DATA 612 Assignment #4

Brian K. Liles

July 2, 2019

Goal

The goal of this assignment is give you practice working with accuracy and other recommender system metrics.

Deliverables

As in your previous assignments, compare the accuracy of at least two recommender system algorithms against your offline data.

Implement support for at least one business or user experience goal such as increased serendipity, novelty, or diversity.

Compare and report on any change in accuracy before and after you've made the change in #2.

As part of your textual conclusion, discuss one or more additional experiments that could be performed and/or metrics that could be evaluated only if online evaluation was possible. Also, briefly propose how you would design a reasonable online evaluation environment.

Libraries

Import Data

From the **recommenderlab** package, and with the guidance of the text **Building a Recommendation System with R** the data for use will be the **MovieLense**

```
data(MovieLense)
MovieLense
```

```
## 943 x 1664 rating matrix of class 'realRatingMatrix' with 99392 ratings.
```

```
# create a new variable entitled MSWebDF in order to take a look at the data
mlDF <- as(MovieLense, 'data.frame')
head(mlDF)
```

```
##     user                                                 item rating
## 1      1                                     Toy Story (1995)      5
## 453    1                                     GoldenEye (1995)      3
## 584    1                                    Four Rooms (1995)      4
## 674    1                                    Get Shorty (1995)      3
## 883    1                                       Copycat (1995)      3
## 969    1 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)      5
```

```
# use the glimpse function to look at the newly created data frame
glimpse(mlDF)
```

```
## Observations: 99,392
## Variables: 3
## $ user   <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ item   <fctr> Toy Story (1995), GoldenEye (1995), Four Rooms (1995),...
## $ rating <dbl> 5, 3, 4, 3, 3, 5, 4, 1, 5, 3, 2, 5, 5, 5, 5, 5, 3, 4, 5...
```

```
# summarize the rating variable
summary(mlDF$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.00    4.00    3.53    4.00    5.00
```

Based off the findings from the **glimpse** and **summary** functions we see that the **MovieLense** dataset has 99,392 observations and 3 variables. In addition, the **ratings** column has a mean of 3.53, median of 4.00; datais slightly skewed to the left and scored on a scale from 1 to 5.

Similarity Matrix

```
similarity_users <- similarity(MovieLense[1:4,],
                              method = "cosine",
                              which = "users")
cat("Similarity Users Matrix Output","\n")
```

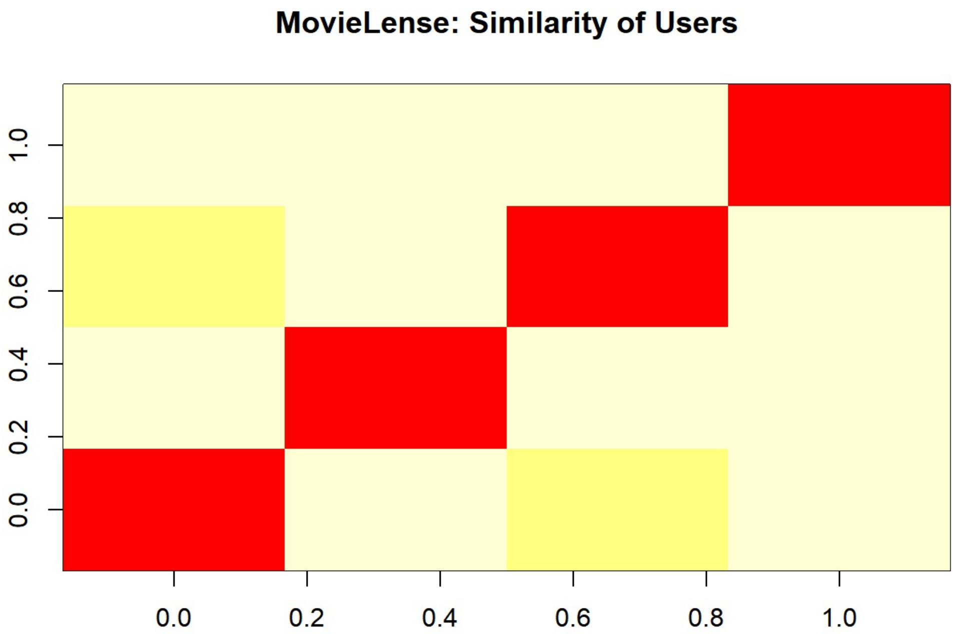
```
## Similarity Users Matrix Output
```

```
(similarityMatrix <- as.matrix(similarity_users))
```

```
##           1         2         3         4
## 1 0.0000000 0.9605820 0.8339504 0.9192637
## 2 0.9605820 0.0000000 0.9268716 0.9370341
## 3 0.8339504 0.9268716 0.0000000 0.9130323
## 4 0.9192637 0.9370341 0.9130323 0.0000000
```

User similarity can also be viewed using the **image** function

```
image(as.matrix(similarity_users), main = "MovieLense: Similarity of Users")
```



Recommender Models

The **recommenderlab** package has several algorithms which can create recommender models

```
recommender_models <- recommenderRegistry$get_entries(
    dataType = "realRatingMatrix")
names(recommender_models)
```

```
## [1] "ALS_realRatingMatrix"          "ALS_implicit_realRatingMatrix"
## [3] "IBCF_realRatingMatrix"         "POPULAR_realRatingMatrix"
## [5] "RANDOM_realRatingMatrix"       "RERECOMMEND_realRatingMatrix"
## [7] "SVD_realRatingMatrix"          "SVDF_realRatingMatrix"
## [9] "UBCF_realRatingMatrix"
```

