Appendix

Analysis and testing of different Recommendation Systems using Machine Learning Techniques

Author: Adam Akram

Student Number: 1842752

Supervisor: Dr Senir Dinar

Date: 22/04/2022

Contents

1 Appendix A **2**

1.1 Instructions . . . 2

1.2 Walk-through: . . . 2

2 Appendix B **4**

2.1 Declaration . . . 1

2.2 Packages Used . . . 2

2.3 Data Set . . . 2

2.4 Code Inspiration . . . 3

2.5 Source Code . . . 5

3 Appendix C **42**

3.1 Graphs and Charts . . . 42

Chapter 1

Appendix A

1.1 Instructions

1.2 Walk-through:

1. Install ”R Studio” or the ”R” extension for ”Visual Studio Code”
2. Check that all the specified packages have been installed:
   (a) mlr
   (b) recommenderlab
   (c) tidyverse
   (d) tidyr
   (e) dplyr
   (f) readr
   (g) knitr
   (h) AUC
   (i) doParallel
3. Make sure the relevant code and all data files are present in the folder:
   (a) Artists.dat
   (b) user_artists.dat
   (c) user_taggedartists.dat

(d) Recommendation.r

If any of the files a,b or c are missing, please download and extract the files from the data set linked below

4. Open the Recommendation.r file
5. On line 1, set the working directory to the location of the project folder on the machine:

```
1  setwd('C:/Users/Student/Desktop/PROJECT')
```

6. Click on the code and press Cntrl+A to select all lines
7. Press Cntrl+Enter to run the code
8. As the code is run, the output of the recommendation systems is printed, with a few graphs and histograms
9. This process takes long depending on the machine's RAM

Chapter 2

Appendix B

Source Code

Adam Akram

22/04/2022

2.1 Declaration

Declaration I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary. Adam Akram — 22/04/2022

2.2 Packages Used

As mentioned in the code comments, some packages were used to create the project.

- 'recommenderlab' found at: `https://github.com/mhahsler/recommenderlab`, (mhahsler, Sept 2021) (Accessed 11/03/2022)
- 'knitr' found at: `https://github.com/yihui/knitr` (Yihui, April 2022) (Accessed 16/03/2022)
- 'mlr' found at: `https://github.com/mlr-org/mlr/` (mlr-org, April 2022) (Accessed 17/03/2022)
- 'tidyverse' found at: `https://github.com/tidyverse` (tidyverse, April 2022) (Accessed 12/03/2022)
- 'tidyr' found at: `https://github.com/tidyverse/tidyr` (tidyverse, April 2022) (Accessed 12/03/2022)
- 'dplyr' found at: `https://github.com/tidyverse/dplyr` (tidyverse, April 2022) (Accessed 12/03/2022)
- 'readr' found at: `https://github.com/tidyverse/readr` (tidyverse, April 2022) (Accessed 12/03/2022)
- 'doParallel' found at: `https://github.com/HenrikBengtsson/parallelly` (Bengtsson, April 2022) (Accessed 15/03/2022)
- 'AUC' found at: `https://rdrr.io/cran/DescTools/src/R/StatsAndCIs.r` (Signorell, April 2022) (Accessed 20/03/2022)

2.3 Data Set

As mentioned in the report, the data set was taken from kaggle:

Data Set Link:

`https://www.kaggle.com/code/kerneler/starter-last-fm-music-artist-scrobbles-45e52454-b/data?select=lastfm_user_scrobbles.csv` (KAGGLE KERNELER, June 2020) (Accessed 22/2/2022)

2.4 Code Inspiration

Inspiration used to help with the development of the recommendation systems was taken from the following sources:

2.4.1 Recommendation Systems

- `https://github.com/Ravi8889/Recomendations-systems/blob/main/Recomendation%20Systems.ipynb` (Ravi8889, June 2021) (Accessed 2/1/2022)

2.4.2 Evaluation Metrics

- `https://github.com/bhattbhavesh91/classification-metrics-python/blob/master/ml_a.ipynb` (bhattbhavesh91, Dec 2018) (Accessed 5/1/2022)

