HarvardX: PH125.9x - MovieLens Project

Gerald Kandulu

May 22, 2020

1.0 Introduction

The aim of this project is to develop a machine learning algorithm for a movie recommendation system based on user movie ratings. The recommendation system is developed from the 10M version of the MovieLens dataset. In the dataset, a user rating of 0.5 indicate a poor movie while a 5.0 rating indicate an excellent movie. The dataset is split into 90:10 for training and validations sets. The Root Mean Squared Error (RMSE) method is used in the validation set to evaluate how close the predictions are to the true values.

The prediction algorithms used in the recommendation systems follows the model that was learnt in the HarvardX: Data Science Professional Certificate program, where user and movie effects are used for the model. Regularization was used to penalize large estimates that were formed using small sample sizes. While other approaches may have produced better estimates, choice of the algorithm in the recommendation system was based on project requirements which were, use tools that were shown in the program, and the system to be run on a standard student computer. The later restriction was introduced to allow running the scripts on unknown computer for peer grading. This allowed for the developed learning algorithms to be robust, scalable, and adaptable.

This report contains the data exploratory analysis, modeling and data analysis, results and concluding remarks.

2.0 Methods and analysis

Scripts were provided to download data from the GroupLens research lab website (http://files.grouplens.org/datasets/movielens/ml-10m.zip) and split the data set into 90:10 for training and validation sets. The training set is called edx which is used to develop the algorithm while the validation set is called validation which is used to final test the algorithm.

Helper Functions

As indicated in the introduction, RSME is used to validate the algorithm. Since this method is used when creating the model on test set and also evaluating the algorithm on the validation set, the below rsme R function is created.

```
RMSE <- function(true_rating, predicted_rating){
  #function computes the RSME between a vector of true rating and predicated ratings.
    sqrt(mean((true_rating - predicted_rating)^2))
}
```

2.1 Exploratory data analysis

The script to download and pre-process the data was provided as part of the course. Inspection of the data indicate that the data is in tidy format with about 9 million rows (9,000,055) and 6 columns with each row representing one user and one movie rating.

```
#view structure of edx dataset
edx %>% glimpse()
```

```
## Rows: 9,000,061
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 231, 292, 316, 329, 355, 356, 362, 364, 370, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumber (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy", "Act...
```

Furthermore, the data set also indicated have no null values.

```
#checking proportional of null values in training set
mean( is.na(edx))
```

```
## [1] 0
```

2.1.1 Users and movies

The code below indicate that the edx dataset has 69, 878 unique users, 10, 677 unique movies with an average movie rating of 3.5 with a standard deviation of 1.060331.

```
#summarise the data set to count distinct users and movies, and the mean and standard deviations
#opted this approach as opposed to using the generic R summary generic because was concerned of computa
edx %>% summarize(n_users = n_distinct(userId),
                  n_movies = n_distinct(movieId),
                  mean_rating = round(mean(rating), 1),
                  sd_rating = sd(rating)) %>% knitr::kable()
```

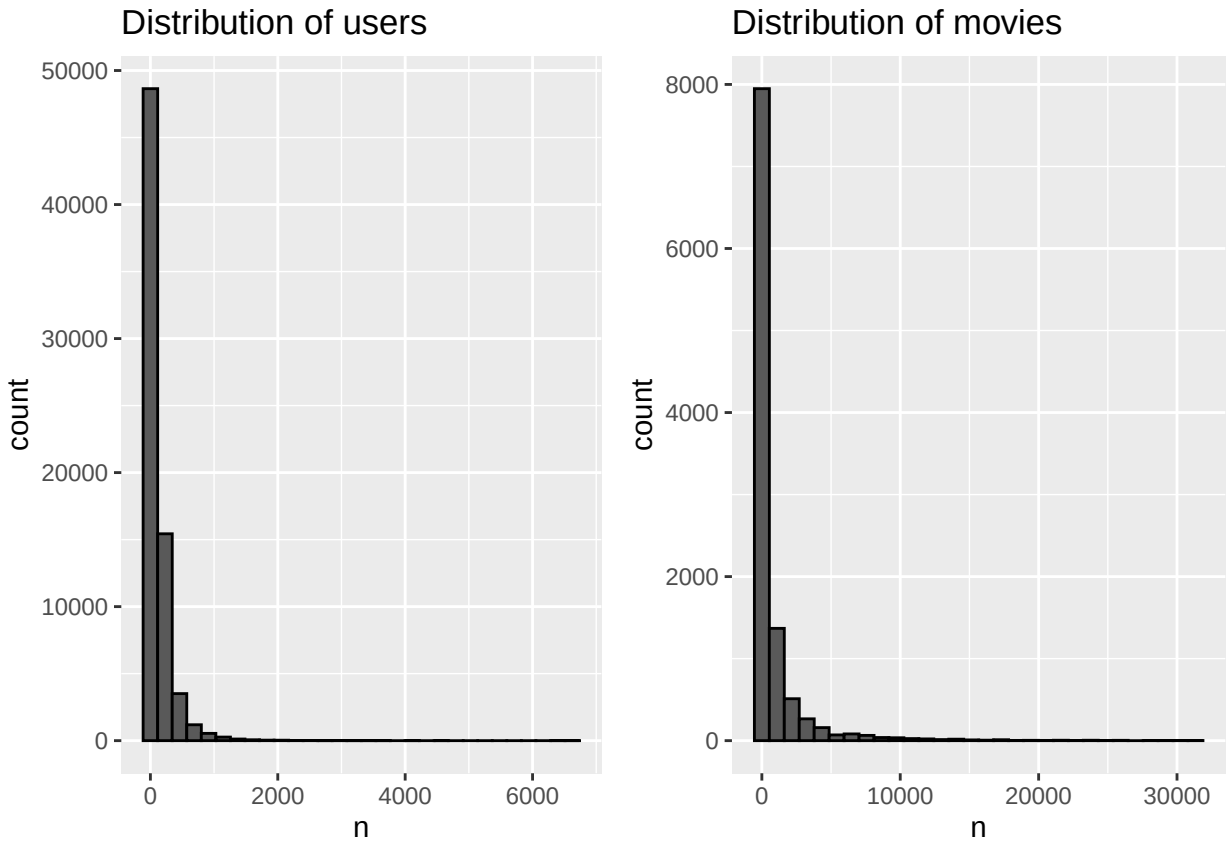
n_users	n_movies	mean_rating	sd_rating
69878	10677	3.5	1.060393

