Movie Recommendation System: Movielens Database

Mahdi Karami

2023-01-23

1- Introduction

The present report aims to develop a movie recommendation system based on the MovieLens database of over 10M ratings given by numerous individuals to a large set of movies. A movie recommendation system is a technology that uses machine learning algorithms to suggest films to users based on their ratings and preferences. These systems analyze user behavior, preferences, and their rating history, as well as information about the movies themselves, to generate personalized recommendations. The goal of such a system is to help users discover new films they will enjoy and to make the process of finding something to watch more efficient by taking into account the users' ratings of movies. This allows the recommendation system to understand the user's taste and suggest similar movies accordingly. The linear model with regularization is implemented in R programming language to predict the users' ratings that would be given to the movies by considering the history of other ratings.

2- Data Wrangling & Preparation

The MovieLens rating database is located on their website (mentioned in the R script) which should be downloaded first by the R script. The downloaded zip-file contains two separate files, one for ratings ("ratings.dat") and another file representing the information about all the movies. The whole information are extracted and stored in a unique data frame called "movielens". The mentioned data frame is then divided into two separated groups, namely "edx" and "final_holdout_test" (by respectively portions of 90% to 10%), where the former is engaged to build the predicting model while the latter is meant for testing the predicting model and kept untouched until the model is finalized.

The "edx" dataset is consequently split into two further parts, namely "train_set" (90%) and "validation_set" (10%). They will be used for building models and optimize the hyper-parameters, respectively. This section is titled as "Part #1" in the provided R script.

3- Database Inspection

The given database is inspected in this section. It should be mentioned that we only investigate the "edx" set, i.e., the available dataset for building the model, as we are not allowed to touch the test set until finalizing the predicting model. In the other word, we assume that there is no information of the test set in this stage.

3-1- Overview of the MovieLens Database

The MovieLens dataset contains the history of over 10M ratings given by multiple users. The overall information of the generated datasets are shown in table below. There are 6 columns in the data frames indicating user's ID, movie ID, the given rating, time-stamp, movie title, and the movies genres.The time-stamp is an integer indicating the time-date that the rating has been given to the movie in terms of seconds since January 1, 1970.

