# Capstone_Project [1]: MovieLens report

Abedulla M. A. El-Saidy

12/12/2021

## 1. Introduction

Data Science: Capstone, is the final course in the HarvardX Professional Certificate in Data Science. For this project, I will create a movie recommendation system using the MovieLens dataset.

I will create my own recommendation system using all the tools that have been shown throughout the courses in this series. I will use the 10M version of the MovieLens dataset to make the computation a little easier. I will download the MovieLens data and run code provided to generate my datasets. I will train a machine learning algorithm using the inputs to predict movie ratings.

## 2. Loading Data

**#Loading Data from MovieLens Dataset zip file,**

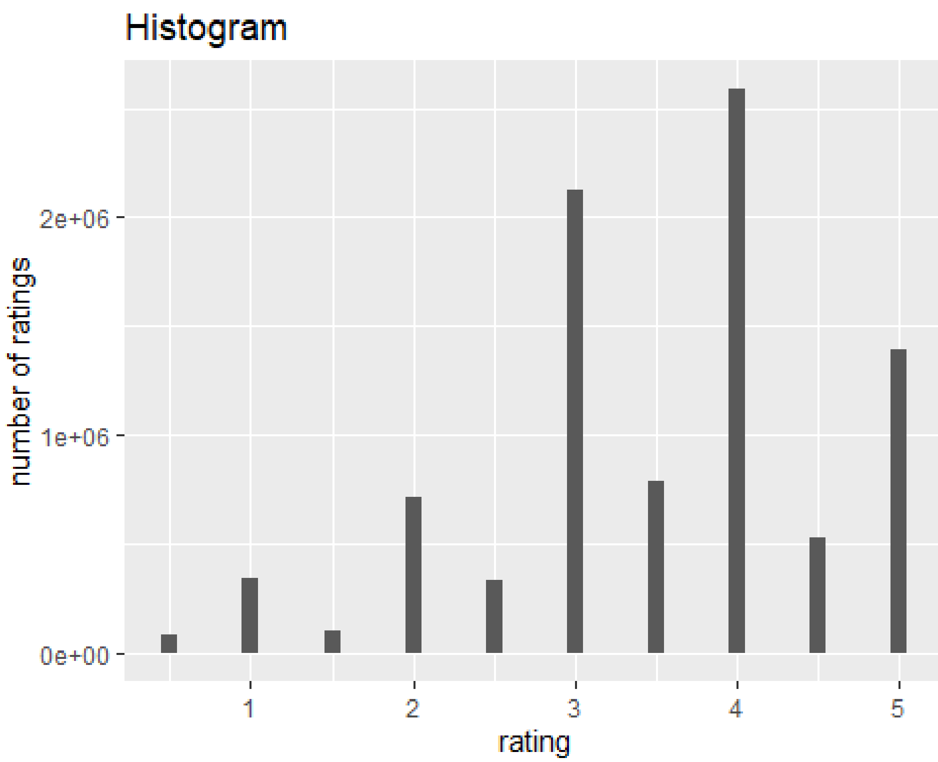
```
Note: this process could take a couple of minutes

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

library(tidyverse) library(caret) library(data.table)


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("::", ", 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, sample.kind="Rounding") # if using R 3.5 or earlier, use 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)
```

#then I store it in file called "gresdy.RData"

