# Movielens Recommendation System - Capstone Project

*Harsh Navin Gupta*

*10 January 2020*

### Dataset Analysis

For this project, we make use of the **Movielens 10M Dataset**. The Movielens datasets were collected by the *GroupLens Research Project* at the *University of Minnesota*.

In this project, we make a *Recommendation System*, which recommends movies to watch to the users, based on the various features that are present in the dataset, which are the features that are present in the ratings given by other users.

The **Movielens 10M Dataset** consists of the follows features :
1. **Movie ID** : It is an unique numeric value assigned to every movie, that has been rated and is available in the dataset.
2. **User ID** : It is an unique numeric value assigned to every user, who has rated atleast one or more movies.
3. **Title** : This is a character field, which stores the title of the movie.
4. **Genre** : This is a character field, which stores the genre of the movie.
5. **Timestamp** : This field stores the timestamp, for the time when the rating was given. It is a numeric value, which the stores the time in reference to *January 1,1970*.

The **Movielens 10M Dataset** can be downloaded from https://grouplens.org/datasets/movielens/10m/.

### Dataset Download

The dataset is available in *zip* file, which consists of many files, such as *movies.dat*, *genres.dat*, which are extracted and combined together to form a tidy dataset.

```
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes",
                                       repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")

library(ggthemes)
library(tidyverse)
library(caret)
library(lubridate)
library(knitr)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
```

```r
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>%
  mutate(movieId = as.numeric(levels(movieId))[movieId],title = as.character(title),genres = as

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = movielens$rating,
                                  times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

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

After the dataset has been downloaded we split the dataset into two parts :
1. edx : This contains 90% of the **Movielens 10M Dataset**, and is used for training of our models.
2. validation : This contrains 10% of the **Movielens 10M Dataset**, and is used for validation of our models.

### Dataset Preprocessing

First we observe a few entries of our **edx** dataset.

```
##   userId movieId rating timestamp                    title
## 1      1     122      5 838985046         Boomerang (1992)
```

```
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1               Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

Here, it is observed that the title of the movie contains the movie name along with the release year, thus we extract the *release year* from the movie title and store it in a new column named *year*.

```
edx <- edx %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)))

validation <- validation %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)))
```

One movie can be listed under many genres, and as seen in our dataset, the genres for all movies, are provided as character strings, seperated by | seperator. Thus, the genres are split, and a new entry is made for every genre listed.

```
split_edx <- edx %>% separate_rows(genres,sep = "\\|")
split_validation <- validation %>% separate_rows(genres,sep = "\\|")
```

The first few entries of our edx dataset after the pre-processing.

```
head(split_edx)
```

```
##   userId movieId rating timestamp            title   genres year
## 1      1     122      5 838985046 Boomerang (1992)   Comedy 1992
## 2      1     122      5 838985046 Boomerang (1992)  Romance 1992
## 3      1     185      5 838983525  Net, The (1995)   Action 1995
## 4      1     185      5 838983525  Net, The (1995)    Crime 1995
## 5      1     185      5 838983525  Net, The (1995) Thriller 1995
## 6      1     292      5 838983421  Outbreak (1995)   Action 1995
```

### Exploratory Data Analysis

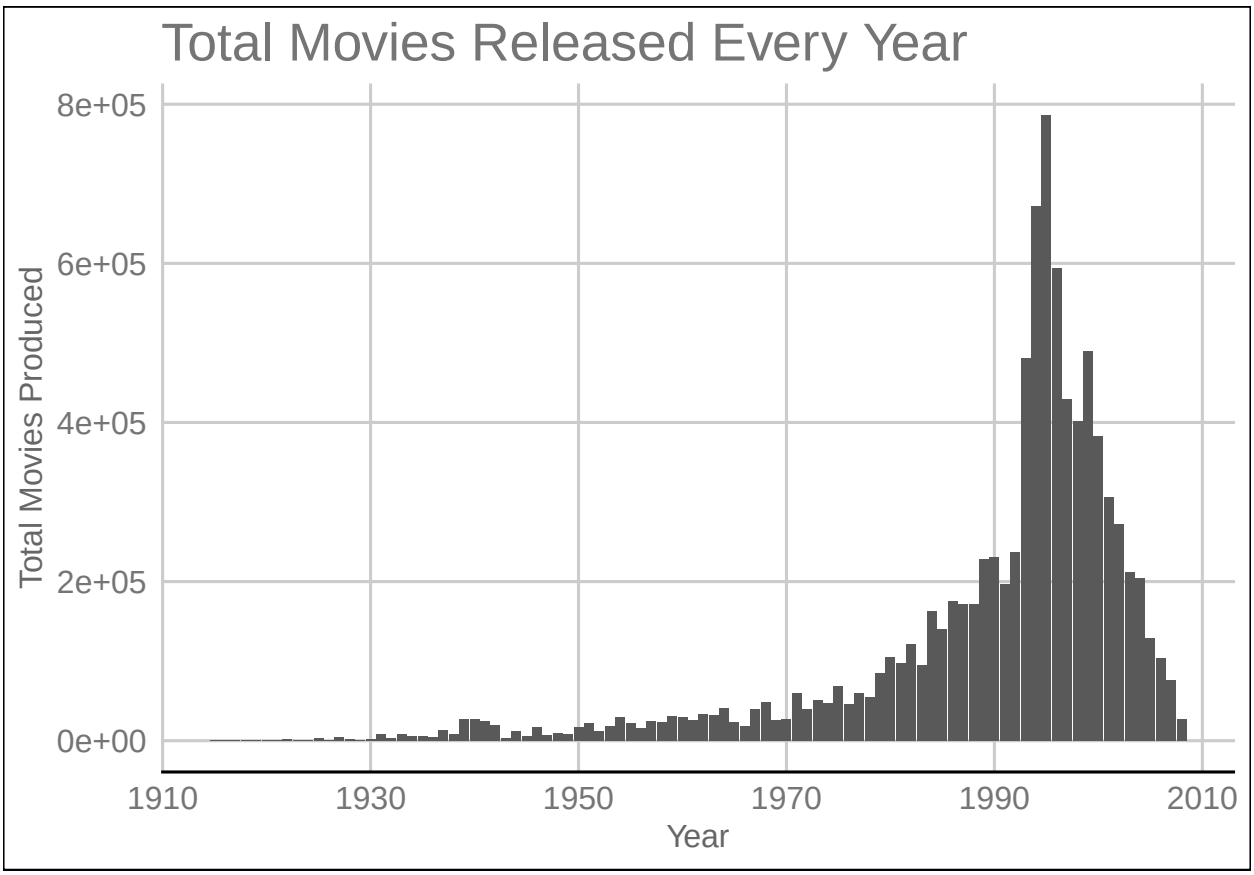
The **Movielens 10M Dataset** contains ratings given by multiple users, and every user has given a rating to one or movies. Hence, we determine the number of users who have given the rating, and number of movies that have been rated.

```
edx %>%
  summarise(Users=n_distinct(userId),Movies=n_distinct(movieId)) %>%
  knitr::kable()
```

| Users | Movies |
|---|---|
| 69878 | 10677 |

Now, we visulise the distribution of the number of movies which have been released over the different years.

