MovieLens HarvardX Capstone

Ana Hristova

05/06/2020

1. Introduction

Recommendation systems use ratings given by users for specific products/ services to make recomendations based on predictions. Ratings given by a user or users exhibiting similar characteristics are used to predict a rating for an item, which, if high, is used to recommend the item to the user. Ratings typically range from 1 to 5, where 1 means the item is not desirable and 5 indicates the customer would love it.

Online retail companies as well as video/ music streaming services amongst others use such systems to tailor their content to each user.

This project uses the MovieLens 10M dataset generated by GroupLens reasearch lab and released in January 2009 to predict movie ratings.

2. Data

The MovieLens dataset contains 10M observations of six variables and for the purpose of creating a predictive model and validating its predidictions, it is split into two sets - edx, containing 90% of the observations and validation, containing the remaining 10% and only used to calculate the RMSE of the output of the model vs the actual ratings.

The dataset is provided clean, with no missing values.

3. Methodology

As the aim of this project is to create a predictive model that predicts ratings for a set of movies and users, simulating a real modelling environment, only the edx data set was used for the exploratory analysis of the data.

The exploratory analysis of the data includes deep dive into each variable, to understand any dependancies and data types included.

Following the exploratory analysis, the edx dataset was split into train and test set to allow for examining different models and variables and to arrive to a model that would deliver the desired RMSE of below 0.8649.

The predictive models explored are: * Basic model using only the average rating * Model using just the movie effect * Model using a combination of movie and user effect * Model using a combination of movie, user and genre effect * Regularized model using a combination of movie, user and genre effect * Regularized model using a combination of movie, user, genre and time of rating effect

4. Exploratory data analysis

4.1. Exploring the data set

The edx dataset contains 6 variables and 9000055 observations. The variables included are: userId, movieId, rating, timestamp, title, genres

The set includes data for 69878 users who have rated 10677 movies that belong to different combinations of 20 possible genres. The first rating included in the dataset is from 1995-01-09 11:46:49 and the last is from 2009-01-05 05:02:16.

4.2. Exploring the users included in the dataset

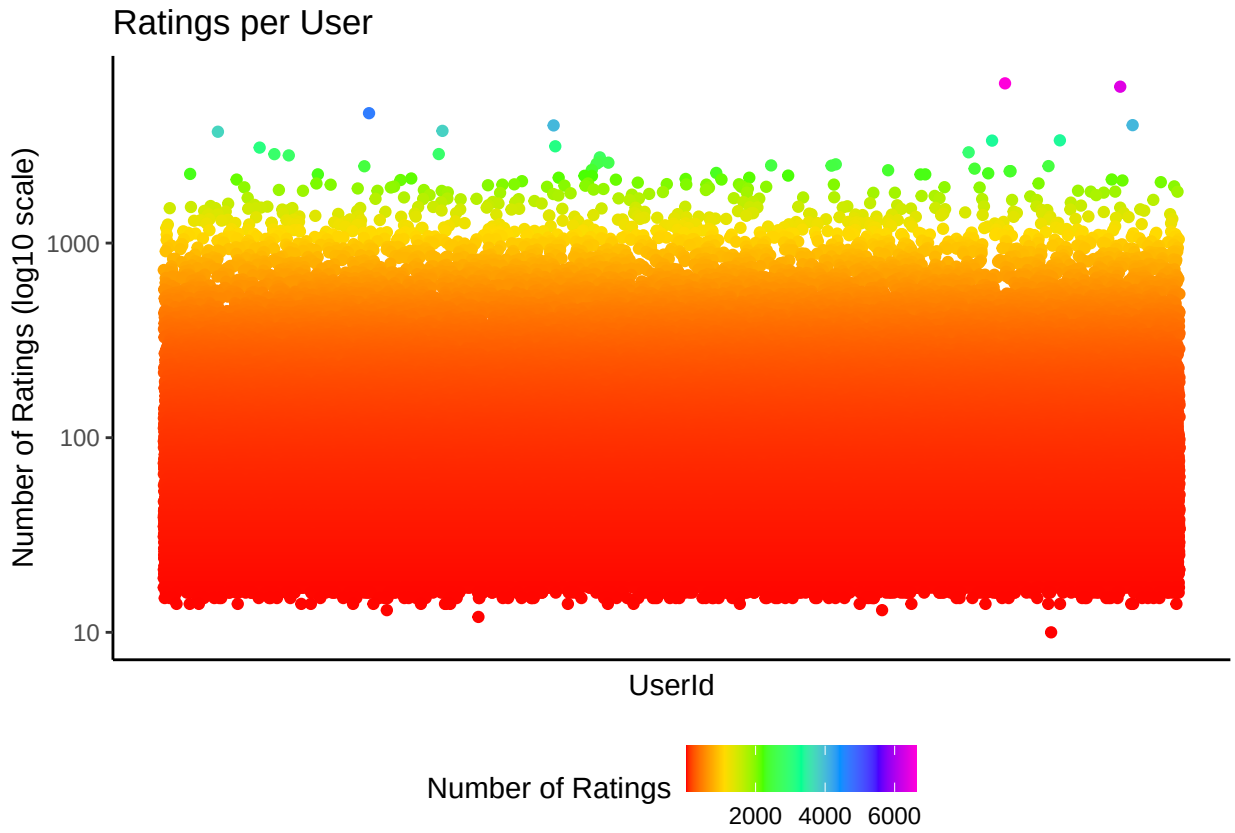
There are 69878 users included in the edx dataset.

