Capstone - Movielens Project

Rishabh

January 4, 2020

Abstract

This project seeks to create a recommender system. A recommender system essentially seeks to predict the "rating" or "preference" a user would give to an item. For this purpose the movielens dataset has been used which has approximately 10 Million rows. Of these (10%) will be used to create a validation dataset and the rest 9 Million will be used to train the predictive models.

The movielens dataset contains genres such as Action, Adventure, Horror, Drama, Thriller etc. There are about 70k distinct userIds , and about 11k movies.

The method to evaluate how good a recommnder system will be done based on its RMSE value.

The goal is to try to achieve an RMSE less than 0.8649.

Data Exploration

Here we check if the data is consistent and check some data summaries to get familiar with it.

The edx dataset has rating score between 0.5 and 5. There is no missing values (NAs).

The validation dataset also does not have NAs.

The Mean rating is 3.512.

The training dataset

```
##   Users Movies
## 1 69878  10677
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

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

The Validation dataset

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

```
##   Users Movies
## 1 68534   9809
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   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:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Both the data have 6 features which have the following information.

userId contains a unique user identification number. movieId contains a unique identification number for each movie. rating contains a rating of one movie by one user. timestamp contains a timestamp for the rating provided by one user. title contains the movie title and release year. genres contains a list of genre each movie falls into.

Data Wrangling

Here we will convert the data available to a format which would be suited to be entered into a machine learning model.

Our data exploration suggests that there are several processing steps that we require :

1.The year of release of the movie needs to be put in a separate column .

2.The timestamp of each user for every rating needs to be converted to year and month columns.

3.The genre column contains multiple genre and this has to be made unique so that every movie gets counted as a movie in all genre that it lies in.

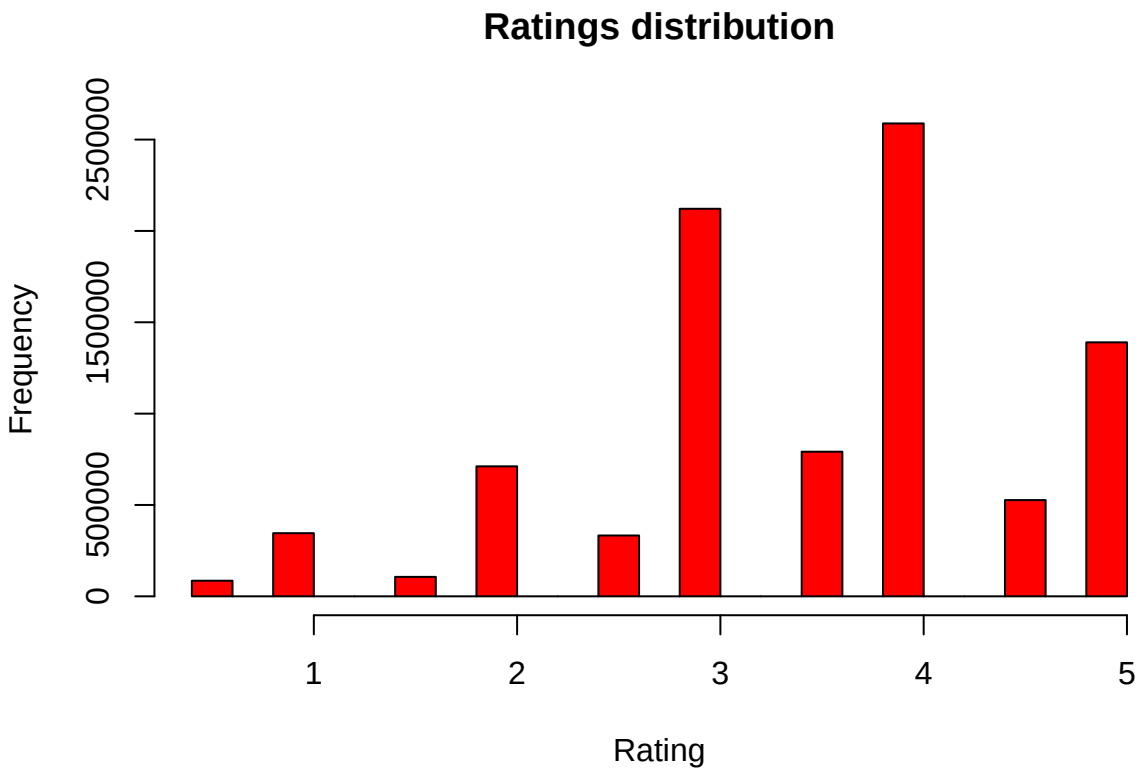
Final processed dataset

```
##   userId movieId rating rating_year                          title
## 1      1     122      5   838985046              Boomerang (1992)
## 2      1     185      5   838983525               Net, The (1995)
## 3      1     292      5   838983421               Outbreak (1995)
## 4      1     316      5   838983392               Stargate (1994)
## 5      1     329      5   838983392 Star Trek: Generations (1994)
## 6      1     355      5   838984474       Flintstones, The (1994)
##                         genres                date Rating_year
## 1                Comedy|Romance 1996-08-02 16:54:06        1996
## 2         Action|Crime|Thriller 1996-08-02 16:28:45        1996
## 3  Action|Drama|Sci-Fi|Thriller 1996-08-02 16:27:01        1996
## 4       Action|Adventure|Sci-Fi 1996-08-02 16:26:32        1996
## 5 Action|Adventure|Drama|Sci-Fi 1996-08-02 16:26:32        1996
## 6       Children|Comedy|Fantasy 1996-08-02 16:44:34        1996
##   Rating_month release_year
## 1            8         1992
## 2            8         1995
## 3            8         1995
## 4            8         1994
## 5            8         1994
## 6            8         1994
```

