Movie Lens Final project

Subramanian

June 16, 2019

1. Introduction

This project is to develop model to predict rating of a movie based on given data set which has around 10M records. Goal is to develop and anlayse different models and recommend the best based on the one which has lowers RMSE value. Towards this objects first we shall download and split the data into two groups - training (90%) and validation (10%)

2. Initial project setup

As First step of the exercise we shall download the data and get it fine tuned to create the model. For this we shall also download and install necessary packages & add required libraries

```r
## Data Loading and Preparation

#############################################################
# Create edx set, validation set, and submission file
#############################################################

# 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(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")

# Add the required libraries
library(caret)
library(tidyverse)
library(lubridate)

# 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 <- read.table(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")
```

```r
# Validation set will be 10% of MovieLens data

set.seed(1)
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)
```

3. Data Analysis

Before we develop our model we shall first manipulate the given data to extract the year of evaluation and year of release. Also we shall remove unwanted columns from the data set as it will unnecessarily strain the internal memory space

3.1 Data Manipulaion

```r
# Extract the year of release information from the title column and add that info to a new column year_
edx <- edx %>% mutate(year_release = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year_release = as.numeric(str_sub(title,-5,-2)))


# Add a new column called year_eval and update it with year of Evaluation information from the timestam
edx <- edx %>% mutate(year_eval = year(as_datetime(timestamp)))
validation <- validation %>% mutate(year_eval = year(as_datetime(timestamp)))


# To save memory remove the timestamp & title columns from both the training and validation data set
edx <- edx %>% select(-c(timestamp,title))
validation  <- validation  %>% select(-c(timestamp, title))
```

3.2 Exploration

Having finetuned the given data set let us now deep dive into the data to understand more about it before start developing the models

```r
# Summary details of the data set
```

Below detail shows the data between trainign and validation set are evenly distributed at high level without big variation

```r
summary(edx)
```

```
##      userId         movieId          rating         genres         
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:9000055    
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   Class :character  
##  Median :35738   Median : 1834   Median :4.000   Mode  :character  
##  Mean   :35870   Mean   : 4122   Mean   :3.512                     
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000                     
##  Max.   :71567   Max.   :65133   Max.   :5.000                     
##   year_release    year_eval   
##  Min.   :1915   Min.   :1995  
##  1st Qu.:1987   1st Qu.:2000  
##  Median :1994   Median :2002  
##  Mean   :1990   Mean   :2002  
##  3rd Qu.:1998   3rd Qu.:2005  
##  Max.   :2008   Max.   :2009  
```

```r
summary(validation)
```

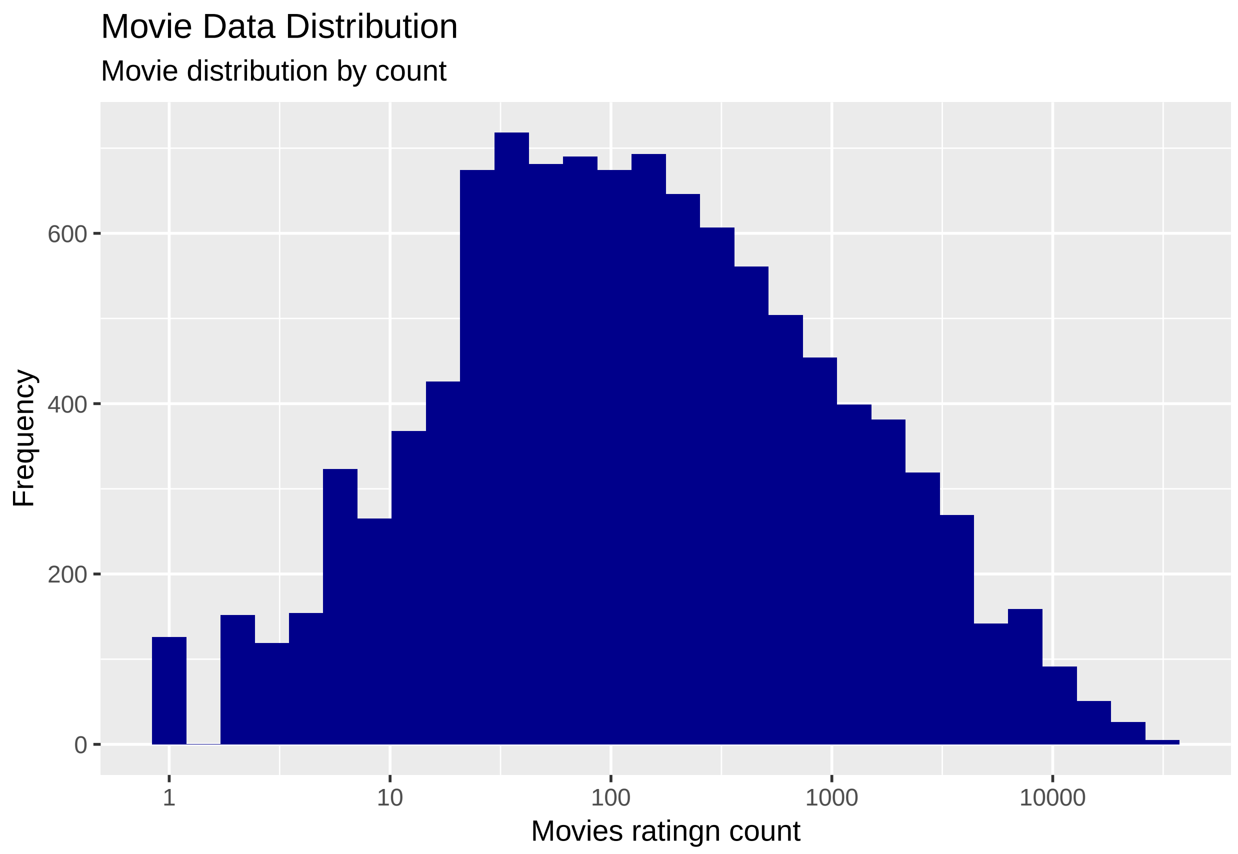
```
##      userId         movieId          rating         genres         
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:999999     
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   Class :character  
##  Median :35768   Median : 1827   Median :4.000   Mode  :character  
##  Mean   :35870   Mean   : 4108   Mean   :3.512                     
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000                     
##  Max.   :71567   Max.   :65133   Max.   :5.000                     
##   year_release    year_eval   
##  Min.   :1915   Min.   :1995  
##  1st Qu.:1987   1st Qu.:1999  
##  Median :1994   Median :2002  
##  Mean   :1990   Mean   :2002  
##  3rd Qu.:1998   3rd Qu.:2005  
##  Max.   :2008   Max.   :2009  
```

```r
# Unique list of values across different fields
edx %>% summarize(UnqGenre = n_distinct(genres), UnqMid = n_distinct(movieId) ,
                  UnqUid = n_distinct(userId), UnqYRls = n_distinct(year_release),
                  UnqYEva = n_distinct(year_eval))
```

```
##   UnqGenre UnqMid UnqUid UnqYRls UnqYEva
## 1      797  10677  69878      94      15
```