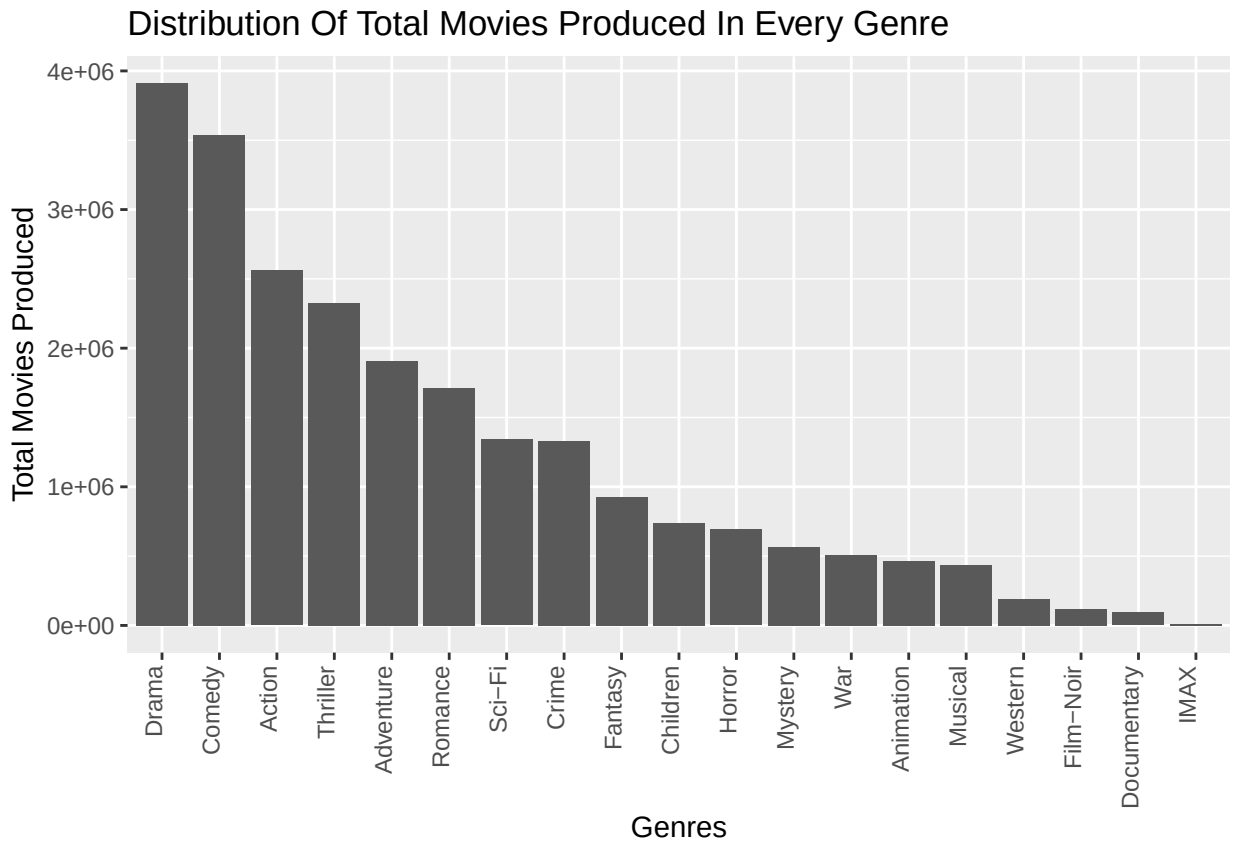
```
edx %>% group_by(year) %>%
  summarise(total_movies = n()) %>%
  ggplot(aes(year,total_movies)) +
  geom_bar(stat = "identity") +
  theme_gdocs() +
  xlab("Year") +
  ylab("Total Movies Produced") +
  ggtitle("Total Movies Released Every Year")
```




In the **Movielens 10M Dataset**, every movie has a property *genres*. A single movie may be consisting of many genre values. Here, we visualise the distribution of number of movies which have made for the various *genres*.

```
split_edx %>% filter(genres != "(no genres listed)") %>%
  group_by(genres) %>%
  summarise(total_movies = n()) %>%
  ggplot(aes(reorder(genres,-total_movies),total_movies)) +
```

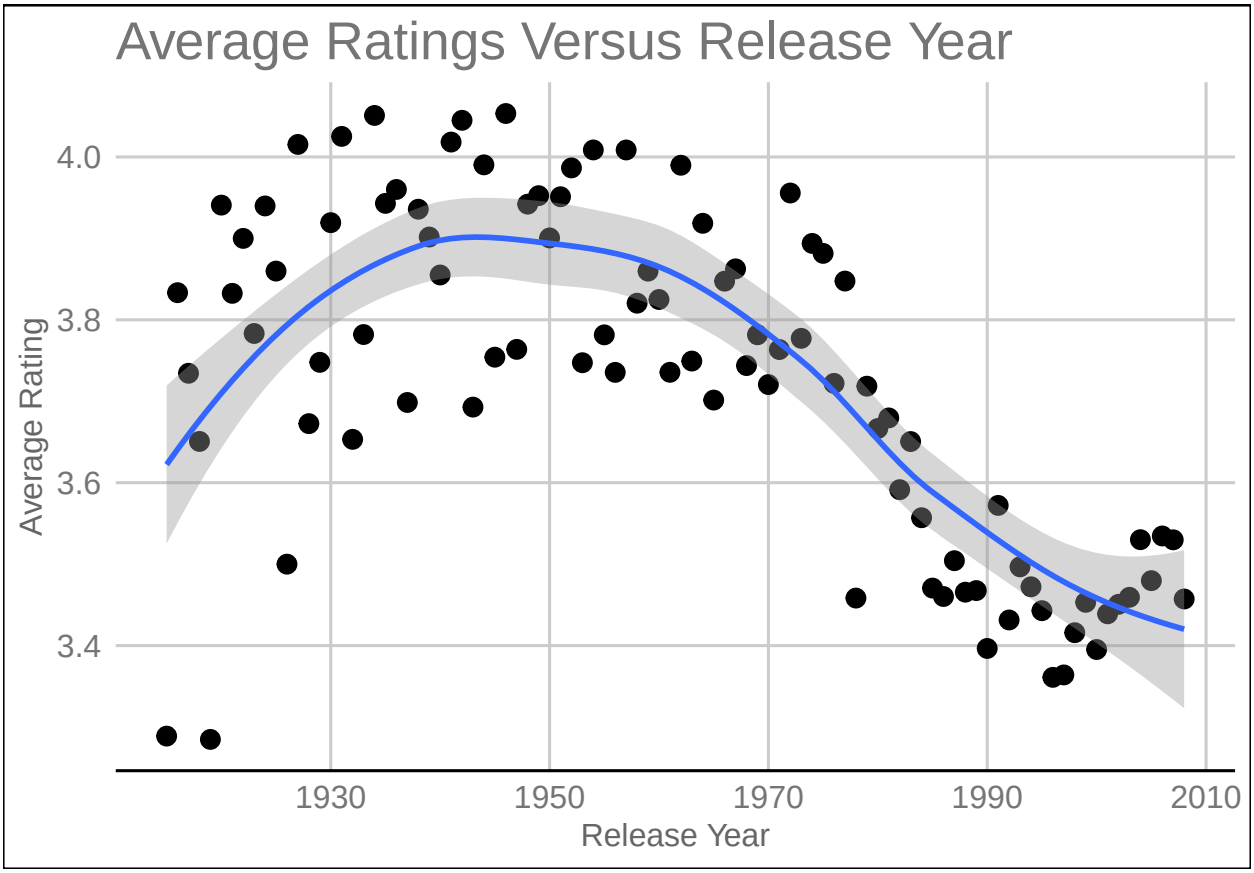
```
  geom_bar(stat = "identity") +
  theme(axis.text.x=element_text(angle=90,hjust=1, vjust=0)) +
  xlab("Genres") +
  ylab("Total Movies Produced") +
  ggtitle("Distribution Of Total Movies Produced In Every Genre")
```




Now we observe the trend followed by the *average rating (mean rating)* given to the movies based on their release years.

```
edx %>% group_by(year) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot(aes(year,mean_rating)) +
  geom_point(size = 3) +
  geom_smooth() +
  ggtitle("Average Ratings Versus Release Year") +
  xlab("Release Year") +
  ylab("Average Rating") +
  theme_gdocs()
```