Insights from the data about the distribution of Ratings

Rating Distrubition

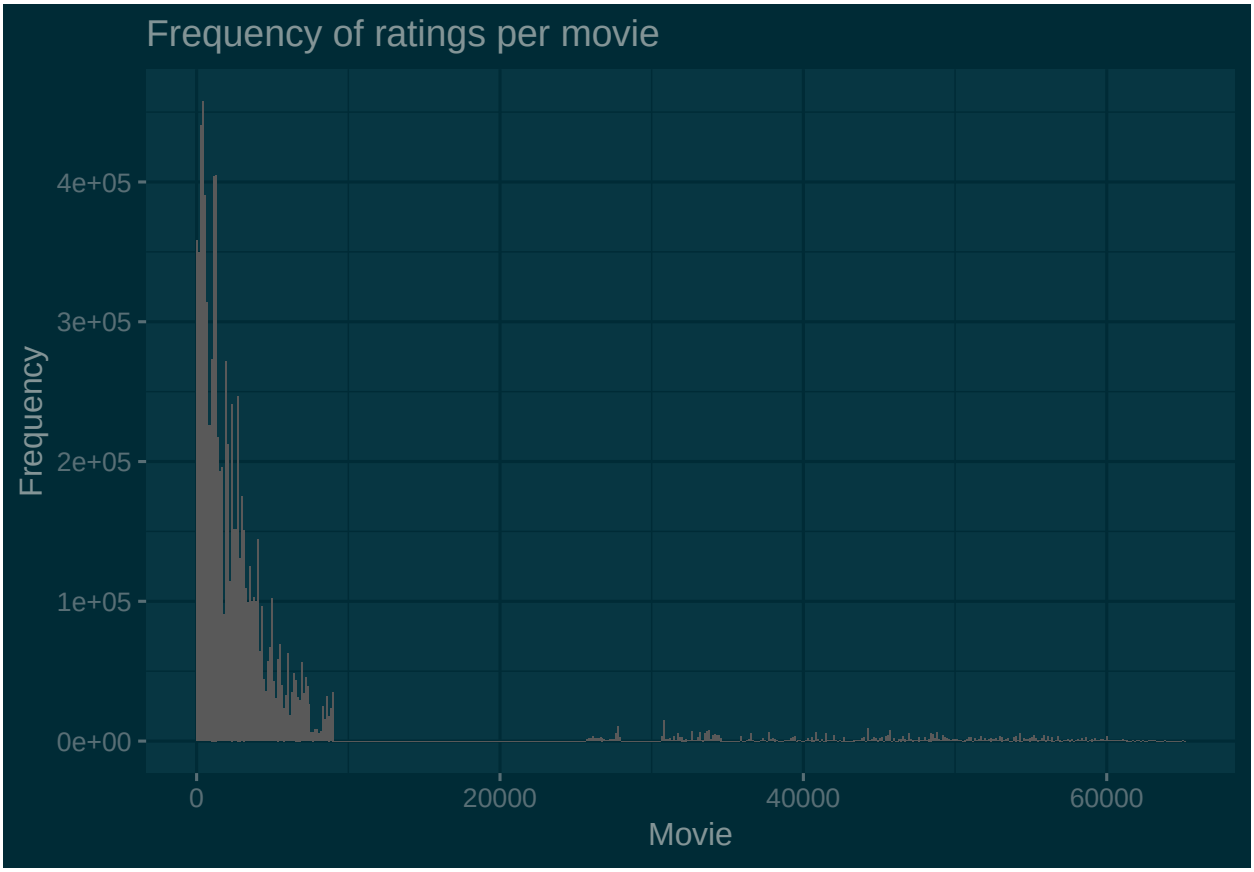
It can be inferred from the below distribution that users tend to give whole number ratings much more than those with half point ranges(e.g. 3.5).Also most of the ratings are centered around the mean which is 3.512.