According to the textbook, a rating equal to 0 represented a missing value. In addition, the **summary** function stated the column held values from 1 to 5; based off the text we will visualize the ratings

```
ratings <- as.vector(MovieLense@data)
cat("Table of MOvie Lense Ratings")
```

```
## Table of MOvie Lense Ratings
```

```
(table_ratings <- table(ratings))
```

```
## ratings
##       0       1       2       3       4       5
## 1469760    6059   11307   27002   33947   21077
```

When creating a recommender system, a potential customer would feel more comfortable with information from reliable sources. In the text, users who have rated at least 50 movies and watched 100 were used.

```
ratings_movies <- MovieLense[rowCounts(MovieLense) > 50,colCounts(MovieLense) > 100]
ratings_movies
```

```
## 560 x 332 rating matrix of class 'realRatingMatrix' with 55298 ratings.
```

Create Training Sets & Models for Comparison

Based off the **ratings_movies** we will build models based off k-fold to validate models. In the text a rating_threshold was set at 3, which is slightly under the mean which was 3.53.

```
test <- evaluationScheme(ratings_movies, method = "split", train = 0.8, k = 4, given = 15, goodRating = 3)

# method: this is the way to split the data
# train: this is the percentage of data in the training set
# given: number of items to keep
# goodRating: rating threshold
# k: number of times to run the evaluation
```

The next step is to evaluate the models using the IBCF, UBCF, ALS, POPULAR methods.In addition, the measures of accuracy will also be performed for each method: RMSE, MSE, MAE

IBCF

```
# IBCF Models
ibcf_recMod <- Recommender(getData(test, "train"), "IBCF")
ibcf_pred <- predict(ibcf_recMod, getData(test, "known"), type = "ratings")
cat("IBCF Method: RMSE, MSE, MAE","\n","\n")
```

```
## IBCF Method: RMSE, MSE, MAE
##
```

```
(ibcf <- calcPredictionAccuracy(ibcf_pred, getData(test, "unknown")))
```

```
##     RMSE      MSE      MAE
## 1.410003 1.988109 1.079129
```

```
ibcf_results <- evaluate(test, method  = "IBCF", n = seq(10,100,10))
```

```
## IBCF run fold/sample [model time/prediction time]
##   1  [1.02sec/0.75sec]
##   2  [0.91sec/0.24sec]
##   3  [0.94sec/0.14sec]
##   4  [0.93sec/0.15sec]
```

