MovieLens Project

Nik Pevnev

2025-03-19

Intro

MovieLens project is based on creating a rating machine learning algorithm trained on provided edx data and RMSE tested with final_holdout_test set. Final deliverables of the project will be: .Rmd file, .PDF document, and an R script. Below is the general info of a dataset worked with

Datasets

MovieLens 10 million movie rating dataset (http://grouplens.org/datasets/movielens/10m/) contains 10m ratings with 100,000 tags applied to 10,000 movies by 72,000 users. Dataset is then split per project requirements into 2 datasets to create efficient analysis of machine learning algorithm created: edx with 9000055 rows and final_holdout_test with 999999 rows, each dataset has 6 columns.

```r
# Check data sets created
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```r
glimpse(edx) # [1] 9000055       6
```

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
## $ movieId   <int> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ 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, 838983421, 838983392, 838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

```
summary(final_holdout_test)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
glimpse(final_holdout_test) # [1] 999999      6
```

```
## Rows: 999,999
## Columns: 6
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ movieId   <int> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 85, ~
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3.0, ~
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86824~
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alone ~
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|Come~
```

```
# How many movies with 1 rating only? - 126 movies

ratings_count <- edx %>%
  group_by(movieId) %>%
  summarise(ratings_count = n())

movies_with_one_rating <- ratings_count %>%
  filter(ratings_count == 1)

num_movies_with_one_rating <- nrow(movies_with_one_rating)
num_movies_with_one_rating
```

```
## [1] 126
```

No ratings of 0 were noted in both datasets. Now we can proceed with machine learning algorithms that can predict user ratings.

Machine Learning

Method 1 - Mean of ratings as prediction algorithm

This is most simple prediction of each rating that is constant and represents mean of all movie ratings

```
# (1) Calculate mean across all edx ratings
method1Predictions_Mean <- mean(edx$rating)
# method1Predictions # [1] 3.512465
# On average rating given by user is 3.51

# (2) Calculate RMSE of method 1 prediction
rmse1 <- RMSE(final_holdout_test$rating, method1Predictions_Mean)
rmse1
```

```
## [1] 1.061202
```

Method 2 - Remediate effects of each movie rating on the final predicted score based on mean of ratings as a basis and their difference

This is a manual process of predicting ratings and using join functions to recalculate each mean prediction by movie effect deviation

```
# (1) Generate movie effect lookup table that can be used
# as a lookup to normalize simple mean prediction
movieEffectLookupTable <- edx %>%
  group_by(movieId) %>%
  summarize(movieBias = mean(rating - method1Predictions_Mean))
# movieEffectLookupTable

# (2) Calculate predictions based on
# final_holdout_test dataset and assess its RMSE
method2Predictions <- final_holdout_test %>%
  left_join(movieEffectLookupTable, by = "movieId") %>%
  mutate(tailoredPrediction
         = method1Predictions_Mean + coalesce(movieBias, 0)) %>%
  pull(tailoredPrediction)
# method2Predictions

# (3) Calculate RMSE of method 2 prediction
rmse2 <- RMSE(final_holdout_test$rating, method2Predictions)
rmse2
```

```
## [1] 0.9439087
```

Method 3 - Remediate effects of user bias by introducing a new variable in linear model prediction, the final predicted score is based on mean of ratings as a basis and movie effects from method 2

This is a manual process of predicting ratings and using join functions to recalculate each mean prediction by movie effect and user bias deviations

```r
# (1) Generate user effect lookup table that can be used
# as a lookup to normalize simple mean prediction
userEffectLookupTable <- edx %>%
  left_join(movieEffectLookupTable, by='movieId') %>%
  group_by(userId) %>% # Group by user to calculate user bias
  summarize(userBias = mean(rating - method1Predictions_Mean - movieBias))
# userEffectLookupTable

# (2) Calculate predictions based on
# final_holdout_test dataset and assess its RMSE
method3Predictions <- final_holdout_test %>%
  left_join(movieEffectLookupTable, by = "movieId") %>%
  left_join(userEffectLookupTable, by='userId') %>%
  mutate(tailoredPrediction
         = method1Predictions_Mean
         + coalesce(userBias, 0) + coalesce(movieBias, 0)) %>%
  pull(tailoredPrediction)
# method3Predictions

# (3) Calculate RMSE of method 3 prediction
rmse3 <- RMSE(final_holdout_test$rating, method3Predictions)
rmse3
```

```
## [1] 0.8653488
```

