MovieLens Project Report

Dmitry Kuzmin

February 21, 2023

Contents

Introduction

This report presents the results of an analysis of the MovieLens 10M dataset using various models to predict movie ratings. The goal of this project is to compare the performance of these models and determine which is the most effective at predicting ratings. The dataset was downloaded from grouplens.org and contains ratings data for over 10,000 movies from more than 70,000 users. We start with a simple prediction strategy that assumes all movies have the same average rating and gradually build up to more complex models that take into account the average rating for each movie and the average deviation of each user from the dataset mean. Finally, we apply a regularized movie and user effect model that adds a penalty term to control for overfitting and use cross-validation to select the optimal value for the tuning parameter. The models are evaluated based on their root mean squared error (RMSE), and the model with the lowest RMSE is considered the best for predicting the ratings in the MovieLens dataset.

Data The data was downloaded from grouplens.org.

Data summary

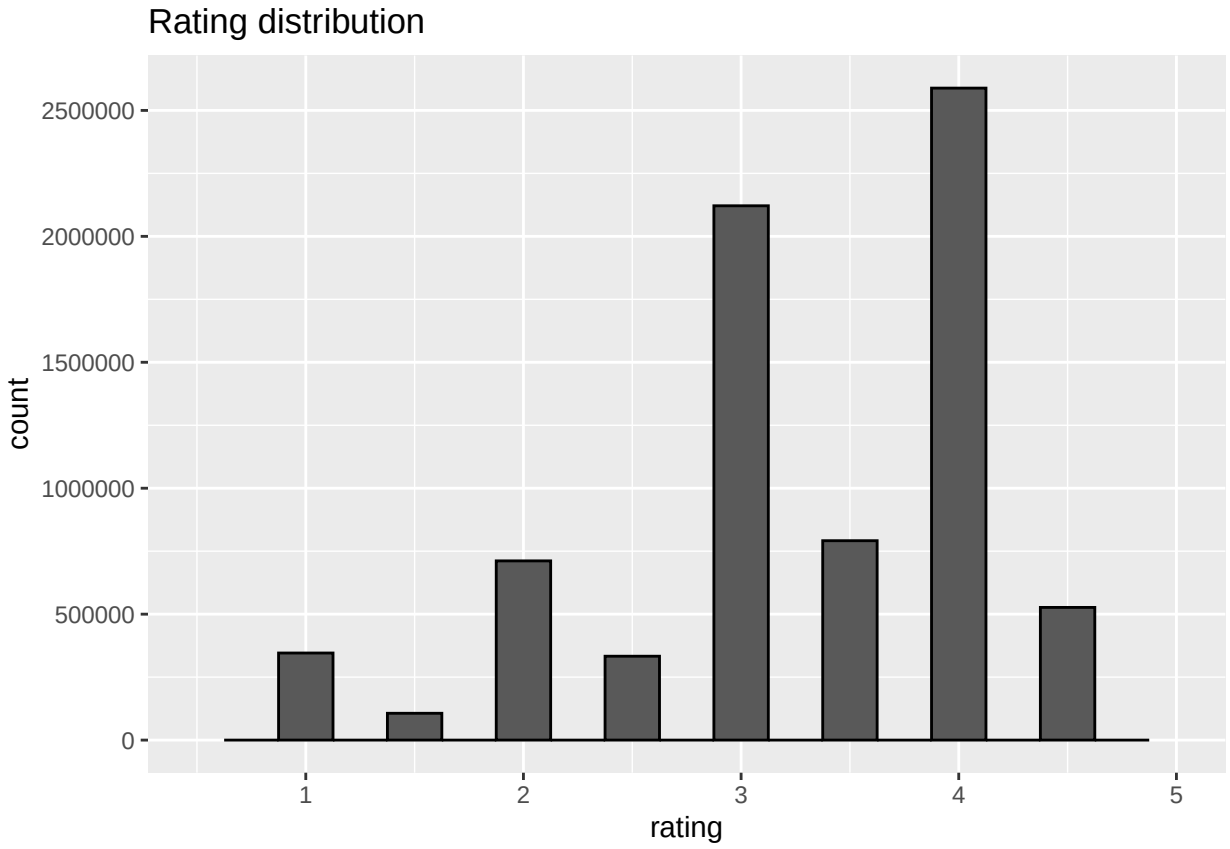
The MovieLens Project Report analyzes the MovieLens 10M dataset to predict movie ratings using several different models. The report begins with an introduction and overview of the data before diving into the analysis.

```r
# Number of unique movies and users in the edx dataset
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

```r
# Ratings distribution
edx %>%
  filter(!is.na(rating) & is.finite(rating)) %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "black") +
  scale_x_continuous(limits = c(0.5, 5.0)) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ggtitle("Rating distribution")
```

```
## Warning: Removed 2 rows containing missing values (‘geom_bar()‘).
```



Modelling approach

In this project, we apply several models to predict movie ratings and compare their performance using RMSE.

Simple prediction

We start by using a simple prediction strategy that assumes all movies have the same average rating.

