Data Mining

Google Play Store Apps

Lecturer:

Dr. Omer Zuk

Serving:

Ron Borower

Itay Danino

Contents

Attached is the data file and the Python notebook

Data_Mining_Google Play_Store_Apps.ipy

googleplaystore.xlsx

Abstract

The project aims to provide app developers with a tool that aims to give a prediction based on a set of data taken from the Kaggle website, through which they can test several parameters measured in apps from the Google App Store, which we finally want to try and repeat how to create a "successful" app, i.e., receive a high amount of installs and also high ratings.

After cleaning up the information, we analyzed the existing information and used the tools we learned in the course and were able to provide several conclusions by which to define what is a "successful" app.

To get a "successful" app, we set minimum installs of over one million downloads and a rating above 4.0.

Using additional columns in the data table, we looked for a certain correlation through which we reached our main conclusion, and indeed at the data analysis stage we were able to show that from a certain number of reviews that app users have made to a particular app, then there is a very high chance that an app will be successful, so we recommend that those app developers emphasize requesting auditing from each user of the app, beyond common highlights such as convenience, clear visuals, free and more.

Introduction

Is it possible to predict the success of an app in the Google Store (Android) that will be reflected in the number of downloads and high ratings?

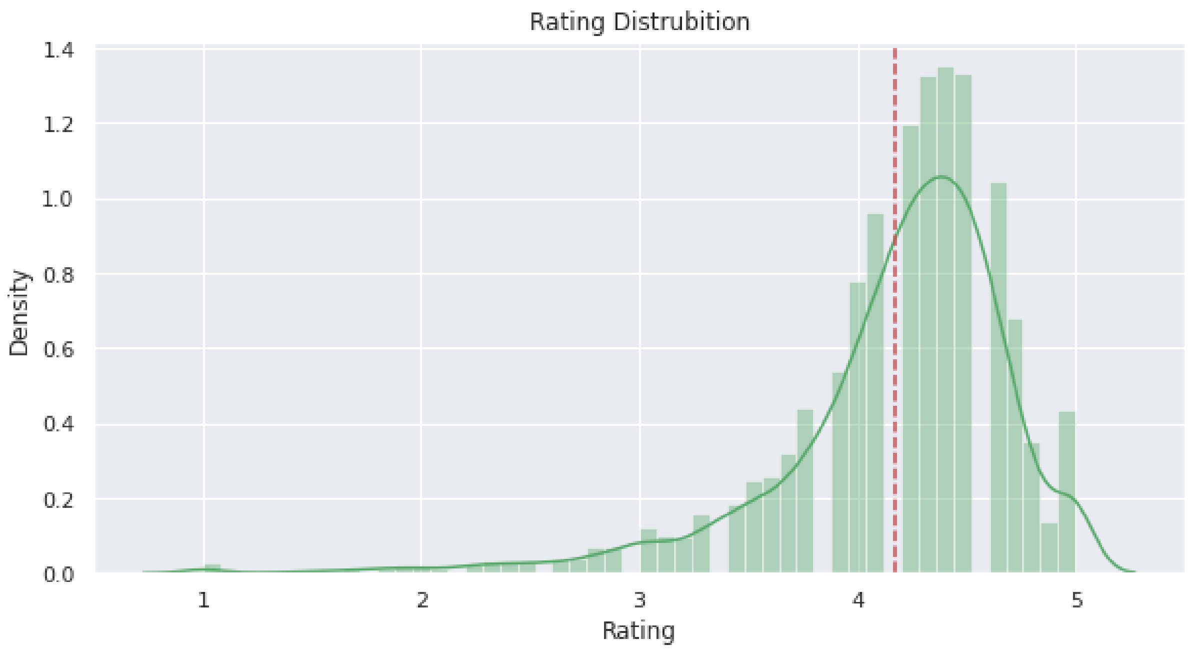
The purpose of this question is to provide app developers with a platform through which they can gain insights and guidance regarding the potential to develop an app that is considered "successful".

A company or group of people who wish to develop an app will do everything possible to reach the highest number of downloads and then establish the status of the app as a "successful" app by getting a high rating from the users.

