Movielens Project - Capstone

Eline Morais

31/05/2020

1. Introduction

This project aims to create a movie recommendation system using the MovieLens dataset as part of the final evaluation to the Data Science Professional Certificate from Harvardx-Edx, PH125.9x:Data Science: Capstone.

Recommendation systems are more complicated than other machine learning challenges because each outcome has a different set of predictors. For example, different users rate a different number of movies and rate different movies.To build this recomendation system we must use the 10M version of the MovieLens dataset to make the computation a little easier and we must use all the tools we have learned throughout the courses in this series.

The dataset will be divided into two parts: 1- Edx set: which must be used to build and train the algorithm to make movie rating predictions, and 2- Validation set: which must be used only at the end to evaluate how close predictions are to the true values by the Root-mean-square error (RMSE).

To this project is expected to obtain a RMSE smaller than 0.8649.

This report contains all steps taken to achieve the goal, starting by the given code to download dataset and create edx and validation sets, followed by an exploratory analysis of data, application of machine learning algorithms and results achieved.

2. Analysis section

2.1 Create test and validation sets

First, we use the code provided to download the MovieLens data and create the trainig and test set. We also call all the libraries required to analyse data.

2.2 Exploratory analysis

The second step is to understand our Edx data looking for standards, distributions, to have insights that can help us to increase accuracy of our algorithm.

Edx dataset shows 9.000.055 observations and 6 variables. Each observation (rows) represents a rating given by one user to a different movie. The 6 variables availables are: userID, movieID, rating, date and time of rating (timestamp), title and genres as shown:

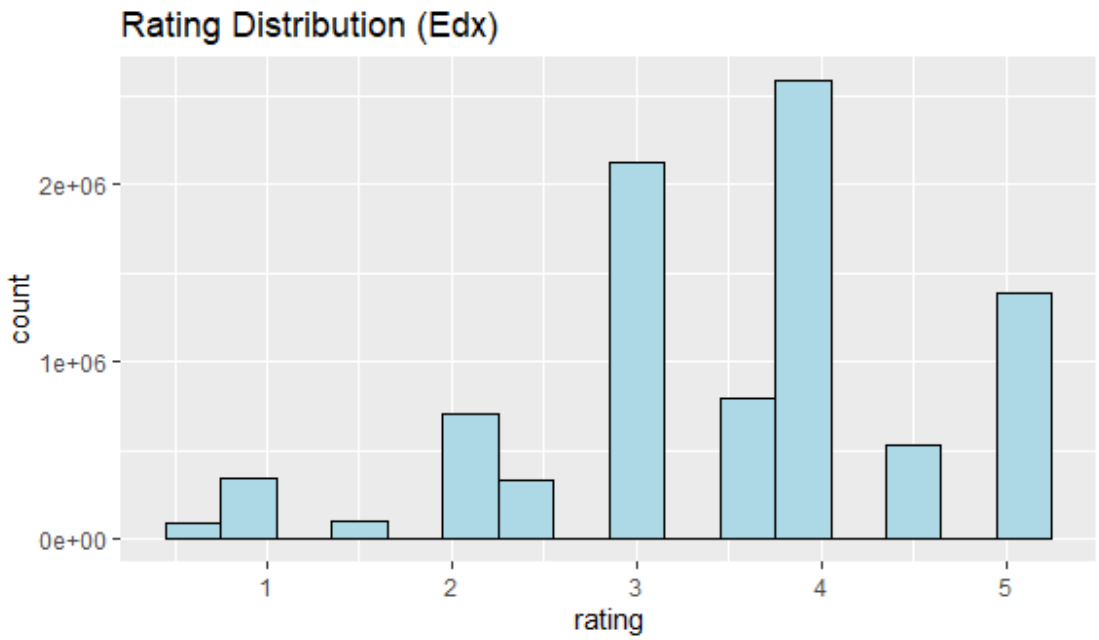
```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               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
```

Edx presents 10.677 different movies and 69.878 different users. Let's also have a look in the summary of edx:

