Movielens report

Santiago 1975

2025-05-04

1. Introduction

Recommendations systems use historic data to make suggestions that specific users will likely find relevant or interesting. Providers of digital entertainment like Netflix develop algorithms that use historic movie ratings to predict the ratings of specific users. From the predicted ratings, recommendations can be made concerning the next movies a user should watch. In 2006, Netflix popularized recommendation system algorithms through a million dollar challenge. Here, a predictive movie ratings model reminiscent of the nearly 20-year old algorithms of the Netflix challenge were developed as a learning exercise.

As source data, the publicly available **MovieLens 10M** data set was used. The data set contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens. Users included in the data set were selected at random from those having rated at least 20 movies. Each user is represented by an id, but no other demographic information is provided. Data are contained in three files, `movies.dat`, `ratings.dat`, and `tags.dat`. Tags were not used in the exercise.

The goal of the exercise was to build from what has been learned in the course and textbook, and develop a movie ratings algorithm predicting the number of stars that users will rate individual movies. Specific Steps included:

- Acquisition, cleaning, and splitting of data into train & holdout sets using provided scripts
- Utilization of the training set and an empirical modeling framework to monitor RSME in response to different model predictors
- Identification of predictors and predictor combinations with the greatest capacity to reduce RSME
- Validation of selected model predictors using the holdout set against a target RSME of 0.865

2. Methods

Data acquisition, cleaning, and processing

Data was acquired, cleaned, and processed according to the directives and scripts provided in the assignment (Irizarry, 2025). Briefly, the complete data set was downloaded as a zip file. The zip file was then extracted to `movies.dat` and `ratings.dat`. These files were read into R as data frames and minor string processing performed. Variables for use in modeling were then selected, renamed, and formatted. The two data sets were joined into a single data frame that was then split into training (`edx`) and test sets (`final_holdout_test`) using the following code.

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

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

A seed was set to make the random test-train split reproducible. A `test_index` was made using the `createDataPartition()` function from the `caret` package. Only 10% (NB: p = 0.1) of the data was allocated to the test set. In making the data sets, the index was combined with a series of 'semi' and 'anti' joins to ensure continuity in movies and users between the `edx` and `final_holdout_test` data sets. From this point the `final_holdout_test` was not used until evaluation of the best performing model.

Exploratory data analysis

The first few lines of the `edx` training set can be observed in **Table 1**. The target variable, `rating`, is measured on a scale of 1 to 5 stars with some users also providing half stars. For this reason, the target was handled as a continuous rather than an ordinal multi-class variable. Each movie `title` is keyed to a unique `movieId` such that the two columns provide the same information. Each `userId` is a unique individual key but not linked to other variables in the data set. Each rating also included an Unix `timestamp` and a `genres` entry. While these variables were experimented with, they were not incorporated into the final model described here.

Table 1. The first few lines of the training set data frame

userId	movieId	rating	timestamp	title	genres
1	122	5	838,985,046	Boomerang (1992)	Comedy|Romance
1	185	5	838,983,525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838,983,421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838,983,392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838,983,392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

In total, the `edx` training set consisted of 9,000,055 observations of 6 variables. There were 10,677 unique movies and 69,878 unique users. Overall, users assigned a mean rating of 3.51 ± 1.06 standard deviations.

The distribution of number of times a movie was rated is displayed in **Figure 1**. While nearly two-thirds of movies were rated less than 500 times, some movies received over 30,000 ratings.

