MovieLens

Zubair Khan

15-Aug-2020

Contents

Introduction

This report is part of HarvardX's Capstone project for PH125.9x Data Science course. The objective of this project is to predict the rating of unseen data using the operational understanding of R, its toolset and acquired skills in data analytics and data mining. This report will not just propose a recommendation system but also investigate potential limitations in the dataset and hidden patterns for future predictions.

The MovieLens 10M dataset is provided by GroupLens, a research lab at the University of Minnesota. The dataset is built from feedback on a dynamic user group. Through this report we will attempt to predict rating for unseen data. The initial data model approach that was used is to identify hidden patterns by exploiting the genre variable. It was later modified to condition a prediction that yields a RMSE value of lower than 0.86 as per course requirement.

Inorder to achieve our objectives the project will flow in the following order:

- Acquisition of the Dataset
- Data preparation by splitting into training and test dataset
- Verifying data consistency
- Loading required libraries
- Filtering data
- Data conversion to readable format
- Variable selection using Data exploration & Analysis
- Run ML on training dataset and verify result

- Run ML on test dataset and export CSV generated

Method

Dataset acquisition

The project requirements dictate that the data set is to be downloaded from the following path.

Path = "http://files.grouplens.org/datasets/movielens/ml-10m.zip"

The dataset is then split into the following 2 parts:

- Imported Dataset:
  - edx: contains 90% of the Movielens dataset
  - validation: contains 10% of Movielens dataset

The code required to split the relevant datasets are already provided. Refer to ***Data Preparation*** section below.

Data Preparation

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

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")

# 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")

# if using R 3.6 or earlier
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
# if using R 4.0 or later
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(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)
# 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)

#############################################################################
```

Once the required data is loaded into the environment, we will begin our investigation process.

Data Verification

In this section we will be examining the dataset to verify data structure and consistency. This will be done by visually examining the a sample of the dataset, generating a summary of it and examining the structural output by using the **str** function.

The EDX dataset contains 9000061 and 6 attributes. There are a total of 69878 **Users** accounts and 10677 **Movies** titles being represented in the dataframe. The titles column hold's movie release details. The genre column has genre's piped alongside various genre categories.

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	231	5	838983392	Dumb & Dumber (1994)	Comedy
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi

```
## [1] "Table  1: Top 5 rows of dataset"
```

. .

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000061     Length:9000061
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
```

```
##
##
```

```
## [1] "Table  2: Summary of dataset"
```

```
. .
```

```
## Classes 'data.table' and 'data.frame':   9000061 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 838983392 838984474 838983653 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-Fi|Thriller"
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
## [1] "Table  3: Structure of dataset"
```

Libraries

The following libraries will be utilized through out the project.

```
#Working with edx library
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(anytime)) install.packages("anytime")
if(!require(sjmisc)) install.packages("sjmisc")
if(!require(scales)) install.packages("scales")
if(!require(dplyr)) install.packages("dplyr")
if(!require(tinytex)) install.packages("tinytex")

library(ggplot2)
library(anytime)
library(sjmisc)
library(scales)
library(dplyr)
library(ggplot2)
library(formattable)
library(tinytex)
```

Filtering Dataset

Movies have been categorized with a collection of genres that are separated by the "|" delimiter. We will be splitting the genres and saving the result in an attribute. Then using the unique function, identify the number of unique genres contained in the dataset.

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

```
## [1] "Table  4: Genres"
```

Looking at the **genres** variable, we can identify that there are a total of 20 genres with movie titles holding a maximum of 8. Splitting the genre column and building a matrix that will hold Boolean value if movie title belongs to that genre. This Data matrix will later assist us in further data categorization and explorations.

The below table represents the genre distribution across various movie titles. Higher number represents movies created within the dataset time frame that hold specified genre.

genres	count
Drama	3909401
Comedy	3541284
Action	2560649
Thriller	2325349
Adventure	1908692
Romance	1712232
Sci-Fi	1341750
Crime	1326917
Fantasy	925624
Children	737851
Horror	691407
Mystery	567865
War	511330
Animation	467220
Musical	432960
Western	189234
Film-Noir	118394
Documentary	93252
IMAX	8190
(no genres listed)	6

```
## [1] "Table  5: Genre count in descending order"
```

Further data exploration is required. To create the data matrix mention in the **Filtering & Analyzing Dataset** section, we will create a new dataset and migrate only relevant columns in it. Then based on genre count as calculated above, we will add relevant columns in the same sequence.

Data Conversion

The **timestamp** variable represents the time frame the rating was captured. In order to get the movie release time frame, we need to extract the release year from the right most side of the **title** variable with the following code. We will only be extracting the year value from the timestamp data.

```
##  [1] "1992" "1995" "1994" "1993" "1990" "1991" "1937" "1970" "1977" "1996"
## [11] "1972" "1971" "1989" "1974" "1967" "1997" "1985" "2000" "1982" "2001"
## [21] "2002" "2003" "2004" "2005" "1976" "1958" "1942" "1939" "1941" "1950"
## [31] "1951" "1979" "1955" "1962" "1984" "1980" "1988" "1975" "1987" "1981"
## [41] "1969" "1998" "1999" "1986" "1952" "1959" "1946" "1944" "1940" "1954"
## [51] "1983" "1949" "1960" "1973" "1948" "1933" "1931" "1963" "1922" "1943"
## [61] "1956" "1957" "1953" "1965" "1978" "1964" "1968" "1966" "1961" "1935"
## [71] "1938" "2006" "2007" "1932" "2008" "1934" "1945" "1936" "1947" "1927"
## [81] "1925" "1930" "1929" "1923" "1928" "1921" "1926" "1915" "1924" "1916"
## [91] "1917" "1918" "1920" "1919"
```

```
## [1] "Table  6: List of all unique moive release years "
```