User Ratings

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.0    32.0    62.0   128.8   141.0  6616.0
```

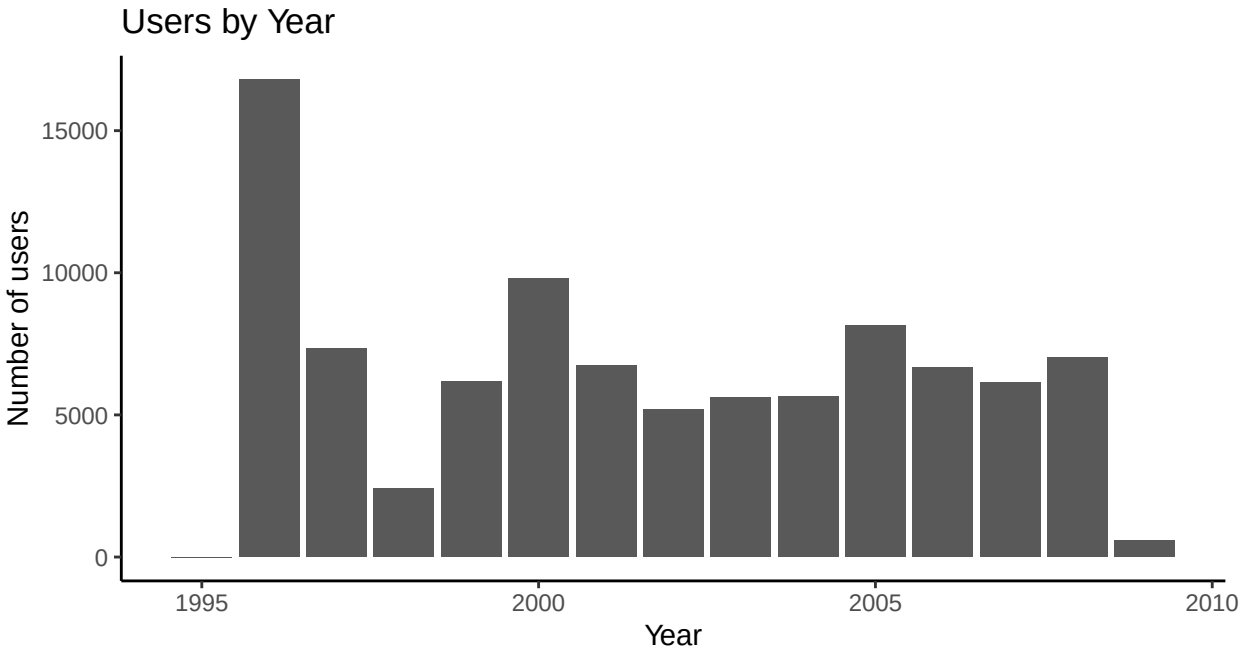
The average number of ratings provided by the users is 129. One user has been particularly active, providing 6616 ratings in total, while the minimum number of ratings submitted by a user is 10, with median number of ratings 62, suggesting that user ratings are heavily influenced by a small number of users who have provided exceptionally high number of ratings.

This is visualised in the below plot showing user activity.



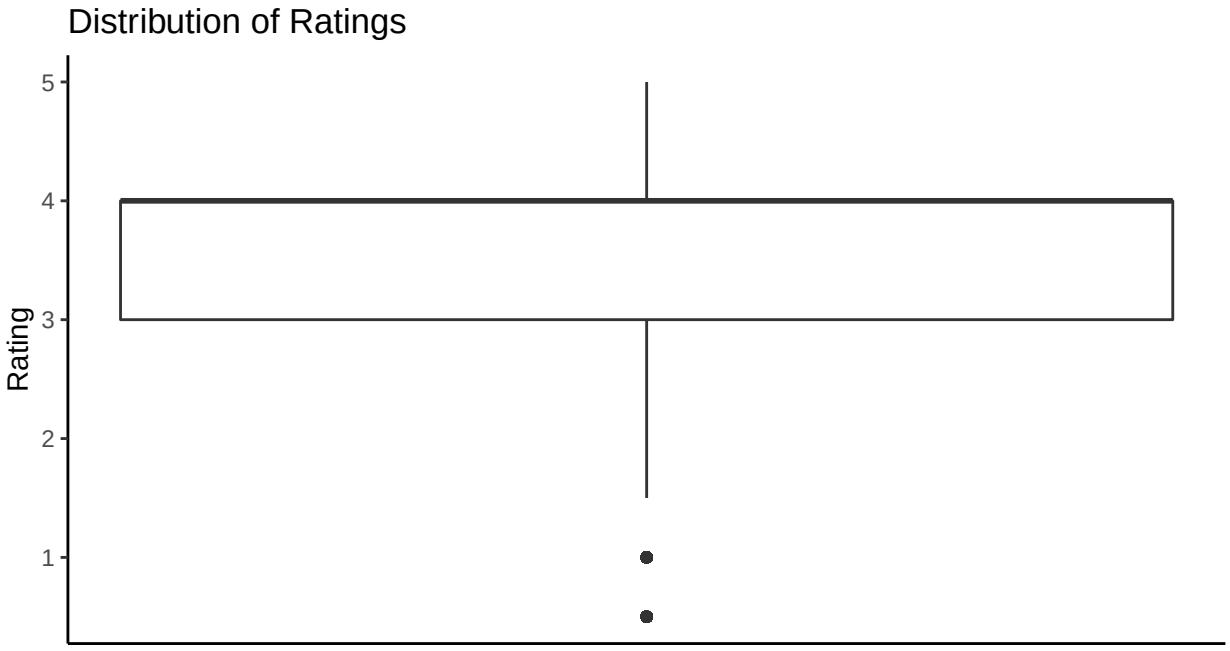
Next, we will look at the user activity for each year included in the data (1995 - 2009).

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1    5404    6164    6288    7176   16796
```



