Movielens project report

Mathilde Couteyen Carpaye

Introduction

This report describes the creation of a movie recommendation system using the 10M version of the MovieLens dataset.

The MovieLense dataset contains movie ratings performed by anonymous users. It was divided into training and test sets. The test set or validation contained 10% of MovieLens data which is about 1 million ratings. The train set called edx contained about 9 million ratings on m = 10,677 different movies given by n = 69,878 users. Rating values range from 1 to 5 stars.

The goal of the project was to train a machine learning algorithm on the edx data set to predict ratings for the validation set with the lowest Root Mean Squared Error (RMSE). Let be $y_{u,i}$ the rating for movie i by user u and $\hat{y}_{u,i}$ our prediction, the RMSE is defined as :

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

with N the number of User/movie combinations. If the RMSE is larger than 1 it means that our typical error is larger than 1 star [1].

We first explored the data and tried to get any insight on what parameters could help to create the most accurate model. After spotting these parameters and finding a way to compute them, we could train different prediction models on our edx set. We then tested our algorithm on the validation set and compared the RMSEs obtained.

Methods

Data exploration

The data set is ordered in tidy format. Each row represents a rating by one user for one movie. We have information on the userId, movieId , timestamp, title, genre and rating.

The first thing to note is that not all movies were rated by every users. Some movies were much more rated than others and some were only rated once.Figure 1 shows the distribution of the number of ratings by movie.

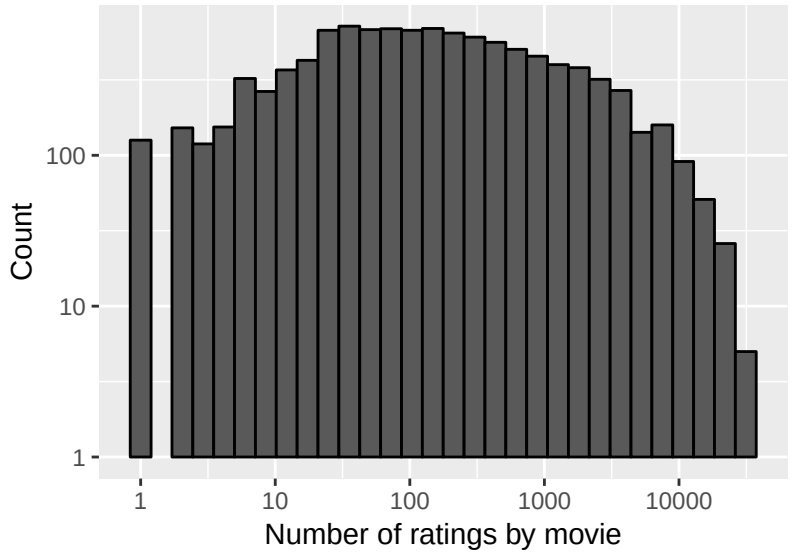

Fig 1. Histogram of log10 of the number of movies depending on log10 of the number of ratings they get.

Some users were much more active at rating movies than others. Figure 2 shows the distribution of the number of movies rated by users.

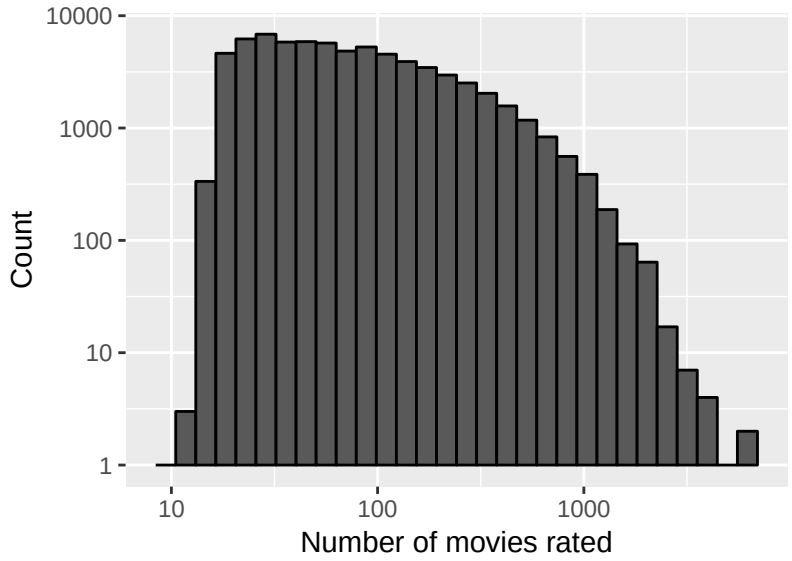

Fig 2. Histogram of log10 of the number of users depending on log10 of the number of ratings they give

The five most given ratings in order - from most to least - are : 4, 3, 5, 3.5 and 2, as we can see in Figure 3.

