Recommendation_project

1- Introduction

For this first Project in Capstone we have the goal to develop an recommendation system considering the dataset Movielens, more precisely an subset with 10M.It is a problem with large dataset.The dataset was extracted from EDX and the data wrangling was initiated for us.

Define the variables Identification of Movie and User: MovieID and UserId rating - rating of the movies timestamp -> a number of the date tht the user have rated the film

Data -> Subsets:edx (train) and final_holdout_test (test)

1.1-Loading Libraries

1.2-Data Preprocessing

Data Loading

Code that was provided from EDX to download the dataset and start the preprocessing, as split the dataset and organize the data with joins. The principal dataset is called movielens and it wil be separated in edx (training set) and final_holdout_test (test set without some rows that do not exist in edx)

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

1.3- QUIZ from EDX

This entire chunk was developed to answer the QUIZ for this Capstone.

```
zero <- edx%>%filter(rating=="0")
three <- edx%>%filter(rating=="3")

ids <- edx %>%
  summarize(n_users = n_distinct(userId),
  n_movies = n_distinct(movieId))

#How many users?
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
#Q4- How many different movies are in the edx dataset?
n_distinct(edx$movieId)
```

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

```
# Q5 - str_detect
genres = c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(g) {sum(str_detect(edx$genres, g))})
```

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

```
#Q6 - Q8
edx %>% group_by(movieId,title) %>% summarize(count = n()) %>%arrange(desc(count))
```

```
## `summarise()` has grouped output by 'movieId'. You can override using the
## `.groups` argument.

## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        count
##      <int> <chr>                                                        <int>
##  1     296 Pulp Fiction (1994)                                          31362
##  2     356 Forrest Gump (1994)                                          31079
##  3     593 Silence of the Lambs, The (1991)                             30382
##  4     480 Jurassic Park (1993)                                         29360
##  5     318 Shawshank Redemption, The (1994)                             28015
##  6     110 Braveheart (1995)                                            26212
##  7     457 Fugitive, The (1993)                                         25998
##  8     589 Terminator 2: Judgment Day (1991)                            25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                             24284
## # i 10,667 more rows
```

```
edx%>%group_by(rating)%>%summarize(count=n())%>%top_n(10)%>%arrange(desc(count))
```

```
## Selecting by count

## # A tibble: 10 x 2
##    rating   count
##     <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
```

#2-Analysis ##2.1 Preprocessing Handling Missing Data - remove the entire rows that contain any NA Edit columns: timestamp as date and then as Year

