**Dataset Link :** <https://www.kaggle.com/lava18/google-play-store-apps>

**Publish Date:** April , 2018

**Data Type:** Web scraped data of 10k Play Store apps for analyzing the Android market.

**Prediction Models:** App ‘Rating’ prediction utilizing Neural Network, Decision Tree, Gradient Boosted Decision trees, and Linear regression models

**Parameters:** The models are compared based on mean absolute error (MAE), Mean square Error(MSE), Root mean square error (RMSE), on test data set.

**1.1 INTRODUCTION**

With over 3.5 Million applications on it, the Google Play Store boasts of being the most widely utilized location for application downloads. The App making business is highly competitive and any actionable insights into the factors promoting an Apps’s success is vital.

The sample dataset on Kaggle is a subset of the google playstore . It contains information about 10,000 applications across 33 broad categories like gaming, productivity, family, etc. The data provides information about the respective sizes, downloads, reviews, category etc. of the app. Upon initial analysis it becomes clear that some of these factors definitely have a role to play in an app’s rating and performance.

User Rating was narrowed down to be the focus target of this project. The main reason underlying this selection was that the average user rating can be perceived as a reflection of the general sentiment towards the app. This almost eventually materializes into Installs and increased usage. Additionally, App store ratings are crucial to discovery, downloads, and in-app purchases. They’re an integral part of the search ranking algorithm in both Apple’s App Store and Android’s Play Store. (<https://testlio.com/blog/app-ratings-matter-increase/>) Lower ranked Applications are usually penalized and removed from the store.

This project done on this dataset utilizes 4 different prediction models to predict user rating. These are Neural Networks, Decision Trees, Gradient Boosted Decision trees, and Linear regression models. Prior to running the prediction models the important task of data cleaning & detecting outliers was performed. This was followed by a few data visualization insights to better understand the data.

**1.2 QUESTIONS AND FINDINGS**

How many categories of applications are covered in this Dataset?

What is a good model to predict User Rating?

Which parameters affect the ratings the most?

**1.3 DATA DESCRIPTION**

Google Play Store uses sophisticated modern-day techniques (like dynamic page load) using JQuery making scraping more challenging. Each app (row) has values for category, rating, size, and more. This information is scraped from the Google Play Store.

The different data fields in the dataset are as follows:

**App:** The name of the application

**Category:** The category to which the app belongs

**Rating:** User rating of the app when scraping was done

**Reviews:** Number of user reviews for the app

**Size:** Size of the app

**Installs:** Number of user downloads/installs for the app

**Type:** A binary variable which denotes whether a

**Price:** The cost of the app in US Dollars

**Content Rating:** Age group the app is targeted at - Children / Mature 21+ / Adult

**Genres:** An app can belong to multiple genres (apart from its main category). For e.g., a musical family game will belong to Music, Game, Family genres.

**Last Updated:** Date when the app was last updated on Play Store

**Current Version:** Current version of the app available on Play Store

**Android Version:** Min required Android version

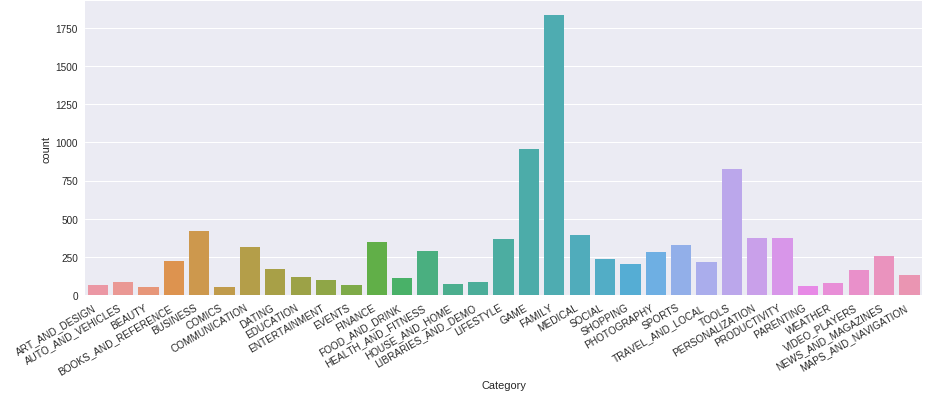
In total there are 13 attributes and 10841 Records in the original raw dataset. Each row corresponds to a single application and its performance across the 13 parameters. The 12 other attributes were used to predict “Rating” which is the user rating, where 5.0 is the maximum and 0 corresponds to the minimum user rating for the app.

