Project report - Recommender system for movies

Ingrid Brizotti

This project builds recommender systems for movies using Item-Based Collaborative Filtering (IBCF) approach to compare users and items.

The idea is to compare these system using just the information if the user watched the movie or not (called binary in this project) and use the rating gave for each user (called rating). Also, compare different approaches for calculate the similarity between users and items. These measures are: - cosine distance - euclidean distance - Pearson correlation - Jaccard index

Require package and datasets used

```
library(data.table)
library(ggplot2)
library(recommenderlab)
```

```
## Loading required package: Matrix
```

```
## Loading required package: arules
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

```
## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## Loading required package: registry
```

```
library(countrycode)

load("~/Desktop/Ryerson/3.Data_Analytics_Capstone/MovieTweetings/latest/ratings.Rda")
load("~/Desktop/Ryerson/3.Data_Analytics_Capstone/MovieTweetings/latest/movies4.Rda")
```

Data preparation

```
# most common movie
t_m <- aggregate(cbind(count = rating) ~ movie_id,
                 data = ratings,
                 FUN = function(x){NROW(x)})
# merge
r <- merge(x=ratings, y=t_m ,by="movie_id", all.x=TRUE)

# select movies with more than 50 ratings
r <- r[r$count >= 50,]

# count movies per user
t_m_u <- aggregate(cbind(count_movie = movie_id) ~ user_id,
                   data = r,
                   FUN = function(x){NROW(x)})
# merge
r2 <- merge(x=r, y=t_m_u ,by="user_id", all.x=TRUE)

# select users with more than 20 movies
r2 <- r2[r2$count_movie >= 20,]


# delete count and timestamp
r2 <- subset(r2, , -c(rating_timestamp,count,count_movie))

# convert it into a data table
r2 <- data.table(r2)
```

I will start building recommender systems using just the information if the user watched or not the movie. So, 1 if the user rated the movie, and 0 otherwise.

Binary recommender systems

```
# delete rating
r_binary <- subset(r2, , -c(rating))

# reshape
r_binary[, value := 1]
r_wide <- reshape(data = r_binary,
                  direction = "wide",
                  idvar = "user_id",
                  timevar = "movie_id",
                  v.names = "value",
                  drop = NULL)

# keep only the columns containing ratings
# the user name will be the matrix row names, so we need to store them in the vector_users vector
```

```
vector_users <- r_wide[, user_id]
r_wide <- r_wide[ ,user_id := NULL]


# have the column names equal to the item names
setnames(x = r_wide,
         old = names(r_wide),
         new = substring(names(r_wide), 7))

# store the rating matrix within a recommenderlab object:
# 1) convert r_wide in a matrix
# 2) set the row names equal to the user names
matrix_wide <- as.matrix(r_wide)
rownames(matrix_wide) <- vector_users

# replace NA for zero
matrix_wide[is.na(matrix_wide)] <- 0
head(matrix_wide[, 1:6])
```

```
##    816711 2726560 3079380 1091191 2381249 1398426
## 2       1       1       1       1       1       1
## 18      0       0       0       1       0       0
## 26      1       0       0       0       0       0
## 38      0       0       0       0       0       0
## 48      0       0       0       0       0       0
## 49      0       0       1       0       0       0
```

```
# coercing matrix_wide into a binary rating matrix
ratings_matrix <- as(matrix_wide, "binaryRatingMatrix")
ratings_matrix
```

```
## 4938 x 1788 rating matrix of class 'binaryRatingMatrix' with 287862 ratings.
```

Compare recommender sytems using different approaches to calculate the similarity

```
# split the data into the training and the test set
which_train <- sample(x = c(TRUE, FALSE),
                      size = nrow(ratings_matrix),
                      replace = TRUE,
                      prob = c(0.7, 0.3))
recc_data_train <- ratings_matrix[which_train, ]
recc_data_test <- ratings_matrix[!which_train, ]

percentage_training <- 0.7

# minimum number of movies watched by any user
min(rowCounts(ratings_matrix))
```

```
## [1] 20
```

```r
# [1] 20

# keep 15 movies
items_to_keep <- 15

# the rating = 1 (watched the movie) will be considered good to evaluate the model
rating_threshold <- 1

# how many times we want to run the evaluation
n_eval <- 1

eval_sets <- evaluationScheme(data = ratings_matrix,
                              method = "split",
                              train = percentage_training,
                              given = items_to_keep,
                              goodRating =rating_threshold,
                              k = n_eval)

models_to_evaluate <- list(
  IBCF_jac = list(name = "IBCF", param = list(method = "jaccard")),
  IBCF_cos = list(name = "IBCF", param = list(method = "cosine")),
  IBCF_cor = list(name = "IBCF", param = list(method = "pearson")),
  IBCF_euc = list(name = "IBCF", param = list(method = "euclidean")),