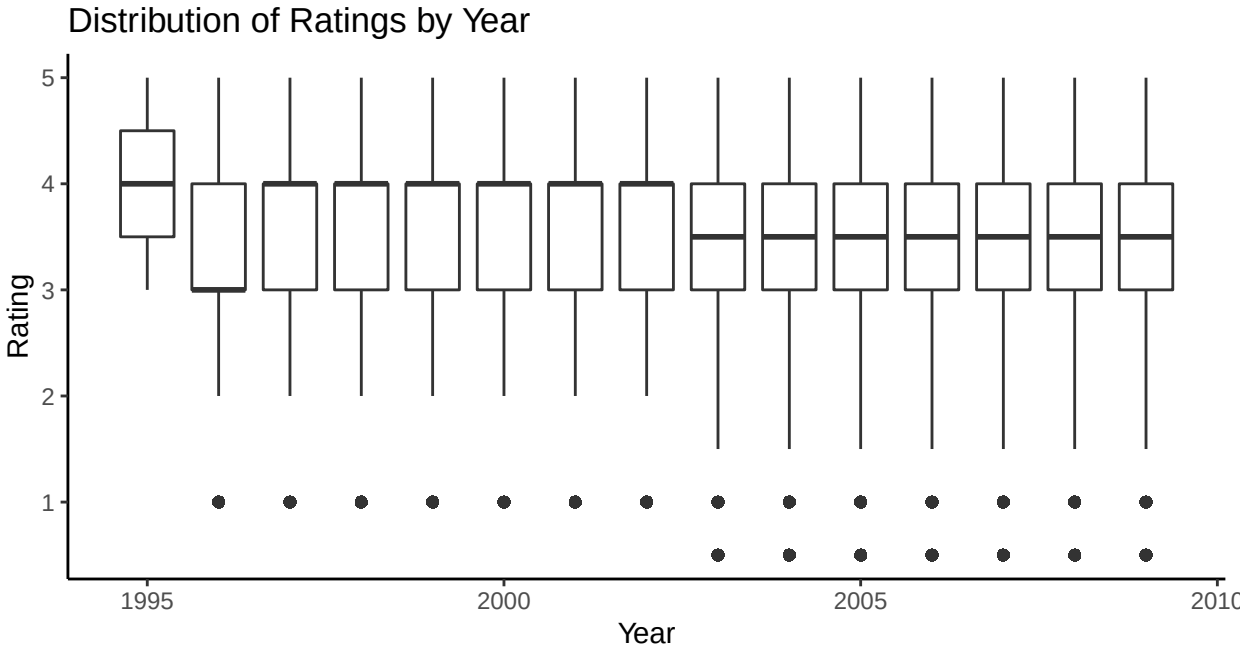
From the Users by Year plot above we see that 1995 and 2009 are the years with the lowest number of active users. This is due to the fact that the data for these years is only partial. The year with the highest number of active users is 1996 where there were 16796 active users. In the following section we will examine how that affects the ratings provided.

4.3. Exploring the ratings

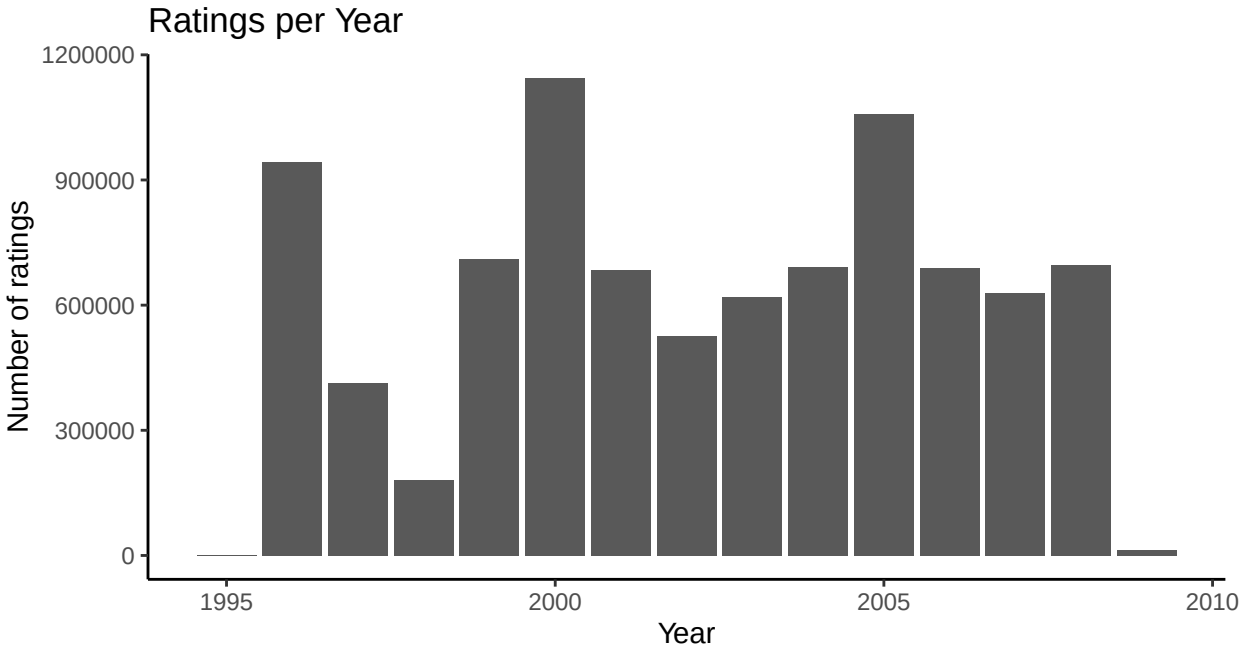


The Distribution of Ratings boxplot shows that generally people like the movies they have rated, with average rating of nearly 4. Most ratings provided are above 3 stars with only a handful of ratings being below or equal to 1 star.

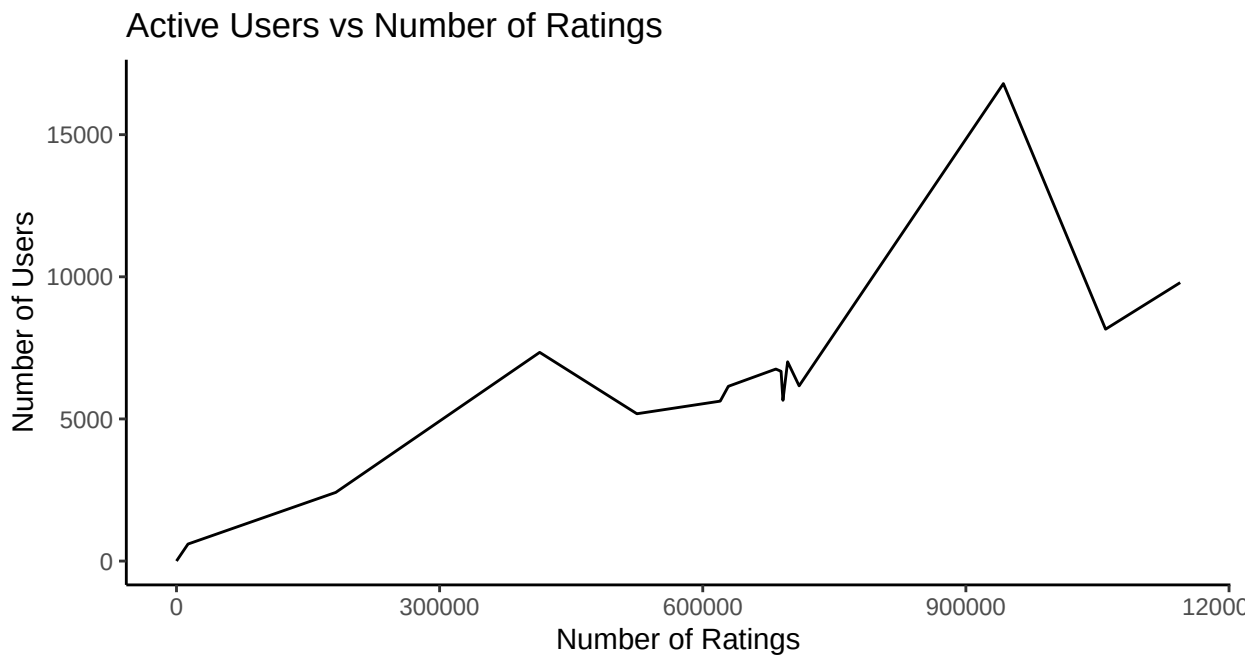
Looking at how the distribution of ratings varies per year, the .5 ratings appear to be introduced in 2003. Further investigation confirms they were introduced on 18 Feb 2003, and before that only whole star ratings were permitted.



