Movie Recommendation System

Capstone: Final Writeup

Andrew Chikunga

02 August 2020

1. INTRODUCTION

Movie Recommendation

Every successful Data Scientist has built at least one recommendation engine in his career. A movie recommendation is important in our social life due to its strength in providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although, a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the existing users efficiently or to a new user by any means. In this project, I built a movie rating prediction system based on selected training sets provided by MovieLens and the RMSE (Root-Mean-Square-Error) is applied as the main criteria to evaluate the performance of the model.

What is a Recommendation System?

A recommendation system provides suggestions to the users through a filtering process that is based on user preferences and browsing history. The information about the user is taken as an input. The information is taken from the input that is in the form of browsing data. This information reflects the prior usage of the product as well as the assigned ratings. A recommendation system is a platform that provides its users with various contents based on their preferences and likings. A recommendation system takes the information about the user as an input. The recommendation system is an implementation of the machine learning algorithms.

Problem Statement:

To analyze the MovieLens data set in order to understand trends and patterns that will help to predict movie ratings and recommend new movies to users and to come up with a RMSE less than 0.86490 as a criteria to evaluate the perfomance of the model.

Data Set Description:

The data set used for this project was taken from MovieLens dataset and after processing a new dataset called edx was created. In this project,I am developing algorithms to predict movie ratings.

Importing Essential Libraries

In our Data Science project, we will make use of these packages – ‘tidyverse’,‘dslabs’ ‘ggplot2’, ‘data.table’ ,‘caret’,‘MatrixStats’ and ‘reshape2’.

2. DATA CLEANING

Retrieving the Data

We will now retrieve our data from link(http://files.grouplens.org/datasets/movielens/ml-10m.zip) into ratings.dat and movies.dat. We will use the str () function to display information about the movies dataframe.

```
##  chr [1:10681, 1:3] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12" "13" ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "movieId" "title" "genres"
```

Data Overview

We can overview the summary of the movies using the summary () function. We will also use the head () function to print the first six lines of movie_data

```
summary(movies)

##     movieId
##  1      :    1
##  10     :    1
##  100    :    1
##  1000   :    1
##  1001   :    1
##  1002   :    1
##  (Other):10675
##
title
##  War of the Worlds (2005)
:    2
##  ...All the Marbles (a.k.a. The Cal-
ifornia Dolls) (1981)   :    1
##  ...And God Created Woman (Et
Dieu... créa la femme) (1956):    1
##  ...And God Spoke (1993)
:    1
##  ...And Justice for All (1979)
:    1
##  'burbs, The (1989)
:    1
##  (Other)
:10674
##              genres
##  Drama          :1817
##  Comedy         :1047
##  Comedy|Drama   : 551
##  Drama|Romance  : 412
##  Comedy|Romance: 379
##  Documentary    : 350
##  (Other)        :6125
```

```
head(movies)

##      movieId title
## [1,] "1"     "Toy Story (1995)"
## [2,] "2"     "Jumanji (1995)"
## [3,] "3"     "Grumpier Old Men
(1995)"
## [4,] "4"     "Waiting to Exhale
(1995)"
## [5,] "5"     "Father of the Bride
Part II (1995)"
## [6,] "6"     "Heat (1995)"
##      genres
## [1,] "Adven-
ture|Animation|Children|Comedy|Fantasy"
## [2,] "Adventure|Children|Fantasy"
## [3,] "Comedy|Romance"
## [4,] "Comedy|Drama|Romance"
## [5,] "Comedy"
## [6,] "Action|Crime|Thriller"
```

Similarly, we can output the summary as well as the first six lines of the 'rating_data' dataframe.

```
summary(ratings)

##      userId          movieId
rating        timestamp
##  Min.   :    1   Min.   :    1
Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18123   1st Qu.:  648   1st
Qu.:3.000   1st Qu.:9.468e+08
##  Median :35740   Median : 1834   Me-
dian :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4120
Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53608   3rd Qu.: 3624   3rd
Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133
Max.   :5.000   Max.   :1.231e+09
```

```
head(ratings)

##    userId movieId rating timestamp
## 1:      1     122      5 838985046
## 2:      1     185      5 838983525
## 3:      1     231      5 838983392
## 4:      1     292      5 838983421
## 5:      1     316      5 838983392
## 6:      1     329      5 838983392
```

