Final Report: Statistical Modeling and Analysis Results for the Google Play Store Apps Data study

Submitted to:

Prof. Matthew S. Gerber

Department of Systems and Information Engineering, University of Virginia

Report Prepared By:

Shaoran Li
Fang You
Wenxi Zhao

December 11, 2018

Shaoran Li (sl4bz), Wenxi Zhao (wz8nx), Fang You (fy6vj)

Problem Description

While many public datasets online provide Apple App Store data, there are not many counterpart datasets available for Google Play Store Apps anywhere on the web. The Play Store Apps data has enormous potential to drive App-making businesses to success. Actionable insights can be drawn for developers to work on and capture the Android market.

Our project analyzes the Android app market using a Kaggle-hosted, web scraped dataset which describes 10k Google Play Store apps. We will do Exploratory Data Analysis, supplemented by some empirically competent statistical models to explore which features affect the rating of apps. We can also use our model to make predictions on app "Rating" given their features.

Objectives and Metrics

The objective of this project is to identify the relevant predictors that correlate to the "Rating" of the apps and to develop a model that could predict "Rating" while minimizing mean squared error and constraining variance. We are also studying patterns in various types of predictors and use these patterns to find correlations between predictors and response variable(i.e. "Rating"). There are two types of predictions that could be made based on our studies. One type of prediction is to forecast the potential popularity and ranking of apps in the near future based on existing metrics. Another type of prediction is to determine how app developers can adjust metrics available to them to maximize apps' probability of earning high rankings and popularity.

The metrics we can use to make such predictions are Apps' Category, Rating, Reviews, Size, (no. of) Installs, Type (paid or free), Price, Content Rating, Genre, Last Updated, Current Ver, Android Ver, Translated_Review, Sentiment, Sentiment_Polarity and Sentiment Subject. Among these metrics, there are only a few metrics such as Type, Price and those associated with version

that could be controlled by App developers. As such, the objectives of our second type of prediction would be limited since App developers can only manipulate a limited number of predictor metrics to affect their Apps' popularity. Nonetheless, we will be able to use these variables to construct multiple models and compare their effectiveness to help us make our first type of prediction.

Existing Methods

Datasets of App stores from Google play and Android market have been closely examined by various sources. In paper *Mining and Analysis of Apps in Google Play*, which will be referred to as *Mining*, both Google Play and Android Market App datasets have been discussed in depth. The first dataset is crawled from Google Play in November 2012 and accommodates 21,065 Apps from 24 categories. The second dataset is provided by Frank et al. [2012] crawled from Android Market (the older version of Google Play), they collected information of 450,933 Android Apps. Similarly in paper *Android Apps and User Feedback: A Dataset for Software Evolution and Quality Improvement*, which will be referred to as *Android*, a dataset containing 288,065 reviews extracted from Google Play related to 395 open source Apps mined from F-Droid2.

In *Mining*, two data analysis methods were used: correlation and clustering. Correlation analysis was useful for revealing intrinsic properties when it is applied to software repositories [Harman et al., 2012] and it depicts Google Play's position in the market to help developers to understand the market, customers' desire and their attitude. Clustering analysis was used to examine if applications placed in the same category are also functionally similar or whether App developers tend to develop Apps from the same category. In *Android*, they applied classification analysis on application's code to classify user reviews' feedback according to software maintenance and evolution categories, then computed software quality indicators for each version of the mined Apps.

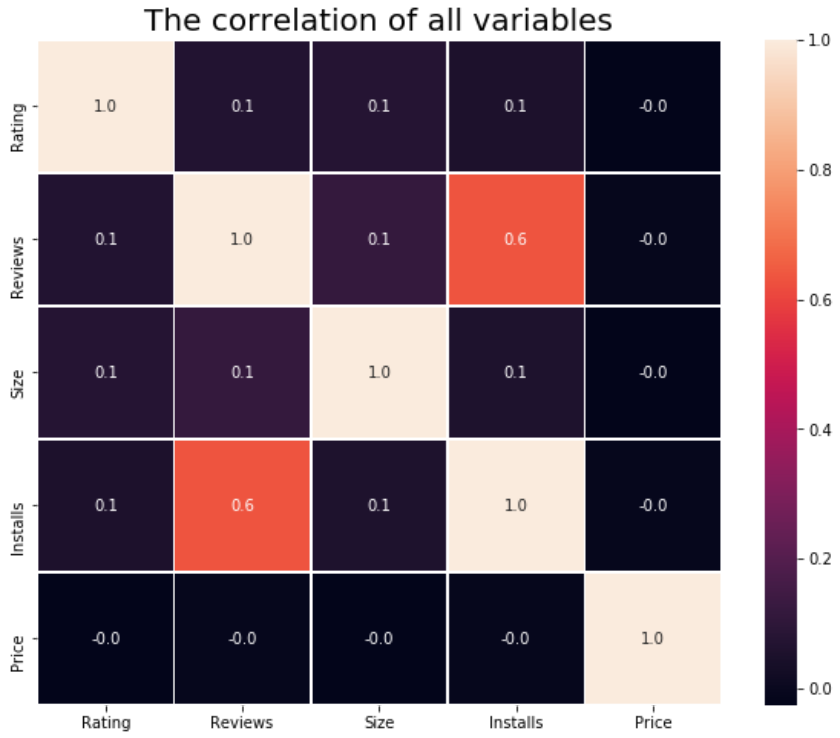
In *Mining*, the evaluation method used was to apply the same analysis on both datasets from Google Play and Android Market App. The authors observed quite similar correlation trend in certain metrics such as participation against price or participation against rating. However, they also detected minor differences in correlation coefficients for price against rating. To further evaluate this relationship, they had depicted graph of correlation measures for both datasets across different "Rating" and found they are quite similar. Hence they concluded that after expanding the smaller dataset to the larger dataset, the correlation analysis results would remain the same.

In both papers, data exploratory and analysis are conducted on Google Play's market position and customer feedback for developer's purpose. Our focus is to increase rankings and popularity to maximize profitability, by making predictions on existing metrics. Our focus was not addressed in previous related works, although analysis from previous work was conducted on similar metrics compared to ours to derive results that suffice different purposes.