Listings

2.1 Pearson Correlation Functions . . . 5
2.2 Cosine Similarity Functions . . . 8
2.3 Pearson Correlation for Item Based Collaborative Filtering . . . 14
2.4 Cosine Similarity for Item Based Collaborative Filtering . . . 16
2.5 Cluster Based Collaborative Filtering . . . 17
2.6 Content Based Filtering . . . 19
2.7 Predictive Accuracy . . . 21
2.8 Classification Accuracy . . . 22
2.9 Ranking Accuracy . . . 23
2.10 Data Manipulation For Collaborative Based Filtering . . . 25
2.11 Performing Evaluation Metrics . . . 29
2.12 Data Manipulation For Content Based Filtering . . . 31
2.13 Quantitative Evaluations between systems . . . 35
2.14 Qualitative Results . . . 36
2.15 Hybrid Recommendation System Results . . . 38

2.5 Source Code

2.5.1 Pearson Correlation Functions

```r
setwd('C:/Users/Student/Desktop/PROJECT')
####Functions for user based Collaborative Filtering using pearson correlation ####

meandiff <- function(data,i){
  data[i,] - mean(data[i,],na.rm=TRUE)
}

 #User Based Collaborative Filtering for multiple users
 UserBasedCF_pearson <- function(train_data, test_data, N, NN, onlyNew=TRUE){

   ### similarity ###
   similarity_matrix <- matrix(, nrow = nrow(test_data), ncol = nrow(train_data),
                               dimnames = list(rownames(test_data), rownames(train_data)))

   for (i in rownames(test_data)){
     for (j in rownames(train_data)){
       sim = sum(meandiff(test_data,i) * meandiff(train_data,j), na.rm = TRUE) /
         (sqrt(sum(meandiff(test_data,i)^2,na.rm=TRUE)) *
            sqrt(sum(meandiff(train_data,j)^2,na.rm=TRUE)))
       similarity_matrix[i,j] <- sim
     }
   }
   print("similarity calculation done")
   ### Nearest Neighbors ###
   similarity_matrix_NN <- similarity_matrix

   for (k in 1:nrow(similarity_matrix_NN)){
     crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
     similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val, similarity_matrix_NN[k,], NA)
   }

   print("Nearest Neighbor selection done")
   ### Prediction ###
   # Prepare
```

```r
  prediction <- matrix(, nrow=nrow(test_data), ncol(test_data),
                       dimnames=list(rownames(test_data), colnames(test_data)))
  prediction2 <- matrix(, nrow=nrow(test_data), ncol(test_data),
                        dimnames=list(rownames(test_data), colnames(test_data))
   )

  TopN <- matrix(, nrow=nrow(test_data), ncol=N, dimnames=list(rownames(
   test_data)))
  ### Numerator ###
  for (u in rownames(test_data)){
    similarity_vector <- na.omit(similarity_matrix_NN[u, ])

    NN_norm <- train_data[rownames(train_data) %in% names(similarity_vector),]

    CM <- colMeans(train_data, na.rm=TRUE)
    for (l in 1:ncol(NN_norm)){
      NN_norm[,l] <- NN_norm[,l] - CM[l]
    }
    NN_norm[is.na(NN_norm)] <- 0

    # Numerator
    Num = similarity_vector %*% NN_norm

    #Prediction
    prediction[u, ] =  mean(test_data[u, ], na.rm=TRUE)  + (Num/sum(
   similarity_vector, na.rm=TRUE))



    if (onlyNew == TRUE){
      unseen <- names(test_data[u, is.na(test_data[u,])])
      prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
   , ], NA)
    }else{
      prediction2[u, ] <- prediction[u, ]
    }

    TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])

  }

  print("Prediction done")
```

```
  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}

#Pearson Correlation function for one user

UserBasedCFOneUser_Pearson <- function(dataet, user, N, NN, onlyNew=TRUE){

  ### similarity ###
  similarity_vect <- vector(, nrow(dataet))
  names(similarity_vect) <- rownames(dataet)
  for (i in rownames(dataet)){
    if (i != user){
      #sim <- sum(dataet[user, ]*dataet[i,], na.rm=TRUE)/sqrt(sum(dataet[user,
 ]^2, na.rm=TRUE) * sum(dataet[i, ]^2, na.rm=TRUE))
      sim = sum(meandiff(dataet,user) * meandiff(dataet,i), na.rm = TRUE) /
        (sqrt(sum(meandiff(dataet,user)^2,na.rm=TRUE)) *
           sqrt(sum(meandiff(dataet,i)^2,na.rm=TRUE)))
      similarity_vect[i] <- sim
    }
  }

  ### Nearest Neighbors ###
  crit_val <- -sort(-similarity_vect)[NN]
  similarity_vect <- na.omit(ifelse(similarity_vect >= crit_val,
 similarity_vect, NA))

  ### Prediction ###
  # Prepare
  NN_norm <- dataet[rownames(dataet) %in% names(similarity_vect),]
  CM <- colMeans(dataet, na.rm=TRUE)
  for (l in 1:ncol(NN_norm)){
    NN_norm[,l] <- NN_norm[,l] - CM[l]
  }
  NN_norm[is.na(NN_norm)] <- 0

  # Numerator
  Num = similarity_vect %*% NN_norm

