

Recommendation System

Analysis on Movie Lens Dataset

ESC Rennes - Retail data analytics

ASKARI, Hadi

CHAUDHARY, Karan

SHARMA, Kanika

Contents

[Introduction 2](#_Toc513539873)

[Dataset 2](#_Toc513539874)

[Data Exploration 4](#_Toc513539875)

[Item Based Collaborative Filtering 7](#_Toc513539876)

[Collaborative Filtering 7](#_Toc513539877)

[Create Train and test Data 8](#_Toc513539878)

[Build Recommendation Model 8](#_Toc513539879)

[Prediction 8](#_Toc513539880)

[Evaluation 9](#_Toc513539881)

[USER Based Collaborative Filtering 11](#_Toc513539882)

[Build Model 11](#_Toc513539883)

[Prediction 11](#_Toc513539884)

[Evaluation 12](#_Toc513539885)

# Introduction

This report provides practical introduction to recommendation system (RS) and different techniques that can be used to create RS. RS can be used for recommending users relevant products or items based on user preference. User preference is a very subject topic and many articles are published on the same. For simplicity, in this report we give recommendation based on content consumed by user in past.

There are three different techniques for recommendation system as below, but we will use the first one only.

1. Collaborative Filtering. 2.) Content-Based Filtering. 3.) Hybrid method.

## Dataset

Movie Lens is a non-commercial, personalized movie recommendation system. It is a web-based system, where a virtual community recommends movies based on what user have watched in past. In year 2003 Movie Lens made 1 million records available. This dataset has been since then researched a lot. We have three different .csv files.

**User.csv: UserID—Gender—Age—Occupation--Zip-code**

Gender is denoted by a "M" for male and "F" for female

- Age is chosen from the following ranges:

1: "Under 18"

18: "18-24"

25: "25-34"

35: "35-44"

45: "45-49"

50: "50-55"

56: "56+"

- Occupation is chosen from the following choices:

0: "other" or not specified

1: "academic/educator"

2: "artist"

3: "clerical/admin"

4: "college/grad student"

5: "customer service"

6: "doctor/health care"

7: "executive/managerial"

8: "farmer"

9: "homemaker"

10: "K-12 student"

11: "lawyer"

12: "programmer"

13: "retired"

14: "sales/marketing"

15: "scientist"

16: "self-employed"

17: "technician/engineer"

18: "tradesman/craftsman"

19: "unemployed"

20: "writer"

**MovieID.CSV: Title--Genres**

- Titles are identical to titles provided by the IMDB (including

year of release)

- Genres are pipe-separated and are selected from the following genres:

1. Action
2. Adventure
3. Animation
4. Children's
5. Comedy
6. Crime
7. Documentary
8. Drama
9. Fantasy
10. Film-Noir
11. Horror
12. Musical
13. Mystery
14. Romance
15. Sci-Fi
16. Thriller
17. War
18. Western

- Some MovieIDs do not correspond to a movie due to accidental duplicate

entries and/or test entries

- Movies are mostly entered by hand, so errors and inconsistencies may exist

**Rating.csv: UserID—MovieID—Rating--Timestamp**

1. UserIDs range between 1 and 6040
2. MovieIDs range between 1 and 3952
3. Ratings are made on a 5-star scale (whole-star ratings only)
4. Timestamp is represented in seconds since the epoch as returned by time (2)
5. Each user has at least 20 ratings

# Data Exploration

Recommendation in R is done using Package recommenderlab using collaborative filtering, content based or hybrid. This package works on data structure of rating matrix to provide common interface for rating data. Many methods of matrix structures are used in this package for e.g.: dim (), rowcounts (), colmeans (), rowsums (). Hence before proceeding further, lets convert out dataset into matrix. We need to join two datasets ratings and movie into one and remove the movieid and timestamp. After merging, we should be able to get a data frame like below Figure 1.

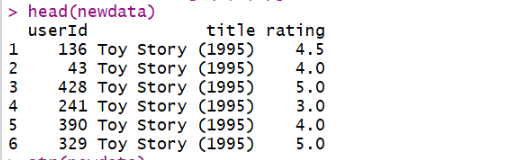


Figure : data frame of MovieLens dataset in R

Let’s change this data frame to realratingsmatrix format using below function in Figure 2.

