Capstone

Virginia Carneiro

December 22nd, 2020

Contents

1 INTRODUCTORY

1.1 OVERVIEW

THIS PROJECT WILL FOCUS TO GENERATE A MOVIE RECOMMENDATION SYSTEM USING THE *MOVIELENS-10M* DATASET.

THE MODEL WILL PREDICT THE RATING THAT A USER WILL GIVE TO A MOVIE.

I WILL CREATE MY OWN RECOMMENDATION SYSTEM USING THE TOOLS LEARNING DURING THE 8 COURSES INCLUDED IN THE "DATA SCIENCE PROFESSIONAL CERTIFICATE" AND, ALSO OTHER RESOURCES I WILL INCLUDING IN THE RESOURCES SECTION.

TO GENERATE THE TRAIN AND VALIDATION DATASETS I WILL TAKE THE CODE PROVIDED BY HARVARD STAFF.

AFTER THAT, I WILL MAKE SOME VALIDATIONS BETWEEN THEM AND VERIFY THE DATASETS ARE CONSISTENT.

THEN, I WILL WORK WITH FEATURE ENGINEERING AND TARGET ENCODING TOOLS TO DEVELOP MY RECOMMENDATION SYSTEM ALGORITHMS.

FINALLY, I WILL CHOOSE THE BEST RECOMMENDATION SYSTEM ALGORITHMS ACCORDING TO THE SMALLEST RMSE VALUE.

INTO THE MOVIELENS PROJECT WILL BE THREE FILES:

1. MY REPORT IN PDF FORMAT.
2. A REPORT IN RMD FORMAT.
3. A SCRIPT IN R FORMAT THAT GENERATES YOUR PREDICTED MOVIE RATINGS AND RMSE SCORE.

2 MOVIELENS DATASET

2.1 DATA VALIDATION

AFTER RUNNING THE SCRIPT PROVIDED BY HARVARD STAFF, I GOT TWO DATASETS WITH THE FOLLOW STRUCTURES:

2.1.1 EDX DATASET

```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
```

2.1.2 VALIDATION DATASET

```r
head(validation)
```

```
##    userId movieId rating timestamp
## 1:      1     231      5 838983392
## 2:      1     480      5 838983653
## 3:      1     586      5 838984068
## 4:      2     151      3 868246450
## 5:      2     858      2 868245645
## 6:      2    1544      3 868245920
##                                                       title
## 1:                                      Dumb & Dumber (1994)
## 2:                                      Jurassic Park (1993)
## 3:                                         Home Alone (1990)
## 4:                                            Rob Roy (1995)
## 5:                                     Godfather, The (1972)
## 6: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                         genres
## 1:                                      Comedy
## 2:       Action|Adventure|Sci-Fi|Thriller
## 3:                             Children|Comedy
## 4:                   Action|Drama|Romance|War
## 5:                                 Crime|Drama
## 6: Action|Adventure|Horror|Sci-Fi|Thriller
```

TO MAKE THE REPORT EASY TO READ I WILL SHOW JUST THE ANALYSIS FOR *EDX DATASET*, BUT THE SAME VALIDATION WILL BE AVARIABLE ON R SCRIPT FOR VALIDATION DATASET AS WELL.

- THERE ARE NOT *NA* VALUES IN THE DATASET.

```r
sum(is.na(edx))
```

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

- THERE ARE 69878 DIFFERENT USERID.

```r
edx_users <- edx %>%
  select(userId) %>%
  distinct()
str(edx_users)
```

```
## Classes 'data.table' and 'data.frame':   69878 obs. of  1 variable:
##  $ userId: int  1 2 3 4 5 6 7 8 9 10 ...
```

- THERE ARE 10677 DIFFERENT MOVIEID.

```r
edx_movieId <- edx %>%
  select(movieId) %>%
  distinct()
str(edx_movieId)
```

```
## Classes 'data.table' and 'data.frame':   10677 obs. of  1 variable:
##  $ movieId: num  122 185 292 316 329 355 356 362 364 370 ...
```

- THERE ARE 10676 DIFFERENT TITLES. THAT MEANS THERE IS A DUPLICATE TITLE. IT HAS DIFFERENT MOVIEID AND GENRES. I'M NOT GOING TO REMOVE THEM BECAUSE ONE GENRES INCLUDE THE OTHER ONE.

```r
edx_movieId_title <- edx %>%
  select(movieId,title) %>%
  distinct()
str(edx_movieId_title)
```

```
## Classes 'data.table' and 'data.frame':   10677 obs. of  2 variables:
##  $ movieId: num  122 185 292 316 329 355 356 362 364 370 ...
##  $ title  : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
```

```r
which(duplicated(edx_movieId_title$title))
```

```
## [1] 6538
```

```r
edx_movieId_title[6538]
```

```
##    movieId                      title
## 1:   64997 War of the Worlds (2005)
```

```
edx %>% select(movieId,title,genres) %>%
  filter(title == "War of the Worlds (2005)") %>%
  group_by (movieId,title,genres) %>%
  distinct(n())
```

```
## # A tibble: 2 x 4
## # Groups:   movieId, title, genres [2]
##   movieId title                    genres                           `n()`
##     <dbl> <chr>                    <chr>                            <int>
## 1   34048 War of the Worlds (2005) Action|Adventure|Sci-Fi|Thriller  2460
## 2   64997 War of the Worlds (2005) Action                              28
```

- THERE ARE 10 DIFFERENT RATINGS. IT GOES FROM 0.5 TO 5 AND INCREASES BY 0.5.

```
edx_rating <- edx %>%
  select(rating) %>%
  distinct()
str(edx_rating)
```

```
## Classes 'data.table' and 'data.frame':   10 obs. of  1 variable:
##  $ rating: num  5 3 2 4 4.5 3.5 1 1.5 2.5 0.5
```

