Movie Recommendation System

Luca Bartolomei

27 aprile, 2021

Contents

1 Introduction

The purpose of the present project is to create a recommendation system for predicting the rating of movies.

A recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.

As a data source for training and final evaluation of the system will be used the MovieLens 10M Dataset from https://grouplens.org/datasets/movielens/10m/.

The residual mean squared error (RMSE) will be used as the evaluation system of the recommendation system. The ultimate goal is to obtain an RMSE value of less than 0.86490.

In a first phase, the data will be imported and the sub-data sets configured for training and final evaluation.

We will start to represent the base data structure using table and charts in order to highlight the relationship between the rating and the features; once it is clear how each feature effects the outcome, it is possible to discard the ineffective ones.

A data cleanup will then be performed to focus only on relevant data.

Finally, selected algorithms will be implemented and then respective RMSE values compared

2 Methods

2.1 Model evaluation

As already mentioned in the introduction, the evaluation model used is the residual mean squared error.

The residual Mean Squared Error (RMSE), is the square root of the the average squared error of the predictions and it's the typical metric to evaluate recommendation systems.

The RMSE penalizes large deviations from the mean and is appropriate in cases where small errors are not relevant.

In general terms RMSE can be described by the following formula

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

Where $y_{u,i}$ is the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$.

2.2 Modeling

The models that will be implemented are described below

2.2.1 Random Prediction

Random prediction is performed using the probability distribution observed during the data exploration.

It is only implemented to indicate the worst case scenario. It will be implemented through the Monte Carlo method

2.2.2 Linear Regression

Linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables.

Linear regression is based on this formula

$$Y_i = \beta_0 + \beta_1 x_{i,1} + ... + \beta_n x_{i,n} + \varepsilon_i, i = 1, \ldots, n$$

Where Y_i are the dependent variables, $x_{i,1}...x_{i,n}$ are the independents variables, β_0 is a constant, $\beta_1...\beta_n$ are coefficients and $\varepsilon_{u,i}$ is the error distribution.

So, to solve that equation we need to find the constant β_0 and $\beta_1...\beta_n$ coefficients

2.2.3 Regularization

Regularization is the process of adding information in order to prevent overfitting by penalizing models with extreme parameter values. Overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.

The aim of regularization is to regularize or to shrink the coefficient estimates towards zero.This technique discourages feeding a more complex or flexible model, so as to avoid the risk of overfitting.

If we consider this formula

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i} - \mu - \beta_i - \beta_u\right)^2 + \lambda\left(\sum_i \beta_i^2 + \sum_u \beta_u^2\right)$$

The first term is just the sum of squares and the second is a penalty that gets larger when many β_i and/or β_u are large. So, the idea is to find a value for λ that minimizes the above equation.

An effective method to choose λ that minimizes the RMSE is running simulations with several values of λ.

2.2.4 Matrix factorization

Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two rectangular matrices of lower dimension. In simpler terms, Factorization is the method of expressing something big as a product of smaller factors.

2.3 Data preparation

Let's start by installing the necessary packages.

```
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")

#Install package ggthemes to manage themes, geoms, and scales for 'ggplot2'
if(!require(ggthemes))
  install.packages("ggthemes", repos = "http://cran.us.r-project.org")

#This package simplifies the way to manipulate the HTML or 'LaTeX' codes
#generated by 'kable()' and allows users to construct complex tables
#and customize styles using a readable syntax
if(!require(kableExtra))
  install.packages("kableExtra", repos = "http://cran.us.r-project.org")

#This package make Dealing with Dates a Little Easier
if(!require(lubridate))
  install.packages("lubridate", repos = "http://cran.us.r-project.org")


library(tidyverse)
library(caret)
library(data.table)
library(ggthemes)
library(scales)
library(kableExtra)
library(lubridate)
```

We download the data necessary for the creation of the recommendation system, We split the dataset in two parts: the training set "edx" and the evaluation set "validation", with 90% and 10% of the original dataset respectively.

Then, we split "edx" in two parts: the train set "train_set" and test set "test_set", with 90% and 10% of edx set respectively. We are going to use "train_set" to train the models which will be tested with "test_set".

The best model will be trained with "edx" and validated with "validation".

```
dl <- tempfile()

download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t",
                             readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                             col.names = c("userId",
                                           "movieId",
                                           "rating",
                                           "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")),
                          "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
```

```r
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")


# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating,
                                  times = 1,
                                  p = 0.1,
                                  list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]


# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)

#The edx set is used for training and testing, and the validation set
#is used for final validation to simulate the new data.

#Here, we split the edx set in 2 parts: the training set and the test set.

#The model building is done in the training set, and the test set is
#used to test the model. When the model is complete, we use the validation
#set to calculate the final RMSE. We use the same procedure used
#to create edx and validation sets.

#The training set will be 90% of edx data and the test set
#will be the remaining 10%.
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating,
                                  times = 1,
                                  p = 0.1,
                                  list = FALSE)
train_set <- edx[-test_index,]
temp <- edx[test_index,]

# Make sure userId and movieId in test set are also in train set
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test set back into train set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
```

```
rm(test_index, temp, removed)
```

2.4 Data exploration

Let's analyze the structure of the data.

```
#Structure
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 83898488
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Actio
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
#Dimension
dim(edx)
```

```
## [1] 9000055       6
```

edx has six variables:

userId	integer
movieId	numeric
rating	numeric
timestamp	integer
title	character
genres	character

The variable "rating" is the desired outcome. The other variables are the potential predictors.