From the above table, we observe that the userId column, as well as the movieId column, consist of integers. Furthermore, we need to convert the genres present in the movie_data dataframe into a more usable format by the users. In order to do so, we will first create a one-hot encoding to create a matrix that comprises of corresponding genres for each of the films.

3. DATA PREPROCESSING

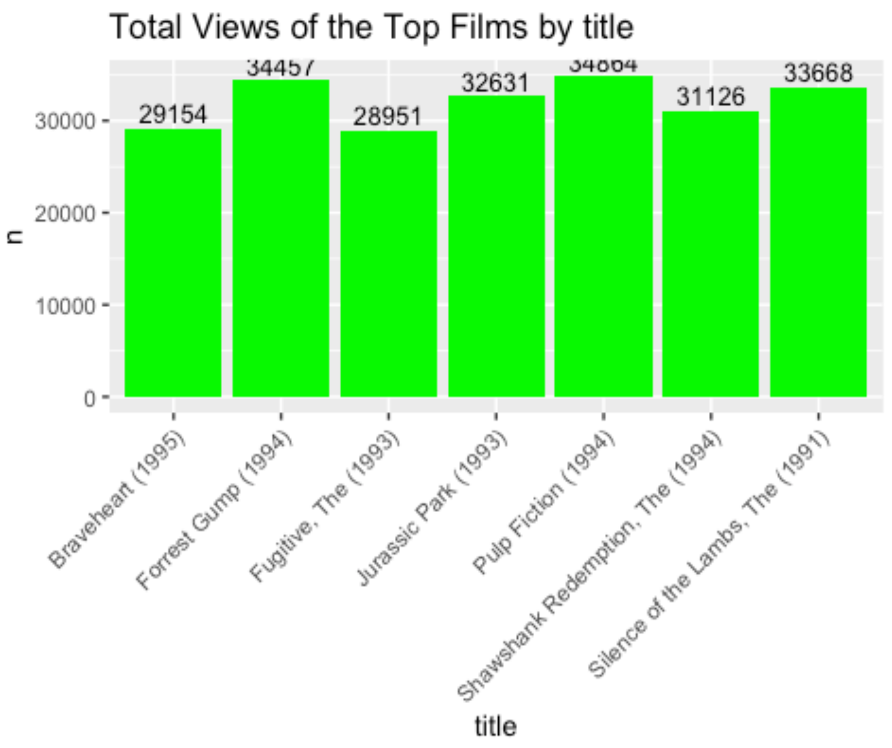
```
# summary of the movielens dataset
summary(movielens)

##      userId          movieId
rating        timestamp
##  Min.   :    1   Min.   :    1
Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18123   1st Qu.:  648   1st
Qu.:3.000   1st Qu.:9.468e+08
##  Median :35740   Median : 1834   Me-
dian :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4120
Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53608   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:10000054    Length:10000054
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
```

```
str(movielens)

## Classes 'data.table' and 'da-
ta.frame':   10000054 obs. of  6 varia-
bles:
##  $ 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 838984885
838983707 ...
##  $ title    : chr  "Boomerang
(1992)" "Net, The (1995)" "Dumb & Dumb-
er (1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance"
"Action|Crime|Thriller" "Comedy" "Ac-
tion|Drama|Sci-Fi|Thriller" ...
##  - attr(*,
".internal.selfref")=<externalptr>
```

