MovieLens

Greg

2020-08-19

INTRODUCTION AND PROJECT OVERVIEW

DATA SOURCE

The data set is sourced from **GroupLens** project that was originally built as a research lab to recommend news by collecting ratings from news readers. The ratings were used to predict whether other readers would like an article before they read it. Then, it was extended to other products such as books and movies, referred to as *BooksLens* and *MovieLens* respectively. Among many other accomplishments, the GroupLens research team was renowned for releasing several data sets during the early years in the field of recommendation systems, when data sets were not easily available for benchmarking. Consequently, their publicly available set with 10M ratings is utilized for this project. While there exists a larger 100M dataset, 10M is sufficient for the project at hand while allowing relative ease of computation on any machine with 16GB of RAM.

PROJECT OVERVIEW AND GOALS

We will attempt to construct a movie recommendation system using the MovieLens dataset as described. Using 90/10 partition, the data is split into training (edx) and validation sets. The final model Root Mean Squared Error (RMSE) from the true ratings from the validation set is returned. As the validation is "set aside" and can not be used for training the algorithm, it will only be used for evaluating the RMSE of the final model.

The RMSE is a measure of the differences between values predicted by a model and the values observed (i.e., model accuracy), and is calculated as follows:

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

Similar to other R-squared like measures such as Mean Absolute (MAE) or Mean Bias (MBE) Errors, RMSE is a *negatively-oriented* metric, i.e., the lower values are better. However, due to squared errors, RMSE tends to give more weight to large errors relative to the others and therefore, is more sensitive to outliers. The RMSE is always larger or equal to MAE for a sample size n as in:

$$MAE \leq RMSE \leq \sqrt{n}MAE$$

As RMSE is a common metric used in the the famous Netflix challenge, there is no need to consider other measures for purposes of this project.

Table 1: Sample Data: First Look

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	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

Table 2: Sample Data: extracted realease year and year rated

userId	movieId	rating	title	released	genres	year_rated
1	122	5	Boomerang	1992	Comedy|Romance	1996
1	185	5	Net, The	1995	Action|Crime|Thriller	1996
1	292	5	Outbreak	1995	Action|Drama|Sci-Fi|Thriller	1996
1	316	5	Stargate	1994	Action|Adventure|Sci-Fi	1996
1	329	5	Star Trek: Generations	1994	Action|Adventure|Drama|Sci-Fi	1996
1	355	5	Flintstones, The	1994	Children|Comedy|Fantasy	1996

METHODS AND ANALYSIS

EXTRACT-TRANSFORM-LOAD & DATA CLEANING

The first 6 rows of the initial training data set, **edx**, as extracted are displayed in Table 1.

It has 9,000,055 rows and 6 columns as follows: *userId, movieId, rating, timestamp, title, genres*. It appears that *timestamp* is too precise of a feature and we won't be able to explore the full effects of seasonality on the ratings, i.e., whether some users are rating higher in spring/summer, for instance.

However, we can further improve the raw data by extracting the year of the rating from *timestamp* as well as the release year from *title* and potentially investigating the lag between the two. For example, would the new release skew the ratings away from the mean, more positively for "good" and negatively for "bad" movies? The year could be extracted from *title* with this regular expression. Please contrast Table 2 to Table 1 for the transformed data appearance (new *released* and *year_rated* columns).

Also, we note that the original edx data set is in the form of a *long* data format with each row correspond to a single user/single movie which is not really suitable for our analysis and will be converted to a *wide* (rating matrix) format with users in rows and movies in columns.

DATA EXPLORATION & VISUALIZATION

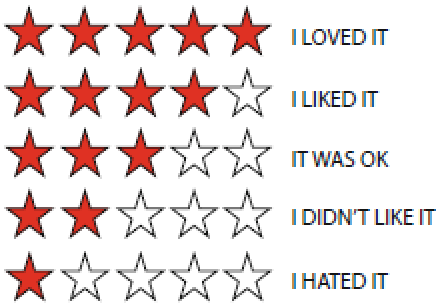


Figure 1: 5-point interval ratings

As we are predicting ratings, it can be seen that ratings are **discrete** with half point intervals, ranging from 0.5 to 5 with 4 being the most common. Also, the half star are given a lot less frequently than whole star ratings, as evident from Figure 2 (red vs blue).