**1.4 Data Preparation and Exploratory Analysis**

The first step in this process was to detect duplicate rows and extreme records with missing values. These create problems during prediction and hence were removed.

Upon removing duplicates and outliers using the ‘ drop\_duplicates’ function and removing a single incomplete record the total number of clean records in the data set was 9659. This indicated that approximatel 10% of the raw data set was duplicated and had to be removed.

To get an overall picture of the dataset and how the different apps are present across different categories a categorical count plot was created using the python seaborn package. The image is as below:

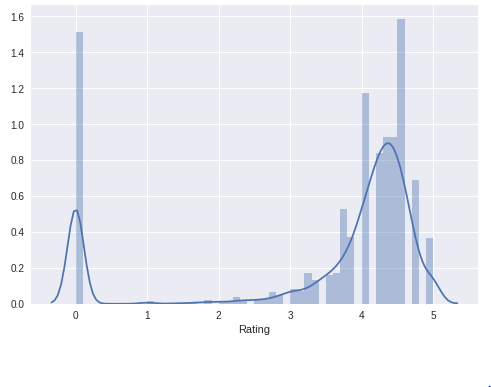


From the above analysis it was seen that the top 10 categories that the apps belonged to were as follows:



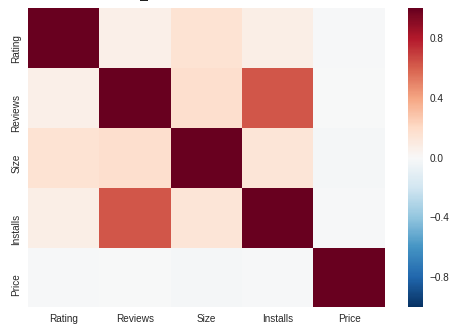
The top three categories Family, Game, and Tools nearly accounted for more than 30% of the data.

The next visualization was to get an idea of how the prediction variable ‘Rating’ was distributed across the data. This was found by creating a distribution plot using the seaborn package.



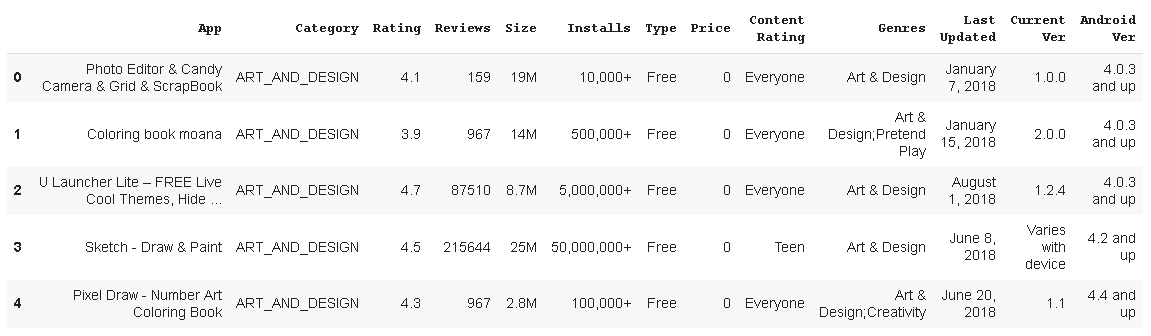
The above analysis yielded the insight that apart from the 0 rated app’s the distribution was a type of a normal distribution from 3.0 to 5.0 with the average peak being somewhere close to 4.5.

A correlation matrix was then created to find the relationship between the numerical variables namely Rating, Reviews, Size, Installs, and Price. This correlation plot can be seen below:



An interesting point that could be inferred from the above analysis was that the variables Installs and Reviews are fairly correlated which makes sense as because the usage of the app increases the number of reviews also increases.

