MovieLens Recommender Model

Scott Rushford

2024-12-02

Introduction

The purpose of this assignment is to develop a recommender model to predict how a user would rate a movie in the MovieLens data set for the purpose of recommending movies to users that were rated highly by similar users. ("MovieLens 10M Dataset" 2009)

The objective is to create a model with a root mean square error (RMSE) equal to or less than 0.86490.

Pre-processing of the data set resulted in the creation of two data frames: 1) edx - consisting of 9,000,055 observations and, 2) final_holdout_test - consisting of 999,999 observations. Both sets contain the following 6 variable: (Irizarry, n.d.)

1. User ID - a number identifying the user.
2. Movie ID - a number given to a movie for identification purposes.
3. Time Stamp - the date and time the movie was rated by the user.
4. Rating - the rating a user assigned to the movie on a scale of 0 - 5 (least to most favorable).
5. Title - movie title.
6. Genre - a column consisting of the names of the movie genre each movie falls under.

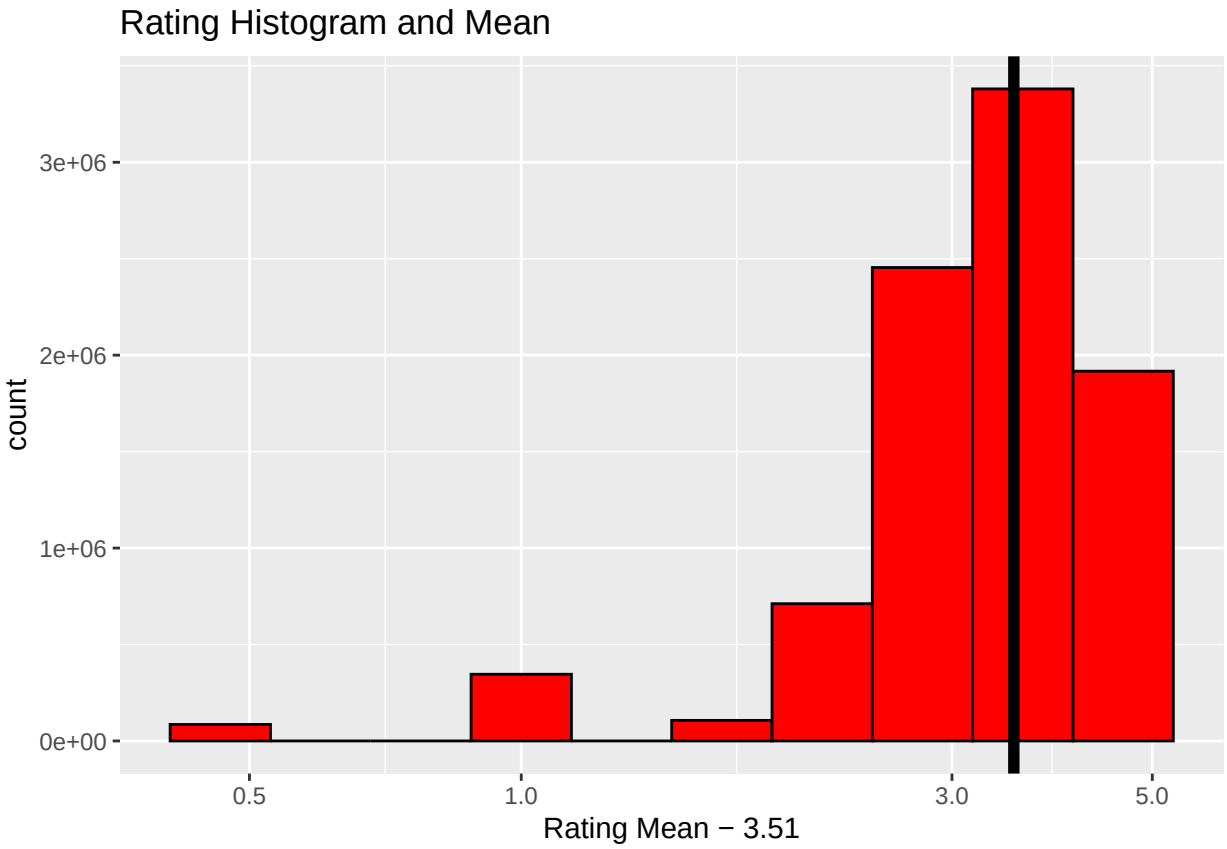
The key steps performed in this study include:

1. Summarize and visualize the edx data.
2. Conduct a correlation and principal component analysis.
3. Reduce the dimensions of the data set based on summary and analyses results.
4. Train and test various algorithms to find the best model.
5. Review results
6. Test the chosen model using the final_holdout_test set.

