Project 2

Shipra Ahuja, Nathan Groom, Saheli Kar

June 24, 2016

Introduction

This project uses book ratings data to implement and configure two different recmmender systems: one using a hybrid collaborative filtering technique taking into account both the genre of the books and their user ratings, the other a content based recommendation system.

Recommender System - Collaborative Filtering

This function is essentially a hybrid collaborative recommender function which does the following: first, it calculates the overall average rating of each title (as rated by users) from 1-5 and places these ratings in a data frame. We then create an empty recommendation matrix with users as rows and book titles as columns. Finally we fill in the matrix with each user's recommendations, which are the top rated books, not yet read by that user, sorted with that user's preferred genre appearing first.

UBCF using manual approach

```
suppressPackageStartupMessages(library(pROC))

#Read the books and Ratings file
books <- read.csv("books.csv",header=TRUE)
ratings <- read.csv("Ratings.csv",header=FALSE)
books_num<-nrow(books)

#Add column names as ISBN of the books
names(ratings)<-c("User", as.character(books$ISBN))

#Get the average rating of each book
books=as.data.frame(cbind(books, avg_score=unname(mapply(mean, ratings[,2:(books_num+1)]))))

readers=ratings[,1]
categories<-unique(books$Category1)

# Matrix to hold the category preference of users
categories_ratings_matrix<-matrix(0,nrow=length(readers), ncol=length(categories),
                        dimnames=list(readers,categories))

Authors<- unique(books$Author)

# Matrix to hold the Athor preference of users
Authors_ratings_matrix<-matrix(0,nrow=length(readers), ncol=length(Authors),
                        dimnames=list(readers,Authors))
```

```
for(rownum in (1:nrow(ratings))) {

  for (colNum in (2:(books_num+1))){
    readerName<-as.character(ratings[rownum,1])
    if(ratings[rownum,colNum]!=0){
      Category1<- books[books$ISBN==(names(ratings)[colNum]),]$Category1
      Category2<- books[books$ISBN==(names(ratings)[colNum]),]$Category2
      Author<-    books[books$ISBN==(names(ratings)[colNum]),]$Author

      categories_ratings_matrix[readerName, Category1] = categories_ratings_matrix[readerName, Category
      categories_ratings_matrix[readerName, Category2] = categories_ratings_matrix[readerName, Category
      Authors_ratings_matrix[readerName, Author] = Authors_ratings_matrix[readerName, Author]+1

    }
  }
}

categories_ratings_matrix=categories_ratings_matrix/rowSums(categories_ratings_matrix)

#Fetch the preferred books for for an user for a particular category
getPreferredBooks<-function(User, Category, booksToFetch){
  #fetch all the books from the category
  booksInCategory<-books[books$Category1==Category | books$Category2==Category,c(1:3,6)]

  #Sort the books with average score
  booksInCategory<-booksInCategory[ order(booksInCategory[, "avg_score"], decreasing = TRUE), ]

  #Initialize vector for the books to recommend
  booksToRecommend<-c()
  count=as.numeric(booksToFetch)

  for(i in 1:length(booksInCategory)){
    bookISBN = booksInCategory[i, "ISBN"]

    #Add the book into the recommendation list if the user haven't read it
    if(ratings[ratings$User==User,bookISBN]==0){
      recommendBook<- paste0("Title::", booksInCategory[i,"Title"], "; Author::", booksInCategory[i,"Au
                             "; ISBN::", booksInCategory[i,"ISBN"])
      booksToRecommend<-c(booksToRecommend, recommendBook)

      #Decrease the count by 1 as one book is recommended
      count=(count-1)
    }

    #If no more boo to fetch then return the list
    if(count==0){
      return(booksToRecommend)
    }
  }
  return(booksToRecommend)
}

#Initialize the empty recommendation matrix
```

```
reco_matrix<- matrix(NA, nrow = length(readers), ncol = 5 )
rownames(reco_matrix)<-as.character(readers)

#Compute recommendation for all the users

for(i in 1:length(readers)){
  #Get users category preference
  temp_df <- cbind(Category=colnames(categories_ratings_matrix), Score=categories_ratings_matrix[i,])
  temp_df<-temp_df[ order(temp_df[, "Score"], decreasing = TRUE), ]

  #Initialize number of books to be recommended
  booksToFetch=5
  count=booksToFetch

  #Start with the  ost preferred category
  Category_index=1
  selected.category=temp_df[Category_index,1]
  booksToRecommend<-c()
  while(booksToFetch>0){

     booksToRecommend<-c(booksToRecommend, getPreferredBooks(as.character(readers[i]), as.character(sel
     booksToFetch=booksToFetch-length(booksToRecommend)

     #if the preferred category doesnt have enough book to offer to the reader go for the next category
     Category_index=Category_index+1

     # If the number of recommended books fetched then stop
     if(Category_index>length(categories)){
       break
     }
    selected.category=temp_df[Category_index,1]

  }

  booksToRecommend=c(booksToRecommend,rep("",(count-length(booksToRecommend))))
  #Add the recommended book in the matrix
  reco_matrix[i,]=rbind(booksToRecommend)

}

# Function to recommend books
getRecommendation<-function(User=NA){
  if(is.na(User)){
    return(reco_matrix)
  }
  else{
    return(reco_matrix[User,])
  }
}
```

