Movie Recommendation System

HarvardX Data Science PH125.9x Capstone - MovieLens

Erik Martins Tonon

September 2022

Contents

Introduction

Machine learning is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment (El Naqa and Murphy (2015)).

Recommendation systems have become prevalent in recent years as they deal with the information overload problem by suggesting to users the most relevant products from a massive amount of data (Wang et al. (2014)).

With over 10 million ratings and more than 10 thousand unique movies, the MovieLens dataset (Harper and Konstan (2015)) will be used in this project to create a movie recommendation system using Machine Learning.

The users were selected at random for inclusion and each user is represented by an id, and no other information was provided. The dataset contains not only ratings but movie genres, titles and the date when the review was submitted.

The goal of this project is to predict movie ratings based on the MovieLens dataset. To achieve this, the information collected from this dataset will be divided into two other datasets: the train dataset *(edx)* and the validation *(validation)* dataset, which consists of 10% of the original dataset and will not be used along the development of the recommendation system, only at the final hold-out test.

The objective is to predict ratings with a Root Mean Square Error *(RMSE)* less than **0.86490** against the ratings in the validation set.

The RMSE is defined as :

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

Exploratory Analysis

The dataset *'edx'* contains precisely 9.000.055 rows, 6 variables, ratings provided by 69.878 unique users and 10.677 unique movies.

A simple analysis already tells us that not every user had rated a movie, otherwise, we would have more than 746 million ratings.

Table 1: First 10 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
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

Users

Taking a closer look at "$userId", it is possible to identify some patterns of ratings in which some users have rated more movies than others (Figure 1).

It indicates that some users are more active than others which could suggest a potential user bias. This study could help improve the accuracy of the algorithm.

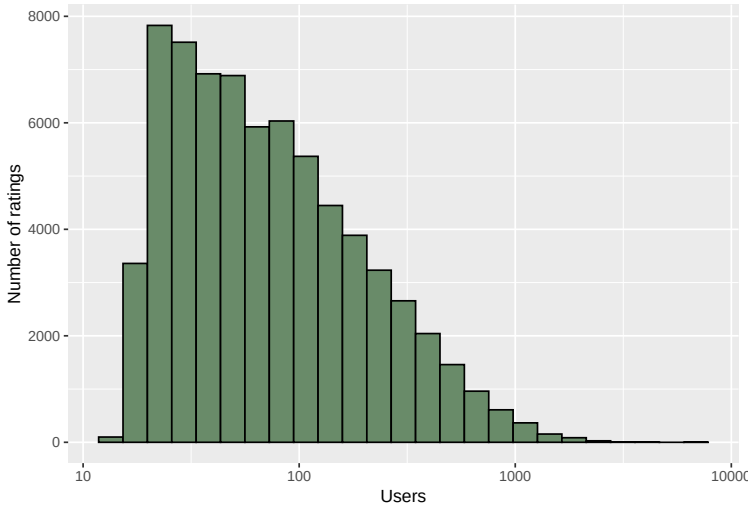

Figure 1: Ratings per User

Movies

The analysis of the movie data collected demonstrates significant variation in the number of ratings per movie (Figure 2). Some movies received higher ratings than others (Figure 3), stating that there is a movie effect on the rating, which can potentially improve our training algorithm.

