Movie Recommendation System

Bernardo Centeno

2023-07-28

Introduction

In October of 2006, Netflix launched a challenge to the data science community, where the goal was to improve their current movie recommendation system by 10%, offering a prize of one million dollars. A database with over 20 million ratings for over 27,000 movies by more than 138,000 users was provided. Similar to the Netflix challenge, the goal of this project was to build a movie recommendation system using the dataset called edx, a 10M version of the MovieLens dataset, which contains information about how different users rated different movies with diverse features. To achieve the purpose of predicting the rating by a given user for a given movie, the following steps were taken: 1) Data exploratory analysis, visualization, and cleaning, 2) Data splitting into train and test sets, 3) Model development and testing and 4) Model validation. A model with four terms was developed, with the first term representing the average of all movies, the second the movie-to-movie variation, the third the user-to-user variation, and the fourth the genres-to-genres variation. To account for average estimates based on few observations, the concept of regularization was implemented. Deploying this approach, a rmse of 0,8648 was obtained for the final model.

Methods/Analysis

Data cleaning, exploration and visualization

After loading the data and the necessary libraries, it was very important to know the structure of the edx dataset, a data frame with 9000055 observations and 6 columns representing each a different variable; the names of the columns were: userId (user), movieId, rating, timestamp, title (movie title), genres (movie genres)

```
library(tidyverse)

## ── Attaching packages ───────────────────────────────────────
tidyverse 1.3.1 ──

## ✓ ggplot2 3.4.0     ✓ purrr   0.3.4
## ✓ tibble  3.1.7     ✓ dplyr   1.0.9
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.2.2
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggplot2)
load("C:/Users/berna/Downloads/edx.RData")
load('C:/Users/berna/Downloads/final_holdout_test.RData')
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 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 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392
838984474 838983653 838984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak
(1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller"
"Action|Drama|Sci-Fi|Thriller" "Action|Adventure|Sci-Fi" ...
```

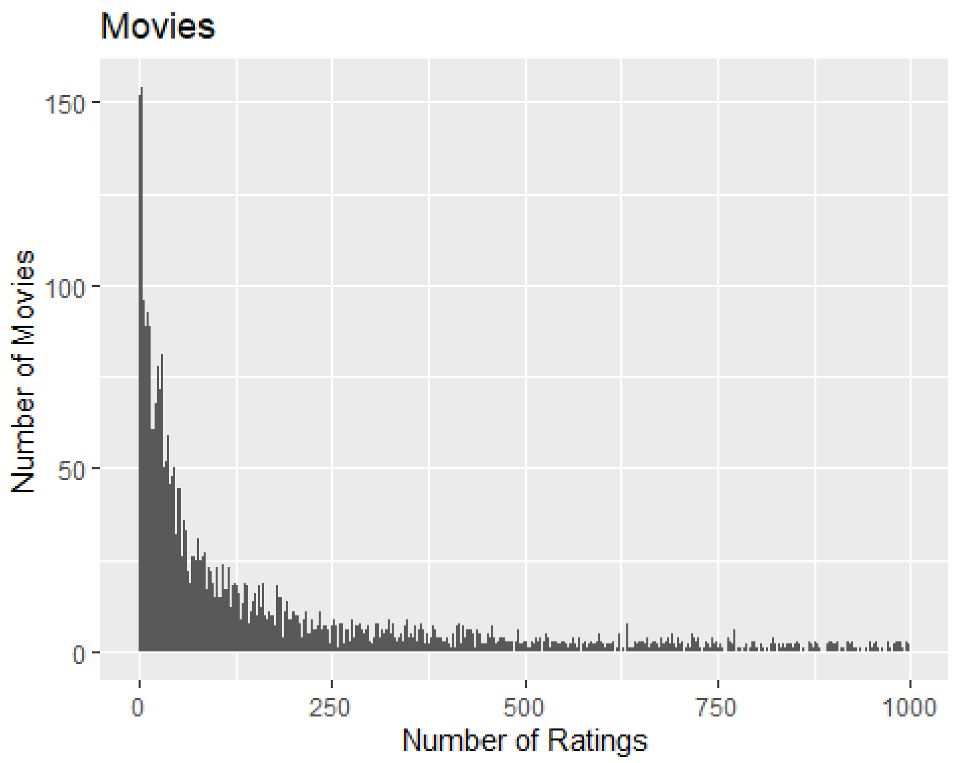
Then, it was relevant to rule out the presence of missing values (NAs):

```
any(is.na(edx))
```

