MovieLens Project Report

Rajashekar Vasanth

22nd December 2019

Introduction

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such streaming services, recommender systems have garnered importance and have become a forefront of any consumer facing technology. From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences) to streaming services(suggesting users which movie/TV show they should be watching next), recommender systems are today unavoidable in our daily online journeys.

The aim of this project is to build a movie recommendation system which will will recommend new movies to existing users using the MovieLens dataset.

MovieLens Dataset

The MovieLens data set was collected by GroupLens Research.Before we head to how the data looks like, let us install and import the different libraries we will be using during the course of building the recommendation system.

Importing the required libraries

Writing a function to install and import libraries.

```r
#Writing a function to install and import libraries.
fn_install_packages <- function(library_name) {
#Check if the library is present, if not install it
   if(!require(library_name,character.only=TRUE))
     install.packages(library_name,character.only=TRUE,repos = "http://cran.us.r-project.org")

#Import the library
   library(library_name,character.only=TRUE)
}
```

Creating a list of libraries required and importing all required libraries

Data Ingestion

The code provided by edX is run as below to ingest the movielens 10M dataset. The data is then split into train and tes set with the train set having 90% of the data.

We also ensure that the userIds and movieIds present in the validation set is also present in the train set.

```r
################################
# Create edx set, validation set
################################
```

```r
# Note: this process could take a couple of minutes
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# 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.character(genres))

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

# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# 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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

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

Exploratory Data Analysis

Looking at the top 5 rows of the dataset shows that it is made up of 6 columns. UserId, movieId, timestamp of rating, title of the movie and a list of genres associated with the movie. We also see here that the title of the movie contains the year of its release as well. We would, at a later stage, extract this information and use that to predict the ratings.

```r
head(edx)
```

```
##   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
```

The train dataset has 9000055 observations. While userId, movieId, timestamp and rating are numeric columns, title and genres are character columns.

```r
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 8
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
```

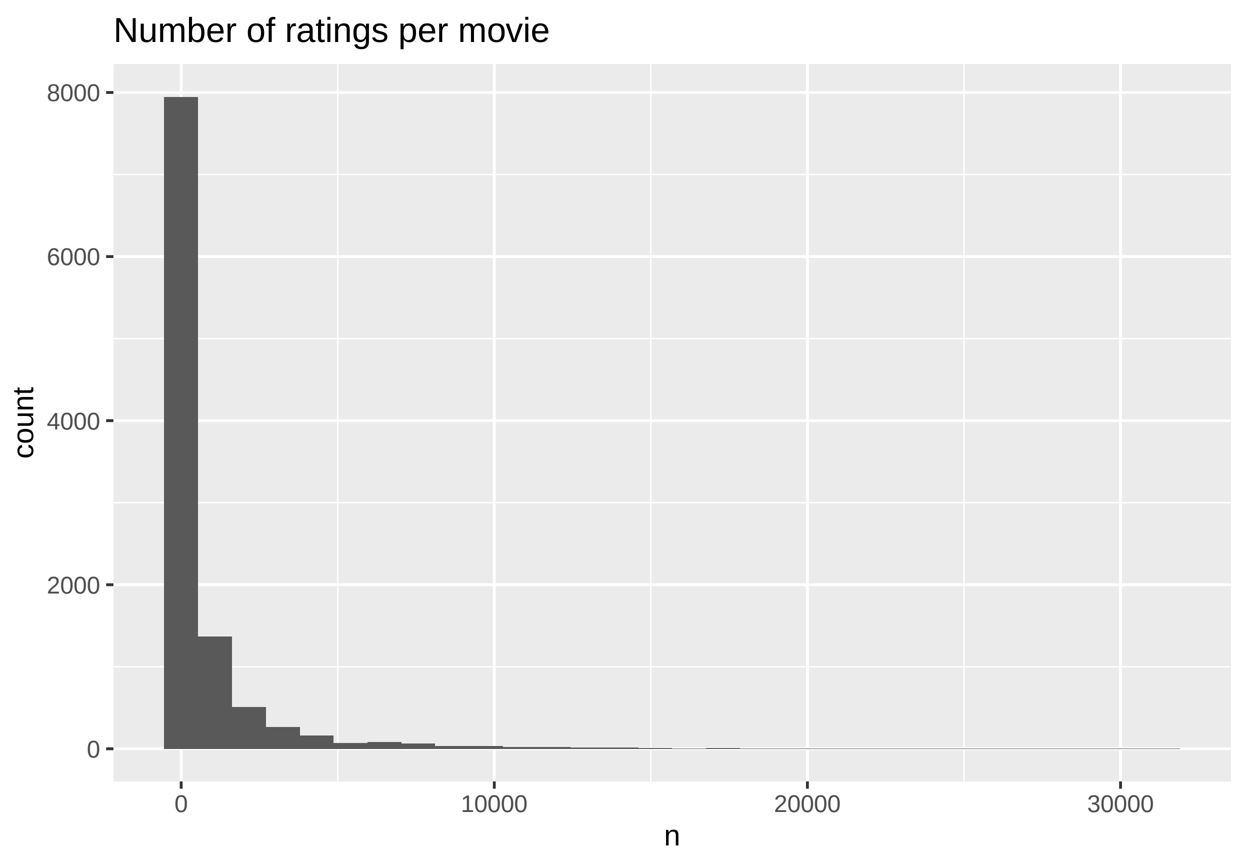
How many movies are there in total and how many users are there in total?

```r
edx %>% summarize(distinct_users = n_distinct(userId), distinct_movies = n_distinct(movieId))
```

```
##   distinct_users distinct_movies
## 1          69878           10677
```

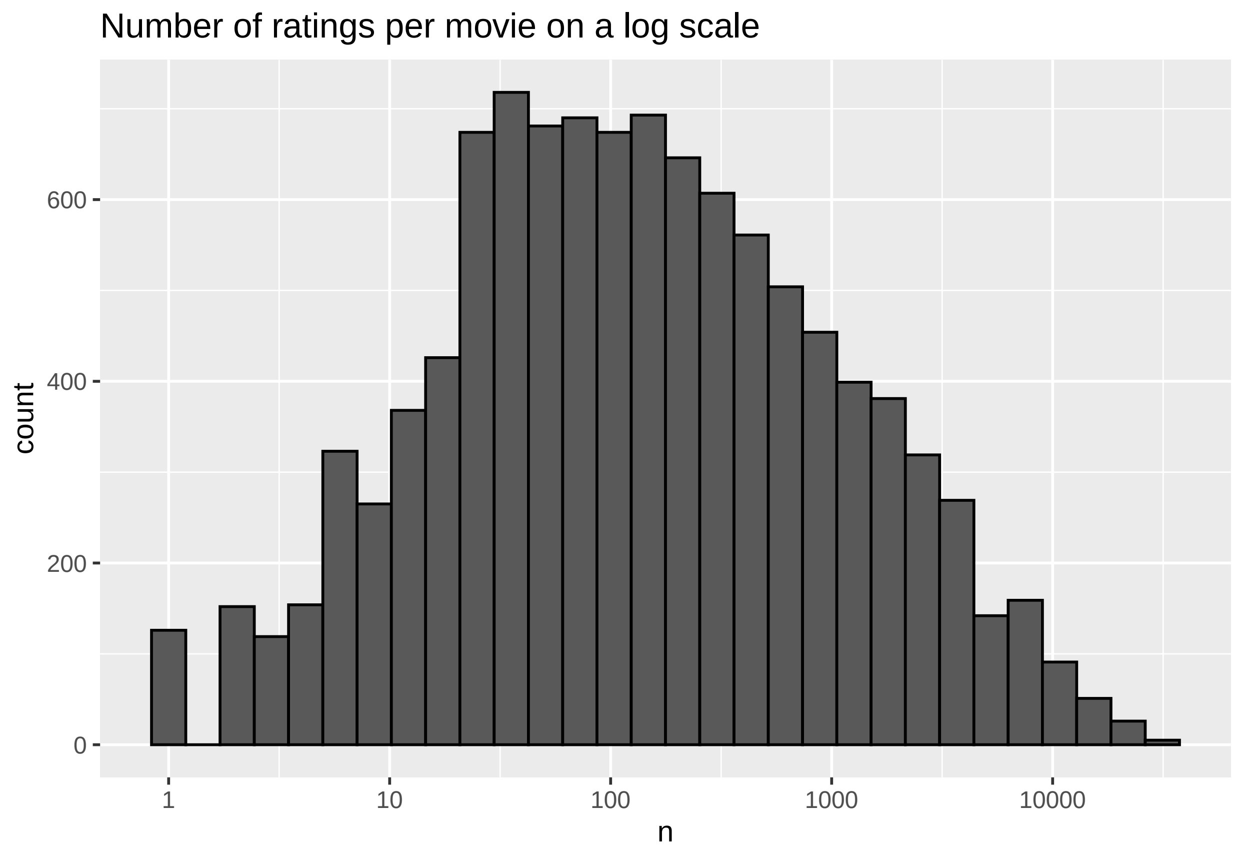
Visualizing the number of ratings per movie as a histogram. The graph below shows that just a few movies have several ratings, while a majority of the movies have very few ratings.