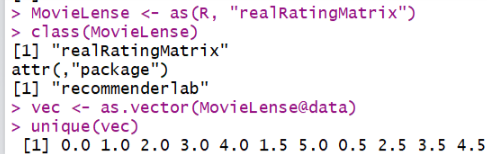


Figure : matrix format of data

Now in this data structure, each row corresponds to a user, each column to a movie, and each value to a rating. We can extract the size of the data structure using dim function, which gives the information that we have 591 users and 138 movies. A rating equal to 0 represents a missing value, so we can remove them from matrix. Now, we can build a frequency plot of the ratings. To visualize a bar plot with frequencies, we can use ggplot2. Let's convert them into categories using factor and build a quick chart.

It is clear from the Figure 3, four and three ratings are the most common ratings and one is least ratings. Let’s explore the number of times movie have been viewed. Fir this, Colcounts and colmeans are the effective functions to find out the number of non-missing values and average value in each column. Let’s try to visualize the data using ggplot.

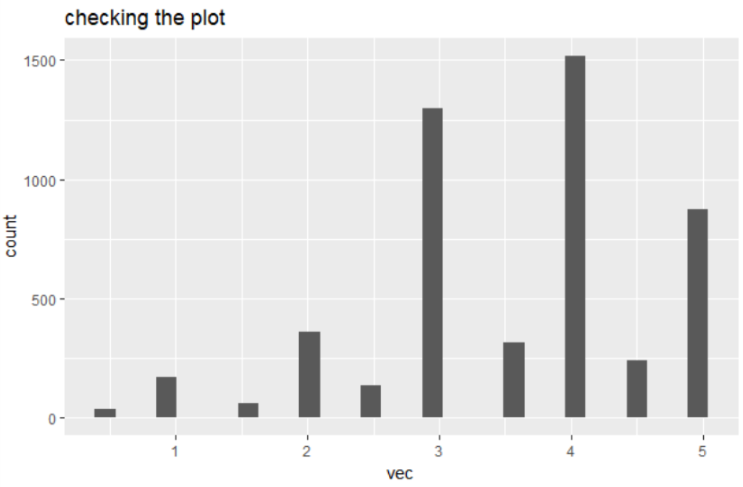
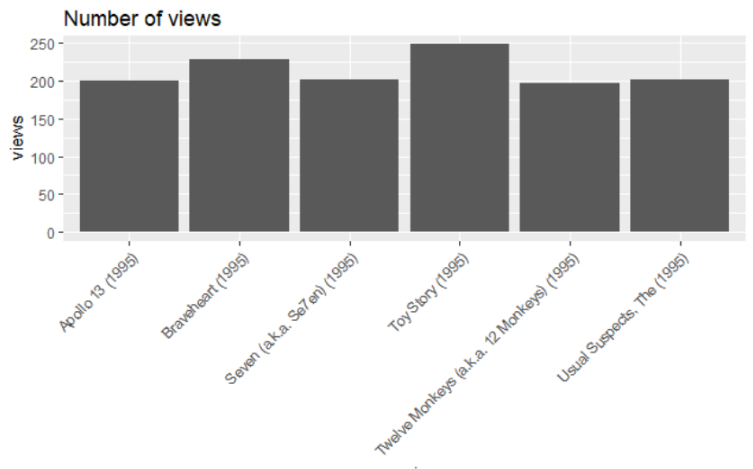


Figure : Distribution of ratings



In the preceding chart, Figure 4, you can notice that **Star Wars (1977)** is the most viewed movie, exceeding the others by about 100 views.

