Harvard Data Science - Capstone Project

Andrea De Nardi

2025-11-20

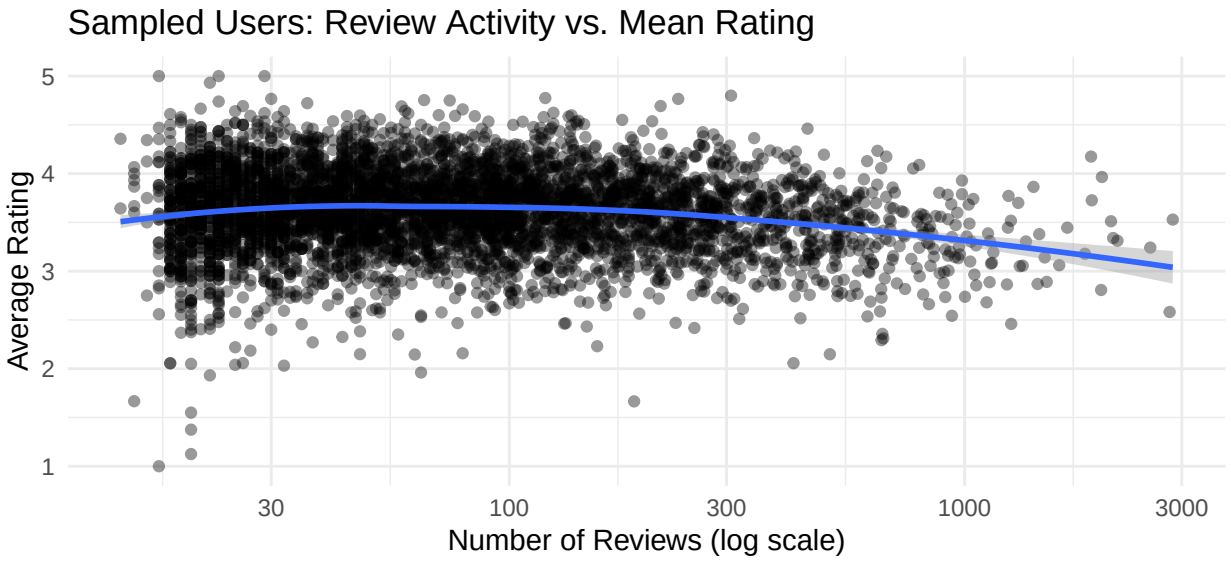
Executive Summary

In this project, we will be working with the MovieLens dataset. This dataset contains millions of reviews given across thousands of movies. The goal of our analysis is to produce a movie recommendation system trained using the MovieLens data. To begin with, we will perform a brief exploratory analysis to understand what predictors can be used to adequately model movie ratings.

We will then tackle our modeling approach with a simple, naive model. We will build upon this baseline model to train more complex models in order to gradually reduce our prediction error. Our final model will be evaluated using a root mean square error approach, where we will compare the mean euclidian distance between our predictions and the true ratings.

Analysis

Let us begin our analysis with some visualizations. Our first hypothesis is that users that give more reviews (movie buffs) may be more critical than those who have given less reviews. In order to check whether this intuition is true, we can take a sample of users, and create a scatterplot of their average rating vs the number of ratings they have given.



The graphs highlights a slight negative relationship between the number of ratings given and the average rating. However, the practical significance seems quite limited. Let us now take a look at ratings across movie genres. Perhaps, certain types of movies tend to attract harsher reviewers that others?

Table 1: Mean Rating by Movie Genre

genre	prop_ratings	mean_rating	count
genre_Film.Noir	0.0132	4.0116	118541
genre_Documentary	0.0103	3.7835	93066
genre_War	0.0568	3.7808	511147
genre_IMAX	0.0009	3.7677	8181
genre_Mystery	0.0631	3.6770	568332
genre_Drama	0.4345	3.6731	3910127
genre_Crime	0.1475	3.6659	1327715
genre_X.no.genres.listed.	0.0000	3.6429	7
genre_Animation	0.0519	3.6006	467168
genre_Musical	0.0481	3.5633	433080
genre_Western	0.0210	3.5559	189394
genre_Romance	0.1902	3.5538	1712100
genre_Thriller	0.2584	3.5077	2325899
genre_Fantasy	0.1028	3.5019	925637
genre_Adventure	0.2121	3.4935	1908892
genre_Comedy	0.3934	3.4369	3540930
genre_Action	0.2845	3.4214	2560545
genre_Children	0.0820	3.4187	737994
genre_Sci.Fi	0.1490	3.3957	1341183
genre_Horror	0.0768	3.2698	691485

