# MovieLens Capstone Report

Yvonne Kirschler

2025-04-11

## Introduction

This capstone project is part of the HarvardX Data Science Professional Certificate.
The goal is to build a model that predicts movie ratings based on the MovieLens 10M dataset.
Model performance is evaluated using RMSE.
The final model is tested only once on `final_holdout_test` in accordance with project rules.

## Data Preparation

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)

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

ratings <- as.data.frame(str_split(read_lines("ml-10M100K/ratings.dat"), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
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(read_lines("ml-10M100K/movies.dat"), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))
```

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

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

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

removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

# Exploratory Data Analysis

```r
glimpse(edx)
```

```
## Rows: 9,000,061
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ movieId   <int> 122, 185, 231, 292, 316, 329, 355, 356, 362, 364, 370, 377, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421, 838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumber (1994)~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy", "Action~
```

```r
dim(edx)
```

```
## [1] 9000061       6
```

```r
n_users <- n_distinct(edx$userId)
n_movies <- n_distinct(edx$movieId)

print(n_users)
```

```
## [1] 69878
```

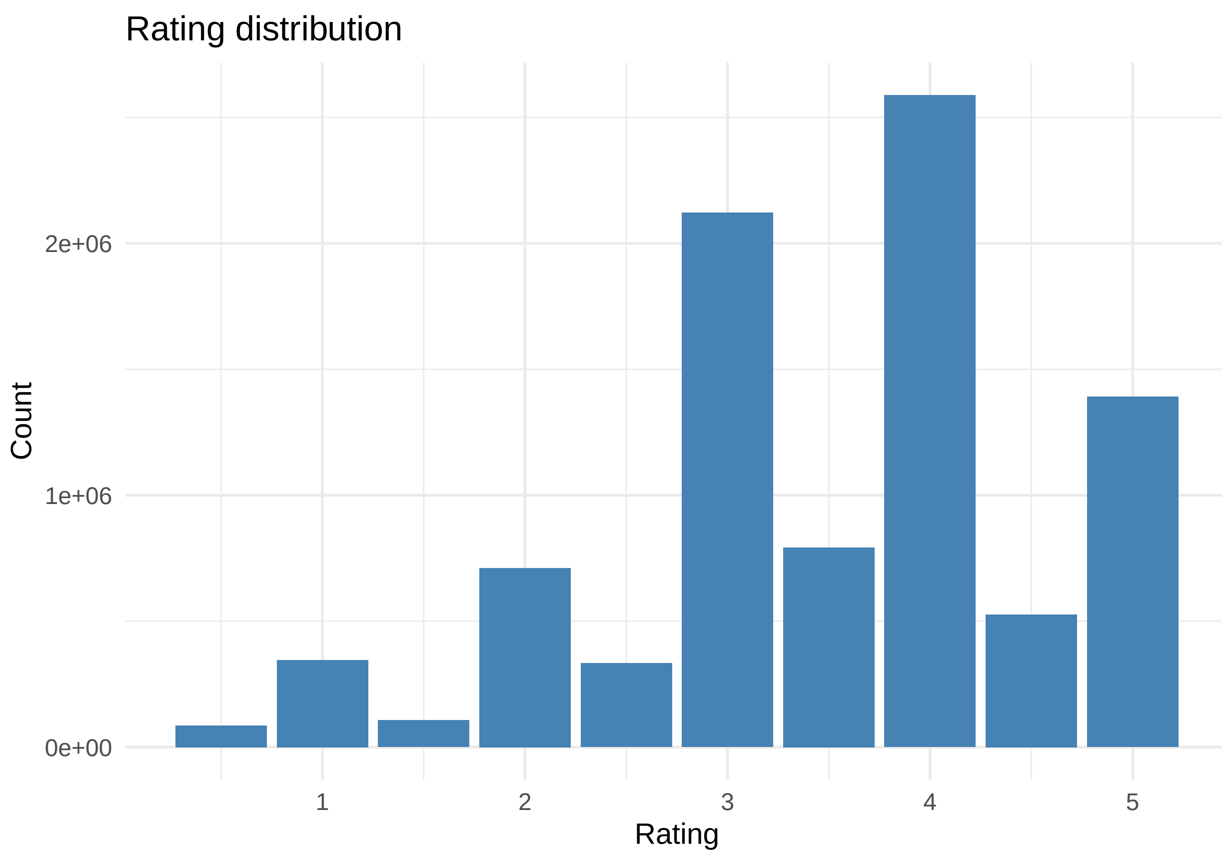
```r
print(n_movies)
```

