MovieLens Recommender System Capstone Project Report

Compiled by Mahesh Halkeri

Contents

1 Executive Summary

The purpose for this project is creating a recommender system using MovieLens dataset.

The version of movielens dataset used for this final assignment contains approximately 10 Milions of movies ratings, divided in 9 Milions for training and one Milion for validation. It is a small subset of a much larger (and famous) dataset with several millions of ratings. Into the training dataset there are approximately **70.000 users** and **11.000 different movies** divided in 20 genres such as Action, Adventure, Horror, Drama, Thriller and more.

After a initial data exploration, the recommender systems builted on this dataset are evaluated and choosen based on the RMSE - Root Mean Squared Error that should be at least lower than **0.87750**.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_{2t}}$$

For accomplishing this goal, the **Regularized Movie+User+Genre Model** is capable to reach a RMSE of **0.8628**, that is really good.

2 Exploratory Data Analysis

2.1 Inital data Exploration

The 10 Millions dataset is divided into two dataset: edx for training purpose and validation for the validation phase.

The edx dataset contains approximately 9 Millions of rows with 70.000 different users and 11.000 movies with rating score between 0.5 and 5. There is no missing values (0 or NA).

edx dataset

Users	Movies
69878	10677

Missing Values per Column

	x
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

The features/variables/columns in both datasets are six:

- **userId** <integer> that contains the unique identification number for each user.
- **movieId** <numeric> that contains the unique identification number for each movie.
- **rating** <numeric> that contains the rating of one movie by one user. Ratings are made on a 5-Star scale with half-star increments.
- **timestamp** <integer> that contains the timestamp for one specific rating provided by one user.
- **title** <character> that contains the title of each movie including the year of the release.
- **genres** <character> that contains a list of pipe-separated of genre of each movie.

First 6 Rows of edx dataset

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	231	5	838983392	Dumb & Dumber (1994)	Comedy
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

2.2 Dataset Pre-Processing and Feature Engineering

After a initial data exploration, we notice that the genres are pipe-separated values. It's necessary to extract them for more consisten, robust and precise estimate. We also observe that the title contains the year where the movie war released and this it could be necessary to predic the movie rating. Finally, we can extract the year and the month for each rating.

The pre-processing phase is composed by this steps:

1. Convert timestamp to a human readable date format;
2. Extract the month and the year from the date;
3. Extract the release year for each movie from the title;
4. Separate each genre from the pipe-separated value. It increases the size of both datasets. After preprocessing the data, edx dataset looks like this:

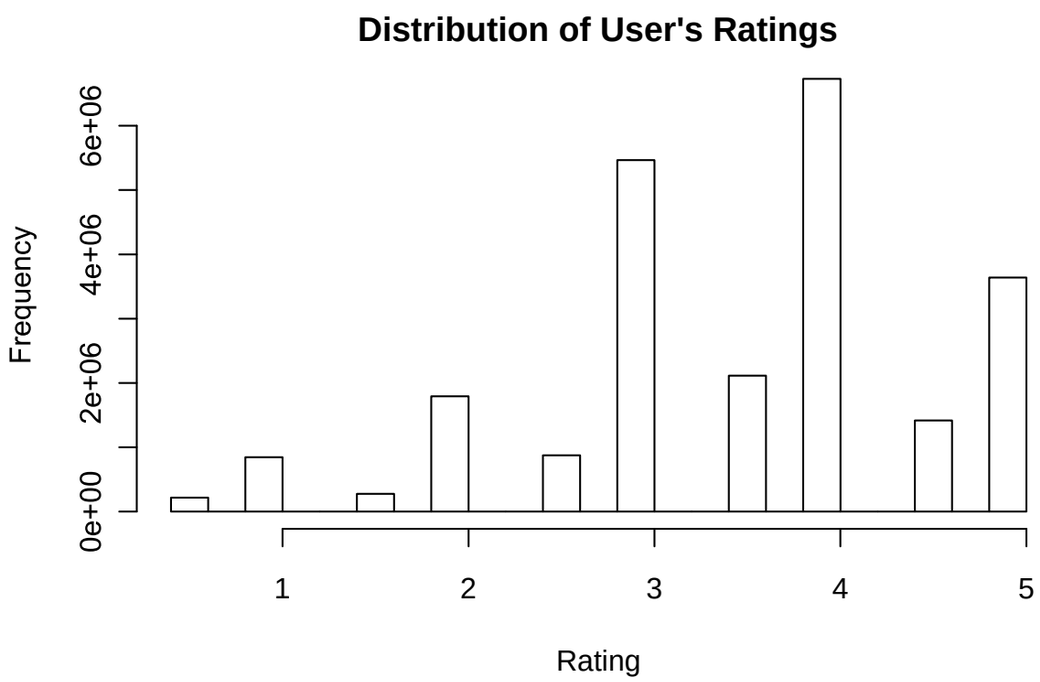
Processed edx datadaset

userId	movieId	rating	title	genre	release	yearOfRate	monthOfRate
1	122	5	Boomerang	Comedy	1992	1996	8
1	122	5	Boomerang	Romance	1992	1996	8
1	185	5	Net, The	Action	1995	1996	8
1	185	5	Net, The	Crime	1995	1996	8
1	185	5	Net, The	Thriller	1995	1996	8
1	231	5	Dumb & Dumber	Comedy	1994	1996	8