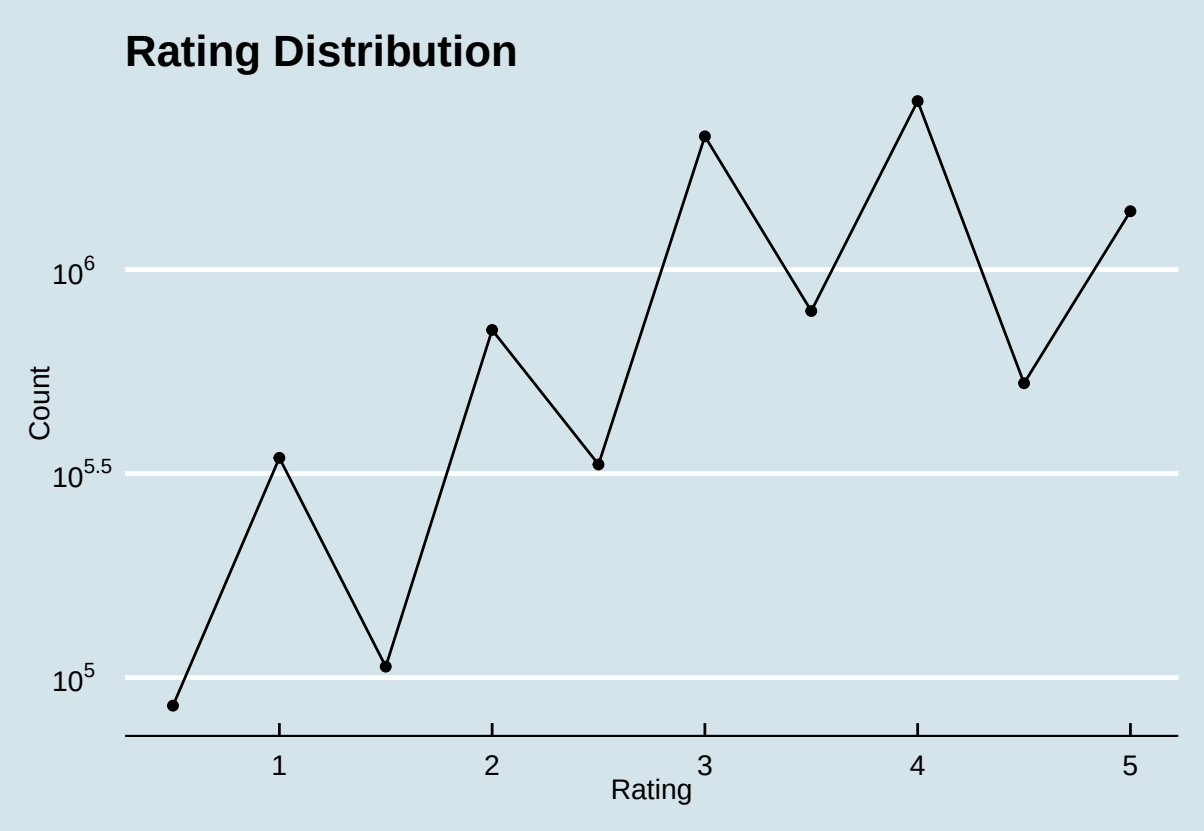
2.5 Visualization

Now we analyze the distribution of the rating and the distribution of the rating with respect to users, movies, genres and dates to get a first idea of the effect of these predictors on the rating

2.5.1 Rating analysis

Round values receive more ratings than decimals and higher ratings are prevalent. High ratings are predominant

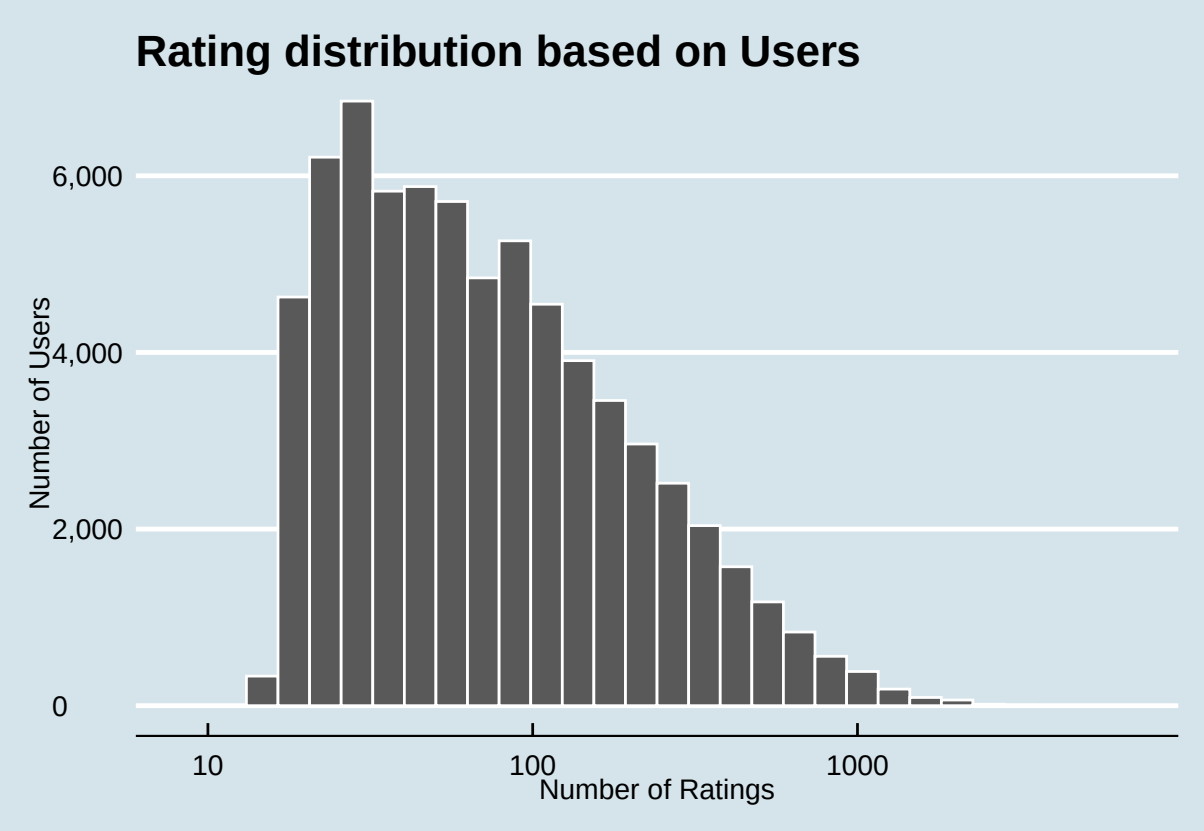
```
edx %>% group_by(rating) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=rating, y=count)) +
  geom_line() +
  geom_point() +
  scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x))) +
  ggtitle("Rating Distribution") +
  xlab("Rating") +
  ylab("Count") +
  theme_economist()
```



2.5.2 Users analysis

Rating distribution based on Users is right skewed.