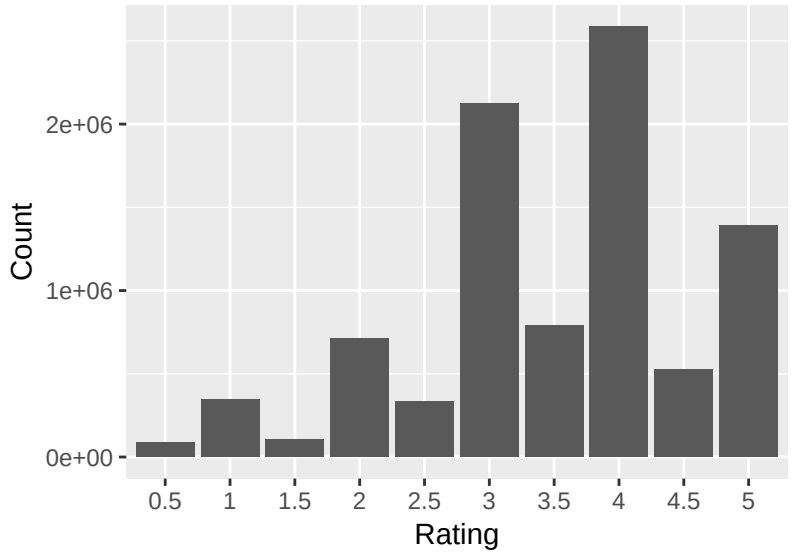

Fig 3. Histogram of the number of different ratings that were given

Plus, some movies are generally rated higher than others. Figure 4 illustrates the distribution of the average rating for a movie.

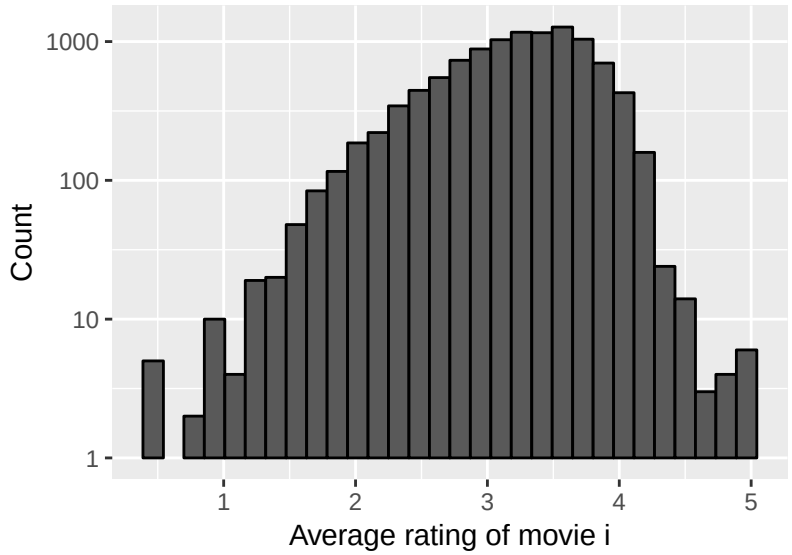

Fig 4. Histogram of log10 of the number of movies depending on their average rating

By computing the average rating for user u (only for those who have rated over 100 movies) we can see in Figure 5 that some are more likely to give very good rating and others not at all.

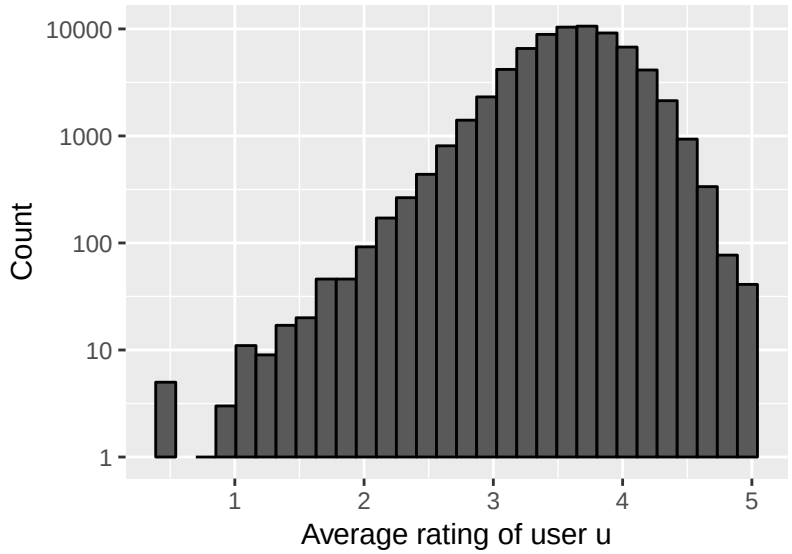

Fig 5. Histogram of log10 of the number of users depending on the average rating they give