- THERE ARE 797 DIFFERENT GENRES. THERE ARE 7 OBS. AS "(NO GENRES LISTED)". I WILL REMOVE THEM BECAUSE THIS IS NOT A CORRECT CLASSIFICATION.

```
edx %>% filter(genres == "(no genres listed)")
```

```
##    userId movieId rating  timestamp                title             genres
## 1:   7701    8606    5.0 1190806786 Pull My Daisy (1958) (no genres listed)
## 2:  10680    8606    4.5 1171170472 Pull My Daisy (1958) (no genres listed)
## 3:  29097    8606    2.0 1089648625 Pull My Daisy (1958) (no genres listed)
## 4:  46142    8606    3.5 1226518191 Pull My Daisy (1958) (no genres listed)
## 5:  57696    8606    4.5 1230588636 Pull My Daisy (1958) (no genres listed)
## 6:  64411    8606    3.5 1096732843 Pull My Daisy (1958) (no genres listed)
## 7:  67385    8606    2.5 1188277325 Pull My Daisy (1958) (no genres listed)
```

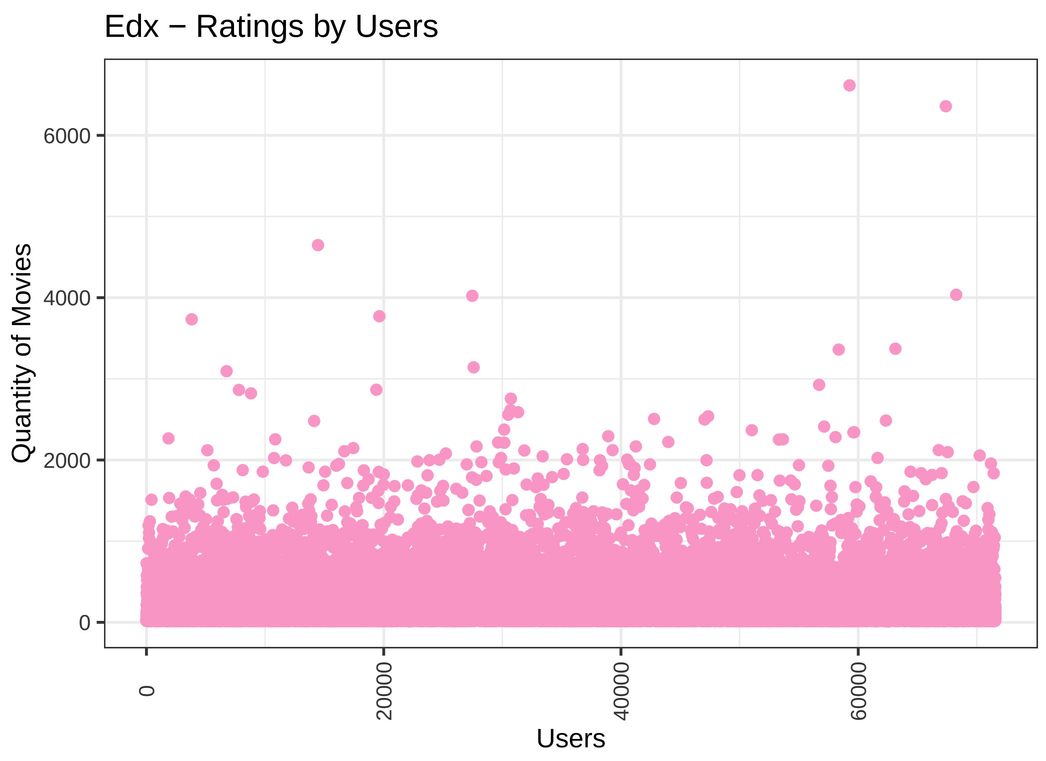
2.2 DATA VISUALIZATION

BEFORE THE BELOW PLOTS, I'LL APPLY SOME FEATURES ENGINEERING FOR BETTER VISUALIZATION. THESE FEATURES WILL BE APPLIED IN EDX AND VALIDATION DATASETS TO KEEP THE CONSISTENCIES.

- ADD COLUMNS FOR EACH GENRES FROM GENRES COLUMN.
- ADD A COLUMN FOR THE MOVIE YEAR FROM THE TITLE COLUMN.
- ADD YEAR, MONTH AND DAY OF RATED FROM TIMESTAMP.
- REMOVE TITLE, TIMESTAMP AND GENRES COLUMNS.
- ADD DIF_RATE_MOVIE (YEAR_RATED - YEAR_MOVIE).

2.2.1 EDX RATINGS BY USERS

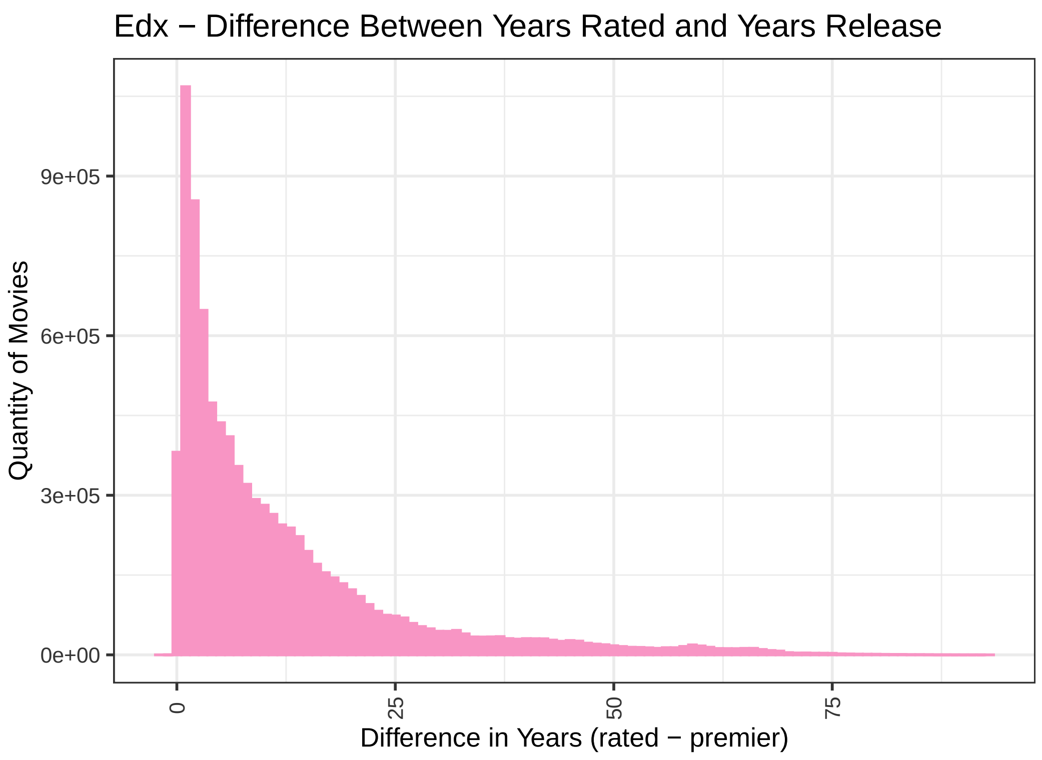
```r
edx_rating_users %>%
  ggplot(aes(x = userId, y = qty)) +
  geom_point(color= '#f895c4')+
  labs(title = "Edx - Ratings by Users",
       x = "Users",
       y = "Quantity of Movies")+
  theme_bw() +
  theme(axis.text.x = element_text(colour = "grey20", size = 8, angle = 90, hjust = 0.5,
                                   vjust = 0.5),
        axis.text.y = element_text(colour = "grey20", size = 8),
        strip.text = element_text(face = "italic"),
        text = element_text(size = 10))
```