We shall deep dive into different aspects of the data that will have impact on the models

```r
# Analysing how many times different Movies have been rated
edx %>% count(movieId) %>%  ggplot(aes(n)) +
  geom_histogram(fill = "darkblue") +
  scale_x_log10() +
  labs(title = "Movie Data Distribution",
       subtitle = "Movie distribution by count",
       x = "Movies ratingn count",
       y = "Frequency")
```



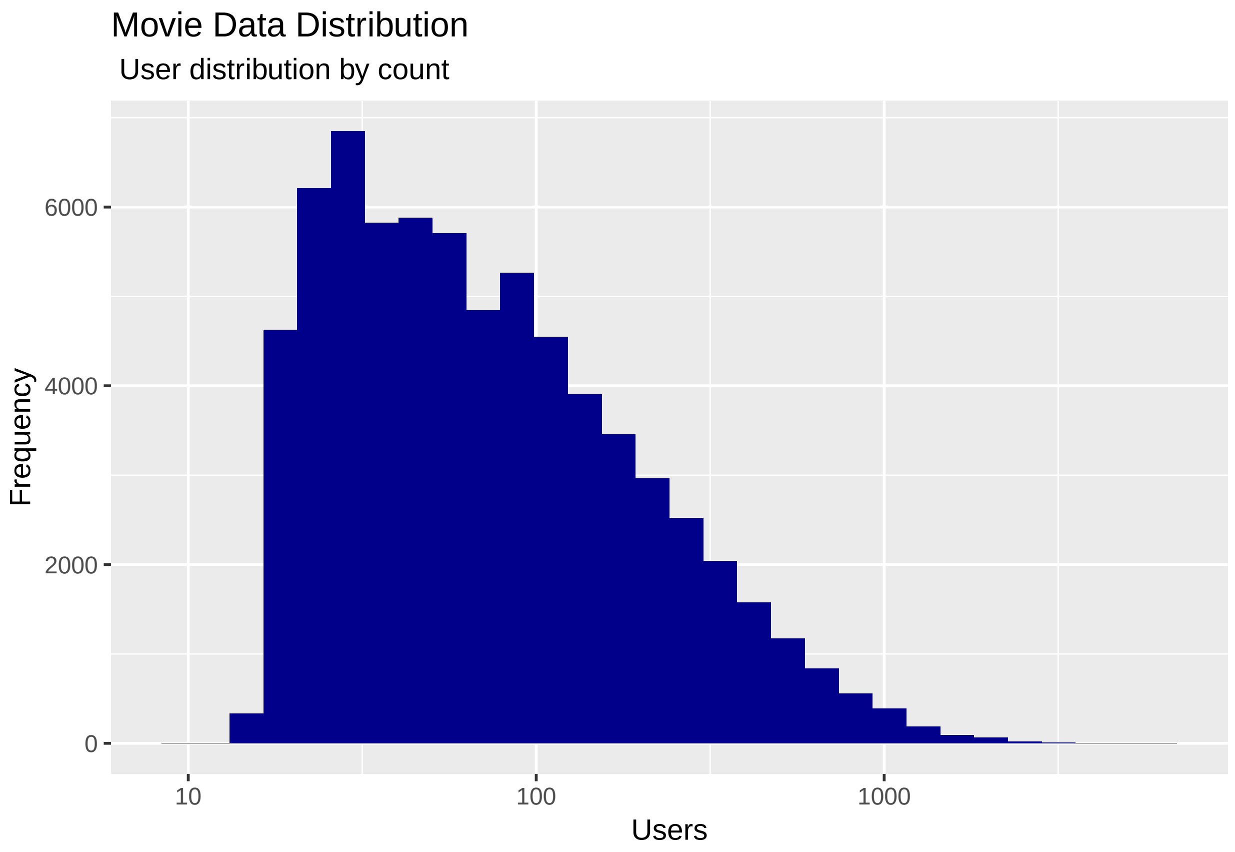
The above historgram shows that there are some movies which were rated nearly 1000 times where as some moves are rated only 10 times. This will have impact on the output of the model when rating for a movie which was less rated has to be predicted where as the prediction of rating for a movie which was rated more will be more closure to actual value

