Capstone Project 1 : The MovieLens

2023-11-15

Overview

Recommendation systems are very crucial in various business segments, including but not limited to online shopping and movies streaming services. The key idea is that given the user history in rating the items and the other users history in rating the items in additions to the information about the items themselves and other factors, what is expected user rating for the items that he/she never see or have? estimating this rating will help the businesses to recommend other items that the user most probably interested in.

In this project I have a data set of 10 million ratings for different users on different movies, this data set includes the following: the rating value out of 5, the user id, the movie information and the time at which the user rated that particular movie, this data set is splitted into two parts: the model building part (*edx*) of 90% of the entries, and the final testing part (*final_holdout_test*) of 10% of the entries, the *edx* data set is then splitted into model training part (*edx_train*) and model testing part (*edx_test*).

The *edx_train* and *edx_test* data sets will be used to build, evaluate, and select the best performing model that predicts the movie rating with the minimum root mean square error (RMSE), after selecting the best model it will be tested using the *final_holdout_test* data set that is never used in training or evaluating the model performance before.

After splitting the data sets, I started with the features engineering to select the features of interest and to extract some new features based on the existing ones, I did the following steps:

- Extract the movie release year from the movie title, the release year is the last part of the movie title in this format (release_year).
- Extract the rating year, rating month, rating week of the year, rating day of the week from the rating timestamp.
- Transform the movie genres to one-hot-coding, where each genre will be in a separate column.
- Did some correlation analysis to remove some of the features that are highly correlated with the others.

Then, I start using the features in incremental manner to build different models and evaluate them using the testing data set, after selecting the best model I used regularization to to improve the model performance.

The best regularized model is then used with the whole model building data set *edx* and evaluated against the final testing data set *final_holdout_test*, the RMSE of 0.864299 was achieved.

Contents

Features Engineering and Exploratory Data Analysis

In this section I will study the original data set and transform some features to different formats in addition to extract new features from the given ones. The original data set looks like

```
head(edx_org_sample) %>% knitr::kable(caption = 'The Original data set format')
```

Table 1: The Original data set format

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

In this data set, every row is representing a *rating* given by one user whose ID is *userId* to a movie that has the ID *movieId* and the title *title* and belongs to the genres given in the | separated list *genres*, this rating happened at the time given in *timestamp*.

This section is organized as the following: Extracting the release year, Formatting the rating date and time, and Genres encoding.

Extracting the release year

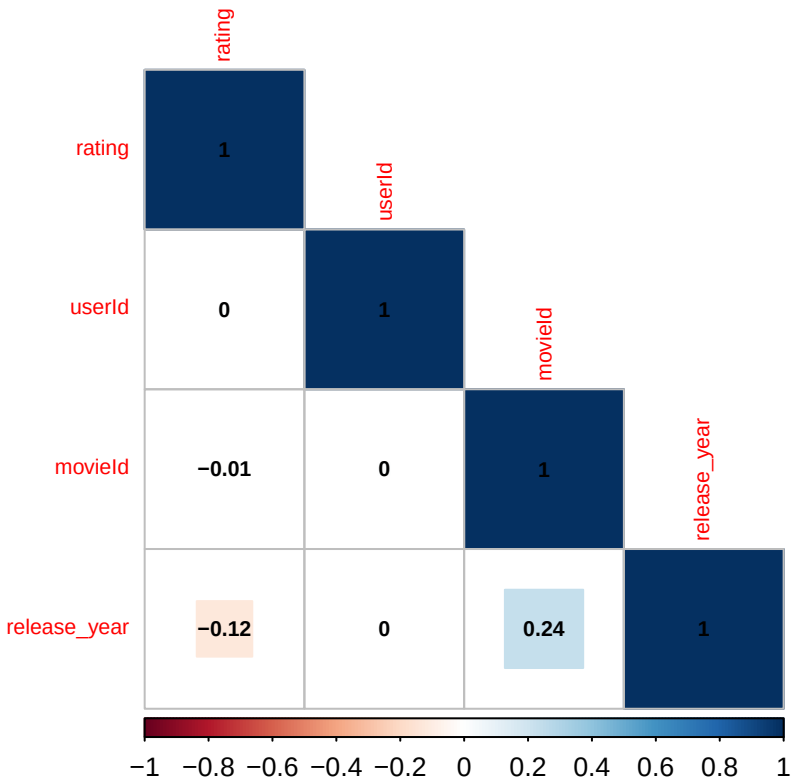
I noted that the movie title contains the release date, it is 1992 for the first movie in the table above, after doing some exploring, I found some movies contains more than () pair like this movie: *Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)*, which mean I need to take the value between the last pair of () as the release date, I developed the following function to extract the release date from the input movie title:

```
## the get_release_date function, to extract the release date from the title.
get_release_date <- function(m_title){
  ## get all strings encapsulated between the ()
  all_matches <- str_match_all(m_title, "\\((.*?)\\)")
  ## the matches will be given in all_matches[[1]]
  ## the first dimension of it is the number of matches
  matches_count <- dim(all_matches[[1]])[1]
  ## the last match is located at the matches_count
  ## the first value will be with (), the second will be without the () which what we want
  as.integer(all_matches[[1]][matches_count,2])
}
```

Then I used *sapply* function to add the *release_year* to both *edx* and *final_holdout_test* data sets.

Th correlation between *userId*, *movieId*, *release_year*, and *rating* is plotted using the following code:

```
cr <- cor(edx %>% select(rating,userId,movieId,release_year))
corrplot(cr,addCoef.col="black",number.cex=0.7,tl.cex = 0.7,type = 'lower',method = 'square')
```



As per the previous plot, it is clear that *rating* is not linearly correlated with neither *userId* nor *movieId*, however it is slightly negatively correlated the the *release_year* which means the new movies tend to have lower rating than the new ones. The *movieId* is slightly positively correlated with the *release_year*, this is expected since the ID is a sequence number that is incrementing over time so the new ratings for a new movies will happen after the ratings for an old ones, which means they will tend to have higher ID number.