It seems that indeed, there are differences in average rating across movie genres. Likewise, it seems reasonable to posit that there are rating differences across users and across movies. Based on the above assumptions, we come up with the following plan to model our recommendation system:

1. Split the data into a training and testing set.
2. Train a naive model, where our predicted rating is simply the average rating across all movies, so the true rating is the average rating plus an error term: $Y_{u,i} = \mu + \varepsilon_{u,i}$
3. Add an additional term to the naive model to take into account movie bias: $Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$, where the movie bias is defined as the difference between the average rating for a given movie and the average rating across all movies: $b_i = \frac{1}{n_i} \sum_{u \in i} (Y_{u,i} - \mu)$
4. Add an additional term to factor in user bias: $Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$, where the user bias is defined as $b_u = \frac{1}{n_u} \sum_{i \in u} (Y_{u,i} - \mu - b_i)$
5. We will adjust this model by shrinking movie and user biases toward zero ("no effect") when the number of observations is small. This is based on the assumption that movies or users with only a few ratings may skew the ratings due to low sample size (a user with only 2 movies rated 5 stars will have an artificially high b_u and therefore inflate predicted ratings. In order to mitigate this, we can introduce a regularization term in the denominator of our movie bias and user bias, that will shrink biases for lower sample sizes. Our user bias and movie bias become: $b_u(\lambda) = \frac{\sum_i (Y_{u,i} - \mu - b_i(\lambda))}{n_u + \lambda}$ and $b_i(\lambda) = \frac{\sum_u (Y_{u,i} - \mu)}{n_i + \lambda}$. The equation for movie rating is therefore: $\hat{Y}_{u,i} = \mu + b_i(\lambda) + b_u(\lambda)$.
6. We will try a different approach and create a model using *recosystem*, an R library created specifically for the design of recommendation systems.
7. Finally, we will evaluate our best performing model (the one with lowest training RMSE), by applying it to our testing data, and thus calculate the testing RMSE).

Results

Table 2: Training RMSE Comparison Across Models

Model	Training_RMSE
Naive mean model	1.0603
Movie effect model	0.9423
Movie + user effect model	0.8567
Regularized movie + user model	0.8567
Matrix factorization (recosystem)	0.7937

As we can see, our naive model provides a solid baseline with a training RMSE of: 1.0603. Taking into account movie-specific and user-specific effects helps increase the performance, with an RMSE reduced to 0.8567. We further improve the model by shrinking down movie and user biases for low sample sizes (regularization), but the gain is minimal. The most significant improvement is obtained with the *recosystem* library, which models the ratings as the inner product of two matrices (item matrix and user matrix) obtained by minimizing error over the known ratings in the training dataset. The resulting model has a training RMSE of 0.7937.

Table 3: Final Hold-Out Test RMSE - Recosystem Model

Test_RMSE
0.8217

After applying the model to the test data, we can evaluate the performance of our recommendation system on fresh new data, and we find a final test RMSE of 0.8217.

Conclusion

In this project, we developed a recommendation system using the MovieLens dataset by progressively increasing the complexity of our models and evaluating their performance. Beginning with a naive baseline, we incorporated movie-specific and user-specific effects, and subsequently applied regularization to mitigate the impact of low-sample biases. While these incremental improvements yielded modest gains, the most substantial reduction in prediction error was achieved using a matrix factorization approach via the *recosystem* library. This method, which models user–item interactions through latent factors, produced the lowest error on both the training data and the final hold-out test set, with a test RMSE of **0.8217**. Overall, our results highlight the value of latent-factor models in recommendation systems and demonstrate that even simple bias-based models can capture meaningful structure in user rating behavior. Future improvements could include hyperparameter tuning, cross-validation, and incorporating temporal effects to further refine predictive accuracy.