EDX Capstone - Movielens project

Alexandre Bort

13 June 2019

About me

Welcome to my report. I'm Alexandre Bort, a french student from ISIMA engineering school. My English isn't very good, but I will try to do my best ! Hope you will enjoy reading it!

Plan of the study

1. Introduction
2. Methods - Analysis
    1. Data overview
    2. Pre-processing
    3. Memory limits
    4. Model
3. Results
4. Conclusion
5. Extra

I. Introduction

Film industry didn't stop growing the last decades. The development of the new technologies provides an easy access to films for everyone. Film producers and especially film distributors such as Netflix or even YouTube try to satisfy their users as best as possible. To do this, they use data analysis approaches. For example, that gives them the ability to improve their film suggestion system.

Here, we will try to **predict the interest that a user could have for a film**. I will minimise the `RMSE` (Root Mean Square Error) criteria as seen in the course.

We will build out study on the ***MoviesLens*** dataset provided by the edx team. The dataset is available this link. The dataset is composed by two main dataframes, one called `edx` for training and one called `validation` for testing our models.

In my work, I will study several approaches. The first one is to predict the rating from a linear regression model. Then, we will build a **decision tree** and finally a **random forest model**.

II. Methods - Analysis

II.1. Data overview

Data:

```r
head(edx)
```

```
##   userId movieId rating timestamp                   title
## 1      1     122      5 838985046                Boomerang
## 2      1     185      5 838983525                 Net, The
## 3      1     231      5 838983392            Dumb & Dumber
## 4      1     292      5 838983421                 Outbreak
## 5      1     316      5 838983392                 Stargate
## 6      1     329      5 838983392 Star Trek: Generations
##                          genres year
## 1                Comedy|Romance 1992
## 2         Action|Crime|Thriller 1995
## 3                        Comedy 1994
## 4  Action|Drama|Sci-Fi|Thriller 1995
## 5       Action|Adventure|Sci-Fi 1994
## 6 Action|Adventure|Drama|Sci-Fi 1994
```

Let's see some **statistic** about our `edx` and `evaluation` dataframe:

```r
summary(edx)
```

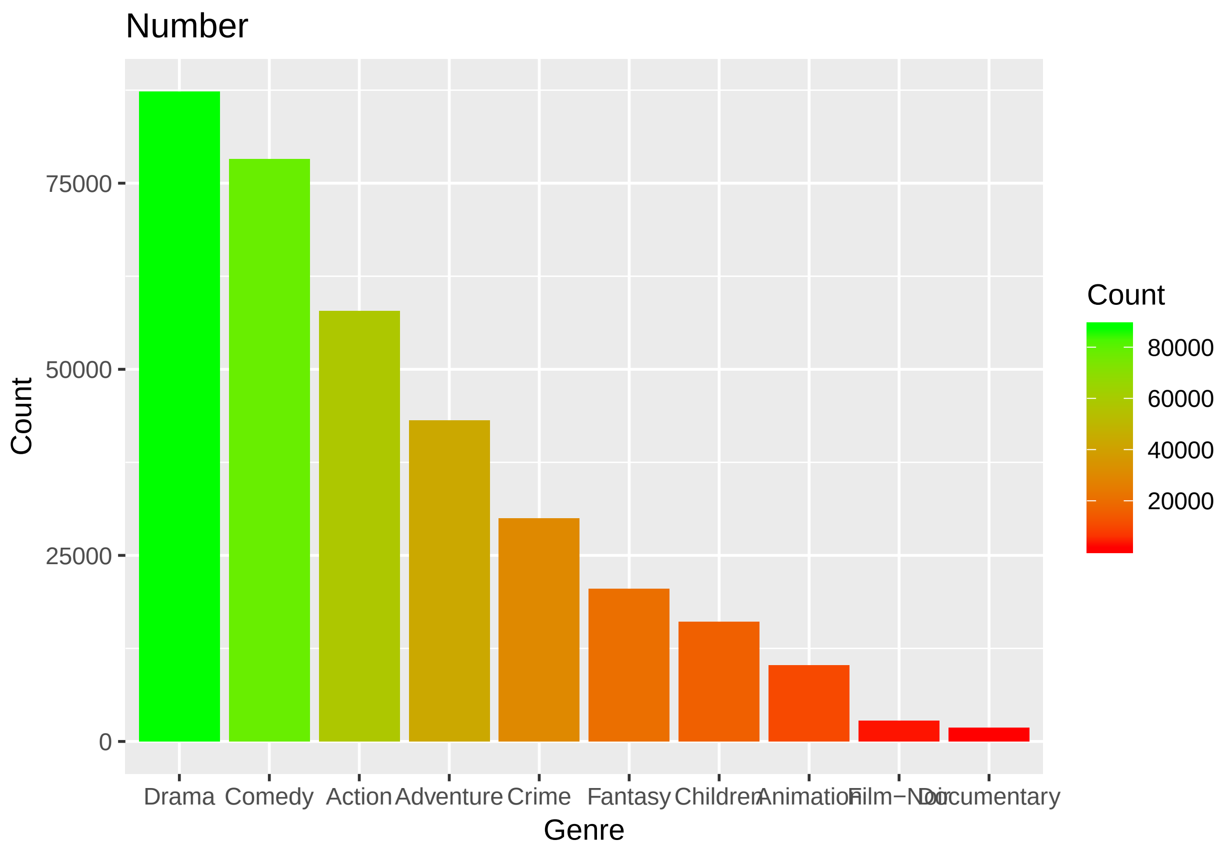
```
##      userId         movieId          rating         timestamp
##  Min.   :   1   Min.   :    1   Min.   :0.500   Min.   :8.281e+08
##  1st Qu.:2177   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.462e+08
##  Median :4234   Median : 1722   Median :4.000   Median :1.024e+09
##  Mean   :4214   Mean   : 3921   Mean   :3.521   Mean   :1.027e+09
##  3rd Qu.:6273   3rd Qu.: 3481   3rd Qu.:4.000   3rd Qu.:1.120e+09
##  Max.   :8269   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres               year
##  Length:1000000     Length:1000000     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