.

Plotting the user and movie distribution indicate that data is right skewed. This is expected because in general blockbuster or excellent movies tend to be watched by more users and will in turn have more rating than little known movies which are watched by few users.

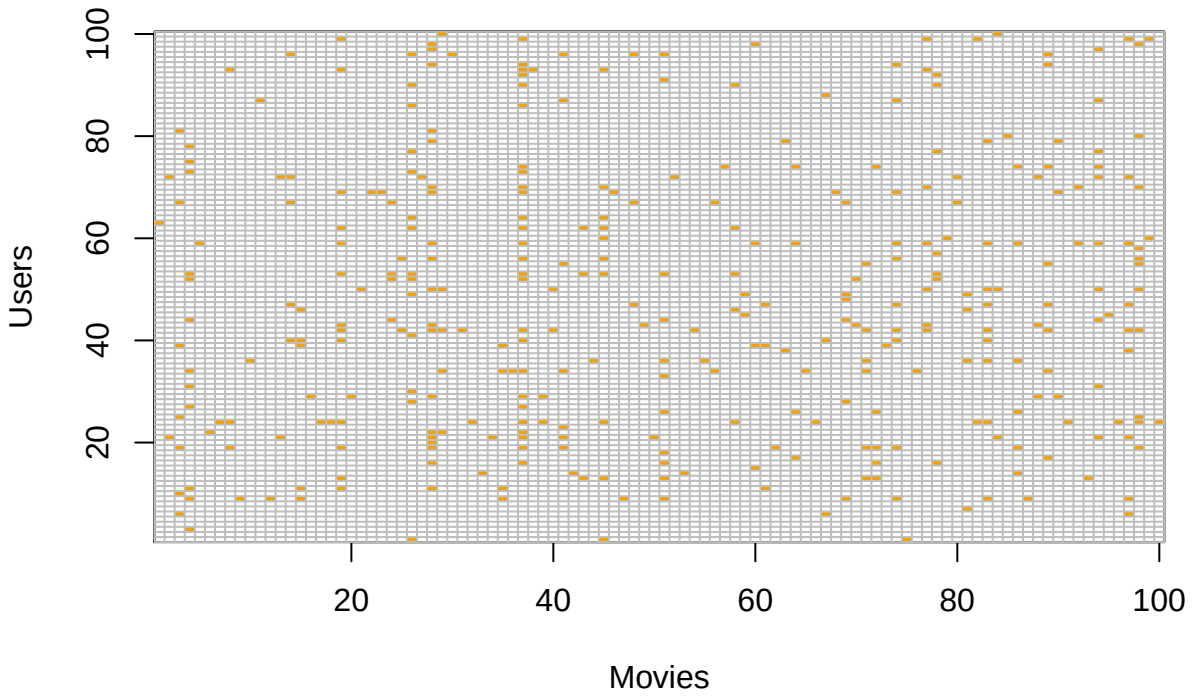
```
#using ggplot to distribution of user rating
p1 <- edx %>% group_by(userId) %>%summarise(n = n())%>%
                                  ggplot(aes(n)) +
                                  geom_histogram(bins = 30, color = "black") +
                                  ggtitle("Distribution of users")
```

```
#using ggplot to distribution of movie rating
p2 <- edx %>% group_by(movieId) %>%summarise(n = n())%>%
                                    ggplot(aes(n)) +
                                    geom_histogram(bins = 30, color = "black") +
                                    ggtitle("Distribution of movies")

#show plots in grid
grid.arrange(p1, p2, ncol = 2)
```



Using an approach that was suggested by Irizarry(2020) in a similar MovieLens recommendation system, an analysis of 100 unique random users confirms that some users rate more movies than other and also some movies are rated more often than others.

```
#sample 100 unique users
users <- sample(unique(edx$userId), 100 )

#plot a grid of movies rated by the 100 unique random users
edx %>% filter(userId %in% users) %>% select(userId, movieId, rating) %>%
                                      mutate(rating = 1) %>%
                                      spread(movieId, rating) %>% select(sample(ncol(.), 100)) %>%
                                      as.matrix() %>% t(.) %>%
                                      image(1:100, 1:100,. , xlab="Movies", ylab="Users")
                                      abline(h=0:100+0.5, v=0:100+0.5, col = "grey")
```