Rating Distribution

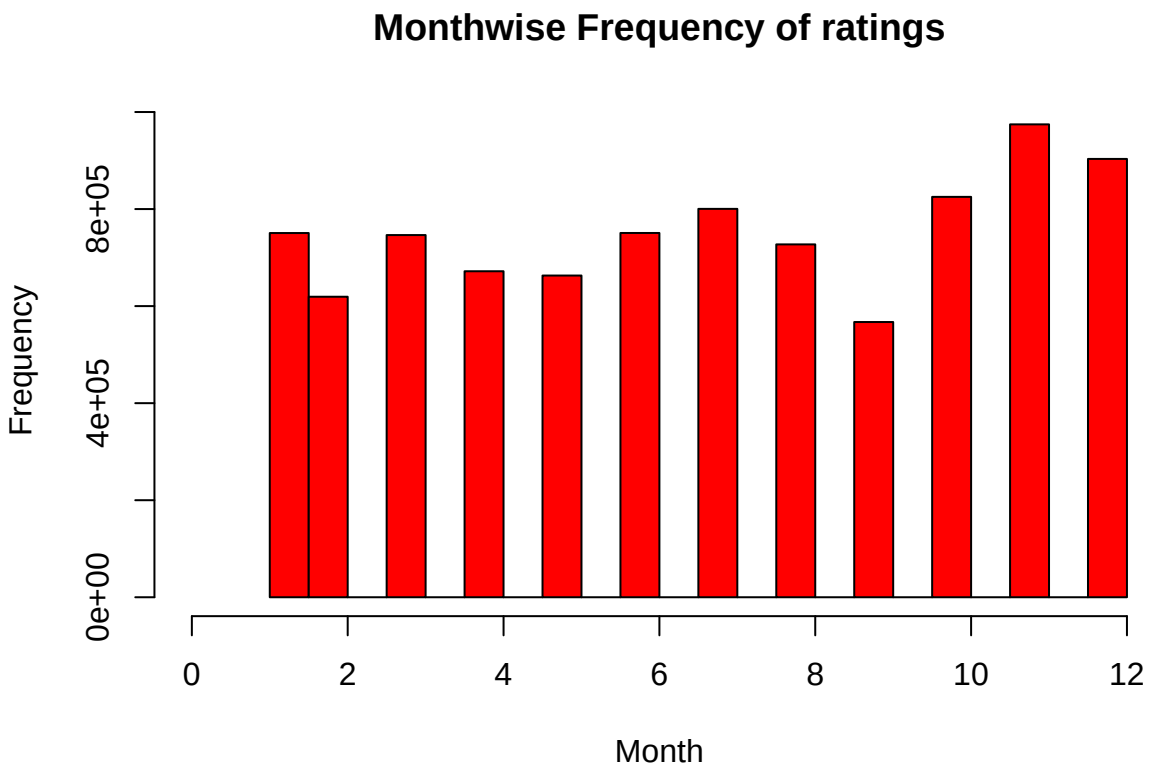
The plot below suggests that there tend to be a lot of ratings for a very less number of movies and verly less number of rating for most of the movies.This is an improrant fact which will matter while training our model as this gives rise to prevalence.