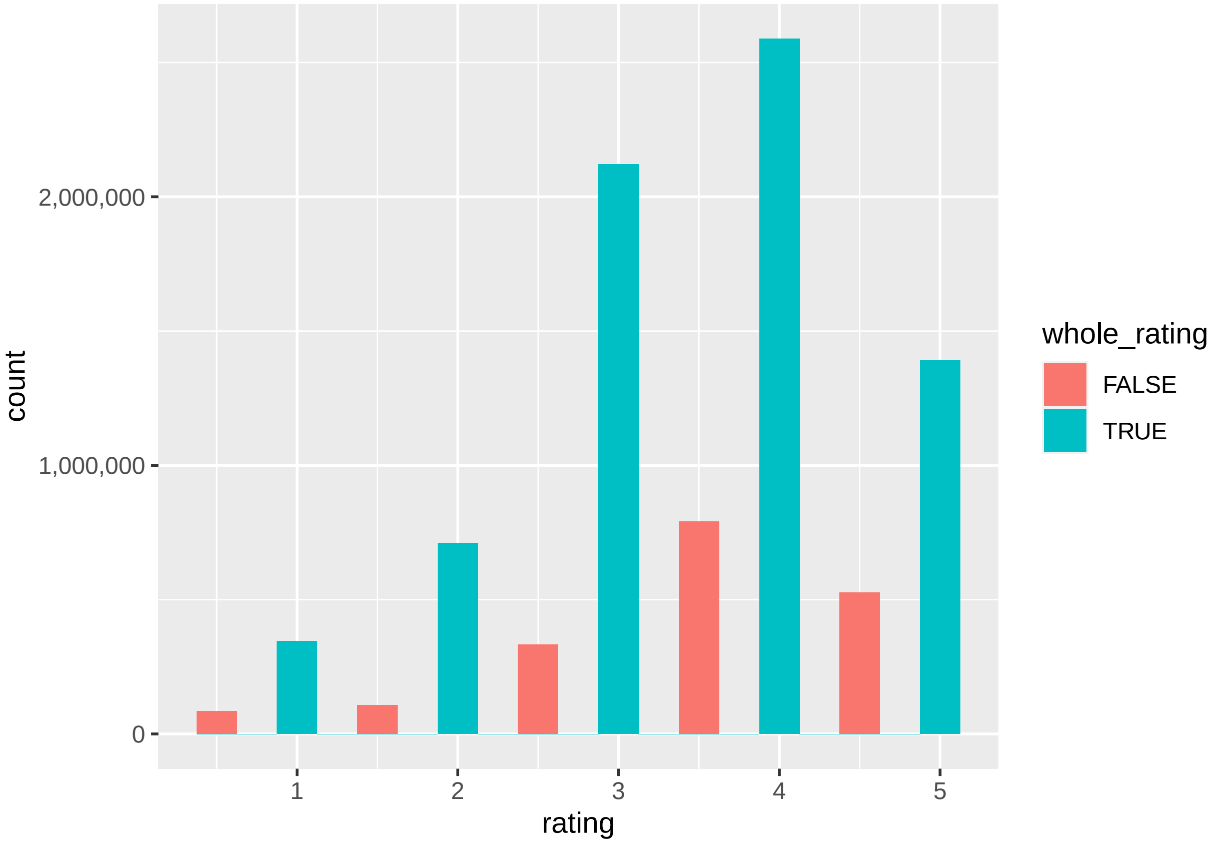


Figure 2: Frequency Plot, Movie Ratings

We can observe the frequency plots in Figure being highly skewed. They show that only a relatively small portion of users are active (high frequency ratings) and, likewise, a small portion of movies are popular. We note that after about 6,000 most active users and 500 most popular movies the counts tail off. Because of the rarity of observed ratings in the long tail it is generally more difficult to provide robust rating predictions for tail users or movies. In fact, many recommendation algorithms have a tendency to suggest popular rather than infrequent movies. This also has a negative impact on diversity as users may often become bored by receiving the same set of recommendations of popular movies.