Preparation involves the proper selection and conversion of the predictor variables into numerical values which facilitates much more accurate predictions. The head of the dataset will give a better indication of this.



The initial categories were first looked at from a logical standpoint to judge whether they are useful predictors. This procedure yielded the removal of the categories app’s name, genres, last updated and, current version.

So the main predictors were of two types:

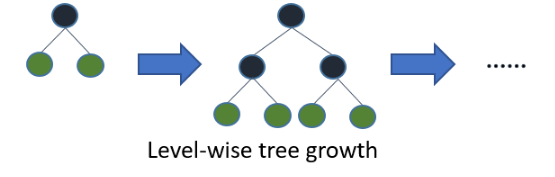
* **Categorical Predictors:** Category, Type, Content Rating, Android Ver
* **Numerical Predictors:** Reviews, Size, Installs, Price

Creating dummy variables was the next step to create binary categories from the 4 categorical predictors. This increased the total predictor variables to 76. Minor changes were made to convert the numerical predictors to purely numerical columns. These involved removing the ‘M’ from the size column, the ‘+’ sign from the Installs category and a complete conversion of these categories to the float data type. Similarly the prediction variable ‘Rating’ was also converted to the float format.

**2. Prediction Models**

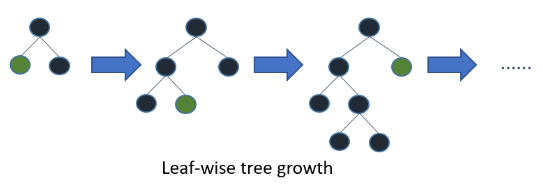
Four powerful and widely used machine learning models were implemented to make ‘Rating’ predictions for the various apps. They are as follows:

* **Multiple Linear Regression:** Multiple linear regression tries to model the relationship between the predictor variables and the response variable ‘Rating’ by fitting a linear equation to the observed data. (<http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm>)
* **Decision Tree Regression:** It decomposes a dataset into a series of smaller subsets while simultaneously developing an associated decision tree incrementally. The final result is a tree with decision nodes and leaf nodes which grows level wise. (<https://www.saedsayad.com/decision_tree_reg.htm>)



*(source: analyticsvidhya.com, 2017)*

* **Deep Learning Neural Network:** A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs. (<http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html>)
* **Light Gradient Boosted Model (LightGBM):** Light GBM is a high-performance gradient boosting framework based on classical decision trees. The model splits the tree leaf wise instead of splitting trees depth wise or level wise as implemented in other boosting algorithms.



*(source: analyticsvidhya.com, 2017)*

**2.1 Model Creation and Implementation of regression models**

In creating the model the most basic step to split the data into 4 different and random sets is vital. The data was split into the following :

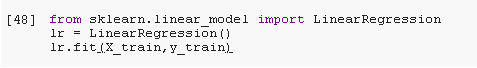
* **X\_train** - Predictor Variables for training
* **y\_train** - Target variable for training
* **X\_test** - Known predictor Variables for testing
* **y\_test** - Target variable for testing

To summarize the original dataset was split into a training dataset and testing dataset using a 70% - 30% split. The train\_test\_split functionality from the sklearn library was used to perform this action. The small code snippet to perform this split can be seen below

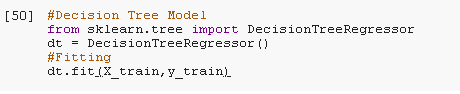
**

From the sklearn library the two regressors LinearRegression, and Decision Tree Regressor were imported and the data was fit into these regressors. Two code snippets showing this process can be seen below:

* Multiple Linear Regression:

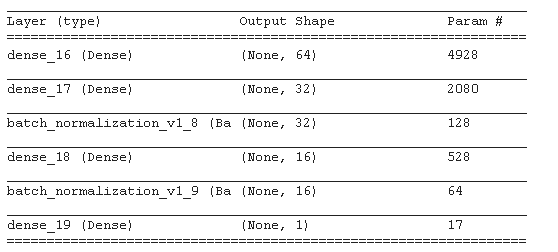


* Decision Tree:

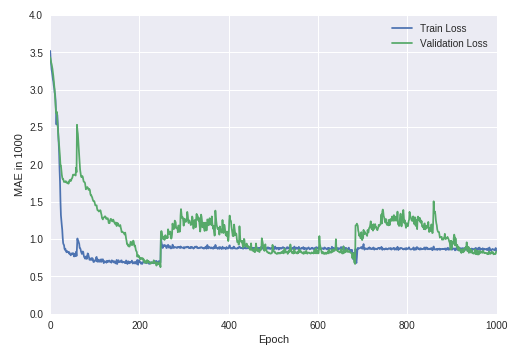


**2.2 Deep Learning Neural Network**

The Keras API was used to create a neural network model with 3 Dense input layers and 1 output layer with 64, 32,16, and 1 nodes respectively. Two Batch normalisation layers were inserted to increase the chances of smoothening of the error curve. The Adam optimizer was used in compiling the model .The model creation and its layers can be seen below:



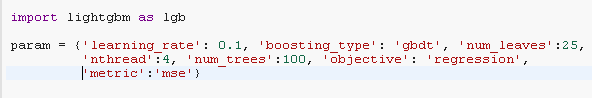
The model was run for 1000 iterations(epochs) which shows how the neural network gradually learns to predict the ‘Rating’ eventually predicting with a low mean absolute error of 0.8. The following graph shows this :



**2.3 Light Gradient Boosted Model (LightGBM)**

The model which predicted the best was the Light Gradient Boosted Model (LightGBM) model. Considering the model’s efficiency and speed the Light GBM was implemented on the dataset. It requires and mostly relies on the correct parameter tuning of the learning rate and the number of leaves.

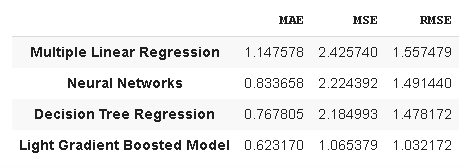
The model is structured as below:



Gridsearchcv method was used to find the optimal parameters for learning rate and number of leaves. LGBM provided the least error parameters with a learning rate of 0.1 and number of leaves at 25.

**2.4 Model Comparison**

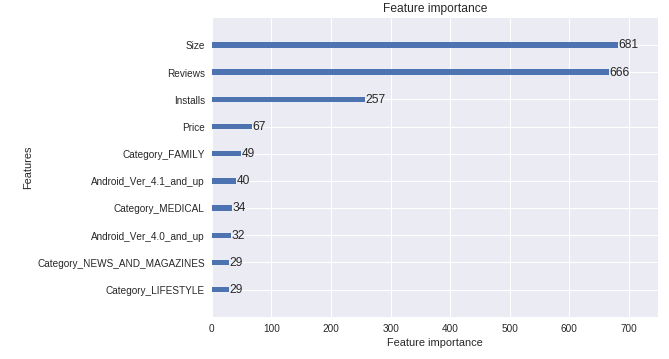
All the four above models were compared based on three parameters MAE(Mean Absolute Error), MSE(Mean Squared Error), RMSE(Root mean Squared Error). The comparison table can be found below:



The Light Gradient Boosted Model provided the best results with MAE, MSE, and RMSE values as 0.62, 1.06, and 1.03 respectively.

**3. Conclusion and further improvements**

The LightGBM model can be an excellent starting point to predict ratings for additional apps given the categories used in the model. To further analyze the insights provided the model the feature importance function of the model was implemented. This gave an interesting insight on which parameters helped the model to predict rating with low errors. This can be better understood by the following figure



The above analysis shows that from the current dataset it can be inferred that the variables Size, and Reviews are affecting the predictions the most.

This capstone project thus gives a broad overview of how the data set was distributed across 33 categories of apps. It also shows how the LightGBM model predicted the ‘Rating’ variable with very low errors. The predictions for ratings can be further improved by gathering more data and arriving at more variables than currently available. Furthermore normalizing the entire predictor variable dataset can also help in arriving at more accurate predictions.

The approach adopted for this project is extremely flexible and can be scaled to accommodate huge datasets and many more predictor variables. Utilizing the methods in the project can thus predict user ratings beforehand, and vitally help in achieving a good competitive advantage in the tough Google Play Store market space.