Creating a Movie Recommendation System Using the MovieLens 10M Dataset

Nikhil Garg

12/30/2020

Overview

The goal of this project is to create a reliable recommendation system for movies based on a dataset that was provided on MovieLens.

MovieLens is a website run by GroupLens, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities specializing in recommender systems as well as other topics. For more information on this organization, please visit their website.

To evaluate my recommendation system, I used the Root Mean Squared Error, or RMSE, metric. When I make recommendations using my model, a smaller RMSE indicates better recommendations for the dataset I'm predicting on.

This project uses a dataset that was created in 2009. It contains 10 million ratings of 10,000 movies rated by 72,000 users. There are a total of 100,000 tag applications. I used this dataset rather than the latest dataset provided on MovieLens because computing the predictions would be more feasible with a smaller dataset rather than a larger (but more recent) dataset.

For this project, the dataset was split into a training set and a validation set. The idea was to create a model that is trained on the training set, and then test that model's performance on the validation set.

Methods

Before I created my model, I first wanted to clean the data and explore it to understand it better.

One of the variables in the dataset is the *timestamp* of when the rating was recorded. This variable is recorded as Unix time (the number of seconds since the Unix Epoch, which is January 1, 1970, or 00:00:00 UTC). Since this format doesn't help with the analysis, I converted it to a standard date format (January 1, 1970). Additionally, I created two more variables that I extracted from this date variable: the year and month when the ratings were submitted.

Next, I briefly explored the data by using the *summarize* function on the edx and validation datasets. This revealed to me that the minimum rating given in both datasets was 0.5, and the maximum was 5. This was important to know because I used this knowledge to limit the range of my predictions in my model. In other words, if my model made a prediction that a rating would be greater than 5, I would adjust it to make it exactly 5. Similarly, if there was a predicted rating that was less than 0.5, I would adjust it to make it exactly 0.5. This was done in order to attempt to reduce the final RMSE.

Now, I needed to determine which model I was going to use. Initially, I wanted to use the *train* function in the *caret* package with models such as *lm*, *loess*, *glm*, and *rf* (random forest). An example of what I wanted to do is this:

```
train(train_x, train_y, method = "lm") #This didn't work
```

However, when I tried doing this, I found that running this code takes *way* too long. This is probably because the dataset is so massive that computing any model using a complicated algorithm takes forever. So, I sought another method of modeling the data.

That is when I decided to implement a bias-based model, similar to what was employed in the eighth course of the HarvardX Data Science certificate program. This model is based around the idea that each feature in the data has a certain effect on the outcome. For example, each *movie* has an effect on the rating of each movie, and this effect is usually significant because each movie could be considered good or bad, and that would affect its rating. The other features I chose for this model were *users* (those who rated the movies), *genres*, *year*, and *month* of the rating's submission.

To train the model, I split the edx training set into a smaller training set and a test set that I could use to test the model so that I can tweak the model accordingly. I used a probability of 0.1 to determine how much data from the edx set will go to the test set. The rest would go to the training set.

Results

The bias-based model was the best-performing model I created. It was improved once I performed the step of adjusting the range of my predictions from 0.5 to 5.

When I tested my model on the test edx set, I got an RMSE of 0.8653667. This was the go-ahead for me to finalize this model.

I then validated my model using the validation set, and I got an RMSE of 0.8647269. This was a satisfactory result, and it indicated that my model performed well.

Conclusion

The bias-based model was the most feasible I could have employed to predict the ratings, as other models using the *train* function from the *caret* package were taking *way* too long to run.

The limitations of this model is that there has to be information about the movie, user, genre, and time of the ratings in order for this model to work. On top of that, the final RMSE value was not ideal and could have been improved with a better model or better selection of biases.

If I were to continue working on this project in the future, I would have tried other combinations of biases and possibly include new biases I hadn't included in the original model. I would also seek other types of models to improve my RMSE, without sacrificing the feasibility of the computation.