MovieLens: Movie recommendation system

HarvardX PH125.9x - Data Science: Capstone

Safeen Ghafour

November 23, 2020

Contents

1 Introduction

1.1 Definition

This assignment is part of HarvardX PH125.9x - Data Science: Capstone course. We will create a movie recommendation system using MovieLens dataset.

1.2 Recommendation systems

Recommendation systems use ratings that users have given items to make specific recommendations. These systems, are taking an important place in the world of machine learning applications of Artificial Intelligence.

The evolution of such systems is directly connected to their commercial use by tech giants like Netflix, Amazon and others to serve personalised content to the audience.

The movie recommendation system is based on ratings given by users to movies. It will continuously undergo improvement as more data and interaction become available.

1.3 The Data

For this project we will use a portion of GroupLens research lab database that contains over 20 million ratings for over 27,000 unique movies rated by more than 138,000 unique users. Our data subset is available at http://files.grouplens.org/datasets/movielens/ml-10m.zip and contains 10 million ratings.

Essentially, the dataset consists of movie ratings in a many-to-many relationship with users.

Each movie has one or more categories. The data does not contain any user demographics to identify or segregate users.

1.4 Objective

The objective of this project is to predict ratings for movies that are not included in the training subset.

Predictions will be evaluated using the root mean square error (RMSE) method (the lower the numerical value, the better). The goal of the project is to achieve an RMSE lower than **0.86490**.

2 Data preparation

We will first download the MovieLens subset. The compressed file contains two important files for our purpose:

movies.dat: MovieId, Title and genres separated by pipe. The title contains the publication year of the movie.

ratings.dat: userId, MovieId, rating and the timestamp of the rating.

After joining the data from the two files in a new data-frame using movieId as key, we will split it to two random partitions, of which 10% is used for test and 90% for training

3 Exploratory data analysis

3.1 Summaries

To examine the data we will print the first six records:

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

A summary of the training subset confirms that it contains 90% of the ratings, i.e. 9,000,055 records.

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055
1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
Mean :35870	Mean : 4122	Mean :3.512	Mean :1.033e+09		
3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09		
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09		

And that the test subset contains 10% which is equivalent to 999,999 records.

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:999999	Length:999999
1st Qu.:18096	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.467e+08	Class :character	Class :character
Median :35768	Median : 1827	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
Mean :35870	Mean : 4108	Mean :3.512	Mean :1.033e+09		
3rd Qu.:53621	3rd Qu.: 3624	3rd Qu.:4.000	3rd Qu.:1.127e+09		
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09		

There are **6** variables in the dataset:

userId, **movieId**, **rating**, **timestamp**, **title**, **genres**.

3.2 Analysis

To better understand the data we will perform some basic data analysis.

3.2.1 The totals

The total number of unique movies is **10,677** rated by **69,878** unique users.

3.2.2 The best movies

As we see from the table below, 'Pulp Fiction' is the most rated movie, however if we sort the table by the highest average of ratings, 'Who's Singin' Over There?' has the highest average rating.

'Pulp Fiction'has 31,362 ratings with an average of 4.154.

movieId	Title	ratings_count	rating_average
296	Pulp Fiction (1994)	31362	4.154789
356	Forrest Gump (1994)	31079	4.012822
593	Silence of the Lambs, The (1991)	30382	4.204101
480	Jurassic Park (1993)	29360	3.663522
318	Shawshank Redemption, The (1994)	28015	4.455131
110	Braveheart (1995)	26212	4.081852
457	Fugitive, The (1993)	25998	4.009155
589	Terminator 2: Judgment Day (1991)	25984	3.927859

movieId	Title	ratings_count	rating_average
260	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672	4.221311
150	Apollo 13 (1995)	24284	3.885789

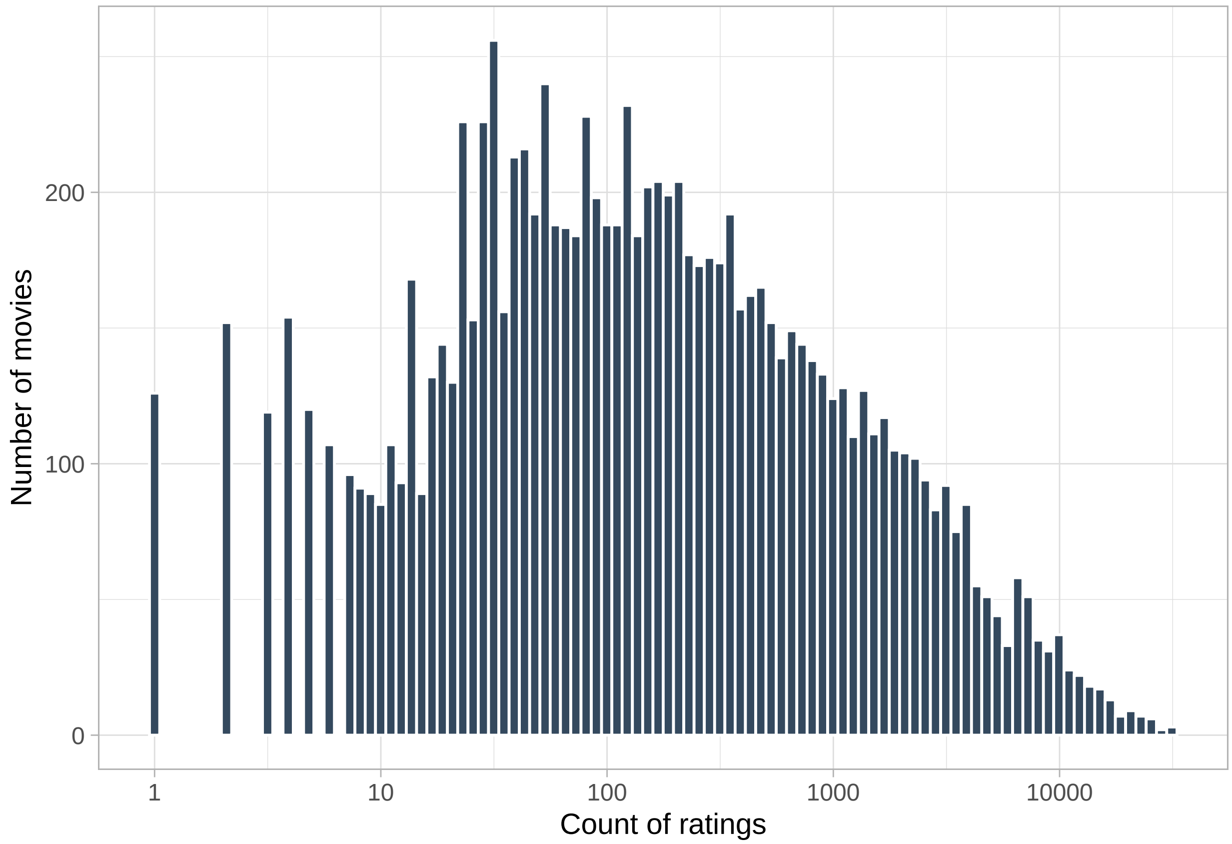
'Who's Singin' Over There?' has a rating average of 4.75 however it is rated 4 times only.