```
# Compute the dataset's mean rating
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

```
# Test results based on simple prediction
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

```
## [1] 1.061202
```

```
# Save prediction in data frame
rmse_results <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
```

```
# Check results
rmse_results %>% knitr::kable()
```

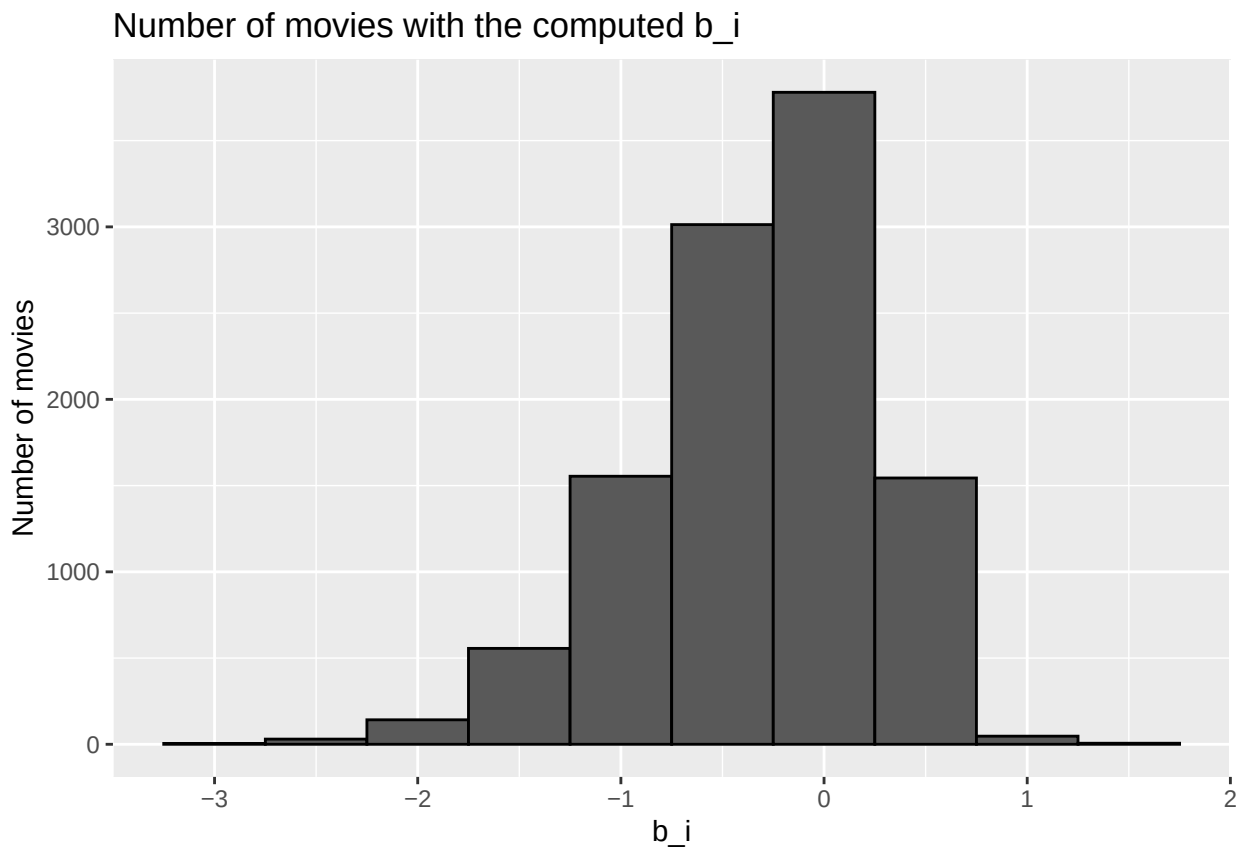
method	RMSE
Average movie rating model	1.061202

Movie effect model

Next, we build a model that takes into account the average rating for each movie.