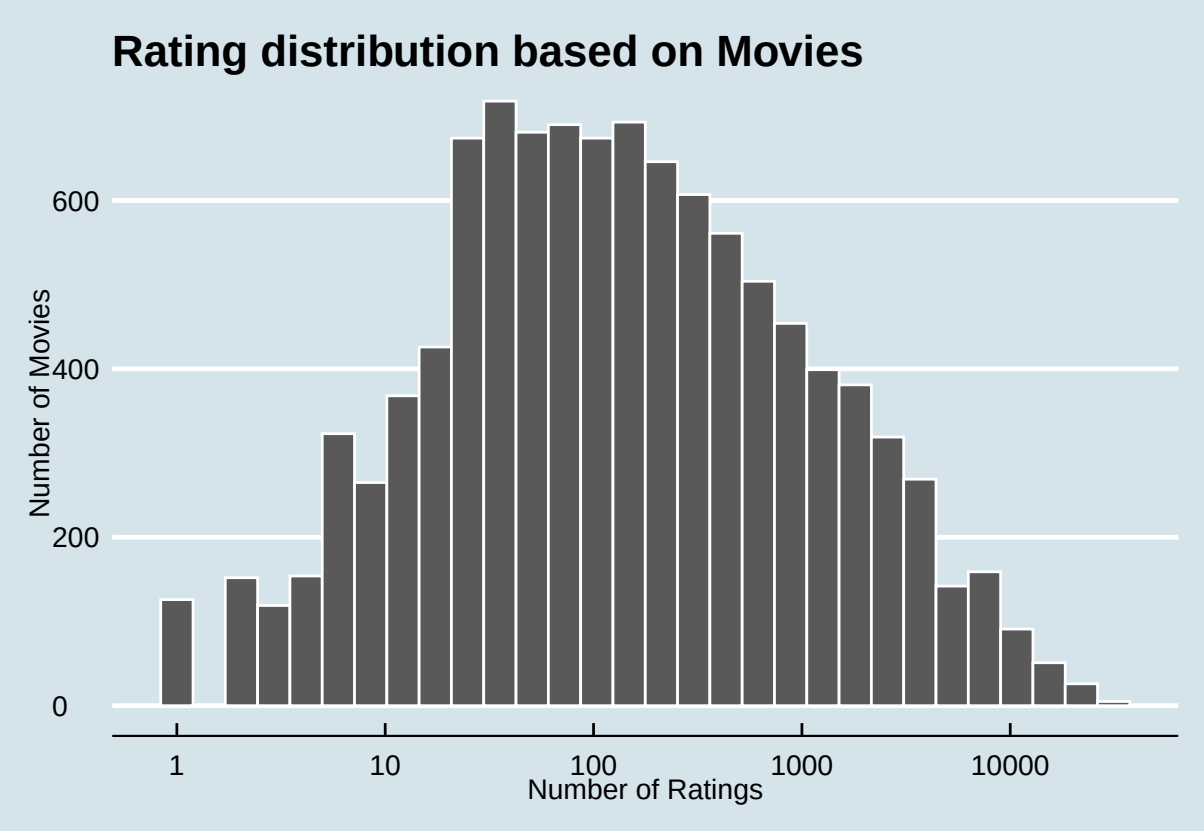
```
edx %>% group_by(userId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "white") +
  scale_x_log10() +
  ggtitle("Rating distribution based on Users") +
  xlab("Number of Ratings") +
  ylab("Number of Users") +
  scale_y_continuous(labels = comma) +
  theme_economist()
```



2.5.3 Movies analysis

Rating distribution based on Movies is almost symmetric

```
edx %>% group_by(movieId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "white") +
  scale_x_log10() +
  ggtitle("Rating distribution based on Movies") +
  xlab("Number of Ratings") +
  ylab("Number of Movies") +
  theme_economist()
```



2.5.4 Genres analysis

Some movies fall under several genres

```
head(edx, 20) %>% kable(caption = "Movies") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 2: Movies

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
1	356	5	838983653	Forrest Gump (1994)	Comedy|Drama|Romance|War
1	362	5	838984885	Jungle Book, The (1994)	Adventure|Children|Romance
1	364	5	838983707	Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical
1	370	5	838984596	Naked Gun 33 1/3: The Final Insult (1994)	Action|Comedy
1	377	5	838983834	Speed (1994)	Action|Romance|Thriller
1	420	5	838983834	Beverly Hills Cop III (1994)	Action|Comedy|Crime|Thriller
1	466	5	838984679	Hot Shots! Part Deux (1993)	Action|Comedy|War
1	520	5	838984679	Robin Hood: Men in Tights (1993)	Comedy
1	539	5	838984068	Sleepless in Seattle (1993)	Comedy|Drama|Romance
1	588	5	838983339	Aladdin (1992)	Adventure|Animation|Children|Comedy|Musical
1	589	5	838983778	Terminator 2: Judgment Day (1991)	Action|Sci-Fi
1	594	5	838984679	Snow White and the Seven Dwarfs (1937)	Animation|Children|Drama|Fantasy|Musical
1	616	5	838984941	Aristocats, The (1970)	Animation|Children
2	110	5	868245777	Braveheart (1995)	Action|Drama|War

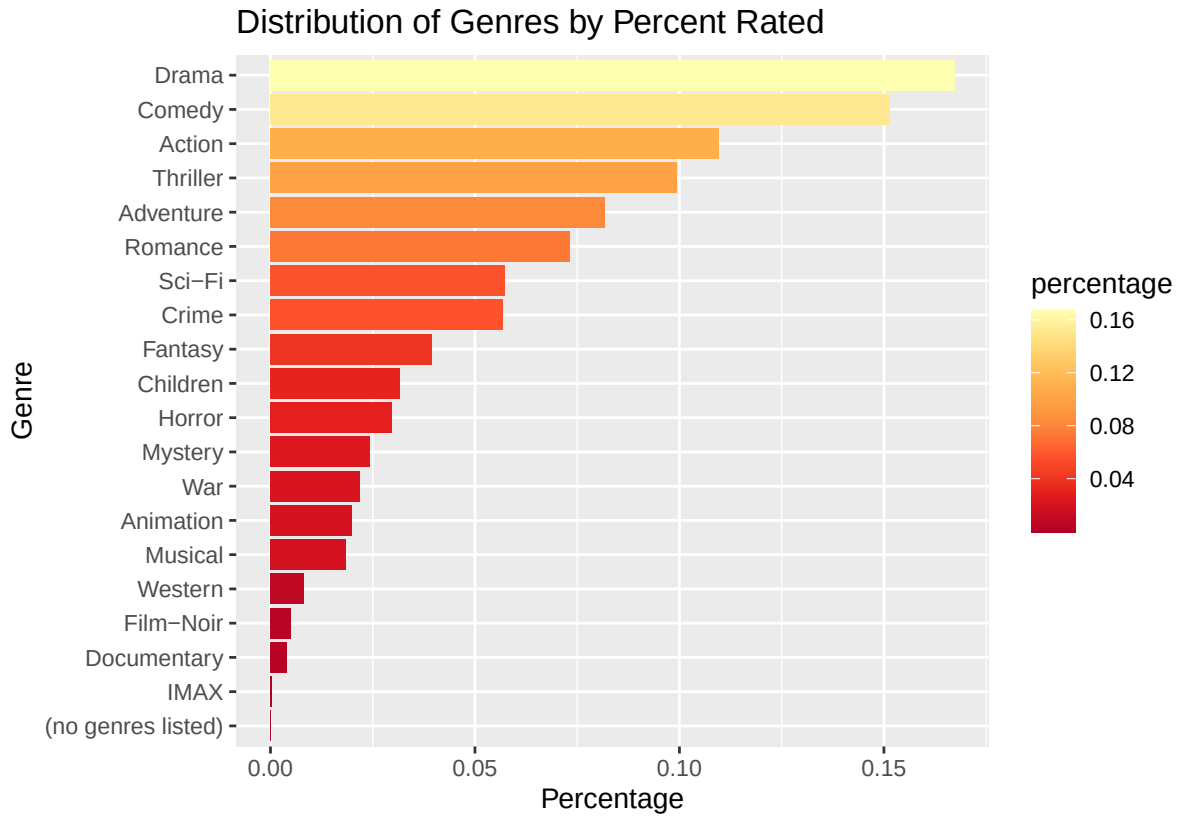
```
#We can see that different movies belong to multiple genres