```r
edx %>% group_by(movieId) %>% summarize(n = n()) %>%
  ggplot(aes(x=n)) + geom_histogram(bins = 30) +
  ggtitle("Number of ratings per movie")
```



Converting the X-Axis to log scale to get a better understanding of the above histogram:

```
edx %>%
     dplyr::count(movieId) %>%
     ggplot(aes(n)) +
     geom_histogram(bins = 30, color = "black") +
     scale_x_log10()+ggtitle("Number of ratings per movie on a log scale")
```



Data transformation

Performing some transformations to get a better understanding of the movie genres and the time they were released.

I have performed three transformations:

1. Splitting genres into separate rows. Getting a distinct movie-genre list and then performing separate_rows operation as doing it in a single step on the edx dataset was consuming a lot of memory and my machine did not have sufficient memory.
2. Extracting the year of rating from the 'timestamp' column and extarcting the 'year of rating' from the movie title.
3. Apart from year of release and year of rating, another factor that might influence its rating its the during for which it has been released. The column 'age_while_rated' shows the duration for which the movie was released when it was rated.

```
#Data Transformation

#Getting the year of rating by converting the 'timestamp' column to year.
edx <- edx %>% mutate(year_rating = as.numeric(format(as.Date(as.POSIXct(timestamp, origin="1970-01-01")

#Getting the year of release from the title of the movie
edx <- edx %>% mutate(year_release = str_sub(str_sub(title,-5,-1),1,4))
```

```r
#Apart from these, also getting the age of the movie in years when it was rated.
edx <- edx %>% mutate(age_while_rated = as.numeric(year_rating) - as.numeric(year_release))

#Performing the same transformation on the validation set.
validation <- validation %>%
  mutate(year_rating = as.numeric(
    format(as.Date(as.POSIXct(timestamp, origin="1970-01-01")),"%Y")))
validation <- validation %>%
  mutate(year_release = str_sub(str_sub(title,-5,-1),1,4))
validation <- validation %>%
  mutate(age_while_rated = as.numeric(year_rating) - as.numeric(year_release))

#Separating the genres into separate rows.
#Getting a list of distinct movies and genres to reduce the dataset size. This can be done in a single
movies <- edx %>% distinct(movieId,genres)

#Separating the genres into separate rows.
movies_genre <- movies %>% separate_rows(genres, sep = "\\|")

#Combining it with the original dataset
edx_genre <- edx %>% inner_join(movies_genre,by="movieId")

#Viewing the new data
head(edx_genre)
```

```
##   userId movieId rating timestamp            title
## 1      1     122      5 838985046 Boomerang (1992)
## 2      1     122      5 838985046 Boomerang (1992)
## 3      1     185      5 838983525  Net, The (1995)
## 4      1     185      5 838983525  Net, The (1995)
## 5      1     185      5 838983525  Net, The (1995)
## 6      1     292      5 838983421  Outbreak (1995)
##                         genres.x year_rating year_release age_while_rated
## 1                 Comedy|Romance        1996         1992               4
## 2                 Comedy|Romance        1996         1992               4
## 3          Action|Crime|Thriller        1996         1995               1
## 4          Action|Crime|Thriller        1996         1995               1
## 5          Action|Crime|Thriller        1996         1995               1
## 6 Action|Drama|Sci-Fi|Thriller        1996         1995               1
##   genres.y
## 1   Comedy
## 2  Romance
## 3   Action
## 4    Crime
## 5 Thriller
## 6   Action
```

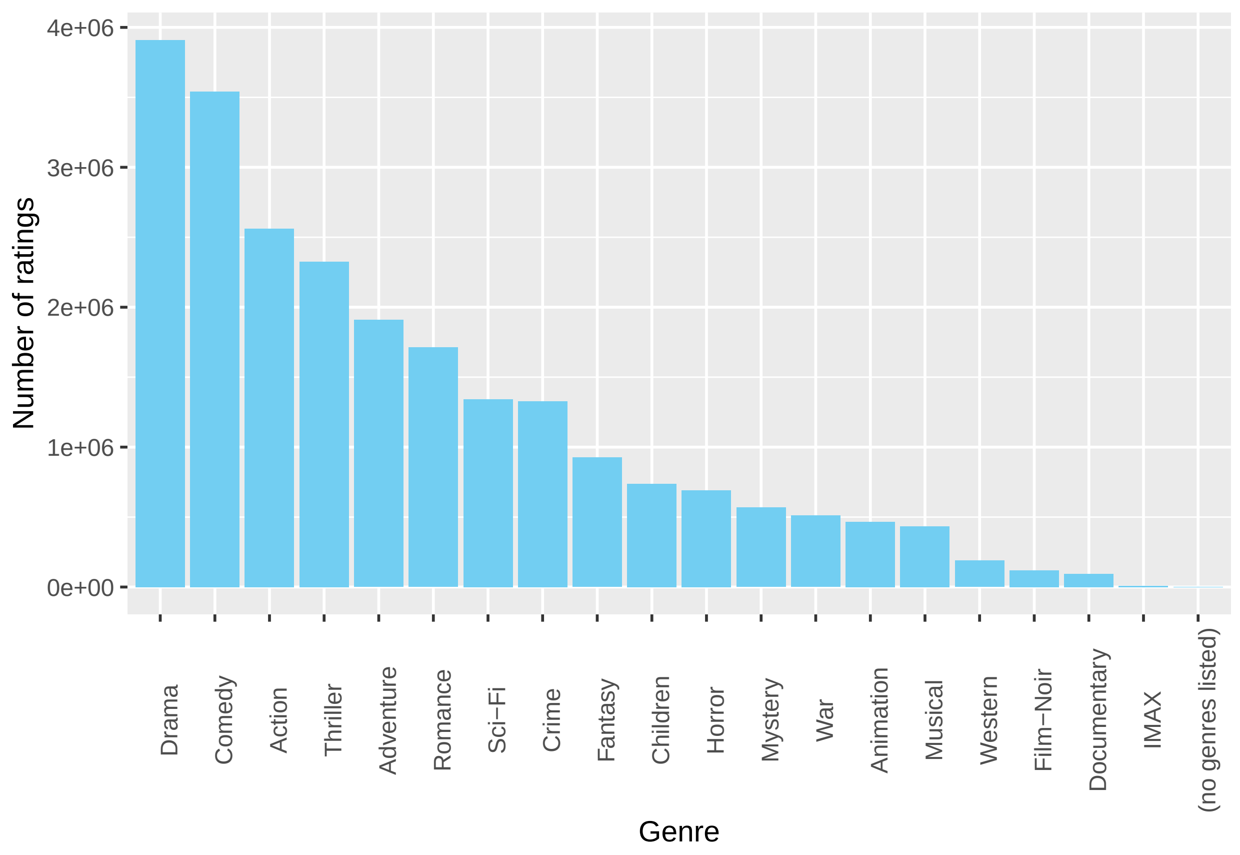
Counting the ratings that each genre has received. Drama, Comedy and Action are the top 3 most popular genres.