- THE MAJORITY OF THE USERS RATED LESS THAN 1000 MOVIES.
- WE CAN OBSERVE THAT THERE ARE SOME OUTLIERS AT THE TOP OF THE PLOT.

2.2.2 EDX YEAR RATED VS YEAR RELEASE

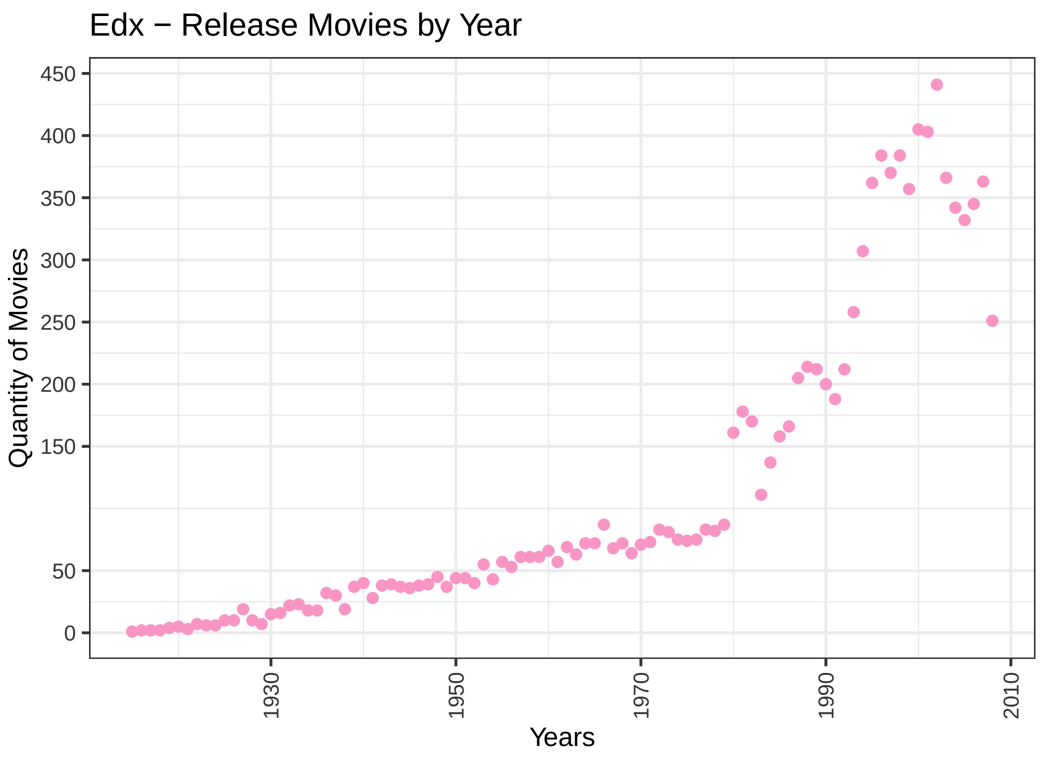
```r
ggplot(data = edx_movie_release_rated, aes(x = difference, y = Qty)) +
  geom_bar(stat = "identity", color= '#f895c4', fill = '#f895c4') +
  labs(title = "Edx - Difference Between Years Rated and Years Release",
       x = "Difference in Years (rated - premier)",
       y = "Quantity of Movies") +
  theme_bw() +
  theme(axis.text.x = element_text(colour = "grey20", size = 8, angle = 90, hjust = 0.5,
                                   vjust = 0.5),
        axis.text.y = element_text(colour = "grey20", size = 8),
        strip.text = element_text(face = "italic"),
        text = element_text(size = 10))
```



- THE MAJORITY OF THE MOVIES HAVE BEEN RATED A YEAR AFTER THE PREMIER.
- THERE ARE A FEW MOVIES HAVE BEEN RATED BEFORE THE PREMIER. THESE ONES CAN BE CONSIDERED AS OUTLIERS.