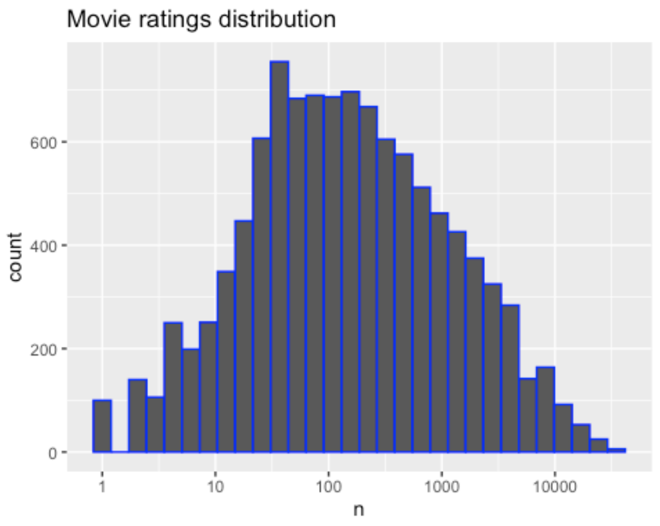
```
# Unique users that provided ratings


Total number of unique users: 69878



Total number of unique movies: 10677
```

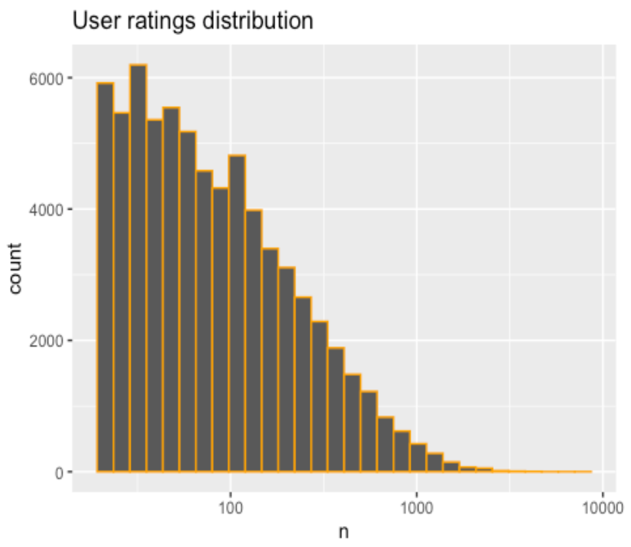
```
Top_5_movies

1.Pulp Fiction (1994):296

2. Shawshank Redemption, The (1994):318


3. Forrest Gump (1994):356

4.Jurassic Park (1993) :480


5. Silence of the Lambs, The(1991):593
```

Now, we will visualize a bar plot for the total number of views of the top films.



Distribution of movie and user ratings





Creating training and validation sets

Now in order to study the structure of our data set, we call the str() method. This gives us a descriptive summary of all the predictor variables present in the data set and check for missing values.

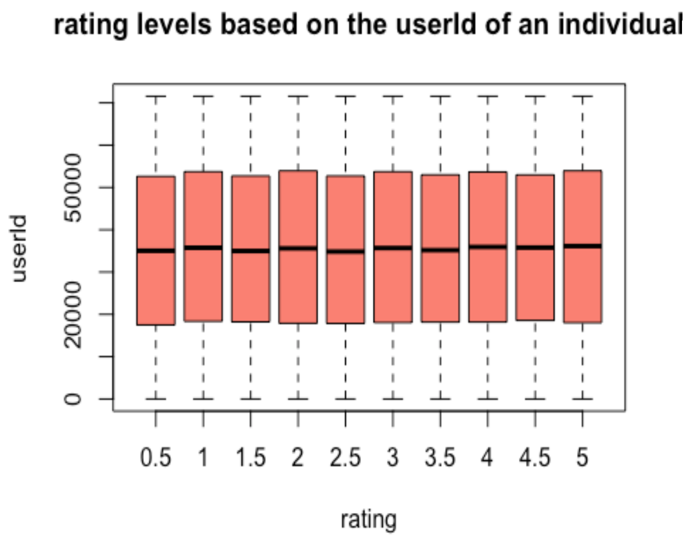
```
#Checking for missing values
table (complete.cases (edx))
##    TRUE
## 9000055

#Display structure of the data
str (edx)

## Classes 'data.table' and 'da-
ta.frame':   9000055 obs. of  6 varia-
bles:
##  $ 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 838983707
838984596 ...
##  $ title    : chr  "Boomerang
(1992)" "Net, The (1995)" "Outbreak
(1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance"
"Action|Crime|Thriller" "Ac-
tion|Drama|Sci-Fi|Thriller" "Ac-
tion|Adventure|Sci-Fi" ...
##  - attr(*,
".internal.selfref")=<externalptr>
```