```r
edx_genre %>% group_by(genres.y) %>% summarize(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    genres.y              count
##    <chr>                 <int>
##  1 Drama               3910127
##  2 Comedy              3540930
##  3 Action              2560545
##  4 Thriller            2325899
##  5 Adventure           1908892
##  6 Romance             1712100
##  7 Sci-Fi              1341183
##  8 Crime               1327715
##  9 Fantasy              925637
## 10 Children             737994
## 11 Horror               691485
## 12 Mystery              568332
## 13 War                  511147
## 14 Animation            467168
## 15 Musical              433080
## 16 Western              189394
## 17 Film-Noir            118541
## 18 Documentary           93066
## 19 IMAX                   8181
## 20 (no genres listed)        7
```

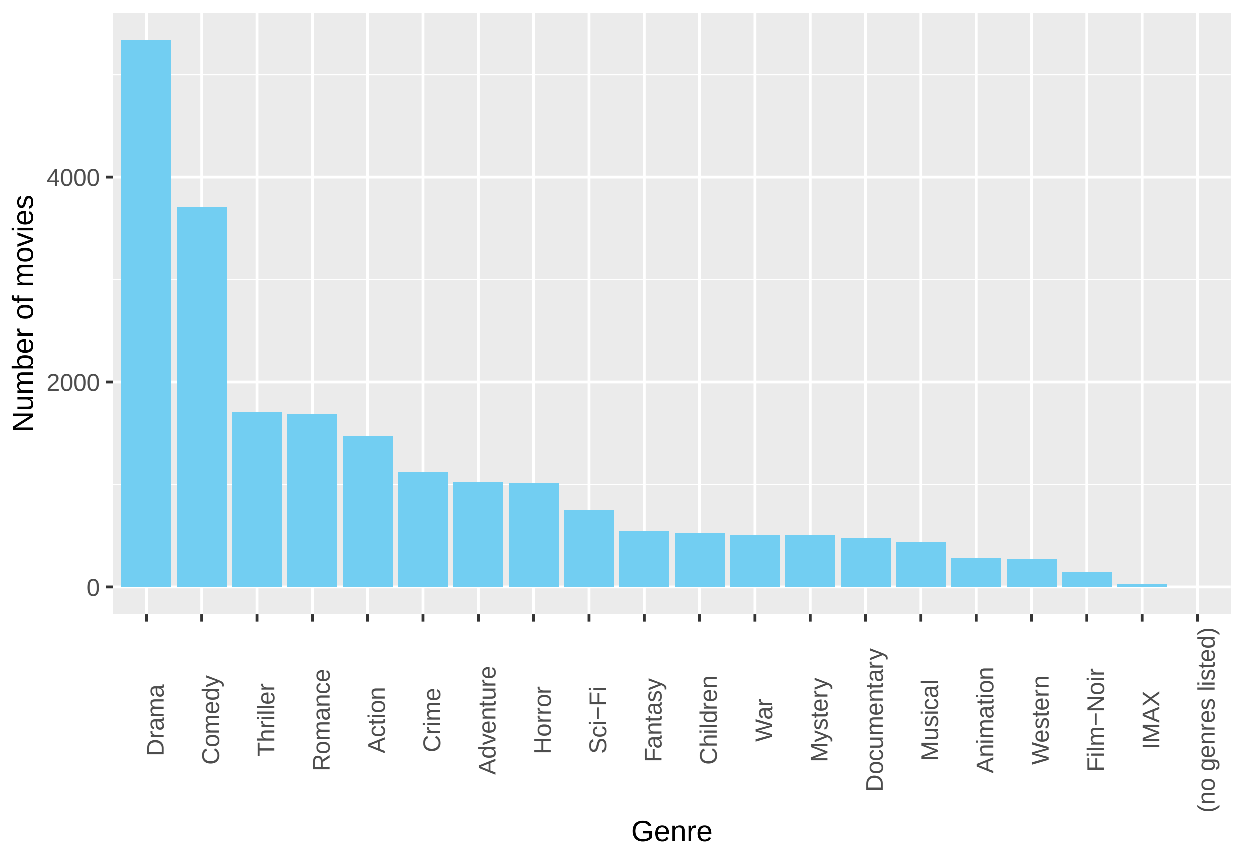
Visualizing the popularity on a bar graph.

```
edx_genre %>% group_by(genres.y) %>% summarize(count = n()) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x= reorder(as.factor(genres.y),-count),y=count)) +
  geom_bar( fill = "#72cef2",stat = "identity") +
  theme(axis.text.x = element_text(angle = 90))+
  labs(y="Number of ratings", x = "Genre")
```



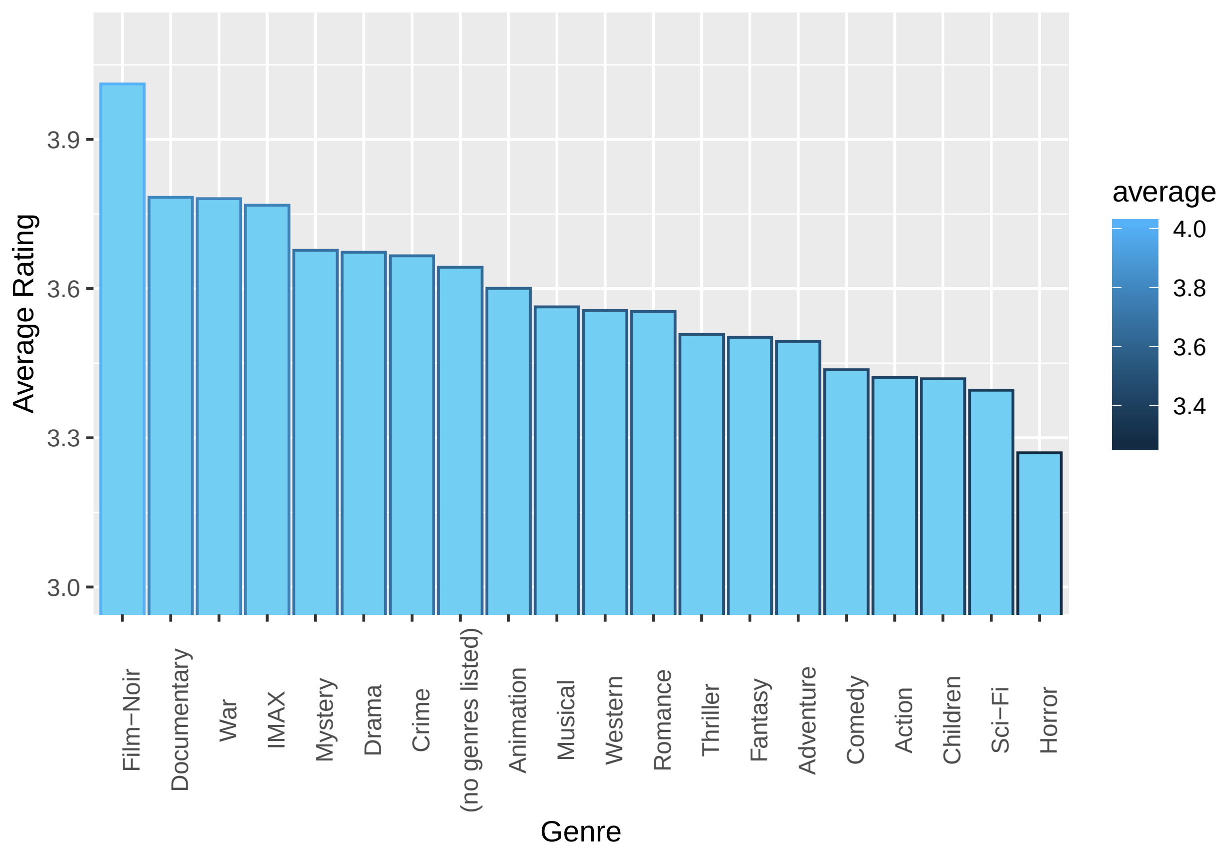
Visualizing the number of movies falling under each genre. This seems to follow the popularity trend as above.