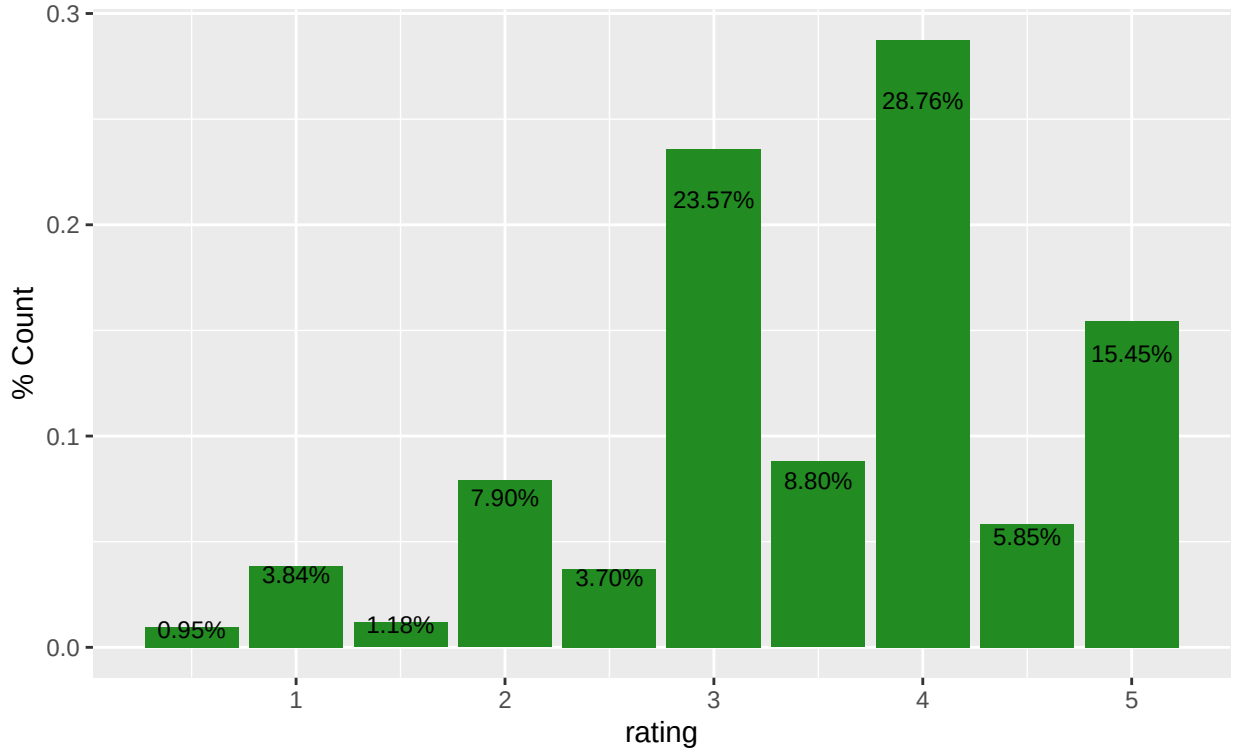
The final data set contains a better categorization of genre variable. Although this has added more columns in our dataset, it will allow us to perform better filtering and data exploration. I believe exploring the data matrix will give us a richer perspective of user behavior in choosing movie titles. Unfortunately, I am at a press of time for submission and need to do a bit more research on using ML with matrices. We will proceed with RMSE calculations without it.

userId	movieId	rating	Drama	Comedy	Action	Thriller	Year	RelYear
1	122	5	FALSE	TRUE	FALSE	FALSE	1996	1992
1	185	5	FALSE	FALSE	TRUE	TRUE	1996	1995
1	231	5	FALSE	TRUE	FALSE	FALSE	1996	1994
1	292	5	TRUE	FALSE	TRUE	TRUE	1996	1995
1	316	5	FALSE	FALSE	TRUE	FALSE	1996	1994

```
## [1] "Table  7: Visual representaion of final dataset"
```

