MovieLens

Aditya Taktode - Harvard Data Science Professional - Capstone

20/06/2020

```r
knitr::opts_chunk$set(echo = T, fig.align = 'center', cache = F, cache.lazy = F)
```

Executive Summary

The purpose of this project is creating a recommender system using the MovieLens dataset.

The version of movielens dataset used for this final assignment contains approximately 10 Million movie ratings, divided in 9 Milion for training and one Milion for validation.

It is a small subset of a much larger (and famous) dataset with several millions of ratings.

After a initial data exploration, the different recommender systems built on this dataset are evaluated and choosen based on the RMSE (Root Mean Squared Error) that should be at least lower than **0.87750**.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

The RMSE function

```r
RMSE <- function(predictions, actuals){
  d <- predictions - actuals
  d <- d^2
  sqrt(mean(as.numeric(d), na.rm = TRUE))
}
```

Getting set-up with the required libraries

```r
#Install all the needded libraries if not pressent already
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(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(biglm)) install.packages("biglm", repos = "http://cran.us.r-project.org")
if(!require(broom)) install.packages("broom", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(recommenderlab)) install.packages("recommenderlab", repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem", repos = "http://cran.us.r-project.org")

#Loading all the required libraries
library(tidyverse)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(gbm)
library(stringr)
library(biglm)
library(broom)
library(data.table)
library(lubridate)
library(recommenderlab)
library(recosystem)
```

Load and prepare the data

The 10 Million movielens dataset is divided into two sets: `dataset` for training purpose and `validation` for validation (i.e. final test) purpose.

```
# 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")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
dataset <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in dataset set
validation <- temp %>%
  semi_join(dataset, by = "movieId") %>%
  semi_join(dataset, by = "userId")

# Add rows removed from validation set back into dataset set
removed <- anti_join(temp, validation)
```

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

```
dataset <- rbind(dataset, removed)
```

Data Exploration

Preliminary data exploration

```
#Summary of dataset
glimpse(dataset)
```

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
```

The features in dataset are six:

- **userId** `<integer>` that contains the unique identification number for each user.
- **movieId** `<numeric>` that contains the unique identification number for each movie.
- **rating** `<numeric>` that contains the rating of one movie by one user. Ratings are made on a 5-Star scale with half-star increments.
- **timestamp** `<integer>` that contains the timestamp for one specific rating provided by one user.
- **title** `<character>` that contains the title of each movie including the year of the release.
- **genres** `<character>` that contains a list of pipe-separated of genre of each movie.

After having a glimpse of summary of our dataset, it has come to notice that the `genres` are pipe-separated values.

```
#peeking into 'genres'
head(dataset$genres, n = 20)
```

```
##  [1] "Comedy|Romance"
##  [2] "Action|Crime|Thriller"
##  [3] "Action|Drama|Sci-Fi|Thriller"
##  [4] "Action|Adventure|Sci-Fi"
##  [5] "Action|Adventure|Drama|Sci-Fi"
##  [6] "Children|Comedy|Fantasy"
##  [7] "Comedy|Drama|Romance|War"
##  [8] "Adventure|Children|Romance"
##  [9] "Adventure|Animation|Children|Drama|Musical"
## [10] "Action|Comedy"
## [11] "Action|Romance|Thriller"
## [12] "Action|Comedy|Crime|Thriller"
## [13] "Action|Comedy|War"
## [14] "Comedy"
## [15] "Comedy|Drama|Romance"
## [16] "Adventure|Animation|Children|Comedy|Musical"
## [17] "Action|Sci-Fi"
## [18] "Animation|Children|Drama|Fantasy|Musical"
## [19] "Animation|Children"
## [20] "Action|Drama|War"
```

It is necessary to extract them for better, robust and precise estimation.