movieId	Title	ratings_count	ratings_average
5194	Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4	4.75
26048	Human Condition II, The (Ningen no joken II) (1959)	4	4.75
26073	Human Condition III, The (Ningen no joken III) (1961)	4	4.75
33264	Satan's Tango (Sátántangó) (1994)	2	5.00
65001	Constantine's Sword (2007)	2	4.75
3226	Hellhounds on My Trail (1999)	1	5.00
42783	Shadows of Forgotten Ancestors (1964)	1	5.00
51209	Fighting Elegy (Kenka erejii) (1966)	1	5.00
53355	Sun Alley (Sonnenallee) (1999)	1	5.00
64275	Blue Light, The (Das Blaue Licht) (1932)	1	5.00

3.2.3 Movie effect

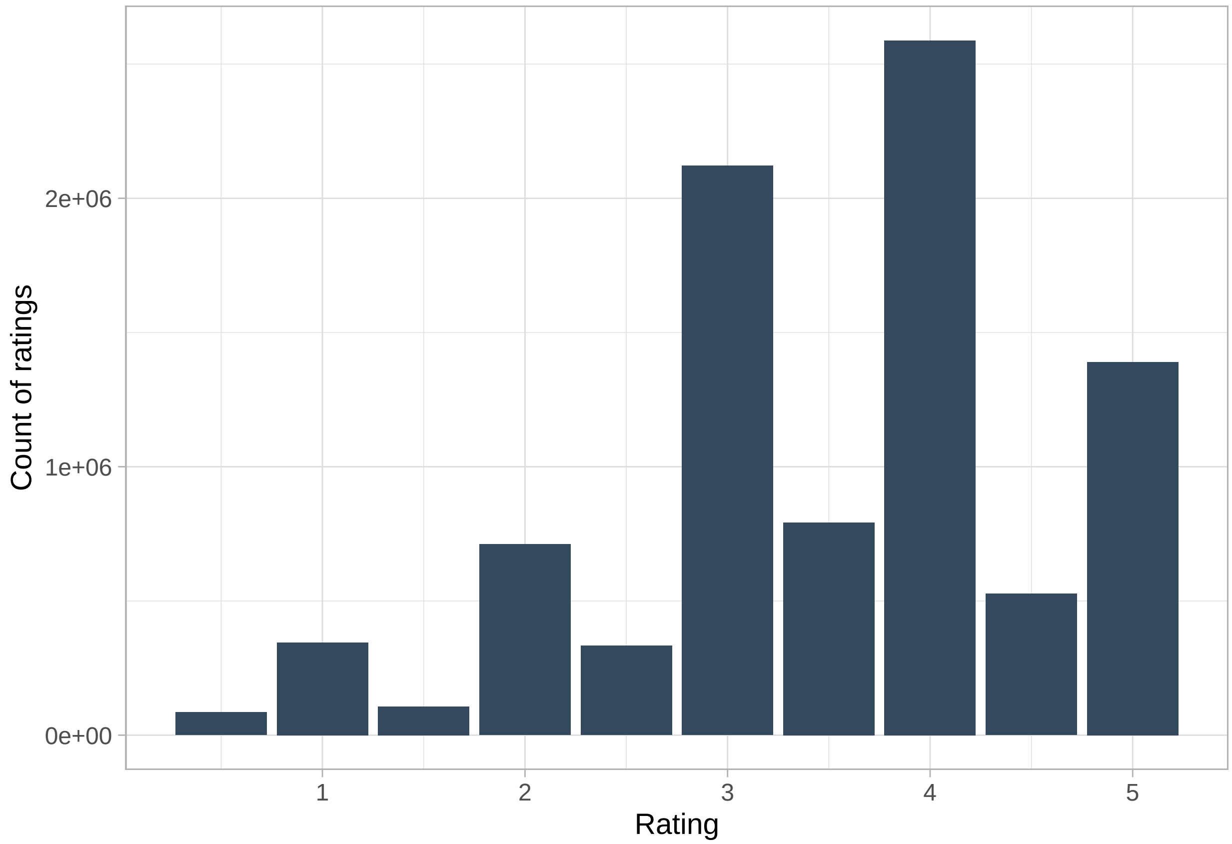
There is a wide variation in the quantity of ratings each movie has received. Not every movie is rated the same. There are **126** movies that have received only **1** rating while only **3** movies have been rated **30,000** or more times.

The average number of ratings per movie is **843**.