```
# Simple model taking into account the movie effect b_i
# Subtract the rating minus the mean for each rating the movie received
# Plot number of movies with the computed b_i
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
                     ylab = "Number of movies", main = "Number of movies with the computed b_i")
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```




```
# Test and save RMSE results
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_1_rmse ))

# Check results
rmse_results %>% knitr::kable()
```

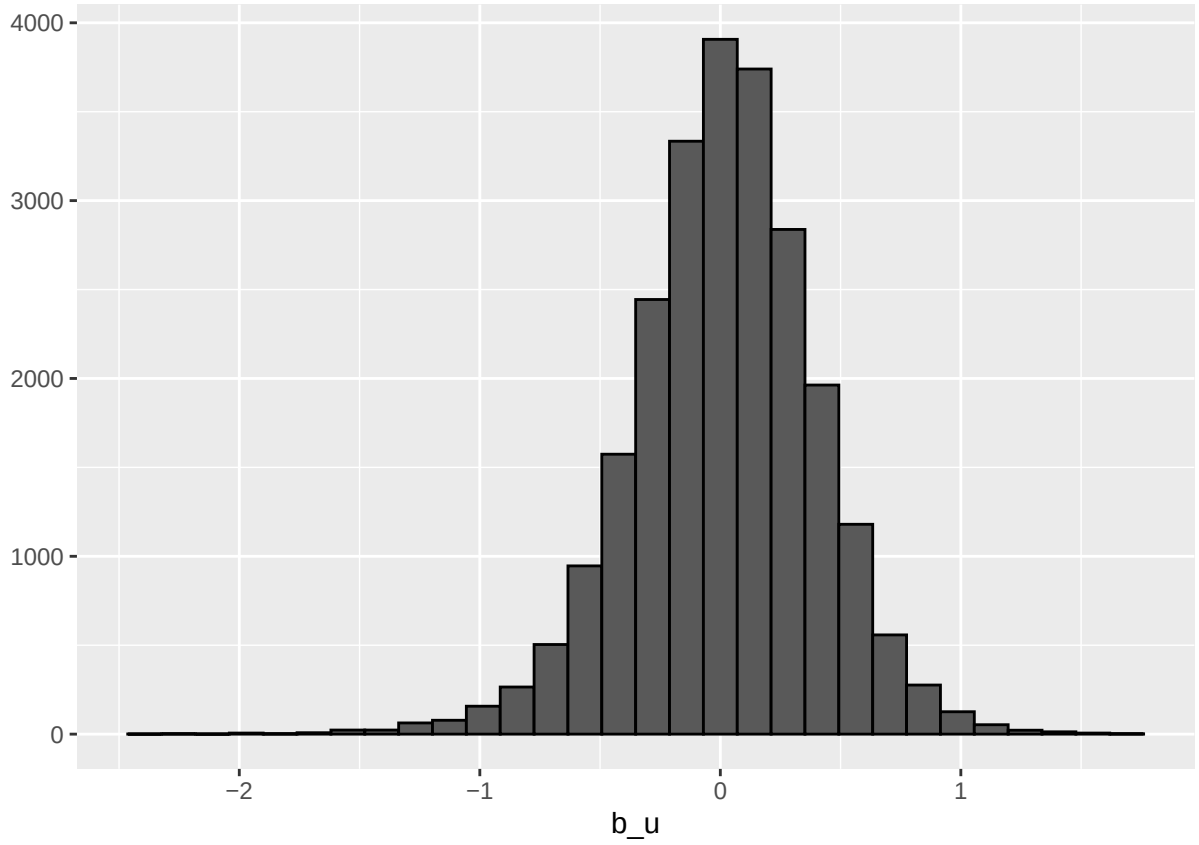
method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