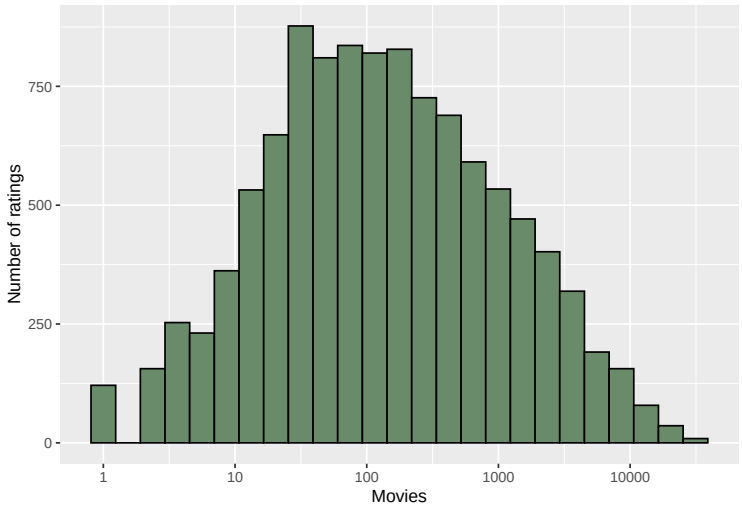

Figure 2: Ratings per Movie

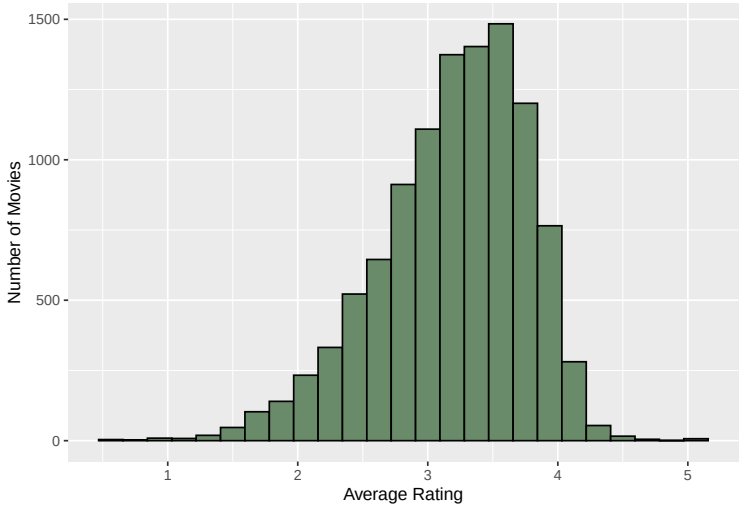

Figure 3: Average rating per Movie

Movie Genres

As seen in Table 1, the genres have been treated together *("Drama|Comedy|Romance")*, and a movie can be tagged with more than one genre.

"Drama", for instance, had the most rating (Table 2), while Documentary and IMAX (Table 3) had the fewest.

We can see that 6 ratings were provided for movies without genre.

Table 2: Most Rated

genres	count	rating
Drama	3909401	3.67
Comedy	3541284	3.44
Action	2560649	3.42

Table 3: Least Rated

genres	count	rating
Documentary	93252	3.78
IMAX	8190	3.76
(no genres listed)	6	3.50

Among the highest rates, *'Film-Noir'* has an average of 4.01 whereas *'Western'* has 3.56 (Table 4).

Table 4: Highest rates

genres	count	rating
Film-Noir	118394	4.01
War	511330	3.78
Documentary	93252	3.78
IMAX	8190	3.76
Mystery	567865	3.68
Drama	3909401	3.67
Crime	1326917	3.67
Animation	467220	3.60
Musical	432960	3.56
Western	189234	3.56

Movie Title

As seen before, the variable *'$Title'* contains both the name of the movie and the year of release. We see the top 10 most-rated movies topped by *'Pulp Fiction (1994)'* as the most rated movie (Table 5).

Table 5: Top 10 Most Rated Movies

title	n
Pulp Fiction (1994)	31336
Forrest Gump (1994)	31076
Silence of the Lambs, The (1991)	30280
Jurassic Park (1993)	29291
Shawshank Redemption, The (1994)	27988
Braveheart (1995)	26258
Terminator 2: Judgment Day (1991)	26115
Fugitive, The (1993)	26050
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25809
Batman (1989)	24343

Rating

We can also observe in the distribution of ratings (Figure 4) that 4 star ratings out of 5 were the most frequent and that half-integers stars were less frequent.

