikit-learn.org/stable/modules/tree.html`

[23] Data Analytics Post. (2020). *RANDOM FOREST*. Consulted the 14/05/2020 at `https://dataanalyticspost.com/Lexique/random-forest/`

[24] Chakure, A. (2019). *Random Forest Classification.* Consulted the 14/05/2020 at `https://towardsdatascience.com/random-forest-classification-and-its-implementation-d5d840dbead0`

[25] scikit-learn. (2019). *Ensemble methods.* Consulted the 14/05/2020 at `https://scikit-learn.org/stable/modules/ensemble.html`

[26] Harrison, O. (2018). *Machine Learning Basics with the K-Nearest Neighbors Algorithm.* Consulted the 14/05/2020 at `https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761`

[27] scikit-learn. (2019). *Nearest Neighbors.* Consulted the 14/05/2020 at `https://scikit-learn.org/stable/modules/neighbors.html`

[28] Yadav, A. (2018). *SUPPORT VECTOR MACHINES(SVM).* Consulted the 15/05/2020 at `https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589`

[29] scikit-learn. (2019). *Support Vector Machines.* Consulted the 15/05/2020 at `https://scikit-learn.org/stable/modules/svm.html`

[30] Brabazon, A., O'Neill, M., McGarraghy, S. (2015). *Natural Computing Algorithms.* Leiden : Springer.

[31] scikit-learn. (2019). *Neural network models (supervised).* Consulted the 15/05/2020 at `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`

[32] Brownlee, J. (2019). *When to Use MLP, CNN, and RNN Neural Networks.* Consulted the 15/05/2020 at `https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/`

[33] 42matters. (2020). *Programmatic Access to Mobile Data.* Consulted the 18/05/2020 at `https://42matters.com/app-market-data`

[34] 42matters. (2020). *Advanced Query API for Android Apps.* Consulted the 18/05/2020 at `https://42matters.com/docs/app-market-data/android/apps/advanced-query-api`

[35] 42matters. (2020). *Advanced Query API for iOS Apps.* Consulted the 18/05/2020 at `https://42matters.com/docs/app-market-data/ios/apps/advanced-query-api`

[36] Agarwal, R. (2019). *The 5 Feature Selection Algorithms every Data Scientist*

should know. Consulted the 19/05/2020 at (https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2

[37] Elite Data Science. (2019). *Dimensionality Reduction Algorithms: Strengths and Weaknesses.* Consulted the 19/05/2020 at https://elitedatascience.com/dimensionality-reduction-algorithms#feature-selection

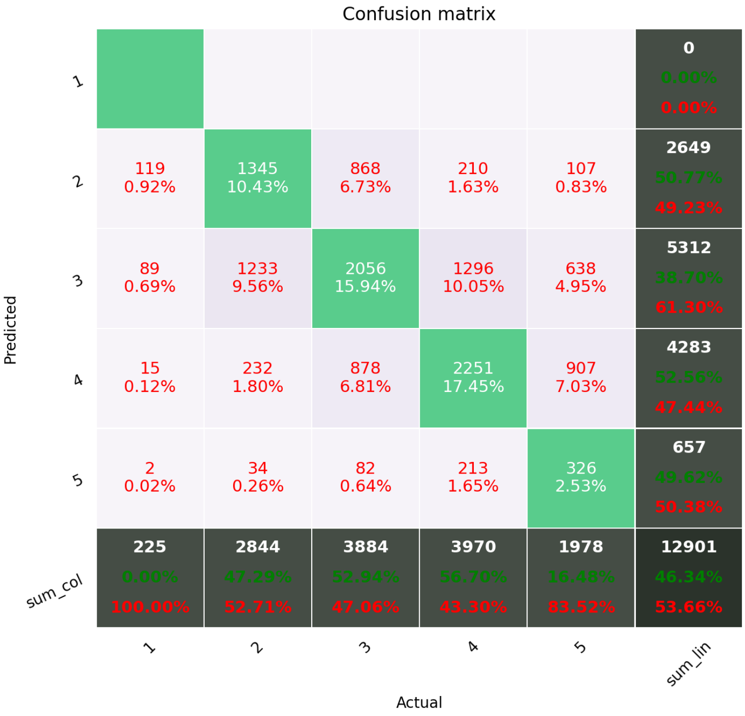
[38] scikit-learn. (2019). *Choosing the right estimator.* Consulted the 19/05/2020 at https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