  UBCF_jac = list(name = "UBCF", param = list(method = "jaccard")),
  UBCF_cos = list(name = "UBCF", param = list(method = "cosine")),
  UBCF_cor = list(name = "UBCF", param = list(method = "pearson")),
  UBCF_euc = list(name = "UBCF", param = list(method = "euclidean"))
#  ,random = list(name = "RANDOM", param=NULL)
)

# number of recommendations
n_recommendations <- c(1, 5, seq(10, 30, 5))

# We are ready to run and evaluate the models.
# The only difference from code 5.evaluate binary CF is now the input method is a list of models
list_results <- evaluate(x = eval_sets,
                         method = models_to_evaluate,
                         n = n_recommendations)
```

```
## IBCF run fold/sample [model time/prediction time]
##   1  [63.83sec/1.502sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [61.962sec/1.934sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [51.393sec/1.484sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [109.019sec/0.888sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.002sec/43.387sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.001sec/42.397sec]
## UBCF run fold/sample [model time/prediction time]
```

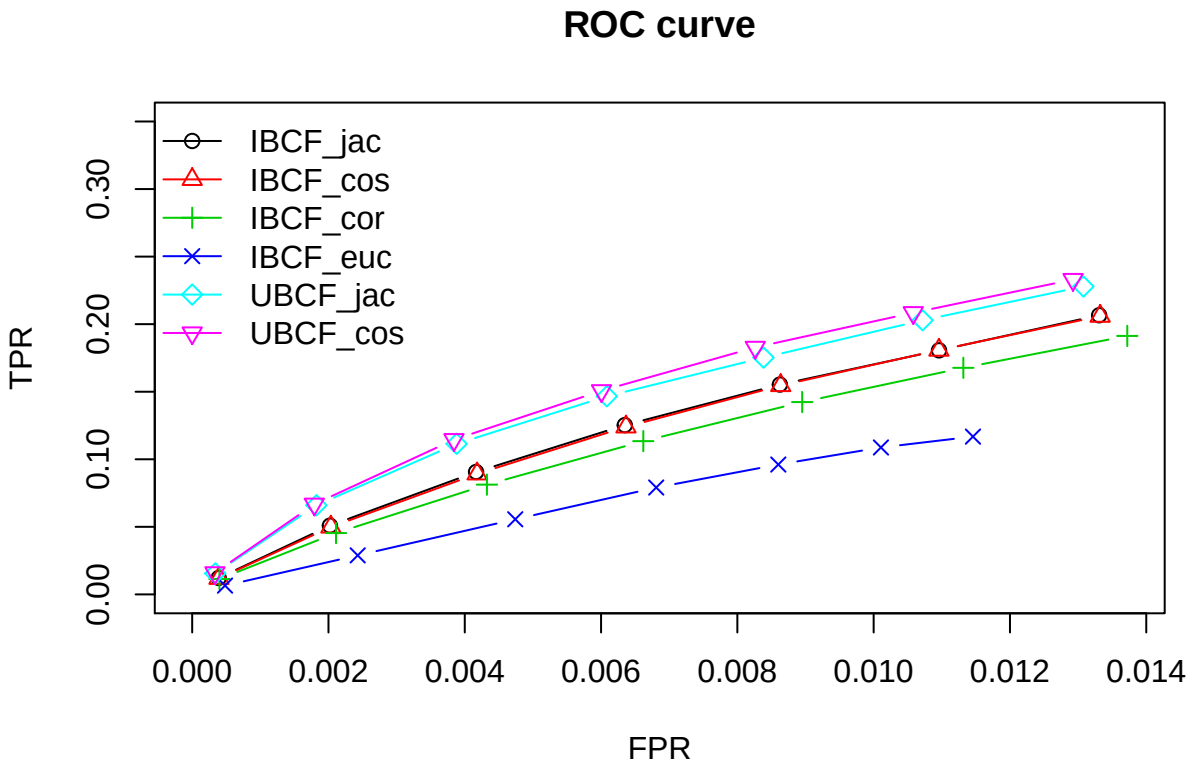
```
##   1  Timing stopped at: 18.449 0.747 19.828
## UBCF run fold/sample [model time/prediction time]
##   1  Timing stopped at: 0.044 0.026 0.072
```