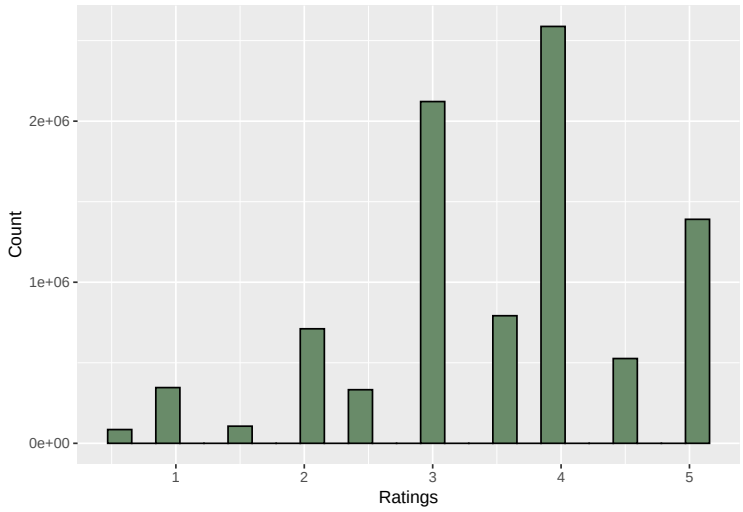

Figure 4: Histogram of ratings per movie

Review Date

The dataset was provided with a review date *($timestamp)* recorded when the rating was submitted. However, the variable needs to be transformed into human-readable information once it is in Epoch format.

We can observe that the earliest review was in 1995 while the Year 2000 has the highest amount of reviews. (Figure 5)

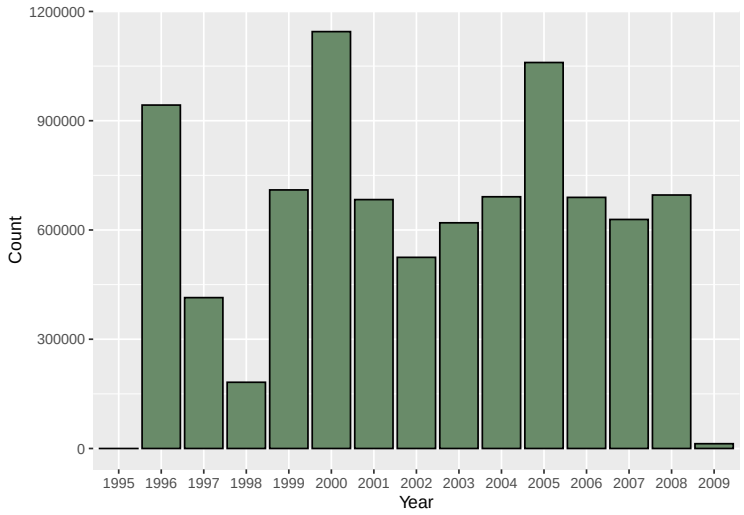

Figure 5: Number of ratings per Year

Methods

We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If this number is larger than 1, it means our typical error is larger than one star, which is not good. (Irizarry (2020))

Naive Recommendation

Due to the large size of the dataset, it will not be possible to fit the model using the *lm()* function as it can be very slow using normal computers, then we will calculate it manually.

The naive model is the most simple and it predicts that the movies will have the same rating regardless of user, genre or movie and the estimate that minimizes the RMSE is the average of all ratings.

We can define a model that assumes the same rating for all movies and users and for this model we have:

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

Method	RMSE
Objective	0.86490

Calculate the overall average rating across all movies included in train set and calculate RMSE between each rating included in test set and the overall average.

Method	RMSE
Objective	0.86490
Simple Average	1.06055

Movie Effect (b_i)

As we have seen before, some movies have a higher number of ratings than others, therefore we can improve our model by adding the movie effect (b_i).

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

```
#Dropping hat to represent estimates
mu <- mean(train_set$rating)

b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

Predict ratings adjusting for movie effects

```
pred_ratings <- mu + test_set %>%
  left_join(b_i, by = "movieId") %>%
  pull(b_i)
```

Calculate RMSE based on movie effects model

```
b_i_rmse <- format(round(RMSE(pred_ratings, test_set$rating),5), nsmall = 5)

rmse_results <- rmse_results %>%
  rbind(c("Movie Effect (b_i)", b_i_rmse))
```

We can see an improvement to the model as shown in the RMSE below:

Method	RMSE
Objective	0.86490
Simple Average	1.06055
Movie Effect (b_i)	0.94266

Movie, User effect (b_u)

We can also see the same bias by users (b_u), where some users rate less than others.

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

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

pred_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

```
b_u_rmse <- format(round(RMSE(pred_ratings, test_set$rating),5), nsmall = 5)

rmse_results <- rmse_results %>%
  rbind(c("Movie + User Effect (b_u)", b_u_rmse))
```

Adding users (b_u) to the algorithm has greatly improved the model, as suspected in the initial analysis.

Method	RMSE
Objective	0.86490
Simple Average	1.06055
Movie Effect (b_i)	0.94266
Movie + User Effect (b_u)	0.86460

Movie, Users and Genre Effect (b_g)

As we have seen in the data exploration, the movie genres (b_g) are treated together for each movie. Let's evaluate them using the following formula:

$$Y_{u,i} = \mu + b_i + b_u + b_g + \varepsilon_{u,i}$$

