NYU | TANDON SCHOOL OF ENGINEERING

Play Store Trends and Predictions

Big Data Programming Fall 2019

Sarth Desai (sjd445)
Crystal Cheung (cc5680)

Table of Contents

Introduction 3
Motivation 3
Solution 3
Technology and Architecture 3
ETL Process 4
Visualizations 5
Insights 10
Results 11
Future Scope
14
References 14

Introduction

There are 3.3 billion smartphone users in the world.[1] With that, comes great opportunity to capitalize on mobile apps. There are 250 million downloads per day and it is only increasing.[1] Last year, $39.7 billion dollars were made in revenue from apps worldwide. To put it into perspective, if an app is able to generate 1 million downloads, the creators should earn about $10,000 - $15,000 per month.[1]

With this opportunity, there is also competition. There are 2,714,499 apps in the Google Play Store, 3739 apps are added daily,[1] and app discovery is at its lowest. Users in the US only use about 20.1 apps per month.[2] Chances of being successful in the play store is slim.

Motivation

We wanted to analyze mobile app trends to get some insight on what is popular and successful. Through analysis, hopefully there are concrete results that can help app developers decide where to put their time and energy to produce the most auspicious for their apps, because there have been studies that found that 80% of developers don’t earn enough to turn their development into viable businesses.[3]

Solution

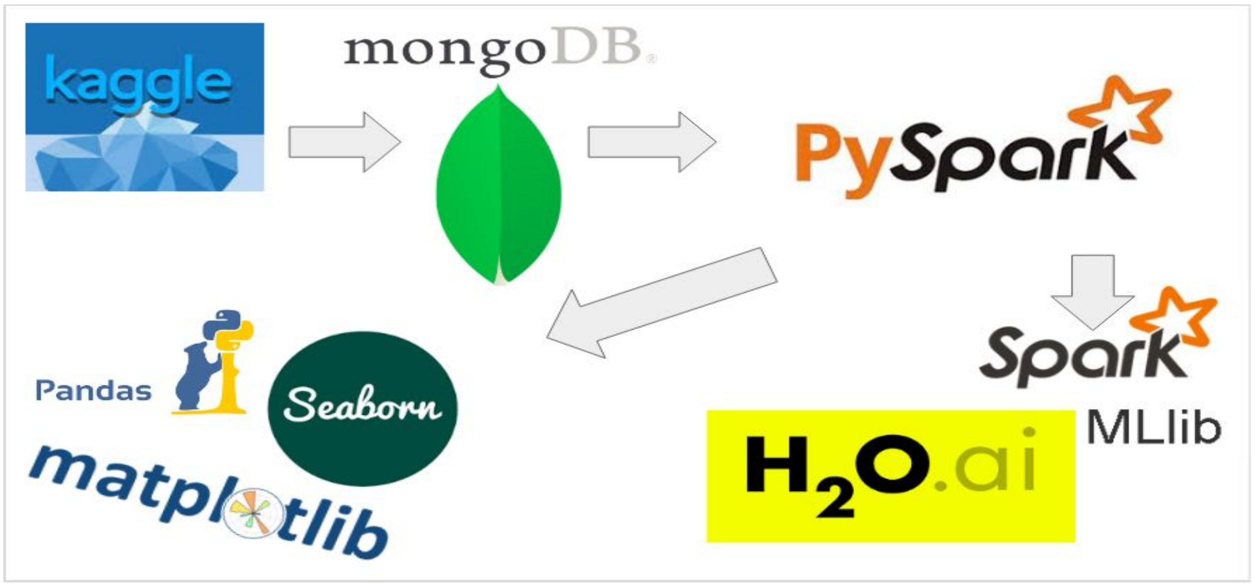
We will find trends in the Google Play Store to discover if there is a correlation between app attributes and install rates. We took in account attributes such as app size, category, number of reviews, price, generes, and content rating. After finding trends within these features, we trained several different machine learning models and used the most accurate one to run our predictions.

Technology and Architecture

A pipeline needed to be implemented because there does not exist one product that can handle the whole process. Our source of data was from Kaggle and it was stored in a csv format. The next step was persistent storage for the data. There are two reasons why we chose to use a NoSQL database rather than load in CSVs to Spark. A NoSQL database provided a structured way of letting data persist permanently, while CSVs is just rows and rows of data points. Also NoSQL databases also allows the output to be structured and stored so computations in Spark do not need to be repeatedly redone just to see the output. The second reason is that using a database, it is possible to constrain the data before loading it into Spark. Even though we did not use this feature, it is always good to know that we have the option to if we ever need/want to in the future. A NoSQL database was chosen over traditional relational databases because they are distributed and efficient.[4] We chose to use MongoDB as out NoSQL database.

We used Apache Spark to do computations on our data because it is a powerful platform that is able to handle big data processing. We decided to use H2O.ai because it is a distributed machine learning platform. We used Spark Machine Learning Library for training out Linear Regression model because we were not able to find it on H2O.ai. Of course all the technology we used are horizontally scalable and distributed, or else it would not be compatible with big data processes. Pandas, Matplotlib, and Seaborn were used to display charts within jupyter notebook once we were certain we have aggregated the data using Spark, to ensure that we did not make those 3 platforms handle big data, which they were not made for.