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



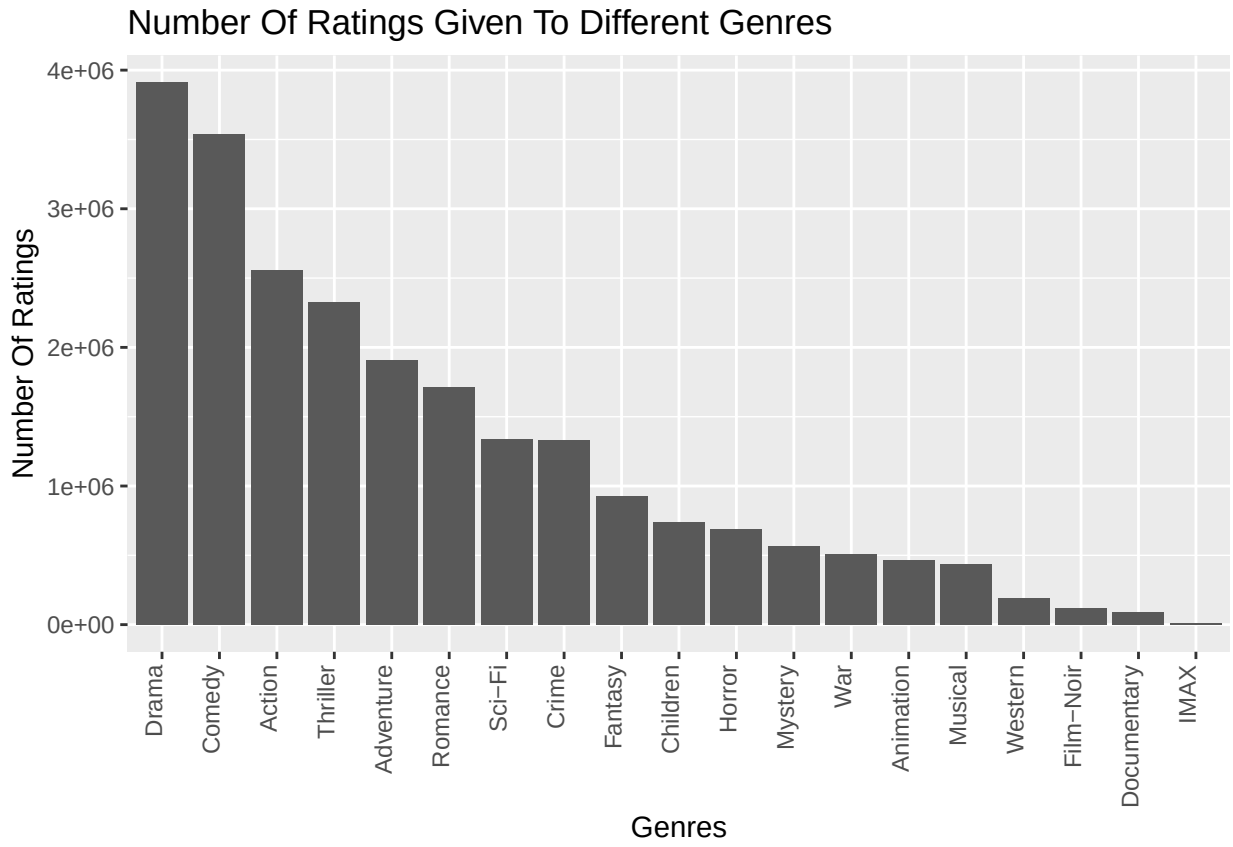

Now we explore the number of ratings given to every genre.

```
split_edx %>% filter(genres != "(no genres listed)") %>%
  group_by(genres) %>%
  summarise(total_ratings = n()) %>% knitr::kable()
```

| genres | total_ratings |
|---|---|
| Action | 2560545 |
| Adventure | 1908892 |
| Animation | 467168 |
| Children | 737994 |
| Comedy | 3540930 |
| Crime | 1327715 |
| Documentary | 93066 |
| Drama | 3910127 |
| Fantasy | 925637 |
| Film-Noir | 118541 |
| Horror | 691485 |
| IMAX | 8181 |
| Musical | 433080 |
| Mystery | 568332 |
| Romance | 1712100 |
| Sci-Fi | 1341183 |

| genres | total_ratings |
|---|---|
| Thriller | 2325899 |
| War | 511147 |
| Western | 189394 |

```
split_edx %>% filter(genres != "(no genres listed)") %>%
  group_by(genres) %>%
  summarise(total_ratings = n()) %>%
  ggplot(aes(reorder(genres,-total_ratings),total_ratings)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x=element_text(angle=90,hjust=1, vjust=0)) +
  ylab("Number Of Ratings") +
  xlab("Genres") +
  ggtitle("Number Of Ratings Given To Different Genres")
```



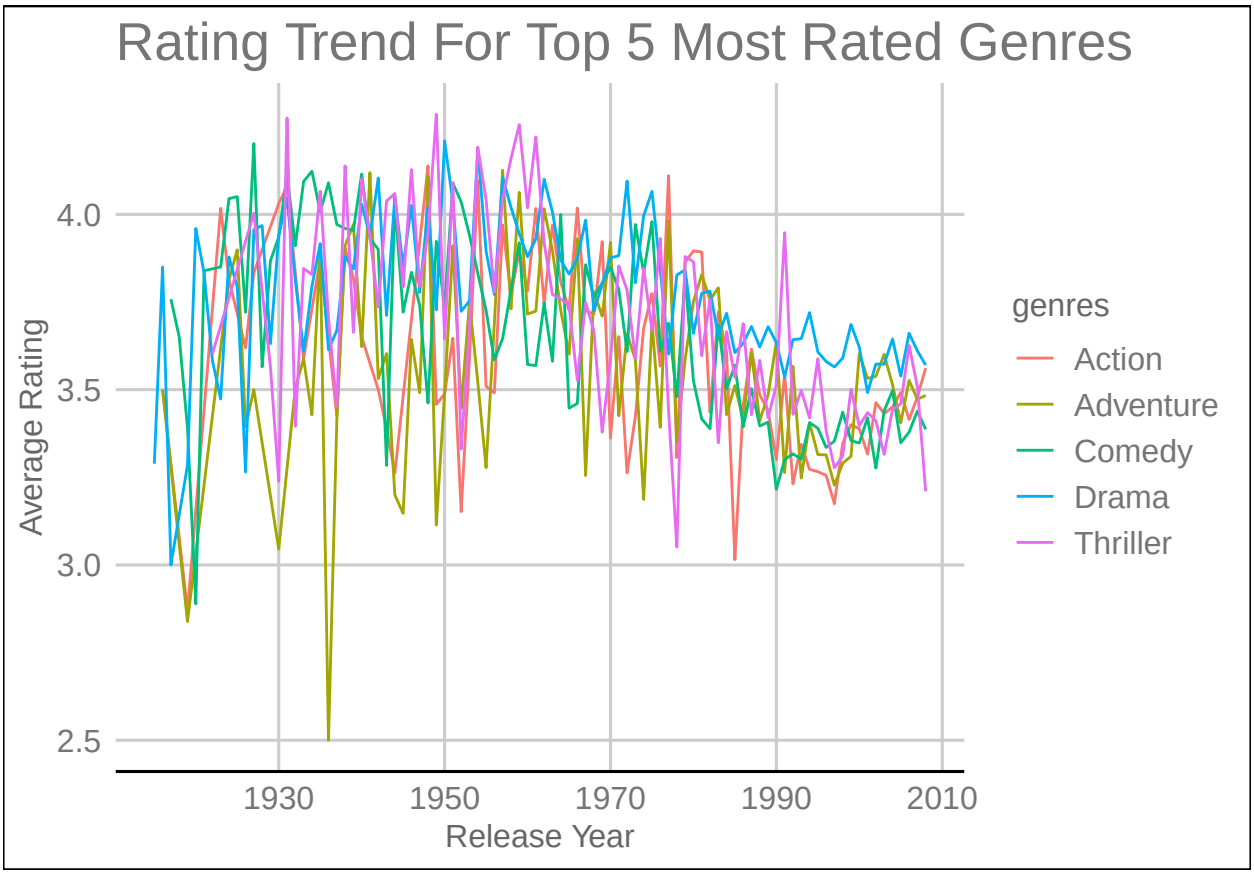

Now we filter out and view the *Top 5 Most Rated Movie Genres*.

```
split_edx %>% group_by(genres) %>%
  summarise(total_ratings = n()) %>%
  arrange(desc(total_ratings)) %>%
  head(5) %>% knitr::kable()
```

| genres | total_ratings |
|---|---|
| Drama | 3910127 |
| Comedy | 3540930 |
| Action | 2560545 |
| Thriller | 2325899 |
| Adventure | 1908892 |

Now we visualise the trend of the average ratings given over the years to *Top 5 Most Rated Movie Genres*.

```
sel_g <- c("Drama","Comedy","Action","Thriller","Adventure")
split_edx %>%
  filter(genres %in% sel_g) %>%
  group_by(year,genres) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot(aes(year,mean_rating,color = genres)) +
  geom_line() +
  ggtitle("Rating Trend For Top 5 Most Rated Genres") +
  xlab("Release Year") +
  ylab("Average Rating") +
  theme_gdocs()
```




Now we filter out the *Top 10 Most Rated Movies*.

```
edx %>% group_by(title) %>%
  summarise(total_ratings = n()) %>%
  arrange(desc(total_ratings)) %>%
  head(10) %>% knitr::kable()
```

| title | total_ratings |
|---|---|
| Pulp Fiction (1994) | 31362 |
| Forrest Gump (1994) | 31079 |
| Silence of the Lambs, The (1991) | 30382 |
| Jurassic Park (1993) | 29360 |
| Shawshank Redemption, The (1994) | 28015 |
| Braveheart (1995) | 26212 |
| Fugitive, The (1993) | 25998 |
| Terminator 2: Judgment Day (1991) | 25984 |
| Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) | 25672 |
| Apollo 13 (1995) | 24284 |