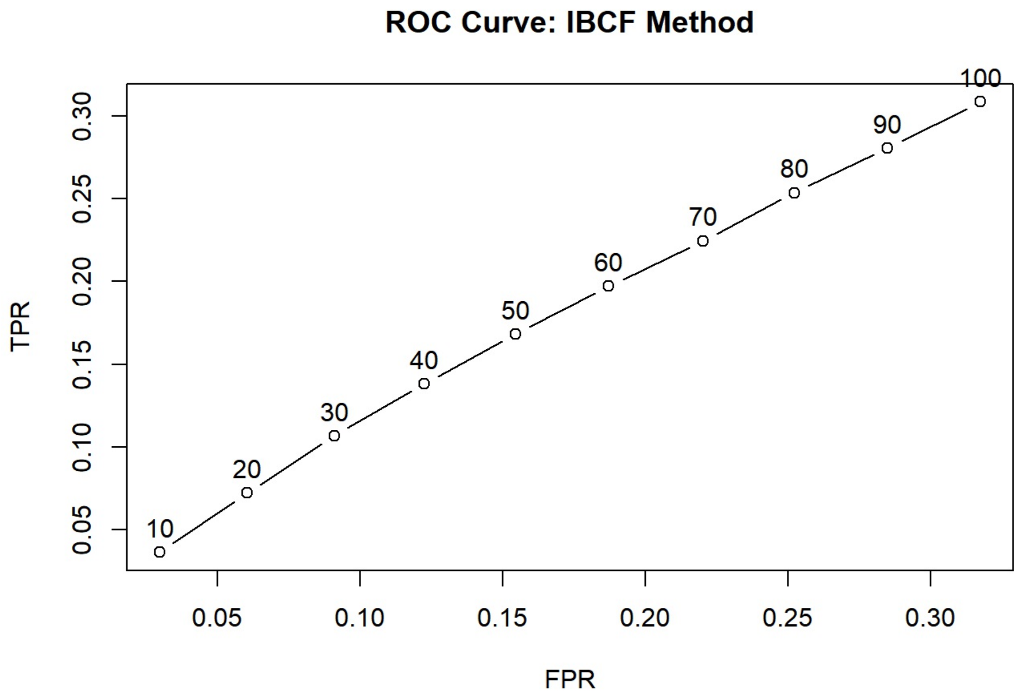
```
class(ibcf_results)
```

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

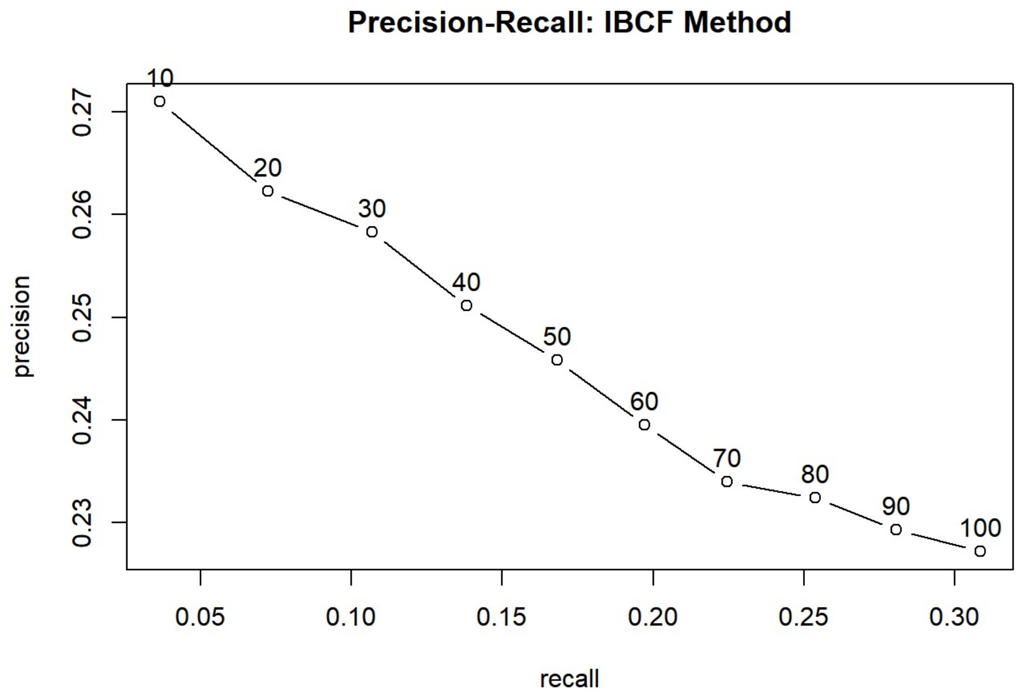
```
head(getConfusionMatrix(ibcf_results)[[1]])
```

```
##            TP        FP       FN       TN precision     recall        TPR
## 10  2.410714  7.589286 64.74107 242.2589 0.2410714 0.03346173 0.03346173
## 20  4.714286 15.285714 62.43750 234.5625 0.2357143 0.07155819 0.07155819
## 30  6.794643 23.205357 60.35714 226.6429 0.2264881 0.10386921 0.10386921
## 40  8.767857 31.232143 58.38393 218.6161 0.2191964 0.13313444 0.13313444
## 50 10.839286 39.160714 56.31250 210.6875 0.2167857 0.16319675 0.16319675
## 60 12.848214 47.151786 54.30357 202.6964 0.2141369 0.19429792 0.19429792
##           FPR
## 10 0.03010303
## 20 0.06095402
## 30 0.09288718
## 40 0.12529073
## 50 0.15694280
## 60 0.18916946
```

```
plot(ibcf_results, annotate = TRUE, main = "ROC Curve: IBCF Method")
```




```
plot(ibcf_results, "prec/rec", annotate = TRUE, main = "Precision-Recall: IBCF Method")
```



ALS

```
# ALS Models
als_recMod <- Recommender(getData(test, "train"), "ALS")
als_pred <- predict(als_recMod, getData(test, "known"), type = "ratings")
cat("ALS Method: RMSE, MSE, MAE","\n","\n")
```

```
## ALS Method: RMSE, MSE, MAE
##
```

```
(als <- calcPredictionAccuracy(als_pred, getData(test, "unknown")))
```

```
##      RMSE       MSE       MAE
## 0.9533999 0.9089713 0.7562991
```

```
als_results <- evaluate(test, method  = "ALS", n = seq(10,100,10))
```

```
## ALS run fold/sample [model time/prediction time]
##   1  [0.2sec/36.33sec]
##   2  [0sec/36sec]
##   3  [0sec/36.39sec]
##   4  [0sec/35.49sec]
```

