HarvardX - Final Project

Sorin Simion

16 June 2019

Contents

Introduction

The goal of this project is to create a recommendation system for Apps on Google Play using the Google Play Store App data set from Kaggle.
We need to train a machine learning algorithm using the inputs in one subset to predict App ratings in the validation set, develop our algorithm using a new subset and finally test out prediction for App ratings in the validation set as if they were unknown. RMSE will be used to evaluate how close our predictions are to the true values in the validation set.

Dataset description and Analysis

I used the Google Playstore App data set downloaded the data set from Kaggle. The dataset presents each Application by Category, Rating, Reviews, Size, Installs, Type, Price, Content Rating, Genres, Last Updated, Current Version, Android Version. In total there are 12 variables qualifying each application.

Data Structure

Using the Tidyverse Package, we can easily analyse the structure of the datasets. In order to better understand the challenge of this project, we need to see the general properties of the data.

Data set GooglePlaystoreApp, which is the test dataset and has a dimension of 10841 observations and 13 variables.

With following internal structure:

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 10841 obs. of  13 variables:
##  $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Coloring book moana" "U La
##  $ Category      : chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" ...
##  $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
##  $ Reviews       : num  159 967 87510 215644 967 ...
##  $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
##  $ Installs      : chr  "10,000+" "500,000+" "5,000,000+" "50,000,000+" ...
##  $ Type          : chr  "Free" "Free" "Free" "Free" ...
##  $ Price         : chr  "0" "0" "0" "0" ...
##  $ Content Rating: chr  "Everyone" "Everyone" "Everyone" "Teen" ...
##  $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Design" "Art & Design" ...
##  $ Last Updated  : chr  "January 7, 2018" "January 15, 2018" "August 1, 2018" "June 8, 2018" ...
##  $ Current Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
##  $ Android Ver   : chr  "4.0.3 and up" "4.0.3 and up" "4.0.3 and up" "4.2 and up" ...
##  - attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame': 2 obs. of  5 variables:
##   ..$ row     : int  10473 10473
##   ..$ col     : chr  "Reviews" NA
##   ..$ expected: chr  "no trailing characters" "13 columns"
##   ..$ actual  : chr  "M" "12 columns"
##   ..$ file    : chr  "'googleplaystore.csv'" "'googleplaystore.csv'"
##  - attr(*, "spec")=
##   .. cols(
##   ..   App = col_character(),
##   ..   Category = col_character(),
##   ..   Rating = col_double(),
##   ..   Reviews = col_double(),
##   ..   Size = col_character(),
##   ..   Installs = col_character(),
##   ..   Type = col_character(),
##   ..   Price = col_character(),
##   ..   `Content Rating` = col_character(),
##   ..   Genres = col_character(),
##   ..   `Last Updated` = col_character(),
##   ..   `Current Ver` = col_character(),
##   ..   `Android Ver` = col_character()
##   .. )
```

So there are 2 columns with numeric inputs and 11 with character inputs.

Next step is to analyse how many non-numeric input there are in numeric columns.

First column to be analysed is the Rating variable. There are 1474 entries as *NaN* in Rating Column.

By looking at a snapshot of several Apps with "NaN" as entry for Rating, we notice that other variable entries are correct hence at this stage we will not delete any of the rows with NaN as Rating.

```
## # A tibble: 1,474 x 13
##    App   Category Rating Reviews Size  Installs Type  Price
##    <chr> <chr>     <dbl>   <dbl> <chr> <chr>    <chr> <chr>
##  1 Mcqu~ ART_AND~    NaN      61 7.0M  100,000+ Free  0
##  2 Wrin~ BEAUTY      NaN     182 5.7M  100,000+ Free  0
##  3 Mani~ BEAUTY      NaN     119 3.7M  50,000+  Free  0
##  4 Skin~ BEAUTY      NaN     654 7.4M  100,000+ Free  0
##  5 Secr~ BEAUTY      NaN      77 2.9M  10,000+  Free  0
##  6 Reci~ BEAUTY      NaN      35 3.1M  10,000+  Free  0
##  7 Lady~ BEAUTY      NaN      30 9.9M  10,000+  Free  0
##  8 Anon~ BOOKS_A~    NaN     161 2.7M  10,000+  Free  0
##  9 SH-0~ BOOKS_A~    NaN       2 7.2M  50,000+  Free  0
## 10 URBA~ BOOKS_A~    NaN     114 7.3M  100,000+ Free  0
## # ... with 1,464 more rows, and 5 more variables: `Content Rating` <chr>,
## #   Genres <chr>, `Last Updated` <chr>, `Current Ver` <chr>, `Android
## #   Ver` <chr>
```

Next, we are checking the column with Reviews to see if all entries are numeric.

There is 1 entry as non-numeric for Reviews column.

Printout of the observation with non-numeric Review entry:

```
##  [1] "Life Made WI-Fi Touchscreen Photo Frame"
##  [2] "1.9"
##  [3] "19"
##  [4] "NA"
##  [5] "1,000+"
##  [6] "Free"
##  [7] "0"
##  [8] "Everyone"
##  [9] "NA"
## [10] "February 11, 2018"
## [11] "1.0.19"
## [12] "4.0 and up"
## [13] "NA"
```

This line will be taken out later when we clean the dataframe for the "Installs" column.

Next the column representing the app size will be analysed. For analyzing the "Size" column, first we identified the pattern used for entries for size which is the patern where the size starts with a number. We identify if there is any size entry not starting with a number and run another analysis and check what kind of entries are there.

There are 1695 entries that do not start with a number.

List of size variable :

Size	count
Varies with device	1695
11M	198
12M	196
14M	194
13M	191
15M	184
17M	160
19M	154
16M	149
26M	149
25M	143
20M	139
21M	138
10M	136
24M	136
18M	133
23M	117
22M	114
29M	103
27M	97

We see that all 1695 mismatch are the same, and it actually is not a mistake but the entry is quite logical since the application size depends on the device.

Next, column “Type” will be analysed:

Type	count
Free	10039
Paid	800
0	1
NaN	1

Printout line containing Type as 0:

```
##  [1] "Command & Conquer: Rivals" "FAMILY"
##  [3] "NaN"                       "0"
##  [5] "Varies with device"        "0"
##  [7] "NaN"                       "0"
##  [9] "Everyone 10+"              "Strategy"
## [11] "June 28, 2018"             "Varies with device"
## [13] "Varies with device"
```

There is one NaN and one 0, and the line with Type=0 is the same line that will be taken out when clearing for Reviews, and it is in line number 9149.

Next, column “Installs” is analyzed:

The pattern for Installs is numbers ending with a “+”. Then, we check which lines do not match the pattern or more precisely, we check which line starts with a letter, and there are 1 lines starting with a letters instead of a numbers in the “Installs” column.

and we check the line number, which is line number 10473.

Where we see that generally this app has wrong entries in nearly all columns:

```
##  [1] "Life Made WI-Fi Touchscreen Photo Frame"
##  [2] "1.9"
##  [3] "19"
##  [4] "NA"
##  [5] "1,000+"
##  [6] "Free"
##  [7] "0"
##  [8] "Everyone"
##  [9] "NA"
## [10] "February 11, 2018"
## [11] "1.0.19"
## [12] "4.0 and up"
## [13] "NA"
```

Cleaning the Data set

Now we can start cleaning the dataset.

First we take out the 2 lines containing the wrong “Installs” entry and “Type” entry, since it is also correlating with the wrong “Reviews” entries and rename the new dataset as *New_googleplaystore*:

Below is shown the new structure if the new data set renamed as “New_googleplaystore”:

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10839 obs. of  13 variables:
##  $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Coloring book moana" "U La
##  $ Category      : chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" ...
##  $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
##  $ Reviews       : num  159 967 87510 215644 967 ...
##  $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
##  $ Installs      : chr  "10,000+" "500,000+" "5,000,000+" "50,000,000+" ...
##  $ Type          : chr  "Free" "Free" "Free" "Free" ...
##  $ Price         : chr  "0" "0" "0" "0" ...
##  $ Content Rating: chr  "Everyone" "Everyone" "Everyone" "Teen" ...
##  $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Design" "Art & Design" ...
##  $ Last Updated  : chr  "January 7, 2018" "January 15, 2018" "August 1, 2018" "June 8, 2018" ...
##  $ Current Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
##  $ Android Ver   : chr  "4.0.3 and up" "4.0.3 and up" "4.0.3 and up" "4.2 and up" ...
##  - attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame': 2 obs. of  5 variables:
##   ..$ row     : int  10473 10473
##   ..$ col     : chr  "Reviews" NA
##   ..$ expected: chr  "no trailing characters" "13 columns"
##   ..$ actual  : chr  "M" "12 columns"
##   ..$ file    : chr  "'googleplaystore.csv'" "'googleplaystore.csv'"
```

Since for validation set and test set needs the rating to be numerical and not *NaN*, we will need to take out the *NaN* entries from the dataframe:

The table is cleaned, and now we present a review of the new data set:

Type	count
Free	8719
Paid	647

Installs	count
1,000,000,000+	58
1,000,000+	1577
1,000+	713
1+	3
10,000,000+	1252
10,000+	1010
10+	69
100,000,000+	409
100,000+	1150
100+	309
5,000,000+	752
5,000+	432
5+	9
50,000,000+	289
50,000+	467
50+	56
500,000,000+	72
500,000+	538
500+	201

Price	count
0	8719
$2.99	114
$0.99	107
$4.99	70
$1.99	59
$3.99	58
$1.49	31
$2.49	21
$5.99	18
$9.99	16
$6.99	13
$399.99	11
$14.99	10
$4.49	9
$3.49	7
$7.99	7
$29.99	6
$11.99	5
$12.99	5
$19.99	5

Content Rating	count
Adults only 18+	3
Everyone	7420
Everyone 10+	397
Mature 17+	461
Teen	1084
Unrated	1

Genres	count
Tools	733
Entertainment	533
Education	468
Action	358
Productivity	351
Medical	350
Sports	333
Communication	328
Finance	323
Photography	317
Personalization	314
Lifestyle	313
Business	303
Health & Fitness	297
Social	259
Shopping	238
News & Magazines	233
Travel & Local	225
Arcade	207
Dating	195

App	count
ROBLOX	9
CBS Sports App - Scores, News, Stats & Watch Live	8
8 Ball Pool	7
Candy Crush Saga	7
Duolingo: Learn Languages Free	7
ESPN	7
Bleacher Report: sports news, scores, & highlights	6
Bowmasters	6
Bubble Shooter	6
Helix Jump	6
Nick	6
slither.io	6
Sniper 3D Gun Shooter: Free Shooting Games - FPS	6
Subway Surfers	6
Temple Run 2	6
Zombie Catchers	6
Angry Birds Classic	5
BeautyPlus - Easy Photo Editor & Selfie Camera	5
Calorie Counter - MyFitnessPal	5
eBay: Buy & Sell this Summer - Discover Deals Now!	5

It seems that 789 apps are present more than once in the table.

See below example of most frequently repated Apps:

As an example of repeated app:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres
Candy Crush Saga	GAME	4.4	22426677	74M	500,000,000+	Free	0	Everyone	Casual
Candy Crush Saga	GAME	4.4	22428456	74M	500,000,000+	Free	0	Everyone	Casual
Candy Crush Saga	GAME	4.4	22428456	74M	500,000,000+	Free	0	Everyone	Casual
Candy Crush Saga	GAME	4.4	22429716	74M	500,000,000+	Free	0	Everyone	Casual
Candy Crush Saga	GAME	4.4	22430188	74M	500,000,000+	Free	0	Everyone	Casual
Candy Crush Saga	GAME	4.4	22430188	74M	500,000,000+	Free	0	Everyone	Casual
Candy Crush Saga	FAMILY	4.4	22419455	74M	500,000,000+	Free	0	Everyone	Casual

Next step is to take out the duplicated App entries.

Next: change the variable type for "Installs" from chr to number since it will simplify analyses.