Hypotheses from Exploratory Data Analysis

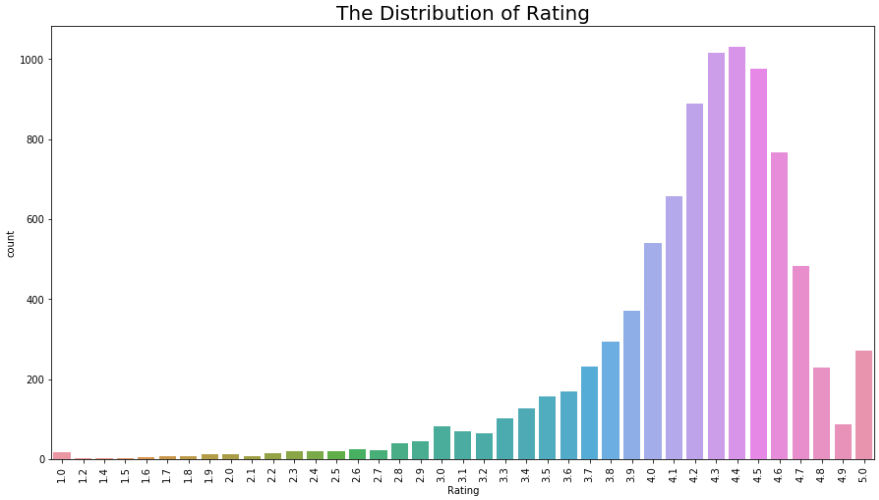
In order to propose hypothesis, we did exploratory data analysis for our data. Since our main goal is to build models to predict "Rating" for applications, our purpose for exploratory data analysis is essentially finding relationships among all variables and response variable "Rating" and proposing our hypothesis.

After preprocessing our data, we have 5 numerical variables, including our response "Rating", and 6 categorical variables. First we check the heatmap of correlation of those 5 numerical variables.



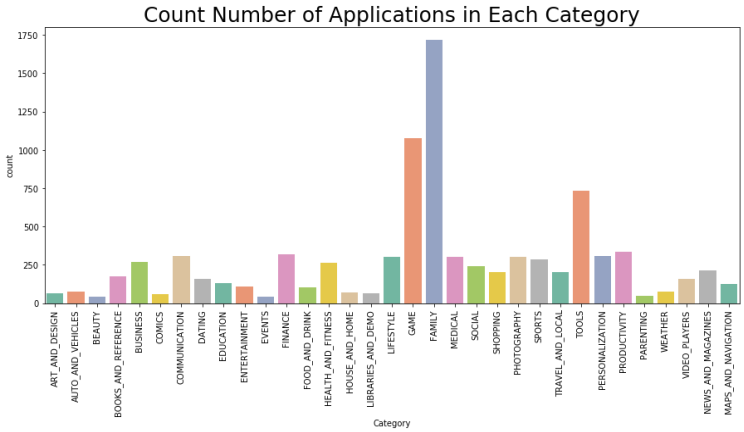
Based on the heatmap, we can see that "Rating" would be probably affected by "Reviews", "Size", "Installs", but not "Price". And "Reviews" and "Installs" are highly correlated which means an application received more reviews will be installed more.

Then we check the range of "Rating". By plotting the distribution of "Rating", we get the bell-curve distribution which shows that most of our applications get rated in the range of 4.0-4.7, a relatively high range.

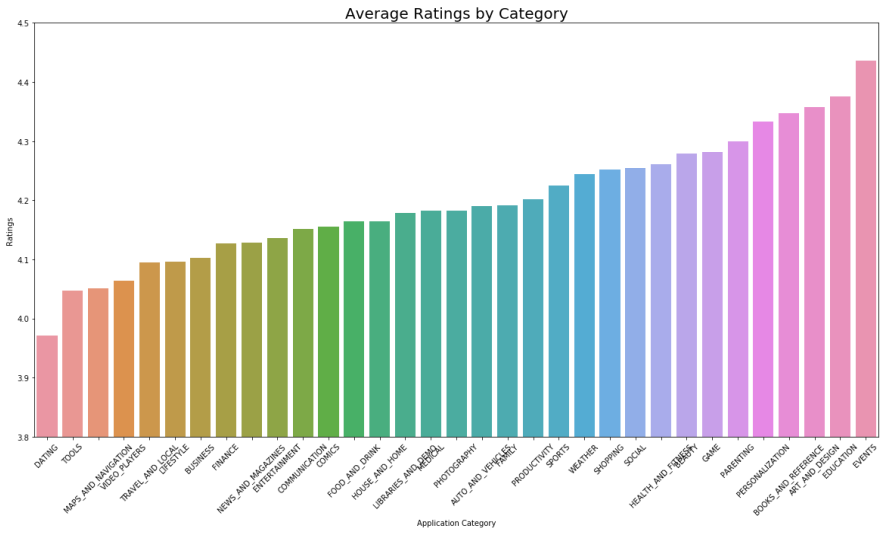


We also examine the relationship between “Rating” and each predictor variable.

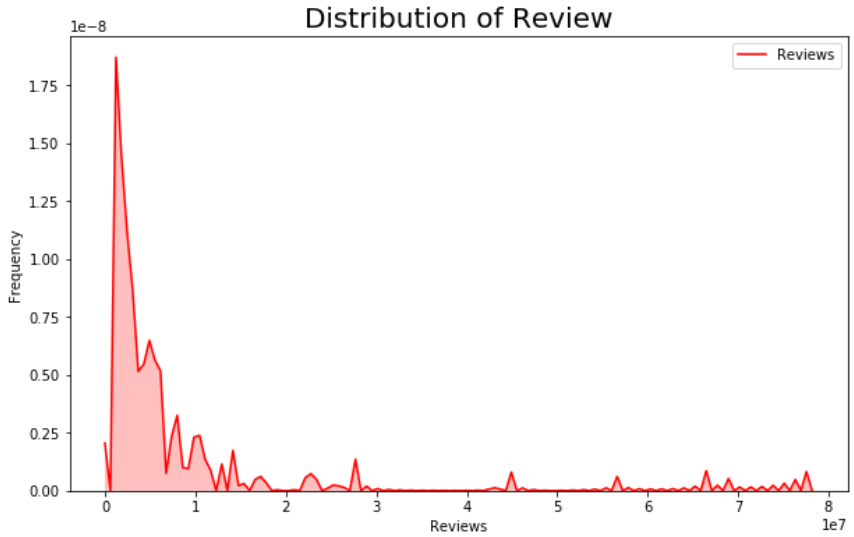
For the first predictor variable “Category”, we see that FAMILY has a largest number of applications, and then category GAME and TOOLS also have a lot of applications. This result is reasonable since people would like to use practical applications to facilitate their life and playing games in spare time. So category GAME and TOOLS would have a lot of applications since developers design applications based on popularity.