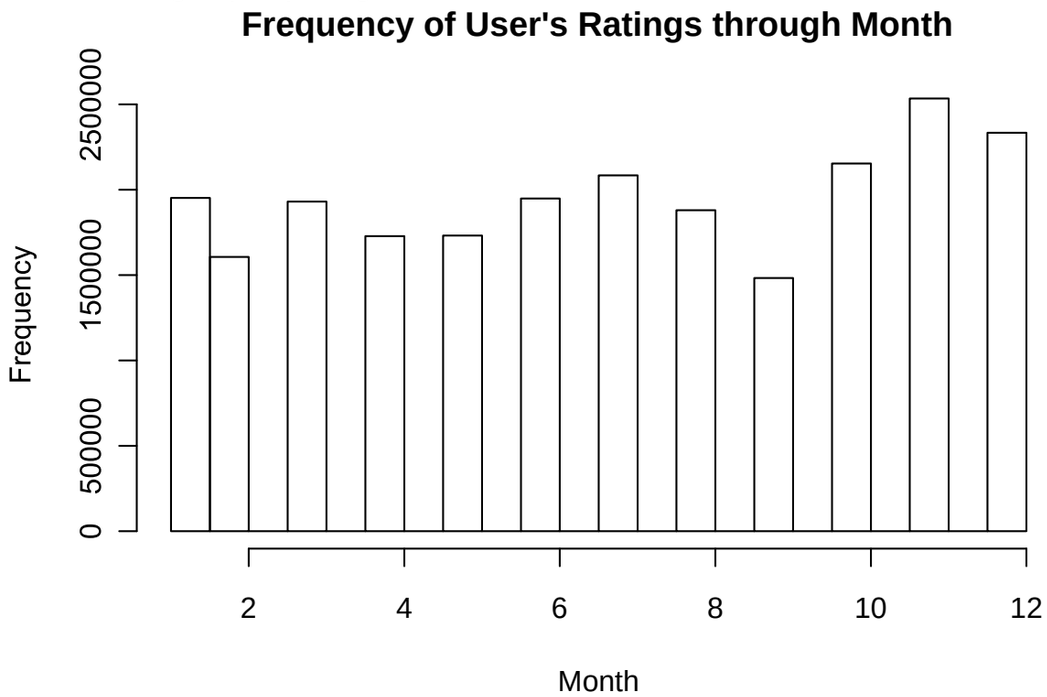
2.3 Rating Distribution

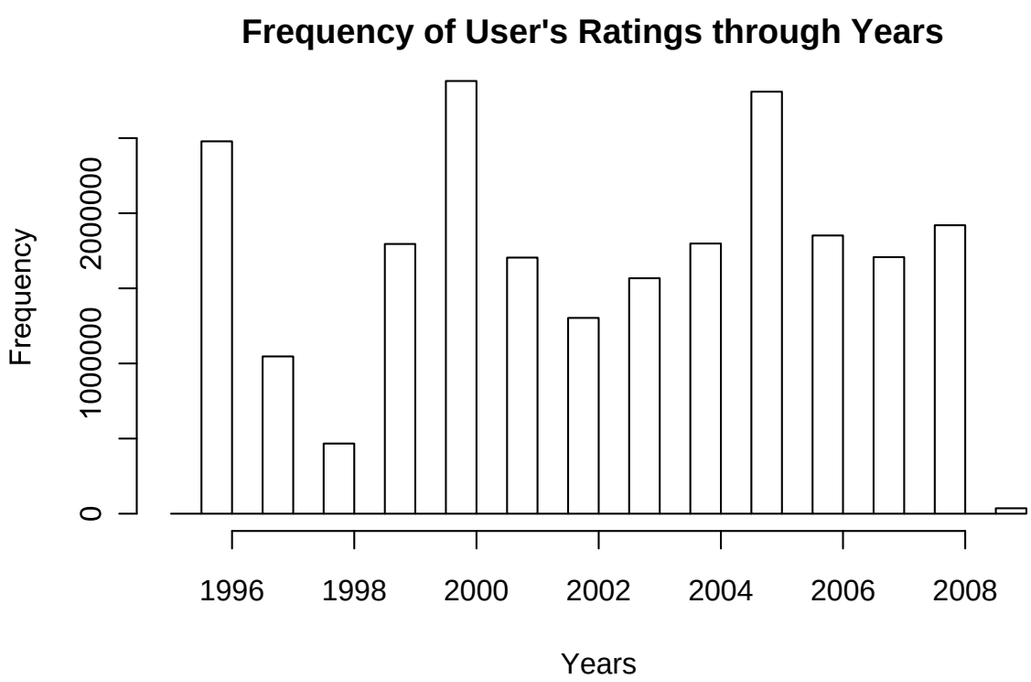
Overview of Rating Distribution

According to the histogram below, it shows that there are a small amount of negative votes (below 3). Maybe, the user tends to give a vote if he liked the movie. Half-Star votes are less common than "Full-Star" votes.