```
## Warning in .local(x, method, ...):
##   Recommender 'UBCF_cor' has failed and has been removed from the results!
##   Recommender 'UBCF_euc' has failed and has been removed from the results!
```

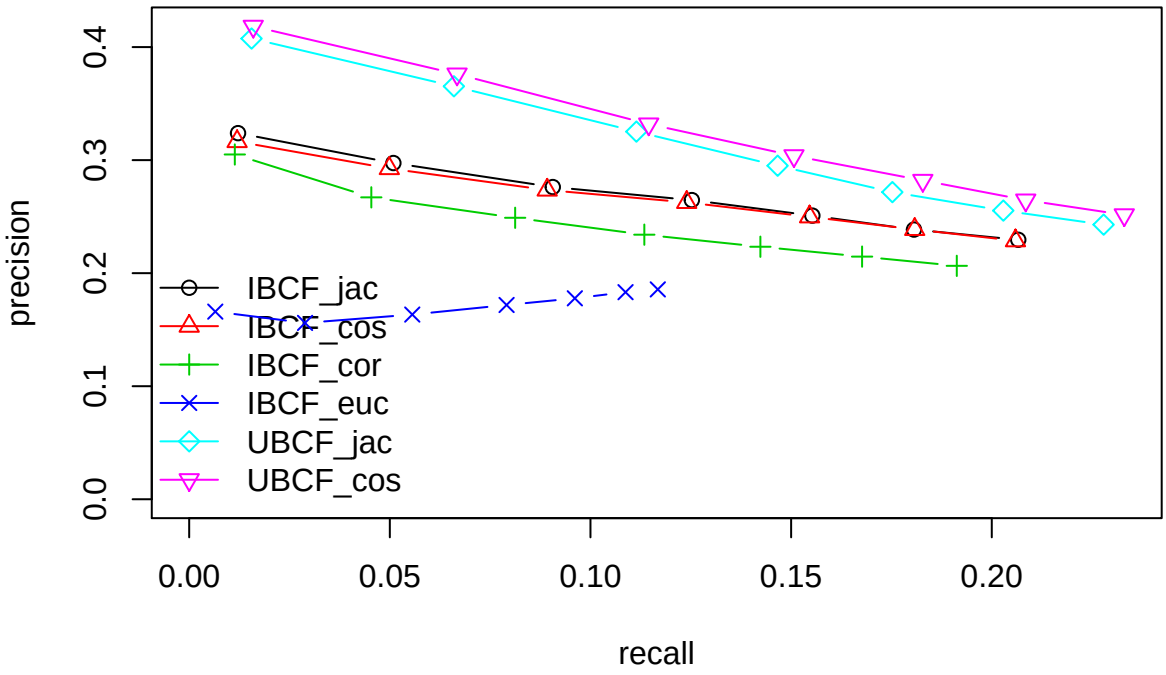
```
class(list_results)
```

```
## [1] "evaluationResultList"
## attr(,"package")
## [1] "recommenderlab"
```

```
# Calculate the ROC curve
plot(list_results, legend = "topleft",ylim = c(0, 0.35))
title("ROC curve")
```




```
# Calculate the precision and recall
plot(list_results, "prec/rec", legend = "bottomleft")
title("Precision-recall binary models")
```




```
# average confusion matrix
avg_matrices <- lapply(list_results, avg)

# IBCF cosine distance
head(avg_matrices$IBCF_cos[,1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.3164642  0.6835358 45.05128 1726.949 0.3164642 0.01194274 0.01194274
## 5  1.4628880  3.5371120 43.90486 1724.095 0.2925776 0.04987898 0.04987898
## 10 2.7354926  7.2645074 42.63225 1720.368 0.2735493 0.08921586 0.08921586
## 15 3.9392713 11.0607287 41.42848 1716.572 0.2626181 0.12397890 0.12397890
## 20 4.9973009 15.0026991 40.37045 1712.630 0.2498650 0.15459799 0.15459799
## 25 5.9676113 19.0323887 39.40013 1708.600 0.2387045 0.18079897 0.18079897
##             FPR
## 1  0.0003934291
## 5  0.0020353745
## 10 0.0041810570
## 15 0.0063657448
## 20 0.0086363172
## 25 0.0109575799
```