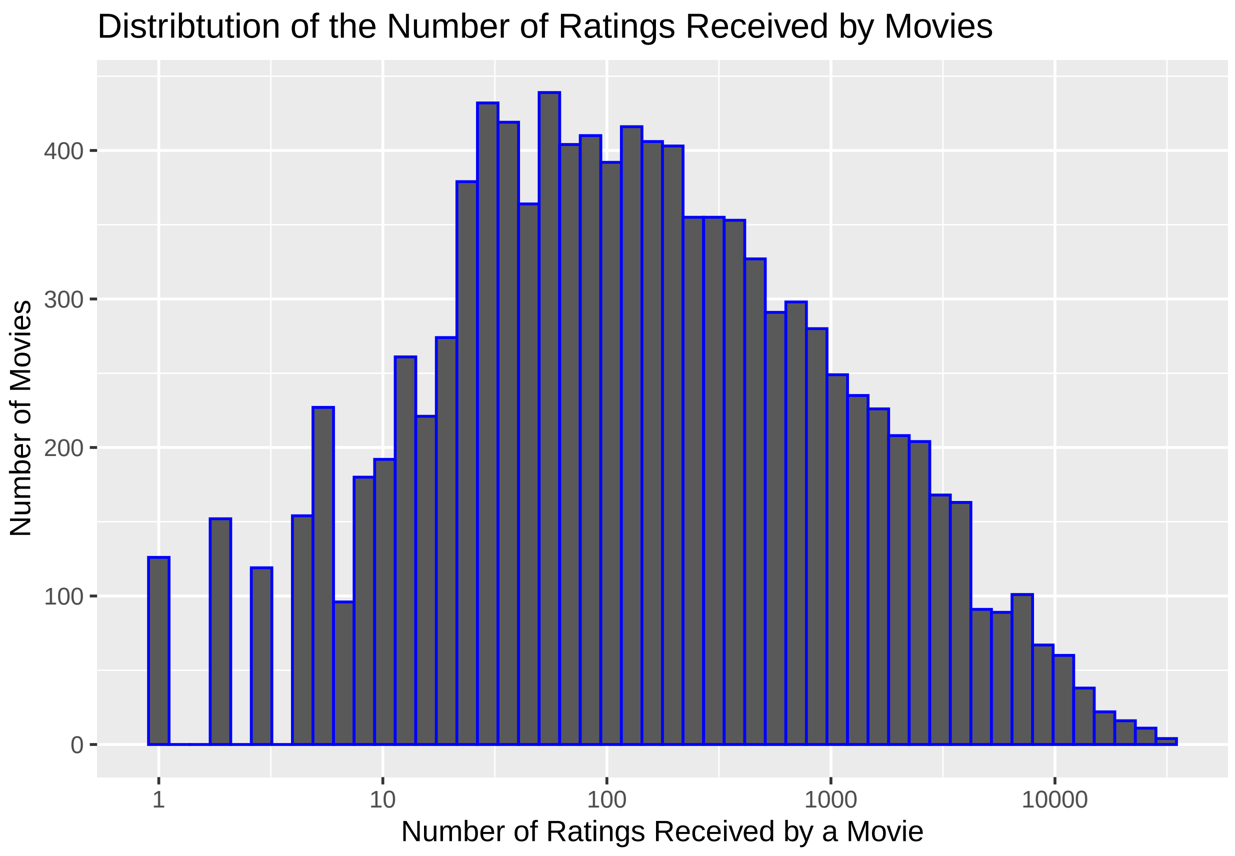
Table 1: Overview of Data Frames

Data Frame	No. Rows	No. Columns
Train Set	8100067	6
Validation Set	899988	6
Test Set	999999	6
Total (MovieLens)	10000054	6

3-2- Movies

There are totally 10,677 individual movies identified by unique Movie IDs in the "edx" database. The number of ratings each movie received varies considerably in the dataset, ranging from only 1 rating (for more than 100 movies) to 31,362 rating received by "Pulp Fiction" (movieID = 296). The average and median of the number of ratings for each movie are respectively ~843 and 122.

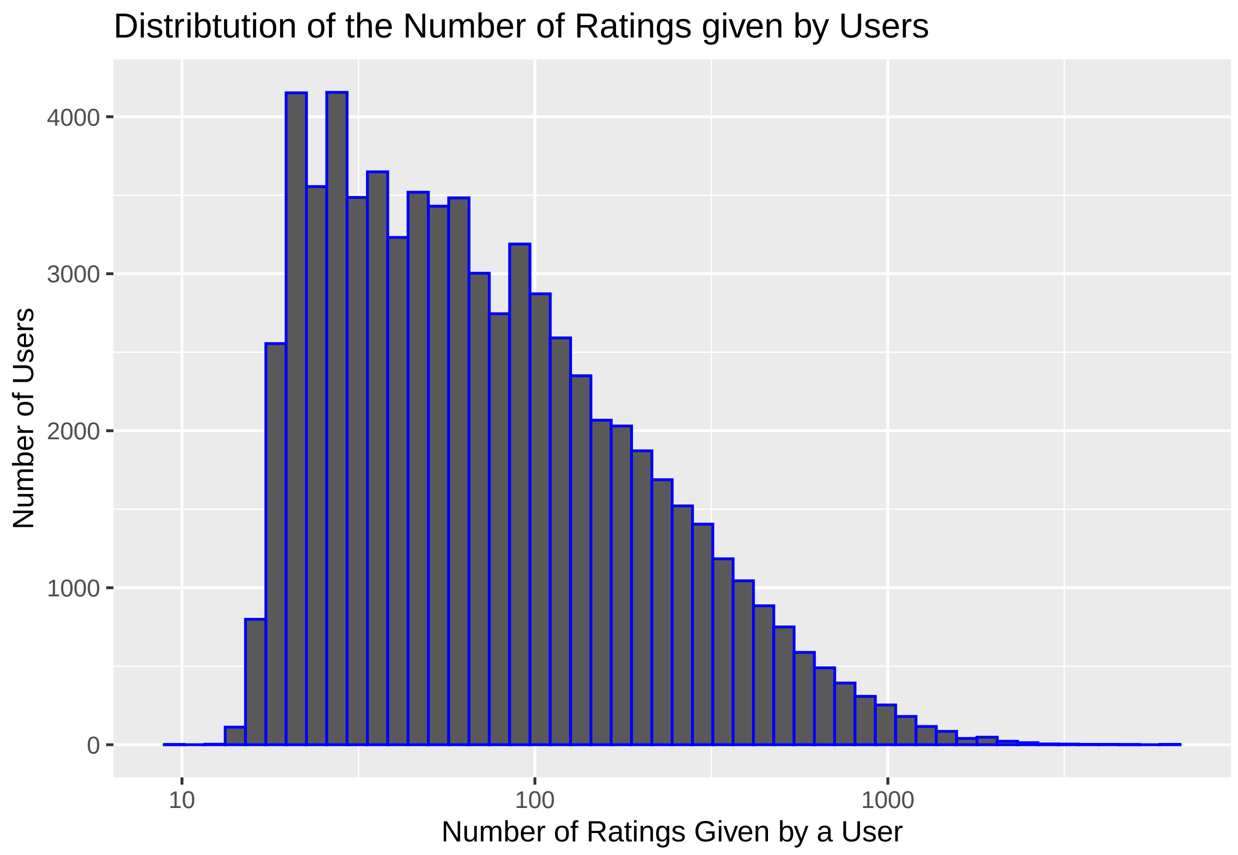
The distribution of received rating by each movie is depicted as a histogram in figure below where the horizontal axis is shown in log-scale. For instance, one can check the number of movies received only one rating (126).



