Exploratory Data Analysis and Rating Prediction on Google Play Store Apps

Mohammad Zarei
University of California, San Diego
Electrical and Computer Engineering
Email: mozarei@ucsd.edu

Payam Khorramshahi
University of California, San Diego
Electrical and Computer Engineering
Email: pkhorram@ucsd.edu

***Abstract*—we performed exploratory data analysis and visualization on the google play store app dataset to better understand the correlation of different attributes with the rating of an application. Furthermore, we implemented a series of supervised and unsupervised learning algorithms such as linear regression, support vector machine, random forest, and k-mean clustering for modeling rating prediction. Among all model K-mean + KNN model achieved the highest accuracy 81.5% on the test data.**

1. Introduction

The history of mobile application is quite interesting.Google store officially started with 500 apps in a platform.This was a new era that gave people all around the world this idea to build apps. Nowadays, there are almost 2.7 millions apps in Google store which makes the Google store the biggest number of available apps platform. Quite number of the apps have made our lives easier because of their advantages. For example, they made our world much smaller and distance is no longer a problem. With all being said, we encounter this question. What if I want to build an app? what kind of apps have higher probability of success?

One of the crucial business factors that should be considered by app developers is to have an understanding of the level of success that could be faced by their apps once released into the market. One way to absorb this understanding is by looking into dataset which expose the history of previous applications. Studying each app from a large dataset is a cumbersome and nearly impossible task for human. Thus, one solution is to rely the power of computers to do this analysis for us. In this research project we will perform a comprehensive analysis of the Android app market via comparing 10,000 apps in Google Play across different categories. We will then pick the best attributes that contribute to the ratings. The ultimate goal of this project is to evaluate the success of an apps by making prediction on their ratings based on other features. A total of 4 learning algorithms (Linear regression, K-mean Clustering, Random Forest, and Support Vector Machine (SVM) are considered to be implemented and evaluated on this dataset.

2. Dataset

The dataset used in this project is the Google-Play-Store-Apps taken from Kaggle. The raw dataset is composed of a total of 10841 rows. After excluding rows with missing values and duplicates the dataset was reduced to 8484 rows. The dataset is representative of 13 columns: [AppName, Category, Reviews, Size, Installs, Type, Rating, Price, Content Rating, Genres, LastUpdated, CurrentVer, AndroidVer]. Among these 13 columns, LastUpdated and Reviews are excluded since these information would not be present for newly released apps. Columns Type and Genres are also excluded for statistical reasons that will be explained shortly.

3. Data Visualization

The goal of this project is to predict ratings of each application. Thus, its important to have an insight on this feature in the dataset. We obtained the distribution of ratings over all android application.

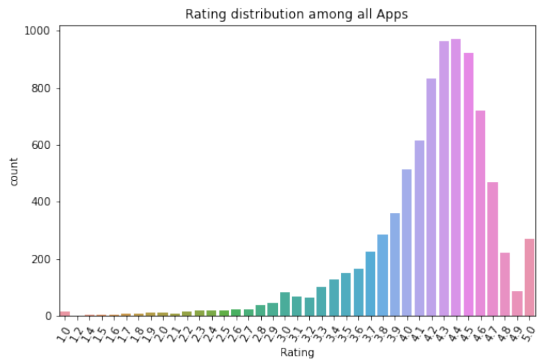
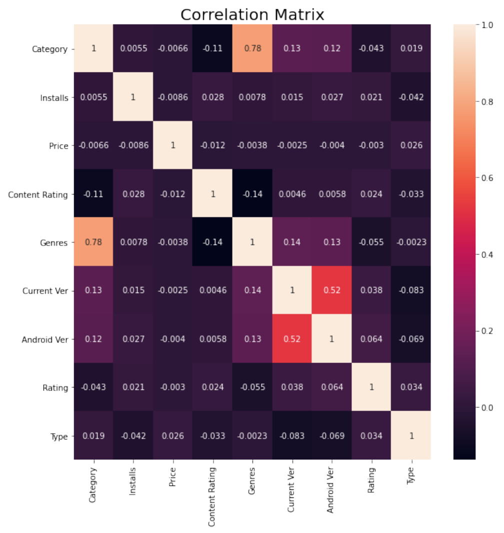