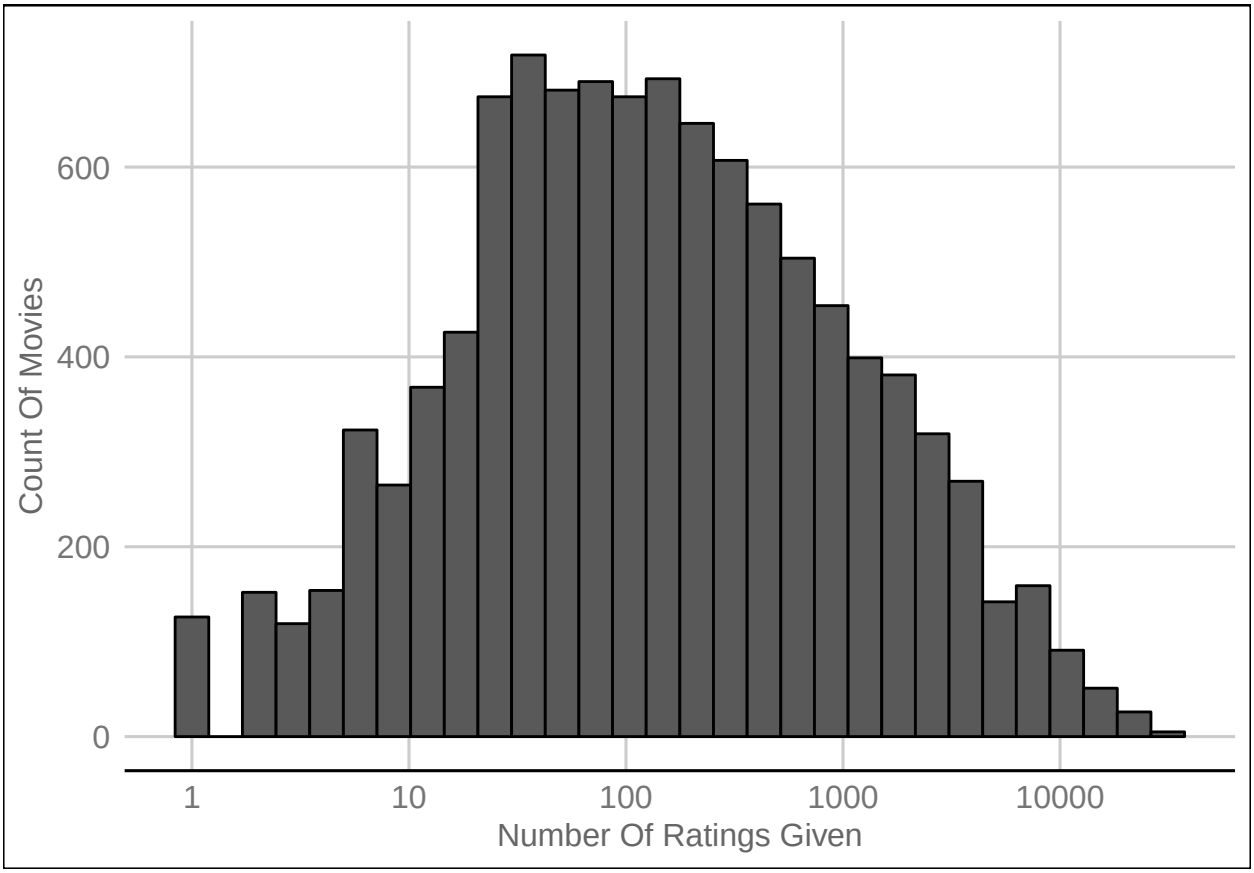
Now we filter out the *Top 10 Highest Rated Movies*.

```
edx %>% group_by(title) %>%
  summarise(average_rating = mean(rating)) %>%
  arrange(desc(average_rating)) %>% head(10)
```

```
## # A tibble: 10 x 2
##    title                                                    average_rating
##    <chr>                                                             <dbl>
##  1 Blue Light, The (Das Blaue Licht) (1932)                           5
##  2 Fighting Elegy (Kenka erejii) (1966)                               5
##  3 Hellhounds on My Trail (1999)                                      5
##  4 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                                5
##  5 Shadows of Forgotten Ancestors (1964)                              5
##  6 Sun Alley (Sonnenallee) (1999)                                     5
##  7 Constantine's Sword (2007)                                         4.75
##  8 Human Condition II, The (Ningen no joken II) (1959)                4.75
##  9 Human Condition III, The (Ningen no joken III) (1961)              4.75
## 10 Who's Singin' Over There? (a.k.a. Who Sings Over There) ~           4.75
```

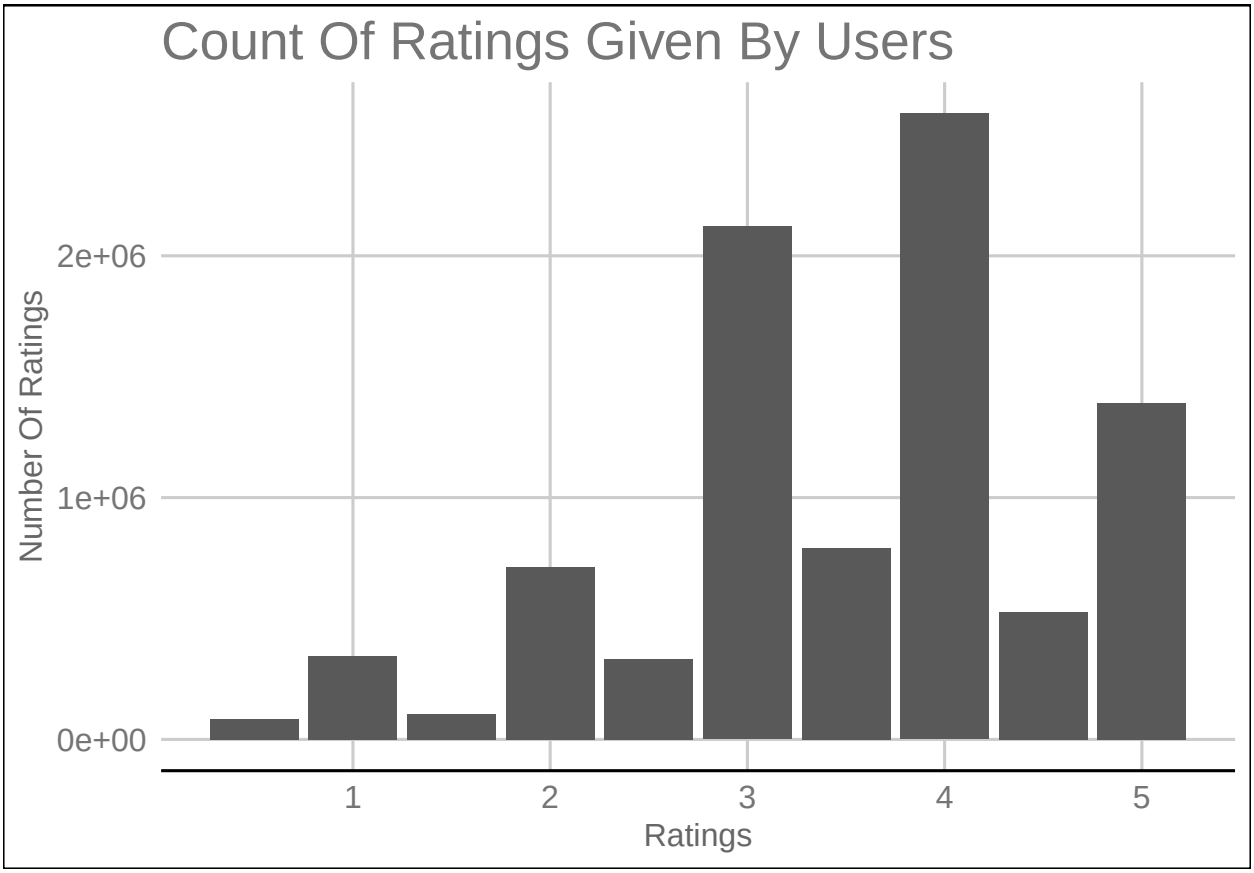
Now we visualise the overall distribution of the ratings given to the movies.

```
edx %>% count(movieId) %>% ggplot(aes(n)) +
  geom_histogram(bins = 30,color = "black") +
  scale_x_log10() +
  xlab("Number Of Ratings Given") +
  ylab("Count Of Movies") +
  theme_gdocs()
```



