Miguel_Sanchez_Movie_Lens_report_rmd_final

Miguel Sanchez

September 05-2020

Context

Machine Learning is a subset of Data Science and it´s becoming a strategic piece of digital transformation processes.

Predictive algorithms provide additional insights to make better decisions and will enable proactive actions on a prticular business pain point.

I have implemented a system to predict movie ratings, using provided data sets (Movie Lens) that includes data regarding users, ratings and movies variables.This exercise is intended to demonstrate the usage and power of predictive algorithms.

This report is composed of four parts: the context has provided an introduction and presented the problem, the summary describes the dataset and some transformations performed to split the training/test set; the methods describes the model and its implementation in the attached R file; finally the conclusion shares the results.

##Summary

EDX Data set structure and sample data

```r
head(edx)
```

```
##    userId movieId rating timestamp                         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
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

Data Set (sample) with 6 variables

```r
cat("Train set dimension :",dim(edx))
```

```
## Train set dimension : 9000055 6
```

Data set with 9.000.055 records and 6 variables

```r
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ 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 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
##  - attr(*, ".internal.selfref")=<externalptr>
```

2 categorical and 4 continuous variables

#Number of unique movies

```r
unique_movies<- edx$movieId %>% unique() %>% length()
unique_movies
```

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

#Number of unique users

```r
unique_users<- edx$userId %>% unique() %>% length()
unique_users
```

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

#Movies with most ratings

```r
edx %>% group_by(movieId) %>%
  summarise(n_ratings=n(), title=first(title)) %>%
  top_n(10, n_ratings)
```

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

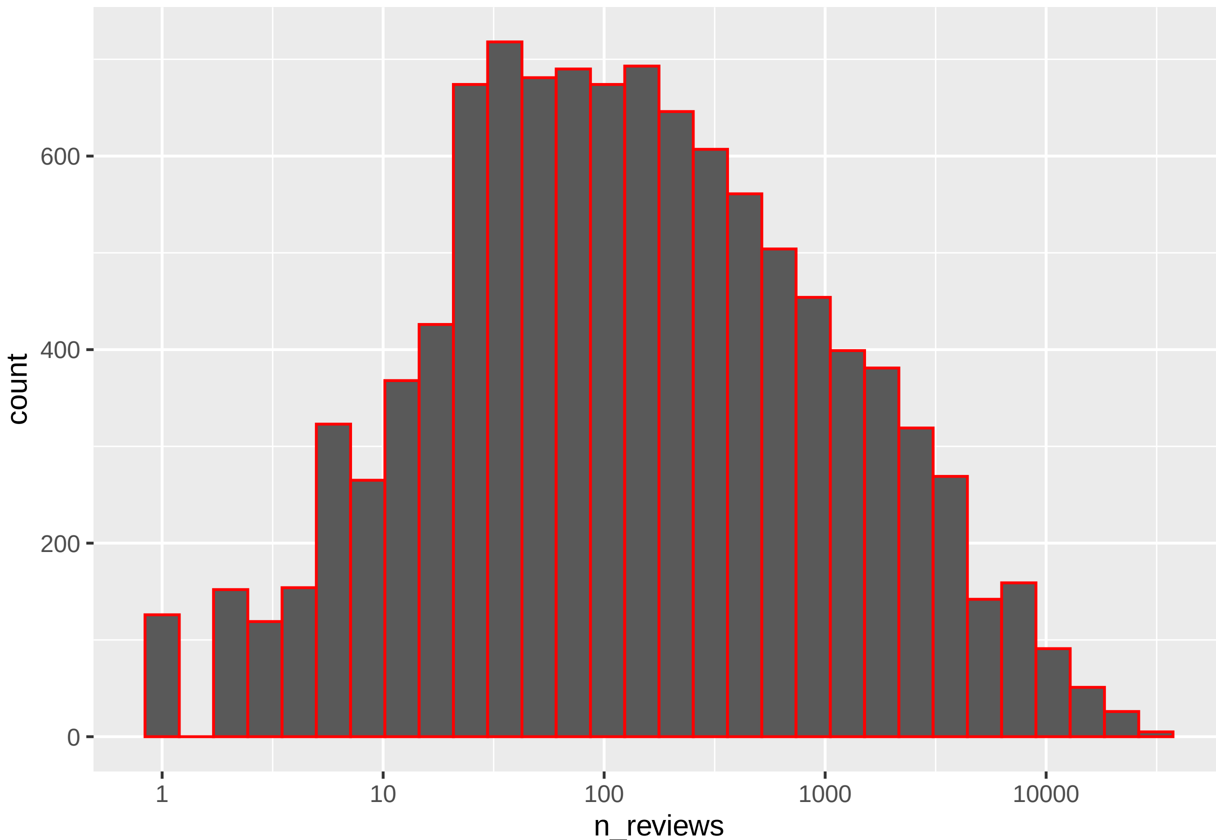
```
## # A tibble: 10 x 3
##    movieId n_ratings title
##      <dbl>     <int> <chr>
##  1     110     26212 Braveheart (1995)
##  2     150     24284 Apollo 13 (1995)
##  3     260     25672 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (197~
##  4     296     31362 Pulp Fiction (1994)
##  5     318     28015 Shawshank Redemption, The (1994)
##  6     356     31079 Forrest Gump (1994)
##  7     457     25998 Fugitive, The (1993)
##  8     480     29360 Jurassic Park (1993)
##  9     589     25984 Terminator 2: Judgment Day (1991)
## 10     593     30382 Silence of the Lambs, The (1991)
```

#Histogram of number of reviews for each movie