```
## [1] 10677
```

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

```r
  ggplot(aes(x = rating, y = n)) +
  geom_col(fill = "steelblue") +
  labs(title = "Rating distribution", x = "Rating", y = "Count") +
  theme_minimal()

print(plot_rating_dist)
```




```r
top_movies <- edx %>%
  group_by(title) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  slice(1:10)

print(top_movies)
```

```
## # A tibble: 10 x 2
##    title                                                         count
##    <chr>                                                         <int>
##  1 Pulp Fiction (1994)                                           31336
##  2 Forrest Gump (1994)                                           31076
##  3 Silence of the Lambs, The (1991)                              30280
##  4 Jurassic Park (1993)                                          29291
##  5 Shawshank Redemption, The (1994)                              27988
##  6 Braveheart (1995)                                             26258
##  7 Terminator 2: Judgment Day (1991)                             26115
```

```
##  8 Fugitive, The (1993)                                       26050
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25809
## 10 Batman (1989)                                              24343
```

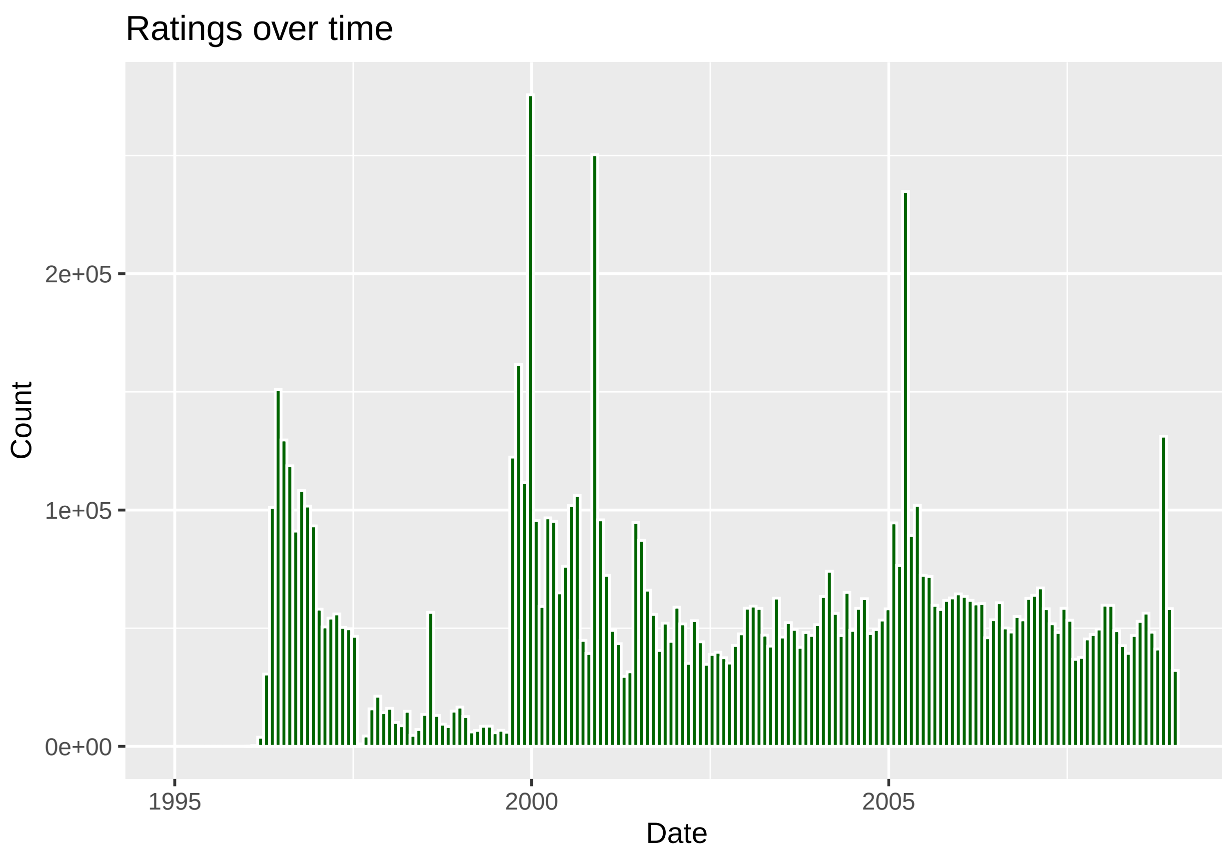
```r
top_genres <- edx %>%
  separate_rows(genres, sep = "\\|") %>%
  count(genres, sort = TRUE) %>%
  slice(1:10)