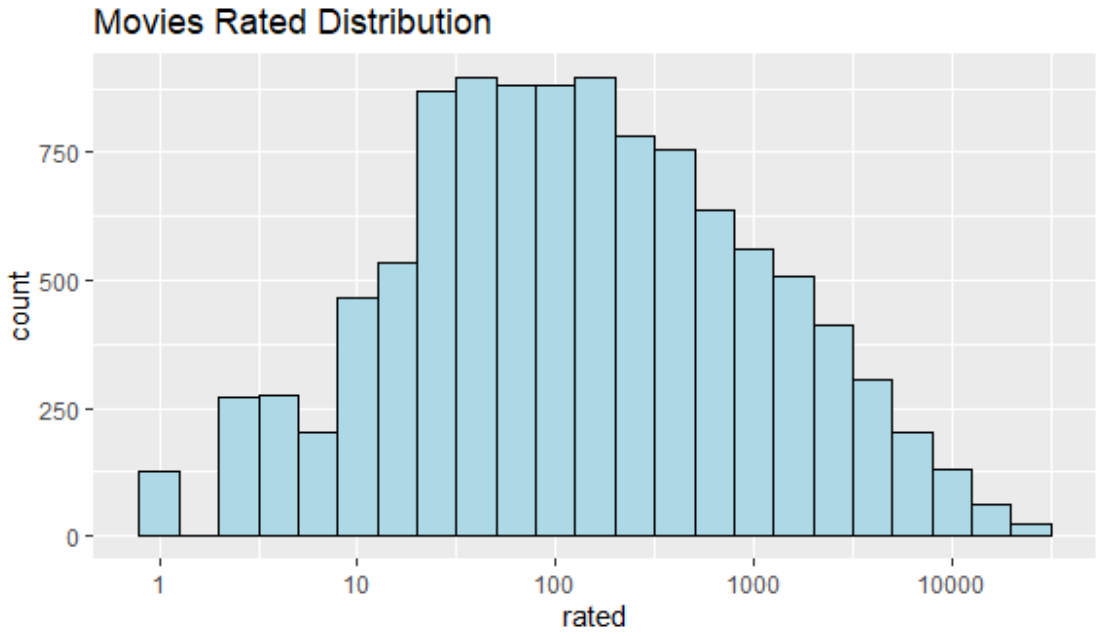
```
## # A tibble: 1 x 4
##   Movies Users Genres Ratings
##    <int> <int>  <int>   <int>
## 1  10677 69878    797      10

##      userId         movieId          rating       timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.50   Min.   :7.90e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.00   1st Qu.:9.47e+08
##  Median :35738   Median : 1834   Median :4.00   Median :1.04e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.51   Mean   :1.03e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.00   3rd Qu.:1.13e+09
##  Max.   :71567   Max.   :65133   Max.   :5.00   Max.   :1.23e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

We can see 10 different rating scores, between 0.5 and 5, being more common high and whole grades: 4, 3 and 5, as shown on the histogram below.



Some movies are rated more often than others. There are about 1150 movies rated less than 10 times and a very few movies rated more than 10.000 times, as we can see below:

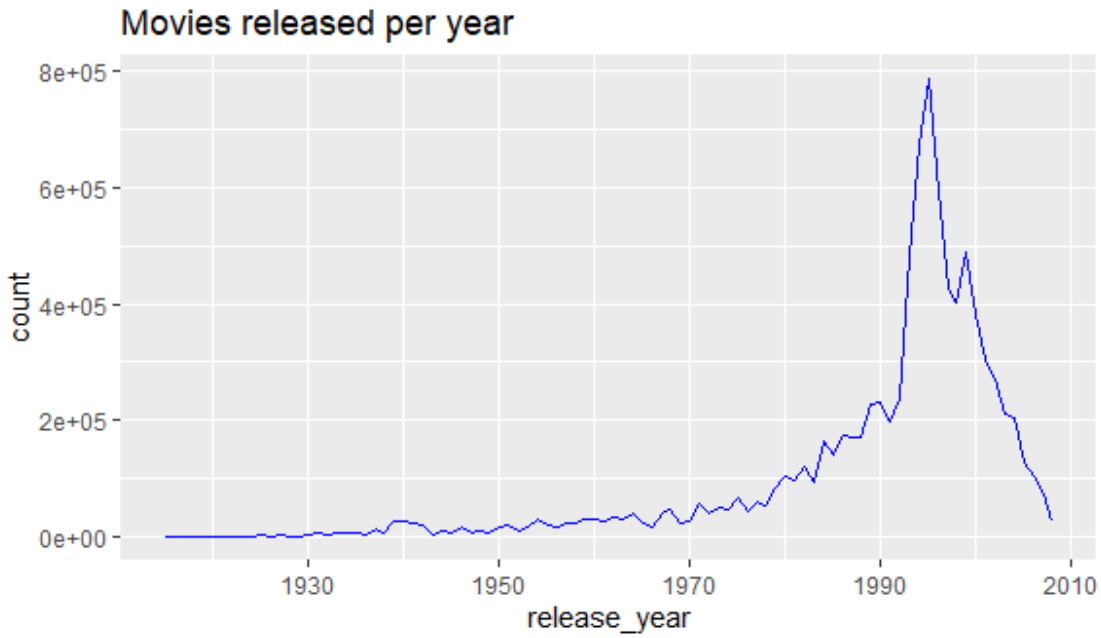


To see if there is any relation between released year and ratings it will be necessary to extract release year from “title” into a separate column. After doing this, we can observe that:

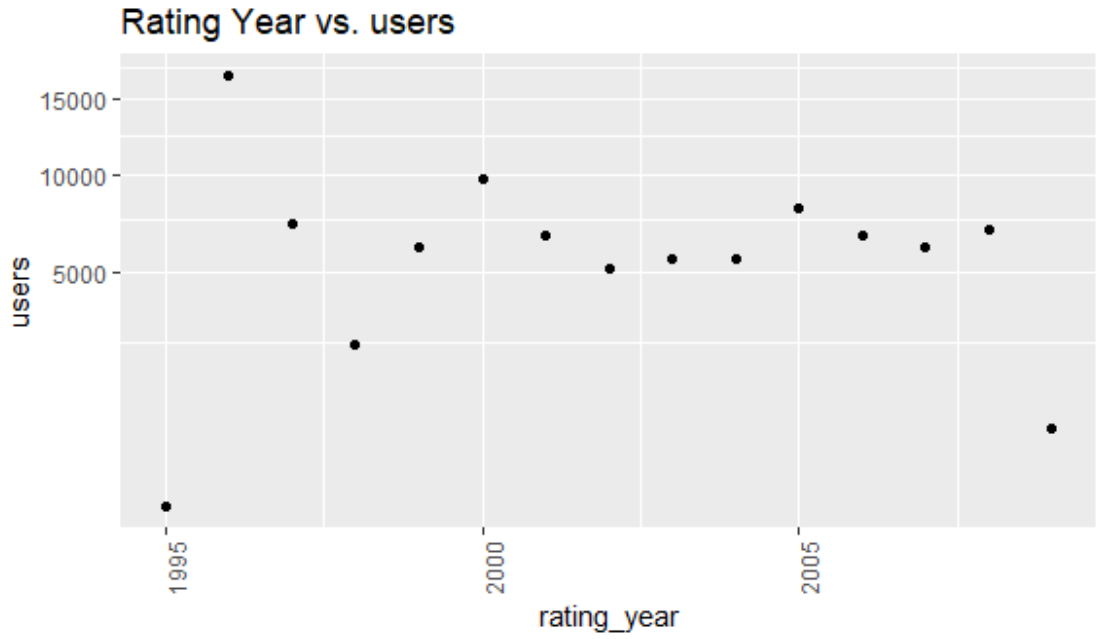
1- Rated movies were released from 1915 until 2008 but evaluation (ratings) period was from 1995 to 2009.