```r
edx %>%
  group_by(movieId) %>%
  summarise(n_reviews=n()) %>%
  ggplot(aes(n_reviews)) +
  geom_histogram(color="red") +
  scale_x_log10()
```

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

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



###Data Wrangling

Data sets Management: In order to deal with test and train data, the EDX data set will be splitted; considering 70% for training and 30% for test.

```r
set.seed(1, sample.kind="Rounding")
```

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

```r
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.3, list = FALSE)
train_data_set <- edx[-test_index,]
temporal <- edx[test_index,]
```

```
test_data_set <- temporal %>%
  semi_join(train_data_set, by = "movieId") %>%
  semi_join(train_data_set, by = "userId")

borrar <- anti_join(temporal, test_data_set)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
train_data_set <- rbind(train_data_set, borrar)

rm(test_index, temporal, borrar)
```

#Checking number of records for test data set - 30%

```
dim(test_data_set)
```

```
## [1] 2699928       6
```

#Checking number of records for training data set - 70%

```
dim(train_data_set)
```

```
## [1] 6300127       6
```

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

For model evaluation we will be using RMSE

##Methods 1 Machine Learning Model: A linear model will be used for prediction;only Movie and User variables are being considered (due to performace in my laptop) movie effect = bi & bu = user effect $\hat{y} = \mu + b_i + b_u + \epsilon_{u,i}$

For comparison purposes I will use the mean as the first prediction and calculate the RMSE (only Mu) $\hat{y} = \mu + \epsilon_{u,i}$

```
mu<- mean(train_data_set$rating)
rmse_initial<-sqrt(mean((test_data_set$rating - mu)^2))
rmse_initial
```

```
## [1] 1.060597
```

I will try now with movie effect –> bi $\hat{y} = \mu + b_i + \epsilon_{u,i}$

```
bi<- train_data_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

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

Predicting the rating using mu + bi

```
y_hat_bi <- mu + test_data_set %>%
  left_join(bi, by = "movieId") %>%
  .$b_i
#y_hat_bi
```

Calculate the RMSE using mu + bi

```
rmse_mu_bi<- sqrt(mean((test_data_set$rating - y_hat_bi)^2))
rmse_mu_bi
```

```
## [1] 0.9443758
```

Including the user effect –>bu $\hat{y}_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$

```
bu <- train_data_set %>%
  left_join(bi, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

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

Predicting the rating using mu + bi + bu

```
y_hat_bi_bu <- test_data_set %>%
  left_join(bi, by='movieId') %>%
  left_join(bu, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
#y_hat_bi_bu
```

Calculate the RMSE using mu + bi +bu