```
# IBCF Pearson correlation
head(avg_matrices$IBCF_cor[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.3049933  0.6950067 45.06275 1726.937 0.3049933 0.01137293 0.01137293
## 5  1.3353576  3.6646424 44.03239 1723.968 0.2670715 0.04538827 0.04538827
## 10 2.4898785  7.5101215 42.87787 1720.122 0.2489879 0.08123494 0.08123494
```

```
## 15 3.5107962 11.4892038 41.85695 1716.143 0.2340531 0.11341921 0.11341921
## 20 4.4676113 15.5323887 40.90013 1712.100 0.2233806 0.14234366 0.14234366
## 25 5.3663968 19.6336032 40.00135 1707.999 0.2146559 0.16768605 0.16768605
##             FPR
## 1  0.0004000936
## 5  0.0021102720
## 10 0.0043253723
## 15 0.0066196578
## 20 0.0089512772
## 25 0.0113157768
```

```
# IBCF jaccard
head(avg_matrices$IBCF_jac[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.3238866  0.6761134 45.04386 1726.956 0.3238866 0.01217523 0.01217523
## 5  1.4871795  3.5128205 43.88057 1724.119 0.2974359 0.05088415 0.05088415
## 10 2.7618084  7.2381916 42.60594 1720.394 0.2761808 0.09060615 0.09060615
## 15 3.9696356 11.0303644 41.39811 1716.602 0.2646424 0.12526513 0.12526513
## 20 5.0182186 14.9817814 40.34953 1712.650 0.2509109 0.15529337 0.15529337
## 25 5.9642375 19.0357625 39.40351 1708.596 0.2385695 0.18061832 0.18061832
##             FPR
## 1  0.0003891262
## 5  0.0020213464
## 10 0.0041659378
## 15 0.0063482765
## 20 0.0086240180
## 25 0.0109594116
```

```
# IBCF Eucliden
head(avg_matrices$IBCF_euc[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.1659919  0.8340081 45.20175 1726.798 0.1659919 0.00650464 0.00650464
## 5  0.7746289  4.2024291 44.59312 1723.430 0.1559829 0.02887252 0.02887252
## 10 1.5742240  8.2004049 43.79352 1719.432 0.1633943 0.05559240 0.05559240
## 15 2.3481781 11.7746289 43.01957 1715.858 0.1718435 0.07908990 0.07908990
## 20 2.9770580 14.8663968 42.39069 1712.766 0.1777548 0.09610670 0.09610670
## 25 3.4635628 17.4628880 41.90418 1710.169 0.1831799 0.10873960 0.10873960
##             FPR
## 1  0.0004820242
## 5  0.0024290700
## 10 0.0047406859
## 15 0.0068092014
## 20 0.0086007651
## 25 0.0101067197
```

```
# UBCF cosine distance
head(avg_matrices$UBCF_cos[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.4183536  0.5816464 44.94939 1727.051 0.4183536 0.01596944 0.01596944
```

```
## 5  1.8805668  3.1194332 43.48718 1724.513 0.3761134 0.06675333 0.06675333
## 10 3.3191633  6.6808367 42.04858 1720.951 0.3319163 0.11451138 0.11451138
## 15 4.5573549 10.4426451 40.81039 1717.190 0.3038237 0.15070815 0.15070815
## 20 5.6403509 14.3596491 39.72740 1713.273 0.2820175 0.18282154 0.18282154
## 25 6.6160594 18.3839406 38.75169 1709.248 0.2646424 0.20846502 0.20846502
##             FPR
## 1  0.0003339639
## 5  0.0017923477
## 10 0.0038414728
## 15 0.0060063893
## 20 0.0082629222
## 25 0.0105814364
```

```
# UBCF jaccard
head(avg_matrices$UBCF_jac[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.4075574  0.5924426 44.96019 1727.040 0.4075574 0.01555400 0.01555400
## 5  1.8265857  3.1734143 43.54116 1724.459 0.3653171 0.06597394 0.06597394
## 10 3.2523617  6.7476383 42.11538 1720.885 0.3252362 0.11144961 0.11144961
## 15 4.4257760 10.5742240 40.94197 1717.058 0.2950517 0.14664062 0.14664062
## 20 5.4338731 14.5661269 39.93387 1713.066 0.2716937 0.17523697 0.17523697
## 25 6.3839406 18.6160594 38.98381 1709.016 0.2553576 0.20288348 0.20288348
##             FPR
## 1  0.0003401687
## 5  0.0018238739
## 10 0.0038807414
## 15 0.0060850176
## 20 0.0083846592
## 25 0.0107185442
```

Now, I will use the rating information to build the recommender systems.

