# HarvardX PH125.9x Data Science: Capstone
# MovieLens Rating Prediction Project

Joshua Hendrix

2025-03-26

## Contents



## Introduction

Movie or TV show recommendations are an essential feature of most content distribution services, especially streaming services. The recommendation algorithm is an important part of keeping the user engaged by providing personal suggestions and promoting greater satisfaction. The recommendation algorithm also needs to account for several sources of bias when providing a recommendation to the user.

The goal of this project is to develop a recommendation algorithm using statistical analysis and machine learning techniques. The dataset analyzed comes from MovieLens and contains 10 million entries. The analysis is performed using R version 4.4.1. To evaluate the final prediction model, ten percent of the data will be randomly selected and stored in the `final_holdout_test` variable. The remaining data is split into training and test data sets for model development. Root Mean Squared Error (RMSE) is used as the evaluation metric throughout the algorithm development to measure prediction accuracy.

## Data Analysis

### Data Cleaning

Before moving forward with the analysis, the dataset was examined for missing or invalid values. The `is.na()` function checks for null entries in the dataset.

```
hasNulls <- any(is.na(edx))
hasNulls
```

```
## [1] FALSE
```

The data did not contain any missing values therefore no data removal or adjusting was necessary.

## Data Exploration and Visualization

### Summary Statistics

Getting a basic understanding of the data is crucial when developing a prediction model. Basic data analysis and visualizations can help identify potential issues or biases that may influence the final model. A summary statistic printout is a great place to start.

```
summary(edx$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