```
b_g <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

pred_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  left_join(b_g, by = c("genres")) %>%
  mutate(pred = mu + b_i  + b_u + b_g) %>%
  pull(pred)

b_g_rmse <- format(round(RMSE(pred_ratings, test_set$rating),5), nsmall = 5)

rmse_results <- rmse_results %>%
  rbind(c("Movie + User + Genre Effect (b_g)", b_g_rmse))
```

Adding genres (b_g) to the model has not improved the algorithm much.

Method	RMSE
Objective	0.86490
Simple Average	1.06055
Movie Effect (b_i)	0.94266
Movie + User Effect (b_u)	0.86460
Movie + User + Genre Effect (b_g)	0.86425

Movie, Users, Genre + Review Year Effect (b_y)

The dataset provides us information in Epoch format for when the review (b_y) was submitted and we can improve our model with the following formula:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_y + \varepsilon_{u,i}$$

```
b_y <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  left_join(b_g, by = "genres" ) %>%
  group_by(review_year) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u - b_g))

pred_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "review_year") %>%
  mutate(pred = mu + b_i  + b_u + b_g + b_y) %>%
  pull(pred)

b_y_rmse <- format(round(RMSE(pred_ratings, test_set$rating),5), nsmall = 5)

rmse_results <- rmse_results %>%
  rbind(c("Movie + Users + Genre + Review Year Effect (b_y)", b_y_rmse))
```

Adding the Review Year (b_y) to the model has slightly improved the algorithm, but we can achieve better results by adding regularization.

Method	RMSE
Objective	0.86490
Simple Average	1.06055
Movie Effect (b_i)	0.94266
Movie + User Effect (b_u)	0.86460
Movie + User + Genre Effect (b_g)	0.86425
Movie + Users + Genre + Review Year Effect (b_y)	0.86424

Regularization

To make better estimates, we have to penalize cases where movies were rated by fewer users. Otherwise, we will have noisy estimates. The general idea behind regularization is to constrain the total variability of the effect sizes. (Irizarry (2020))

$$\sum_{u,i} \left(y_{u,i} - \mu - b_i - b_u - b_g - b_y\right)^2 + \lambda \left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2 + \sum_y b_y^2\right)$$

Below at Figure 6, the value for λ that obtains the minimum RMSE for our model is 5.25.

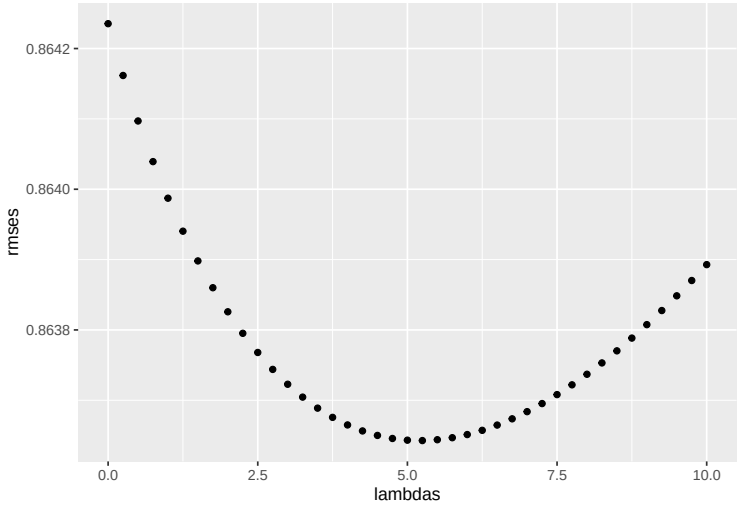

Figure 6: Minimum Lambda per RMSE

x
5.25

Applying the most optimized λ to the model.

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

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

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

b_y <- train_set %>%
```

```
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId" ) %>%
  left_join(b_g, by = "genres" ) %>%
  group_by(review_year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u - b_g)/(n()+lambda))

pred_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  left_join(b_g, by = "genres" ) %>%
  left_join(b_y, by = "review_year") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
  pull(pred)

reg_rmse <- format(round(RMSE(pred_ratings, test_set$rating),5), nsmall = 5)

rmse_results <- rmse_results %>%
  rbind(c("Regularized Movie + User + Genre + Year Effect", reg_rmse))
```