Alongside the historgram analysis We shall also look the top 5 records for each of the evaluation parameter

```r
#Top 5 movies which has more review rating than others
top5Movies <- edx %>% group_by(movieId) %>% summarize(count = n()) %>%
  arrange(desc(count)) %>% head(5)

top5Movies
```

```
## # A tibble: 5 x 2
##   movieId count
##     <dbl> <int>
## 1     296 31362
## 2     356 31079
## 3     593 30382
## 4     480 29360
## 5     318 28015
```

```r
# Analysing how many times different user have rated movies
edx %>% count(userId) %>%  ggplot(aes(n)) +
  geom_histogram(fill = "darkblue" , bins=30) +
  scale_x_log10() +
  labs(title = "Movie Data Distribution",
       subtitle = " User distribution by count",
       x = "Users",
       y = "Frequency")
```

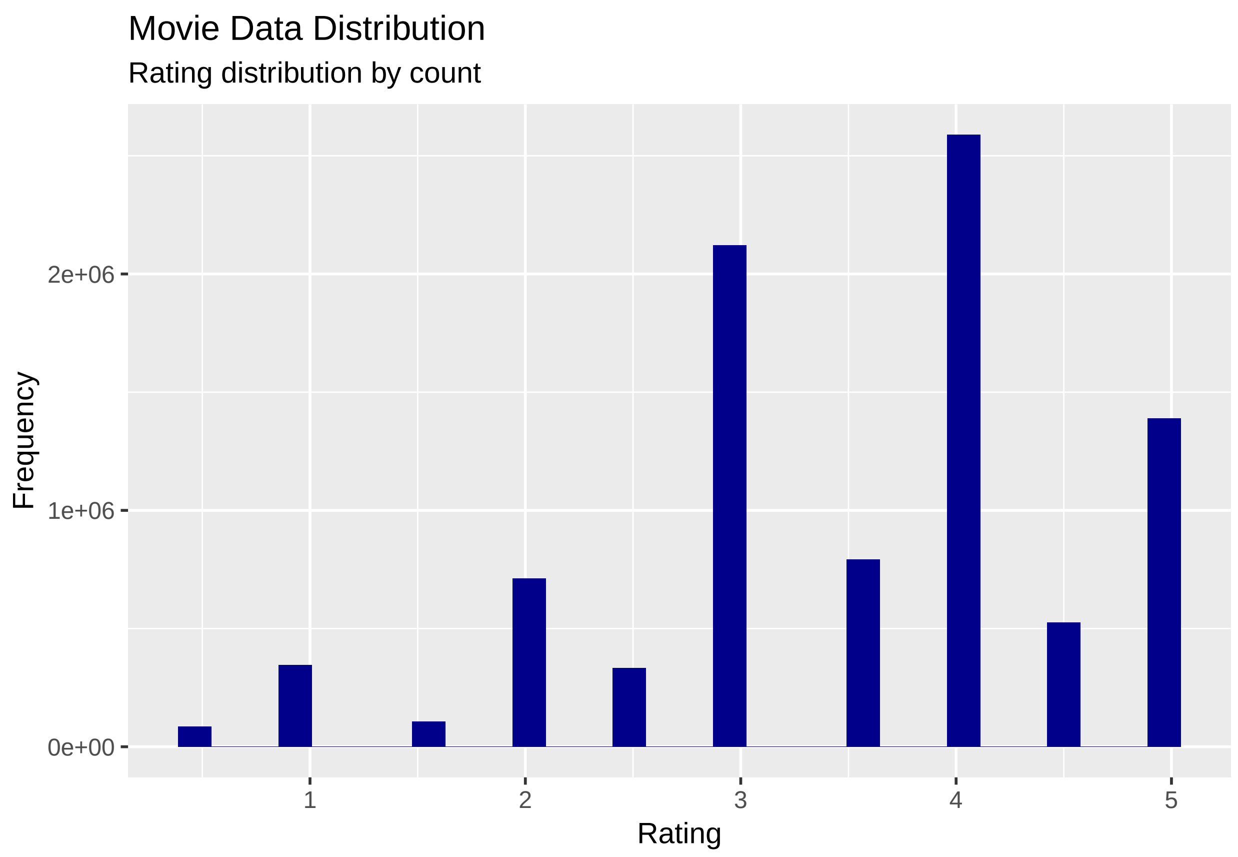


The output clearly showsf that quite large number ofo users have rated only very few movies where some users have rated more. Similar to above analysis on movie histogram this also will impact the rating prediction based on for user the rating has to be predicted

```r
# Top 5 users who rated more movies than others
top5Users  <- edx %>% group_by(userId) %>% summarize(count = n()) %>%
  arrange(desc(count)) %>% head(5)

top5Users
```

```
## # A tibble: 5 x 2
##   userId count
##    <int> <int>
## 1  59269  6616
## 2  67385  6360
## 3  14463  4648
## 4  68259  4036
## 5  27468  4023
```

```
# Analysing how many times different movies have scored similar ratings
edx %>% ggplot(aes(rating)) +
  geom_histogram(fill = "darkblue") +
  labs(title = "Movie Data Distribution",
       subtitle = "Rating distribution by count",
       x = "Rating",
       y = "Frequency")
```