```
edx_genre %>% group_by(genres.y) %>%
  summarize(count = n_distinct(movieId)) %>%
  arrange(desc(count)) %>%
  ggplot(aes(x= reorder(as.factor(genres.y),-count),y=count)) +
  geom_bar( fill = "#72cef2",stat = "identity") +
  theme(axis.text.x = element_text(angle = 90))+
  labs(y="Number of movies", x = "Genre")
```



Visualizing the average rating for each genre, we realize that Documentary and War movies are rated higher whereas Sci-Fi and horror movies are rated lower. As the absolute differences are marginal and even the small marginality makes a significant difference, I have curtailed the Y axis to be in the range of 3 to 4. This makes the difference more pronounced.

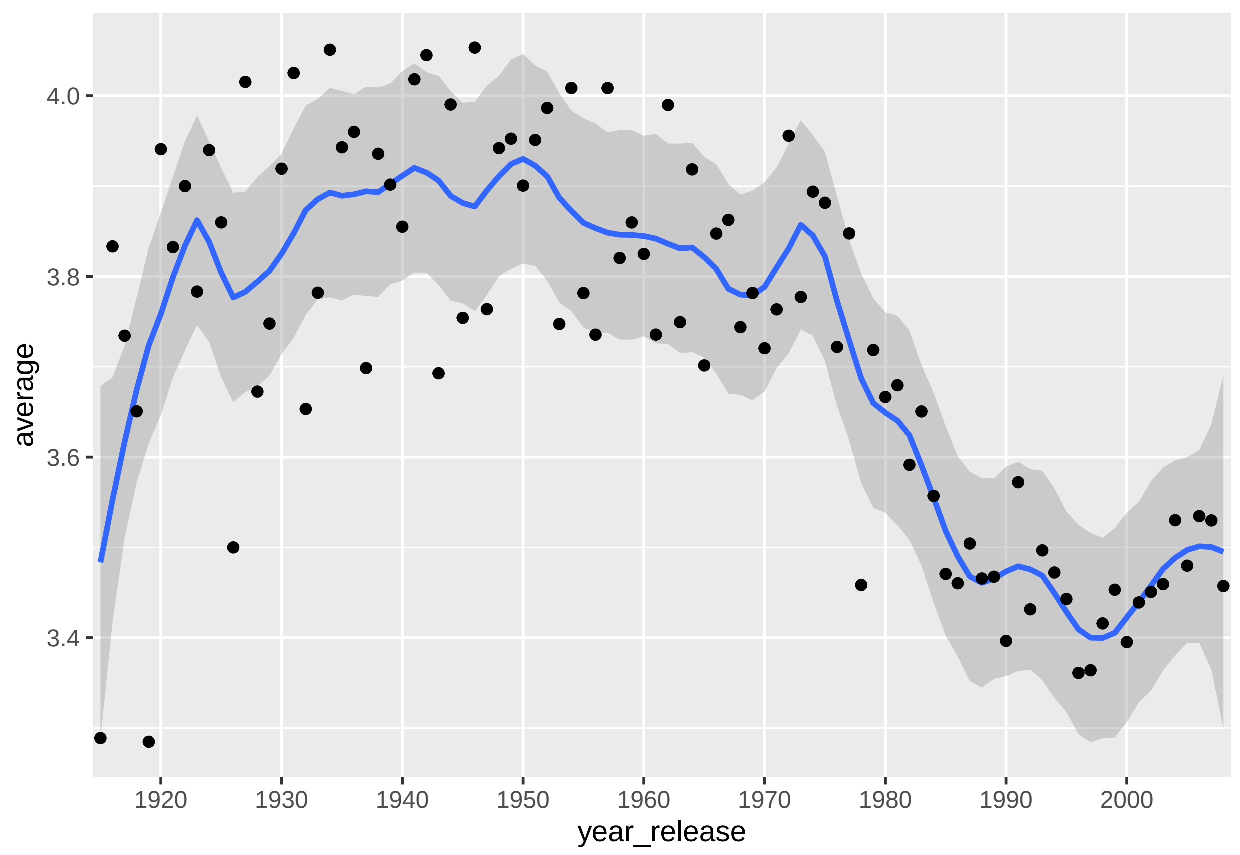
```r
edx_genre %>% group_by(genres.y) %>%
  summarize(average = mean(rating)) %>%
  arrange(desc(average)) %>%
  ggplot(aes(x= reorder(as.factor(genres.y),-average),y=average, color = average)) +
  geom_bar( fill = "#72cef2",stat = "identity") +
  theme(axis.text.x = element_text(angle = 90))+
  labs(y="Average Rating", x = "Genre") +
  coord_cartesian(ylim = c(3, 4.1))
```