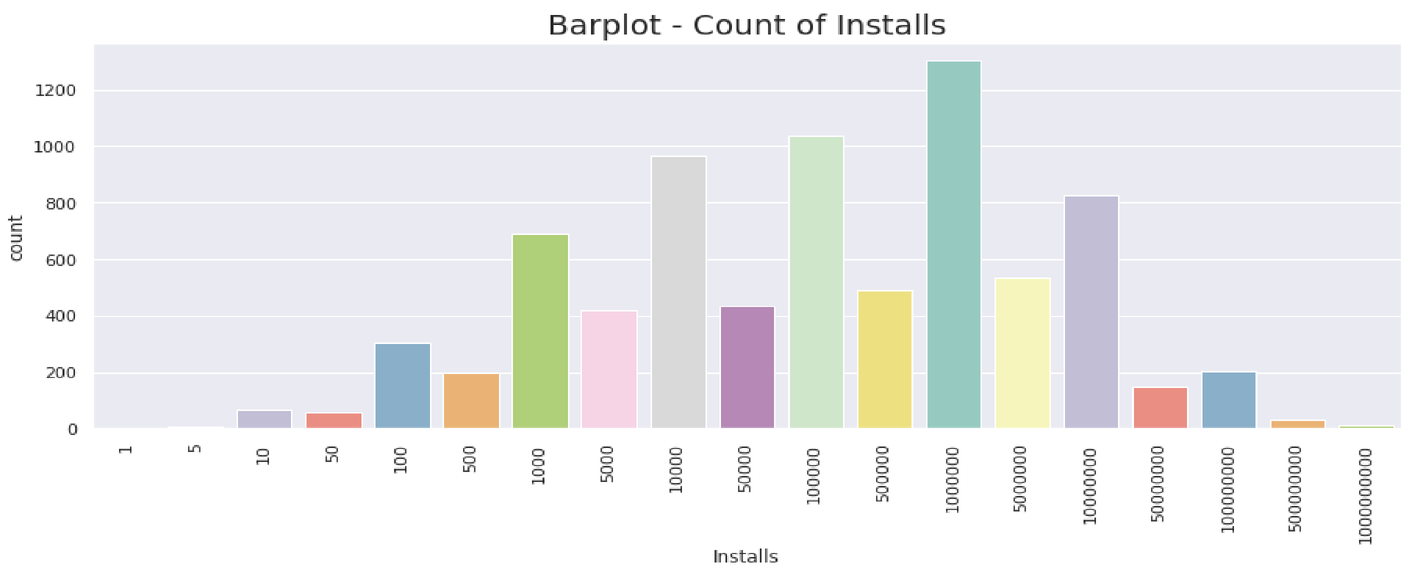
In recent years, we can see an increase in the number of applications in the market, when not necessarily all applications do manage to "harm" the real need of users in terms of several parameters such as need, design, convenience, price, and more, and therefore before developing the app, developers must make a prediction, according to several parameters relevant to that app, according to which they will be able to determine whether the app they wish to develop will succeed.

Methodology

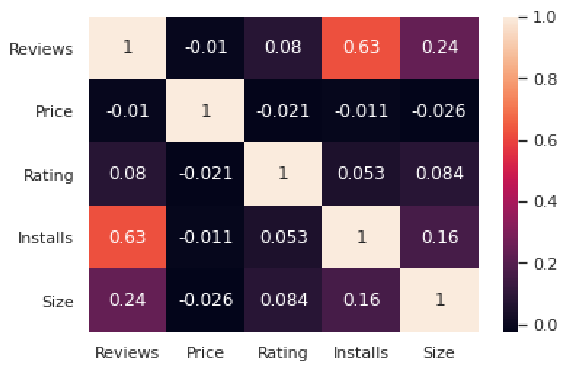
- ❖ Data Cleaning
    - ➢ First, we converted the values from the "Size" column that are not numeric to numeric values and deleted the irrelevant values.
    - ➢ When you view at able, there are 3138 Null values throughout the table, so we will first delete those values, it is not a large amount of data, nor do they represent any time series, so you can delete them.
    - ➢ We will set a target value that will be called "Success", and it depends on two features "Rating" and "Installs"
    - ➢ For the "Installs" column – we downloaded characters and turned the values in the column into numeric values.
    - ➢ The Rating column has only numeric values.
    - ➢ We then set a threshold rating and a minimum amount of downloads to define "success"- a rating above 4.0 and several downloads over a million, and we now want to see if our initial determination is correct against the data available in the table.
    - ➢ In this graph, we can see that the average rating of apps in the Google Store is 4.17 and from here we learn that our initial determination on the subject of ranking made sense.