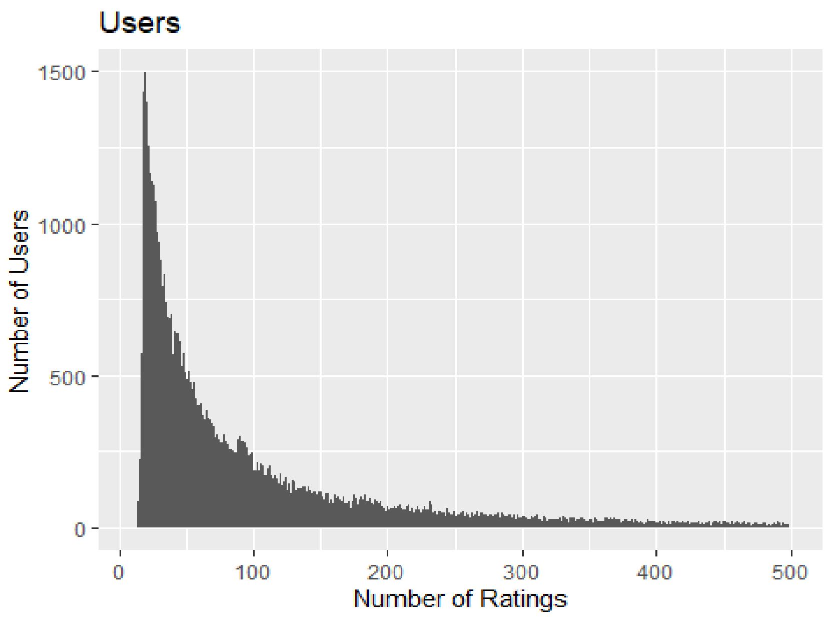
```
## [1] FALSE
```

From data visualization it was evident that:

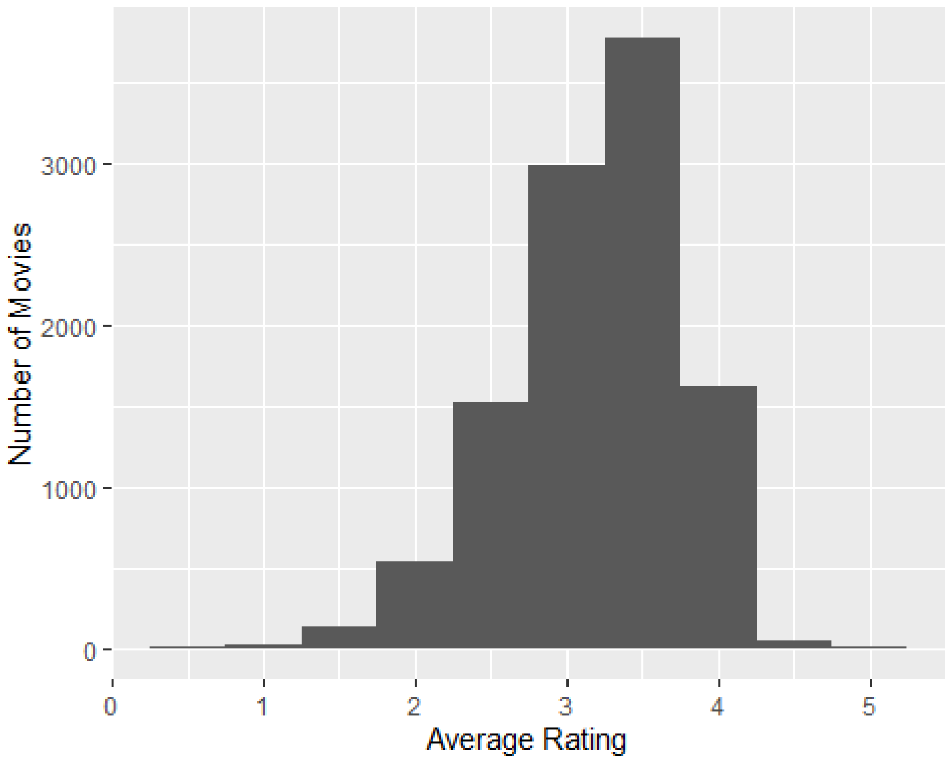
1) Some movies were rated more often than others:



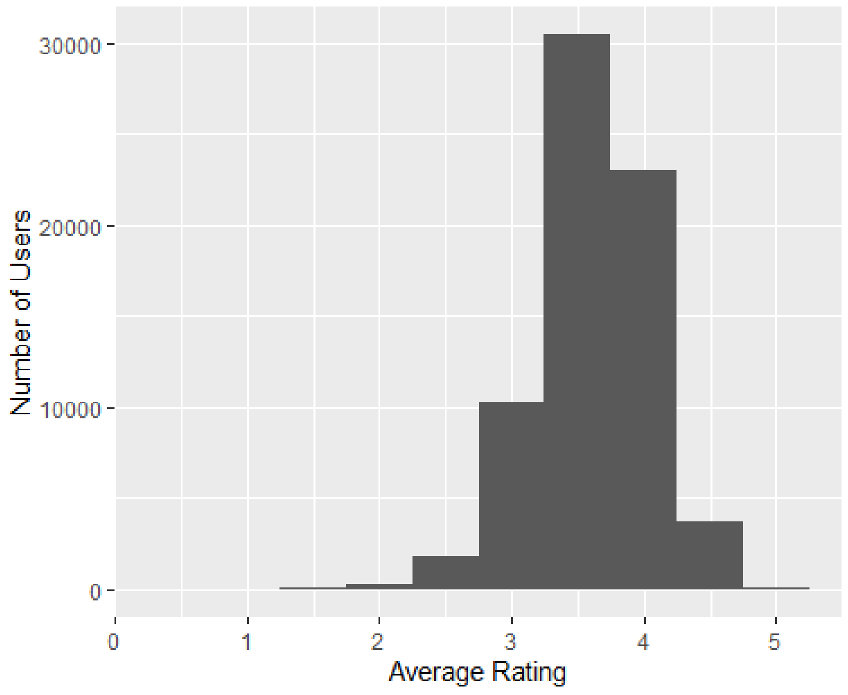
2) Some users rated more movies than others:



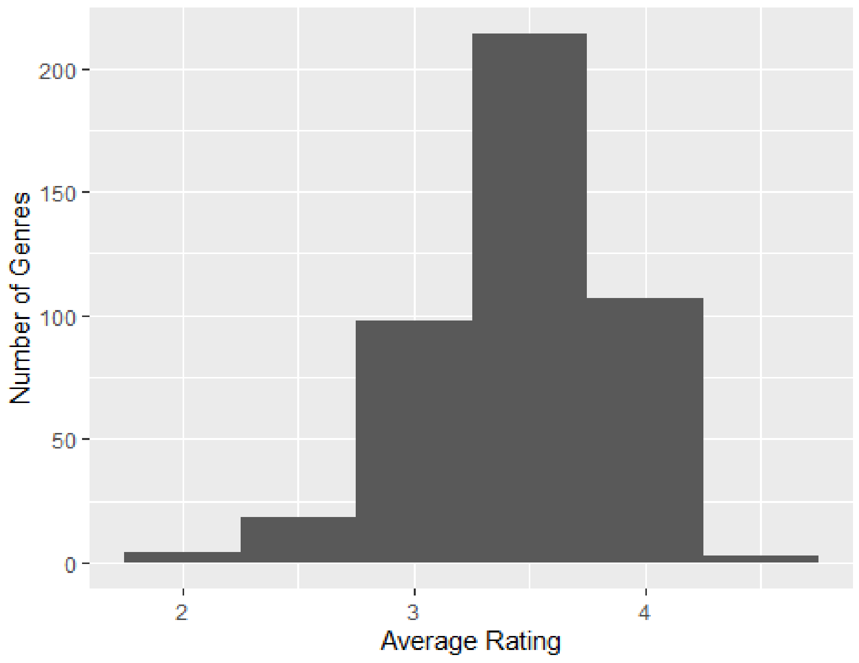
3) Some movies were rated higher than others:



4) Some users tend to rate movies higher:



5) Some genres were rated higher than others:



Data splitting

To have a set for training the model and a set to test it, the edx dataset was partitioned into a train_set and a test_set respectively:

```
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1,
list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

To make sure that the test_set didn´t have a movie or a user not present in the train_set, the following code was executed:

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

Model development and testing

Model 1

The first model assumes that the rating for any movie by any user (Yui) is the average of all movies (μ), and that random variation explains the differences (eu,i)

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

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

To test models, the residual mean square error (rmse) was used. The rmse reflects the average difference between the predicted and actual movie ratings, with a lower rmse indicating better predictive performance.

```
rmse1 <- RMSE(test_set$rating,mu)
rmse1

## [1] 1.0597
```

Model 2

As shown before, it was clear that some movies were rated higher than others; to model that variation, a new term was added: bi, the average rating for a movie i

Yu,i = μ + bi + eu,i

```
movie_avgs <- train_set %>% group_by(movieId)%>% summarise(bi =
mean(rating - mu))
pred1 <- left_join(test_set,movie_avgs)%>% mutate(pred = mu + bi)%>%
pull(pred)

## Joining, by = "movieId"

rmse2 <- RMSE(pred1,test_set$rating)
rmse2

## [1] 0.9430962
```

Model 3

Data also showed that some users tend to rate higher than others; to represent this user-to-user variation, the term bu was added to the model. bu was estimated as the average rating from user u

Yu,i = μ + bi + bu + eu,i

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

pred2 <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + bi + bu) %>%
  pull(pred)

rmse3 <- RMSE(pred2, test_set$rating)
rmse3

## [1] 0.8646243
```

Model 4

Since many movies were rated only a few times (sometimes just once), and as a consequence average rating estimates were based on few (or only one) observations, those estimates were unreliable. This was also true for average user rating estimates. To penalize those estimates based on few observations, regularization was applied.

First, the lambda parameter was defined

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  bi <- train_set %>%
    group_by(movieId) %>%
    summarize(bi = sum(rating - mu)/(n()+l))
  bu <- train_set %>%
    left_join(bi, by="movieId") %>%
    group_by(userId) %>%
    summarize(bu = sum(rating - bi - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(bi, by = "movieId") %>%
    left_join(bu, by = "userId") %>%
    mutate(pred = mu + bi + bu) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})
lambda <- lambdas[which.min(rmses)]
```

Then, both movie and user regularized averages were computed

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

user_reg_avgs <- train_set %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(bu = sum(rating - mu - bi)/(n()+lambda), n_i = n())
```

and the model was tested

```
pred3 <- test_set %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  left_join(user_reg_avgs, by='userId') %>%
  mutate(pred = mu + bi + bu) %>%
  pull(pred)
rmse4 <- RMSE(pred3, test_set$rating)
rmse4

## [1] 0.8640705
```

