Movielens Recommendation System - edX Data science course PH125.9x

Gianluca Turco

February 27, 2019

1 Introduction

A recommendation system is a machine learning model that helps a user discover products and content by predicting the user’s rating of each item and showing them the items that they would rate highly. Recommendation systems are at the core of today’s most successful online companies such as Amazon, Google, Netflix and Spotify. By recommending new products that suit their customers’ tastes, these companies managed to effectively secure customer loyalty.

For this project, we will build a movie recommendation system using the 10M MovieLens Dataset, collected by GroupLens Research, which includes 10,000,000 ratings on 10,000 movies by 72,000 users.

10% percent of this dataset was partitioned to create a validation set that will be used to measure how well our algorithm scores on unknown data. I have further partitioned with the same ratio the rest of the data, labelled `edx`, into a training set, labelled `edx_train`, and a test set labelled `edx_test`, to test my algorithm before submission. The metric used to measure its performance will be the root-mean-square error (RMSE), i.e. the typical error of the predicted ratings compared to the actual ones.

The first part of the report will focus on exploring the dataset. Then, we will train different algorithms to find a model with the best possible RMSE. Finally, we will collect the results and conclude. All the analysis will be conducted using the `dplyr`, `tidyverse` and `caret` packages.

Note:

- For the sake of clarity in the context of this project, I have included in the report most of the code. This would not be the case in a real working environment.

- I was not able to install on the hardware at my disposal Tex, which is required to create a PDF output in R Markdown. I had to extract the output in HTML format and then convert it into PDF. I have left `pdf_document` as output in this file in case somebody wants to run it.

2 Dataset preparation and exploration

2.1 Data preparation

The dataset is downloaded and splitted into a training set (`edx`, 90% of the data) and a test set (`validation`, 10% of the data). All necessary R packages are downloaded and loaded to execute the analysis. The `edx` dataset is then further splitted with the same ratio into a train set `edx_train` (90% of the data) and a test set `edx_test` (10%).

2.2 Exploratory data analysis

Let’s start by exploring the `edx` dataset. Each row represents a single rating of a user for a single movie, and includes the movie title, genre and timestamp at which the rating was given.

```
  userId movieId rating timestamp                         title
1      1     122      5 838985046               Boomerang (1992)
2      1     185      5 838983525                Net, The (1995)
4      1     292      5 838983421                Outbreak (1995)
5      1     316      5 838983392                Stargate (1994)
6      1     329      5 838983392 Star Trek: Generations (1994)
7      1     355      5 838984474       Flintstones, The (1994)
                       genres
1                Comedy|Romance
2          Action|Crime|Thriller
4  Action|Drama|Sci-Fi|Thriller
5       Action|Adventure|Sci-Fi
6 Action|Adventure|Drama|Sci-Fi
7       Children|Comedy|Fantasy
```

The dataset contains over 9 million movie ratings of approximately 70,000 unique users giving ratings to over 10,600 different movies, with no missing values:

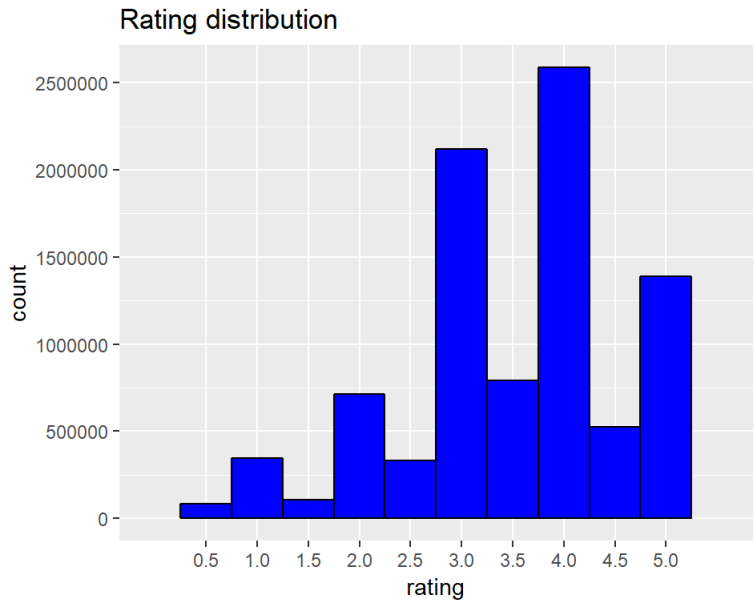
```
     userId         movieId          rating        timestamp
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
 1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
 Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
 Mean   :35870   Mean   : 4122   Mean   :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
    title              genres
 Length:9000055     Length:9000055
 Class :character   Class :character
 Mode  :character   Mode  :character
```

```
  number_of_users number_of_movies
1           69878            10677
```