The number of ratings per year appears to vary significantly. 1995 and 2009 are the years with the fewest ratings provided, which is explained by the fact that we only have partial records for this years and in line with the lower number of actve users, also due to the same reason.

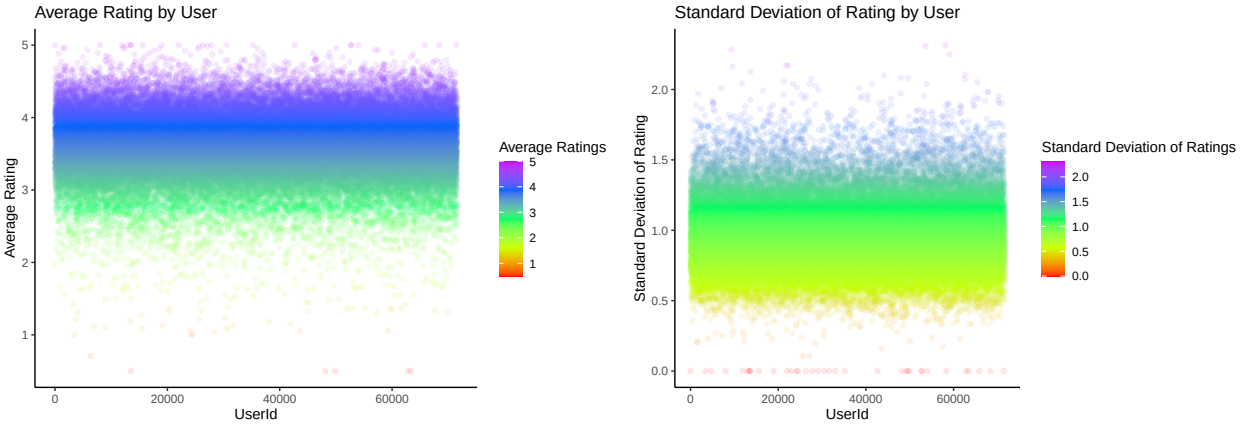


Lets explore the relationship between the number of ratings per year and the number of active users per year.



Unsurprisingly, we notice that the more active users there are, the more ratings are provided, with a strong positive correlation of 0.81.

Exploring the ratings by user, we notice that the average rating is 3 to 4 stars, with a small number of users that loved every movie they saw and even fewer who hated every movie they saw. The standard deviation of ratings by user plot shows that the majority of users are consistent with theyr ratings with about 1 star standard deviation, with only a few people who gave a wide range of ratings.



4.4.Exploring the movies

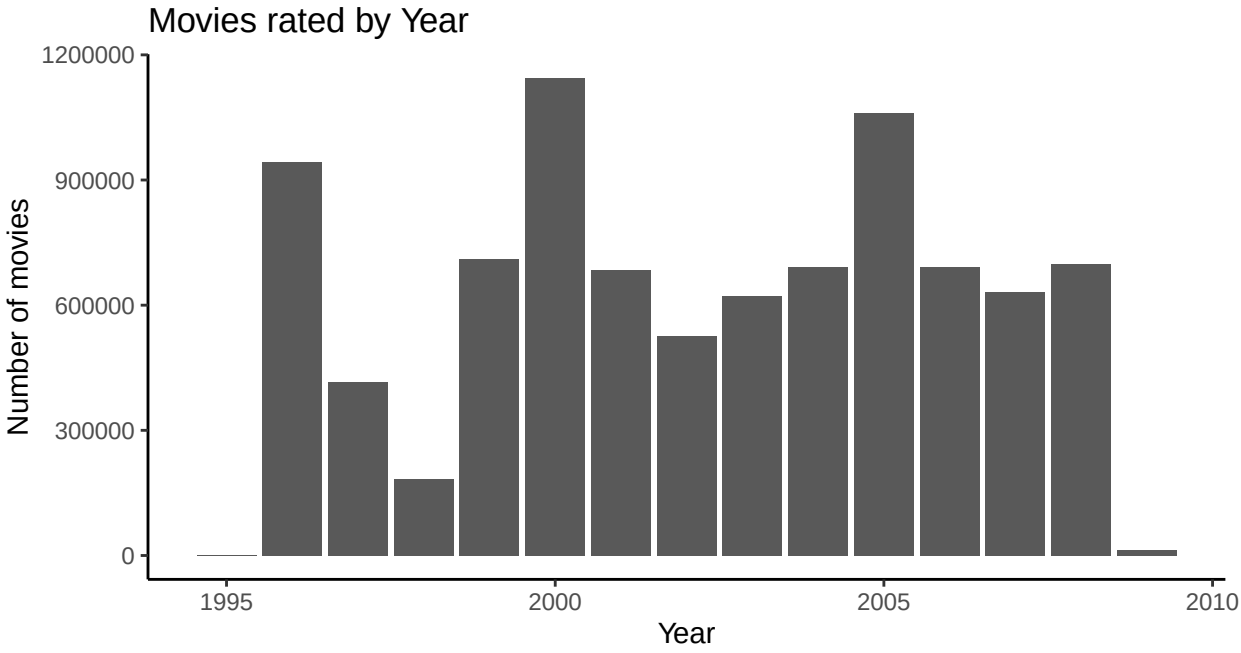
There are 10677 movies included in the edx dataset.

