Movielens Capstone Project

Michelle Salandy

6/3/2019

Introduction

This project is based on developing a movie recommendation system using the MovieLens dataset. Several factors were explored as predictors for determining the prediction of movie ratings. The Root Mean Squared Error (RMSE) was utilized to evaluate performance and determine the best machine learning algorithm for this task.

The impact of the user, movie and the genre of a movie, on movie ratings in the dataset were explored but only the algorithm with user and movie effects was chosen. Additionally, after data exploration, Regularization was introduced in the model to *penalize* large estimates in ratings for some movies being derived from a small number of total ratings, as well as being rated by only a few users. Thus, shrinking deviations from the average toward zero and improving the generalization abilities of the model.

The project has five sections. Following this brief introduction, the second section will identify the dataset and illustrate how 90% of the data is used as the training set and the remaining 10% the validation set. The third section will discuss the general properties of the dataset. The fourth section presents a detailed discussion on the penalized approach of the chosen algorithm. The report ends with a summary of the major findings and the identification of the RMSE of the "*best-fit*" algorithm.

Dataset: Downloading and Creating Test and Validation Sets

The following code was used to generate the datasets. The dataset was then split into training set denoted as edx and a validation set denoted as validation, which is used to test the algorithm for predicting movie ratings as if they were unknown.

MovieLens 10M dataset:

```
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")
```

```
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(movieId))[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, 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", "genres")
```

```
edx <- rbind(edx, removed)

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

Data Exploration: edx dataset

The general properties of the edx dataset is explored below.

1. There are 9000055 rows and 6 columns.

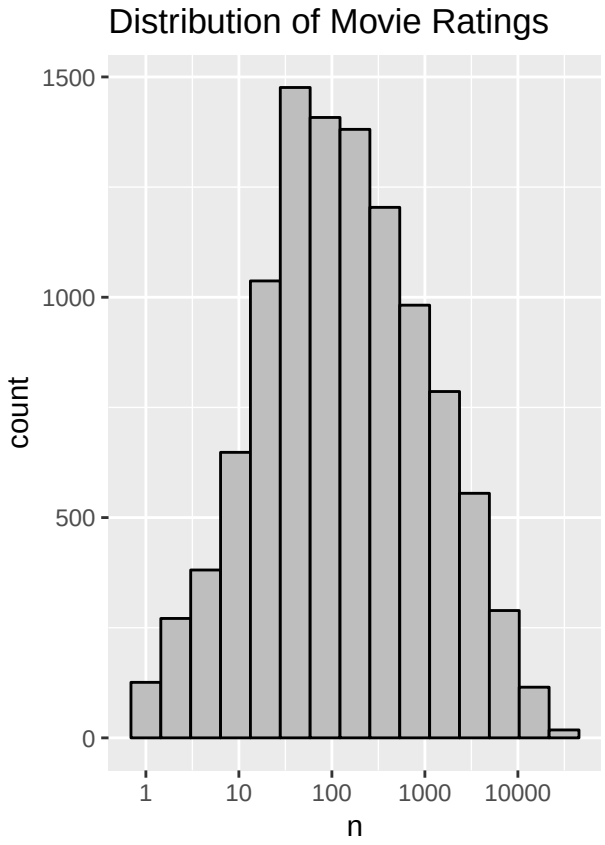
```
dim(edx)
```

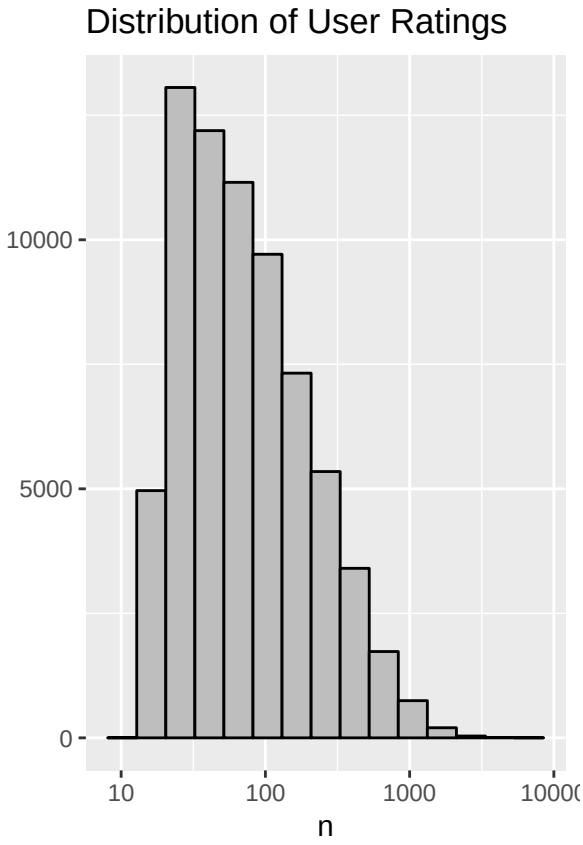
2. There are 10677 unique movies, 69878 unique users, unique genre categories and 10 unique ratings ranging from 0.5 to 5.