- In the following graph, we can also clearly see here that our initial hypothesis regarding the number of downloads was good, that is, the highest number of downloads is over one million installs.

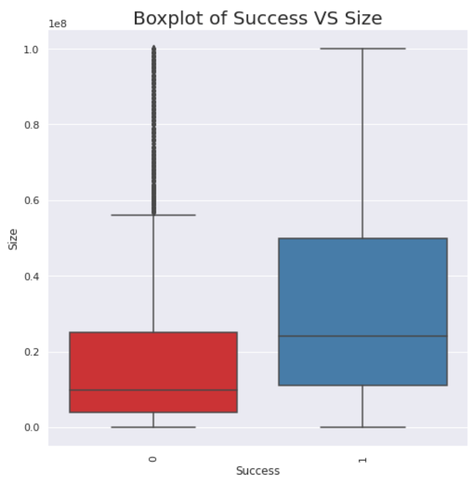


- Before downloading the Installs and Rating columns, we will examine a correlation between these columns and other numeric columns that we have in the table, which we estimate may be some correlation.
- We convert to the "Price" column from categorical arrays to numeric values, for example, we will go from '$18'to'18.
- We will run the Pearson test for features "Rating", "Installs", "Price", "Reviews".
- We will first notice that there is no correlation between the "Rating" and "Installs" columns (0.053).
- After that, we will notice that the Price column also does not affect the Rating and Installs columns, as is the case with the Size column.
- Finally, we will notice that between the "Installs" and "Reviews" column there is a certain correlation (0.63).



	Reviews	Price	Rating	Installs	Size
Reviews	1	-0.01	0.08	0.63	0.24
Price	-0.01	1	-0.021	-0.011	-0.026
Rating	0.08	-0.021	1	0.053	0.084
Installs	0.63	-0.011	0.053	1	0.16
Size	0.24	-0.026	0.084	0.16	1

- Now, we download the "Rating" and "Installs" columns after we have defined a new "Success" column.
- We will look for a correlation between the Target Value and one of the columns. It can be seen through Boxplot that more successful apps then probably have their size higher.



- We will perform a Chi-square test to see a correlation between Success and the other features:

	Variable	Chi squared value	p-value
0	Reviews	7639.556250	0.0
1	Size	1082.290598	0.0
2	Genres	981.990479	0.0
3	Price	263.453084	0.0

It can be seen that when the p-value is less than alpha=0.05, there is a certain correlation between the Target Value and the features.

- Finally, we will download the columns that we have determined are irrelevant (in the dataset).

