

COEN 240: Term Project

Spring 2022 M/W 7:10 pm - 9:00 pm

Weijia Li and Sumit Agrawal

[**Abstract**](#_heading=h.1t3h5sf) **3**

[1.1 Problem statement and Objective](#_heading=h.4d34og8) 3

[1.2 Assumptions](#_heading=h.2s8eyo1) 3

[1.3 Methodology](#_heading=h.17dp8vu) 3

[**Data Exploration**](#_heading=h.3rdcrjn) **4**

[2.1 Data Description](#_heading=h.26in1rg) 4

[2.2 Preprocessing](#_heading=h.lnxbz9) 4

[2.3 Feature Engineering](#_heading=h.35nkun2) 5

[**Methods**](#_heading=h.1ksv4uv) **6**

[3.1.1 Baseline Model](#_heading=h.44sinio) 6

[3.1.2 Other Models](#_heading=h.2jxsxqh) 6

[3.2 Parameter Tuning](#_heading=h.z337ya) 6

[3.3 Model Evaluation](#_heading=h.3j2qqm3) 7

[**Results**](#_heading=h.1y810tw) **7**

[**Discussion & Limitation**](#_heading=h.4i7ojhp) **7**

[**References**](#_heading=h.2xcytpi) **7**

# 

# 

# 

# 

# 

# 

**Google Play Store App Ratings Using Data Mining Techniques**

**Weijia Li** and [Sumit Agrawal](mailto:sragrawal@scu.edu)

# 

# **Abstract**

The Google Play Store apps analysis is designed to assist application developers and app-making businesses in the areas of business and app development in the Android market. The focus was placed on predicting the rating of the application using descriptive data about mobile applications. To achieve this objective, the analysis is broken down into three parts.

The first part of the analysis is to identify the current apps, their characteristics, and their ratings. The problem is supervised in nature, therefore new predictions will be implemented based on the data that has been already tagged. 92% of all the apps are free with the most popular genre being - Tools.

The second part is to clean the dataset, remove fields and rows with the least predictive power and prepare the data for the modeling. During this process, columns such as Price, Size, Installs and Reviews were converted into numerical values. The Genres field was reduced from 120 to 47 and numerical fields were standardized using the scikit-learn package StandardScaler. The categorical variables were improved using the one-hot encoding technique. Variables such as App Name, App versions and Last Updated had very little predictive information, therefore were removed.

The third part of the analysis is to predict the ratings of the apps and evaluate

the performances of three different models to choose the best performing model. Random Forest gave the best MAE, MSE, and RMSE values.

This model gave Reviews, Installs, Days Since Last Updated, and Size as most predictive variables. The most optimal parameters are - 50 estimators and max depth 10.

**1. Introduction**

## **1.1 Problem statement and Objective**

The project was conducted to meet two key objectives:

1. Identify technical and behavioral traits that influence users’ app ratings.
2. Outline actionable insights and tactics for developers to work on capturing the Android market.

## **Assumptions**

1. Data sample represents apps’ characteristics in the Google Play Store, and they can change over time.
2. App ratings are coherent with the real scenario and represent actual ratings of people. They haven’t been affected by both ratings and reviews.

## **1.3 Methodology**

To meet the project objectives, the SEMMA (Sample, Explore, Modify, Model, Assess)

approach was followed. This methodology provided a structured approach to both the analysis and devising actionable recommendations.

* Sample - Due to the small size of the dataset, entire data was used to derive insights.
* Explore - Information was visualized using Python and various packages. The data was described, summarized, and aggregated through techniques such as boxplotting, scatter plotting, bar and column graphing.
* Modify - To increase the predictive power of the data, new binary columns were created using techniques such as one-hot encoding and modified existing fields using conversions, normalization, standardization, combination, and removal. All steps are outlined in the code.
* Model - Being a regression problem, linear regression, random forest, K-nearest neighbor, and support vector machine algorithms were utilized to predict the future app ratings. The emphasis was put on tuning the parameters to achieve the best error performance.
* Assess - All models were compared using the common regression metrics such as MAE, MSE, and RMSE. The model with the best overall performances in all metrics was chosen.