```
#separate rows for extracting different genres
dataset <- dataset %>% separate_rows(genres, sep = "\\|")

#the new genres are
levels(factor(dataset$genres))
```

```
##  [1] "(no genres listed)" "Action"             "Adventure"
##  [4] "Animation"          "Children"           "Comedy"
##  [7] "Crime"              "Documentary"        "Drama"
## [10] "Fantasy"            "Film-Noir"          "Horror"
## [13] "IMAX"               "Musical"            "Mystery"
## [16] "Romance"            "Sci-Fi"             "Thriller"
## [19] "War"                "Western"
```

The new genres look better and well segregated.

We'll have to do the same for validation set as well

```
#separate rows for extracting different genres
validation <- validation %>% separate_rows(genres, sep = "\\|")
```

check for any NAs

```
#check for any missing values
sum(is.na(dataset))
```

```
## [1] 0
```

So, there are no missing values, cheers!!

Now, diving into the number of unique values for some features we've got

```
#total number of unique users
n_distinct(dataset$userId)
```

```
## [1] 69878
```

```
#total number of unique movies
n_distinct(dataset$movieId)
```

```
## [1] 10677
```

```
#total number of unique genres
n_distinct(dataset$genres)
```

```
## [1] 20
```

Ratings explorating analysis

Ratings range

```
range(dataset$rating)
```

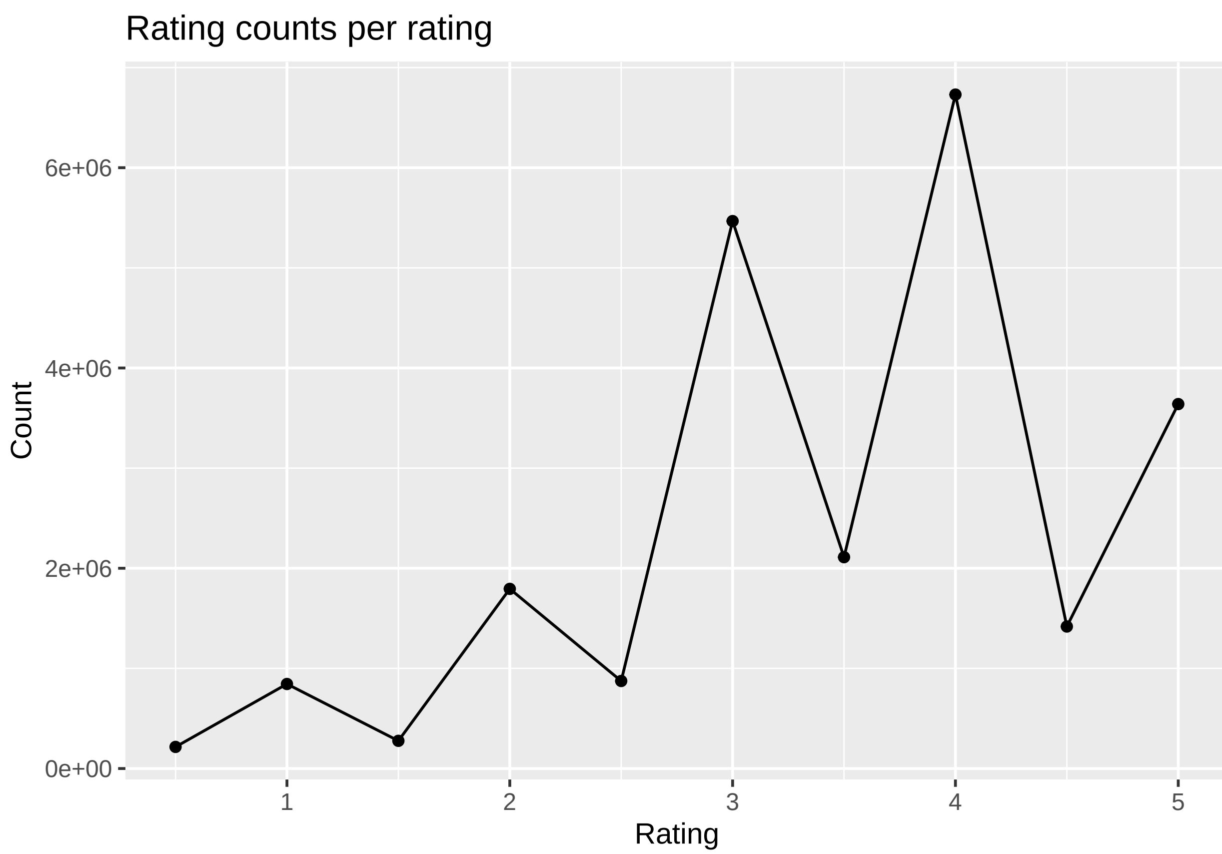
```
## [1] 0.5 5.0
```

The minimum possible rating is 0.5 whereas the maximum is 5

Rating distribution exploration