```
##   Movies Users Genres Rating
## 1  10677 69878    797     10
```

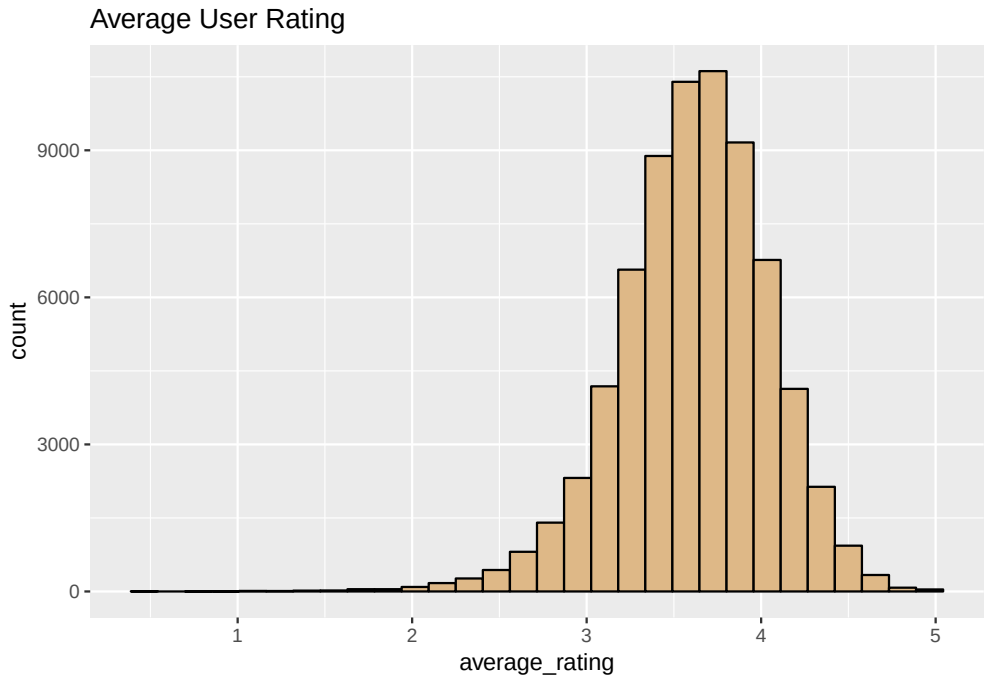
```
range(edx$rating)
```

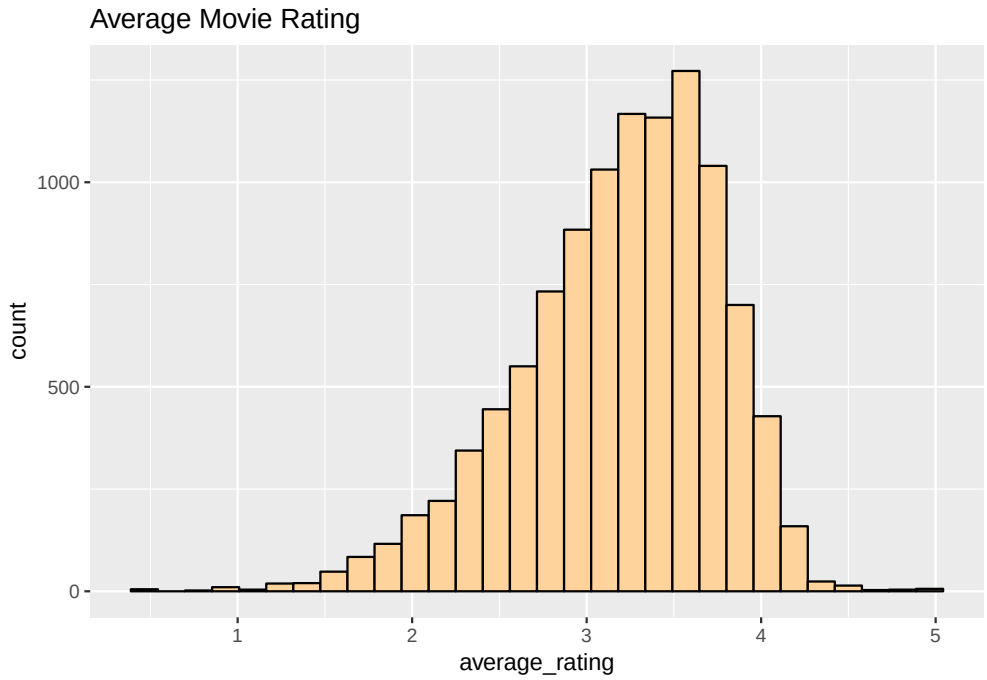
3.The distribution of the number of movie ratings and the number of user ratings is displayed below. The figure shows that some movies are rated more often than others. This variation is also seen as some users also actively rate more movies. Some users have rated more than 1000 movies while others have only rated a few.





