Google Play Store Apps Analysis

Jiamin Tang
Data Science Initiative, Brown University

https://github.com/jiamin-tang/Data-1030-Project

1 Introduction

1.1 Motivation

As app development is an increasingly innovative industry nowadays, ratings in this competitive market has become a significant index to measure the success of an app. It not only has the potential to give developers a hint how to make a more popular product, but also become a determinant for users' purchase decisions. High-rating apps are more likely to be attractive to users, thus can drive business success in app-developing business. Besides, the vast availability of the features of existing apps published by app stores provides the possibility to predict apps' ratings.

1.2 Dataset Description

In this case, we mainly focus on predicting the rating of an app in Android market based on the collected features of published apps. The target variable will be the app ratings which is an average of user ratings out of 5 stars. Also, this is a problem of regression.
The dataset is from Kaggle and it is for Google play Store Apps which is web scraped of about 10k Play Store apps for analyzing the Android market. It contains the objective statistics of the app including size, installs etc.. The goal is to predict ratings of apps with the given features.
The dataset contains 10840 entries and 12 features with each app, accompanied with the target variable rating. The description of each feature is given below:
1. App: Application name
2. Category: Category the app belongs to
3. Rating: Overall user rating of the app out of 5 stars(as when scraped)
4. Reviews: Number of user reviews for the app (as when scraped)
5. Size: Size of the app (as when scraped)
6. Installs: Number of user downloads/installs for the app (as when scraped)
7. Type: Paid or Free

8. Price: Price of the app (as when scraped)
9. Content Rating: Age group the app is targeted at – Children / Mature 21+ / Adult
10. Genres: An app can belong to multiple genres (apart from its main category). e.g. a musical family game will belong to Music, Game, Family genres
11. Last Updated: Date when the app was last updated on Play Store (as when scraped)
12. Current Ver: Current version of the app available on Play Store (as when scraped)
13. Android Ver: Min required Android version (as when scraped)

2 EDA

The histogram of the target variable is left skewed and the most rated interval is between 4 and 4.5, where the average 4.19 falls into.

It is hard to look into the relations between rating and category since the categories are trivial. Thus I select categories which have more than 220 apps, which is about the median among all categories. In the box plot from figure 2, some categories achieve good performance such as health and fitness and personalization, for half of apps in these categories are rated higher than the average. While there are some categories that both have high ratings and low ratings such as finance, family and tools. It implies that people may be more selective for these apps. Still, these are trivial clues we can get.

From the scatter plot from figure 3, apps that are large in size have higher

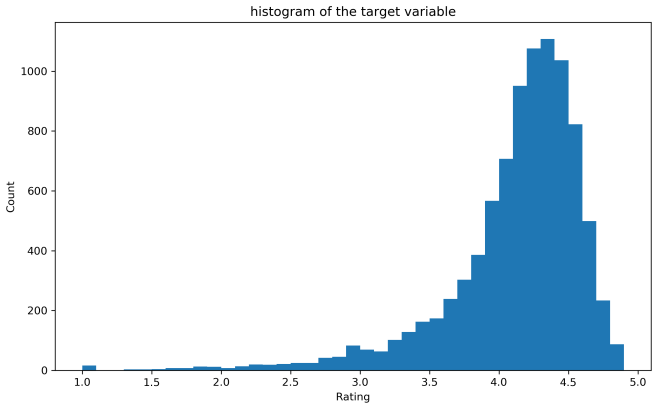

Figure 1: Histogram – Rating

ratings on average. The feature Size may have something to do with Rating.

On average, paid apps achieves higher ratings, which can be seen from figure 4. The feature Type which implies the apps are paid or not, and the feature Price may affect Rating to some extent.

When we look into the relationships between features from a scatter matrix from figure 5, there are no obvious clues from the features except Installs, Size and Reviews. Thus we strive to discover underlying relationships using ML models as follows.