```
dataset %>% group_by(rating) %>% summarise(count = n()) %>% ggplot(aes(rating, count)) + geom_point() +
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

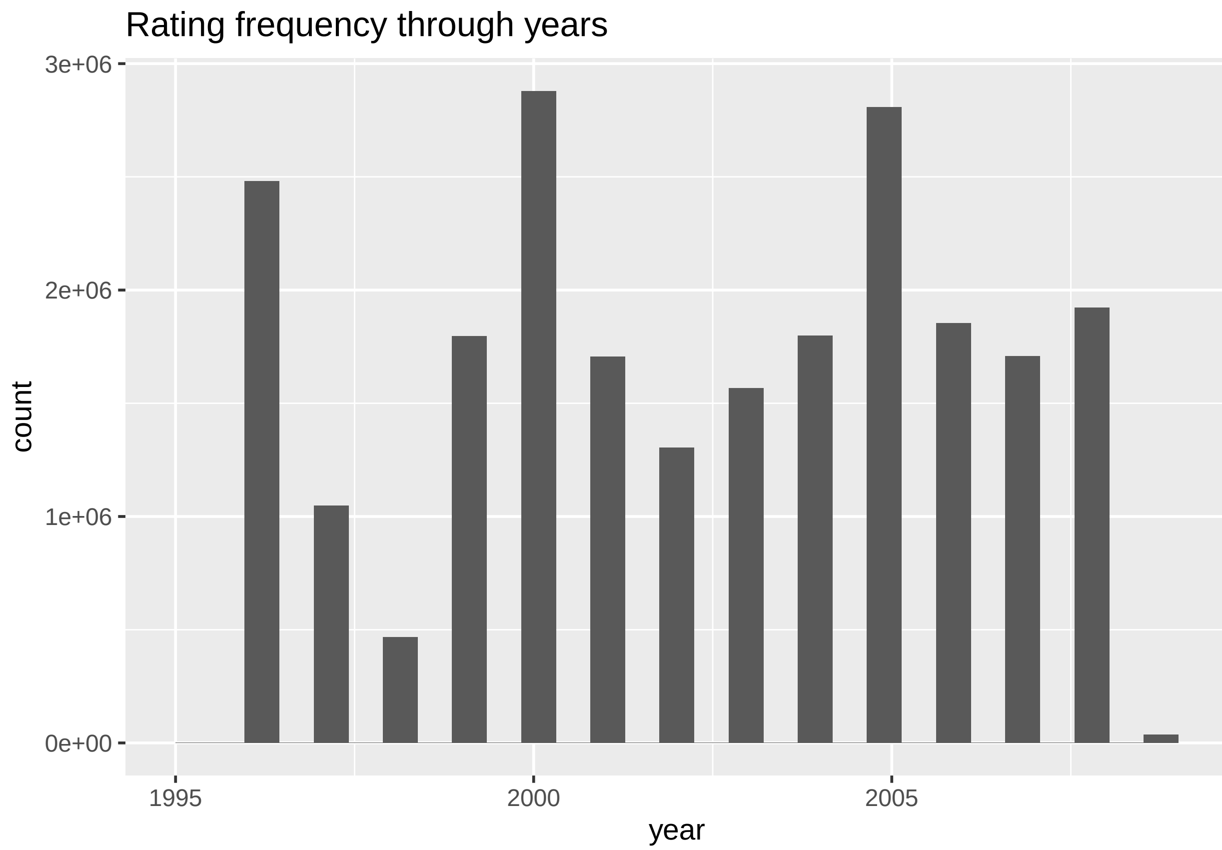


This visualizationg shows that there is a small amount of negative votes(i.e. below 3). Maybe, the users give ratings only if they like it. Also, half-star ratings are less likely as compared to the full-star ratings.