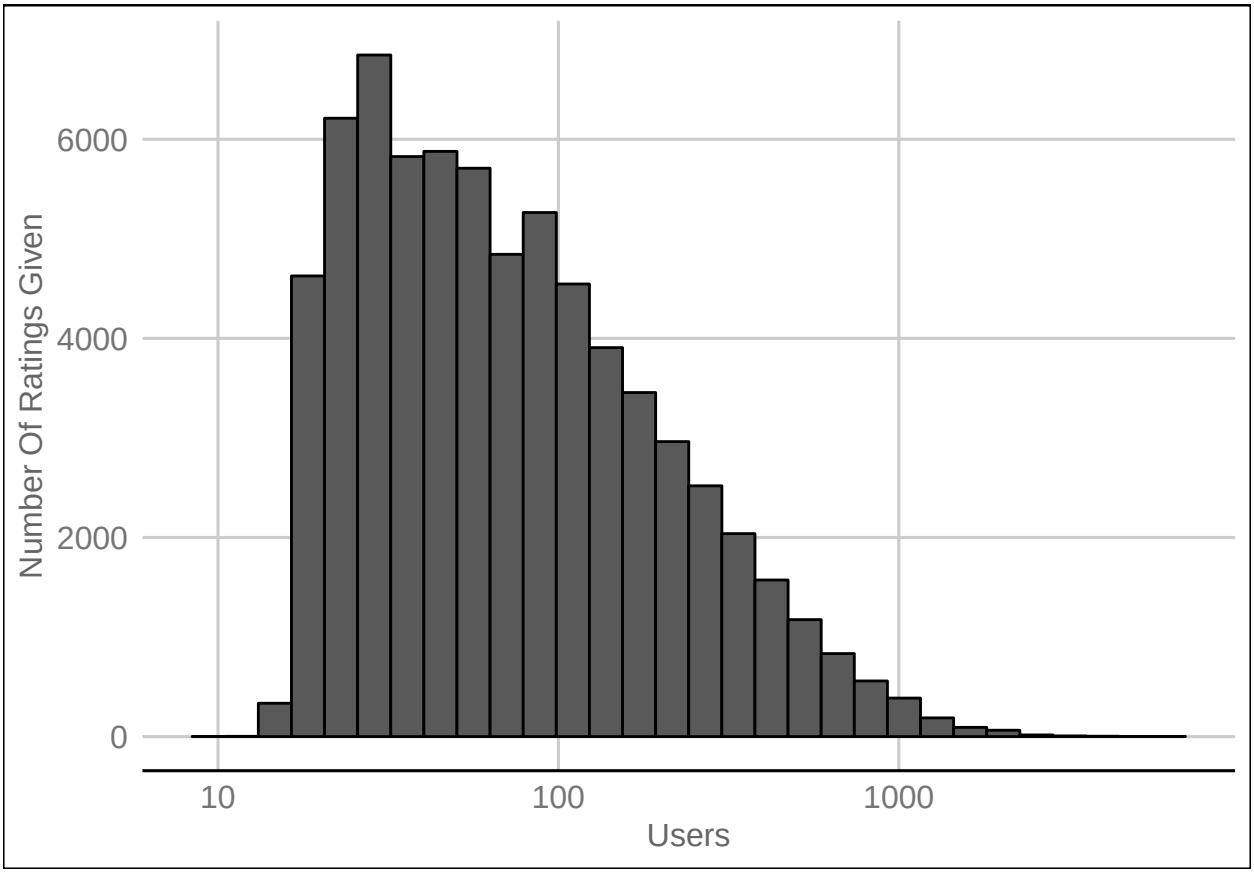

Now we visualise the overall distribution of the frequency of the ratings given by the users.

```
edx %>% group_by(rating) %>%
  summarise(total_ratings = n()) %>%
  ggplot(aes(rating,total_ratings)) +
  geom_bar(stat = "identity") +
  xlab("Ratings") +
  ylab("Number Of Ratings") +
  ggtitle("Count Of Ratings Given By Users") +
  theme_gdocs()
```




Now we visualise the overall distribution of the number of the ratings given by the users.

```
edx %>% count(userId) %>% ggplot(aes(n)) +
  geom_histogram(bins = 30,color = "black") +
  scale_x_log10() +
  xlab("Users") +
  ylab("Number Of Ratings Given") + theme_gdocs()
```




### Recommendation Models

First we define a function, to compute the *RMSE (Root Mean Squared Error)*.

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
```

### Baseline Model

In this model we simply use the mean rating given to all the movies as a prediction of the rating that can be given by the user.

```
mu <- mean(edx$rating)
```

The RMSE for *Baseline Model*.

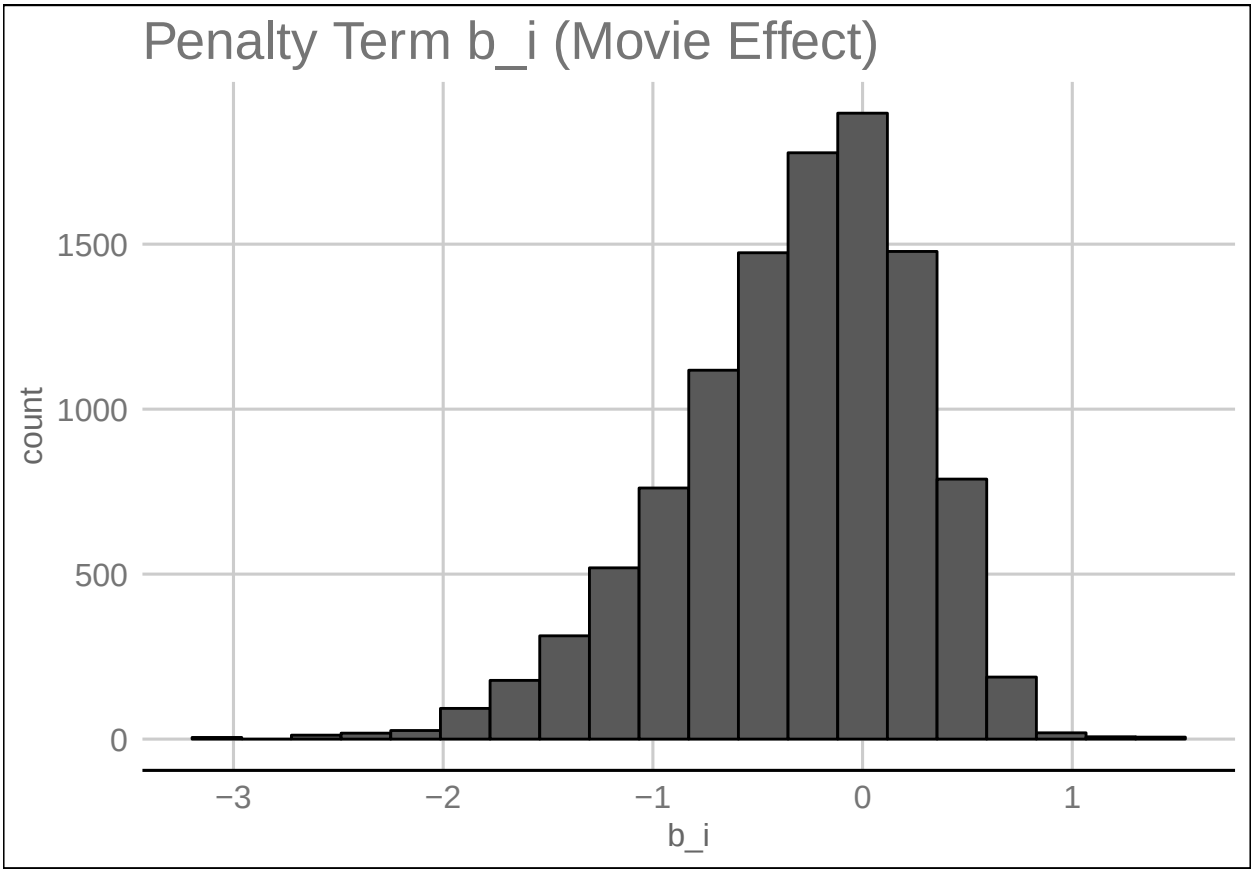
```
rmse <- RMSE(validation$rating,mu)
rmse_results <- data.frame(Method = "Using Mean Only",RMSE = rmse)
rmse_results %>% knitr::kable()
```

| Method | RMSE |
|---|---|
| Using Mean Only | 1.061202 |

### Movie Effect Model

As it was observed in the Exploratory Data Analysis, every movie is rated by different number of users, and thus, this factor should also be considered while predicting the rating, along with the average rating.