```r
summary(validation)
```

```
##      userId          movieId          rating         timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :8.229e+08
##  1st Qu.:18166   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35801   Median : 1833   Median :4.000   Median :1.035e+09
##  Mean   :35900   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53649   3rd Qu.: 3635   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres               year
##  Length:999991      Length:999991      Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

Let have a look at the most rated genres:

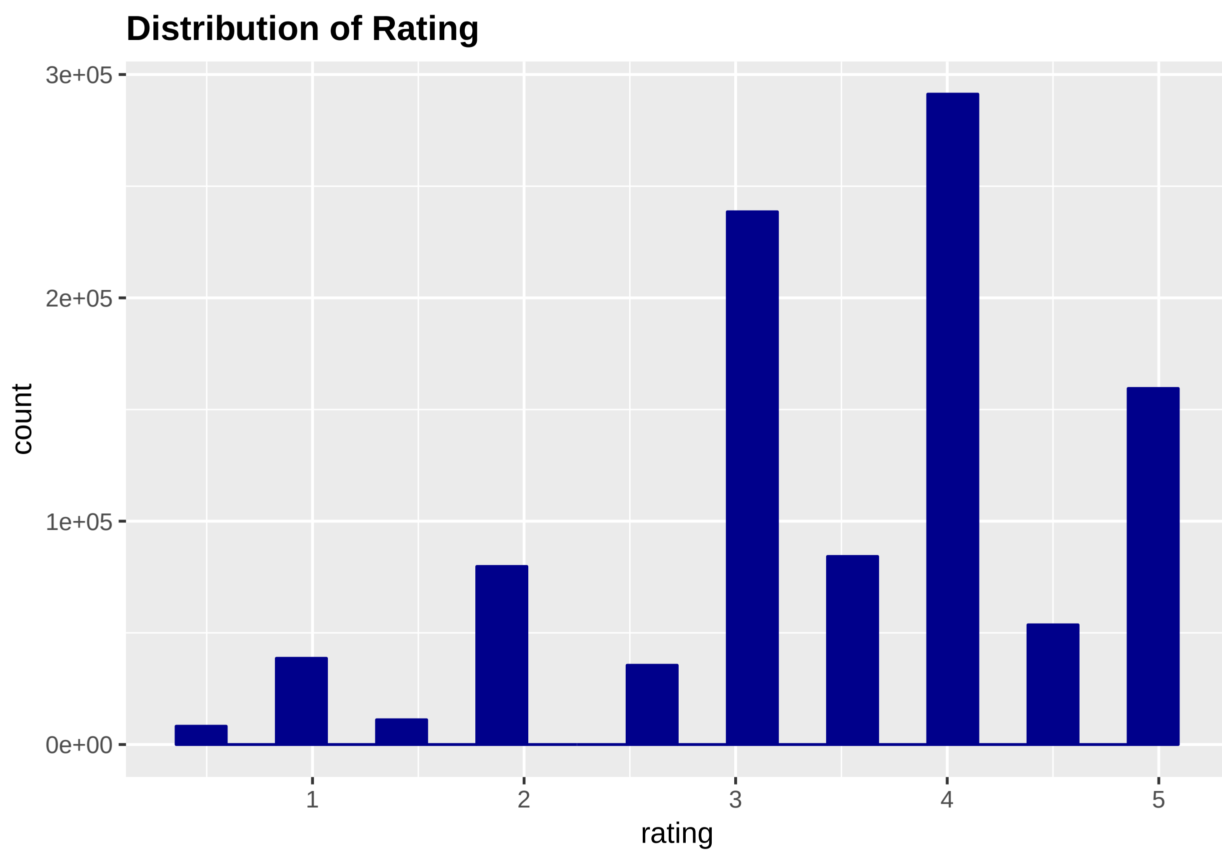
```
plot1
```



People don't seem to be really studious, they widely prefer looking a drama film than a documentary film!

How are they used to rate a film ? The following graph shows that they don't really like comma number rate. Also, most of the time, they seem to like what they have looked.

```
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(bins = 20,color="darkblue", fill="darkblue")+
  ggtitle("Distribution of Rating") +
  theme(plot.title = element_text(color="black", size=13, face="bold"))
```



II.2. Pre-processing

After a brief overview of the dataset, we can notice two things: - The `title` column contains a **year** between parenthesis. We can consider that the interest of customer changes across the year. We can extract them as a new column with the bellow code:

```
df <- edx[seq(1,100000),]
# Extract year as a new column
df$year = as.numeric(substr(df$title, nchar(df$title)-4, nchar(df$title)-1))
df$title = substr(df$title, 0 , nchar(df$title)-7)
head(df)
```

```
##   userId movieId rating timestamp                  title
## 1      1     122      5 838985046              Boomerang
## 2      1     185      5 838983525               Net, The
## 3      1     231      5 838983392           Dumb & Dumber
## 4      1     292      5 838983421               Outbreak
## 5      1     316      5 838983392               Stargate
## 6      1     329      5 838983392 Star Trek: Generations
##                           genres year
## 1                 Comedy|Romance 1992
## 2          Action|Crime|Thriller 1995
## 3                         Comedy 1994
## 4  Action|Drama|Sci-Fi|Thriller 1995
## 5        Action|Adventure|Sci-Fi 1994
```