Recommendation Samples

This displays some samples of 5 recommended books per user.

```r
head(getRecommendation())
```

```
##          [,1]
## Ben      "Title::Holes; Author::Louis Sachar; ISBN::978-0440414803"
## Moose    "Title::Holes; Author::Louis Sachar; ISBN::978-0440414803"
## Reuven   "Title::Holes; Author::Louis Sachar; ISBN::978-0440414803"
## Cust1    "Title::Holes; Author::Louis Sachar; ISBN::978-0440414803"
## Cust2    "Title::Holes; Author::Louis Sachar; ISBN::978-0440414803"
## Francois "Title::Holes; Author::Louis Sachar; ISBN::978-0440414803"
##          [,2]
## Ben      "Title::Bleach (graphic novel); Author::Tite Kubo; ISBN::978-1421539928"
## Moose    "Title::Hatchet; Author::Gary Paulsen; ISBN::978-1416936473"
## Reuven   "Title::Naruto; Author::Masashi Kishimoto; ISBN::978-1421584935"
## Cust1    "Title::To Kill a Mockingbird; Author::Harper Lee; ISBN::978-0446310789"
## Cust2    "Title::To Kill a Mockingbird; Author::Harper Lee; ISBN::978-0446310789"
## Francois "Title::To Kill a Mockingbird; Author::Harper Lee; ISBN::978-0446310789"
##          [,3]
## Ben      "Title::Bone Series; Author::Jeff Smith; ISBN::978-0439706407"
## Moose    "Title::Naruto; Author::Masashi Kishimoto; ISBN::978-1421584935"
## Reuven   "Title::Bleach (graphic novel); Author::Tite Kubo; ISBN::978-1421539928"
## Cust1    "Title::The Bourne Series; Author::Robert Ludlum; ISBN::978-1780485799"
## Cust2    "Title::The Da Vinci Code; Author::Dan Brown; ISBN::978-0307474278"
## Francois "Title::The Bourne Series; Author::Robert Ludlum; ISBN::978-1780485799"
##          [,4]
## Ben      "Title::Maus: A Survivor's Tale; Author::Art Spiegelman; ISBN::978-0394747231"
## Moose    "Title::Bleach (graphic novel); Author::Tite Kubo; ISBN::978-1421539928"
## Reuven   "Title::Maus: A Survivor's Tale; Author::Art Spiegelman; ISBN::978-0394747231"
## Cust1    "Title::The Hitchhiker's Guide To The Galaxy; Author::Douglas Adams; ISBN::978-0345391803"
## Cust2    "Title::The Hitchhiker's Guide To The Galaxy; Author::Douglas Adams; ISBN::978-0345391803"
## Francois "Title::The Hitchhiker's Guide To The Galaxy; Author::Douglas Adams; ISBN::978-0345391803"
##          [,5]
## Ben      ""
## Moose    ""
## Reuven   ""
## Cust1    "Title::Hatchet; Author::Gary Paulsen; ISBN::978-1416936473"
## Cust2    "Title::The Da Vinci Code; Author::Dan Brown; ISBN::978-0307474278"
## Francois "Title::Hatchet; Author::Gary Paulsen; ISBN::978-1416936473"
```

```r
# Compute average score
collaborative_reco_matrix<-matrix(NA, nrow=86, ncol=55)

for(i in 1:55){
  collaborative_reco_matrix[,i]<-books[i,]$avg_score
}

head(collaborative_reco_matrix)
```

```
##          [,1]      [,2]     [,3]       [,4]        [,5]        [,6]
## [1,] 1.604651 0.3953488 0.255814 0.06976744 -0.02325581 -0.01162791
## [2,] 1.604651 0.3953488 0.255814 0.06976744 -0.02325581 -0.01162791
## [3,] 1.604651 0.3953488 0.255814 0.06976744 -0.02325581 -0.01162791
## [4,] 1.604651 0.3953488 0.255814 0.06976744 -0.02325581 -0.01162791
## [5,] 1.604651 0.3953488 0.255814 0.06976744 -0.02325581 -0.01162791
```

```