Figure 1

As shown by Figure (1) the behavior of the rating attribute is similar to a Gaussian distribution with mean of 4.4 and standard deviation of 0.4. The majority of the ratings are distributed in the range of 4.1 to 4.6. There exist a large class imbalance for ratings in the range of 1-3.4. This makes the problem challenging in predicting lower ratings. It's noteworthy that although rating is a continues variable, however in this setting where all rating values have been rounded to one decimal place, we consider that as a discrete random variable. This is in fact a significant feature that we considered when picking prediction algorithms. Now that we have an understanding of the overall rating distribution, its time to have an insight on how other attributes in the dataset would contribute to the ratings. Initially, we created a correlation matrix for 9 of the most important features. From Figure (2) it can be observed that there exist a very low correlation among most attributes in the dataset. However, a correlation of 0.58 exists between column ”Current Version” and ”Android Version”. Also a correlation of 0.78 exists between attributes ”Genres” , and ”Category”. Both of these correlations are high and mightly induce the colliniearity problem in the learning algorithms. It is always desired to have a dataset which cosists of features with low correlation because this makes each feature unique and thus exposing new information about each data-point. In other words, learning models tend to learn more feature from data with less correlation and thus making better prediction. In addition, this helps models such as K-mean clustering and SVM to do a better job in separating each region corresponding to a specific ratings.



	Category	Installs	Price	Content Rating	Genres	Current Ver	Android Ver	Rating	Type
Category	1	0.0055	-0.0066	-0.11	0.78	0.13	0.12	-0.043	0.019
Installs	0.0055	1	-0.0086	0.028	0.0078	0.015	0.027	0.021	-0.042
Price	-0.0066	-0.0086	1	-0.012	-0.0038	-0.0025	-0.004	-0.003	0.026
Content Rating	-0.11	0.028	-0.012	1	-0.14	0.0046	0.0058	0.024	-0.033
Genres	0.78	0.0078	-0.0038	-0.14	1	0.14	0.13	-0.055	-0.0023
Current Ver	0.13	0.015	-0.0025	0.0046	0.14	1	0.52	0.038	-0.083
Android Ver	0.12	0.027	-0.004	0.0058	0.13	0.52	1	0.064	-0.069
Rating	-0.043	0.021	-0.003	0.024	-0.055	0.038	0.064	1	0.034
Type	0.019	-0.042	0.026	-0.033	-0.0023	-0.083	-0.069	0.034	1

Figure 2

In the dataset, column ”Types” provides if the app is paid or free. Thus this column is composed of only 2 categories. Note from the pie chart in figure (3a) that the 93.2% of the apps are free. Since there is a high class imbalance between the two categories, this could deteriorate the performance of prediction model on the minor class. There is another column in the dataset called ”Price” which provides similar information but in a wide range of categories, specifying more detailed information about applications in the smaller class (Figure (3b)) which could have a positive impact on the learning process. Therefore only the ”Price” column is used and the ”Type” column is excluded.

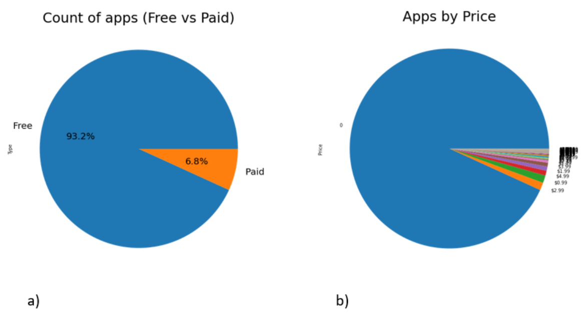


Figure 3

Next column to investigate represents the ”Category” attribute. As shown in figure (4b) GAME and FAMILY are the most common attributes observed in the dataset providing the most promising information regarding the ratings. Compared with the ”Type” column, this attribute is composed of a more diverse classes. Figure(4a) represents how this column is distributed with respect to the app rating. Although this distribution is similar to being uniform, however the presence of fluctuation among different categories could help the prediction models to isolate apps based on rating.

