Contents

MovieLens Project – Sarat Sarngadharan

Introduction

Overview

MovieLens (http://www.movielens.org/) is a research site that is maintained by University of Minnesota. This site makes recommendations of movies to users that they might enjoy and generates a prediction of movies that a user would enjoy. The website allows users to rate a particular film from 1 – 5 stars with an increment of 0.5 stars. For this project, we would be creating a movie recommendation system using the [10M version of the MovieLens]() dataset.

This project report elaborates how to analyze the existing movie ratings and train a recommendation ML algorithm to predict the user ratings for a movie. RMSE (Root Mean Squared Error) is used to evaluate the accuracy of the predictions to the true values in the validation set.

Dataset and Initial exploration

The 10M version of the MovieLens dataset contains

- 9000055 rows
- 5 different columns (*userId, movieId, rating, timestamp,title and genres*)
- Ratings from 0.5 – 5
- 71567 users

```
> 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
```

[Distinct Users and Movies]()

In this dataset there 10677 distinct movies and 69878 distinct userids

```
n_distinct(edx$movieId)
[1] 10677

> n_distinct(edx$userId)
[1] 69878
```

Ratings

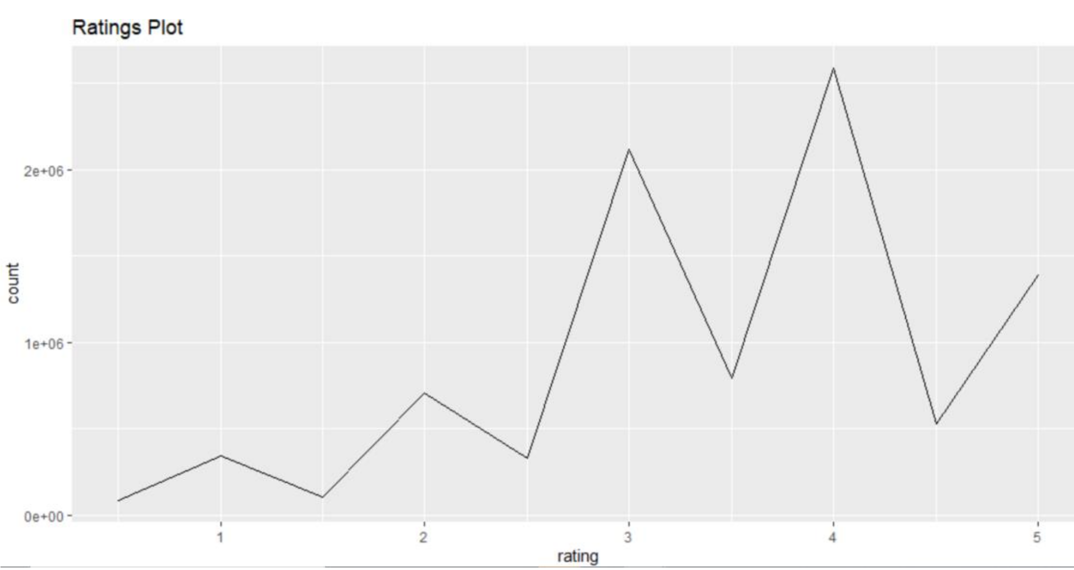
The most common ratings that was given to a movie is 4 (count = 2588430) followed by 3 (count = 2121240) with a mean rating of 3.512477

```
> #Ratings
> edx %>% separate_rows(rating, sep = "\\|") %>%
+   group_by(rating) %>%
+   summarize(count = n()) %>%
+   arrange(desc(count))
# A tibble: 10 x 2
   rating   count
   <chr>    <int>
 1 4      2588430
 2 3      2121240
 3 5      1390114
 4 3.5     791624
 5 2       711422
 6 4.5     526736
 7 1       345679
 8 2.5     333010
 9 1.5     106426
10 0.5      85374
```

```
 > mean(edx$rating)
 [1] 3.512477

 >
```

```
> edx %>%
+   group_by(rating) %>%
+   summarize(count = n()) %>%
+   ggplot(aes(x = rating, y = count)) +
+   geom_line()+
+   ggtitle("Ratings Plot")
```