```
table(edx$rating)
```

```
##
##     0.5       1     1.5       2     2.5       3     3.5       4     4.5       5
##   85374  345679  106426  711422  333010 2121240  791624 2588430  526736 1390114
```

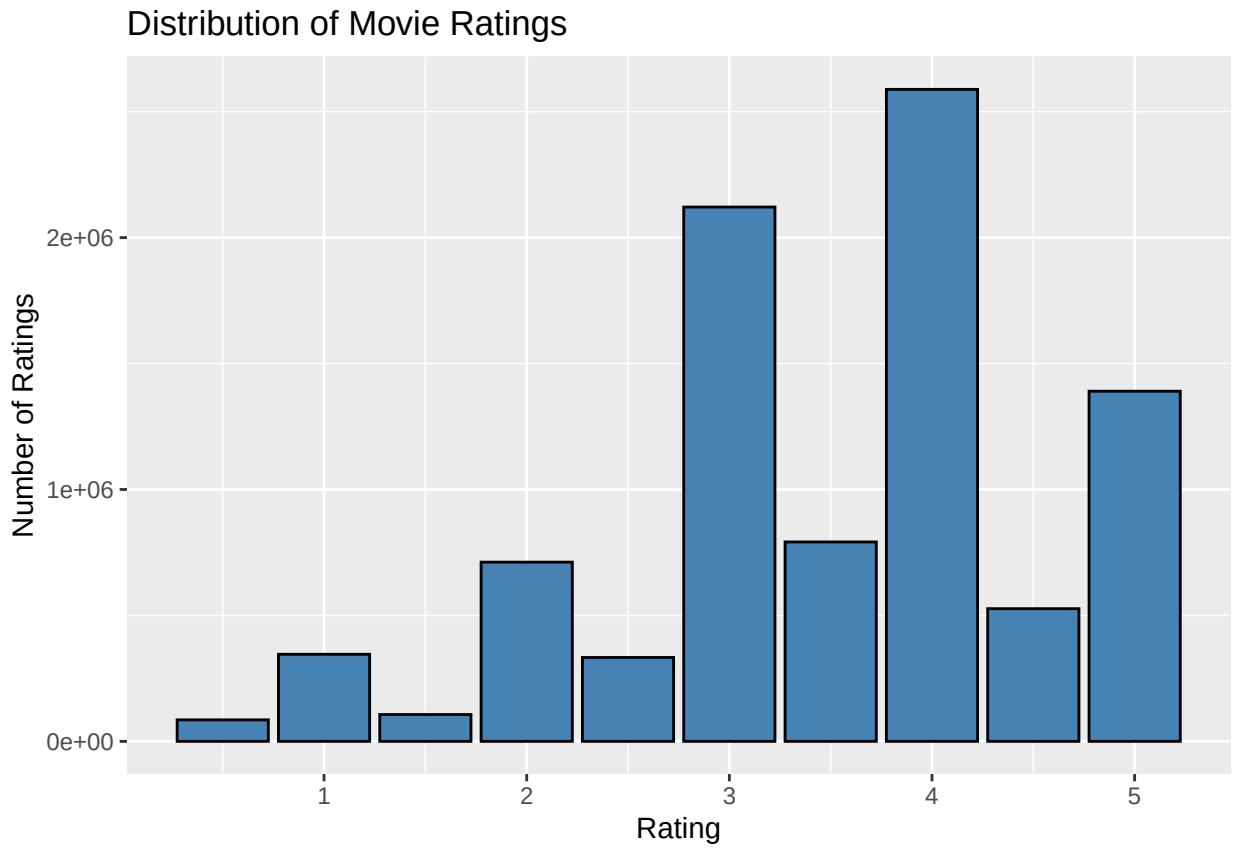


Figure 1: Distribution of Movie Ratings

Figure 1 shows a tendency towards higher ratings, with the mean rating around 3.5. Interestingly, the data suggests that users prefer to use whole number ratings over half-step ratings. This could be due to rating systems that only allow integer ratings. Given this data, the mean rating may be a great starting point for the prediction model, helping to estimate how users might rate a movie.

### Impact of Movie Popularity on Rating

Popular movies tend to receive more ratings and often have higher than average ratings. In contrast, movies that are less popular may receive fewer ratings and are likely more biased as a result. By plotting the number of ratings each movie has received versus their average rating, trends can be identified.

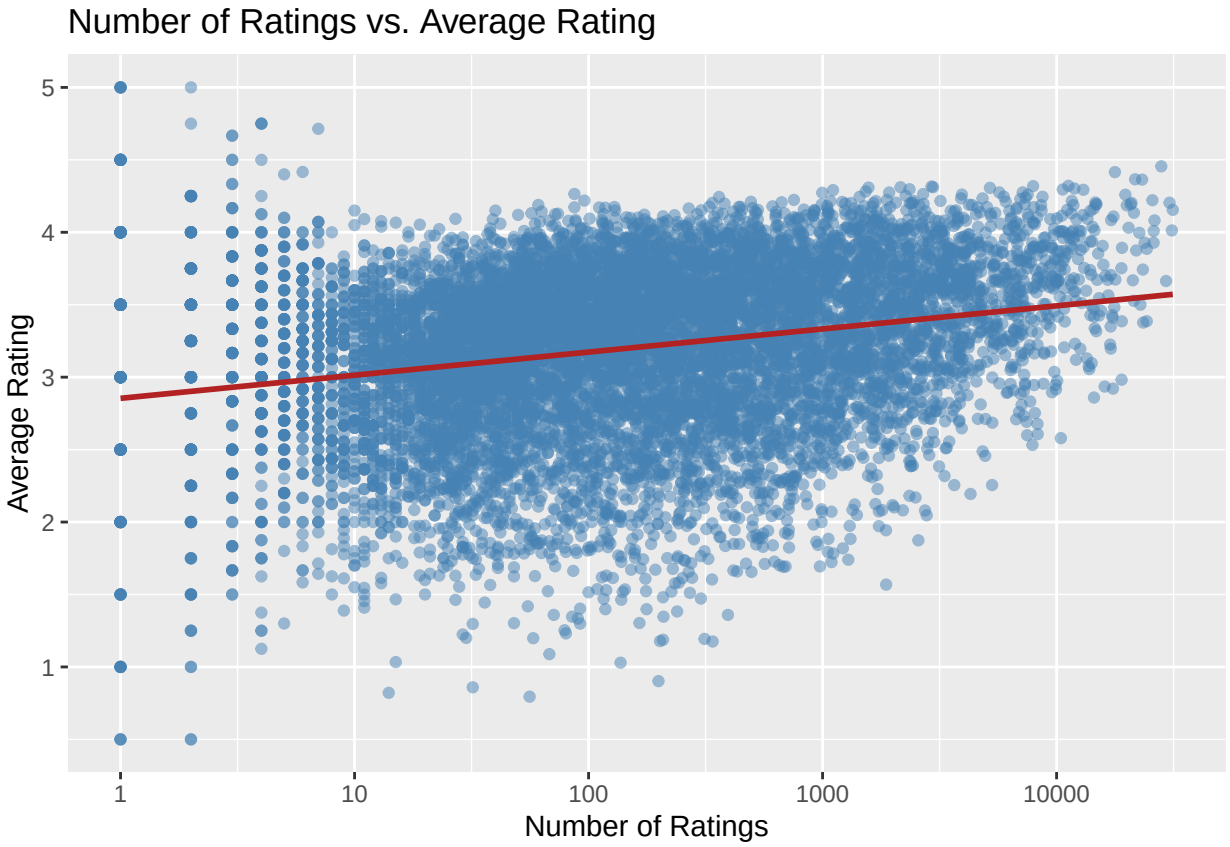


Figure 2: Impact of Movie Popularity on Rating

Figure 2 shows a definitive upward average rating trend line as movies receive more ratings. The prediction model will need to take into account the popularity of movies and adjust its rating accordingly.

However, movies with very few ratings (around ten or less) exhibit high variance in their average ratings, which may introduce bias. The prediction model should be cautions to not weigh them too heavily in the prediction process.