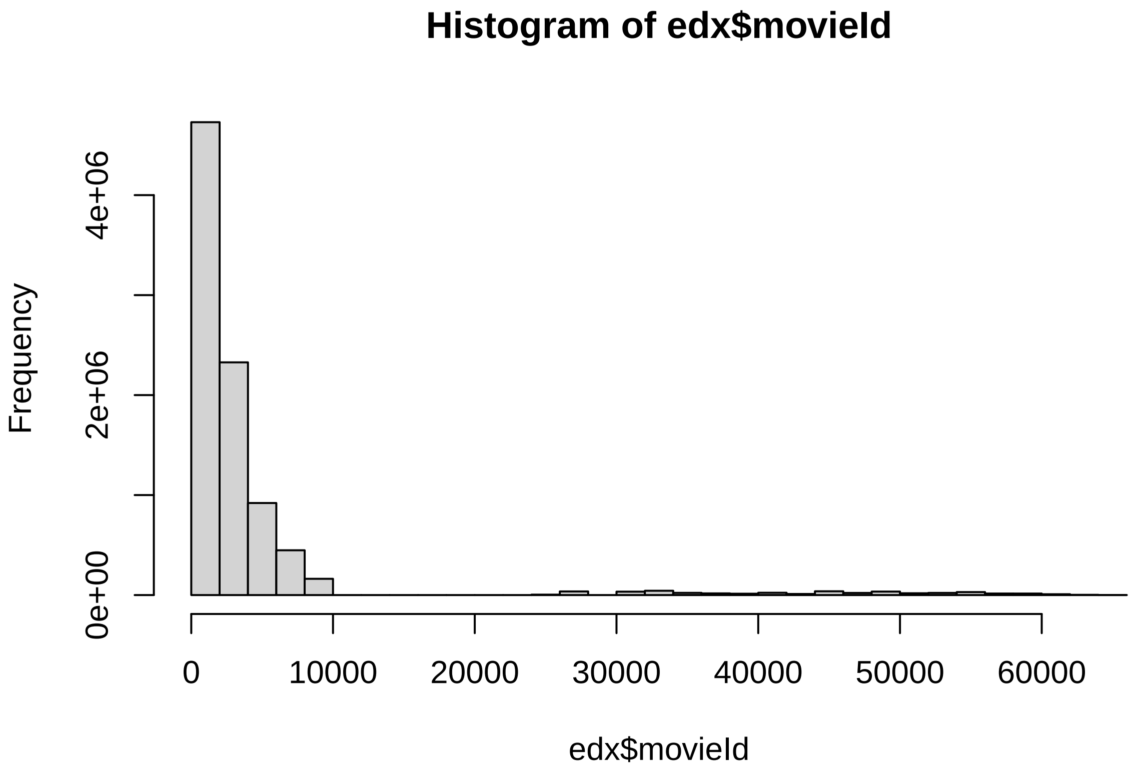
```
## Missing Data
edx <- edx %>% drop_na()
## Create  the Year Column for edx and final_holdout_test
edx <- edx%>%mutate(date= as_datetime(timestamp))%>%mutate(Year = year(date))

final_holdout_test <- final_holdout_test%>%mutate(date= as_datetime(timestamp))%>%mutate(Year = year(da
```

2.2 Dataviz - Exploratory Data Analysis (EDA)

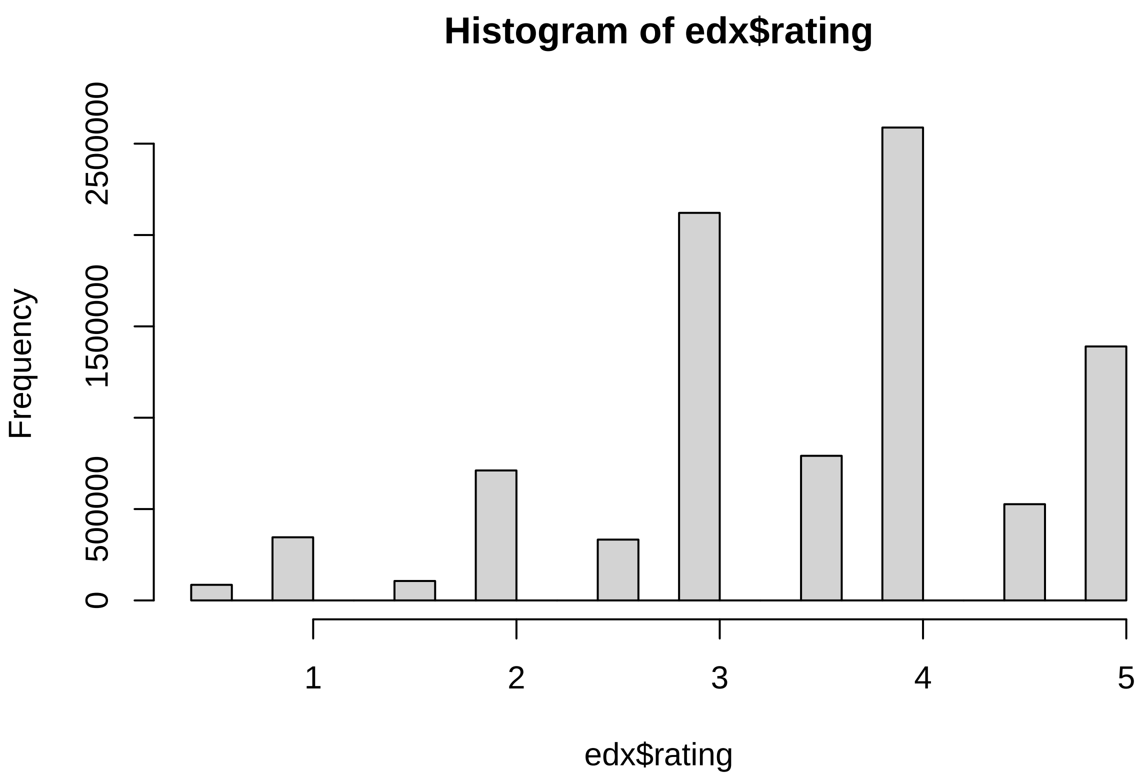
```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres                date Year
## 1                 Comedy|Romance 1996-08-02 11:24:06 1996
## 2          Action|Crime|Thriller 1996-08-02 10:58:45 1996
## 4  Action|Drama|Sci-Fi|Thriller 1996-08-02 10:57:01 1996
## 5        Action|Adventure|Sci-Fi 1996-08-02 10:56:32 1996
## 6 Action|Adventure|Drama|Sci-Fi 1996-08-02 10:56:32 1996
## 7        Children|Comedy|Fantasy 1996-08-02 11:14:34 1996
```