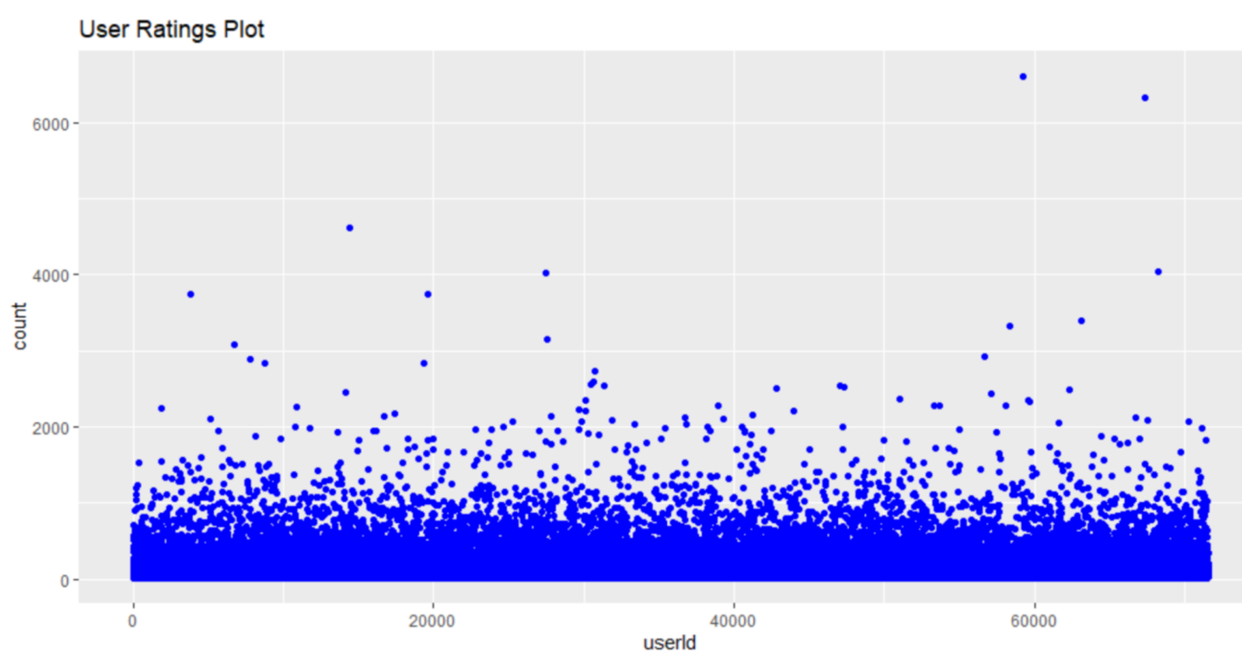
The Top ten rated movies are shown below with Pulp Fiction rated the top.

```
> # Highest Rated Movies
>
> edx %>% group_by(movieId, title) %>%
+   summarize(count = n()) %>%
+   arrange(desc(count))
# A tibble: 10,677 x 3
# Groups:   movieId [10,677]
   movieId title                                                          count
     <dbl> <chr>                                                          <int>
 1     296 Pulp Fiction (1994)                                            31362
 2     356 Forrest Gump (1994)                                            31079
 3     593 Silence of the Lambs, The (1991)                               30382
 4     480 Jurassic Park (1993)                                           29360
 5     318 Shawshank Redemption, The (1994)                               28015
 6     110 Braveheart (1995)                                              26212
 7     457 Fugitive, The (1993)                                           25998
 8     589 Terminator 2: Judgment Day (1991)                              25984
 9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   25672
10     150 Apollo 13 (1995)                                               24284
```

Users & Rating Behavior

Most of the user rated under 100 movies while there are few users who rated more than 2000 movies.

```
> edx %>% group_by(userId) %>%
+   summarize(count = n()) %>%
+   ggplot(aes(x = userId, y = count)) +
+   geom_point(color="blue")+
+   ggtitle("User Ratings Plot")
```



Genre

Drama and Comedy are the rated most by the sample users

```
> # Genres
>
> 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

>
```

Analysis

RMSE

Root Mean Square Error (RMSE) is commonly used in regression analysis to verify experimental results. It is the standard deviation of the residuals (prediction errors) and can be calculated as below

```
# RMSE
RMSE <- function(tr_ratings, pr_ratings){
  sqrt(mean((tr_ratings - pr_ratings)^2))
}
```

tr_ratings = true ratings

pr_ratings= predicted ratings

If RMSE is greater than0.8775, it indicates that our error is almost by a star, which is not good.