tibble(count = str_count(edx$genres, fixed("|")), genres = edx$genres) %>%
  group_by(count, genres) %>%
  summarise(n = n()) %>%
  arrange(-count) %>%
  head() %>% kable(caption = "Genres") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 3: Genres

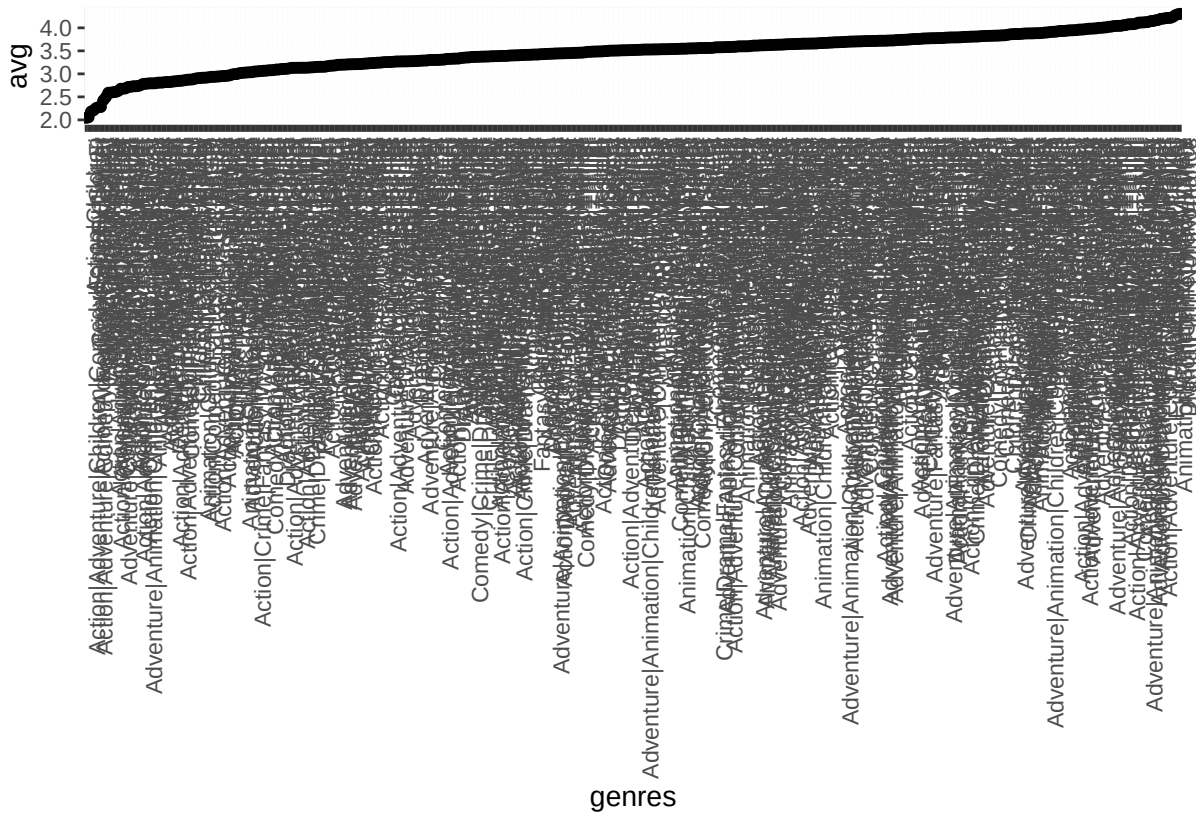
count	genres	n
7	Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller	256
6	Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery	10975
6	Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery	355
6	Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance	515
5	Action|Adventure|Animation|Children|Comedy|Fantasy	187
5	Action|Adventure|Animation|Children|Comedy|IMAX	66

The plots show a clear effect of genres on the rating

```
temp_genre <- edx %>%
  separate_rows(genres,sep = "\\|") %>% mutate(value=1) %>%
  group_by(genres) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  mutate(sumN = sum(n), percentage = n/sumN) %>%
  arrange(-percentage)

temp_genre %>%
  ggplot(aes(reorder(genres, percentage), percentage, fill= percentage)) +
  geom_bar(stat = "identity") + coord_flip() +
  scale_fill_distiller(palette = "YlOrRd") +
  labs(y = "Percentage", x = "Genre") +
  ggtitle("Distribution of Genres by Percent Rated")
```