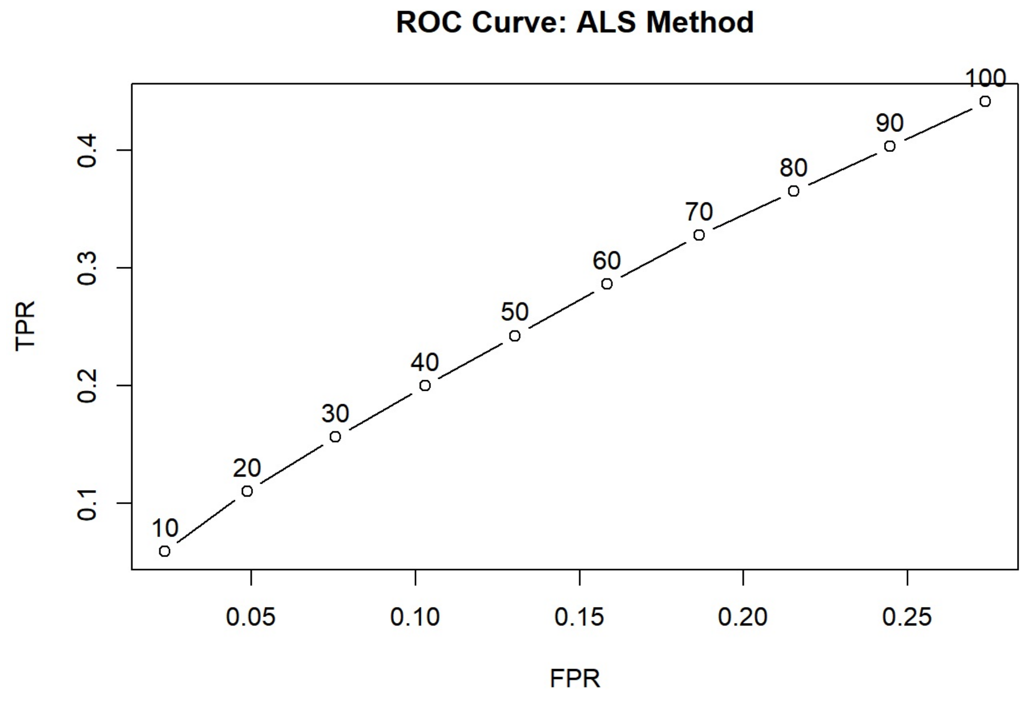
```
class(als_results)
```

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

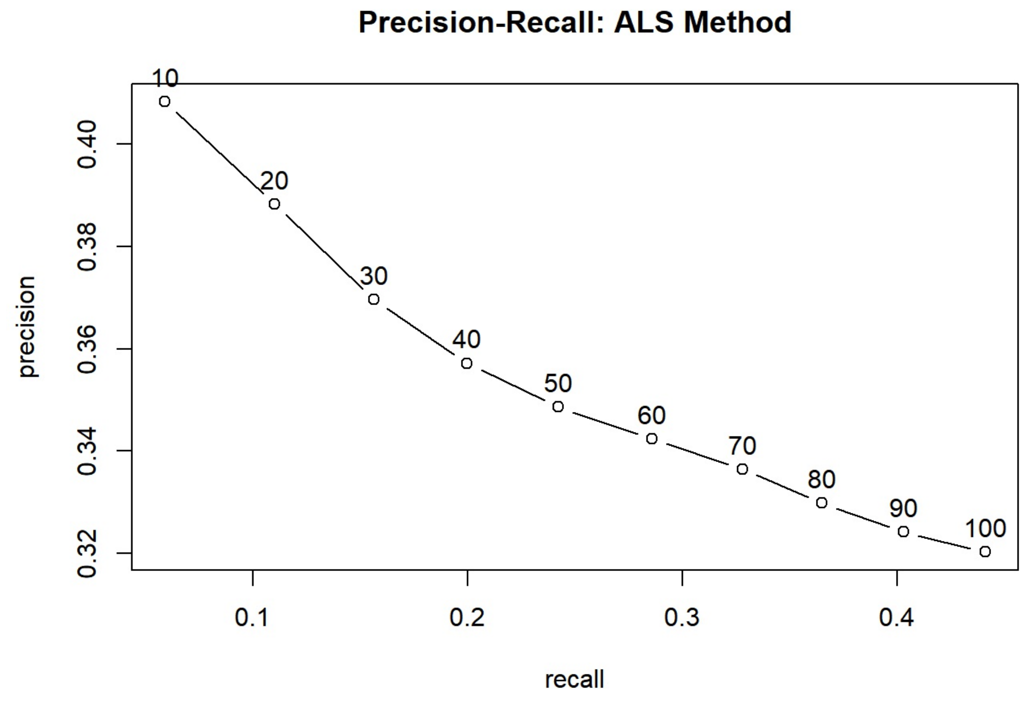
```
head(getConfusionMatrix(als_results)[[1]])
```

```
##           TP        FP       FN       TN precision     recall        TPR
## 10  3.607143  6.392857 63.54464 243.4554 0.3607143 0.05652401 0.05652401
## 20  7.017857 12.982143 60.13393 236.8661 0.3508929 0.10737235 0.10737235
## 30 10.008929 19.991071 57.14286 229.8571 0.3336310 0.15066021 0.15066021
## 40 12.928571 27.071429 54.22321 222.7768 0.3232143 0.19177430 0.19177430
## 50 15.705357 34.294643 51.44643 215.5536 0.3141071 0.23220013 0.23220013
## 60 18.508929 41.491071 48.64286 208.3571 0.3084821 0.27514931 0.27514931
##           FPR
## 10 0.02508648
## 20 0.05078683
## 30 0.07833238
## 40 0.10619151
## 50 0.13461937
## 60 0.16311340
```

```
plot(als_results, annotate = TRUE, main = "ROC Curve: ALS Method")
```




```
plot(als_results, "prec/rec", annotate = TRUE, main = "Precision-Recall: ALS Method")
```



UBCF

```
# UBCF Models
ubcf_recMod <- Recommender(getData(test, "train"), "UBCF")
ubcf_pred <- predict(ubcf_recMod, getData(test, "known"), type = "ratings")
cat("UBCF Method: RMSE, MSE, MAE","\n","\n")
```

```
## UBCF Method: RMSE, MSE, MAE
##
```

```
(ubcf <- calcPredictionAccuracy(ubcf_pred, getData(test, "unknown")))
```

```
##      RMSE       MSE       MAE
## 1.0279037 1.0565860 0.8206121
```

```
ubcf_results <- evaluate(test, method  = "UBCF", n = seq(10,100,10))
```

```
## UBCF run fold/sample [model time/prediction time]
##    1  [0.1sec/0.58sec]
##    2  [0.02sec/0.56sec]
##    3  [0.02sec/0.6sec]
##    4  [0sec/0.59sec]
```