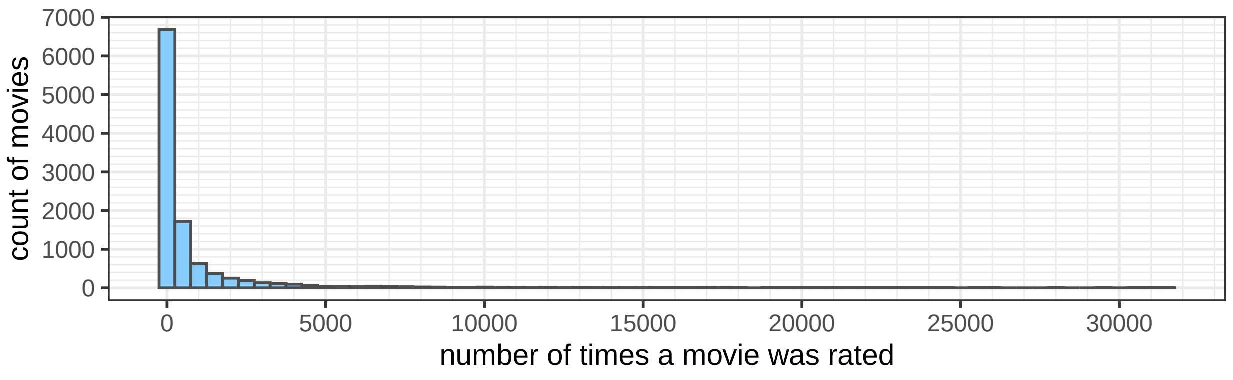

Figure 1. Distribution of the number of times a movie was rated

The top 5 movies in terms of the number of ratings is provided for interest in **Table 2**.

Table 2. Top 5 movies in terms of ratings and the number of times they were rated

movieId	title	n
296	Pulp Fiction (1994)	31,362
356	Forrest Gump (1994)	31,079
593	Silence of the Lambs, The (1991)	30,382
480	Jurassic Park (1993)	29,360
318	Shawshank Redemption, The (1994)	28,015

The distribution of the number of times a user has rated a movie is provided in **Figure 2**. While a little more than half of user rated more than 100 movies, two users rated more than 6000 movies. As stated in the `MovieLens 10M Dataset README.txt`, all users selected had rated at least 20 movies.

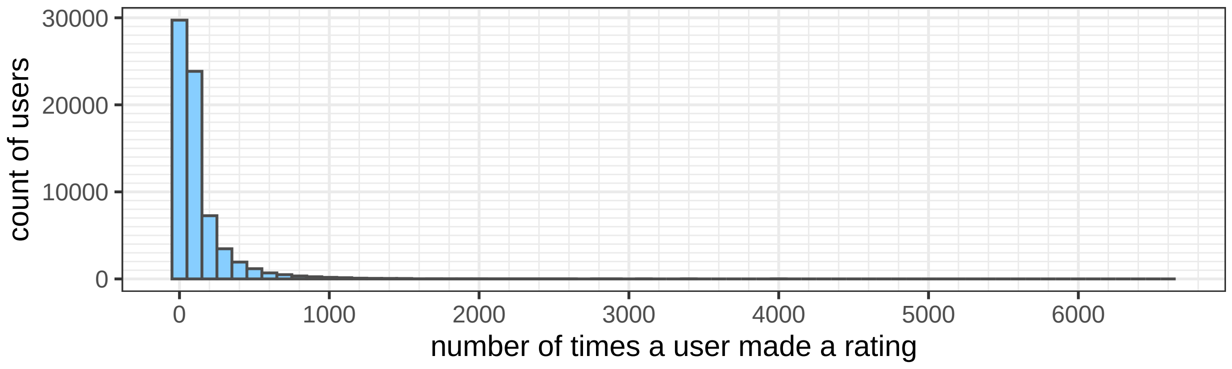

Figure 2. Distribution of the number of times a user made a rating

Insights gained

While not reproduced here, in both the course materials (Irizarry, 2024) and textbook (Irizarry, 2019), evidence for the hypothesis that users who rate specific movies similarly tend to rate other movies similarly. For example, through matrix factorization, it was shown that users either liked or disliked gangster movies such as “The Godfather” and “Goodfellas”. Similarly, a correlation in ratings for the Tom Hanks and Meg Ryan movies “Sleepless in Seattle” and “You’ve Got Mail” was apparent. Here, these putative movie x user effects ultimately proved effective in reducing the RSME model to below 0.865.