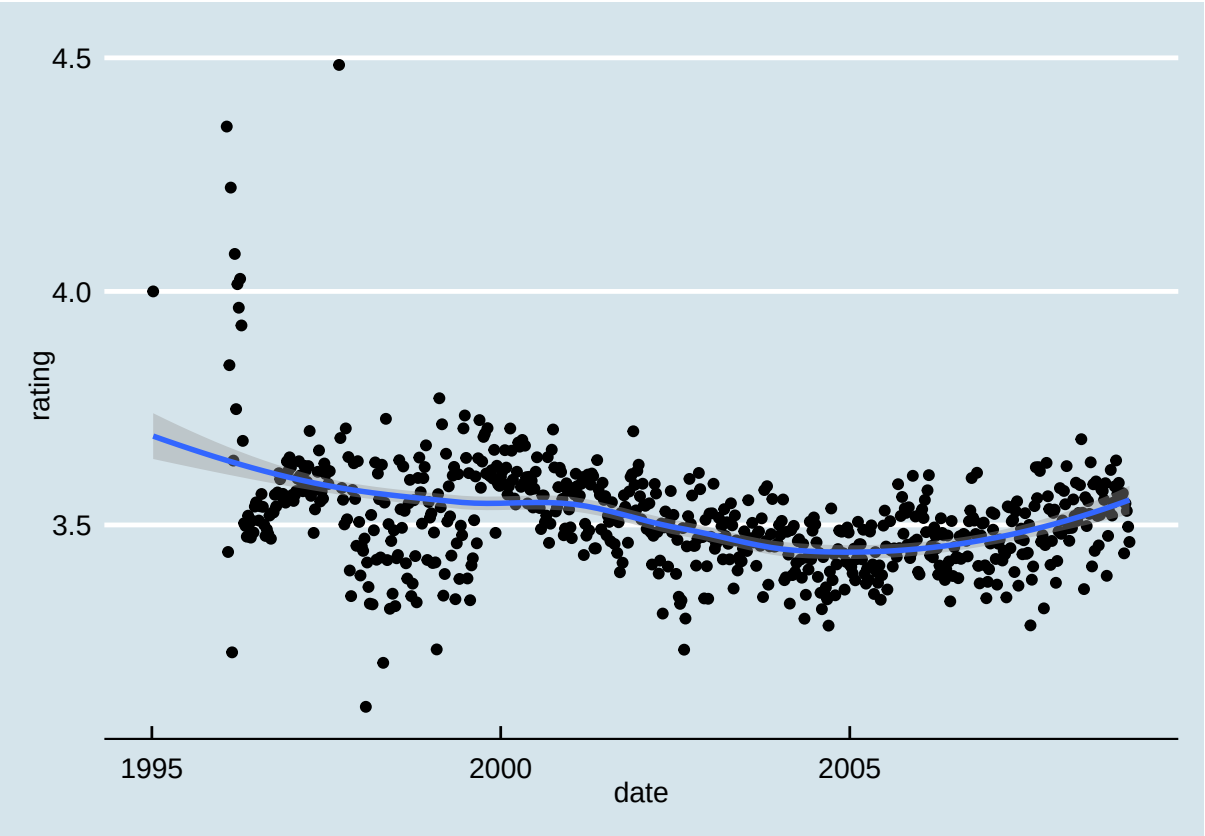

```
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 1000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



2.5.5 Date analysis

The plot shows that time has no strong effect on average ratings

```
edx <- mutate(edx, date = as_datetime(timestamp))
validation <- mutate(validation, date = as_datetime(timestamp))
test_set <- mutate(test_set, date = as_datetime(timestamp))
train_set <- mutate(train_set, date = as_datetime(timestamp))


edx %>% mutate(date = round_date(date, unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  theme_economist()
```


4.5
4.0
3.5
rating
1995
2000
2005
date

2.6 Data cleaning

It is necessary to reduce the number of predictors as otherwise the complexity of the analysis would be too high and, above all, it would require an amount of memory generally not available in a laptop. For example, genres are computationally very expensive as they would require to separate multiple genres for many films in multiple variables. For this analysis we will only share movie and user

```
train_set <- train_set %>% select(userId, movieId, date, rating, title)
test_set  <- test_set  %>% select(userId, movieId, date, rating, title)
```

3 Results

3.1 Model evaluation

For the calculation of the RMSE we will use the following function

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

3.2 Random Prediction

We randomly predict ratings using the probabilities observed in the training set. We start by calculating the probability of each assessment in the training set, then predict the assessment for the test set and compare it to the actual assessment.

```
set.seed(1, sample.kind = "Rounding")

#Create the probability of each rating
random_prob <- function(x, y) mean(x == y)
rating <- seq(0.5,5,0.5)

# Estimate the probability of each rating with Monte Carlo simulation
B <- 10^3
monte_carlo <- replicate(B, {
  rec <- sample(train_set$rating, 100, replace = TRUE)
  sapply(rating, random_prob, x = rec)
})

monte_carlo_prob <- sapply(1:nrow(monte_carlo),
                           function(ind)
                             mean(monte_carlo[ind,]))

#Predict random ratings
random_prediction <- sample(rating, size = nrow(test_set),
                                replace = TRUE,
                                prob = monte_carlo_prob)

#Create a table with the error results
monte_carlo_result <- tibble(Method = "Random prediction",
                             RMSE = RMSE(test_set$rating,
                                         random_prediction))

result <- tibble(Method = "Project target", RMSE = 0.86490)

result <- bind_rows(result,
                    monte_carlo_result)

result %>% kable(caption = "Result") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 4: Result

Method	RMSE
Project target	0.864900
Random prediction	1.497303

3.3 Linear Regression

Linear regression euqation applied to our case is this one

$$\hat{Y} = \mu + \beta_i + \beta_u + \varepsilon_{u,i}$$

Where μ is the mean of rating, β_i is the movie bias, β_u is the user bias and $\varepsilon_{u,i}$ is the error distribution