2.2.3 EDX RELEASE MOVIES BY YEAR

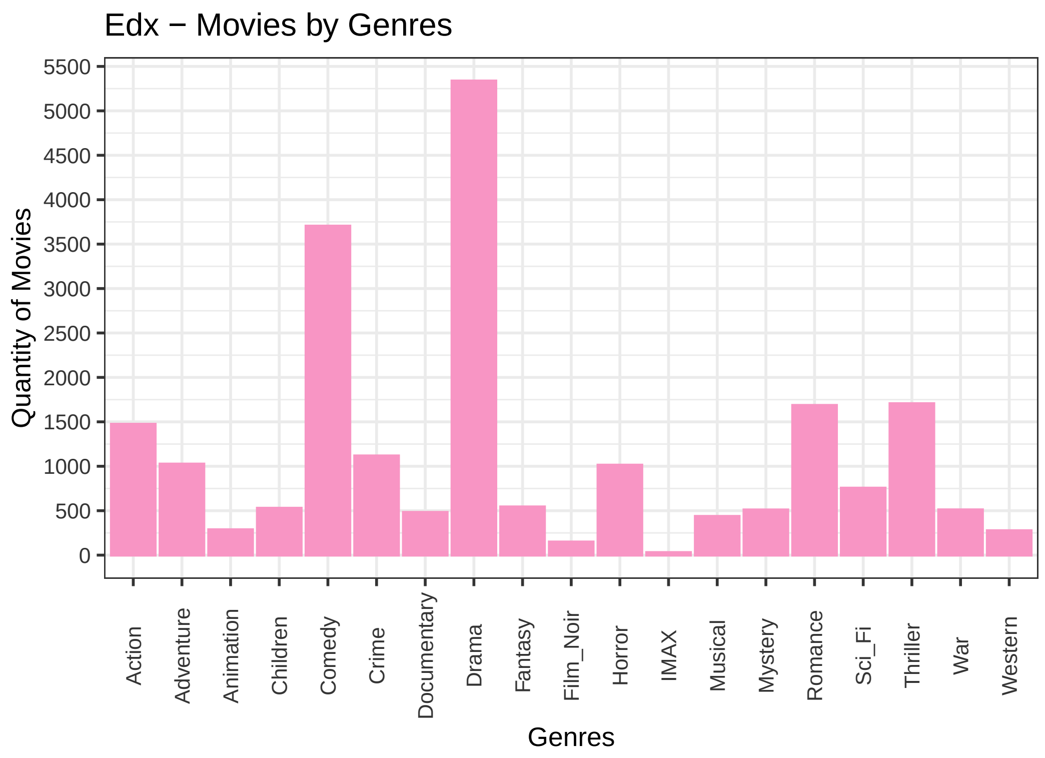
```
edx_movies_year_movie %>%
  ggplot(aes(x = year_movie, y = qty)) +
  geom_point(color= '#f895c4')+
  scale_y_continuous(breaks = c(0, 50, 150, 200, 250, 300, 350, 400, 450))+
  labs(title = "Edx - Release Movies by Year",
       x = "Years",
       y = "Quantity of Movies")+
  theme_bw() +
  theme(axis.text.x = element_text(colour = "grey20", size = 8, angle = 90, hjust = 0.5,
                                   vjust = 0.5),
        axis.text.y = element_text(colour = "grey20", size = 8),
        strip.text = element_text(face = "italic"),
        text = element_text(size = 10))
```



- WE HAVE FEW MOVIES RELEASED BETWEEN 1915 AND 1930.
- STARTING IN 1931, THE PREMIERES OF THE FILMS GREW PROGRESSIVELY UNTIL 1979.
- BEGAN IN 1980, WE CAN OBSERVED A PRONOUNCED GROWTH UNTIL REACHING THE MAXIMUM GROWTH PEAK BETWEEN 1996 TO 2003.
- AFTER 2004 MOVIE PREMIERES BEGAN TO DECLINE BUT NOT TOO MUCH.

2.2.4 EDX MOVIES BY GENRES

```
ggplot(data = edx_movies_genres_colSums, aes(x = rownames(edx_movies_genres_colSums),
                                              y = Qty)) +
  geom_bar(stat = "identity", color= '#f895c4', fill = '#f895c4') +
  scale_y_continuous(breaks = c(0, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000,
                                4500, 5000, 5500))+
  labs(title = "Edx - Movies by Genres",
       x = "Genres",
       y = "Quantity of Movies") +
  theme_bw() +
  theme(axis.text.x = element_text(colour = "grey20", size = 8, angle = 90, hjust = 0.5,
                                   vjust = 0.5),
        axis.text.y = element_text(colour = "grey20", size = 8),
        strip.text = element_text(face = "italic"),
        text = element_text(size = 10))
```



- THE TOP MOVIES BY GENRES RATED WERE DRAMA, COMEDY, THRILLER AND, ROMANCE.

3 MOVIE RECOMMENDATION SYSTEM

THE RECOMMENDATION SYSTEM WILL BE EVALUATED ACCORDING TO THE RMSE METRIC.

THE RMSE() FUNCTION AVAILABLE IN THE PACKAGE IN R IS USED TO CALCULATE ROOT MEAN SQUARE ERROR BETWEEN ACTUAL VALUES AND PREDICTED VALUES. THE LOWER VALUES OF RMSE INDICATE A BETTER FIT. RMSE IS A GOOD MEASURE OF HOW ACCURATELY THE MODEL PREDICTS THE RESPONSE, AND IT IS THE MOST IMPORTANT CRITERION FOR FIT IF THE MAIN PURPOSE OF THE MODEL IS PREDICTION.

I TRIED WITH DIFFERENT ALGORITHMS SUCH AS LINEAR REGRESSION, LOGISTIC REGRESSION, REGRESSION TREE AND RANDOM FOREST.

DUE TO THE HARDWARE LIMITATION OF MY COMPUTER, I COULDN'T OPTIMIZE THESE ALGORITHMS TO REACH AN ACCEPTABLE RMSE. BUT THE STUDY OF DECISION TREES FOR MACHINE LEARNING CAUGHT MY INTEREST.

FOR THAT REASON I PUT HIGHER EMPHASIS ON THE RF MODEL, BUT I HAVE NOT ACHIEVED AN ACEPTABLE RMSE.

BECAUSE THESE LIMITATIONS, I STARTED LOOKING FOR DIFFERENT OPTIONS AND I FOUND A LIBRARY IN R CALLED XGBOOST (EXTREME GRADIENT BOOSTING) WHICH IS POPULAR FOR KAGGLE COMPETITIONS. IT IS AN OPEN-SOURCE IMPLEMENTATION OF GRADIENT BOOSTED TREES ALGORITHM AND IT IS DESIGNED FOR SPEED AND PERFORMANCE.

RF AND XGBOOST ARE SETS OF DECISION TREES, BUT THEY HAVE SOME DIFFERENCES:

1. BUILDING TREES:
    - RF: INDEPENDENT TREES
    - GRADIENT BOOSTING: ONE TREE AT A TIME(FORWARD STAGE-WISE MANNER)
2. COMBINING RESULTS:
    - RF: END OF THE PROCESS
    - GRADIENT BOOSTING: AT THE BEGINNING
3. TUNING:
    - RF: MEDIUM
    - GRADIENT BOOSTING: HARD (BETTER PERFORMANCE)

3.1 ABOUT XGBOOST

XGBOOST IS USED FOR SUPERVISED LEARNING PROBLEMS, WHERE WE USE THE TRAINING DATA (WITH MULTIPLE FEATURES) x_i TO PREDICT A TARGET VARIABLE y_i. XGBOOST CAN BE USED TO SOLVE REGRESSION, CLASSIFICATION, RANKING, AND USER-DEFINED PREDICTION PROBLEMS. FOR MY RECOMMENDATION SYSTEM I'M USING REGRESSION.