Figure : the number of views of the top movies

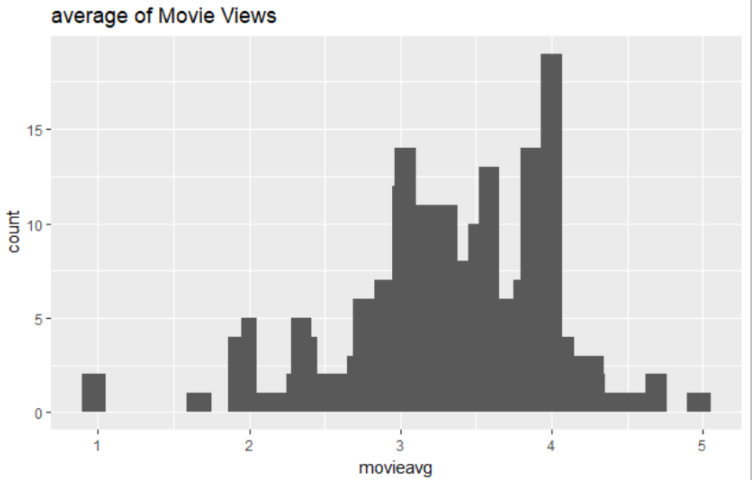


Figure : distribution of the average movie ratings

In the Figure 5 the highest value is around 4, and there are a few movies whose rating is either 1 or 5. Probably, the reason is that these movies received a rating from a few people only, so, we shouldn't take them into account. We can remove the movies whose number of views are below a defined threshold. 100.Now we can different average rating s.

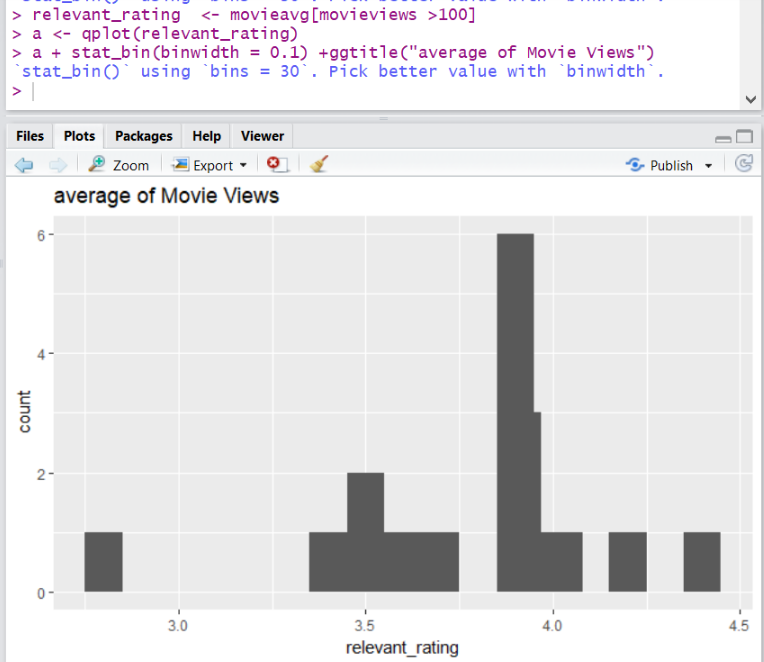


Figure : the average of movie views after removing the extremes

In the Figure 6 all the rankings are between 3.5 and 4. As expected, we removed the extremes. The highest value changes, and now, it is around 4. We can visualize the matrix by building a heat map whose colors represent the ratings. But also, we need to clean the data and take the quantile containing the quality data (Figure 7).

|  |
| --- |
|  |

Figure : heatmap of the rating matrix

The reason to do perform filtering is that we see a lot of white area in the first image and it indicates that there are too many users that have rated very less movies and hence it does not make sense to read such data. we select the most relevant users and items, and this means visualizing only the users who have seen many movies and the movies that have been seen by many users.

We need to determine the minimum number of users per movie and vice versa. The correct solution comes from an iteration of the entire process of preparing the data, building a recommendation model, and validating it. Since we are implementing the model for the first time, we can use a rule of thumb. After having built the models, we can come back and modify the data preparation. We will define ratings\_movies containing the matrix that we will use. It takes account of: Users who have rated at least 10 movies and Movies that have been watched at least 10 times. Now this matrix in Figure 8 contains about 157 users and 83 movies in comparison with MovieLense.

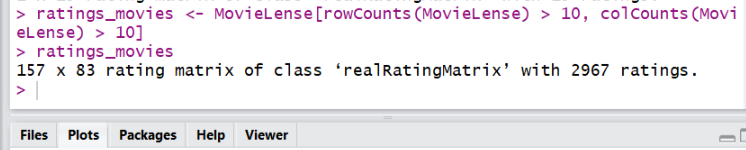


Figure : Figure 8: rating matrix of class 'realRatingMatrix'

Let's take account of the users having watched more movies. Most of them have seen all the top movies, and this is not surprising. We can notice some columns that are darker than the others. These columns represent the highest-rated movies. Conversely, darker rows represent users giving higher ratings. This might bias the results. We can remove this effect by normalizing the data in such a way that the average rating of each user is 0.