# **Data Exploration**

## **2.1 Data Description**

For this project, our data source is the Google Play Store Apps dataset on Kaggle at https://www.kaggle.com/lava18/google- play-store-apps. This area is experimentally convenient because most of the information about the application is publicly available, and such data sets are already organized. An app has a wide range of attributes, such as its type, package size, price, etc., and people

can evaluate it in a variety of ways, but based on the rating system of Google Play Store, we can always describe an app with an average score. Therefore, we do not need to manually tag data for supervised learning or model evaluation. It is worth mentioning that the machine learning techniques we utilize are not app store-specific and should be easily applied to other areas if adequate training data is available.

The Google Play Store Apps dataset annotates each application with a score. Each score is between 1 and 5, with higher scores representing better ratings. For the work described in this article, we focused only on predicting the application's score within this interval. There are more than 10,000 observations in this data set.

## 

## **2.2 Preprocessing**

There are some imperfections in this dataset that affect our modeling. Therefore, we corrected these imperfections using feature engineering. As the dataset was retrieved through a crawler, it had an unexpected 483 repeated rows. We removed them first. Also, we removed the outliers such as a rating of 19 and a category of “1.9” as they did not convey any substantive information.

In this section, we also explored the patterns of some main variables. Among the 10,000+ apps, the most common category is Family, which accounts for about 20%; the second and third places are Games and Tools. 92% of the apps are free, implying that the rate is very imbalanced. Most top-rated apps have an optimal size between 2MB to 40MB, a moderate size that would be convenient for users to download and update.

## **2.3 Feature Engineering**

Generally, numerical variables can be directly fitted by the model, but some are regarded as object variables that need to be converted. For example, we manually remove the dollar sign for some prices and uniformly replace the values “Everyone” with their mode 0. After these two steps, we convert the Price field into a numeric variable. Similarly, the number of installed apps is not a real number, it simply denotes whether more than 50,000, 1 million, or 50 million. Thus, the Install field has commas and plus signs. The invalid text also exists in some values. After processing, we also convert the Install field into numbers that can be understood by the model. For the Reviews field, it is much tidier, so we just replace the “3.0M” with “3000000”.

Next, we normalized the date variables by standard date packages. A date like “January 7, 2018” will be converted to “2018-1-7”. Note that even these standardized dates convey limited information and are hard to be processed by machine learning models, so we need to convert the date of the last update to the number of days since the last update, a continuous variable with mathematical meaning. For this newly generated variable, we notice a median of 108 days and a mean of 295 days, indicating that it is heavily influenced by outliers. We will scale the data to make it uniform later.

On smartphones, the size of most applications is greater than 1MB, but in our data set, there are still some apps smaller than 1MB (Mainly efficiency apps), expressed in KB. We convert these values to be expressed in MB by dividing 1024. We further observe that the size of certain apps varies with different devices, and they are directly denoted as “Varies with device” in the dataset. For these apps, we set their size to 0.

For categorical variables, we firstly dropped some variables that are irrelevant to our analysis, like names and versions. The genre of the apps has multiple classifications. For instance, music is one of the second-class classifications of entertainment, the first-class classification. Such cases are separated by semicolons in our dataset, such as “Role-Playing; Action & Adventure”. To reduce the dimensionality of the data, we only retain the first level of classification. The genres, shown in Fig. 1, are much more compact than before.

| **Genre, original dataset (counts)** | **Genre postprocessing (counts)** |
| --- | --- |
| Action; Action & Adventure (15) | Action (371) |
| Education; Action & Adventure (5) | Education (610) |
| Education; Pretend Play (18) | Education (610) |

Fig. 1: Examples of genres before and after processing

Even so, the text-based fields still cannot be fitted by most algorithms, so we use one-hot encoding to convert it into a number. This is a way to remove the integer encoding variable and add a new binary variable for each unique integer value. A one-hot encoding is suitable for categorical variables where there is no evident relationship between categories. By default, the encoder will output data with a sparse representation, which is efficient given that most values are “0” in the encoded representation (Rodríguez et al., 2018). For each unique value in the Genres field, a one-hot encoding will generate a new variable, like Genres\_Action. It will be either 1 or 0 depending on whether its genre is “Action”. After encoding in this way, all categorical variables are digitized, and the data become sparser.

# **Methods**

The aim of this project is to predict an application's rating based on a given attribute. We tested three algorithms: linear regression, random forest, and K-nearest neighbors.