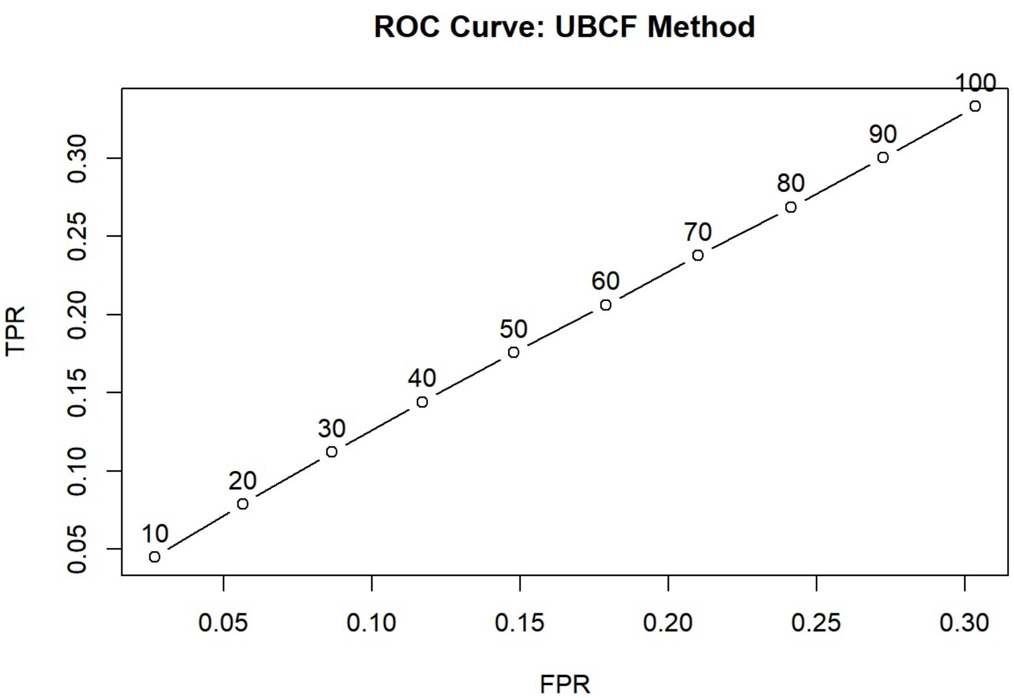
```
class(ubcf_results)
```

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

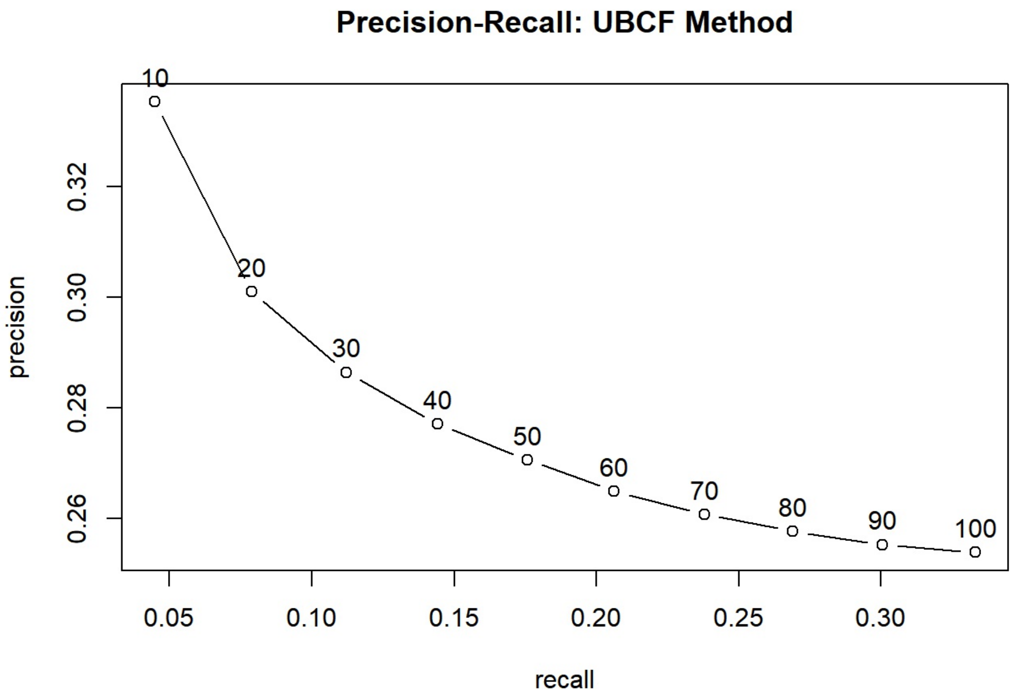
```
head(getConfusionMatrix(ubcf_results)[[1]])
```

```
##            TP        FP       FN       TN precision     recall        TPR
## 10  3.205357  6.794643 63.94643 243.0536 0.3205357 0.04577616 0.04577616
## 20  5.464286 14.535714 61.68750 235.3125 0.2732143 0.07599436 0.07599436
## 30  7.812500 22.187500 59.33929 227.6607 0.2604167 0.11013581 0.11013581
## 40  9.982143 30.017857 57.16964 219.8304 0.2495536 0.14021526 0.14021526
## 50 12.241071 37.758929 54.91071 212.0893 0.2448214 0.17301485 0.17301485
## 60 14.312500 45.687500 52.83929 204.1607 0.2385417 0.20265402 0.20265402
##           FPR
## 10 0.02656125
## 20 0.05721992
## 30 0.08762384
## 40 0.11880667
## 50 0.14957927
## 60 0.18127923
```

```
plot(ubcf_results, annotate = TRUE, main = "ROC Curve: UBCF Method")
```




```
plot(ubcf_results, "prec/rec", annotate = TRUE, main = "Precision-Recall: UBCF Method")
```