# Item Based Collaborative Filtering

## Collaborative Filtering

Collaborative filtering is a branch of recommendation that takes account of the information about different users. The word "collaborative" refers to the fact that users collaborate with each other to recommend items. In fact, the algorithms take account of user purchases and preferences. The core algorithm is based on these steps:

1. For each two items, measure how similar they are in terms of having received similar ratings by similar users

2. For each item, identify the *k*-most similar items

3. For each user, identify the items that are most like the user's purchases

## Create Train and test Data

We split the data in training and test data where training includes users from which model learns and test dataset include users to whom we recommend movies (Figure 9). Here, for each item, it identifies its k most similar items and stores it.

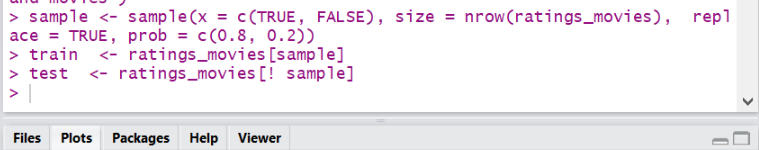


Figure : splitting data in training and test

## Build Recommendation Model

Figure 10 shows how to build the model using recommender function and defining the method “IBCF” and similarity function like cosine and value of k.

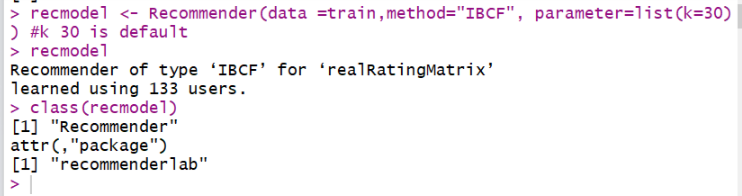


Figure : Building recommendation system model

## Prediction

Now we should be able to recommend movies to the user (Figure 11). Albeit we need to provide the number of items to be recommended in the test. For each user, the algorithm extracts its rated movies. For each movie, it identifies all its similar items, starting from the similarity matrix. Then, the algorithm ranks each similar item in this way. Extract the user rating of each purchase associated with this item. The rating is used as a weight. Extract the similarity of the item with each purchase associated with this item. Multiply each weight with the related similarity and sum everything up.

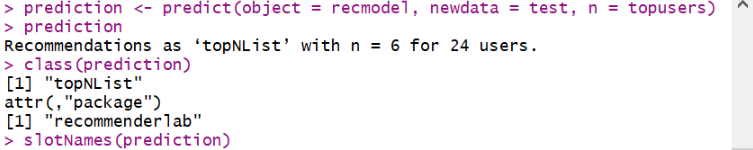


Figure : prediction the model on the test

Here we gave 6 as the number of items to be recommended to users. Let’s see which movies they are.

Let’s find out the most recommended movie by plotting the matrix in Figure 12.

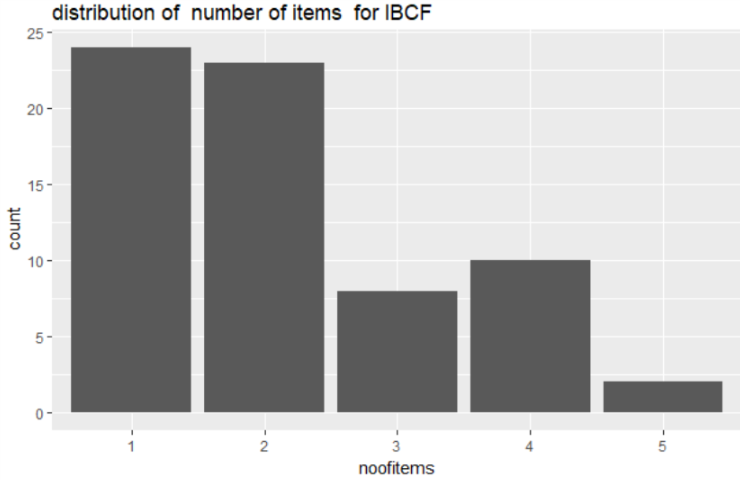


Figure : distribution of the most recommended movie

Figure 13 shows the item number 1 and 2 has been recommended the highest times as compared to rest items. Let’s see their names. Top recommended movies are as following.

