MovieLens Recommendation System - Capstone

Martinez Maldonado Matías

2023-02-17

Contents

0.1 Dataset Description

For this project, i will be creating a movie recommendation system using the **MovieLens dataset**. The version of movielens that i will use contains around 9 Millions movie ratings for train my model, which implies that i will divide this 9 Millions movie ratings into train and test set. And to validate the model i will use *final_holdout_test* with around 6.7 Millions movie ratings. This model will be evaluated by RMSE (Root Mean Squared Error).

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

```
# All Libraries we need
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.0
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(lubridate)
```

```
## Loading required package: timechange
##
## Attaching package: 'lubridate'
```

```
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(stringr)
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

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

options(timeout = 120)

## DataSet Preparation
dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

# Final hold-out test set will be 10% of MovieLens data
```

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

```
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)

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

# Function that return the RMSE value
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

0.2 Pre-Processing Data

First of all, we pre-processing the data to manipulate it easier.

```
sapply(edx, function(x) sum(is.na(x))) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("hover", "responsive", "bordered"),
                position = "center",
                font_size = 10,
                full_width = FALSE) %>%
  column_spec(1, color = "black", bold = TRUE) %>%
  column_spec(2, color = "white",
              background = "red") %>%
  row_spec(0, color = "black", bold = TRUE)
```

```
# Convert timestamp predictor into a human most readable format
edx$year <- edx$timestamp %>% as_datetime() %>% year()
edx$month <- edx$timestamp %>% as_datetime() %>% month()

#Extract the release date from title to a new predictor
```

```
edx <- edx %>% mutate(release_date = title %>% str_extract_all("\\([0-9]{4}\\)") %>%
                 str_extract("[0-9]{4}") %>% as.numeric(),
               title = title %>% str_remove("\\([0-9]{4}\\)")%>% str_trim("right"))

# Doing the same with the validation dataset in one step
final_holdout_test <- final_holdout_test %>% mutate(release_date = title %>% str_extract_all("\\([0-9]{4
                               str_extract("[0-9]{4}") %>% as.numeric(),
                             title = title %>% str_remove("\\([0-9]{4}\\)")%>% str_trim("right"),
                             year = timestamp %>% as_datetime() %>% year(),
                             month = timestamp %>% as_datetime() %>% month())


edx_temp <- edx %>% select(-timestamp)
```

0.3 Data Exploration

After pre-processing the data, we start the exploratory data analysis (EDA) to complain how best to manipulate data sources to get the answers we need, making it easier for us to discover patterns, spot anomalies, test a hypothesis, or check assumptions. Then we ask ourselves a series of questions.

