MovieLens

Sowmiya

6/17/2019

Introduction

This project is a movie recommendation system using the movielens dataset which contains millions of movie ratings by users. The insights from this analysis are used to generate predictions of movies which are compared with the actual ratings to check the quality of the prediction algorithm.

Summary

The report is split in three sections. 1) Analysis after the data set is loaded. 2) Exploratory datanalysis after the data is partitioned from edx data set. 3) Machinelearning algorithm creates predictions which are then exported for a final test. A 'Penalized Root Mean Squared Error' approach was one of the few algorithm. is algorithm achieved a RMSE of 0.86 and an accuracy of 36.05% on the validation set.

Method

Due to the large dataset, an efficient method was needed to predict movie ratings based on an user id and a movie id. The penalized least squares approach is based on the mean movie rating. This average is adjusted for user-effects and movie-effects. Analysis shows that ratings from users who rate just a few movies and movies with a small number of total ratings tend to have more volatile ratings than users who rate lots of movies and movies with lots of ratings. To adjust for these effects, a penalty - lambda - is taken into account. Onve a prediction is made, it has to be translated from a continuous number into a number from 0.5 to 5.0 in 0.5 steps. The so derived predicted values get compared with the actual valuse to calculate an accuracy value.

Download MovieLens Data

```
if(!require(tidyverse)) install.packages("tidyverse", repos ="http://cran.us.
r-project.org")

## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------
------- tidyverse 1.2.1 --

## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts -------------------------------------------------------
- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org")

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

 # MovieLens 10M dataset:
  # https://grouplens.org/datasets/movielens/10m/
  # http://files.grouplens.org/datasets/movielens/ml-10m.zip

# MovieLens 10M dataset:
  # https://grouplens.org/datasets/movielens/10m/
  # http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip",dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K
/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp
"))
library(stringr)
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\:
:", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieI
d))[movieId],title = as.character(title),genres = as.character(genres))

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

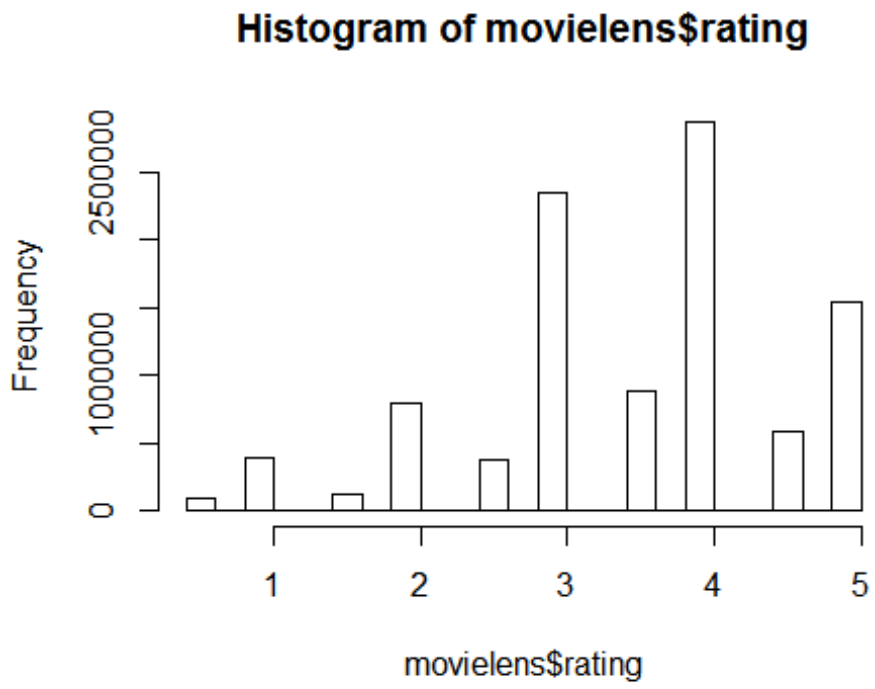
Data analysis of Movielens

```
str(movielens)