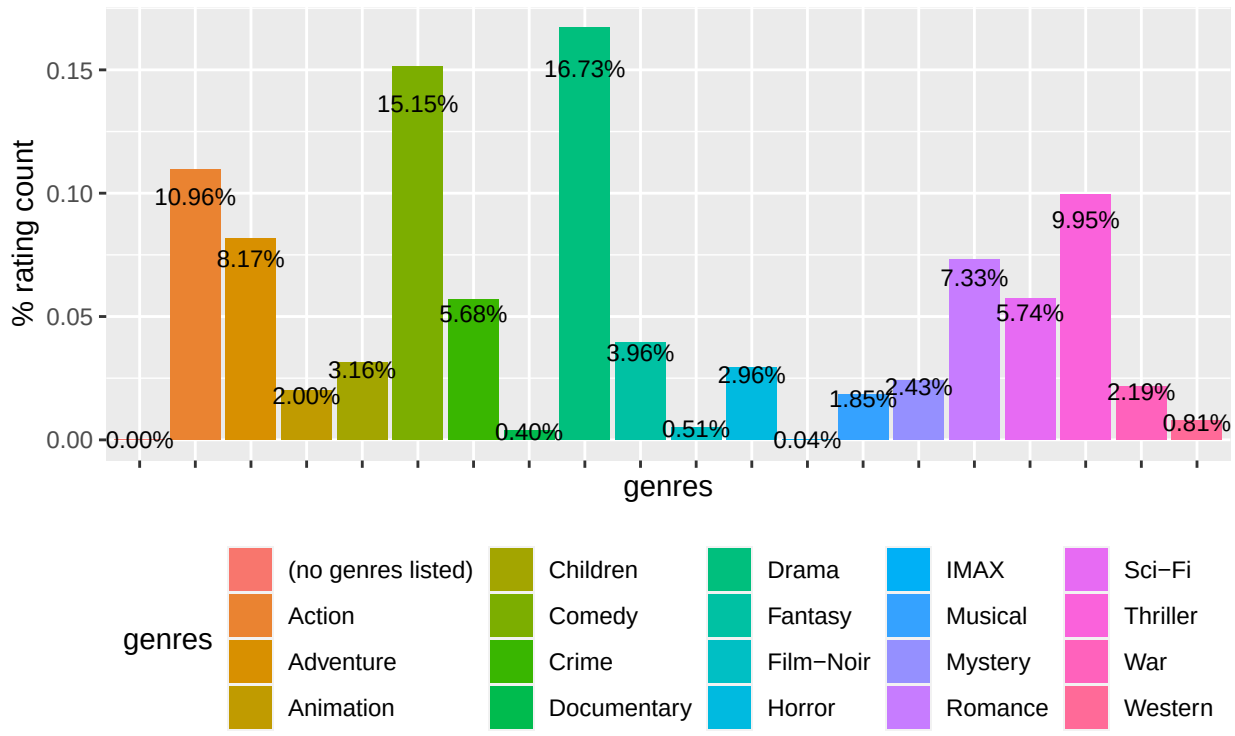
Data Exploration & Analytics

Let's examine the distribution of ratings to get a clear understanding of how users have rated titles.




```
## [1] "Figure  1: Total rating count"
```

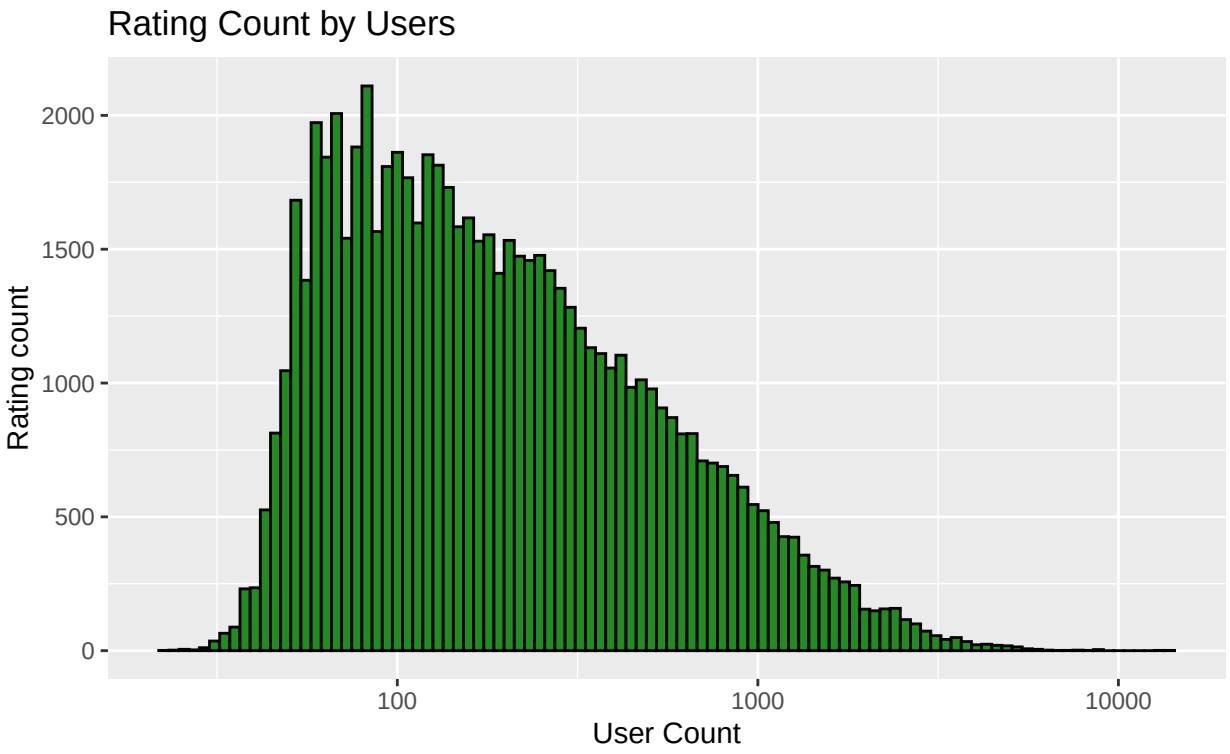
Although the above chart depicts that majority titles have been rated at an average of 4, It is however not clear that if users have given all titles equal rating opportunity. Rating count can be biased based on popularity of title. Unfamiliar titles may not even be rated or might not have adequate volume to create impact.




```
## [1] "Figure  2: Total genre count"
```