Fig 1: Architecture

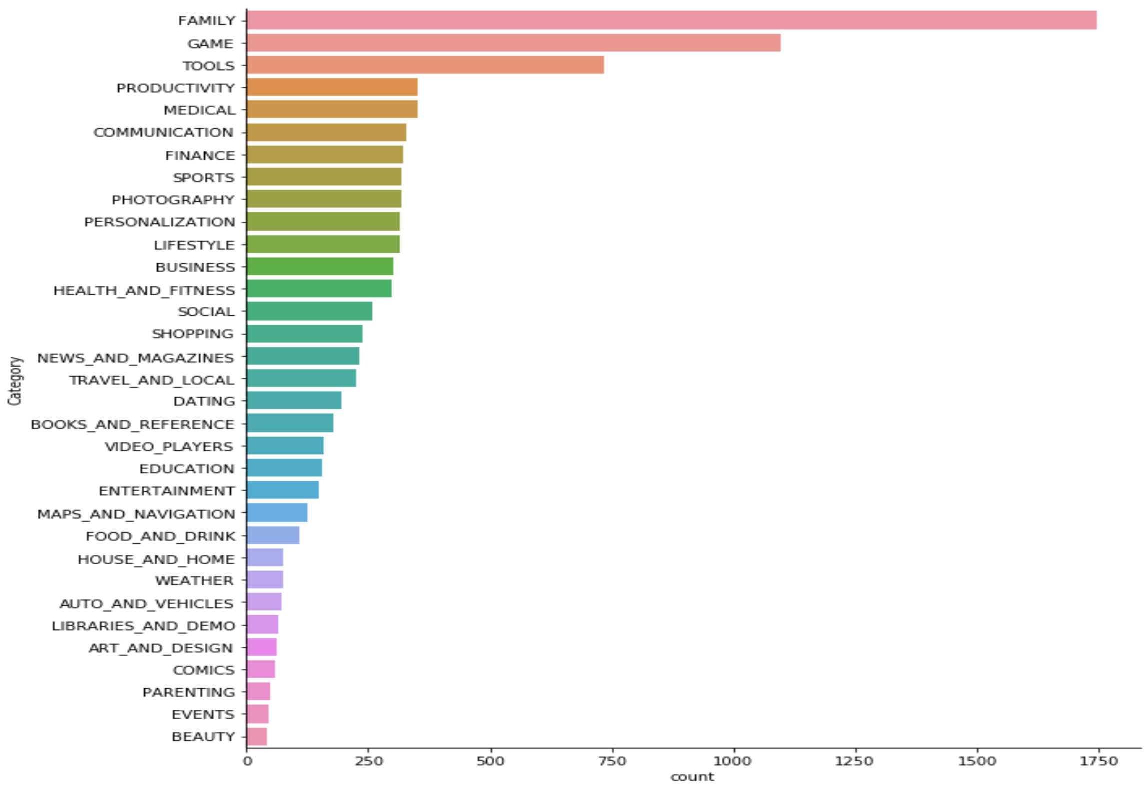


ETL Process

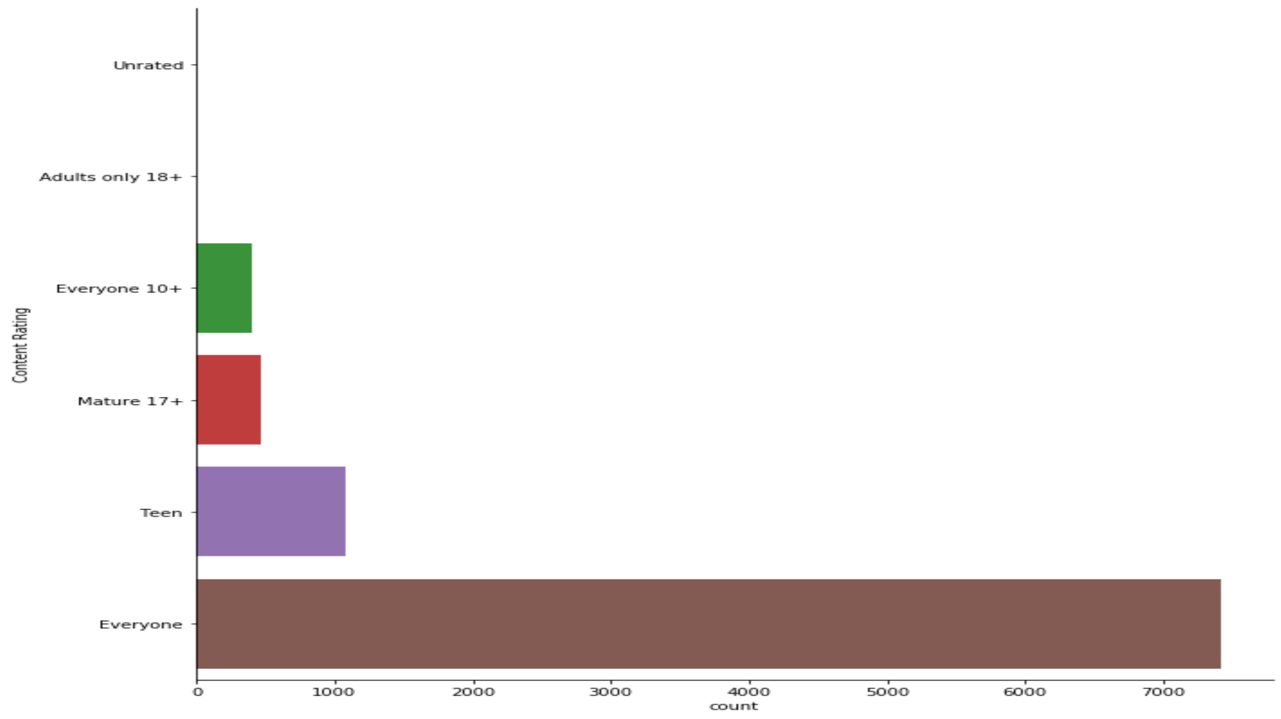
The first step in our process was to extract data from Kaggle, which we then loaded into MongoDB. Using jupyter notebook, we ran Spark and loaded data from the database into Spark. In Spark, we cleaned up the data for analysis by turning columns that were strings into numerical values when suitable. We also took out invalid values such as NULL, “varies by device” in app size, and 19 in ratings when the scale only went up to 5. To clean the data for machine learning, we turned the remaining columns into numerical values even if they were categorical data. We did this by assigning a numeric value to each unique data point in a column. We also had to vectorize the data after cleaning it so the machine learning model could digest it.