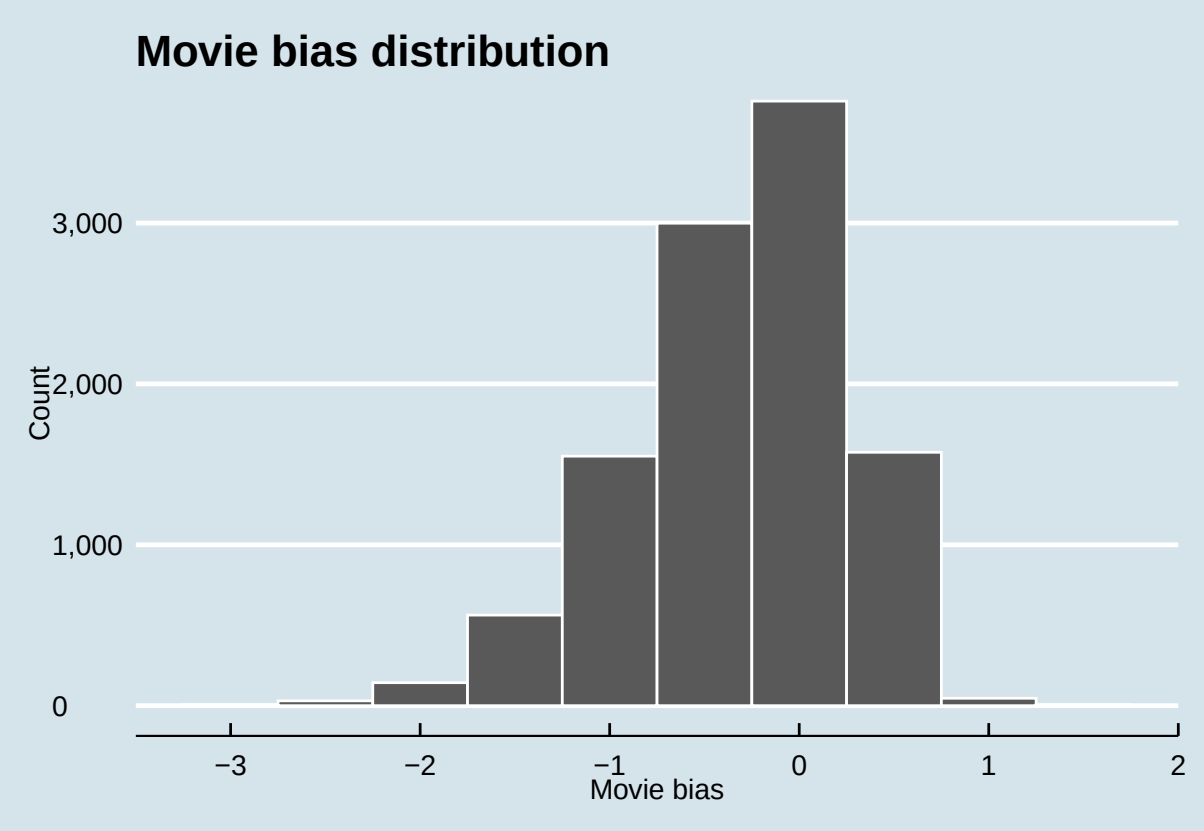
We will not use the lm () function as it is not recommended for large databases

```
set.seed(1, sample.kind="Rounding")
#Evaluate mean of the ratings
mean_rating <- mean(train_set$rating)
```

```
#Evaluate movie bias
movie_bias <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mean_rating))
```

Movie bias has an asymmetric distribution

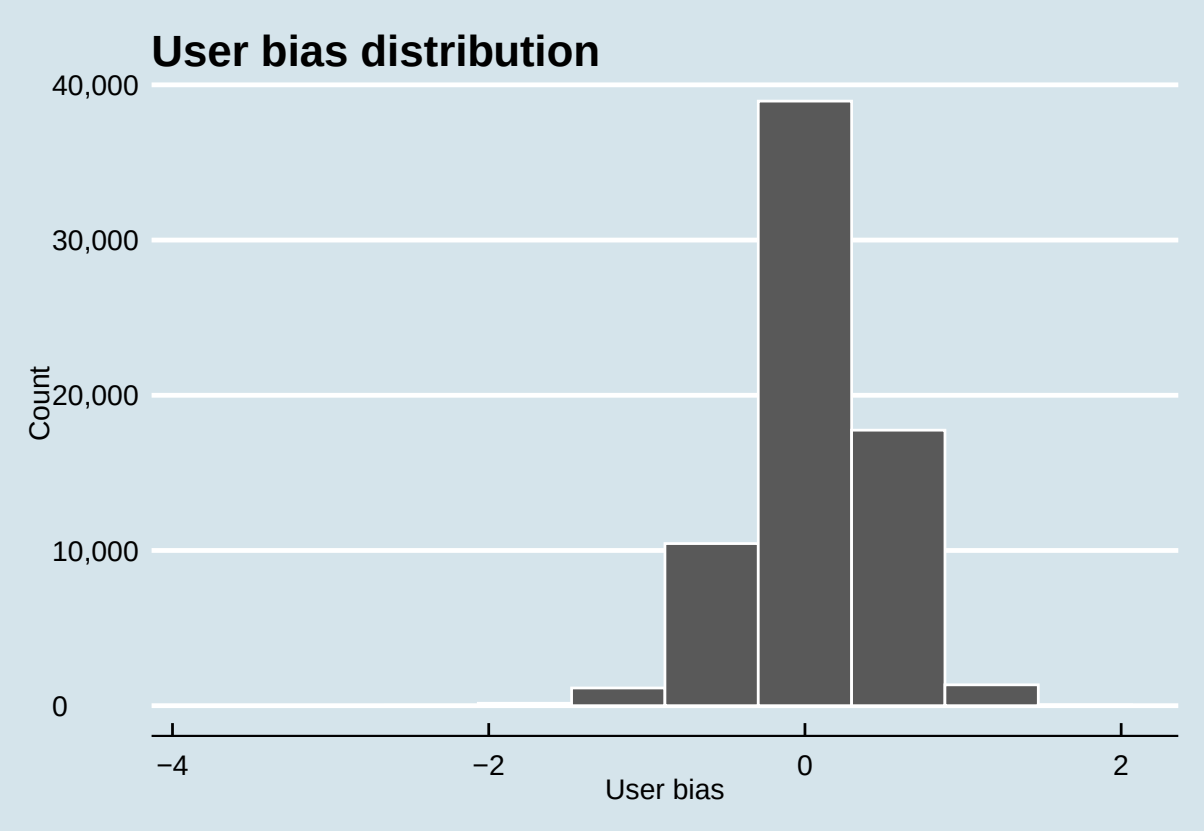
```
movie_bias %>% ggplot(aes(x = b_i)) +
  geom_histogram(bins=10, col = I("white")) +
  ggtitle("Movie bias distribution") +
  xlab("Movie bias") +
  ylab("Count") +
  scale_y_continuous(labels = comma) +
  theme_economist()
```




```
#Evaluate user bias
user_bias <- train_set %>%
  left_join(movie_bias, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mean_rating - b_i))
```