```
# Data preparation
# reshape
r_wide <- reshape(data = r2,
                  direction = "wide",
                  idvar = "user_id",
                  timevar = "movie_id",
                  drop = NULL)

head(r_wide[, 1:5, with = FALSE])
```

```
##    user_id rating.816711 rating.2726560 rating.3079380 rating.1091191
## 1:       2             8              9              8              7
## 2:      18            NA             NA             NA              8
## 3:      26             5             NA             NA             NA
## 4:      38            NA             NA             NA             NA
## 5:      48            NA             NA             NA             NA
## 6:      49            NA             NA             10             NA
```

```
vector_users <- r_wide[, user_id]
r_wide <- r_wide[ ,user_id := NULL]

# have the column names equal to the item names
setnames(x = r_wide,
         old = names(r_wide),
         new = substring(names(r_wide), 7))

matrix_wide <- as.matrix(r_wide)
rownames(matrix_wide) <- vector_users
head(matrix_wide[, 1:6])
```

```
##    .816711 .2726560 .3079380 .1091191 .2381249 .1398426
## 2        8        9        8        7        5        8
## 18      NA       NA       NA        8       NA       NA
## 26       5       NA       NA       NA       NA       NA
## 38      NA       NA       NA       NA       NA       NA
## 48      NA       NA       NA       NA       NA       NA
## 49      NA       NA       10       NA       NA       NA
```

```
# coercing matrix_wide into a binary rating matrix
ratings_matrix <- as(matrix_wide, "realRatingMatrix")
ratings_matrix
```

```
## 4938 x 1788 rating matrix of class 'realRatingMatrix' with 287862 ratings.
```

```
# Train and test
which_train <- sample(x = c(TRUE, FALSE),
                      size = nrow(ratings_matrix),
                      replace = TRUE,
                      prob = c(0.7, 0.3))
recc_data_train <- ratings_matrix[which_train, ]
recc_data_test <- ratings_matrix[!which_train, ]

eval_sets <- evaluationScheme(data = ratings_matrix,
                              method = "split",
                              train = percentage_training,
                              given = items_to_keep,
                              goodRating =rating_threshold,
                              k = n_eval)

models_to_evaluate <- list(
  IBCF_jac = list(name = "IBCF", param = list(method = "jaccard")),
  IBCF_cos = list(name = "IBCF", param = list(method = "cosine")),
  IBCF_cor = list(name = "IBCF", param = list(method = "pearson")),
  IBCF_euc = list(name = "IBCF", param = list(method = "euclidean")),

