RMarkDown

2024-09-13

Introduction

In this project, we build a movie recommendation system using the MovieLens dataset. The goal is to predict user ratings for movies they haven't seen based on patterns in their past ratings.

Data Loading

```r
# Load the data
ratings <- read.csv("ratings.csv")
movies <- read.csv("movies.csv")

# Merge the ratings and movies datasets on the movieId column
merged_data <- merge(ratings, movies, by = "movieId")

# Check the structure and the first few rows of the merged dataset
str(merged_data)
```

```
## 'data.frame':    100836 obs. of  6 variables:
##  $ movieId  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ userId   : int  1 555 232 590 601 179 606 328 206 468 ...
##  $ rating   : num  4 4 3.5 4 4 4 2.5 5 5 4 ...
##  $ timestamp: int  964982703 978746159 1076955621 1258420408 1521467801 852114051 1349082950 149421
##  $ title    : chr  "Toy Story (1995)" "Toy Story (1995)" "Toy Story (1995)" "Toy Story (1995)" ...
##  $ genres   : chr  "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Animation|Children|Comed
```

```r
head(merged_data)
```

```
##   movieId userId rating  timestamp            title
## 1       1      1    4.0  964982703 Toy Story (1995)
## 2       1    555    4.0  978746159 Toy Story (1995)
## 3       1    232    3.5 1076955621 Toy Story (1995)
## 4       1    590    4.0 1258420408 Toy Story (1995)
## 5       1    601    4.0 1521467801 Toy Story (1995)
## 6       1    179    4.0  852114051 Toy Story (1995)
##                                        genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2 Adventure|Animation|Children|Comedy|Fantasy
## 3 Adventure|Animation|Children|Comedy|Fantasy
## 4 Adventure|Animation|Children|Comedy|Fantasy
## 5 Adventure|Animation|Children|Comedy|Fantasy
## 6 Adventure|Animation|Children|Comedy|Fantasy
```

Data Preparation

We split the data into training and validation sets to avoid overfitting and ensure that the model generalizes well.

```r
# Set seed for reproducibility
set.seed(1)

# Split the data into edx (90%) and final_holdout_test (10%)
test_index <- createDataPartition(merged_data$rating, p = 0.1, list = FALSE)
edx <- merged_data[-test_index, ]
final_holdout_test <- merged_data[test_index, ]

# Ensure final_holdout_test has only users and movies that are also in edx
final_holdout_test <- final_holdout_test %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

Baseline Model: Just the Average Rating

We start by calculating the RMSE for a simple baseline model that uses the average movie rating to predict all user ratings.

```r
# Baseline Model: Just the average rating
mu <- mean(edx$rating)

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

baseline_rmse <- rmse(final_holdout_test$rating, mu)
cat("Baseline RMSE:", baseline_rmse, "\n")
```

```
## Baseline RMSE: 1.04253
```

Movie Effect Model

The next model incorporates the effect of movies by adjusting for the average rating of each movie.

```r
# Movie Effect Model
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings_movie <- final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
```

```r
movie_effect_rmse <- rmse(final_holdout_test$rating, predicted_ratings_movie)
cat("Movie Effect Model RMSE:", movie_effect_rmse, "\n")
```

```
## Movie Effect Model RMSE: 0.9617058
```

Movie + User Effect Model

We further refine the model by incorporating user-specific effects in addition to the movie effects.

```r
# Movie + User Effect Model
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

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

movie_user_effect_rmse <- rmse(final_holdout_test$rating, predicted_ratings_movie_user)
cat("Movie + User Effects Model RMSE:", movie_user_effect_rmse, "\n")
```

```
## Movie + User Effects Model RMSE: 0.8731295
```

Regularized Movie + User Effect Model

To avoid overfitting, we regularize the model by adding a penalty term to both the movie and user effects.

```r
# Regularized Movie + User Effect Model
lambdas <- seq(0, 10, 0.1)
best_lambda <- 0
best_rmse <- Inf

for (l in lambdas) {
  movie_reg_avgs <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + l))

  user_reg_avgs <- edx %>%
    left_join(movie_reg_avgs, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu) / (n() + l))

  predicted_ratings_reg <- final_holdout_test %>%
    left_join(movie_reg_avgs, by='movieId') %>%
    left_join(user_reg_avgs, by='userId') %>%
    mutate(pred = mu + b_i + b_u) %>%
```

```r
    pull(pred)

  model_rmse <- rmse(final_holdout_test$rating, predicted_ratings_reg)

  if (model_rmse < best_rmse) {
    best_rmse <- model_rmse
    best_lambda <- l
  }
}

cat("Best Regularized RMSE:", best_rmse, "\n")
```

```
## Best Regularized RMSE: 0.8527238
```

```r
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 3.1
```

Conclusion

In this project, we explored multiple models to predict movie ratings based on the MovieLens dataset. The regularized Movie + User Effects Model achieved the best RMSE score. Future improvements could include incorporating time-based effects or using matrix factorization techniques for better recommendations.

```r
# Final Regularized Model with the best lambda
movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + best_lambda))

user_reg_avgs <- edx %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu) / (n() + best_lambda))

predicted_ratings_final <- final_holdout_test %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  left_join(user_reg_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

final_rmse <- rmse(final_holdout_test$rating, predicted_ratings_final)
cat("Final Regularized Model RMSE:", final_rmse, "\n")
```

```
## Final Regularized Model RMSE: 0.8527238
```