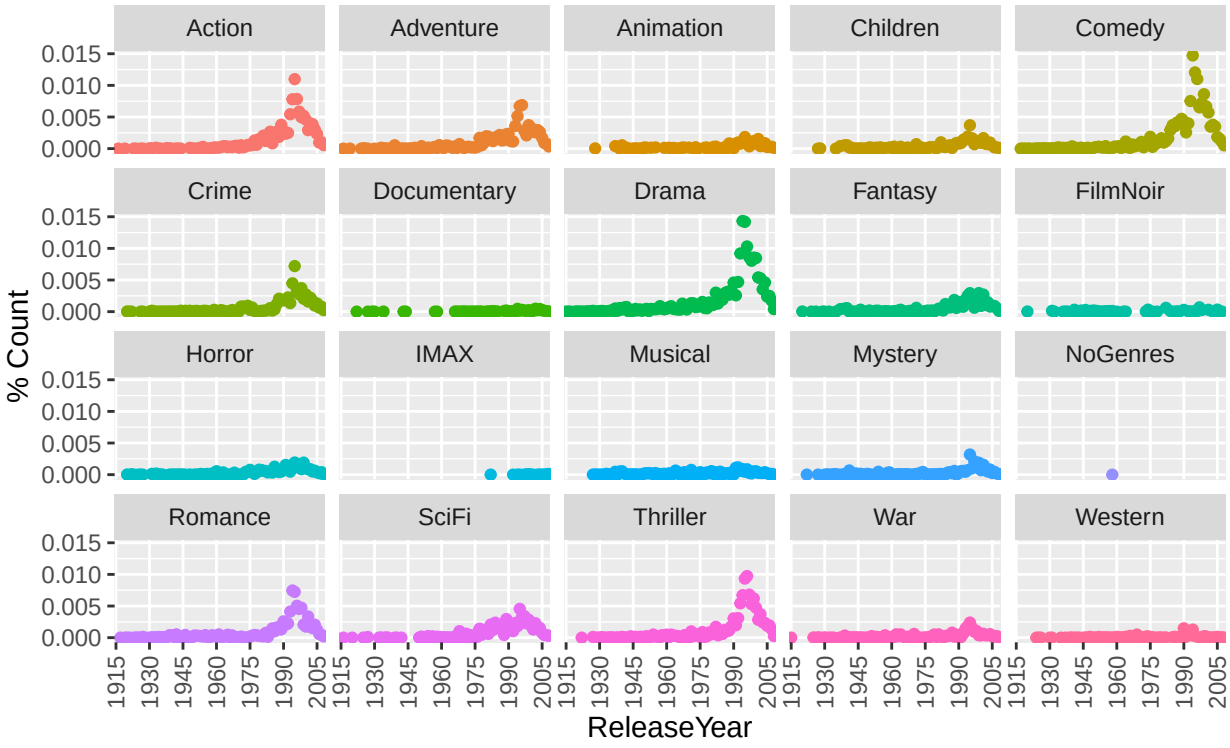
This can be confirmed by reviewing the genre distribution as shown above. Only two genre's have crossed the 80% distribution. The hypothesis is that certain genre's may not receive the adequate exposure required as the genres are fairly new compared to the dataset time frame that they are being assessed on.

Let's plot the user distribution to get an perspective of how users are distributed based on their amount of rating feedback.




```
## [1] "Figure  3: Rating distribution"
```

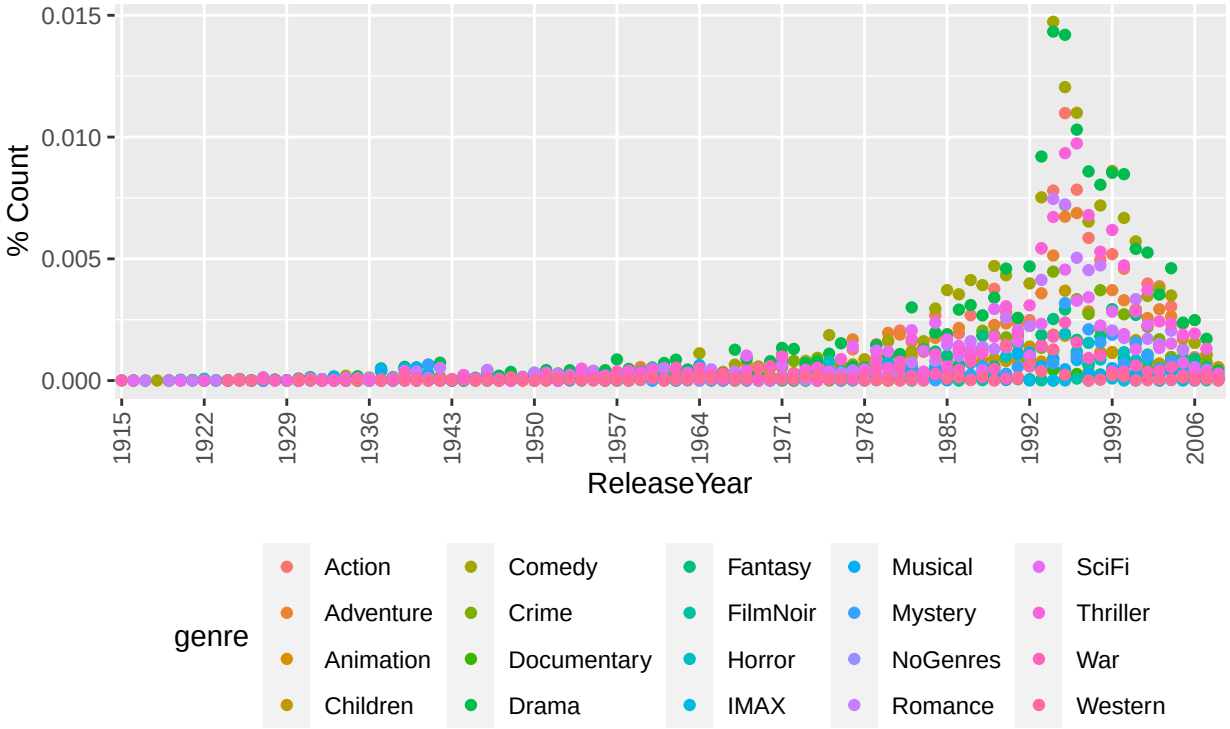
The above chart indicated that user rating is not evenly distributed. Rating count has had substantial growth from 50 and then a gradual decrease post 100.



```
## [1] "Figure  4: Genre count by release year"
```