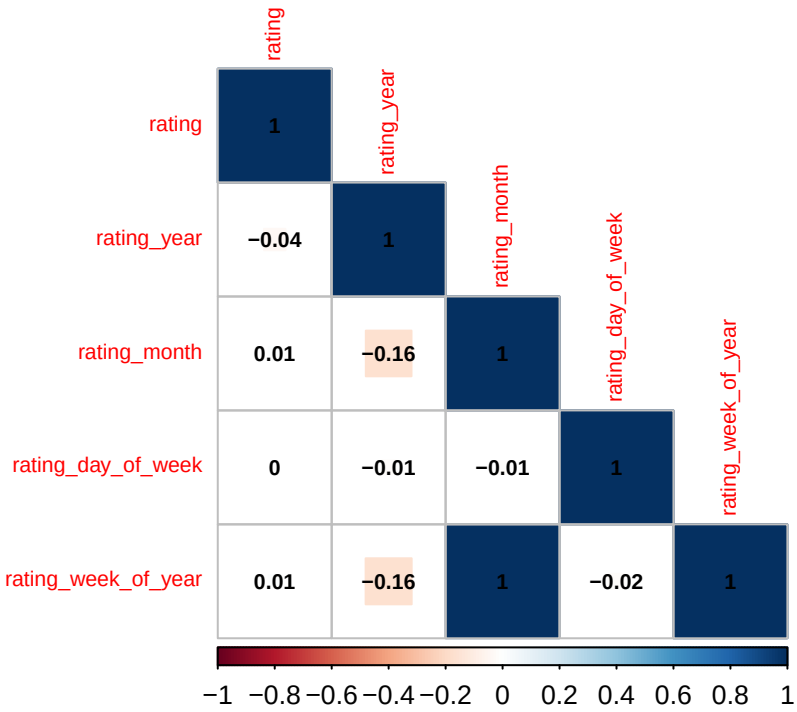
Formatting the rating date and time

The rating date and time is given as timestamp, I used the following code to extract *rating_year*, *rating_month*, *rating_day_of_week*, and *rating_week_of_year* from the *edx* data set, also I did the same for the *final_holdout_test* data set:

```
## I used as.POSIXct to convert the timestamp into date object, then use format function
## to extract the required info: %Y- for the year, %m- for the month, %w- for the day of
## the week (0 to 6), and %W- for the week of the year (0-53)
edx <- edx %>% mutate(rating_date=as.POSIXct(timestamp,origin="1970-01-01")) %>%
  mutate(rating_year= as.integer(format(rating_date,"%Y")),
         rating_month=as.integer(format(rating_date,"%m")),
         rating_day_of_week=as.integer(format(rating_date,"%w")),
         rating_week_of_year=as.integer(format(rating_date,"%W")))
```

To study the correlation between the features extracted from the *timestamp*, I plotted the correlation between them as follows:

```
cr <- cor(edx %>% select(rating,rating_year,rating_month,
                         rating_day_of_week,rating_week_of_year))
corrplot(cr,addCoef.col="black",number.cex=0.7,tl.cex = 0.7,type = 'lower',method = 'square')
```



We can see that *rating_month* and *rating_week_of_year* are fully correlated, I will drop one of them when building the model, I will choose the one that will improve the model more. Also *rating_year* is slightly negatively correlated with both of the previously mentioned features.

Genres encoding

Genres in the original data set (Table 1) are encoded as a single string of values that are separated by |: for example the movie in the first rating has the genres value of "Comedy|Romance" which means that it is belongs to the Comedy genre and Romance genre. This format is human friendly but can't be used as it is in the models building, I splitted this value into different columns, one column for each possible genre where the value of 1 indicated that movies belongs to the genre and the value of zero indicates that the movie not belong to the genre, this format is called one-hot-coding.

Converting the genres into one-hot-coding format will required those steps: the first step is to separate the values in the genres using the | as splitting character, this will create many rows for the same data set entry where every genre the movie belongs to will add a new row, the second step is to summarize the rows for the same data set entry back into one row, to do so we need to group the rows of the same original data set entry together, that is why the initial step should be adding a unique *rowID* before the splitting.

The following code snippet is used to add a unique *rowID* to the original data set entries

```
edx$rowID = seq(1,nrow(edx))
```

The following code snippet is used to split the genres into multiple rows

```
#Split the genres into single values, now we will have multiple row for the same ratting
#entry. we can identify the entry later by the rowID
edx<- edx %>% separate_rows(genres, sep = "\\|")
```

```
#Get the list of unique genres
gens <- unique(edx$genres)

#Add a column for each genre, for Comedy as example, the column name will be genres_Comedy,
#and it will have either the value of 1 or 0
for(gen in gens)
{
  edx <- edx %>% mutate( "genres_{gen}" := ifelse(genres==gen,1,0))
}

#For movies without any genres we will have genres_(no genres listed) column, the
#following line will remove it.
edx <- edx %>% select(-`genres_(no genres listed)`)
```

We have the following genres:

```
colnames(edx %>% select(starts_with('genres_')))
```

```
##  [1] "genres_Comedy"      "genres_Romance"     "genres_Action"      "genres_Crime"
##  [5] "genres_Thriller"    "genres_Drama"       "genres_Sci_Fi"      "genres_Adventure"
##  [9] "genres_Children"    "genres_Fantasy"     "genres_War"         "genres_Animation"
## [13] "genres_Musical"     "genres_Western"     "genres_Mystery"     "genres_Film_Noir"
## [17] "genres_Horror"      "genres_Documentary" "genres_IMAX"
```