POPULAR

```
# POPULAR Models
popular_recMod <- Recommender(getData(test, "train"), "POPULAR")
popular_pred <- predict(popular_recMod, getData(test, "known"), type = "ratings")
cat("POPULAR Method: RMSE, MSE, MAE","\n","\n")
```

```
## POPULAR Method: RMSE, MSE, MAE
##
```

```
(popular <- calcPredictionAccuracy(popular_pred, getData(test, "unknown")))
```

```
##      RMSE       MSE       MAE
## 0.9720629 0.9449062 0.7634149
```

```
popular_results <- evaluate(test, method  = "POPULAR", n = seq(10,100,10))
```

```
## POPULAR run fold/sample [model time/prediction time]
##   1  [0.09sec/0.58sec]
##   2  [0.02sec/0.61sec]
##   3  [0.01sec/0.57sec]
##   4  [0.03sec/0.56sec]
```

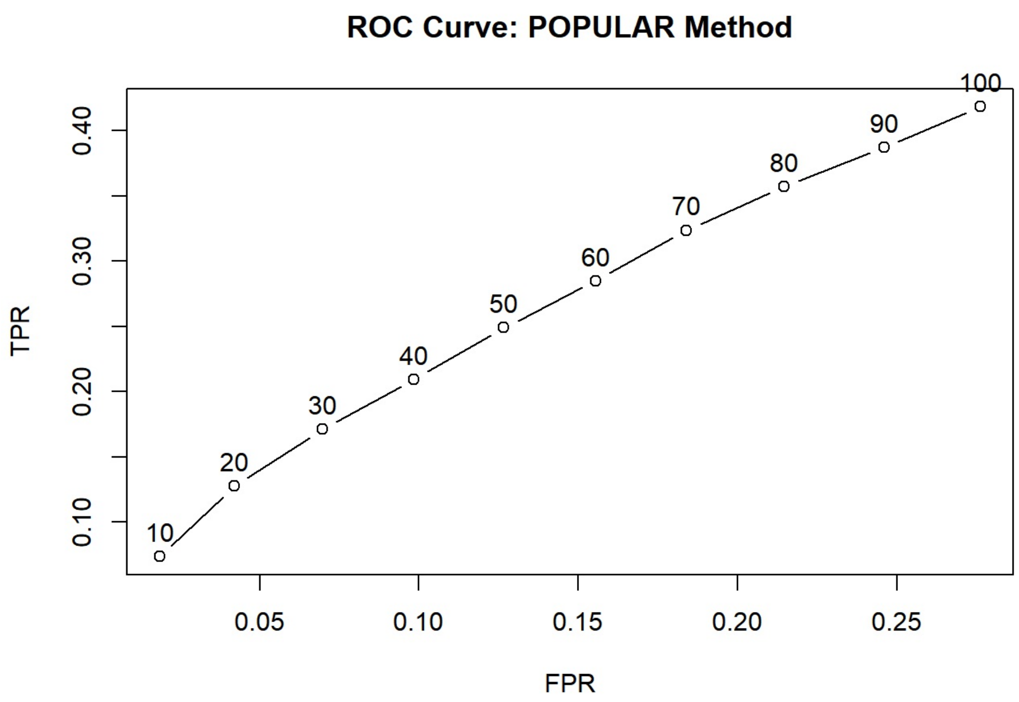
```
class(popular_results)
```

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

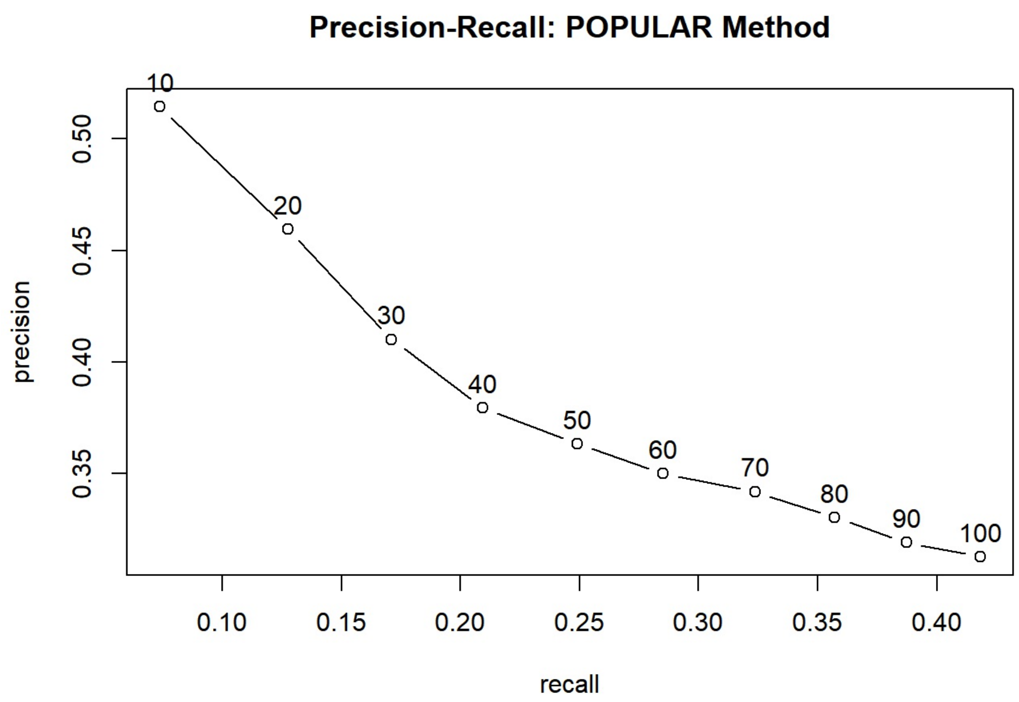
```
head(getConfusionMatrix(popular_results)[[1]])
```

```
##            TP        FP       FN       TN precision     recall        TPR
## 10  4.571429  5.428571 62.58036 244.4196 0.4571429 0.06788259 0.06788259
## 20  8.178571 11.821429 58.97321 238.0268 0.4089286 0.11811623 0.11811623
## 30 11.044643 18.955357 56.10714 230.8929 0.3681548 0.15977698 0.15977698
## 40 13.428571 26.571429 53.72321 223.2768 0.3357143 0.19384917 0.19384917
## 50 15.982143 34.017857 51.16964 215.8304 0.3196429 0.22985776 0.22985776
## 60 18.607143 41.392857 48.54464 208.4554 0.3101190 0.26653364 0.26653364
##           FPR
## 10 0.02059542
## 20 0.04526826
## 30 0.07332163
## 40 0.10339700
## 50 0.13277207
## 60 0.16174385
```

```
plot(popular_results, annotate = TRUE, main = "ROC Curve: POPULAR Method")