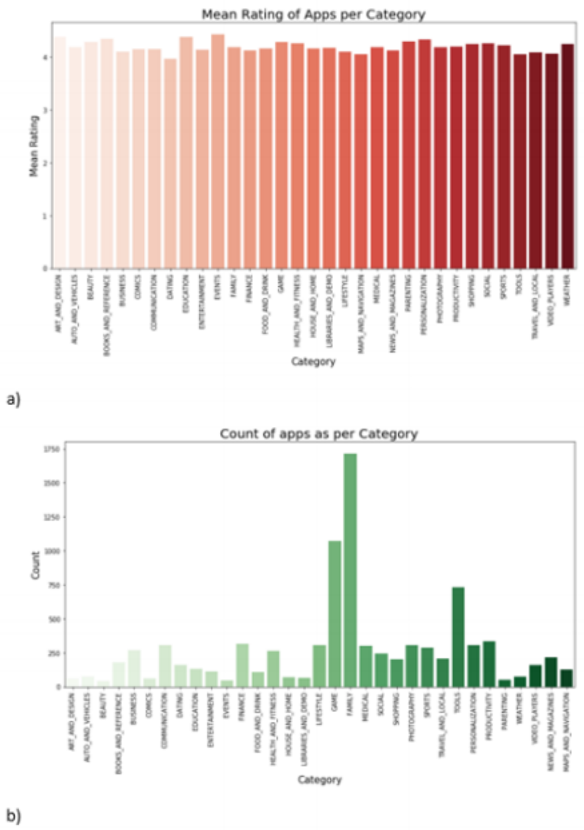
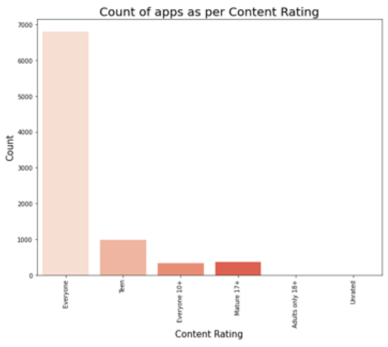
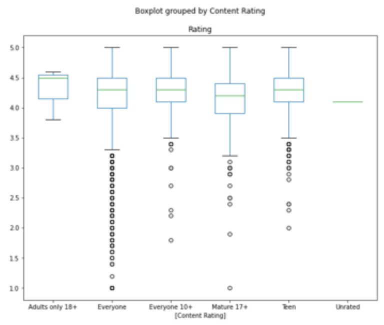


Figure 4

Another column in this dataset provides content rating of the apps. There are a total 6 classes in this columns as shown in figure(5a). The majority of the apps (83.2%) are available for everyone's use. The remaining 16.8% require age restriction. From the box plot shown in Figure(5b) it can be seen that the distribution of the two most populated contents (Everyone and Teen) are similar. The position of the median and quartiles for both columns possess a close range of values. However, small fluctuation among different contents could contribute to better learning. Despite, the "Category" column "Content Rating" is not as diverse in terms of number of classes. Therefore, it might do poorly in terms of learning more complex features compared with the "Category" attribute.



a)



b)

Figure 5

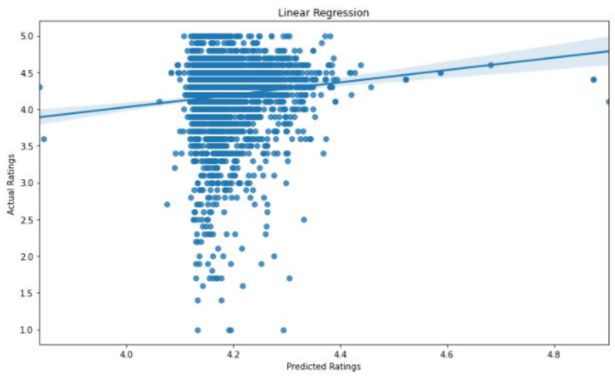
4. Models

We performed different machine learning algorithms in our data set to predict the rating. Features that we used for training are Price, and Content Rating, category, android version, current version.

4.1. Linear Regression

Algorithm Info: Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It's used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog). There are two main types:simple and multivariate. More information is linked in the references.

Since we want to predict the rating based on different features such as size, android version and etc. , we decided to perform linear regression in our data-set. The rating feature is the dependent variables and others are independent in our model. Here is the plot of linear regression for predicting rating in our data-set.



Linear regression model on our data-set is plotted above. The plot axis's are actual vs predicted results.The error values based on different loss function are as follows,

Loss Function	Error Value
Mean Squared Error	0.26331416544107183
Mean absolute Error	0.35565708267114116
Mean squared Log Error	0.011655702079437644

4.2. Support Vector Machine

Algorithm Info: The objective of the support vector machine algorithm is to find a hyper-plane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.To separate the two classes of data points, there are many possible hyper-planes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Considering the fact that some of the features are non linear, we decided to use support vector machine to predict the rating.In this model, SVM does not seem to perform well because of the number of features are little and our data-set might no be large enough. Here is the SVM plot for predicting rating in our data-set.