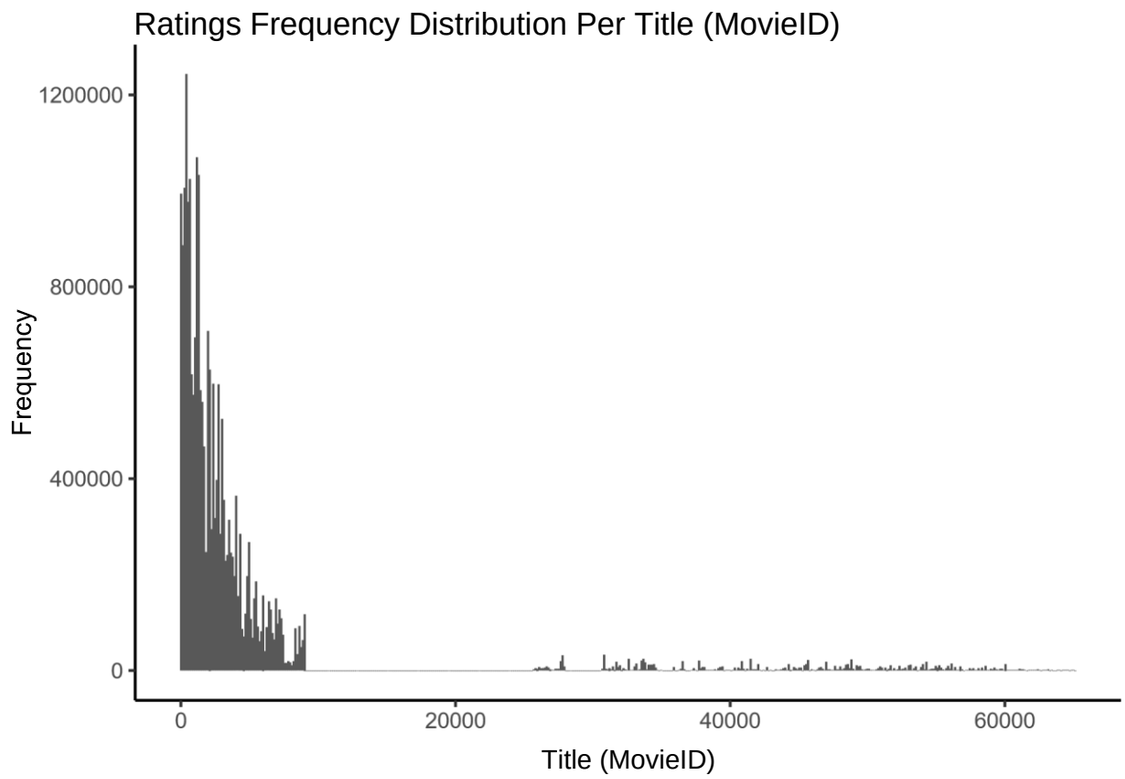


Overview of Rating Frequency through Months and Years

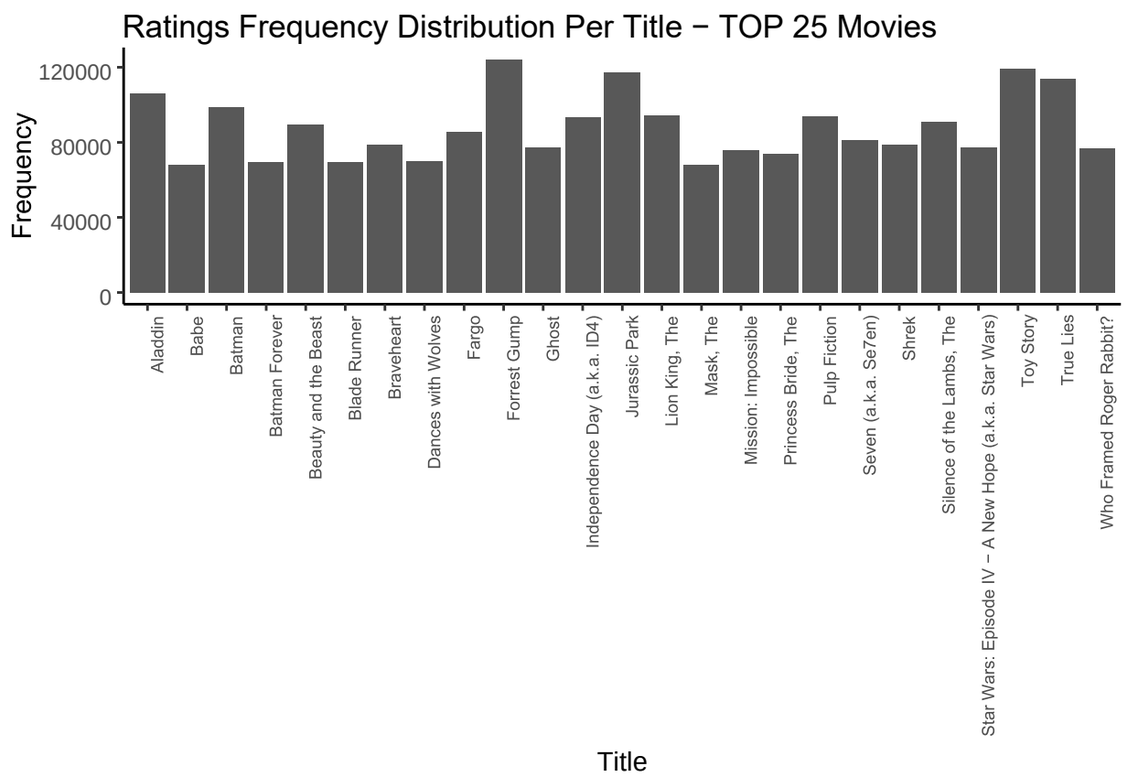



Frequency of User's Ratings through Years
Frequency
0
1000000
2000000
1996
1998
2000
2002
2004
2006
2008
Years

2.3.1 Numbers of Ratings per Movie



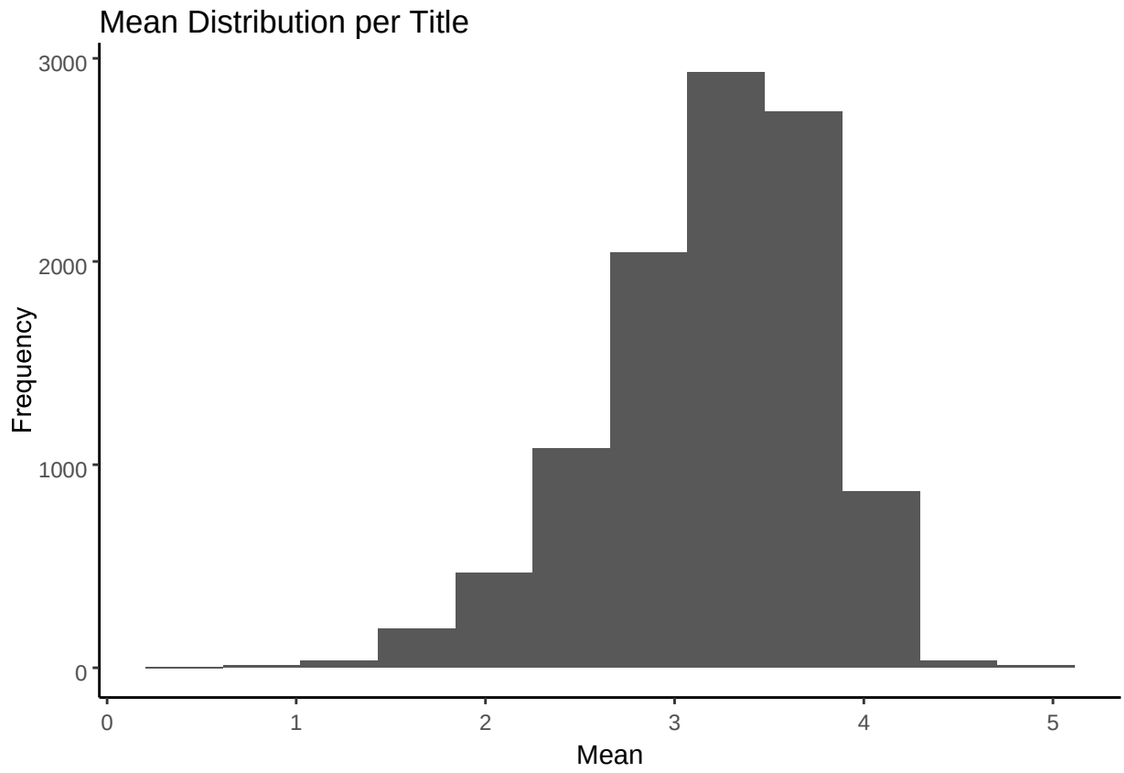
2.3.2 Top Rated Movies



title	count
Forrest Gump	124304
Toy Story	119130
Jurassic Park	117164
True Lies	113930
Aladdin	106070
Batman	98656
Lion King, The	94435
Pulp Fiction	94008
Independence Day (a.k.a. ID4)	93440
Silence of the Lambs, The	90840
Beauty and the Beast	89315

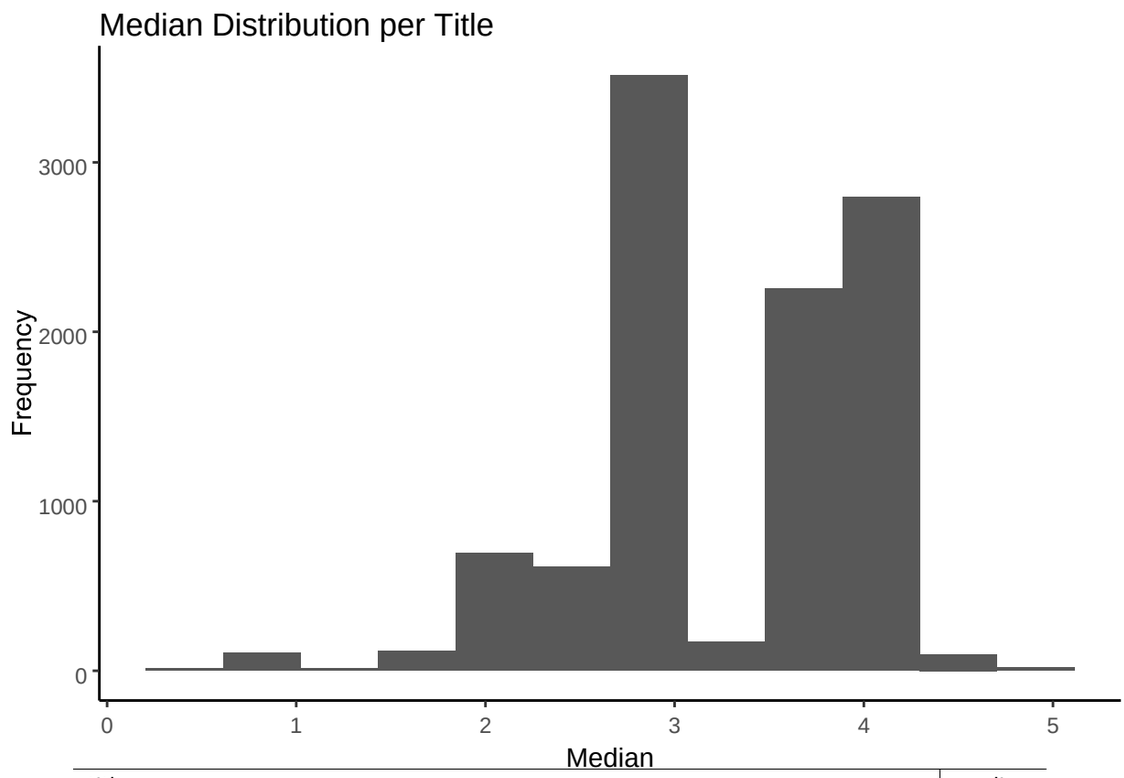
Fargo	85480
Seven (a.k.a. Se7en)	81084
Braveheart	78774
Shrek	78564
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)	77427
Ghost	77335
Who Framed Roger Rabbit?	76825
Mission: Impossible	75876
Princess Bride, The	74045
Dances with Wolves	69936
Blade Runner	69615
Batman Forever	69432
Mask, The	68200
Babe	68140