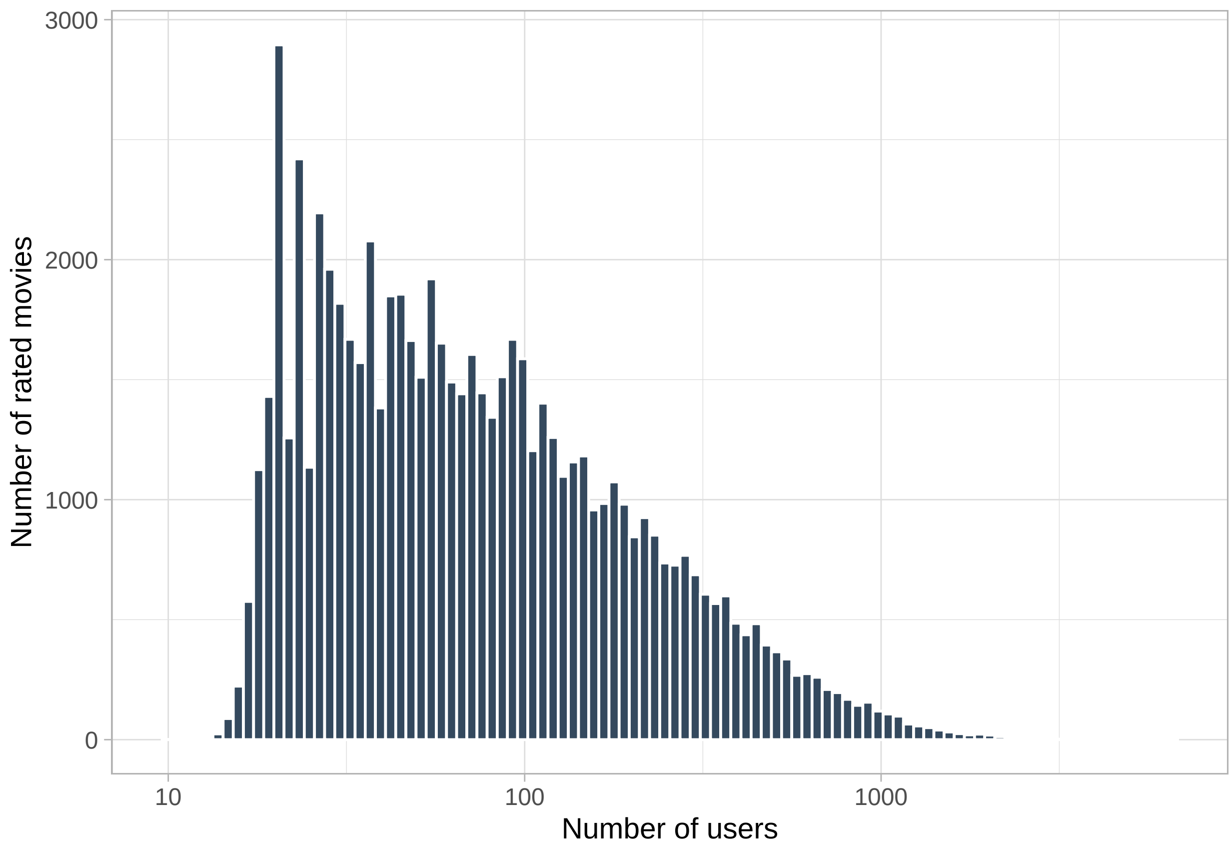


3.2.4 User effect

Grouping the data by rating gives a clear view of rating distribution. A **4** rating is by far the most popular. In this act of generosity by users **50%** of the movies got a 4 or higher.



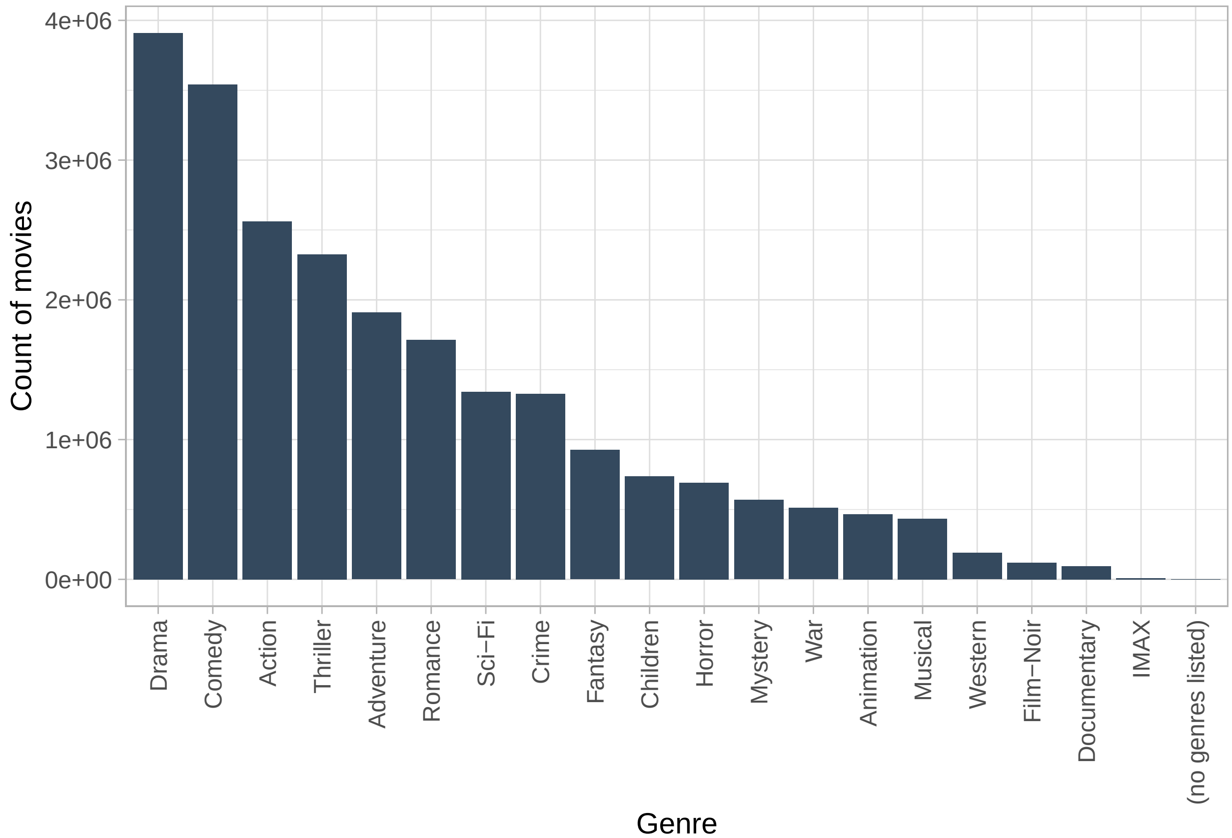
User activity varies between a low of **10** and a maximum of **6,616** ratings per user. The average number of ratings made by the user population is **129** ratings per user.