User bias is almost symmetrical

```
user_bias %>% ggplot(aes(x = b_u)) +
  geom_histogram(bins=10, col = I("white")) +
  ggtitle("User bias distribution") +
  xlab("User bias") +
  ylab("Count") +
  scale_y_continuous(labels = comma) +
  theme_economist()
```




```
#Predict rating
regression_prediction <- test_set %>%
  left_join(movie_bias, by='movieId') %>%
  left_join(user_bias, by='userId') %>%
  mutate(prediction = mean_rating + b_i + b_u) %>%
  .$prediction

regression_rmse <- RMSE(test_set$rating,
                            regression_prediction)

regression_rmse
```

```
## [1] 0.8646843
```

```
regression_prediction_result <- tibble(Method = "Linear regression prediction",
                                       RMSE = RMSE(test_set$rating,
                                                   regression_prediction))

result <- bind_rows(result,
                    regression_prediction_result)

result %>% kable(caption = "Results") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 5: Results

Method	RMSE
Project target	0.8649000
Random prediction	1.4973032
Linear regression prediction	0.8646843

Linear regression method improves the performance in the calculation of the RMSE compared to Random prediction

3.4 Regularization

We want to regularize movie and user bias adding a penalty factor λ and find a value to pick the best value that minimizes the RMSE. We use cross-validation through the use of the function sapply()

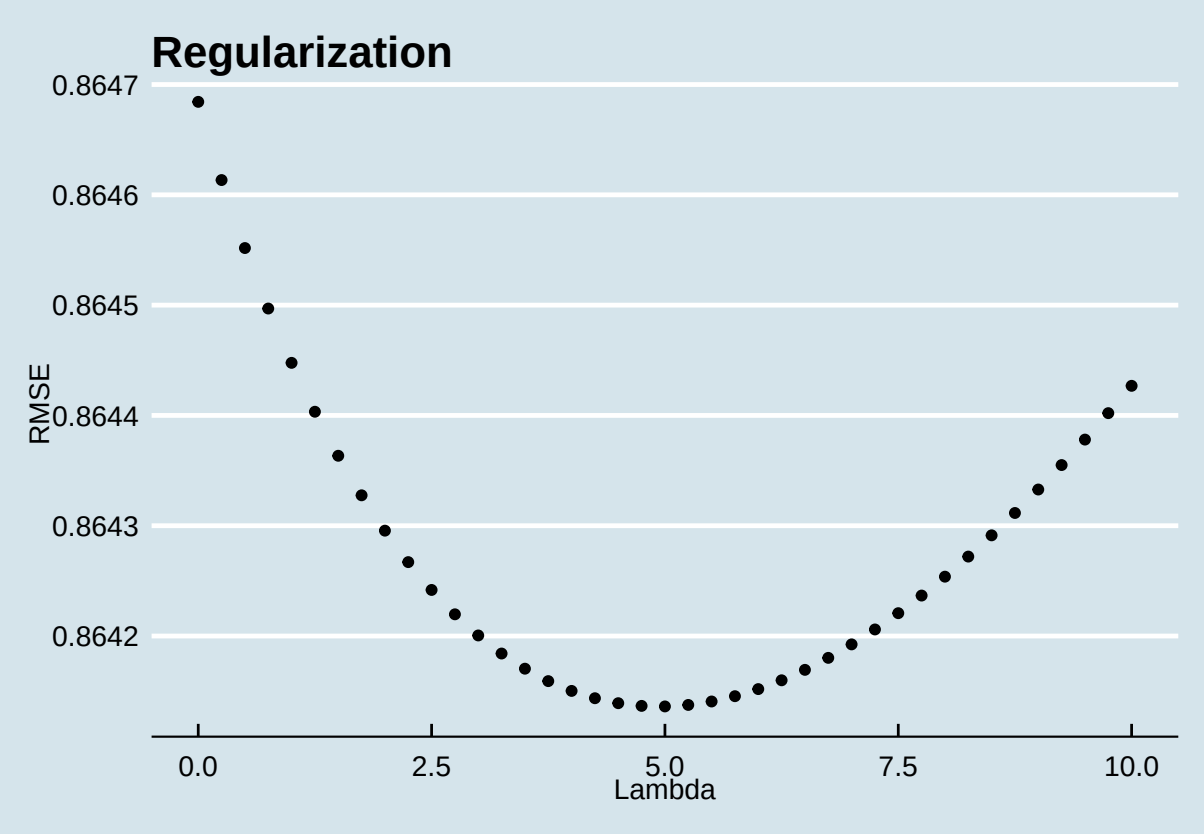
```
set.seed(1, sample.kind="Rounding")
#Regularization function
regularization <- function(lambda, training, test){

  #Mean
  mean_rating <- mean(training$rating)

  #Movie bias
  movie_bias <- training %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mean_rating)/(n()+lambda))

  #User bias
  user_bias <- training %>%
    left_join(movie_bias, by="movieId") %>%
    filter(!is.na(b_i)) %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mean_rating - b_i)/(n()+lambda))

  #Prediction
  predicted_ratings <- test %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias, by = "userId") %>%
    filter(!is.na(b_i), !is.na(b_u)) %>%
    mutate(predicted = mean_rating + b_i + b_u) %>%
    pull(predicted)

  return(RMSE(test$rating, predicted_ratings))
}
```

```
# Definition of lambdas
lambdas <- seq(0, 10, 0.25)
```

```
#Tuning
rmses <- sapply(lambdas,
```

```
                regularization,
                training = train_set,
                test = test_set)
```

```
# Plot lambdas
tibble(Lambda = lambdas, RMSE = rmses) %>%
  ggplot(aes(x = Lambda, y = RMSE)) +
    geom_point() +
    ggtitle("Regularization") +
    theme_economist()
```




```
# Pick the best value
lambda <- lambdas[which.min(rmses)]
lambda
```

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

The best value is 0.5

```
# Regularization prediction

#Mean
mean_rating <- mean(train_set$rating)

#Movie bias
movie_bias <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mean_rating)/(n()+lambda))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
#User bias
user_bias <- train_set %>%
  left_join(movie_bias, by="movieId") %>%
  filter(!is.na(b_i)) %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mean_rating - b_i)/(n()+lambda))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
#Prediction
regularization_prediction <- test_set %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(user_bias, by = "userId") %>%
  filter(!is.na(b_i), !is.na(b_u)) %>%
  mutate(predicted = mean_rating + b_i + b_u) %>%
  pull(predicted)

regularization_rmse <- RMSE(test_set$rating, regularization_prediction)

regularization_result <- tibble(Method = "Regularization prediction",
                                RMSE = regularization_rmse)
result <- bind_rows(result,
                    regularization_result)

result %>% kable(caption = "Results") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 6: Results

Method	RMSE
Project target	0.8649000
Random prediction	1.4973032
Linear regression prediction	0.8646843
Regularization prediction	0.8641362

Regularization provides a slight improvement in the RMSE estimate

3.5 Matrix factorization

To perform matrix factorization first of all we need to convert the data in a user-movie matrix and then approximate this matrix as a product of two smaller matrices. These operations could be very expensive in terms of memory, so we are going to use the recosystem package, which provides the complete solution for a recommendation system using matrix factorization.