A sample of the data set is shown below, only some columns are displayed for visibility

```
head(edx_before_summarize_sample %>% select(userId,movieId,rating,genres_Comedy,genres_Romance,genres_A
  knitr::kable(caption = 'Data set with some generes - Before grouping')
```

Table 2: Data set with some generes - Before grouping

userId	movieId	rating	genres_Comedy	genres_Romance	genres_Action
1	122	5	1	0	0
1	122	5	0	1	0
1	185	5	0	0	1
1	185	5	0	0	0
1	185	5	0	0	0
1	292	5	0	0	1

As you can see, there is a row for every data entry for each genre with value 1, in the following code snippet, I will use the *summarize* function to group those rows into one row per entry:

```
#Notes:
# - From the non-genres columns we will select: userId, movieId, rating, title,
# release_year, rating_year, rating_month, rating_day_of_week, and rating_week_of_year
# - In the genre related columns, we have columns that contains - which will make further
# processing a little hard (genres_Sci-Fi and genres_Film-Noir),
# I will replace the - with _, namely


edx <- edx %>% group_by(rowID) %>% summarise(
  ## in the following columns I will use first, since all rows belongs to the rating entry
  ## have the same exact value
  userId = first(userId),
  movieId = first(movieId),
```

```
  rating = first(rating),
  title = first(title),
  release_year = first(release_year),
  rating_year = first(rating_year),
  rating_month = first(rating_month),
  rating_day_of_week = first(rating_day_of_week),
  rating_week_of_year = first(rating_week_of_year),
  ## in the following columns I will use sum, since only on row belongs to the rating
  ## entry may have 1 or 0.
  genres_Comedy = sum(`genres_Comedy`),
  genres_Romance = sum(`genres_Romance`),
  genres_Action = sum(`genres_Action`),
  genres_Crime = sum(`genres_Crime`),
  genres_Thriller = sum(`genres_Thriller`),
  genres_Drama = sum(`genres_Drama`),
  genres_Sci_Fi = sum(`genres_Sci-Fi`),
  genres_Adventure = sum(`genres_Adventure`),
  genres_Children = sum(`genres_Children`),
  genres_Fantasy = sum(`genres_Fantasy`),
  genres_War = sum(`genres_War`),
  genres_Animation = sum(`genres_Animation`),
  genres_Musical = sum(`genres_Musical`),
  genres_Western = sum(`genres_Western`),
  genres_Mystery = sum(`genres_Mystery`),
  genres_Film_Noir = sum(`genres_Film-Noir`),
  genres_Horror = sum(`genres_Horror`),
  genres_Documentary = sum(`genres_Documentary`),
  genres_IMAX = sum(`genres_IMAX`)
)
```

A sample of the data set is shown below, only some columns are displayed for visibility

```
head(edx %>% select(userId,movieId,rating,genres_Comedy,genres_Romance,genres_Action)) %>%
  knitr::kable(caption = 'Data set with some generes - After grouping')
```

Table 3: Data set with some generes - After grouping

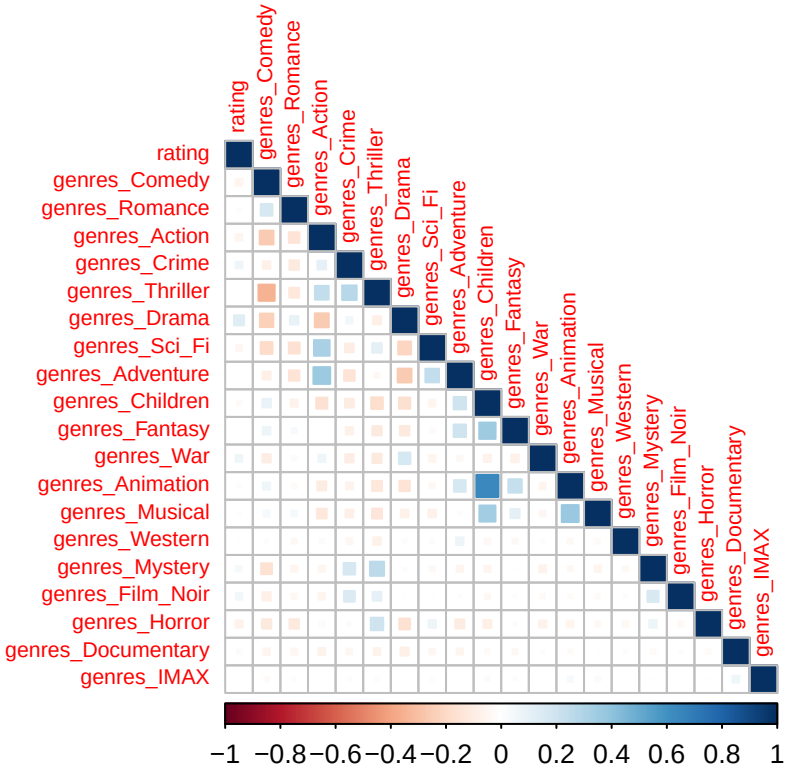
userId	movieId	rating	genres_Comedy	genres_Romance	genres_Action
1	122	5	1	1	0
1	185	5	0	0	1
1	292	5	0	0	1
1	316	5	0	0	1
1	329	5	0	0	1
1	355	5	1	0	0

Now, it is clear that each data entry is in one row, for example, the movie with *movieId* 122 belongs at least to the Comedy and Romance genres at the same time.

I did the same with the *final_holdout_test* data set.