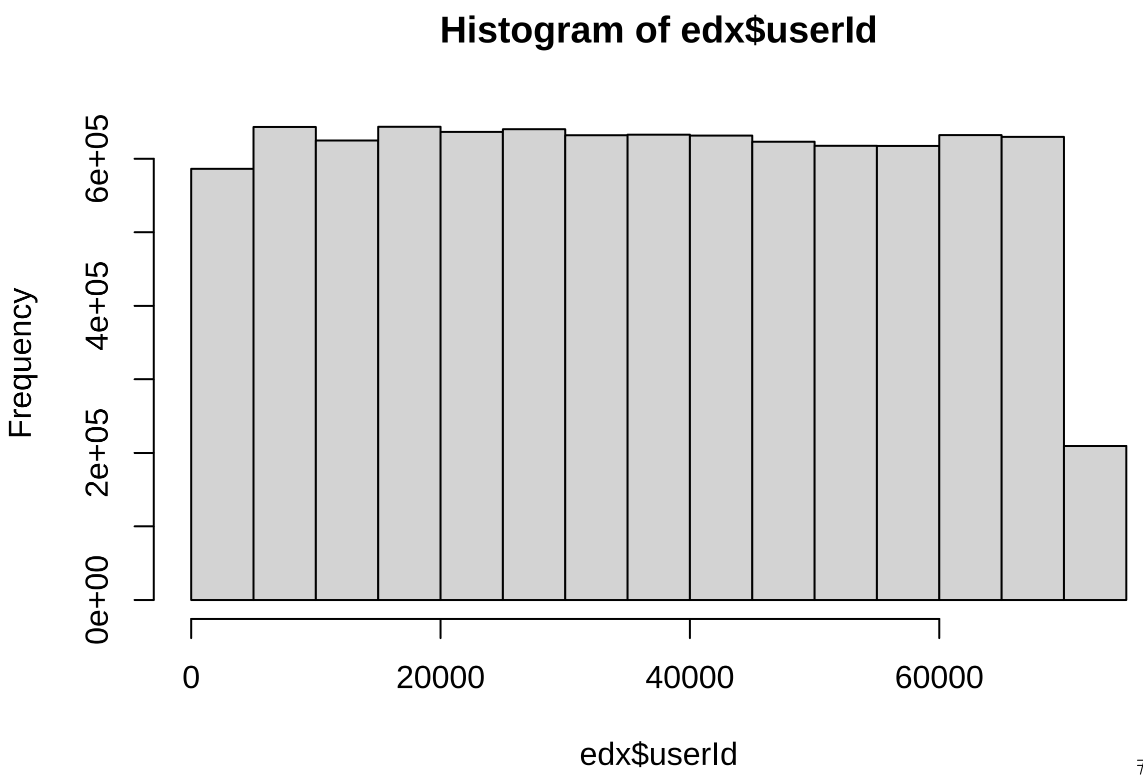
```r
hist(edx$movieId)
```



```r
hist(edx$rating)
```

