Movies Recommendation

2024-07-22

Used smallest dataset

Create edx and final_holdout_test sets

Note: this process could take a couple of minutes

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
options(timeout = 120)
```

```
dl <- "ml-latest-small.zip"
```

```
ratings_file <- read.csv("ml-latest-small/ratings.csv")
```

```
str(ratings_file)
```

```
## 'data.frame':    100836 obs. of  4 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  1 3 6 47 50 70 101 110 151 157 ...
##  $ rating   : num  4 4 4 5 5 3 5 4 5 5 ...
##  $ timestamp: int  964982703 964981247 964982224 964983815 964982931 964982400 96498086
8 964982176 964984041 964984100 ...
```

```
movies_file <- read.csv("ml-latest-small/movies.csv")
```

```
str(movies_file)
```

```
## 'data.frame':    9742 obs. of  3 variables:
##  $ movieId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ title  : chr  "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting
to Exhale (1995)" ...
##  $ genres : chr  "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fant
asy" "Comedy|Romance" "Comedy|Drama|Romance" ...
```

```
ratings <- read_csv("ml-latest-small/ratings.csv", col_names = TRUE, col_types = cols(
  userId = col_integer(),
  movieId = col_integer(),
  rating = col_double(),
  timestamp = col_integer()
))
```

```
movies <- read_csv("ml-latest-small/movies.csv", col_names = TRUE, col_types = cols(
  movieId = col_integer(),
  title = col_character(),
  genres = col_character()
))
```

```
head(ratings)
```

```
## # A tibble: 6 × 4
##   userId movieId rating timestamp
##    <int>   <int>  <dbl>     <int>
## 1      1       1      4 964982703
## 2      1       3      4 964981247
## 3      1       6      4 964982224
## 4      1      47      5 964983815
## 5      1      50      5 964982931
## 6      1      70      3 964982400
```

```
head(movies)
```

```
## # A tibble: 6 × 3
##   movieId title                              genres
##     <int> <chr>                              <chr>
## 1       1 Toy Story (1995)                   Adventure|Animation|Children|Comed…
## 2       2 Jumanji (1995)                     Adventure|Children|Fantasy
## 3       3 Grumpier Old Men (1995)            Comedy|Romance
## 4       4 Waiting to Exhale (1995)           Comedy|Drama|Romance
## 5       5 Father of the Bride Part II (1995) Comedy
## 6       6 Heat (1995)                        Action|Crime|Thriller
```

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

```
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind = "Rounding")  # for R 3.6 or later
```

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

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

```
# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

```
# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

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

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

```
# Remove unnecessary objects
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

```
# Verify the split
nrow(edx)  # Check number of rows in edx
```

```
## [1] 91128
```

```
nrow(final_holdout_test)  # Check number of rows in final_holdout_test
```

```
## [1] 9708
```

```
# write.csv(edx, "edx.csv", row.names = FALSE)
# write.csv(final_holdout_test, "test.csv", row.names = FALSE)
```

Script and RMSE

```
# Load necessary libraries
library(tidyverse)
library(caret)
```

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

```
# Split edx data into training and validation sets (80-20 split)
set.seed(1, sample.kind = "Rounding")
```

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

```
train_index <- createDataPartition(y = edx$rating, times = 1, p = 0.8, list = FALSE)
train_set <- edx[train_index, ]
validation_set <- edx[-train_index, ]
```

```
# Simple model: predict average rating for all movies
mu <- mean(train_set$rating)
naive_rmse <- RMSE(validation_set$rating, mu)
```

```
# Create results dataframe to store RMSEs of different models
rmse_results <- data.frame(
  method = "Average Rating Model",
  RMSE = naive_rmse
)
```

```
# Train a more complex model: movie effects model
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
# Predict ratings using the movie effects model
predicted_ratings <- validation_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
```

```r
# Calculate RMSE for movie effects model
movie_effect_rmse <- RMSE(validation_set$rating, predicted_ratings)
rmse_results <- bind_rows(rmse_results, data.frame(
  method = "Movie Effects Model",
  RMSE = movie_effect_rmse
))
```