[39] 42matters. (2020). *Android App Object.* Consulted the 20/05/2020 at https://42matters.com/docs/app-market-data/android/apps/object

[40] Rençberoğlu, E. (2019). *Fundamental Techniques of Feature Engineering for Machine Learning.* Consulted the 20/05/2020 at https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114

Appendix A

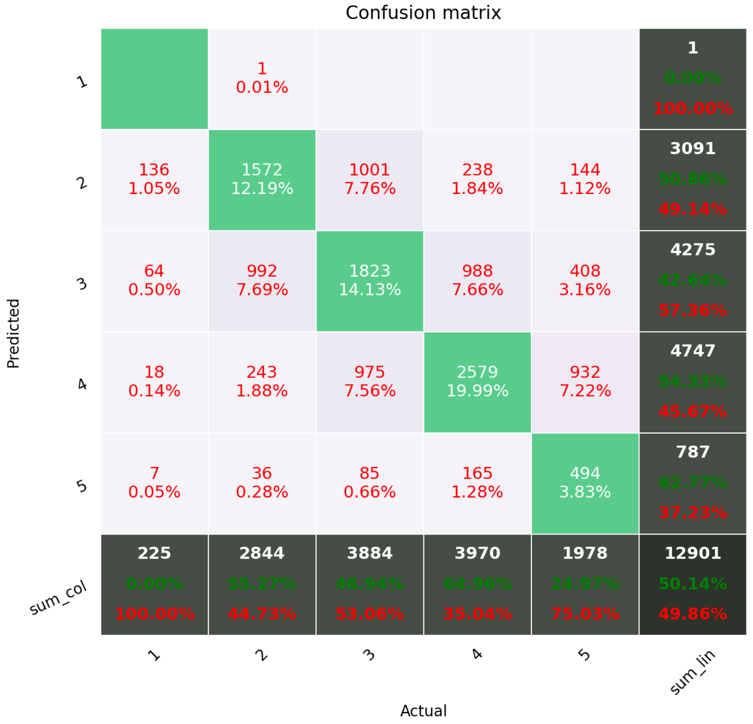
Confusion matrices for Android apps



Confusion matrix

Predicted \ Actual	1	2	3	4	5	sum_lin
1						0 0.00% 0.00%
2	119 0.92%	1345 10.43%	868 6.73%	210 1.63%	107 0.83%	2649 50.77% 49.23%
3	89 0.69%	1233 9.56%	2056 15.94%	1296 10.05%	638 4.95%	5312 38.70% 61.30%
4	15 0.12%	232 1.80%	878 6.81%	2251 17.45%	907 7.03%	4283 52.56% 47.44%
5	2 0.02%	34 0.26%	82 0.64%	213 1.65%	326 2.53%	657 49.62% 50.38%
sum_col	225 0.00% 100.00%	2844 47.29% 52.71%	3884 52.94% 47.06%	3970 56.70% 43.30%	1978 16.48% 83.52%	12901 46.34% 53.66%

