Movie Recommendation System

Hitesh Gupta

06/06/2019

Objective

The objective of this data science project is to build a machine learning (ML) algorithm that will predict movie rating(s) for a given movie (or a set of movies)

Evaluation Criteria

To evalaute and compare multiple ML approaches, we will use the principal of Root Mean Squared Error (RMSE). RMSE can be defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$$

where $\hat{y}_i$= output predicted by selected approach and y_i= Actual output of the observation. The objective is to decrease RMSE as much as possible to find our optimal algorithm

Note that other evaluation metrics such as sensitivity, specificity, accuracy etc are more apt for cases where number of categorical cases in output are low (or binary in nature). Hence such metrics are of limited use in this case

Data Set

For this project we are using the Movielens data. Movielens is a free movie recommendation website run by University of Minnesota, USA. While the entire dataset is pretty big (27 million ratings), we will be using a small subset with 10 million ratings

The dataset has 6 types of data (columns) for each rating (row)

- **userId**: Unique id for each user who has submitted the rating
- **movieId**: Unique id for the movie
- **rating**: The rating given by the user. It is ordinal in nature and is spans from 1 (worst) to 5(best) with an interval of 0.5
- **timestamp**: Timestamp when the rating was given
- **movieId**: Title of the given movie along with year of release
- **genres**: The genre(s) a movie belongs to. In case movie belongs to multiple genres, the list is separated by pipe characters

Methods & Analysis

Approach

Machine learning algorithms can be classified into 2 types

1. **Supervised Algorithms** Supervised algorithms are used in cases where we already have outcome or output values for our sample data and the machine then tries to learn from a sub-set of this data (training data set) to predict the outcome of another mutually exclusive subset (test data set) or entirely new data set. An example of this could be - Given data for 1 million patients tested for a given medical condition, predicting the probability of a patients testing positive for that condition
2. **Un-supervised Algorithms** These type of algorithms do not have any prior data or prior outputs to learn from. These algorithms try to find a pattern, structure, similarity etc within given data set and then try to classify the given data into different clusters or categories

For this project, we will be using supervised algorithms

Training and Test Sets

We have randomly selected 10% of our data as the training set and stored in *edx* variable. The rest of the data will used for validation and is stored in the *validation* variable.

The script to generate these training and validation sets are provided in the problem itself and are included in setup script of source RMD file

Data Wrangling

It's a set of techniques to process the raw data in usable or tidy format which can then easily be used for further processing. We take the following steps to check the data wrangling measures required on our data (if any)

Converting to tidy format - data frame

The class of data is already "data frame" and data is alredy present in tidy format

Checking for empty (NA) values

Each column in the data was checked with **is.na()** and no empty values were found in each of the columns

Checking for Null

Each column in the data was checked with **is.null()** and no empty values were found in each of the columns

Checking for NaN in movieId, userId, Rating and Timestamp

Each column in the data was checked with **is.nan()** and no empty values were found in each of the columns

Checking of invalid values

Structure of the data table

The structure of the data table was analyzed using **str()** and the data types used for each column were found as expected. No factor variables found

Rating

The count of unique ratings and **there are 10 distinct values found as expected**. No invalid values were found

Timestamp

The timestamp given in the data is in UNIX format and has to be converted in human readable format so that one can use **lubridate package** and extract corresponding values in it

```
edx<- edx %>% mutate(dateTime=as.Date(as.POSIXct(timestamp, origin="1970-01-01")))
```