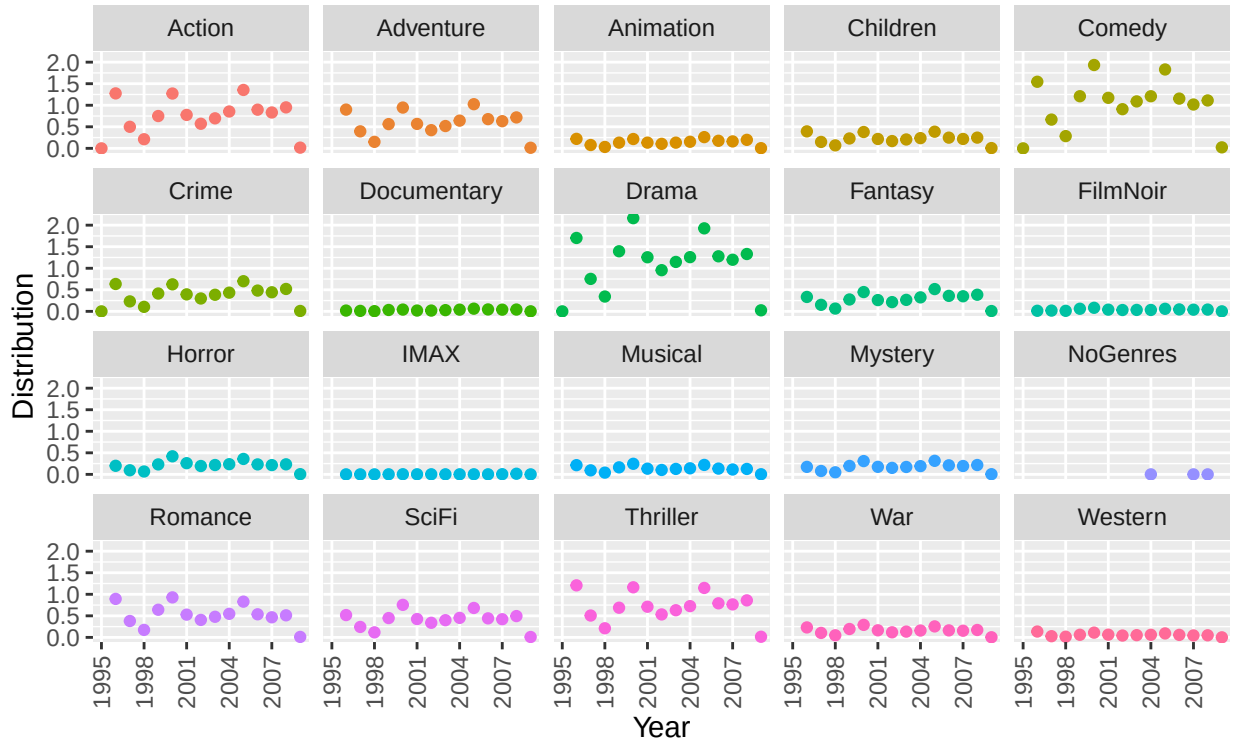
The graph above represents ratings captured by user on all genres represented separately. Based on the collage, we can identify that the certain genres do not follow the general pattern. There is an introduction of new genre's that are not available earlier in the timeline and there has been an increase in movie rating in the late 90's and then a gradual decrease moving toward 1999 and onwards. The gradual decrease could be due to lack of opportunity for never movie titles to be rated.

Throughout the timeline **Action**, **Adventure**, **Comedy**, **Drama**, ***Romance** and **Thriller** have sustained an overall strong presence. It can be stipulated that movies produced with either of the genres or combination of genres will sustain a strong user attention.