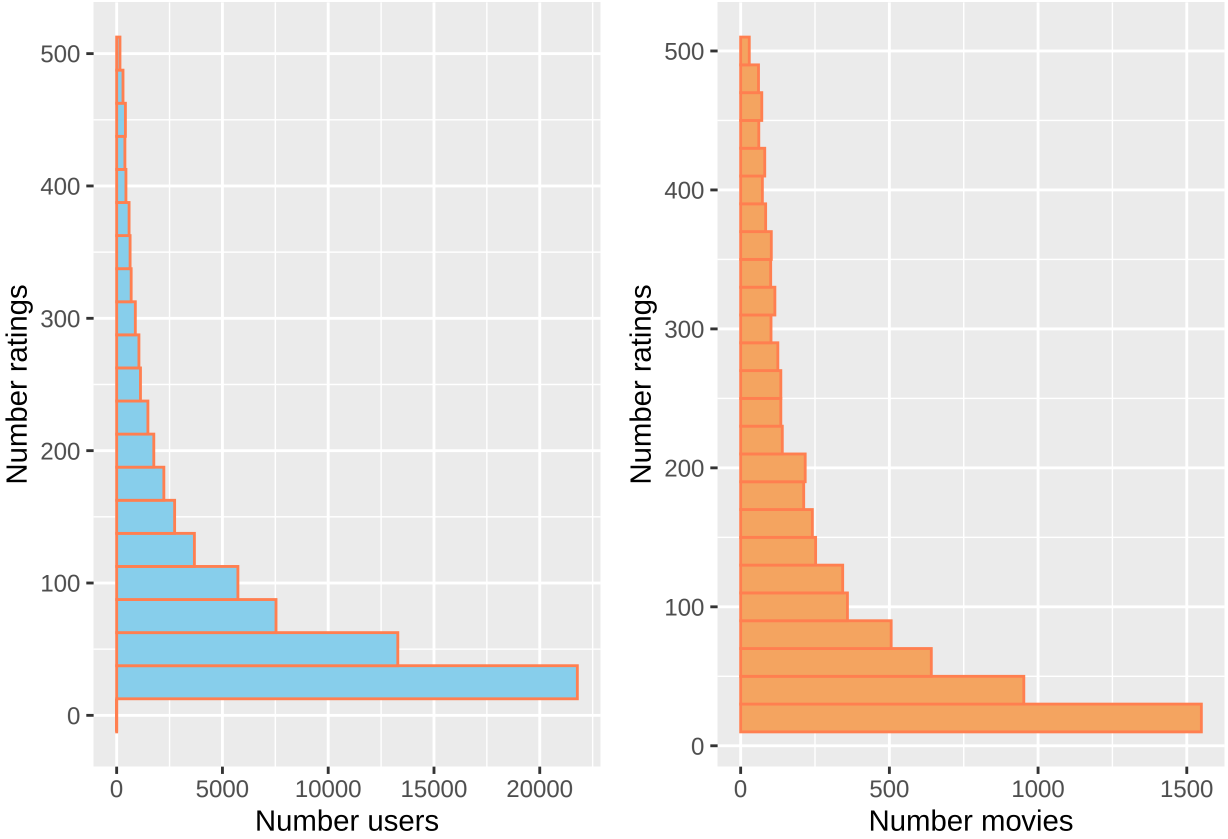


Figure 3: Long Tailed Plots: Active vs Tail Users and Popular vs. Unrecognized Movies

For the training set under consideration, there are 10,677 different movies and 69,878 users or raters with "Pulp Fiction" being the most rated movie and five most given ratings in decreasing order are 4, 3, 5, 3.5, and 2. [Source: Quiz]

The data could be viewed as an $n \times m$ rating matrix with n rows for rating users and m columns for movies. While there are 69,878 rows and 10,677 columns which is a matrix of 746,087,406 entries or cells, only 1% of overall movies are rated. As there are almost 7 times less movies than users, movie based algorithm would probably be less computationally prohibitive to utilize from the computational power standpoint.

The fact that our rating matrix is 99% empty or *extremely* sparse in other words, may lead to additional bias. This is important as essentially we are trying to perform a missing value estimation. Additionally, some well-known methods such as kNN neighborhood-based interpolation may not work as well on a very sparse dataset.

In general per (Aggarwal 2016), the methods for such data as ours are called **collaborative filtering** methods. Collaborative filtering methods use the collaborative power of the ratings provided by multiple users to make recommendations. Most users would have viewed only a small fraction of the large universe of available movies. As a result, most of the ratings are unspecified. The specified ratings are also referred to as **observed** ratings. With this data, we are dealing with a matrix with a lot of **unobserved** ratings. To illustrate, consider the below (transposed) matrix representing 10 most active users ratings for 10 most rated movies compared to a sample of random user ratings for a sample of random movies:

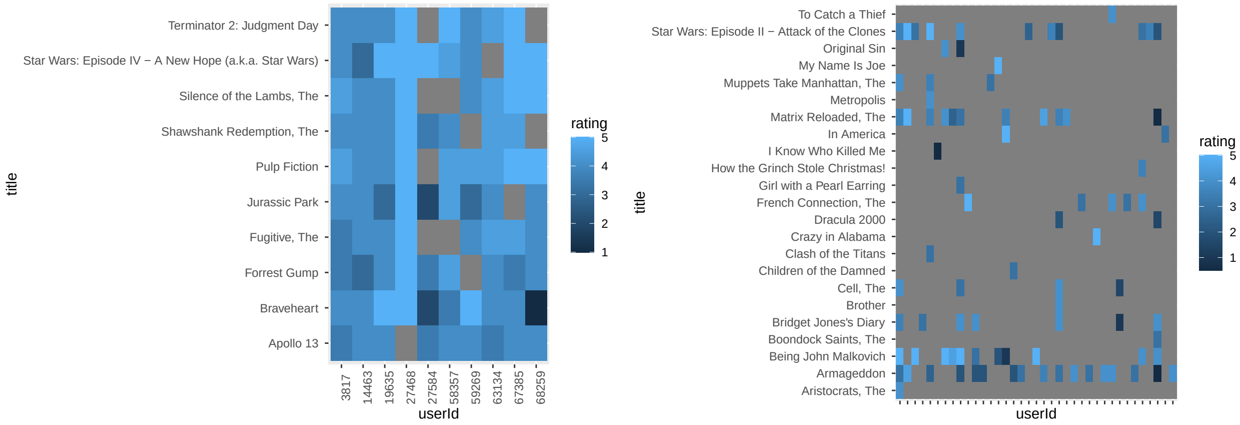


Figure 4: top 10 users rate 10 top movies vs. random users rate random movies

Even for the top ten, there are movies that were not rated! For instance, *The Fugitive* was not rated by users *‘27584’* and *‘58357’*, per Figure 4. The *observed* ratings represent explicit interval ratings from .5 to 5.

Important characteristics of ratings, such as sparsity and the long tail, need to be taken into account during the recommendation process. The main challenge in designing collaborative filtering methods is that the underlying ratings matrices are sparse. By adjusting the recommendation algorithms to take such properties into account, it is possible to obtain more meaningful predictions.