Method 4 -Regularized movie and user bias

We are going to minimize prediction error by introducing a regularization parameter Lambda to prevent overfitting issues, where we introduce a regularization parameter n()+l to adjust movie ratings bias for movies with less ratings than popular ones, as shown below.

$$\text{Bias} = \frac{\sum_i(\text{rating}_i - \text{mean prediction})}{n + \lambda}$$

This is a multi-step process where we first find most optimal lambda for most efficient accuracy. Once we know best value to take for prediction model, we import it to our method 4 algorithm.

```r
# (1) Calculate list of optimal prediction regression
# models that can fit best to predict the best
optimalResult <- optimize(function(l) {
  # (a) Generate regularized movie effect lookup table that
  # can be used as a lookup to normalize simple mean prediction
  movieBiasLookUp <- edx %>%
    group_by(movieId) %>%
    summarize(movieBias = sum(rating
                              - method1Predictions_Mean) / (n() + l))

  # (b) Generate regularized user effect lookup table that
  # can be used as a lookup to normalize simple mean prediction
  userBiasLookUp <- edx %>%
    left_join(movieBiasLookUp, by = "movieId") %>%
    group_by(userId) %>%
```

```r
    summarize(userBias = sum(rating - coalesce(movieBias, 0)
                             - method1Predictions_Mean) / (n() + 1))

  # (c) Calculate predictions based on
  # final_holdout_test dataset and assess its RMSE
  method4Predictions <- final_holdout_test %>%
    left_join(movieBiasLookUp, by = "movieId") %>%
    left_join(userBiasLookUp, by = "userId") %>%
    mutate(tailoredPrediction = method1Predictions_Mean
           + movieBias + userBias) %>%
    pull(tailoredPrediction)

  # (d) Calculate RMSE of optimization method
  return(RMSE(final_holdout_test$rating, method4Predictions))
}, lower = 0, upper = 10)

# (2) Extract the most optimal lambda and RMSE
optimalLambda <- optimalResult$minimum
optimalLambda
```

```
## [1] 5.240516
```

```r
optimalRMSE <- optimalResult$objective
optimalRMSE
```

```
## [1] 0.864817
```

```r
# (3) Calculate most optimal prediction regression
# models that can fit best to predict the best
# ------------------------------------------------

# (a) Generate regularized movie effect lookup table that
# can be used as a lookup to normalize simple mean prediction
movieBiasLookUp <- edx %>%
  group_by(movieId) %>%
  summarize(movieBias = sum(rating - method1Predictions_Mean)
            / (n() + optimalLambda))

# (b) Generate regularized user effect lookup table that
# can be used as a lookup to normalize simple mean prediction
userBiasLookUp <- edx %>%
  left_join(movieBiasLookUp, by = "movieId") %>%
  group_by(userId) %>%
  summarize(userBias = sum(rating - coalesce(movieBias, 0)
                           - method1Predictions_Mean) / (n() + optimalLambda))

# (c) Calculate predictions based on
# final_holdout_test dataset and assess its RMSE
method4Predictions <- final_holdout_test %>%
  left_join(movieBiasLookUp, by = "movieId") %>%
  left_join(userBiasLookUp, by = "userId") %>%
  mutate(tailoredPrediction = method1Predictions_Mean
```

```r
        + movieBias + userBias) %>%
  pull(tailoredPrediction)

# (4) Calculate RMSE of method 4 prediction
rmse4 <- RMSE(final_holdout_test$rating, method4Predictions)
rmse4
```

```
## [1] 0.864817
```

Summary

Here is summary of all RMSE for methods used in machine learning section. We can observe gradual increase in algorithm accuracy as we add more granularity to account for statistical variations.

```r
# Summary of RMSE across ML methods
rmseSummary <- tibble(
  Method = c("Method 1: Mean", "Method 2: Movie Effect"
             , "Method 3: Movie & User Effects"
             , "Method 4: Regularized Movie & User Effects"),
  RMSE = c(rmse1, rmse2, rmse3, rmse4)
)

kable(rmseSummary, caption = "RMSE Comparison of Different Methods")
```

Table 1: RMSE Comparison of Different Methods

Method	RMSE
Method 1: Mean	1.0612018
Method 2: Movie Effect	0.9439087
Method 3: Movie & User Effects	0.8653488
Method 4: Regularized Movie & User Effects	0.8648170