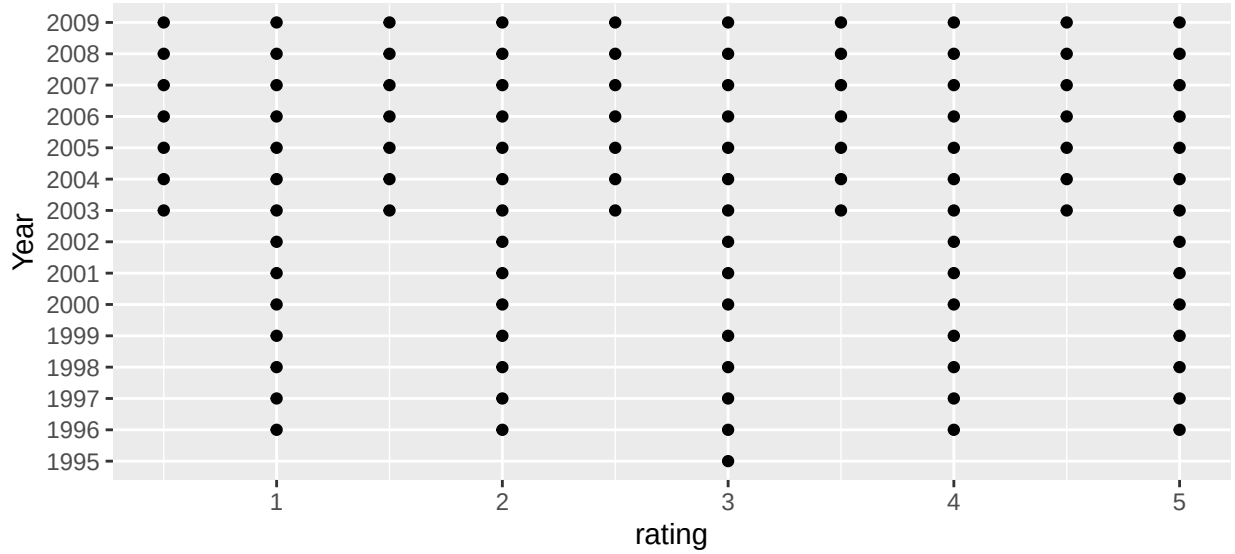

```
## [1] "Figure  5: Consolidated genre count by release year"
```

Reviewing the above chart we can stipulate that there is a substantial increase in ratings in 1990's. This is later identified to be corelated with the year the rating system was introduced.




```
## [1] "Figure  6: Consolidated genre count by year rating was captured"
```

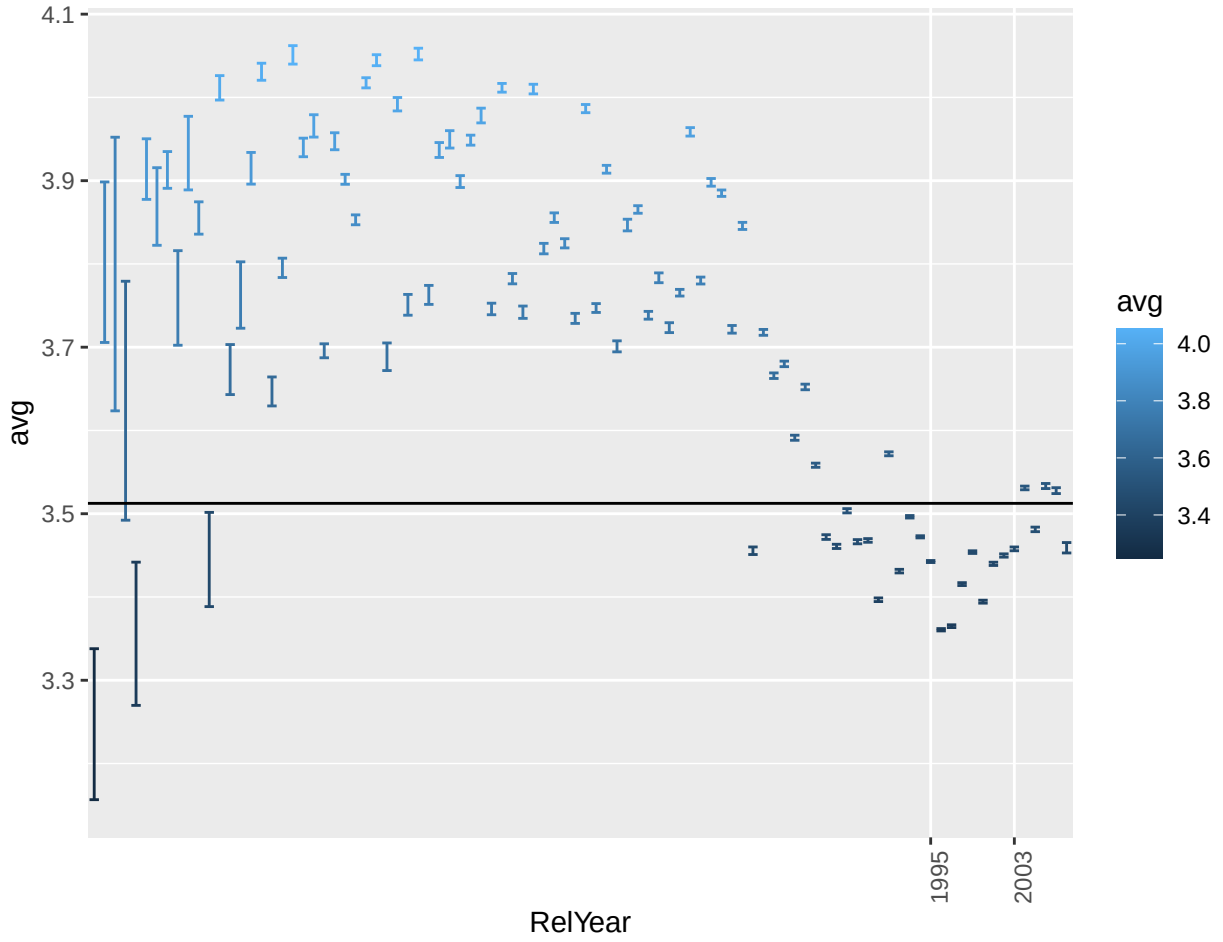
The above genre collage indicates that the initial ratings started in 1995 and hence forth. We can identify that the **Action**, **Comedy** , **Drama** & **thriller** genres has the most versatile user feedback. One other aspect to notice is that genres captured in the early 90's have had a sharper variance while same genres had a smoother rating capture after 2000.




```
## [1] "Figure  7: Distribution of year ratings were captured"
```

