Optimization in the Google Play Store

Owen Levinthal, Shangzhen Wu, Daniel Momeni, Jonathan Jiang

Owen Levinthal's Contribution:
-Drafted report
-Data cleaning
-Exploratory analysis
-KNN, Neural Network

Shangzhen Wu's Contribution:
-Methodology Description
-Logistic GAM Modeling

Daniel Momeni's Contribution:
-SVM, KNN, Neural Network, Random Forest, Regression
-Writing Results Section
-Organized Code

Jonathan Jiang's Contribution:
-Exploratory Analysis
-Final code cleanup
-Neural Network

Introduction

The Google Play Store contains roughly 3.4 million apps[1], which collectively generate north of 50 billion dollars as of 2021 [2]. With so much money up for grabs on the Google Play Store platform, it is essential that app developers understand what factors go into creating a successful app. This question starts with its preposition, what makes a successful app? Using data scrapped directly from the google play store [3], we have three quantifiable metrics by which to measure app success: number of app installs, number of reviews, and Play Store rating. Furthermore, our datasets allow for insights not just on the developer side, but on the platform side too. Since Google gets a cut of app revenue on their platform, they are incentivized to set up apps for success by properly categorizing apps, both in genre and in content rating.

This report is focused on optimizing the market from both ends; informing app creators as to what makes for a successful app, and improving the marketing of apps with the most accurate classification models. On the developer side, our results show that reviews, and installs are the most important factors in creating a successful app. On the platform side, the best models for classifying content rating, genre, other, are:

Methodology

For this report we are working with data scrapped directly from the Google Play store. This data is available from Kaggle, courtesy of Gurugram, Haryana [3]. The data comes in the form of 9660 unique observations, with 10 features for each observation. The features are as follows: App, Category, Rating, Reviews (# of reviews), Size, Installs (# of installs), Type (free or paid), Price, Content Rating, Genres, Last Update, Android Ver. Of these features, Category, Installs, Type, Content Rating, and Genres are categorical variables while the rest are quantitative. Installs have the interesting property of not being an exact number, but rather a series of ranges, making the data categorical rather than quantitative. Exploratory analysis can be found in the 'EDA' section of our supplementary material.

In order to analyze the most important factors for the success of an app, we fit a simple linear regression model to identify factors that significantly impact both app type and rating. This model assumes the presuppositions for linear regression, which are: a linear relationship among variables, multivariate normality, minimal collinearity, and homoscedasticity. Confidence intervals are generated for each predictor to get a sense for how accurately we predict the significance of a predictor to be.

The bulk of our report comes in the form of classification models, for which we have four: K-Nearest Neighbors, Random Forest, Support Vector Machine, and Neural Net. These models are used to classify the following for new data that contains all features except explicitly categorical features, and the ones being classified: Rating, Installs, Category, and Content Rating. The purpose of including these models is to determine the most accurate model for classification. Accurate classification of Rating and Installs allows for the prediction of app success, whereas accurate classification of Category and Content Rating allow for more appropriate placement within the app store. For each type of classification model, we chose to stay away from explicitly categorical input variables, as the four models mentioned have difficulty working with categorical variables.

Implementation Details

Our analysis is completed in Python, with the full code being included in the supplemental file to this report. We begin by reading in the data to a data frame and cleaning it for future analysis. This cleaning primarily involves converting columns to a single data type, either strings, integers, or floats. Dollar signs and commas are removed to make this possible. After converting each column to a workable data type, we perform exploratory data analysis to get a better sense of the data we are working with. This involves creating the graphs seen in the “EDA” section of supplementary details. While these results are not particularly important to the analysis, they served to guide our project. Following our exploratory analysis, we fit our four classification models to the following variables: Installs, Rating, Category, and Content Rating. Prior to fitting our classification models, the Rating feature was converted to a binary classification, with ratings above a 4 being “good” and ratings below a 4 being “bad.” This allowed us to predict Rating.