```




```
plot(popular_results, "prec/rec", annotate = TRUE, main = "Precision-Recall: POPULAR Method")
```



```
cat("IBCF Model Time/Prediction Time Average","\n","\n")
```

```
## IBCF Model Time/Prediction Time Average
##
```

```
(ibcf_mean <- mean(c(0.15,0.24,0.14,0.14)))
```

```
## [1] 0.1675
```

```
cat("\n","ALS Model Time/Prediction Time Average","\n","\n")
```

```
##
##  ALS Model Time/Prediction Time Average
##
```

```
(als_mean <- mean(c(37.38,36.89,37.8,37.5)))
```

```
## [1] 37.3925
```

```
cat("\n","UBCF Model Time/Prediction Time Average","\n","\n")
```

```
##
##  UBCF Model Time/Prediction Time Average
##
```

```
(ubcf_mean <- mean(c(0.64,0.61,0.62,0.57)))
```

```
## [1] 0.61
```

```
cat("\n","POPULAR Model Time/Prediction Time Average","\n","\n")
```

```
##
##  POPULAR Model Time/Prediction Time Average
##
```

```
(popular_mean <- mean(c(0.71,0.66,0.58,0.58)))
```

```
## [1] 0.6325
```

It is clear that the **IBCF** has the fastest model time/prediction time, while **ALS** is the slowest out of the 4 methods.

```
rbind(ibcf,als,ubcf,popular)
```

```
##              RMSE       MSE       MAE
## ibcf    1.4100032 1.9881090 1.0791286
## als     0.9533999 0.9089713 0.7562991
## ubcf    1.0279037 1.0565860 0.8206121
## popular 0.9720629 0.9449062 0.7634149
```

Next, we will compare the **IBCF,UBCF** techniques

```
compare <- list(IBCF_cos = list(name = "IBCF", param = list(method = "cosine")),IBCF_cor = list(name = "IBCF",
param = list(method = "pearson")), UBCF_cos = list(name = "UBCF", param = list(method = "cosine")), UBCF_cor =
list(name = "UBCF", param = list(method = "pearson")))
```