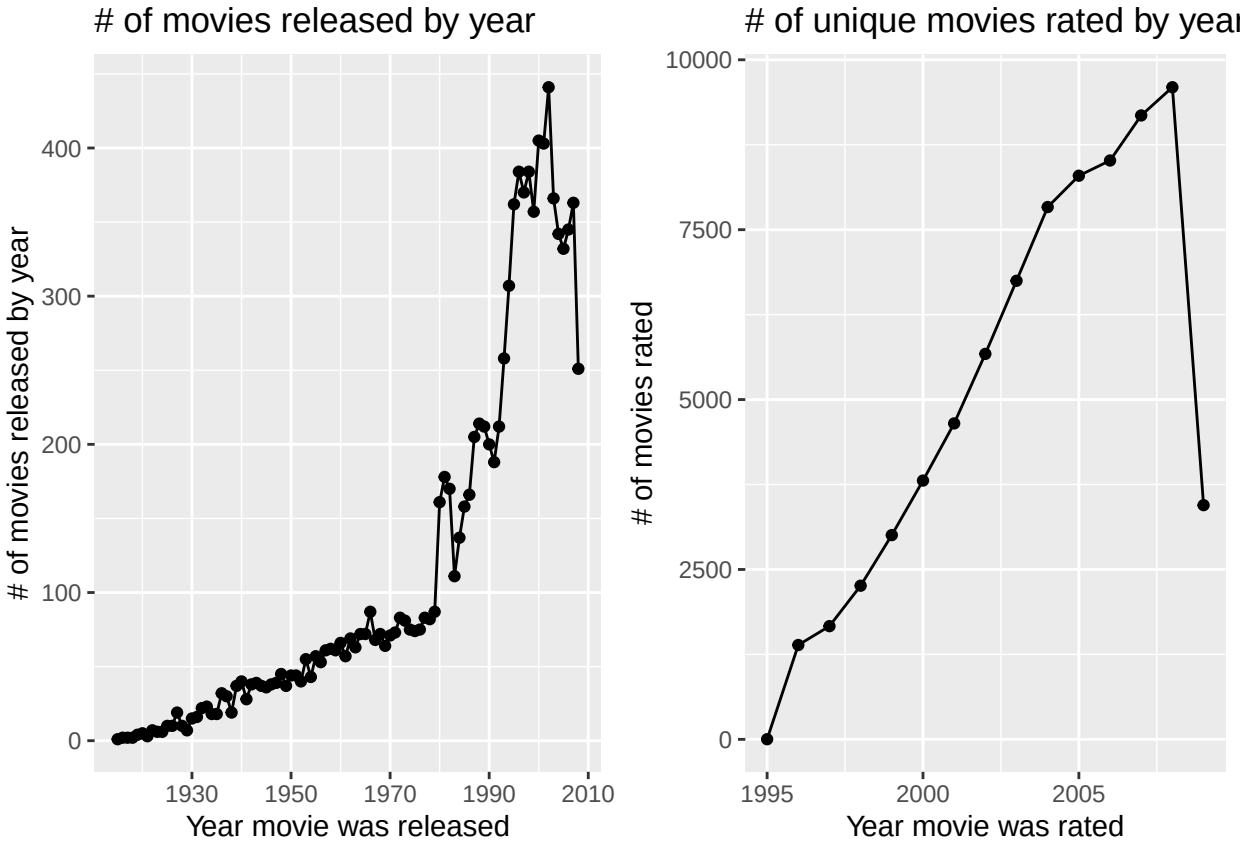


2.1.2 Day the movie is rated

The day of the week, year movie was rated, and year the movie was released were some of the variable that were explored in the project. The day of the week "wkday", and year movie was rated "year_rated" were derived from the timestamp variable by using the "wday" and "year" functions from the lubridate package. While, the year the movie was released "release_year" was extracted from the title variable by using "str_extract" function and a regular expression.

```
#year patern from title variable
yr_pattern_in_title <- "\\(\\d{4}\\)"

#using str_extract function to extract movie release year from title variable and,
#year and wkday function from lubridate to extract year and weekday respectively from timestamp
edx <- edx %>% mutate(release_year = as.numeric(str_extract(
                                    str_extract(.$title, yr_pattern_in_title ), '\\d{4}')),
                      year_rated = year(as_datetime(timestamp)),
                      wkday = wday(as_datetime(timestamp), label = TRUE)
                     )

#use ggplot to plot numbers of movies released in each year
p1 <- edx %>% group_by(release_year) %>%
              summarize(n = n_distinct(movieId)) %>%
              ggplot(aes(release_year, n)) +
              geom_line() + geom_point() + ylab('# of movies released by year') +
              xlab("Year movie was released") + ggtitle("# of movies released by year")
```

```
#use ggplot to plot numbers of unique movies rated in each year
p2 <- edx %>% group_by(year_rated) %>%
              summarize(n = n_distinct(movieId)) %>%
              ggplot(aes(year_rated, n)) +
              geom_line() + geom_point() +
              xlab("Year movie was rated") + ylab('# of movies rated') +
              ggtitle("# of unique movies rated by year")

#show plots in grid
grid.arrange(p1, p2, ncol = 2)
```