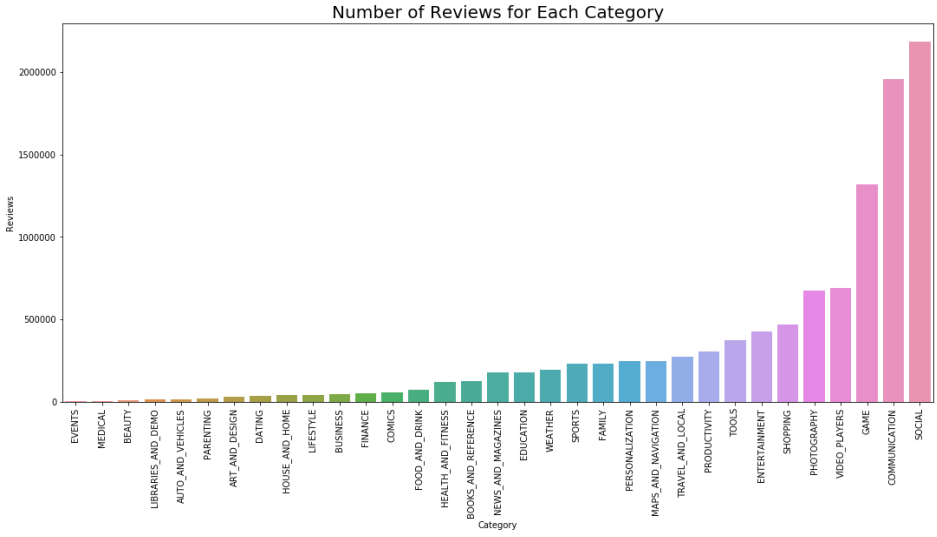
Grouping our “Category” by “Rating”, we can see that the category EVENTS received highest rating, and then category EDUCATION and ART_AND_DESIGN also got very high rating. The average rating for category GAME is also not bad. But for category TOOLS, although it has a lot of applications, the average rating is quite low. So developers should consider improve those applications categorized as TOOLS.



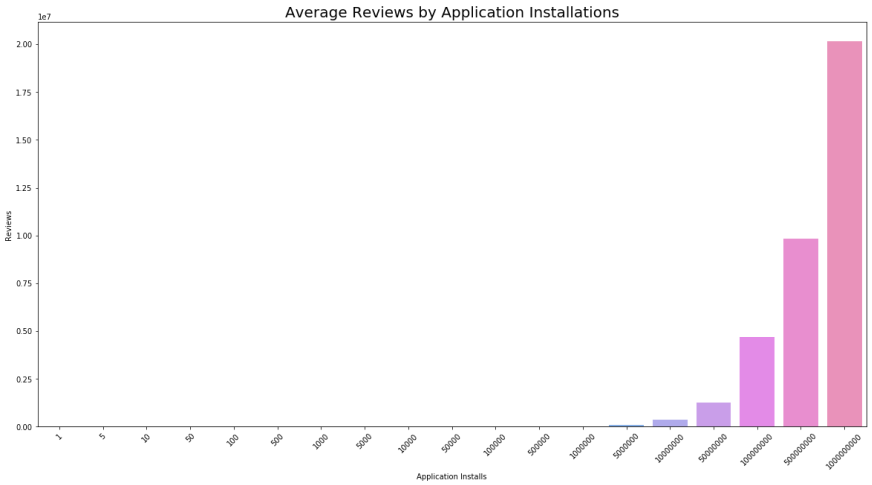
From the distribution plot of "Reviews", we see that most of applications in the store have less than 1M in reviews. And there are some applications got a lot of reviews, after examining the data, the four applications received most reviews are Facebook, Whats App, Instagram and Messenger, all of them are social network softwares.



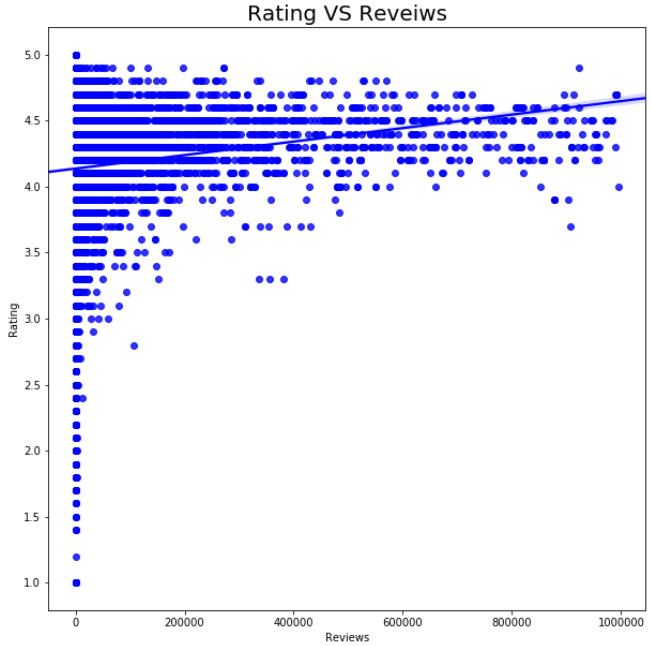
By counting the number of "Reviews" for each "Category", the SOCIAL, COMMUNICATION and GAME applications received a lot of reviews. This makes sense since those three are the most common applications that people used a lot in real life.



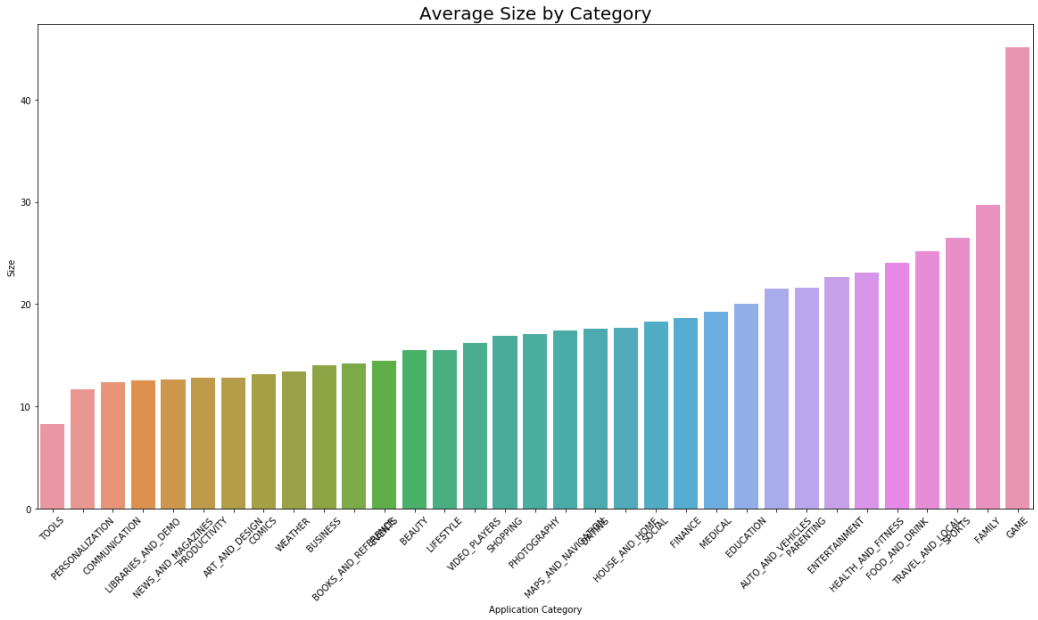
By counting the number of “Reviews” for “Installs”, we see that the most installed application got lots of reviews. There is an obvious positive relationship between “Installs” and “Reviews”, which is the same conclusion we got from our heatmap of correlation and this means that people would like to install applications with more reviews.