```

```
  #Prediction
  prediction = mean(dataet[user, ], na.rm=TRUE) + (Num/sum(similarity_vect, na.rm=TRUE))
  names(prediction) = colnames(dataet)

  if (onlyNew == TRUE){
    unseen <- names(dataet[user, is.na(dataet[user,])])
    prediction <- prediction[names(prediction) %in% unseen]
  }
  TopN <- head(-sort(-prediction), N)

  return(TopN)
}
```

Listing 2.1: Pearson Correlation Functions

2.5.2 Cosine Similarity Functions

```
####Functions for user based collaborative filtering using cosine similarity ####

#Cosine function for multiple users
UserBasedCF <- function(train_data, test_data, N, NN, onlyNew=TRUE){

  ### similarity ###
  similarity_matrix <- matrix(, nrow = nrow(test_data), ncol = nrow(train_data),
                              dimnames = list(rownames(test_data), rownames(train_data)))

  for (i in rownames(test_data)){
    for (j in rownames(train_data)){
      sim <- sum(test_data[i, ]*train_data[j,], na.rm=TRUE)/sqrt(sum(test_data[i, ]^2, na.rm=TRUE) * sum(train_data[j, ]^2, na.rm=TRUE))
      similarity_matrix[i,j] <- sim
    }
  }
  print("similarity calculation done")
  ### Nearest Neighbors ###
  similarity_matrix_NN <- similarity_matrix

  for (k in 1:nrow(similarity_matrix_NN)){
```

```
  crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
  similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val, similarity_matrix_NN[k,], NA)
}

print("Nearest Neighbor selection done")
### Prediction ###
# Prepare
prediction <- matrix(, nrow=nrow(test_data), ncol(test_data),
                     dimnames=list(rownames(test_data), colnames(test_data)))
prediction2 <- matrix(, nrow=nrow(test_data), ncol(test_data),
                      dimnames=list(rownames(test_data), colnames(test_data)))

TopN <- matrix(, nrow=nrow(test_data), ncol=N, dimnames=list(rownames(test_data)))
### Numerator ###
for (u in rownames(test_data)){
  similarity_vector <- na.omit(similarity_matrix_NN[u, ])

  NN_norm <- train_data[rownames(train_data) %in% names(similarity_vector),]

  CM <- colMeans(train_data, na.rm=TRUE)
  for (l in 1:ncol(NN_norm)){
    NN_norm[,l] <- NN_norm[,l] - CM[l]
  }
  NN_norm[is.na(NN_norm)] <- 0

  # Numerator
  Num = similarity_vector %*% NN_norm

  #Prediction
  prediction[u, ] =  mean(test_data[u, ], na.rm=TRUE)  + (Num/sum(similarity_vector, na.rm=TRUE))


  if (onlyNew == TRUE){
    unseen <- names(test_data[u, is.na(test_data[u,])])
    prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u, ], NA)
  }else{
    prediction2[u, ] <- prediction[u, ]
```

```
    }

    TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])

  }

  print("Prediction done")

  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}

#Cosine function for single user
UserBasedCFOneUser <- function(dataet, user, N, NN, onlyNew=TRUE){

  ### similarity ###
  similarity_vect <- vector(, nrow(dataet))
  names(similarity_vect) <- rownames(dataet)
  for (i in rownames(dataet)){
    if (i != user){
      sim <- sum(dataet[user, ]*dataet[i,], na.rm=TRUE)/sqrt(sum(dataet[user, ]^2, na.rm=TRUE) * sum(dataet[i, ]^2, na.rm=TRUE))
      similarity_vect[i] <- sim
    }
  }

  ### Nearest Neighbors ###
  crit_val <- -sort(-similarity_vect)[NN]
  similarity_vect <- na.omit(ifelse(similarity_vect >= crit_val, similarity_vect, NA))

  ### Prediction ###
  # Prepare
  NN_norm <- dataet[rownames(dataet) %in% names(similarity_vect),]
  CM <- colMeans(dataet, na.rm=TRUE)
  for (l in 1:ncol(NN_norm)){
    NN_norm[,l] <- NN_norm[,l] - CM[l]
  }
  NN_norm[is.na(NN_norm)] <- 0