## [6,] 1.604651 0.3953488 0.255814 0.06976744 -0.02325581 -0.01162791
##           [,7]      [,8]      [,9]    [,10]     [,11]    [,12]     [,13]
## [1,] 0.4186047 0.1046512 0.1162791 1.046512 0.2209302 1.197674 0.3139535
## [2,] 0.4186047 0.1046512 0.1162791 1.046512 0.2209302 1.197674 0.3139535
## [3,] 0.4186047 0.1046512 0.1162791 1.046512 0.2209302 1.197674 0.3139535
## [4,] 0.4186047 0.1046512 0.1162791 1.046512 0.2209302 1.197674 0.3139535
## [5,] 0.4186047 0.1046512 0.1162791 1.046512 0.2209302 1.197674 0.3139535
## [6,] 0.4186047 0.1046512 0.1162791 1.046512 0.2209302 1.197674 0.3139535
##          [,14]     [,15]      [,16]     [,17]     [,18]     [,19]
## [1,] 0.1162791 0.4418605 0.08139535 0.8372093 0.3604651 0.6162791
## [2,] 0.1162791 0.4418605 0.08139535 0.8372093 0.3604651 0.6162791
## [3,] 0.1162791 0.4418605 0.08139535 0.8372093 0.3604651 0.6162791
## [4,] 0.1162791 0.4418605 0.08139535 0.8372093 0.3604651 0.6162791
## [5,] 0.1162791 0.4418605 0.08139535 0.8372093 0.3604651 0.6162791
## [6,] 0.1162791 0.4418605 0.08139535 0.8372093 0.3604651 0.6162791
##          [,20]      [,21]      [,22]      [,23]     [,24]     [,25]
## [1,] 0.6511628 0.02325581 0.03488372 0.01162791 0.5465116 0.1744186
## [2,] 0.6511628 0.02325581 0.03488372 0.01162791 0.5465116 0.1744186
## [3,] 0.6511628 0.02325581 0.03488372 0.01162791 0.5465116 0.1744186
## [4,] 0.6511628 0.02325581 0.03488372 0.01162791 0.5465116 0.1744186
## [5,] 0.6511628 0.02325581 0.03488372 0.01162791 0.5465116 0.1744186
## [6,] 0.6511628 0.02325581 0.03488372 0.01162791 0.5465116 0.1744186
##          [,26]      [,27]     [,28]     [,29]      [,30]    [,31]    [,32]
## [1,] 0.4883721 0.02325581 0.4069767 0.4069767 0.01162791 1.081395 1.883721
## [2,] 0.4883721 0.02325581 0.4069767 0.4069767 0.01162791 1.081395 1.883721
## [3,] 0.4883721 0.02325581 0.4069767 0.4069767 0.01162791 1.081395 1.883721
## [4,] 0.4883721 0.02325581 0.4069767 0.4069767 0.01162791 1.081395 1.883721
## [5,] 0.4883721 0.02325581 0.4069767 0.4069767 0.01162791 1.081395 1.883721
## [6,] 0.4883721 0.02325581 0.4069767 0.4069767 0.01162791 1.081395 1.883721
##         [,33]     [,34] [,35]     [,36]     [,37]     [,38]    [,39] [,40]
## [1,] 0.627907 0.1395349     0 0.1860465 0.1860465 0.7906977 1.395349     1
## [2,] 0.627907 0.1395349     0 0.1860465 0.1860465 0.7906977 1.395349     1
## [3,] 0.627907 0.1395349     0 0.1860465 0.1860465 0.7906977 1.395349     1
## [4,] 0.627907 0.1395349     0 0.1860465 0.1860465 0.7906977 1.395349     1
## [5,] 0.627907 0.1395349     0 0.1860465 0.1860465 0.7906977 1.395349     1
## [6,] 0.627907 0.1395349     0 0.1860465 0.1860465 0.7906977 1.395349     1
##          [,41]    [,42]    [,43]   [,44]     [,45] [,46]     [,47]
## [1,] 0.4418605 1.093023 2.639535 1.77907 0.3139535     0 0.4069767
## [2,] 0.4418605 1.093023 2.639535 1.77907 0.3139535     0 0.4069767
## [3,] 0.4418605 1.093023 2.639535 1.77907 0.3139535     0 0.4069767
## [4,] 0.4418605 1.093023 2.639535 1.77907 0.3139535     0 0.4069767
## [5,] 0.4418605 1.093023 2.639535 1.77907 0.3139535     0 0.4069767
## [6,] 0.4418605 1.093023 2.639535 1.77907 0.3139535     0 0.4069767
##          [,48]     [,49]    [,50]    [,51]     [,52]    [,53]     [,54]
## [1,] 0.2093023 0.2674419 1.627907 1.860465 0.1627907 0.755814 0.1511628
## [2,] 0.2093023 0.2674419 1.627907 1.860465 0.1627907 0.755814 0.1511628
## [3,] 0.2093023 0.2674419 1.627907 1.860465 0.1627907 0.755814 0.1511628
## [4,] 0.2093023 0.2674419 1.627907 1.860465 0.1627907 0.755814 0.1511628
## [5,] 0.2093023 0.2674419 1.627907 1.860465 0.1627907 0.755814 0.1511628
## [6,] 0.2093023 0.2674419 1.627907 1.860465 0.1627907 0.755814 0.1511628
##          [,55]
## [1,] 0.4534884
## [2,] 0.4534884
## [3,] 0.4534884
```

```
## [4,] 0.4534884
## [5,] 0.4534884
## [6,] 0.4534884
```