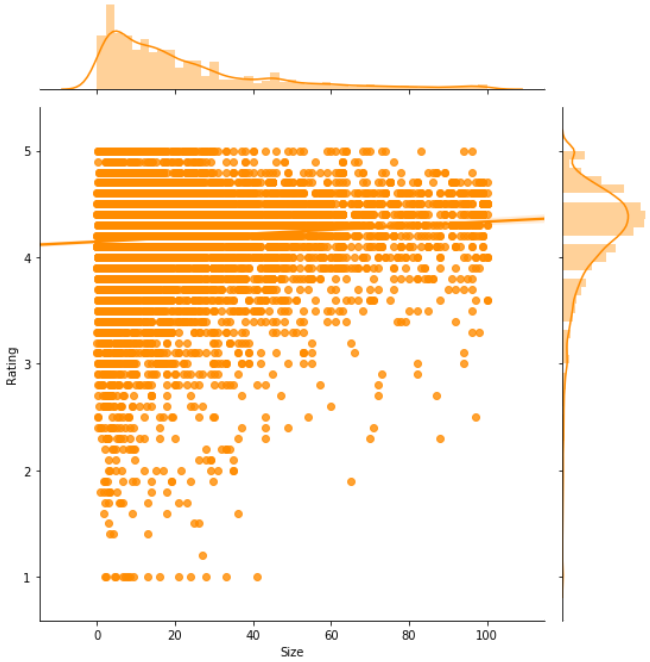
By checking the relationship between “Rating” and “Reviews”, we draw the regression plot which clearly shows that application received more reviews also get higher rating. To explain this phenomena, we know that reputation of an app is based on reviews.If an application has very low ratings, people won’t even install or use it, let alone write a review. So low rating applications won’t receive more reviews than others since they cannot attract new users.



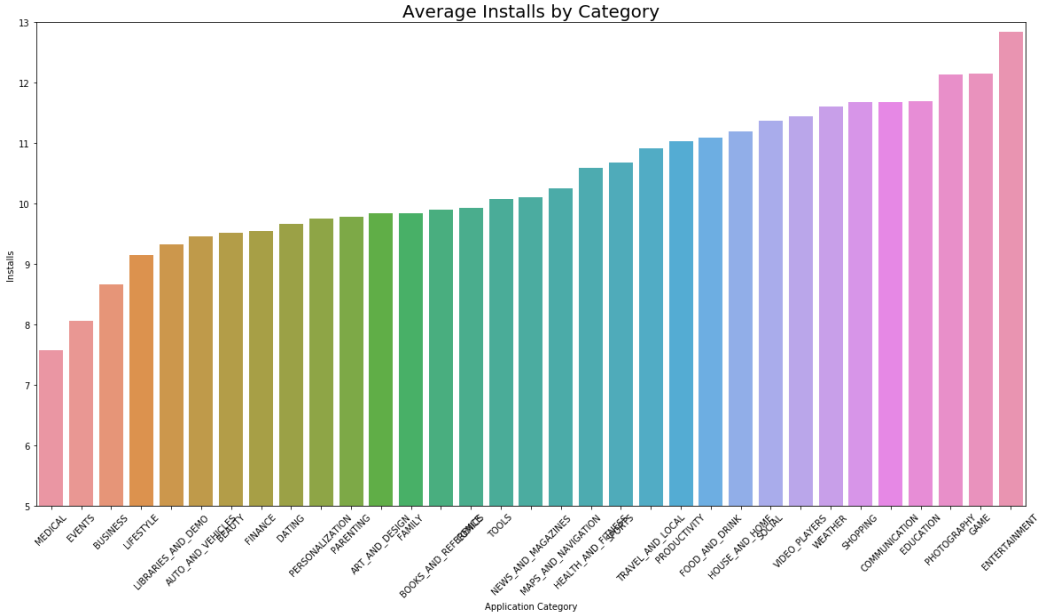
The third variable is “Size”. Grouping our application “Size” by “Category” first, we see that category GAME has the biggest size. Based on our previous plots, applications of GAME also received a lot of reviews and in general their average rating is good. We can see that the applications with smallest size are TOOLS. From previous plot we know that TOOLS also received lowest rating. So from here we can make a hypothesis that applications with larger size will receive better rating.



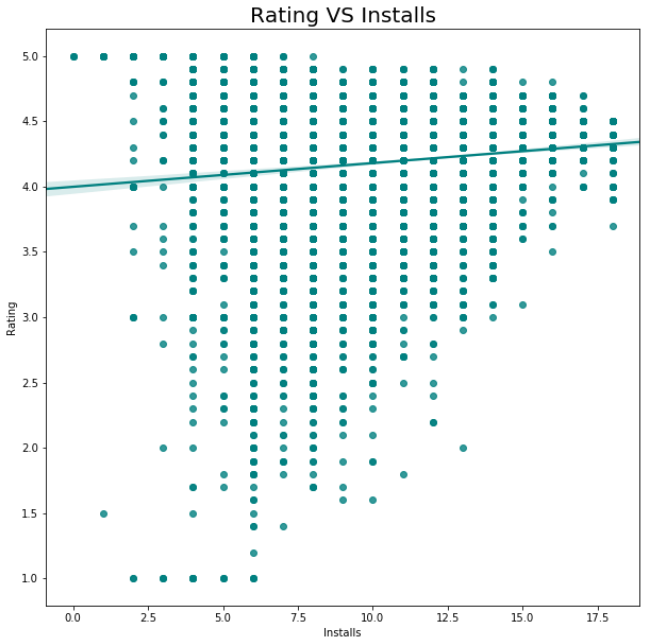
Based on the distribution and regression line for “Size” vs “Rating” below, the relationship between “Size” vs “Rating” is not obvious, but applications with larger size have slightly higher rating. Larger applications will generally have a lot of functions and in most cases, people would like at least one function and give great feedback based on the functions that are useful for them. However, for small applications, they might have fewer functions and if people don’t like those functions, small applications would not receive a high rating.



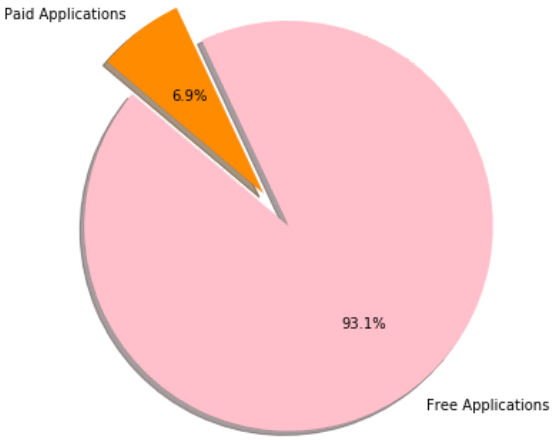
The fourth variable is “Installs”. By plotting the average number of installs for each category, we can see that category ENTERTAINMENT, GAME, PHOTOGRAPHY, and COMMUNICATION got most installations. As we mentioned before, people would like to use apps to play games, contact with friends and perform practical actions. It’s self-explanatory that those four categories of applications got most installations.