3-3- Users

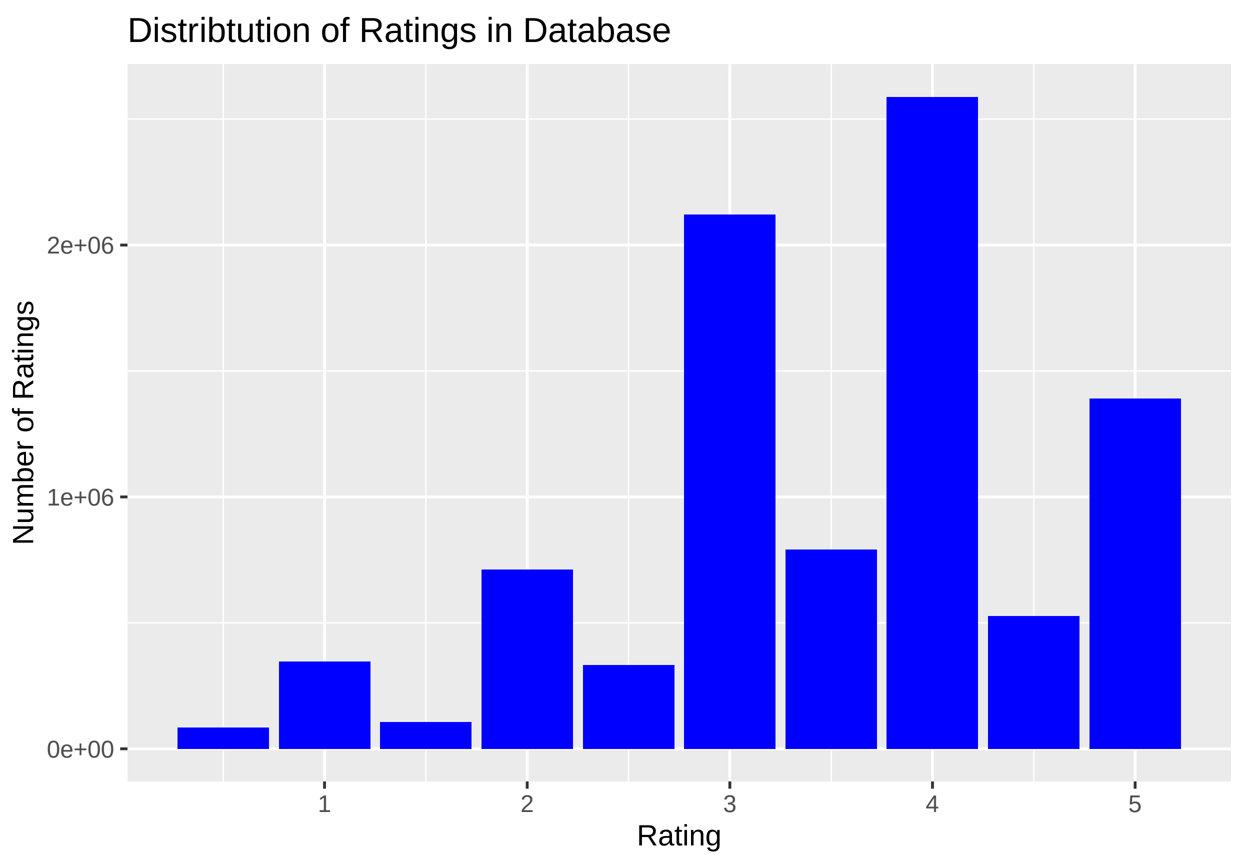
there are 69,878 unique users in the rating database with the minimum and maximum of respectively 10 and 6,616 number of ratings for each user. The average and median of the number of ratings given by each user are ~129 and 62, respectively.

the distribution of the number of ratings among the users is demonstrated in figure below.