Modeling approach

The predictor set was derived empirically starting with those discussed in the course and textbook (mean, movie, and user effects). The `edx` data set was first split into train and test sets using the same strategy described previously. Predictors were then added sequentially as the residual of the preceding model. The objective was to achieve an RSME of less than 0.865 with a minimum number of predictors beyond the mean, movie, and user effects. Time and genre effects, as well as regularization, were attempted but resulted in only marginal improvements. In the end, the most promising results were achieved through principal component analysis (PCA) of the putative movie x user effects. The selected model took the form:

$y = u + b_i + b_u + p_1 q_1 + p_2 q_2 + e$

where:

- y was the predicted rating
- u was the mean rating for all movies and users
- b_i was the movie effect
- b_u was the user effect
- p_1q_1 was the first principal component & loading of the movie x user effect
- p_2q_2 was the second principle component & loading of the movie x user effect
- e was the error term from which the RSME was determined

Performance of the selected model on the `edx` data set; and relative to mean, movie, and user effects models is summarized in the **Table 3**.

Table 3. Train and test RSME of the selected model relative to mean, movie, and user effects models.

model	equation	rsme_train	rsme_test
mean_ratings	y = u + e	1.060	1.061
movie_effects	y = u + bi + e	0.942	0.944
user_effects	y = u + bi + bu + e	0.856	0.866
movie_x_user_effects	y = u + bi + bu + p1q1 + p2q2 + e	0.843	0.857

More detail concerning the derivation of each predictor term in the selected model is provided. Example code is shown relative to training on the entire `edx` data set.

Mean rating (mu) For all movies and users, the mean rating (mu) was determined using the following code.

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

Movie effects (b_i) Movie effects were derived from the residual of the *mean_rating* model. The mean residual for each `movieId` was derived using the following code. Movie effects (b_i) were stored in a data frame keyed to `movieId` for use in making predictions.

```
movie_effects <- edx %>%
  mutate(u = mu) %>%
  group_by(movieId) %>%
  summarize(bi = mean(rating - u))
```

User effects (u_i) User effects were derived from the residual of the *mean_rating + movie_effects* model. The mean residual for each `userId` was derived using the following code. User effects (u_i) were stored in a data frame keyed to `userId` for use in making predictions.

```
user_effects <- edx %>%
  mutate(u = mu) %>%
  left_join(movie_effects, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(bu = mean(rating - (u + bi)))
```

Movie x user effects ($p_n q_n$) *Movie x user effects* were derived from the residual of the *mean_rating + movie_effects + user_effects* model. The first two principal components (p) and loadings (q) of the movie x user matrix of residuals. The following scripts were used to derive the effects.

The residual of the *mean_rating + movie_effects + user_effects* model was achieved by first joining the mean, movie and user effects to the `edx` data set. For each observation, the sum of the predictor coefficients was then subtracted from the actual rating to achieve the residual. Residuals were then arranged in a `movieId` (rows) by `userId` (columns) matrix. To prepare the matrix for PCA, NA were converted to 0. Since the ratings scale is naturally standardized, no further transformation of the residuals was conducted.

```r
y <- edx %>%
  mutate(u = mu) %>%
  left_join(movie_effects, by = 'movieId') %>%
  left_join(user_effects, by = 'userId') %>%
  mutate(residual = rating - (u + bi + bu)) %>%
  select(userId, movieId, residual) %>%
  pivot_wider(names_from = "userId",
              values_from = "residual",
              values_fill = 0) %>%
  remove_rownames %>%
  column_to_rownames("movieId") %>%
  as.matrix()
```