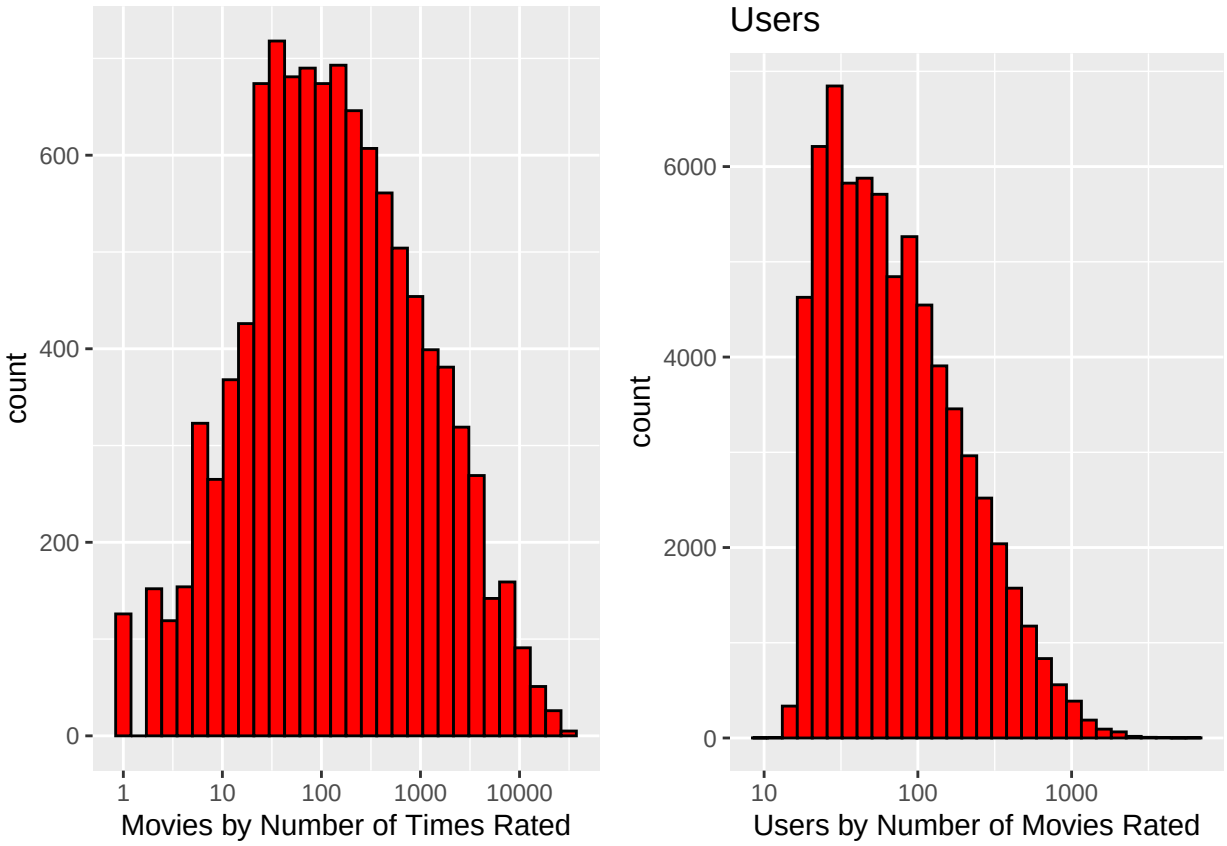
Methods and Analysis

The data summary indicates that 69,878 users rated 10,677 unique movies for a total of 9,000,055 ratings. The average rating is 3.51 out of 5. Each user rated on average 128.8 movies. Each movie was rated 843 times on average, with Pulp Fiction receiving the most reviews at 31,362.

Summary Data Visualization




```
## $title
## [1] "Movies"
##
## attr(,"class")
## [1] "labels"
```