### User Bias

User preferences can bias the prediction model by skewing ratings higher or lower than the mean rating. Some users may prefer to rate more negatively where others may prefer to rate more positively. A histogram of user bias can help visualize how much deviation there is from the mean rating.

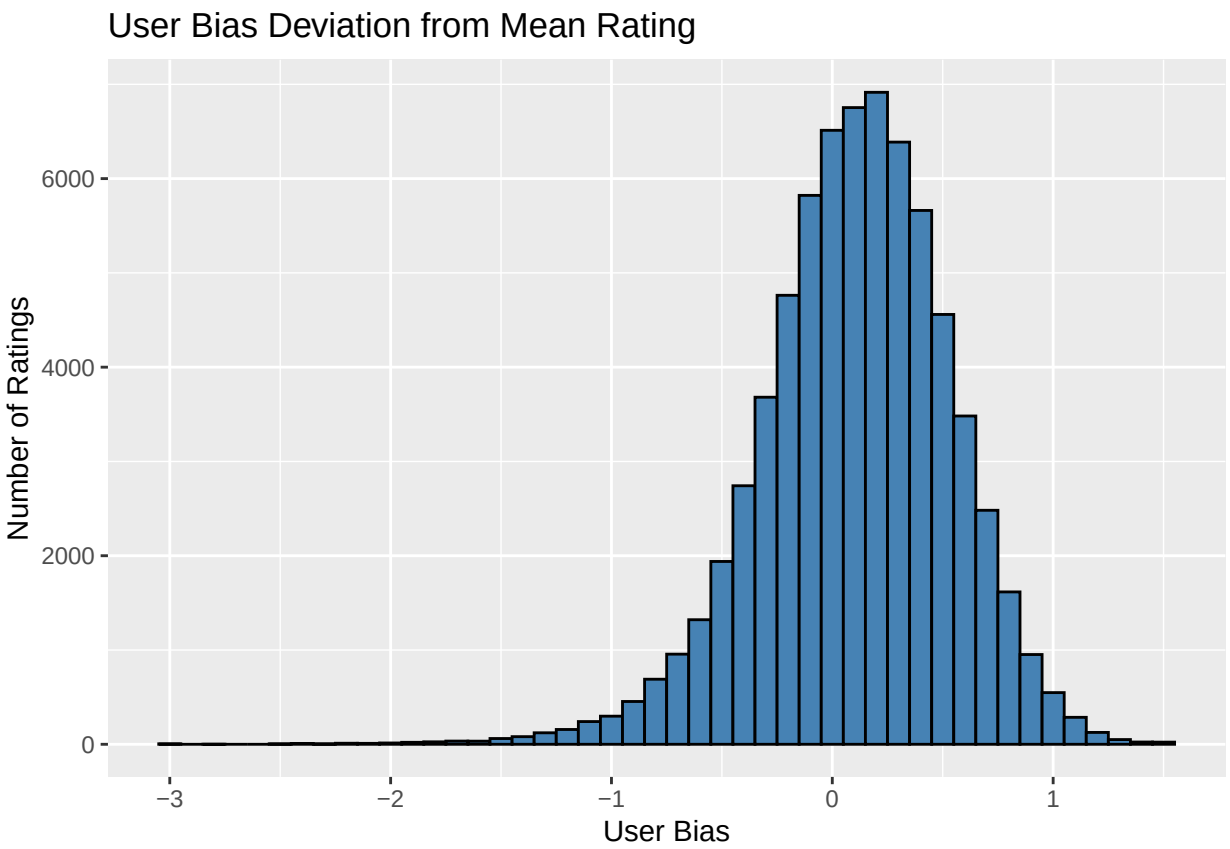


Figure 3: User Bias Distribution

Reviewing Figure 3, it can be seen that the user ratings deviation from mean rating follow a normal distribution with the mean being close to the movie's mean rating. The mean user bias is 0.1011364 with a standard deviation of 0.4306889. Using the standard deviation, the data suggests that approximately 68% of users will rate a movie within a half point of the mean, and about 95% of the users will rate within one point of the mean. However, some users stray from the mean by several standard deviations. Therefore, the prediction model will need to weigh the outliers less than the users that rate closer to the mean rating of movies.

## Model Development Process

The prediction model was developed in stages, starting with a simple baseline model and progressing through to models that take into account movie and user effects. Finally, regularization was used to reduce overfitting of data. All models were developed by splitting the training data into two sets: test and train. The initial dataset stored in `final_holdout_test` is saved for the final model testing and will not be used during model development.

### Test and Train Datasets

To train the models, the data must be split into training and testing sets. To prevent overfitting of data, each iteration of the model will use a different set of training and testing data selected using a random seed and the `createDataPartition` method. This step is performed for all models with varying seeds as shown:

```
set.seed(100, sample.kind="Rounding")
trainIndex <- createDataPartition(edx$rating, p=0.8, list=FALSE)
train <- edx[trainIndex,]
```

```
test <- edx[-trainIndex,]
```

where `train` is the larger dataset used to train the model and `test` is used to validate the model's accuracy utilizing RMSE.

### Simple Baseline Model

To get the baseline model, a simple average of all movie's rating can be taken.

```
avgRating = mean(train$rating)

simpleRmse <- RMSE(test$rating, avgRating)
```

The RMSE for the baseline is 1.0608086

### Movie Effects Model

The training data is grouped by `movieId` and an average rating is taken for each one. This is then subtracted from the overall average movie rating to create the movie bias.

```
avgRating = mean(train$rating)

movieAvg <- train %>%
  group_by(movieId) %>%
  summarize(movieBias = mean(rating - avgRating))

predictedRating <- avgRating + test %>%
  left_join(movieAvg, by="movieId") %>%
  pull(movieBias)

movieEffectRmse <- RMSE(test$rating, predictedRating)
```

Adding movie effects to the model produces an RMSE of 0.9446603, showing an improvement over the baseline model.

### User Effects Model

To improve on the model, user effects are added by grouping by `userId` and finding each users average rating that can skew ratings.

```
avgRating = mean(train$rating)

movieAvg <- train %>%
  group_by(movieId) %>%
  summarize(movieBias = mean(rating - avgRating))

userAvg <- train %>%
  group_by(userId) %>%
  summarize(userBias = mean(rating - avgRating))

predictedRating <- test %>%
  left_join(userAvg, by="userId") %>%
  left_join(movieAvg, by="movieId") %>%
  mutate(pred = avgRating + userBias + movieBias) %>%
  pull(pred)

userEffectRmse <- RMSE(test$rating, predictedRating)
```

The extended model results in an RMSE of 0.8862914, which further improves on the model.

### Regularized Model

To address overfitting, regularization is applied to both the user and movie bias terms. This aids in minimizing the impact that a low number of ratings can have on the model. To do this, a penalty term is added for large biases which is tuned using the `lambda` parameter. To tune the model, several values for lambda are iterated and the one that minimizes the final RMSE value is chosen.