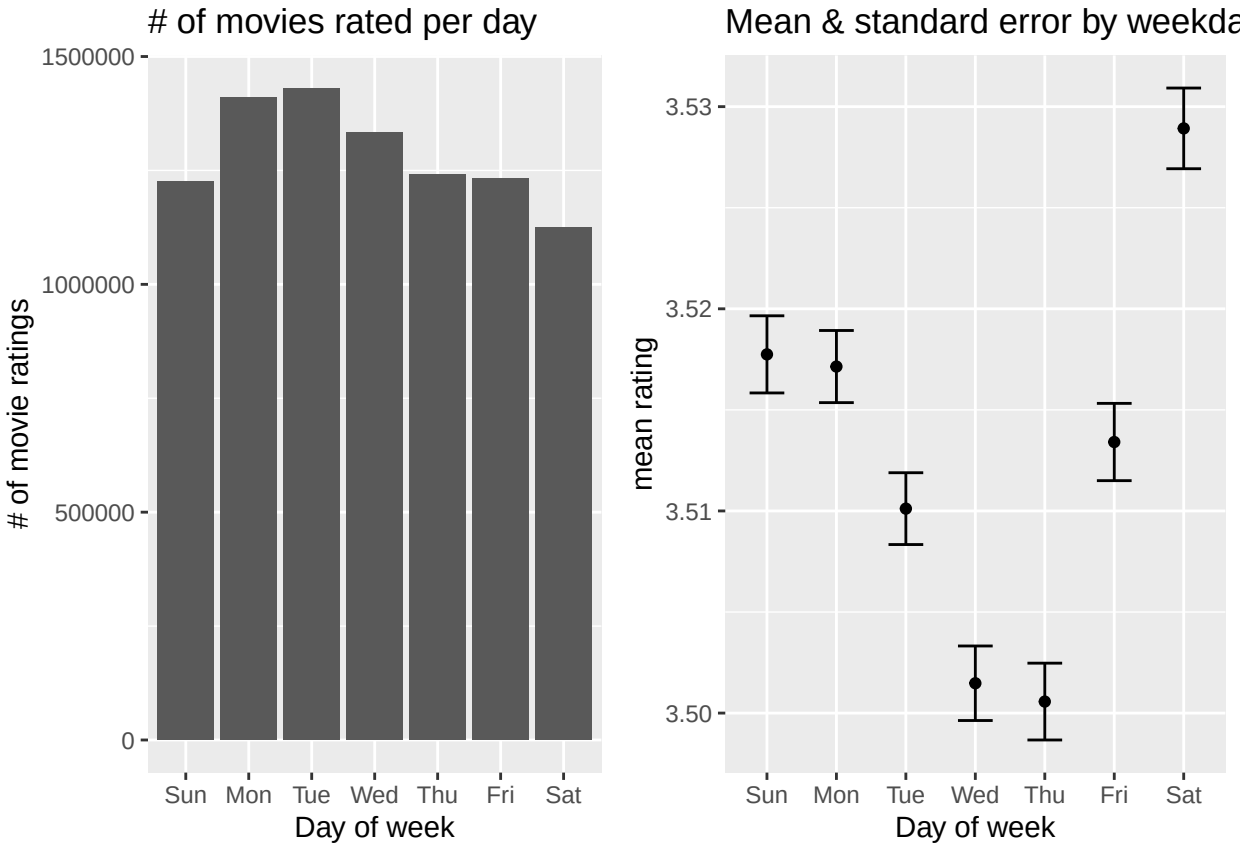


The above graphs, indicate that the highest number of movies (441) were released in 2002, and most movies (9,590) were rated in 2008. Also, the data indicate that there has been a steady increase in numbers of unique movies rated by users before reaching the pick at 9,590 movies in 2008.

While the plots of day of the week below indicate that saturday have the least amount of movies rated but is also the day with the highest mean rating. The plot for mean rating per day with the standard errors further shows strong evidence of a day effect on the rating.

```
#use ggplot to plot for weekday and number of ratings
p1 <- edx %>% group_by(wkday) %>% summarise(n = n()) %>%
                                  ggplot(aes(wkday, n)) + geom_bar(stat = "identity") +
                                  xlab("Day of week") + ylab("# of movie ratings") +
                                  ggtitle("# of movies rated per day")


#use ggplot to plot mean rating and standard error
```

```
p2 <- edx %>% group_by(wkday) %>% summarise(mu = mean(rating),
                                            se = sd(rating)/sqrt( n())) %>%
                                            ggplot(aes(wkday, mu, ymin= mu - 2 * se, ymax = mu + 2 * se)) +
                                            geom_point() + geom_errorbar(aes(width = 0.5)) +
                                            xlab("Day of week") + ylab("mean rating") +
                                            ggtitle("Mean & standard error by weekday")


#show plots in grid
grid.arrange(p1, p2, ncol = 2)
```



2.1.3 Movie genre

The movie genres are provided as a composite variable with each genre separated by a slash (|). To better explore genre variable, the composite "genres" was decomposed into a "genre" variable. A temporally dataset was created to have a copy of the edx data set in a long format with each genre record as a row.