Evaluation metrics for collaborative filtering using manual approach

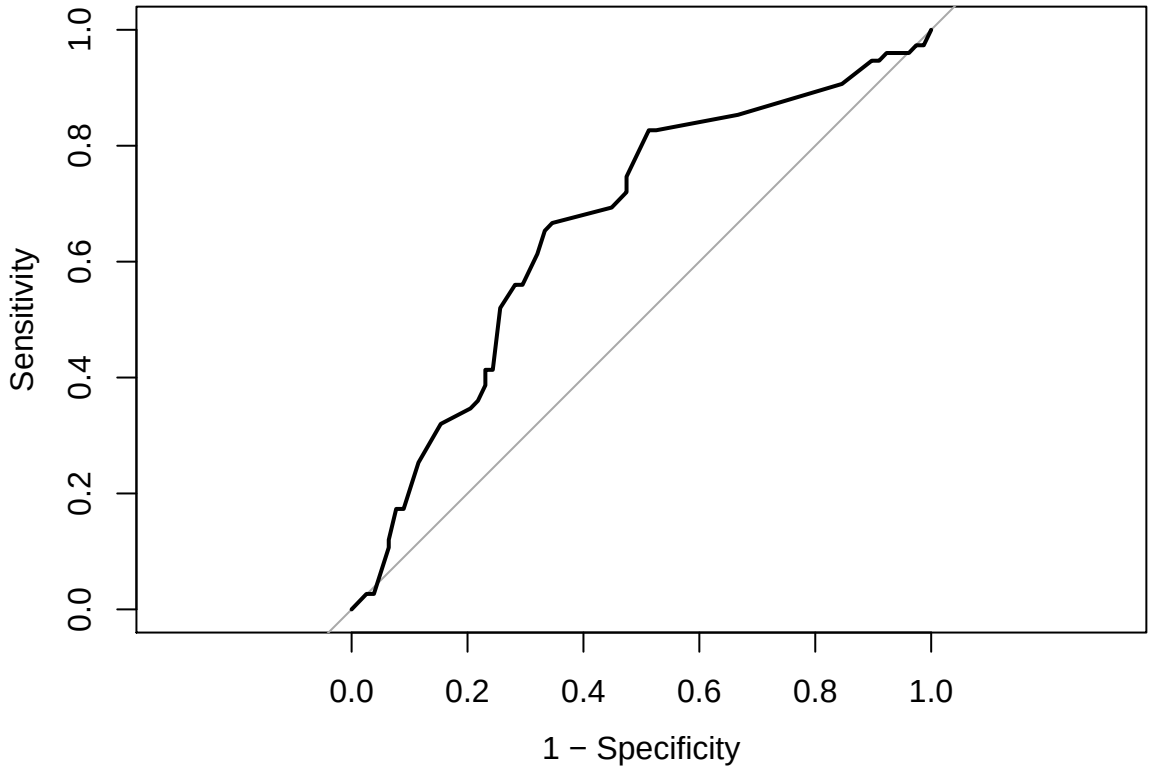
```
suppressPackageStartupMessages(library(recommenderlab))

r <- as((collaborative_reco_matrix), "realRatingMatrix")

# Create 90/10 split into training/test datasets
eval <- evaluationScheme(r[1:85,], method="split", train=0.9,
                         k=1, given=9)

# Create a UBCF recommender system using training data
r <- Recommender(getData(eval, "train"), "UBCF")

# Create predictions for test data using known ratings
pred <- predict(r, getData(eval, "known"), type="ratings")

# Compute the average metrics for all readers - RMSE, MSE, MAE
calcPredictionAccuracy(pred, getData(eval, "unknown"),given=85,goodRating=5, byuser=FALSE)
```

```
##       RMSE        MSE        MAE
## 0.14528616 0.02110807 0.11817921
```