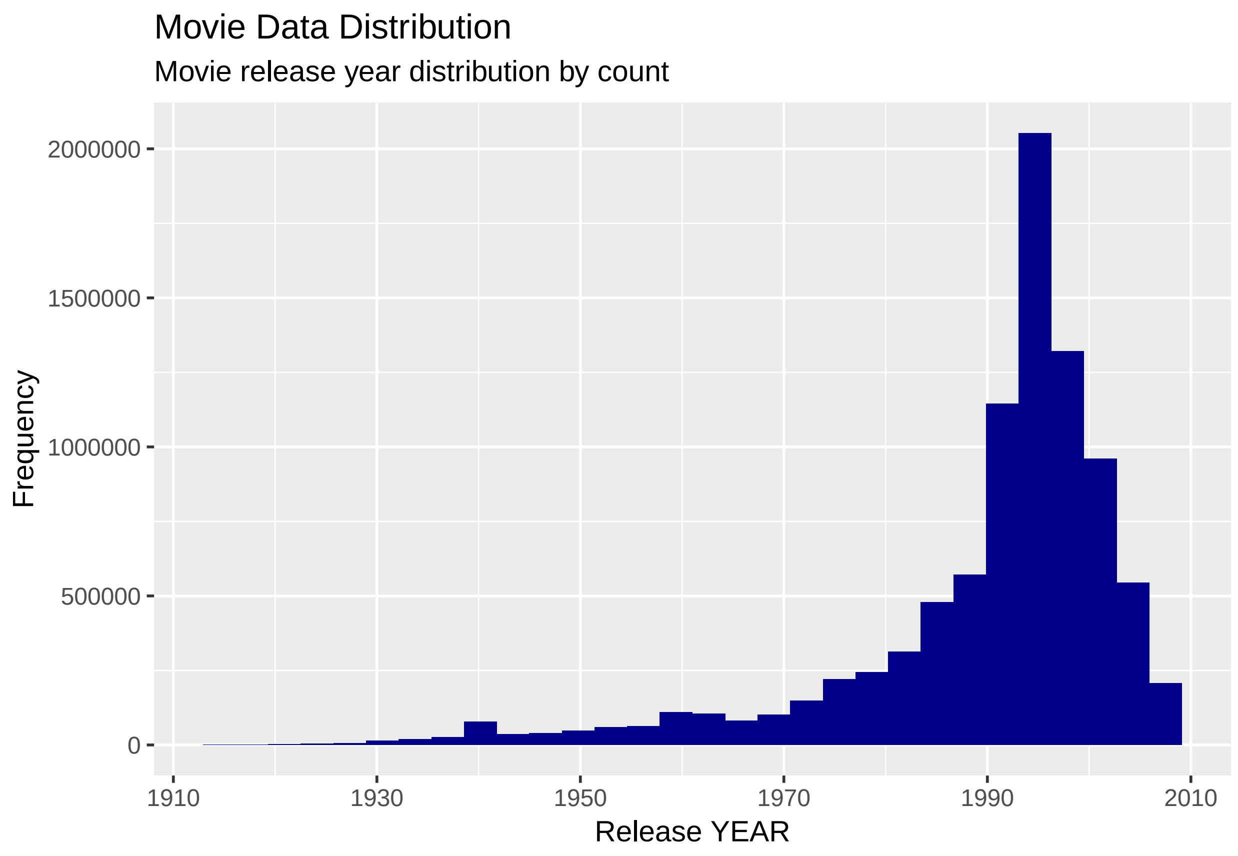


We saw above that there is huge gap between number of time different movies are rated. This will have rippling impact on the rating also based how good the movie was that was rated more. From the histogram above it looks most of the frequently rated movies were good and hence we see many movies with rating 4 and 3

```
# Ratings which has more count during movie review
top5Rating <- edx %>% group_by(rating) %>% summarize(count = n()) %>%
  arrange(desc(count)) %>% head(5)

top5Rating
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1     4    2588430
## 2     3    2121240
## 3     5    1390114
## 4     3.5   791624
## 5     2     711422
```

```
# Analysing how old the different movies are that were rated.
edx %>% ggplot(aes(year_release)) +
  geom_histogram(fill = "darkblue") +
  labs(title = "Movie Data Distribution",
       subtitle = "Movie release year distribution by count",
       x = "Release YEAR",
       y = "Frequency")
```