  UBCF_jac = list(name = "UBCF", param = list(method = "jaccard")),
  UBCF_cos = list(name = "UBCF", param = list(method = "cosine")),
  UBCF_cor = list(name = "UBCF", param = list(method = "pearson")),
  UBCF_euc = list(name = "UBCF", param = list(method = "euclidean"))
#  ,random = list(name = "RANDOM", param=NULL)
```

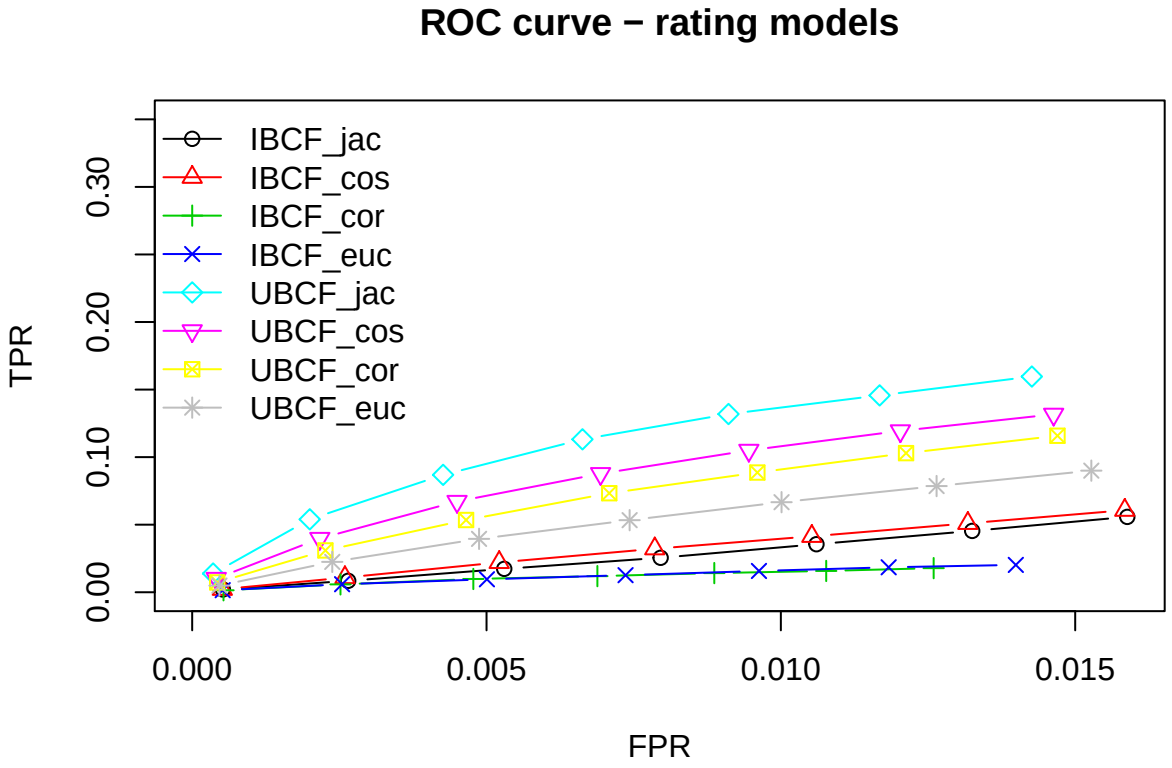
```
)

list_results <- evaluate(x = eval_sets,
                         method = models_to_evaluate,
                         n = n_recommendations)
```

```
## IBCF run fold/sample [model time/prediction time]
##   1  [81.655sec/2.628sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [116.702sec/2.005sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [66.172sec/1.344sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [86.755sec/1.342sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.042sec/98.061sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.042sec/143.112sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.042sec/38.178sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.04sec/90.231sec]
```

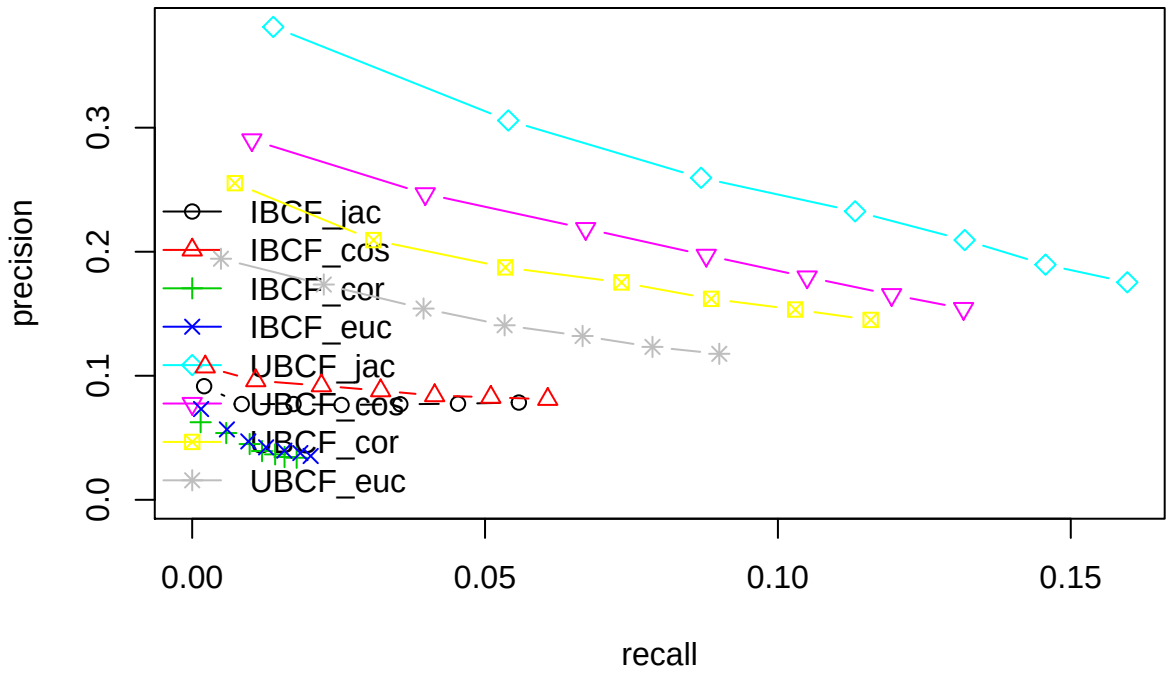
```
class(list_results)
```