In the following figure, I plotted the correlation between all genres:

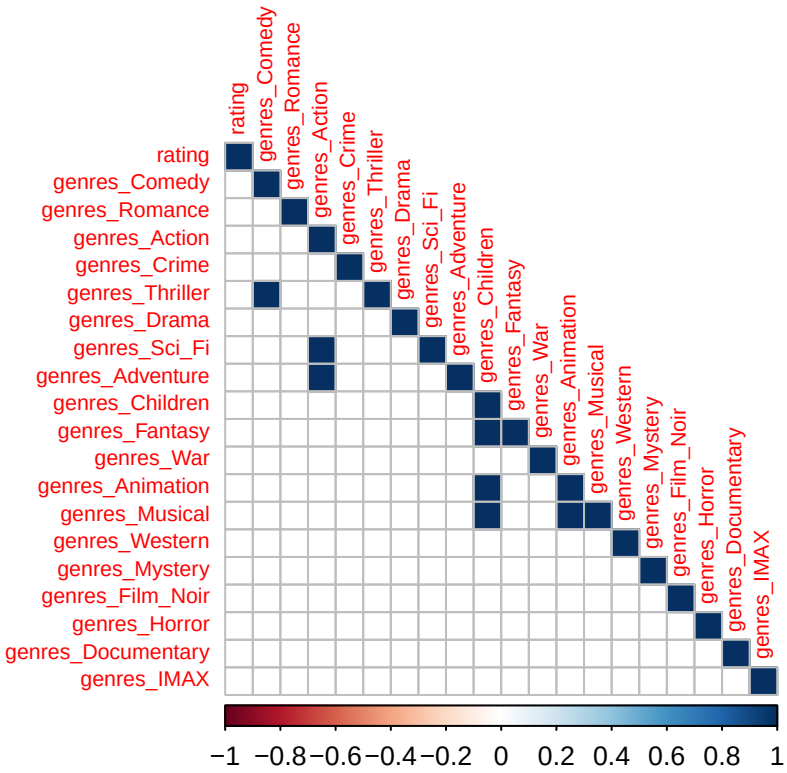
```
gen_correlation = cor(edx%>%select("rating",starts_with('genres_')))
corrplot(gen_correlation,tl.cex = 0.7,type = 'lower',method = 'square')
```



Since we have 19 genres, during the model building, we will assume the linear independence among them, however, it is clear that are we some genres are highly correlated, to maintain the assumption of linear independence, I will assume any correlation value above 0.3 to be highly correlated

In the following correlation plot, the blue square means there is a correlation of 0.3 or higher

```
gen_correlation = cor(edx%>%select("rating",starts_with('genres_')))
corrplot(abs(gen_correlation)>0.3,tl.cex = 0.7,type = 'lower',method = 'square')
```



Based on the above two correlation plot, I have the following notes:

- rating is not highly correlated with any movie genre, however is slightly positively correlated (tend to have high rating) with Drama, War, Crime and Film Noir and slightly negatively correlated (tend to have low rating) with Action, Horror, Comedy and Sci-Fi.
- Action is highly correlated with Sci_Fi and Adventure, this is true, movies in one of those genres is usually belongs to one or both of the others. Action genre will be excluded from the model building.
- Children is highly correlated with Animation and Musical, this is true as well. Children genre will be excluded from the model building.
- Comedy is highly correlated with Thriller, but this time negatively (as per the first correlation plot), this is true as well, Comedy and Thriller can't be in the same movie. Comedy genre will be excluded from the model building.
- Animation is highly correlated with Musical, Animation genre will be excluded from the model building.

Models Bulding

In this section, the data set processed in the previous section will be used to build the prediction model. At first, I tried to use *caret* package to build Linear, Loess, and Random Forest models, but my r-session kept crashing, I debugged the problem and it is resulted from the memory consumption of the training process, I have 64GB RAM, so I mounted another 64GB swap file as virtual memory but the problem got worse and the computer start to crash due to the extreme high CPU utilization (because of copying memory back and forth between the RAM and the swap file). At the end I decided to extended the linear model we built as an example in Chapter 8.

This section is organized as the following: Splitting the edx into training and testing data sets, building he base model using the mean of the ratings (Model 1), then adding the movie effect (Model 2), then adding the user effect to Model 2 to get Model 3, after that adding the release date effect to get Model 4, then Model 5

is constructed by adding the rating date and time effect to Model 4, to get Model 6 the genres effect is added to Model 5, then regularization is done on Model 6 to get Model7, Finally the best regularized model is used to predict the ratings in the *final_holdout_test* data set.

Splitting the edx data sets into training and testing data sets.

I splitted the *edx* data set into two data sets, the *edx_train* data set that contains 80% of the entries and the *edx_test* data set that contains 20% of the entries, I used the following code for splitting it:

```
set.seed(1, sample.kind="Rounding") # I am using R 4.3.1
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
edx_train <- edx[-test_index,]
temp <- edx[test_index,]

# Make sure userId and movieId in the edx_test are also in edx_train
edx_test <- temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Add rows removed from final edx_test back into edx_train set
removed <- anti_join(temp, edx_test)
edx_train <- rbind(edx_train, removed)
```

The following code is to double check if we got the 80/20 split

```
test_perc <- nrow(edx_test)/nrow(edx)
train_perc <- nrow(edx_train)/nrow(edx)
sprintf('We got %1.5f%% for the testing set, and %1.5f%% fot the training set',
        test_perc*100,train_perc*100)
```

```
## [1] "We got 19.99950% for the testing set, and 80.00050% fot the training set"
```

Model 1: Only the mean of the ratings

I started with the base-line model; using the average rating in the training data set as predicted rating for all entries in the testing data set.