Data Exploration & Insights

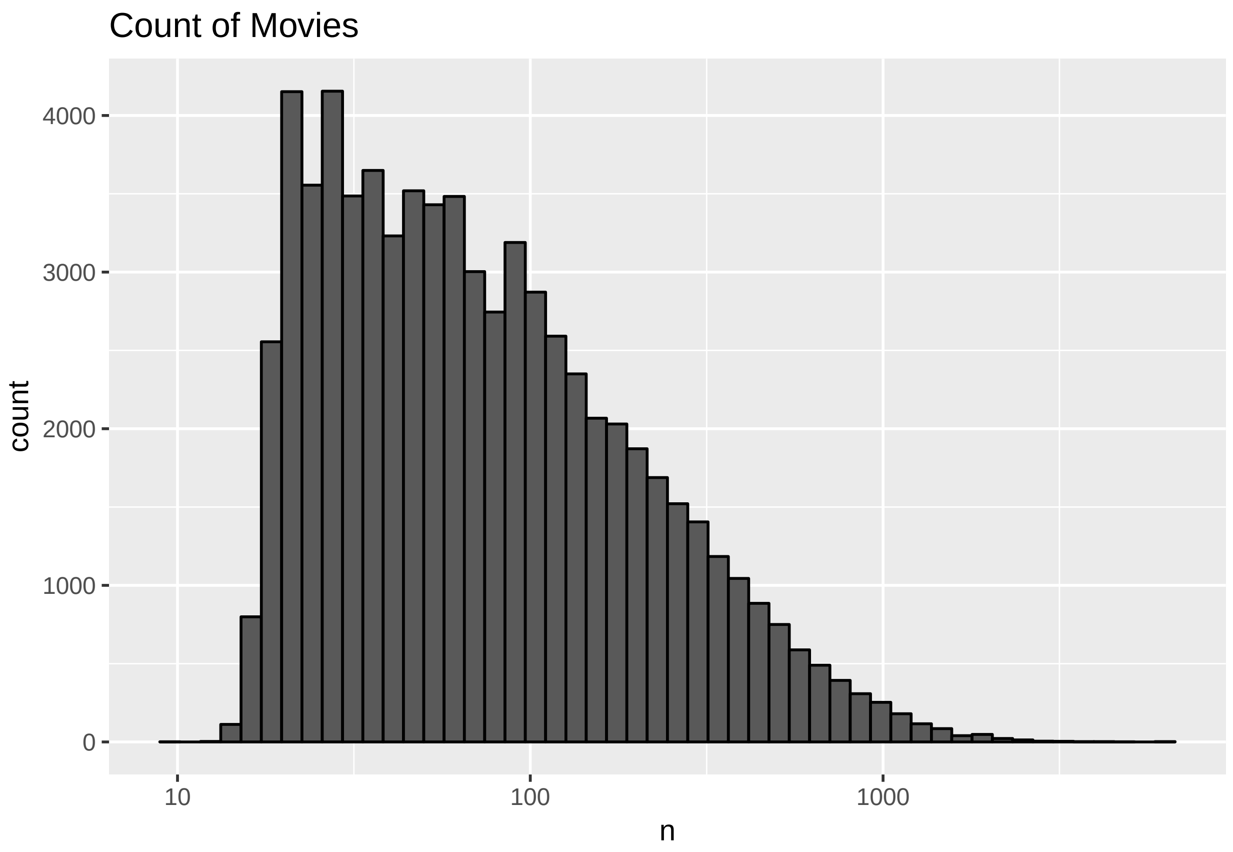
Exploring the training data set (edx) gives us some key insights.

- There are 9000055 entries in the dataset
- 69878 different users have given ratings to 10677 unique movies (they have one to many and many to one relationship).

Effect of movie's content and popularity

Plotting the frequency distribution for movies tells us that some movies get much higher number of ratings than others - which in turn effects the final ratings (higher denominator can lead to lower ratings). The movies with very few ratings can also be outliers.

