MovieLens Project : Movie Rating Prediction using MovieLens Data

Arun Kataria

5/10/2021

Introduction : Project

This is an R Markdown document that contains the code & report to showcase the developments in achieving a statistically usable prediction model that predicts the movie ratings based on various external factors like Movie, User, Genre etc

The statistical viability of this model and various other models developed during the modeling to predict the movie ratings is evaluated based on the RMSE (Root Mean Square Error) method. The Goal is to reduce the RMSE as much as i can.

Method/Analysis :

Overview of DataSet (train_set): The movielens dataset edx has further been divided into train_set(90%) and test_set(10%) to train and test various models. Hereafter we will be using the train_set for training of models and test_set to test the RMSE of various models.s

The train_set dataset contains 8100065 records.

Following is the structure of the source Dataset 8100065 rows and 6 columns. The columns of the dataset available are userId, movieId, rating, timestamp, title, genres.

The dependent variable or the variable that we are interested to predict here is : **rating**

The independent variables or the variables that we will be using to predict rating are: - userId - movieId - timestamp - title - genres

```
head(train_set,10)
```

A Basic exploration of the data in the train_set

```
##     userId movieId rating timestamp                                 title
##  1:      1     122      5 838985046                      Boomerang (1992)
##  2:      1     292      5 838983421                       Outbreak (1995)
##  3:      1     316      5 838983392                       Stargate (1994)
##  4:      1     329      5 838983392           Star Trek: Generations (1994)
##  5:      1     355      5 838984474                 Flintstones, The (1994)
##  6:      1     356      5 838983653                     Forrest Gump (1994)
##  7:      1     362      5 838984885                 Jungle Book, The (1994)
```

```
##  8:      1     364      5 838983707                     Lion King, The (1994)
##  9:      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (1994)
## 10:      1     377      5 838983834                              Speed (1994)
##                                     genres
##  1:                            Comedy|Romance
##  2:               Action|Drama|Sci-Fi|Thriller
##  3:                    Action|Adventure|Sci-Fi
##  4:              Action|Adventure|Drama|Sci-Fi
##  5:                    Children|Comedy|Fantasy
##  6:                  Comedy|Drama|Romance|War
##  7:                 Adventure|Children|Romance
##  8: Adventure|Animation|Children|Drama|Musical
##  9:                              Action|Comedy
## 10:                    Action|Romance|Thriller
```

Below are the distinct values in :

- userId =69878
- movieId =10677
- title =10676
- genres =797

Lets start analyzing the data more:

Data Analysis & Modeling Mean Rating = 3.5124556

First Model

At first we started with the mean rating across the complete train_set and tested out the RMSE of this simple model.

```
mu_hat <- mean(train_set$rating)
mu_hat
```

```
## [1] 3.512456
```

```
#Validating the RMSE value for simple mean, matching with ratings in test set
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
```

```
## [1] 1.060054
```

```
# A RMSE results table/dataframe created to store RMSEs of various models as we analyze it
rmse_results <- data_frame(method = "Simple average", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
# Print rmse results for simple average - A baseline RMSE
rmse_results
```

```
## # A tibble: 1 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Simple average  1.06
```

Lets see if we can improve the RMSE of this model using other independent variables available. We will start with MovieId and see if we can improve the RMSE value For this we need to calculate average ratings by movieId, so this will capture the impact of movieId on overall ratings

Second Model

```
# Calculate the Average rating across the train_set and storing in mu
mu <- mean(train_set$rating)


# Calculating the Average impact of movieId on rating
# For this we first substract the mu (average rating across train_set) from actual rating
# and then calculate impact of movieId on this and store in new data frame - movie_avgs
movie_avgs <- train_set %>%
  group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))


# Join the movie_avgs dataframe created above with test_set on movieId and add
# the movie Impact (b_i) to the overall mean (mu) to get predicted ratings
# (store in predicted_ratings dataset)
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i


# Test out the Root mean square error of predicted ratings calculated above
# to actual ratings in test_set using RMSE function
# append/add the RMSE value results to rmse_results dataframe
model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))

# View the RMSE values just added for Movie Model
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615

Using MovieId along with Overall means (mu) improved the RMSE value significantly. However we can try using other independent variables like userId, genres etc to see if they can improve it further.

Third Model