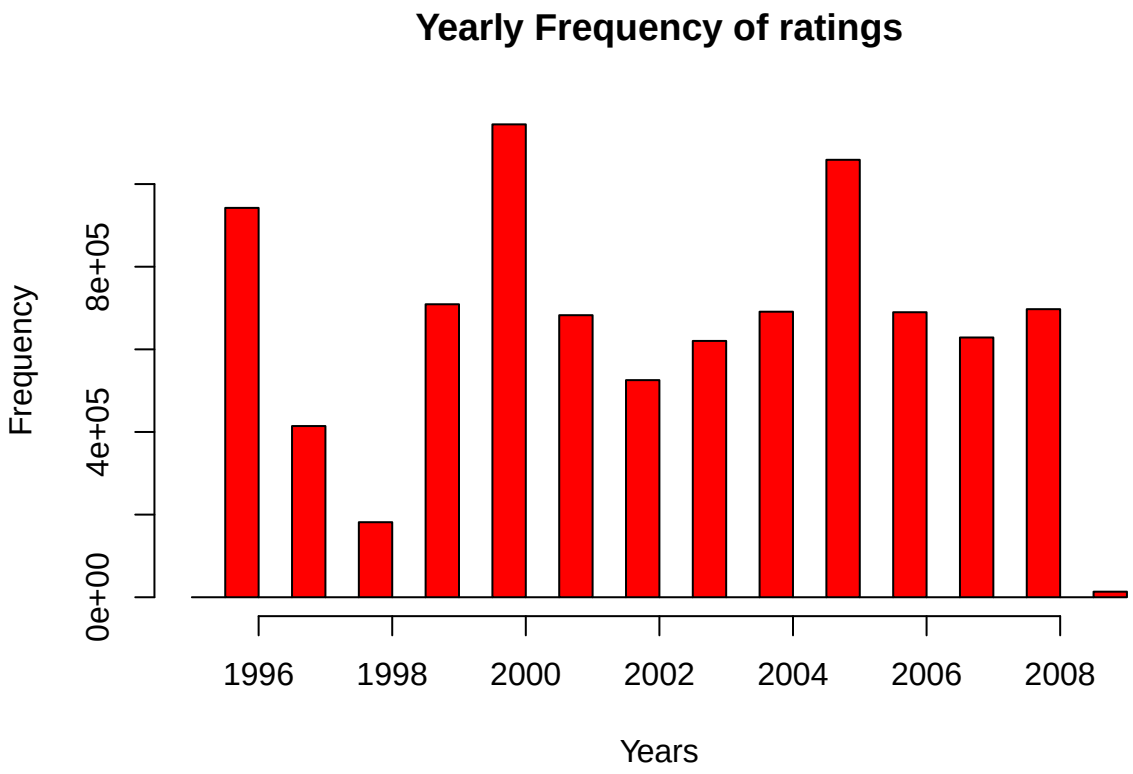
Frequency of Ratings per Movie



Rating Frequency through Months and Years

The below distributions tell us that the frequency of ratings in some months are significantly higher than others. This can again have an effect on our model in the form of prevalence.