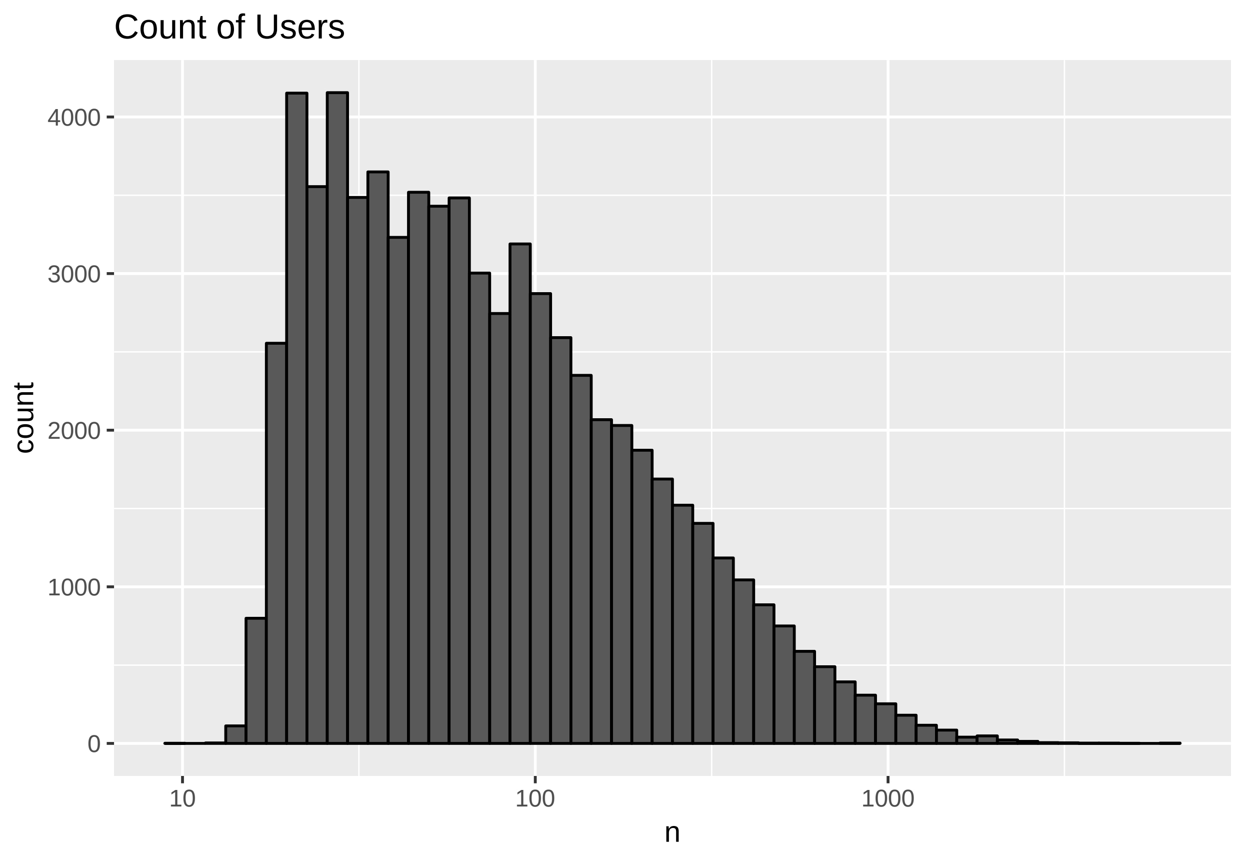
```
edx %>% count(userId) %>% ggplot(aes(n)) + geom_histogram(bins = 50, color = "black") + scale_x_log10()
```



Effect of user behaviour

Plotting the frequency distribution for users tells us that some users give much higher number of ratings than others - this can lead to bias of user behaviour

```
edx %>% count(userId) %>% ggplot(aes(n)) + geom_histogram(bins = 50, color = "black") + scale_x_log10()
```