```
## 6 Action|Adventure|Drama|Sci-Fi 1994
```

- The `genres` column is defined in one string having several genres. The matter with this representation is that we do not distinguish the genre inside the text. One idea is to binarize the genre column in new genre columns. Here's the code:

```r
# Select all genres (uniques)
list_genre <- unique(separate_rows(data = df["genres"], genres, sep = "\\|"))
nb_genres <- nrow(list_genre)
# Create empty matrix
genres_col <- as.data.frame(matrix(0, ncol = nb_genres, nrow = nrow(df)))

# Set col names
colnames(genres_col) <- as.list(list_genre)[[1]]
# Add columns to edx dataframe
df <- cbind(df, genres_col)

rm(genres_col)

set_genre <- function(row){
  genres <- strsplit(row["genres"], "\\|")[[1]]
  row[genres] <- 1
  return(row)
}
df <- as.data.frame(t(apply(df, MARGIN=1, FUN=set_genre)))

# remove genre columns
df <- df[setdiff(names(df), c("genres","title", "timestamp"))]
head(df)
```

```
##   userId movieId rating year Comedy Romance Action Crime Thriller Drama
## 1      1     122    5.0 1992      1       1      0     0        0     0
## 2      1     185    5.0 1995      0       0      1     1        1     0
## 3      1     231    5.0 1994      1       0      0     0        0     0
## 4      1     292    5.0 1995      0       0      1     0        1     1
## 5      1     316    5.0 1994      0       0      1     0        0     0
## 6      1     329    5.0 1994      0       0      1     0        0     1
##   Sci-Fi Adventure Children Fantasy War Animation Musical Western Horror
## 1      0         0        0       0   0         0       0       0      0
## 2      0         0        0       0   0         0       0       0      0
## 3      0         0        0       0   0         0       0       0      0
## 4      1         0        0       0   0         0       0       0      0
## 5      1         1        0       0   0         0       0       0      0
## 6      1         1        0       0   0         0       0       0      0
##   Film-Noir Mystery Documentary IMAX
## 1         0       0           0    0
## 2         0       0           0    0
## 3         0       0           0    0
## 4         0       0           0    0
## 5         0       0           0    0
## 6         0       0           0    0
```