ROC plot for Collaborative filtering using manual approach

The ROC plot is created for manual UBCF recommender system

```
ratings.binary<- unlist(ratings[,2:56])
reco.binary<-as.numeric(collaborative_reco_matrix)
rocCurve<-roc(response=ratings.binary,predictor=reco.binary,threshhold=2)
plot(rocCurve, legacy.axes = TRUE)
```




```
##
## Call:
## roc.default(response = ratings.binary, predictor = reco.binary,     threshhold = 2)
##
## Data: reco.binary in 78 controls (ratings.binary -5) < 75 cases (ratings.binary -3).
## Area under the curve: 0.6648
```

Recommendation using recommenderlab package for UBCF filtering

```
# Get all book names from ratings matrix
ratings_matrix<-as.matrix(ratings[,2:56])

# Get all user names from ratings dataset and place it as individual rows of column 1
rownames(ratings_matrix)<-c(as.character(ratings[,1]))

# Get all books names from books dataset and assign them as column names
colnames(ratings_matrix)<-as.character(books$Title)

# Convert the matrix as realRatingMatrix to compress it
r <- as(ratings, "realRatingMatrix")

# Get unique values of the ratings
vector_ratings <- as.vector(r@data)
unique(vector_ratings)
```

```
## [1]  0 -5 -3  5  3  1
```

```
# Group the ratings
table_ratings <- table(vector_ratings)

# Create recommender system model

reco.model <- Recommender(r[1:nrow(r)],method="UBCF",param=list(method="Cosine",k=30))
```

```
## Available parameter (with default values):
## method    =  cosine
## nn    =  25
## sample    =  FALSE
## normalize     =  center
## minRating     =  NA
## verbose   =  FALSE
```

```
# Recommend books for all users
books.pred <- predict(reco.model,r[1:nrow(r)],n=5)

rec_matrix <- sapply(books.pred@items,function(x){

  colnames(ratings_matrix)[x]
})

head(rec_matrix)
```

```
## [[1]]
## [1] "The Hitchhiker's Guide To The Galaxy"
## [2] "The Five People You Meet in Heaven"
## [3] "Speak"
## [4] "I Know Why the Caged Bird Sings"
##
## [[2]]
## [1] "The Hitchhiker's Guide To The Galaxy"
## [2] "The Five People You Meet in Heaven"
## [3] "Speak"
## [4] "I Know Why the Caged Bird Sings"
##
## [[3]]
## character(0)
##
## [[4]]
## [1] "The Hitchhiker's Guide To The Galaxy"
## [2] "The Five People You Meet in Heaven"
## [3] "Speak"
## [4] "I Know Why the Caged Bird Sings"
##
## [[5]]
## [1] "The Hitchhiker's Guide To The Galaxy"
## [2] "The Five People You Meet in Heaven"
## [3] "Speak"
## [4] "I Know Why the Caged Bird Sings"
```

```
##
## [[6]]
## character(0)
```

Evaluation Metrics of Collaborative Filtering

Metrics for User Based Collaborative Filtering System

The Root Mean Square Error, Mean Absolute Error and and Mean Square Error have been computed for the UBCF recommendaton system.

Compute RMSE, MAE, MSE

```
# Create real rating matrix

r <- as(ratings_matrix, "realRatingMatrix")

# Create 90/10 split into training/test datasets
eval <- evaluationScheme(r[1:85,], method="split", train=0.9,
                         k=1, given=9)

# Create a UBCF recommender system using training data
r <- Recommender(getData(eval, "train"), "UBCF")

# Create predictions for test data using known ratings
pred <- predict(r, getData(eval, "known"), type="ratings")

# Compute the average metrics for all readers - RMSE, MSE, MAE
calcPredictionAccuracy(pred, getData(eval, "unknown"),given=85,goodRating=5, byuser=FALSE)
```

```
##     RMSE      MSE      MAE
## 1.777594 3.159840 1.168171
```

Compute confusion matrix

```
r <- as(ratings_matrix, "realRatingMatrix")

eval <- evaluationScheme(r[1:85,], method="split", train=0.9,
                           k=1, given=9, goodRating=5)

results <- evaluate(eval,method="UBCF",n=seq(10,80,10))
```