4.Substantial variablity also exists in the average rating across users and movies. Some users give higher average ratings than others.



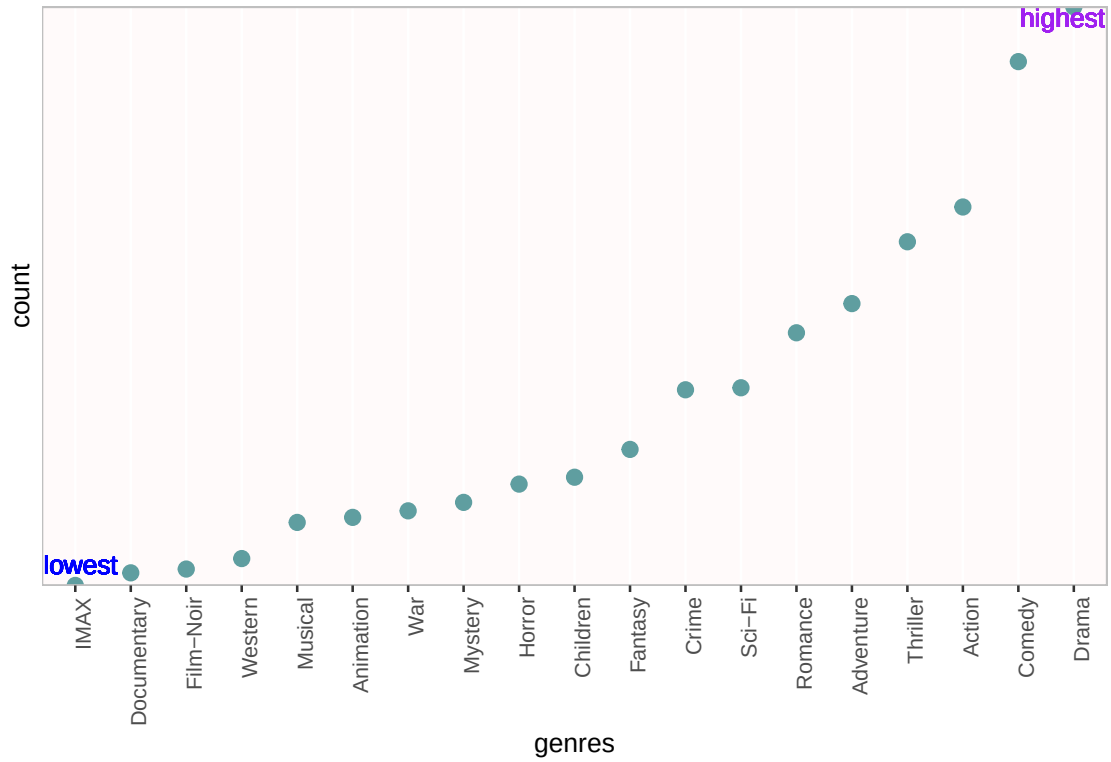


Average ratings are also higher for some more popular movies that others.

5. Preferences in genre type can also be assumed as there are various numbers of movie ratings in each of the following genre categories. Ranging from a minimum of 2 ratings to max of 733296 ratings in the Drama Category.