3.2.5 Genres effect

In the training dataset there is a genres column which is a string, with multiple components separated by a pipe character. There are **797** rated combinations with 'Drama' at the top.

genres	N	avg
Drama	733296	3.712364
Comedy	700889	3.237858
Comedy|Romance	365468	3.414486
Comedy|Drama	323637	3.598961
Comedy|Drama|Romance	261425	3.645824
Drama|Romance	259355	3.605471
Action|Adventure|Sci-Fi	219938	3.507407
Action|Adventure|Thriller	149091	3.434101
Drama|Thriller	145373	3.446345
Crime|Drama	137387	3.947135



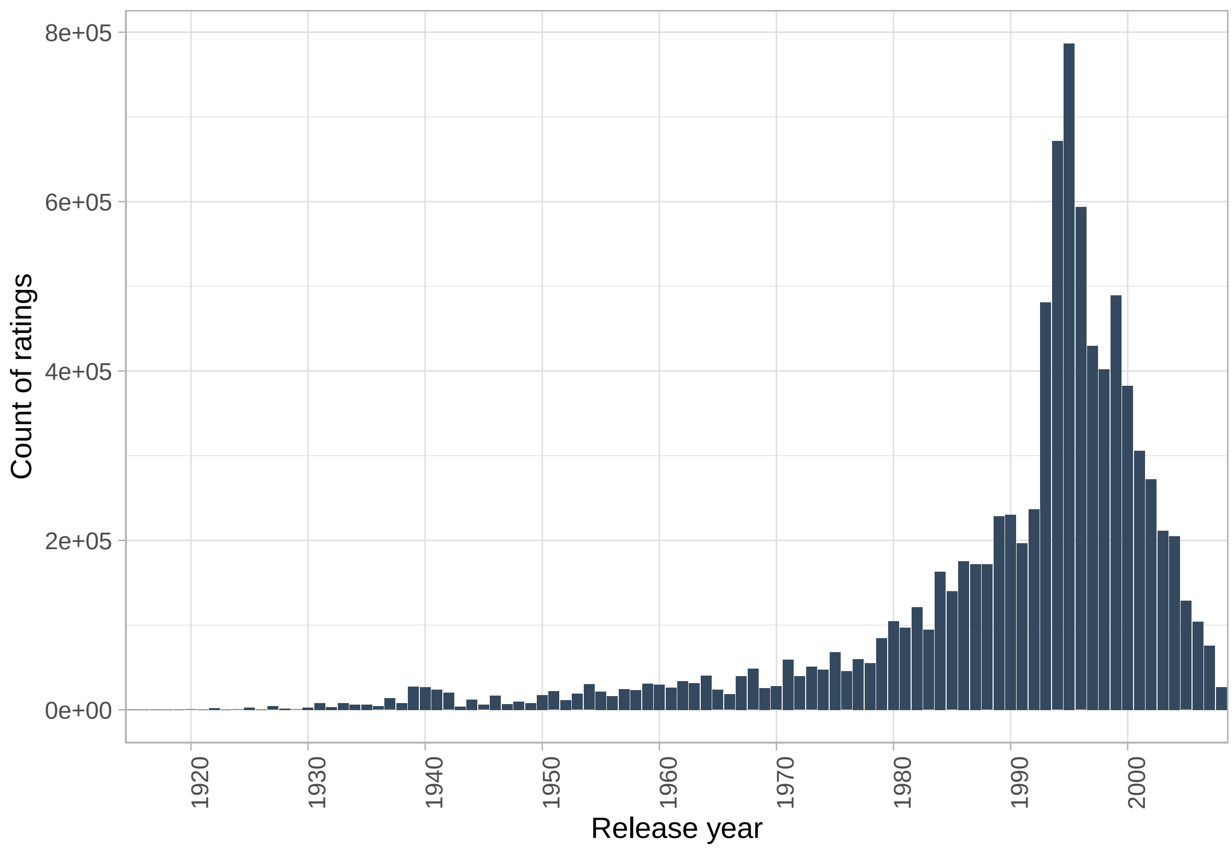
If we split the genres column and arrange it by the number of ratings we get the following results and only **20** genres:

genres	N	avg
Drama	3910127	3.673131
Comedy	3540930	3.436908
Action	2560545	3.421405
Thriller	2325899	3.507676
Adventure	1908892	3.493544
Romance	1712100	3.553813
Sci-Fi	1341183	3.395743
Crime	1327715	3.665925
Fantasy	925637	3.501946
Children	737994	3.418715