The 5 most rated movies are:

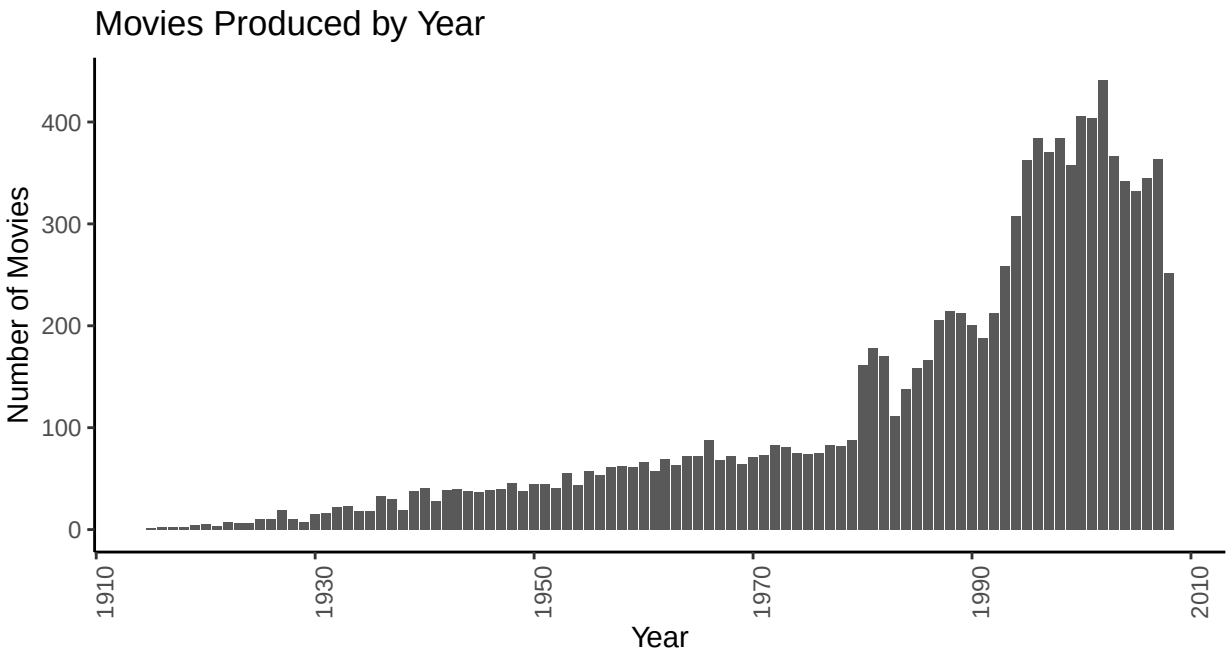
title	n_ratings	avg	sd
Pulp Fiction (1994)	31362	4.154789	1.0041181
Forrest Gump (1994)	31079	4.012822	0.9715636
Silence of the Lambs, The (1991)	30382	4.204101	0.8394029
Jurassic Park (1993)	29360	3.663522	0.9381197

title	n_ratings	avg	sd
Shawshank Redemption, The (1994)	28015	4.455131	0.7170227
Braveheart (1995)	26212	4.081852	0.9526711

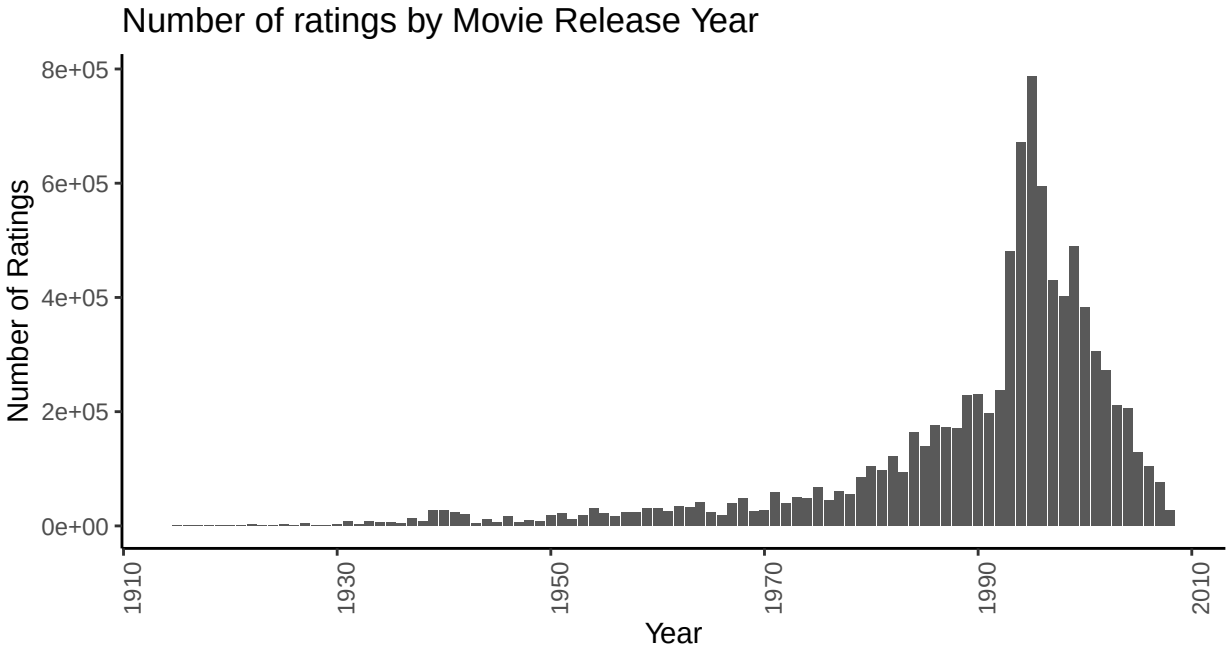
In the below chart, we see that the number of movies rated by year follows the same pattern as the number of ratings per year. 1995 and 2009 have the lowest numbe of movies rated due to them being only partially inclueded in the dataset. 2000, 2005 and 1996 are the years with most movies rated.