## 3. Data exploration

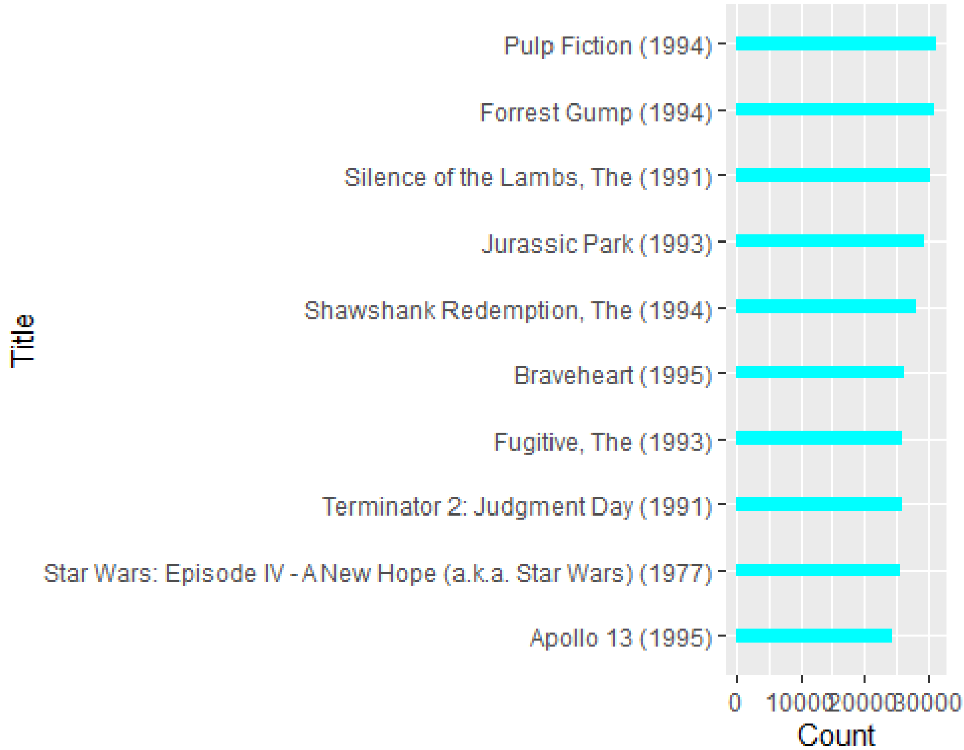
```
#Data exploration


summary(edx)

##      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
##
# histogram of ratings

ggplot(edx, aes(x= edx$rating)) +   geom_histogram( binwidth = 0.1) +
  labs(x="rating", y="number of ratings") +  ggtitle("Histogram")
```



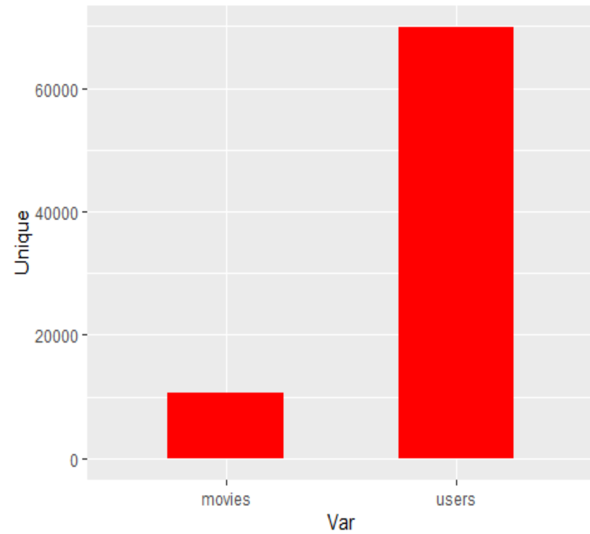

```
# top 10 movies
top_10 <- edx %>%  group_by(title) %>%  summarize(count=n()) %>%
  top_n(10,count) %>%  arrange(desc(count))
print(top_10)

## # A tibble: 10 x 2
##    title                                                        coun
t
##    <chr>                                                        <int
>
##  1 Pulp Fiction (1994)                                          3136
2
##  2 Forrest Gump (1994)                                          3107
9
##  3 Silence of the Lambs, The (1991)                             3038
2
##  4 Jurassic Park (1993)                                         2936
0
##  5 Shawshank Redemption, The (1994)                             2801
5
##  6 Braveheart (1995)                                            2621
2
##  7 Fugitive, The (1993)                                         2599
8
##  8 Terminator 2: Judgment Day (1991)                            2598
4
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 2567
```

```
2
## 10 Apollo 13 (1995)                                              2428
4
```