The first model we fit is the Knn algorithm, which effectively measures the euclidean distance between features. Since similar data has features that are “close” to each other, the Knn allows for classification based on similar data, with similarity being comparable to shorter distance between features. This model is easy to interpret, yet has difficulty at scale given the need to calculate distance between so many observations. Despite compiling quickly, this model lacked accuracy in classifying each of the aforementioned features.

The second model we looked at was the Random Forest model. We set the bins at 4 and below to be considered bad and 4 and up to be considered good. Even though ratings below 4 may not be bad, many users will look towards a “better” app at a higher rating. The results of this model were great for 2 of the 5 tests while 3 were lower but still good. The accuracy for each classification was fairly constant compared to the other models. Overall, Random Forest fared arguably the best than all the other machine learning models we chose.

Another model we looked at was the SVM model. We also kept the bins at 4 and below to be considered bad and 4 and up to be considered good. Even though a rating of slightly below 4 is not really bad but more like average we kept it like this because most of the apps had high ratings and this would help us more easily differentiate between an average app to a good app in SVM. The results of this model were fairly decent for 3 out of the 5 of our tests while 2 were subpar. SVM did overall fare better than most of the other machine learning models that we experimented with.

The last model we fit is the neural network algorithm. Similar to the other models, this one does not handle categorical variables as well, and as such we limited our predictors to Reviews, Size, Installs, and Rating (with only three predictors when one of these variables is the subject of classification). Implementation of this model relied heavily on trial and error, as we ran the algorithm for different numbers of hidden layers to determine which configuration would yield the highest classification rate. This model proved to be computationally expensive, especially as the number of hidden layers increased.

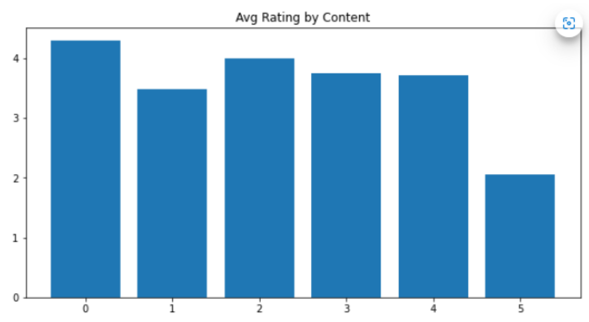
Results and Interpretation

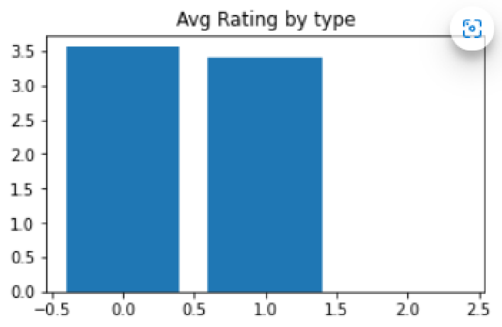
Looking at all of our models and their performance in classifying Rating, Installs, Content Rating, and Category, we believe that random forest is the best in regards to our data set. This was not a surprise, as a random forest is good at dealing with bins in data which we have constructed for categories such as Rating. Based on the results of random forest, logistic regression, and our confidence intervals we come to the conclusion that number of installs and reviews are the best predictors of an app's success. When one of these quantities is missing, the other can be used to get a sense of how popular an app will be, given their significance in predicting the classification of each other. The significance of these predictors, and the effectiveness of Random Forest and Support Vector Machine algorithms are the most important takeaways from this report. It is also notable that application size is an important factor in predicting the rating, and by extension the success of an application.

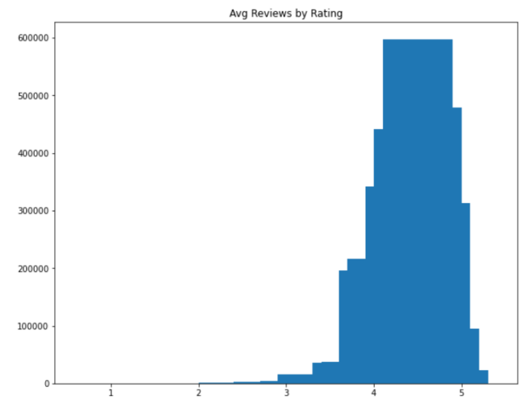
Methods and Accuracy	**General**	**Installs**	**Rating**	**Category**	**Content Rating**
KNN	**0.92**	**0.29**	**0.68**	**0.21**	**0.81**
RF	**0.68**	**0.67**	**0.67**	**0.76**	**0.79**
SVM	**0.59**	**0.59**	**0.70**	**0.76**	**0.81**
NN	**0.63**	**0.44**	**0.68**	**0.25**	**0.81**