Looking at the relationship between rating and the year it was captured we notice that the half ratings were absent prior to 2003. The dataset is not coherent in regard to the rating variable. This can be corrected by rounding up the half variances. As the dataset has a combination of two unevenly-sized distribution, we will

continue as is and exclude the rating variable from our RMSE calculation.




```
## [1] "Figure  8: Average rating by release year"
```

Looking at the above distribution we conclude that older movie titles have had a better rating as compared to newer titles. This can be contributed to the fact that older movies has less count but better rating and newer movies have tighter variability and are closer to the mean. From the data represented in **Release Year** chart we can conclude this is directly related to amount of ratings captured.

In our correlation plot above, we can observe that there is a positive correlation between rating and release year. Movies being rated shortly after they are released may have the volume but lack the diverse rating spectrum. Similar behavior can be seen in music title where major new releases general hit the top 5 rating.

Machine learning Algorithm

As the dataset we are working with is already categorical and labeled, my choice of machine learning algorithm was **KMeans**. But due to high computational requirements which I found out in the end, I opted to change the ML algorithm that was lighter in resources. The decision was based on resources available to the evaluators, the course requirement of submitting the result with RMSE and more specifically not modifying the validating dataset, after a bit of searching I have opted to use the PLS ML algorithm. The rationale behind PLS is to approximate the observed data by balancing the conflicts between the exactness to original data and the roughness of fitting data. This on top of searching for a lighter performing algorithm that would not take a long time to complete.

Findings

One key factor to keep in mind is that the course appoints RMSE grading based on a specific structure which is tied to the **UserID** & **MovieID** columns. As per my findings which will be covered later on, if **RMSE** is calculated without a combination of these two columns the result will not be lower than ***0.9000***.

#Results

As stipulated in the Method section I will be using the PLS ML algorithm to finetune my RMSE results. My system configurations are as follows: + CPU: Core i5 - 2.4Ghz + RAM: 16GB

Goal

While examining the **Dataset** we observer that the **Genre** variable is a composite of multiple genre categories. Prior to splitting the dataframe we need to consider what is required of it. We can look at splitting it into a new line entry for each category and then using that as our baseline category and proceed with identifying patterns. I have opted to construct a data matrix from the genre section and then populate it with Boolean values if it matches a certain category

One thing to note is that, Movies that just have one genre only should be considered as anomalies or errors and should be excluded from our final dataset as in real world scenario they are a combination of genres. We will proceed as it is for now.

The sequence is set from 0 to 3 with 0.25 increments.

```
#Lambda is set to the specified sequence after repeated testes yealded no benefit to changes in it.
lambda <- seq(0, 3, 0.25)

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

My initial idea was to run the below function on the Release year, UserID and Genre variables. The outcome did not yield a RMSE value lower than 9000. In fact, multiple combinations were tested with no improvements. The movie and user attributes therefore have the highest impact on our predictions. Either increase or decrease in these variables will affect the movie rating in lower or higher but will remain close to the overall mean. The user has the tendency to maintain it within that proximity.

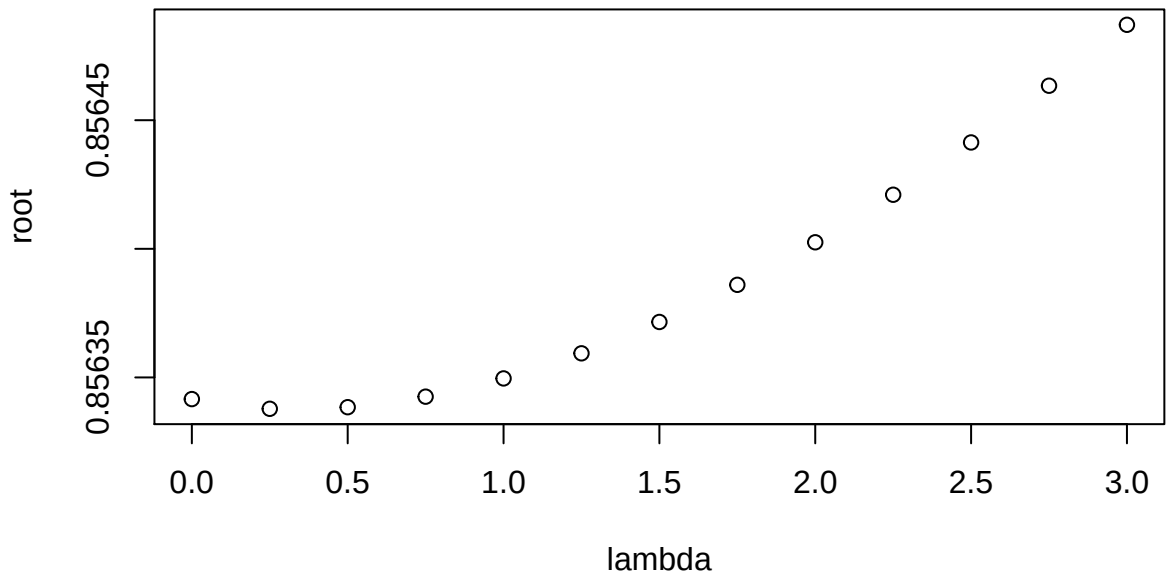
```
root <- sapply(lambda, function(l){
  mu <- mean(edx_final$rating)
  AdjustG <- edx_final %>% group_by(movieId) %>% summarise(AdjustG = sum(rating - mu)/(n()+l))