The following code is to calculate the average rating and to use it for the prediction, the RMSE value for the predicted values is also calculated.

```
mu <- mean(edx_train$rating)
model1_rmse <- RMSE(edx_test$rating, mu)
sprintf('Model-1 RMSE is %1.9f',model1_rmse)
```

```
## [1] "Model-1 RMSE is 1.059904289"
```

Model 2: Movie Effect

The movie effect is added to the model as in the following code:

```
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
predicted_ratings <- mu + edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model2_rmse <- RMSE(predicted_ratings, edx_test$rating)
sprintf('Model-2 RMSE is %1.9f',model2_rmse)
```

```
## [1] "Model-2 RMSE is 0.943742914"
```

Model 3: Movie + User Effect

The user effect is also added to build the third model as in the following code:

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


predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model3_rmse <- RMSE(predicted_ratings, edx_test$rating)
sprintf('Model-3 RMSE is %1.9f',model3_rmse)
```

```
## [1] "Model-3 RMSE is 0.865931913"
```

Model 4: Movie + User + Release Year Effect

To improve the model performance, I the release year effect is included in the model as in the following code:

```
rel_year_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(release_year) %>%
  summarize(b_ry = mean(rating - mu - b_i - b_u))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  mutate(pred = mu + b_i + b_u + b_ry) %>%
  pull(pred)

model4_rmse <- RMSE(predicted_ratings, edx_test$rating)
sprintf('Model-4 RMSE is %1.9f',model4_rmse)
```

```
## [1] "Model-4 RMSE is 0.865611708"
```

Model 5: Movie + User + Release Year Effect + Rating Date and Time

I had 4 features extracted from the rating timestamp which are: *rating_year*, *rating_month*, *rating_day_of_week*, and *rating_week_of_year*, in the previous section we showed that *rating_month* and *rating_week_of_year* are fully correlated and we have to choose one of them to include in the model.

In the following code snippet,those 4 features were added one at a time independently to the previous model to see which one will improve the model performance.

```r
#will rating year increase model performance? let's see
rat_year_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  group_by(rating_year) %>%
  summarize(b_rat_y = mean(rating - mu - b_i - b_u - b_ry))


predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by='rating_year') %>%
  mutate(pred = mu + b_i + b_u + b_ry + b_rat_y ) %>%
  pull(pred)

model5_rat_year_rmse <- RMSE(predicted_ratings, edx_test$rating)

#will the rating month increase model performance? let's see
rat_m_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  group_by(rating_month) %>%
  summarize(b_rat_m = mean(rating - mu - b_i - b_u - b_ry))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_m_avgs, by='rating_month') %>%
  mutate(pred = mu + b_i + b_u + b_ry +  b_rat_m) %>%
  pull(pred)

model5_rat_month_rmse <- RMSE(predicted_ratings, edx_test$rating)

#will the rating week of the year increase model performance? let's see
rat_wy_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  group_by(rating_week_of_year) %>%
  summarize(b_rat_wy = mean(rating - mu - b_i - b_u - b_ry))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  mutate(pred = mu + b_i + b_u + b_ry +  b_rat_wy) %>%
  pull(pred)
```

```
model5_rat_wy_rmse <- RMSE(predicted_ratings, edx_test$rating)

#will the day of the week increase model performance? let's see
rat_dw_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  group_by(rating_day_of_week) %>%
  summarize(b_rat_dw = mean(rating - mu - b_i - b_u - b_ry))


predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_dw_avgs, by='rating_day_of_week') %>%
  mutate(pred = mu + b_i + b_u + b_ry + b_rat_dw) %>%
  pull(pred)

model5_rat_dw_rmse <- RMSE(predicted_ratings, edx_test$rating)

# cat is used here to print the new lines, otherwise, they will be printed as raw \n
cat(sprintf(" Base RMSE (Model 4) = %1.9f
 RMSE (Rating Year)  = %1.9f
 RMSE (Rating Month) = %1.9f
 RMSE (Week of Year) = %1.9f
 RMSE (Day of Week)  = %1.9f",
 model4_rmse,model5_rat_year_rmse,model5_rat_month_rmse,model5_rat_wy_rmse,
 model5_rat_dw_rmse ))
```

```
##  Base RMSE (Model 4) = 0.865611708
##  RMSE (Rating Year)  = 0.865541684
##  RMSE (Rating Month) = 0.865604092
##  RMSE (Week of Year) = 0.865599674
##  RMSE (Day of Week)  = 0.865612499
```

For the fully correlated features (*rating_month* and *rating_week_of_year*), the *rating_week_of_year* improved the RMSE slightly more than the *rating_month* so the *rating_month* will be dropped, also *rating_day_of_week* didn't improv the RMSE at all, so I it will be dropped as well, In the code snippet model I added the *rating_year* and *rating_week_of_year* to the model 4 to construct the model 5.

```
rat_wy_updated_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  group_by(rating_week_of_year) %>%
  summarize(b_rat_wy = mean(rating - mu - b_i - b_u - b_ry - b_rat_y))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
```

```
  mutate(pred = mu + b_i + b_u + b_ry + b_rat_y +  b_rat_wy) %>%
  pull(pred)

model5_rat_ywy_rmse <- RMSE(predicted_ratings, edx_test$rating)
sprintf('Model-5 RMSE is %1.9f',model5_rat_ywy_rmse)
```

```
## [1] "Model-5 RMSE is 0.865534463"
```

Model 6: Movie + User + Release Year Effect + Rating Date (Year + Week of Year) + Genres

Dealing with genres is tricky since a movie can belong to multiple genre at the same time, to facilitate this task, I will assume that the genres are linearly independent. Then the effect of the genres on the movie i (G_i) can be modeled as:

$$G_i = \sum_{j=1}^{m} g_{i,j} \times u_j$$

where m is the number of genres, u_j is the average effect for the genre j, while $g_{i,j}$ will be 1 if the movie i belongs to the genre j, 0 otherwise. In the following code I will find the u_j for every genre j,

```
# The genres are represented in the data set as one-hot-coding, if the movie belongs
# to the genre the value will be 1, otherwise 0
# The effect of the genres on the rating is the sum of the effects of all genres the
# movie belongs to, so it the the sum of the following for all genres