2.3.3 Mean Distribution per Title (Movie ID)



title	mean
Blue Light, The (Das Blaue Licht)	5.000000
Constantine’s Sword	5.000000
Fighting Elegy (Kenka erejii)	5.000000
Hellhounds on My Trail	5.000000
Satan’s Tango (SÃ¡tÃ¡ntangÃ3)	5.000000
Shadows of Forgotten Ancestors	5.000000
Sun Alley (Sonnenallee)	5.000000
Human Condition II, The (Ningen no joken II)	4.833333
Human Condition III, The (Ningen no joken III)	4.750000
Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	4.750000
Class, The (Entre les Murs)	4.666667
I’m Starting From Three (Ricomincio da Tre)	4.666667
Man Who Planted Trees, The (Homme qui plantait des arbres, L’)	4.571429
Bad Blood (Mauvais sang)	4.500000
CaÃ3tica Ana	4.500000
Demon Lover Diary	4.500000
End of Summer, The (Kohayagawa-ke no aki)	4.500000
Fires on the Plain (Nobi)	4.500000
Ladrones	4.500000
Life of Oharu, The (Saikaku ichidai onna)	4.500000
Man Named Pearl, A	4.500000
Mickey	4.500000
Please Vote for Me	4.500000
Power of Nightmares: The Rise of the Politics of Fear, The	4.500000
Testament of Orpheus, The (Testament d’OrphÃ©e)	4.500000

2.3.4 Median Distribution per Title (Movie ID)

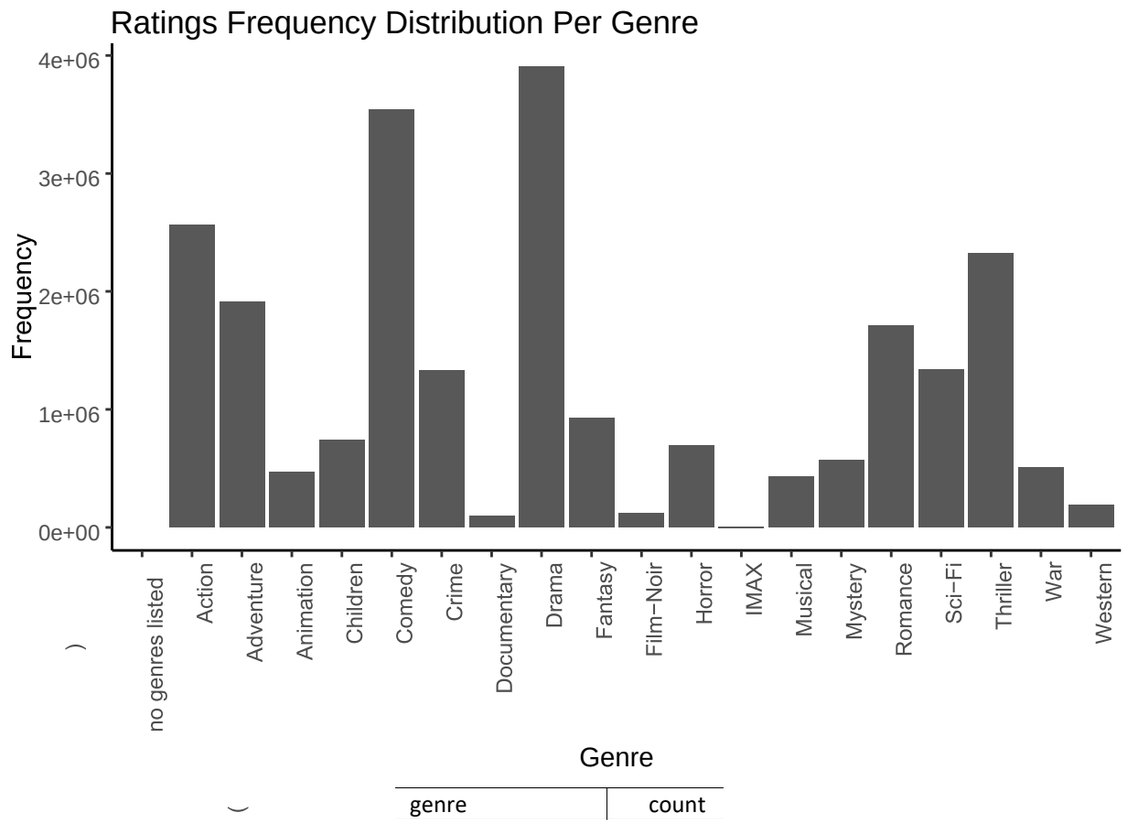


title	median
Aerial, The (La Antena)	5.00
Blue Light, The (Das Blaue Licht)	5.00
Class, The (Entre les Murs)	5.00
Constantine’s Sword	5.00
Fighting Elegy (Kenka erejii)	5.00
Godfather, The	5.00
Hellhounds on My Trail	5.00
Human Condition II, The (Ningen no joken II)	5.00
Jesus	5.00
Kids of Survival	5.00
Man Who Planted Trees, The (Homme qui plantait des arbres, L’)	5.00
Parallel Sons	5.00
Satan’s Tango (SÃ¡tÃ¡ntangÃ3)	5.00
Shadows of Forgotten Ancestors	5.00
Shawshank Redemption, The	5.00
Sun Alley (Sonnenallee)	5.00
Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)	5.00
World of Apu, The (Apur Sansar)	5.00
Human Condition III, The (Ningen no joken III)	4.75
400 Blows, The (Les Quatre cents coups)	4.50
49 Up	4.50
Amelie (Fabuleux destin d’AmÃ©lie Poulain, Le)	4.50
American Beauty	4.50
Andrei Rublev (Andrey Rublyov)	4.50
Bad Blood (Mauvais sang)	4.50