Basically recosystem performs the following operations:

- Create a model object by calling Reco();

- Call the tune() method to select best tuning parameters along a set of candidate values;
- Call train() method to train the model;
- Call predict() method to compute predicted values.

This package has several parameters whose values can be calibrated to increase performance, in particular, the nthread parameter which sets the number of threads for parallel computing.

```
#Install recosystem package
if(!require(recosystem))
  install.packages("recosystem", repos = "http://cran.us.r-project.org")
```

```
## package 'RcppProgress' successfully unpacked and MD5 sums checked
## package 'recosystem' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\DELL\AppData\Local\Temp\RtmpWMUi3V\downloaded_packages
```

```
library(recosystem)
```

```
set.seed(1, sample.kind = "Rounding")

#Create matrices
train_data <-  with(train_set, data_memory(user_index = userId,
                                           item_index = movieId,
                                           rating     = rating))

test_data  <-  with(test_set,  data_memory(user_index = userId,
                                           item_index = movieId,
                                           rating     = rating))
```

```
#Create the model object
model_object <-  Reco()
```

```
#Select the best tuning parameters
best_tuning <-  model_object$tune(train_data,
                           opts = list(dim = c(10, 20, 30),
                           lrate = c(0.1, 0.2),
                           costp_l1 = c(0.01, 0.1),
                           costq_l1 = c(0.01, 0.1),
                           nthread  = 4,
                           niter = 10))
```

```
#Train model
model_object$train(train_data,
                   opts = c(best_tuning$min, nthread = 4, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9624   1.1298e+07
##    1       0.8738   1.0025e+07
##    2       0.8459   9.6766e+06
##    3       0.8261   9.4486e+06
##    4       0.8098   9.2725e+06
##    5       0.7962   9.1320e+06
##    6       0.7847   9.0204e+06
##    7       0.7748   8.9256e+06
```

```
##    8       0.7661   8.8463e+06
##    9       0.7585   8.7764e+06
##   10       0.7519   8.7209e+06
##   11       0.7458   8.6671e+06
##   12       0.7403   8.6205e+06
##   13       0.7355   8.5804e+06
##   14       0.7313   8.5432e+06
##   15       0.7273   8.5099e+06
##   16       0.7239   8.4805e+06
##   17       0.7208   8.4536e+06
##   18       0.7182   8.4291e+06
##   19       0.7156   8.4063e+06
```

```
#Predict
matrix_prediction <-  model_object$predict(test_data, out_memory())

matrix_rmse <-  RMSE(test_set$rating,
                     matrix_prediction)

matrix_result <- tibble(Method = "Matrix factorization prediction",
                        RMSE = matrix_rmse)

result <- bind_rows(result,
                    matrix_result)

result %>% kable(caption = "Results") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 7: Results

Method	RMSE
Project target	0.8649000
Random prediction	1.4973032
Linear regression prediction	0.8646843
Regularization prediction	0.8641362
Matrix factorization prediction	0.7931145

Matrix factorization further improves the performance in calculating the RMSE

3.6 Final validation

From the previous results we can see that Matrix factorization gives the best results on RMSE evaluation.

In these final step we train Matrix factorization model with complete edx dataset and then we will evaluate the RMSE value on validation dataset

3.6.1 Matrix factorization

```
set.seed(1, sample.kind = "Rounding")

#Create matrix
edx_data <-  with(edx, data_memory(user_index = userId,
                                   item_index = movieId,
                                   rating     = rating))

validation_data  <-  with(validation,  data_memory(user_index = userId,
                                                   item_index = movieId,
                                                   rating     = rating))
```

```
#Create the model object
final_model_object <-  Reco()
```

```
#Select the best tuning parameters
final_best_tuning <-  final_model_object$tune(edx_data,
                                opts = list(dim = c(10, 20, 30),
                                       lrate = c(0.1, 0.2),
                                       costp_l1 = c(0.01, 0.1),
                                       costq_l1 = c(0.01, 0.1),
                                       nthread  = 4,
                                       niter = 10))
```

```
#Train model
final_model_object$train(edx_data,
                  opts = c(final_best_tuning$min,
                           nthread = 4,
                           niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9538   1.2406e+07
##    1       0.8680   1.1056e+07
##    2       0.8410   1.0685e+07
##    3       0.8214   1.0440e+07
##    4       0.8056   1.0252e+07
##    5       0.7930   1.0112e+07
##    6       0.7827   9.9961e+06
##    7       0.7738   9.9029e+06
##    8       0.7659   9.8222e+06
##    9       0.7591   9.7543e+06
##   10       0.7531   9.6976e+06
##   11       0.7476   9.6417e+06
##   12       0.7424   9.5926e+06
##   13       0.7382   9.5520e+06
##   14       0.7343   9.5133e+06
##   15       0.7308   9.4789e+06
##   16       0.7276   9.4442e+06
```

```
##    17       0.7248   9.4160e+06
##    18       0.7224   9.3906e+06
##    19       0.7202   9.3656e+06
```

```
 #Predict
 final_matrix_prediction <-  final_model_object$predict(validation_data,
                                                        out_memory())

 final_matrix_rmse <- RMSE(validation$rating,
                           final_matrix_prediction)

final_matrix_result <- tibble(Method = "Final Matrix factorization prediction",
                              RMSE = final_matrix_rmse)

result <- tibble(Method = "Project target", RMSE = 0.86490)

result <- bind_rows(result,
                    final_matrix_result)

result %>% kable(caption = "Results") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down","HOLD_position"))
```

Table 8: Results

Method	RMSE
Project target	0.8649000
Final Matrix factorization prediction	0.7899689

4 Conclusion

The objective of this project was to design a recommendation system for predicting the rating of movies.

The goal was to have a RMSE equal to or less than 0.86490

We started by analyzing the dataset to understand the structure of the data

We have therefore defined a subset of predictors also considering the limitations related to memory

We identified the methods to be implemented to create the system and then evaluated their quality in minimizing the RMSE

The Matrix factorization method has been identified as providing the best performance in terms of error reduction

A computer not limited by the memory factor could allow to implement a system that also takes into consideration other predictors besides those considered (movies and users), such as genre. It is reasonable to assume that such a system would lead to a reduction in the value of RMSE