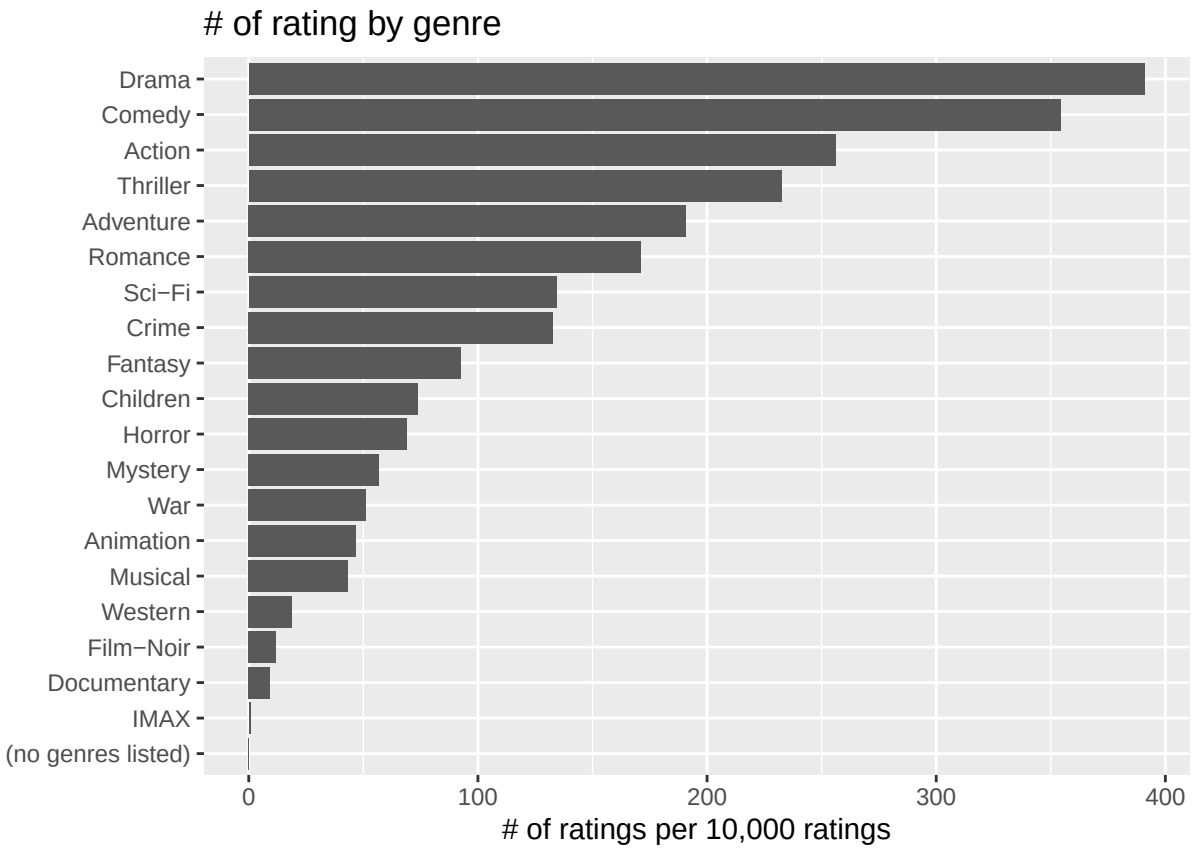
Below steps indicate code used to decompose the genres column. To avoid the process of having each genre on individual row taking unnecesarily longer, an temporally dataset "df" was created to store the unique genres and then separate each genre into its own row. This temporally dataset was then merged to the edx dataset. To ensure that the structure of the training set (edx) is maintained, result of the merge was stored in the temporally "df" dataset.

```
# ! Warning: this chunk of code takes a bit longer ( 2 - 5 minutes)
#get unique genres from genres column
```

```
df <- tibble(unique(edx$genres))

#rename column name
colnames(df) <- 'genres'

#make copy of the genres to genre column
df$genre <- df$genres

#count # of decomposed genres in each composite genre
df$no_genres <- sapply(df$genres, function (x) length(unlist(str_split(x, pattern ="\\|"))))

#merge the no of genres to the edx dataset
edx <- df %>% select(genres, no_genres) %>%left_join(edx, ., by = c("genres"))

#split each decomposed genre to be on its own row
df <- df %>% separate_rows(genre, sep ="\\|")


#merge edx to df, using left join
df <- df %>% select(genres, genre) %>% left_join(edx, ., by =c('genres'))
```

The charts below indicate that the genre of drama has the most ratings and that there are some movies rated which had no genre indicated.

```
#plot # of rating in each genre
df  %>% group_by(genre) %>%
              summarize(n = n()/10^4) %>% arrange(desc(n)) %>%
              ggplot(aes(reorder(genre, n), n)) +  geom_bar(stat = "identity")  +
              ylab("# of ratings per 10,000 ratings") + xlab(" ") + coord_flip() +
              ggtitle("# of rating by genre")
```