Visualizing how the average rating is impacted by the year of release of movie.We see that older movies have significantly higher movie ratings compared to the newer ones. This can be one of the important features while we build models for prediction.

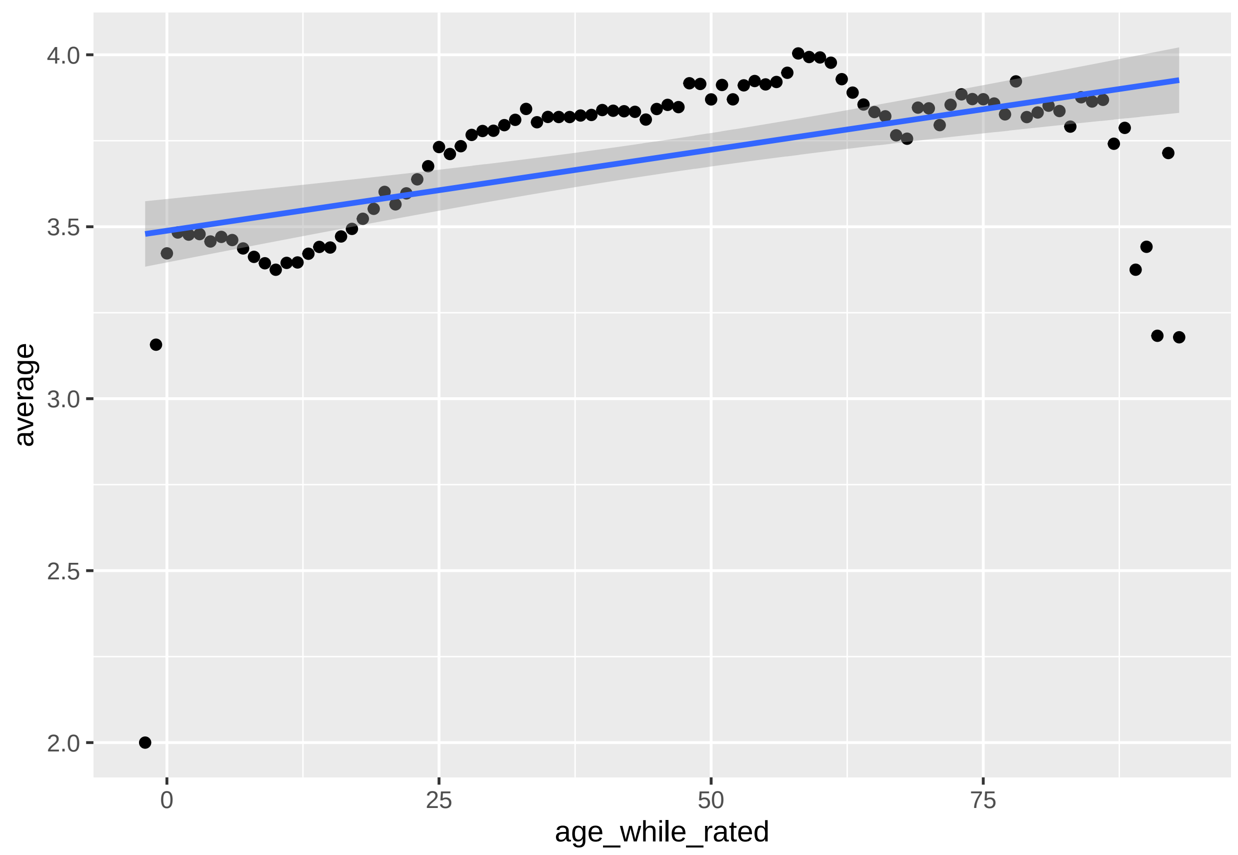
```
edx %>% group_by(year_release) %>%
  summarize(count = n(), average = mean(rating)) %>% arrange(desc(count,average)) %>%
  ggplot(aes(year_release,average,group=1)) +
  geom_smooth(span = 0.15) +
  geom_point() +
  scale_x_discrete(breaks=seq(1900, 2030, 10))
```

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



To understand this better, I created another feature which has the age of movie when it was rated.The below graph shows that newer movies get lesser of a rating and as the age of the movie increases, its average rating tends to get better.

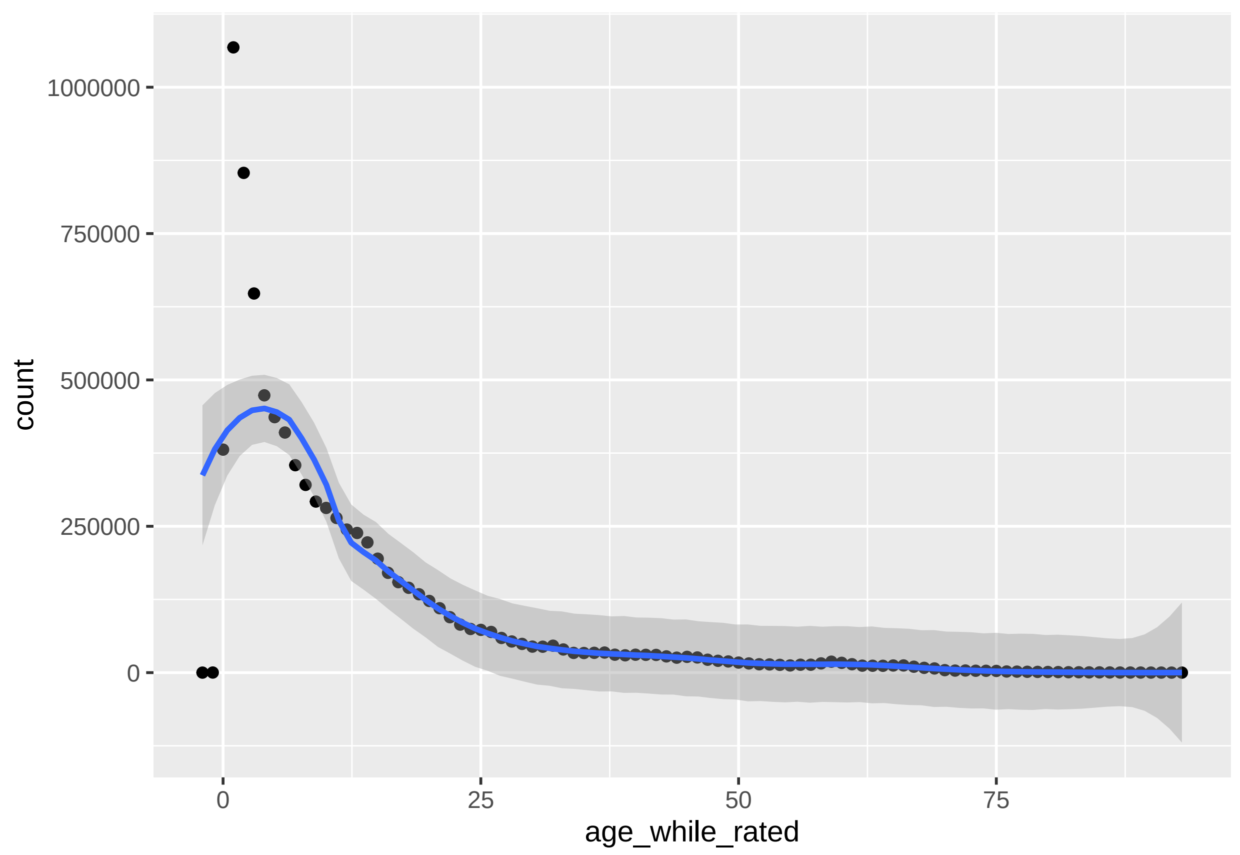
```
edx %>% group_by(age_while_rated) %>%
  summarize(average = mean(rating)) %>%
  ggplot(aes(age_while_rated,average)) +
  geom_point() +
  geom_smooth(method = lm)
```



Visualizing the number of ratings along with the age of the movie when rated shows that there are a lot of ratings as soon as the movie is released and it sharply decreases with time.