Overview of rating frequency through years

```
dataset %>% mutate(year = year(as_datetime(timestamp, origin = '1970-01-01'))) %>% ggplot(aes(x = year)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



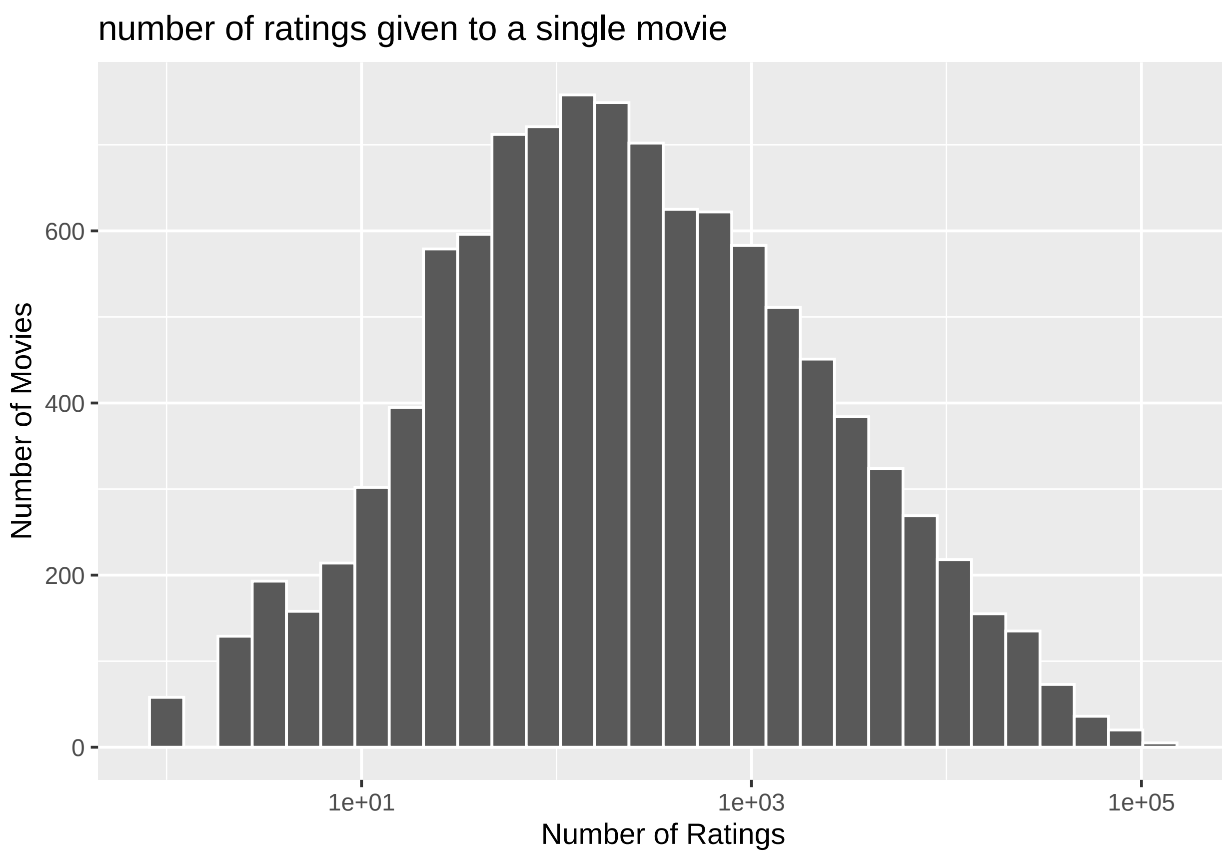
This visualizes frequench of user's ratings through years

Number of ratings given to a movie

```
dataset %>% group_by(movieId) %>% summarise(count = n()) %>% ggplot(aes(count)) + geom_histogram(col =
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