```
range((as.data.frame(genres))$count)
```

6. There are various numbers of movie ratings in each of the following individual genre categories. Drama receives the most ratings and IMAX the least.

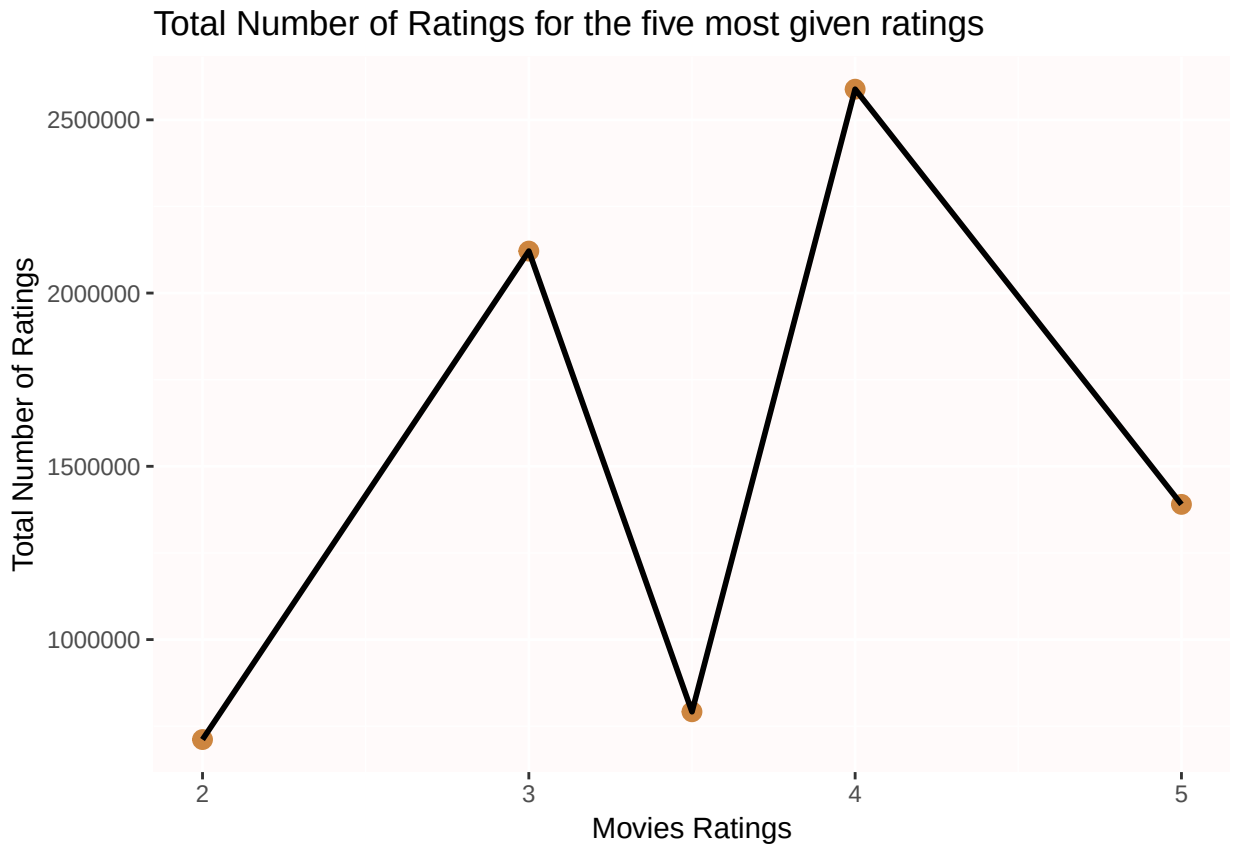


7. Pulp Fiction has the greatest number of ratings.