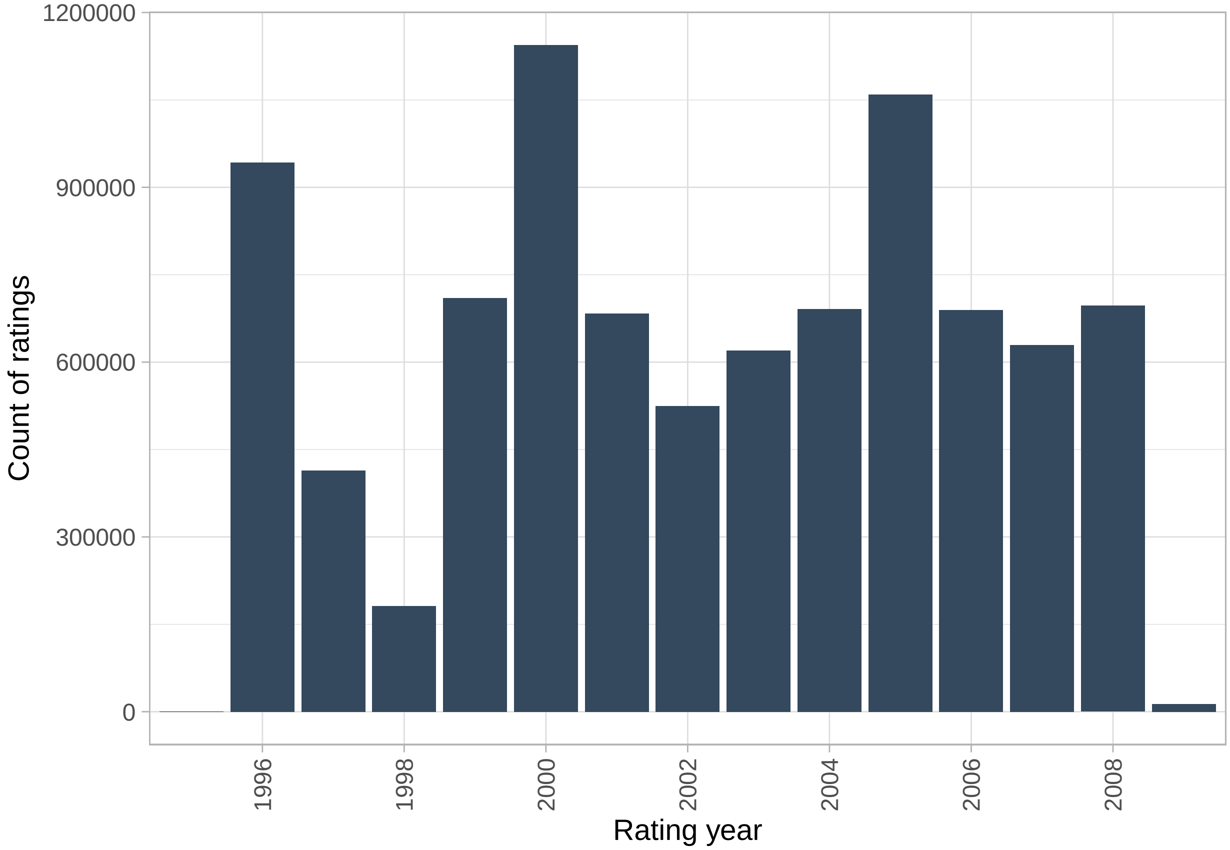
For the sake of simplicity and the purposes of this analysis we disregard the accuracy or relevance of individual movie classifications into specific genres.

3.2.6 Time effect

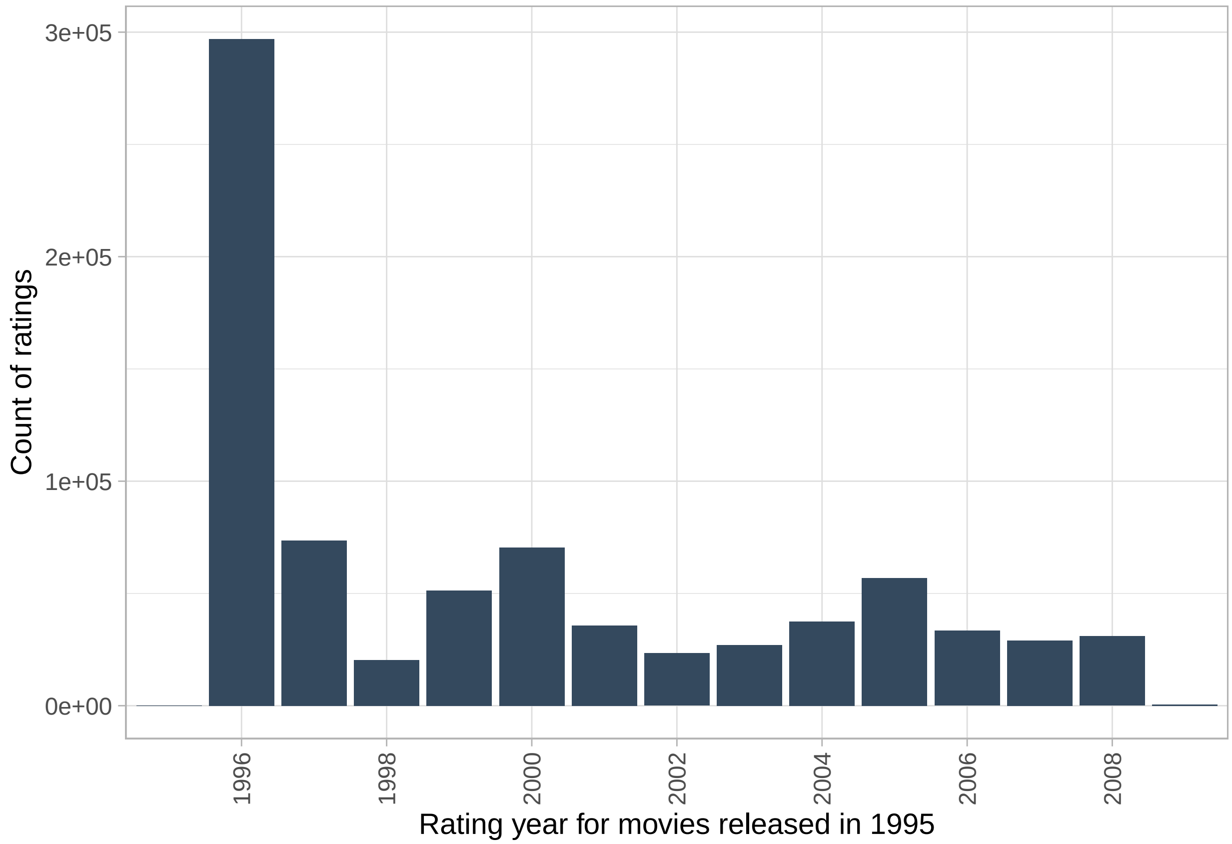
We find that there is an exponential increase in rating counts for movies released from around 1970 onward, reaching a peak in **1995**. Notable also is the even steeper exponential fall from the peak for movies released after **1995**.



The chart above does not correlate with the year in which ratings were made. For example, the same data when presented based upon the year during which ratings occurred, a relatively consistent volume of user activity can be noted.



It is thought that the reason for this disparity is that users continuously rate movies from prior years. This seems clear from the distribution of the year of ratings for the movies released in **1995**. Clearly most ratings for 1995 releases happened in 1996 but user activity does continue there after.



4 The model

For this assignment we will use a loss function to determine the viability of the model. If the predictions of the model are less accurate, the residual mean squared error (RMSE) loss function will output a higher value, the more accurate the prediction, the lower the RMSE value.