2.4 Genre Analysis

2.4.1 Rating Distribution per Genre

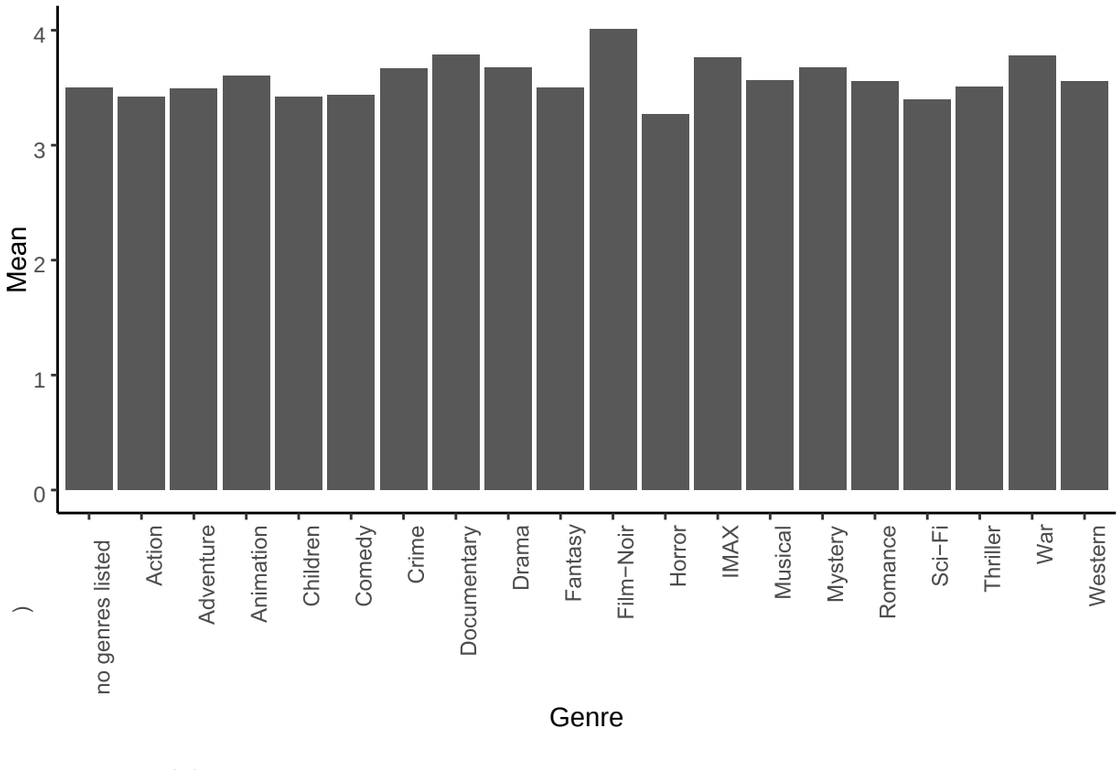
Overview of Rating distribution over Genre



genre	count
Drama	3909401
Comedy	3541284
Action	2560649
Thriller	2325349
Adventure	1908692
Romance	1712232
Sci-Fi	1341750
Crime	1326917
Fantasy	925624
Children	737851
Horror	691407
Mystery	567865
War	511330
Animation	467220
Musical	432960
Western	189234
Film-Noir	118394
Documentary	93252
IMAX	8190
(no genres listed)	6

2.4.2 Mean Distribution per Genre

Mean Distribution per Genre



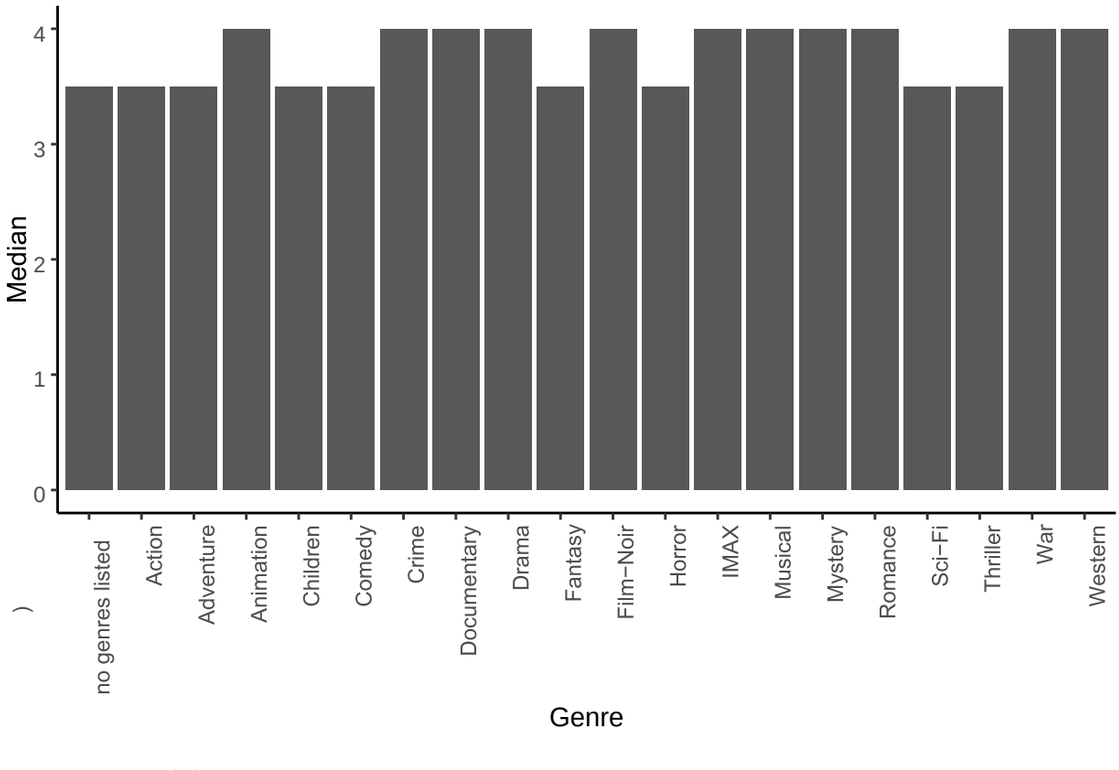
(

)

genre	mean
Film-Noir	4.011732
Documentary	3.784385
War	3.779457
IMAX	3.761844
Mystery	3.677412
Drama	3.673047
Crime	3.666151
Animation	3.599588
Musical	3.562761
Western	3.555122
Romance	3.553594
Thriller	3.506879
Fantasy	3.502419
(no genres listed)	3.500000
Adventure	3.494076
Comedy	3.437040
Action	3.421589
Children	3.418673
Sci-Fi	3.396756
Horror	3.269523

2.4.3 Median Distribution per Genre

Median Distribution per Genre



genre	median
Animation	4.0
Crime	4.0
Documentary	4.0
Drama	4.0
Film-Noir	4.0
IMAX	4.0
Musical	4.0
Mystery	4.0
Romance	4.0
War	4.0
Western	4.0
(no genres listed)	3.5
Action	3.5
Adventure	3.5
Children	3.5
Comedy	3.5
Fantasy	3.5
Horror	3.5
Sci-Fi	3.5
Thriller	3.5

3 Analysis - Model Building and Evaluation

3.1 Naive Baseline Model

The simplest model that someone can build, is a Naive Model that predict ALWAYS the mean. In this case, the mean is approximately 3.5.

[1] "The mean is: 3.52700364195256"

3.1.1 Naive Mean-Baseline Model The formula

used is:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0.