```
edx %>% group_by(age_while_rated) %>%
  summarize(count = n()) %>%
  ggplot(aes(age_while_rated,count)) +
  geom_point() +
  geom_smooth(span=0.3)
```

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



Converting the edx dataframe to a matrix to apply machine learning models. We pivot the genre column so that all the genres are transformed to columns and the values filled will be 1/0 based on whether the movie belongs to that genre or not.

```
#Removing the old list of genres.
edx_genre <- edx_genre %>% select(-genres.x)

#Adding a placeholder value of 1 so that we can pivot the genres as columns.
edx_genre$val <- 1

#Pivoting the genres from rows to columns
edx_matrix <- spread(edx_genre,key = genres.y,value = val, fill = 0)

#Viewing the pivoted matrix
head(edx_matrix)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##   year_rating year_release age_while_rated (no genres listed) Action
## 1        1996         1992               4                  0      0
## 2        1996         1995               1                  0      1
## 3        1996         1995                 1                  0      1
## 4        1996         1994                 2                  0      1
## 5        1996         1994                 2                  0      1
## 6        1996         1994                 2                  0      0
##   Adventure Animation Children Comedy Crime Documentary Drama Fantasy
## 1         0         0        0      1     0           0     0       0
## 2         0         0        0      0     1           0     0       0
## 3         0         0        0      0     0           0     1       0
## 4         1         0        0      0     0           0     0       0
## 5         1         0        0      0     0           0     1       0
## 6         0         0        1      1     0           0     0       1
##   Film-Noir Horror IMAX Musical Mystery Romance Sci-Fi Thriller War
## 1         0      0    0       0       0       1      0        0   0
## 2         0      0    0       0       0       0      0        1   0
## 3         0      0    0       0       0       0      1        1   0
## 4         0      0    0       0       0       0      1        0   0
## 5         0      0    0       0       0       0      1        0   0
## 6         0      0    0       0       0       0      0        0   0
##   Western
## 1       0
## 2       0
## 3       0
## 4       0
## 5       0
## 6       0
```

Model Training

I have not repeated what was taught during the course - the movie effect, user effect and regularization models. I have instead explored the different models caret has to offer and also modelled it using the matrix factorization approach.

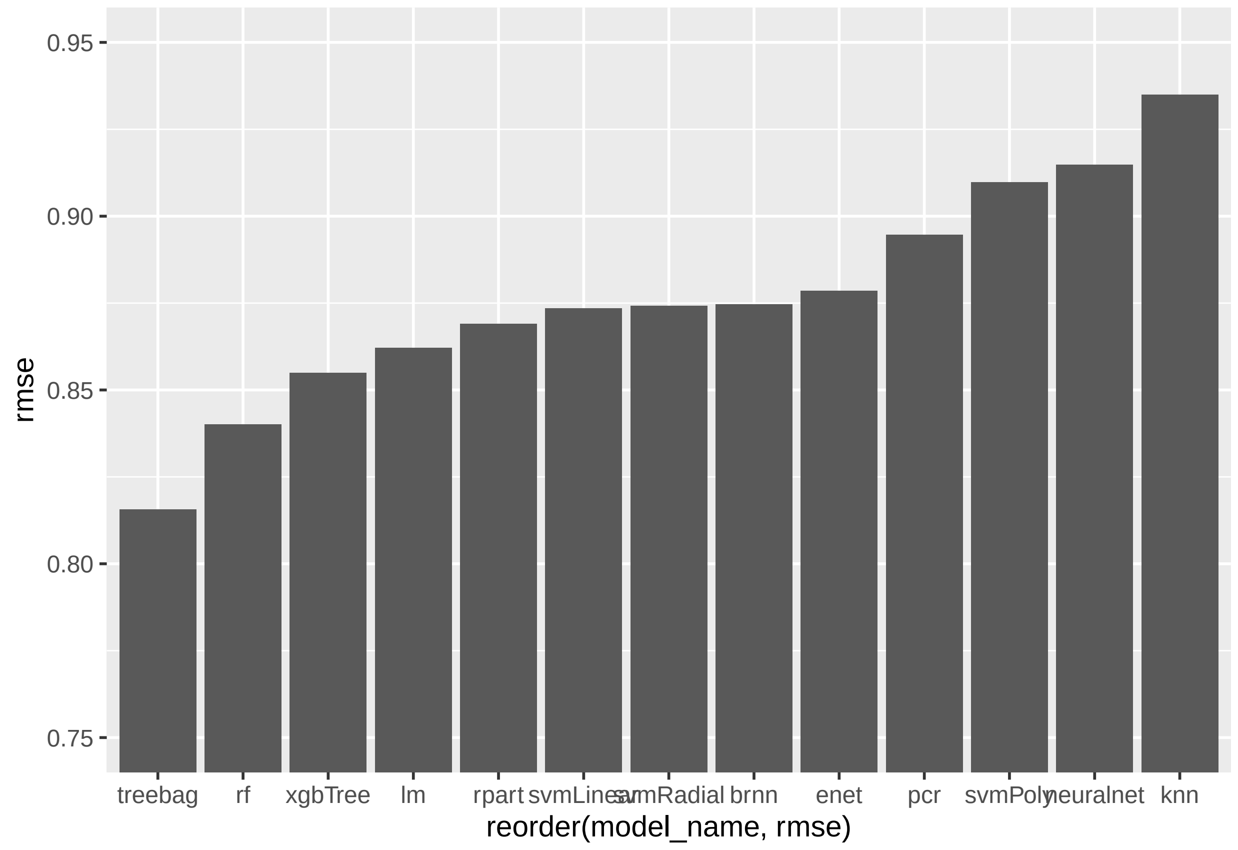
Modelling Approach 1: CARET Regression Models

I first convert the edx dataframe to a matrix to apply machine learning models.

Regression models like lm“,”neuralnet“,”rpart“,”enet“,”brnn“,”pcr“,”treebag“,”xgbTree“,”gamLoess“,”knn“,”svmLinear“,”svr
and “rf” are applied to train set and a dataframe consisting of the model name and train RMSE is generated.

Plotting the RMSEs of all the trained models and viewing it on a bar graph. We see that the treebag and the random forest models have the lowest RMSEs on the trained model. Before we zero in on a model, let us look at a different approach to solving this recommendation problem - Matrix Factorization.

```
df_models %>%
  filter(model_name != "gamLoess")%>%
  ggplot(aes(reorder(model_name,rmse),rmse)) +
  geom_bar(stat = "identity") +
  coord_cartesian(ylim = c(0.75, 0.95))
```



Modelling Approach 2: Matrix Factorization

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. This family of methods became widely known during the Netflix prize challenge due to its effectiveness as reported by Simon Funk in his 2006 blog post, where he shared his findings with the research community.

We will apply Matrix Factorization with parallel stochastic gradient descent. With the help of "recosystem" package it is an R wrapper of the LIBMF library which creates a Recommender System by Using Parallel Matrix Factorization. The main task of recommender system is to predict unknown entries in the rating matrix based on observed values.

More info on the recosystem package and the techniques here: https://www.rdocumentation.org/packages/recosystem/versions/0.3

```
#Function to calculate RMSE
RMSE <- function(true_ratings, predicted_ratings) {
    sqrt(mean((true_ratings - predicted_ratings)^2))
}

edx_mf <- edx %>% select(movieId, userId, rating)
validation_mf <- validation %>% select(movieId, userId, rating)

#The data file for training set needs to be arranged in sparse
#matrix triplet form, i.e., each line in the
```

```r
#file contains three numbers in order to use recosystem
#package we create 2 new matrices (our train and our
#validation set) with the below 3 features

edx_mf <- as.matrix(edx_mf)
validation_mf <- as.matrix(validation_mf)

#Importing Recosystem library
if(!require(tidyverse)) install.packages("recosystem", repos = "http://cran.us.r-project.org")
library(recosystem)

#Recosystem needs data to be saved as files on harddisk.

write.table(edx_mf, file = "trainingset.txt", sep = " ", row.names = FALSE,
    col.names = FALSE)

write.table(validation_mf, file = "validationset.txt", sep = " ",
    row.names = FALSE, col.names = FALSE)


#Setting seed to 1 to get repeatable results.
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
#Getting the training and testing datasets from the file stored on the hard disk.
training_dataset <- data_file("trainingset.txt")
validation_dataset <- data_file("validationset.txt")

#Creating a model object
r = Reco()

#call the $tune() method to select best tuning parameters along a set of candidate values.
opts = r$tune(training_dataset, opts = list(dim = c(10, 20, 30), lrate = c(0.1,
    0.2), costp_l1 = 0, costq_l1 = 0, nthread = 1, niter = 10))
```

Let us now train the model using matrix factorization approach and parameters obtained from the previous step.