```
# Calculate the impact that userId has on the movie ratings. Create dataset user_avgs for this
# grouping on userId and substracting previous calculated impacts (mu, b_i)
# filter out outliers
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n()>3) %>%
  summarize(b_u = mean(rating - mu - b_i))


# Generate predicted ratings using movie avgs and user avgs and mu
predicted_ratings_mu <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + ifelse(is.na(b_i),0,b_i) + ifelse(is.na(b_u),0,b_u)) %>%
  .$pred


# Test out the RMSE with the latest predicted ratings and append these to existing rmse_results datafra
model_2_rmse <- RMSE(predicted_ratings_mu, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843

This has further improved the RMSE value, the model looks good. Lets proceed further to search for hidden insights into data. For this we would need to do some data wrangling or enrichment.

We will be creating additional columns that capture following information in the data. 1. nt - This columns should hold the length of the title 2. s - Column to hold the count of spaces in title (words) 3. gs - Column to hold the count of pipe symbol "|" in genres (~no of genres) 4. yr - Column to hold the release year of movie (extracted from title)

It would be great to find if these hidden/derived parameters actually have an effect over the ratings or not. eg:- it would be great to know if the length of name of a movie can be used to predict the rating? Can we predict the movie rating by the no of words its title has? are the recently released movies preferred over the older ones (release year)? Can a movies rating be predicted by counting the no of genres it has? To find out lets process all of the dataset (train_set, test_set and validation) with these additional columns.

```
# DATA PREPRATION :
# Further data preparation
```

```
# we will further use these enrichments and try to predict the ratings using them as well

# Add column to all datasets "nt" that holds the length of the title column
train_set<- train_set %>% mutate(nt=str_length(title))
test_set <- test_set %>% mutate(nt=str_length(title))
validation <- validation %>% mutate(nt=str_length(title))


# Add column to all datasets "s" that holds the count of spaces in the title
train_set <- train_set %>% mutate(s=str_count(title,"\\ "))
test_set <- train_set %>% mutate(s=str_count(title,"\\ "))
validation <- validation %>% mutate(s=str_count(title,"\\ "))


# Add column to all datasets "gs" that holds the count of pipe symbol in genres
# (similar to space for title)
train_set <- train_set %>% mutate(gs=str_count(genres,"\\|"))
test_set <- test_set %>% mutate(gs=str_count(genres,"\\|"))
validation <- validation %>% mutate(gs=str_count(genres,"\\|"))


# Create column "yr" and add that to all datasets, this holds the year part from title
# Extract (last 4 characters from title)
train_set<-train_set %>% mutate(yr=substr(title,str_length(title)-4,str_length(title)-1))
test_set <- test_set %>% mutate(yr=substr(title,str_length(title)-4,str_length(title)-1))
validation <- validation %>% mutate(yr=substr(title,str_length(title)-4,str_length(title)-1))


# sneak peek into enriched train_set now
head(train_set)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     292      5 838983421               Outbreak (1995)
## 3:      1     316      5 838983392               Stargate (1994)
## 4:      1     329      5 838983392 Star Trek: Generations (1994)
## 5:      1     355      5 838984474       Flintstones, The (1994)
## 6:      1     356      5 838983653           Forrest Gump (1994)
##                             genres nt s gs   yr
## 1:                Comedy|Romance 16 1  1 1992
## 2:  Action|Drama|Sci-Fi|Thriller 15 1  3 1995
## 3:       Action|Adventure|Sci-Fi 15 1  2 1994
## 4: Action|Adventure|Drama|Sci-Fi 29 3  3 1994
## 5:       Children|Comedy|Fantasy 23 2  2 1994
## 6:      Comedy|Drama|Romance|War 19 2  3 1994
```