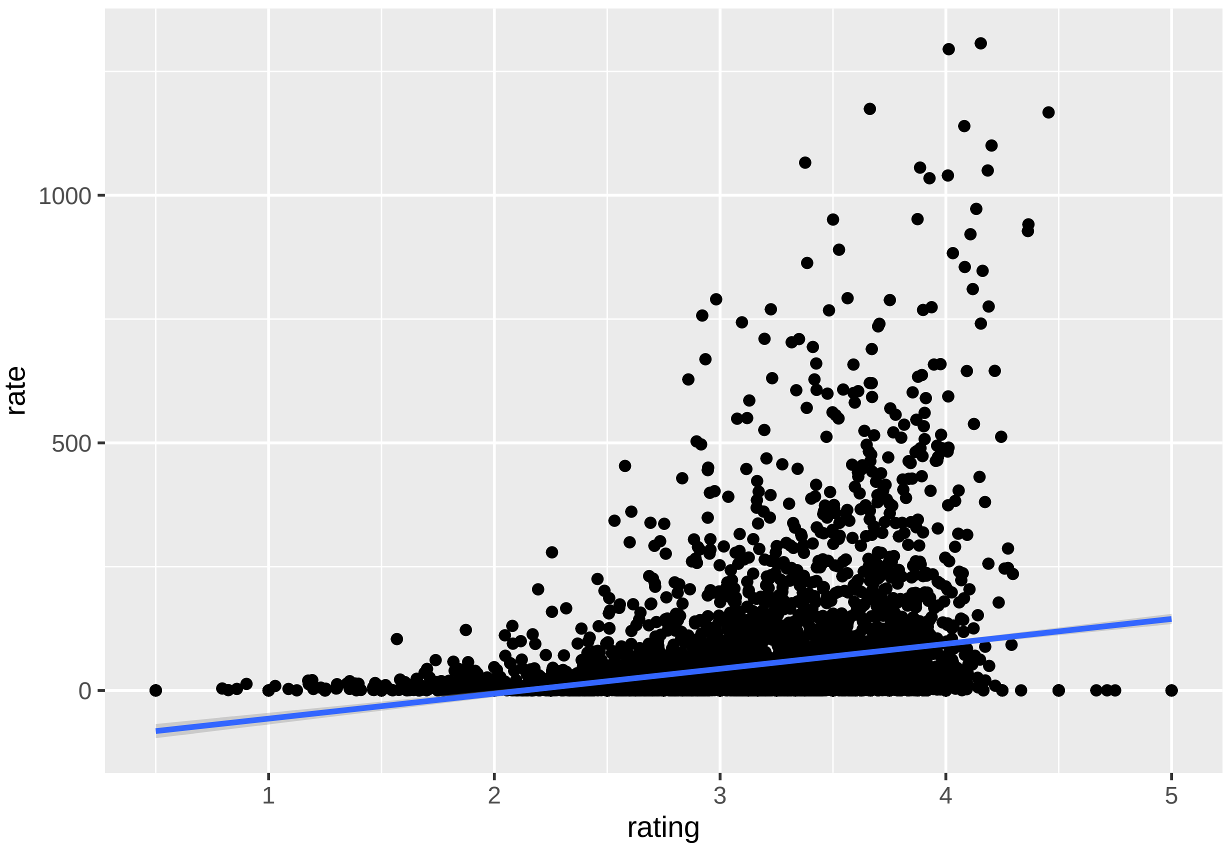


Figure 5: Average ratings by frequency

The plot in Figure 5 explores the premise that the top movies have above average ratings. We stratify the post-1993 movies by ratings per year and compute their average ratings. To calculate number of ratings per year, the year 2018 was used as the end year. When the rate (average rating) plotted versus ratings per year (rate in Figure 5) with an estimated trend, the trend does support the above premise.

The data also has a *genres* feature. It includes every genre that applies to the movie as some movies fall under several genres. Let's define a category as any combination than could appear but only categories with more than 1,000 ratings. Then we will compute the average and standard error for each category. The averages by genre plotted as error bar plots show that genre does have a significant effect, per Figure 6.