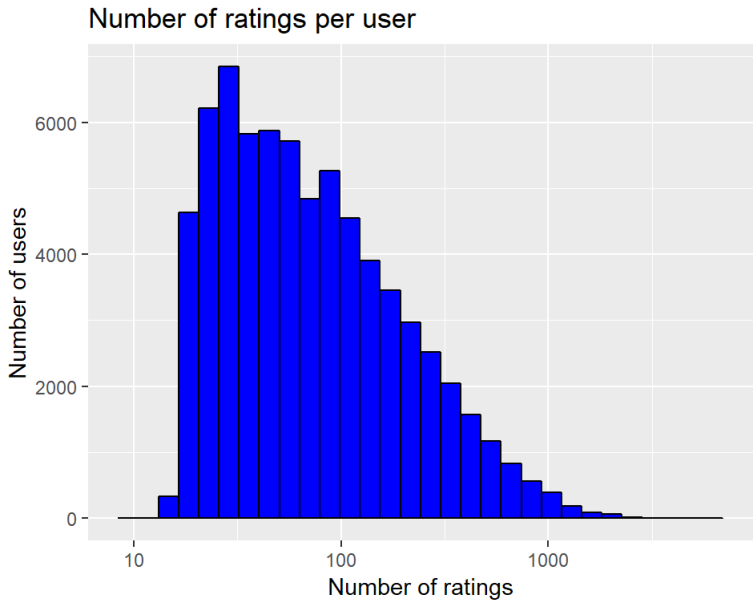
There are 10 distinct ratings, spanning from 0 to 5 with a 0.5 unit of variation. The distribution of ratings reveals that 4 is the most common rating, followed by 3 and 5:

```
   rating       n
1     4.0 2588430
2     3.0 2121240
3     5.0 1390114
4     3.5  791624
5     2.0  711422
6     4.5  526736
7     1.0  345679
8     2.5  333010
9     1.5  106426
10    0.5   85374
```

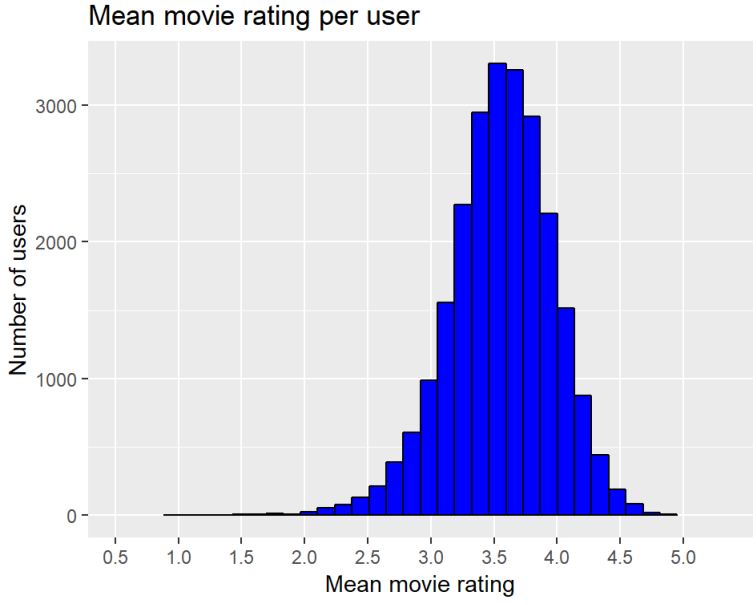
The following graph clearly shows that half-star ratings are much less frequent than full-star ratings:



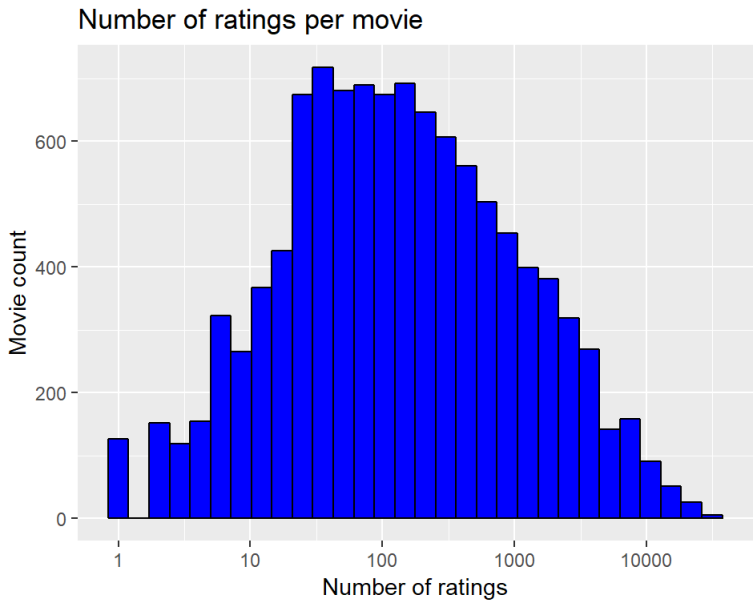
Users behaviour varies considerably. Some users rated several hundred movies, others only a few:



Morevover, we observe a large variation in how users rated the movies. Some users tend in fact to give consideraly lower or higher ratings than average. The following graph shows the mean movie rating only for users that have rated at least 100 movies:



Another important factor to consider is the number of times each movie have been rated, as this will influence the accuracy of our model. We note that some movies have been rated only a handful of times, while others have been rated thousands of times:



We even have 126 movies which were rated only once:

```
# A tibble: 1 x 2
  count     n
  <int> <int>
1     1   126
```

These variations will have to be taken into account when building our models.

3 Modelling Approaches

In this section we will explore diffenet approaches, creating and training several algorithms in order to identify the best one. The metric used to measure the performance of each will be the Residual Mean Squared Error (RMSE), i.e. the typical star rating error we would make upon predicting a movie rating. The lower the RMSE, the better the model will perform.

We define $y_{u,i}$ the rating for movie i given by user u and N the number of user/movie combinations. We denote our rating predictions from user u for movie i as $\hat{y}_{u,i}$.

$$ RMSE = \sqrt{\frac{1}{N}\displaystyle\sum_{u,i} (\hat{y}_{u,i}-y_{u,i})^{2}} $$

Note: In this report we will drop the estimator "hat" (as in $\hat{y}$) from the following equations.

3.1 Model based approach: Simple Average Algorithm

The first step will be to set a baseline by predicting new movie ratings based on the average rating value in the train edx set. The model is simply expressed as

$$Y_{u,i}=\mu+\epsilon_{u,i}$$

with sampling error ϵ and μ being single average value of all ratings. The average is then evaluated against the rating in the test set, and this evaluation will be our baseline RMSE to compare future modelling approaches against. We note that the mean movie rating μ is slightly more than 3.5 stars.

```
mu <- mean(edx_train$rating)
mu
```

```
## [1] 3.512457
```

For easier comparison, we will record our approaches and the RMSEs they generate in a table. With this first approach, by predicting a new rating based on the average rating we are typically off by over one star in rating (RMSE > 1.06).

```
#The following code generates a function that will calculate the RMSE for actual values (true_ratings) from
our test set
# to their corresponding predictors from our models
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# The RMSE function computes the root mean squared error between actual and predicted ratings (mu).
baseline_rmse <- RMSE(edx_test$rating,mu)

# Let's create a results table with this baseline model approach:
rmse_results <- data_frame(Method = "Baseline", RMSE = baseline_rmse)
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline	1.060054

To improve upon this simple approach, we will factor in some of the insights we gained during the exploratory data analysis.

3.2 “Movie-effect” model

This model takes into account the fact that not all movies are equally popular or good. Each movie will therefore have its own bias effect on the ratings, which will be higher or lower than the average movie rating \(\mu\).

We will then calculate the estimated deviation of each movie’s mean rating from the total mean of all movies \(\mu\). The resulting variable will be called “b” (as in “bias”) for each movie “i”: \({b}_{i}\), which represents the average ranking of movie \(i\). We will then include this variable to our previous model equation:

\[Y_{u, i} = \mu + b_{i} + \epsilon_{u, i} \]

Our new prediction will take into account the fact that movies are rated differently by adding the computed \(b_{i}\) to \(\mu\). If a movie is on average rated better than the average rating of all movies \(\mu\), we predict that it will be rated better than \(\mu\) by \(b_{i}\), the difference of the individual movie average from the total average.