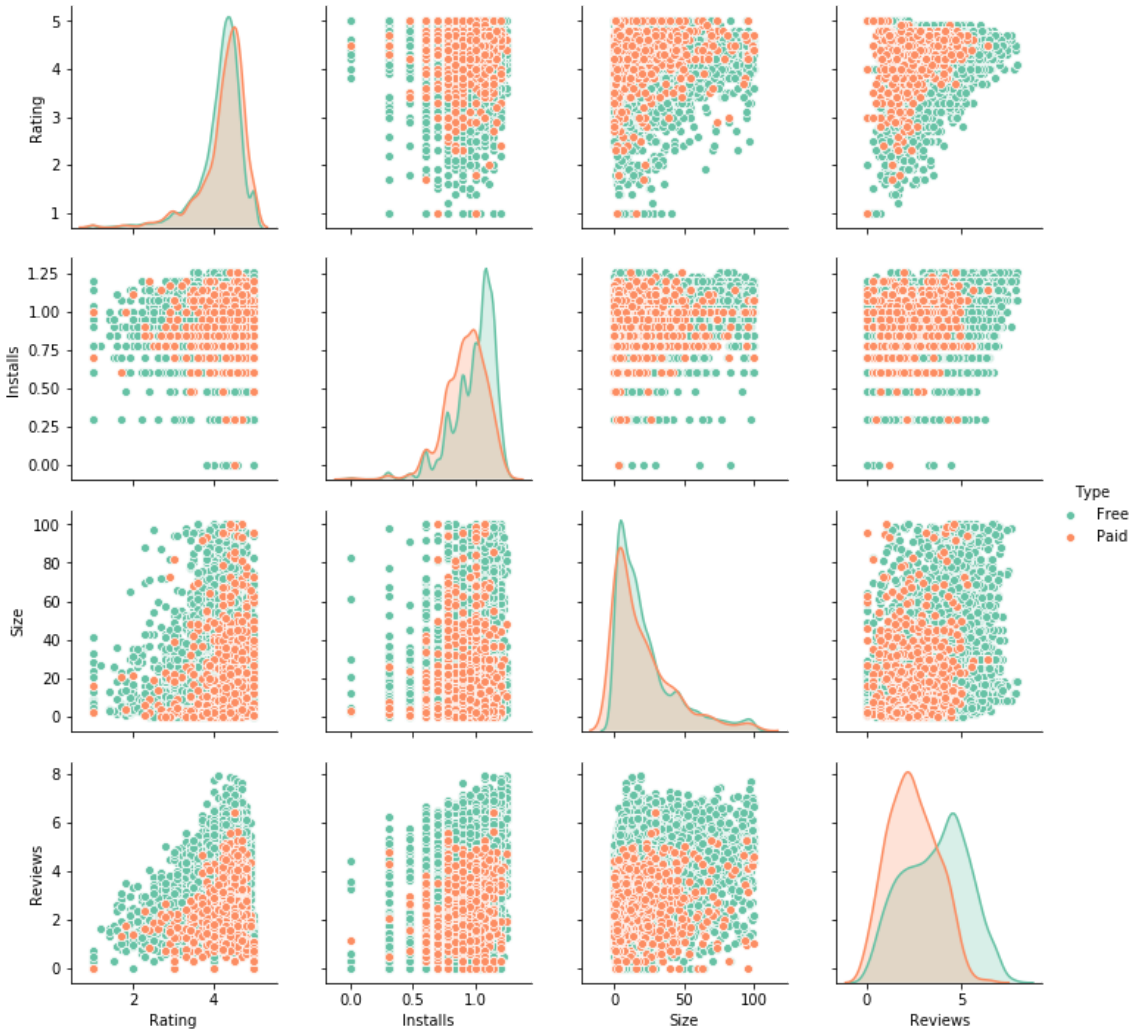
By plotting the regression plot, we can see that “Rating” increases when the number of installs increases, which is the same result from our heatmap of correlation. The reputation of an application should also base on their rating, since from common sense people would download applications which have higher "Rating".



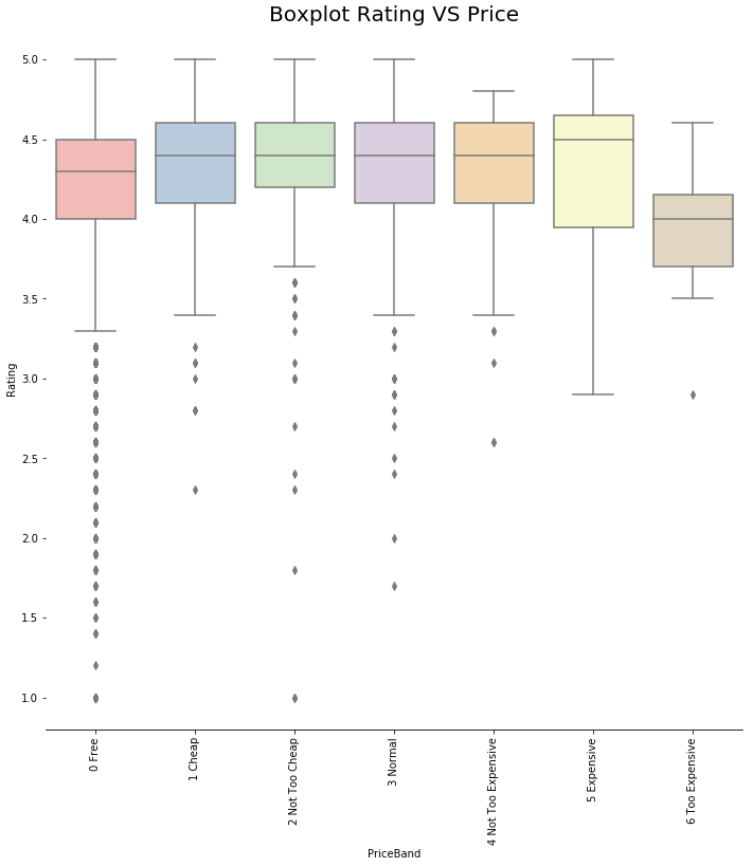
The fifth variable is “Type”, a categorical variable with binary values of paid applications and free applications. From the pie chart, we see that 93.1% applications are free and only 6.9% applications are not free. The average rating for paid application is much higher than free applications. This could be interpreted as people would like to pay for the higher rating applications. Also, paid applications are designed much better than free ones.