Since the resulting matrix was large (~6 GB), a randomized PCA algorithm was used in place of traditional PCA algorithms (NB: `prcomp`) that can suffer from a lack of memory and long computational times. The first two principal components (p_1 and p_2) were wrangled into the `p` data frame and keyed to `movieId` for use in making predictions. The first two loadings were wrangled into the `q` data frame and keyed to `userId` for use in making predictions.

```r
m2 = pca(y, ncomp = 2, rand = c(5, 1))

p <- data.frame(movieId = rownames(m2$res$cal$scores),
                p1 = as.numeric(m2$res$cal$scores[,1]),
                p2 = as.numeric(m2$res$cal$scores[,2])) %>%
  mutate(movieId = as.numeric(movieId))

q <- data.frame(userId = rownames(m2$loadings),
                q1 = as.numeric(m2$loadings[,1]),
                q2 = as.numeric(m2$loadings[,2])) %>%
  mutate(userId = as.numeric(userId))
```

Testing and validation

Model performance was assessed through the RSME function defined in the following code.

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

Predictions were made as described in the following script. The test set data frame was used as a scaffold. The mean rating (*mu*) was then added as a column. Each of the movie, user, p, and q effects data frames were then joined using their relative key values (NB: `movieId` or `userId`). Predicted ratings were then calculated using the model's formula. True ratings were pulled directly from the test set data frame for entry into the `RMSE` function defined earlier.

```
true_ratings <- edx %>%
  pull(rating)

predicted_ratings <- edx %>%
  mutate(u = mu) %>%
  left_join(movie_effects, by = 'movieId') %>%
  left_join(user_effects, by = 'userId') %>%
  left_join(p, by = 'movieId') %>%
  left_join(q, by = 'userId') %>%
  mutate(rating_hat = u + bi + bu + p1*q1 + p2*q2) %>%
  pull(rating_hat)

RMSE(true_ratings, predicted_ratings)
```

3. Results

After training on the entire `edx` data set, model performance was assessed on the `final_holdout_test`. The same script as described previously was used except the `final_holdout_test` data frame was used as the scaffold for joining the effects data frames developed in training. Similarly, true ratings were pulled directly from the `final_holdout_test` for entry into the previously defined `RMSE` function. Comparison of model performance on both the `edx` training set and the validation `final_holdout_test` was performed for interest.

```
true_ratings <- final_holdout_test %>%
  pull(rating)

predicted_ratings <- final_holdout_test %>%
  mutate(u = mu) %>%
  left_join(movie_effects, by = 'movieId') %>%
  left_join(user_effects, by = 'userId') %>%
  left_join(p, by = 'movieId') %>%
  left_join(q, by = 'userId') %>%
  mutate(rating_hat = u + bi + bu + p1*q1 + p2*q2) %>%
  pull(rating_hat)

RMSE(true_ratings, predicted_ratings)
```

An RSME of 0.854 was achieved for the validation `final_holdout_test` and 0.843 for the training `edx` data sets.

4. Conclusion

Extending the mean, movie, and user effects model through movie x user effects PCA was a relatively straightforward means of reducing the RSME to below 0.865. Nonetheless, the PCA step was computationally demanding for a single computer. The 746 billion element, 6 GB matrix resulting from the `MovieLens 10M dataset` was best processed using a randomized PCA algorithm and a CPU with 16+ GB RAM. The 27,000 movies x 138,000 users full data set would result in a 3.7 trillion element matrix. With more and more data being added at an ever-increasing rate, one can imagine problems of scale with the PCA step employed here. Future work might look further into matrix factorization of the genre x user effects matrix. Similar results might be achieved through a matrix growing primarily in only one-dimension (NB: new genres are not being made like new movies) rather than the two-dimensions of the movie x user effects matrix.

5. References

- Irizarry, Rafael A. (2019) Introduction to data science: Data analysis and prediction algorithms with R. Chapman and Hall/CRC
- Irizarry, Rafael A. (2024) 'Recommendation Systems', PH125.8x: Machine Learning. HarvardX
- Irizarry, Rafael A. (2025) 'Create Train and Final Hold-out Test Sets', PH125.9x: Capstone. HarvardX