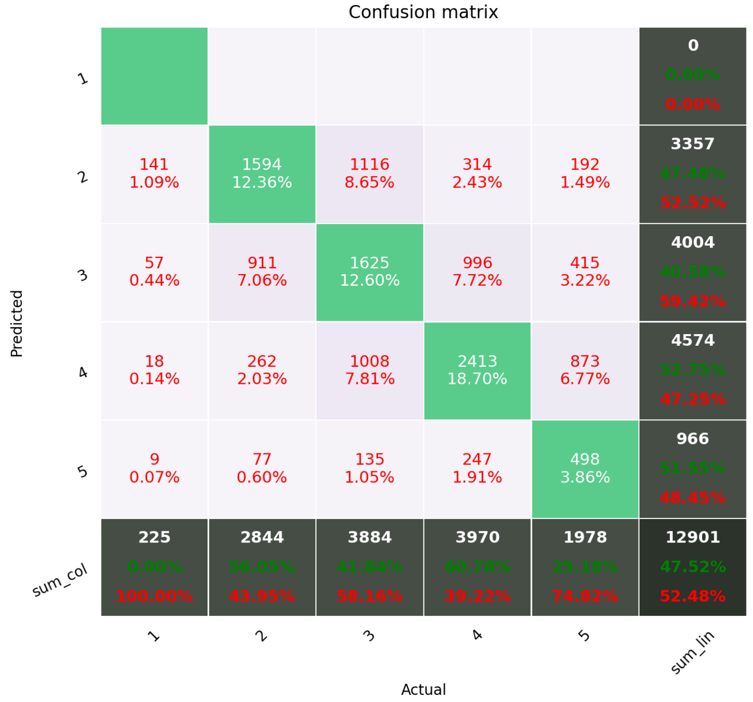
Figure A.1: Decision Tree



Confusion matrix

Predicted \ Actual	1	2	3	4	5	sum_lin
1		1 0.01%				1 0.00% 100.00%
2	136 1.05%	1572 12.19%	1001 7.76%	238 1.84%	144 1.12%	3091 50.86% 49.14%
3	64 0.50%	992 7.69%	1823 14.13%	988 7.66%	408 3.16%	4275 42.64% 57.36%
4	18 0.14%	243 1.88%	975 7.56%	2579 19.99%	932 7.22%	4747 54.33% 45.67%
5	7 0.05%	36 0.28%	85 0.66%	165 1.28%	494 3.83%	787 62.77% 37.23%
sum_col	225 0.00% 100.00%	2844 55.27% 44.73%	3884 46.94% 53.06%	3970 64.96% 35.04%	1978 24.97% 75.03%	12901 50.14% 49.86%