```
str(edx_temp)
```

```
## 'data.frame':    9000055 obs. of  8 variables:
##  $ userId      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId     : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating      : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ title       : chr  "Boomerang" "Net, The" "Outbreak" "Stargate" ...
##  $ genres      : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Actio
##  $ year        : num  1996 1996 1996 1996 1996 ...
##  $ month       : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ release_date: num  1992 1995 1995 1994 1994 ...
```

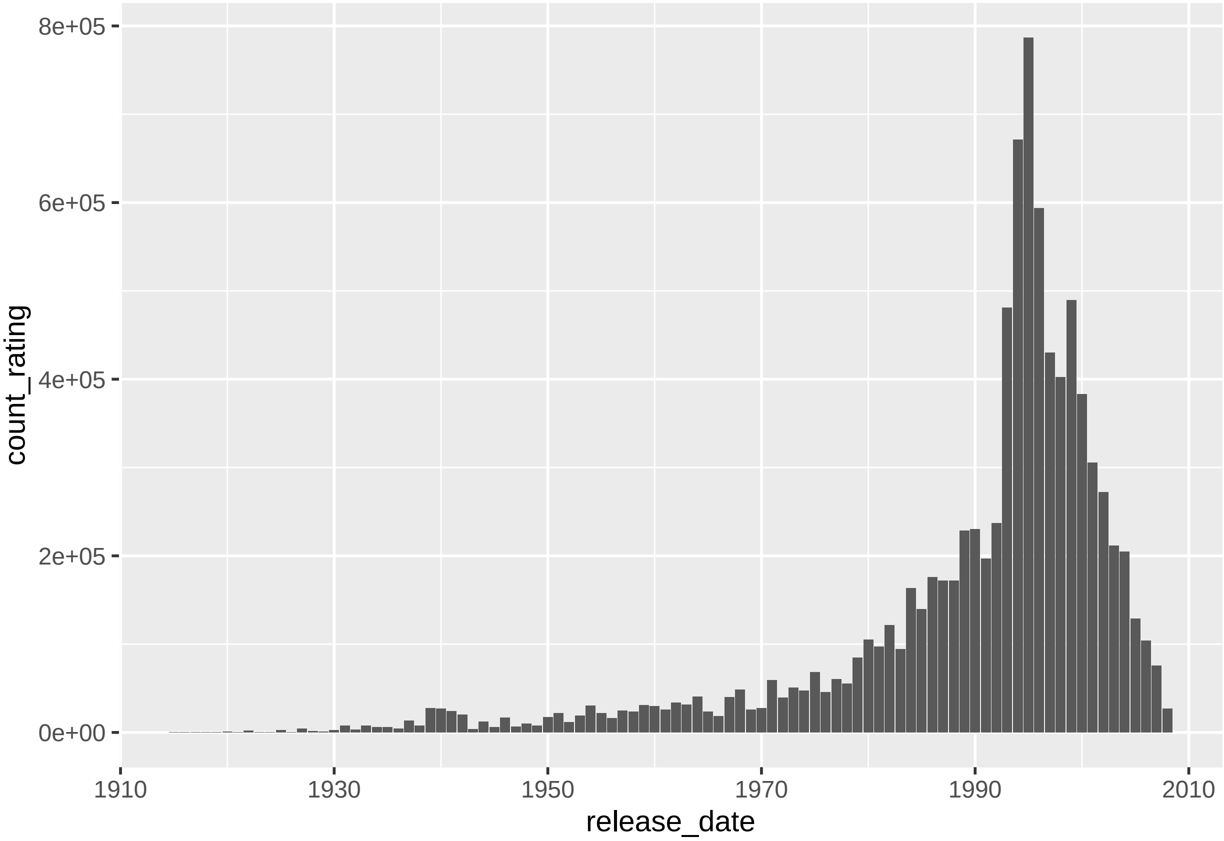
The features in both datasets are:

- **userId** `<integer>` that contains the unique identification number for each user.
- **movieId** `<integer>` 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.
- **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.
- **year** `<numeric>` that contains the year of each rating.
- **motnh** `<numeric>` that contains the month of each rating.
- **release_date** `<numeric>` that contains the year of each movie release date.

```
edx_temp %>% select(rating, title) %>% group_by(rating) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