The year in which a movie is produced is included in the movie title column after the movie title, which permits us to explore how many movies are produced by year of production in the graph below. There is an exponential growth in movies produced from 1910 until 2000 which then drops in the following years.



The number of ratings by movie release year follows a similar pattern with the exponential growth peaking for movies released in the 90's. Tis is likely due to the fact that the ratings included in the data set begin in the 90's and movies which were then new and popular have had the longest time to receive ratings.



The top 10 movies **by number of ratings received** are:

title	n_ratings	avg	sd
Pulp Fiction (1994)	31362	4.154789	1.0041181
Forrest Gump (1994)	31079	4.012822	0.9715636
Silence of the Lambs, The (1991)	30382	4.204101	0.8394029
Jurassic Park (1993)	29360	3.663522	0.9381197

title	n_ratings	avg	sd
Shawshank Redemption, The (1994)	28015	4.455131	0.7170227
Braveheart (1995)	26212	4.081852	0.9526711
Fugitive, The (1993)	25998	4.009155	0.7776601
Terminator 2: Judgment Day (1991)	25984	3.927859	0.9059733
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	25672	4.221311	0.9139770
Apollo 13 (1995)	24284	3.885789	0.8518878

The average ratings for these movies vary from 3.66 to 4.46 which is a large variance for this dataset. Let's look at the top 10 movies **by rating**:

title	n_ratings	avg	sd
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	2	5.00	0.0000000
Blue Light, The (Das Blaue Licht) (1932)	1	5.00	NaN
Fighting Elegy (Kenka erejii) (1966)	1	5.00	NaN
Hellhounds on My Trail (1999)	1	5.00	NaN
Shadows of Forgotten Ancestors (1964)	1	5.00	NaN
Sun Alley (Sonnenallee) (1999)	1	5.00	NaN
Human Condition II, The (Ningen no joken II) (1959)	4	4.75	0.2886751
Human Condition III, The (Ningen no joken III) (1961)	4	4.75	0.2886751
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4	4.75	0.5000000
Constantine's Sword (2007)	2	4.75	0.3535534

Those movies have a very small number of ratings (between 1 and 4), so to gain an understanding of the most popular movies (those with a large number of ratings and a high average rating), let's review the top 10 movies **by rating with 1k or more ratings**:

title	n_ratings	avg	sd
Shawshank Redemption, The (1994)	28015	4.455131	0.7170227
Godfather, The (1972)	17747	4.415366	0.8059105
Usual Suspects, The (1995)	21648	4.365854	0.7589700
Schindler's List (1993)	23193	4.363493	0.8056913
Casablanca (1942)	11232	4.320424	0.8223351
Rear Window (1954)	7935	4.318651	0.7270165
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	2922	4.315880	0.7869492
Third Man, The (1949)	2967	4.311426	0.7697156
Double Indemnity (1944)	2154	4.310817	0.7540440
Paths of Glory (1957)	1571	4.308721	0.7572455

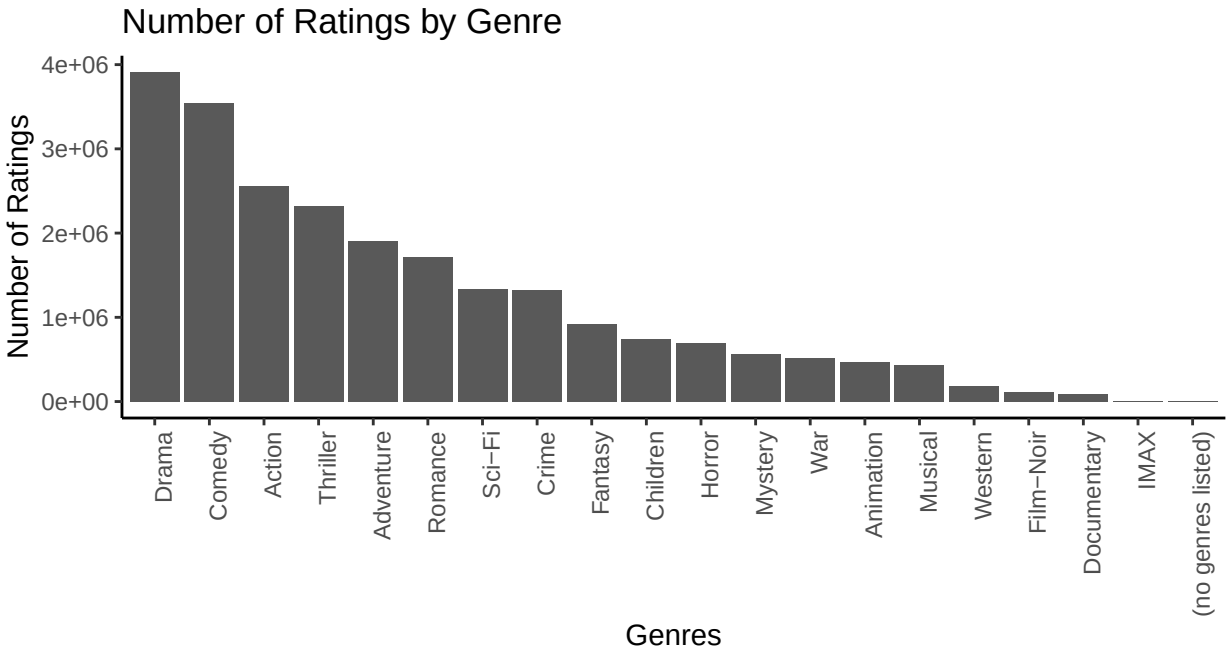
Now this is a list of well known classics which are always listed in "must watch" movie lists!