print(top_genres)
```

```
## # A tibble: 10 x 2
##    genres          n
##    <chr>       <int>
##  1 Drama     3909401
##  2 Comedy    3541284
##  3 Action    2560649
##  4 Thriller  2325349
##  5 Adventure 1908692
##  6 Romance   1712232
##  7 Sci-Fi    1341750
##  8 Crime     1326917
##  9 Fantasy    925624
## 10 Children   737851
```

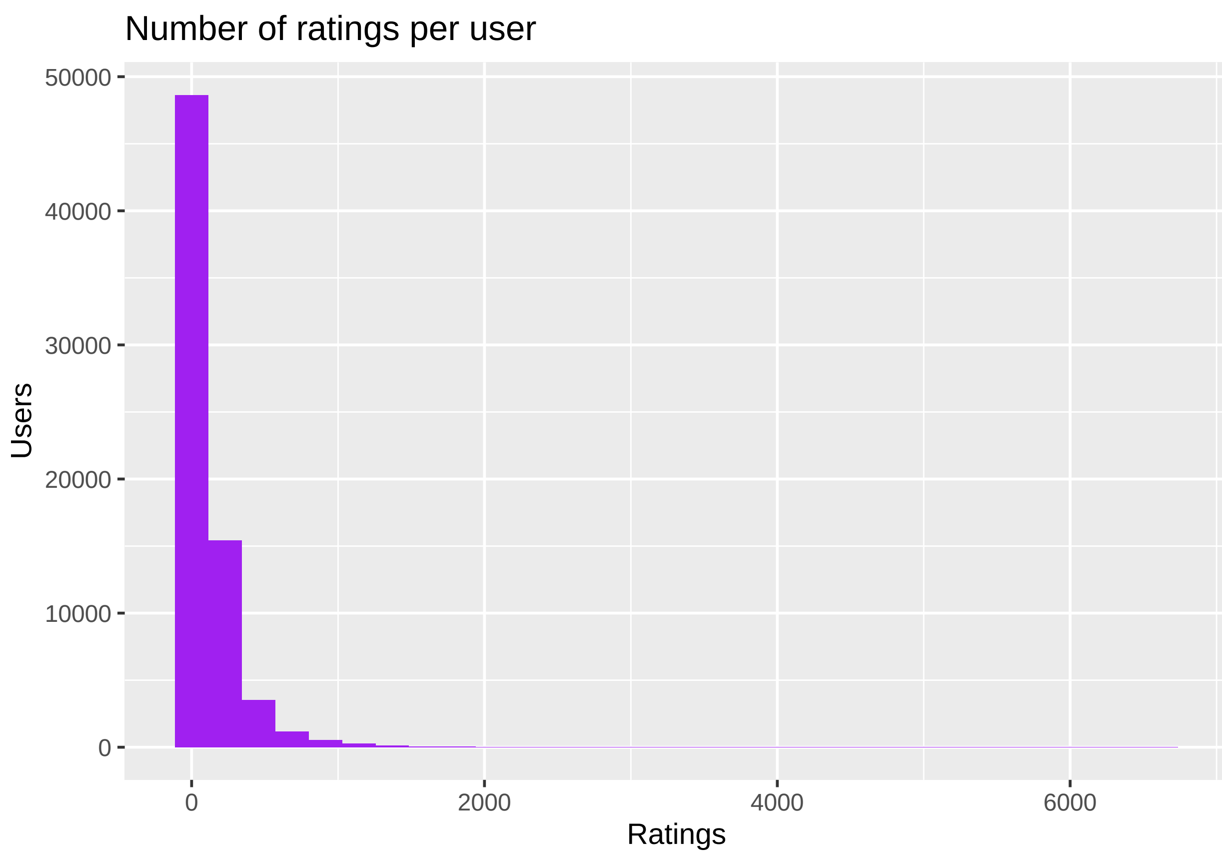
```r
plot_time <- edx %>%
  mutate(date = as_datetime(timestamp)) %>%
  ggplot(aes(x = date)) +
  geom_histogram(binwidth = 30*24*60*60, fill = "darkgreen", color = "white") +
  labs(title = "Ratings over time", x = "Date", y = "Count")