## **3.1.1 Baseline Model**

We use linear regression as the baseline model. Linear regression was developed in the field of statistics and has been studied as a model to understand the relationship between input numerical variables and output numerical variables. It has been borrowed by machine learning. This approach treats the data as a matrix and uses linear algebraic operations to estimate the best values for the coefficients. This means that all the data must be available, and there must be enough memory to fit the data and perform matrix operations (Uyanık & Güler, 2013). When dealing with complex data, this algorithm is not the best choice. Therefore, we use it as our baseline model.

## **3.1.2 Other Models**

## **Random forest** utilizes ensemble learning, a technique that aggregates many models to provide predictions to complicated problems. The “forests”, generated by random forest algorithms, are trained by bagging or bootstrap aggregating. It makes predictions by taking the average or average of the outputs of various trees, so increasing the number of trees will improve the accuracy of the results. Compared with the decision tree, it reduces the overfitting of the dataset and improves the accuracy (Biau & Scornet, 2016).

**K-nearest neighbors** algorithm (KNN) is a non-parametric classification method in which the function is approximated only locally and all calculations are deferred until the function evaluation. It can also be used for regression problems. Since the algorithm is based on the distance for prediction, normalization of the data can significantly improve its accuracy if the variables depict different physical units or have distinct scales (Mani & Zhang, 2003).

## **3.2 Parameter Tuning**

Our work shows that when the number of estimators of the random forest model increases within a certain range (25 to 175), the score of prediction will increase slightly by up to 0.4%.

## **3.3 Model Evaluation**

Using the optimized models, the results obtained are shown in Fig. 2

| Metrics  \Model | **MAE** | **MSE** | **RMSE** |
| --- | --- | --- | --- |
| Linear Regression | 0.2828 | 0.2037 | 0.4513 |
| Random Forest | 0.2564 | 0.1837 | 0.4288 |
| K-Nearest Neighbor | 0.3011 | 0.2237 | 0.4729 |

Fig. 2: Three evaluation methods for models

# **Results**

The evaluation of the models proved that after tuning the parameters, the random forest algorithm performed best on accuracy. Linear regression model yields a similar result but is at a disadvantage.

In terms of feature importance, the two better models give different answers. In linear regression, reviews and days since last update play an important role, and some genres also show strong predictive power, such as events, business, as well as food and drink. In the random forest model, the top five important features are reviews, installs, days since last updated, size, and price. The importance of price is 0.02199, which is much less important than size 0.1068.

The intersection of these two models is reviews, and the difference in the evaluation of the importance of reviews indicates that reviews are important but not decisive for a user to rate an app. The random forest model indicates that reviews, installs, and updates have a great influence on user reviews, and most paid apps are designed and developed to cater to specific functionalities and hence are not bulky.

# **Discussion and Limitation**

# Based on the previous analysis, we found that on this dataset, random forest achieved the most accurate prediction effect.

# What kinds of apps can get higher ratings? To answer this, we further inspected the models. It is important for app developers to control the category of apps and the size of the packages (especially for paid apps). For example, they need to reduce their size to attract enough users. Also, their pricing should not be too exaggerated (say, more than $30). In addition, if the app’s rating is low, the effect of continuing to release updates is limited (Ruiz et al., 2015).

# We believe that the next step can include analyzing the user's reviews of the apps, which requires natural language processing (NLP) technology, such as identifying whether a user’s review is positive or negative. This points out the direction of future work.

# **References**

Biau, G., & Scornet, E. (2016). A random

forest-guided tour. *Test*, *25*(2), 197-227.

Mani, I., & Zhang, I. (2003, August). kNN

approach to unbalanced data distributions: a case study involving information extraction. In *Proc. of workshop on learning from imbalanced datasets* (Vol. 126). United States: ICML.

Ruiz, I. J. M., Nagappan, M., Adams, B.,

Berger, T., Dienst, S., & Hassan, A. E. (2015). Examining the rating system used in mobile app stores. *Ieee Software*, *33*(6), 86-92.

Rodríguez, P., Bautista, M. A., Gonzalez, J.,

& Escalera, S. (2018). Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, *75*, 21-31.

Uyanık, G. K., & Güler, N. (2013). A study

on multiple linear regression analysis. *Procedia-Social and Behavioral Sciences*, *106*, 234-240