```
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

# The new predicition takes into account the fact that different movies are rated differently by adding the
movie bias (b_i) to the total mean of all movies (mu).

predicted_ratings <- mu + edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

Movie_effect_rmse <- RMSE(predicted_ratings, edx_test$rating)

rmse_results <- bind_rows(rmse_results, data_frame(Method = "Movie Effect Model", RMSE = Movie_effect_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline	1.0600537
Movie Effect Model	0.9429615

We note a significant improvement on the RMSE using this approach. Let’s see if our previous observations on user rating distribution could allow us to improve on this result.

3.3 “Movie + User-effect” model

We observed above that different users have a different rating behaviour: some tend to be very generous, some less so. Also, we noted that the mean rating in the dataset exceeds 3.5 stars. In this approach we will compute this bias, the “User-effect” \(b_{u}\). We will then predict the ratings taking into account movie and user effects together (\(b_{i}\) + \(b_{u}\)).

Our new model is the following:

\[Y_{u, i} = \mu + b_{i} + b_{u} + \epsilon_{u, i} \]

And the corresponding RMSE value:

```
# The following code computes the user effect, which we will call "b_u".
user_avgs <- edx_train %>%
  left_join(movie_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

# We predict ratings taking into account "b_i" and "b_u".
predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

User_effect_RMSE <- RMSE(predicted_ratings, edx_test$rating)

rmse_results <- bind_rows(rmse_results, data_frame(Method = "Movie & User Effects Model", RMSE = User_effect
_RMSE))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline	1.0600537
Movie Effect Model	0.9429615
Movie & User Effects Model	0.8646844

Including the user-effect b_{u} in our rating predictions allowed us to further reduce the RMSE, which is now close to 0.865.

This model, however, does not take into consideration the fact that some rating averages are based on a very limited number of ratings, as we observed above. We can counterbalance this effect by applying the regularization technique, which permits us to penalize large estimates that are formed using small sample sizes.