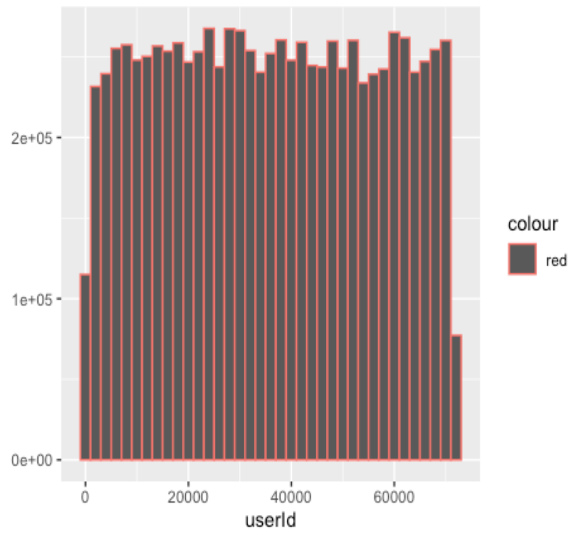
4. DATA EXPLORATION

Data Exploration involves analyzing each feature variable to check if the variables are significant for building the model.

Exploring the userId variable

```
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1   18124   35738   35870   53607   71567 

 69878
```



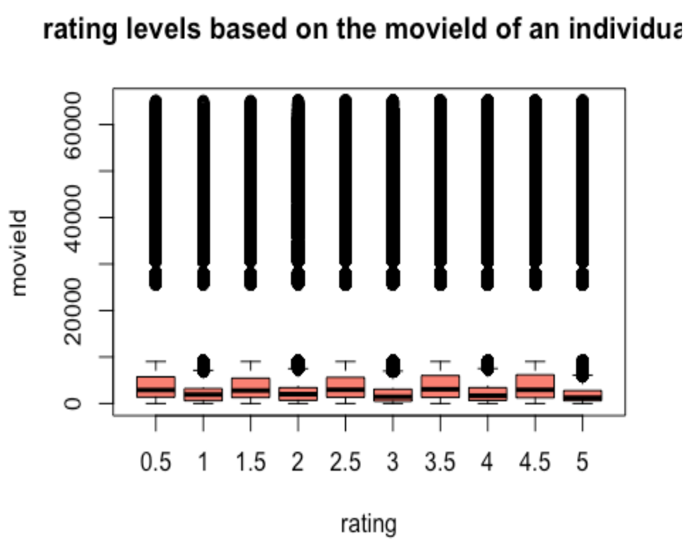


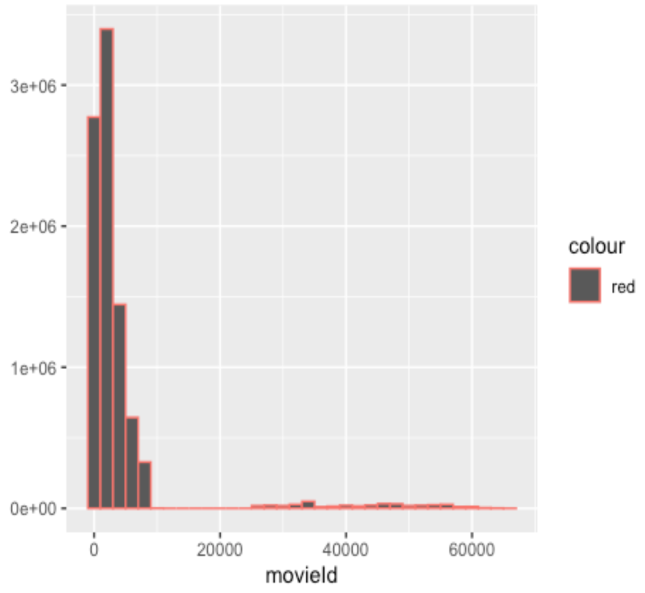
The diagrams above shows that the userId variable is not varying with the level of rating and hence it is not a strong predictor variable.