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

4.1 The first model

Our first model assumes that every movie will get the same rating based on the total average.

```
# The mean of all the ratings
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

Calculate the RMSE

```
# Calculate the RMSE
naive_rmse  <- RMSE(validation$rating, mu)
naive_rmse
```

```
## [1] 1.061202
```

The result is more than **1** which is slightly more than one star and much higher than our goal; less than **0.86490**.

We add our first result to an output table.

```
options(pillar.sigfig = 5)
results <- tibble(method = "Average", RMSE = naive_rmse)
results
```

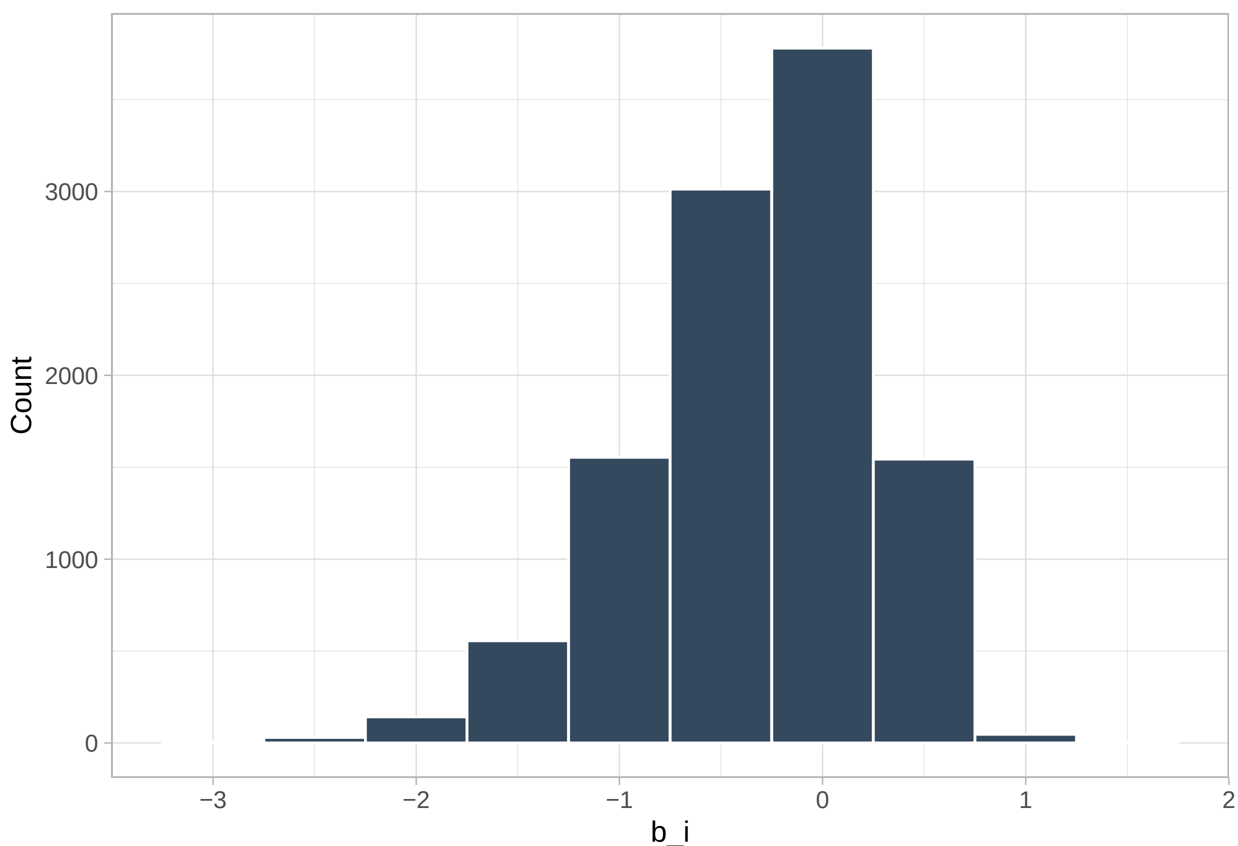
```
## # A tibble: 1 x 2
##   method   RMSE
##   <chr>   <dbl>
## 1 Average 1.0612
```

4.2 Movie effect

From sections 3.2.3 and 3.2.6 we learn that some movies are rated more often than others. To calculate this we could use the (lm) function, however, due to the amount of data it will be very slow. We know that the least squares estimate is just the average of (rating - mu) for each movie.

```
#Movie Effect
movie_effect <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))
```

The least square estimate plot shows a variation between -3 and 1.5, which again shows that the majority of the movies are rated around the 3.5 average.



```r
#Movie effect prediction
prediction <- validation %>%
  left_join(movie_effect, by='movieId') %>%
  mutate(p = mu + b_i)
```

We see an improvement of more than 11% compared with our first approach; RMSE **0.94391**.

```
## # A tibble: 2 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Average      1.0612
## 2 Movie effect 0.94391
```

4.3 User effect

In section **3.2.4** we have seen that users differently rate movies and realise that not every user rated the same amount of movies.

```r
#User Effect
user_effect <- edx %>%
  left_join(movie_effect, by='movieId') %>%
  group_by(userId) %>%
  summarise(u_i = mean(rating - mu - b_i))
```

```r
#User effect prediction
prediction <- validation %>%
  left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by='userId') %>%
  mutate(p = mu + b_i + u_i)
```

We see again an improvement with a **0.86535** RMSE.

```
## # A tibble: 3 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Average      1.0612
## 2 Movie effect 0.94391
## 3 User effect  0.86535
```

4.4 Genre effect

Form our data analysis we can conclude that some genres are better rated, section **3.2.5**. We also believe that the compound genres should not be divided because it eventually loses its meaning.

```r
#Genre Effect
genre_effect <- edx %>%
  left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by='userId') %>%
  group_by(genres) %>%
  summarise(g_i = mean(rating - mu - b_i - u_i))
```

```r
#Genre effect prediction
prediction <- validation %>%
  left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by='userId') %>%
  left_join(genre_effect, by='genres') %>%
  mutate(p = mu + b_i + u_i + g_i)
```

Applcation of this adjustment has the effect of lowering the RMSE slightly to **0.86495**.

```
## # A tibble: 4 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Average      1.0612
## 2 Movie effect 0.94391
## 3 User effect  0.86535
## 4 Genre effect 0.86495
```

4.5 Year effect

In section **3.2.6** we have shown that movies released in 1995 were more often rated than other years.