```

```
  # Numerator
  Num = similarity_vect %*% NN_norm

  #Prediction
  prediction = mean(dataet[user, ], na.rm=TRUE) + (Num/sum(similarity_vect, na.rm=TRUE))
  names(prediction) = colnames(dataet)

  if (onlyNew == TRUE){
    unseen <- names(dataet[user, is.na(dataet[user,])])
    prediction <- prediction[names(prediction) %in% unseen]
  }
  TopN <- head(-sort(-prediction), N)

  return(TopN)
} ####Functions for user based collaborative filteringg using cosine simillarity ####

#Cosine function for multiple users
UserBasedCF <- function(train_data, test_data, N, NN, onlyNew=TRUE){

  ### similarity ###
  similarity_matrix <- matrix(, nrow = nrow(test_data), ncol = nrow(train_data),
                              dimnames = list(rownames(test_data), rownames(train_data)))

  for (i in rownames(test_data)){
    for (j in rownames(train_data)){
      sim <- sum(test_data[i, ]*train_data[j,], na.rm=TRUE)/sqrt(sum(test_data[i, ]^2, na.rm=TRUE) * sum(train_data[j, ]^2, na.rm=TRUE))
      similarity_matrix[i,j] <- sim
    }
  }
  print("similarity calculation done")
  ### Nearest Neighbors ###
  similarity_matrix_NN <- similarity_matrix

  for (k in 1:nrow(similarity_matrix_NN)){
    crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
    similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val, similarity_matrix_NN[k,], NA)
```

```r
}

print("Nearest Neighbor selection done")
### Prediction ###
# Prepare
prediction <- matrix(, nrow=nrow(test_data), ncol(test_data),
                     dimnames=list(rownames(test_data), colnames(test_data)))
prediction2 <- matrix(, nrow=nrow(test_data), ncol(test_data),
                      dimnames=list(rownames(test_data), colnames(test_data))
 )

TopN <- matrix(, nrow=nrow(test_data), ncol=N, dimnames=list(rownames(
 test_data)))
### Numerator ###
for (u in rownames(test_data)){
  similarity_vector <- na.omit(similarity_matrix_NN[u, ])

  NN_norm <- train_data[rownames(train_data) %in% names(similarity_vector),]

  CM <- colMeans(train_data, na.rm=TRUE)
  for (l in 1:ncol(NN_norm)){
    NN_norm[,l] <- NN_norm[,l] - CM[l]
  }
  NN_norm[is.na(NN_norm)] <- 0

  # Numerator
  Num = similarity_vector %*% NN_norm

  #Prediction
  prediction[u, ] =  mean(test_data[u, ], na.rm=TRUE)  + (Num/sum(
 similarity_vector, na.rm=TRUE))


  if (onlyNew == TRUE){
    unseen <- names(test_data[u, is.na(test_data[u,])])
    prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
 , ], NA)
  }else{
    prediction2[u, ] <- prediction[u, ]
  }

  TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])
```

```

  }

  print("Prediction done")

  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}

#Cosine function for single user
UserBasedCFOneUser <- function(dataet, user, N, NN, onlyNew=TRUE){

  ### similarity ###
  similarity_vect <- vector(, nrow(dataet))
  names(similarity_vect) <- rownames(dataet)
  for (i in rownames(dataet)){
    if (i != user){
      sim <- sum(dataet[user, ]*dataet[i,], na.rm=TRUE)/sqrt(sum(dataet[user,
 ]^2, na.rm=TRUE) * sum(dataet[i, ]^2, na.rm=TRUE))
      similarity_vect[i] <- sim
    }
  }

  ### Nearest Neighbors ###
  crit_val <- -sort(-similarity_vect)[NN]
  similarity_vect <- na.omit(ifelse(similarity_vect >= crit_val,
 similarity_vect, NA))

  ### Prediction ###
  # Prepare
  NN_norm <- dataet[rownames(dataet) %in% names(similarity_vect),]
  CM <- colMeans(dataet, na.rm=TRUE)
  for (l in 1:ncol(NN_norm)){
    NN_norm[,l] <- NN_norm[,l] - CM[l]
  }
  NN_norm[is.na(NN_norm)] <- 0

  # Numerator
  Num = similarity_vect %*% NN_norm