Exploring the movieId variable

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     648    1834    4122    3626   65133
```

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



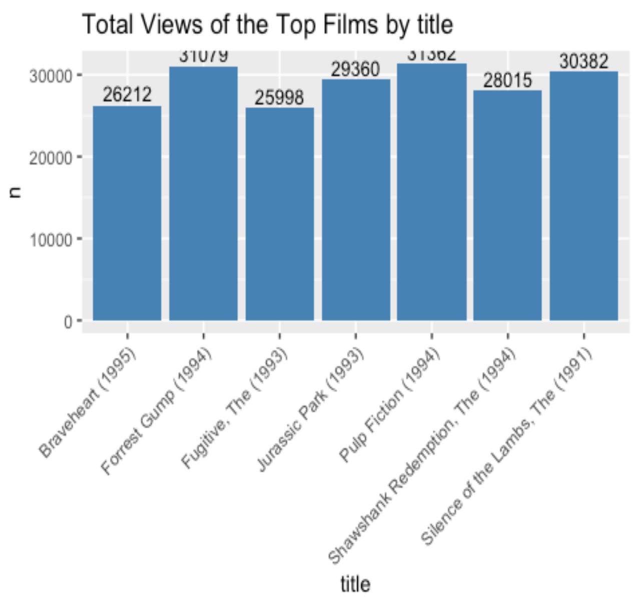


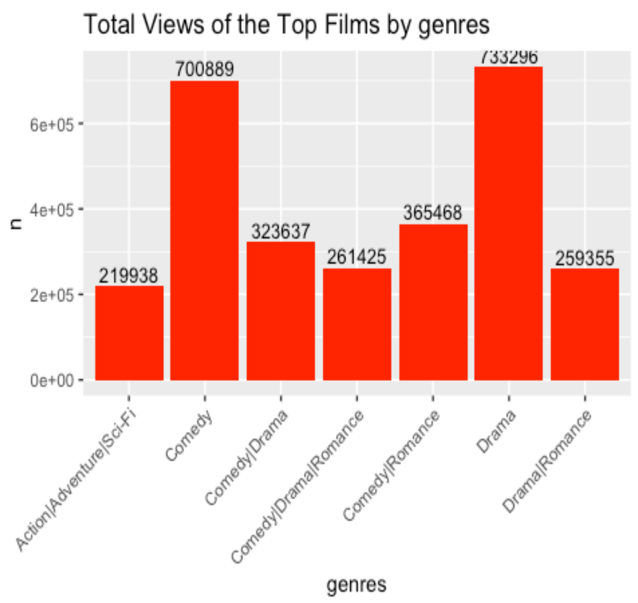
The above illustrations show that the movieId variable is varying with the level of rating and hence it is a strong predictor variable.

Similarly, we'll be evaluating categorical variables as well.

In the below section I've created qplots for each variable and after evaluating the plots, it is clear that these variables are essential for predicting the rating of a movie.