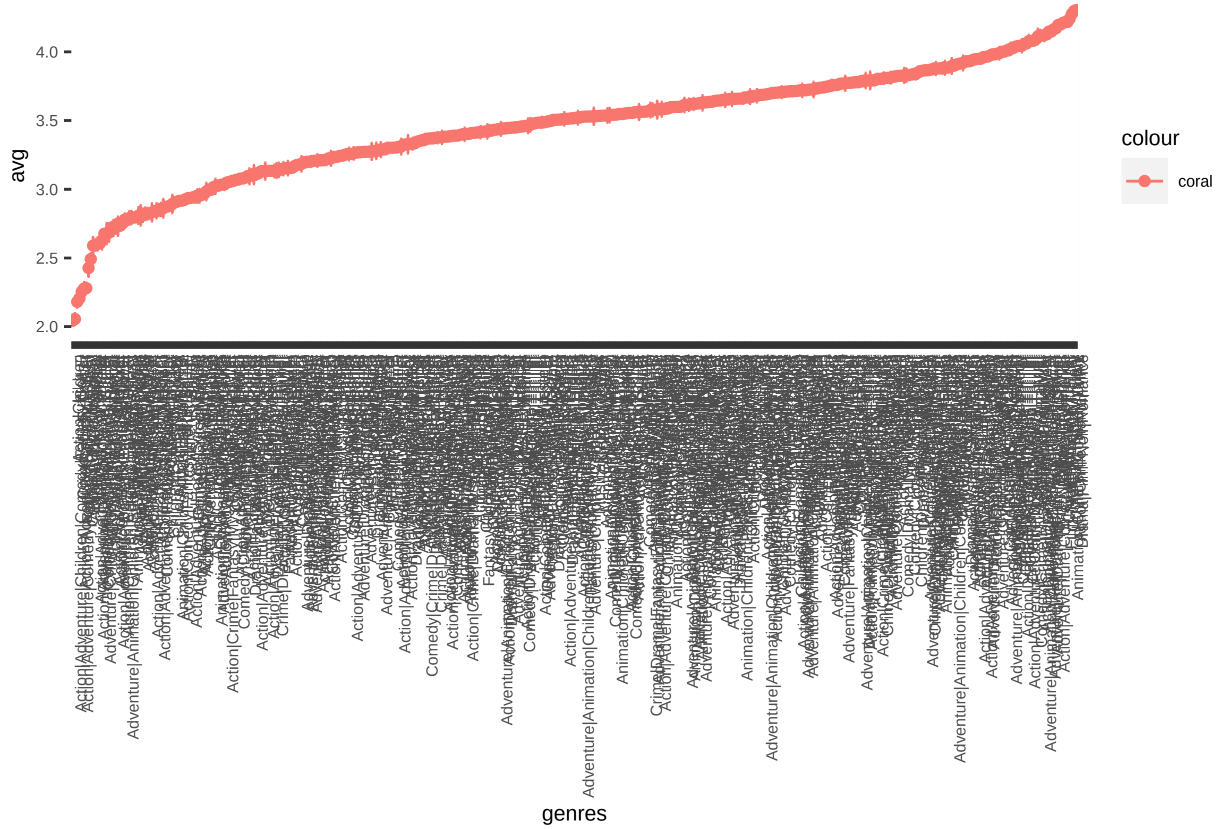


Figure 6: Average ratings by genre

ANALYSIS: BASELINE MODEL

"The secret to getting ahead is getting started." –Mark Twain

As we need to start somewhere anyway, let's just use mean as the predictor for our first model. However, before any modeling, the *edx* set has to be split into train and test sub sets using the same 10% partition as discussed before. The *edx_test* set will be used to asses RMSE for our trained model to select the best final model with lowest RMSE. That model's RSME will be cross checked on the validation set at the project end. The Root Mean Squared Error > 1 indicates an error greater than 1 star, i.e., model's predicted rating of more than 1 star off the actual is not an acceptable prediction. In this project, we'll target an RMSE < 0.8649 as acceptable.

The base model predicts the same rating for all movies by all users (i.e., calculating mean rating for entire dataset):

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

The $\epsilon_{u,i}$ is an independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies.

Table 3: RMSE: baseline model

model	RMSE	Improvement vs Target
Mean	1.060054	-0.1951537

RMSE result for our *starter* model is 1.0601 which is greater than 1 and is obviously unacceptable indicating an error of greater than 1 star, but we got a baseline at the very least!

ANALYSIS: BASELINE VARIATIONS - MOVIE EFFECTS

"When you're in something as successful as 'Transformers,' you can't use it as a sales piece for your ability as an actress because it's all about the special effects." –Megan Fox

Collaborative Filtering methods try to capture the interactions between users and movies that produce the different rating values. However, much of the observed rating values are due to effects associated with either users or movies, independently of their interaction. Our data exhibit large user and movie biases or effects; i.e., systematic tendencies for some users to give higher ratings than others, and for some movies to receive higher ratings than others. We will encapsulate those effects, which do not involve user-item interaction, within the baseline predictors (also known as biases). Assuming that more popular movies get rated higher, movie bias is calculated as follows:

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

where b_i is a movie effects.