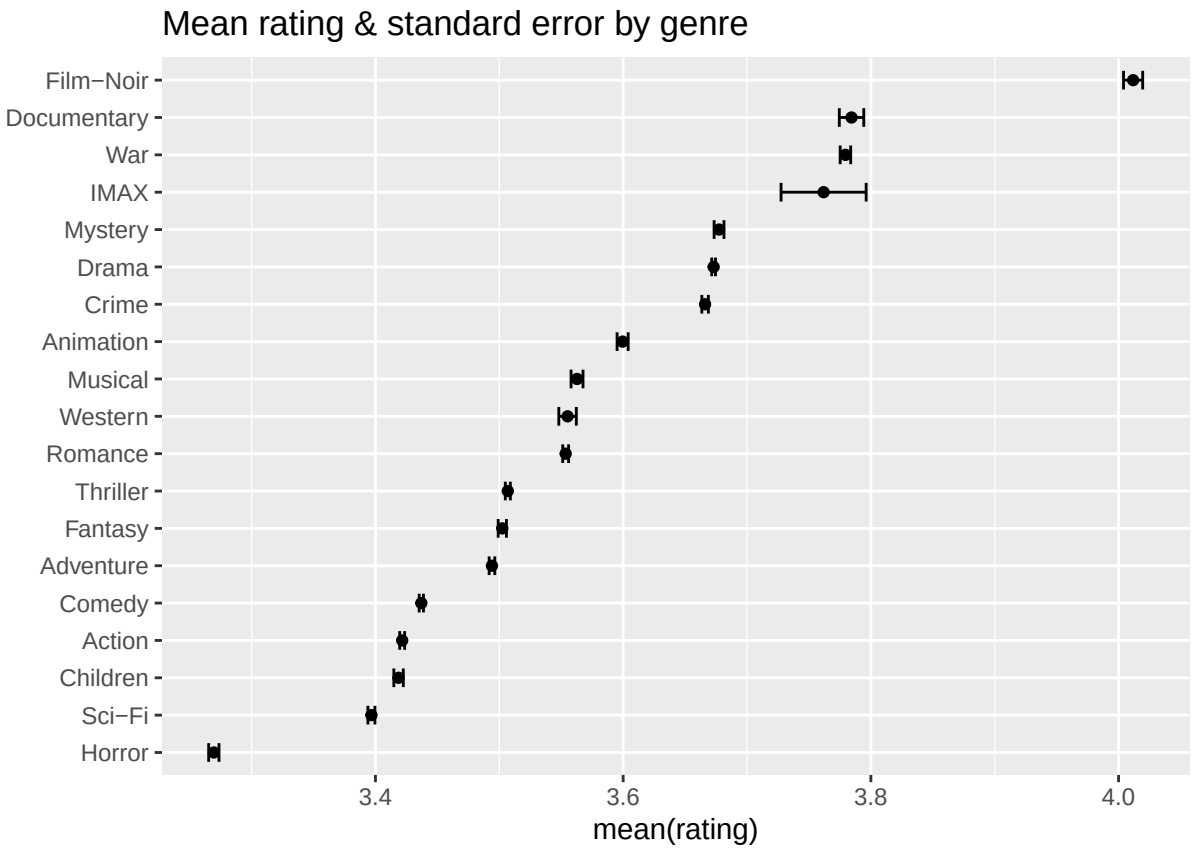


Inspection of the "(no genres listed)", indicate that the movie "Pull My Daisy (1958)" has 7 rows with no genre listed.

```
#show the movies with no genre
df %>% filter(genre == '(no genres listed)') %>% select(title, genre, year_rated, rating) %>%knitr::kabl
```

title	genre	year_rated	rating
Pull My Daisy (1958)	(no genres listed)	2007	5.0
Pull My Daisy (1958)	(no genres listed)	2007	4.5
Pull My Daisy (1958)	(no genres listed)	2004	2.0
Pull My Daisy (1958)	(no genres listed)	2008	3.5
Pull My Daisy (1958)	(no genres listed)	2004	3.5
Pull My Daisy (1958)	(no genres listed)	2007	2.5

```
#plot mean rating and standard error for each genre with at least 1000 ratings
df %>% group_by(genre) %>% summarise(mu = mean(rating),
                                     se = sd(rating)/sqrt( n()),
                                    n = n()) %>% filter(n >= 1000)  %>%
                                ggplot(aes(reorder(genre, mu), mu, ymin= mu - 3*se, ymax = mu + 3*se)) +
                                geom_point() + coord_flip() +
                                geom_errorbar(aes(width = 0.5)) +
                                ylab('mean(rating)') + xlab("") +
                                ggtitle("Mean rating & standard error by genre")
```