```
lambdas <- seq(3, 7, 0.25)
avgRating = mean(train$rating)

rmses <- sapply(lambdas, function(lambda) {
  # Movie Effect
  b_movie <- train %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - avgRating) / (n() + lambda))

  # User Effect
  b_user <- train %>%
    left_join(b_movie, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_m - avgRating) / (n() + lambda))

  # Prediction
  predicted_ratings <- test %>%
    left_join(b_movie, by="movieId") %>%
    left_join(b_user, by="userId") %>%
    mutate(prediction = avgRating + b_u + b_m) %>%
    pull(prediction)

  return(RMSE(predicted_ratings, test$rating))
})

bestLambda <- lambdas[which.min(rmses)]
regularizedRmse <- rmses[which.min(rmses)]
```

The best value for `lambda` was found to be 5. To verify that a value close to a local minima was chosen, a plot can be created that shows each lambda and their respective RMSE values.

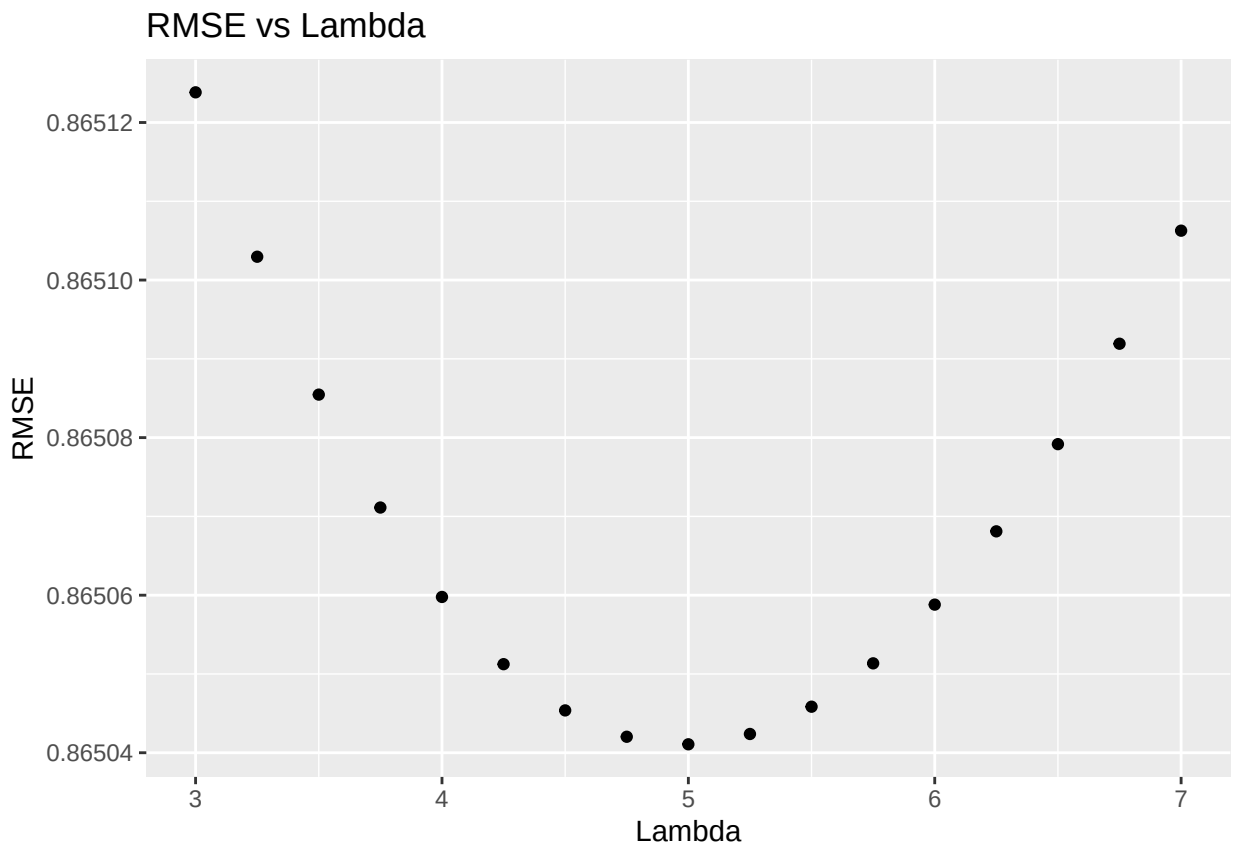


Figure 4: Lambda Tuning

As seen in Figure 4, the chosen lambda is the best value out the chosen test values. When using this lambda in the regularization model, the RMSE value is 0.8650411.

## Results

The final rating prediction model takes into account movie and user biases as well as applying regularization. The final test is to use the developed model against the `final_holdout_test` dataset to see how well the model performs.