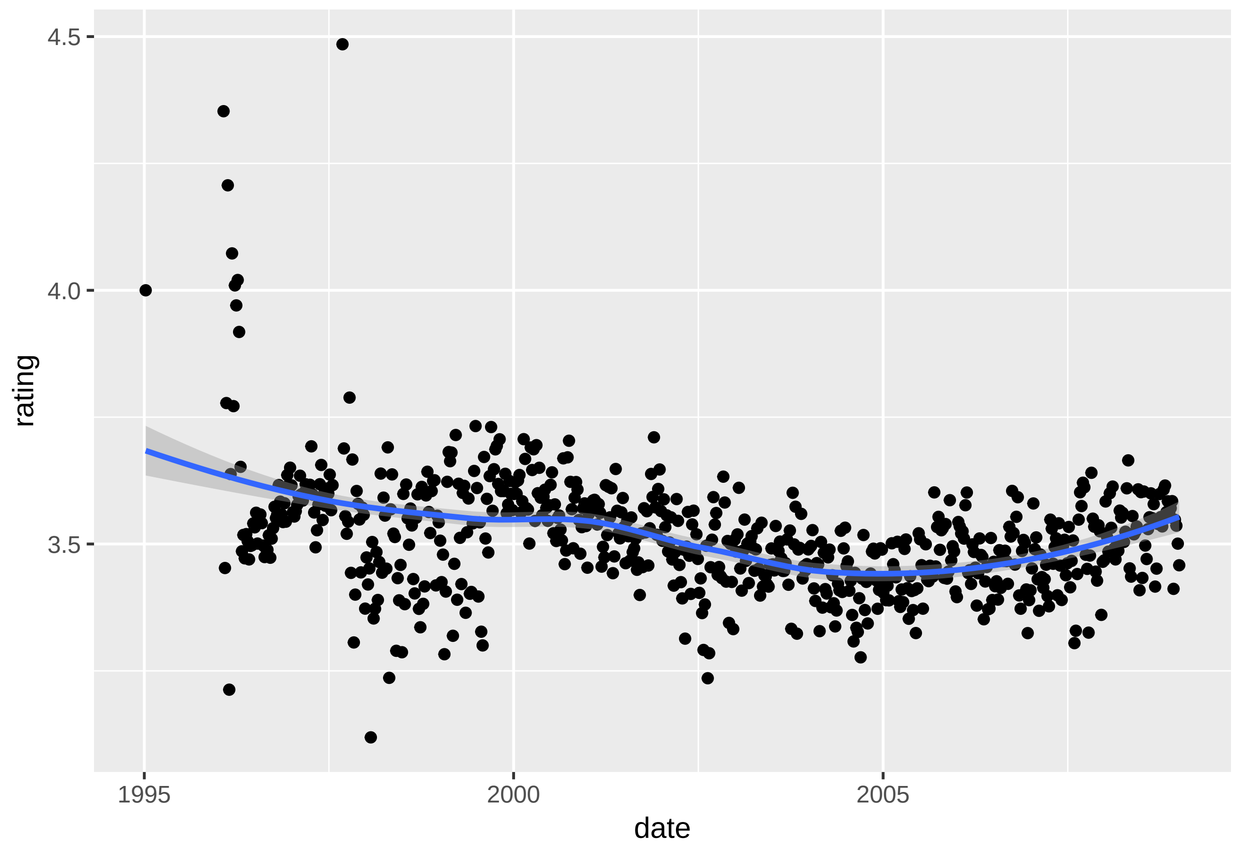


Time Dependency

To check if time of release has any observable impact on rating, we plot the week vs rating graph and try to see any observable pattern when plotted by using the code below

```
edx %>% mutate(date = round_date(dateTime, unit = "week")) %>%
    group_by(date) %>%
    summarize(rating = mean(rating)) %>%
    ggplot(aes(date, rating)) +
    geom_point() +
    geom_smooth()
```

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



One can see from the graph above that there is no strong correlation between the two. Hence we can ignore the time of release as a factor from our analysis

Other Observations

- Not all users have given ratings to all the movies. So if we try to think of a matrix with users on rows and movies on columns and the corresponding rating as a value, we will have lot of empty cells (sparsely populated).

Modelling Approach

Creating a reusable RMSE function

As the heading suggests, we will create a reusable function to compute RMSE values for our various models

```
RMSE <- function(actual, predicted){
  sqrt(mean((actual - predicted)^2))
}
```

Model 1: Using Average of Testing data set

The most basic model, which we can use as a start is to assume that all predicted output values $Y_{u,i}$ for user u and movie i are equal to mean rating of training data set. This model can be described by equation below, where μ is the true mean and ϵ is the sampled errors with center at 0

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

The RMSE value can then be found using the R code below

```r
rmse_avg<- RMSE(validation$rating,mean(edx$rating))
rmse_table <- data_frame(method = "Training Set average",
                         RMSE = rmse_avg)
paste("RMSE_Average =",rmse_avg,sep = " ")
```

```
## [1] "RMSE_Average = 1.06120181029262"
```