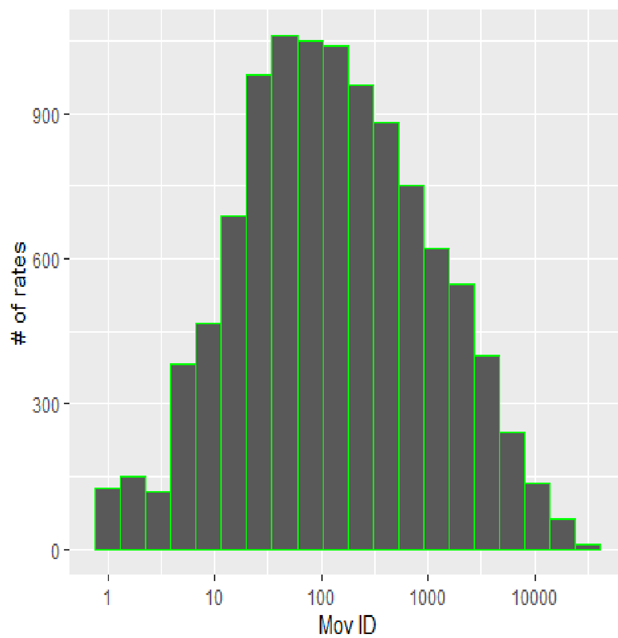
```
ggplot(top_10, aes(x=reorder(title,count),y=count))+geom_bar(stat='iden
tity', fill="cyan", width=0.2)+ coord_flip()+ylab("Count")+xlab("Title"
)
```



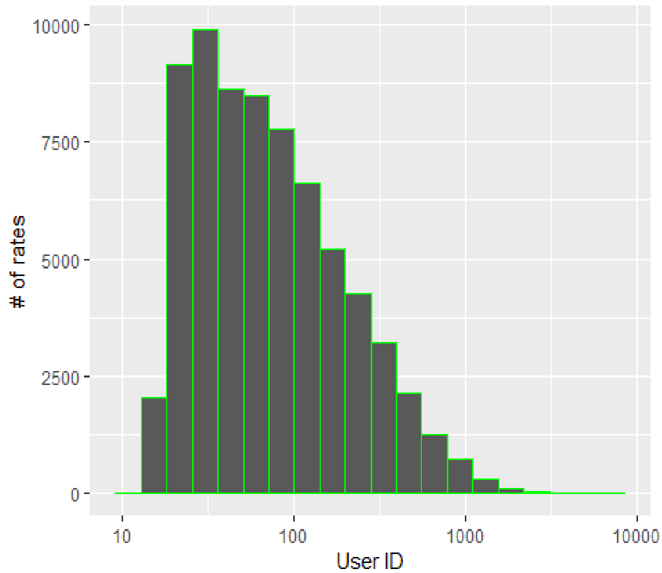

```
#The number of unique values for movieId and userId

unique=data.frame(movies=length(unique(edx$movieId)),users=length(uniqu
e(edx$userId)))
unique

##   movies users
## 1  10677 69878

tunique=as.data.frame( t(unique))
ggplot(tunique, aes(x=row.names(tunique),y=tunique[,1]))+geom_bar(stat=
'identity', fill="red", width=.5)+xlab("Var")+ylab("Unique")
```




```
# histogram  movie Versus ratings
edx %>%  count(movieId) %>%   ggplot(aes(n)) + geom_histogram(bins=20,color="Green")+ scale_x_log10() +xlab("Mov ID")+ylab("# of rates")
```




```
# histogram  User Versus ratings
edx %>%  count(userId) %>%   ggplot(aes(n)) + geom_histogram(bins=20,color="Green")+ scale_x_log10() +xlab("User ID")+ylab("# of rates")
```

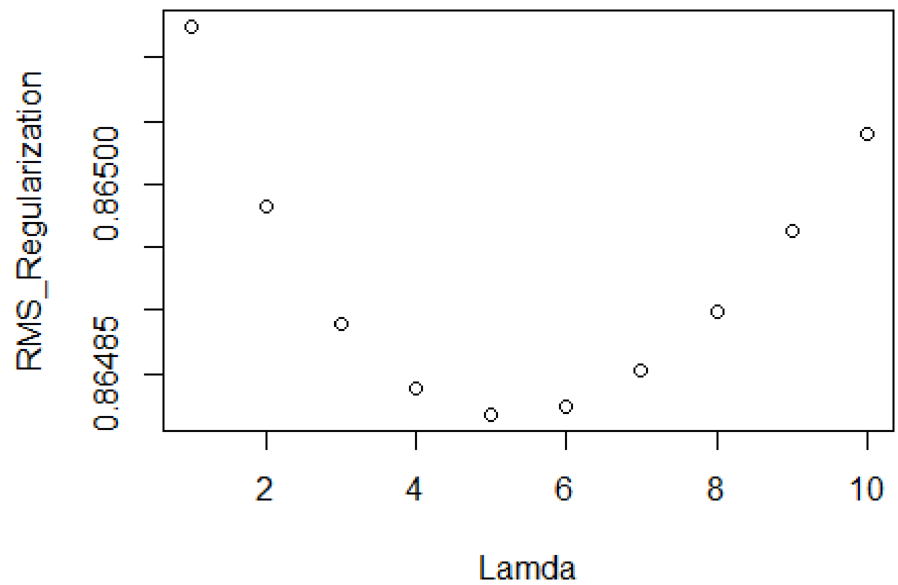



```
#some movies get rated more than others, and some users are more active than others


#  the relation between rating versus  year


time=round_date(as_datetime(edx$timestamp), unit = "year")
timedat=data.frame(time,edx$rating)
timedat %>%  group_by(time) %>% summarize(edx.rating = mean(edx.rating)
) %>% ggplot(aes(time, edx.rating)) +   geom_smooth( colour="red", size
=0.5)

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