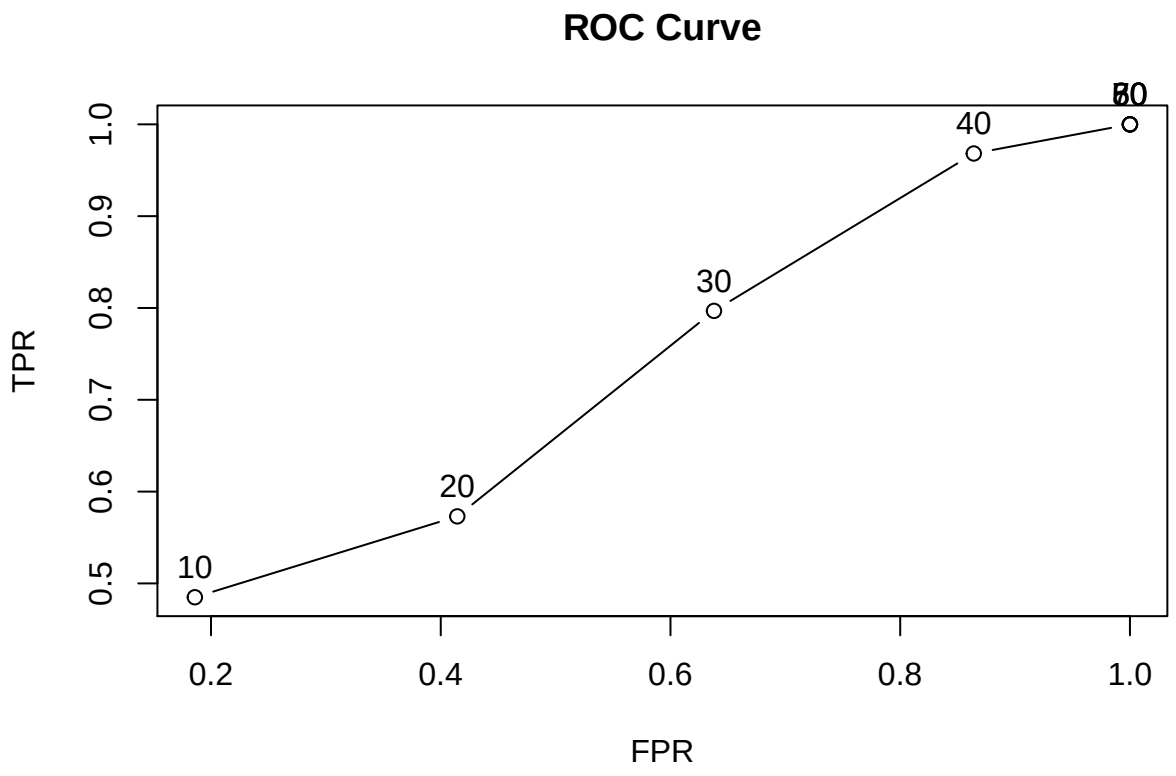
```
## UBCF run fold/sample [model time/prediction time]
##   1  [0.01sec/0.07sec]
```

```
getConfusionMatrix(results)
```

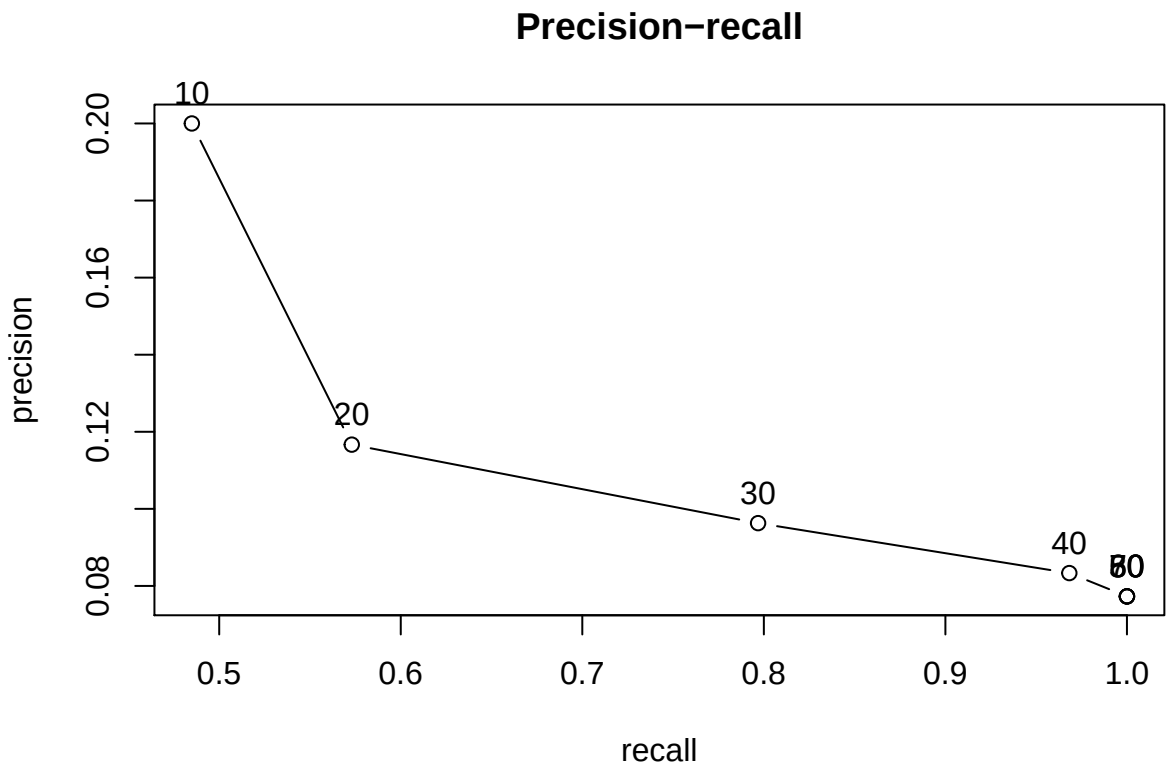
```
## [[1]]
##          TP       FP        FN        TN  precision    recall       TPR
## 10 2.000000  8.00000 1.5555556 34.444444 0.20000000 0.4849206 0.4849206
## 20 2.333333 17.66667 1.2222222 24.777778 0.11666667 0.5730159 0.5730159
## 30 2.888889 27.11111 0.6666667 15.333333 0.09629630 0.7968254 0.7968254
## 40 3.333333 36.66667 0.2222222  5.777778 0.08333333 0.9682540 0.9682540
## 50 3.555556 42.44444 0.0000000  0.000000 0.07729469 1.0000000 1.0000000
## 60 3.555556 42.44444 0.0000000  0.000000 0.07729469 1.0000000 1.0000000
## 70 3.555556 42.44444 0.0000000  0.000000 0.07729469 1.0000000 1.0000000
## 80 3.555556 42.44444 0.0000000  0.000000 0.07729469 1.0000000 1.0000000
##          FPR
## 10 0.1859387
## 20 0.4144098
## 30 0.6379009
## 40 0.8640434
## 50 1.0000000
## 60 1.0000000
## 70 1.0000000
## 80 1.0000000
```