```
##   first_release last_release first_rating last_rating
## 1          1915         2008         1995        2009
```

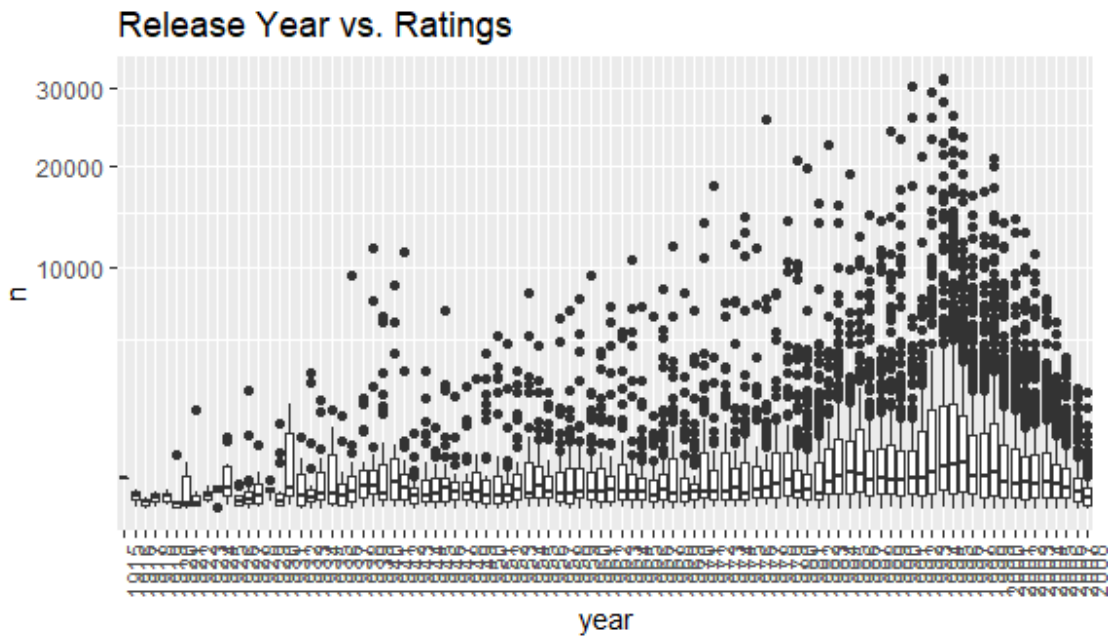
2- There is an exponential growth of releasing movies after 1990



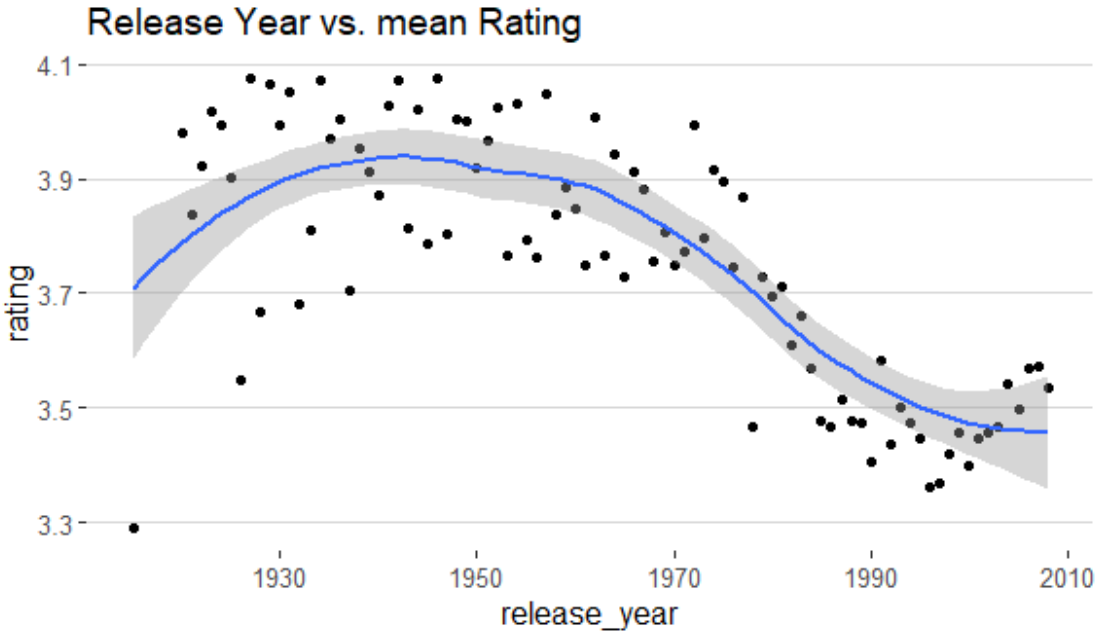
In addition, movies started to be rating in 1995 and after 3 years there was a stabilization of the number of users rating movies. We can see a linear trend since 1999 until 2009 which data ratings seems to be incomplete.



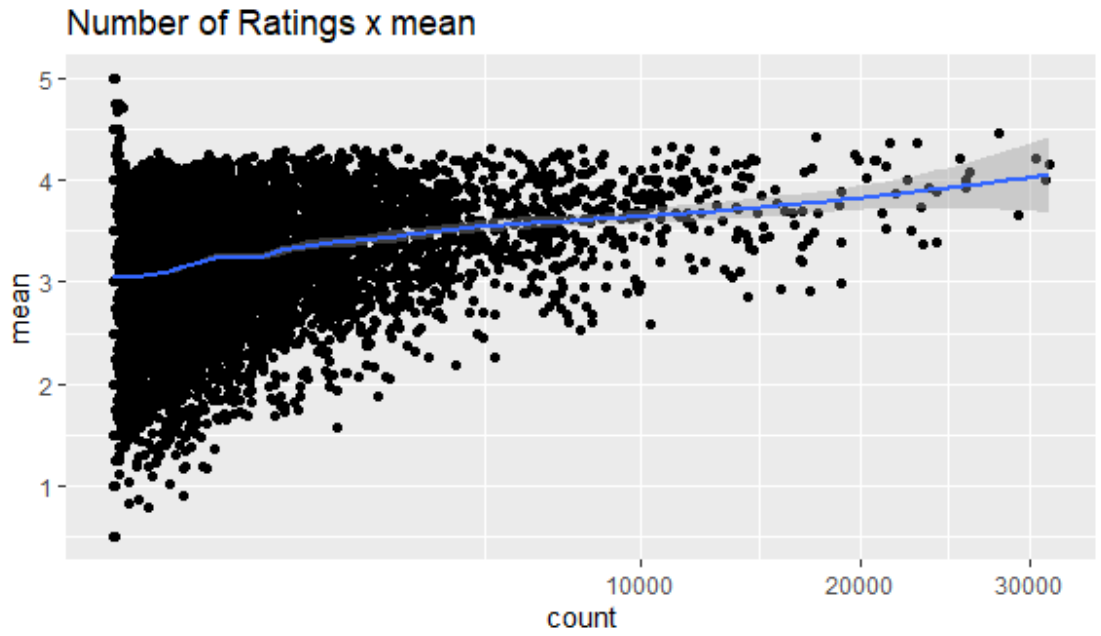
We also see that, on average, movies that came out after 1993 get more ratings.



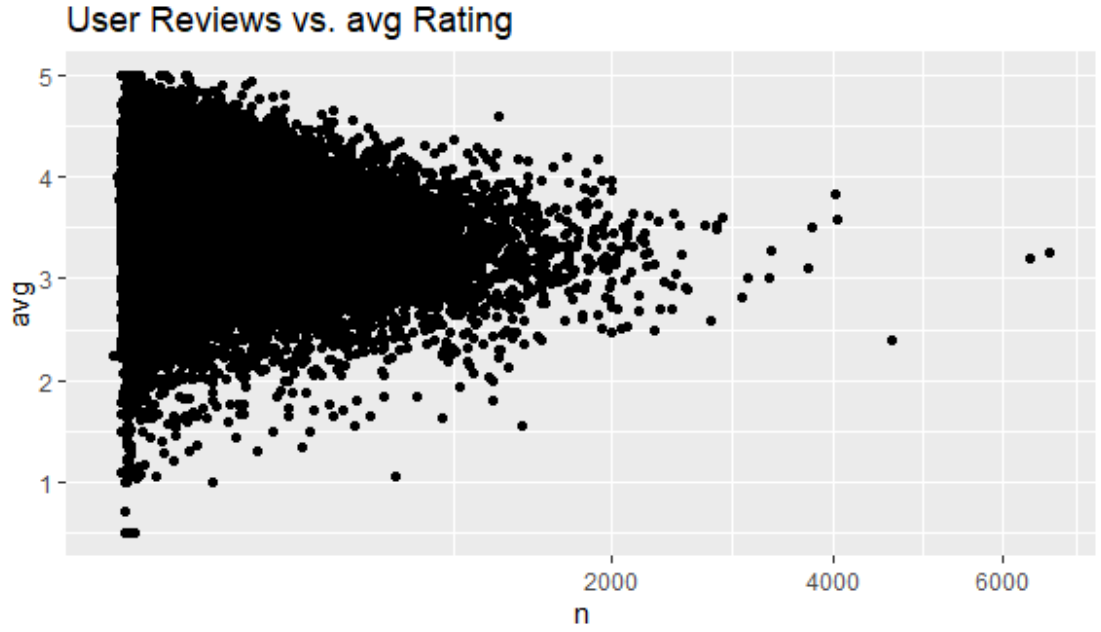
We see there is a influence of released year and the mean rating, so we can say that older “classics” get less and higher ratings.



Another interesting trend is that the most rated movies tend to have above average ratings. This is related to the fact that more people watch popular movies. To confirm this, follow the plot of average rating versus number of ratings with an estimate of the trend.

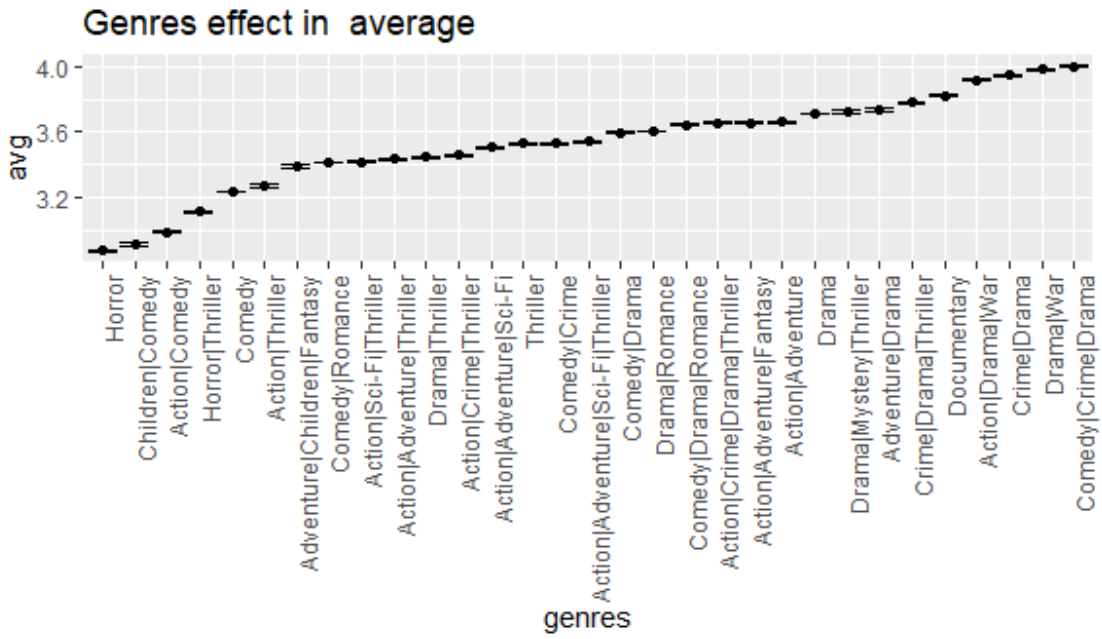


Yet, we can look for correlation between number of times each user has reviewed movies and its average. It seems that there is no correlation between number of ratings by one user and its average. We can say only that some users are chunky and others like most movies they watch.



Some movies fall under several genres, which results in the 797 different combination of genres present on the dataset. Defining a category as whatever combination appears in this “genres” column we can plot a bar plot to see if there is also genre effect.

To facilitate the plot visualization we are going to keep only categories with more than 50,000 ratings.



The plot shows strong evidence of genre effect.

2.3 Model Building and Training

Now that we have analysed data we can start building our prediction model. The first step consists in split our edx dataset in training and test set, once we can only use validation set at the end when we have our best predictive modelling.

Now we have a training dataset with 8.100.065 rows and 8 columns and a test set with 899.990 rowns and 8 columns since we have added 2 columns: release_year and date.

Second, we define the function that will be used for calculate the RMSE.

