MovieLens Report

Beauniah Kustaa

2025-06-06

Introduction

This project uses the MovieLens 10M dataset to build a recommendation system that predicts movie ratings using collaborative filtering methods. We use RMSE (Root Mean Squared Error) to evaluate our model's accuracy.

Data Preparation

We begin by downloading and preparing the MovieLens dataset. We split it into `edx` for training and `final_holdout_test` for final evaluation, making sure there's no leakage.

```r
dl <- "ml-10M100K.zip"
if (!file.exists(dl)) {
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
}
ratings <- read_lines("ml-10M100K/ml-10M100K/ratings.dat")
movies <- read_lines("ml-10M100K/ml-10M100K/movies.dat")

ratings <- as.data.frame(str_split(ratings, fixed("::"), simplify = TRUE))
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(movies, fixed("::"), simplify = TRUE))
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>% mutate(movieId = as.integer(movieId))

movielens <- left_join(ratings, movies, by = "movieId")

set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

```r
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, final_holdout_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

```r
edx <- rbind(edx, removed)
```

Data Exploration

Here we check the structure of the dataset — including number of users, movies, and how ratings are distributed.

```r
dim(edx)
```

```
## [1] 1314396       6
```

```r
n_distinct(edx$userId)
```

```
## [1] 10448
```

```r
n_distinct(edx$movieId)
```

```
## [1] 9959
```

```r
edx %>% count(rating) %>% arrange(desc(n))
```

```
##    rating      n
## 1     4.0 380046
## 2     3.0 312852
## 3     5.0 206297
## 4     3.5 112191
## 5     2.0 105330
## 6     4.5  72688
## 7     1.0  51000
## 8     2.5  47729
## 9     1.5  14999
## 10    0.5  11264
```

Model Development

We build our model in steps: - Start with the global average rating - Add **movie effects** - Add **user effects**

```r
mu <- mean(edx$rating)

movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

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

Model Evaluation on edx

We use RMSE to evaluate performance on the training dataset.

```r
predicted_ratings <- edx %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

rmse <- sqrt(mean((predicted_ratings - edx$rating)^2))
rmse
```

```
## [1] 0.8549815
```

Final Model and Holdout Evaluation

We apply the trained model to the `final_holdout_test` set and calculate the RMSE.

```r
final_predictions <- final_holdout_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

final_rmse <- sqrt(mean((final_predictions - final_holdout_test$rating)^2))
final_rmse
```

```
## [1] 0.8717645
```

Conclusion

Our final model uses a simple yet effective approach combining global average, movie effects, and user effects. The RMSE achieved on the final holdout test set is:

0.8717645

This model can be further improved by: - Adding regularization to avoid overfitting - Trying matrix factorization (`recosystem`) - Using hybrid methods combining content and collaborative filtering