```
## [1] "evaluationResultList"
## attr(,"package")
## [1] "recommenderlab"
```

```
# Calculate the ROC curve
plot(list_results, legend = "topleft",ylim = c(0, 0.35))
title("ROC curve - rating models")
```




```r
# Calculate the precision and recall
plot(list_results, "prec/rec", legend = "bottomleft")
title("Precision-recall rating models")
```



```r
# average confusion matrix
avg_matrices <- lapply(list_results, avg)

# IBCF cosine distance
head(avg_matrices$IBCF_cos[,1:8])
```

```
##           TP         FP       FN       TN  precision      recall
## 1  0.1066127  0.8879892 44.86707 1727.138 0.10719132 0.002231592
## 5  0.4777328  4.4952767 44.49595 1723.531 0.09606513 0.010857502
## 10 0.9149798  9.0310391 44.05870 1718.995 0.09199457 0.022062247
## 15 1.3124157 13.6066127 43.66127 1714.420 0.08796924 0.032177022
## 20 1.6707152 18.2213225 43.30297 1709.805 0.08398915 0.041397161
## 25 2.0566802 22.8083671 42.91700 1705.218 0.08271370 0.050984408
##            TPR          FPR
## 1  0.002231592 0.0005122931
## 5  0.010857502 0.0025950535
## 10 0.022062247 0.0052153509
## 15 0.032177022 0.0078584410
## 20 0.041397161 0.0105257930
## 25 0.050984408 0.0131759576
```

```r
# IBCF Pearson correlation
head(avg_matrices$IBCF_cor[, 1:8])
```

```
##            TP         FP       FN       TN  precision      recall
## 1  0.06140351  0.9203779 44.91228 1727.106 0.06254296 0.001446846
## 5  0.26045884  4.3569501 44.71323 1723.669 0.05382589 0.005808662
## 10 0.42240216  8.2516869 44.55128 1719.775 0.04495173 0.009813845
## 15 0.53171390 11.8846154 44.44197 1716.142 0.03931299 0.011922175
## 20 0.63292848 15.3070175 44.34076 1712.719 0.03663704 0.014163713
## 25 0.71524966 18.5829960 44.25843 1709.443 0.03448819 0.015765982
##            TPR          FPR
## 1  0.001446846 0.0005321392
## 5  0.005808662 0.0025192432
## 10 0.009813845 0.0047761147
## 15 0.011922175 0.0068831885
## 20 0.014163713 0.0088692688
## 25 0.015765982 0.0107704038
```

```r
# IBCF jaccard
head(avg_matrices$IBCF_jac[, 1:8])
```

```
##            TP         FP       FN       TN  precision      recall
## 1  0.09109312  0.9035088 44.88259 1727.123 0.09158752 0.002038986
## 5  0.38394062  4.5890688 44.58974 1723.437 0.07720488 0.008455769
## 10 0.76788124  9.1781377 44.20580 1718.848 0.07720488 0.017305251
## 15 1.14035088 13.7786775 43.83333 1714.248 0.07643600 0.025502802
## 20 1.53238866 18.3596491 43.44130 1709.667 0.07703528 0.035529894
## 25 1.92645074 22.9385965 43.04723 1705.088 0.07747626 0.045382171
##            TPR          FPR
## 1  0.002038986 0.0005213819
## 5  0.008455769 0.0026501149
```

```
## 10 0.017305251 0.0053011412
## 15 0.025502802 0.0079579261
## 20 0.035529894 0.0106047162
## 25 0.045382171 0.0132502283
```

```
# IBCF Eucliden
head(avg_matrices$IBCF_euc[, 1:8])
```

```
##           TP         FP       FN       TN  precision      recall
## 1  0.0708502  0.8974359 44.90283 1727.129 0.07317073 0.001521440
## 5  0.2719298  4.4021592 44.70175 1723.624 0.05672474 0.005929194
## 10 0.4433198  8.6491228 44.53036 1719.377 0.04708534 0.009585519
## 15 0.5843455 12.7125506 44.38934 1715.314 0.04227051 0.012623287
## 20 0.7179487 16.6214575 44.25574 1711.405 0.03963719 0.015736500
## 25 0.8380567 20.4203779 44.13563 1707.606 0.03766949 0.018465357
##            TPR          FPR
## 1  0.001521440 0.0005185225
## 5  0.005929194 0.0025466127
## 10 0.009585519 0.0050064645
## 15 0.012623287 0.0073618511
## 20 0.015736500 0.0096279057
## 25 0.018465357 0.0118314209
```