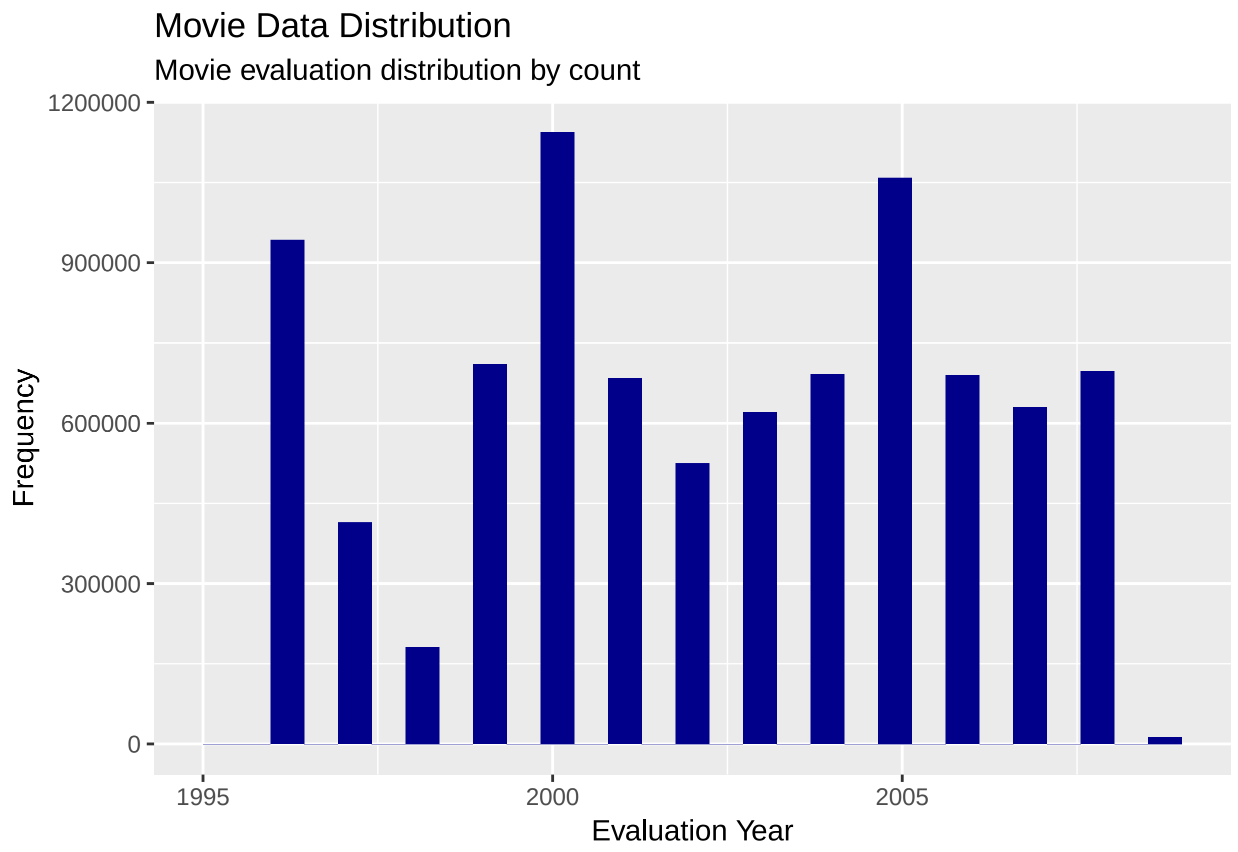
The output shows that out of the evaluated movies that were rated many were between 90s and early 2000s. This will have profound impact on the rating provided becuase even though the old movies are good enough with the advancement of technology they may not be rated much when evaluated in present situation. For eg. Horror movie taken before 1950s may not be that much appealing now as computer graphics and cinematography has advanced by many leap now

```
# The top 5 year where there are more movie releases
top5YrRls  <- edx %>% group_by(year_release) %>%
  summarize(count = n()) %>% arrange(desc(count)) %>% head(5)

top5YrRls
```

```
## # A tibble: 5 x 2
##   year_release  count
##          <dbl>  <int>
## 1         1995 786762
## 2         1994 671376
## 3         1996 593518
## 4         1999 489537
## 5         1993 481184
```

Analysing the count of when the movies were evaluated. The reason why we consider this parameter also is that as explained later the year of evaluation might have some impact when rating older movies

```r
# Analysing how many ovies were rated during different years
edx %>% ggplot(aes(year_eval)) +
  geom_histogram(fill = "darkblue") +
  labs(title = "Movie Data Distribution",
       subtitle = "Movie evaluation distribution by count",
       x = "Evaluation Year",
       y = "Frequency")
```



As explained for the year of Release parameter histogram here we can that our data contains evaluation only post 1995. So there is high probablity that rating of older movies might have been less. Also there is very less data post 2008. This will have profound impact on our prediction when the evaluation done post 2008 has to be predicted.

```r
#Top 5 years when more movies were evaluated when compared to rest of the period
top5YrEval  <- edx %>% group_by(year_eval) %>%
  summarize(count = n()) %>% arrange(desc(count)) %>% head(5)

top5YrEval
```

```
## # A tibble: 5 x 2
##   year_eval   count
##       <dbl>   <int>
## 1      2000 1144349
## 2      2005 1059277
## 3      1996  942772
## 4      1999  709893
## 5      2008  696740
```

```r
# We shall also look at the top 5 movie Genres which were reviewed more than others
top5Genres <- edx %>% group_by(genres) %>%
  summarize(count = n()) %>% arrange(desc(count)) %>% head(5)
top5Genres
```

```
## # A tibble: 5 x 2
##   genres                count
##   <chr>                 <int>
## 1 Drama                733296
## 2 Comedy               700889
## 3 Comedy|Romance       365468
## 4 Comedy|Drama         323637
## 5 Comedy|Drama|Romance 261425
```

4. Model deveopment

First we shall evaluate the baseline model where in the impact of different paramters of evaluation like user, genre, movie are not taken into consideration

4.1 Baseline model

```r
mu_edx_rating <- mean(edx$rating)
model_baseline <- RMSE(validation$rating, mu_edx_rating)

rmse_report <- tibble(method = "Base Line average Model", RMSE = model_baseline)
rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.061202

The RMSE value which is greater than 1 is significantly higher and has to be brought down

4.2 Movie genre impact model

We shall finetune the model by taking into consideration the moviegenre