4.5. Exploring the genres

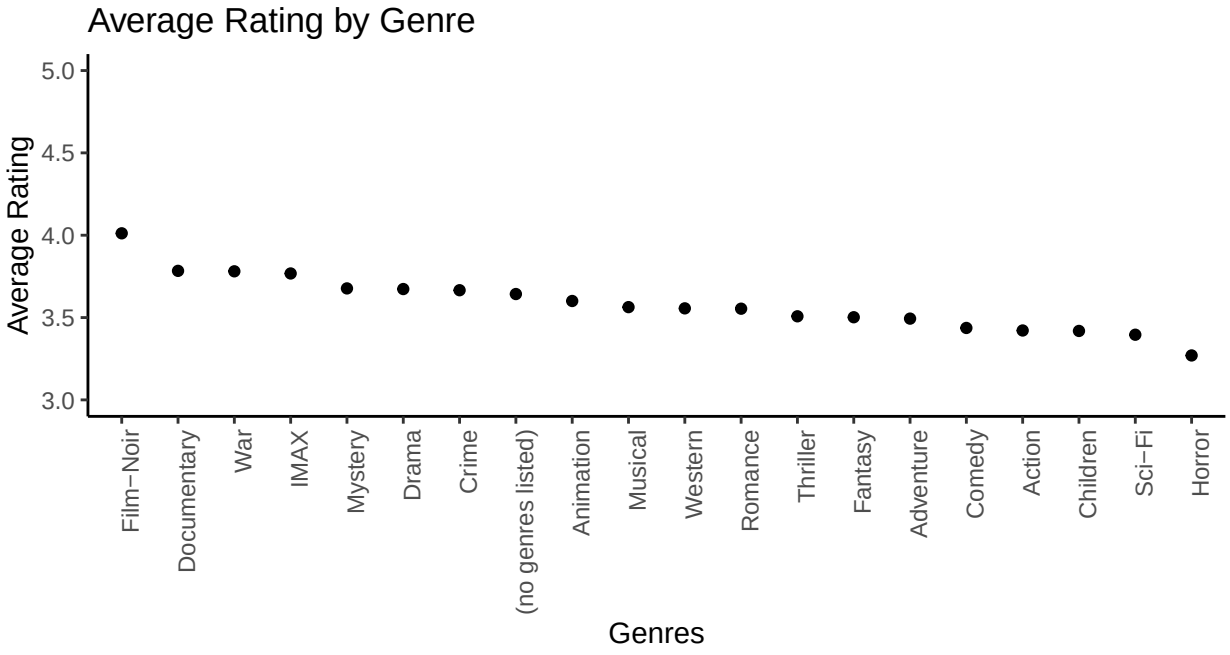
The last variable to explore are the movie genres.

The movies included in the edx dataset belong to combinations of 19 genres with 1 movie having no genre listed. There are 797 combinations of genres included.

Lets look at a breakdown of the genres.



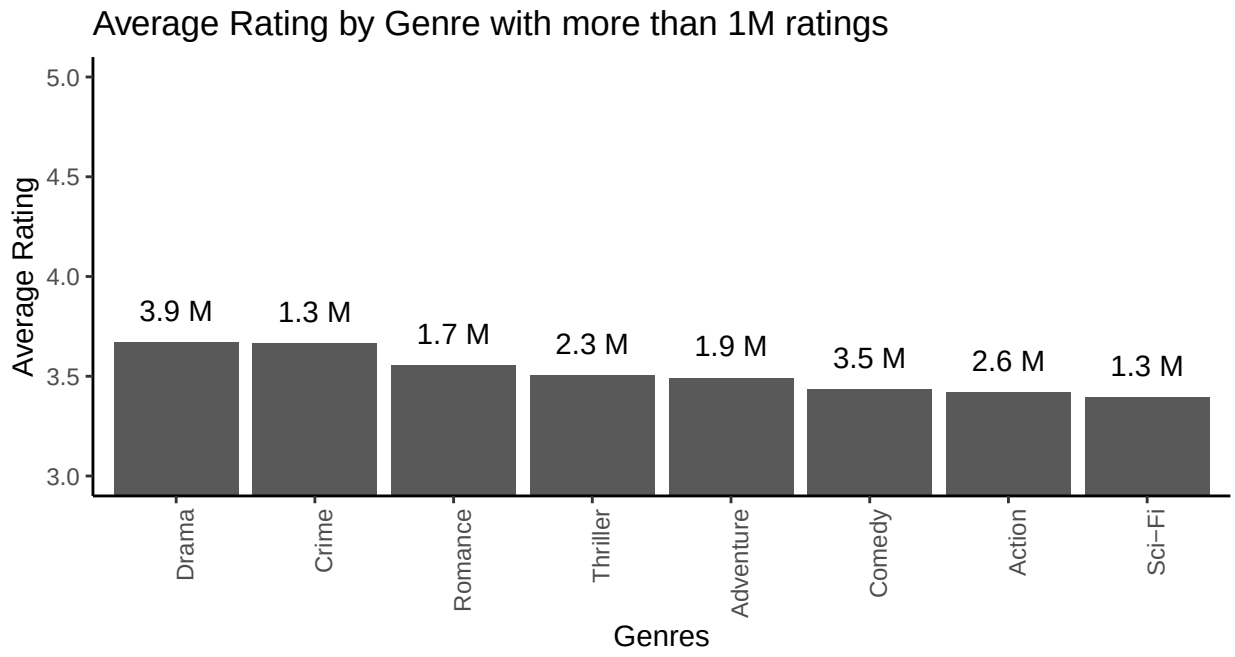
The genres that have received the highest number of ratings are Drama, Comedy and Action, which aligns with the genres of most blockbuster movies which are likely to be viewed by many people and receive many ratings.



When we look at the average rating for each genre, we notice that the highest rated genres Film-Noir, Documentary and War were on the tail of the graph that showed the genres by number of ratings. We can assume these genres are watched and reviewed mostly by connoiseurs who generally enjoy these particular genres.

To get an understanding of the most popular genres - those with high avereage rating and large volume of reviews, the below graph shows genres with over 1M ratings ranked by average rating. Drama appears on

first position again, which is also aligned with the most popular movies seen above.



5. Creating predictive model

In this section we will take a look at 6 predictive models to choose the best one amongst them, which will yield a RMSE of below 0.86490.

For this purpose, the edx dataset explored above is split into a train and test set, which will allow for the model to be tested and fitted before applying the best one to the validation set.

5.1. Basic model with just the average

For this model, we use the avergae of the ratings in the train set as our predicted value for the ratings in the test set.The RMSE of this method is over 1, which is much larger than the one we are trying to achieve.

```
mu_hat<- mean(train_set$rating)
naive_rmse<- RMSE(test_set$rating, mu_hat)


#creating a tibble for storing RMSE results
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)

knitr::kable(rmse_results)
```

method	RMSE
Just the average	1.059904

Let's start using the variables to try and get better predictions.