XGBOOST IS A PERFECT COMBINATION OF SOFTWARE AND HARDWARE OPTIMIZATION TECHNIQUES TO YIELD SUPERIOR RESULTS USING LESS COMPUTING RESOURCES IN THE SHORTEST AMOUNT OF TIME.

3.2 WHY I CHOSE XGBOOST

BELOW ARE THE PRINCIPAL FEATURES OF THE MODEL (FOR MORE DETAILS SEE THE RESOURCES SECTION).

3.2.1 SYSTEM FEATURES

- PARALLELIZATION OF TREE CONSTRUCTION USING ALL OF YOUR CPU CORES DURING TRAINING.
- DISTRIBUTED COMPUTING FOR TRAINING VERY LARGE MODELS USING A CLUSTER OF MACHINES.
- OUT-OF-CORE COMPUTING FOR VERY LARGE DATASETS THAT DON'T FIT INTO MEMORY.
- CACHE OPTIMIZATION OF DATA STRUCTURES AND ALGORITHM TO MAKE BEST USE OF HARDWARE.

3.2.2 ALGORITHM FEATURES

THE ALGORITHM WAS ENGINEERED FOR THE EFFICIENCY OF COMPUTE TIME AND MEMORY RESOURCES. THE ALGORITHM FEATURES INCLUDE:

- SPARSE AWARE IMPLEMENTATION WITH AUTOMATIC HANDLING OF MISSING DATA VALUES.
- BLOCK STRUCTURE TO SUPPORT THE PARALLELIZATION OF TREE CONSTRUCTION.
- CONTINUED TRAINING SO THAT YOU CAN FURTHER BOOST AN ALREADY FITTED MODEL ON NEW DATA.

3.2.3 XGBOOST PARAMETERS:

TO KEEP THE MODEL SIMPLE, I WILL SHOW THE IMPROVEMENT OF THE MODEL THROUGH THE SETTING OF THESE PARAMETERS. THERE ARE A LOT OF PARAMETERS TO SET, BUT I'M GOING TO MENTION THE ONES THAT I'M USING FOR MY RECOMMENDATION SYSTEM.

- NROUNDS = MAXIMUM ROUND QUANTITY.
- VERBOSE = SHOW THE OPTIMIZATION PROGRESS.
- EARLY_STOPPING_ROUNDS = STOP TRAINING AFTER x ROUNDS WITHOUT ANY IMPROVEMENT.
- WATCHLIST = TRAIN / TEST PAIR TO CHECK.
- EVAL_METRIC = METRIC TO EVALUATE (RMSE).
- MAX_DEPTH (6 DEFAULT) = MAXIMUM DEPTH OF A TREE. USED TO CONTROL OVERFITTING.
- ETA (0.3 DEFAULT) = LEARNING RATE . IT IS SIZE OF THE SHRINKAGE USED TO PREVENT OVERFITTING.
- MIN_CHILD_WEIGHT (DEFAULT=1) = THE MINIMUM SUM OF INSTANCE WEIGHT (HESSIAN) NEEDED IN A CHILD. THE LARGE IT IS, THE MORE CONSERVATIVE THE ALGORITHM WILL BE. THIS IS TO IMPROVE OVERFITTING.
- TREE_METHOD (DEFAULT = auto) = IT IS THE TREE CONSTRUCTION ALGORITHM. I CHOSE **HIST** BECAUSE IT IS FASTER THAN THE APPROX TREE METHOD (WHICH IS SELECTED IF I DONT MENTION THE TREE METHOD) WITH THE SAME RESULTS.

4 THE RECOMMENDATION SYSTEM

FROM THIS POINT ON I AM GOING TO SHOW THE DIFFERENT CONFIGURATIONS THAT I DID TO ACHIEVE AN ACCEPTABLE RMSE.

I SPLIT THE EDX DATASET INTO TWO DATASETS. ONE FOR TRAINING AND ANOTHER ONE FOR TEST (20%).

I WILL USE THE VALIDATION DATASET WHEN I GET AN ACCEPTABLE RMSE ON TEST DATASET.

THERE IS A WAY TO FIND THE BEST TUNING WITH XGBOOST, BUT IT WILL TAKE MORE THAN 2 WEEKS TO RUN ON MY COMPUTER. THIS IS WHY I COULDN'T APPLY IT TO MAKE IT EASY.

THE PARAMETERS TO TUNING ARE:

- NROUNDS (100, 200 ROUNDS).
- MAX_DEPTH (10, 15, 20, 25).
- COLSAMPLE_BYTREE (SUBSAMPLING OF COLUMNS WITH 0.5, 0.6, 0.7, 0.8, 0.9).
- ETA (0.1).

5 PREPARE DATASET TO RUN XGBOOST

- TO GIVE THE ALGORITHM MORE FLEXIBILITY FOR OPTIMIZATION PROPOSE. I'M GOING TO APPLY TARGET ENCODING AROUND OUR TARGET VARIABLE **RATING**.
- TARGET ENCODING ONLY APPLIED FOR TRAINING PURPOSE.