Plot ROC and Precision-Recall Curves

```
# Plot ROC Curve
plot(results,annotate=TRUE,main="ROC Curve")
```



```
# Plot Precision-Recall Curve
plot(results, "prec/rec", annotate = TRUE, main = "Precision-recall")
```



Content Based Filtering

This recommendation system is content based and provides recommendations by normalizing each user's ratings. The algorithm is recommending items for each user that are similar to its past purchases.

```
Category_books_count<- matrix(0, nrow=length(categories), ncol=1)
rownames(Category_books_count)<- categories

for(i in 1:nrow(books)){
  Category1 = books[i,]$Category1
  Category2 = books[i,]$Category2
  Category_books_count[Category1,1] = Category_books_count[Category1,1]+1
  Category_books_count[Category2,1] = Category_books_count[Category2,1]+1
}


books_profile<- matrix(NA, nrow = length(books$ISBN), ncol = length(categories))
rownames(books_profile)<- as.character(books$ISBN)
colnames(books_profile)<- as.character(categories)
```

```
idf<- log(nrow(books)/Category_books_count)

for(i in 1:nrow(books_profile)){
  for(j in 1: ncol(books_profile)){
    category1<-as.character(books[i,"Category1"])
    category2<-as.character(books[i,"Category2"])
    idf1=0
    idf2=0
    if(as.character(categories[j])==category1){
      idf1<-idf[category1,]
    }
    if(as.character(categories[j])==category2){
      idf1<-idf[category2,]
    }
    books_profile[i,j]<- books[i,"avg_score"]*(idf1+idf2)

  }
}

avg_rating_by_user = mean(unlist(ratings[,2:56]))

user_profile<-matrix(0, nrow = nrow(ratings), ncol = nrow(books))
rownames(user_profile)<- ratings[,1]
colnames(user_profile)<- books$ISBN

user_profile<- t(as.matrix(ratings[, 2:56]- avg_rating_by_user))
books_profile = rowSums(books_profile)

rec_profile<- user_profile+books_profile
rec_profile[rec_profile>5]=5
```