```

```
  #Prediction
  prediction = mean(dataet[user, ], na.rm=TRUE) + (Num/sum(similarity_vect, na.
   rm=TRUE))
  names(prediction) = colnames(dataet)

  if (onlyNew == TRUE){
    unseen <- names(dataet[user, is.na(dataet[user,])])
    prediction <- prediction[names(prediction) %in% unseen]
  }
  TopN <- head(-sort(-prediction), N)

  return(TopN)
}
```

Listing 2.2: Cosine Similarity Functions

2.5.3 Pearson Correlation for Item Based Collaborative Filtering

```
#Item based collaborative filtering using pearson correlation
meandiff2 <- function(data,i){
  data[,i] - mean(data[,i],na.rm=TRUE)
}


ItemBasedCF_pearson <- function(train_data, test_data, N, NN, onlyNew=TRUE){
  similarity_matrix = matrix(, ncol=ncol(test_data), nrow=ncol(train_data),
   dimnames = list(colnames(test_data), colnames(train_data)))
  for (i in colnames(test_data)){
    for (j in colnames(train_data)){
      sim = sum(meandiff2(test_data,i) * meandiff2(train_data,j), na.rm = TRUE)
    /
        (sqrt(sum(meandiff2(test_data,i)^2,na.rm=TRUE)) *
           sqrt(sum(meandiff2(train_data,j)^2,na.rm=TRUE)))
      similarity_matrix[i,j] <- sim
    }
  }
  print("Similarity calculation done")

  # Nearest Neighbor
  similarity_matrix_NN <- similarity_matrix

  for (k in 1:ncol(similarity_matrix_NN)){
```

```
    crit_val <- -sort(-similarity_matrix_NN[,k])[NN]
    similarity_matrix_NN[,k] <- ifelse(similarity_matrix_NN[,k] >= crit_val, similarity_matrix_NN[,k], NA)
  }
  similarity_matrix_NN[is.na(similarity_matrix_NN)] <- 0

  train_data[is.na(train_data)] <- 0

  test_data2 <- test_data
  test_data2[is.na(test_data2)] <- 0

  print("Nearest neighbor selection done")

  ### Prediction ###
  prediction <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
                       dimnames=list(rownames(test_data), colnames(test_data)))
  prediction2 <- matrix(, nrow=nrow(test_data), ncol(test_data),
                        dimnames=list(rownames(test_data), colnames(test_data)))
  TopN <- matrix(, nrow=nrow(test_data), N, dimnames=list(rownames(test_data)))

  for (u in rownames(test_data)){
    # Numerator
    Num <-  test_data2[u, ] %*% similarity_matrix_NN

    # Denominator
    Denom <- colSums(similarity_matrix_NN, na.rm=TRUE)

    # Prediction
    prediction[u, ] <- Num/Denom

    if (onlyNew == TRUE){
      unseen <- names(test_data[u, is.na(test_data[u,])])
      prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u, ], NA)
    }else{
      prediction2[u, ] <- prediction[u, ]
    }

    TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])

  }
```

```

  print("Prediction done")

  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}
```