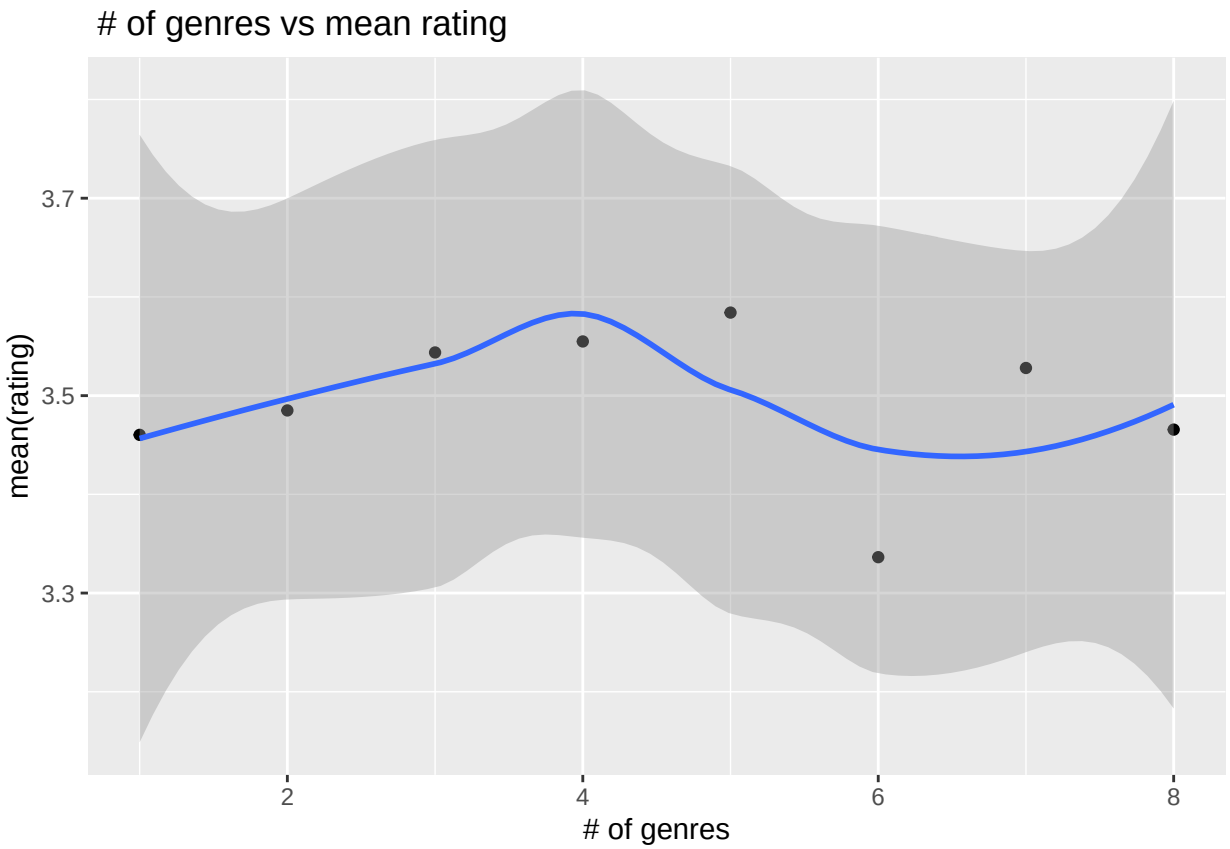


The plot of mean rating per genre indicate the mean rating is between 3.27 and 4.01, with the genre "Film-Noir" having the highest mean rating.

Furthermore, the plot of number of genres and mean rating indicate that the mean rating increases as the number of genres increase until the number of genres is 4, there after the rating decrease with an increase in number of genres.

```
#Use ifelse to assign no genres listed to be 0 = no_genres
edx <- edx %>% mutate(no_genres = ifelse(genres == '(no genres listed)', 0, no_genres))


#use ggplot to plot no_genres vs their mean(rating)
edx %>% filter(no_genres >= 1) %>% group_by(no_genres) %>% summarize(rating = mean(rating)) %>%
                                   ggplot(aes(no_genres, rating)) +
                                   geom_point() + geom_smooth() +
                                   ylab('mean(rating)') + xlab("# of genres")  +
                                   ggtitle(" # of genres vs mean rating")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Both the plot for "# of genres vs mean rating" and the "Mean rating & standard error by genre" indicate strong genre effect on rating.

Based on the foregoing discussion, the movie ratings are influenced by the following "biases": user, movie, day of the week, and genre. The following section will discuss the data modeling approach for the machine learning algorithm.

2.2 Modeling machine learning algorithm

The machine learning algorithm is modeled by using the approach discussed in the course, where "bias" terms are incrementally added to the model. The "edx" dataset is used to train the model while the "validation" dataset was used to test and report accurancy of the model.

From the data exploration, we notice that predicting the movie user rating is influenced by the user, the movie, day of the week, and genres. The user and movie effect has been found to have a model effect in movie rating predictions systems (Irizarry, 2020).

The model used to develop the prediction algorithm follows that from the course, where the mean μ rating is modified by one or more "bias" terms with a residual error ε expected. Hence the base model can be represented as:

$$Y_{u,i} = \mu + \varepsilon_{i,u,g}$$

```
#calculate mean rating in the training set
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512464
```

```
#> [1] 3.54
```

Using the mean, as the base model. The RSME for the base model is: 1.060331.

```
#calculate the RSME
model_rmse <- RMSE(edx$rating, mu)
model_rmse
```

```
## [1] 1.060393
```

```
#add the rsme result to a tibble
rmse_results <- tibble(method = "Train: Base model - using only mean", RMSE = model_rmse)
```

Using the approach discussed in the course, the movie b_i and user b_u effects were added to the model as bias terms below. The resulting model was:

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{i,u,g}$$

```
#Calculating the mean for movie effect model
movie_avgs <- edx %>% group_by(movieId) %>%
                      summarize(b_i = mean(rating - mu))


#Calculating the mean for user and movie effect model
user_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>%
                      group_by(userId) %>%
                      summarize(b_u = mean(rating - mu - b_i))
```

Using the movie and user biases, the RSME for this model is: 0.8567039.

```
#create predictions for movie and user effect model
predicted_ratings <- edx %>%  left_join(movie_avgs, by='movieId') %>%
                              left_join(user_avgs, by='userId') %>%
                              mutate(pred = mu + b_i + b_u) %>%
                              pull(pred)

#calculate RSME
model_rmse <- RMSE(predicted_ratings, edx$rating)

#save RSME to tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Train: Movie + User effects model",
                                     RMSE = model_rmse ))