Visualization

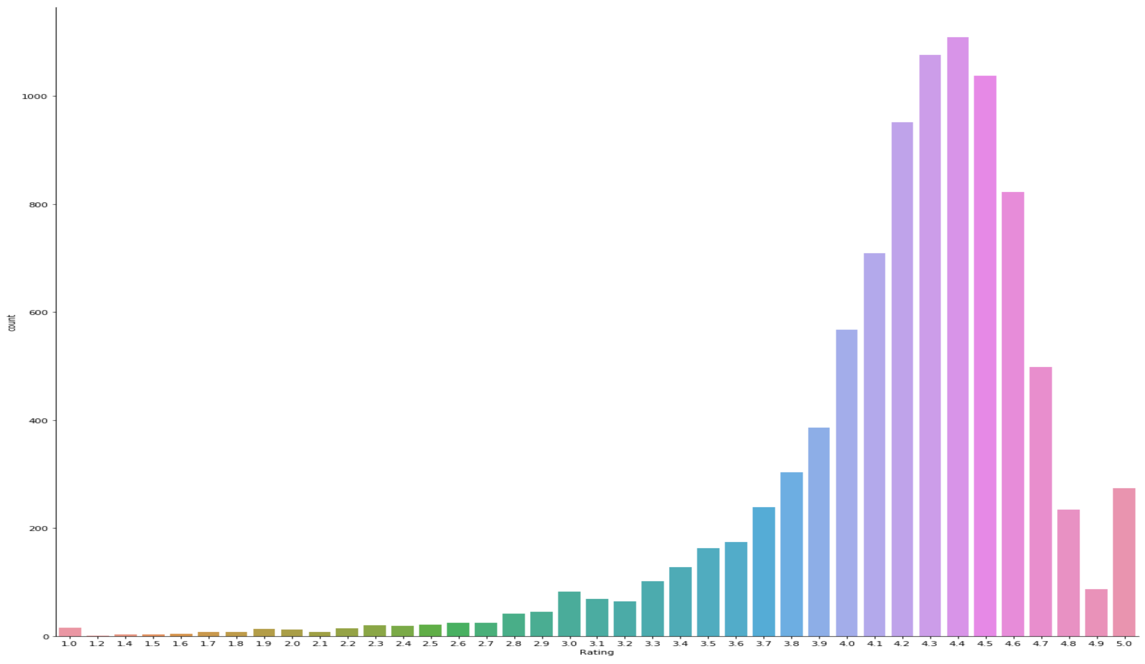
Graph 1: Number of Apps in Each Category



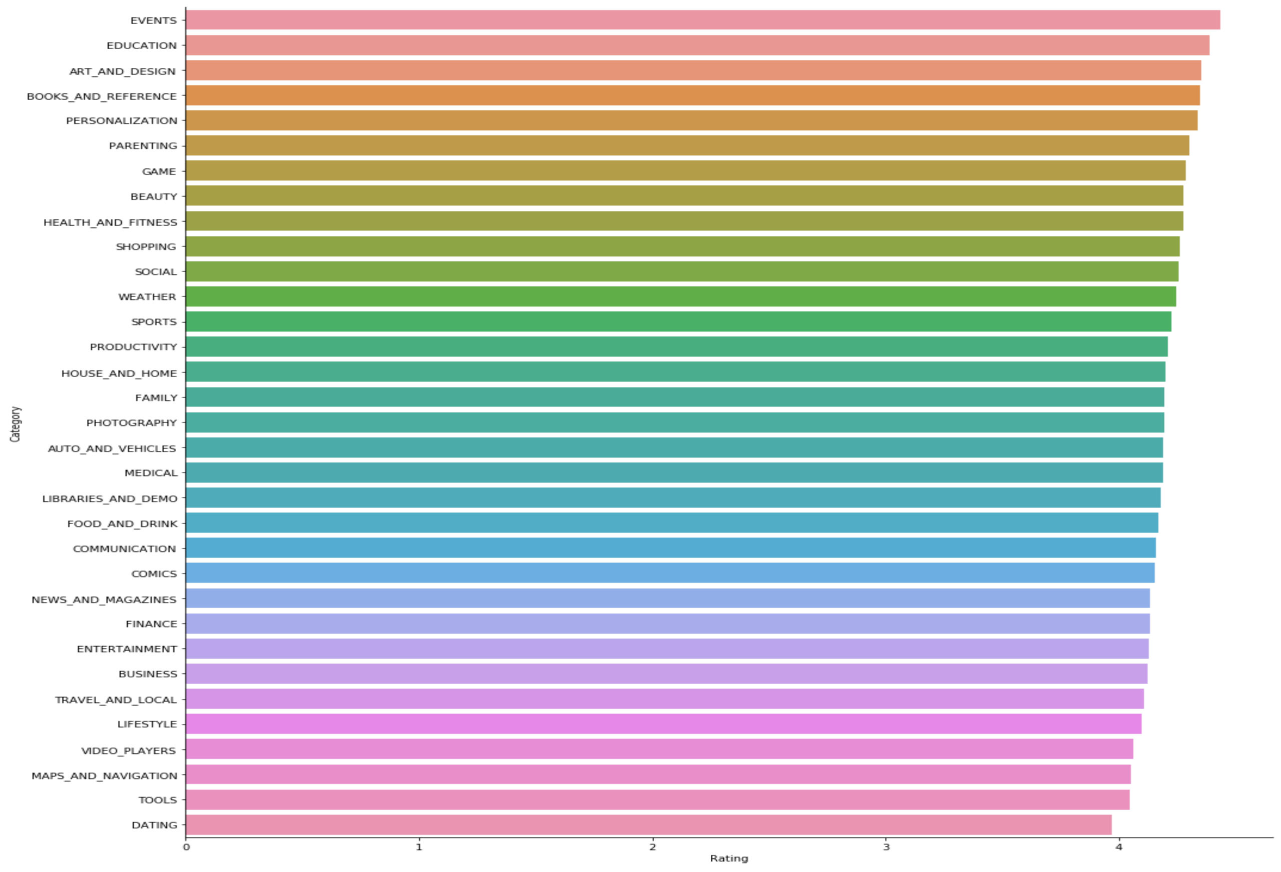
Graph 2: Number of Apps in each Content Rating Group