Now that we have exploded the `genre` column, we can see the repartition.

```r
q <- quantile(apply(df[,genre_cols], 2, sum))
q
```

```
##    0%   25%   50%   75%  100%
##    68  4837  7880 20340 43745
```

Some genres are not very representative, so we can group them in a unique column `Other`. The following code groups the column below the first quartile.

```r
column_below_q1 <- genre_cols[q < q[[2]]]

keep <- function(row){
  return (if(sum(row) > 0) 1 else 0)
}
df$other <- apply(df[column_below_q1], 1, keep)
df <- df[, !names(df) %in% column_below_q1, drop=F]
head(df)
```

```
##   userId movieId rating year Romance Action Crime Thriller Sci-Fi
## 1      1     122    5.0 1992       1      0     0        0      0
## 2      1     185    5.0 1995       0      1     1        1      0
## 3      1     231    5.0 1994       0      0     0        0      0
## 4      1     292    5.0 1995       0      1     0        1      1
## 5      1     316    5.0 1994       0      1     0        0      1
## 6      1     329    5.0 1994       0      1     0        0      1
##   Adventure Children Fantasy Animation Musical Western Horror Mystery
## 1         0        0       0         0       0       0      0       0
## 2         0        0       0         0       0       0      0       0
## 3         0        0       0         0       0       0      0       0
## 4         0        0       0         0       0       0      0       0
## 5         1        0       0         0       0       0      0       0
## 6         1        0       0         0       0       0      0       0
##   Documentary IMAX other
## 1           0    0     1
## 2           0    0     0
## 3           0    0     1
## 4           0    0     1
## 5           0    0     0
## 6           0    0     1
```

Training - Validation dataset

Great ! This task has already been done for us ! The datset provided by the edx team is composed of two datasets: `edx` and `validation`. The `edx` dataset will be used to build our models and the `evaluation` dataset to evaluate them.

```r
train_set <- edx
test_set  <- validation
```

II.3. Memory limits

Unfortunately, all the precedent pre-processing tasks are resource consuming. My current PC does not give me the opportunity to perform the task in a reasonable time (still processing `edx` after 15min when trying to explode the genre column).

The year extraction can be to be done in a reasonable time, but not the second one. Trying to run it on the whole `edx` dataset takes a very long time. I haven't been able to perform this task on my personal computer. As you will read in the next chapters, I have performed this task on an `edx` dataset sample (100 000 rows) for the Decision tree and Random Forest models. However, for the linear model, I use the whole `edx` dataset without applying this pre-process.

II.4. Model

To answer the initial question, I implemented three models: a linear model, a decision tree model and a random forest model.

II.4.1. Linear model

The **linear model** assignes the rating of a movie to the mean of all ratings minus the mean of the ratings grouped by user and movie Id (code is simpler than long sentences). Then I computed the RMSE.

The model is fitting the following equation:

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

where : $\epsilon_{u,i}$ is a random error term μ is the overall mean of all ratings across all users b_i is the movie specific mean b_u is the user specific mean