```
rmse_mu_bi_bu<- sqrt(mean((test_data_set$rating - y_hat_bi_bu)^2))
rmse_mu_bi_bu
```

```
## [1] 0.8673028
```

##Observation Based on the results I have gotten from the algorithm, the RMSE has been reduced reduced thus the prediction has improved

```
EVALUATION<- c("RMSE_MU", "RMSE_MU+BI", "RMSE_MU+BI+BU")
#result
VALUES<- c(rmse_initial, rmse_mu_bi, rmse_mu_bi_bu)
#result2
Result3<- data.frame(EVALUATION, VALUES)
Result3
```

```
##      EVALUATION    VALUES
## 1       RMSE_MU 1.0605972
## 2    RMSE_MU+BI 0.9443758
## 3 RMSE_MU+BI+BU 0.8673028
```

It´s now time to process against the validation data set

```
mu_v2<- mean(train_data_set$rating)
rmse_initial_v2<-sqrt(mean((validation$rating - mu)^2))
rmse_initial_v2
```

```
## [1] 1.061202
```

```
bi_v2<- train_data_set %>%
  group_by(movieId) %>%
  summarize(b_i_v2 = mean(rating - mu))
```

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

```
y_hat_bi_v2 <- mu_v2 + validation %>%
  left_join(bi_v2, by = "movieId") %>%
  .$b_i_v2
#y_hat_bi_v2

rmse_mu_bi_v2<- sqrt(mean((validation$rating - y_hat_bi_v2)^2))
rmse_mu_bi_v2
```

```
## [1] 0.9441642
```

```
bu_v2 <- train_data_set %>%
  left_join(bi_v2, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u_v2 = mean(rating - mu_v2 - b_i_v2))
```

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

```
y_hat_bi_bu_v2 <- validation %>%
  left_join(bi_v2, by='movieId') %>%
  left_join(bu_v2, by='userId') %>%
  mutate(pred = mu_v2 + b_i_v2 + b_u_v2) %>%
  .$pred
#y_hat_bi_bu_v2

rmse_mu_bi_bu_v2<- sqrt(mean((validation$rating - y_hat_bi_bu_v2)^2))
```

#Preliminary Results Results obtained processing against validation Data Set

```
rmse_mu_bi_bu_v2
```

```
## [1] 0.8672596
```

The results are good, but I believe the RMSE can be improved using the linear model with regularisation ##Methods 2 A linear model with regularisation will be implemented.The first step is to calculate the best lambda

```
lambdas <- seq(from=0, to=10, by=0.25)
```

Calculate the RMSE on each defined LAMBDA, using MU+BI+BU