```r
#Train the model by calling the $train() method.
#A number of parameters can be set inside the function,
#possibly coming from the result of $tune().
r$train(training_dataset, opts = c(opts$min, nthread = 1, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9736  1.2042e+007
##    1       0.8721  9.8790e+006
##    2       0.8378  9.1536e+006
##    3       0.8168  8.7473e+006
##    4       0.8012  8.4729e+006
##    5       0.7892  8.2723e+006
##    6       0.7795  8.1204e+006
##    7       0.7713  7.9959e+006
##    8       0.7646  7.9044e+006
##    9       0.7588  7.8245e+006
##   10       0.7537  7.7575e+006
##   11       0.7492  7.7015e+006
##   12       0.7452  7.6522e+006
##   13       0.7415  7.6098e+006
##   14       0.7382  7.5712e+006
##   15       0.7351  7.5374e+006
##   16       0.7322  7.5068e+006
##   17       0.7296  7.4768e+006
##   18       0.7271  7.4528e+006
##   19       0.7249  7.4299e+006
```

Training using matrix factorization gives us an RMSE of 0.7247 on the training set which is very less compared to the other caret models that we used before. Hence, I will continue using the matrix factorization approach to build a prediction on the validation dataset and calculate the final RMSE.

```
#Write predictions to a tempfile on HDisk
stored_prediction = tempfile()

#With the $predict() method we will make  predictions
#on validation set and will calculate RMSE:
r$predict(validation_dataset, out_file(stored_prediction))
```

```
## prediction output generated at C:\Users\RAJASH~1.VAS\AppData\Local\Temp\RtmpMHIvi5\file44a4e503502
```

```
#Store real ratings
real_ratings <- read.table("validationset.txt", header = FALSE, sep = " ")$V3

#Store predicted ratings
pred_ratings <- scan(stored_prediction)

#Calculate RMSE
rmse_of_model_mf <- RMSE(real_ratings, pred_ratings)
```

Let us now view the RMSE on validation dataset. The matrix factorization approach gives us an RMSE of 0.783 on the validation dataset.

```
rmse_of_model_mf
```

```
## [1] 0.7829341
```

Results

The lowest RMSE that could be achieved was 0.783 using the matrix factorization approach and using just two features: movieId and userId.

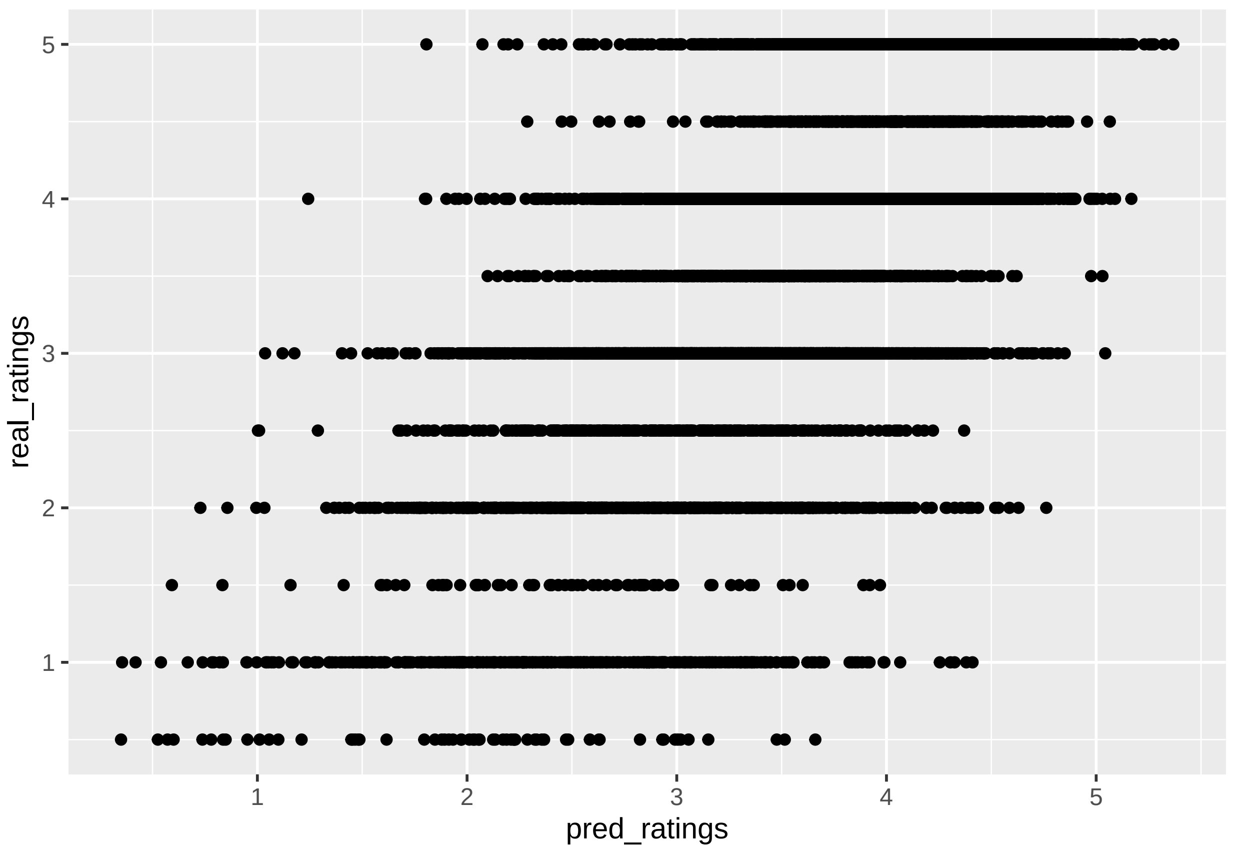
The second lowest RMSE was by treebag model which gave an RMSE of 0.83.

Limitations and future work

The above approaches have a few limitations. Firstly, I was unable to train the caret models using the full dataset as there were memory constraints. I am sure we can obtain a better RMSE for the caret models using the complete dataset. One of the areas that demands some more research is optimization of training of caret models.

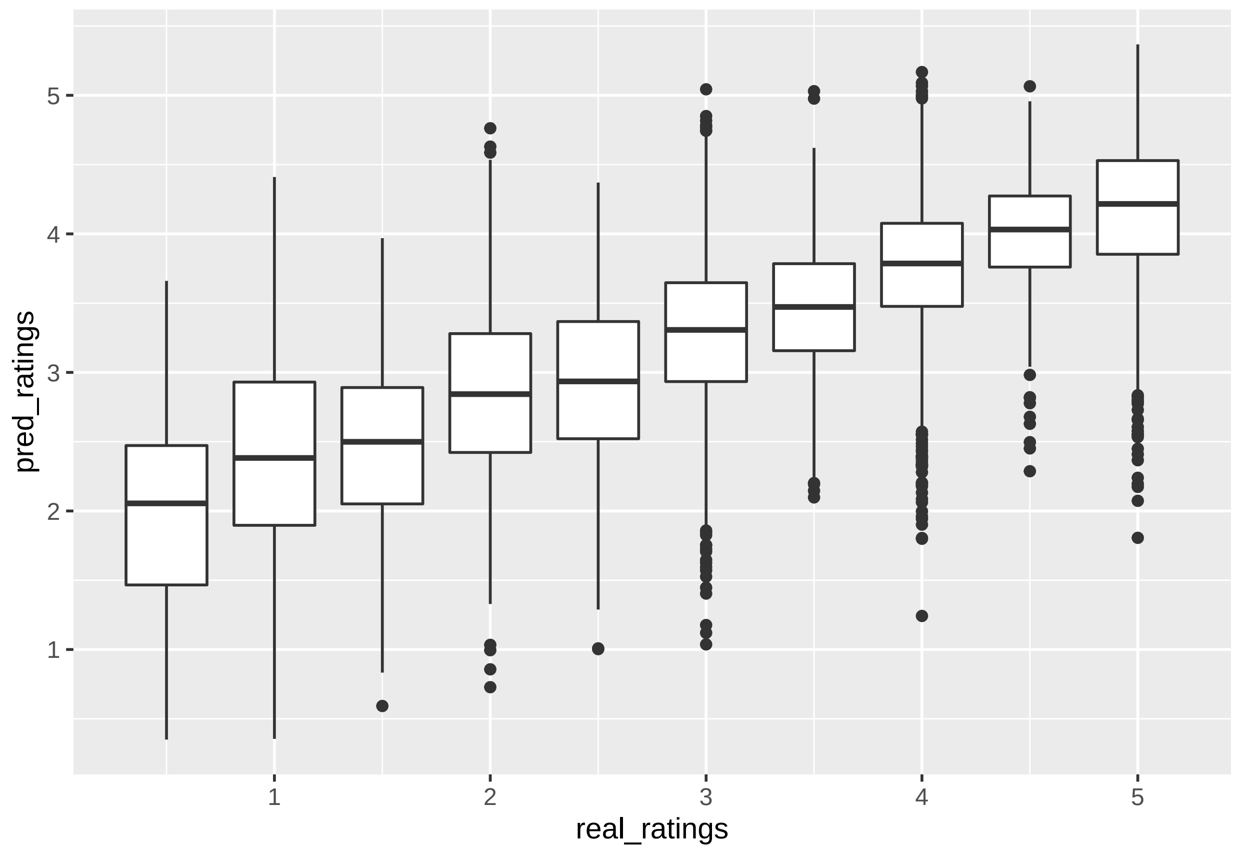
Secondly, with matrix factorization approach, on looking at the predicted values and plotting them as a scatterplot we can notice that there when grouped by real values, the predicted values seem follow some distribution pattern around the real ratings.