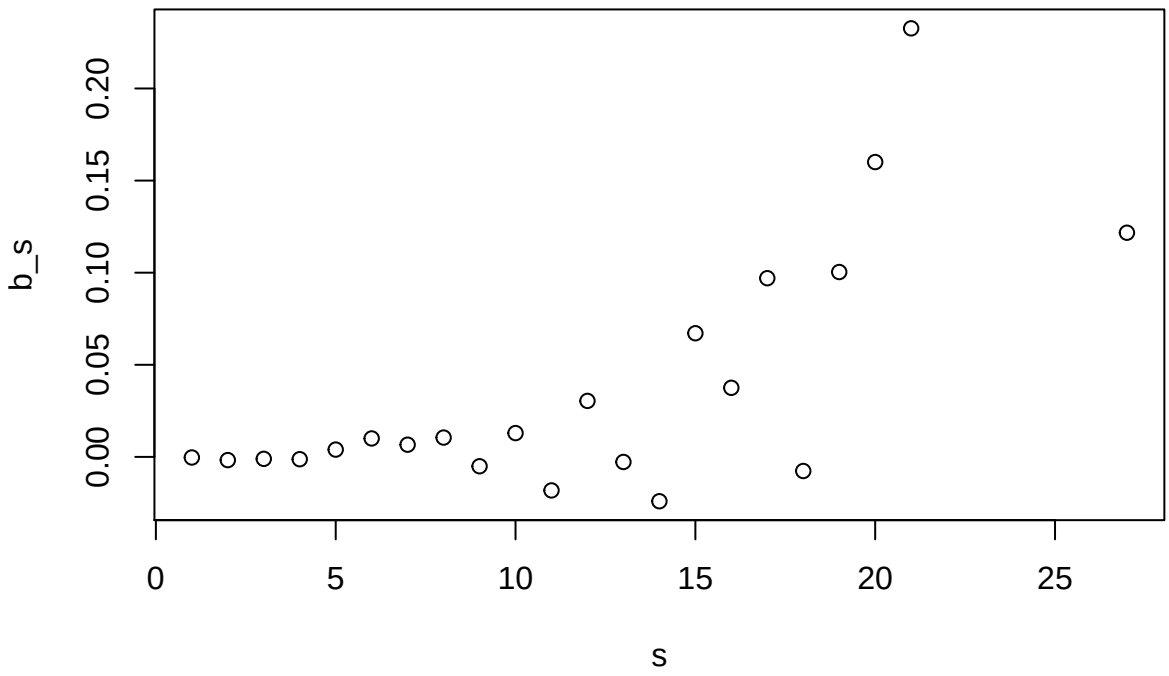
Now lets predict the ratings using these enriched dataset columns (gs,s,nt,yr)

```
# Train further using these additional columns

# Use train_set to calculate raing averages across year(substring of tiles)
# Year
yr_avgs <- train_set %>%
```

```
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(yr) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u))

# Calcuate predicted values using the year averages as well
# Predicted Rating = Overall mean + movie impact + user impact + year impact
predicted_ratings_muy <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  mutate(pred = mu + b_i + b_u + b_y) %>%
  .$pred

# Calcuate the RMSE value of predicted ratings and add these to the rmse_results dataframe
model_3_rmse <- RMSE(predicted_ratings_muy, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + year Effects Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843
Movie + User + year Effects Model	0.8560307

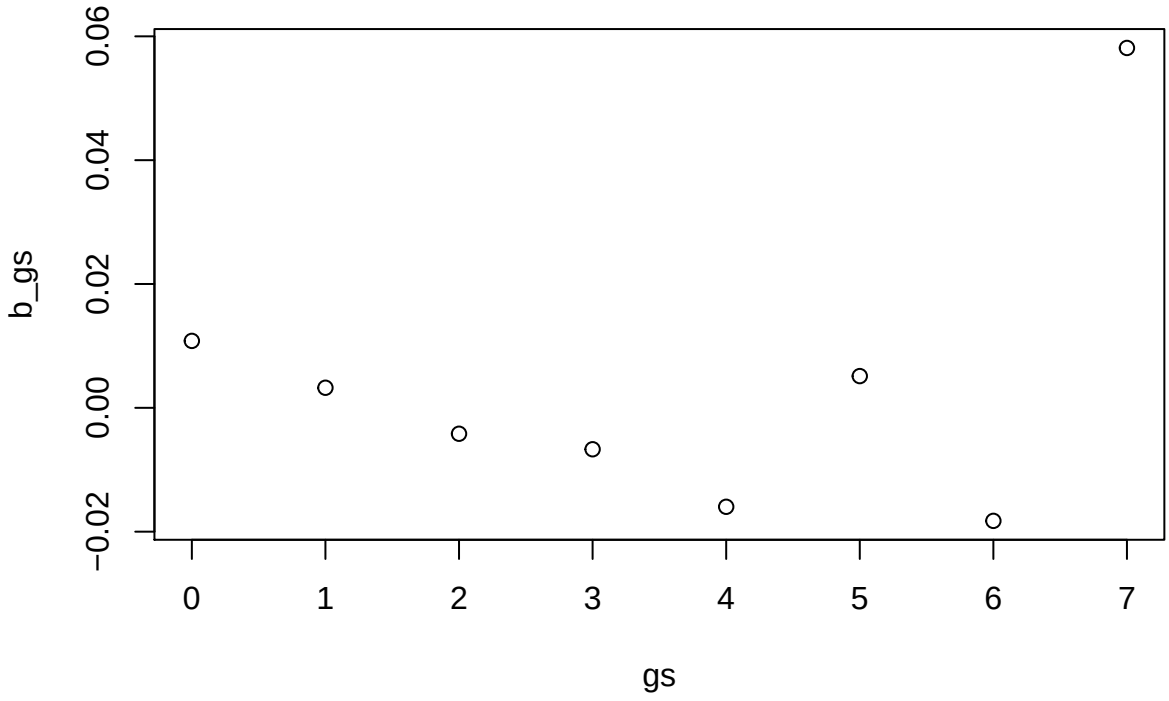
```
# Use train_set to calculate rating averages across spaces in title "s" column
# No of Spaces in title impact
s_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  group_by(s) %>%
  summarize(b_s = mean(rating - mu - b_i - b_u - b_y))