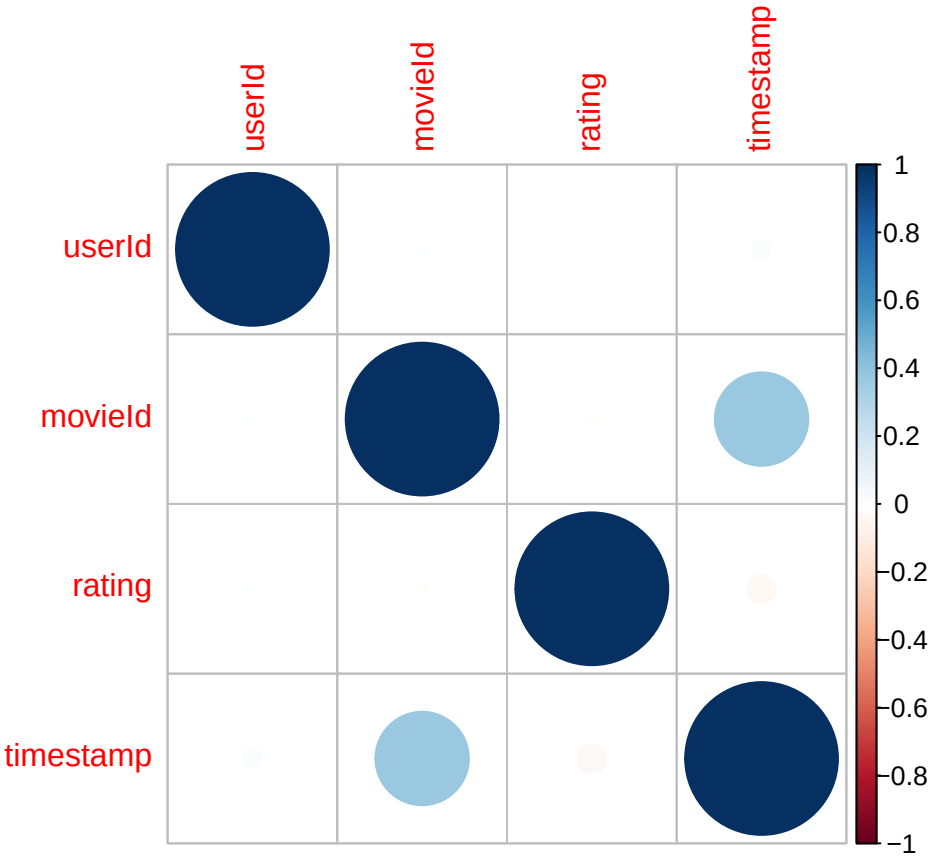


(Irizarry 2019)

Principle Component Analysis (PCA) and Correlation Analysis

To determine if the dimensions of the data could be reduced a principal component analysis and a correlation analysis was performed on the data.

```
## Importance of components:
##                           PC1    PC2    PC3    PC4
## Standard deviation     1.1733 1.0011 0.9981 0.7905
## Proportion of Variance 0.3442 0.2506 0.2490 0.1562
## Cumulative Proportion  0.3442 0.5947 0.8438 1.0000
```



The PCA analyzed movie ID, user ID, rating and time stamp and reveals none of these components explain more that 35% of the variance. Time stamp has a correlation with movie rating at 0.374. This is explained by the fact that the movie ID is assigned as the movie is released. It is expected that a movie will receive more ratings after it is released.

Time stamp is removed from the data set for this reason. Title is removed since it is a character string and is already represented by movie ID. Genre is also removed as it is also a character string and genre classification is subjective and incomplete.

Methods

The recommenderlab package (Hahsler 2023a) was developed for the purpose of creating and evaluating recommender models. This study will train and evaluate 2 collaborative filtering algorithms and 2 single value decomposition methods to determine which performs best on the MovieLens data.

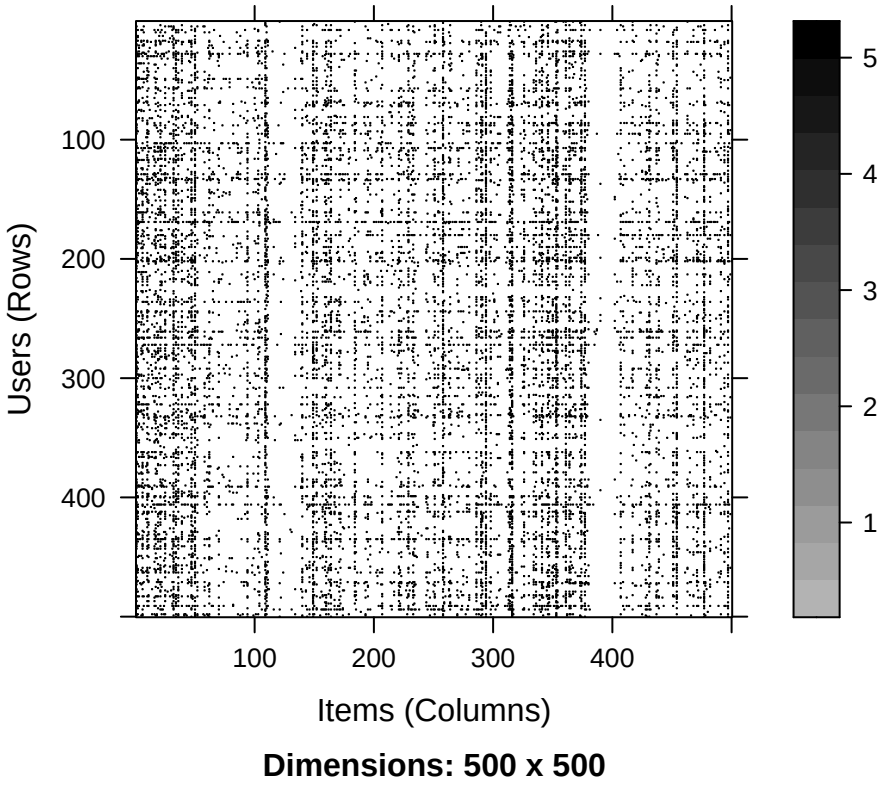
The recommenderlab package offers two types of data prediction, “ratings” and “topNList”. Type “ratings” will predict the rating a user would give to a certain movie. The “topNList” will recommend a list of a selected number (n) of movies to a particular user. The accuracy for the “topNList” type is measured as a percentage of True Positives divided by the total number of observations. Type “ratings” accuracy is measured as Root Mean Squared Error (RMSE). As the evaluation matric of the assignment is RMSE, the “ratings” type is used for each model.