5.2. Model using just the movie effect

In this model we will explore what effect do the movies have in estimating the rating by calculating movie bias. From the table we can see that the RMSE has indeed improved slightly but is far from the desired value.

```
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
movie_rmse<-RMSE(predicted_ratings, test_set$rating)

rmse_results<- bind_rows(rmse_results, data_frame(method = "Movie effect model",
                                                  RMSE = movie_rmse))
knitr::kable(rmse_results)
```

method	RMSE
Just the average	1.0599043
Movie effect model	0.9437429

5.3. Model using a combination of movie and user effect

Building on the previous model, we will now include both movie and user bias as ratings differ by movie but also two different users can give the same movie a different rating due to their own preferences. This imporoved further our model.

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

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

movie_user_rmse<-RMSE(predicted_ratings, test_set$rating)
rmse_results<- bind_rows(rmse_results, data_frame(method = "Movie + user effect model",
                                                  RMSE = movie_user_rmse))
knitr::kable(rmse_results)
```

method	RMSE
Just the average	1.0599043
Movie effect model	0.9437429
Movie + user effect model	0.8659319

5.4. Model using a combination of movie, user and genre effect

As we are still away from achieveing the desired RMSE in our train and test sets, lets look into including the bias genre introduces into the ratings, as different users enjoy different genres.

```
genre_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
```

```
  left_join(user_avgs, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

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

movie_user_genre_rmse<-RMSE(predicted_ratings, test_set$rating)
rmse_results<- bind_rows(rmse_results, data_frame(method = "Movie + user + genre effect model",
                                                  RMSE = movie_user_genre_rmse))
knitr::kable(rmse_results)
```

method	RMSE
Just the average	1.0599043
Movie effect model	0.9437429
Movie + user effect model	0.8659319
Movie + user + genre effect model	0.8655941

5.5. Regularized model using a combination of movie, user and genre effect

The RMSE is getting closer to the desired value but there is still some space for tunning the model. During the exploratory data analysis we noticed that some movies have thousands or ratings while others only a handful, some users gave many ratings, some gave few and some genres were much more popular than others. In order to fit our model better we need to include a penalty term that will help even the playing field. First we have to find out what that penalty term should be by running the model with several different terms.

```
lambdas <- seq(0, 10, 0.25) #looking for the best lambda (penalty trem)

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

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

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

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

  predicted_ratings <-
    test_set %>%
```

```
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    pull(pred)

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

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

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

The best penalty term in this model which accounts for user, movie and genre effect is 4.75. Knowing that, we can now run the model and see that RMSE has indeed improved.

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

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

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

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

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

regularised_m_u_g_rmse<-RMSE(predicted_ratings, test_set$rating)

rmse_results<- bind_rows(rmse_results, data_frame(method = "Regularised movie + user + genre effect mod
                                                  RMSE = regularised_m_u_g_rmse))
knitr::kable(rmse_results)
```

method	RMSE
Just the average	1.0599043
Movie effect model	0.9437429
Movie + user effect model	0.8659319
Movie + user + genre effect model	0.8655941
Regularised movie + user + genre effect model	0.8649407

5.6. Regularized model using a combination of movie, user, genre and time of rating effect

Lastly, we also noticed that the number of ratings and their average also vary by year, so we will add one more bias term - the timestamp. Agan due to the variability accross users, movies, genres and time of rating, we first calculate penalty term.

```
# regularising the estimates to account for variability in the number of observations

# transforming the timestamp into date format
class(train_set$timestamp)<- c('POSIXt','POSIXct')
class(test_set$timestamp)<- c('POSIXt','POSIXct')

train_set$timestamp<- format(as.Date(train_set$timestamp), "%Y-%m")
test_set$timestamp<- format(as.Date(test_set$timestamp), "%Y-%m")

lambdas <- seq(0, 10, 0.25) #looking for the best lambda (penalty trem)

rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

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

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

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

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

  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_t, by="timestamp") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})
```

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

The best penalty term that minimises RMSE is now 5.

Running this more finely tuned model finally delivers RMSE in the desirable range.

method	RMSE
Just the average	1.0599043
Movie effect model	0.9437429
Movie + user effect model	0.8659319
Movie + user + genre effect model	0.8655941
Regularised movie + user + genre effect model	0.8649407
Regularised movie + user + genre + timestamp effect model	0.8648562

This is the model we choose to use to predict the ratings in the validation set.

6. Results

Using the regularized model accounting for a combination of movie, user, genre and time of rating effect with a penalty term, user bias, movie bias, genre bias and time of rating bias, we achieve a predictive model that when trained on the edx set produces RMSE of 0.8643 wich brings us in the desired range (RMSE <= 0.86490).

```
mu <- mean(edx$rating)
lambda<- 5 #from the above fitting

# creating the reglurarised estimates
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

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

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

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

#predicting the ratings
predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
```

```
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_t, by="timestamp") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_t) %>%
  pull(pred)

# RMSE result ----
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8643743
```

7. Conclusion

This report took us through the MoveLens 10M data set provided by GroupLens and pre-processed by HarvardX to allow for smooth exploratory analysis and modelling.

The entire set was split 90/10 to achieve edx and validation sets, used for modelling the data and validating the final model respectively.

The set contained 10M observations of 6 variables wich were explored in detail in the Exploratory Analysis section of this report. It also gave interesting insight into most popular movies, genres and user bahaviour.

The predictive model that best fit the data and provided the lowest RMSE amongst the models explored was the last one, which accounted for regularised user, movie, genre and time of rating biases, thereby providing a close estimate of the actual ratings presented in the validation set. The results of each of the models explored are summarised in the table below:

method	RMSE
Just the average	1.0599043
Movie effect model	0.9437429
Movie + user effect model	0.8659319
Movie + user + genre effect model	0.8655941
Regularised movie + user + genre effect model	0.8649407
Regularised movie + user + genre + timestamp effect model	0.8648562

8. Limitations and further work

Only 6 models were explored in predicting the ratings in the validation set, and due to the data size and low processing computer power those were all "processing-light" methods which allow for the code to be run on any machine.

To create more precise predictions, other predictive methodologies can be explored, such as regression trees for example.