Data Science Capstone - MovieLens Project

Author: Raven Houck

3/24/2019

Introduction

This report is part of the "HarvardX: PH125.9x Data Science: Capstone" course. The main goal of this project is to use all acquired skills that have been learned throughout the Data Science Professional Certificate program, and to apply them to a real-world problem. The given task is to use machine learning techniques to create a movie recommendation system based on predicted movie ratings. The given task will be accomplished using the MovieLens dataset – a set of Big Data that includes over ten million movie ratings. A machine learning algorithm will be trained using the inputs in one subset to predict movie ratings in the validation set. The movie rating predictions will then be compared to the true ratings in the validation set using Root Mean Squared Error (RMSE).

Methods and Analysis

A RMSE approach will be used to create this algorithm, since it is based on the mean movie rating. However, this value will be adjusted for based on two varying effects. First, the movie-affect variable must be accounted for. This is based on the idea that in general, some movies are just rated higher than others. The user-affect variable will also be accounted for, due to the substantial variability across users who rate movies. Once both affects are implemented into the algorithm, it will be used to create a movie recommendation system through the resulting predicted movie ratings. The movie rating predictions will then be compared to the true ratings in the validation set using RMSE.

Creating Test and Validation Sets

In order to develop a machine learning algorithm, the MovieLens data must first be downloaded and split into training and test sets.

```
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
"))

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")

# Validation set will be 10% of MovieLens data
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = 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")

edx <- rbind(edx, removed)

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

Data Exploration

This section of the report explores the data in order to get familiarized with it.

```
#What are the dimensions of the edx dataset?
dim(edx)

## [1] 9000055       6
```

The dataset has 9000055 rows and 6 columns.

```
#How many zeros and threes are there in the edx dataset?
edx %>% filter(rating == 3) %>% tally()

##         n
## 1 2121240
```

There are no movies that have a rating of zero – this is because the movies are rated from 0 to 5 in increments of 0.5. However, there are 2121240 threes given as ratings.

```
#How many different movies are in the edx dataset?
edx %>% summarize(n_movies=n_distinct(movieId))
```

```
##   n_movies
## 1    10677
```

There are 10677 unique movies in the dataset.

```
#How many different users are in the edx dataset?
edx %>% summarize(n_users=n_distinct(userId))

##   n_users
## 1   69878
```

There are 69878 unique users in the dataset.

```
#How many movie ratings are there for drama, comedy, thriller, and romance ge
nres?
edx %>% separate_rows(genres, sep = "\\|") %>%
    group_by(genres) %>%
    summarize(count = n()) %>%
    arrange(desc(count))

## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                 8181
## 20 (no genres listed)      7
```

There are 3910127 drama movie ratings, 3540930 comedy movie ratings, 2325899 thriller movie ratings, and 1712100 romance movie ratings in the dataset.

```
#Which movie group has the highest number of ratings?
edx %>% group_by(title) %>% summarize(number = n()) %>% arrange(desc(number))

## # A tibble: 10,676 x 2
##    title                                                     number
```

```
##    <chr>                                                          <int>
##  1 Pulp Fiction (1994)                                            31362
##  2 Forrest Gump (1994)                                            31079
##  3 Silence of the Lambs, The (1991)                               30382
##  4 Jurassic Park (1993)                                           29360
##  5 Shawshank Redemption, The (1994)                               28015
##  6 Braveheart (1995)                                              26212
##  7 Fugitive, The (1993)                                           25998
##  8 Terminator 2: Judgment Day (1991)                              25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   25672
## 10 Apollo 13 (1995)                                               24284
## # ... with 10,666 more rows
```

Pulp Fiction has the greatest number of ratings.

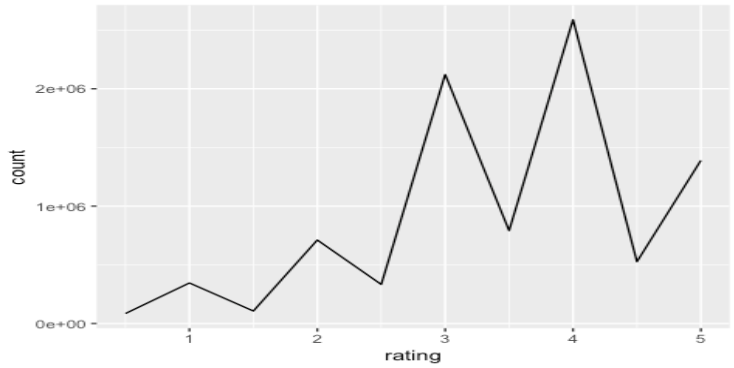
```
#What are the five most given ratings?
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>% arrange(
desc(count))

## Selecting by count

## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

The five most given ratings are 4, 3, 5, 3.5, and 2.

```
#Are half-star ratings less common than whole star ratings?
edx %>%
    group_by(rating) %>%
    summarize(count = n()) %>%
    ggplot(aes(x = rating, y = count)) +
    geom_line()
```



This plot shows that half-star ratings are less common than whole star ratings.