```
# UBCF cosine distance
head(avg_matrices$UBCF_cos[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.2901484  0.7098516 44.68354 1727.316 0.2901484 0.01018963 0.01018963
## 5  1.2334683  3.7665317 43.74022 1724.260 0.2466937 0.03979122 0.03979122
## 10 2.1862348  7.8137652 42.78745 1720.213 0.2186235 0.06718063 0.06718063
## 15 2.9561404 12.0438596 42.01754 1715.982 0.1970760 0.08776730 0.08776730
## 20 3.5917679 16.4082321 41.38192 1711.618 0.1795884 0.10498868 0.10498868
## 25 4.1342780 20.8657220 40.83941 1707.161 0.1653711 0.11941242 0.11941242
##             FPR
## 1  0.0004087764
## 5  0.0021682408
## 10 0.0044995445
## 15 0.0069378490
## 20 0.0094563526
## 25 0.0120302592
```

```
# UBCF jaccard
head(avg_matrices$UBCF_jac[, 1:8])
```

```
##           TP         FP       FN       TN precision     recall        TPR
## 1  0.3812416  0.6187584 44.59244 1727.408 0.3812416 0.01384676 0.01384676
## 5  1.5290148  3.4709852 43.44467 1724.555 0.3058030 0.05396808 0.05396808
## 10 2.5958165  7.4041835 42.37787 1720.622 0.2595816 0.08688196 0.08688196
## 15 3.4885290 11.5114710 41.48516 1716.515 0.2325686 0.11318172 0.11318172
## 20 4.1882591 15.8117409 40.78543 1712.215 0.2094130 0.13189186 0.13189186
## 25 4.7395412 20.2604588 40.23414 1707.766 0.1895816 0.14568772 0.14568772
##             FPR
```

```
## 1  0.0003555711
## 5  0.0019977789
## 10 0.0042627475
## 15 0.0066302096
## 20 0.0091104678
## 25 0.0116780869
```

```
# UBCF euclidean
head(avg_matrices$UBCF_euc[, 1:8])
```

```
##           TP        FP       FN       TN precision     recall        TPR
## 1  0.1943320  0.805668 44.77935 1727.221 0.1943320 0.00491603 0.00491603
## 5  0.8677463  4.132254 44.10594 1723.894 0.1735493 0.02247098 0.02247098
## 10 1.5418354  8.458165 43.43185 1719.568 0.1541835 0.03950176 0.03950176
## 15 2.1106613 12.889339 42.86302 1715.137 0.1407108 0.05333939 0.05333939
## 20 2.6410256 17.358974 42.33266 1710.667 0.1320513 0.06665376 0.06665376
## 25 3.0816464 21.918354 41.89204 1706.108 0.1232659 0.07858338 0.07858338
##             FPR
## 1  0.0004636386
## 5  0.0023796893
## 10 0.0048732083
## 15 0.0074307641
## 20 0.0100116683
## 25 0.0126453276
```

```
# UBCF correlation
head(avg_matrices$UBCF_cor[, 1:8])
```

```
##           TP         FP       FN       TN precision      recall
## 1  0.2537112  0.7402159 44.71997 1727.286 0.2552614 0.007281998
## 5  1.0404858  3.9291498 43.93320 1724.097 0.2093686 0.030941939
## 10 1.8623482  8.0769231 43.11134 1719.949 0.1873727 0.053461079
## 15 2.6126856 12.2962213 42.36100 1715.730 0.1752433 0.073286201
## 20 3.2206478 16.6578947 41.75304 1711.368 0.1620163 0.088599897
## 25 3.8117409 21.0364372 41.16194 1706.990 0.1534012 0.102957772
##            TPR         FPR
## 1  0.007281998 0.000425691
## 5  0.030941939 0.002262984
## 10 0.053461079 0.004652751
## 15 0.073286201 0.007084640
## 20 0.088599897 0.009600603
## 25 0.102957772 0.012126494
```