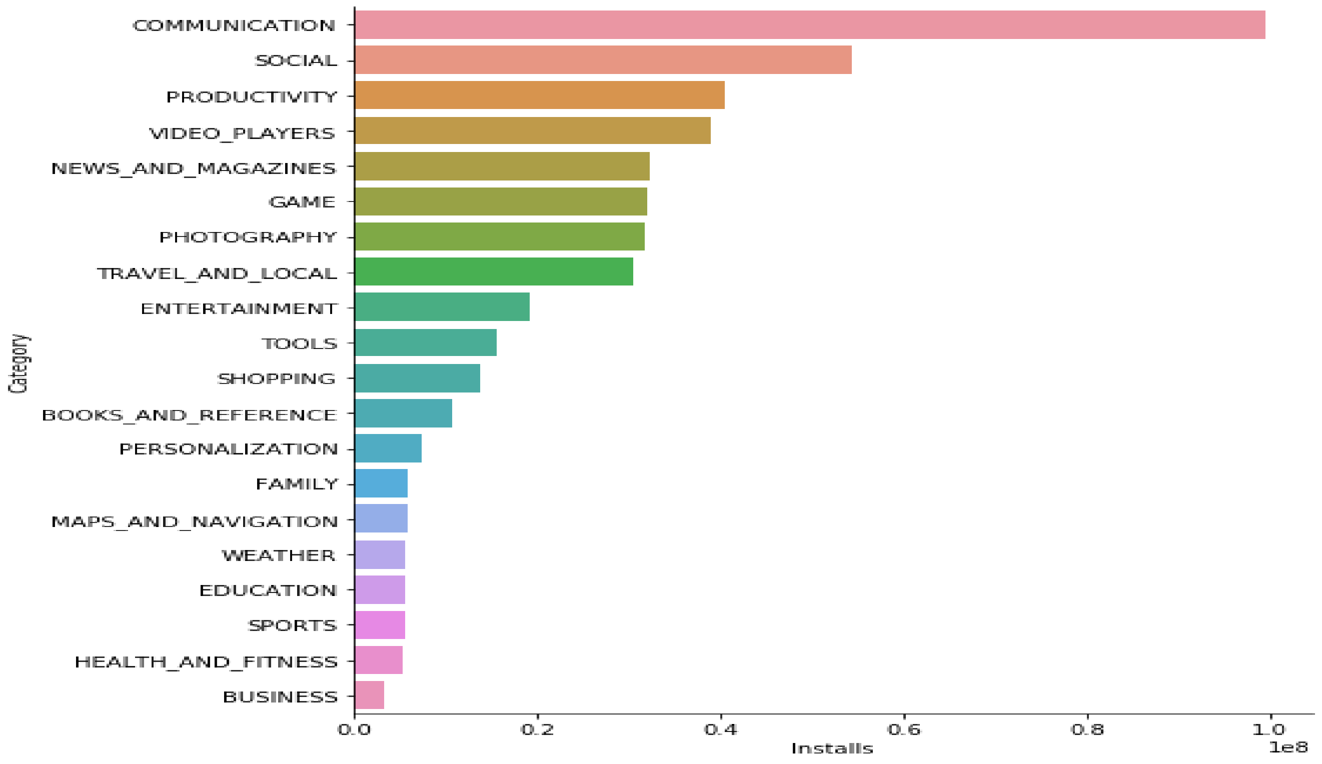
Graph 3: Number of Ratings for Each Rate



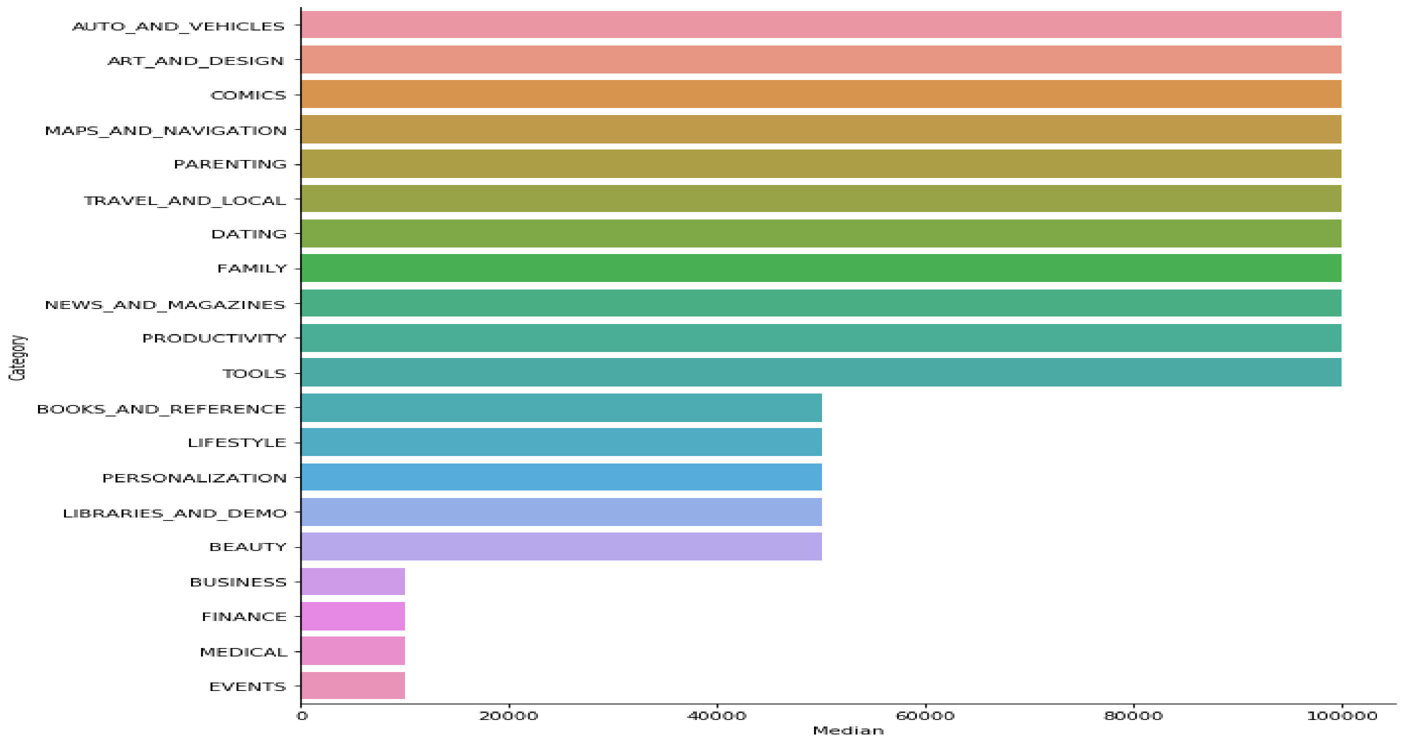
Graph 4: Average Rating for Each Category