```
mf_ratings <- as.data.frame(real_ratings)
mf_ratings <- mf_ratings %>% mutate(pred_ratings = pred_ratings)
head(mf_ratings,10000) %>%
  ggplot(aes(pred_ratings,real_ratings)) +
  geom_point()
```



Plotting a boxplot of the same we see that the median of the predicted ratings somewhat regress towards the mean (that is 2.5). Can we add another bias variable which would push it away from the mean giving a more accurate rating? This is something that can be considered for further study.

```
head(mf_ratings,10000) %>%
  ggplot(aes(y = pred_ratings,x = real_ratings,group = real_ratings)) +
  geom_boxplot()
```



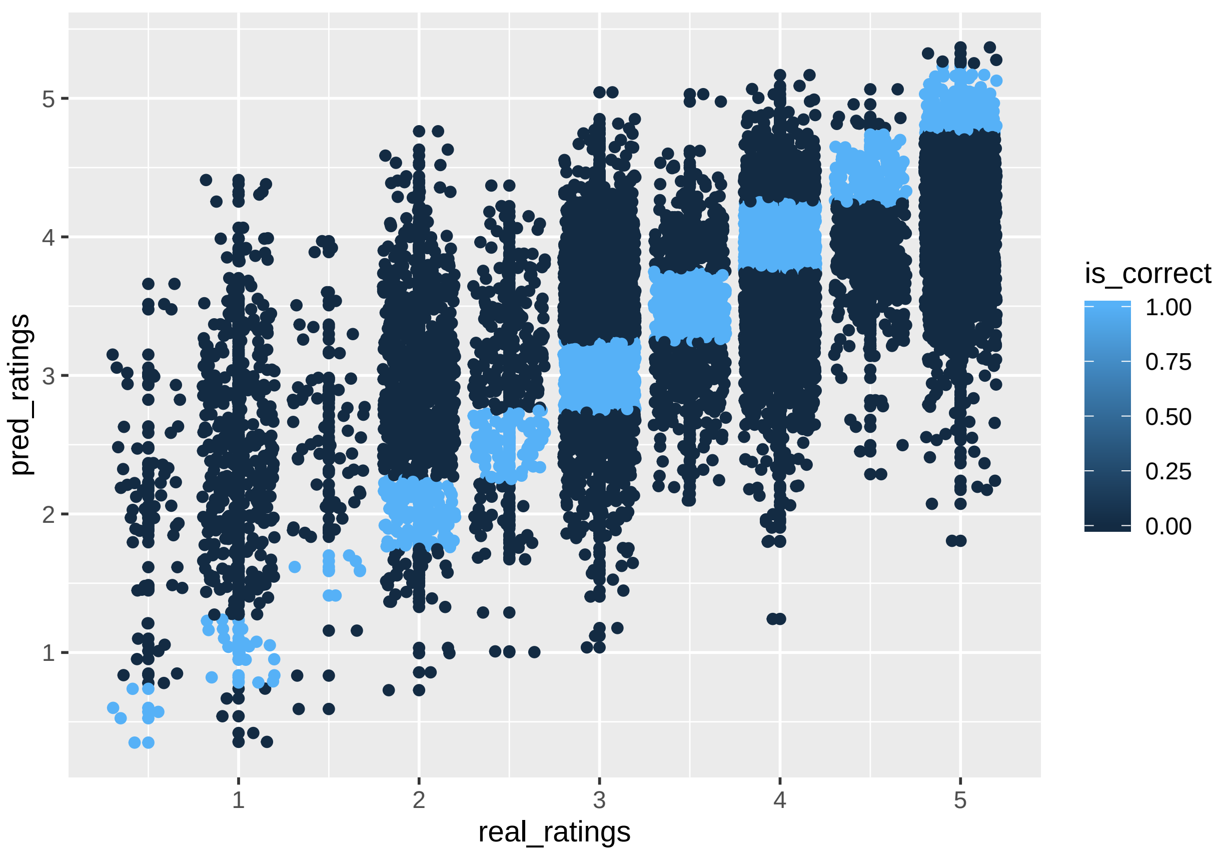
Looking at the predicted results and rounding them off to the nearest 0.5, we see that the number of ratings that are actually predicted accurately is still very less even if the RMSE is acceptable. This is another case that should be looked into.

```
pred_ratings_rounded <- round(pred_ratings/0.5) * 0.5

mf_ratings <- mf_ratings %>%
  mutate(pred_ratings_rounded = pred_ratings_rounded)

mf_ratings <- mf_ratings %>%
  mutate(is_correct = ifelse(pred_ratings_rounded==real_ratings,1,0))

head(mf_ratings,10000) %>%
  ggplot(aes(y = pred_ratings,x = real_ratings,color = is_correct)) +
  geom_point() +geom_jitter()
```


pred_ratings
5
4
3
2
1
real_ratings
is_correct
1.00
0.75
0.50
0.25
0.00