## 'data.frame':    10000054 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
```

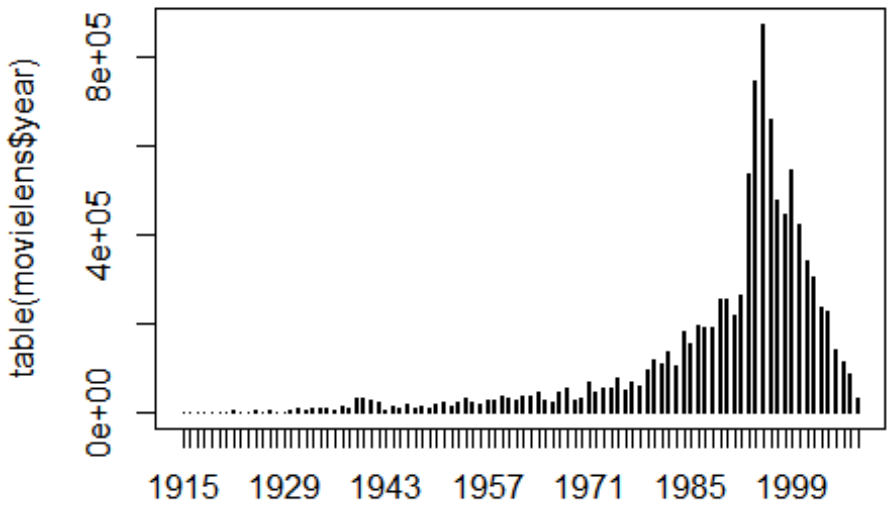
```
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 83898
3392 838984474 838983653 838984885 838983707 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (19
94)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Acti
on|Drama|Sci-Fi|Thriller" ...
```

The movielens dataset has more than 10 million ratings. Each rating comes with a userId, a movieId, the rating, a timestamp and information about the movie like title and genre.

```
hist(movielens$rating)
```

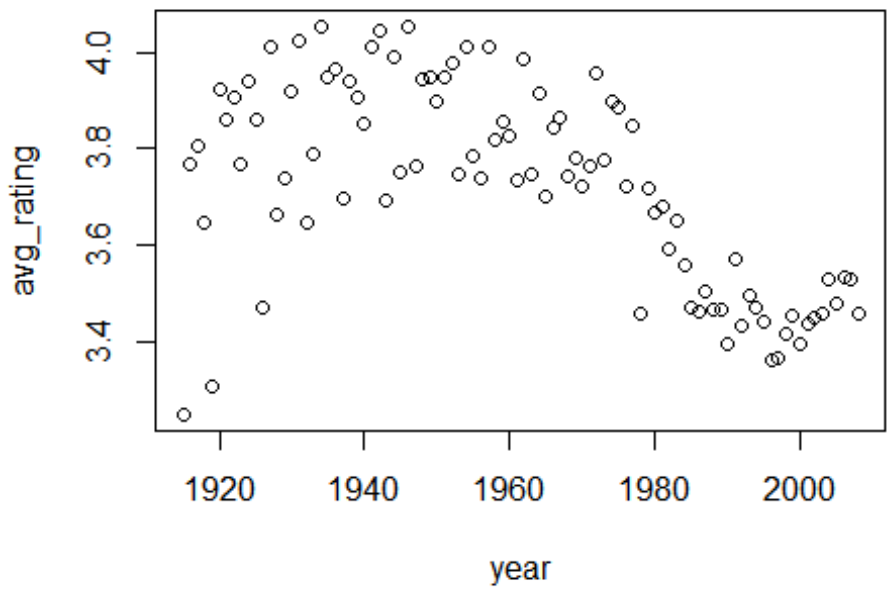



```
summary(movielens$rating)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000

movielens$year <- as.numeric(substr(as.character(movielens$title),nchar(as.ch
aracter(movielens$title))-4,nchar(as.character(movielens$title))-1))

plot(table(movielens$year))
```



More recent movies get more userratings. Movies earlier than 1930 get few ratings, whereas newer movies, especially in the 90s get far more ratings.

```
avg_ratings <- movielens %>% group_by(year) %>% summarise(avg_rating = mean(r
ating))
plot(avg_ratings)
```



Movies from earlier decades have more volatile ratings, which can be explained by the lower frequence of movie ratings.

Validation set will be 10% of MovieLens data

```
set.seed(1)
  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 validation set are also in edx set

  validation <- temp %>%
    semi_join(edx, by = "movieId") %>%
    semi_join(edx, by = "userId")

  # Add rows removed from validation set back into edx set

  removed <- anti_join(temp, validation)

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genr
es", "year")

  edx <- rbind(edx, removed)

  # Learners will develop their algorithms on the edx set
```

```
  # For grading, learners will run algorithm on validation set to generate ra
tings

  validation <- validation %>% select(-rating)
```

Exploratory Data Analysis on edx data set

```
paste('The edx dataset has',nrow(edx),'rows and',ncol(edx),'columns.')