  AdjustU <- edx_final %>% left_join(AdjustG, by="movieId") %>% group_by(userId) %>%
    summarise(AdjustU = sum(rating - AdjustG - mu)/(n()+l))

  AdjustM <- edx_final %>% left_join(AdjustG, by="movieId") %>%
    left_join(AdjustU, by="userId") %>% group_by(RelYear) %>%
    summarise(AdjustM = sum(rating - AdjustG -AdjustU - mu)/(n()+l))

  prediction <- edx_final %>% left_join(AdjustG, by = "movieId") %>% left_join(AdjustU, by = "userId") %
    left_join(AdjustM, by = "RelYear") %>%
    mutate(predict = AdjustG + AdjustU +AdjustM + mu) %>% .$predict
  return(RMSE(prediction, edx_final$rating))
})


plot(lambda, root)
```




```
fig_nums("Test", "Trainging for optimal Lambda value")
```

```
## [1] "Figure  9: Trainging for optimal Lambda value"
```

```
min(root)
```

```
## [1] 0.8563378
```

```
lambda[which.min(root)]
```

```
## [1] 0.25
```

Test dataset verification

As the RSME results fall within the required parameters, we will utilize the selected ML algorithm on our validation dataset. The result will be extracted into a CSV file for review.

```
lambda <- lambda[which.min(root)]

#we need to modify the validation set to include Release year

#dim(validation)
#head(validation)

#Extract release year of Title
RelYear <- substr(validation$title, nchar(validation$title)-4,nchar(validation$title)-1)
validation$RelYear <- as.factor(RelYear)

Fdate <- anydate(validation$timestamp)
#Creating a column that will contain Year
validation$Year <- as.factor(format(Fdate, "%Y"))
```

```
root <- sapply(lambda, function(l){
  mu <- mean(validation$rating)
  AdjustG <- validation %>% group_by(movieId) %>%
    summarise(AdjustG = sum(rating - mu)/(n()+l))

  AdjustU <- validation %>% left_join(AdjustG, by="movieId") %>% group_by(userId) %>%
    summarise(AdjustU = sum(rating - AdjustG - mu)/(n()+l))

  AdjustM <- validation %>% left_join(AdjustG, by="movieId") %>%
    left_join(AdjustU, by="userId") %>% group_by(RelYear) %>%
    summarise(AdjustM = sum(rating - AdjustG -AdjustU - mu)/(n()+l))

  prediction <- validation %>% left_join(AdjustG, by = "movieId") %>%
    left_join(AdjustU, by = "userId") %>%
    left_join(AdjustM, by = "RelYear") %>%
    mutate(predict = AdjustG + AdjustU +AdjustM + mu) %>% .$predict
  return(RMSE(prediction, validation$rating))
})

write.csv(validation, file = "submission.csv", row.names = FALSE)
```

Conclusion

After reviewing the dataset it can be concluded that we were able to predict above 85% accuracy by just using two columns (UserId & MovieID). Adding an additional attribute further improves the results. A ML algorithm tends to be useful but being able to understand how data needs to be represented and structured is a challenging aspect and should remain the primary focus before deciding on a specific algorithm. I see potential in administering genres as a matrix set and will try to incorporate it in my future work.

The one limiting factor that I have come across is the demand of resources certain algorithms have. I am certain that options like cluster processing and code optimization are available and should be explored.

Final Remarks

Working on my first complete project in R I have listed below few points that I would consider prior to engaging on my next one.

- Identify the audience and means on content delivery.

  submitting the results to an audience in pdf is different than submitting code that you expect them to compile. I have wasted a lot of time streamlining the Kmeans algorithm only to find out that it is resource hungry. My current profession allows me to resource high end systems to perform required calculations but reviewers may not have access to similar resources.

- Consider cluster or GPU processing

  I am familiar with using python to improve my computation speed. Pursuing with R as my source of mathematical computation, I would surely look into how to utilize these features

- Have a flat rich data structure

  Often splitting data into columns instead of rows allows you to grasp patterns in data like constructing a data matrix. I see potential in this method but unfortunately, I have been at a press of time and could not persue this area.

- Sort out your markdown theme in the beginning

I have spent quite a bit of time fixing visual aspects of my file. I wasted time changing the code and finetuning the visual aspect to fit the result.

- Complete the project in one go

  Do not give your project breaks in completion. Often you will find that rebuilding the entire concept in your mind requires motivation on its own. Something that only requires a few days to complete could be dragged to weeks.

- Use captioner to label tables & figures

  The inbuild fig.cap function has given me problems while compiling in pdf. the figures jumble into a group during the final built. Using the captioner function allowed me to maintain the document structure.