In order to use recommenderlab all data must be in the form of a "realRatingMatrix". The edx and final_holdout_test data frames are coerced into a "realRatingMatrix".

Data Visualization - Sparsity

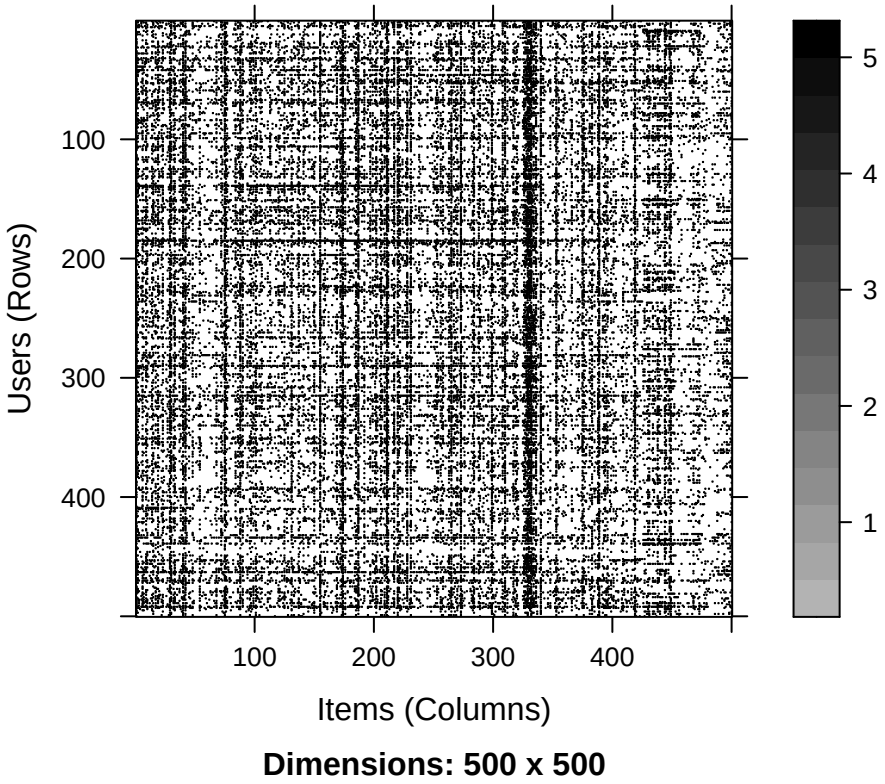
The data summary indicates that there is some sparsity in the data since not all users rated all movies. A visualization was created to examine the sparsity among the first 500 users and 500 movies.

The white areas represent the sparsity in the data as it indicates no ratings from that user exists for a particular movie.



The means obtained from the data summary are used to reduce the level of sparsity the in the data by including only those users that have rated at least 129 movies and those movies that were rated at least 843 times.

Once these adjustments were made, sparsity was visualized again.