```
movie_avg <- edx %>% group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

movie_avg %>%
  ggplot(aes(b_i)) +
  geom_histogram(bins = 20,color = "black") +
  ggtitle("Penalty Term b_i (Movie Effect)") +
  theme_gdocs()
```




The RMSE for *Movie Effect Model*.

```
movie_avg_pred <- validation %>%
  left_join(movie_avg,by = "movieId") %>%
  mutate(pred = (mu + b_i)) %>%
  pull(pred)

rmse <- RMSE(validation$rating,movie_avg_pred)
rmse_df <- data.frame(Method = "Using Movie Effect",RMSE = rmse)
rmse_df %>% knitr::kable()
```

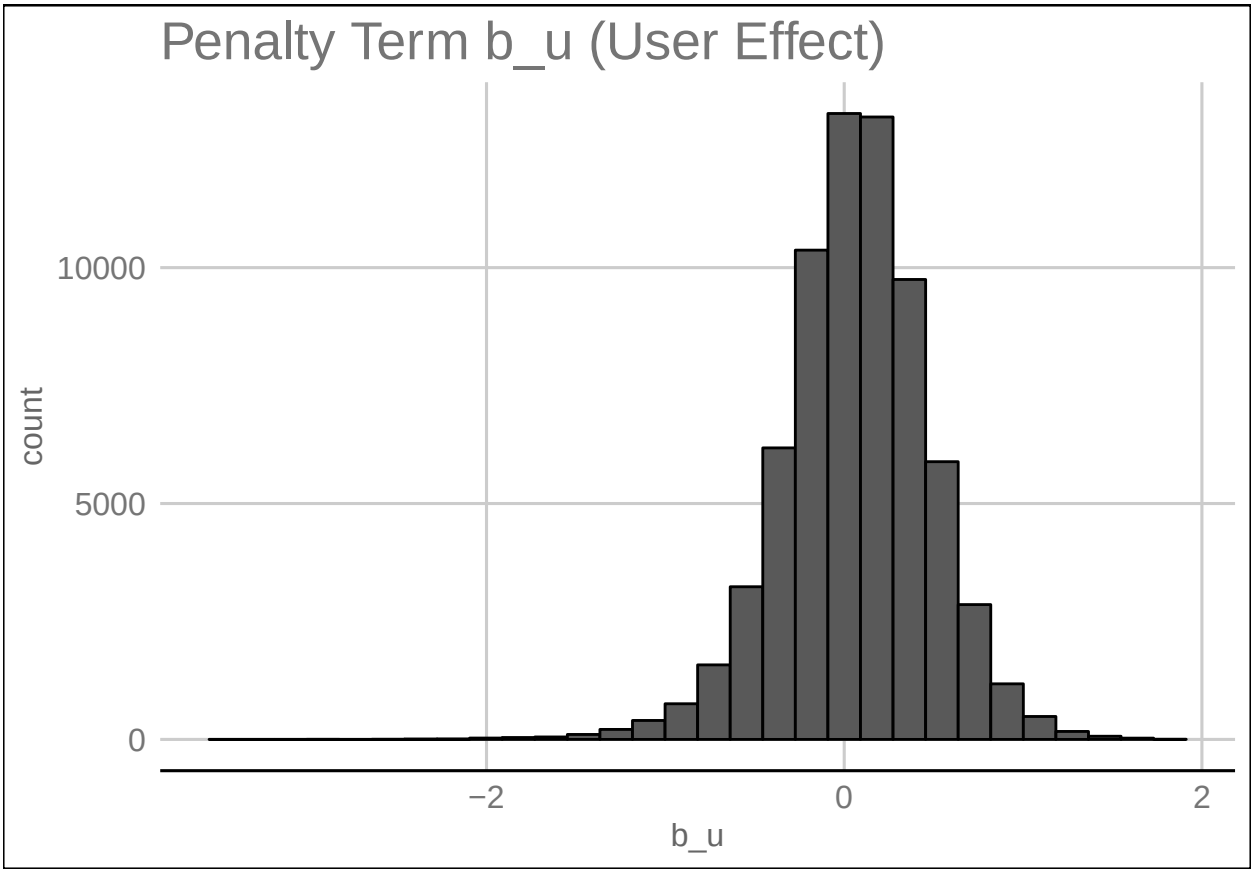
| Method | RMSE |
|---|---|
| Using Movie Effect | 0.9439087 |

```
rmse_results <- bind_rows(rmse_results,rmse_df)
```

### User Effect Model

As seen in *Exploratory Data Analysis*, every user has rated different number of movies, and this can be used as a factor for predicting the rating for every movie.

```
user_avg <- edx %>% left_join(movie_avg,by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

user_avg %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30,color = "black") +
  ggtitle("Penalty Term b_u (User Effect)") +
  theme_gdocs()
```




The RMSE for *User Effect Model*.

```
user_avg_pred <- validation %>%
  left_join(movie_avg,by = "movieId") %>%
  left_join(user_avg,by = "userId") %>%
  mutate(pred = b_u + b_i + mu) %>%
  pull(pred)

rmse <- RMSE(validation$rating,user_avg_pred)
rmse_df <- data.frame(Method = "Using Movie & User Effect",RMSE = rmse)
rmse_df %>% knitr::kable()
```

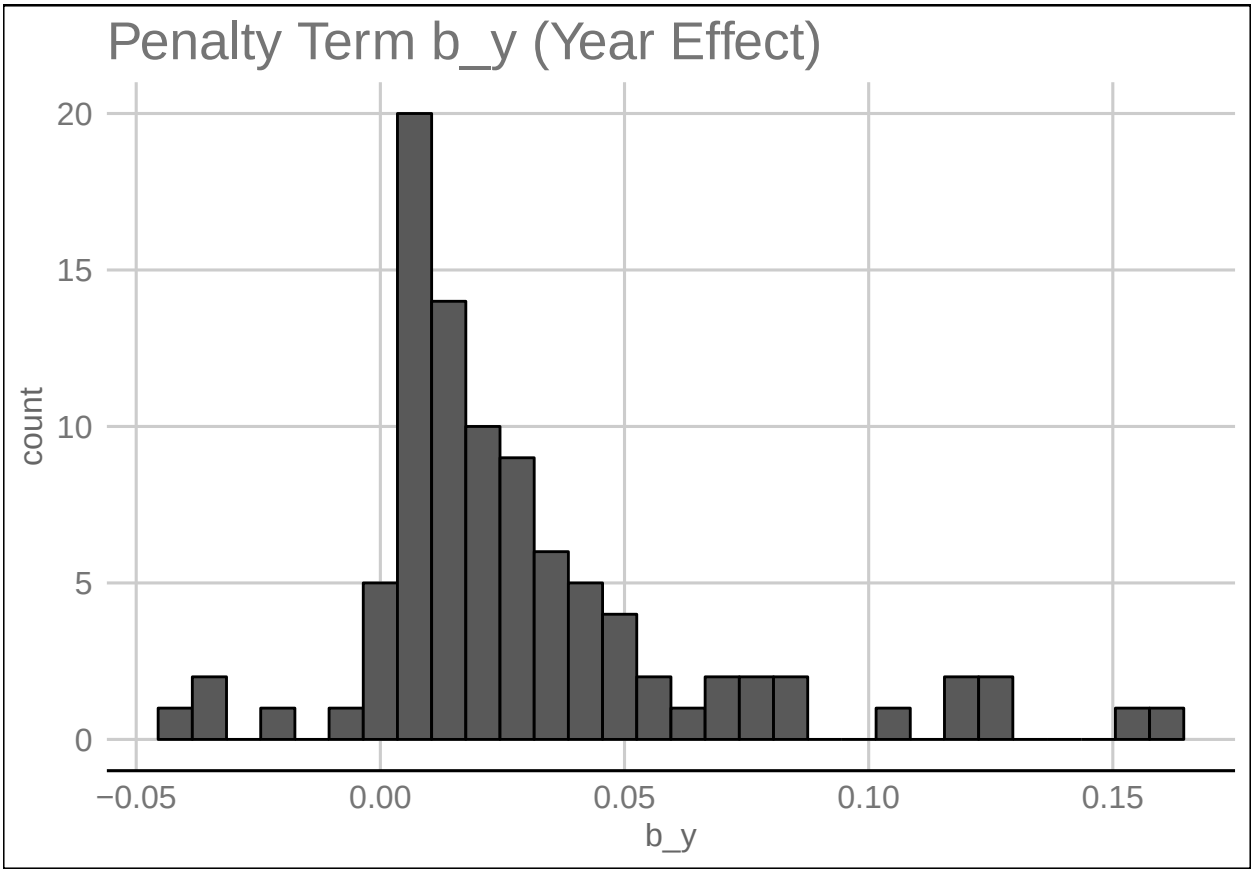
| Method | RMSE |
|---|---|
| Using Movie & User Effect | 0.8653488 |

```
rmse_results <- bind_rows(rmse_results,rmse_df)
```

### Year Effect Model

As observed during Exploratory Data Analysis, the rating trend various over different years, and thus release year, can also be used as a factor to predict the movie rating.