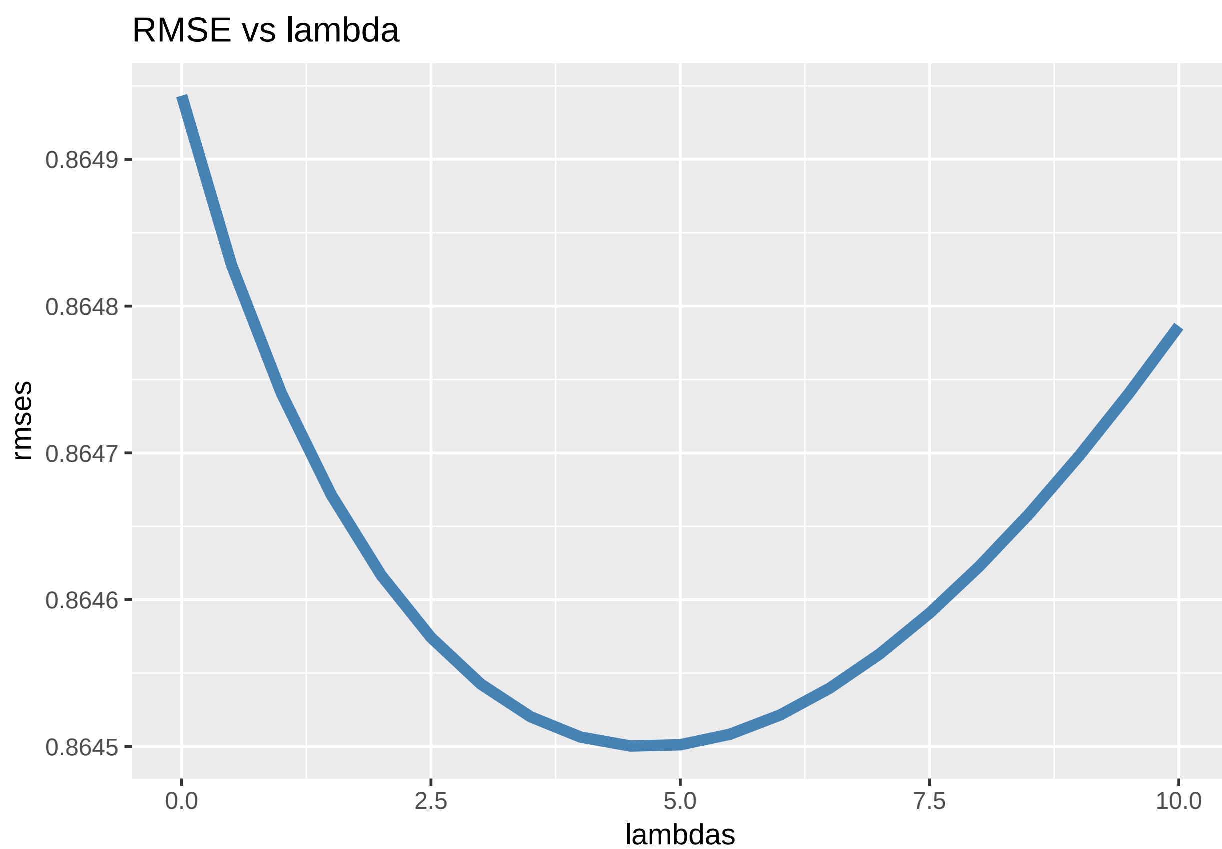
The crucial step in the building process is to minimize the lambda value. To answer this job, we select a range of lambda values from `0` to `10` and compute the RMSE for all the lambda candidates.

```r
# Renaing variable as usual analysis
train_set <- edx
test_set  <- validation

# Prepare test_set for linear model
test_set_lm <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Compute RMSE
compute_RMSE <- function(lambda, train_set, test_set){

  mu <- mean(train_set$rating)

  b_i <- train_set %>%
            group_by(movieId) %>%
            summarize(b_i = sum(rating - mu)/(n()+lambda))

  b_u <- train_set %>%
            left_join(b_i, by="movieId") %>%
            group_by(userId) %>%
            summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
```

```
  predicted_ratings <- test_set %>%
                        left_join(b_i, by = "movieId") %>%
                        left_join(b_u, by = "userId") %>%
                        mutate(pred = mu + b_i + b_u) %>%
                        pull(pred)

  # print(paste("Lambda:", lambda))

  return(RMSE(predicted_ratings, test_set$rating))
}


# Create a set of lambdas to test for regularization
lambdas <- seq(0, 10, 0.5)

#For each value of lambda in lambdas, calculate the RMSE
rmses <- sapply(lambdas, compute_RMSE, train_set=train_set, test_set=test_set_lm)
```

The following plot shows the `RMSE` per `lambda` candidate.

```
ggplot(as.data.frame(cbind(lambdas, rmses)), aes(lambdas, rmses)) +
              geom_line(color='steelblue', size=2) +
              ggtitle("RMSE vs lambda")
```



We can now find the best `lambda` that minimize the RMSE:

```r
paste("Landa minimizing RMSE:", lambdas[which.min(rmses)])
```

```
## [1] "Landa minimizing RMSE: 4.5"
```

