MovieLens Capstone Project

Mayesha Maliha Proma

Introduction

This project is the final part of the HarvardX Data Science Professional Certificate. The goal is to build a movie recommendation system using the MovieLens 10M dataset. The objective is to predict user ratings for movies as accurately as possible using data science techniques learned throughout the course. The final performance is evaluated using Root Mean Squared Error (RMSE).

Data Preparation

```r
ratings_file <- "/Users/mayeshamalihaproma/Downloads/ml-10M100K/ratings.dat"
movies_file  <- "/Users/mayeshamalihaproma/Downloads/ml-10M100K/movies.dat"

ratings <- read_lines(ratings_file) %>%
  str_split_fixed("::", 4) %>%
  as.data.frame(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 <- read_lines(movies_file) %>%
  str_split_fixed("::", 3) %>%
  as.data.frame(stringsAsFactors = FALSE)
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 <- bind_rows(edx, removed)
```

RMSE Function

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

Model Development

During the course, I worked with models using bias terms (global mean, movie effect, and user effect). I implemented those and achieved decent results. However, I wanted to reduce RMSE further and decided to explore matrix factorization methods.

I researched how to build better models and came across the `recosystem` package. With the help of ChatGPT, I learned how to prepare the data and tune the model using matrix factorization. I did not simply copy the approach; I understood it and implemented it step-by-step on my own.

After understanding the structure and hyperparameters, I felt more confident applying it, and I believe this method significantly improved performance. I am now comfortable using it in future recommendation system tasks as well.

```r
train_data <- edx %>% select(userId, movieId, rating)
test_data <- final_holdout_test %>% select(userId, movieId)

train_file <- tempfile()
test_file <- tempfile()
out_file <- tempfile()

write.table(train_data, file = train_file, sep = " ", row.names = FALSE, col.names = FALSE)
write.table(test_data, file = test_file, sep = " ", row.names = FALSE, col.names = FALSE)

r <- Reco()
opts <- r$tune(train_file, opts = list(dim = c(10, 20, 30),
                                       lrate = c(0.1, 0.2),
                                       costp_l2 = c(0.01, 0.1),
                                       costq_l2 = c(0.01, 0.1),
                                       nthread = 4, niter = 10))
```

```
## Warning in r$tune(train_file, opts = list(dim = c(10, 20, 30), lrate = c(0.1, : API has changed sinc
## use data_file(path) for argument 'train_data' instead
```

```r
r$train(train_file, opts = c(opts$min, nthread = 4, niter = 20))
```

```
## Warning in r$train(train_file, opts = c(opts$min, nthread = 4, niter = 20)): API has changed since v
## use data_file(path) for argument 'train_data' instead

## iter      tr_rmse          obj
##    0       0.9723   1.2002e+07
##    1       0.8727   9.8829e+06
##    2       0.8383   9.1662e+06
##    3       0.8160   8.7415e+06
##    4       0.8006   8.4649e+06
##    5       0.7886   8.2668e+06
##    6       0.7787   8.1127e+06
##    7       0.7704   7.9930e+06
##    8       0.7635   7.8951e+06
##    9       0.7578   7.8160e+06
##   10       0.7527   7.7500e+06
##   11       0.7483   7.6937e+06
##   12       0.7443   7.6447e+06
##   13       0.7407   7.6024e+06
##   14       0.7374   7.5630e+06
##   15       0.7344   7.5313e+06
##   16       0.7315   7.5005e+06
##   17       0.7289   7.4725e+06
##   18       0.7264   7.4479e+06
##   19       0.7242   7.4251e+06
```

```r
r$predict(test_file, out_file)
```

```
## Warning in r$predict(test_file, out_file): API has changed since version 0.4
## use data_file(path) for argument 'test_data' instead

## Warning in r$predict(test_file, out_file): API has changed since version 0.4
## use out_file(path) for argument 'out_pred' instead

## prediction output generated at /var/folders/vt/fg6gjf616ggdqqktqg66mlz40000gn/T//RtmpwHFMgP/file2cc7
```

```r
predicted_ratings <- scan(out_file)
```

Results

```r
final_rmse <- RMSE(final_holdout_test$rating, predicted_ratings)
cat("Final RMSE from recosystem model:", final_rmse, "\n")
```

```
## Final RMSE from recosystem model: 0.7824692
```

The RMSE I achieved was well below the benchmark of 0.8649. This shows that matrix factorization was effective in improving prediction accuracy.

Conclusion

This project brought together everything I've learned in this program: data cleaning, visualization, modeling, evaluation, and critical thinking. I also extended my knowledge by learning to use the `recosystem` package for matrix factorization, which helped me improve the model significantly.

Even though it was not taught directly in the course, I used ChatGPT and online resources to understand and apply it. This not only gave me a better result but also helped me feel more confident in exploring tools beyond the curriculum.

I am proud of my work and the learning journey, and I now feel more capable of applying data science skills in real-world applications like recommendation systems.