```
##    Drama   Comedy Thriller  Romance
##  3910127  3540930  2325899  1712100

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

## Selecting by n
```





All these graphs above show that these set of predictor variables are significant for building our predictive model.

5. BUILDING THE MODEL

So, after evaluating all our predictor variables, it is finally time to perform Predictive analytics. In this stage, we'll build a predictive model that will recommend movies to users based on their past preferences.

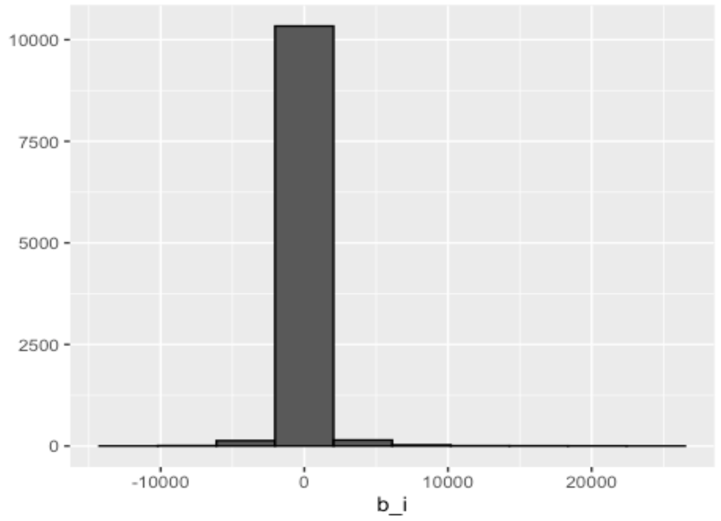
Loading and evaluating the validation data set: edx

We start with a model that assumes the same rating for all movies and all users, with all the differences explained by random variation: If u represents the true rating for all movies and users. u represents independent errors sampled from the same distribution centered at zero, then:

```
#least squares estimate of the mean mu
mu <- mean(edx$rating)
mu

3.512465
```

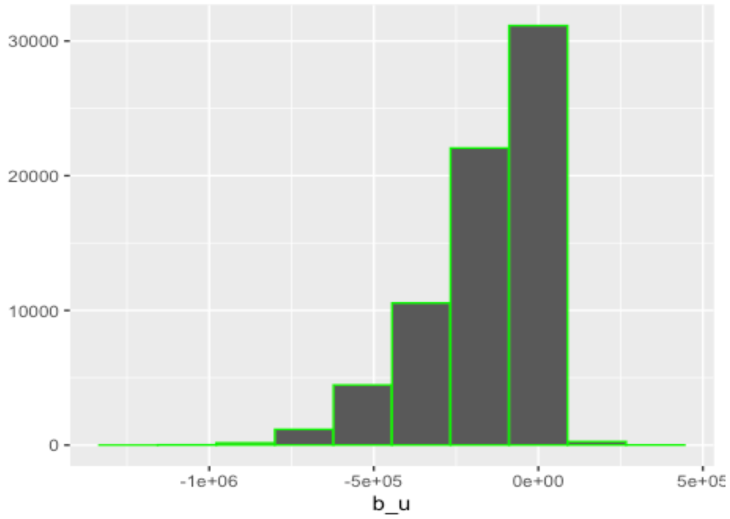
Investigating the effects of the variables that affect the movie rating

Average rating for movie:



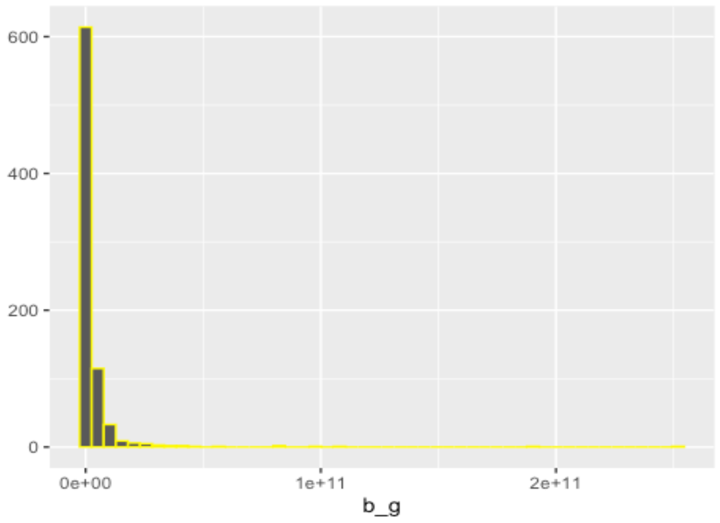
Histogram above suggest that movie average is normalized. We can see that these estimates vary substantially, not surprisingly, some movies are good and Other movies are bad.

User specific effect:



Note that there is substantial variability across users, as well. Some users are very cranky while others love every movie they watch, while others are somewhere in the middle.

genres specific effect:



The plot above shows that there is substantial variability across all genres, as well. Some genres are more popular while others are rarely watched.

Regularization

Recommendation systems are more complicated machine learning challenges because each outcome has a different set of predictors. For example, different users rate a different number of movies and rate different movies.

To compare different models or to see how well we're doing compared to a baseline, we will use root mean squared error (RMSE) as our loss function. We can interpret RMSE similar to standard deviation.

If N is the number of user-movie combinations $y_{u,i}$ is the rating for movie i by user u and $\hat{y}_{u,i}$ is our prediction, then RMSE is defined as follows: $\sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$

Motivation

We start with a model that assumes the same rating for all movies and all users, with all the differences explained by random variation: If μ represents the true rating for all movies and users and $\varepsilon_{u,i}$ represents independent errors sampled from the same distribution centered at zero, then: $Y_{u,i} = \mu + \epsilon_{u,i}$

We can improve our model by adding a term b_i that represents the average rating for movie i: $Y_{u,i} = \mu + b_i + \epsilon_{u,i}$. b_i is the average of $Y_{u,i}$ minus the overall mean for each movie i .

We can further improve our model by adding b_u the user-specific effect and b_g the genres-specific effect: $Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$

Note that because there are thousands of b's , the lm() function will be very slow or cause R to crash, so we don't recommend using linear regression to calculate these effects.