```
movieGenres_avgs_norm <- edx %>% group_by(genres) %>%
  summarize(b_movieGenres = mean(rating - mu_edx_rating))

predicted_movieGenres_norm <- validation %>%
  left_join(movieGenres_avgs_norm, by='genres') %>%
  mutate(rating = mu_edx_rating + b_movieGenres )

model_movieGenres_rmse <- RMSE(predicted_movieGenres_norm$rating , validation$rating)

rmse_report <- rbind(rmse_report, tibble(method = "Movie Genre Model",
                                         RMSE = model_movieGenres_rmse))

rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.061202
Movie Genre Model	1.018406

we see that there is slight improvement in the RMSE value though not significant

4.3 Movie genre & User impact model

```
movieGenres_userId_avgs_norm <- edx %>%
  left_join(movieGenres_avgs_norm, by='genres') %>%
  group_by(userId) %>%
  summarize(b_movieGenres_userId =
              mean(rating - mu_edx_rating - b_movieGenres))

predicted_movieGenres_userId_norm <- validation %>%
  left_join(movieGenres_avgs_norm, by='genres') %>%
  left_join(movieGenres_userId_avgs_norm, by='userId') %>%
  mutate(rating = mu_edx_rating + b_movieGenres +  b_movieGenres_userId  )

model_movieGenres_userId_rmse <-
  RMSE(predicted_movieGenres_userId_norm$rating , validation$rating)

rmse_report <- rbind(rmse_report,
                     tibble(method = "Movie Genre & User effect Model",
                            RMSE = model_movieGenres_userId_rmse))
rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137

The RMSE value has improved significantly by inclusion of user detail

4.4 Movie genre , User& Year Release impact model

We shall now improvise further by adding Year of Release value

```
movieGenres_userId_yearRelease_avgs_norm <- edx %>%
  left_join(movieGenres_avgs_norm, by='genres') %>%
  left_join(movieGenres_userId_avgs_norm, by='userId') %>%
  group_by(year_release) %>%
  summarize(b_movieGenres_userId_yearRelease =
              mean(rating - mu_edx_rating -
                   b_movieGenres - b_movieGenres_userId))

predicted_movieGenres_userId_yearRelease_norm <- validation %>%
  left_join(movieGenres_avgs_norm, by='genres') %>%
  left_join(movieGenres_userId_avgs_norm, by='userId') %>%
  left_join(movieGenres_userId_yearRelease_avgs_norm, by='year_release') %>%
  mutate(rating = mu_edx_rating + b_movieGenres +  b_movieGenres_userId +
           b_movieGenres_userId_yearRelease  )

model_movieGenres_userId_yearRelease_rmse <-
  RMSE(predicted_movieGenres_userId_yearRelease_norm$rating ,
       validation$rating)

rmse_report <-
  rbind(rmse_report,
        tibble(method = "Movie Genre, User & Release Year effect Model",
               RMSE = model_movieGenres_userId_yearRelease_rmse))

rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200

The RMSE value has improved further but still it is above 0.93.

4.5 Movie genre, User, Release year & Evaluation year impact model

```
movieGenres_userId_yearRelease_yearEval_avgs_norm <- edx %>% left_join(movieGenres_avgs_norm, by='genre
  left_join(movieGenres_userId_avgs_norm, by='userId') %>%
  left_join(movieGenres_userId_yearRelease_avgs_norm, by='year_release') %>%
  group_by(year_eval) %>%
  summarize(b_movieGenres_userId_yearRelease_yearEval =
            mean(rating - mu_edx_rating - b_movieGenres -
                   b_movieGenres_userId - b_movieGenres_userId_yearRelease))
```

```
predicted_movieGenres_userId_yearRelease_yearEval_norm <- validation %>%
  left_join(movieGenres_avgs_norm, by='genres') %>%
  left_join(movieGenres_userId_avgs_norm, by='userId') %>%
  left_join(movieGenres_userId_yearRelease_avgs_norm, by='year_release') %>%
  left_join(movieGenres_userId_yearRelease_yearEval_avgs_norm, by='year_eval') %>%
  mutate(rating = mu_edx_rating + b_movieGenres +  b_movieGenres_userId +
         b_movieGenres_userId_yearRelease +
         b_movieGenres_userId_yearRelease_yearEval  )

model_movieGenres_userId_yearRelease_yearEval_rmse <-
  RMSE(predicted_movieGenres_userId_yearRelease_yearEval_norm$rating ,
       validation$rating)

rmse_report <-
  rbind(rmse_report,
        tibble(method = "Movie Genre, User, Release Year & Evaluation Year effect Model",
rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200
Movie Genre, User, Release Year & Evaluation Year effect Model	0.9330743

There is no significant impact by including the year of evaluation in the model

So far we tried model based on Movie Genre as it will help if in future we get any new movie (not in training set but belongs to genre available in training set) to be predicted for rating . Now we shall slightly change track and try model based on the Movie Id to see whether we can improve the RMSE further though this model will not be able to help if a new movie which is not in the existing data set has to be predicted for rating even thought it belongs to existing genre

4.6 Movie Id impact model