```
# Movie Effect
b_movie <- edx %>%
group_by(movieId) %>%
summarize(b_m = sum(rating - avgRating) / (n() + bestLambda))

# User Effect
b_user <- edx %>%
left_join(b_movie, by="movieId") %>%
group_by(userId) %>%
summarize(b_u = sum(rating - b_m - avgRating) / (n() + bestLambda))

# Prediction
predicted_ratings <- final_holdout_test %>%
left_join(b_movie, by="movieId") %>%
```

```
left_join(b_user, by="userId") %>%
mutate(prediction = avgRating + b_u + b_m) %>%
pull(prediction)

finalRmse <- RMSE(predicted_ratings, final_holdout_test$rating)
```

The RMSE for the final test dataset is 0.8648178 indicating that the model has achieved good predictive performance.

# Conclusion

In this project, a movie rating prediction model was developed based on data from the MovieLens dataset. The model started as a simple average and progressed to include movie and user biases, as well as applying regularization to prevent overfitting. The final model achieved an RMSE value of 0.8648178 on the holdout test dataset.

While the model performs reasonably well, there are still potential areas for improvement. Future work could try to account for biases caused by user preferences in movie genres or how a movie with multiple genres affects its rating. Additionally, the model is limited to existing movies in the dataset. It would not accurately predict newer titles initially starting with the average movie rating until the model was refined with new ratings.

# Reference

## Figure Code

### Figure 1 Code

```
# Plot the distribution of movie ratings.
edx %>%
  ggplot(aes(x = rating)) +
  geom_bar(fill = "steelblue", color = "black") +
  labs(title = "Distribution of Movie Ratings", x = "Rating", y = "Number of Ratings")
```

### Figure 2 Code

```
# Get each movie's average rating
movie_avg_ratings <- edx %>%
  group_by(movieId) %>%
  summarize(avg_rating = mean(rating), count = n())

# Plot movie popularity (number of ratings) vs. average rating
movie_avg_ratings %>%
  ggplot(aes(x = count, y = avg_rating)) +
  geom_point(alpha = 0.5, color = "steelblue") +
  geom_smooth(method = "lm", color = "firebrick", se = FALSE) +
  scale_x_log10() +
  labs(title = "Number of Ratings vs. Average Rating",
       x = "Number of Ratings", y = "Average Rating")
```

### Figure 3 Code

```
avg_rating <- mean(edx$rating)
user_bias <- edx %>%
  group_by(userId) %>%
  summarize(user_avg_rating = mean(rating), user_bias = user_avg_rating - avg_rating)

user_bias %>%
  ggplot(aes(x = user_bias)) +
  geom_histogram(binwidth = 0.1, fill = "steelblue", color = "black") +
  labs(title = "User Bias Deviation from Mean Rating",
       x = "User Bias", y = "Number of Ratings")
```

### Figure 4 Code

```
data.frame(lambdas, rmses) %>%
  ggplot(aes(lambdas, rmses)) +
  geom_point() +
  labs(title = "RMSE vs Lambda", x = "Lambda", y = "RMSE")
```

## External References

The following are references that assisted in creating this rmarkdown file used to generate both the html and pdf documents.

**PDF Figure Position Fix**

Figures that could not fit on the remainder of a pdf page would be pushed to the next page. However, the text that followed the figures would be placed on the prior page above the image. This rearrangement created a difference in the output between the html and pdf formats. The below forum post provided the answer to ensure the figures remained in the same locations as defined in the rmarkdown file.

https://forum.posit.co/t/cant-control-position-of-tables-and-figures-in-knitted-pdf-document/37364