```
## # A tibble: 10 x 3
##    movieId     n title
##      <dbl> <int> <chr>
##  1     296 31362 Pulp Fiction (1994)
##  2     356 31079 Forrest Gump (1994)
##  3     593 30382 Silence of the Lambs, The (1991)
##  4     480 29360 Jurassic Park (1993)
##  5     318 28015 Shawshank Redemption, The (1994)
##  6     110 26212 Braveheart (1995)
##  7     457 25998 Fugitive, The (1993)
##  8     589 25984 Terminator 2: Judgment Day (1991)
##  9     260 25672 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1~
## 10     150 24284 Apollo 13 (1995)
```

8. Its also observed that the five most given ratings in order from most to least are 4, 3, 5, 3.5, 2.

```
##   rating       n
## 1    4.0 2588430
## 2    3.0 2121240
## 3    5.0 1390114
## 4    3.5  791624
## 5    2.0  711422
```



Discussion: Regularization of the Algorithmn and RMSE results

Using *Penalized* least squares estimates to determine the best algorithm to determine ratings based on users and movieId.

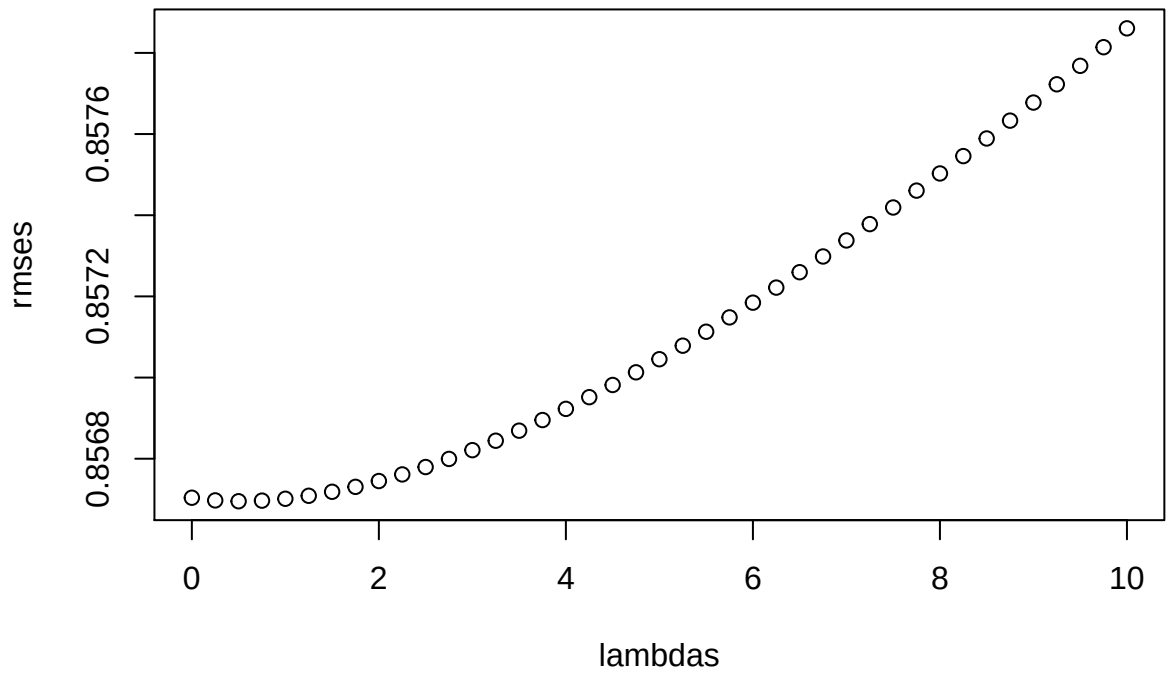
Computing the RMSE for a vector of ratings and their corresponding predictors

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

Methodology:Determining optimal value for lambda

Cross-validation is used to choose the tuning parameter lambda.

```
lambdas<-seq(0,10,0.25)
rmses<-sapply(lambdas, function(l){
  mu<-mean(edx$rating)

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

  bu<-edx %>% left_join(bi, by="movieId") %>%
    group_by(userId) %>%
    summarize(bu=sum(rating-bi-mu)/(n()+l))

  yhat<-edx %>% left_join(bi, by="movieId") %>%
    left_join(bu, by="userId") %>%
    mutate(pred=mu+bi+bu) %>%
    .$pred

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

lambda<-lambdas[which.min(rmses)]

plot(lambdas,rmses)
```



Cross validation on the training set with suggests that lambda of 0.5 will minimize the RMSE.

Methodology:Computing the lowest RMSE

Using lambda (l) of 0.5, the code below shows how the RMSE was calculated from the validation set.

```
l<-0.5
mu<-mean(validation$rating)

bi<-validation %>% group_by(movieId) %>%
summarize(bi=sum(rating-mu)/(n()+l))

bu<-validation %>% left_join(bi, by="movieId") %>%
    group_by(userId) %>%
    summarize(bu=sum(rating-bi-mu)/(n()+l))

yhat<-validation %>% left_join(bi, by="movieId") %>%
    left_join(bu, by="userId") %>%
    mutate(pred=mu+bi+bu) %>%
    .$pred

RMSE(yhat, validation$rating)
```

```
## [1] 0.8258487
```

Conclusion

This algorithm achieved a RMSE of 0.83. This is lowest RSME reflecting the closet predictions to the true values of ratings in the validation set from the various algorithms utilized (algorithms which provided lower RMSEs were not included in the document).