5.1 TRAIN DATASET STRUCTURE

```r
str(train_edx)
```

```
## Classes 'data.table' and 'data.frame':   7200037 obs. of  49 variables:
##  $ userId                   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId                  : num  185 316 329 355 364 377 420 539 588 589 ...
##  $ rating                   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ genres                   : num  578 138 170 32773 65596 ...
##  $ year_movie               : num  1995 1994 1994 1994 1994 ...
##  $ year_rated               : num  1996 1996 1996 1996 1996 ...
##  $ month_rated              : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ day_rated                : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ dif_rated_movie          : num  1 2 2 2 2 2 2 3 4 5 ...
##  $ userId_sd                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ userId_mean              : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_median            : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_mode              : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ movieId_sd               : num  0.955 0.941 0.942 0.961 0.941 ...
##  $ movieId_mean             : num  3.13 3.35 3.34 2.49 3.75 ...
##  $ movieId_median           : num  3 3 3 3 4 3.5 3 3.5 4 4 ...
##  $ movieId_mode             : num  3 3 3 3 4 4 3 4 4 4 ...
##  $ year_movie_sd            : num  1.07 1.07 1.07 1.07 1.07 ...
##  $ year_movie_mean          : num  3.44 3.47 3.47 3.47 3.47 ...
##  $ year_movie_median        : num  3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 4 ...
##  $ year_movie_mode          : num  3 3 3 3 3 3 3 4 4 4 ...
##  $ year_rated_sd            : num  0.995 0.995 0.995 0.995 0.995 ...
##  $ year_rated_mean          : num  3.55 3.55 3.55 3.55 3.55 ...
##  $ year_rated_median        : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ year_rated_mode          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ month_rated_sd           : num  1.06 1.06 1.06 1.06 1.06 ...
##  $ month_rated_mean         : num  3.48 3.48 3.48 3.48 3.48 ...
##  $ month_rated_median       : num  3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 ...
##  $ month_rated_mode         : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ day_rated_sd             : num  1.06 1.06 1.06 1.06 1.06 ...
##  $ day_rated_mean           : num  3.52 3.52 3.52 3.52 3.52 ...
##  $ day_rated_median         : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ day_rated_mode           : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ userId_year_rated_sd     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ userId_year_rated_mean   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_year_rated_median : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_year_rated_mode   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_month_rated_sd    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ userId_month_rated_mean  : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_month_rated_median: num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_month_rated_mode  : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_day_rated_sd      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ userId_day_rated_mean    : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_day_rated_median  : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ userId_day_rated_mode    : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ movieId_year_movie_sd    : num  0.955 0.941 0.942 0.961 0.941 ...
##  $ movieId_year_movie_mean  : num  3.13 3.35 3.34 2.49 3.75 ...
##  $ movieId_year_movie_median: num  3 3 3 3 4 3.5 3 3.5 4 4 ...
##  $ movieId_year_movie_mode  : num  3 3 3 3 4 4 3 4 4 4 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

5.2 TEST DATASET STRUCTURE

```
str(test_edx)
```

```
## Classes 'data.table' and 'data.frame':   1800011 obs. of  49 variables:
##  $ userId                   : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ movieId                  : num  122 292 356 362 370 466 520 594 260 376 ...
##  $ rating                   : num  5 5 5 5 5 5 5 5 5 3 ...
##  $ genres                   : num  16385 674 147489 16396 3 ...
##  $ year_movie               : num  1992 1995 1994 1994 1994 ...
##  $ year_rated               : num  1996 1996 1996 1996 1996 ...
##  $ month_rated              : num  8 8 8 8 8 8 8 8 7 7 ...
##  $ day_rated                : num  2 2 2 2 2 2 2 2 7 7 ...
##  $ dif_rated_movie          : num  4 1 2 2 2 3 3 59 20 3 ...
##  $ userId_sd                : num  0 0 0 0 0 ...
##  $ userId_mean              : num  5 5 5 5 5 ...
##  $ userId_median            : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ userId_mode              : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ movieId_sd               : num  0.932 0.868 0.972 0.946 1.027 ...
##  $ movieId_mean             : num  2.86 3.42 4.01 3.46 2.96 ...
##  $ movieId_median           : num  3 3 4 3 3 3 3 4 4.5 3 ...
##  $ movieId_mode             : num  3 3 5 3 3 3 3 4 5 3 ...
##  $ year_movie_sd            : num  1.05 1.07 1.07 1.07 1.07 ...
##  $ year_movie_mean          : num  3.43 3.44 3.47 3.47 3.47 ...
##  $ year_movie_median        : num  3.5 3.5 3.5 3.5 3.5 3.5 3.5 4 4 3.5 ...
##  $ year_movie_mode          : num  4 3 3 3 3 4 4 4 5 3 ...
##  $ year_rated_sd            : num  0.995 0.995 0.995 0.995 0.995 ...
##  $ year_rated_mean          : num  3.55 3.55 3.55 3.55 3.55 ...
##  $ year_rated_median        : num  3 3 3 3 3 3 3 3 4 4 ...
##  $ year_rated_mode          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ month_rated_sd           : num  1.06 1.06 1.06 1.06 1.06 ...
##  $ month_rated_mean         : num  3.48 3.48 3.48 3.48 3.48 ...
##  $ month_rated_median       : num  3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 ...
##  $ month_rated_mode         : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ day_rated_sd             : num  1.06 1.06 1.06 1.06 1.06 ...
##  $ day_rated_mean           : num  3.52 3.52 3.52 3.52 3.52 ...
##  $ day_rated_median         : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ day_rated_mode           : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ userId_year_rated_sd     : num  0 0 0 0 0 ...
##  $ userId_year_rated_mean   : num  5 5 5 5 5 ...
##  $ userId_year_rated_median : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ userId_year_rated_mode   : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ userId_month_rated_sd    : num  0 0 0 0 0 ...
##  $ userId_month_rated_mean  : num  5 5 5 5 5 ...
##  $ userId_month_rated_median: num  5 5 5 5 5 5 5 5 3 3 ...
##  $ userId_month_rated_mode  : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ userId_day_rated_sd      : num  0 0 0 0 0 ...
##  $ userId_day_rated_mean    : num  5 5 5 5 5 ...
##  $ userId_day_rated_median  : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ userId_day_rated_mode    : num  5 5 5 5 5 5 5 5 3 3 ...
##  $ movieId_year_movie_sd    : num  0.932 0.868 0.972 0.946 1.027 ...
##  $ movieId_year_movie_mean  : num  2.86 3.42 4.01 3.46 2.96 ...
##  $ movieId_year_movie_median: num  3 3 4 3 3 3 3 4 4.5 3 ...
##  $ movieId_year_movie_mode  : num  3 3 5 3 3 3 3 4 5 3 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

6 XGBOOST MODELS

6.1 MODEL 1 - DEFAULT VALUES

IT WILL BE CREATED WITH THE DEFAULT VALUES. THIS IS THE *BASELINE MODEL*.