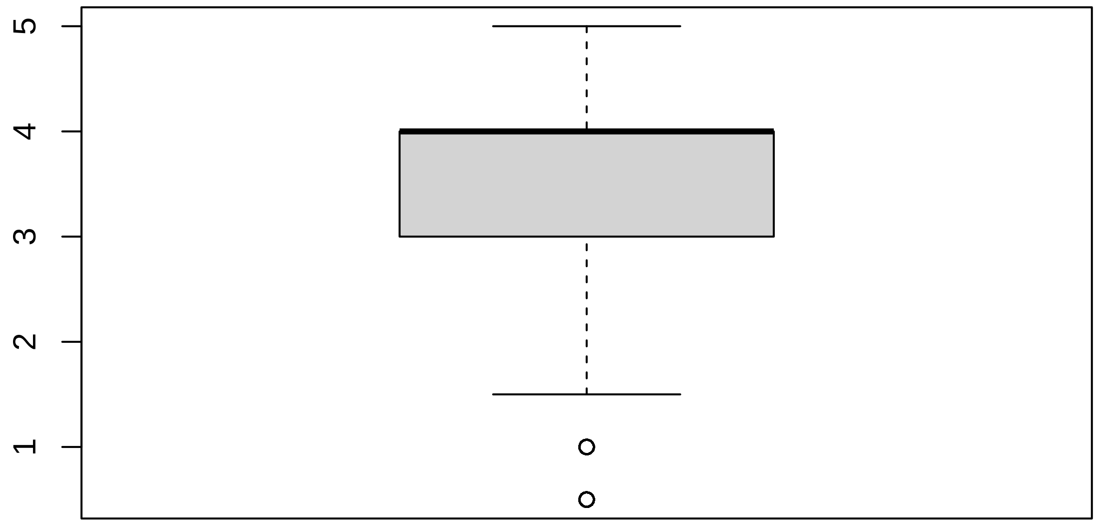


```
hist(edx$userId)
```



###2.2.1Box-
plot -rating

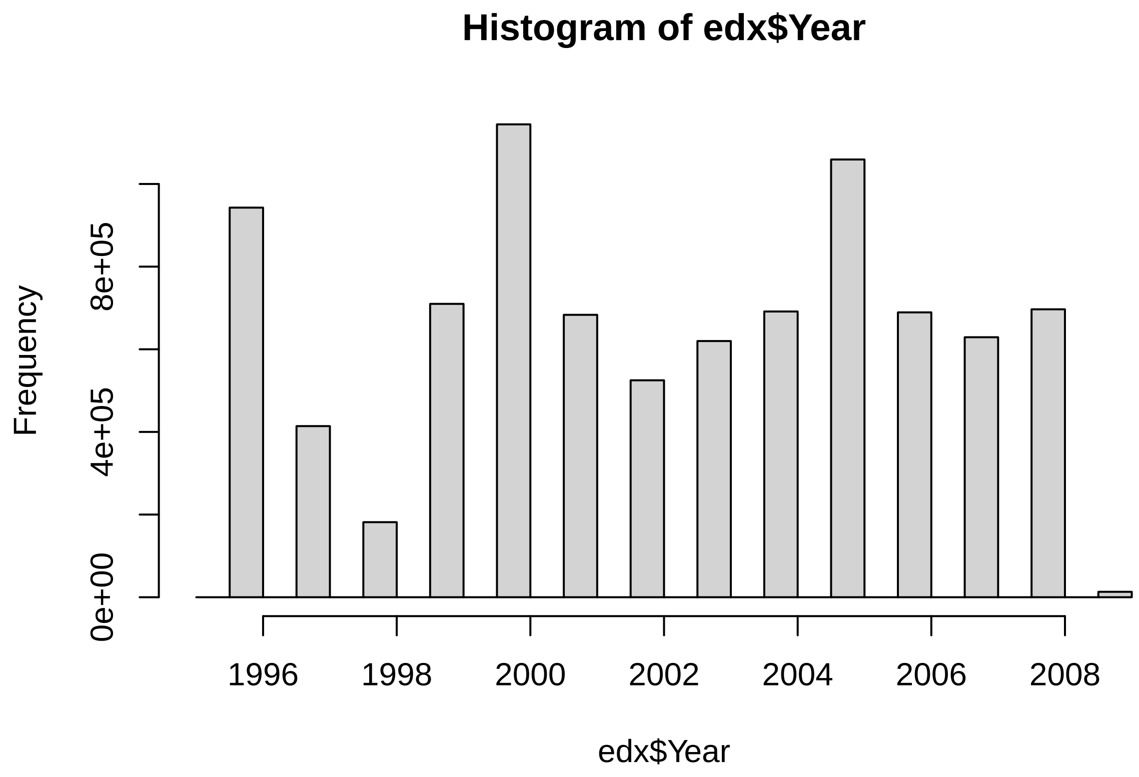
```r
boxplot(edx$rating, data=movielens)
```