Prediction using Mean Rating alone

In this prediction model, we use the mean of the dataset to predict the ratings for all movies and any difference is attributed to a random error

Yui = mu + Eui (Eui – Independent Error, mu=Actual ratings of the movie, Yui=Predicted value)

```
> # RMSE
> RMSE <- function(tr_ratings, pr_ratings){
+   sqrt(mean((tr_ratings - pr_ratings)^2))
+ }
> #Analysis
> # RMSE
> RMSE <- function(tr_ratings, pr_ratings){
+   sqrt(mean((tr_ratings - pr_ratings)^2))
+ }
>
> ## Simple Prediction on mean alone.
> mu <- mean(edx$rating)
> mu
[1] 3.512465
>
> rmse_mean_alone <- RMSE(validation$rating, mu)
> rmse_mean_alone
[1] 1.061202

>
```

As you can see the RMSE is > 1, it indicates that our error is off by more than a star, which is not good.

Prediction Considering the Movie Bias

If you analyze the dataset, you could see that some movies are rated highly and some movies are rated low consistently. To negate the movie effect in this model we consider the bias based on movies average rating and the average ratings of all movies in this dataset

```
> ## Considering the impact of movie
>
> movies_rating <- group_by(edx, title) %>%
+   summarize(n = n(), avg = mean(rating))
>
> movies_mean <- movies_rating  %>%
+   filter(n > 10) %>%
+   arrange(desc(avg), desc(n))
>
> top_mov <- movies_mean[1,]
>
> worst_mov <- movies_mean[nrow(movies_mean),]
>
> top_mov
# A tibble: 1 x 3
  title                                n   avg
  <chr>                            <int> <dbl>
1 Shawshank Redemption, The (1994) 28015  4.46
> worst_mov
# A tibble: 1 x 3
  title                                   n   avg
  <chr>                               <int> <dbl>
1 SuperBabies: Baby Geniuses 2 (2004)    56 0.795
```

```
>
```

Also we can see some movies are rated more number of times and there are some movies that are rated only few number of times

```
> # Some Movies are rated more and some a few rating
> more_ratd <- edx %>% group_by(movieId, title) %>%
+   summarize(count = n()) %>%
+   arrange(desc(count))
> more_ratd
# A tibble: 10,677 x 3
# Groups:   movieId [10,677]
   movieId title                                                        count
     <dbl> <chr>                                                        <int>
 1     296 Pulp Fiction (1994)                                          31362
 2     356 Forrest Gump (1994)                                          31079
 3     593 Silence of the Lambs, The (1991)                             30382
 4     480 Jurassic Park (1993)                                         29360
 5     318 Shawshank Redemption, The (1994)                             28015
 6     110 Braveheart (1995)                                            26212
 7     457 Fugitive, The (1993)                                         25998
 8     589 Terminator 2: Judgment Day (1991)                            25984
 9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
10     150 Apollo 13 (1995)                                             24284
# ... with 10,667 more rows
>
> less_ratd<- edx %>% group_by(movieId, title) %>%
+   summarize(count = n()) %>%
+   arrange((count))
> less_ratd
# A tibble: 10,677 x 3
# Groups:   movieId [10,677]
   movieId title                                                  count
     <dbl> <chr>                                                  <int>
 1    3191 Quarry, The (1998)                                         1
 2    3226 Hellhounds on My Trail (1999)                              1
 3    3234 Train Ride to Hollywood (1978)                             1
 4    3356 Condo Painting (2000)                                      1
 5    3383 Big Fella (1937)                                           1
 6    3561 Stacy's Knights (1982)                                     1
 7    3583 Black Tights (1-2-3-4 ou Les Collants noirs) (1960)        1
 8    4071 Dog Run (1996)                                             1
 9    4075 Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)          1
10    4820 Won't Anybody Listen? (2000)                               1
# ... with 10,667 more rows

>
```