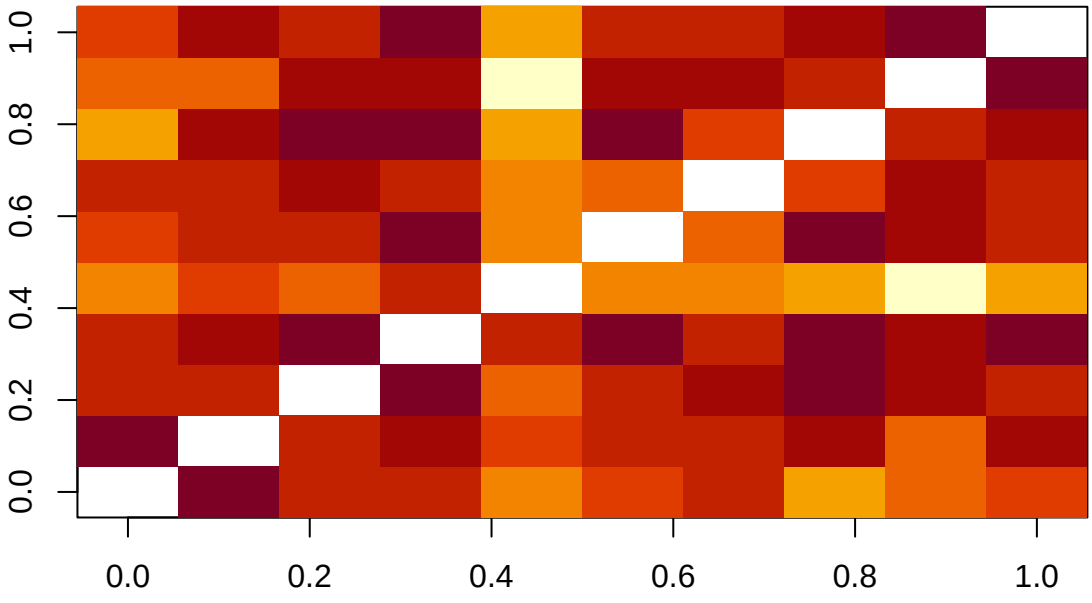


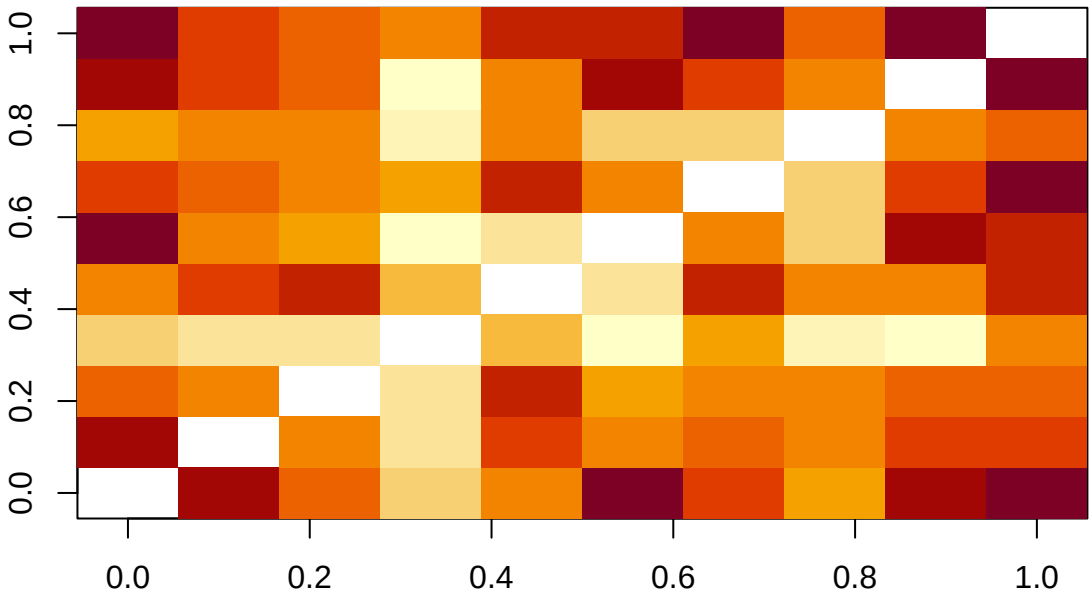
These adjustments reduced the size of the matrix to a:

```
22379 x 1994 rating matrix of class ‘realRatingMatrix’ with 5498215 ratings.
```

A look at the two graphs shows the reduction in the sparsity of the data, as there is far less white space in the second graph.

User and movie similarity matrices were created to how similar the first 10 users are to one another and how similar the first 10 movies are to one another. (73 2023)





Results from the Similarity Analysis indicates more similarity among users (top) than among movies (bottom).

Train, Test and Evaluate Models

There are 10 algorithms available in the recommenderlab package to create models plus a HYBRID option that "aggregates several recommendation strategies using weighted averages" Hahsler (2023a).