3.4 Regularized “Movie-effect” model

The following table shows the most and least popular movies (largest b_{i}) according to the predictions of the movie-effect model:

title	b_i	n
Hellhounds on My Trail (1999)	1.487543	1
Satan’s Tango (Sátántangó) (1994)	1.487543	1
Shadows of Forgotten Ancestors (1964)	1.487543	1
Fighting Elegy (Kenka erejii) (1966)	1.487543	1
Sun Alley (Sonnenallee) (1999)	1.487543	1
Blue Light, The (Das Blaue Licht) (1932)	1.487543	1
Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237543	4
Life of Oharu, The (Saikaku ichidai onna) (1952)	1.237543	2
Human Condition II, The (Ningen no joken II) (1959)	1.237543	4
Human Condition III, The (Ningen no joken III) (1961)	1.237543	4

title	b_i	n
Besotted (2001)	-3.012457	1
Hi-Line, The (1999)	-3.012457	1
Confessions of a Superhero (2007)	-3.012457	1
War of the Worlds 2: The Next Wave (2008)	-3.012457	2
SuperBabies: Baby Geniuses 2 (2004)	-2.767776	47
Disaster Movie (2008)	-2.745790	30

title	b_i	n
From Justin to Kelly (2003)	-2.638140	183
Hip Hop Witch, Da (2000)	-2.603366	11
Criminals (1996)	-2.512457	1
Mountain Eagle, The (1926)	-2.512457	2

These are indeed little-known titles, with only a limited number of ratings (n). The model, in other words, is is too simple and very likely to capture the noise of the data.

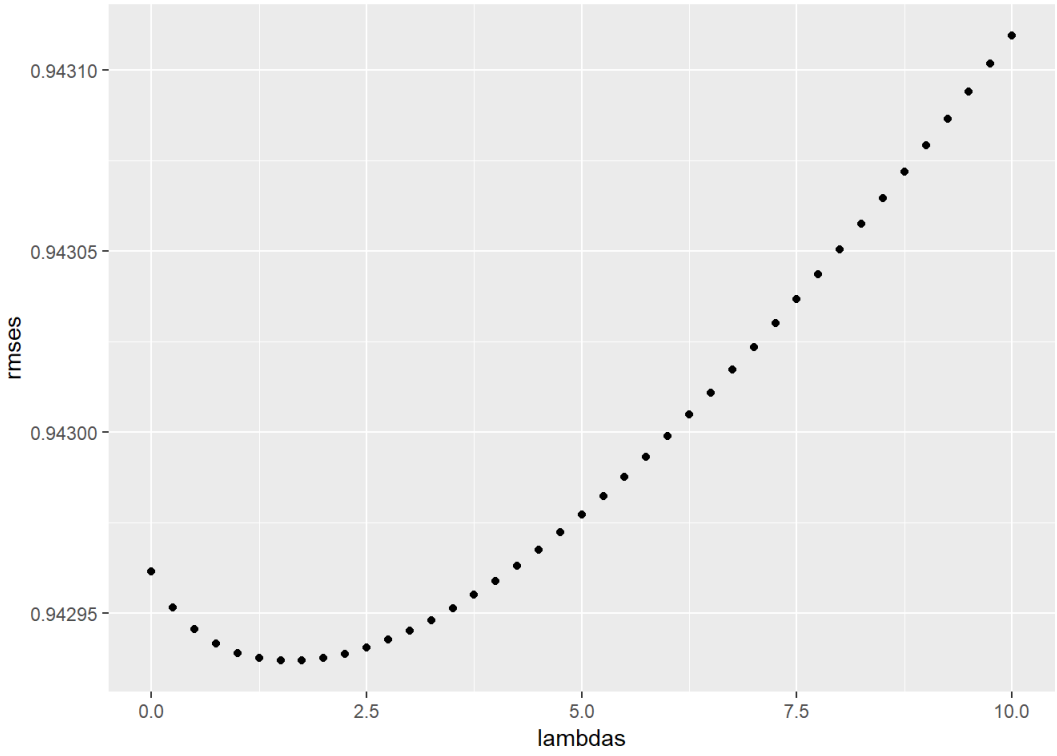
The regularization technique can be applied to mitigate this effect by adding a penalty term lambda \(\lambda\). Essentially, our objective is to penalize/minimize the equation with the lambda in case of small number of ratings. The greater value of the lambda, the more the bias \(b_{i}\) and \(b_{u}\) value will shrink. Consequently, a large number of ratings (n) will reduce the lambda value and give more weigth to the model estimate.

Lambda is a parameter which can be optimised. We will use cross-validation and compute RMSEs for different penalties (from 0 to 10 at a sequence of .25) applied to the movie bias \(b_{i}\). We can thus determine the optimum lambda, i.e. the one that will minimize the RMSE.