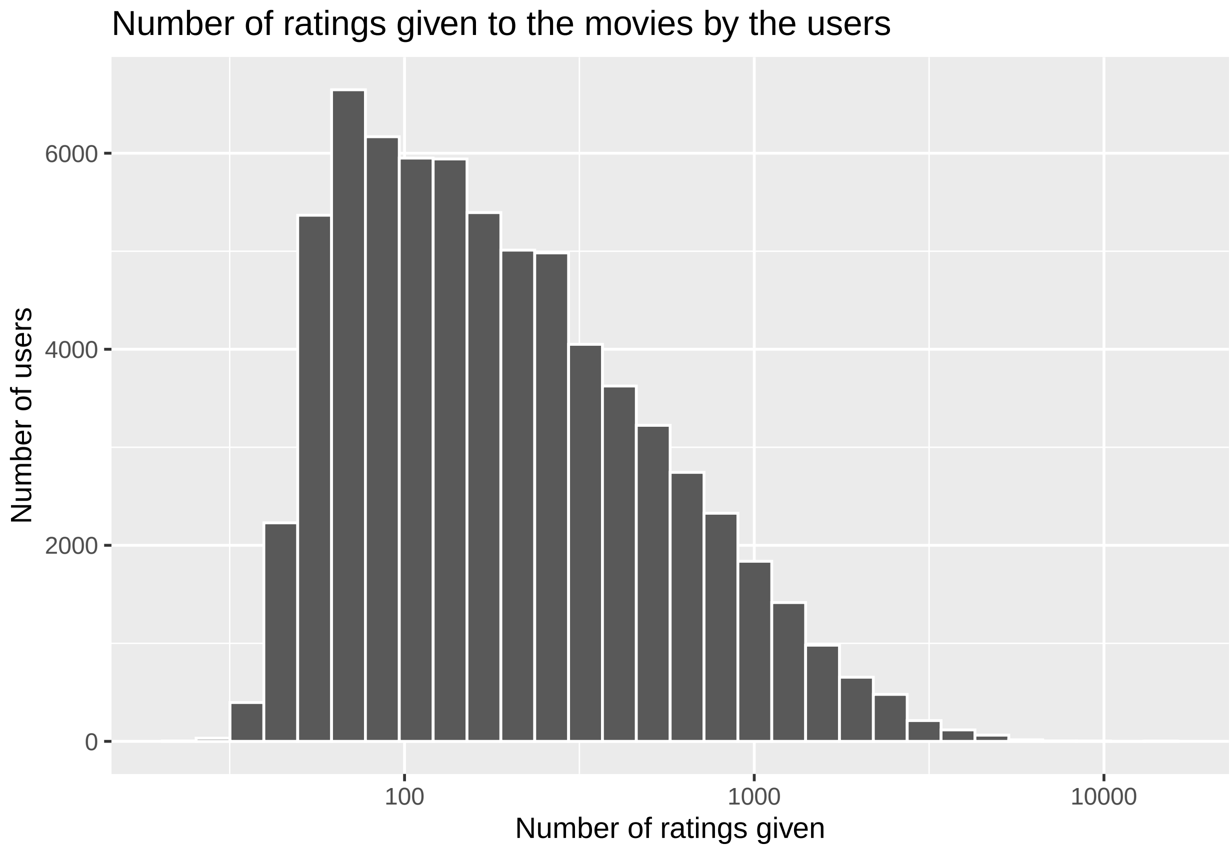
This seems normally distributed (given logarithmic transformation taken on X-axis i.e. for number of ratings) Lets have a leav of faith and hope that this would be of some help in building the model ;)