```
#define a function to calculate the residual mean squared error (RMSE)
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Now we can start to predict rating using different approaches.

2.3.1 Naive approach

The most simple approach to define a baseline to our models comparison is predicting the same rating to any movies and users, calculated by the mean rating of all movies in our training dataset, which gives us a movie rating pretty generous = 3.5.

Using the RMSE funcion we created, we obtain a RMSE of 1.060054, as we can see:

```r
#mean
mu_hat<- mean(edx_train$rating)

#RMSE
mean_rmse <- RMSE(edx_test$rating, mu_hat)
mean_rmse

## [1] 1.06005
```

We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If this number is larger than 1, it means our typical error is larger than one star, which is not good.

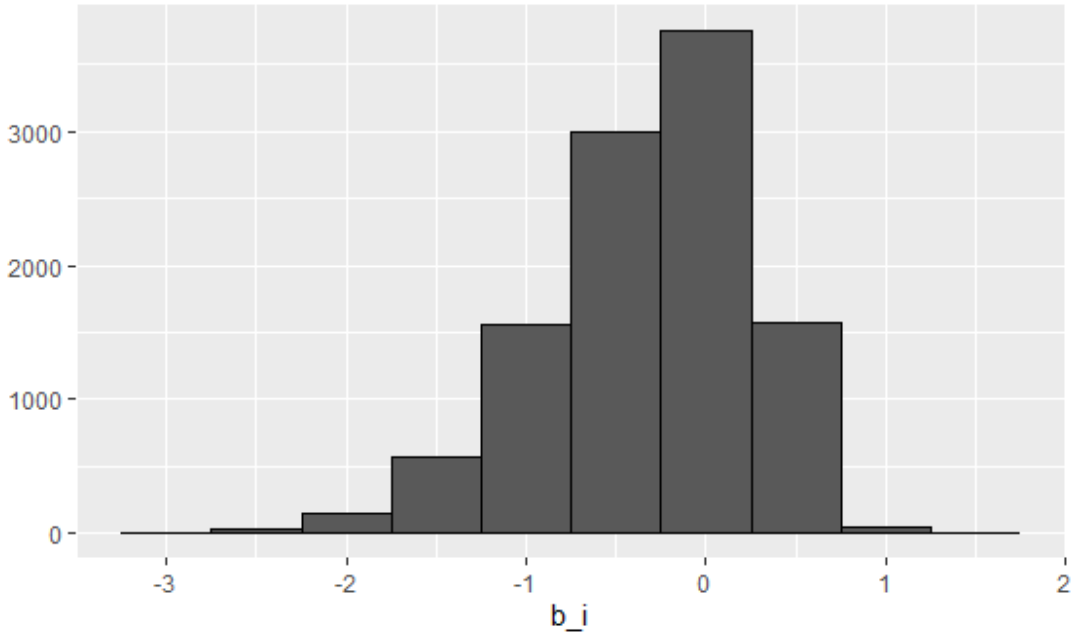
2.3.2 Modeling movie effects

We could think of this data as a very large matrix, with users on rows and movies on columns, with rating on cells and NAs for the movies that users did not rate. But due to the size of the dataset it will crash R.

