Movielens Recommendation System

12.01.2022

1. Introduction

The movielens dataset consists of 9000055 observations of six variables. It includes

- **userId** - the user identification number
- **movieId** - the identification number of each movie
- **timestamp** - the date and time of the rating
- **title** - the movie title
- **genres** - the movie genre
- **rating** - the movie rating raging from 0-5 in 0.5 steps

The aim is to predict the rating as an outcome using the columns aboth.

2. Analysis

2.1 The code provided to generate edx dataset and data format modification

First I used the code provided to generate the edx data set. I changed the data formats and used the "POSIXct"-function to generate a column for year, month and weekday from the $timestamp column.

2.2 Split the data into train and test set

Using to caret function 'createDataPartition' the edx data set is splitted into a test set (20%) and a training set (80%).

```r
data <- edx %>% select(rating, weekday, month, year, genres, movieId, userId)
test_index <- createDataPartition(data$rating, times = 1, p = 0.2, list = FALSE)
train_set <- data %>% slice( - test_index)
test_set <- data %>% slice(test_index)
```

2.3 Calculate baseline RMSE

The root mean square error (RMSE) is significantly higher than 1. The objective is to create an algorithm that achieves a RMSE lower than 1.06071.

Method	RMSE
baseline RMSE	1.06071

2.4 Create a prediction model

Create a prediction model using the
- ‘user-effect’
- ‘movie-effect’
- ‘genres-effect’
- ‘year-effect’
- ‘month-effect’
- ‘weekday-effect’

2.4.1 The effect of user

There were a total of 69878 different users that rated the movies.

```
userId_average <- train_set %>%
  group_by(userId) %>%
  summarize(userId_mean = mean(rating - baseline_mean))

ratings_prediction <- test_set %>%
  left_join(userId_average, by = "userId") %>%
  mutate(prediction = baseline_mean + userId_mean) %>%
  pull(prediction)

RMSE_user_effect <- RMSE(ratings_prediction, test_set$rating, na.rm=T)
```

Method	RMSE
baseline RMSE	1.06071
effect of user	0.979434

2.4.2 The effect of user and movie

There were a total of 10677 different movies rates.

```
movieId_average <- train_set %>%
  left_join(userId_average, by = "userId") %>%
  group_by(movieId) %>%
  summarize(movieId_mean = mean(rating - baseline_mean - userId_mean))

ratings_prediction <- test_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  mutate(prediction = baseline_mean + userId_mean + movieId_mean) %>%
  pull(prediction)

RMSE_user_movie_effect <- RMSE(ratings_prediction, test_set$rating, na.rm = T)
```

2.4.3 The effect of user, movie and genres

There were a total of 797 different genres.

```
genres_average <- train_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  group_by(genres) %>%
  summarize(genre_mean = mean(rating - baseline_mean - userId_mean - movieId_mean))

ratings_prediction <- test_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by = "genres") %>%
  mutate(prediction = baseline_mean + userId_mean + movieId_mean + genre_mean) %>%
  pull(prediction)

RMSE_user_movie_genre_effect <- RMSE(ratings_prediction, test_set$rating, na.rm=T)
```

Method	RMSE
baseline RMSE	1.06071
effect of user	0.979434
effect of user and movie	0.882949
effect of user, movie and genre	0.882949

2.4.4 The effect of user, movie, genres and year

Rating frequency was different over the years and there was a drop in ratings in 1998. The mean rating decreased over the years from 4.0 in 1995 to 3.46 in 2009.

```
years_average <- train_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by = "genres") %>%
  group_by(year) %>%
  summarize(year_mean = mean(rating - baseline_mean - userId_mean - movieId_mean - genre_mean))

ratings_prediction <- test_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by = "genres") %>%
  left_join(years_average, by = "year") %>%
  mutate(prediction = baseline_mean + userId_mean + movieId_mean + genre_mean + year_mean) %>%
  pull(prediction)

RMSE_user_movie_genre_year_effect <- RMSE(ratings_prediction, test_set$rating, na.rm = T)
```

Method	RMSE
baseline RMSE	1.06071
effect of user	0.979434
effect of user and movie	0.882949
effect of user, movie and genre	0.882949
effect of user, movie, genre and year	0.882056

2.4.5 The effect of user, movie, genres, year and month

Users gave more ratings in October, November and December, and ratings tended to be better in those months.

```
month_average <- train_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by= "genres") %>%
  left_join(years_average, by = "year") %>%
  group_by(month) %>%
  summarize(month_mean = mean(rating - baseline_mean - userId_mean - movieId_mean - genre_mean - year_me

ratings_prediction <- test_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by = "genres") %>%
  left_join(years_average, by = "year") %>%
  left_join(month_average, by = "month") %>%
  mutate(prediction = baseline_mean + userId_mean + movieId_mean + genre_mean + year_mean + month_mean)
  pull(prediction)

RMSE_user_movie_genre_year_month_effect <- RMSE(ratings_prediction, test_set$rating, na.rm = T)
```

Method	RMSE
baseline RMSE	1.06071
effect of user	0.979434
effect of user and movie	0.882949
effect of user, movie and genre	0.882949
effect of user, movie, genre and year	0.882056
effect of user, movie, genre, year and month	0.882