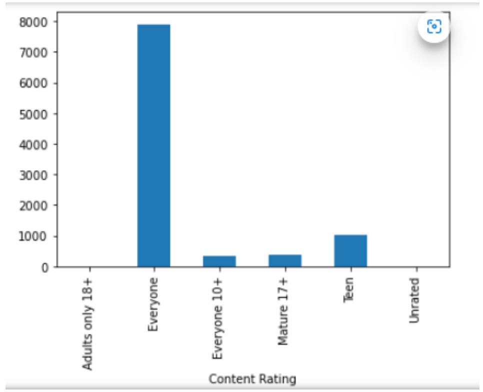
[Linear Regression] Rating ~ Installs/Reviews/Price/Size	Confidence Interval
Installs	(7777504.77206, 7777508.69248)
Reviews	(216590.60361, 216594.52403)
Price	(-0.86091, 3.05950)
Size	(18.38486, 22.30528)

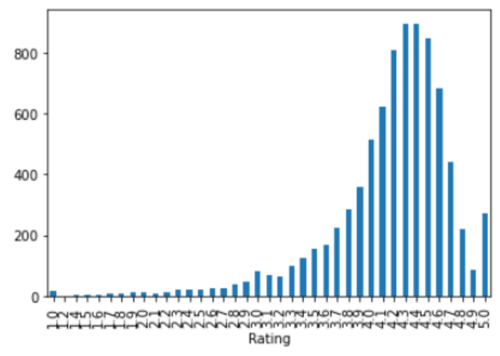
Appendix (Exploratory Data Analysis)

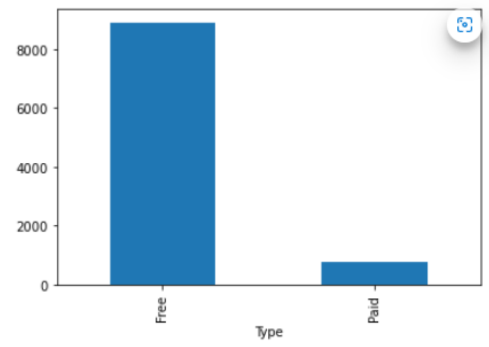


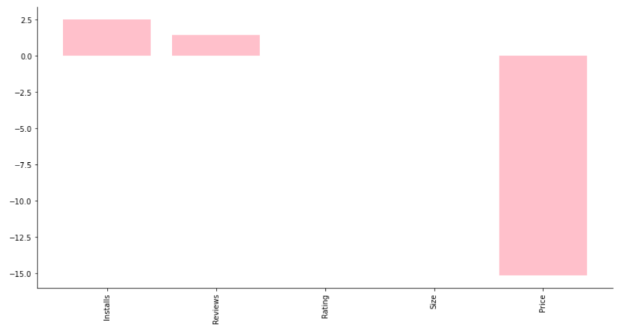


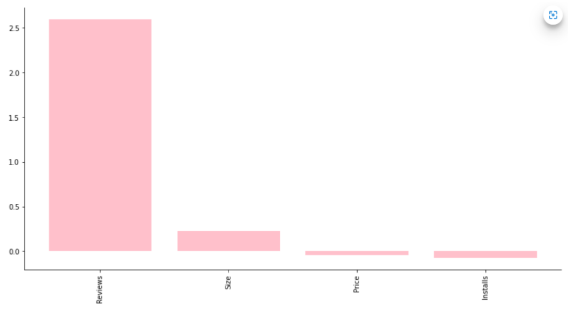












Citations

[1] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[2] https://www.businessofapps.com/data/app-revenues/

[3] https://www.kaggle.com/lava18/google-play-store-apps