As we know from previous analysis some movies are just generally rated higher than others we can add to our previous model the term b_i to represent average ranking for movie i. So, now for each movie we will calculate the average distance from the total average, like this:

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

We can see the distribution below.



Now we can add this b_i term to our naive model and predict ratings on the test set.

```
movie_hat <- mu + edx_test %>%
left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
```

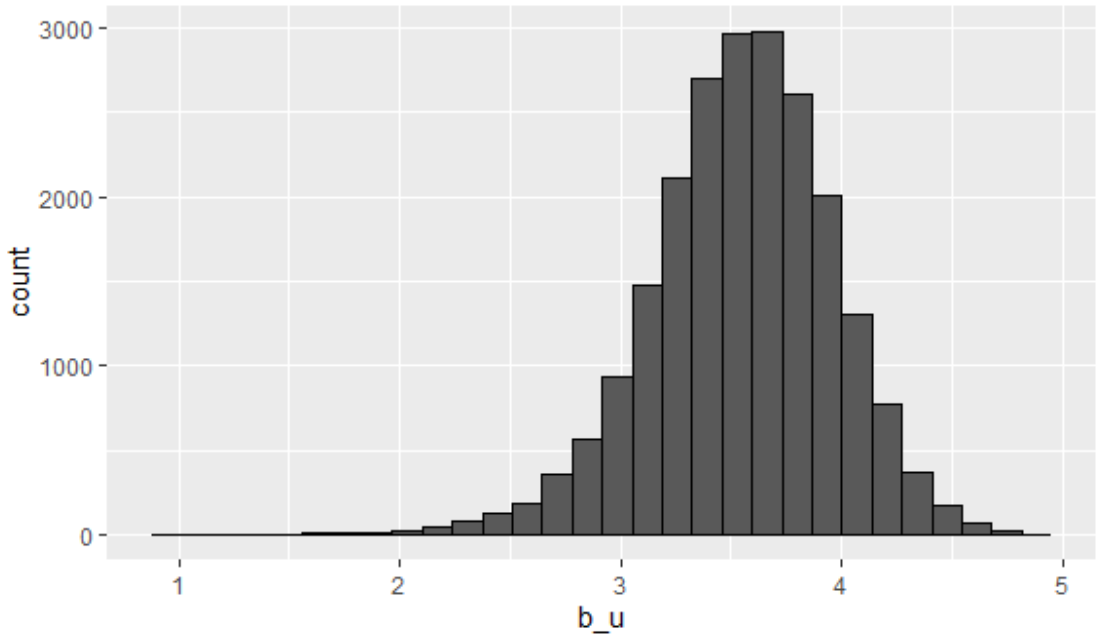
This resulted in a RMSE equal to 0,9429615, an improvement of 10%.

```
movie_rmse<- RMSE(edx_test$rating, movie_hat)
movie_rmse

## [1] 0.942961
```

2.3.3 Modeling user effects

At the same way, we have noticed that there is substantial variability across users: some users are very cranky and others love every movie. This implies that a further improvement to our model may be done adding a term b_u representing the mean distance from the previous prediction and the real rating. We can see the b_u distribution below.




```
user_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
head(user_avgs)

## # A tibble: 6 x 2
##   userId    b_u
##    <int>  <dbl>
## 1      1  1.67
```

```
## 2      2 -0.434
## 3      3  0.268
## 4      4  0.698
## 5      5  0.100
## 6      6  0.336
```

We calculate the b_u, wich will be added to predict the rating for each user, resulting in a RMSE of 0.8646843. Much better than previous one.

```
user_hat <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

user_rmse<- RMSE(edx_test$rating, user_hat)
user_rmse

## [1] 0.864684
```

2.3.4 Modeling genres and year effects

Following this method we can also add genres and release year effects in our prediction since we noticed all variables impacts on ratings values.

The results are shown below.

```
##              method     RMSE
## 1 Just the average 1.060054
## 2     Movie Effect 0.942961
## 3      User Effect 0.864684
## 4    Genres effect 0.864324
## 5      Year effect 0.864126
```

2.3.5 Regularization

We have improved a lot since adding movie and user effect, but it is becoming harder to have significant increases even adding genres and year effect. Let's explore where we have made largest mistakes in our first model, using only movie effects b_i.

Here are the 10 largest mistakes:

title	residual
From Justin to Kelly	4.12568
Shawshank Redemption, The	-3.95657
Shawshank Redemption, The	-3.95657
Godfather, The	-3.91665
Godfather, The	-3.91665

Godfather, The	-3.91665
Godfather, The	-3.91665
Usual Suspects, The	-3.86655
Schindler's List	-3.86409
Schindler's List	-3.86409

Let's see the 10 best and worst movies and the number of ratings they had.