By performing full-cross validation with Regularized Movie + User + Genre + Year has greatly improved the model:

Method	RMSE
Objective	0.86490
Simple Average	1.06055
Movie Effect (b_i)	0.94266
Movie + User Effect (b_u)	0.86460
Movie + User + Genre Effect (b_g)	0.86425
Movie + Users + Genre + Review Year Effect (b_y)	0.86424
Regularized Movie + User + Genre + Year Effect	0.86364

Results

The model that combined the regularized Movie + User + Genre + Year Effect had the best results using the train and test data set, achieving the RMSE goal.

For the final validation, the model should be tested with the *validation* dataset.

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

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

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

b_g <- edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId" ) %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - mu - b_i - b_u)/(n()+lambda))

b_y <- edx %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId" ) %>%
  left_join(b_g, by = "genres" ) %>%
  group_by(review_year) %>%
  summarize(b_y = sum(rating - mu - b_i - b_u - b_g)/(n()+lambda))

pred_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = 'userId' ) %>%
  left_join(b_g, by = "genres" ) %>%
  left_join(b_y, by = "review_year") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
  pull(pred)

reg_rmse <- format(round(RMSE(pred_ratings, validation$rating),5), nsmall = 5)

rmse_results <- rmse_results %>%
  rbind(c("Final Validation", reg_rmse))
```

Finally, we evaluate the final hold-out test with the RMSE below the target.

Method	RMSE
Objective	0.86490
Simple Average	1.06055
Movie Effect (b_i)	0.94266
Movie + User Effect (b_u)	0.86460
Movie + User + Genre Effect (b_g)	0.86425
Movie + Users + Genre + Review Year Effect (b_y)	0.86424
Regularized Movie + User + Genre + Year Effect	0.86364
Final Validation	0.86463

The achieved RMSE 0.86463 with the *validation* data set is below the target and thus, enough for the project expectation.

Table 6: Top 10 Most Rated Movies - Validation data set

title
Shawshank Redemption, The (1994)
Godfather, The (1972)
Godfather, The (1972)
Goodfellas (1990)
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
Pulp Fiction (1994)
Schindler's List (1993)
Blade Runner (1982)
Shawshank Redemption, The (1994)
Usual Suspects, The (1995)

Conclusion

Given the size of the data set and the few variables available, it was possible to achieve the RMSE target by using the Regularized Movie, User, Genre and Review Year model. We explored the data set and identified some patterns around 'Users' and 'Movies' that could improve our algorithm.

The naive recommendation provided the worst result since it is the simple average regardless of user or movie. Afterwards, we saw that combining Movie and User in our algorithm had made a huge impact of around 18% compared to the simple average.

Finally by adding penalty to the model with regularization, it achieved the RMSE of **0.86364** with the test set and **0.86463** with the final validation with the validation data set - both below the target RMSE **0.86490**.

Having further user information, such as age, location and gender for instance, would have enriched this research.

References

El Naqa, Issam, and Martin J. Murphy. 2015. "What Is Machine Learning?" In *Machine Learning in Radiation Oncology: Theory and Applications*, edited by Issam El Naqa, Ruijiang Li, and Martin J. Murphy, 3–11. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-18305-3_1.

Harper, F. Maxwell, and Joseph A. Konstan. 2015. "The MovieLens Datasets: History and Context." *ACM Trans. Interact. Intell. Syst.* 5 (4). https://doi.org/10.1145/2827872.

Irizarry, Rafael A. 2020. *Introduction to Data Science: Data Analysis and Prediction Algorithms with r*. CRC Press.

Wang, Zan, Xue Yu, Nan Feng, and Zhenhua Wang. 2014. "An Improved Collaborative Movie Recommendation System Using Computational Intelligence." *Journal of Visual Languages & Computing* 25 (6): 667–75. https://doi.org/https://doi.org/10.1016/j.jvlc.2014.09.011.