2.4.6 The effect of user, movie, genres, year, month and weekday

Users gave more ratings in the beginning of the week than on the weekend.The mean rating tended to be slightly worse on Wednesdays and Thursdays.

```
weekday_average <- train_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by= "genres") %>%
  left_join(years_average, by = "year") %>%
  left_join(month_average, by = "month") %>%
  group_by(weekday) %>%
  summarize(day_mean = mean(rating - baseline_mean - userId_mean - movieId_mean - genre_mean - year_mear

ratings_prediction <- test_set %>%
  left_join(userId_average, by = "userId") %>%
  left_join(movieId_average, by = "movieId") %>%
  left_join(genres_average, by = "genres") %>%
  left_join(years_average, by = "year") %>%
  left_join(month_average, by = "month") %>%
  left_join(weekday_average, by = "weekday") %>%
  mutate(prediction = baseline_mean + userId_mean + movieId_mean + genre_mean + year_mean + month_mean +
  pull(prediction)

RMSE_user_movie_year_genre_month_day_effect <- RMSE(ratings_prediction, test_set$rating, na.rm = T)
```

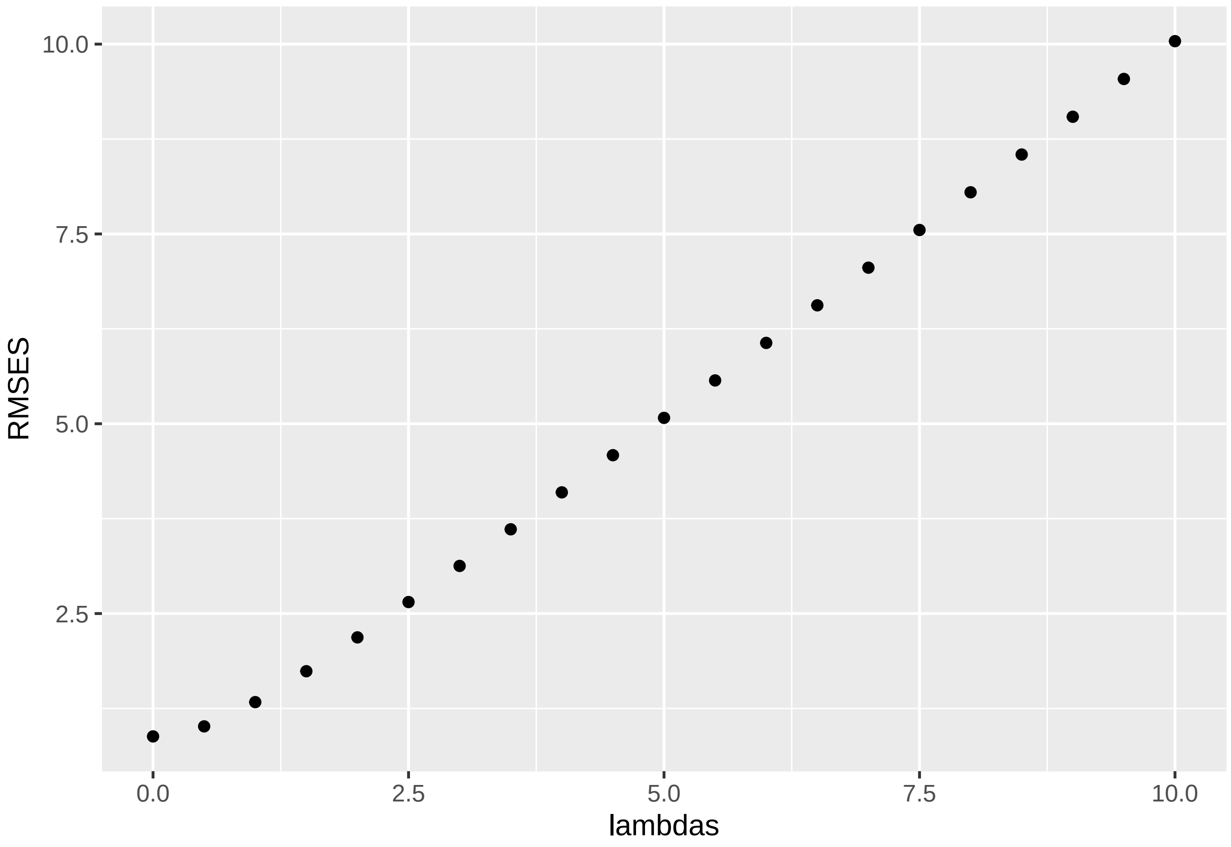
Method	RMSE
baseline RMSE	1.06071
effect of user	0.979434
effect of user and movie	0.882949
effect of user, movie and genre	0.882949

Method	RMSE
effect of user, movie, genre and year	0.882056
effect of user, movie, genre, year and month	0.882
effect of user, movie, genre, year, month and weekday	0.881995

2.5 Regularization

Next I've tried using regularization but the result was that a lambda value of 0 gave the smallest RMSE. It made no difference to the previous RMSE.

```
qplot(lambdas, RMSES)
```



4. Results

Method	RMSE
baseline RMSE	1.06071
effect of user	0.979434
effect of user and movie	0.882949
effect of user, movie and genre	0.882949
effect of user, movie, genre and year	0.882056
effect of user, movie, genre, year and month	0.882
effect of user, movie, genre, year, month and weekday	0.881995
prediction on validation data set	0.881923

4. Conclusion

My final RMSE on the validation data set is **0.8819228**. As predictors I used the effect of user, movie, genre, year, month and weekday. The RMSE is larger than the RMSE required to receive maximal points in the rating by the Havardx Data Science Capstone project but smaller than the initial RMSE of **1.060708** calculated by the average movie ratings. I've tried using regularization but the result was that a lambda value of 0 gave the smallest RMSE.