## [1] "The edx dataset has 9000061 rows and 7 columns."

edx %>% summarize(n_movies = n_distinct(movieId))

##   n_movies
## 1    10677
```

There are 10677 movies

```
edx %>% summarize(n_users = n_distinct(userId))

##   n_users
## 1   69878
```

There are 69878 users.

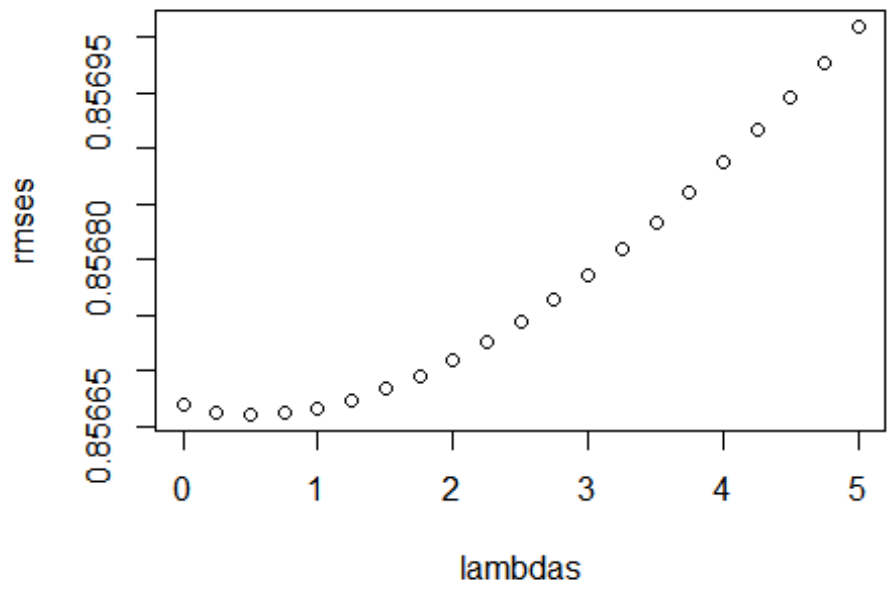
```
drama <- edx %>% filter(str_detect(genres,"Drama"))
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
thriller <- edx %>% filter(str_detect(genres,"Thriller"))
romance <- edx %>% filter(str_detect(genres,"Romance"))

paste('Drama has',nrow(drama),'movies')

## [1] "Drama has 3909401 movies"

paste('Comedy has',nrow(comedy),'movies')

## [1] "Comedy has 3541284 movies"

paste('Thriller has',nrow(thriller),'movies')

## [1] "Thriller has 2325349 movies"

paste('Romance has',nrow(romance),'movies')

## [1] "Romance has 1712232 movies"
```

Results

I have used penealized least squares machine Learning algorithm ti get the best possible rate.

Choose Optimal Penalty Rate 'Lambda'

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

lambdas <- seq(0, 5, 0.25)

rmses <- sapply(lambdas,function(l){

  #Calculate the mean of ratings from the edx training set
  mu <- mean(edx$rating)

  #Adjust mean by movie effect and penalize low number on ratings
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  #ajdust mean by user and movie effect and penalize low number of ratings
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  #predict ratings in the training set to derive optimal penalty value 'lambd
a'
  predicted_ratings <-
    edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred

  return(RMSE(predicted_ratings, edx$rating))
})

plot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
paste('Optimal RMSE of',min(rmses),'is achieved with Lambda',lambda)

## [1] "Optimal RMSE of 0.856661136127576 is achieved with Lambda 0.5"
```

The minimal RMSE of 0.856695227644159 is achieved with Lambda 0.5. So we will use the same for prediction.

Apply Lamda on Validation set

```
lambda <- 0.5

pred_y_lse <- sapply(lambda,function(l){

  #Derive the mearn from the training set
  mu <- mean(edx$rating)

  #Calculate movie effect with optimal lambda
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  #Calculate user effect with optimal lambda
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
```

```
  #Predict ratings on validation set
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred #validation

  return(predicted_ratings)

})
```

Export Prediction

```
write.csv(validation %>% select(userId, movieId) %>% mutate(rating = pred_y_l
se), "submission.csv", na = "" , row.names = FALSE)
```

Conclusion

Hence Penalized least squares machine learning algorithm provides the rating close to the original rating.