2.3 - Method: Evaluate the YEAR influence and try the KNN(method: k-Nearest Neighbors)

One idea was to see if the year was an variable with some importance and the idea was to test a very small subset. EXAMPLE of a year ->CHOOSE A YEAR with => 2009 and try a different method as knn

```r
hist(edx$Year)
```



```r
edx %>% group_by(Year) %>%  summarize(n = n(), avg_rating = mean(rating), se_rating = sd(rating)/sqrt(n
```

```
## # A tibble: 15 x 4
##     Year       n avg_rating se_rating
##    <dbl>   <int>      <dbl>     <dbl>
##  1  1995       2       4      1
##  2  1996  942772       3.55   0.00102
##  3  1997  414101       3.59   0.00157
##  4  1998  181634       3.51   0.00266
##  5  1999  709893       3.62   0.00133
##  6  2000 1144349       3.58   0.00105
##  7  2001  683355       3.54   0.00135
##  8  2002  524959       3.47   0.00154
##  9  2003  619938       3.47   0.00135
## 10  2004  691429       3.43   0.00125
## 11  2005 1059277       3.44   0.00100
## 12  2006  689315       3.47   0.00123
## 13  2007  629168       3.47   0.00130
## 14  2008  696740       3.54   0.00119
## 15  2009   13123       3.46   0.00872
```

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

edx_2009 <- edx%>%filter(Year ==2009)
fit_knn2009 <- knn3(rating ~ movieId, data= edx_2009)
final_holdout_test_2009 <- final_holdout_test%>%mutate(date= as_datetime(timestamp)) %>%mutate(Year = ye
y_hat2009 <- predict(fit_knn2009, final_holdout_test_2009)
RMSE_2009 <- RMSE(final_holdout_test_2009$rating,y_hat2009)
RMSE_2009
```

```
## [1] 3.506498
```

```
# The value was really high ==3.506498
# When I tried to choose a year like 2008, the same code was not able to run.Too many rows.
```

#2.4 - Basic tests (Reference:as EDX explained in textbook)

The idea was to exploring solutions considering the edx solutions presented at Chapter 33 and then add two different variables (year and week)

There were applied 5 models to this dataset: -> simple mean -> consider only the movieID variable, with a simplified calculation for the model fit <- lm(rating ~ as.factor(movieId), data = edx) -> Add userId-consider only the movieID variable and userID NEW -> Add Year (it was not a good idea) -> Add Week (it was a possible variable)

##2.4.1-EDX BASIC -> THE SIMPLE MEAN

```
mu_hat<- mean(edx$rating, rm.NA=TRUE)
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))}

mu_simple <- RMSE(final_holdout_test$rating,mu_hat)
# The mu_simple was 1.061202
```

##2.4.2-EDX BASIC -> as.factor(movieId)

```
mu <- mean(edx$rating)
movie_avgs <- edx %>%group_by(movieId) %>% summarize(b_i = mean(rating - mu))

predicted_ratings1 <- mu + final_holdout_test %>% left_join(movie_avgs, by='movieId') %>% pull(b_i)
```