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    8196 obs. of  13 variables:
##  $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Coloring book moana" "U La
##  $ Category      : chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" ...
##  $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
##  $ Reviews       : num  159 967 87510 215644 967 ...
##  $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
##  $ Installs      : num  1e+04 5e+05 5e+06 5e+07 1e+05 5e+04 5e+04 1e+06 1e+06 1e+04 ...
##  $ Type          : chr  "Free" "Free" "Free" "Free" ...
##  $ Price         : chr  "0" "0" "0" "0" ...
##  $ Content Rating: chr  "Everyone" "Everyone" "Everyone" "Teen" ...
##  $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Design" "Art & Design" ...
##  $ Last Updated  : chr  "January 7, 2018" "January 15, 2018" "August 1, 2018" "June 8, 2018" ...
##  $ Current Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
##  $ Android Ver   : chr  "4.0.3 and up" "4.0.3 and up" "4.0.3 and up" "4.2 and up" ...
```

The final dataset has 8196 entries with 13 variables - 3 numeric and 10 character.

Table Review Summary

The following table presents the total number of rated Apps, Category, Content Rating, OS Version and Genres.

Data set table structure:

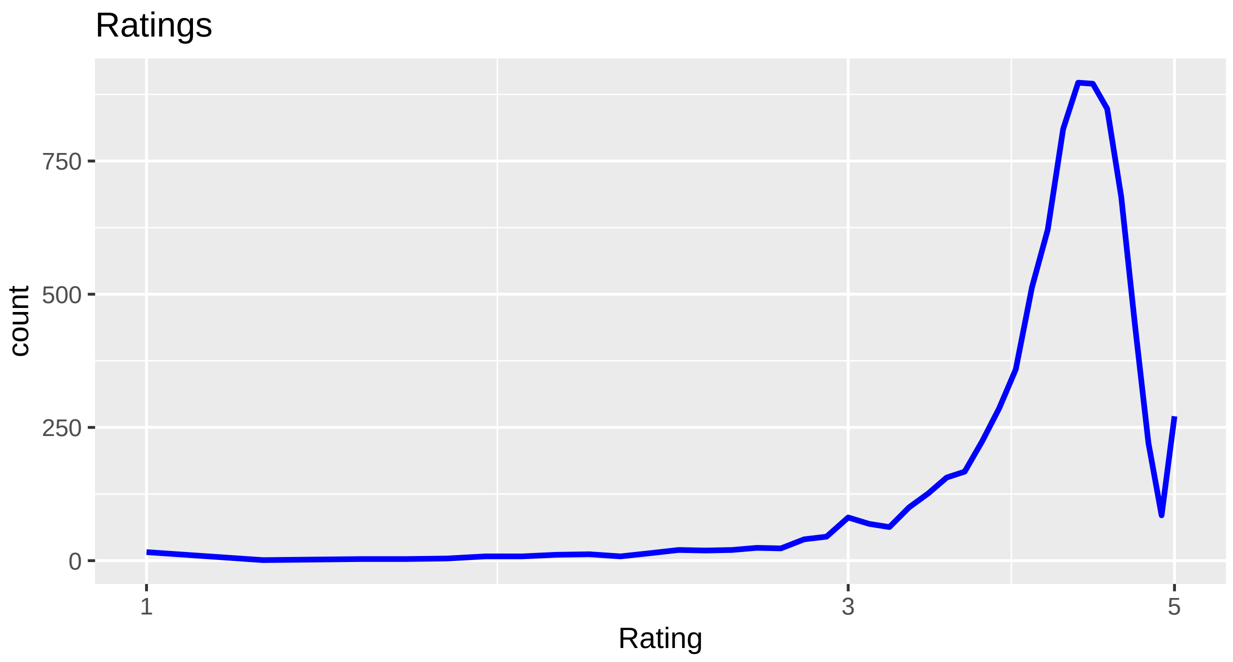
analyse	total
Distinct App	8196
Distinct Category	33
Distinct Content Rating	6
Distinct Android Version	32
Distinct Genres	114

Below is the list of 20 most popular Genres:

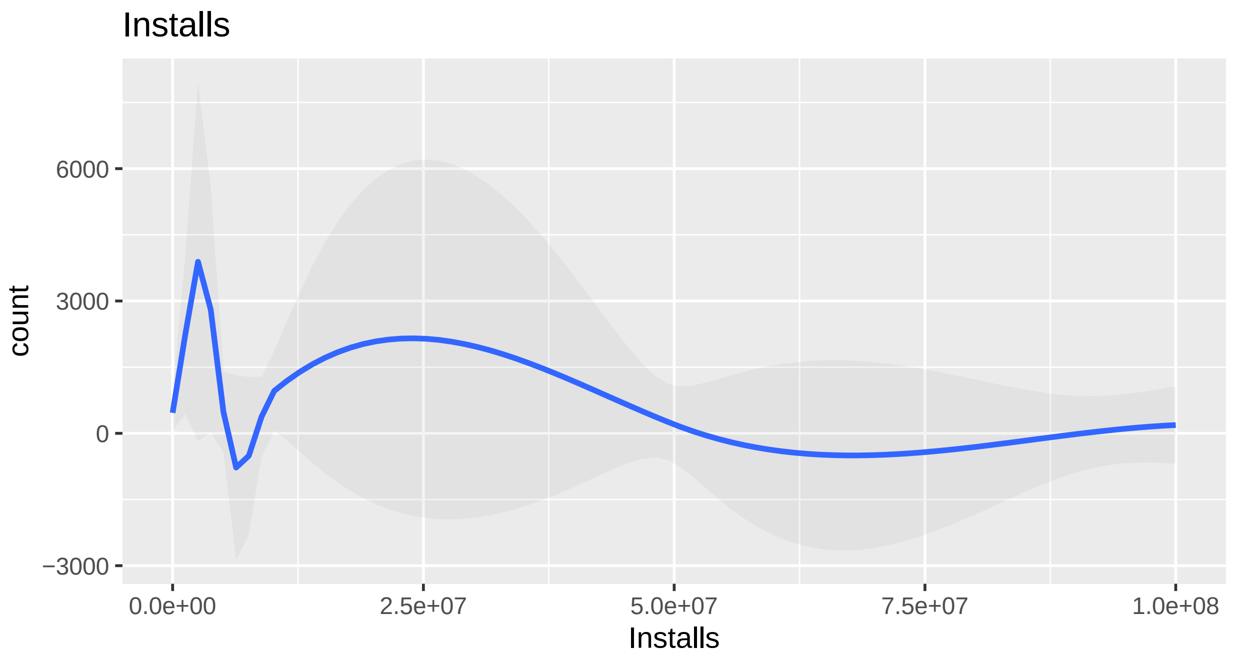
Genres	count
Tools	718
Education	587
Entertainment	502
Action	400
Finance	302
Lifestyle	302
Productivity	301
Personalization	298
Medical	290
Sports	270
Business	263
Photography	263
Communication	257

There are 66 different genres.

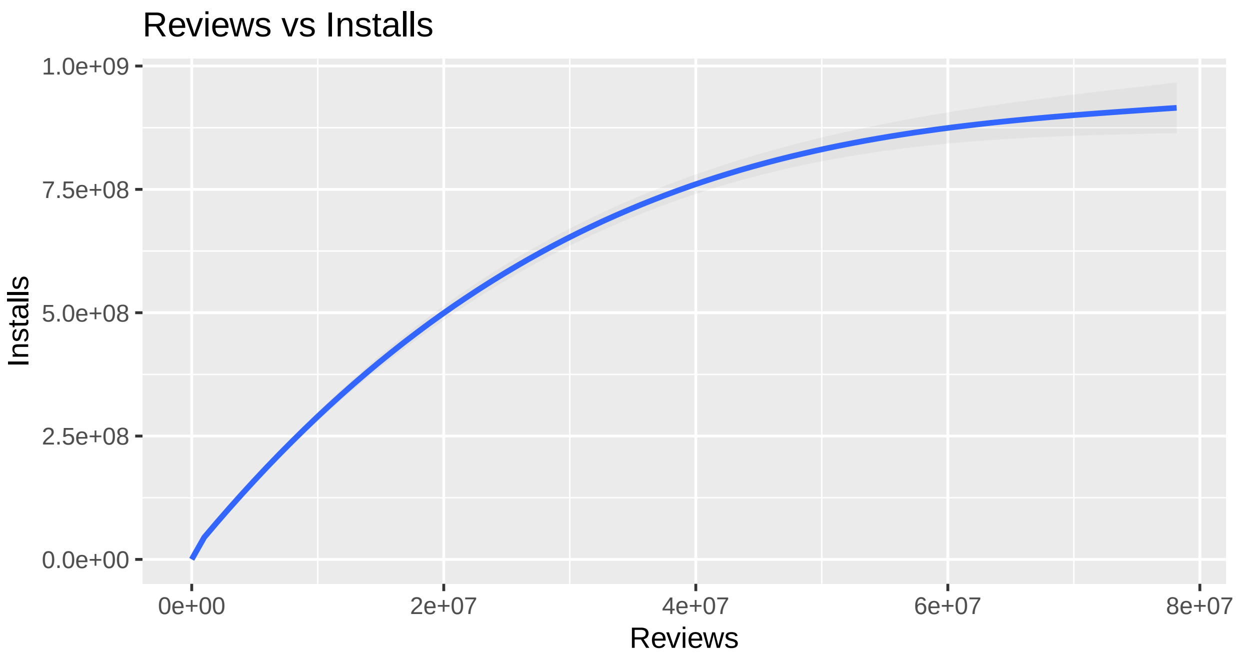
Rating distribution plot by Ratings:



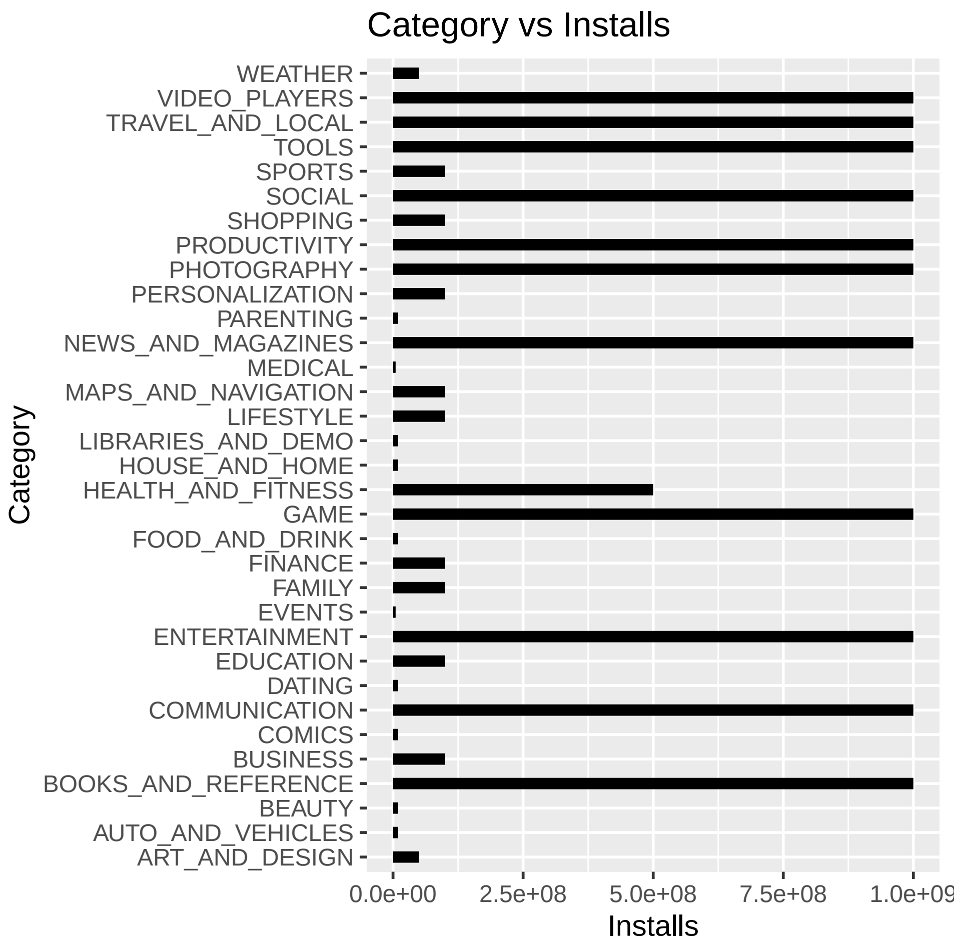
Rating distribution plot by Installs:



Plot by Reviews vs Installs:



Plot by Category vs Installs:



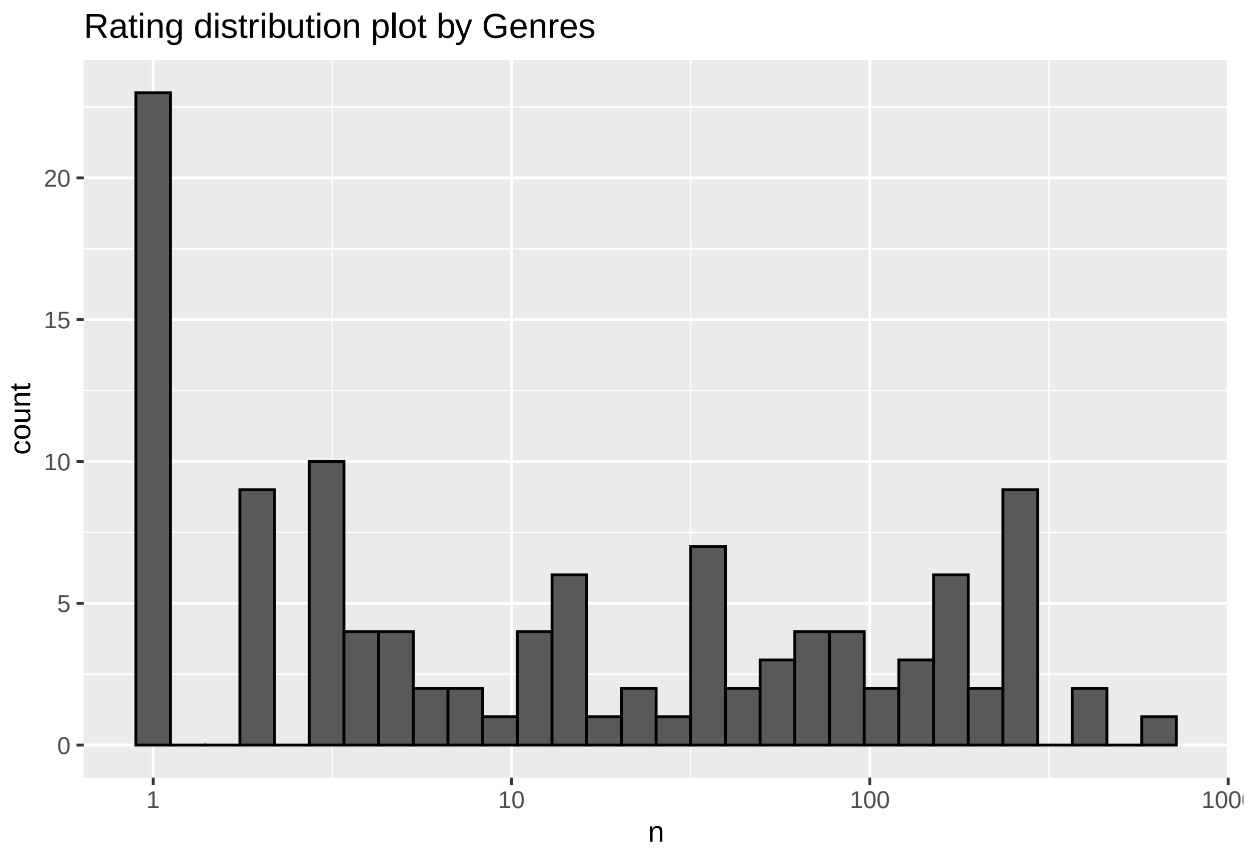
Creating the recommendation system

First, we need to create the validation set and the actual set which will be called "capstone":

1. The validation set will be 10% of Google Play store data,
2. Make sure "Category","Installs" and "Genres" in validation set are also in Capstone set

Let's analyse the new set named "Capstone".

Rating distribution plot by "Genres":



Top 10 reviewed App:

App	Category	Rating	Reviews	Installs	Type	Genres
Facebook	SOCIAL	4.1	78158306	1e+09	Free	Social
WhatsApp Messenger	COMMUNICATION	4.4	69119316	1e+09	Free	Communication
Instagram	SOCIAL	4.5	66577313	1e+09	Free	Social
Messenger – Text and Video Chat for Free	COMMUNICATION	4.0	56642847	1e+09	Free	Communication
Clash of Clans	GAME	4.6	44891723	1e+08	Free	Strategy
Clean Master- Space Cleaner & Antivirus	TOOLS	4.7	42916526	5e+08	Free	Tools
Subway Surfers	GAME	4.5	27722264	1e+09	Free	Arcade
YouTube	VIDEO_PLAYERS	4.3	25655305	1e+09	Free	Video Players & Editors
Security Master - Antivirus, VPN, AppLock, Booster	TOOLS	4.7	24900999	5e+08	Free	Tools
Candy Crush Saga	GAME	4.4	22426677	5e+08	Free	Casual

Top 10 least reviewed App:

App	Category	Rating	Reviews	Installs	Type	Genres
House party - live chat	DATING	1	1	10	Free	Dating
Mindvalley U Tallinn 2018	EVENTS	5	1	100	Free	Events
Labs on Demand	MEDICAL	5	1	10	Free	Medical
Anatomy & Physiology Vocabulary Exam Review App	MEDICAL	5	1	5	Free	Medical
NCLEX Multi-topic Nursing Exam Review-Quiz & notes	MEDICAL	5	1	10	Free	Medical
Basics of Orthopaedics	MEDICAL	5	1	10	Free	Medical
Familyfirst Messenger	MEDICAL	4	1	10	Free	Medical
Zen Leaf	MEDICAL	5	1	100	Free	Medical
Speech Therapy: F	FAMILY	1	1	10	Paid	Education
Android P Style Icon Pack	PERSONALIZATION	5	1	100	Paid	Personalization

Next, we need to create the Loss function, the residual mean squared error (RMSE) on a test set. The interpretation of which is if this number is larger than 1, it means our typical error is larger than one star.