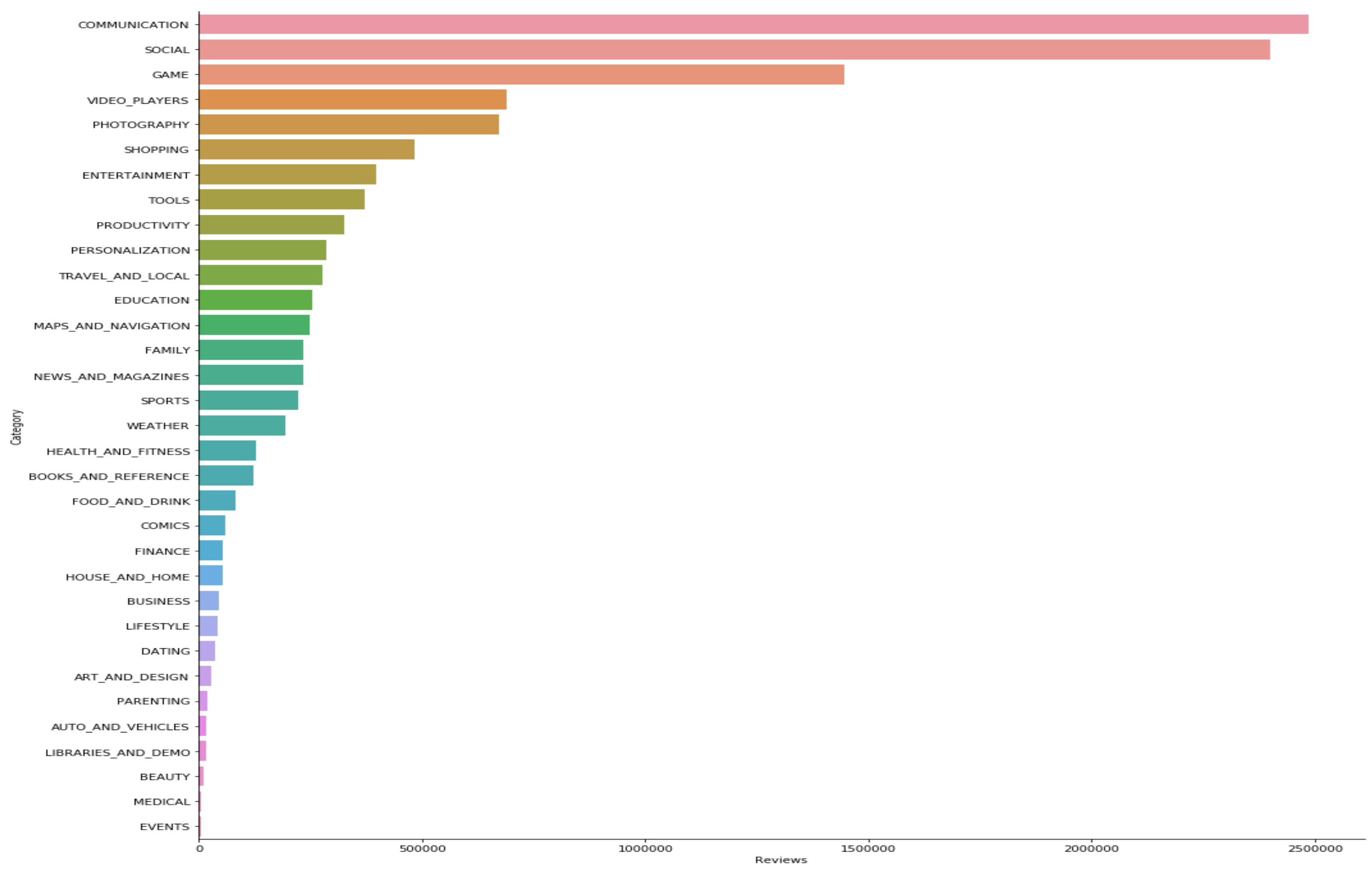
Graph 5: Average Installs in Per Category (Top 20)