Modeling approach

Average

First we predicted the same rating for all movies regardless of the user. We used this model :

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

with $\epsilon_{u,i}$ being independent errors and μ the rating for all movies which can be estimated by computing the average rating $\hat{\mu}$. Thus, our prediction model was :

$$\hat{y}_{u,i} = \hat{\mu}$$

Movie Effect

As we confirmed it exploring the data, different movies are rated differently. We can refer to that as the movie effect.

We augmented our previous model adding the term bi representing the average ranking for movie i.

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

We can get $\hat{b}_i$ by computing the average $y_{u,i} - \hat{\mu}$ of for all movies i.

Our prediction model then looks like this :

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i$$

User Effect

We then augmented our model by considering the fact that some users give much better or much worse rating than others. We added the user-specific effect b_u.

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

For each user u, $\hat{b}_u$ is the average of $y_{u,i} - \hat{\mu} - \hat{b}_i$.

We get this prediction model :

$$\hat{y}_{u,i} = \hat{\mu} + \hat{b}_i + \hat{b}_u$$

Regularization

As Figure 1 depicts it, some movies were only rated once. Thus, the estimates for these movies would be based on just one number which wouldn't be precise at all. Considering these movies as average movies, meaning $b_i = 0$, might provide a better estimate.

As our model is a linear model, to find the values that minimize the fitted model to the data, we can use the least squares equation. (Which could have been used to find $\hat{b}_i$, $\hat{b}_u$,but wasn't because it would have taken much more time.) Hence, we would have this least square equation :

$$\frac{1}{N}\sum_{u,i}(y_{u,i} + \mu + b_i)^2$$

However, instead of minimizing this equation, we add a penalty term that gets larger when many b_i are large so we can control the total variability of the movie effects.

$$\frac{1}{N}\sum_{u,i}(y_{u,i} + \mu + b_i)^2 + \lambda\sum_{i} b_i^2$$

The values of bi that minimize the equation are

$$\hat{b}_i(\lambda) = \frac{n_i}{\lambda + n_i}\frac{\sum_{u=1}^{n_i}(y_{u,i} + \hat{\mu})}{n_i}$$

where n_i is the number of ratings made for movie i. As a result, when n_i is very large, the penalty λ gets ignored. But, when n_i is small then the estimate gets closer to 0.

To choose the penalty λ we ran a cross-validation test. We tried different λ from 0 to 10 with a span of 0.25 and picked out the lambda which returned us the lowest RMSE.

The same process can be followed to regularize both movie and user effects.

Model training

Every model was trained thanks to R on the edx set. Then it was run on the validation set which RMSE was computed thanks to the RMSE function. All RMSEs were stored into a table with their corresponding method for later comparison.

Results

The lowest RMSE was obtained running the Movie + User Effects model. We obtained an RMSE of about 0.86535. The results obtained for each model are presented in Table 1.

Table 1. RMSEs results by method

Method	RMSE
Just the average	1.0612018

Method	RMSE
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie Effect Model	0.9438521

Using only the average to predict the rating returned us an error larger then 1. Adding the movie effect improved our prediction of 11%. And considering the user effect improved it of 7% more.

Regularization didn't improve much our movie effect model. And we didn't get a RMSE for the regularized movie + user effect model because the code took too much time to run.

Conclusion

To build the movie recommendation system, we created a linear prediction model depending on three parameters. First, the average rating μ for all movies. It is the same for all movies and all users. Because not all movies get the same average rating, we added the movie effect parameter. This takes account for the difference between the average rating for all movies and the average rating for movie u. Then, because not every user has the same tendency on rating movies, we added the user effect parameter. This third parameter takes account for the difference between the average rating predicted with the movie effect and the average rating the user i gives.

This model appeared to be pretty efficient because it returned us a RMSE of about 0.865. We improved a prediction with just the average by 18% by adding the movie and user effects parameters.

Regularization though didn't help us improved our prediction much, not even by 1%. Thus, we decided not to go for further improvement of the model, considering the RMSE result returned by the movie + user effect model satisfying.

References

[1]Rafael A. Irizarry, Data Analysis and Prediction Algorithms with R, 34.7, 2019