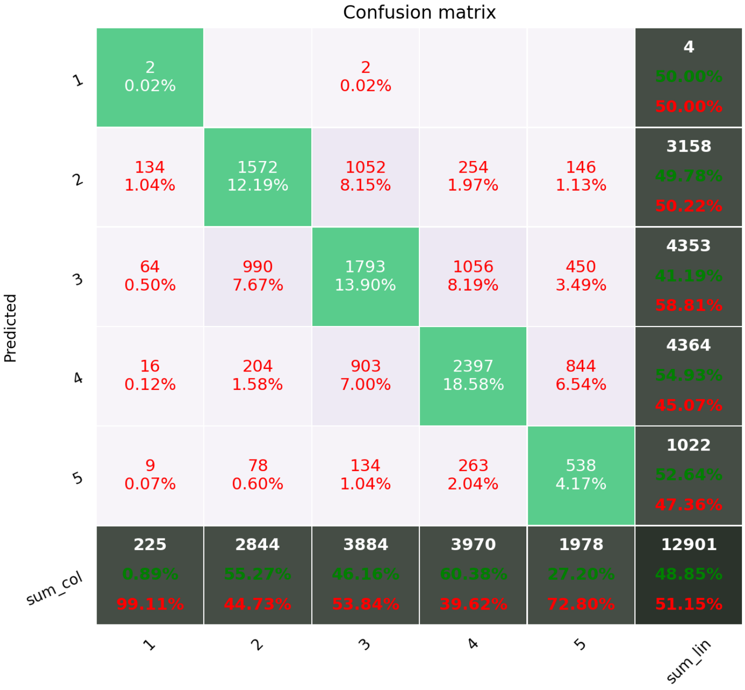
Figure A.2: Random Forest



Confusion matrix

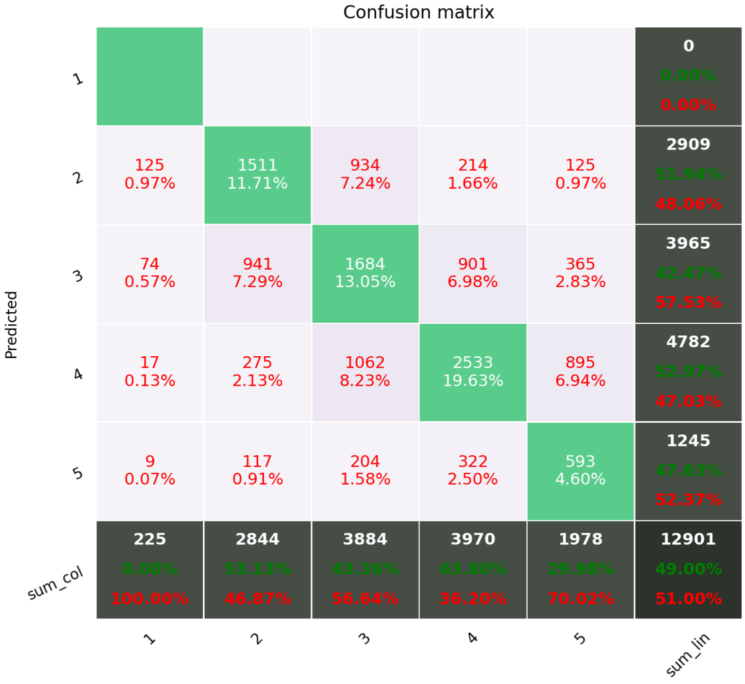
Predicted \ Actual	1	2	3	4	5	sum_lin
1						0 0.00% 0.00%
2	141 1.09%	1594 12.36%	1116 8.65%	314 2.43%	192 1.49%	3357 47.48% 52.52%
3	57 0.44%	911 7.06%	1625 12.60%	996 7.72%	415 3.22%	4004 40.58% 59.42%
4	18 0.14%	262 2.03%	1008 7.81%	2413 18.70%	873 6.77%	4574 52.75% 47.25%
5	9 0.07%	77 0.60%	135 1.05%	247 1.91%	498 3.86%	966 51.55% 48.45%
sum_col	225 0.00% 100.00%	2844 56.05% 43.95%	3884 41.84% 58.16%	3970 60.78% 39.22%	1978 25.18% 74.82%	12901 47.52% 52.48%

Figure A.3: KNN



Predicted \ Actual	1	2	3	4	5	sum_lin
1	2 0.02%		2 0.02%			4 50.00% 50.00%
2	134 1.04%	1572 12.19%	1052 8.15%	254 1.97%	146 1.13%	3158 49.78% 50.22%
3	64 0.50%	990 7.67%	1793 13.90%	1056 8.19%	450 3.49%	4353 41.19% 58.81%
4	16 0.12%	204 1.58%	903 7.00%	2397 18.58%	844 6.54%	4364 54.93% 45.07%
5	9 0.07%	78 0.60%	134 1.04%	263 2.04%	538 4.17%	1022 52.64% 47.36%
sum_col	225 0.89% 99.11%	2844 55.27% 44.73%	3884 46.16% 53.84%	3970 60.38% 39.62%	1978 27.20% 72.80%	12901 48.85% 51.15%