3-4- Ratings

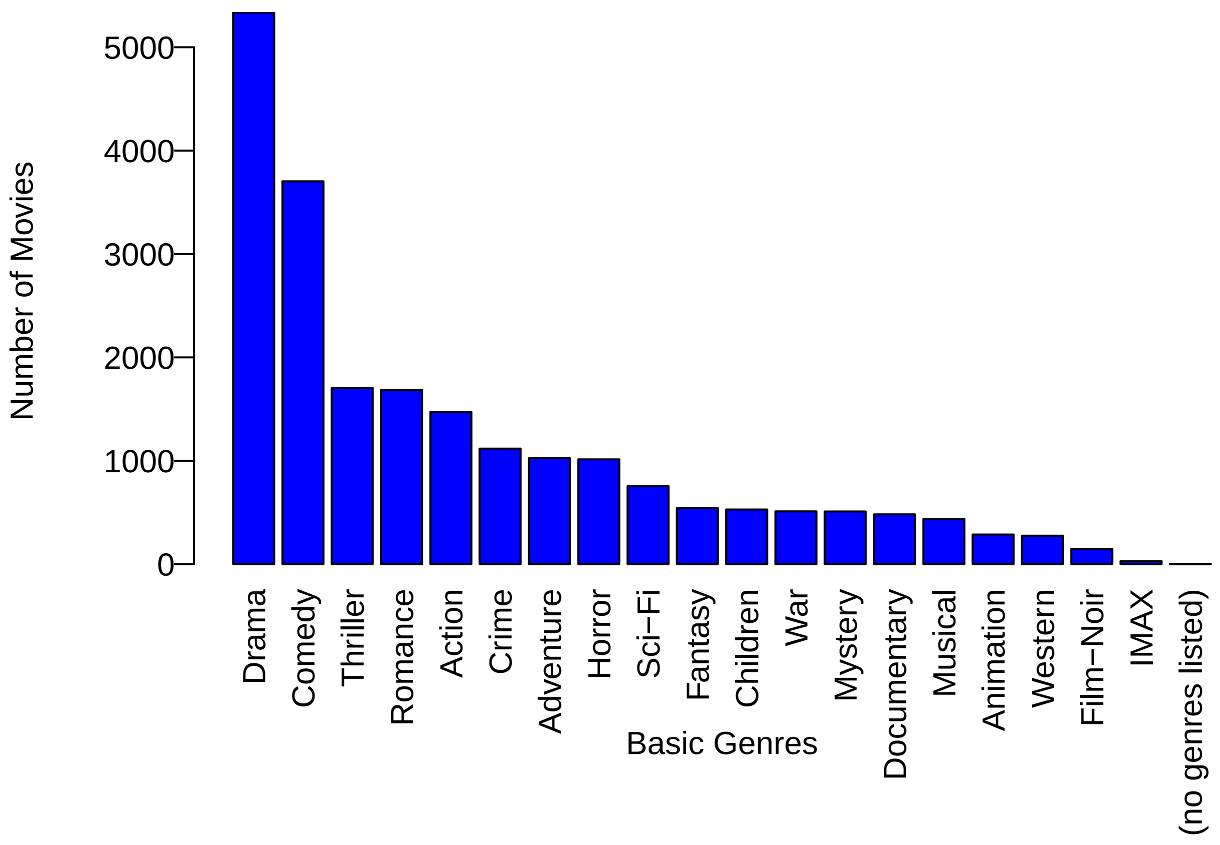
The ratings are discrete values ranging from 0.5 to 5 with the intervals of 0.5. The average, median, and standard deviation of the ratings in "edx" database are 3.51, 4, and 1.06, respectively. Figure below depicts the distribution of ratings given to all the movies. One can see users are more likely to give integer ratings (1,2,) comparing to half ones(0.5,1.5,. . .)!