❖ One Hot

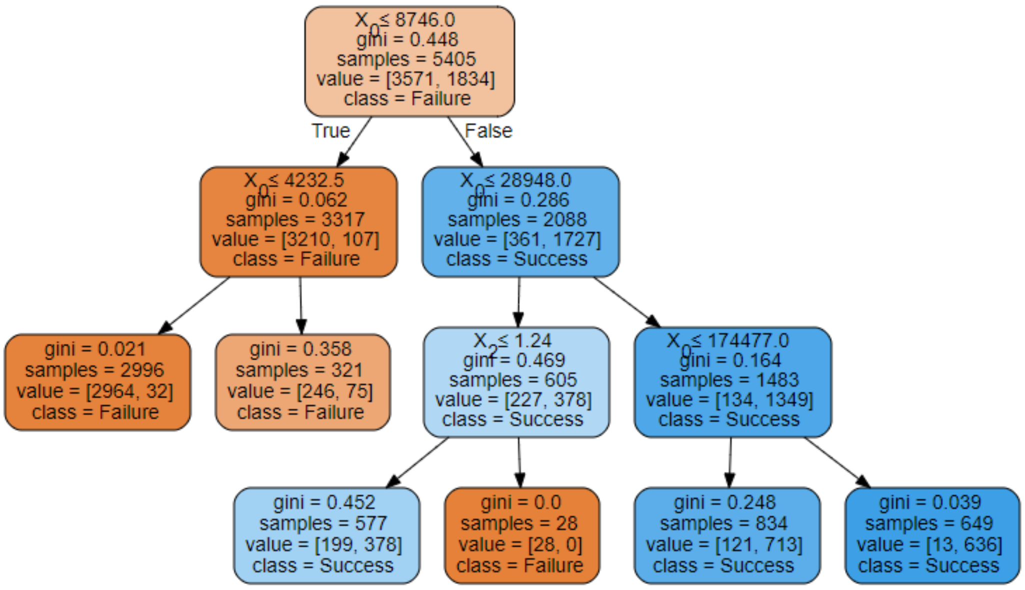
- After not finding a correlation between "Success" and the other columns we examined, we decided to select the "Genres" column where there may be a correlation.
- We convert the categorical column "Genres" to One Hot, to run algorithms.

- Actions before algorithms are run
  - We will set an X variable that will be characterized by a data frame and contain all columns except the Target Value column that is the "Success" column.
  - A variable Y is defined as the Target Value ("Success" column), characterized as a vector that accepts values of 0 and 1.
  - We took 30% of the data for testing, and 70% for training.

Algorithms

❖ Decision Tree

➢ When we activated the algorithm, we received an accuracy of 0.8925 and we wanted to see if we could improve this accuracy, for this, we set alpha=0.001 and max_depth=3, and indeed received an accuracy of 0.9158.



➢ Then, we will look at the Decision Tree and notice that there is one dominant column represented by X0 – "Reviews".

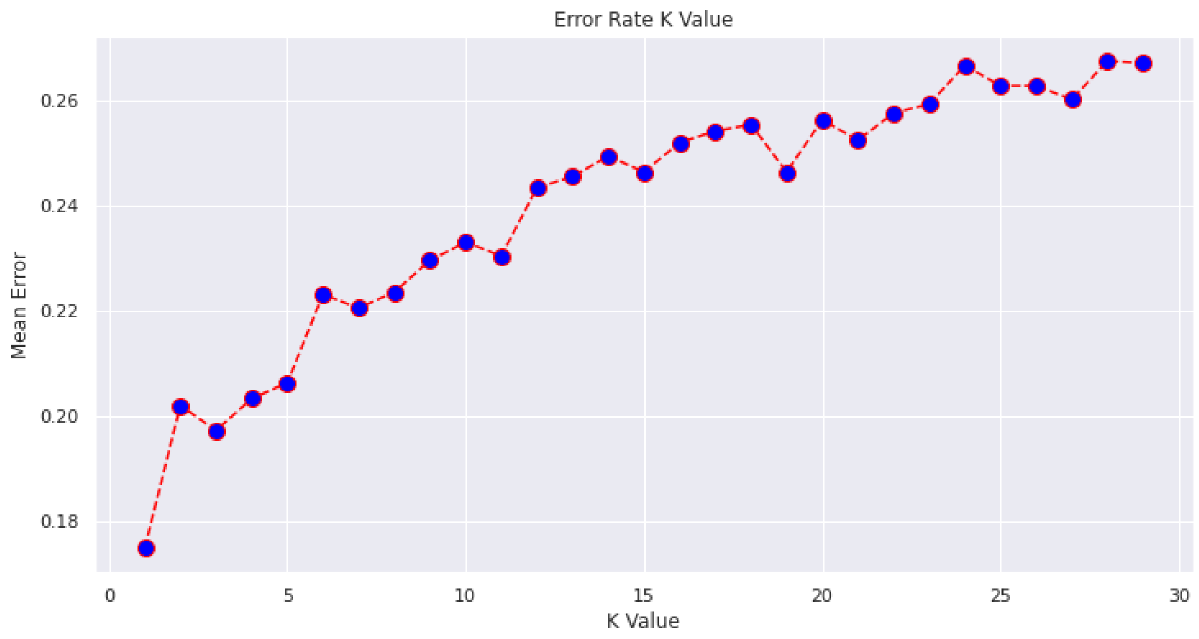
❖ Naive Bayes

➢ After running this algorithm, we received an accuracy = 0.7962.