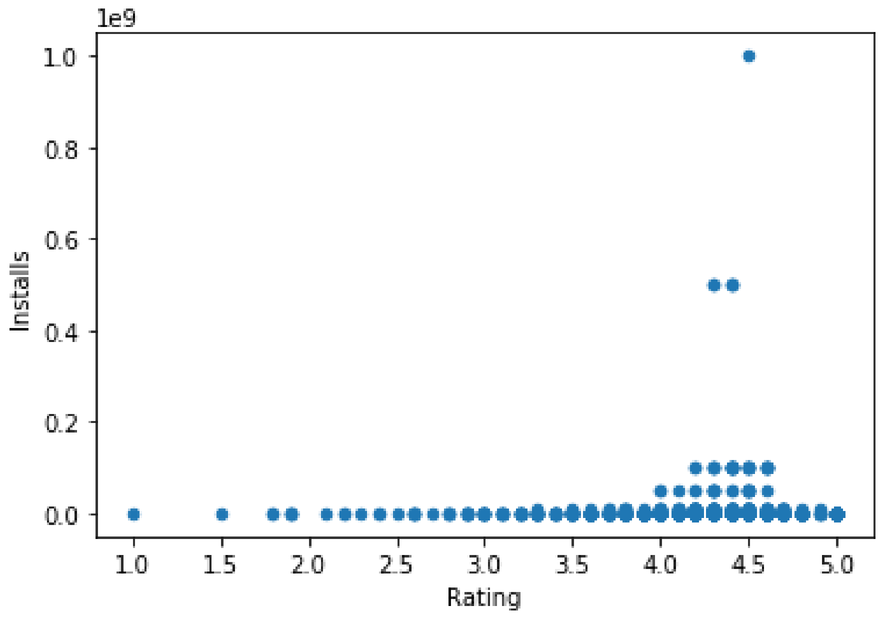
Graph 6: Median Installs Per Category (installs attribute was bucketed)



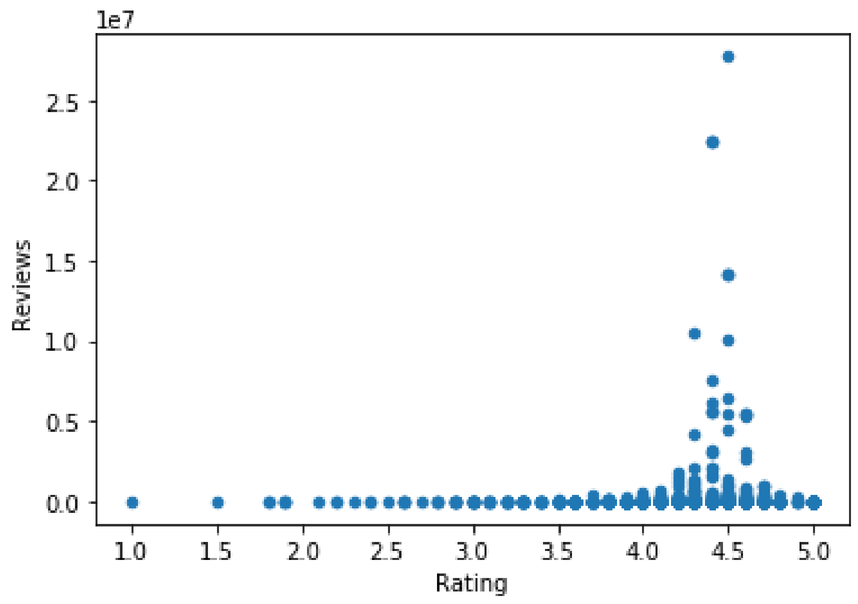
Graph 7: Average Number of Reviews in Each Category



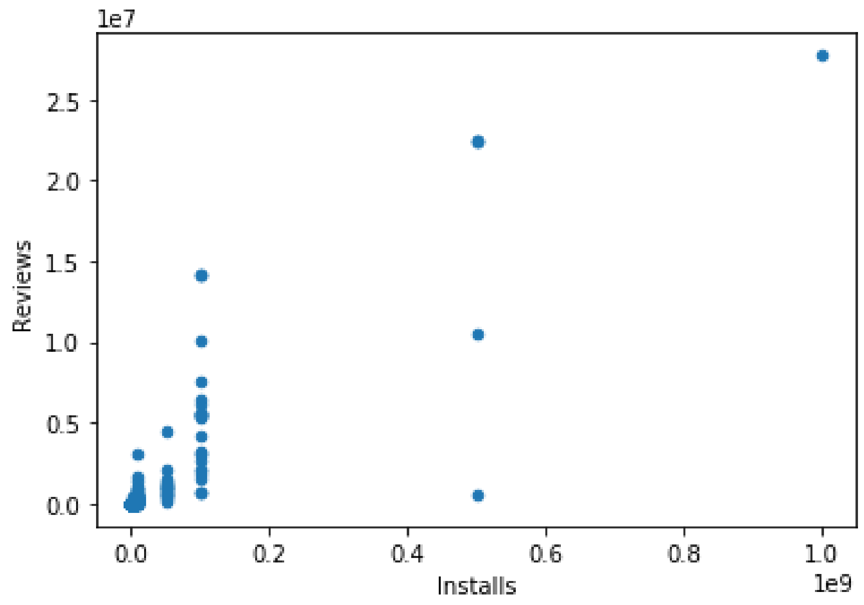
Graph 8: Number of Installs vs Rating from Sampled Data