```r
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

The simple model

The first model is to build the simplest possible recommendation system: same rating for all App regardless of other entries such as “Reviews”, “Installs” and “Genres”.

We will call the simple model RMSE, the naive RMSE.

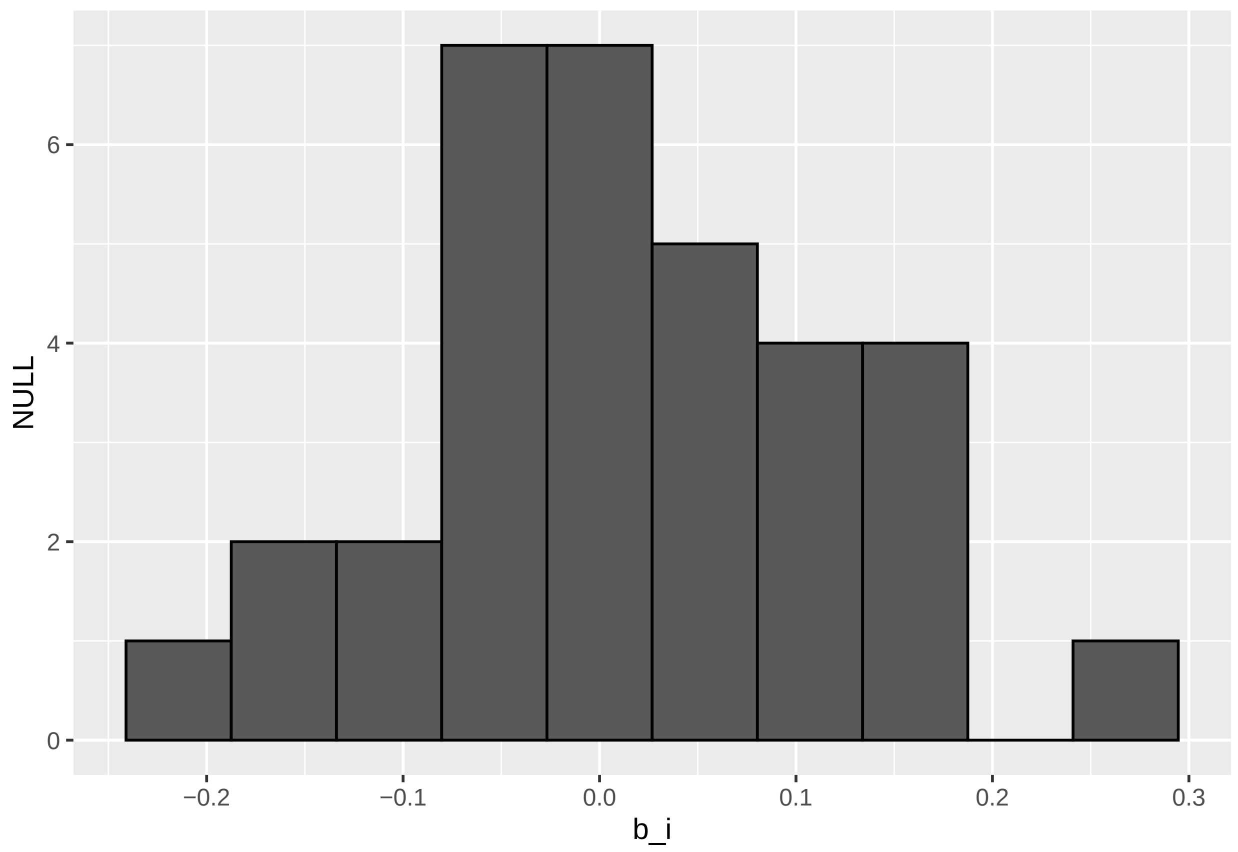
Where mu_hat = 4.171661.

Value of naive RMSE is equal to:

method	RMSE
The average	0.4933486

This table will be used to evaluate the improvement of each model.

This shows us that RMSE is already at 0.4933486, meaning that the prediction has a strong accuracy. Next step is to introduce the Category effect model for predicting the ratings:



method	RMSE
The average	0.4933486
Category Effect Model	0.4873903

The table shows a slight improvement of the prediction using the Category effect, as RMSE is equal to 0.4873903.

Next, we should add also the “Installs”" effect into the model since more “Installs”" means more accurate ratings:

method	RMSE
The average	0.4933486
Category Effect Model	0.4873903
Category + Installs Effects Model	0.4756994

In this case the prediction have highly improved since the RMSE is equal to 0.4756994.

However, the analysis does not stop here since we can add also the genre effect to the model as this factor impacts also the rating values:

method	RMSE
The average	0.4933486
Category Effect Model	0.4873903
Category + Installs Effects Model	0.4756994
Category + Installs +Genres Effects Model	0.4734804