➢ We assume that the accuracy is lower than the decision tree because there is no dependency between the features.

❖ <u>K-Nearest Neighbors</u>

➢ To run this algorithm, we first normalized the X_train, and X_test values (no data normalization is required because it contains binary values).
➢ After initially running the algorithm, we will notice that the Accuracy = 0.7936.
➢ To improve accuracy, we will look at the error graph and understand the optimal number of neighbors.
➢ Since the classification of this algorithm is binary classification, an odd number of neighbors (K = odd) is selected, ensuring an immediate decision.
➢ After running, we noticed that for one neighbor, we would get the best accuracy: 0.8252.



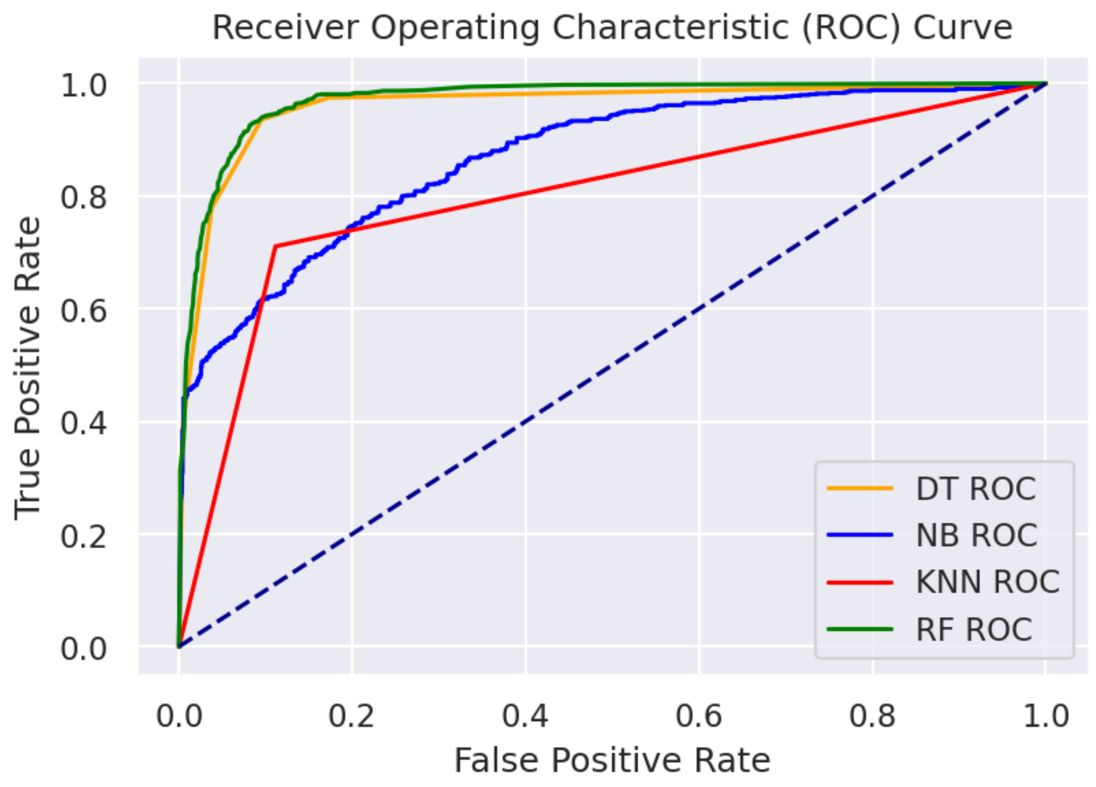
➢ In our case, the advantage of running the KNN algorithm is that it is easy to run, does not require training for our data, and we can always add information easily.