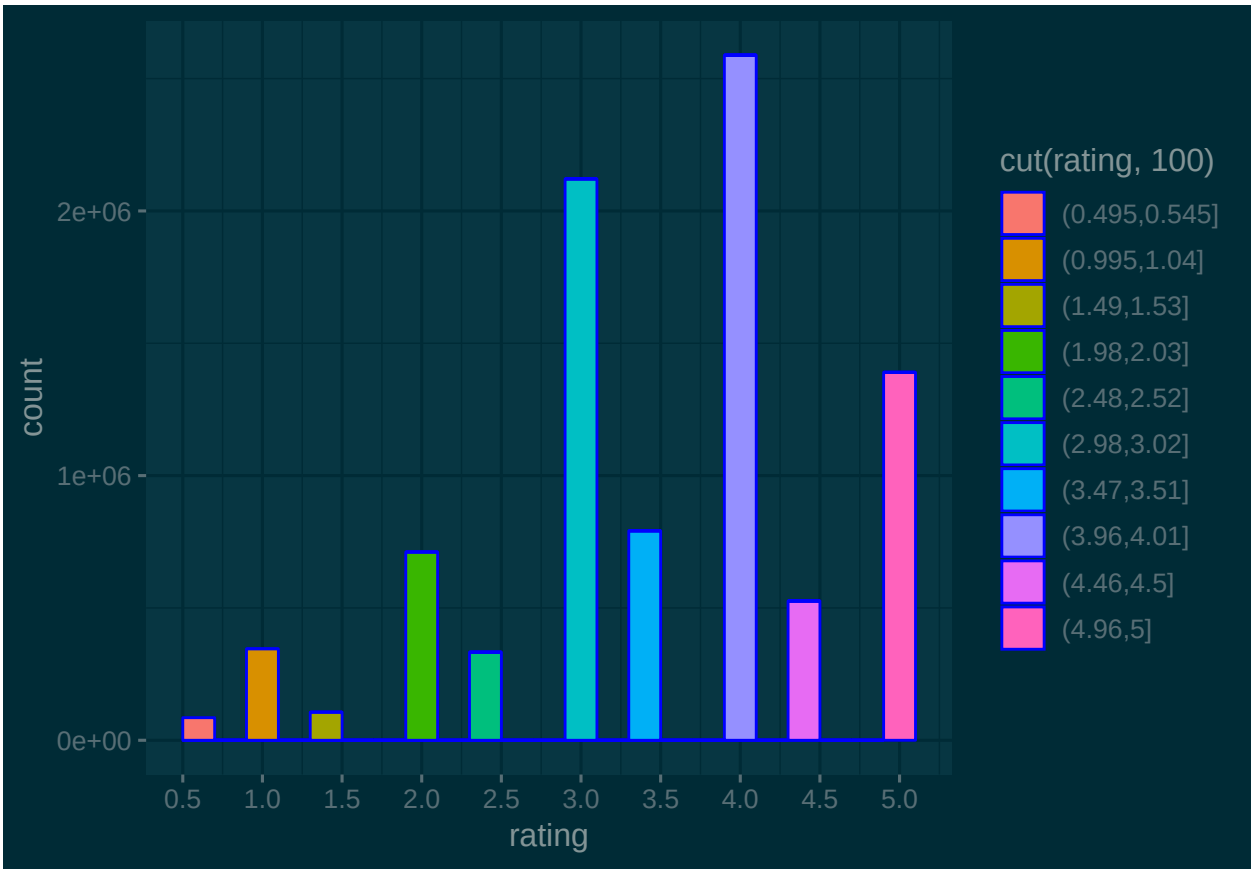

Monthwise Frequency of ratings
Frequency
0e+00
4e+05
8e+05
0
2
4
6
8
10
12
Month



Distribution of rating values

The below graph brings out the fact that the rating of 4 has been the most rated.

```
## # A tibble: 10 x 2
##    rating ratings_distribution_sum
##     <dbl>                    <int>
##  1    4                    2588430
##  2    3                    2121240
##  3    5                    1390114
##  4    3.5                   791624
##  5    2                     711422
##  6    4.5                   526736
##  7    1                     345679
##  8    2.5                   333010
##  9    1.5                   106426
## 10    0.5                    85374
```



Model Building and analysis

1. Naive Mean Model (Simplest - centred around the mean rating i.e. 3.512)

```
## [1] "The mean is: 3.51246520160155"
```

Naive Mean Model

The model formula for Naive Mean Model is as follows:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

Where $\hat{\mu}$ is the mean $\varepsilon_{i,u}$ is the independent errors centered at 0.