Model 2: Factor in movie bias

We know from our data visualization excercise done above that some movies are usually rated higher than other. These could be classic all time favorites (world wide blockbusters) or could be cult movies rated by very few individuals. This movie bias can be explained by below equation. Again μ is the true mean and ϵ is the sampled errors with center at 0

$$Y_{u,i} = b_i + \mu + \epsilon_{u,i}$$

or in other words b_i for each movie i can be written as

$$b_i = Y_{u,i} - \mu - \epsilon_{u,i}$$

The value of b_i can be computed using the below R code

```r
movie_bi <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mean(edx$rating)))
```

The RMSE value can then be found using the R code below

```r
predicted_ratings <- mean(edx$rating) + validation %>%
  left_join(movie_bi, by='movieId') %>% pull(b_i)

rmse_bi <- RMSE(validation$rating, predicted_ratings)
rmse_table <- bind_rows(rmse_table, data_frame(method="Movie Bias Model",
                                    RMSE = rmse_bi))
paste("RMSE Movie Bias =",rmse_bi,sep = " ")
```

```
## [1] "RMSE Movie Bias = 0.943908662806309"
```

Model 3: Factor in user bias

We know from our data visualization excercise done above that some user are give a large number of ratings and some users generally give higher average ratings than others. The other users are either relatively inactive or give low ratings to all the movies. This user bias can be explained by below equation.

$$Y_{u,i} = b_i + b_u + \mu + \epsilon_{u,i}$$

or in other words b_u for each user u can be written as

$$b_u = Y_{u,i} - b_i - \mu - \epsilon_{u,i}$$

The value of b_u can be computed using the below R code

```
user_bu <- edx %>% left_join(movie_bi, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mean(edx$rating) - b_i))
```

The RMSE value can then be found using the R code below

```
predicted_ratings <- validation %>%
  left_join(movie_bi, by='movieId') %>%
  left_join(user_bu, by='userId') %>%
  mutate(pred = mean(edx$rating) + b_i + b_u) %>%
  pull(pred)
rmse_bu <- RMSE(validation$rating, predicted_ratings)
rmse_table <- bind_rows(rmse_table, data_frame(method="Movie + User Bias Model", RMSE = rmse_bu))
paste("RMSE Movie + User  Bias =",rmse_bu,sep = " ")
```

```
## [1] "RMSE Movie + User  Bias = 0.865348824577316"
```

Model 4: Factor in low rating count bias

One of the factors that has a high chance of bringing a bias is the number of ratings. There might be some movies which are rated very few times (or in worst case just 1 time) that can add noise to our models. To prove this we can check for the movies with highest value of b_i as per Model 3, and also find their number of ratings

```
# Create small dataset for just movie titles
titles <- edx %>% select(movieId, title) %>% distinct()
# Find 20 best movies as per b_i
edx %>% count(movieId) %>% left_join(movie_bi) %>%
  left_join(titles, by="movieId") %>% arrange(desc(b_i)) %>%
  select(title, b_i, n) %>% slice(1:20)
```

```
## Joining, by = "movieId"
```

```
## # A tibble: 20 x 3
##    title                                                      b_i     n
##    <chr>                                                    <dbl> <int>
##  1 Hellhounds on My Trail (1999)                             1.49     1
##  2 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                       1.49     2
##  3 Shadows of Forgotten Ancestors (1964)                     1.49     1
##  4 Fighting Elegy (Kenka erejii) (1966)                      1.49     1
##  5 Sun Alley (Sonnenallee) (1999)                            1.49     1
##  6 Blue Light, The (Das Blaue Licht) (1932)                  1.49     1
##  7 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko~ 1.24     4
##  8 Human Condition II, The (Ningen no joken II) (1959)       1.24     4
##  9 Human Condition III, The (Ningen no joken III) (1961)     1.24     4
## 10 Constantine's Sword (2007)                                     1.24      2
## 11 More (1998)                                                    1.20      7
## 12 I'm Starting From Three (Ricomincio da Tre) (1981)             1.15      3
## 13 Class, The (Entre les Murs) (2008)                             1.15      3
## 14 Mickey (2003)                                                  0.988     1
## 15 Demon Lover Diary (1980)                                       0.988     1
## 16 Life of Oharu, The (Saikaku ichidai onna) (1952)               0.988     3
## 17 Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (19~ 0.988     1
## 18 Testament of Orpheus, The (Testament d'OrphÃ©e) (1960)        0.988     1
## 19 Power of Nightmares: The Rise of the Politics of Fear, The ~ 0.988     4
## 20 Kansas City Confidential (1952)                                0.988     1
```