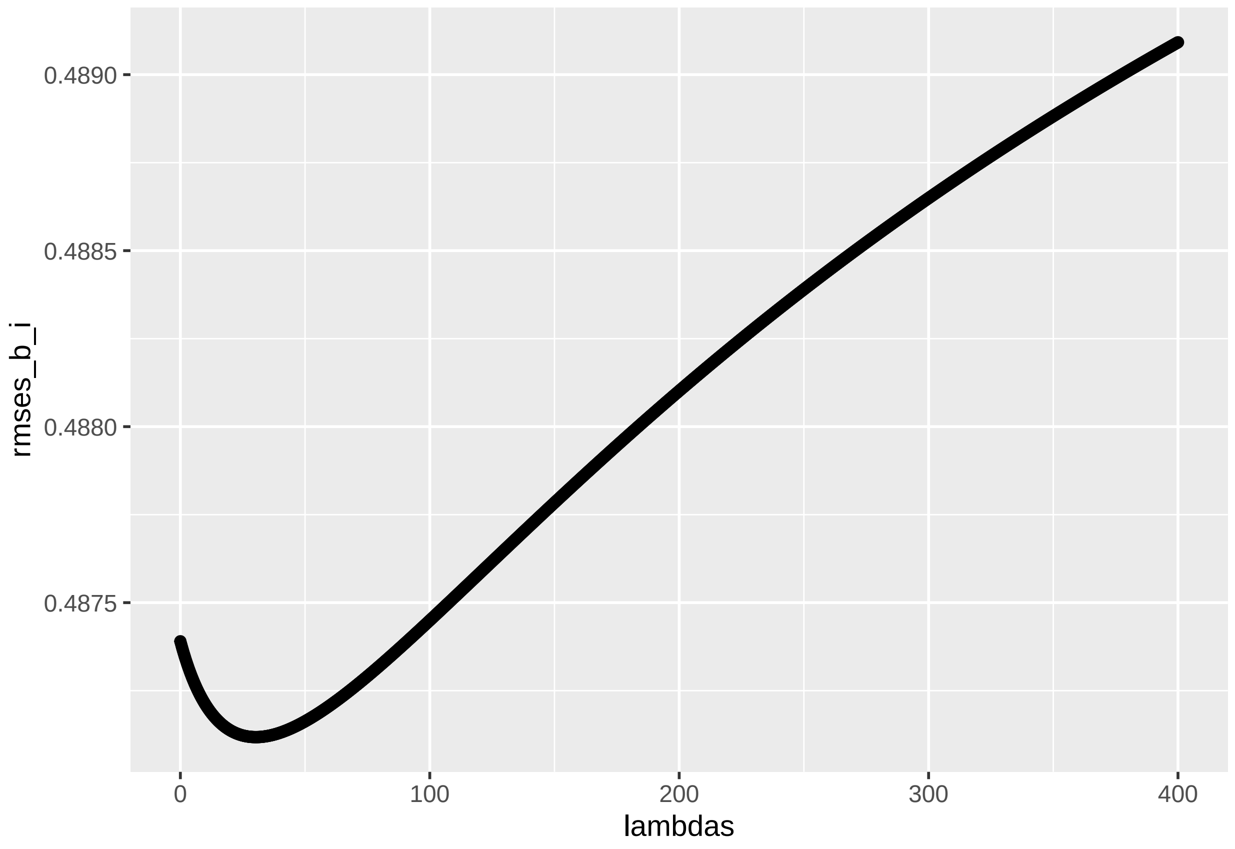
We notice that result are slightly improved with RMSE equaling to 0.4734804.

Regularization

Because data set analysis showed us that some apps are rarely rated , we should add a regularization effect to the prediction. This is done by introducing the Penalized Least squares with Lambda a penalty factor. First, we create a sequence of Lambdas which will be applied to a new function for determining the best lambda fit with lambda between 0 and 400 with steps of 0.25.

```
lambdas <- seq(0, 400, 0.25)
```

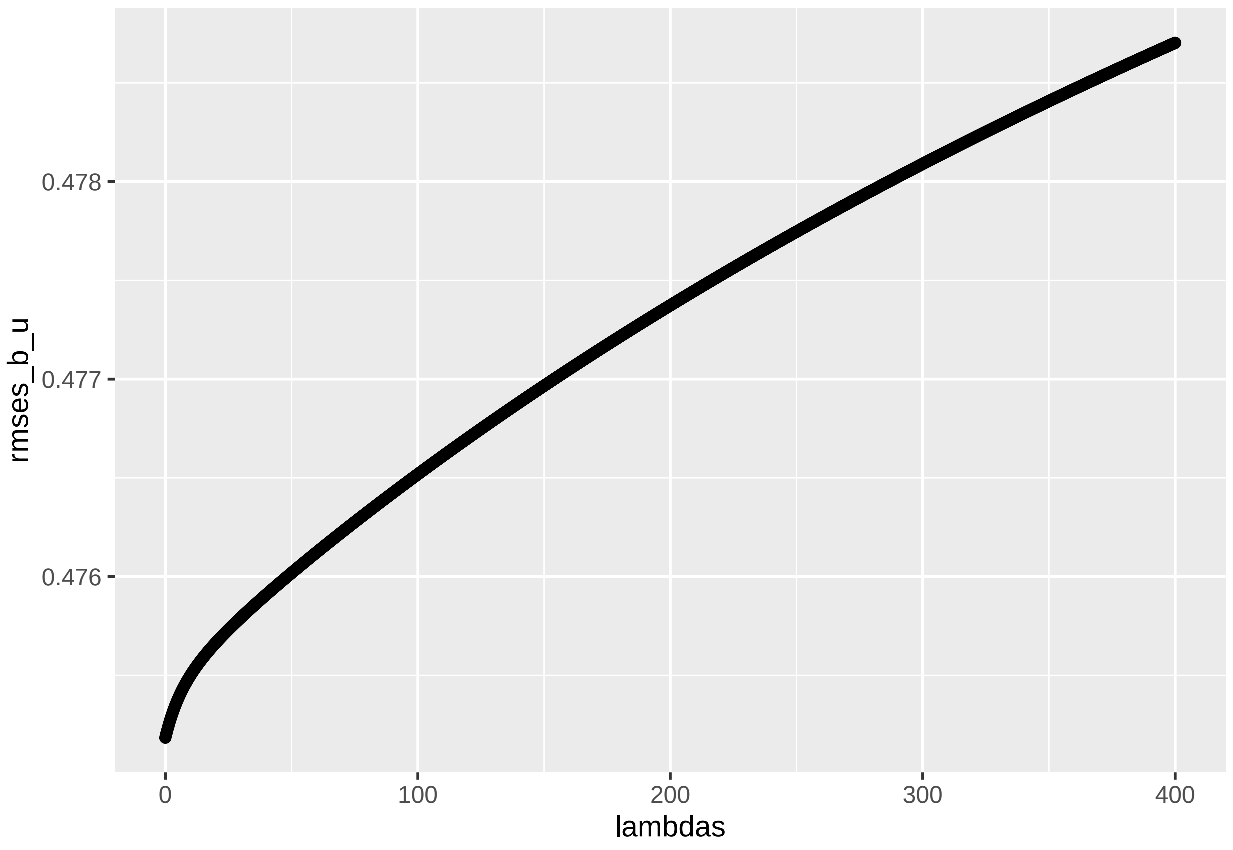
Next, we create the function and plot the evaluation of the lambdas against RMSE with b_i (category effect model):



Thus optimal lambda with category effect model is : 30.25. and we get RMSE result of 0.487118:

method	RMSE
The average	0.4933486
Category Effect Model	0.4873903
Category + Installs Effects Model	0.4756994
Category + Installs +Genres Effects Model	0.4734804
Regularized Category Effect Model	0.4871180

Next step is to add the Installs effect regularized model:

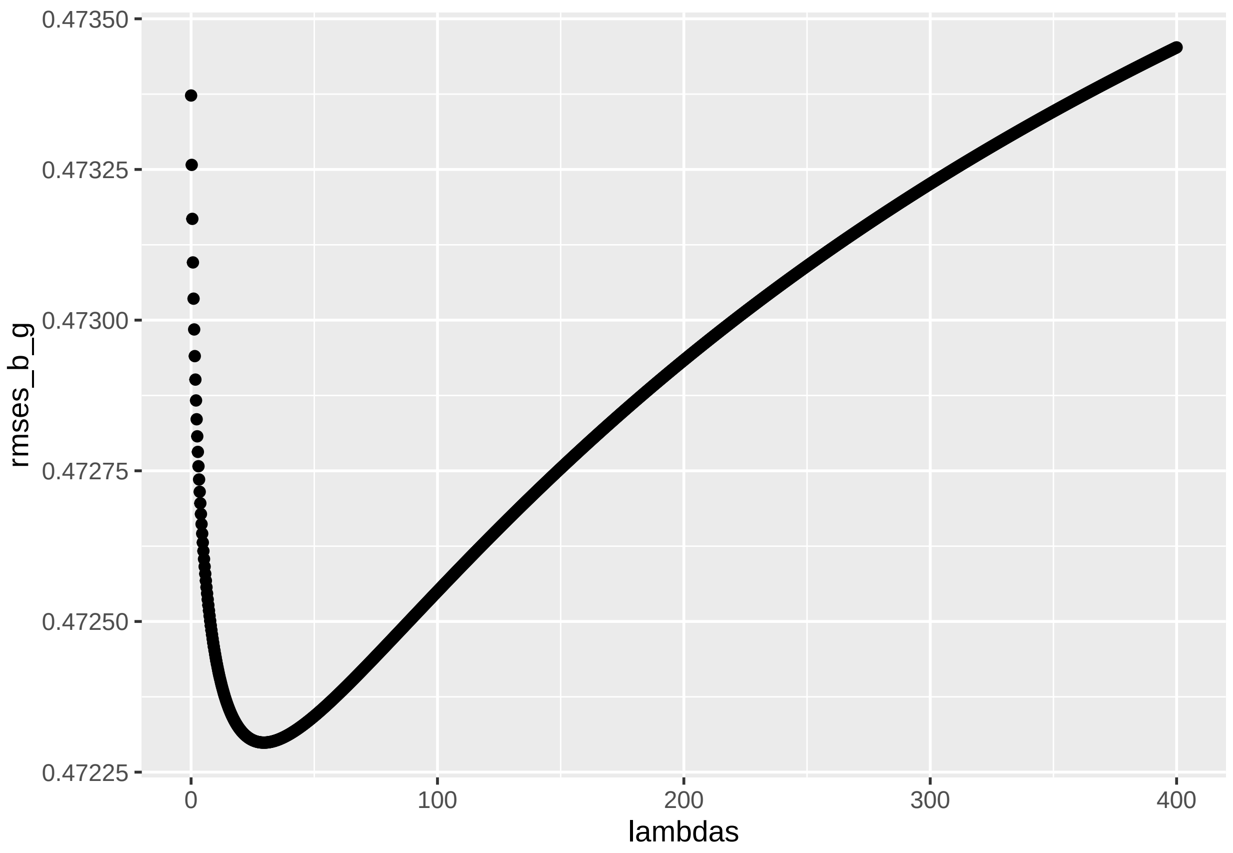


The plot shows us that the optimal lambda for Installs effect model is 0.

And RMSE result are improved as shown on table below where RMSE equals 0.4751846:

method	RMSE
The average	0.4933486
Category Effect Model	0.4873903
Category + Installs Effects Model	0.4756994
Category + Installs +Genres Effects Model	0.4734804
Regularized Category Effect Model	0.4871180
Regularized Category + Installs Effect Model	0.4751846

Last effect to be introduced into the model is the genre effect:



The plot shows us that optimal lambda for genre effect model is 29.5.

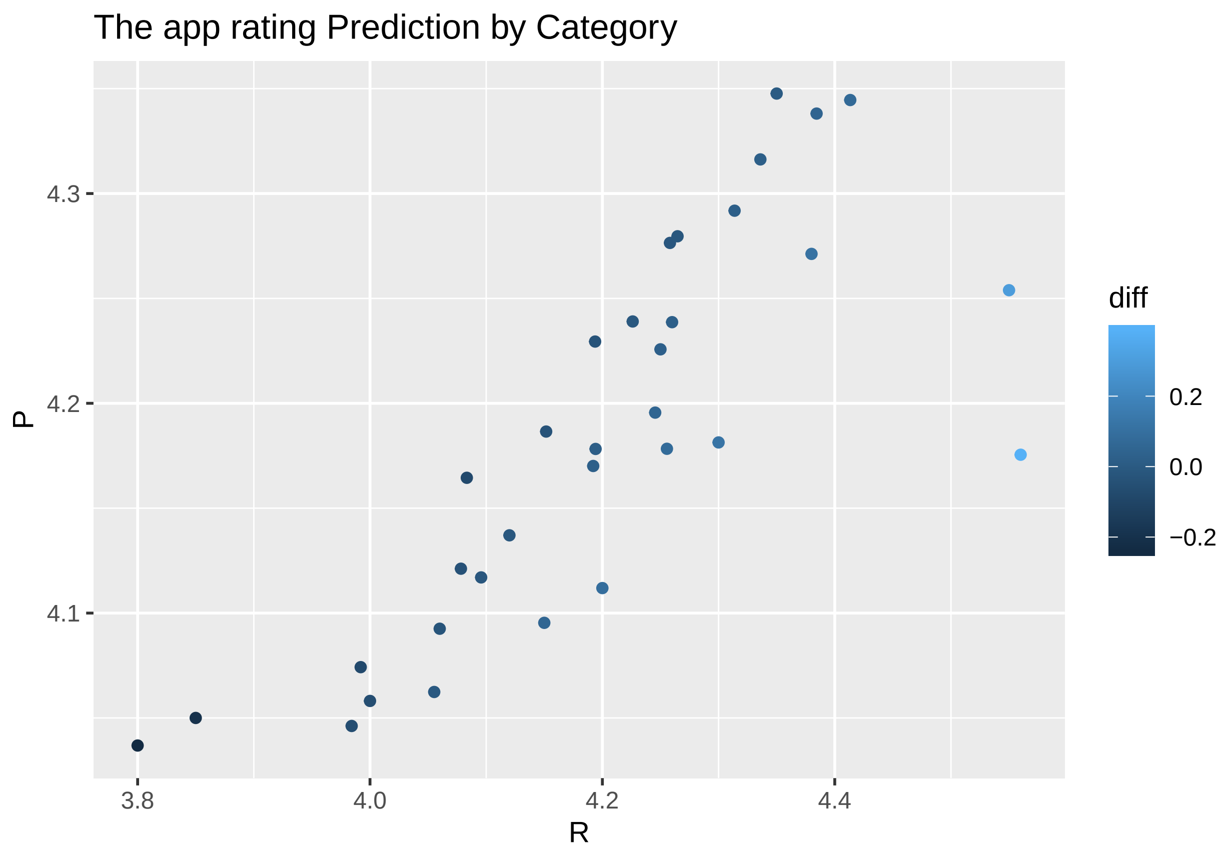
We see that the RMSE for all 3 effects models improves the prediction with RMSE = 0.4722991 as shown on the table below.

