MovieLens - EDX project

Abhisekh

16/11/2022

##Introduction Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

##Uses of ML Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.

##Aim The aim of this project is to create a movie recommendation system using 10M movielens dataset.

##creating a train set and test set using 10M Movielens datset.

```
##########################################################

# Create edx set, validation set (final hold-out test set)

##########################################################

# Note: this process could take a couple of minutes

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

library(tidyverse)
library(caret)
library(data.table)
```

MovieLens 10M dataset are downloaded using below mentioned links:

https://grouplens.org/datasets/movielens/10m/

http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
options(timeout=500)
dl <- tempfile()
download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

```r
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")


# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


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

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `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)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

#lets the consider the ratings are same for all the movies regardless of the user

```r
##################################
# First Model =  simple averaging
##################################

# calculate overall average rating on the training dataset
mu <- mean(edx$rating)

# predict all unknown ratings with mu and calculate the RMSE
first_model = RMSE(validation$rating, mu)
first_model
```

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

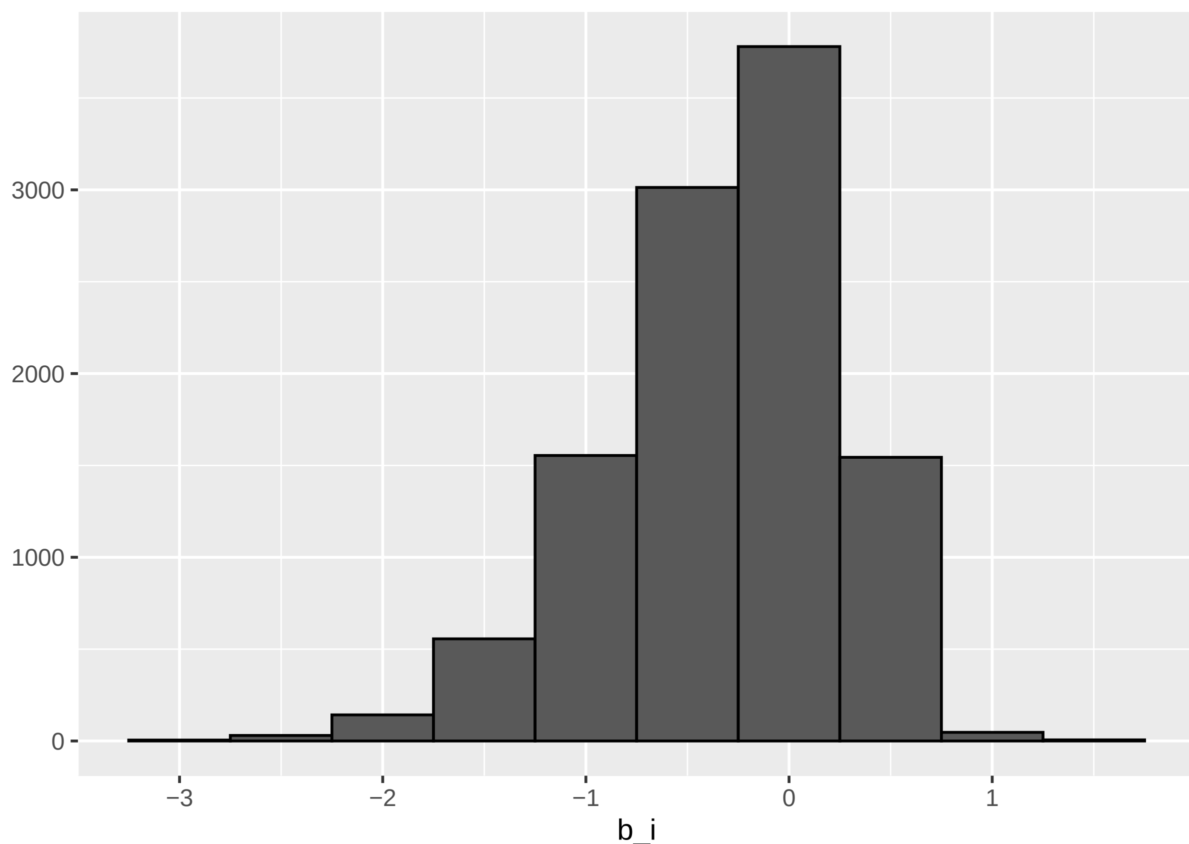
##now this is a simple average, lets consider other effects for ex: Movies

```r
######################
# Second model   = consider Movie effects
######################
```

```r
# add an average ranking for movie i, b_i
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# predict all unknown ratings with mu and b_i
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)