We see that name of the movies are very unpopular and not many people have rated these movies (all of them are single digits).

To counter this noise, we would use the concept of **Regularization** that penalizes observations with large variations but low frequency of occurrence. To counter *count of ratings* we divide the RMSE by number of ratings and to counter *large variation* of b_i we add the parameter b_i^2

The new RMSE can be explained by below equation. Here λ is a constant, and n_i is the number or ratings for movie i

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i)^2+\lambda\sum_i b_i^2$$

or in other words the update b_i for each movie i can be written as

$$b_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu})$$

We currenty do not know the most optimum value of λ. Hence we can cross validate it by creating a map with the values for λ from 1 to 10 with intervals of 1. The updated value of b_u can be computed for each value of λ using the below R code

```r
lambdas <- seq(1, 10, 1)

rmses <- sapply(lambdas, function(l){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mean(edx$rating))/(n()+l))

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

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mean(edx$rating) + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, validation$rating))
})
rmse_table <- bind_rows(rmse_table, data_frame(method="Regularized Movie + User Bias Model", RMSE = min
paste("RMSE Low Number of Rating  Bias =",min(rmses),sep = " ")
```

```
## [1] "RMSE Low Number of Rating  Bias = 0.864817746466669"
```

The min value of RMSE is found using $\lambda = 5$

Results

The results of our various models can be seen in the table below

```
rmse_table
```

```
## # A tibble: 4 x 2
##   method                              RMSE
##   <chr>                              <dbl>
## 1 Training Set average               1.06
## 2 Movie Bias Model                   0.944
## 3 Movie + User Bias Model            0.865
## 4 Regularized Movie + User Bias Model 0.865
```