The best:

title	b_i	n
Hellhounds on My Trail	1.48754	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³)	1.48754	1
Shadows of Forgotten Ancestors	1.48754	1
Fighting Elegy (Kenka erejii)	1.48754	1
Sun Alley (Sonnenallee)	1.48754	1
Blue Light, The (Das Blaue Licht)	1.48754	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	1.23754	4
Life of Oharu, The (Saikaku ichidai onna)	1.23754	2
Human Condition II, The (Ningen no joken II)	1.23754	4
Human Condition III, The (Ningen no joken III)	1.23754	4

The worst:

title	b_i	n
Besotted	-3.01246	1
Hi-Line, The	-3.01246	1
Accused (Anklaget)	-3.01246	1
Confessions of a Superhero	-3.01246	1
War of the Worlds 2: The Next Wave	-3.01246	2
SuperBabies: Baby Geniuses 2	-2.76777	47
Disaster Movie	-2.74579	30
From Justin to Kelly	-2.63814	183
Hip Hop Witch, Da	-2.60337	11
Criminals	-2.51246	1

The supposed "best" and "worst" movies were rated by very few users, in most cases just 1. These movies were mostly obscure ones. This is because with just a few users, we have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely. These are noisy estimates that we should not trust, especially when it

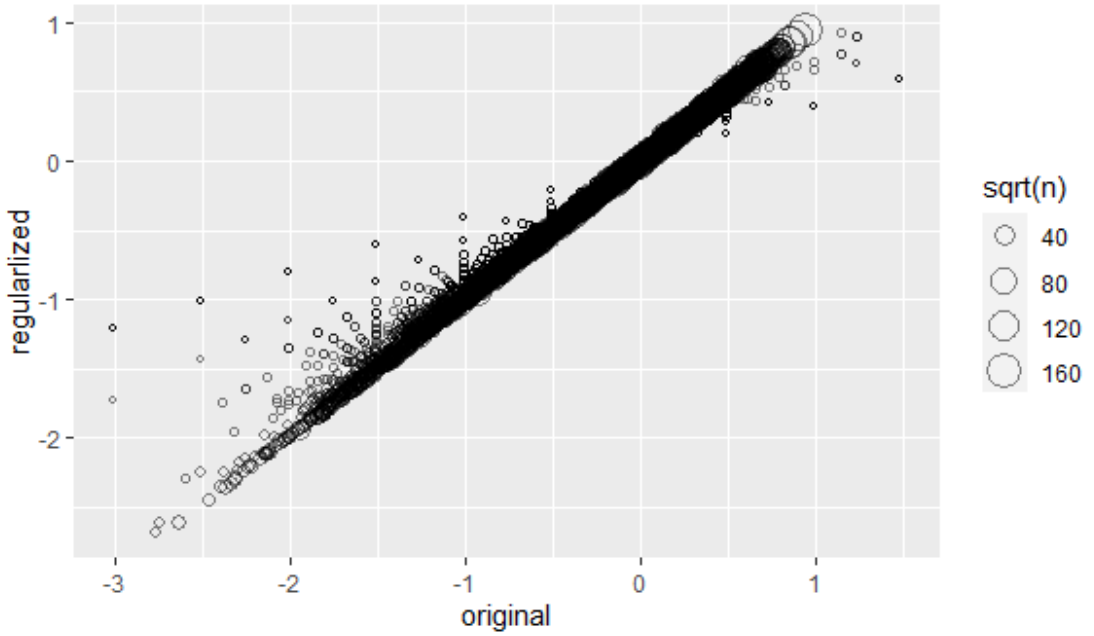
comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when unsure.

Regularization permits us to penalize large estimates that are formed using small sample sizes. When our sample size n_i is very large, a case which will give us a stable estimate, then the penalty λ is effectively ignored since n_i+λ ≈ n_i. However, when the n_i is small, then the estimate b_i(λ) is shrunken towards 0. The larger λ, the more we shrink.

Let's compute these regularized estimates of b_i using lambda=1.5. Then, look at the top 10 best and worst movies.

```
lambda <- 1.5
movie_reg_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
```

To see how the estimates shrink, we can make a plot of the regularized estimates versus the least squares estimates.



Now, let's look at the top 10 best and worst movies based on the penalized estimates ^b_i (λ):

The best:

title	b_i	n
Shawshank Redemption, The	0.944055	25188
More	0.923369	6
Godfather, The	0.904110	15975

Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	0.900032	4
Human Condition II, The (Ningen no joken II)	0.900032	4
Human Condition III, The (Ningen no joken III)	0.900032	4
Usual Suspects, The	0.854030	19457
Schindler's List	0.851568	20877
Rear Window	0.812320	7115
Casablanca	0.807068	10141

The worst:

title	b_i	n
SuperBabies: Baby Geniuses 2	-2.68217	47
From Justin to Kelly	-2.61669	183
Disaster Movie	-2.61504	30
PokÃ©mon Heroes	-2.44259	124
Barney's Great Adventure	-2.35369	186
Carnosaur 3: Primal Species	-2.34016	61
Glitter	-2.32760	311
Gigli	-2.30266	281
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie)	-2.29204	188
Hip Hop Witch, Da	-2.29096	11

Now we can use the regularized b_i to make predictions and see if it impacts the RMSE.

```
## [1] 0.942937
```

Did we improve our results? Regularization movie effects brought very small improvements to the results RMSE. But what if we penalize as well small sizes of users, genres and year? And what value we should assing to λ to minimize the estimated RMSE? Let's do a cross validation to pick a λ:

```
lambdas <- seq(0, 10, 0.5)

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

  mu <- mean(edx_train$rating)

  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx_train %>%
```