```
#When compared to current times, the rating was high in the past.
```

# 4. methods/analysis

```
# the RMSE function  -library(caret)  -> RMSE()

#1.simple methods
#The mean of all movies
meo=mean(edx$rating)
mean_only_RMSE=RMSE(validation$rating,meo)
mean_only_RMSE

## [1] 1.061202

# movie effect
e_mov_mean = edx %>%  group_by(movieId) %>%  summarize(e_mov_avg = mean
(rating-meo))
pred_rat_eachmov = validation %>% left_join(e_mov_mean, by='movieId') %
>% mutate(pr1 = meo + e_mov_avg)
mean_each_movie_RMSE = RMSE(validation$rating,pred_rat_eachmov$pr1)
mean_each_movie_RMSE

## [1] 0.9439087

# user+movie effect
e_user_mean = edx %>% left_join(e_mov_mean, by = "movieId")  %>% group_
by(userId) %>%  summarize(e_user_avg = mean(rating-meo-e_mov_avg))
pred_rat_eachusr = validation %>% left_join(e_mov_mean, by='movieId') %
>%left_join(e_user_mean, by='userId') %>% mutate(pr2 = meo + e_mov_avg
+ e_user_avg)
mean_each_user_RMSE = RMSE(validation$rating,pred_rat_eachusr$pr2)
mean_each_user_RMSE

## [1] 0.8653488

#Regularization
Lamda <- seq(1, 10, 1)
RMS_Regularization <- sapply(Lamda, function(Lam){
  meo=mean(edx$rating)
  e_mov_avg <- edx %>%  group_by(movieId) %>% summarize(e_mov_avg = sum
(rating - meo)/(n()+Lam))
  e_user_mean <- edx %>% left_join(e_mov_avg, by="movieId") %>% group_b
y(userId) %>% summarize(e_user_mean = sum(rating - e_mov_avg - meo)/(n(
)+Lam))
  pred_rat <- validation %>%left_join(e_mov_avg, by = "movieId") %>%lef
t_join(e_user_mean, by = "userId") %>%mutate(pr = meo + e_mov_avg + e_u
ser_mean)

  return(RMSE(validation$rating, pred_rat$pr))
})
plot(Lamda, RMS_Regularization)
```




```
min(RMS_Regularization)

## [1] 0.8648177

#2.Caret Prediction
#sampling
set.seed(1000, sample.kind="Rounding")

## Warning in set.seed(1000, sample.kind = "Rounding"): non-uniform 'Ro
unding'
## sampler used

miniedx=edx[sample(9000055,10000),]

#Split the mini edx
indx = createDataPartition(miniedx$rating, times=1, p=0.8,list=FALSE)
train = miniedx[indx,]
test = miniedx[-indx,]

#Machine Learning Models
#Linear Model
lm = train(rating~userId+movieId+timestamp, data=train, method="lm")

## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind =
NULL) :
## non-uniform 'Rounding' sampler used

lm
```

```
## Linear Regression
##
## 8001 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8001, 8001, 8001, 8001, 8001, 8001, ...
## Resampling results:
##
##   RMSE      Rsquared     MAE
##   1.055194  0.002703584  0.8524619
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
pred1 = predict(lm, test)
lm_RMSE = RMSE(test$rating, pred1 )
lm_RMSE
```