# View variability across no of spaces
s_avgs %>% plot()
```




```
# Use No of Spaces "s" column to predict the ratings
# Pred Ratings =  Overall Mean +  Movie Impact + UserImpact + Year Impact + NoofSpaces Impact
predicted_ratings_muys <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_s) %>%
  .$pred

# Calcuate the RMSE value of predicted ratings and add these to the rmse_results dataframe
model_4_rmse <- RMSE(predicted_ratings_muys, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + year + NoOfSpaces(title) Effects Model",
                                     RMSE = model_4_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843
Movie + User + year Effects Model	0.8560307
Movie + User + year + NoOfSpaces(title) Effects Model	0.8560211

```r
# Use train_set to calculate rating averages across no of pipe "|" in genres "gs" column
# No of pipe symbol in genres impact
gs_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
    group_by(gs) %>%
  summarize(b_gs = mean(rating - mu - b_i - b_u - b_y - b_s))


# Plot no of pipe in genres vs variability in ratings
gs_avgs %>% plot()
```




```r
# Use No of pipes "gs" column to predict the ratings
# Pred Ratings =  Overall Mean +  Movie Impact + UserImpact + Year Impact +
# + NoofSpaces(title) Impact + NoOfPipes(Genres)
predicted_ratings_muysgs <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  left_join(gs_avgs, by='gs') %>%
    mutate(pred = mu + b_i + b_u + b_y + b_s + b_gs) %>%
  .$pred
```

```r
# Calcuate the RMSE value of predicted ratings and add it to rmse_results dataframe
model_5_rmse <- RMSE(predicted_ratings_muysgs, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + year + nSpaces(title)
                                     + nPipes(Genres) Effects Model",
                                     RMSE = model_5_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843
Movie + User + year Effects Model	0.8560307
Movie + User + year + NoOfSpaces(title) Effects Model	0.8560211
Movie + User + year + nSpaces(title) + nPipes(Genres) Effects Model	0.8559915

```r
# Use train_set to calculate rating averages across length of title "nt" column
# Length of tile "nt" impact
nt_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  left_join(gs_avgs, by='gs') %>%
  group_by(nt) %>%
  summarize(b_nt = mean(rating - mu - b_i - b_u - b_y - b_s - b_gs))



# Use Length of title "nt" column to predict the ratings
# Pred Ratings =  Overall Mean +  Movie Impact + UserImpact + Year Impact
# + NoofSpaces(title) Impact + NoOfPipes(Genres) + LengthOfTitle
predicted_ratings_muysgsnt <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  left_join(gs_avgs, by='gs') %>%
  left_join(nt_avgs, by='nt') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_s + b_gs + b_nt) %>%
  .$pred