Decision tree

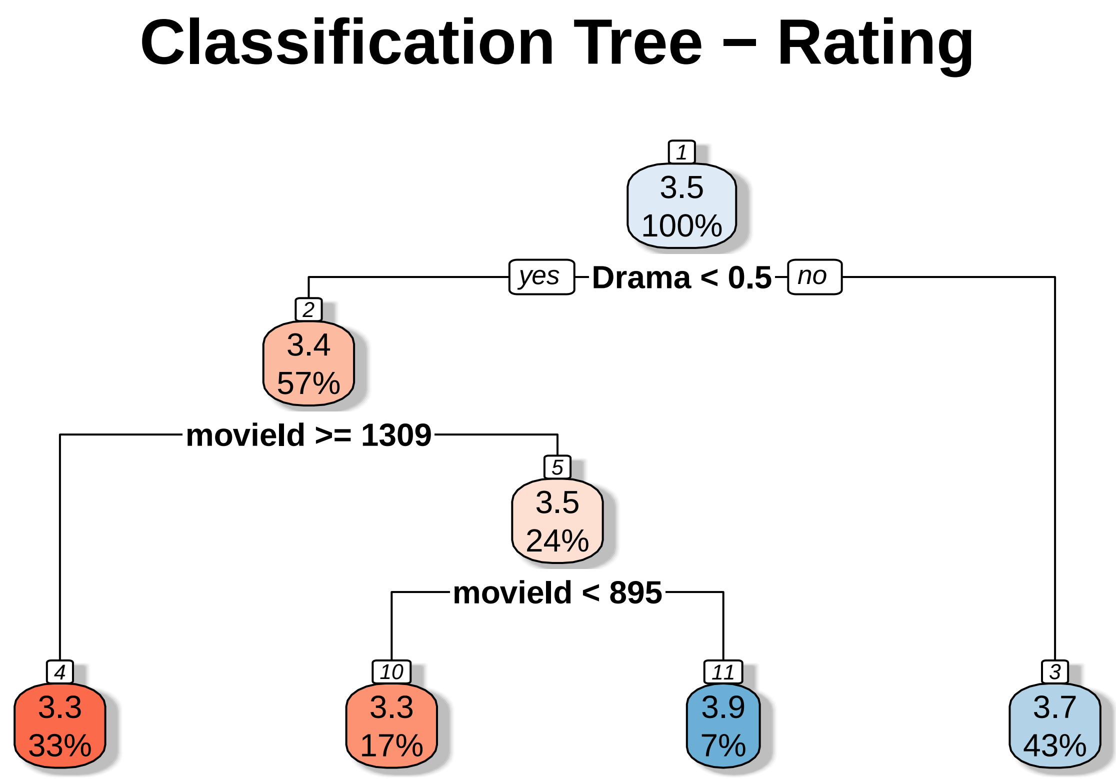
The second model is a decision tree. I build it with the `rpart` function from rpart module. Building a decision tree is resource consuming. For this reason, I use a sample of the dataset (100 000rows) to build it. Also, because I deal with a fewer dataset, I explode the `genres` column in binary columns and group the least frequent (see chapter II - Preprocessing).

```r
# Renaming variables
train_set <- edx
test_set  <- validation

# Reduce size on training and test dataset
train_set <- train_set[sample(nrow(train_set), 100000), ]
test_set <- test_set[sample(nrow(test_set), 50000),]

# Pre-process
train_set <- pre_process_data_FT(train_set)
test_set <- pre_process_data_FT(test_set)

genre_names <- colnames(train_set)[seq(4, length(colnames(train_set)))]

train_set <- group_genres_columns(train_set, genre_names)
test_set <- group_genres_columns(test_set, genre_names)

feature_names <- colnames(train_set)[! colnames(train_set) %in% c('rating')]

# Prediction formula
predictor_formula <- paste("rating ~", paste(feature_names, collapse = " + "))
predictor_formula
[1] "rating ~ userId + movieId + Action + Drama + War + Horror + Comedy + Romance + Children + Musical +
Mystery + Western + Crime + Documentary + IMAX + Sci_Fi + Film_Noir + other"

# Build model
model_tree <- rpart(predictor_formula, data=train_set)
```

The decision tree is the following:

```r
rpart.plot(model_tree, cex.main=2,
           main="Classification Tree - Rating",
           box.palette="RdBu", shadow.col="gray", nn=TRUE)
```

```
## Warning: Cannot retrieve the data used to build the model (model.frame: objet 'Drama' introuvable).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



Random Forest model

I build the random forest model with `randomForest` method from randomForest module. The model is built from the sampled dataset (100 000rows). I apply the whole pre-process steps on.