The RMSE on the validation dataset is **1.05**. It is very far for the target RMSE (below 0.87) and that indicates poor performance for the model.

3.2 Movie-Based Model, a Content-based Approach

The first Non-Naive Model takes into account the content. In this case the movies that are rated higher or lower resperct to each other. The formula used is:

$Y_{u,i} = \hat{\mu} + b_i + {}_{u,i}$ With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie $_i$, i.e. the bias of movie $_i$.

The RMSE on the validation dataset is **0.94**. It better than the Naive Mean-Baseline Model, but it is also very far from the target RMSE (below 0.87) and that indicates poor performance for the model.

3.3 Movie + User Model, a User-based approach

The second Non-Naive Model consider that the users have different tastes and rate differently.

The formula used is:

$Y_{u,i} = \hat{\mu} + b_i + b_u + {}_{u,i}$ With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie $_i$, i.e. the bias of movie $_i$. The b_u is a measure for the mildness of user $_u$, i.e. the bias of user $_u$.

The RMSE on the validation dataset is **0.8635** and this is very good. The Movie+User Based Model reaches the desidered performance but applying the regularization techniques, can improve the performance just a little.

3.4 Movie + User + Genre Model, the Genre Popularity

The formula used is:

$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + {}_{u,i}$ With $\hat{\mu}$ is the mean and $\varepsilon_{i,u}$ is the independent errors sampled from the same distribution centered at 0. The b_i is a measure for the popularity of movie $_i$, i.e. the bias of movie $_i$. The b_u is a measure for the mildness of user $_u$, i.e. the bias of user $_u$. The $b_{u,g}$ is a measure for how much a user $_u$ likes the genre $_g$.

The RMSE on the validation dataset is **0.8634** and this is very good. The Movie+User+Genre Based Model reaches the desidered performance but adding the genre predictor, doesn't improve significantly the model's performance. Applying the regularization techniques, can improve the performance just a little.

3.5 Regularization

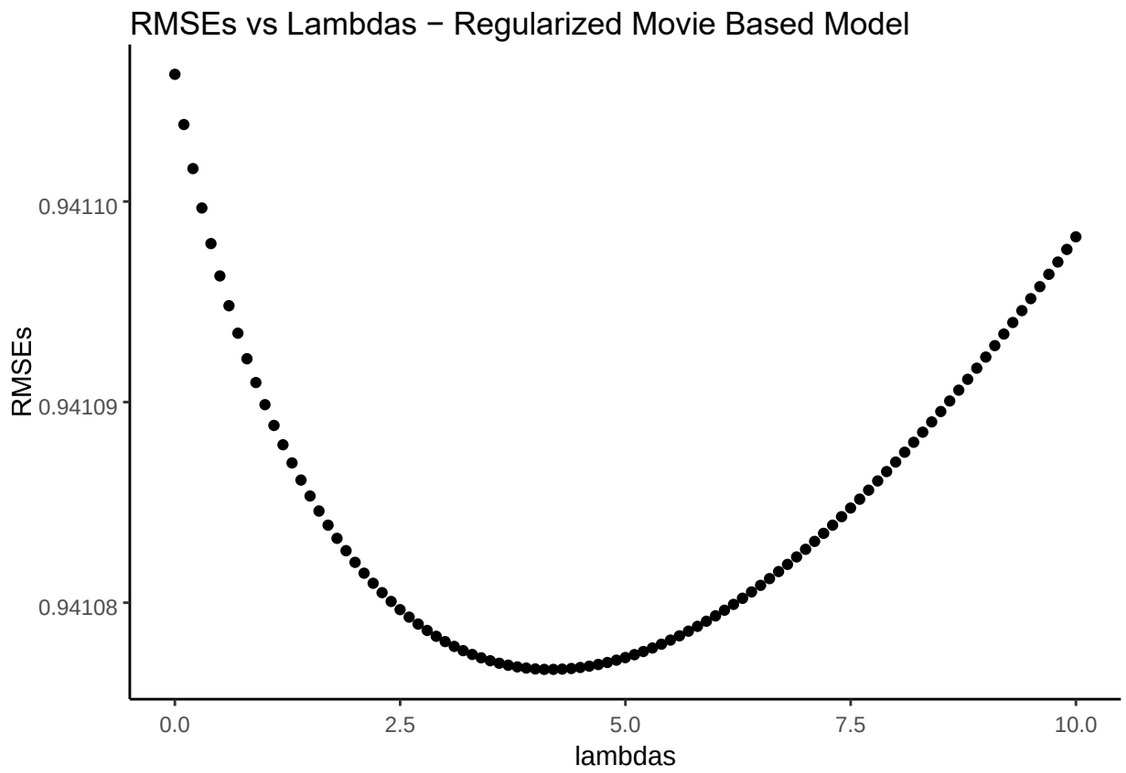
The regularization method allows us to add a penalty λ (lambda) to penalizes movies with large estimates from a small sample size. In order to optimize b_i, it necessary to use this equation:

$$\frac{1}{N_{u,i}} \sum (y_{u,i} - \mu - b_i)^2 + \lambda \sum b_i^2$$

reduced to this equation:

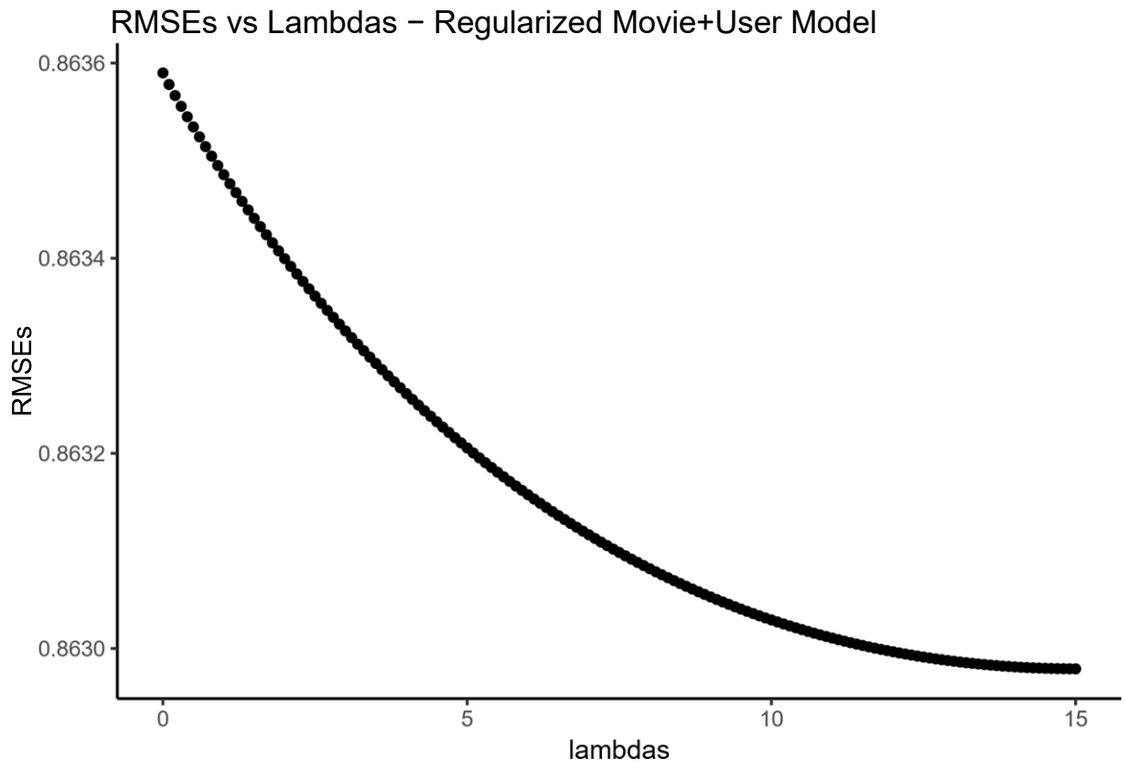
$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