Further Studies and Areas of Improvement

Factor in Time Elapsed since Movie Release

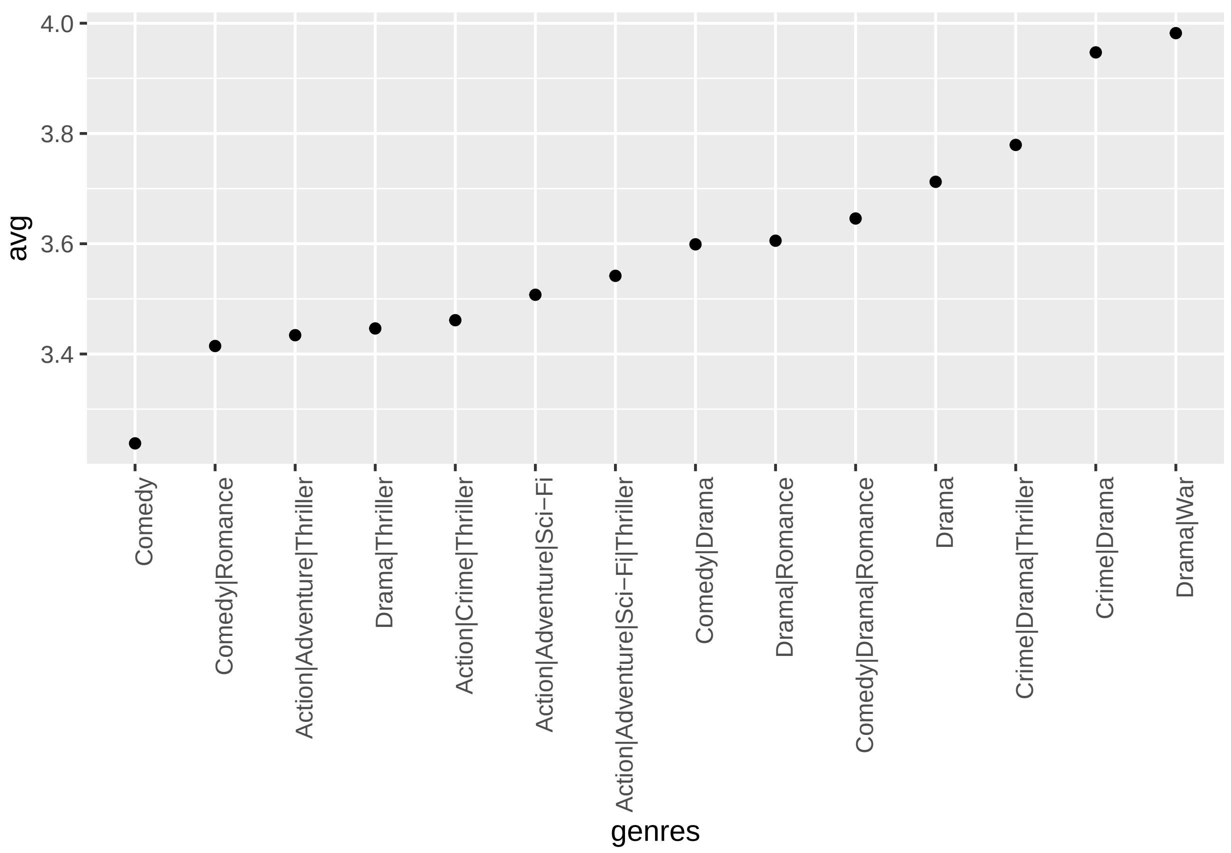
Popular movies lose their popularity with time, while it can be opposite for cult movies. Also the number of ratings for new movies will be lesser than old movies . Hence the time elapsed since the movie has been released can also be a factor that can impact the predicted rating.

We saw in our data inferencing excercise that there is a weak correlation which can further reduce our RMSE score slightly

Impact of Genre

This model does not take into account the effect of Genres. Some genres like Action & Romantic are generally rated much higher than horror movies. This effect in genre can be visualised using the code below

```
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating)) %>%
  filter(n >= 100000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg)) +
  geom_point() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Thus, one can further improve the model by factoring in a variable for various genre categories

Correlation between similar users and movies

This model does not factor in the fact that similar movies have similar rating patters (3 parts of movie trilogy will be rated similar), or movies from some popular actor will be rated similar. Similarly the models in the project do not factor that groups of users will show similar rating patterns. Some users will always rate action movies higher than horror movies.

Such factors can be incorporated in the project by using concepts like **Factorization**

Conclusion

In this paper, we tried to build a movie recommendation system using 10 mn version of MovieLens database.

1. We first analyzed the data and found no significant issues related to empty or invalid values in data set
2. We then did some basic inferencing and data visualization techniques to make the following inferences

i) Some movies are rated more times than others. Also movies rated more tend to have higher rating
ii) Similar behaviour was seen for users. Some users rate more movies than other users. Also their average rating tends to be on extreme and not uniformly distributed
iii) We also found that rating is weakly correlated with time of release. This factor was not used in any of our models

We then started building our model step by step iteratively and starting adding the required factors

1. Base Model (Model 1): Predicted rating is same as average rating of sample size
2. Model 2: Factor in the movie popularity and content bias
3. Model 3: Factor in the user behaviour and choice bias
4. Model 4: Factor in low rating count bias. We tested our model for various value of λ and found the least value of RMSE occurs **0.8648177** at $\lambda = 5$

Overall Model 4 performs best of all the evaulated models

References

1. Course Notes from Edx course HarvardX - PH125.8x, Data Science: Machine Learning [https://www.edx.org/course/data-science-machine-learning]
2. Introduction to Machine Learning by Rafael A. Irizarry [https://rafalab.github.io/dsbook/introduction-to-machine-learning.html]
3. Machine Learning online course by MIT at coursera.org [https://www.coursera.org/learn/machine-learning]