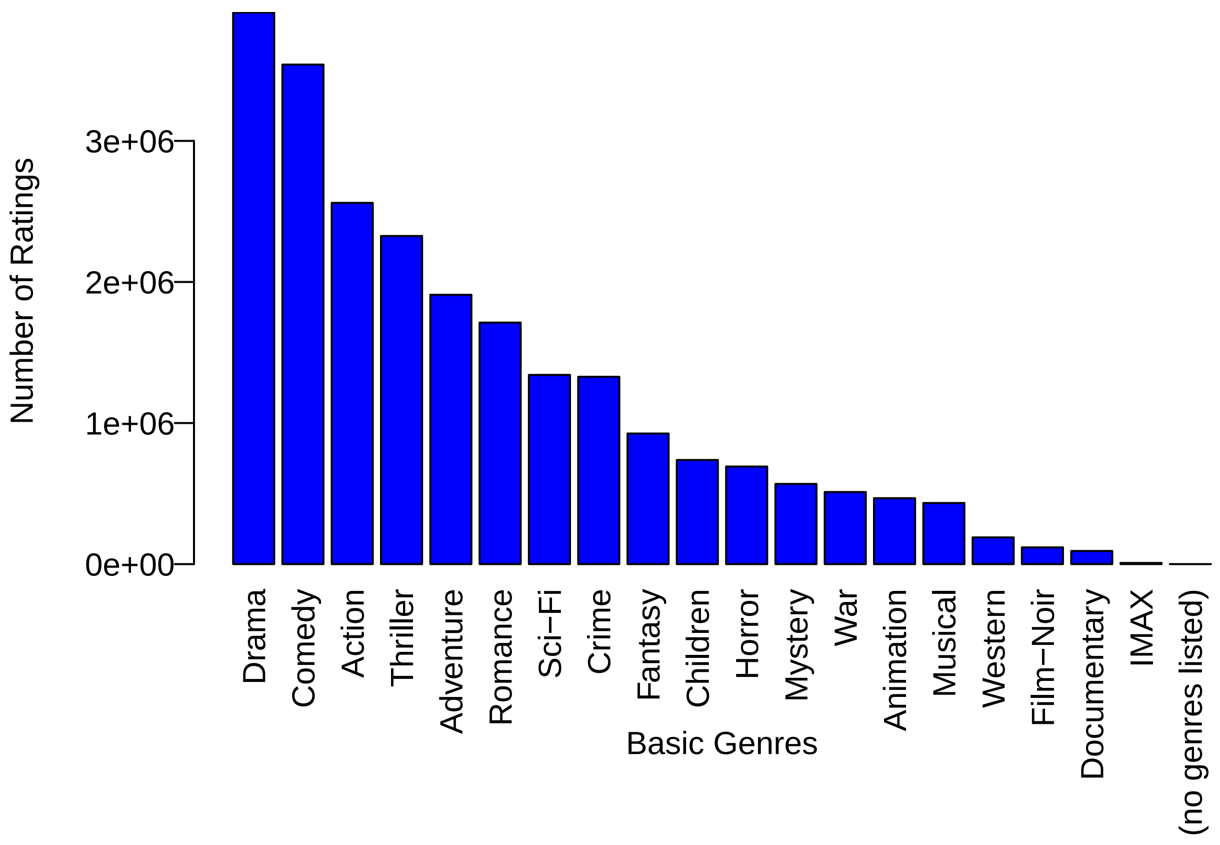
3-5- Genres

The genre attribute of the "edx" database is a text containing corresponding genres which are separated by "|" character. There are 19 basic genres whose combinations generate 797 different genres in the database. There are also 7 ratings given to "Pull My Daisy (1958)" which is a movie indicated as "(no genres listed)".

The figure below depicts the number of movies corresponding to each basic genres in the database. It should be mentioned that a movie can be found in multiple genres, hence the summation of movies in the plot exceeds the total number of movies in the database.



Another graph is shown below representing the number of ratings for each basic genres in the database. Similar to the above plot, each rating may be considered for multiple genres. One can see a similar pattern in two figures, despite of some differences. The most popular genre in the movies (and also in the ratings) is Dramma, which is followed by Comedy.



4- Methodology

The methodology of the proposed movie recommendation system is illustated in this section.

4-1- Linear Model

The linear estimation is considered for predicting users' ratings to the movies in the database. The following formula is assumed for calculating the ratings.

$$\hat{R}_{u,i} = \mu + B_i + B_u + B_g + \epsilon_{i,u,g}$$

where $\hat{R}_{u,i}$ indicates the estimated rating to movie i by user u and the average rating is indicated by μ. Due to the differences among movies, we apply a bias term (B_i) which implies the overall tendency of all the users toward movie i. The differences among individual movies are taken into account by bias effect, which indicates the tendency of all users to rate a special movie i differently from the others, which is shown in above formula by B_i. There is also another bias coefficient for each user, B_u, which models the differences in users' behavior when rating movies. The last bias term (B_g) indicates the effect of genres on the ratings. The error of estimation is considered by term $\epsilon_{i,u,g}$ which is assumed to have a normal distribution and will be minimized by linear model.considering $R_{u,i}$ as the actual ratings in the database, the terms of the linear model can be calculated by the following equations.

$$\mu = AVG(R)$$

$$B_i = \frac{1}{N_i} \sum_{u=1}^{N_i} (R_{u,i} - \mu)$$

where for each movie i, N_i indicates the number of ratings given by the users. The users' bias term can also be calculated as following.

$$B_u = \frac{1}{N_u} \sum_{u=1}^{N_u} (R_{u,i} - \mu - B_i)$$

where N_u demonstrates the number of ratings given by user u. Accordingly, the genre bias would be found by the following equation.

$$B_g = \frac{1}{N_t} \sum_{u=1}^{N_t} (R_{u,i} - \mu - B_i - B_u)$$

where N_t indicates the total number of ratings existing in the database.