```r
#Year effect
year_effect <- edx %>%
  left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by='userId') %>%
  left_join(genre_effect, by='genres') %>%
  mutate(year = str_sub(title, -5, -2))  %>%
  group_by(year) %>%
  summarise(y_i = mean(rating - mu - b_i - u_i - g_i))
```

```r
#Year effects prediction
prediction <- validation %>%
  left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by='userId') %>%
  left_join(genre_effect, by='genres') %>%
  mutate(year = str_sub(title, -5, -2))  %>%
  left_join(year_effect, by='year') %>%
  mutate(pred = mu + b_i + u_i + g_i + y_i)
```

The release year has not a significant effect but already enough to reach our objective.

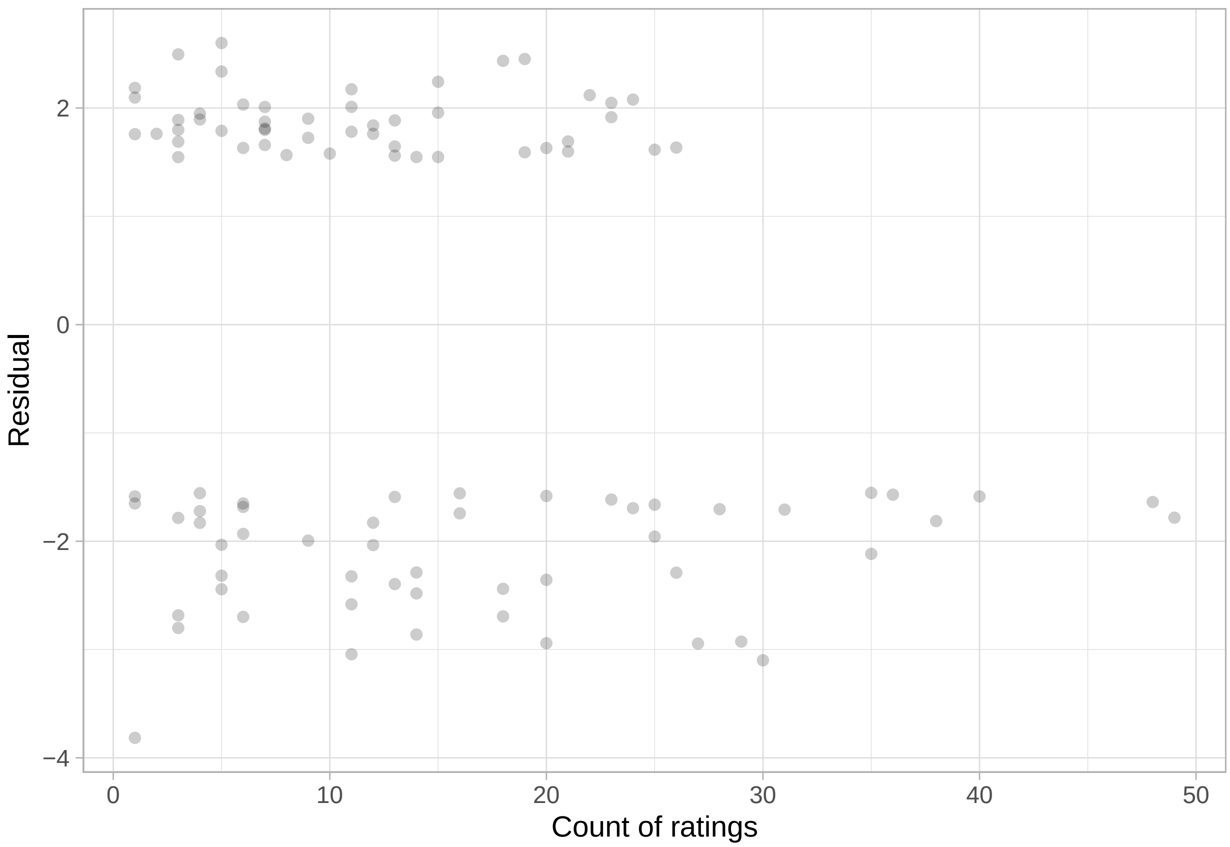
```
## # A tibble: 5 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Average      1.0612
## 2 Movie effect 0.94391
## 3 User effect  0.86535
## 4 Genre effect 0.86495
## 5 Year effect  0.86476
```

4.6 Regularisation

To improve our prediction we will examine the movies where there are large differences between actual ratings and those predicted by this model.

```r
#The residual is the difference between the average actual rating and the prediction
residuals <- prediction %>%
  group_by(movieId) %>%
  summarise(residual = mean(rating) - mean(pred)) %>%
  arrange(desc(abs(residual))) %>%
  slice(1:100)

#Count movies
edx_count <- edx %>%
```

```
  group_by(movieId) %>%
  summarise(n = n())

#plot the highest residuals in relation with the number of ratings of the training set
residuals %>%
  inner_join(edx_count, by = "movieId") %>%
  ggplot(aes(x = n, y = residual)) +
  geom_point(alpha = 2/10) +
  xlab("Count of ratings") +
  ylab("Residual") +
  theme_light()
```



We see from this plot that the movies with the highest residual are rated less than 40 times and in some cases just once. Knowing that the average number of ratings per movies is **843**, the prediction for movies rated less than **40** times in the training set is less accurate.

Show me the movies

```
#Get movie title
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()

residuals %>%
  inner_join(movie_titles, by = "movieId") %>%
  inner_join(edx_count, by = "movieId") %>%
  arrange(n) %>%
```