```
rmses <- sapply(lambdas, function(l){
  # calculate average (MU)
  mu_v3 <- mean(edx$rating)
  # calculate the bi
  b_i_v3 <- edx %>%
    group_by(movieId) %>%
    summarize(b_i_v3 = sum(rating - mu_v3)/(n()+l))
  # calculate the bu
  b_u_v3 <- edx %>%
    left_join(b_i_v3, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_v3 = sum(rating - b_i_v3 - mu_v3)/(n()+l))
  # Run the predictions on validation data set
  predicted_ratings <- validation %>%
    left_join(b_i_v3, by = "movieId") %>%
    left_join(b_u_v3, by = "userId") %>%
    mutate(pred_v3 = mu_v3 + b_i_v3 + b_u_v3) %>%
    pull(pred_v3)
  # Print RMSE on predictions
  return(RMSE(predicted_ratings, validation$rating))
})
```

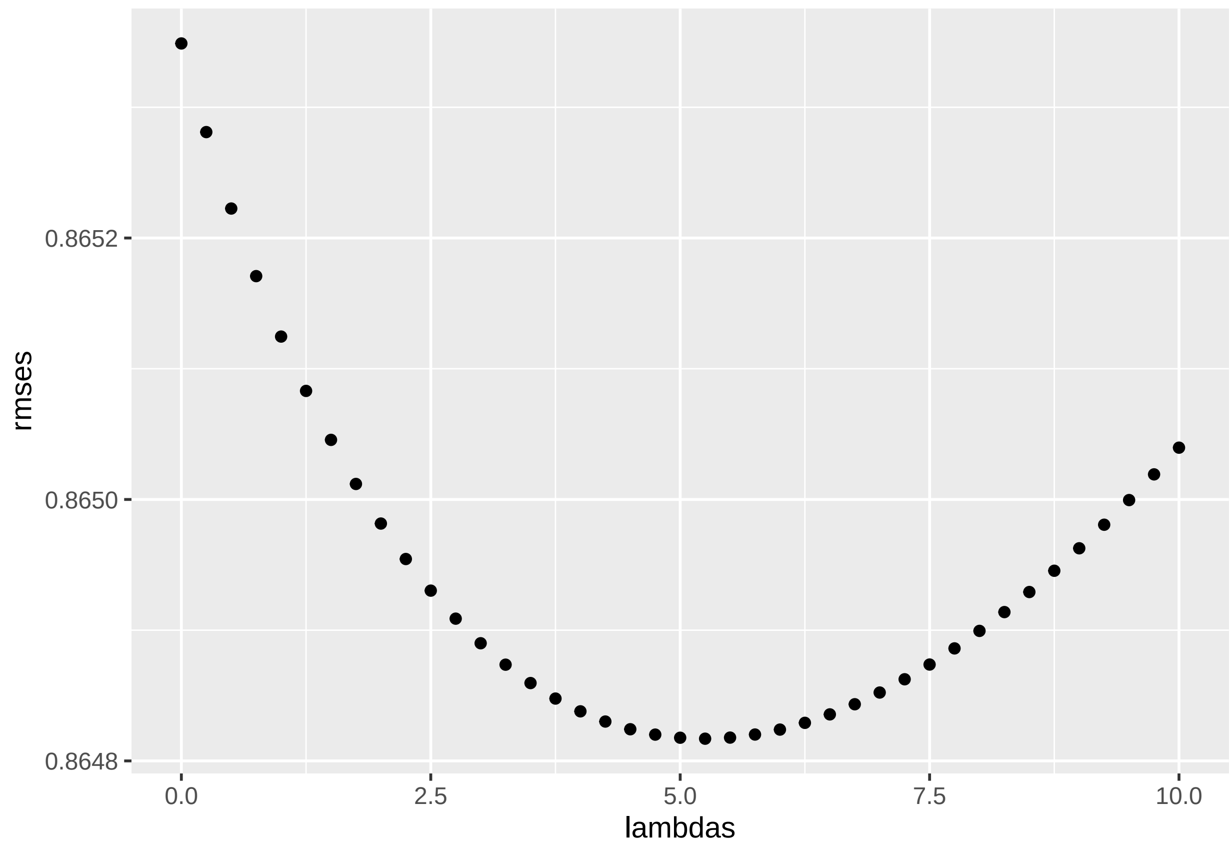
```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()`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

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

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

Plot of RMSE vs lambdas

```
qplot(lambdas, rmses)
```



Minimum RMSE

```
min(rmses)
```

```
## [1] 0.864817
```

LAMBDAS

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

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

Exceution – Model with regularized movie –>BI and user effect –> BU Linear model with the minimizing lambda

```r
# Calculate the mean (MU)
mu_v4 <- mean(edx$rating)
# Calculate the BI
b_i_v4 <- edx %>%
  group_by(movieId) %>%
  summarize(b_i_v4 = sum(rating - mu_v4)/(n()+lambda))
```

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

```r
# Calculate the BU
b_u_v4 <- edx %>%
  left_join(b_i_v4, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u_v4 = sum(rating - b_i_v4 - mu_v4)/(n()+lambda))
```

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

```r
# Run predictions using calculated BI & BU
predicted_ratings <- validation %>%
  left_join(b_i_v4, by = "movieId") %>%
  left_join(b_u_v4, by = "userId") %>%
  mutate(pred = mu_v4 + b_i_v4 + b_u_v4) %>%
  pull(pred)
```

output RMSE of predictions

```r
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.864817
```

##Conslusion / Results The linear model considering movie (BI) anf user effects (BU) created an acceptable resut; however,the result was improved (< 0.86490) using regularisation