The full list of algorithms are available by loading the recommenderlab library and running:

recommenderRegistry$get_entries(dataType = "realRatingMatrix")

The first step in creating a Recommender Model is to determine the evaluation scheme to use. The evaluation scheme is used to determine how to split the data for training and evaluation. The key arguments for this function are the **method** (split, cross-validation, and bootstrap), the argument **given** - "single number of items given for evaluation or a vector of length of data giving the number of items given for each observation. Negative values implement all-but schemes. For example, `given = -1` means all-but-1 evaluation" and **goodRating** the "threshold at which ratings are considered good". Hahsler (n.d.)

The method chosen for the evaluation scheme in this study is cross-validation using the default value k = 10. This will split the data into 3 sets: a train set, a known set, and an unknown set. The known test data has the ratings specified by the argument given and the unknown has the remaining ratings, which is used to validate the predictions made using known." (Malshe 2019)

As the average rating is 3.51 any rating equal to or greater than 4 is considered good and is used for the argument goodRating.

The argument given was selected by going through the four experimental with holding protocols called Given 2, Given 5, Given 10 and All-but-1. Hahsler (n.d.)

The given value was finally selected at 25.

The train, known and unknown data was extracted from the evaluation scheme and assigned to their own real rating matrix:

- t_edx_r - train set 15273 x 1825 rating matrix with 4,288,752 ratings
- k_edx_r - known set 1704 x 1825 rating matrix with 41,903 ratings
- u_edx_r - unknown set 1704 x 1825 rating matrix with 431,871 ratings.

To establish a measurement baseline and to select a value for given, the "POPULAR", which recommends movies that are deemed popular based on ratings, algorithm was run on the data.

The algorithms chosen for this study are:

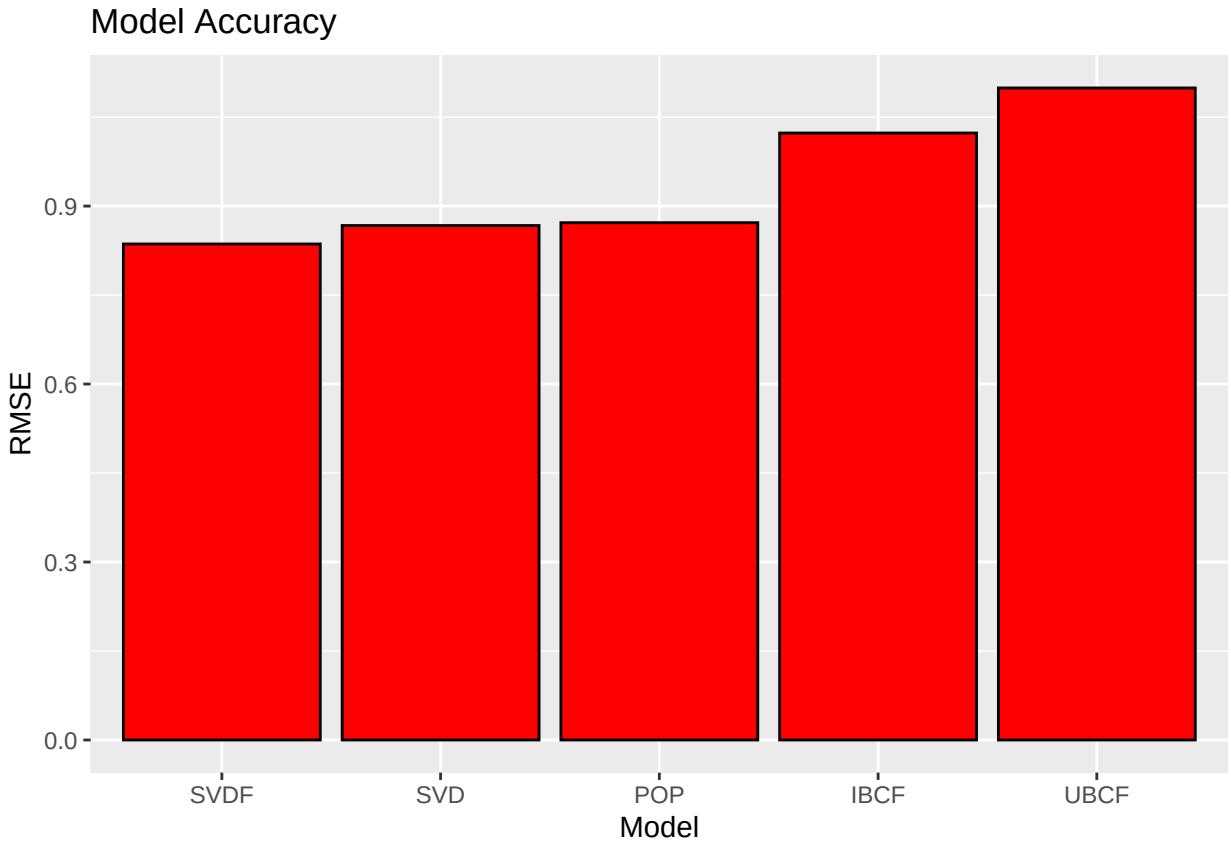
1. User Based Collaborative Filtering (UBCF)
2. Item Based Collaborative Filtering (IBCF)
3. Singular Value Decomposition (SVD) - based on SVD approximation with column mean imputation
4. Funk Singular Value Decomposition (SVDF) - based on Funk SVD with gradient descend.