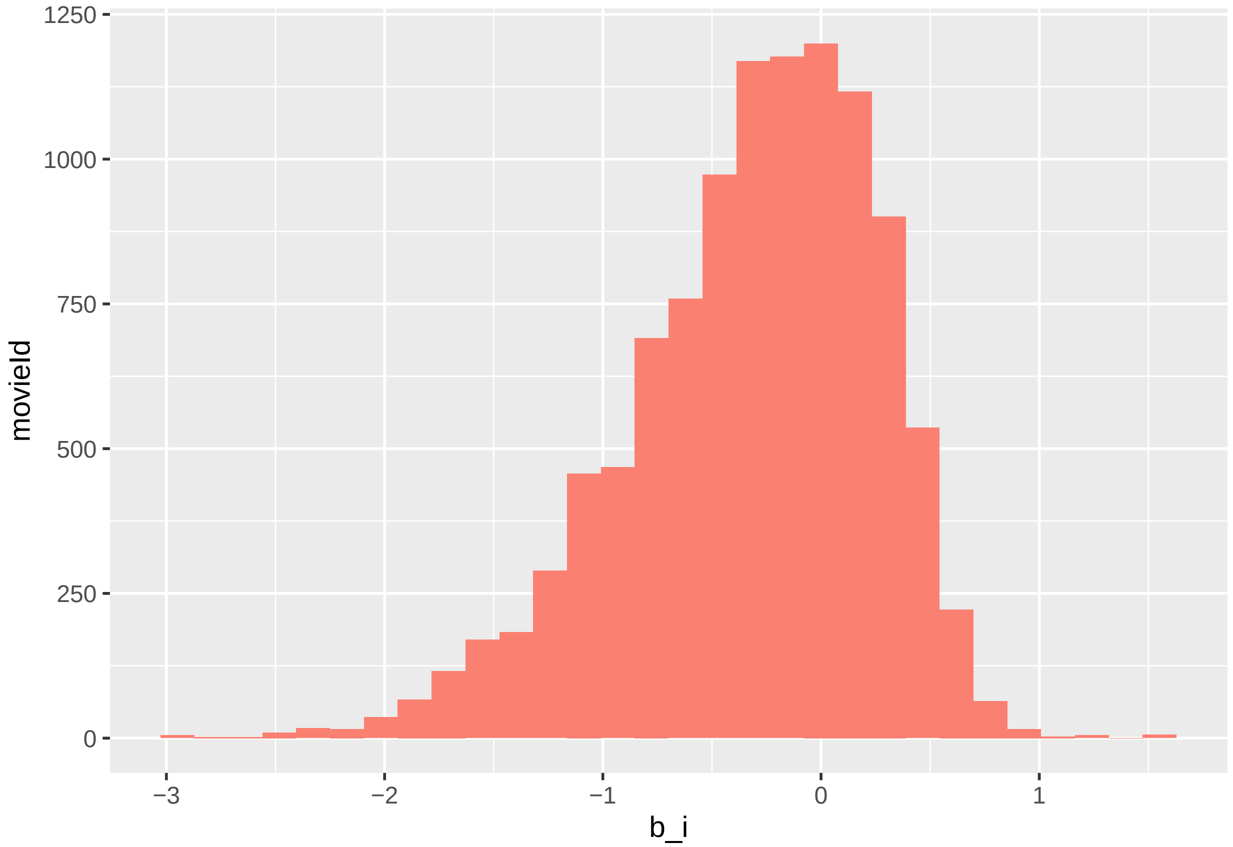


Figure 7: Movie bias: movie individual ratings normalized by overall avg

By adding the computed b_i to μ, the predictive model includes movie rating bias or the difference between individual movie average and the total movie average. It will predict higher rating for a movie which has generally rated higher than average of all movies, and lower rating for a movie that rated lower than overall average. The resulting RMSE is much better than baseline's, but still is fairly high.

Table 4: RMSE Results

model	RMSE	Improvement vs Target
Mean	1.0600537	-0.1951537
Mean + Movie effects	0.9429615	-0.0780615

ANALYSIS: BASELINE VARIATIONS - COMBINE MOVIE AND USER EFFECTS

"No good movie is too long and no bad movie is short enough." –Roger Ebert

Similarly to movie bias, there exist user bias as some users are rating movies higher than others and vice versa. We could add user bias as another variable to our predictive model as follows:

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

where b_u variable is for user-specific effects. The resulting RMSE is in line with the target!

Table 5: RMSE Results

model	RMSE	Improvement vs Target
Mean	1.0600537	-0.1951537
Mean + Movie effects	0.9429615	-0.0780615
Mean + Movie and User effects	0.8646843	0.0002157