To negate the movie effect in this model we consider the bias based on movies average rating and the average ratings of all movies in this dataset

Yui = mu + bias_ind_mov + Eui
(Yu,i is the prediction, mu the mean rating for all movies, and bias_ind_mov is the bias for each movie, Eui is the independent error)

```
Browse[1]> #Movie Bias
Browse[1]> mu <- mean(edx$rating)
Browse[1]> mu
[1] 3.512465
Browse[1]>
> mov_avg <- edx %>%
+   group_by(movieId) %>%
+   summarise(bias_ind_movie = mean(rating - mu))
>
> pr_ratings <- mu + validation %>%
+   left_join(mov_avg, by='movieId') %>%
+   pull(bias_ind_movie)
>
> rmse_movie_bias<- RMSE(pr_ratings, validation$rating)
> rmse_movie_bias
[1] 0.9439087

>
```

We could see that considering the individual movie bias improved the RMSE to 0.9439087

Prediction Considering the User Bias and Movie Bias

If you analyze the dataset you could see that there are generous users who consistently rate movies higher than the mean and some users who rate movies consistently lower than the mean. In this model we are trying to negate for the individual user bias

```
> #Generous / Critical User
> users_rating <- group_by(edx, userId) %>%
+   summarize(n = n(), avg_user_rating = mean(rating))
>
> users_ranking <- users_rating  %>%
+   arrange(desc(avg_user_rating))
>
> generous_user <- users_ranking[1,]
>
> critical_user <- users_ranking[nrow(users_ranking),]
>
> generous_user
# A tibble: 1 x 3
  userId     n avg_user_rating
   <int> <int>           <dbl>
1      1    19               5
> critical_user
# A tibble: 1 x 3
  userId     n avg_user_rating
   <int> <int>           <dbl>
1  63381    18             0.5
```

Generous user consistently rated 5 and critical user rated 0.5

Some users review a lot of movies compared to others

```
> #user who ranked more
> user_number_rating <- users_ranking  %>%
+   arrange(desc(n))
>
```

```
> most_ratings_user <- user_number_rating[1,]
>
> least_ratings_user  <- user_number_rating[nrow(user_number_rating),]
>
> most_ratings_user
# A tibble: 1 x 3
  userId     n avg_user_rating
   <int> <int>           <dbl>
1  59269  6616            3.26
> least_ratings_user
# A tibble: 1 x 3
  userId     n avg_user_rating
   <int> <int>           <dbl>
1  62516    10            2.25
```

To negate the movie effect in this model we consider the user bias

Yui = mu + bias_ind_mov + bias_user_rating+Eui
(Yu,i is the prediction, mu the mean rating for all movies, and bias_ind_mov is the bias for each movie, bias_user_rating=individual user bias, Eui is the independent error)

```
>

> # User Bias Model
>
> user_avg <- edx %>%
+   left_join(mov_avg, by="movieId") %>%
+   group_by(userId) %>%
+   summarise(bias_user_rating = mean(rating - mu - bias_ind_movie))
>
> pr_ratings <- validation %>%
+   left_join(mov_avg, by='movieId') %>%
+   left_join(user_avg, by='userId') %>%
+   mutate(pred_user = mu + bias_ind_movie + bias_user_rating) %>%
+   pull(pred_user)
>
> rmse_user_bias <- RMSE(pr_ratings, validation$rating)
> rmse_user_bias
[1] 0.8653488

>


>
```

We could see that considering the individual user bias too improved the RMSE to 0.8653488