Figure A.4: SVM

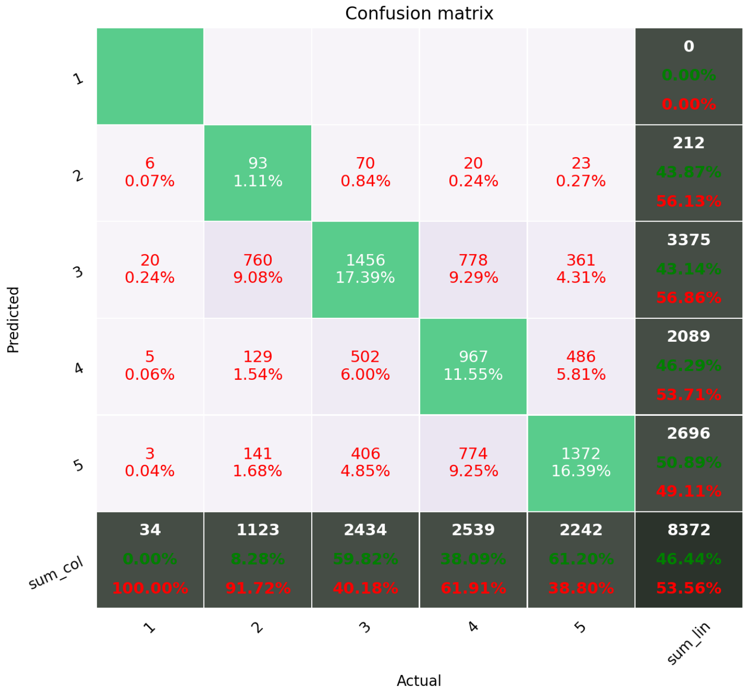


Predicted \ Actual	1	2	3	4	5	sum_lin
1						0 0.00% 0.00%
2	125 0.97%	1511 11.71%	934 7.24%	214 1.66%	125 0.97%	2909 51.94% 48.06%
3	74 0.57%	941 7.29%	1684 13.05%	901 6.98%	365 2.83%	3965 42.47% 57.53%
4	17 0.13%	275 2.13%	1062 8.23%	2533 19.63%	895 6.94%	4782 52.97% 47.03%
5	9 0.07%	117 0.91%	204 1.58%	322 2.50%	593 4.60%	1245 47.63% 52.37%
sum_col	225 0.00% 100.00%	2844 53.13% 46.87%	3884 43.36% 56.64%	3970 63.80% 36.20%	1978 29.98% 70.02%	12901 49.00% 51.00%

Figure A.5: MLP

Appendix B

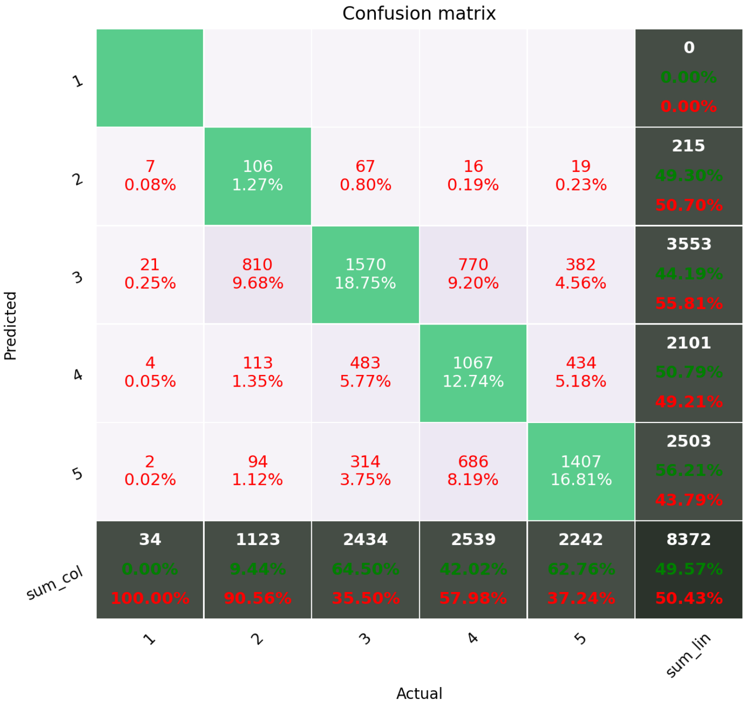
Confusion matrices for IOS apps



Confusion matrix

	1	2	3	4	5	sum_lin
1						0 0.00% 0.00%
2	6 0.07%	93 1.11%	70 0.84%	20 0.24%	23 0.27%	212 43.87% 56.13%
3	20 0.24%	760 9.08%	1456 17.39%	778 9.29%	361 4.31%	3375 43.14% 56.86%
4	5 0.06%	129 1.54%	502 6.00%	967 11.55%	486 5.81%	2089 46.29% 53.71%
5	3 0.04%	141 1.68%	406 4.85%	774 9.25%	1372 16.39%	2696 50.89% 49.11%
sum_col	34 0.00% 100.00%	1123 8.28% 91.72%	2434 59.82% 40.18%	2539 38.09% 61.91%	2242 61.20% 38.80%	8372 46.44% 53.56%