# calculate RMSE of movie ranking effect
RMSE(validation$rating, predicted_ratings)
```

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

Now lets plot the distribution of b_i's and we can see that how these estimates varies substantially



#after movie effects, lets consider user effects

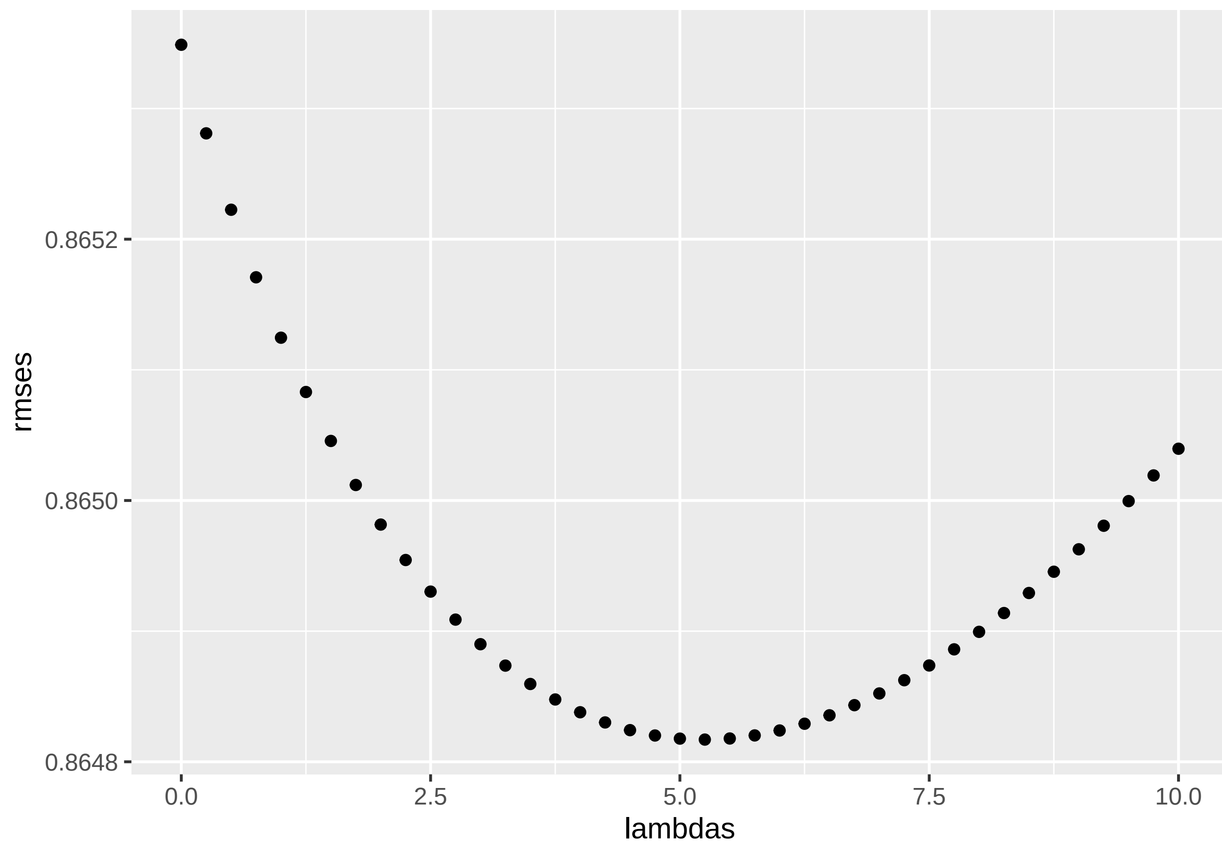
```r
################################
# Third model = consider user effect
################################

# compute average rating for user , b_u
b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# predict new ratings with movie and user bias
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# calculate RMSE of movie ranking effect
RMSE(predicted_ratings, validation$rating)
```

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

#lets regularise movie and user effects

```r
#############################################################
# Fourth Model = consider regularizing movie and user effects
#############################################################

# determine best lambda
lambdas <- seq(from=0, to=10, by=0.25)

# RMSE output of each lambda, (repeating with regularization)
rmses <- sapply(lambdas, function(l){
  # calculate average rating
  mu <- mean(edx$rating)
  # compute regularized movie effects
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  # compute user effect (regularization)
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  # compute predictions 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) %>%
    pull(pred)
  # output RMSE of these predictions
  return(RMSE(predicted_ratings, validation$rating))
```

```r
})

# plot of RMSE vs lambdas
qplot(lambdas, rmses)
```




```r
# print least RMSE
min(rmses)
```

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

#finally we have the final model and an ML algorithm

```r
######################################################
# Final model = regularized movie and user effects
######################################################

# The final linear model with least lambda
lam <- lambdas[which.min(rmses)]

b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lam))

# compute regularized user effect
```

```r
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lam))

# compute predictions 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) %>%
  pull(pred)

# output RMSE of these predictions
RMSE(predicted_ratings, validation$rating)
```

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

#RMSE improves as we regularised effect and get the desired the RMSE #An important note is to study the data set we are working on and understand the variability and effects of other parameters and how this affect our data.