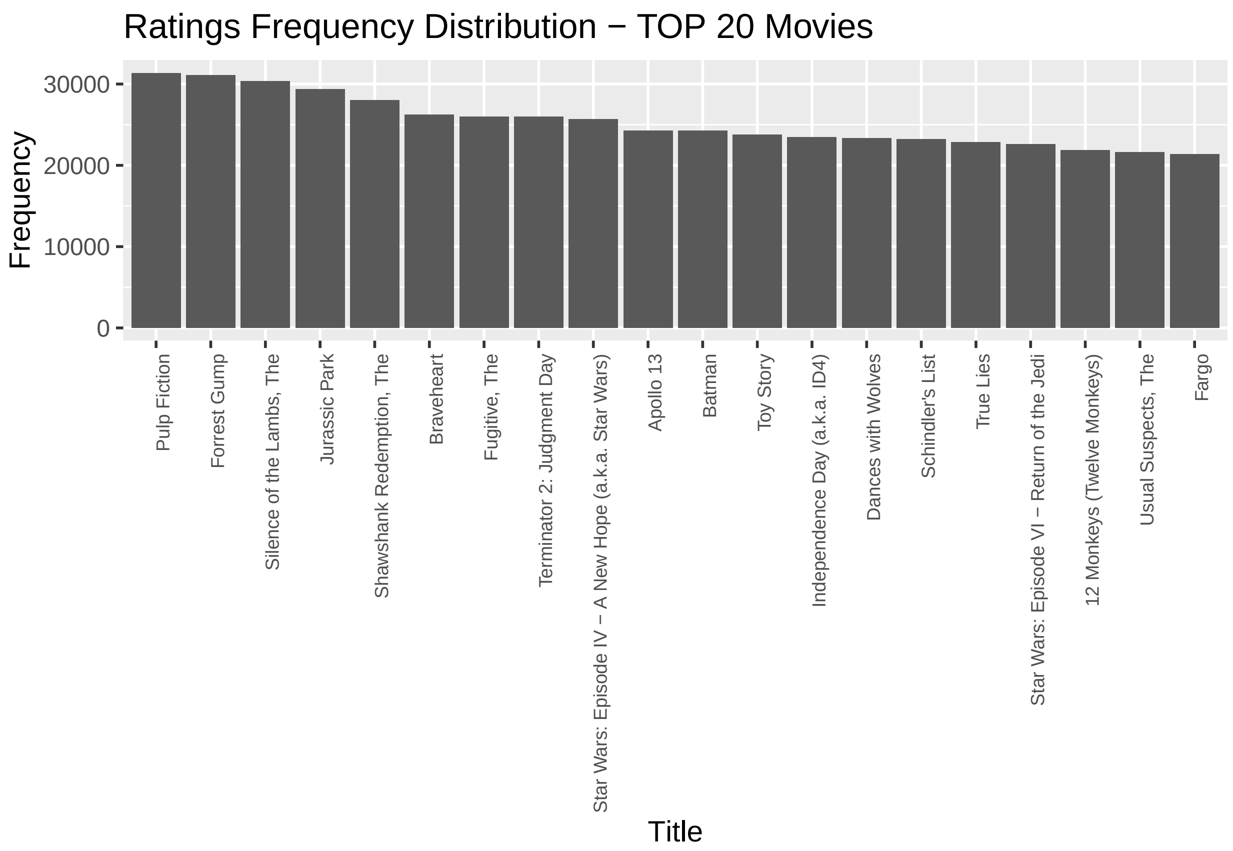
```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    4   2588430
##  2    3   2121240
##  3    5   1390114
##  4    3.5  791624
##  5    2    711422
##  6    4.5  526736
## 7     1    345679
## 8     2.5  333010
## 9     1.5  106426
## 10    0.5   85374
```

```
edx %>% group_by(release_date) %>% summarize(count_rating = n()) %>%
  ggplot(aes(release_date, count_rating)) +
  geom_col()
```



```
# Most 20 rated movies
edx %>% group_by(movieId, title) %>%
  summarize(count_rates = n()) %>%
  arrange(desc(count_rates)) %>% head(20) %>%
  ggplot(aes(reorder(title, count_rates, decreasing = TRUE), count_rates)) +
  geom_col() + theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 7)) +
  labs(title = "Ratings Frequency Distribution - TOP 20 Movies",
       x = "Title", y = "Frequency")
```

```
## `summarise()` has grouped output by 'movieId'. You can override using the
## `.groups` argument.
```