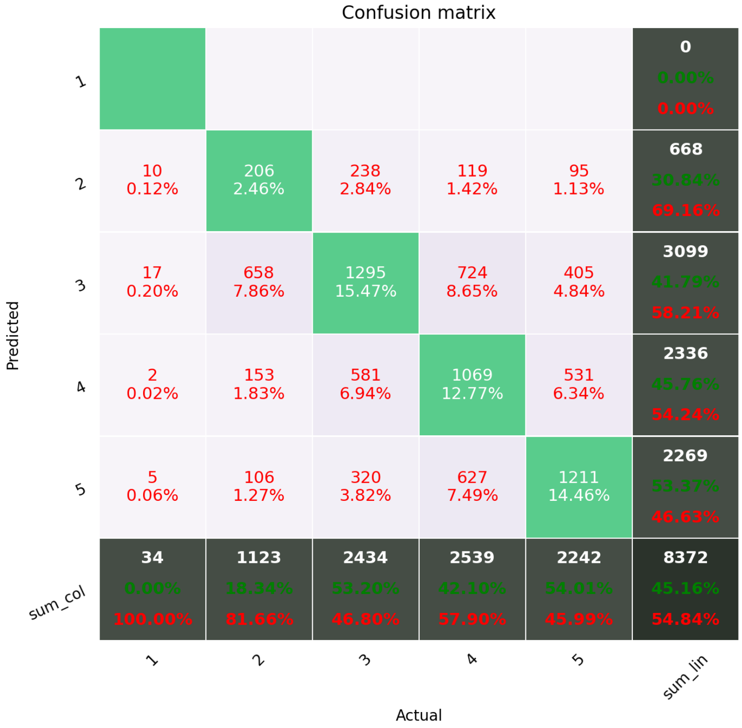
Figure B.1: Decision Tree



Confusion matrix

Predicted \ Actual	1	2	3	4	5	sum_lin
1						0 0.00% 0.00%
2	7 0.08%	106 1.27%	67 0.80%	16 0.19%	19 0.23%	215 49.30% 50.70%
3	21 0.25%	810 9.68%	1570 18.75%	770 9.20%	382 4.56%	3553 44.19% 55.81%
4	4 0.05%	113 1.35%	483 5.77%	1067 12.74%	434 5.18%	2101 50.79% 49.21%
5	2 0.02%	94 1.12%	314 3.75%	686 8.19%	1407 16.81%	2503 56.21% 43.79%
sum_col	34 0.00% 100.00%	1123 9.44% 90.56%	2434 64.50% 35.50%	2539 42.02% 57.98%	2242 62.76% 37.24%	8372 49.57% 50.43%

Figure B.2: Random Forest



Confusion matrix

Predicted \ Actual	1	2	3	4	5	sum_lin
1						0 0.00% 0.00%
2	10 0.12%	206 2.46%	238 2.84%	119 1.42%	95 1.13%	668 30.84% 69.16%
3	17 0.20%	658 7.86%	1295 15.47%	724 8.65%	405 4.84%	3099 41.79% 58.21%
4	2 0.02%	153 1.83%	581 6.94%	1069 12.77%	531 6.34%	2336 45.76% 54.24%
5	5 0.06%	106 1.27%	320 3.82%	627 7.49%	1211 14.46%	2269 53.37% 46.63%
sum_col	34 0.00% 100.00%	1123 18.34% 81.66%	2434 53.20% 46.80%	2539 42.10% 57.90%	2242 54.01% 45.99%	8372 45.16% 54.84%