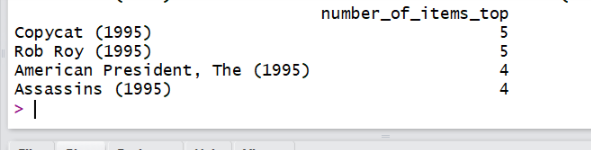


Figure : the name of the most recommended movies

IBCF recommends items based on the similarity matrix. It's an eager-learning model, that is, once it's built, it doesn't need to access the initial data. For each item, the model stores the k-most similar, so the amount of information is small once the model is built. This is an advantage in the presence of lots of data. In addition, this algorithm is efficient and scalable, so it works well with big rating matrices. Its accuracy is rather good, compared with other recommendation models.

## Evaluation

Evaluation scheme is the function used for the creating the evaluation sets from the data. We get following scheme (after executing this function

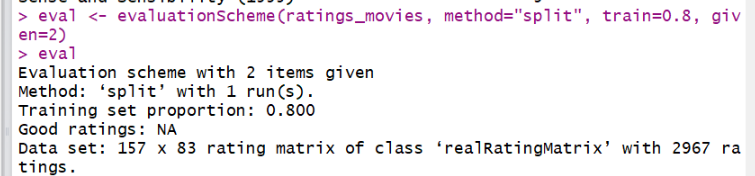


Figure : evaluation scheme with 2 items given

To extract the sets, we need to use get Data. There are three sets:

* train: This is the training set
* known: This is the test set, with the item used to build the recommendations
* unknown: This is the test set, with the item used to test the recommendations

Now is the time to evaluate the ratings given by algorithm need to define the model to evaluate. Let's build it using the Recommender function and the matrix with the predicted ratings using the predict function. The function to measure the accuracy is calcPredictionAccuracy and it computes the following aspects:

* **Root mean square error (RMSE)**: This is the standard deviation of the difference between the real and predicted ratings.
* **Mean squared error (MSE)**: This is the mean of the squared difference between the real and predicted ratings. It's the square of RMSE, so it contains
* the same information.
* **Mean absolute error (MAE)**: This is the mean of the absolute difference between the real and predicted ratings

Let's look at the distribution of RMSE by a user (Figure 15): To have a performance index of the whole model, we need to compute the average indices, specifying by User = FALSE.



Figure : distribution of movies per user by IBCF

We get Root square mean error of IBCF to be 1.51. Such values help us to compare the models.

# USER Based Collaborative Filtering

## Build Model

In this section, we will use the opposite approach (Figure 16). First, given a new user, we will identify its similar users. Then, we will recommend the top-rated items purchased by similar users. Identify the most similar users. Take account of the top k users (k-nearest neighbors). Take account of the users whose similarity is above a defined threshold. Rate the items purchased by the most similar users. The rating is the average rating among similar users and the approaches are Average rating or Weighted average rating, using the similarities as weights and lastly, Pick the top-rated items.

We provide the value of similarity function among users and the number of similar users to be found in the function and we get following results.

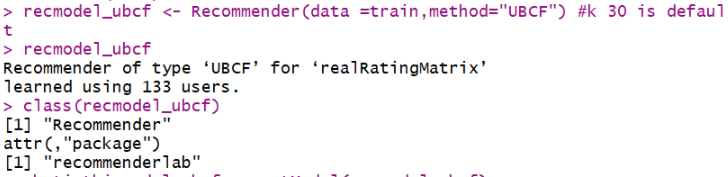


Figure : user-based Collaborative Filtering

This method is a lazy-learning technique, which means that it needs to access all the data to perform a prediction.

## Prediction

Let’s try to predict results on test data with this model and see top 6 recommendations for each user.