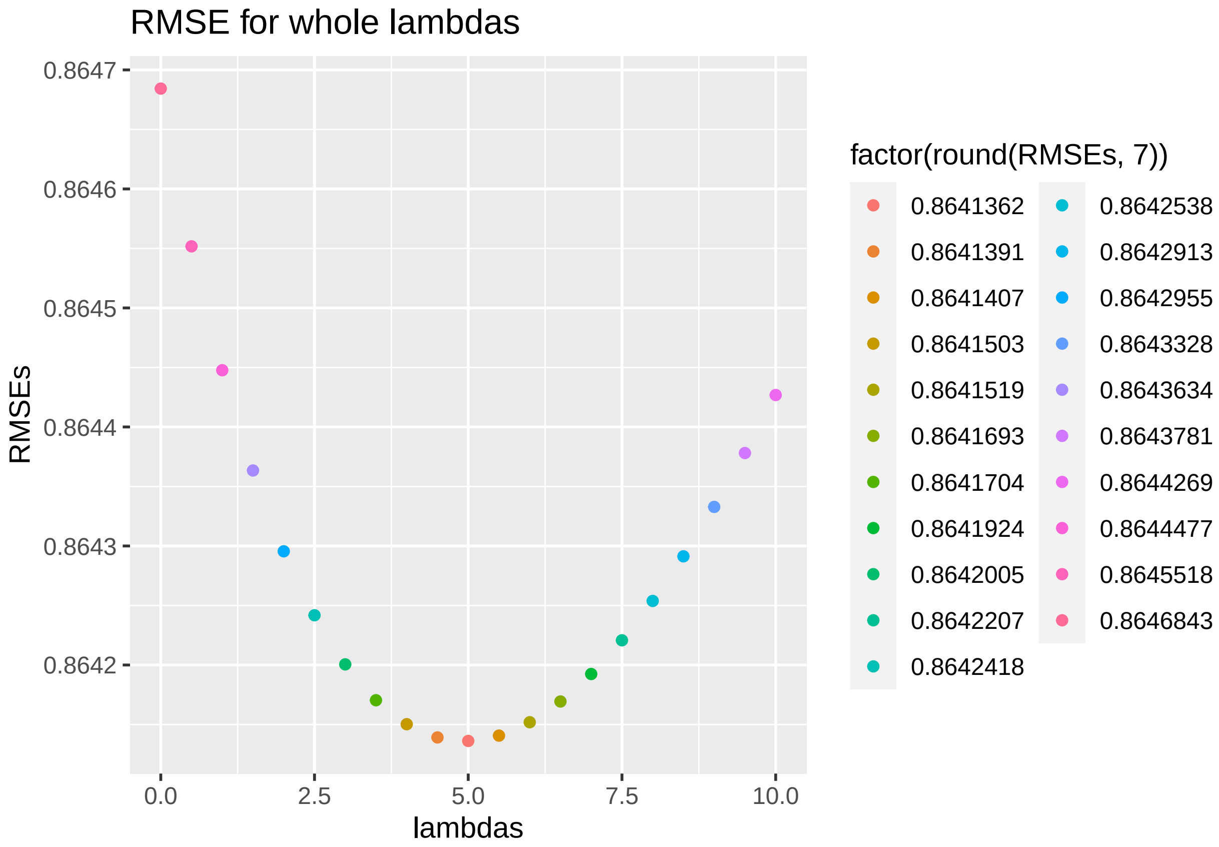
ANALYSIS: REGULIRIZED MOVIE AND USER EFFECTS MODEL

“The stiffer the penalty, the greater the message is sent.” – Lou Brock

While the last model’s RMSE of 0.8647 is fairly decent, there is also a potential hidden issue with the above models, as supposedly “best” and “worst” movies could be rated by a handful of users, introducing noise to the model. The noisy estimates are not to be trusted as they can lead to incorrect predictions (model overfitting), larger prediction errors and higher RMSE as a result.

Regularization is a technique that penalizes large estimates that are based on small sample sizes. It essentially introduces a penalty coefficient that smothers the total variability of the effects tested above.