- max_depth = 6 (default)
- eta = 0.3 (default)
- min_child_weight = 1 (default)

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

y <- train_edx$rating
x <- data.matrix(train_edx[,!("rating")])
d_train <- xgb.DMatrix(data = x, label = y)

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])
d_test <- xgb.DMatrix(data = x, label = y)
```

```
watchlist <- list(train = d_train, eval = d_test)

fit <- xgb.train(data=d_train,
                 nrounds = 5000,
                 verbose = FALSE,
                 tree_method = 'hist',
                 early_stopping_rounds = 20,
                 watchlist = watchlist,
                 eval_metric = 'rmse')

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])

yhat = predict(fit, x)
RMSE_model_I <- RMSE(y, yhat)
```

Table 1: RMSEs

Method	RMSE
BASELINE - TEST	0.8310898

6.2 MODEL 2 - MAX TREE DEPTH = 8

IN THIS MODEL I'M CHANGING THE TREE DEPTH FROM 6 TO 8. WILLING AN IMPROVEMENT IN THE RMSE.

- max_depth = 8
- eta = 0.3 (default)
- min_child_weight = 1 (default)

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

y <- train_edx$rating
x <- data.matrix(train_edx[,!("rating")])
d_train <- xgb.DMatrix(data = x, label = y)

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])
d_test <- xgb.DMatrix(data = x, label = y)
```

```r
watchlist <- list(train = d_train, eval = d_test)

fit <- xgb.train(data=d_train,
                 nrounds = 5000,
                 verbose = FALSE,
                 tree_method ='hist',
                 max_depth = 8,
                 early_stopping_rounds = 20,
                 watchlist = watchlist,
                 eval_metric = 'rmse')

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])

yhat = predict(fit, x)
RMSE_model_II <- RMSE(y, yhat)
```

Table 2: RMSEs

Method	RMSE
BASELINE - TEST	0.8310898
MAX TREE DEPTH = 8 - TEST	0.8302726

6.3 MODEL 3 - MAX TREE DEPTH = 10

BECAUSE THERE WAS AN IMPROVEMENT IN MODEL 2, I'M INCREASING THE TREE DEPTH FROM 8 TO 10.

- max_depth = 10
- eta = 0.3 (DEFAULT)
- min_child_weight = 1 (default)

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

y <- train_edx$rating
x <- data.matrix(train_edx[,!("rating")])
d_train <- xgb.DMatrix(data = x, label = y)

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])
d_test <- xgb.DMatrix(data = x, label = y)
```

```r
watchlist <- list(train = d_train, eval = d_test)

fit <- xgb.train(data=d_train,
                 nrounds = 5000,
                 verbose = FALSE,
                 tree_method = 'hist',
                 max_depth = 10,
                 early_stopping_rounds = 20,
                 watchlist = watchlist,
                 eval_metric = 'rmse')

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])

yhat = predict(fit, x)
RMSE_model_III <- RMSE(y, yhat)
```

Table 3: RMSEs

Method	RMSE
BASELINE - TEST	0.8310898
MAX TREE DEPTH = 8 - TEST	0.8302726
MAX TREE DEPTH = 10 - TEST	0.8316796

THERE WAS NOT IMPROVEMENT IN THIS MODEL CHANGING THE TREE DEPTH TO 10. THEN, FOR THE REST OF THE MODELS I'M KEEPING THE TREE DEPTH = 8 AS THE MOST OPTIMAL.

6.4 MODEL 4 - MAX TREE DEPTH = 8 AND MIN CHILD WEIGHT = 300

IN THIS MODEL I'M CHANGING THE CHILD WEIGHT FROM THE DEFAUL 1 TO 300. WILLING AN IMPROVEMENT IN THE RMSE.

- max_depth = 8
- eta = 0.3 (DEFAULT)
- min_child_weight = 300

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

y <- train_edx$rating
x <- data.matrix(train_edx[,!("rating")])
```

```r
d_train <- xgb.DMatrix(data = x, label = y)

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])
d_test <- xgb.DMatrix(data = x, label = y)
```

```r
watchlist <- list(train = d_train, eval = d_test)

fit <- xgb.train(data=d_train,
                 nrounds = 5000,
                 verbose = FALSE,
                 tree_method = 'hist',
                 max_depth = 8,
                 min_child_weight = 300,
                 early_stopping_rounds = 20,
                 watchlist = watchlist,
                 eval_metric = 'rmse')

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])

yhat = predict(fit, x)
RMSE_model_IV <- RMSE(y, yhat)
```

Table 4: RMSEs

Method	RMSE
BASELINE - TEST	0.8310898
MAX TREE DEPTH = 8 - TEST	0.8302726
MAX TREE DEPTH = 10 - TEST	0.8316796
MAX TREE DEPTH = 8 AND CHILD WEIGHT = 300 - TEST	0.8288294

THERE WAS AN IMPROVEMENT IN THESE MODEL CHANGING THE CHILD WEIGHT. THEN, FOR THE REST OF THE MODELS I'M KEEPING THE CHILD WEIGHT = 300 AS THE MOST OPTIMAL.

6.5 MODEL 5 - MAX TREE DEPTH = 8, ETA = 0.1 AND MIN CHILD = 300

FOR THIS MODEL I'M CHANGING THE ETA FROM THE DEFAULT 0.3 TO 0.1. WILLING AN IMPROVEMENT IN THE RMSE.

- max_depth = 8
- eta = 0.1
- min_child_weight = 300

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

y <- train_edx$rating
x <- data.matrix(train_edx[,!("rating")])
```

```r
d_train <- xgb.DMatrix(data = x, label = y)

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])
d_test <- xgb.DMatrix(data = x, label = y)
```

```r
watchlist <- list(train = d_train, eval = d_test)

fit <- xgb.train(data=d_train,
                 nrounds = 5000,
                 verbose = FALSE,
                 tree_method = 'hist',
                 max_depth = 8,
                 min_child_weight = 300,
                 eta = 0.1,
                 early_stopping_rounds = 20,
                 watchlist = watchlist,
                 eval_metric = 'rmse')

y <- test_edx$rating
x <- data.matrix(test_edx[,!("rating")])

yhat = predict(fit, x)
RMSE_model_V <- RMSE(y, yhat)
```

Table 5: RMSEs

Method	RMSE
BASELINE - TEST	0.8310898
MAX TREE DEPTH = 8 - TEST	0.8302726
MAX TREE DEPTH = 10 - TEST	0.8316796
MAX TREE DEPTH = 8 AND CHILD WEIGHT = 300 - TEST	0.8288294
MAX TREE DEPTH = 8, ETA = 0.1, MIN CHILD = 300 - TEST	0.8249370

BECAUSE THIS MODEL REACHED AN ACCEPTABLE RMSE IN TEST. I AM GOING TO USE THE VALIDATION DATASET TO PREDICT THE RATING THAT A USER WILL GIVE TO A MOVIE.