print(plot_time)
```




```
plot_user_activity <- edx %>%
  count(userId) %>%
  ggplot(aes(x = n)) +
  geom_histogram(bins = 30, fill = "purple") +
  labs(title = "Number of ratings per user", x = "Ratings", y = "Users")

print(plot_user_activity)
```




## Movie Effect Model

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

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

predicted_ratings <- final_holdout_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i)

rmse_movie_effect <- sqrt(mean((predicted_ratings$rating - predicted_ratings$pred)^2))

print(rmse_movie_effect)
```

```
## [1] 0.9437046
```

## Movie + User Effect Model

```r
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))

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

rmse_user_effect <- sqrt(mean((predicted_ratings$rating - predicted_ratings$pred)^2))

print(rmse_user_effect)
```

```
## [1] 0.8655329
```

## Regularized Model

```r
lambdas <- seq(0, 10, 0.25)

RMSE <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

rmse_results <- sapply(lambdas, function(lambda) {
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + lambda), .groups = "drop")

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

  predicted_ratings <- final_holdout_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(
      b_i = coalesce(b_i, 0),
      b_u = coalesce(b_u, 0),
      pred = mu + b_i + b_u
    ) %>%
    pull(pred)
```

```r
  RMSE(final_holdout_test$rating, predicted_ratings)
}, simplify = TRUE)

rmse_results_df <- data.frame(lambda = lambdas, rmse = rmse_results)

print(rmse_results_df)
```

```
##    lambda      rmse
## 1    0.00 0.8655329
## 2    0.25 0.8654680
## 3    0.50 0.8654111
## 4    0.75 0.8653602
## 5    1.00 0.8653141
## 6    1.25 0.8652723
## 7    1.50 0.8652344
## 8    1.75 0.8652000
## 9    2.00 0.8651688
## 10   2.25 0.8651406
## 11   2.50 0.8651153
## 12   2.75 0.8650927
## 13   3.00 0.8650725
## 14   3.25 0.8650548
## 15   3.50 0.8650394
## 16   3.75 0.8650261
## 17   4.00 0.8650149
## 18   4.25 0.8650056
## 19   4.50 0.8649982
## 20   4.75 0.8649926
## 21   5.00 0.8649887
## 22   5.25 0.8649864
## 23   5.50 0.8649857
## 24   5.75 0.8649865
## 25   6.00 0.8649888
## 26   6.25 0.8649924
## 27   6.50 0.8649974
## 28   6.75 0.8650036
## 29   7.00 0.8650111
## 30   7.25 0.8650197
## 31   7.50 0.8650294
## 32   7.75 0.8650403
## 33   8.00 0.8650522
## 34   8.25 0.8650651
## 35   8.50 0.8650789
## 36   8.75 0.8650937
## 37   9.00 0.8651094
## 38   9.25 0.8651260
## 39   9.50 0.8651434
## 40   9.75 0.8651616
## 41  10.00 0.8651806
```

```r
best_result <- rmse_results_df %>% arrange(rmse) %>% slice(1)
print(best_result)
```

```
##   lambda      rmse
## 1    5.5 0.8649857
```

## Final Model & RMSE

```
lambda <- 5.25

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + lambda), .groups = "drop")

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

final_predictions <- final_holdout_test %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(
    b_i = coalesce(b_i, 0),
    b_u = coalesce(b_u, 0),
    pred = mu + b_i + b_u
  )

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

print(final_rmse)
```

```
## [1] 0.8649864
```

## RMSE Summary

```
data.frame(
  Model = c("Movie Effect", "Movie + User Effect", "Regularized Final"),
  RMSE = c(rmse_movie_effect, rmse_user_effect, final_rmse)
)
```

```
##                 Model      RMSE
## 1        Movie Effect 0.9437046
## 2 Movie + User Effect 0.8655329
## 3   Regularized Final 0.8649864
```

## Conclusion

The final model achieved an RMSE of 0.864986, using regularized Movie + User effects and lambda tuning (`lambda = 5.25`).
This model performs better than the simpler baselines and meets the capstone criteria.