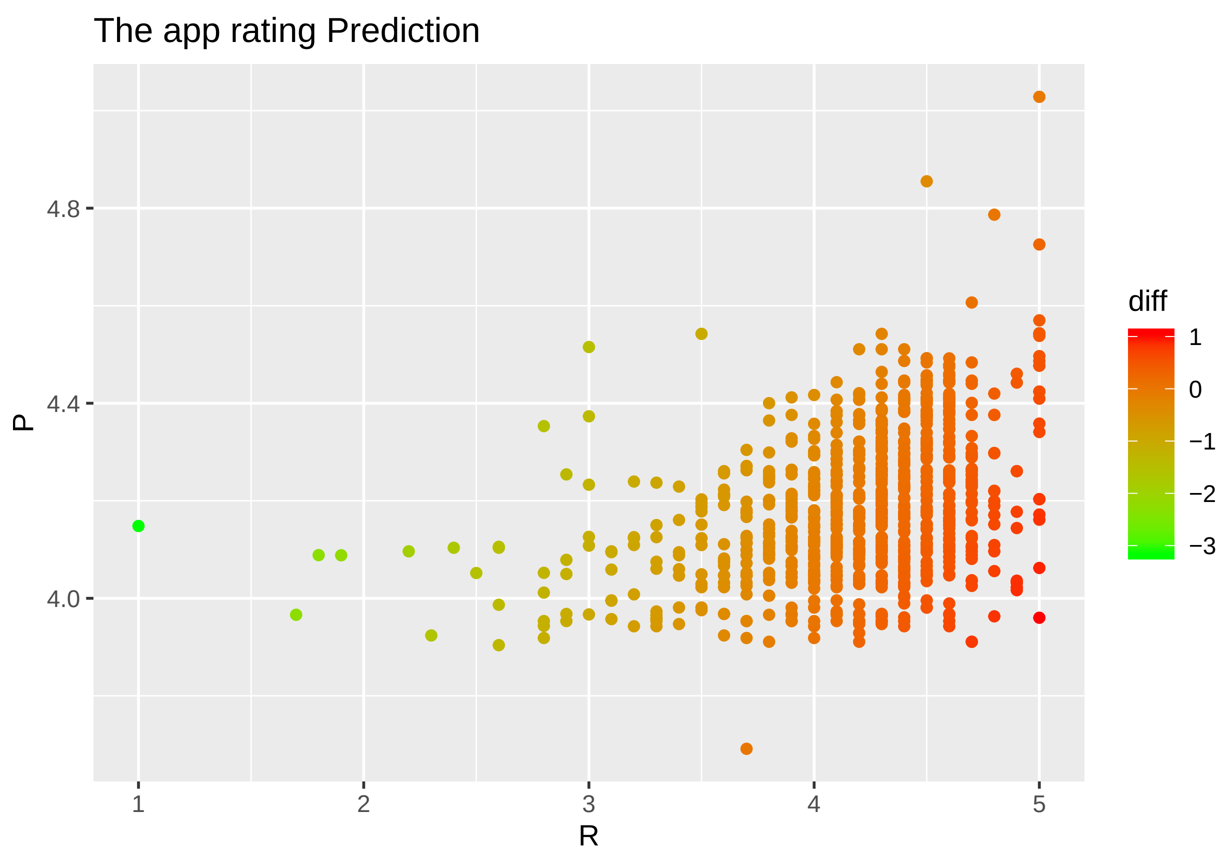
method	RMSE
The average	0.4933486
Category Effect Model	0.4873903
Category + Installs Effects Model	0.4756994
Category + Installs +Genres Effects Model	0.4734804
Regularized Category Effect Model	0.4871180
Regularized Category + Installs Effect Model	0.4751846
Regularized Category + Installs + Genres Effect Model	0.4722991

Results and Prediction table

Now, that we have the final model with RMSE of 0.4722991, we will continue to produce the whole prediction table and present some snapshots of the table.

Performing the verification of the prediction model.





The chart above shows that most predictions based on categories are between 0 and 0.4 point difference between predicted ratings and real ratings with very few outliers, which shows that the recommandation system renders very accurate results, whereas the second chart shows that even per app the rating prediction is between 0 and 1 point difference between ratings and predicted ratings and is another proof that the recommandation system is fairly accurate.

Snapshot of 10 App prediction:

Category	n	avg_ratings	pred_ratings
FAMILY	1527	4.2	4.2
GAME	907	4.2	4.3
TOOLS	717	4.0	4.1
FINANCE	302	4.1	4.2
PRODUCTIVITY	301	4.2	4.2
LIFESTYLE	300	4.1	4.1
PERSONALIZATION	298	4.3	4.3
MEDICAL	289	4.2	4.2
BUSINESS	263	4.1	4.1
PHOTOGRAPHY	263	4.2	4.1

Snapshot of predicted rating for 10 first ratings by Category:

App	Rating	Category	Type	Genres	pred_c
Photo Editor & Candy Camera & Grid & ScrapBook	4.1	ART_AND_DESIGN	Free	Art & Design	4.1
U Launcher Lite – FREE Live Cool Themes, Hide Apps	4.7	ART_AND_DESIGN	Free	Art & Design	4.3
Sketch - Draw & Paint	4.5	ART_AND_DESIGN	Free	Art & Design	4.4
Paper flowers instructions	4.4	ART_AND_DESIGN	Free	Art & Design	4.1
Smoke Effect Photo Maker - Smoke Editor	3.8	ART_AND_DESIGN	Free	Art & Design	4.1
Infinite Painter	4.1	ART_AND_DESIGN	Free	Art & Design	4.2
Garden Coloring Book	4.4	ART_AND_DESIGN	Free	Art & Design	4.2
Text on Photo - Fonteee	4.4	ART_AND_DESIGN	Free	Art & Design	4.2
Name Art Photo Editor - Focus n Filters	4.4	ART_AND_DESIGN	Free	Art & Design	4.2
Tattoo Name On My Photo Editor	4.2	ART_AND_DESIGN	Free	Art & Design	4.4

Conclusion

The goal of this project was to create a Recommandation System for Apps available on Google Playstore. In the project, we downloaded the Google Play Store App data set from Kaggle(. With first analysis we see that there are around 10000 observations with 13 variables. Starting the analysis of the Data and Data structure we notice that the dataset needs to be cleaned since there were: wrong entries, incompatible entries such as characters instead of numbers, *NaN* entries and Apps that were repeated several times. After cleaning the data, we ended with a dataset of 9000 observations, and the new dataset was renamed to New_googlePlaystore. Next step was to move to the creation of the Recommendation System, train a ML algorithm using the inputs in one subset to predict ratings in the validation set. Developing our algorithm using the Capstone dataset newly created and predicting ratings in the validation set as if they were unknown. RMSE was used to evaluate how close our predictions were to the validation set. For the analysis and creation of the Recommendation System, I started to perform the naive process where we assumed that all ratings are the same. I continued by adding the number of Installs effect to the model and the number of Reviews effect. However an additional factor could be added which is the Genre effect. I ended with a recommendation system using all three effects rendering a good RMSE. Even if we have achieved good results with the 3 effects, the algorithm has been further optimized for apps with fewer downloads, or rated and reviewed rarely. I developped also the regularization effect by using the Penalty least square function and finding best fitted lambdas where lambdas where the penalty factors. I ended with a Recommendation system with a final RMSE = 0.4722991.