```
#First, let’s create a database that connects movieId to movie title:


      movieId                       title
 1:     122             Boomerang (1992)
 2:     185              Net, The (1995)
 3:     292              Outbreak (1995)
 4:     316              Stargate (1994)
 5:     329 Star Trek: Generations (1994)
 6:     355        Flintstones, The (1994)
```

Here are the 10 best movies according to our estimate and how often they are rated:

title	b_i	n
Shawshank Redemption, The (1994)	26408.79	28015
Silence of the Lambs, The (1991)	21013.28	30382
Pulp Fiction (1994)	20144.57	31362
Schindler’s List (1993)	19737.89	23193
Usual Suspects, The (1995)	18474.15	21648
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	18197.49	25672
Godfather, The (1972)	16023.78	17747
Forrest Gump (1994)	15550.59	31079
Braveheart (1995)	14924.76	26212
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	14698.21	19678

Here are the 10 worst movies according to our estimate and how often they are rated:

title	b_i	n
Batman Forever (1995)	-10300.069	17414
Ace Ventura: Pet Detective (1994)	-10047.328	18959
Ace Ventura: When Nature Calls (1995)	-9737.012	10430
Waterworld (1995)	-9434.572	14446
Dumb & Dumber (1994)	-9268.104	16053
Judge Dredd (1995)	-7729.288	7885
Congo (1995)	-7520.011	8304
Honey, I Shrunk the Kids (1989)	-6924.824	8289
Coneheads (1993)	-6830.265	7482
Wild Wild West (1999)	-6664.078	5301

Some of the supposed “worst” movies were rated by very many users. These movies were mostly obscure ones. Therefore, larger estimates of b_i, negative or positive, are more likely.

These are noisy estimates that we should not trust, especially when it comes to prediction. Large errors can increase our RMSE, so we would rather be conservative when unsure.

Penalized least squares

The general idea behind regularization is to constrain the total variability of the effect sizes. Why does this help? Consider a case in which we have movie
$i = 1$ with 100 user ratings and 4 movies $i = 2,3,4,5$ with just one user rating. We intend to fit the model $Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$

To improve our results, we will use regularization. Regularization constrains the total variability of the effect sizes by penalizing large estimates that come from small sample sizes.

To estimate the b ’s, we will now minimize this equation, which contains a penalty term:$\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i)^2 + \lambda\sum_i b_i^2$

The first term is the mean squared error and the second is a penalty term that gets larger when many $b's$ are large. $\hat{b}_i(\lambda) = \frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu})$

The values of b that minimize this equation are given by:$\hat{b}_i(\lambda) = \frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i} - \hat{\mu})$, where n_i is a number of ratings b for movie i