Movie and user effect model

We then extend the previous model to take into account the average deviation of each user from the dataset mean.

```
# Plot penalty term user effect
user_avgs<- edx %>%
```

```
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```



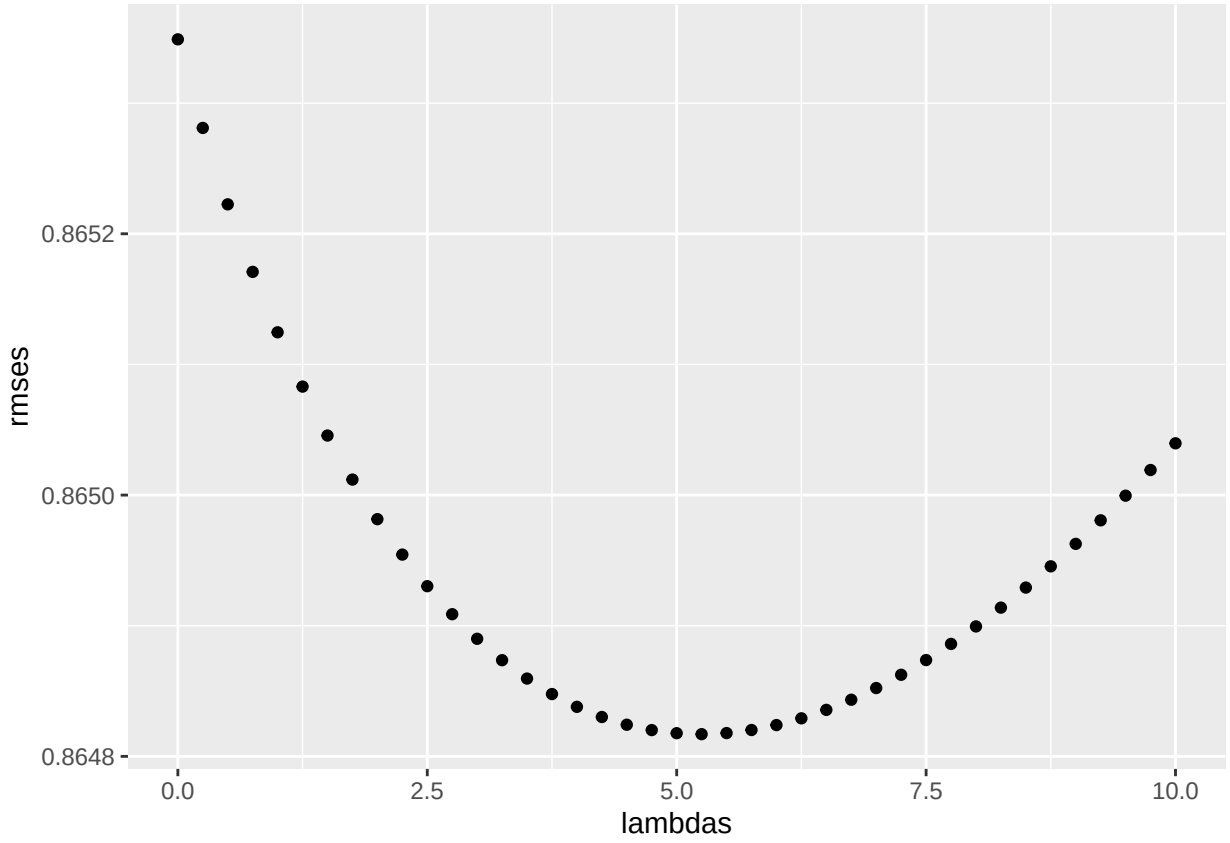

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Test and save RMSE results
predicted_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user effect model",
                                     RMSE = model_2_rmse))

# Check results
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

Regularized movie and user effect model

Finally, we apply a regularized movie and user effect model that adds a penalty term to control for overfitting. We use cross-validation to select the optimal value for the tuning parameter.

```
# lambda is a tuning parameter
# Use cross-validation to choose it.
lambdas <- seq(0, 10, 0.25)

# For each lambda, we find b_i and b_u, and then make rating predictions and test the RMSE
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() + l))

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

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

# Plot RMSEs vs lambdas to select the optimal lambda
qplot(lambdas, rmses)
```




```
# The optimal lambda
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

```
# Test and save results
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model",
                                     RMSE = min(rmses)))

# Check results
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

Conclusion

Based on the results of the analysis, we can conclude that the Regularized movie and user effect model is the most accurate model for predicting movie ratings in the MovieLens 10M dataset. This model had the lowest

root mean squared error (RMSE) compared to the other models tested, including the Average movie rating model, Movie effect model, and Movie and user effect model. The Regularized movie and user effect model takes into account the average rating for each movie and the average deviation of each user from the dataset mean, while also incorporating a penalty term to control for overfitting. This model's superior performance suggests that it is the best approach for predicting movie ratings and can be used for various applications such as movie recommendations and personalization.

```
# RMSE results overview
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170