We obtain an RMSE of 1.0603 from the simplest Naive Mean Model.This is way below our target of 0.864. This is probably because we are not taking into account the effect of rating of previous users,the genre of the movie and perhaps the release year.

We thus try different methods taking these into account.

Model based on Movie rating (Considers the Bias that a movie has alreday been rated a certain way)

This model tries to measure the rating behaviour of the user , when the user knows the existing average rating of the movie and is predisposed to react accordingly.

The Model uses the following formula:

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

where $\hat{\mu}$ is the mean $\varepsilon_{i,u}$ is the independent errors centered at 0 b_i is therating of movie i

The above model gives us an RMSE of 0.941.This is an improvement over out previous model but we still are far from our target of < 0.8649. We thus introduce one more feature which might help us getting a more focused model. i.e. The user based model.

Model based on User behaviour.

This model is based on the assumption that a user has a certain tendency to rate movies higher or lower than others.thus in the model we introduce a new average user rating term.

The model formula is:

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

where $\hat{\mu}$ is the mean $\varepsilon_{i,u}$ is the independent errors centered at 0 b_i is the rating of movie i b_u is a measure of average rating the user gives to movies

With the above approach we get the RMSE to be around 0.863 which is decent enough.However we can still bring this down a little bit using regularization.We thus proceed towards regularization of the models to reduce their RMSE

Regularization for improving RMSE

Regularization is the process of adding information in order to solve an ill-posed problem or to prevent overfitting.

For the Movie based ands the user based model we perform regularization .Here we introduce a penalty lambda and vary it till we obtain the optimum value.

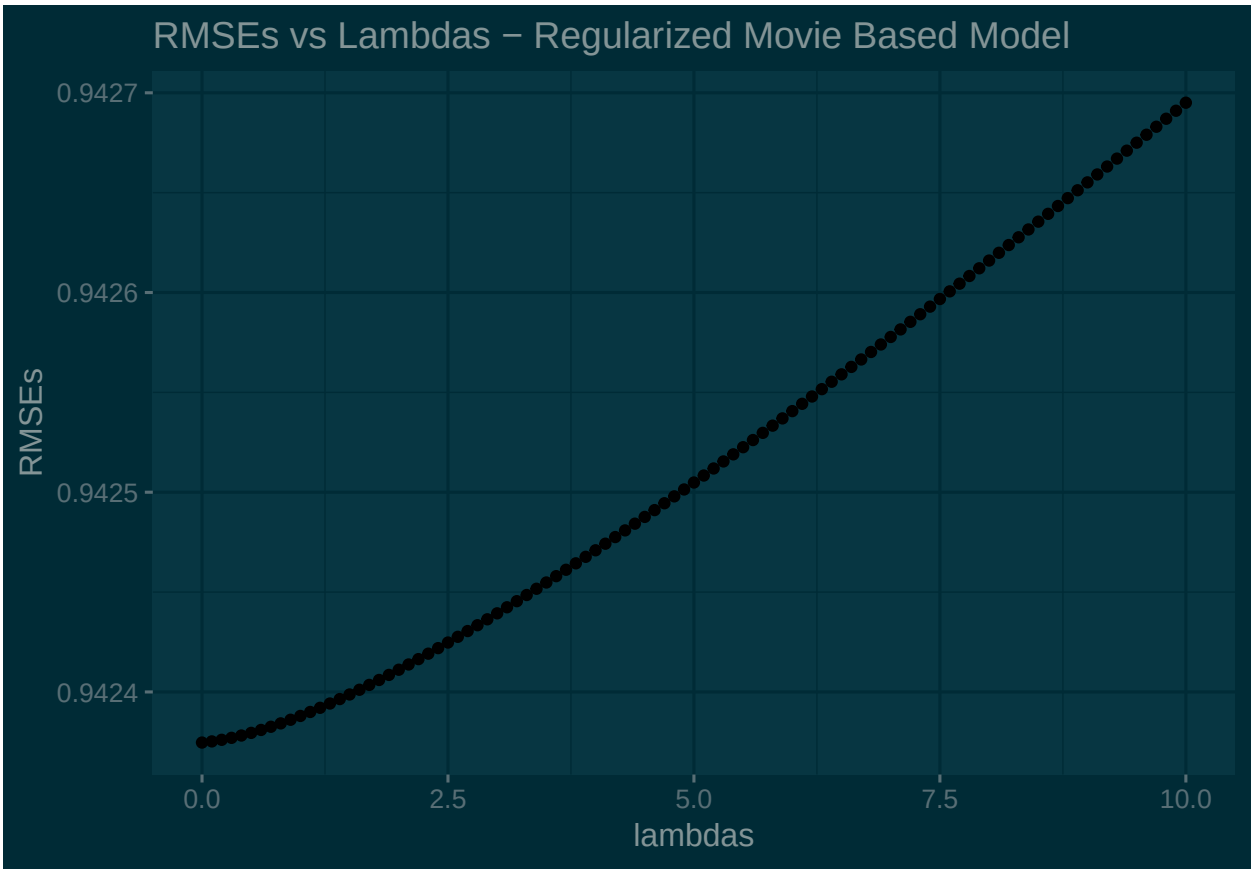
The equation for this operation is as follows:

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

Here we create additional splits of out edx data into training and test sets to compute lambda using regularization. However for optimal lambda we use the entire edx test set.

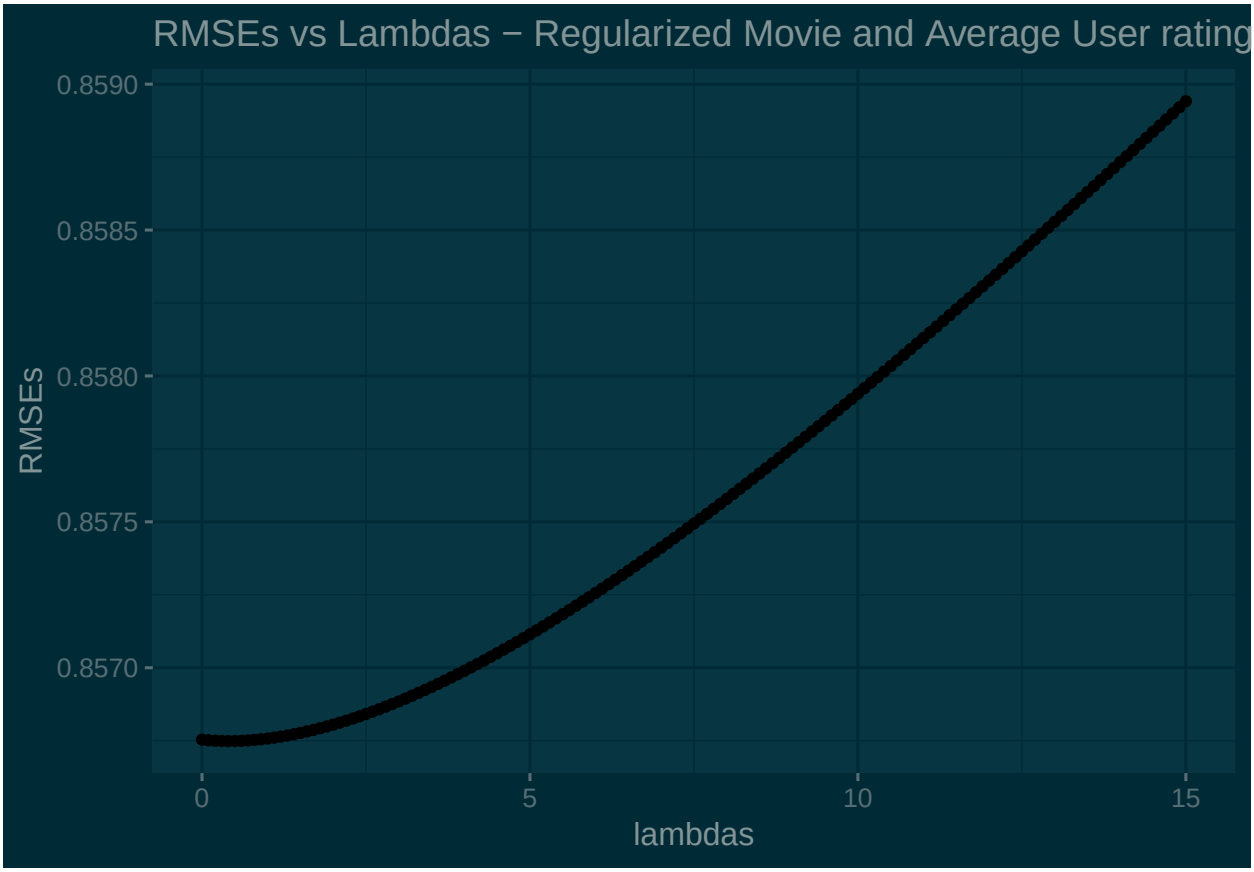
Regularization on Movie-Based Model

Generating a sequence of lambdas



Performing regularization on the Movie rating model gives us the minimum RMSE as 0.9411356. This is a slight improvement on the existing RMSE before regularization but not enough to satisfy out cutoff. Thus we move on to regularizing the Movie and average user model.

Regularization of Movie and Average User rating based Model



Thus our final model which is the model with the average movie rating and the user rating along with Regularization gives us the best RMSE of 0.86355.Though we can stop here, a model with even more granular approach like adding more features - release year , and genre could be even more accurate.

Overall Results

We started out with applying a basic Naive Mean Model only taking into consideration the overall mean of all the movie ratings and tried to predict the movie ratings in validation set. This gave us as very high RMSE of 1.06.

We then moved on to a more complex version of our algorithm, in which we took into account the average Movie rating for every movie to then predict its rating in the validation set.This improved our RMSE from 1.05 to 0.9431.

This hinted us towards an even more granular model in which , along with the average movie rating, the average rating by a user is also used to finally predict a rating that the user would give to a movie. This further improved our RMSE to 0.8646245.