```
# Renaing variable as usual analysis
train_set <- edx
test_set  <- validation

# Reduce size on training and test dataset
train_set <- train_set[sample(nrow(train_set), 10000), ]
test_set <- test_set[sample(nrow(test_set), 5000),]

# Create feature dataset for training
train_x <- train_set[setdiff(names(train_set), c("rating"))]

# Pre-process data
train_x <- pre_process_data_FT(train_x)
test_set <- pre_process_data_FT(test_set)

# Build model - can take a some minutes
model_randomForest <- randomForest(x = train_x, y = train_set$rating)
```

III.Results

Now we have our models, we are ready to evaluate the model on the `evaluate` dataset.

Linear model

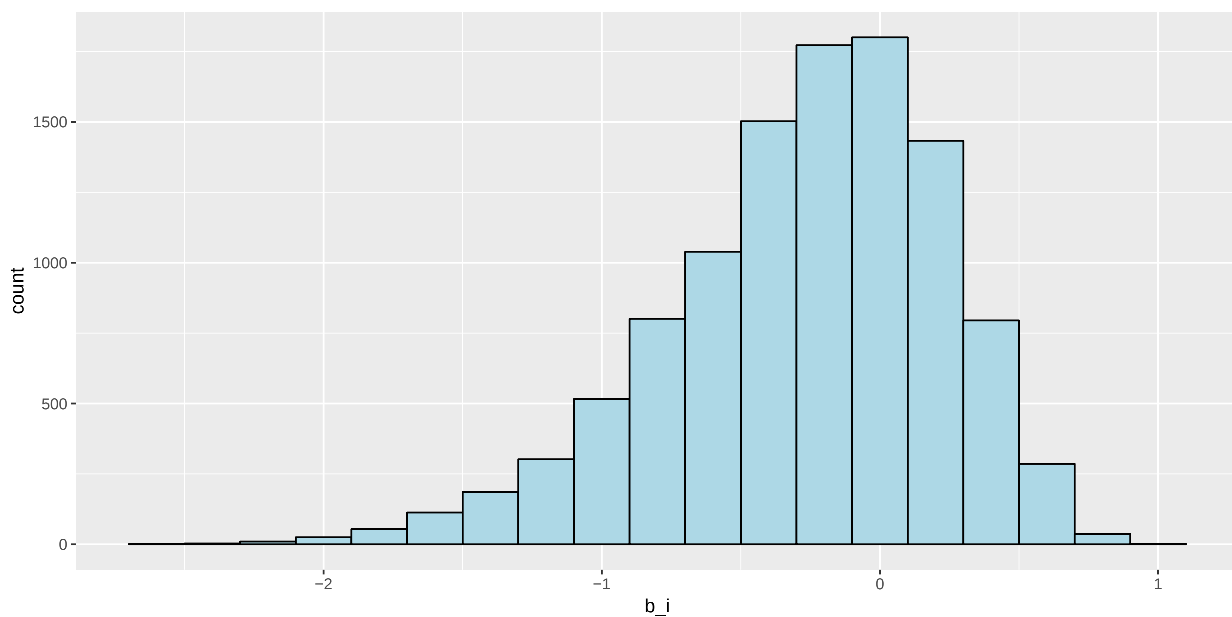
The results are the following:

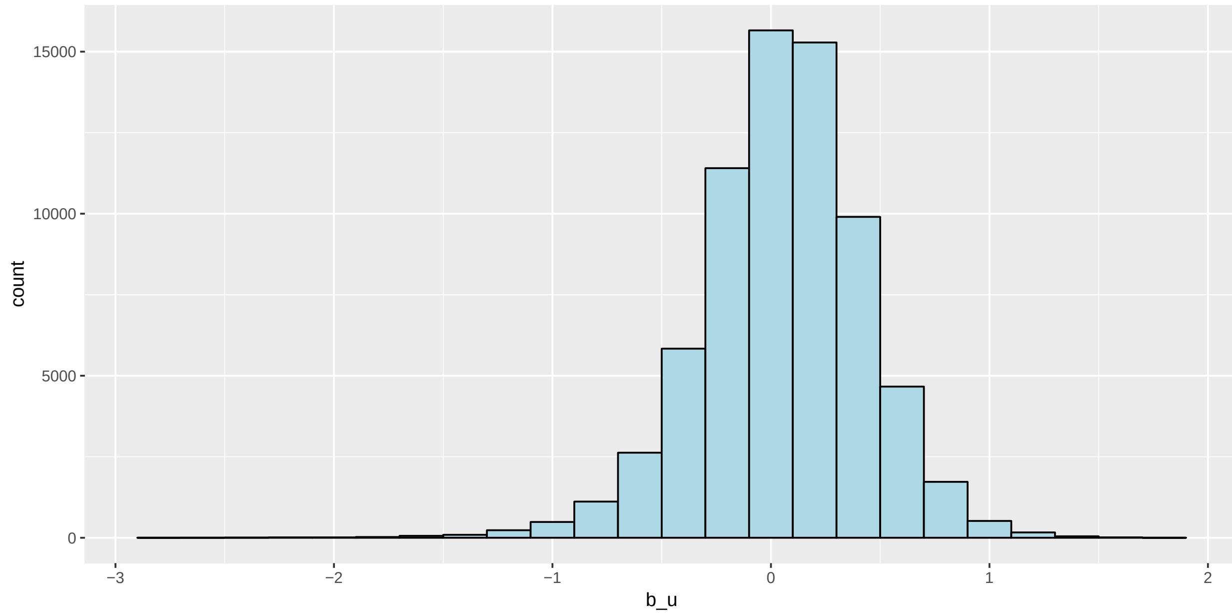
```
## [1] "mu: 3.512"
```