```
year_avg <- edx %>% left_join(movie_avg,by = "movieId") %>%
  left_join(user_avg,by = "userId") %>%
  group_by(year) %>%
  summarise(b_y = mean(rating - mu - b_i - b_u))

year_avg %>%
  ggplot(aes(b_y)) +
  geom_histogram(bins=30,color="black") +
  ggtitle("Penalty Term b_y (Year Effect)") +
  theme_gdocs()
```




The RMSE for *Year Effect Model*.

```
year_avg_pred <- validation %>% left_join(movie_avg,by = "movieId") %>%
  left_join(user_avg,by = "userId") %>%
  left_join(year_avg,by = "year") %>%
  mutate(pred = b_u + b_i + b_y + mu) %>%
  pull(pred)

rmse <- RMSE(validation$rating,year_avg_pred)

title <- "Using Movie,User & Year Effect"
rmse_df <- data.frame(Method = title,RMSE = rmse)
rmse_df %>% knitr::kable()
```

| Method | RMSE |
|---|---|
| Using Movie,User & Year Effect | 0.8650043 |

```
rmse_results <- bind_rows(rmse_results,rmse_df)
```

### Recommendation Model Using Regularisation

We observed the following about the features in the *Exploratory Data Analysis* :
1. **Movie Effect** : It can be seen that the *Most Rated* movies do not feature in *Highest Rated*,

because due to large number of ratings, they tend to have lower average rating. Hence, we need to use regularisation on this feature, to have more accurate predictions.
2. **User Effect** : In the dataset, some users have rated very few movies, whereas some have rated almost all movies, thus, the shorter number of ratings, can lead to larger estimates of rating, and hence this factor needs to be regularised, to have more accurate predictions.
3. **Year Effect** : By the graphs, it was seen that during years 1995-2005, many movies were released, whereas they were comparitively less in other years, and it was also observed, that the average rating, varied across year, thus regularisation of years, is used to get a better accurate predicted rating.
4. **Genre Effect** : The number of movies and the number of rating given to each genre differ, and the regularisation of genres, leads to a more accurate predicted rating.

### Regularised Movie & User Effect Model

First we define a vector of various *Lambda* values, from which we will select, the lambda value which has the *lowest RMSE value*.Here, *Lambda Is A Tuning Parameter*.

```
lambdas <- seq(0, 10, 0.25)
```

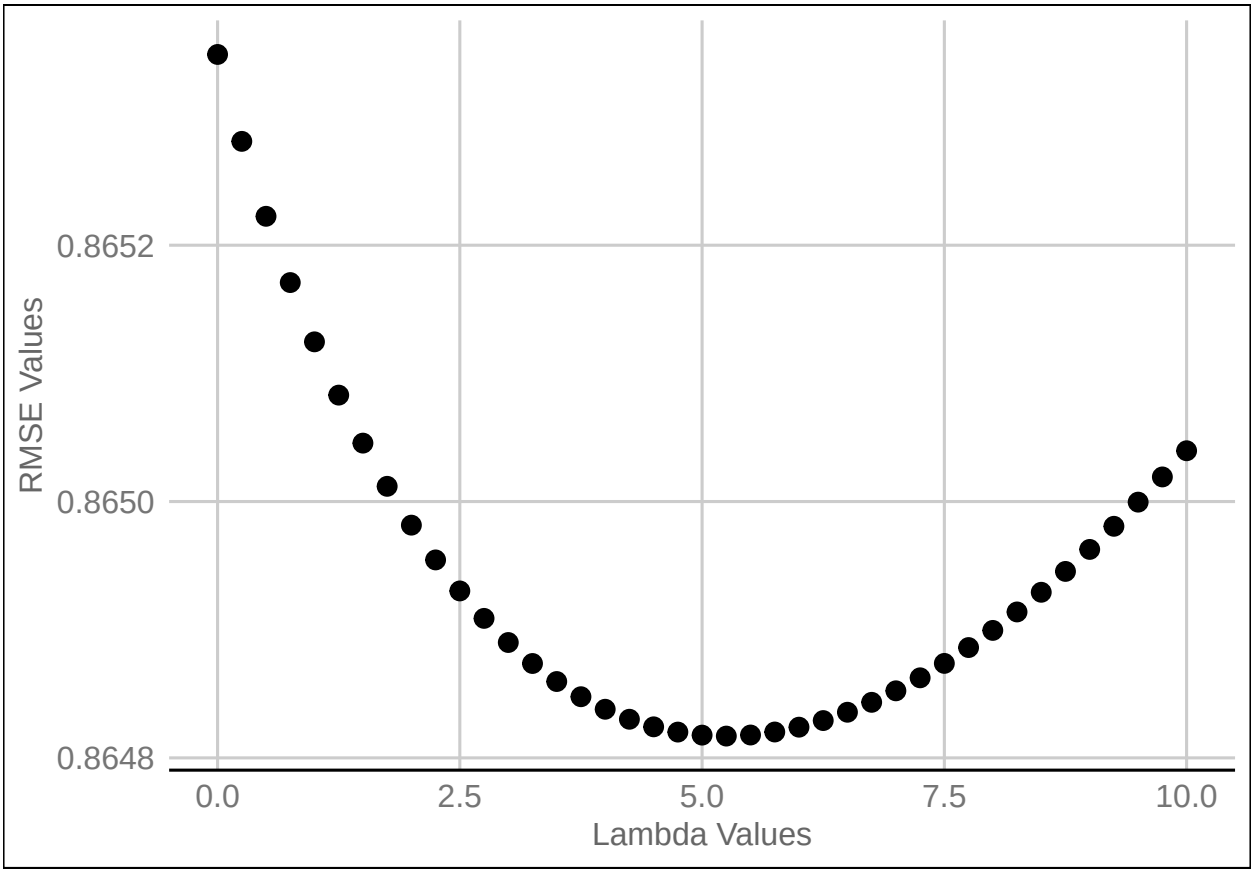
Now, we define a function, which computes the *RMSE* for different lambda values, and returns the list of the *RMSE Results*.

```
rmse_list <- sapply(lambdas, function(l) {
  mu <- mean(edx$rating)

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

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

  reg_user_movie_pred <- validation %>%
    left_join(movie_avg, by = "movieId") %>%
    left_join(user_avg, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>% pull(pred)

  return(RMSE(validation$rating,reg_user_movie_pred))
})
```

Now, we plot the *Lambda Values V/S The RMSE Values*.

```
data.frame(lambda = lambdas,rmse = rmse_list) %>%
  ggplot(aes(lambda,rmse)) +
  geom_point(size = 3) +
  theme_gdocs() +
  xlab("Lambda Values") +
  ylab("RMSE Values") +
```

```
  theme_gdocs()
```




Now, we select the lambda value, which the lowest value for the RMSE.

```
lambda <- lambdas[which.min(rmse_list)]
lambda
```

```
## [1] 5.25
```

Now, we train, the model, based on the select lambda value.