# Calcuate the RMSE value of predicted ratings and add these to the rmse_results dataframe
model_6_rmse <- RMSE(predicted_ratings_muysgsnt, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + year + nSpaces(title)
                                     + nPipes(Genres) + Length(title) Effects Model",
                                     RMSE = model_6_rmse ))
```

```
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843
Movie + User + year Effects Model	0.8560307
Movie + User + year + NoOfSpaces(title) Effects Model	0.8560211
Movie + User + year + nSpaces(title) + nPipes(Genres) Effects Model	0.8559915
Movie + User + year + nSpaces(title) + nPipes(Genres) + Length(title) Effects Model	0.8559671

From the above modeling and predictions we are able to see that these enriched columns do have an effect on the overall ratings predicted. Including these column into the models and using the averages across these columns has definitely reduced the RMSE by some extent in every case. It is hence concluded that these hidden parameters do impact the overall rating.

Lets now try to refine this further and use the genres column as well in the prediction of ratings. As expected the RMSE imrpoved further using the genres column.

Result

After analysis of the predicted ratings i found that the range of predicted ratings went way ahead of top rating > 5 Also the minimum rating came out in negative < 0 The valid range of ratings lies in 0.5 - 5 , per source dataset hence any predicted rating beyond 5 was capped at 5 Also any negative rating or below 0.5 was converted to 0.5

This improved the performance of model further.

Again the predicted vs actual rating for test_set was analyzed by doing an absolute difference of actual and predicted rating on test_set. The max difference comes out to be 4.5 in two cases where (Actual=5 and predicted=0.5) or (Actual=0.5 and predicted=5)

To bridge this gap, ratings in range of 4.75 - 4.95 were converted to 4.75 & ratings in range of 0.55 - 0.75 were converted to 0.75.

This normalization reduced the RMSE to maximise the performance.

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

# Calculate rating averages across genres column
# Genres impact
genres2_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  left_join(gs_avgs, by='gs') %>%
  left_join(nt_avgs, by='nt') %>%
  group_by(genres) %>%
  filter(n()>0) %>%
  summarize(b_g3 = mean(rating - mu - b_i - b_u - b_y - b_s - b_gs - b_nt ))
```

```
# Use genres column to predict the ratings
# Pred Ratings =  Overall Mean +  Movie Impact + UserImpact + Year Impact +
#                 NoofSpaces(title) Impact + NoOfPipes(Genres) + LengthOfTitle + genres
# The Predicted rating has been corrected to 5 if it exceeds 5 and set to 0.5 if its
#   less than 0.5 or negative
# The Predicted rating is "adjusted" to 4.75 if its more than 4.75 and less than 4.95
# The predicted rating is "adjusted" to 0.75 if its more than 0.55 and less than 0.75
# This is done as to normalize the rating
# as the model is clearly overshooting (as there were so many predicted ratings more than 5)
# and undershooting as there were so many predicted ratings in negative as well
predicted_ratings_final <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  left_join(gs_avgs, by='gs') %>%
  left_join(nt_avgs, by='nt') %>%
  left_join(genres2_avgs, by='genres') %>%
  mutate(pred = mu + ifelse(is.na(b_i),0,b_i) + ifelse(is.na(b_u),0,b_u) +
           ifelse(is.na(b_y),0,b_y) +
           ifelse(is.na(b_s),0,b_s) +
           ifelse(is.na(b_nt),0,b_nt) +
           ifelse(is.na(b_gs),0,b_gs) +
           ifelse(str_detect(genres,"\\|"),ifelse(is.na(b_g3),0,b_g3),0)
  ) %>% mutate(pred=ifelse(pred>5,5,ifelse(pred<0.5,0.5,pred))) %>%
  mutate(pred=ifelse(pred>4.75 & pred<4.95,4.75,pred)) %>%
  mutate(pred=ifelse(pred>0.55 & pred<0.75,0.75, pred)) %>%
  .$pred


# Calcuate the RMSE value of predicted ratings and add these to the rmse_results dataframe
# THIS IS THE FINAL RMSE OF THE MODEL
model_7_rmse <- RMSE(predicted_ratings_final, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + year + nSpaces(title) +
                                     nPipes(Genres) + Length(title) + genres Effects Model",
                                     RMSE = model_7_rmse ))