# the genre_value (0 or 1) * genre effect avg.

# now lets find the genre effects for all the selected genres:
# "genres_Romance","genres_Crime","genres_Thriller","genres_Drama","genres_Sci_Fi",
# "genres_Adventure","genres_Fantasy","genres_War","genres_Musical"
# "genres_Western","genres_Mystery","genres_Film_Noir","genres_Horror",
# "genres_Documentary","genres_IMAX"

genres_Romance_avg <- edx_train %>% filter(genres_Romance==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

genres_Crime_avg <- edx_train %>% filter(genres_Crime==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

genres_Thriller_avg <- edx_train %>% filter(genres_Thriller==1) %>%
  left_join(movie_avgs,by='movieId') %>%
```

```
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

genres_Drama_avg <- edx_train %>% filter(genres_Drama==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Sci_Fi_avg` <- edx_train %>% filter(`genres_Sci_Fi`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Adventure_avg` <- edx_train %>% filter(`genres_Adventure`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Fantasy_avg` <- edx_train %>% filter(`genres_Fantasy`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_War_avg` <- edx_train %>% filter(`genres_War`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Musical_avg` <- edx_train %>% filter(`genres_Musical`==1) %>%
```

```
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Western_avg` <- edx_train %>% filter(`genres_Western`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Mystery_avg` <- edx_train %>% filter(`genres_Mystery`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Film_Noir_avg` <- edx_train %>% filter(`genres_Film_Noir`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Horror_avg` <- edx_train %>% filter(`genres_Horror`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)

`genres_Documentary_avg` <- edx_train %>% filter(`genres_Documentary`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)
```

```
`genres_IMAX_avg` <- edx_train %>% filter(`genres_IMAX`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=mean(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)) %>%
  pull(b_g_avg)
```

The predicted ratings for the test data set can be found using the following code:

```
predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_updated_avgs, by='rating_week_of_year') %>%
  mutate(pred = mu + b_i + b_u + b_ry + b_rat_y +  b_rat_wy +
           genres_Romance_avg*genres_Romance+
           genres_Crime_avg*genres_Crime+
           genres_Thriller_avg*genres_Thriller+
           genres_Drama_avg*genres_Drama+
           genres_Sci_Fi_avg*`genres_Sci_Fi`+

           genres_Adventure_avg*`genres_Adventure`+
           genres_Fantasy_avg*`genres_Fantasy`+
           genres_War_avg*`genres_War`+
           genres_Musical_avg*`genres_Musical`+
           genres_Western_avg*`genres_Western`+
           genres_Mystery_avg*`genres_Mystery`+
           genres_Film_Noir_avg*`genres_Film_Noir`+
           genres_Horror_avg*`genres_Horror`+
           genres_Documentary_avg*`genres_Documentary`+
           genres_IMAX_avg*`genres_IMAX`

         ) %>%
  pull(pred)

model6_rmse <- RMSE(predicted_ratings, edx_test$rating)
sprintf('Model-6 RMSE is %1.9f',model6_rmse)
```

```
## [1] "Model-6 RMSE is 0.865421319"
```

Model 7: Regularized Model 6

To further improve the model performance, The final model will be regularized, regularization works by penalizing the large estimates that are formed using small sample sizes, so that the extreme cases will have the minimal effect.

Since the model building and predicting will be used many times, I prepared two functions, the first is called *model_build* to build the model and the other call *model_predict* to do prediction based on the averages produced by the *model_build* function, the two functions are given below:

```
# the function to build the model: dataset_train is the data set to be used,
# l is the regularization level
model_build  <- function(dataset_train,l){
```

```
## generate all the averages
mu <- mean(dataset_train$rating)

movie_avgs <- dataset_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+1))

user_avgs <- dataset_train %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+1))

rel_year_avgs <- dataset_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(release_year) %>%
  summarize(b_ry = sum(rating - mu - b_i - b_u)/(n()+1))

rat_year_avgs <- dataset_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  group_by(rating_year) %>%
  summarize(b_rat_y = sum(rating - mu - b_i - b_u - b_ry)/(n()+1))

rat_wy_avgs <- dataset_train %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  group_by(rating_week_of_year) %>%
  summarize(b_rat_wy = sum(rating - mu - b_i - b_u - b_ry - b_rat_y)/(n()+1))


genres_Romance_avg <- dataset_train %>% filter(genres_Romance==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_Crime_avg <- dataset_train %>% filter(genres_Crime==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_Thriller_avg <- dataset_train %>% filter(genres_Thriller==1) %>%
```

```
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_Drama_avg <- dataset_train %>% filter(genres_Drama==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_Sci_Fi_avg <- dataset_train %>% filter(`genres_Sci_Fi`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_Adventure_avg <- dataset_train %>% filter(`genres_Adventure`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_Fantasy_avg <- dataset_train %>% filter(`genres_Fantasy`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)

genres_War_avg <- dataset_train %>% filter(`genres_War`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+1)) %>%
  pull(b_g_avg)
```

```
genres_Musical_avg <- dataset_train %>% filter(`genres_Musical`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
  pull(b_g_avg)

genres_Western_avg <- dataset_train %>% filter(`genres_Western`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
  pull(b_g_avg)

genres_Mystery_avg <- dataset_train %>% filter(`genres_Mystery`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
  pull(b_g_avg)

genres_Film_Noir_avg <- dataset_train %>% filter(`genres_Film_Noir`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
  pull(b_g_avg)

genres_Horror_avg <- dataset_train %>% filter(`genres_Horror`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
  pull(b_g_avg)

genres_Documentary_avg <- dataset_train %>% filter(`genres_Documentary`==1) %>%
  left_join(movie_avgs,by='movieId') %>%
  left_join(user_avgs,by='userId') %>%
  left_join(rel_year_avgs, by='release_year') %>%
  left_join(rat_year_avgs, by="rating_year") %>%
  left_join(rat_wy_avgs, by='rating_week_of_year') %>%
  summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
  pull(b_g_avg)
```