Results

This machine learning algorithm is based on the mean movie rating, but also takes into account the movie-and-user-effect. This was done using a root mean square approach.

Developing the Algorithm

First, the training set – labelled as the edx set – was used to develop the machine learning algorithm.

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

lambda <- seq(0,5,0.5)

rmse <- sapply(lambda, function(x){

#Begin by determining the mean movie rating from the edx set
mu_hat <- mean(edx$rating)

#Now implement a Movie Effect model that takes this variability into account
movie_avgs <- edx %>% group_by(movieId) %>% summarize(movie_avgs = mean(ratin
g - mu_hat))

#Now take the use variability into account
user_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>%
group_by(userId) %>% summarize(user_avgs = mean(rating - mu_hat - movie_avgs)
)

#Now we determine the predicted ratings based on the edx set
predicted_ratings <- edx %>% left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    mutate(pred = mu_hat + movie_avgs + user_avgs) %>%
    pull(pred)

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

This ultimately results in the following minimum RMSE

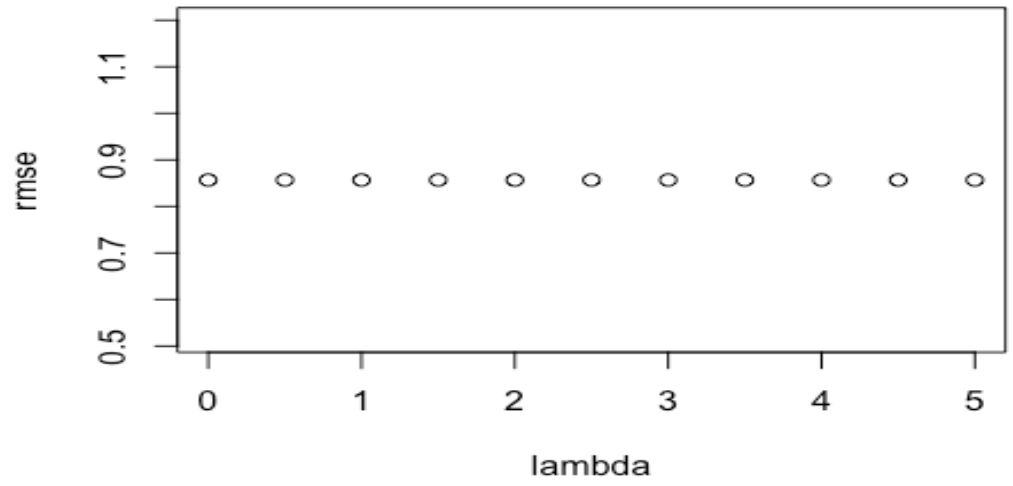
```
min(rmse)

## [1] 0.8567039
```

A minimum rmse of 0.857 was achieved using the Movie & User Variability Method

It is also important to consider the effect of lambda on the RMSE

```
plot(lambda,rmse)
```



Due to the insignificant effect of lambda on rmse, predictions will be made with lambda=0.5 for simplicity

Predicting Movie Ratings

Next, the developed algorithm is tested by predicting the movie ratings in the validation set as if they were unknown. The RMSE is used to evaluate how close the predictions are to the true values.

```
#Lambda will be set to 0.5
lambda <- 0.5

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

#mu is set to be the mean of the training set
mu <- mean(edx$rating)

#Predict the movie effect with lambda
pred_movie_avgs <- edx %>% group_by(movieId) %>% summarize(pred_movie_avgs =
mean(rating - mu))

#Predict the user effect with lambda
pred_user_avgs <- edx %>% left_join(pred_movie_avgs, by='movieId') %>%
group_by(userId) %>% summarize(pred_user_avgs = mean(rating - mu - pred_movie
_avgs))
```

```
#Predict ratings on validation set
predicted_ratings <- validation %>%
    left_join(pred_movie_avgs, by = "movieId") %>%
    left_join(pred_user_avgs, by = "userId") %>%
    mutate(pred = mu + pred_movie_avgs + pred_user_avgs) %>%
    pull(pred)

  return(predicted_ratings)})

#These predicted ratings will be documented on the following .csv file
write.csv(validation %>% select(userId, movieId) %>% mutate(rating = predicti
on),
          "movielens_prediction_submission.csv", na = "", row.names=FALSE)
```

Conclusion

The main goal of this project was successfully achieved by creating a machine learning algorithm that can be used to predict movie ratings from the MovieLens dataset. By comparing the movie rating predictions to the true ratings in the validation set, an RMSE value of 0.857 was achieved.