```r
## Validation Dataset
y <- validation$rating
x <- data.matrix(validation[,!("rating")])

yhat = predict(fit, x)
RMSE_model_VI <- RMSE(y, yhat)
```

Table 6: RMSEs

Method	RMSE
BASELINE - TEST	0.8310898
MAX TREE DEPTH = 8 - TEST	0.8302726
MAX TREE DEPTH = 10 - TEST	0.8316796

Method	RMSE
MAX TREE DEPTH = 8 AND CHILD WEIGHT = 300 - TEST	0.8288294
MAX TREE DEPTH = 8, ETA = 0.1, MIN CHILD = 300 - TEST	0.8249370
MAX TREE DEPTH = 8 AND ETA = 0.1, MIN CHILD = 300 - VALIDATION	0.8582224

THIS IS THE **FINAL MODEL** THAT REACHED AN **ACCEPTABLE RMSE IN VALIDATION DATASET** FOR THIS PROJECT.

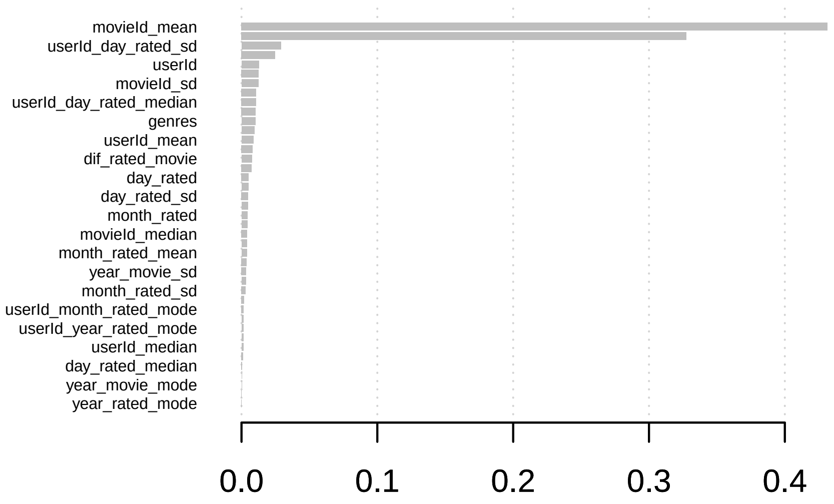
7 RESULT

- THE RMSE TABLE IS A RESULT OF EACH IMPROVEMENT, OR NOT, BETWEEN EACH MODEL.

Table 7: RESULTS

Method	RMSE
BASELINE - TEST	0.8310898
MAX TREE DEPTH = 8 - TEST	0.8302726
MAX TREE DEPTH = 10 - TEST	0.8316796
MAX TREE DEPTH = 8 AND CHILD WEIGHT = 300 - TEST	0.8288294
MAX TREE DEPTH = 8, ETA = 0.1, MIN CHILD = 300 - TEST	0.8249370
MAX TREE DEPTH = 8 AND ETA = 0.1, MIN CHILD = 300 - VALIDATION	0.8582224

- THE IMPORTANCE PLOT SHOWS US THE IMPORTANCE OF EACH FEATURE IN THE FINAL MODEL. THE LONGER IS THE BAR, MORE IMPORTANT IS THE FEATURE. FEATURES ARE CLASSIFIED BY IMPORTANCE AND CLUSTERED BY IMPORTANCE.



8 CONCLUSION

THE DATA SCIENCE MOVIELENS PROJECT REQUESTS TO BUILD A MACHINE LEARNING ALGORITHM TO PREDICT THE USERS RATING FOR A MOVIE.

THE MODEL HAS TO REACH AN ACCEPTABLE RMSE WITHOUT ANY OTHER LIMITATIONS OF HOW TO DO IT.

AFTER REVIEW AND UNDERSTAND THE EXAMPLES PROVIDED BY HARVARD I DECIDED TO KEEP INVESTIGATING OTHER MODELS, SUCH US MATRIX FACTORIZATION AND, THE ONE THAT CAPTURED MY ATTENTION, XGBOOST.

THE LIMITATIONS THAT I HAD WERE THE HARDWARE AND THIS BROUGHT ME ANOTHER ONE “THE TIME”. RUNNING EACH MODEL ON MY COMPUTER HAS TAKEN AT LEAST A DAY!

AS FUTURES IMPROVEMENTS, I WOULD WORK WITH SOME OF THE REGULARIZATION PARAMETERS SUCH AS LAMBDA, GAMMA, ALPHA AND, OTHER ADVANCED PARAMETERS TO TWEAK THE MODEL AND IMPROVE ITS RESULTS. ALSO, CHANGING ETA TO SMALLER VALUES LIKE 0.01 OR 0.001 WOULD HELP CONVERGE TO A BETTER RMSE BUT IT WILL TAKE MORE ROUNDS TO FIND THE BEST RSME.

IN MY RECOMMENDER SYSTEM, THE DEFAULT VALUES USED FOR THE REGULARIZATION PARAMETERS ARE:

- LAMBDA -> DEFAULT = 0.
- ALPHA -> DEFAULT = 0.
- GAMMA -> DEFAULT = 0.

9 REFERENCES

- GENERAL INFORMATION
  - https://rafalab.github.io/dsbook/index.html
- GENERAL INFORMATION IN R
  - https://cran.r-project.org/
- MACHINE LEARNING ALGORITHMS
  - https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/
- FEATURE ENGINEERING
  - https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114
- MATRIX FACTORIZATION
  - https://www.youtube.com/watch?v=ZspR5PZemcs
- XGBOOST
  - https://xgboost.readthedocs.io/en/latest/
  - https://towardsdatascience.com/
  - https://www.kdnuggets.com/2017/10/XGBOOST-CONCISE-TECHNICAL-OVERVIEW.HTML
  - https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/