4-2- Regularization

The regularization technique allow us to penalize data over-fitting of the training set by introducing the regularization parameter, λ to the linear model. The extra parameter(s) entered into the model should be optimized by neither training nor testing sets. Hence, a new set of data should be used to tune these hyper-parameters, where the validation set is taken into account. To make the problem done, we divide the whole dataset into 3 parts, namely train, validation, and test sets. the train set is used to build linear model on while we use the validation set for fine-tuning the hyper-parameters (here λ) to perform regularization and minimize the over-fitting issue. The test set will be used for evaluating the final model then. The following regularization equations are used in the current report although various formulations can be used for this purpose generally.

$$B_i = \frac{1}{N_i + \lambda} \sum_{u=1}^{N_i} (R_{u,i} - \mu)$$

$$B_u = \frac{1}{N_u + \lambda} \sum_{u=1}^{N_u} (R_{u,i} - \mu - B_i)$$

$$B_g = \frac{1}{N_t + \lambda} \sum_{u=1}^{N_t} (R_{u,i} - \mu - B_i - B_u)$$

4-3- Functions Description

For better implementation of the linear model, the script is developed in a procedural format. There are 5 functions in the R script (Part #3) which are described here.

A- Building the model

The linear model is built by $buildLinearModel(DB_{in}\ , \lambda)$ which takes a database and a regularization value to build a linear model on. The input database can be either the train set or "edx". The default set of the function is $\lambda = 0$ which indicates to regularization. The output of the function is a list containing μ, B_i, B_u , and B_g.

B- Predicting the user rating of a given dataset with a given model

The prediction of a given database with the given model is done by using *doPredictionWithModel(DB_{in}, $Model_{in}$, *option*) function. The format of the given model should be the same as the output of *buildLinear-Model*(DB_{in} , λ) function. The output would be a list containing predicted ratings with the same size as that of the input database. The *option* argument defines the level of details used for calculation, describe below. *option =1:* μ option =2: μ, B_i *option =3:* μ, B_i, B_m option =4,5: μ, B_i, B_m, B_g

The values of 4 and 5 for *option* result in the same results in this function. the latter value however indicates the case that regularization is taken into account.

C- Calculating RMSE for a predicted set of result

The root-mean-square error (RMSE) of a predicted rating list is calculated by comparing it with the actual rating using *calcRMSE*(R, $\hat{R}$) where the arguments indicate the actual and predicted ratings, respectively. Both input arguments should be entered with list/vector type. The output is a real number indicating the RMSE of the prediction.

D- Calculating RMSE for a given database with a given pre-built model

The function *calcPredictionError*(DB_{in}, $Model_{in}$, *option*) takes a given database and a pre-built model as inputs. It first calculates the predicted rating through *doPredictionWithModel*(DB_{in}, $Model_{in}$, *option*), and consequently calculates the RMSE of the predicted rating via *calcRMSE*(R, $\hat{R}$). The output will then be simply the RMSE of the prediction. The role of *option* was described in part B.

E- Calculating RMSE of the validation set for a given λ

The function *regularizationErr*(λ) wraps all the previous functions and performs the evaluation of the linear model for a given regularization factor. It takes λ as input and use it in conjunction with the train set to build the corresponding linear model. It then engages the model to predict the ratings for the validation set, and evaluate the RMSE for the predicted value. The output would be a single RMSE value for any given λ.