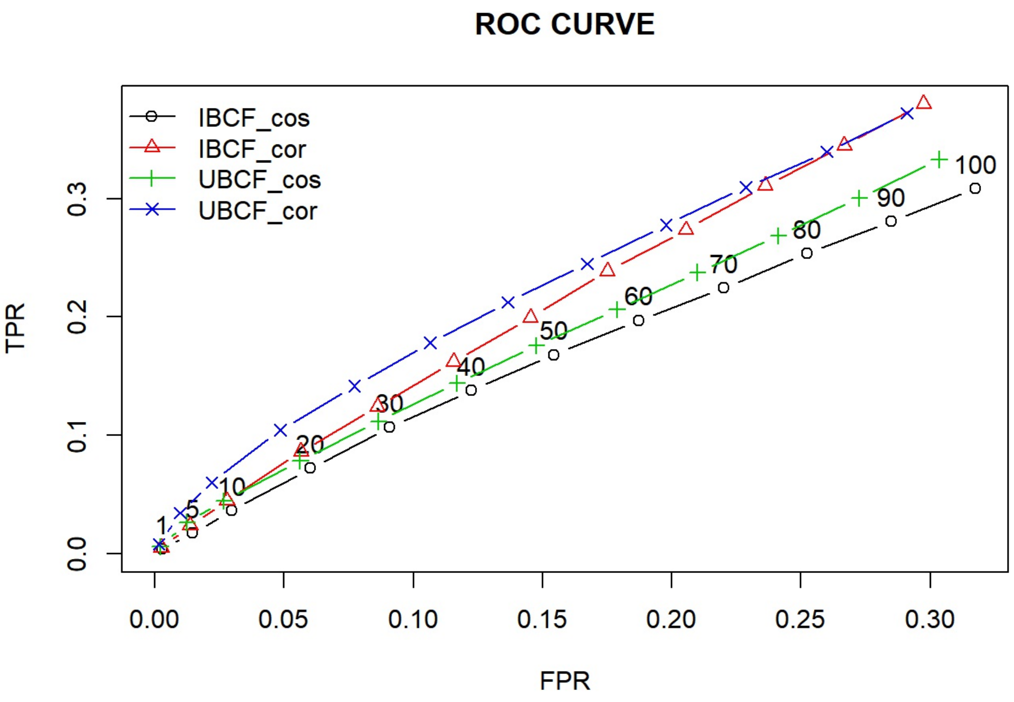
```
compare_results <- evaluate(x = test, method = compare, n = c(1, 5, seq(10,100,10)))
```

```
## IBCF run fold/sample [model time/prediction time]
##   1  [0.99sec/0.15sec]
##   2  [0.98sec/0.16sec]
##   3  [0.98sec/0.14sec]
##   4  [1.13sec/0.15sec]
## IBCF run fold/sample [model time/prediction time]
##   1  [1.14sec/0.16sec]
##   2  [1.17sec/0.14sec]
##   3  [1.18sec/0.16sec]
##   4  [1.1sec/0.16sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.02sec/0.59sec]
##   2  [0sec/0.61sec]
##   3  [0sec/0.59sec]
##   4  [0.01sec/0.6sec]
## UBCF run fold/sample [model time/prediction time]
##   1  [0.03sec/0.72sec]
##   2  [0.01sec/0.73sec]
##   3  [0.02sec/0.72sec]
##   4  [0.02sec/0.72sec]
```

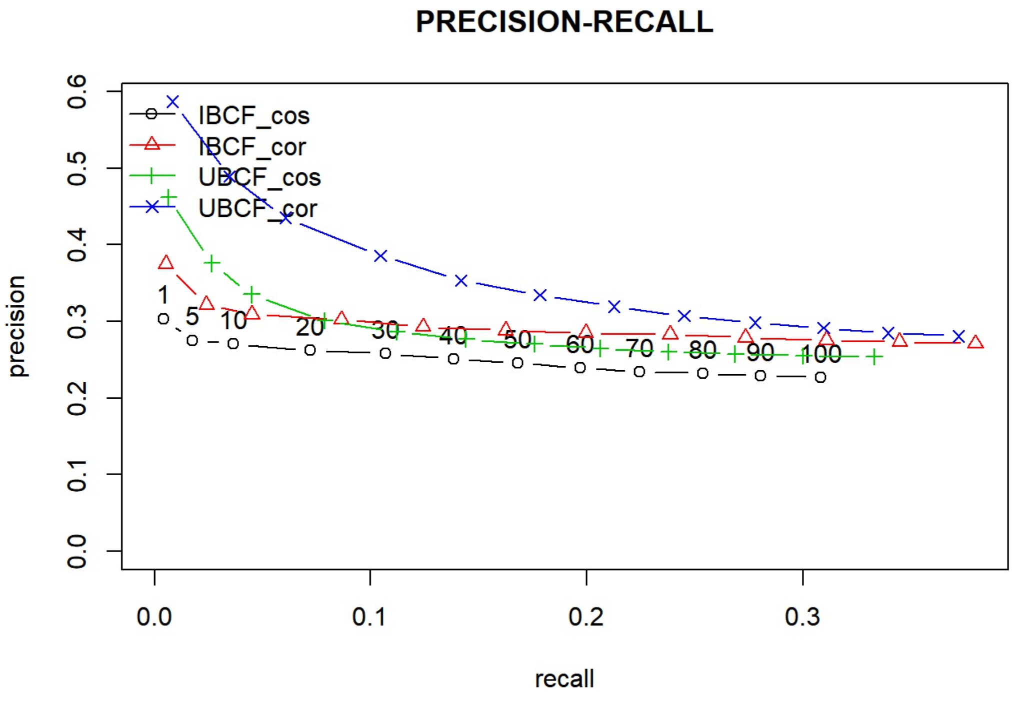
```
class(compare_results)
```

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

```
plot(compare_results, annotate = 1, legend = "topleft")
title("ROC CURVE")
```




```
plot(compare_results, "prec/rec", annotate = 1, legend = "topleft")
title("PRECISION-RECALL")
```


PRECISION-RECALL
IBCF_cos
IBCF_cor
UBCF_cos
UBCF_cor
precision
recall
0.0
0.1
0.2
0.3
0.4
0.5
0.6
1
5
10
20
30
40
50
60
70
80
90
100