$$b_i = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$



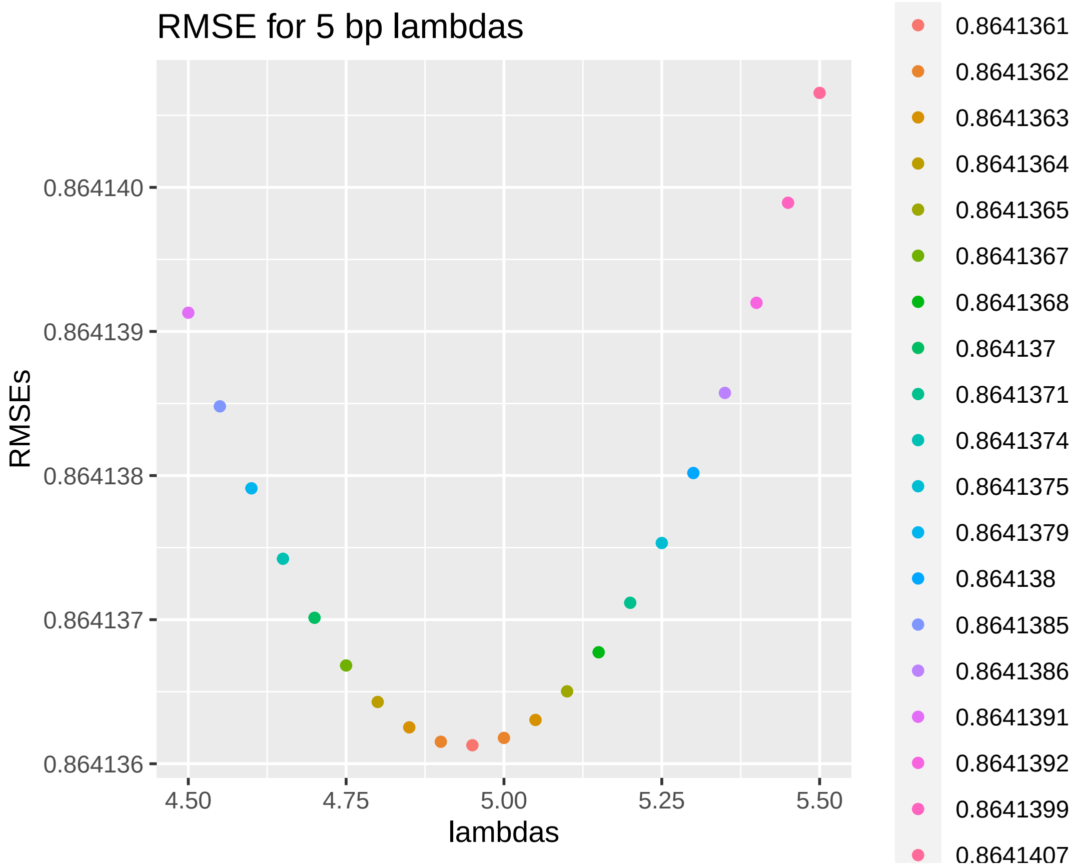


Table 6: RMSE Results

model	RMSE	Improvement vs Target
Mean	1.0600537	-0.1951537
Mean + Movie effects	0.9429615	-0.0780615
Mean + Movie and User effects	0.8646843	0.0002157
Mean + Regularized Movie and User effects	0.8641361	0.0007639

ANALYSIS: MATRIX FACTORIZATION

"What is the Matrix?" – Neo

According to (Koren and Bell 2015), there are two main techniques of collaborative filtering: *the neighborhood approach* and *latent factor* models. Neighborhood methods focus on relationships between movies or, alternatively, between users. On the other hand, matrix factorization models map both users and movies to a joint latent factor space of dimensionality f , such that user-movie interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterizing both movies and users on factors automatically inferred from user feedback. For example, such factors might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well defined dimensions such as depth of character development or "quirkiness"; or completely uninterpretable dimensions.

Accordingly, each movie i is associated with a vector $q_i \in R^f$, and each user u is associated with a vector $p_u \in R^f$. For a given movie i, the elements of q_i measure the extent to which the movie possesses those factors, positive or negative. For a given user u, the elements of p_u measure the extent of interest the user has in movies that are high on the corresponding factors (again, these may be positive or negative). The resulting dot product, $q_i^T p_u = \sum_{k=1}^{f} q_k p_k$, captures the interaction between user u and movie i; i.e., the overall interest of the user in characteristics of the movie. The final rating is created by also adding in the aforementioned baseline predictors that depend only on the user or movie. Thus, a rating is predicted by the rule:

$$Y_{u,i} = \mu + b_i + b_u + q_i^T p_u + \epsilon_{u,i}$$

In order to learn the model parameters (b_u,b_i, p_u and q_i) we minimize the regularized squared error using stochastic gradient descent method via *recosystem* R library, see (Chin et al. 2016) for reference. The resulting RMSE shows a significant improvement over target, and we can proceed to the final model validation.

Table 7: RMSE Results

model	RMSE	Improvement vs Target
Mean	1.0600537	-0.1951537
Mean + Movie effects	0.9429615	-0.0780615
Mean + Movie and User effects	0.8646843	0.0002157
Mean + Regularized Movie and User effects	0.8641361	0.0007639
Matrix Factorization	0.7803237	0.0845763

ANALYSIS: FINAL MODEL VALIDATION

"All models are wrong but some are useful." – George E. P. Box

We can finally train the complete *edx* set on matrix factorization model and estimate the RMSE on the previously unused *validation* set. If the final RMSE stays reasonably close and below the target, we have found a good enough model.

Table 8: Final RMSE Results

model	RMSE	Improvement vs Target
Mean	1.0600537	-0.1951537
Mean + Movie effects	0.9429615	-0.0780615
Mean + Movie and User effects	0.8646843	0.0002157
Mean + Regularized Movie and User effects	0.8641361	0.0007639
Matrix Factorization	0.7803237	0.0845763
Matrix Factorization - Final Validation	0.7837674	0.0811326

CONCLUSION

We have successfully constructed the movie recommendation system using Matrix Factorization with Gradient Descent method by employing *recosystem* **R** library. The achieved RMSE of 0.7837674 is 9% better than a target of 0.8649.

While the dimentionality reducing Matrix Factorization method we used is a a stand along method, it could be potentially combined with neighborhood-based methods. Neighborhood-based methods on their own might not be very robust given the data sparsity and long-tails. Another limitation of neighborhood-based methods is that they are computationally expensive as the neighborhood is used in order to extrapolate the unknown ratings of a record. In order to speed up neighborhood based method, clustering is often used.

As neighborhood-based methods can be viewed as linear models, the potential improvement to the model could be introducing the features that were left out of the analysis such as *year_released*, *year_rated*, and *genre*.

REFERENCES

Aggarwal, Charu. C. 2016. *Recommender Systems*. Textbook. https://www.springer.com/gp/book/9783319296579.

Chin, W.-S., Y. Zhuang, Y.-C. Juan, and C.-J. Lin. 2016. "LIBMF: A Matrix-Factorization Library for Recommender Systems." Weblink. https://www.csie.ntu.edu.tw/~cjlin/libmf/.

Irizarry, Rafael A. 2019. "Introduction to Data Science." Textbook. https://rafalab.github.io/dsbook/.

Koren, Yehuda, and Robert M. Bell. 2015. "Advances in Collaborative Filtering." Textbook. https://link.springer.com/chapter/10.1007/978-1-4899-7637-6_3.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2020. *R Markdown: The Definitive Guide*. Textbook. https://bookdown.org/yihui/rmarkdown/.