Model 5

Considering the genres effect detected during data visualization, a new term called g, representing the genres-to genres variation, was added

Yu,i = μ + bi + bu + g + eu,i

```
genres_avg <- train_set %>%
  left_join(movie_reg_avgs, by = 'movieId')%>%
  left_join(user_reg_avgs, by= 'userId')%>%
  group_by(genres)%>%
  summarize(g = mean(rating- mu - bi - bu))

pred4 <- test_set %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  left_join(user_reg_avgs, by='userId') %>%
  left_join(genres_avg)%>%
  mutate(pred = mu + bi + bu + g) %>%
  pull(pred)

## Joining, by = "genres"

rmse5 <- RMSE(pred4, test_set$rating)
rmse5

## [1] 0.863746
```

Model validation

The final model (model 5) was validated using the final_holdout_test

```
validation_predictions <- final_holdout_test %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  left_join(user_reg_avgs, by='userId') %>%
  left_join(genres_avg, by = 'genres')%>%
  mutate(pred = mu + bi + bu + g) %>%
  pull(pred)
rmse6 <- RMSE(validation_predictions,final_holdout_test$rating)
rmse6

## [1] 0.8648106
```

Results

Here is a summary table showing the RMSE obtained for each model tested on the test_set, and for the final model on the final_holdout_test

```
## # A tibble: 6 × 2
##   method                               RMSE
##   <chr>                               <dbl>
## 1 Just the average                    1.06 
## 2 Movie Effect Model                  0.943
## 3 Movie + User effect Model           0.865
## 4 Mov. + User + Reg. Model            0.864
## 5 Mov. + User + Reg. + Genres Model 0.864
## 6 Final Model Validation              0.865
```

As shown above,during the validation phase, the rmse obtained for the final model on the final_holdout_test was 0,8648.

Conclusion

In this project, we successfully built a movie recommendation system using the edx dataset, a 10M version of the MovieLens dataset, which contains information about how different users rated different movies. The goal was to predict the rating given by a user for a specific movie. A systematic approach was followed to achieve this objective, starting with data exploratory analysis, visualization, and cleaning, followed by data splitting into train and test sets, model development and testing and model validation.

Data was visualized to understand the distribution of ratings, the number of ratings for each movie and user, and the average ratings for movies, users, and genres.

To train and test the models, the dataset was splitted into training and testing sets. The initial model considered the average of all movies and assumed that random variation explained the differences. Subsequently, movie-to-movie and user-to-user variation were introduced , followed by regularization to penalize estimates based on few observations. Finally, we considered the effect of movie genres on ratings.

Upon model validation using the final_holdout_test dataset, the final model achieved an rmse of 0.8648.

Limitations:

Despite the successful results, it is essential to acknowledge the limitations of the movie recommendation system:

Movie Genres: Our model considers movie genres as an additional factor, but it’s essential to recognize that the genre categorization might not be comprehensive or completely accurate. Improving the genre classification could enhance model performance.

User Preferences: The dataset lacks detailed information about individual user preferences beyond movie ratings. Additional user-specific data, such as demographic information or past movie selections, could improve the personalization of the recommendation system.

Temporal Aspect: The final model did not consider the timing of movie ratings. User preferences may evolve over time, and considering temporal dynamics could have led to more dynamic and accurate recommendations.

Cold Start Problem: The model might face challenges when recommending movies for new users or movies with minimal ratings. The cold start problem requires careful consideration and alternative approaches for providing recommendations to new users.

Despite these limitations, this movie recommendation system demonstrates promising results and can serve as a foundation for further improvement and exploration in the field of recommendation systems.

In conclusion, a movie recommendation system was successfully developed to predict movie ratings. The model achieved a competitive rmse, indicating its potential to provide valuable movie recommendations to users. As the field of data science and machine learning continues to advance, further research and refinement in recommendation systems can enhance the accuracy and personalization of movie recommendations for users worldwide.