```
  genres_IMAX_avg <- dataset_train %>% filter(`genres_IMAX`==1) %>%
    left_join(movie_avgs,by='movieId') %>%
    left_join(user_avgs,by='userId') %>%
    left_join(rel_year_avgs, by='release_year') %>%
    left_join(rat_year_avgs, by="rating_year") %>%
    left_join(rat_wy_avgs, by='rating_week_of_year') %>%
    summarise(b_g_avg=sum(rating-mu-b_i-b_u- b_ry - b_rat_y - b_rat_wy)/(n()+l)) %>%
    pull(b_g_avg)

  ## return the list of the averages
  list(
    mu = mu,
    movie_avgs = movie_avgs,
    user_avgs = user_avgs,
    rel_year_avgs = rel_year_avgs,
    rat_year_avgs = rat_year_avgs,
    rat_wy_avgs = rat_wy_avgs,
    genres_Romance_avg = genres_Romance_avg,
    genres_Crime_avg = genres_Crime_avg,
    genres_Thriller_avg = genres_Thriller_avg,
    genres_Drama_avg = genres_Drama_avg,
    genres_Sci_Fi_avg = genres_Sci_Fi_avg,
    genres_Adventure_avg = genres_Adventure_avg,
    genres_Fantasy_avg = genres_Fantasy_avg,
    genres_War_avg = genres_War_avg,
    genres_Musical_avg = genres_Musical_avg,
    genres_Western_avg = genres_Western_avg,
    genres_Mystery_avg = genres_Mystery_avg,
    genres_Film_Noir_avg = genres_Film_Noir_avg,
    genres_Horror_avg = genres_Horror_avg,
    genres_Documentary_avg = genres_Documentary_avg,
    genres_IMAX_avg = genres_IMAX_avg
  )

}

#This function is to find the predictions and the rmse, dataset_test is the testing dataset
# avgs are the averages we got from the model_build function
model_predict <- function(dataset_test,avgs)
{
  dataset_test =data.frame(dataset_test)
  predicted_ratings <- dataset_test %>%
    left_join(avgs$movie_avgs, by='movieId') %>%
    left_join(avgs$user_avgs, by='userId') %>%
    left_join(avgs$rel_year_avgs, by='release_year') %>%
    left_join(avgs$rat_year_avgs, by="rating_year") %>%
    left_join(avgs$rat_wy_avgs, by='rating_week_of_year') %>%
    mutate(pred = mu + b_i + b_u + b_ry + b_rat_y +  b_rat_wy +
             avgs$genres_Romance_avg*genres_Romance+
             avgs$genres_Crime_avg*genres_Crime+
             avgs$genres_Thriller_avg*genres_Thriller+
             avgs$genres_Drama_avg*genres_Drama+
             avgs$genres_Sci_Fi_avg*`genres_Sci_Fi`+
```

```
            avgs$genres_Adventure_avg*`genres_Adventure`+
            avgs$genres_Fantasy_avg*`genres_Fantasy`+
            avgs$genres_War_avg*`genres_War`+
            avgs$genres_Musical_avg*`genres_Musical`+
            avgs$genres_Western_avg*`genres_Western`+
            avgs$genres_Mystery_avg*`genres_Mystery`+
            avgs$genres_Film_Noir_avg*`genres_Film_Noir`+
            avgs$genres_Horror_avg*`genres_Horror`+
            avgs$genres_Documentary_avg*`genres_Documentary`+
            avgs$genres_IMAX_avg*`genres_IMAX`

    ) %>%
    pull(pred)

  rmse <- RMSE(predicted_ratings, dataset_test$rating)
  list(predictions = predicted_ratings,rmse=rmse)
}
```

Regularization is performed using the following code:

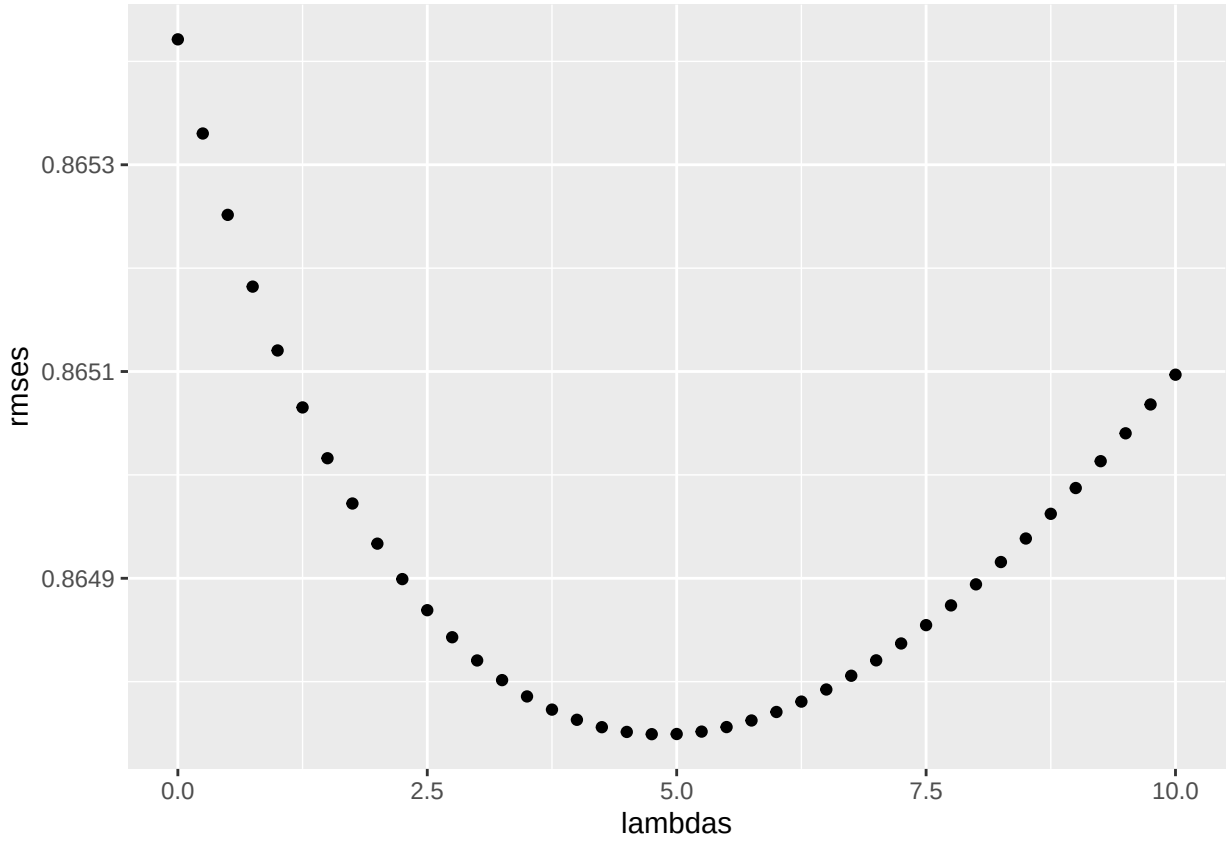
```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  avgs = model_build(edx_train,l)
  result = model_predict(edx_test,avgs)
  return(result$rmse)