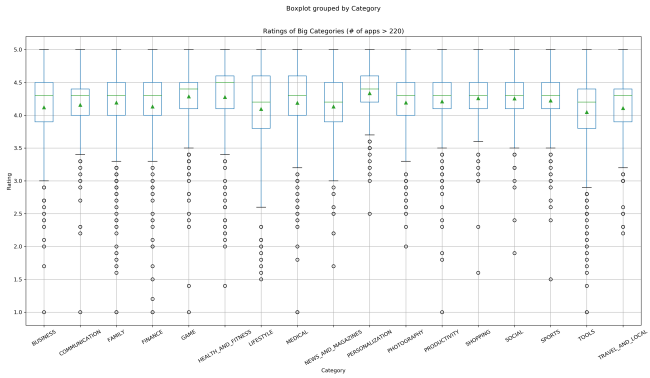

Figure 2: Box Plot – Category and Rating

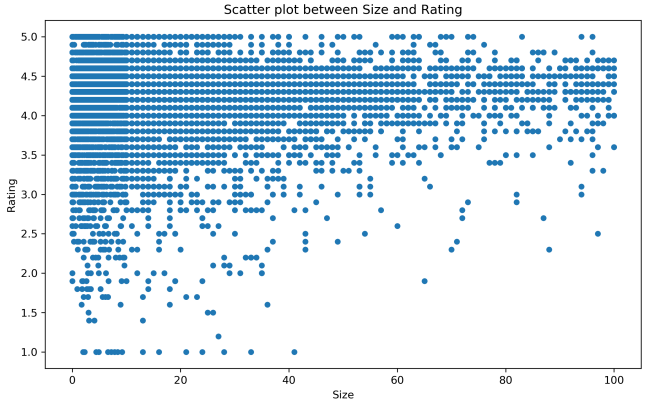

Figure 3: Scatter Plot – Size and Rating

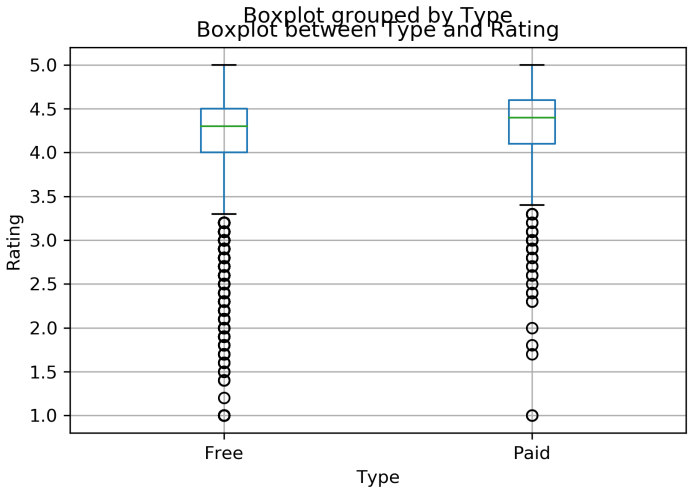

Figure 4: Box Plot – Type and Rating

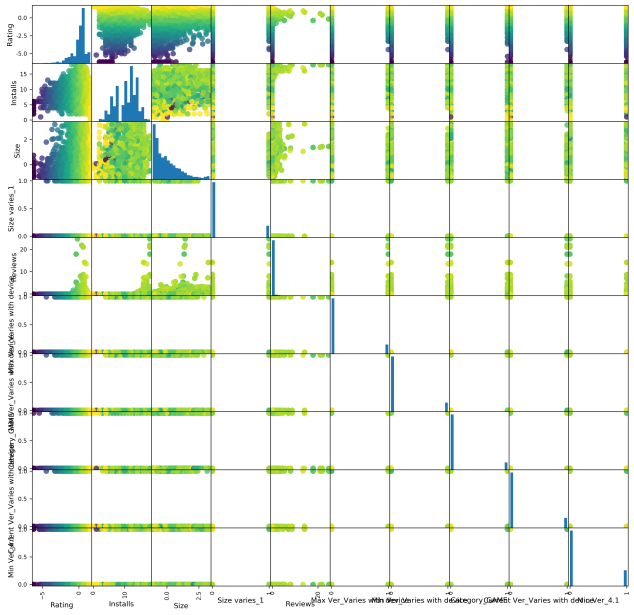

Figure 5: Scatter Matrix

3 Methods

3.1 Preprocessing

The percentage of missing values in total is about 13.66%, with the majority 13.60% in the target variable rating. My solution is to save the missing data points in the target variable as a new dataset and reserve for later test, then drop these missing rows to do the preprocessing. Also, the column of App names is dropped.
The preprocessing steps includes data cleaning and transformations. Some conversions should be made first, including strips, splits, conversions to numeric and unifying units. For some special features, for example, the feature Size which can be converted into numeric later, contains the value "Varies with device". New column is created regarding whether it varies or not and is handled with Onehot Encoder. Then the value in the feature is replaced with nans and is imputed with Iterative Imputer when transforming. Likewise, "varies" in categorical features are treated as a new category when using OneHot Encoder. One categorical feature applied with Ordinal Encoder is Installs, of which the values have a growing sequence. Continuous features are mainly applied with Standard Encoder. There are 9360 data points and 301 features after preprocessing. The columns increase a lot since categorical features with many unique values are transformed with OneHot Encoder.