Figure B.3: KNN

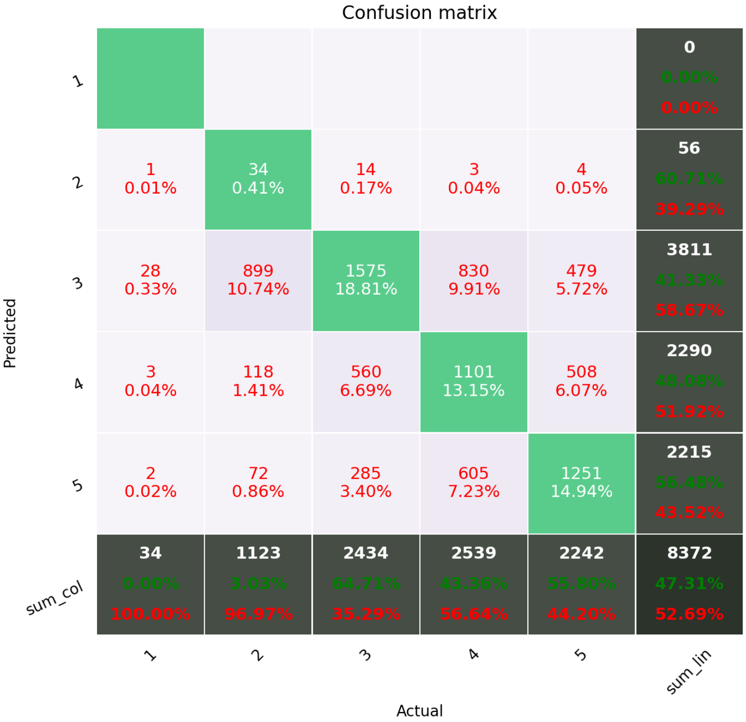


Figure B.4: SVM

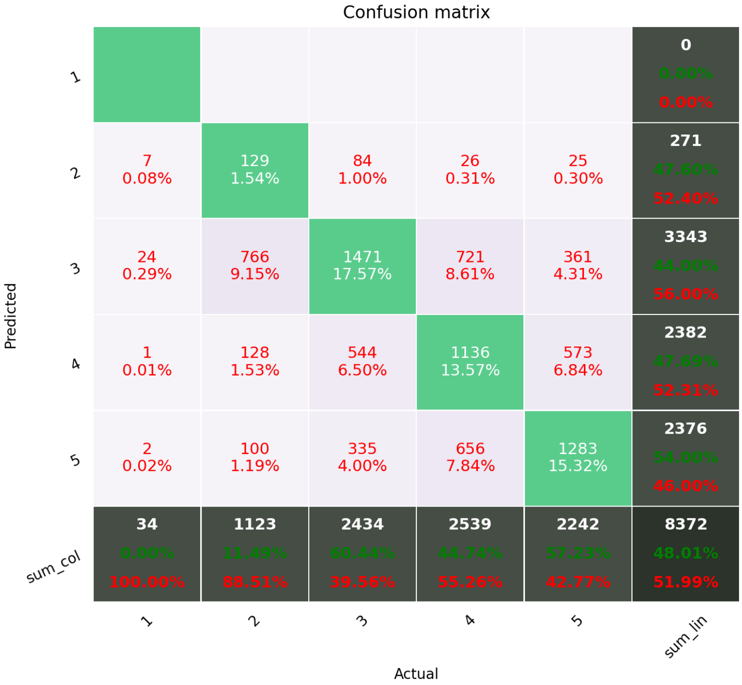


Figure B.5: MLP

Appendix C

Correlation between the most important variables and the rating

	Rating	**Number of screenshots**	**Size**	**Length of description**	**Length of the title**
Rating	1	0.35	0.13	0.27	0.25
Number of screenshots	0.35	1	0.21	0.24	0.14
Size	0.13	0.21	1	0.096	0.044
Length of description	0.27	0.24	0.096	1	0.41
Length of the title	0.25	0.14	0.044	0.41	1

All the coefficients are significant at a 0.01 level

Table C.1: Pearson correlation coefficients for Android

	Rating	**Number of screenshots**	**Size**	**Length of description**	**Length of the title**
Rating	1	0.24	0.19	0.22	0.081
Number of screenshots	0.24	1	0.15	0.25	-0.03
Size	0.19	0.15	1	0.064	-0.077
Length of description	0.22	0.25	0.064	1	0.17
Length of the title	0.081	-0.03	-0.077	0.17	1

All the coefficients are significant at a 0.01 level

Table C.2: Pearson correlation coefficients for IOS