#view RSME tibble
rmse_results %>% knitr::kable()
```

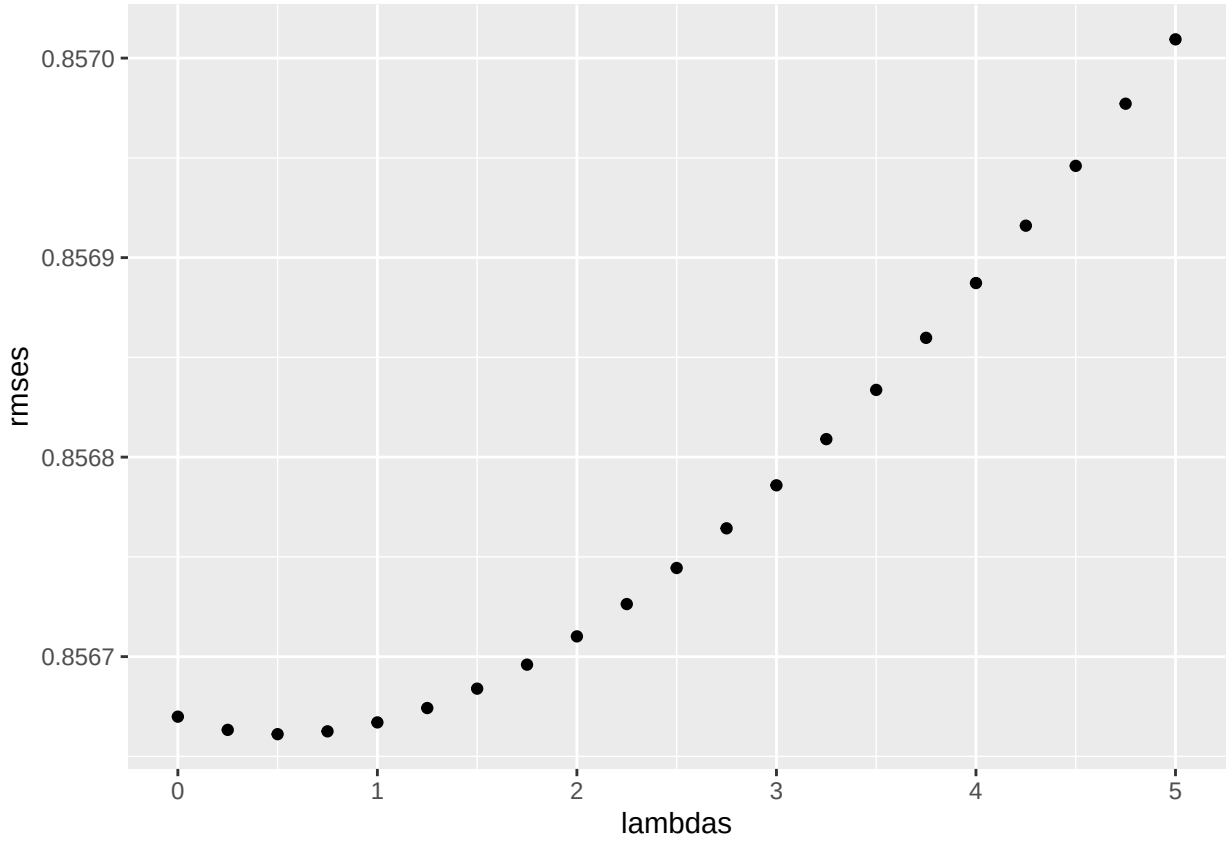
method	RMSE
Train: Base model - using only mean	1.0603926
Train: Movie + User effects model	0.8566699

The project did not consider adding the day of the week, and genre biases to the model because they were considered to be outside the scope of the project requirements.

2.3 Regularization movies and users effect model

The data exploration section indicated that some movies are rated by fewer users and also some users rate more movies than others. Movie and users with fewer ratings results in larger estimates which results in larger errors in the predictions causing a higher RMSE (Irizarry, 2020). Regulazation of the movie and user effects was applied to the model using an approach that was discussed in the course.

```
#Regularazation of the user and movie effect
#tuning parameter
lambdas <- seq(0, 5, 0.25)

# ! warning - this take a bit longer 3 - 5 minutes
#function to calculate rmses using cross validation
rmses <- sapply(lambdas, function(l){

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

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

     predicted_ratings <-
          edx %>%
          left_join(b_i, by = "movieId") %>%
          left_join(b_u, by = "userId") %>%
          mutate(pred = mu + b_i + b_u) %>% pull(pred)

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

#plot the lambdas vs rmses
qplot(lambdas, rmses)
```




```
#print lambda_results_in_min_RSME
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 0.5
```

The plot of lambdas and rmses above indicate that lambda that minimises the RSME in the regularised model is 0.5.

```
#add regularize model rmse to rmse tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Train: Regularized Movie + User Effect Model",
                                 RMSE = min(rmses)))

#print the rmse_result table
rmse_results %>% knitr::kable()
```

method	RMSE
Train: Base model - using only mean	1.0603926
Train: Movie + User effects model	0.8566699
Train: Regularized Movie + User Effect Model	0.8566611

3.0 Results

Using the regularized movie and user effect model with lamdba of 0.5. The code below calculates the RSME on the validation set.

```
#mean rating n validation set
mu_v <- mean(validation$rating)

#model on validation set
b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_v)/(n()+lambda))

b_u <- validation %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu_v)/(n()+lambda))

#predicting rating 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

#calculating RSME on validation set
rmse_v <- RMSE(predicted_ratings, validation$rating)
rmse_v
```

```
## [1] 0.8260103
```

```
#add validation rsme to rsme table
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Validation: Regularized Movie + User Effect Model",
                                     RMSE = rmse_v))
```

The resulting RSME on the validation set is: 0.8258479.

4.0 Conclusion

This project used the approach that was discussed in the course and some of the R code was adapted from the course to implement movie recommendation system based on user movie ratings. The project applied reguralization of the user and movie effect which helped to improve the RSME. The final RSME of 0.82585 was obtained by the project. The RSME achieved by the project meets the project requirement which was to develop a system with lower RSME, where an RSME of less than 0.86490 would indicate an excellent recommendation system.

Future work could consider adding the day of rating, and genre of the movie. These were identified in the data exploration section as having an effect on the movie ratings. The day of rating, and genre were not added in this project because they were considered to be outside the scope of the project. Use of other sophisticated algorithms is also an area that future work could consider.

5.0 Reference

Irizarry, R.A. (2020), “Introduction to Data Science Data Analysis and Prediction Algorithms with R”, (url: https://rafalab.github.io/dsbook/)