Prediction of ratings by Ben Using Content Base Filtering

```
rec_profile[1,]
```

```
##  [1]  5.0000000  5.0000000  5.0000000  5.0000000  5.0000000  5.0000000
##  [7]  5.0000000  4.4974062  4.4974062  4.4974062  4.4974062  5.0000000
## [13]  4.4974062  5.0000000  5.0000000  5.0000000  4.4974062  4.4974062
## [19]  5.0000000  5.0000000  4.4974062  5.0000000  5.0000000  4.4974062
## [25]  4.4974062  5.0000000  4.4974062  4.4974062  4.4974062  4.4974062
## [31]  5.0000000  4.4974062  5.0000000  4.4974062  5.0000000  5.0000000
## [37]  5.0000000  5.0000000  5.0000000  4.4974062  4.4974062  4.4974062
## [43]  5.0000000  4.4974062  4.4974062  4.4974062  4.4974062  4.4974062
## [49]  4.4974062  4.4974062  5.0000000  4.4974062  4.4974062  4.4974062
## [55]  4.4974062  5.0000000  4.4974062  4.4974062  4.4974062  4.4974062
## [61]  5.0000000  4.4974062  4.4974062  5.0000000  5.0000000  4.4974062
## [67]  4.4974062  4.4974062  4.4974062  4.4974062  4.4974062  5.0000000
## [73]  4.4974062  4.4974062  5.0000000  5.0000000  4.4974062 -0.5025938
## [79]  5.0000000  4.4974062  4.4974062  4.4974062  4.4974062  5.0000000
## [85]  4.4974062  5.0000000
```

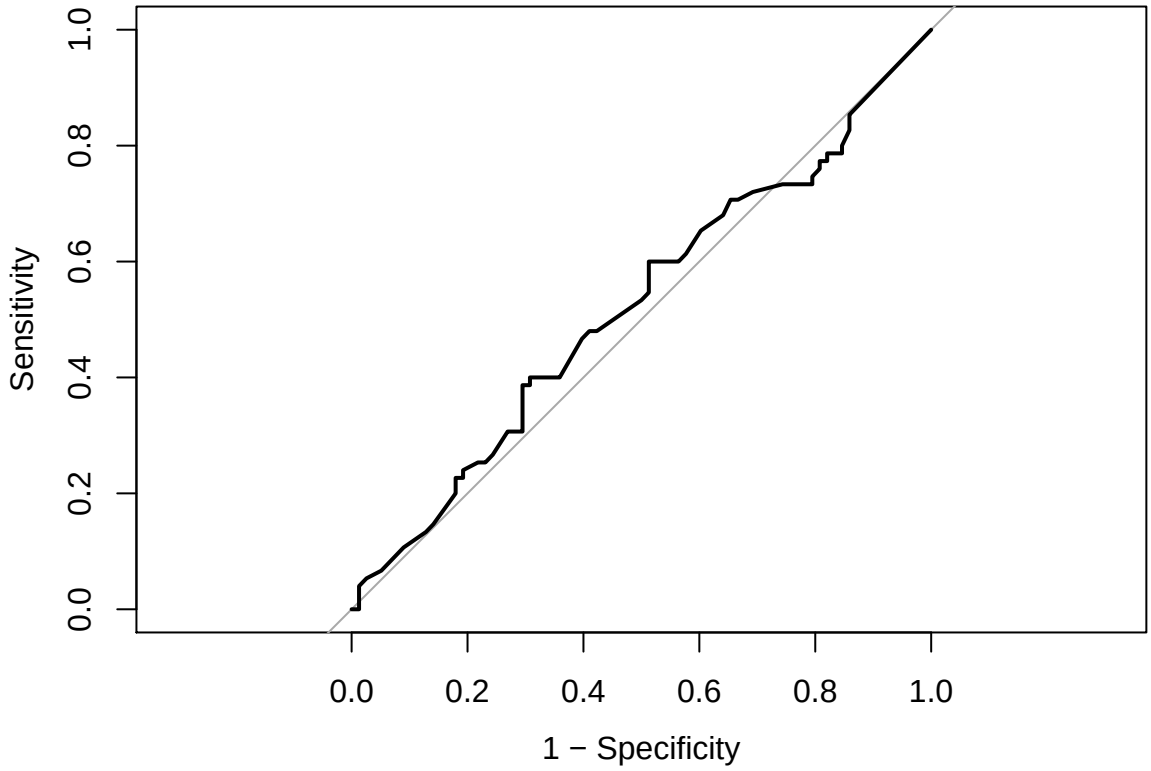
Evaluation metrics for content base filtering

```
RMSE = sqrt(mean((ratings[, 2:56] - t(rec_profile))^2))
RMSE
```

```
## [1] 1.650569
```

ROC plot for content based filtering

```
ratings.binary<- unlist(ratings[,2:56])
reco.binary<-as.numeric(rec_profile)
rocCurve<-roc(response=ratings.binary,predictor=reco.binary,threshhold=2)
plot(rocCurve, legacy.axes = TRUE)
```




```
##
## Call:
## roc.default(response = ratings.binary, predictor = reco.binary,     threshhold = 2)
##
## Data: reco.binary in 78 controls (ratings.binary -5) > 75 cases (ratings.binary -3).
## Area under the curve: 0.5222
```