❖ <u>Random Forest</u>

➢ After running this algorithm, we received an accuracy = 0.9188.
➢ You can see that the accuracy of this algorithm is slightly higher than the level of accuracy we received in the Decision Tree algorithm, which makes sense because the RF algorithm consolidates several decision trees into a final output.

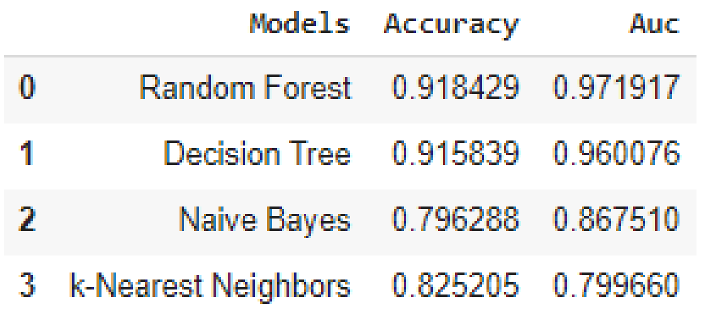
Results

❖ At this point, we made comparisons between all algorithms, using ROC and AUC.



➢ From an initial look at the graph, it is clear that the DT and RF algorithms are superior to the rest of the algorithms.

➢ To decide which of the algorithms is better, we performed a calculation for each algorithm and found the AUC of each:



	Models	Accuracy	Auc
0	Random Forest	0.918429	0.971917
1	Decision Tree	0.915839	0.960076
2	Naive Bayes	0.796288	0.867510
3	k-Nearest Neighbors	0.825205	0.799660

➢ Therefore, you can see that the Random Forest algorithm is the best with a score of 97%, with the following algorithm is Decision Tree with a score of 96%.

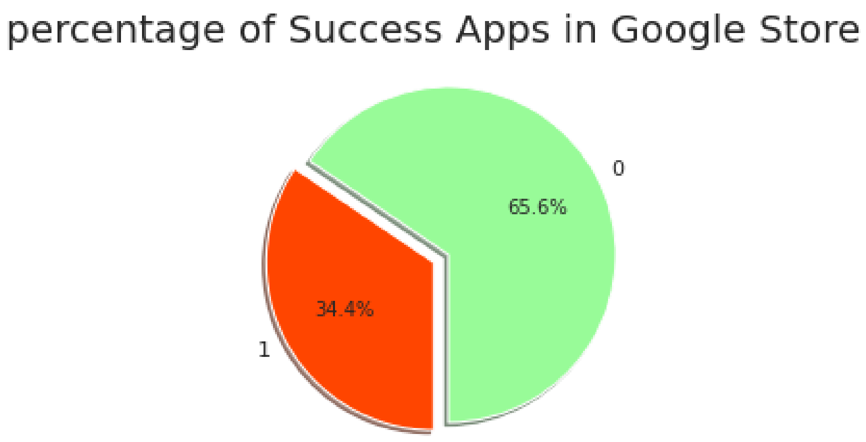
➢ In addition, it is known that there is an assumption that when using the base naive algorithm there is no dependence between the columns. In our case, we got that the algorithm got a low score what decision, and when we performed a test Pearson on the columns, we have seen that there is a correlation between some of the columns, so this is probably the reason. So, the base naive algorithm got a lower score.

Conclusions

After examining all the different algorithms, and the fact that there is no significant difference between DT and RF, we decided to draw conclusions from the decision tree which is visual. Here are the conclusions:

- A key conclusion is that the Reviews column is significant, meaning that from the 8,746 reviews an app receives, it will most likely be "successful."
- A secondary conclusion that can be drawn from the tree is that the Price column will affect success starting with the second branch, the effect is reflected in the fact that if an app costs less than $1.24, then there is a high probability that it will be "successful".

In addition, after all the steps, we received an overall conclusion that 34.4% of all apps in the google store are currently defined as "successful".