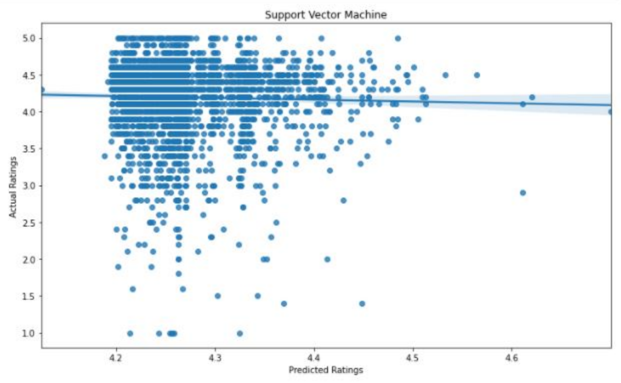


Figure 6

Support vector machine model looks a little bit better that Linear regression as the error values are lesser.The error values based on different loss function are as follows,

Loss Function	Error Value
Mean Squared Error	0.25400979424383546
Mean absolute Error	0.3467728356693965
Mean squared Log Error	0.0120802297101839

4.3. Random Forest Regression

Algorithm Info: A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. What is bagging you may ask? Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.
Despite the fact that SVM was slightly better than linear regression, yet the performance is not satisfactory. Since we use more than one features and technically we have multi-class, we decided to use Random Forest which is intrinsically suited for multi-class problems. Here is the random forest plot for prediction rating.

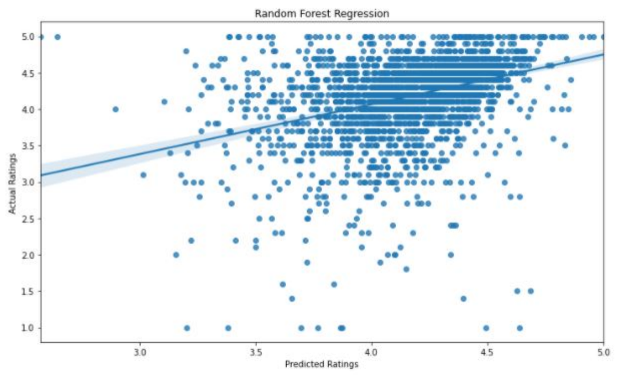


Figure 7

According to the plot, Random forest regression has a better predictive accuracy compare to the previous models. The error values based on different loss functions are as follows,

Loss Function	Error Value
Mean Squared Error	0.21442415172875642
Mean absolute Error	0.30938185371727034
Mean squared Log Error	0.010261427484943313

4.4. K-means + KNN

For this method we used features with the least correlation, since K-mean is one of the most sensitive algorithms that could be fooled by the col-linearity problem. A total of 6 attributes were used which are: Size, Content Rating, Android Version, Price, Category, Rating. Since there were a total of 36 ratings present in the dataset, we expected the best results to be achieved over 36 different clusters. However, best results was achieved through 33 clusters. This is because higher number of clusters requires more complex features to be learned by the model. Training a more complex model on the other hand requires requires more training data. Since the dataset was small, learning all 36 clusters was not achievable. From the plot in figure 8 it can be seen that beyond 33 clusters the model performance significantly decays.

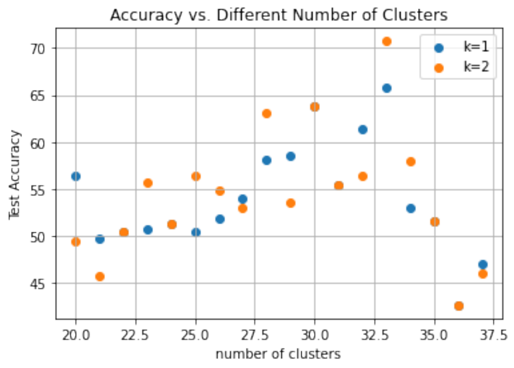


Figure 8

During the testing phase, after assigning each data point to a cluster, we used KNN to find the closest points (determined by k = 1,2,3, ...) to the test data and took their average rating as the predicted value. The maximum accuracy was observed at k = 2. Its noteworthy that when implementing k-mean clustering, the centroids are initially placed in a random position in the multi-dimensional data space and thus their final position after training would be different. Therefore, taking into account the existance of this randomness, after several experiments we concluded that in average best results took place for clusters in the range of [29,34] with k=2. Results, are shown in the table below.

Loss Function	Error Value
Mean Squared Error	0.19516416855246527
Mean absolute Error	0.3256351685155651
Mean squared Log Error	0.0121535133516

5. Conclusion

Its very important to have some prior knowledge on the success of an application before releasing it into the market. This understanding could be attained by extracting information from previous application data. In this paper we discussed the Google-Play-Store-App in details and implemented a few algorithm to learn features that could impact ratings of each app. Although there were some weaknesses in the dataset in terms of size and number of columns, however, upon all, Kmean + Knn returned the best results of 81.5% accuracy. This performance could definitely be boosted on larger dataset with more diverse features.