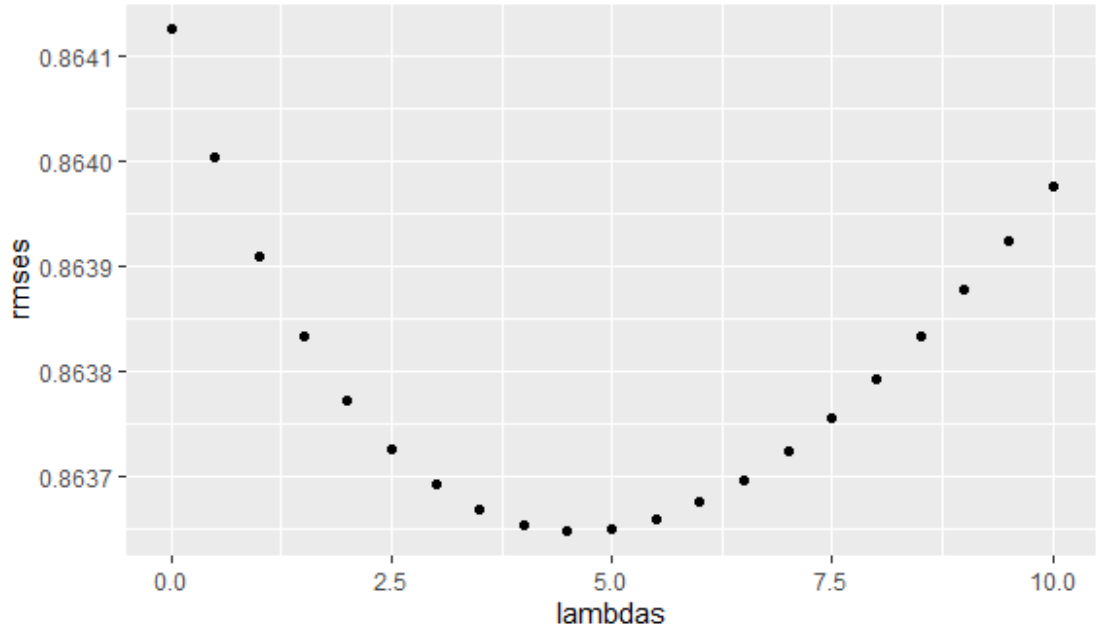
```
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  b_g<- edx_train %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_u - b_i - mu)/(n()+l))

  b_y <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    left_join(b_g, by="genres") %>%
    group_by(release_year) %>%
    summarize(b_y = sum(rating - b_g - b_u - b_i - mu)/(n()+l))

    reg_movie_user_hat <-
    edx_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by= "genres") %>%
    left_join(b_y, by= "release_year") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
    pull(pred)

  return(RMSE(edx_test$rating,reg_movie_user_hat))
})
qplot(lambdas, rmses)
```



We see by the plot, the best value for lambda, which minimize RMSe is 4.5.

```
## [1] 4.5
```

Now let's make predictions using λ = 4,5 and see if it impacts our RMSE.

```
## [1] 0.863648
```

We can see a resume of our results on the following dataframe.

```
##                                                 method     RMSE
## 1                                   Just the average 1.060054
## 2                                       Movie Effect 0.942961
## 3                                        User Effect 0.864684
## 4                                      Genres effect 0.864324
## 5                                        Year effect 0.864126
## 6                     Regularized Movie Effect Model 0.942937
## 7 Regularized Movie + User + Genres + Year effect 0.863648
```

2.4 Predict rating to Validation dataset

Now that we have our final model it is time to use it to predict ratings on the validation dataset. First we add "release_year" and "date" to obtain a dataset in the same structure than edx training set, and after we use the bias obtained for "movie", "users", "genre" and "release year" in the training set.

```
l<-4.5
mu <- mean(edx_train$rating)

b_i <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))

b_u <- edx_train %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))

b_g<- edx_train %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - b_u - b_i - mu)/(n()+l))
b_y <- edx_train %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  left_join(b_g, by="genres") %>%
  group_by(release_year) %>%
  summarize(b_y = sum(rating - b_g - b_u - b_i - mu)/(n()+l))
```

```
validation_hat <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by= "genres") %>%
  left_join(b_y, by= "release_year") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
  pull(pred)
```

Finally calculate the RMSE

```
## [1] 0.864699
```

3. Results

The goal of this project is to build a recommendation system that can predict ratings from the MovieLens 10M data set, with a RMSE under 0.86490 on the validation set.

We have built a model using a mean rating for all movies regardless of users, and later added factors representing movies, users, genres and release year effects. It is important to highlight the final formula for this prediction can be described as: Y = mu + b_i + b_u + b_g + b_y + E. Where E is a term of independent errors that we did not treat in this model, Y is the rating for a single movie rated by one user.

The best RMSE result was obtained on the model where we considered 4 biases and penalized all the small samples with a $\lambda = 4.5$ by a regularization process.

Applying our best model on the validation set, we had a RMSE equal to 0.86469 which means that we reached the goal.

4. Conclusion

This project had a goal to build a recommendation movie system based on the MovieLens with a RMSE less than 0.8649, as part of the final evaluation to the Data Science Professional Certificate from Harvardx-Edx, PH125.9x:Data Science: Capstone.

Our best model resulted in a RMSE = 0.8646 on the validation set. So the goal was reached.

We have a limitation on use machine learning basic algorithms to solve this problem due to the large size of dataset and particularities about recommendation system kind of prediction. Instead other machine learning problems where each cell is influenced only by some variables, in recommedation systems all dataset can de used to predict each cell.

To get more relevant improvements on the model we should use Matrix factorisation through Singular Value Decomposition (SVD) on the residuals errors, where we could decomposing them. But due to the limited time and computational resources to explore it, implementing this approach is beyond the scope of this project.