```
mu <- mean(edx$rating)

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

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

reg_user_movie_pred <- validation %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
```

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

The RMSE for *Regularised Movie & User Effect Model.*

```
rmse <- RMSE(validation$rating,reg_user_movie_pred)

title <- "Regularised Method Using Movie & User Effect"
rmse_df <- data.frame(Method = title,RMSE = rmse)
rmse_df %>% knitr::kable()
```

| Method | RMSE |
|---|---|
| Regularised Method Using Movie & User Effect | 0.864817 |

```
rmse_results <- bind_rows(rmse_results,rmse_df)
```

### Regularised Movie,User,Year & Genre Effect

First we define a vector of various *Lambda* values, from which we will select, the lambda value which has the *lowest RMSE value.*Here, *Lambda Is A Tuning Parameter.*

```
lambdas <- seq(0, 15, 1)
```

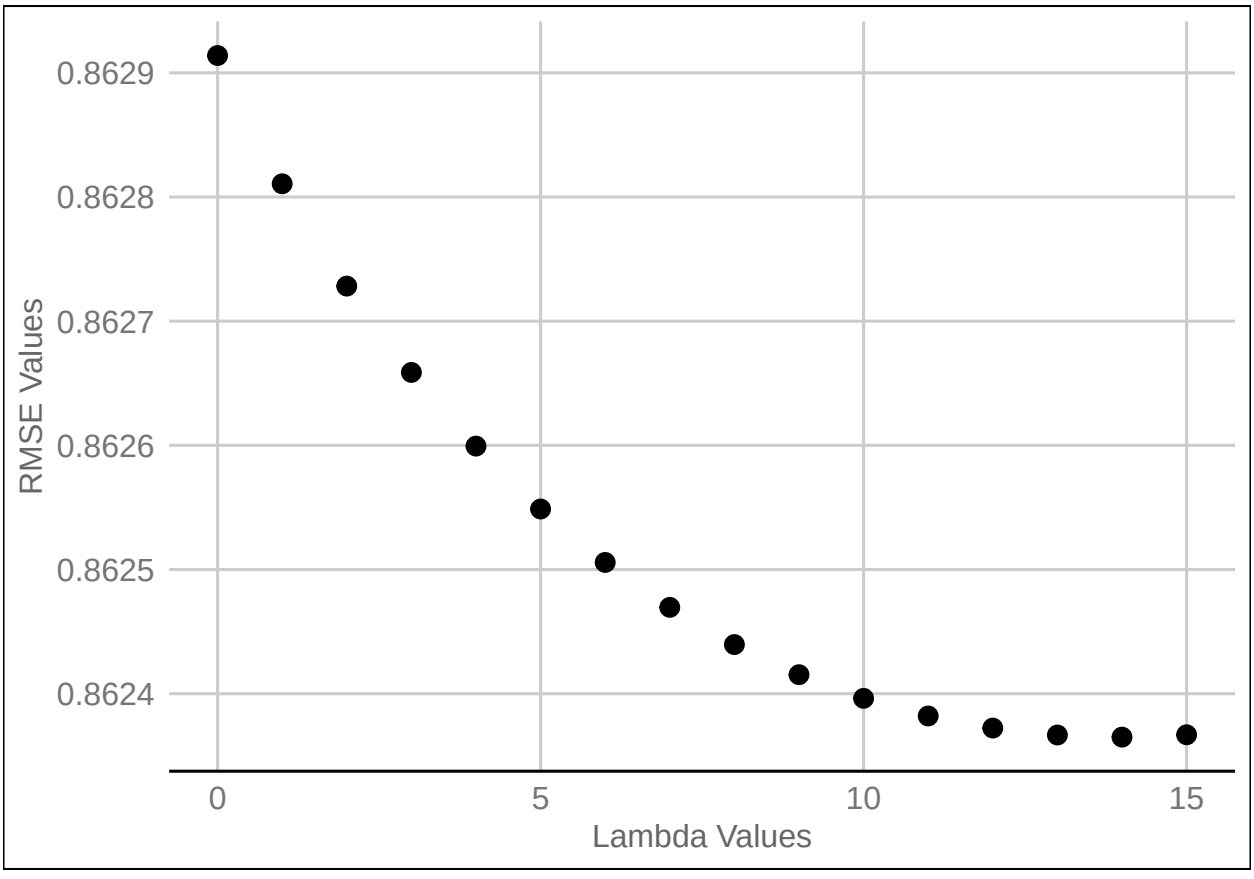
Now, we define a function, which computes the *RMSE* for different lambda values, and returns the list of the *RMSE Results.*

```
rmse_list <- sapply(lambdas, function(l) {
  mu <- mean(edx$rating)

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

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

  year_avg <- split_edx %>%
    left_join(movie_avg, by='movieId') %>%
    left_join(user_avg, by='userId') %>%
    group_by(year) %>%
    summarize(b_y=sum(rating-mu-b_i-b_u)/(n()+l), n_y=n())

  genre_avg <- split_edx %>%
    left_join(movie_avg, by='movieId') %>%
    left_join(user_avg, by='userId') %>%
    left_join(year_avg, by = 'year') %>%
    group_by(genres) %>%
```

```
    summarize(b_g=sum(rating-mu-b_i-b_u-b_y)/(n()+l), n_g=n())

  reg_pred <- split_validation %>%
    left_join(movie_avg, by='movieId') %>%
    left_join(user_avg, by='userId') %>%
    left_join(year_avg, by = 'year') %>%
    left_join(genre_avg, by = 'genres') %>%
    mutate(pred = mu + b_i + b_u + b_y + b_g) %>% pull(pred)

  return(RMSE(split_validation$rating,reg_pred))
})
```

Now, we plot the *Lambda Values V/S The RMSE Values*.

```
data.frame(lambda = lambdas,rmse = rmse_list) %>%
  ggplot(aes(lambda,rmse)) +
  geom_point(size = 3) +
  theme_gdocs() +
  xlab("Lambda Values") +
  ylab("RMSE Values") +
  theme_gdocs()
```




Now, we select the lambda value, which the lowest value for the RMSE.

```
lambda <- lambdas[which.min(rmse_list)]
lambda
```

```
## [1] 14
```

Now, we train, the model, based on the select lambda value.

```
mu <- mean(edx$rating)

movie_avg <- split_edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

user_avg <- split_edx %>%
  left_join(movie_avg, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating-b_i-mu)/(n()+lambda))

year_avg <- split_edx %>%
  left_join(movie_avg, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  group_by(year) %>%
  summarize(b_y=sum(rating-mu-b_i-b_u)/(n()+lambda),n_y=n())

genre_avg <- split_edx %>%
  left_join(movie_avg, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  left_join(year_avg, by = 'year') %>%
  group_by(genres) %>%
  summarize(b_g=sum(rating-mu-b_i-b_u-b_y)/(n()+lambda),n_g=n())

reg_pred <- split_validation %>%
  left_join(movie_avg, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  left_join(year_avg, by = 'year') %>%
  left_join(genre_avg, by = 'genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>% pull(pred)
```

The RMSE for *Regularised Movie,User,Year & Genre Effect Model.*

```
rmse <- RMSE(split_validation$rating,reg_pred)

title <- "Regularised Method Using Movie,User,Year & Genre Effect"
rmse_df <- data.frame(Method = title,RMSE = rmse)
rmse_df %>% knitr::kable()
```

| Method | RMSE |
|---|---|
| Regularised Method Using Movie,User,Year & Genre Effect | 0.862365 |

```
rmse_results <- bind_rows(rmse_results,rmse_df)
```

### RMSE Overview

Here, we take a look at the *RMSE Results* of all the models, which we have trained on **Movielens 10M Dataset**.

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

| Method | RMSE |
|---|---|
| Using Mean Only | 1.0612018 |
| Using Movie Effect | 0.9439087 |
| Using Movie & User Effect | 0.8653488 |
| Using Movie,User & Year Effect | 0.8650043 |
| Regularised Method Using Movie & User Effect | 0.8648170 |
| Regularised Method Using Movie,User,Year & Genre Effect | 0.8623650 |

### Conclusion

On the basis of the *RMSE Values*, it can be concluded that, the *Regularised Model Using the Movie,User,Year & Genre Effect* is the best performing model on the **Movielens 10M Dataset**. The *Mean Only Model* is the worst performing model, with an RMSE over 1, which is not good, as whenever we are predicting, we can overestimate or underestimate a rating by 1 star.
It can be also be observed that *Regularised Models*, perform better in comparison to *Non-Regularised Models*.
This is mainly due to the distribution of the data points, because in some cases, due to fewer data points, without regularisation, a bigger estimate is obtained. But *Regularisation*, helps in penalizing the bigger estimates, and helps lower these estimates near zero, thus providing a better prediction and lower RMSE values.

##Github Link
https://github.com/guptaharshnavin/Movielens_Recommendation_System