We then attempted to perform Regularization on our models so as to avoid overfitting. And thus we separated our edx data further into training and test data.We then iterated lambda values and came up with the lambda that minimizes the RMSE on out edx test data.We used this lambda to calculate the RMSE on our validation set.

After regularization we obtained RMSE of 0.94327 for Movie based approach and 0.8635597 for our Movie and average user based approach.There was no improvement in this case probably because the edx training data was smaller than the edx data for which we obtained the earlier 0.8645076 RMSE.

model	RMSE
Naive Mean-Baseline Model	1.0612018
Movie-Based Model	0.9439087
Movie+User Based Model	0.8653488
Regularized Movie-Based Model	0.9439087
Regularized Movie and Average User rating based Model	0.8652450

Conclusion

The best result was obtained using the Movie and the user rating combined which achieved an RMSE of 0.8645076.

Through our above analysis we found that adding the Movie rating variable component to our Means based model improved our prediction.Adding a user specific rating model improved it even further and brought our RMSE to accceptable levels.

The regularization marginally improved the RMSE of the movie based model and user based model.The process of regularization helps us confirm the stability of the model by reducing overfitting.

Appendix

Acknowledgements

Sources

1.Wikipedia

2.www.towardsdatascience.com

3.www.analyticsvidhya.com

Initial Code privided by edX

```
##########################################################
# 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")

# 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")

# 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)
write.csv(edx, "edx.csv")
```