3.2 Model Discussion

The models I tried are Ridge regression, RandomForestRegressor and XGBoostRegressor.
For Ridge regression, I tune the parameter alpha from the range of 10^{-10} to 10^{10} with 20 numbers spaced evenly on log scale.
For RandomForestRegressor, I tune the parameters max_depth and min_samples_split. The range for max_depth is from 1 to 16 with step 5, and the range for min_samples_split is from 3 to 13 with step 5.
For XGBoostRegressor, I tune the parameters max_depth and learning_rate. The ranges are $[3, 30, 100]$ and $[0.01, 0.05, 0.1, 0.3]$ respectively.
20 % of data points are split into the test set and k-fold cross validation is applied to split the rest 80 % of data points into training set and CV set. Most of the best parameters turn out to be not near the boundaries.

3.3 Evaluation Metric

The evaluation metric I use is R2 score which can be calculated as $R^2(y, y') = 1 - \frac{\sum_{i=1}^{n}(y_i - y_i')^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$, where y and y' are actual target value and predicted target value respectively. It can be easily compared to the baseline model to evaluate the effectiveness of the ML model. Also, I use mean squared error to evaluate the performance between models.

3.4 Uncertainties

The uncertainties are involved in data splitting and RandomForestRegressor, including IterativeImputer using RandomForestRegressor as an estimator. I apply k-fold CV to split the data and for each fold, I tune the best parameters and calculate the best score. The parameter random state ranges from $42 * 0$ to $42 * 9$, following an arithmetic sequence. I calculate the average and standard deviation of R2 score to maintain the uncertainties.

4 Results

4.1 Model Performance

The R2 score of the baseline model of regression is 0, which predicts the mean of the target variable.
The average R2 score of Ridge Regression is 0.065 with standard deviation of 0.012. The small standard deviations implies that the results are stable. It is not a perfect regressor since it is far away from 1 but it is above the baseline model.
The average R2 score of RandomForestRegressor is 0.176 with standard deviation of 0.018. It performs better than Ridge Regression though the score is not good. I try to improve the model performance by changing the range of the parameters I tuned. When the maximum depth of the tree gets larger, the model tends to perform better since more features come to affect the model. The test score from 0.109 to 0.166 when the range of depth change from $(1, 5)$ to $(1, 20)$. Also, since the dataset has more than 9000 data points after preprocessing, overfitting may come to exist when the minimum samples split is too small. The score gets improved from 0.166 to 0.177 when the range of split change from $(3, 15)$ to $(3, 200)$.
The average R2 score of XGBoost regressor is 0.185 with standard deviation of 0.021.
The mean squared error of the three models are 0.246, 0.217, 0.204 and respectively. RandomForestRegressor performs best among three models.

4.2 Global Feature Importance

I evaluate the feature importance using feature_importances_ within RandomForestRegressor and XGBoostRegressor. There are some overlapped features which are assigned higher importance in both models and still there are some differences.
For RandomForestRegressor, I select the best parameter with the best score I have tried and plot several bar plots showing top 10 important features. The five figures (five is the k-fold split) under random state 210 (the random state that gives the best R2 score) are similar, which implies stability to some extent. Also there are some findings which has some real-world meanings:

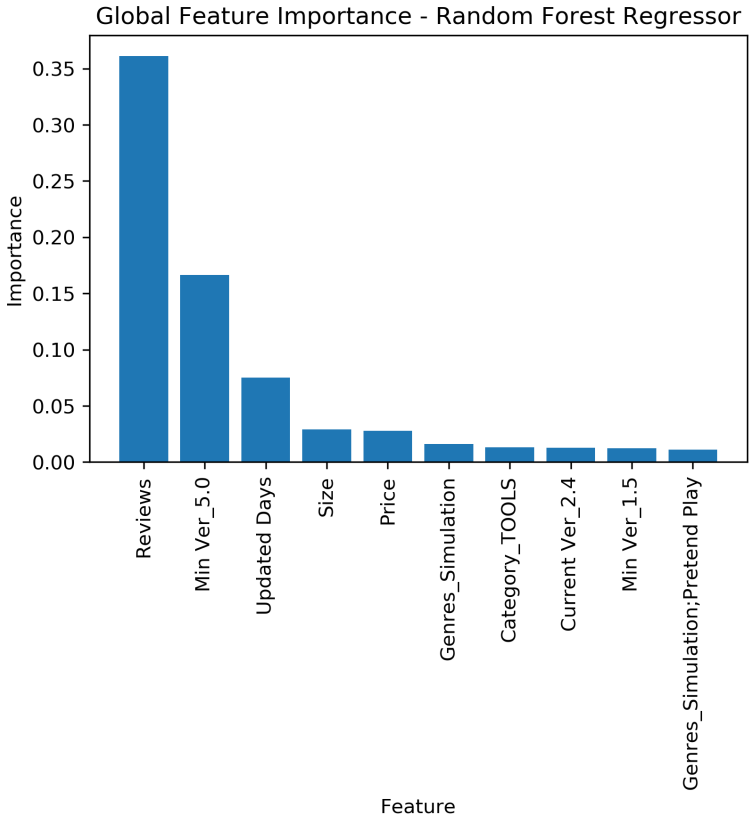

Figure 6: Feature Importance: RandomForestRegressor

1. The feature Reviews is the most important feature, which does not contradict with our intuition that more reviews may contribute to high ratings.
2. Updated Days which means the days from the app's last update till now is another indication of rating. It is not clear from the importance if it contributes to high rating or it is negatively correlated, since for some apps, especially game apps, recent updates can fix bugs and improve some functions, while for others, there are people who prefer to stick to old versions.
3. There are some categories such as Simulation achieving high importance. When I look back on the dataset, I find that these are mainly game Apps and the average rating is over 4, which implies it is positively correlated to ratings.
4. Size and price are among important features, which is in accord with the findings in the EDA part that on average, paid apps that are large in size achieve high ratings. It makes sense that large apps tend to be more functional, which brings better user experience.

For XGBoost, Reviews are also among the most important features. Other selected feature with higher importance are mainly some specific category in-

dicators. Also, Installs can mean a lot to ratings. The difference of important features between the two features may be attributed to the different parameters such as max_depth which select different features when modeling.

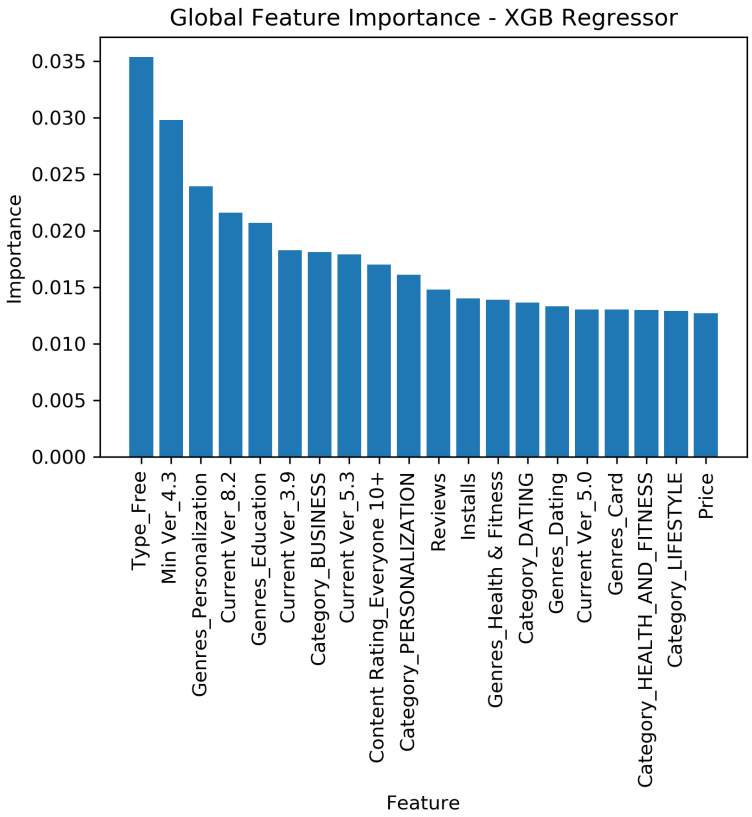

Figure 7: Feature Importance: XGBoost Regressor

5 Outlook

Since neither of the three models perform well on the dataset, I look back on the dataset itself. After preprocessing, the dataset contains hundreds of columns, which slows down the speed of fitting models. Some feature engineering techniques like PCA may help to improve the modeling. Since OneHot Encoder transforms categorical features without underlying sequence into many columns, there may exist some balance between OneHot Encoder and Ordinal Encoder to deal with such features with many unique values. Also, some techniques such as GridSearch might help when finding the best parameter range.

6 References

a. https://www.kaggle.com/lava18/google-play-store-apps
b. https://www.kaggle.com/lava18/all-that-you-need-to-know-about-the-android-market
c. https://medium.com/analytics-vidhya/is-it-possible-to-predict-rating-of-google-play-store-apps-based-on-the-given-information-da9a44a3ac1e
d. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/