The larger λ is, the more we shrink.λ is a tuning parameter, so we can use cross-validation to choose it. We should be using full cross-validation on just the training set, without using the test set until the final assessment.

We can also use regularization to estimate the user effect. We will now minimize this equation: $\frac{1}{N}\sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2)$.

Load and evaluate the validation test data set

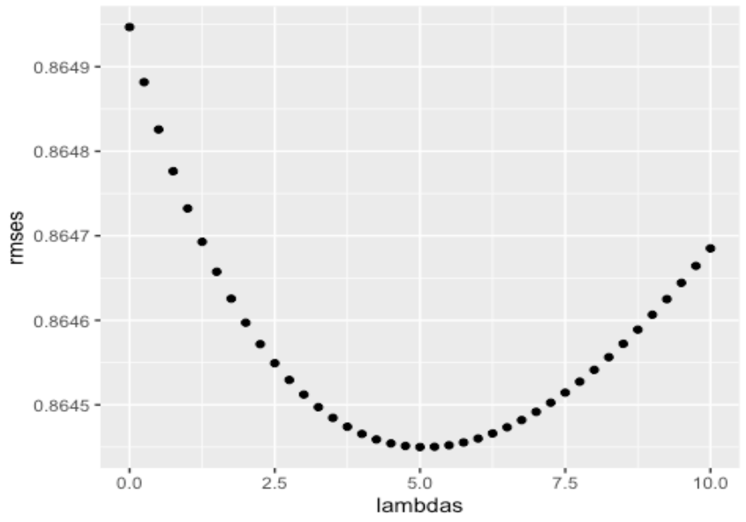
Just like how we cleaned our training data set, our validation test data must also be prepared in such a way that it does not have any null values or unnecessary predictor variables, only then can we use the test data to validate our model.

λ is a tuning parameter. We can use cross-validation to choose it:

Since I am using an RMSE loss algorithm, I’ve also implemented the Cross-Validation technique to prevent overfitting of the model.

Choosing the optimal value of lambda (λ):

Lambdas vs rmes



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

 5
```

Using results obtained above, the value of lambda that minimizes the model is 5,so I am going to use 5 to calculate the RMSE.

Calculating the value of RMSE:

```
Regularized_rmse_results<-data.frame(method="Avg movie rating + UserId-effect  + genres-effect ",RMSE = min(rmses))
Regularized_rmse_results
                                               method      RMSE
 1. Avg movie rating + UserId-effect + genres-effect   0.8644501
```

My generated RMSE is **0.8644501** which is lower than the expected **0.86490** This suggests that the model is able to recommend movies to users with a high degree of accuracy.

6. CONCLUSIONS

Recommendation Systems are the most popular type of machine learning applications that are used in all sectors. They are an improvement over the traditional classification algorithms as they can take many classes of input and provide similarity ranking based algorithms to provide the user with accurate results. These recommendation systems have evolved over time and have incorporated many advanced machine learning techniques to provide the users with the content that they want.

Movie recommendation systems which are existing have poor efficiency due to which movies are suggested in view of aspects for example - movie rated & evaluated by the User. Building a system that achieves good recommendations in new users or coldstart scenario stills is a challenge. In order to create a model with acceptable results, it may be necessary to count with more information, not only about the user's profile but also about the movies, this could allow us to implement other methodologies like Content-based filtering and Hybrid filtering, and it may lead us to more significant results.

7. REFERENCES

1. https://data-flair.training/blogs
2. https://rafalab.github.io/dsbook/large-datasets.html#regularization
3. Book by Ross Mistry:ASP.NET: The Complete Reference