```
## [1] "lambda: 4.5"
```

```
## [1] "RMSE linear model: 0.8645"
```

b_u and b_i were widely distributed as seen below





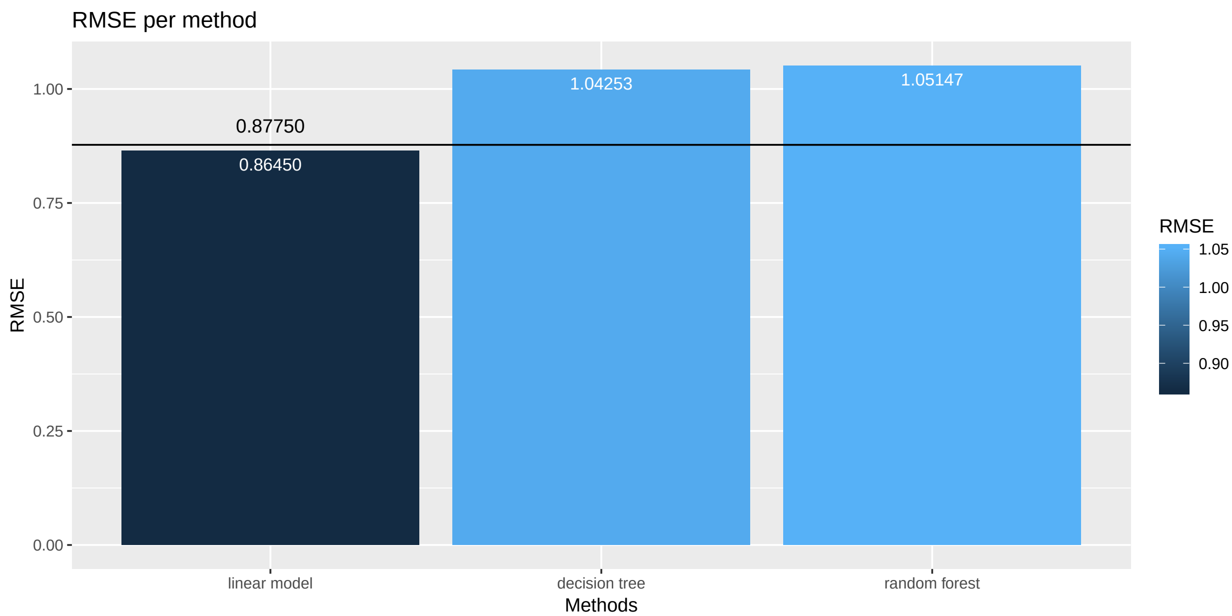
Decision tree:

```
## [1] "RMSE decision tree:  1.04253413798719"
```

Random forest model:

```
## [1] "RMSE random forest model:  1.05146936607111"
```

The following shows the winner : the **linear model** with an RMSE equals to 0.8649659.



IV. Conclusion

Our initial question was to predict the rating from the other columns. I answered by creating 3 models : a linear model, a decision tree and a random forest model. After pre-processing our training dataset, I implemented them. The size of our dataset has shown the efficiency of the linear model over big dataset. However, the decision tree and random forest model are more sensible to big dataset and require bigger computational resources. We reach to test those two precedents models by downsizing our training dataset.

The classement of those 3 methods according the RMSE criteria is the following: 1 : linear model, 2 : decision tree and 3 : random forest model.

The linear model meets the best criteria defined in this project evaluation.

Thank you the edx team for the hard work, thank you for reading ! :-)

If you have any questions, I would be glad to answer you (pop up me on LinkedIn for example). .