3.5.1 Regularized Movie-Based Model



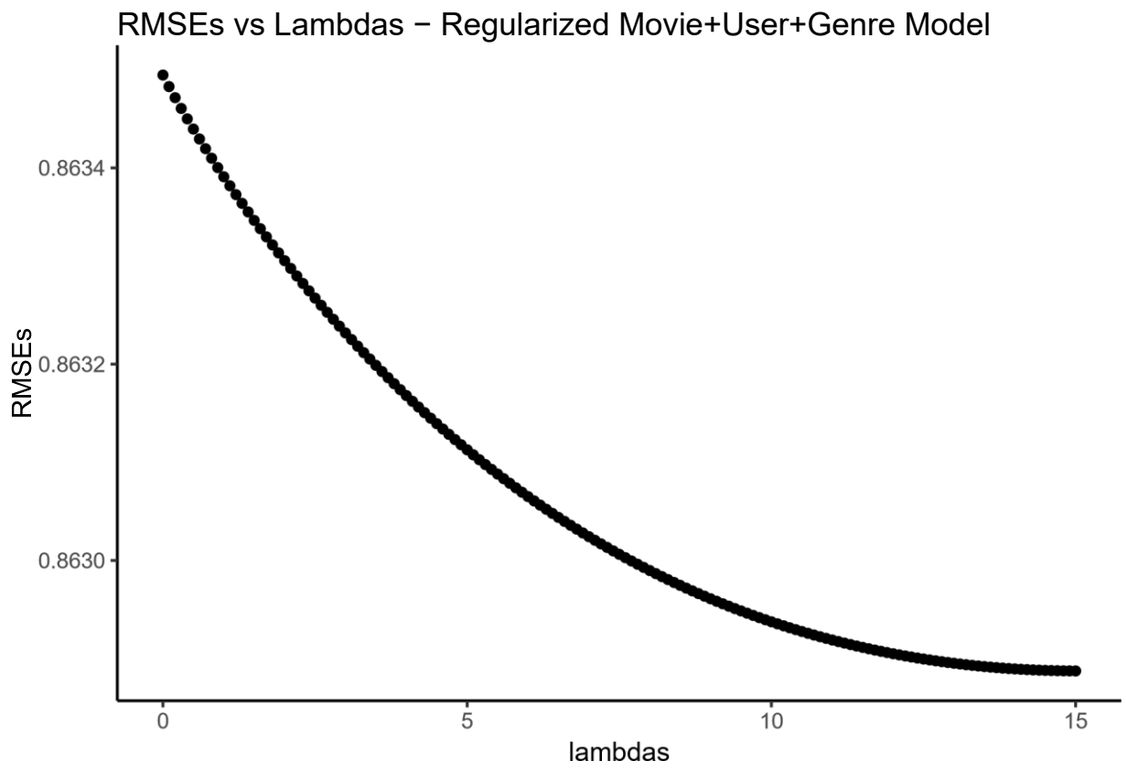
The RMSE on the validation dataset is **0.8635** and this is very good. The Movie+User Based Model reaches the desidered performance but applying the regularization techniques, can improve the performance just a little.

3.5.2 Regularized Movie+User Model



The RMSE on the validation dataset is **0.8629**. The Regularized Movie+User Based Model improves just a little the result of the Non-Regularized Model.

3.5.3 Regularized Movie+User+Genre Model



The RMSE on the validation dataset is **0.8628** and this is the best result of the builted models. The Regularized Movie+User+Genre Based Model improves just a little the result of the Non-Regularized Model. As the Non-Regularized Model, the genre predictor doesn't improve significantly the model's performance.

4 Results

This is the summary results for all the model builted, trained on edx dataset and validated on the validation dataset.

model	RMSE
Naive Mean-Baseline Model	1.0524433
Movie-Based Model	0.9411063
Movie+User Based Model	0.8635899
Movie+User+Genre Based Model	0.8634946
Regularized Movie-Based Model	0.9410767
Regularized Movie+User Based Model	0.8629791
Regularized Movie+User+Genre Based Model	0.8628874

5 Conclusion

After training different models, it's very clear that movieId and userId contribute more than the genre predictor. Without regularization, the model can archieves and overtakes the desidered peformance, but the best is the enemy of the good and applying regularization and adding the genre predictor, it make possible to reach a RSME of **0.8628** that is the best result for the trained models.

6 Appendix

6.1 1a - Initial Code privided by edX

```
##########################################################
# Create edx set, validation set, and submission file
########################################################## #

Note: this process could take a couple of minutes

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")

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

```
ratings <- read.table(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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[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)

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) write.csv(edx, "edx.csv")
```

6.2 1b - Code used in this report - MovieLens Project.R

```
# Install all needed libraries if it is not present

if(!require(tidyverse)) install.packages("tidyverse") if(!require(kableExtra))
install.packages("kableExtra") if(!require(tidyr)) install.packages("tidyr")
if(!require(tidyverse)) install.packages("tidyverse") if(!require(stringr))
install.packages("stringr") if(!require(forcats)) install.packages("forcats")
if(!require(ggplot2)) install.packages("ggplot2")

# Loading all needed libraries

library(dplyr) library(tidyverse)
library(kableExtra) library(tidyr)
library(stringr) library(forcats)
library(ggplot2)

# The RMSE function that will be used in this project is:
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) { sqrt(mean((true_ratings -
      predicted_ratings)^2))
}

# Convert timestamp to a human readable date

edx$date <- as.POSIXct(edx$timestamp, origin="1970-01-01") validation$date <-
as.POSIXct(validation$timestamp, origin="1970-01-01")

# Extract the year and month of rate in both dataset

edx$yearOfRate <- format(edx$date,"%Y") edx$monthOfRate <-
format(edx$date,"%m")

validation$yearOfRate <- format(validation$date,"%Y") validation$monthOfRate <-
format(validation$date,"%m")

# Extract the year of release for each movie in both dataset
# edx dataset

edx <- edx %>%
     mutate(title = str_trim(title)) %>% extract(title,
     c("titleTemp", "release"), regex = "^(.*) \\(([0-9 \\-
     ]*)\\)$", remove = F) %>%
```