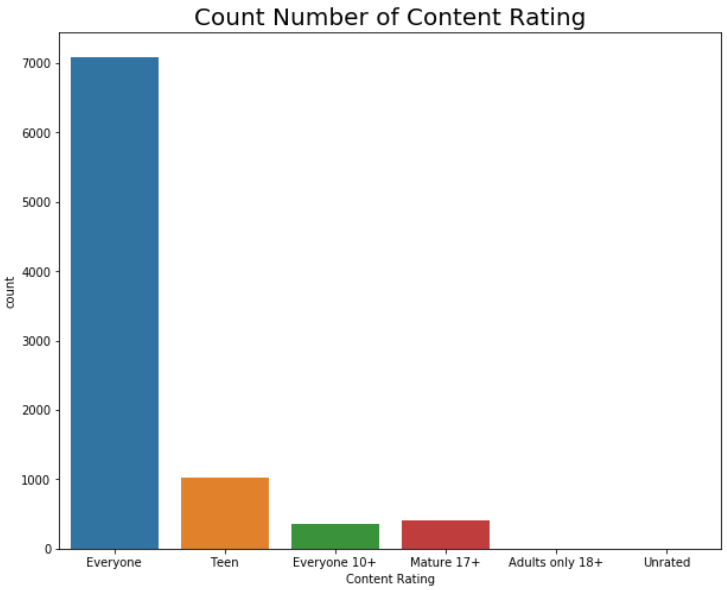
Based on the pairwise plots we conclude that free applications get higher "Rating". Still, the overall distributions of "Rating" for free applications and paid applications respectively are highly similar. The same can be said for the overall distributions of sizes for free applications and paid applications. Meanwhile, although free applications have more installations then paid applications, paid applications received more reviews than free applications.Although paid applications might have higher rating than free ones, people still would like to use free ones instead of paying a lot for downloading an application. So the installation of free applications would be much more than paid ones.



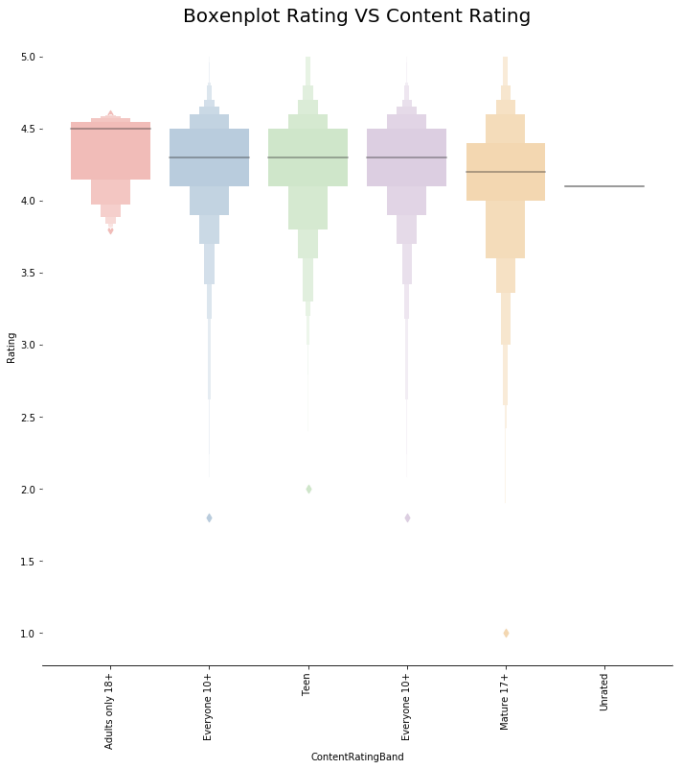
The sixth variable is “Price”. From our heatmap of correlation, we didn’t see any relationship between “Rating” and “Price”. We therefore convert the numerical variable “Price” to categorical variable “Price Band”, which separates the prices into 7 ranges. We then use boxplot to check the relationship between “Rating” and “Price Band”. From the boxplot, we see that free applications got lower rating and expensive applications got higher rating. This makes sense since paid applications should be designed better than free ones since people need to pay to use them. But for the too expensive ones, the rating is very low. People are not satisfied with those very expensive applications because they are too expensive and they do not reach users’ expectations based on their price. We can see that “Price” actually have influenced “Rating” but there is no linear relationship between them.



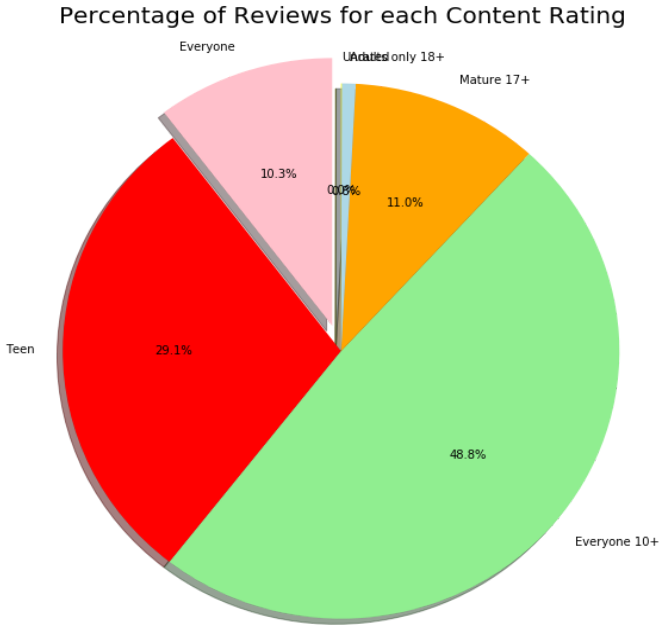
The seventh variable is “Content Rating”, the specific user groups that rate the applications. Based on the barplot, we can see that most applications receive their rating from EVERYONE.



By setting our own band and checking the boxen plot for “Rating” vs “Content Rating”, we can see that Adults Only 18+ give higher "Rating".