Graph 9: Number of Reviews vs Rating from Sampled Data



Graph 10: Number of Installs vs Number of Reviews from Sampled Data



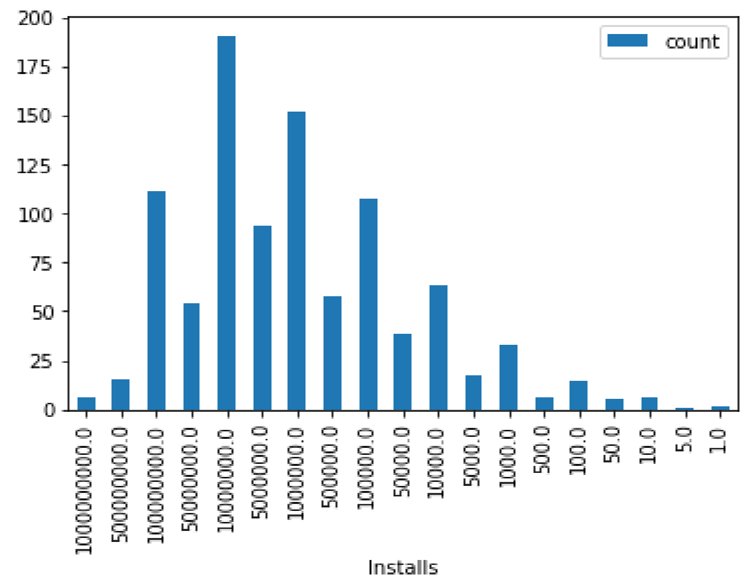
Insights

We decided to base success off of number of installs and ratings. Since ratings and installs had a direct relationship (Graph 8), we decided it would be a good indicator of success as well. Of course not all of our charts provided us with profound insight. But here are some key points we saw.

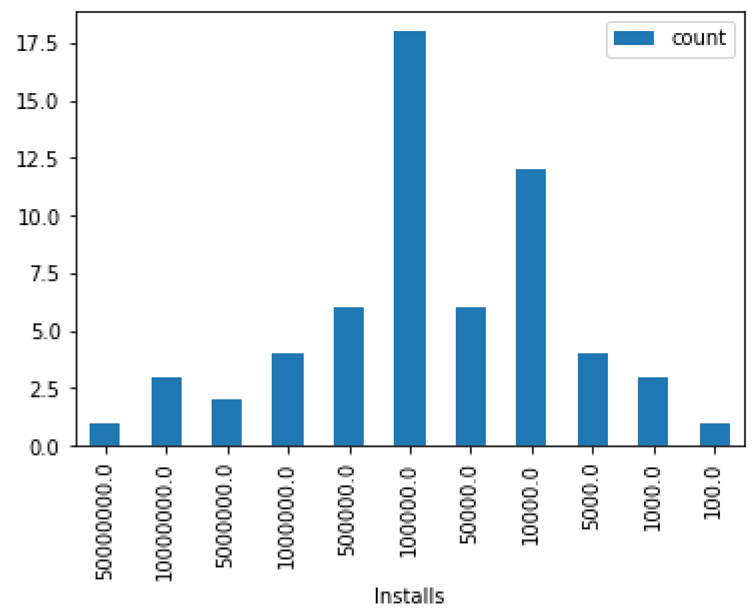
We noticed that (from Graph 5 and Graph 6) the average install rate was much higher than the median for all categories. This means that in all categories, there are a few apps that dominate the market and pull the average up, while most apps are not very successful. The barrier to be successful is high.

We looked at the top 5 categories with the most apps in them (Graph 1) and decided that they were competitive. We compared it with the average downloads by category (Graph 5) and decided that categories that had many apps, yet did not rank as high in the chart with average installs by category was not worthwhile to enter into due to competition. Based on the charts, developers should not enter the Family or Medical category. However, Games and Art and Design were good categories to enter into. For Games, while there is a lot of competition (Graph 11), average install and median install was high, which signals that entering into such a competitive market was worth it. There are few apps in Art and Design (Graph 12), but install rates were comparatively high, so it seems that it is easier to become successful in that category. Perhaps with some more diligent work into what customers actually want, one could disrupt the Art and Design category and become successful.

Graph 11: Installs Distribution within the Games Category



Graph 12: Installs Distribution within Art and Design Category



*Note that the install attribute was bucketed, so the graph is not to scale.

While looking at average ratings over the categories, we considered the possibility that there just weren't good apps in the category if the rating was low, providing another opportunity to disrupt the market with something customers actually want. These categories include: dating, tools, maps and navigation, video players, and lifestyle.
Looking at number of installs vs ratings it is clear that a higher rating correlates with higher installs. Higher ratings also correlate with more reviews.

Results

We implemented three different machine learning models on the dataset to predict the ratings for the applications based on their features.

Table 1: R^2 score and MAE for the machine learning models

ML Model	**R^2 Score**	**Mean Absolute Error**
Linear Regression	0.0133	0.3810
Distributed Random Forest	0.1193	0.3530
Gradient Boosting Machine	0.1503	0.3552

For Linear Regression, we tried two models one for predicting the ratings and the another one for predicting the number of installs for the application. As we can see from the Graph 8, there is a direct relationship between ratings and installs. Our results for predicting installs were not as accurate because the values of the field 'Number of Installs' were quite high (from Graph 5), the mean absolute error was huge and prediction of the number of installs was counterproductive. Since they are correlated the rating would be the next best attribute to measure.

The other model that we implemented is Distributed Random Forest (DRF) using the tool called H2O Driverless AI. It is a powerful classification and regression analysis model.[5] We used the new preprocessed CSV file that we composed in jupyter notebook. And computed DRF for the dataset using 50 trees. Another important reason of using this algorithm is its ability to handle large dataset with higher dimensionality.[6]