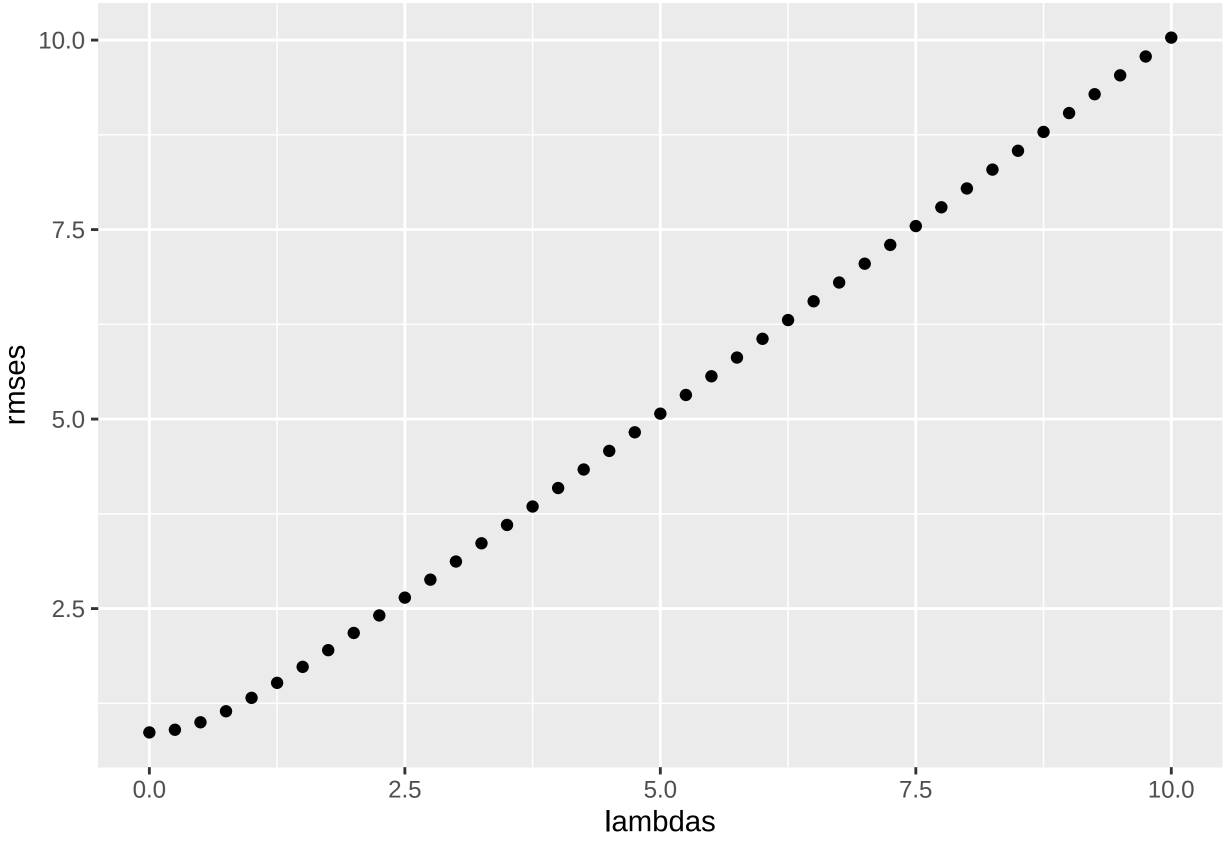


0.4 Creating the Model

```
# Create data partition

index_test <- createDataPartition(edx_temp$rating, times = 1, p=.25, list=FALSE)

train_set <- edx_temp %>% slice(-index_test)
test_set <- edx_temp %>% slice(index_test)

test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

#################################################################################

# Let's start with a naive approach
mu <- mean(train_set$rating)

naive_rmse <- RMSE(test_set$rating, mu)

results <- tibble(method = "Just the average", RMSE = naive_rmse)

# Movie effect method
movie_avgs <- train_set %>% group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```r
movie_effect <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)

rmse_movie_effect <- RMSE(test_set$rating, movie_effect)

results <- results %>% add_row(method="Movie Effect Model", RMSE=rmse_movie_effect)

# Movie + User effect method

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

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

rmse_user_effect <- RMSE(test_set$rating, user_effect)

results <- results %>% add_row(method="Movie + User Effect Model", RMSE=rmse_user_effect)

# Movie + User + Release Date effect method

release_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  group_by(release_date) %>%
  summarize(b_r = mean(rating - mu - b_i - b_u ))

release_effect <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  left_join(release_avgs, by = 'release_date') %>%
  mutate(pred = mu+ b_i + b_u + b_r) %>%
  pull(pred)

rmse_release_effect <- RMSE(test_set$rating, release_effect)

results <- results %>% add_row(method="Movie + User + Release Date Effect Model", RMSE=rmse_release_eff

# Regularization

lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

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

  release_avgs <- train_set %>% left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by = 'userId') %>%
    group_by(release_date) %>%
    summarize(b_r = sum(rating - mu - b_i - b_u )/ n()+ l)


  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(release_avgs, by = 'release_date') %>%
    mutate(pred = mu + b_i + b_u + b_r) %>%
    .$pred

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

qplot(lambdas, rmses)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



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

```r
results <- results %>% add_row(method= 'Regularized Movie + User Effect Model', RMSE = min(rmses))

#################################################################################

# Final test
final_holdout_test <- final_holdout_test %>% select(-timestamp)

final_holdout_test <- train_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

final_b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+ optimal_lambda))

final_b_u <- train_set %>%
  left_join(final_b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+ optimal_lambda))

final_rmse <- final_holdout_test %>%
  left_join(final_b_i, by = "movieId") %>%
  left_join(final_b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
#################################################################################

# Final Result
RMSE(final_holdout_test$rating, final_rmse)
```

```
## [1] 0.8554237
```