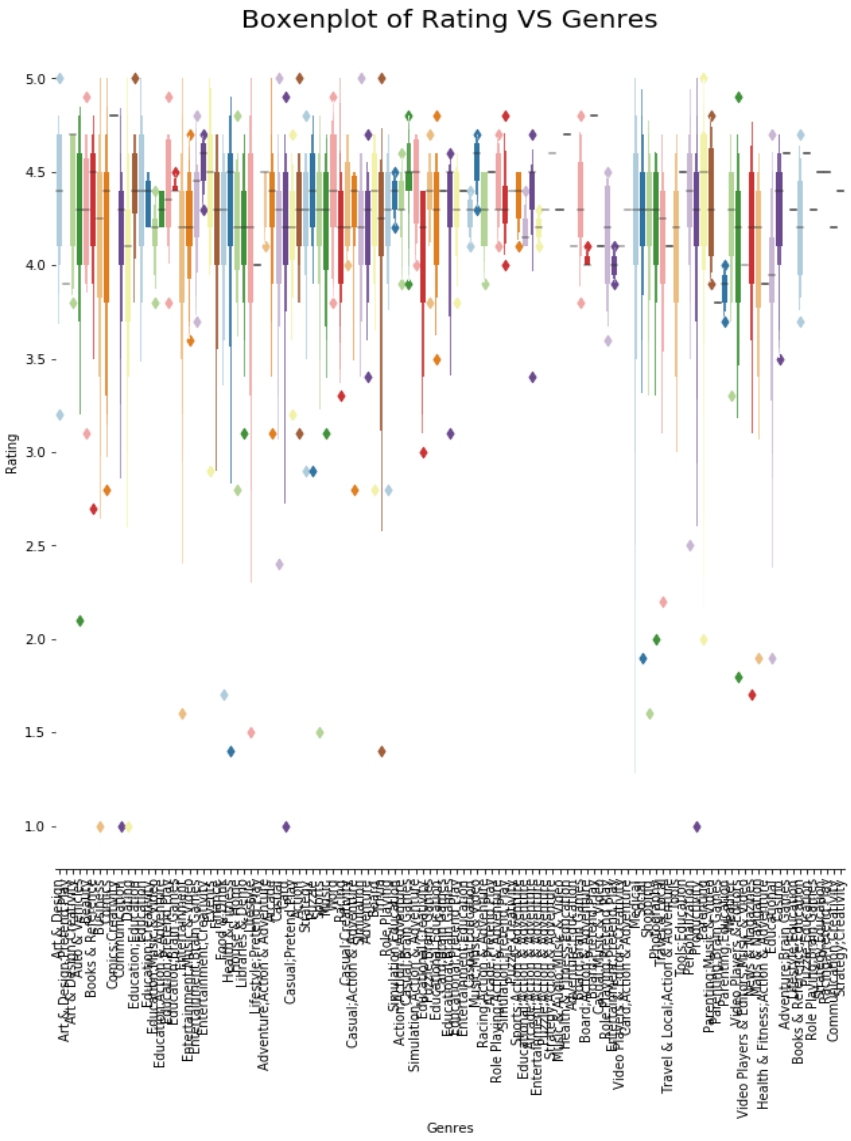


By plotting the pie chart of percentage of reviews for each group, we can see that Teens and Everyone 10+ write a lot of reviews. This shows most applications are used by young people and they are the main target users for app developers.

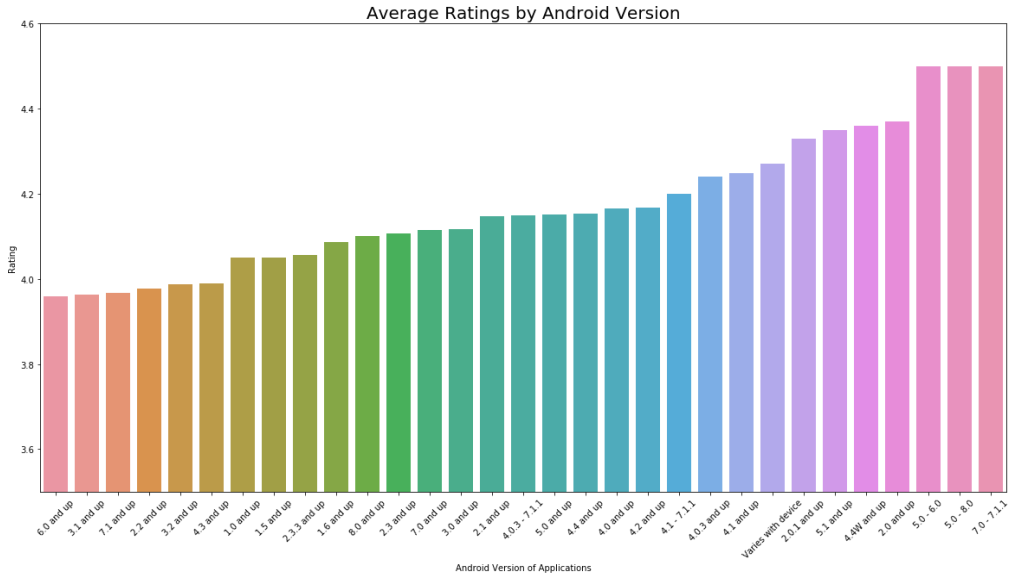


The seventh variable is “Genre”. By checking the standard deviation of “Genre” after grouping by “Rating”, we know that genre does not affect rating too much. Still, we check the box plot of “Rating” vs “Genres”. We cannot say that genres have strong relationship with "Rating". But we can see that the genre of “Comics;Creativity and Board;Pretend Play” got highest rating and genre “Parenting;Brain Games” got lowest rating.

Shaoran Li (sl4bz), Wenxi Zhao (wz8nx), Fang You (fy6vj)



The last variable is “Android Version”. Based on the barplot only, we cannot conclude the relationship between "Rating" and current android version of applications since each application has its own progress on updating the version. So the comparison of versions across all applications does not make sense.



Apart from the csv file for Google Play Store, we also have another csv file for Reviews only. It includes all reviews for each application, but since it does not contain “Rating”, we cannot do sentiment analysis and predict the “Rating” based on the content of “Reviews”. Nonetheless, we draw a word cloud for the corpus of reviews. We can see that the percentages of positive, negative and neutral reviews are 64%, 21% and 15% respectively.

Evaluation Setup

One potential bias in our dataset is that some applications have more than one version included in the dataset, the rating for these apps vary from version to version. If the rating is the same, after dropping the version regressors, these apps could be treated as repetitive data. We treat different versions of apps with different "Rating" as different entity. This is reflected in our exploratory data analysis.

We will consider supervised learning methods only in our project since we are interested in model developing and making predictions instead of studying the data structure. The methods that we will explore are linear regression, knn, svm, random forest and xgboost. Each of these models make certain assumptions, for example, simple linear regression assumes that if the model is adequate, there is constant variance in residual of predicted response against original response variable. Random forest requires that training sample is representative, since we used the cross validation of 8 to 2 as split of training and test, this is not a problem. KNN is non-parametric model, hence it doesn't have specific assumptions for parameters. SVM is classification method, hence it assumes the variables are independently distributed. From correlation matrix, we can see that the highest correlation among regressors is 0.6 between number of reviews and installs, which is a positive relationship but not alarmingly strong. XGBoost assumes that encoded integer values for each input variable have an ordinal relationship, this is addressed by converting categorical variables into binary dummy variables.

We will be evaluating models based on mean squared error, this can be applied across all models. We will validate our models using cross validation method. The training and testing dataset will be split based on 8 to 2 ratio.

Our approach is different from the previous works mentioned in previous section State-of-Art. The two main approaches that Mining and Android used are correlation analysis and clustering. We did employ correlation matrix heatmap in our exploratory data analysis. Nevertheless, from

correlation matrix alone we cannot draw a conclusion of relationship between regressors and the response variable since there are five categorical regressors that cannot be measured against response variable using correlation. Correlation analysis alone is not comprehensive and sophisticated for determining relationship among parameters. As mentioned before, since we are more interested in make predictions compared to study data structure, we exploited supervised learning methods only. Whereas in Mining, clustering is used to examine if applications placed in the same category are also functionally similar or whether App developers tend to develop Apps from the same category. Since our goals are different, the methods used to explore data and reach conclusions are very different in nature, which is understandable.