Regularization

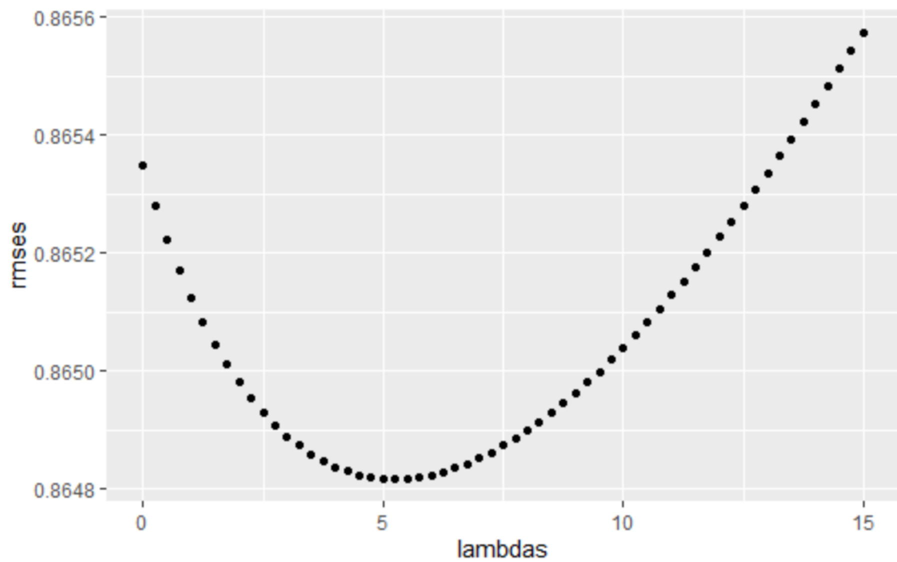
Regularization helps to reduce the error in prediction by using a tuning parameter by removing the outliers that may skew the results.

Yui = mu + bias_ind_movL + bias_user_ratingL+Eui
(Yu,i is the prediction, mu the mean rating for all movies, and bias_ind_movL is the bias for each movie, bias_user_ratingL=individual user bias, Eui is the independent error)

```
Browse[1]> ## Regularization
Browse[1]>
> lambdas <- seq(0, 15, 0.25)
> rmses <- sapply(lambdas, function(l){
+
+   mu <- mean(edx$rating)
+
+   bias_ind_movie <- edx %>%
+     group_by(movieId) %>%
+     summarise(bias_ind_movie = sum(rating - mu)/(n() +l))
+
+   bias_user_rating <- edx %>%
+     left_join(bias_ind_movie, by="movieId") %>%
+     group_by(userId) %>%
+     summarise(bias_user_rating = sum(rating - bias_ind_movie - mu)/(n()+l))
+
+   pr_ratings <- validation %>%
+     left_join(bias_ind_movie, by = "movieId") %>%
+     left_join(bias_user_rating, by = "userId") %>%
+     mutate(rate_pred = mu + bias_ind_movie + bias_user_rating) %>%
+     pull(rate_pred)
+
+   return(RMSE(pr_ratings, validation$rating))
+
+ })
> rmse_regul <- min(rmses)
> rmse_regul
[1] 0.864817

>
```

```
>
> # OptimalLambda
> qplot(lambdas, rmses)
> lambda <- lambdas[which.min(rmses)]
> lambda
[1] 5.25
```



Results and Discussion

The final values of the prediction models are shown below;

Model	RMSE
Prediction using Mean Rating alone	1.061202
Prediction Considering the Movie Bias	0.9439087
Prediction Considering the User Bias and Movie Bias	0.8653488
Regularization	0.864817

Regularized model and model that considered user / movie bias gave the least RMSEs among the prediction models that we analyzed.

Conclusion

A prediction model for ratings were created using Movie Lens dataset. In this project, that the optimum results were obtained when the impact of both user and movie bias were considered. This project was able to create a prediction model with RMSE <= 0.8649

Further analysis could be done on

Movie Genre impact on ratings: Drama and Comedy genres are the most rated by MovieLens users. There could be bias towards these Genres that we could evaluate further and confirm.

Environment Details

```
> #env
> print("Version Info")
[1] "Version Info"
> version
               _
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          5.3
year           2019
month          03
day            11
svn rev        76217
language       R
version.string R version 3.5.3 (2019-03-11)
nickname       Great Truth

>
```

References

- https://rafalab.github.io/dsbook/
- https://movielens.org/
- https://www.statisticshowto.datasciencecentral.com/rmse/