5- Results and Discussions

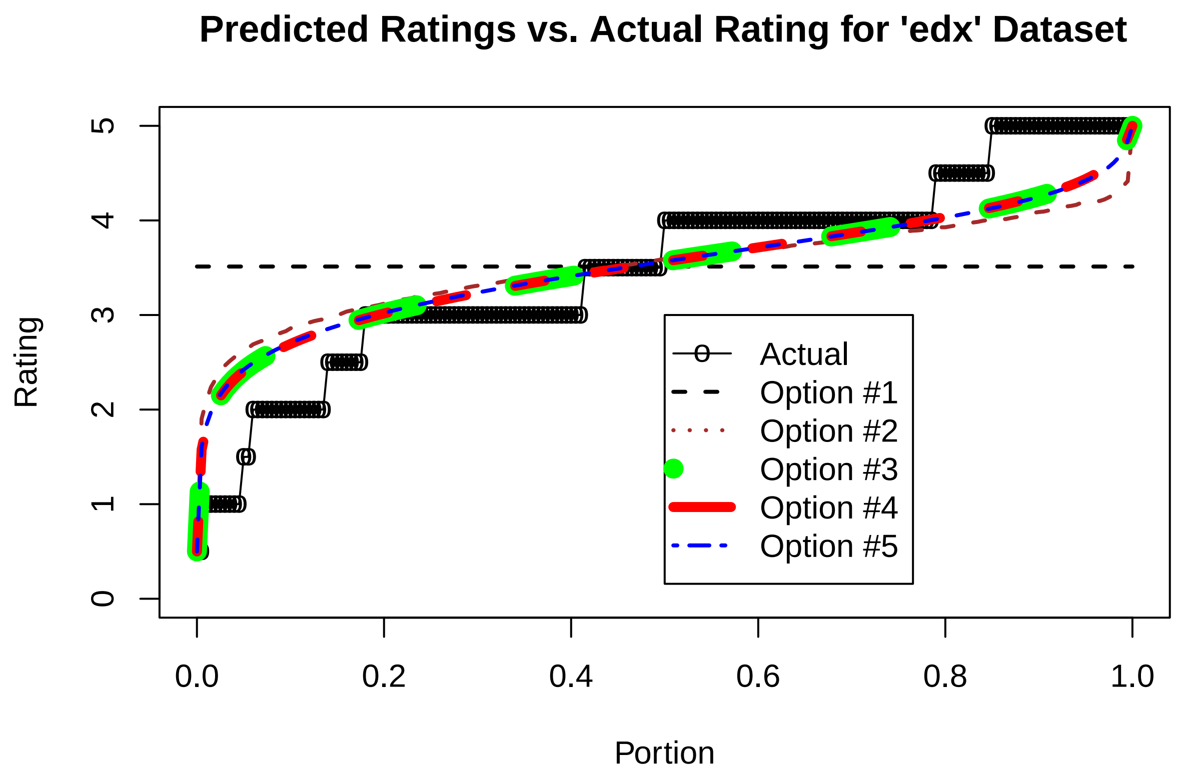
The results of the predicting linear model is presented in this section.

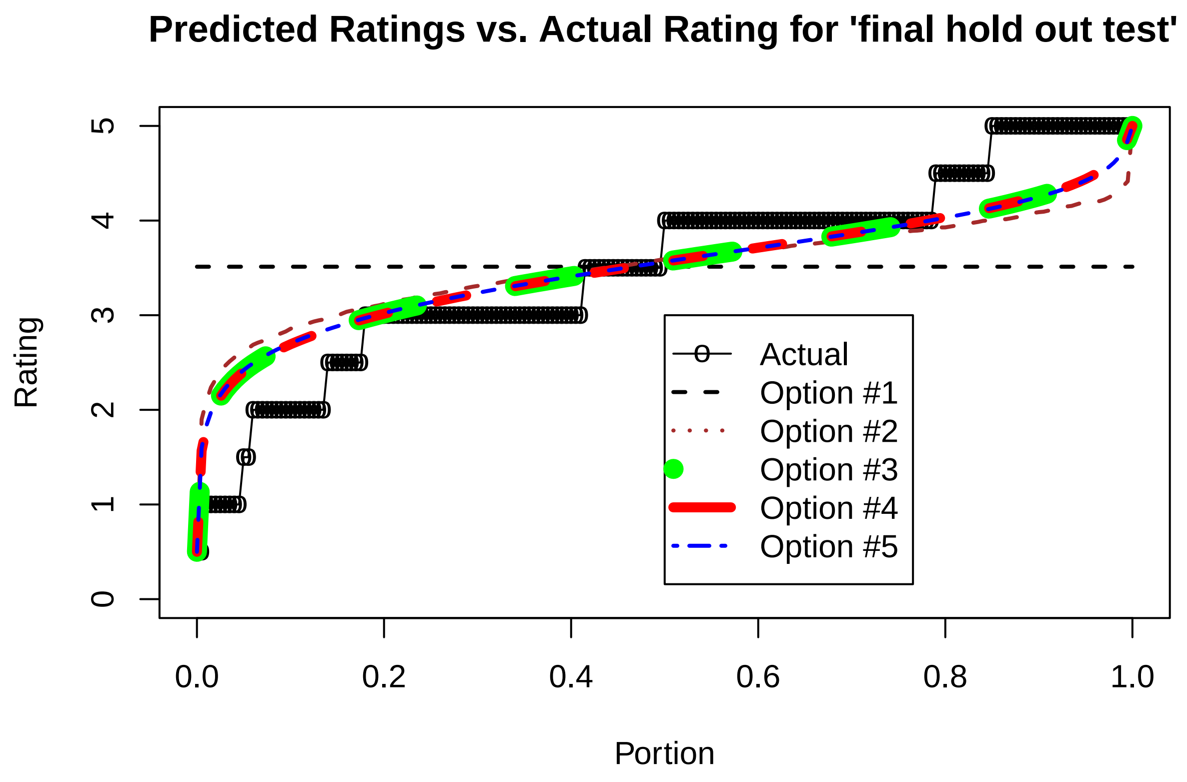
Table below indicates the details of the results of linear model for both 'edx' and the final hold out test (test_set) datasets. According to the given results, the RMSE of the final model for the test set is 0.86434, which is achieved by a regularization factor of $\lambda = 4.10247$.