Fig 2: Distributed Random Forest score deviance

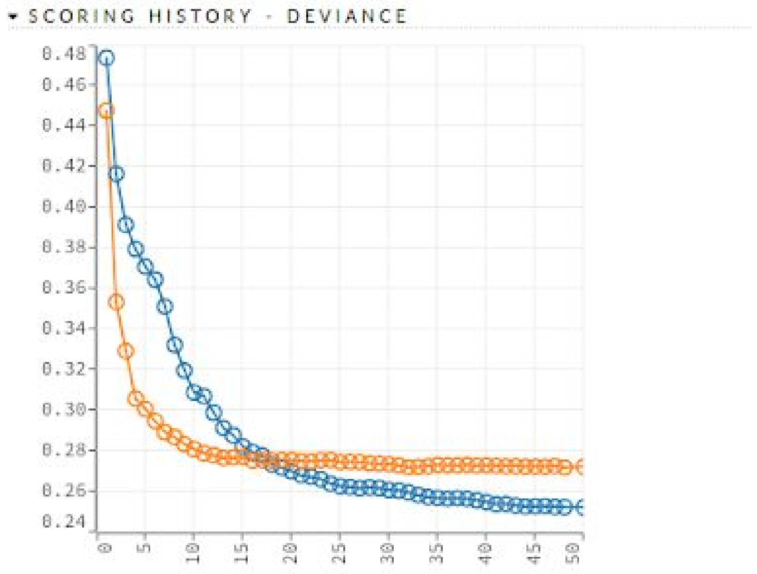
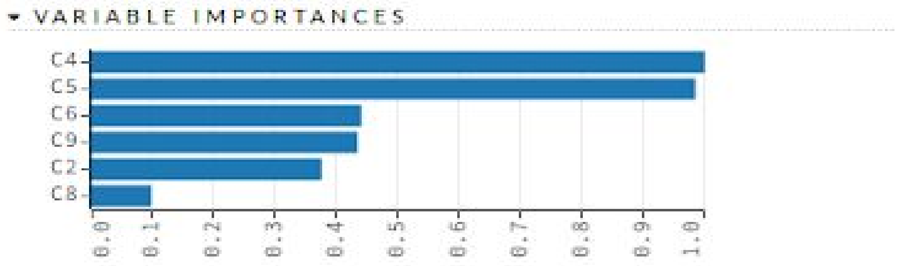


Fig 3: Distributed Random Forest features ranking (refer Table 2)



And the last model is Gradient Boosting Machine (GBM) using Gaussian distribution using the tool called H2O Driverless AI. The guiding heuristic is that good predictive results can be obtained through increasingly refined approximations.[7] This model was computed with 50 trees as the optional parameter and the distribution parameter(i.e. the loss function) was set to Gaussian distribution because the response column(i.e. Ratings) had numeric value.

Fig 4: Gradient Boosting Machine score deviance

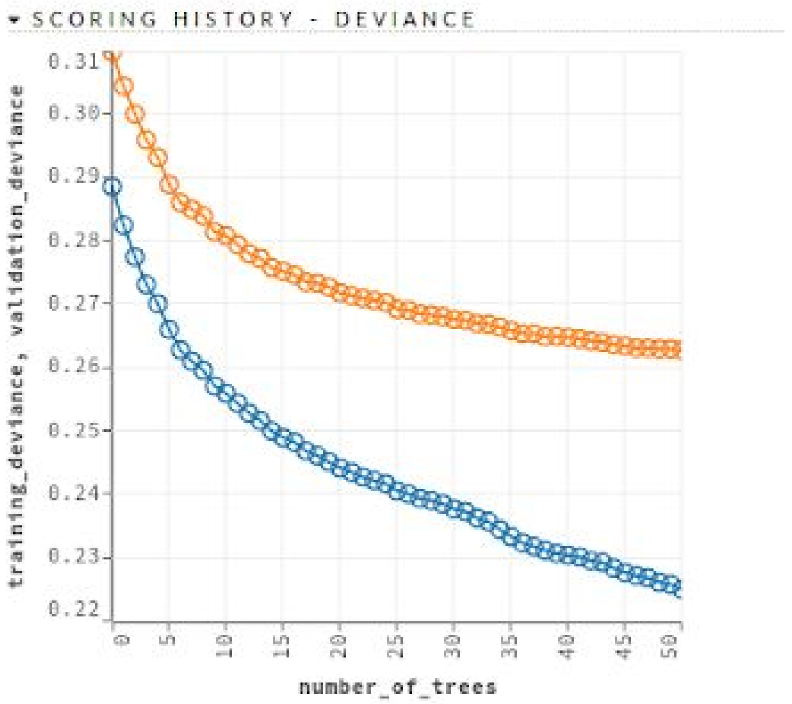


Fig 5: Gradient Boosting Machine features ranking (refer Table 2)

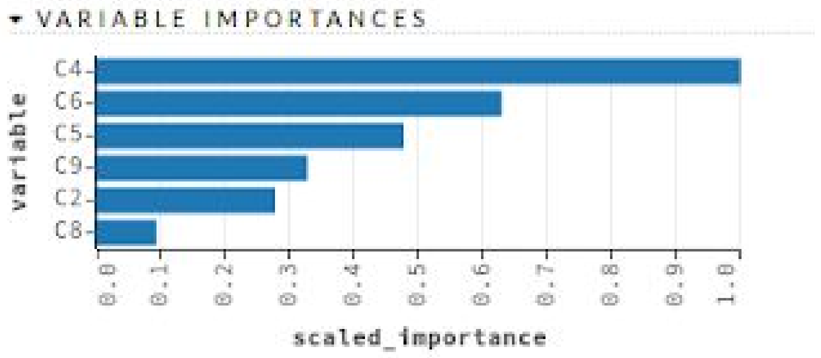


Table 2: Column Labelling

Column_Label	Column Name (as per the Dataset)
C1	App
C2	Category
C3	Rating
C4	Reviews
C5	Size
C6	Installs
C7	Price
C8	Content Rating
C9	Genres

Future Scope

- Hyperparameter optimization for better performance.
- Evaluate and process the models on niche dataset.
  - Splitting data and analyzing by age groups
  - Splitting data and analyzing by the year of the application was published
- Implement Apache Kafka for real-time trend analysis and rating predictions.

References

1. https://appinventiv.com/blog/google-play-store-statistics/
2. https://www.businessofapps.com/data/app-statistics/
3. https://venturebeat.com/2017/09/24/your-chances-of-making-a-successful-mobile-app-are-almost-nil/
4. https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models
5. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html
6. https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706
7. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm.html