```
## [1] 1.048852
```

```
#Decision Tree
TRE = train(rating~userId+movieId+timestamp, data=train, method="rpart"
)
```

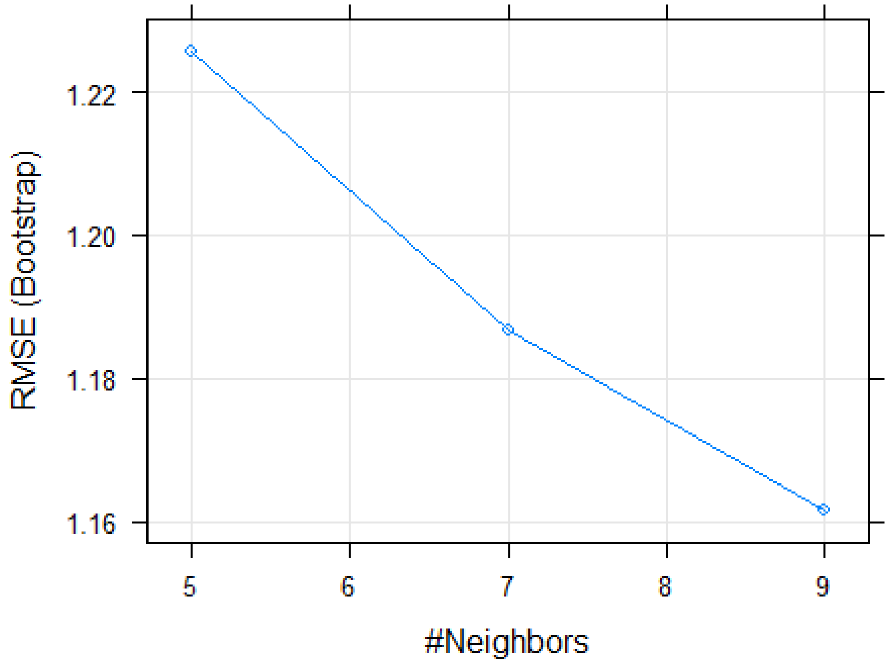
```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind =
NULL) :
## non-uniform 'Rounding' sampler used
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
trainInfo, :
## There were missing values in resampled performance measures.
```

```
(TRE)
```

```
## CART
##
## 8001 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8001, 8001, 8001, 8001, 8001, 8001, ...
## Resampling results across tuning parameters:
##
##   cp           RMSE      Rsquared    MAE
##   0.002365630  1.053217  0.01499636  0.8425199
##   0.002425463  1.052815  0.01501857  0.8425324
##   0.010089754  1.052489  0.01723323  0.8471947
##
```

```
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.01008975.

pred2 = predict(TRE, test)
TRE_RMSE = RMSE(test$rating, pred2)
TRE_RMSE

## [1] 1.049581

#k-Nearest Neighbors
knn = train(rating~userId+movieId+timestamp, data=train, method="knn")

## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind =
NULL) :
## non-uniform 'Rounding' sampler used

knn

## k-Nearest Neighbors
##
## 8001 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8001, 8001, 8001, 8001, 8001, 8001, ...
## Resampling results across tuning parameters:
##
##   k  RMSE      Rsquared      MAE
##   5  1.225627  0.0007518559  0.9704332
##   7  1.186605  0.0006304272  0.9414621
##   9  1.161674  0.0004954929  0.9210902
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 9.

plot(knn)
```




```
pred3 = predict(knn, test)
knn_RMSE = RMSE(test$rating, pred3)
knn_RMSE

## [1] 1.100773
```

# 5. Results

```
Rmse_Result=data.frame(method=c("mean_only","mean_each_movie","mean_eac
h_user","Regularization","Linear Model","Decision Tree","k-Nearest Neig
hbors"  ),RMSE=c(mean_only_RMSE,mean_each_movie_RMSE,mean_each_user_RMS
E,min(RMS_Regularization),lm_RMSE,TRE_RMSE,knn_RMSE))
arrange(Rmse_Result,(RMSE))

##                method      RMSE
## 1      Regularization 0.8648177
## 2      mean_each_user 0.8653488
## 3     mean_each_movie 0.9439087
## 4        Linear Model 1.0488522
## 5       Decision Tree 1.0495815
## 6           mean_only 1.0612018
## 7 k-Nearest Neighbors 1.1007733
```

## 6. Conclusion

The final RMSE is 0.8648177. I constructed and tested numerous models and found that Regularization provided the best accuracy.

## 7. References

Irizzary,R.,Introduction to Data Science