})
```

The following plot showing the RMSE against the regularization factor (lambda), the lambda that gives us the lowest RMSE is the best lambda.

```
qplot(lambdas, rmses)
```



The best lambda is:

```
l = lambdas[which.min(rmses)]
l
```

```
## [1] 4.75
```

The final regularized model is given in the following code snippet:

```
avgs = model_build(edx_train,l ) # get the avgs for the edx_train using the best l
model7_rmse = model_predict(edx_test,avgs)$rmse # get the rmse for edx_test
sprintf('Model-7 RMSE is %1.9f',model7_rmse)
```

```
## [1] "Model-7 RMSE is 0.864749211"
```

Final testing on the *final_holdout_test* data set

In this final sub-section, The last model will be tested using the *final_holdout_test* data set, since I developed the function to build the model and predict the ratings, this sub-section is very short.

To get the most accurate model, The whole *edx* data set will be used to build the model one last time, this final model is used to predict the ratings in the *final_holdout_test*. The code for building the last model and testing it is given below:

```
avgs = model_build(edx,l ) # get the avgs for the edx using the best l

final_holdout_test_rsme <- model_predict(final_holdout_test,avgs)$rmse # get final_holdout_test
sprintf('RMSE for the final_holdout_test is is %1.9f',final_holdout_test_rsme)
```

```
## [1] "RMSE for the final_holdout_test is is 0.864299672"
```

The RMSE of 0.864299672 is less than the target RMSE of 0.86490, so the goal is achieved.

Results

In this project I built 7 different models, the following table shows their names along with their RMSE values achieved on the *edx_test* data sets:

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

Method	RMSE
Model1: Only the mean of the ratings	1.0599043
Model 2: Movie Effect	0.9437429
Model 3: Movie + User Effect	0.8659319
Model 4: Movie + User + Release Year Effect	0.8656117
Model 5: Movie + User + Release Year Effect + Rating Date (Year + Week of Year)	0.8655345
Model 6: Movie + User + Release Year Effect + Rating Date (Year + Week of Year) + Genres	0.8654213
Model 7: Regularized Model 6	0.8647492

As you can see the worst model was the Model 1 which is the naive model that use the mean of the ratings in the *edx_train* to predict the rating in the *edx_test*, when adding the movie effect (Model 2) the RMSE dropped by 11% to 0.9437, when adding the user effect (Model 3) the RMSE dropped by 8% to be 0.8659, in Model 4 I added the release year effect which slightly drop the RMSE by 0.03% to be 0.8656, the addition of the rating date effect in Model 5 will slightly drop the RMSE value by 0.009% to be 0.8655, the same happened when adding the genres effect in Model 6 to have in which the RMSE dropped by 0.013% to be 0.8654, doing regularization on Model 6 resulted in the Model 7 that dropped the RMSE value by 0.077% to be 0.8647.

It is clear that the most important features that increased the model performance were the movie effect, the user effect and the regularization, but I kept all features in the final model to get the lowest possible RMSE.

To include as much data as I can in the model building, I trained the selected model one last time using the whole *edx* data set, I used this final model to predict the rating in the *final_holdout_test* data set, the RMSE for the predictions was 0.864299 which less than the target RMSE of 0.86490

Conclusion

In this project I studied the MovieLens data set, then I did features engineering to extract and transform some features. After that, I built 7 incremental linear models to predict the movie rating in the testing data set, the last final model achieved an RMSE value of 0.8647492 on the *edx_test* data set, and an RMSE value of 0.864299672 on a never seen before *final_holdout_test* data set, which lass better than the RMSE target value specified in the project requirements.

The current model can be still be improved if we assumed the genres not to be linearly independent and we added them incrementally like other features, not all at once. I could also have better models if I used the *caret* package with gamLoess or RandomForest to capture the possible non-linear relation between the features and the rating, I tried this but I couldn't do it due to the limited amount of memory on my workstation.

Since we have a relatively big data set, deep neural networks could be used in this problem, since the data will be fed in batches, the limited memory will not be an issue, but I felt this will be beyond the course scope.