```
            mutate(release = if_else(str_length(release) > 4, as.integer(str_split(release, "-",
                                                                                   simplify = T)[1]),
                                                    as.integer(release))
    ) %>%
    mutate(title = if_else(is.na(titleTemp),
                                    title,
                                    titleTemp) )
            %>%
  select(-titleTemp) #

validation dataset

validation <- validation %>% mutate(title = str_trim(title)) %>%
    extract(title, c("titleTemp", "release"), regex = "^(.*) \\(([0-9
    \\-]*)\\)$", remove = F) %>% mutate(release =
    if_else(str_length(release) > 4,
                                            as.integer(str_split(release, "-", simplify = T)[1]),
                                            as.integer(release))
    ) %>%
    mutate(title = if_else(is.na(titleTemp),
                                    title,
                                    titleTemp) )
            %>%
  select(-titleTemp)

# Extract the genre in edx datasets

edx <- edx %>% mutate(genre = fct_explicit_na(genres, na_level = "(no genres listed)") )
    %>%
    separate_rows(genre, sep =
                        "\\|")

# Extract the genre in validation datasets

validation <- validation %>% mutate(genre = fct_explicit_na(genres, na_level = "(no genres
    listed)") ) %>%
    separate_rows(genre, sep =
                        "\\|")

# remove unnecessary columns on edx and validation dataset

edx <- edx %>% select(userId, movieId, rating, title, genre, release, yearOfRate, monthOfRate) validation <- validation %>%

select(userId, movieId, rating, title, genre, release, yearOfRate, monthOfR

# Convert the columns into the desidered data type

edx$yearOfRate <- as.numeric(edx$yearOfRate) edx$monthOfRate <-
as.numeric(edx$monthOfRate) edx$release <- as.numeric(edx$release)
validation$yearOfRate <- as.numeric(validation$yearOfRate)
validation$monthOfRate <- as.numeric(validation$monthOfRate)
validation$release <- as.numeric(validation$release)

# Calculate the average of all movies mu_hat <-

mean(edx$rating)

# Predict the RMSE on the validation set

rmse_mean_model_result <- RMSE(validation$rating, mu_hat) # Creating a

results dataframe that contains all RMSE results

results <- data.frame(model="Naive Mean-Baseline Model", RMSE=rmse_mean_model_result)

# Calculate the average by movie

movie_avgs <- edx %>% group_by(movieId) %>%
    summarize(b_i = mean(rating - mu_hat))
```

```
# Compute the predicted ratings on validation dataset

rmse_movie_model <- validation %>%
    left_join(movie_avgs, by='movieId') %>%
    mutate(pred = mu_hat + b_i) %>% pull(pred)

rmse_movie_model_result <- RMSE(validation$rating, rmse_movie_model)

# Adding the results to the results dataset

results <- results %>% add_row(model="Movie-Based Model", RMSE=rmse_movie_model_result)

# Calculate the average by user

user_avgs <- edx %>% left_join(movie_avgs,
    by='movieId') %>% group_by(userId) %>%
    summarize(b_u = mean(rating - mu_hat - b_i)) # Compute the

predicted ratings on validation dataset

rmse_movie_user_model <- validation %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>% mutate(pred = mu_hat
    + b_i + b_u) %>% pull(pred)

rmse_movie_user_model_result <- RMSE(validation$rating, rmse_movie_user_model)

# Adding the results to the results dataset
results <- results %>% add_row(model="Movie+User Based Model", RMSE=rmse_movie_user_model_result)

genre_pop <- edx %>% left_join(movie_avgs,
    by='movieId') %>% left_join(user_avgs,
    by='userId') %>% group_by(genre) %>%
    summarize(b_u_g = mean(rating - mu_hat - b_i - b_u)) # Compute

the predicted ratings on validation dataset

rmse_movie_user_genre_model <- validation %>% left_join(movie_avgs,
    by='movieId') %>% left_join(user_avgs, by='userId') %>%
    left_join(genre_pop, by='genre') %>%
    mutate(pred = mu_hat + b_i + b_u + b_u_g) %>% pull(pred)

rmse_movie_user_genre_model_result <- RMSE(validation$rating, rmse_movie_user_genre_model)

# Adding the results to the results dataset

results <- results %>% add_row(model="Movie+User+Genre Based Model", RMSE=rmse_movie_user_genre_model_re lambdas <-

seq(0, 10, 0.1)

# Compute the predicted ratings on validation dataset using different values of lambda rmses <- sapply(lambdas,

function(lambda) {

   # Calculate the average by user

    b_i <- edx %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu_hat) / (n() + lambda)) # Compute the

    predicted ratings on validation dataset

    predicted_ratings <- validation %>% left_join(b_i,
        by='movieId') %>% mutate(pred = mu_hat + b_i)
        %>% pull(pred)

    # Predict the RMSE on the validation set

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

```
# Get the lambda value that minimize the RMSE min_lambda <-
lambdas[which.min(rmses)]

# Predict the RMSE on the validation set
rmse_regularized_movie_model <- min(rmses) #

Adding the results to the results dataset

results <- results %>% add_row(model="Regularized Movie-Based Model", RMSE=rmse_regularized_movie_model) rmses <-

sapply(lambdas, function(lambda) { # Calculate the average by user

    b_i <- edx %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

    # Calculate the average by user

    b_u <- edx %>% left_join(b_i, by='movieId')
        %>% group_by(userId) %>%
        summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

    # Compute the predicted ratings on validation dataset

    predicted_ratings <- validation %>% left_join(b_i,
        by='movieId') %>% left_join(b_u, by='userId') %>%
        mutate(pred = mu_hat + b_i + b_u) %>% pull(pred)

    # Predict the RMSE on the validation set

    return(RMSE(validation$rating, predicted_ratings))
})

# Get the lambda value that minimize the RMSE

min_lambda <- lambdas[which.min(rmses)] # Predict the

RMSE on the validation set

rmse_regularized_movie_user_model <- min(rmses) #

Adding the results to the results dataset

results <- results %>% add_row(model="Regularized Movie+User Based Model", RMSE=rmse_regularized_movie_u lambdas <- seq(0,

15, 0.1)

# Compute the predicted ratings on validation dataset using different values of lambda rmses <- sapply(lambdas,

function(lambda) {

    # Calculate the average by user
    b_i <- edx %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

    # Calculate the average by user

    b_u <- edx %>% left_join(b_i, by='movieId')
        %>% group_by(userId) %>%
        summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

      b_u_g <- edx %>%
        left_join(b_i, by='movieId') %>% left_join(b_u, by='userId')
        %>% group_by(genre) %>%
        summarize(b_u_g = sum(rating - b_i - mu_hat - b_u) / (n() + lambda))

    # Compute the predicted ratings on validation dataset

    predicted_ratings <- validation %>% left_join(b_i,
        by='movieId') %>% left_join(b_u, by='userId') %>%
```

```
        left_join(b_u_g, by='genre') %>% mutate(pred = mu_hat
        + b_i + b_u + b_u_g) %>% pull(pred)

    # Predict the RMSE on the validation set

    return(RMSE(validation$rating, predicted_ratings))
})

# Get the lambda value that minimize the RMSE

min_lambda <- lambdas[which.min(rmses)] # Predict the

RMSE on the validation set

rmse_regularized_movie_user_genre_model <- min(rmses) # Adding

the results to the results dataset

results <- results %>% add_row(model="Regularized Movie+User+Genre Based Model", RMSE=rmse_regularized_m
```