```
RMSE(predicted_ratings1, final_holdout_test$rating)
```

```
## [1] 0.9439087
```

```
# The RMSE was 0.9439087
```

##2.4.3-EDX BASIC -> as.factor(movieId) and as.factor(userId)

```
user_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>% group_by(userId) %>%summarize(b_u = mean(ra

predicted_ratings2 <- final_holdout_test%>% left_join(movie_avgs, by='movieId') %>% left_join(user_avgs
RMSE(predicted_ratings2, final_holdout_test$rating)
```

```
## [1] 0.8653488
```

```
#The RMSE was 0.8653488. It is better than the two others.
```

#3- Results

##3.1- Model1-Add YEAR

Is the YEAR an important variable? as.factor(movieId),as.factor(userId) and as.factor(Year)

```
year_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>% group_by(Year) %>%summarize(bu2 = mean(rati

predicted_ratings3 <- final_holdout_test%>% left_join(movie_avgs, by='movieId') %>% left_join(user_avgs

RMSE(predicted_ratings3, final_holdout_test$rating)
```

```
## [1] 0.8677787
```

```
#The RMSE was 0.8677787. As it was a little higher then the method with just the two variable movieId a
```

##3.2-Model2 -Add WEEK as variable. Is the WEEK of the avaliation an important variable? as.factor(movieId),as.factor(userId) and as.factor(week)

```
#Include the week column
edx <- edx %>% mutate(week= week(date))
final_holdout_test <- final_holdout_test%>% mutate(week= week(date))

# Try the model with
week_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>% group_by(week) %>%summarize(bu_week = mean

predicted_ratings3 <- final_holdout_test%>% left_join(movie_avgs, by='movieId') %>% left_join(user_avgs

RMSE(predicted_ratings3, final_holdout_test$rating)
```

```
## [1] 0.8656555
```

```
# The RMSE was 0.8656555 a little lower than with just movieID and userId. In this case, the week can b
```

#3.3- Recommerderlab- Comparing with new methodologies

In order to begin a study and try to understand some new approaches, after have understood the basic ideas of modeling a recommendation problem. These two references below explained how to start with the package recommederlab. The first orientation in this kind of problem was to use the: User-based collaborative filtering (UBCF).

References: 1) RDocumentation- R package recommenderlab - Lab for Developing and Testing Recommender Algorithms(https://www.rdocumentation.org/packages/recommenderlab/versions/1.0.6)

2) Package 'recommenderlab' - Title Lab for Developing and Testing Recommender Algorithms - (https://cran.r-project.org/web/packages/recommenderlab/recommenderlab.pdf)

```
library(recommenderlab)
edx_Matrix <- as(edx,"realRatingMatrix")
model <- Recommender(edx_Matrix,method="UBCF")
final_holdout_test_Matrix <- as(final_holdout_test,"realRatingMatrix")
```

#4-Conclusion

This Capstone was helpful in trying to solve a problem with large datasets. That showed us that the approaches are not simple and we can not use all the modeling that we have learned. I've tried some other modeling approaches, but it was really difficult and nothing worked.

Finally, in this work , I've tried to use the EDX methodologies presented in Chapter 33 and added other two hypotheses: Does the date of the rating help the prediction? To answer this I have tried to insert Year and week in the models and the compared the result in the loss.The week variable helps to reduce the loss (RMSE).A future work in this case would be study some Regularization issues to opitimaze the model.

Also, for future works, I have started to study "recommederlab" package, and it was easy to develop the model.But I haven't managed yet to predict or evaluate the results. I send the functions here, cause maybe someone would like to finish the challenge and find my error.

4.1- Future Works

```
# The 3 commands listed below are not working. But I will try to understand and I think this kind of pa

#evaluate(model, final_holdout_test_Matrix)

#predictions <- predict(model,final_holdout_test_Matrix, type="rating")

## predictions for the test data
#predict(model, getData(e, "known"), type="topNList", n = 10)
```