```
movieId_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_movieId = mean(rating - mu_edx_rating))

predicted_movieId_norm <- validation %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  mutate(rating = mu_edx_rating + b_movieId )

model_movieId_rmse <- RMSE(predicted_movieId_norm$rating , validation$rating)

rmse_report <-
  rbind(rmse_report, tibble(method = "Movie effect Model",
                            RMSE = model_movieId_rmse))

rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200
Movie Genre, User, Release Year & Evaluation Year effect Model	0.9330743
Movie effect Model	0.9439087

4.7 Movie Id & User impact model

```
movieId_userId_avgs_norm <- edx %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_movieId_userId = mean(rating - mu_edx_rating - b_movieId))

predicted_movieId_userId_norm <- validation %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  mutate(rating = mu_edx_rating + b_movieId +  b_movieId_userId  )

model_movieId_userId_rmse <-
  RMSE(predicted_movieId_userId_norm$rating , validation$rating)

rmse_report <- rbind(rmse_report, tibble(method = "Movie & User effect Model",
                                         RMSE = model_movieId_userId_rmse))

rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200
Movie Genre, User, Release Year & Evaluation Year effect Model	0.9330743
Movie effect Model	0.9439087
Movie & User effect Model	0.8653488

Inclusion of user impact to the movie Id model has significantly brought down the RMSE value to below 0.9 . We shall further explore whether this can further be brought down byincluding Year of Release parameter

4.8 Movie Id, User & Year Release impact model

```
movieId_userId_yearRelease_avgs_norm <- edx %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  group_by(year_release) %>%
  summarize(b_movieId_userId_yearRelease =
    mean(rating - mu_edx_rating - b_movieId - b_movieId_userId))

predicted_movieId_userId_yearRelease_norm <- validation %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  left_join(movieId_userId_yearRelease_avgs_norm, by='year_release') %>%
  mutate(rating = mu_edx_rating + b_movieId +  b_movieId_userId +
           b_movieId_userId_yearRelease  )

model_movieId_userId_yearRelease_rmse <-
  RMSE(predicted_movieId_userId_yearRelease_norm$rating , validation$rating)

rmse_report <-
  rbind(rmse_report, tibble(method = "Movie, User & Release Year effect Model ",
                            RMSE = model_movieId_userId_yearRelease_rmse))

rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200
Movie Genre, User, Release Year & Evaluation Year effect Model	0.9330743
Movie effect Model	0.9439087
Movie & User effect Model	0.8653488
Movie, User & Release Year effect Model	0.8650043

Inclusion of Year of Release has further fine tuned our model. Now we shall try to improvise by adding additional parameter of Year of Evaluation

4.9 Movie Id, User, Release Year & Evaluation Year impact model

```
movieId_userId_yearRelease_yearEval_avgs_norm <- edx %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  left_join(movieId_userId_yearRelease_avgs_norm, by='year_release') %>%
  group_by(year_eval) %>%
  summarize(b_movieId_userId_yearRelease_yearEval =
    mean(rating - mu_edx_rating - b_movieId - b_movieId_userId  -
           b_movieId_userId_yearRelease))

predicted_movieId_userId_yearRelease_yearEval_norm <- validation %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  left_join(movieId_userId_yearRelease_avgs_norm, by='year_release') %>%
  left_join(movieId_userId_yearRelease_yearEval_avgs_norm, by='year_eval') %>%
  mutate(rating = mu_edx_rating + b_movieId +  b_movieId_userId +
         b_movieId_userId_yearRelease + b_movieId_userId_yearRelease_yearEval)
```

```r
model_movieId_userId_yearRelease_yearEval_rmse <-
  RMSE(predicted_movieId_userId_yearRelease_yearEval_norm$rating ,
       validation$rating)

rmse_report <-
  rbind(rmse_report,
        tibble(method = "Movie, User, Release Year & Evaluation Year effect Model",
               RMSE = model_movieId_userId_yearRelease_yearEval_rmse))

rmse_report%>%knitr::kable()
```