# Display the final RMSE of the Model
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843
Movie + User + year Effects Model	0.8560307
Movie + User + year + NoOfSpaces(title) Effects Model	0.8560211
Movie + User + year + nSpaces(title) + nPipes(Genres) Effects Model	0.8559915
Movie + User + year + nSpaces(title) + nPipes(Genres) + Length(title) Effects Model	0.8559671

method	RMSE
Movie + User + year + nSpaces(title) + nPipes(Genres) + Length(title) + genres Effects Model	0.8555076

Below is the Final model that is created using MovieId, User, Genres, SpacesIn-Title, LengthOfTitle, NoOfGenres, Year of Release

The final model is tested out using the Validation data set and it performs well in predicting ratings.

```
############################
# Final Model #

# Predicting Ratings for Validation data set #

predicted_ratings_final <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(yr_avgs, by='yr') %>%
  left_join(s_avgs, by='s') %>%
  left_join(gs_avgs, by='gs') %>%
  left_join(nt_avgs, by='nt') %>%
  left_join(genres2_avgs, by='genres') %>%
  mutate(pred = mu + ifelse(is.na(b_i),0,b_i) + ifelse(is.na(b_u),0,b_u) +
           ifelse(is.na(b_y),0,b_y) +
           ifelse(is.na(b_s),0,b_s) +
           ifelse(is.na(b_nt),0,b_nt) +
           ifelse(is.na(b_gs),0,b_gs) +
           ifelse(str_detect(genres,"\\|"),ifelse(is.na(b_g3),0,b_g3),0)
  ) %>% mutate(pred=ifelse(pred>5,5,ifelse(pred<0.5,0.5,pred))) %>%
  mutate(pred=ifelse(pred>4.75 & pred<4.95,4.75,pred)) %>%
  mutate(pred=ifelse(pred>0.55 & pred<0.75,0.75, pred)) %>%
  .$pred



# THIS IS THE FINAL RMSE OF THE MODEL
Final_Validation_rmse <- RMSE(predicted_ratings_final, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Final Validation RMSE - Movie,User,year,nSpaces(title),nPi
                                     RMSE = Final_Validation_rmse ))

# Display the final RMSE of the Model
rmse_results %>% knitr::kable()
```

method	RMSE
Simple average	1.0600537
Movie Effect Model	0.9429615
Movie + User Effects Model	0.8646843
Movie + User + year Effects Model	0.8560307
Movie + User + year + NoOfSpaces(title) Effects Model	0.8560211
Movie + User + year + nSpaces(title)	

method	RMSE
+ nPipes(Genres) Effects Model	0.8559915
Movie + User + year + nSpaces(title)	
+ nPipes(Genres) + Length(title) Effects Model	0.8559671
Movie + User + year + nSpaces(title) +	
nPipes(Genres) + Length(title) + genres Effects Model	0.8555076
Final Validation RMSE - Movie,User,year,nSpaces(title),nPipes(Genres),Length(title),genres	0.8648957

```
Final_Validation_rmse
```

```
## [1] 0.8648957
```

The Final RMSE using the Validation dataset is 0.8648957

Conclusion

Here in this project we have learnt that with just 4 columns we were able to devise multiple strategies to predict the ratings and were successful to a good extent as well. Underlying the presented data there is so much hidden information that we can derive and plot to see the variability of ratings across it. With simply capturing means and tuning out the RMSE this project clearly showed how powerful the simple approach of using mean can be.

Variability across MovieId, UserId, Year of Release, Genre(s),Spaces in title, No of Genres, Length of title can collectively predict precisely to a great extent the rating of a movie. Individually these parameters have a lesser impact but can collectively be very powerful, simple and efficient in prediction of movie rating. The final RMSE score for this model using validation set is : 0.8648957

Scope of future enhancements :

There is still a good scope in improving the overall predicted ratings / overall RMSE of the model. From a user's perspective the ratings are being predicted in points in a continuous variable fashion. The predicted ratings lies in a range of 0.5-5. However the actual ratings are discrete in nature and falls on 0.5,1,1.5,2,2.5,3,3.5,4,4.5,5 range. The model needs to be improved further to convert the continuous range of ratings to the above discrete range.