(Hahsler 2023b)

Evaluate Models

Each model was evaluated and had its RMSE recorded and the results graphed.

```
##   Model      RMSE
## 1  IBCF 1.0230927
## 2  UBCF 1.0989982
## 3   POP 0.8719016
## 4   SVD 0.8671432
## 5  SVDF 0.8360466
```



Based on the evaluation data the Single Value Decomposition (SVD) models outperformed all other models. The SVDF model produced the lowest RMSE at 0.836, but takes a significant time to run. If time is a factor, SVD and POPULAR provided results within the range of the assignment. Picking from the POPULAR movies outperformed the Item Based and User Based Collaborative Filtering Models.

Final Holdout Test

A disadvantage of the recommender lab package is that a model can only predict or evaluate new data if that data is in a matrix of the same size as the model.

As the final_holdout_test data is of a much smaller size than the edx data a different method of testing was developed. This requires creating a separate final_holdout_test model using the same methods for the creation of the model from the edx data.

Data Summarization Final Holdout Test (fht)

Data from the final_holdout_test (fht) was summarized to determine how to set the evaluation scheme.

The fht data contain 68,534 individual users who rated 9,809 movies for a total number of ratings of 999,999. Like the edx data the average rating is 3.51. Each user rated on average 14.6 movies and the mean for the number of times a movie was rated is 102. Pulp Fiction

received the highest number of ratings at 3,502 which is consistent with the edx data. During the spliting of the data into the edx and fht sets a test was conducted to ensure that all users are represented in each set.

Data Preparation

In order to reduce the dimensions of the data time stamp, title and genres were removed from the fht set.

Sparsity was reduced using the mean value obtained in the data summary. Users who did not rate at least 15 movies and movies that did not get rated at least 102 times were removed.

The same arguments are used for the evaluation scheme of the fht data. The method is cross-validation with k = 10 and given set at 25 and good rating to 4.

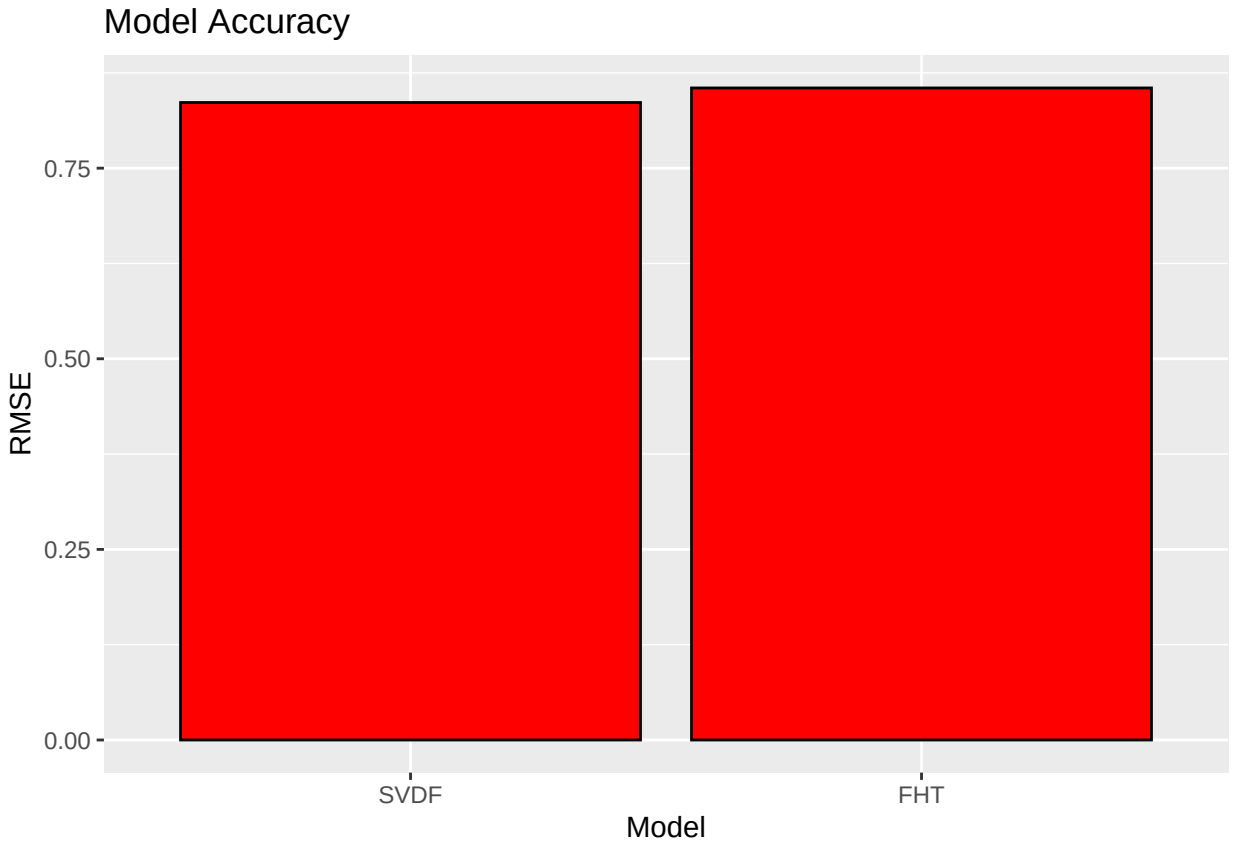
The train, known and unknown data was extracted from the evaluation scheme and assigned to their own real rating matrix:

- t_edx_r - train set 8190 x 2015 rating matrix with 363,528 ratings
- k_edx_r - known set 918 x 2015 rating matrix with 22,983 ratings
- u_ecx_r - unknown set 918 x 2015 rating matrix with 17,848 ratings.

FHT Evaluation

The evaluation of the trained fht model produced the following:

```
##   Model      RMSE
## 1   FHT 0.8551437
## 2  SVDF 0.8360466
```



The results on the final_holdout_test proves that by reducing the dimensions and sparsity of the data by the same proportions and by using the same evaluation scheme similar RMSE scores are obtained on the fht set (0.855) as the edx (0.836) set using the SVDF model.

Conclusion

The objective of this assignment is to create a Recommender Model that produced a RMSE of less than 0.86490 on the edx data set and be able to reproduce it on the final_holdout_test set. Using the recommenderlab package a model was trained on the edx data set that resulted in a RMSE of 0.836.

Model evaluation in recommenderlab is limited to a real rating Matrix of the same size. In order to use the final_holdout_test to evaluate the initial model, the same methods used to train and evaluate the edx data were reproduced to create a separate model on the final_holdout_test set. Once trained and evaluated this model had a RMSE of 0.855, thereby obtaining the target RMSE.

```
## [1] "R version 4.3.2 (2023-10-31 ucrt)"
```

References

,

73, the outlier. 2023. "How to Build a Recommendation Engine in r Using MovieLens Data|cold Start, Long Tail & Data Sparsity," January. https://www.youtube.com/watch?v=Rq07RtDxGh8.

Hahsler, Michael. 2023a. "Recommenderlab: Lab for Developing and Testing Recommender Algorithms." https://CRAN.R-project.org/package=recommenderlab.

———. 2023b. "Recommenderlab: Lab for Developing and Testing Recommender Algorithms." https://CRAN.R-project.org/package=recommenderlab.

———. n.d. "Recommenderlab: An R Framework for Developing and Testing Recommendation Algorithms."

Irizarry, Rafael A. 2019. *Advanced Data Science*. Harvard. https://rafalab.dfci.harvard.edu/dsbook-part-2/.

———. n.d. "MovieLens - Capstone Project: All Learners | Data Science: Capstone | edX." https://learning.edx.org/course/course-v1:HarvardX+PH125.9x+1T2024/block-v1:HarvardX+PH125.9x+1T2024+type@sequential+block@e8800e37aa444297a3a2f35bf84ce452/block-v1:HarvardX+PH125.9x+1T2024+type@vertical+block@e9abcdd945b1416098a15fc95807b5db.

Malshe, Ashwin. 2019. "Collaborative Filtering." In. https://ashgreat.github.io/analyticsAppBook/index.html.

"MovieLens 10M Dataset." 2009, January. https://grouplens.org/datasets/movielens/10m/.