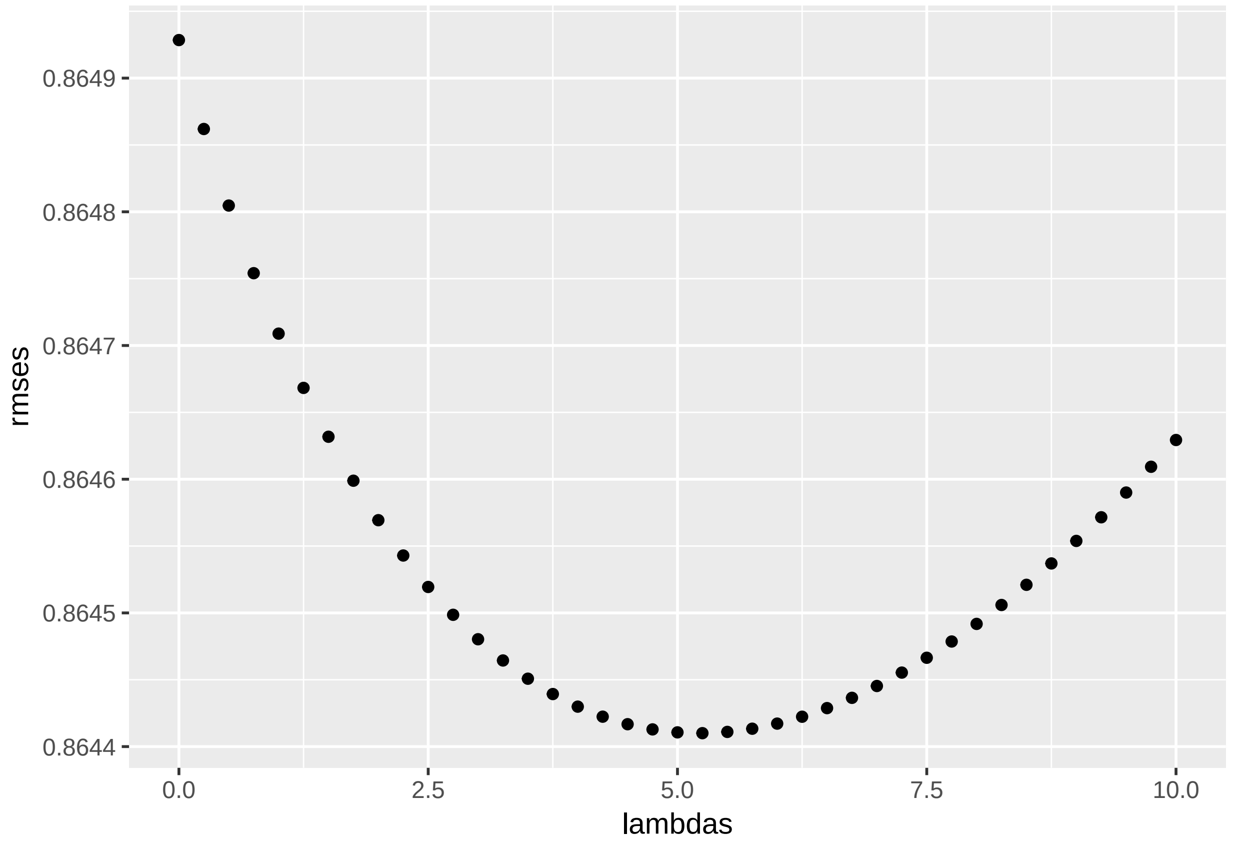
method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200
Movie Genre, User, Release Year & Evaluation Year effect Model	0.9330743
Movie effect Model	0.9439087
Movie & User effect Model	0.8653488
Movie, User & Release Year effect Model	0.8650043
Movie, User, Release Year & Evaluation Year effect Model	0.8649285

The RMSE value has now come to desired value of around 0.85 which is far lower than the base line average.

4.10 Regularization of Movie Id, User, Release Year & Evaluation Year impact model

Now we shall try to improvise this model further with the regularization which helps us to penalize large estimates that are formed from small sample sizes.

```r
lambdas <- seq(0, 10, 0.25)

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

movieId_avgs_norm <- edx %>% group_by(movieId) %>%
  summarize(b_movieId = sum(rating - mu_edx_rating)/(n()+l))

movieId_userId_avgs_norm <- edx %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_movieId_userId = sum(rating - mu_edx_rating - b_movieId)/(n()+l))


movieId_userId_yearRelease_avgs_norm <- edx %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  group_by(year_release) %>%
  summarize(b_movieId_userId_yearRelease =
      sum(rating - mu_edx_rating - b_movieId - b_movieId_userId)/(n()+l))
movieId_userId_yearRelease_yearEval_avgs_norm <- edx %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  left_join(movieId_userId_yearRelease_avgs_norm, by='year_release') %>%
  group_by(year_eval) %>%
  summarize(b_movieId_userId_yearRelease_yearEval =
    sum(rating - mu_edx_rating - b_movieId - b_movieId_userId  -
           b_movieId_userId_yearRelease)/(n()+l))

predicted_movieId_userId_yearRelease_yearEval_norm <- validation %>%
  left_join(movieId_avgs_norm, by='movieId') %>%
  left_join(movieId_userId_avgs_norm, by='userId') %>%
  left_join(movieId_userId_yearRelease_avgs_norm, by='year_release') %>%
  left_join(movieId_userId_yearRelease_yearEval_avgs_norm, by='year_eval') %>%
  mutate(rating = mu_edx_rating + b_movieId +  b_movieId_userId +
         b_movieId_userId_yearRelease + b_movieId_userId_yearRelease_yearEval)


 return(RMSE(predicted_movieId_userId_yearRelease_yearEval_norm$rating ,
       validation$rating))

})

qplot(lambdas, rmses)
```



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

```
## [1] 5.25
```

```r
rmse_report <-
  rbind(rmse_report,
        tibble(method = " Regularized Movie, User, Release Year & Evaluation Year effect Model",
               RMSE = min(rmses)))

rmse_report%>%knitr::kable()
```

method	RMSE
Base Line average Model	1.0612018
Movie Genre Model	1.0184056
Movie Genre & User effect Model	0.9402137
Movie Genre, User & Release Year effect Model	0.9332200
Movie Genre, User, Release Year & Evaluation Year effect Model	0.9330743
Movie effect Model	0.9439087
Movie & User effect Model	0.8653488
Movie, User & Release Year effect Model	0.8650043
Movie, User, Release Year & Evaluation Year effect Model	0.8649285
Regularized Movie, User, Release Year & Evaluation Year effect Model	0.8644101

The Regularization has further reduced the RMSE value and turns out to be the best of all the models that we tried so far . This can be used to predict the rating of any movie

5. Conclusion

Based on the limited input paramters available in the data set our last model based on the 4 parameters (movieId, user, release year & evaluation year) is the best of the model that can be arrived to predict any future movie. If we can have more parameters like the age & gender & preference (taste) of the evaluator this can be further tuned to achieve more precision in predicting values