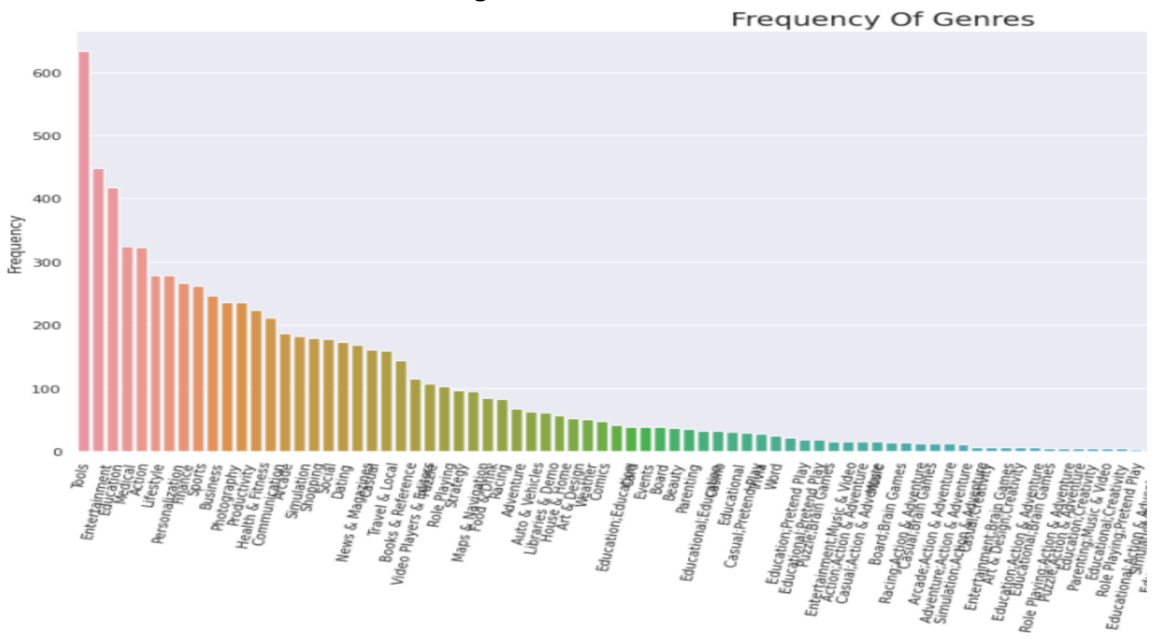
Appendices

- ❖ Appendix A - Sources

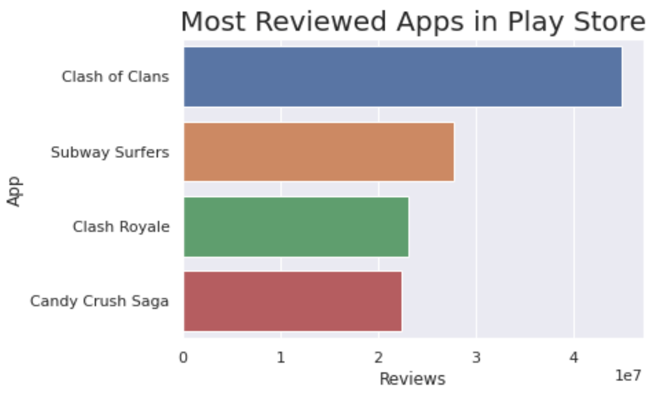
kaggle

- ❖ Appendix B - General Data

  - ➢ Following the initial finding that there is a correlation between "Reviews" and "Installs", we tried to see if there was a connection between "Reviews" and "Rating".
  - ➢ We did this by checking whether there was a link between the genre type from which the highest number of downloads were performed and the apps that received the most reviews.
    Here are the most common genres:



  - ➢ In the chart above, you can see that in the genre "Tools" the most downloads were performed, but you can see in the graph below that precisely the amount of applications with the highest and most reviews are from the "Games" genre, so we were unable to find any connection that would contribute to us drawing conclusions.


Most Reviewed Apps in Play Store
App
Clash of Clans
Subway Surfers
Clash Royale
Candy Crush Saga
Reviews
1e7