Table 2: Results of Linear Model

Options	Description	RMSE of 'edx'	RMSE of 'Final Hold Out' set'
1	μ	1.0603313	1.0612018
2	μ, B_i	0.9423475	0.9439087
3	μ, B_i, B_m	0.8565338	0.8651613
4	μ, B_i, B_m, B_g	0.8561820	0.8647516
5	μ, B_i, B_m, B_g, λ	0.8564937	0.8643371

Figure below depict the quantiles of actual and predicted ratings for 'edx' and 'Final Hold out Test' sets, respectively. Different options of model prediction are also considered. One can see that the corresponding curves of both datasets almost have the same trend. In addition, the results of options #3 to #5 are almost the same.


Predicted Ratings vs. Actual Rating for 'edx' Dataset
Rating
Portion
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
Actual
Option #1
Option #2
Option #3
Option #4
Option #5



6- Conclusion

The movie recommendation system was developed based on the linear model estimation for MovieLens database, which contains the information of about 10M (10^7) ratings. Each row of the database includes the movie title and ID, user's ID, the given rating, the time that the rating has been given (in time-stamp format), and the corresponding genres of the movie. The database was originally split into two separate parts, one for building the predicting model ('edx') and another one for evaluating the final model on (i.e., test set: final hold out set), by the ratio of 90% and 10%, respectively. The 'edx' dataset was consequently divided into train (90%) and validation (10%) sets, where the former was used to build the linear model while the latter was engaged to fine-tune the hyper-parameter (i.e., λ, the regularization parameter). the 'edx' dataset contains the information of more than 10,000 movies which were revived by almost 70,000 unique individuals. The ratings were discrete numbers from 0.5 to 5 (with intervals of 0.5). There are 19 separate basic genres introduced for the movies that generate about 800 different combination of genres in the database. The linear model considered the average of ratings, in conjunction with the bias assumption for movies, users, and genres. The regularization factor was also taken into account to prevent the over-fitting issue. In order to illustrate the importance of multiple terms in the linear model, the results were calculated for different levels of details and both 'edx' and 'final hold out test' sets. For the test set, the best results was achieved for option #5, which included modeling of all parameters with regularization. The optimum value of the regularization factor was calculated as $\lambda = 4.10247$, which resulted in the RMSE of 0.86434 for test set. There are some limitations in the presented model that can be improved in the future works, including neglecting the standard deviations of movies and users, which are important factors in the modeling. In addition, the normalization of the raw database would help to develop a better linear model. Moreover, the users' tendency toward genres is a main factor that was neglected in the current report. It can be added up to the other parameter to develop a more accurate model with interpretable factors. finally, the regularization (which was limited to one parameter here) can be performed with multiple parameters, that probably result in a better prediction.

7- References

[1] https://grouplens.org/datasets/movielens/ [2] Kotkov et al., 2021] Kotkov, D., Maslov, A., and Neovius, M. (2021). Revisiting the tag relevance prediction problem. In Proceedings of the 44th International ACM SIGIR conference on Research and Development in Information Retrieval. https://doi.org/10.1145/3404835.3463019 [3] Vig et al., 2012] Vig, J., Sen, S., and Riedl, J. (2012). The tag genome: Encoding community knowledge to support novel interaction. ACM Trans. Interact. Intell. Syst., 2(3):13:1–13:44. https://doi.org/10.1145/23 62394.2362395 [4] https://rafalab.github.io/dsbook/ [5] https://statisticsbyjim.com/regression/interpret-constant-y-intercept-regression/