```
  slice(1:10)
```

movieId	residual	title	n
31692	-3.815073	Uncle Nino (2003)	1
63828	2.185435	Confessions of a Superhero (2007)	1
60391	2.096151	Aleksandra (2007)	1
64408	1.758392	Sun Shines Bright, The (1953)	1
5616	-1.651177	Mesmerist, The (2002)	1
3226	-1.585884	Hellhounds on My Trail (1999)	1
52561	1.761271	Summer Palace (Yihe yuan) (2006)	2
3193	-2.800696	Creature (1999)	3
56030	-2.684214	Darfur Now (2007)	3
50347	2.495233	Rosario Tijeras (2005)	3

The table (which shows the first 10 of the movies rated least often) indicates that these are obscure titles.

4.6.1 Penalise regression

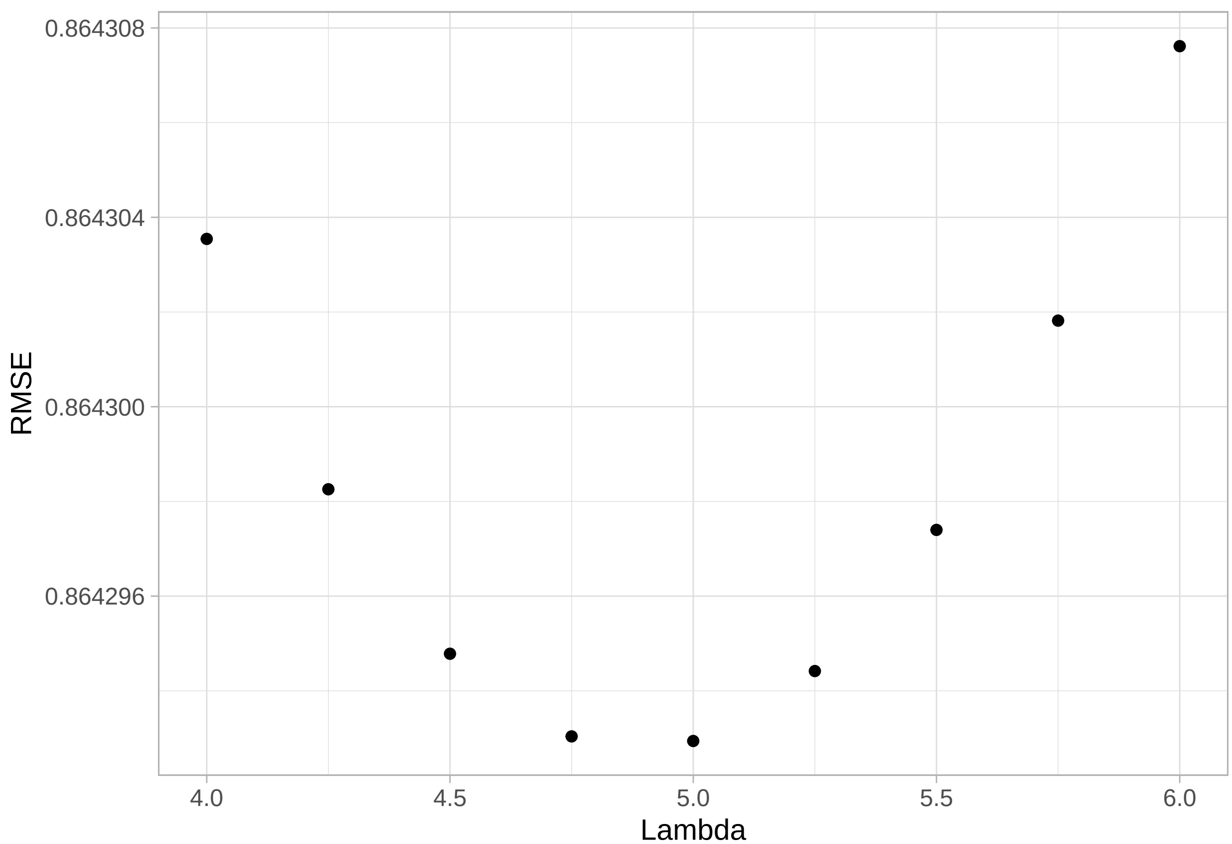
In order to remove the distortion of high ratings made in small numbers for obscure movies, we will adjust our calculations to weight these less highly in our calculations.

The general idea of penalised regression (weighting) is to control the total variability of the movie effects by adding a penalty. When our sample size is very large, the weighting value (represented by Lambda) is effectively ignored. When the sample is small, the estimated Lambda shrinks. However, we need to choose the right Lambda value. If the Lambda value is too high we run the risk of under-fitting the data.

```
#The lambda ranges, to choose an ideal one
lambdas <- seq(4, 6, 0.25)

rmses <- sapply(lambdas, function(l) {

  #Movie Effect
  movie_effect <- edx %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu) / (n() + l))

  #User Effect
  user_effect <- edx %>%
    left_join(movie_effect, by='movieId') %>%
    group_by(userId) %>%
    summarise(u_i = sum(rating - mu - b_i) / (n() + l))

  #Genre Effect
  genre_effect <- edx %>%
    left_join(movie_effect, by='movieId') %>%
    left_join(user_effect, by='userId') %>%
    group_by(genres) %>%
    summarise(g_i = sum(rating - mu - b_i - u_i) / (n() + l))

  #Year Effect
  year_effect <- edx %>%
    left_join(movie_effect, by='movieId') %>%
    left_join(user_effect, by='userId') %>%
```

```r
    left_join(genre_effect, by='genres') %>%
    mutate(year = str_sub(title, -5, -2))  %>%
    group_by(year) %>%
    summarise(y_i = sum(rating - mu - b_i - u_i - g_i) / (n() + l))


  #All effects prediction
  prediction <- validation %>%
    left_join(movie_effect, by='movieId') %>%
    left_join(user_effect, by='userId') %>%
    left_join(genre_effect, by='genres') %>%
    mutate(year = str_sub(title, -5, -2))  %>%
    left_join(year_effect, by='year') %>%
    mutate(p = mu + b_i + u_i + g_i + y_i)

  return(rmse = RMSE(validation$rating, prediction$p))
})
```

```
## [1] 0.8643035 0.8642983 0.8642948 0.8642930 0.8642929 0.8642944 0.8642974
## [8] 0.8643018 0.8643076
```




```
## [1] 5
```

The chart above shows clearly that a Lambda value of **5** produces the lowest RMSE (**0.8642929**)

The final result

```
## # A tibble: 6 x 2
##   method                    RMSE
##   <chr>                    <dbl>
## 1 Average                1.0612 
## 2 Movie effect           0.94391
## 3 User effect            0.86535
## 4 Genre effect           0.86495
## 5 Year effect            0.86476
## 6 Regularisation effect 0.86429
```

FIN