Rating distribution per user

```
dataset %>% group_by(userId) %>% summarise(count = n()) %>% ggplot(aes(count)) + geom_histogram(color =
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

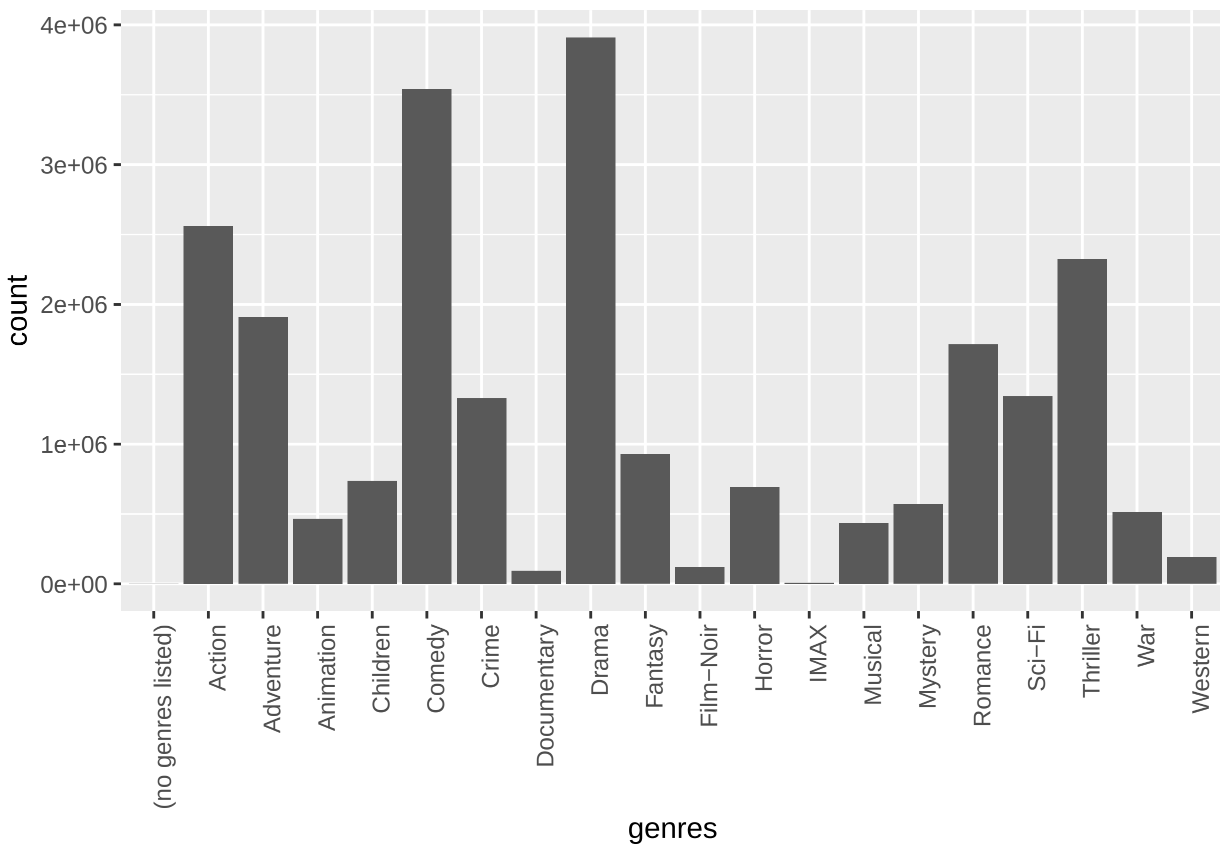


Genre analysis

Overview of rating distribution over different genres

```
dataset %>% group_by(genres) %>% summarise(count = n()) %>% ggplot(aes(genres, count)) + geom_col() + th
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



This gives an overview of rating distribution over different genres.

Preprocessing the data

There's no pre-processing requirements for the models we're gonna build.

Removing unwanted features

Remove 'timestamp', because we are not using it in any of our models and 'title', because 'movieId' serves the same purpose and including it would add redundancy.

```
#removing 'timestamp' and 'title'
dataset <- dataset %>% select(-timestamp, -title)
```

Splitting into training and testing data

We first need to separate our actual training dataset into a training set and a test set so that validation set ramains intact for final expected error evaluation purpose.

```
#splitting the training data into training and testing sets
inTrain <- createDataPartition(y = dataset$rating, times = 1, p = 0.7, list = F)
```

```
training <- dataset[inTrain,]
test <- dataset[-inTrain,]
```

Analysis - Model building and evaluation

Assume that the final rating comprises of average movie rating, movie-specific effect, user-specific effect and the genre popularity effect.

`Rating = Mean + MovieEffect + UserEffect + GenresEffect`

```
#The mean of ratings in training set
raw_mean <- mean(training$rating)

#genres effect
genres_effect <- training %>% group_by(genres) %>% summarise(genreseffect = mean(rating - raw_mean))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#movie effect
movie_effect <- training %>% left_join(genres_effect, by = 'genres') %>% group_by(movieId) %>% summaris
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#user effect
user_effect <- training %>% left_join(genres_effect, by = 'genres') %>% left_join(movie_effect, by = 'mo
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
#Predictor function
predictions <- function(testSet){
  #predicting with the model
  pred <- testSet %>% left_join(movie_effect, by = 'movieId') %>% left_join(user_effect, by = 'userId')
  #adjusting our predictions according to the expected range
  pred[pred < 0.5] <- 0.5
  pred[pred > 5] <- 5
  pred
}
```

#Conclusion

After training the model, test set RMSE and validation set RMSE is calculated as

```r
#test set
predTest <- predictions(test)
RMSE(predTest, test$rating)
```

```
## [1] 0.8623372
```

```r
#validation set
predVal <- predictions(validation)
RMSE(predVal, validation$rating)
```

```
## [1] 0.8684116
```