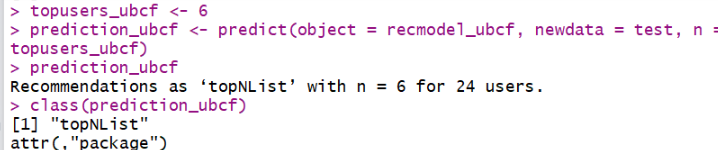


Figure : prediction of user-based Collaborative Filtering

Let’s find out the most recommended movie by plotting the matrix (Figure 18).

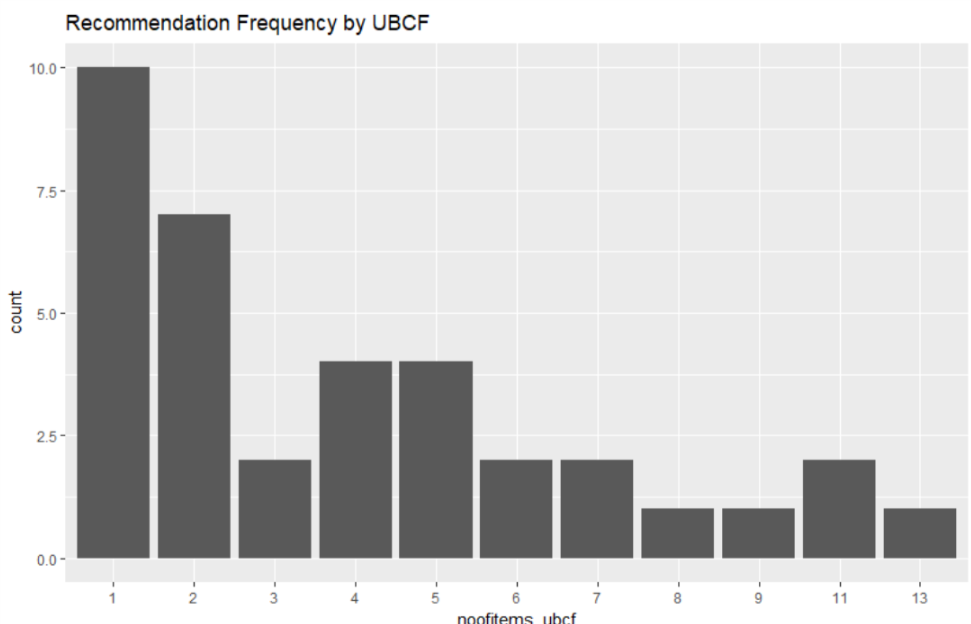


Figure : recommendation frequency of user-based Collaborative Filtering

Compared with the IBCF, the distribution has a longer tail. This means that there are some movies that are recommended much more often than the others. The maximum is 13, compared with 5 for IBCF. Top recommended movies are as following (Figure 19).

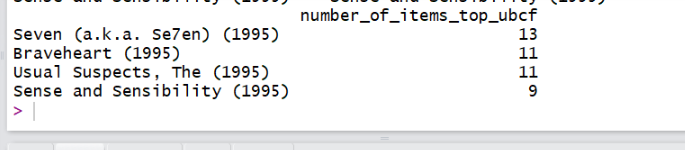


Figure : Top recommendation movies user-based Collaborative Filtering

## Evaluation

Now is the time to evaluate the ratings given by algorithm need to define the model to evaluate. Let's build it using the Recommender function and the matrix with the predicted ratings using the predict function (Figure 20).

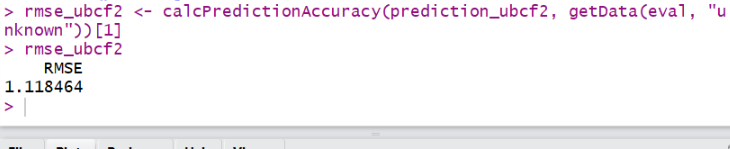


Figure : calculation of prediction accuracy

The Root mean square error of UBCF model is 1.11 which is less than IBCF model’ accuracy. Hence, we should opt of IBCF model of our Data.

REFERNCES

1. Building a Recommendation System with R , Suresh K .Gorakala, Michele Usuelli.

2.[David Goldberg , David Nichols , Brian M. Oki , Douglas Terry, Using collaborative filtering to weave an information tapestry, Communications of the ACM, v.35 n.12, p.61-70, Dec. 1992](https://dl.acm.org/citation.cfm?id=138867)