Approach

We fit 5 models to our dataset, one being the ordinary least squares (OLS), and the other 4 being k nearest neighbors (KNN), support vector machine (SVM), random forest (RF) and gradient boosting (GB). Among these 3 machine learning algorithms, SVM and RF are long standing models, while GB being a relatively new algorithm. We are implementing SVM and RF using SVM and RandomForestRegressor respectively. These two classes are both incorporated in scikit-learn, one of the most widely applicable Python machine learning libraries. We also use scikit-learn to split our dataset and conduct cross validation. We implement GB using XGBoost, an optimized distributed gradient boosting Python library.

These models make certain assumptions about our dataset, one of them being data types of variables. Predictor variables such as Content Rating, Genres, Android Ver are categorical variables and have the data type of string, whereas linear SVM, RF and GB require numerical variable inputs. We transform our data set by converting categorical variables into dummy / indicator variables. Such conversion increases the dimension of our dataset and thus computational complexity of modeling to a reasonable degree, while giving us more freedom to apply and compare various models to our data.

For our OLS model, we constructed two models--one with full regressor matrix and one dropping categorical variables. Note that R-squared is 0.076 and average MSE across five folds is 0.259 for simple linear regression full model. We know that converting too many categorical variables will boost up the dimensionality henceforth the variance, therefore we try dropping the dummy variables. Note that R-squared is 0.917 and average MSE across five folds is 0.267 for simple linear regression full model. This is a great improvement compared to ols model with categorical variables, without too much sacrifice of MSE. The average of ridge mean squared error across five folds is very similar to ols, as 0.259 and 0.267 for full vs dropping categorical models. The average of lasso mean squared error across five folds is slightly better for regressor matrix with categorical variables, as 0.2683 compared to 0.2685 for regressor matrix without categorical variables. But it's no improvement from ols.

With categorical variables, linear models behave better. But we know that dimensionality is an issue here, too many regressors would boost up the variance and eventually result in poor prediction power of the model. We considered method PCA to reduce dimensionality , nevertheless PCA doesn't work well with categorical variables, it only works well with continuous variables. Hence we move on to non-linear methods. KNN,SVM, Random Forest and XGBoost are explored. Among which Random Forest is known as a great way to address the multidimensional issue with large datasets.

For KNN, we construct a graph of MSE against k varying from 1 to 20 for both full regressor matrix and reduced matrix after dropping categorical variables. After comparison, we can see that when k is around 11 to 12, both graphs approach a constant MSE around 0.27. When k is less than 10, clearly the model with categorical variables is doing better with less MSE. When k is above 10, the two graphs merge. At a larger k, MSE for reduced matrix is doing better than full regressor matrix. When k is 15, we get that average MSE from 5 folds for full regressor matrix model is 0.268; average MSE from 5 folds for reduced regressor matrix without categorical variables is 0.265. This is not much improvement from linear regression model.

We prefer linear SVM over non-linear SVM for a few reasons. Our dataset has much more data entries comparing with the no. of predictor variables even after we conduct transformations and create dummy variables. As such, linear SVM should be more efficient in capturing the features in our predictor variables and classify our responses. Linear SVM is also generally more computationally efficient than non-linear SVM. Still, efficiency is not the major concern for our analysis, since our dataset is of moderate size. Using the pipelined linear SVM with its default settings, we get moderately accurate results, with mean prediction error (MPE) of 0.0655 and mean squared error (MSE) of 0.270.

For RF, we iteratively run the algorithm with 2 different settings, one with 50 estimators i.e.: “trees” and the other with 100 estimators. Comparing with SVM, RF requires less assumptions The first iteration achieves a MPE of 0.0200 and a MSE of 0.2248. The second iteration achieves a MPE of 0.0197 and a MSE of 0.2245. It is therefore evident that a higher number of predictors (100 in this case) can better help us improve the accuracy of prediction. Given the relative size of our dataset, the increase in computational complexity with 100 estimators for improved prediction accuracy is acceptable.

GB is an ensemble approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It uses a gradient descent algorithm to minimize the loss and correct the errors made by existing models when adding new models. We know that GB dominates structured or tabular datasets on classification and regression predictive modeling problems, and we expect it to outperform SVM or RF. At a relatively small number of estimator (70, comparing with 100 for RF), we already achieve a MPE of 0.00516, much smaller than that of SVM or RF. GB also produces a MSE of 0.2242, the smallest of all models so far, supporting the assertion that it is a strong model and produces accurate prediction of Rating. Nevertheless, among the process of constructing models, we realize that XGboost usually generates a small MSE with large prediction error. This means that XGBoost faces the risk of overfitting the model.

Conclusion

Based on our models, there is significant correlation between our predictor variables and response, Rating. We can observe that under ordinary linear square, linear regression of the full model outperforms the models with reduced regressor matrix dropping categorical variables of Ridge and Lasso. However, the high dimensionality of full model prevents it from being selected as the ultimate model. As we can see from KNN, the full model is outperformed by reduced model above certain threshold (i.e.: when k >12). Though KNN doesn't give a satisfying MSE compared to Random Forest and XGboost. It is clear that Random forest generates a smallest MSE compared to the rest of the models. XGboost generates a fairly small MSE as well, but since it has a high risk of overfitting the model, we will choose random forest as our final model. Cross validation and splitting the data into train and test sets also allow us to test the prediction power of our models. From exploratory data analysis, we found out that categorical variables such as genre and android version are not as relevant. We also noted that volume of reviews and number of installs are positively correlated. Both issues of categorical variables and multicollinearity are addressed in the Random Forest model. We also know that a higher number of predictors can better help us improve the accuracy of prediction in Random Forest, so it works out very well. Our models are noticeably accurate in predicting response variable "Rating" based on prediction metrics, especially with Random Forest.

Reference

Chen, Tianqi; Guestrin, Carlos (2016). "XGBoost: A Scalable Tree Boosting System". In Krishnapuram, Balaji; Shah, Mohak; Smola, Alexander J.; Aggarwal, Charu C.; Shen, Dou; Rastogi, Rajeev. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* ACM. pp. 785–794. arXiv:1603.02754. doi:10.1145/2939672.2939785.

Grano, G., Sorbo, A. D., Mercaldo, F., Visaggio, C. A., Canfora, G., & Panichella, S. (2017). Android Apps and user feedback: A dataset for software evolution and quality improvement. *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics - WAMA 2017*. doi:10.1145/3121264.3121266

Mokarizadeh, S., & Matskin, M. (2013). Mining and Analysis of Apps in Google Play. *Proceedings of the 9th International Conference on Web Information Systems and Technologies*. doi:10.5220/0004502005270535

Mario Frank, Ben Dong, Adrienne Porter Felt, and Dawn Song. Mining permission request patterns from android and facebook applications. In ICDM, pages 870–875. IEEE Computer Society, 2012.