```
lambdas <- seq(0, 10, 0.25)

mu <- mean(edx_train$rating)
just_the_sum <- edx_train %>%
  group_by(movieId) %>%
  summarise(s = sum(rating - mu), n_i = n())

rmses <- sapply(lambdas, function(l){
  predicted_ratings <- edx_test %>%
    left_join(just_the_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx_test$rating))
})
qplot(lambdas, rmses)
```

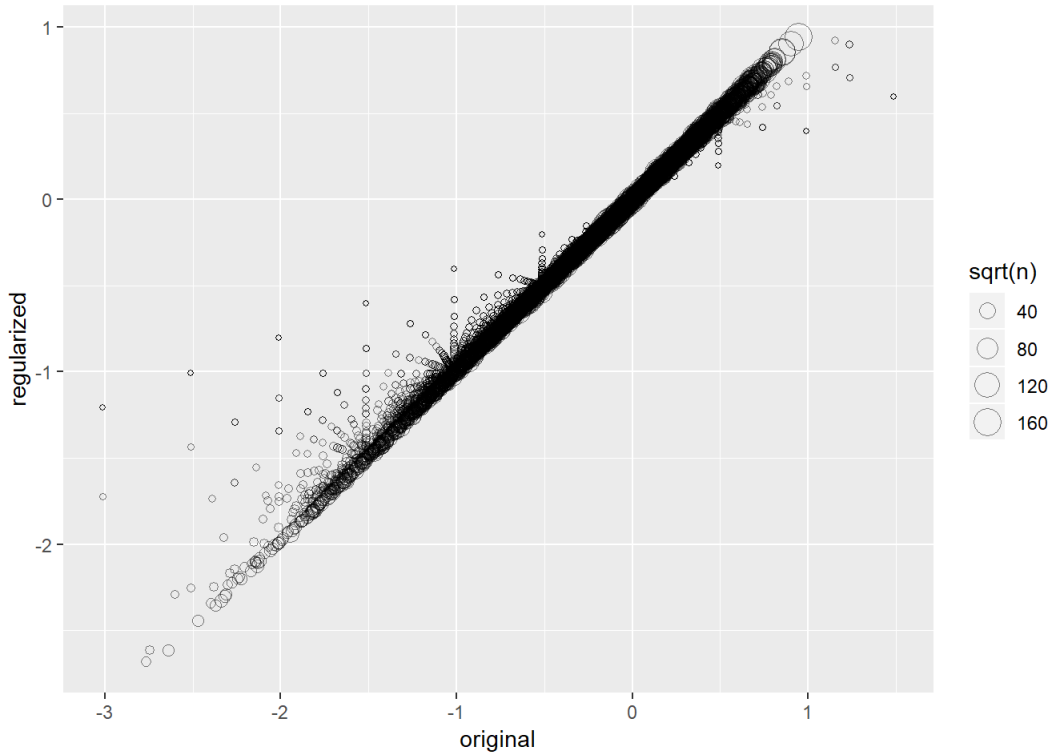



```
lambdas[which.min(rmses)]
```

```
## [1] 1.5
```

The lowest lambda according to this calculation is 1.5. Let's compute the regularized estimates of \(b_{i}\) using this value.

This graph shows how the estimates shrink with the penalty:



Most movies rated the highest and lowest in our prediction after movie-effect regularization are well-known titles which have been rated several times. The following are the top 10 movies after regularization:

```
Joining, by = "movieId"
```

title	b_i	n
Shawshank Redemption, The (1994)	0.9440536	25188
More (1998)	0.9233679	6
Godfather, The (1972)	0.9041093	15975
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	0.9000314	4
Human Condition II, The (Ningen no joken II) (1959)	0.9000314	4
Human Condition III, The (Ningen no joken III) (1961)	0.9000314	4
Usual Suspects, The (1995)	0.8540293	19457
Schindler's List (1993)	0.8515669	20877
Rear Window (1954)	0.8123193	7115
Casablanca (1942)	0.8070669	10141

And these are the lowest-rated ones:

```
Joining, by = "movieId"
```

title	b_i	n
SuperBabies: Baby Geniuses 2 (2004)	-2.682175	47
From Justin to Kelly (2003)	-2.616691	183
Disaster Movie (2008)	-2.615038	30
Pokémon Heroes (2003)	-2.442587	124
Barney's Great Adventure (1998)	-2.353690	186
Carnosaur 3: Primal Species (1996)	-2.340158	61
Glitter (2001)	-2.327597	311

title	b_i	n
Gigli (2003)	-2.302656	281
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002)	-2.292042	188
Hip Hop Witch, Da (2000)	-2.290962	11

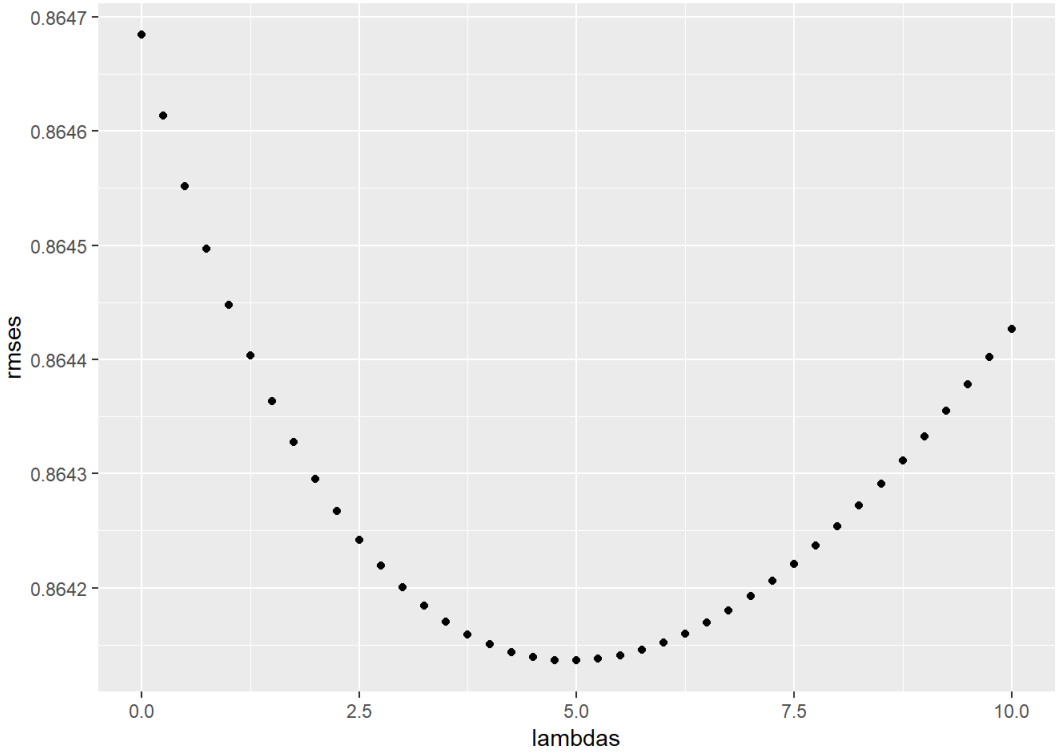
We can now calculate the RMSE for the regularized movie-effect model and compare it to the previous ones:

Method	RMSE
Baseline	1.0600537
Movie Effect Model	0.9429615
Movie & User Effects Model	0.8646844
Regularized Movie Effect Model	0.9429370

We can conclude that regularization did not impact significantly the movie-effect RMSE, but did it improve the best and worst predicted movies.

3.5 Regularized “Movie + user effect” model

To improve upon our RMSE, let’s include a regularized user effect \(b_{u}\) into the model above and apply the same approach.



The lowest lambda in this case is 5:

```
# Lambda for the lowest RMSE:

lambda <- lambdas[which.min(rmses)]
lambda
```

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

Let’s check check how this new model performs on the `edx_test` set and compare it with the previous RMSEs:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Regularized Movie + User Effect Model - test dataset",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline	1.0600537
Movie Effect Model	0.9429615
Movie & User Effects Model	0.8646844
Regularized Movie Effect Model	0.9429370
Regularized Movie + User Effect Model - test dataset	0.8641362

By applying regularization, we observe a slight improvement on our previous “Movie + User Effects Model”.

4 Results

We can now predict ratings on the on the `validation` dataset, which we have ignored so far in our modelling since it is supposed to represent the unknown. We observe that the algorithm is performing slightly worse on `validation` than on the `edx test set`, but still under the threshold indicated for this project.

```
# Calculate b_i with lambda penalty term:
b_i <- edx %>% group_by(movieId) %>% summarise(b_i = sum(rating - mu)/(n()+lambda))

# Calculate b_u with lambda penalty term:
b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>% summarise(b_u = sum(rating - b_i - mu)/
(n()+lambda))

#predict ratings with the validation set:
predicted_ratings <- validation%>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% muta
te(pred = mu + b_i + b_u) %>% .$pred

# Calculate RMSE:
validation_rmse <- RMSE(predicted_ratings, validation$rating)

# Append the results to rmse_results:
rmse_results <- bind_rows(rmse_results,data_frame(Method="Regularized Movie + User Effect Model - validation
dataset",RMSE = validation_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline	1.0600537
Movie Effect Model	0.9429615
Movie & User Effects Model	0.8646844
Regularized Movie Effect Model	0.9429370
Regularized Movie + User Effect Model - test dataset	0.8641362
Regularized Movie + User Effect Model - validation dataset	0.8648177

5 Conclusion

For this capstone project, we have built a model to predict movie ratings which performs reasonably well for the average users. I refrained from exploring other possible approaches (e.g. incorporating in the model factors as the genre, release date and the review date) because of the size of the dataset and the limited hardware at my disposal. Running the code attached was already a challenge form me, as it took a long time and often caused the computer to crash.

We found that taking into account the user and the movie effect together brings the largest improvement in RMSE, which can be still sligtly improved applying the regularization technique. The improvement is especially effective on movies with small sample sizes and allows us to produce more reliable results.