Listing 2.3: Pearson Correlation for Item Based Collaborative Filtering

2.5.4 Cosine Similarity for Item Based Collaborative Filtering

```
#Item based collaborative filtering using cosine similarity

ItemBasedCF <- function(train_data, test_data, N, NN, onlyNew=TRUE){
  # Similarity

  similarity_matrix <- as.matrix(simil(t(train_data), method="cosine"))

  print("Similarity calculation done")
  # Nearest Neighbor
  similarity_matrix_NN <- similarity_matrix

  for (k in 1:ncol(similarity_matrix_NN)){
    crit_val <- -sort(-similarity_matrix_NN[,k])[NN]
    similarity_matrix_NN[,k] <- ifelse(similarity_matrix_NN[,k] >= crit_val,
   similarity_matrix_NN[,k], NA)
  }
  similarity_matrix_NN[is.na(similarity_matrix_NN)] <- 0

  train_data[is.na(train_data)] <- 0

  test_data2 <- test_data
  test_data2[is.na(test_data2)] <- 0

  print("Nearest neighbor selection done")

  ### Prediction ###
  prediction <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
                       dimnames=list(rownames(test_data), colnames(test_data)))
  prediction2 <- matrix(, nrow=nrow(test_data), ncol(test_data),
```

```
                            dimnames=list(rownames(test_data), colnames(test_data))
    )
  TopN <- matrix(, nrow=nrow(test_data), N, dimnames=list(rownames(test_data)))

  for (u in rownames(test_data)){
    # Numerator
    Num <-  test_data2[u, ] %*% similarity_matrix_NN

    # Denominator
    Denom <- colSums(similarity_matrix_NN, na.rm=TRUE)

    # Prediction
    prediction[u, ] <- Num/Denom

    if (onlyNew == TRUE){
      unseen <- names(test_data[u, is.na(test_data[u,])])
      prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
    , ], NA)
    }else{
      prediction2[u, ] <- prediction[u, ]
    }

    TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])

  }

  print("Prediction done")

  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}
```

Listing 2.4: Cosine Similarity for Item Based Collaborative Filtering

2.5.5 Demographic Based Filtering/Cluster Based Collaborative Filtering

```
#Cluster Based collaborative filtering function (similar to demographic based
    filtering)
```

```
DemographicBasedF <- function(data, N, centers, iter, onlyNew=TRUE){

  data2 <- data

  # fill with average product rating
  colmeans <- colMeans(data2, na.rm=TRUE)

  for (j in colnames(data2)){
    data2[, j] <- ifelse(is.na(data2[ ,j]), colmeans[j], data2[, j])
  }

  km <- kmeans(data2, centers=centers, iter.max=iter)

  head(km$cluster)
  head(km$centers)


  # Statistics of the groups
  tab <- table(km$cluster)

  # Assign users to groups
  RES <- cbind(data, as.data.frame(km$cluster))

  # Calculate average ratings for everi cluster
  aggregation <- aggregate(RES, list(RES$"km$cluster"), mean, na.rm=T)
  aggregation <- aggregation[,-1]

  # Make a prediction
  users <- as.data.frame(RES$"km$cluster")
  users <- cbind(users, rownames(RES))
  colnames(users) <- c("km$cluster", 'rn')
  rec()

  prediction = merge(users, aggregation, by="km$cluster")
  rownames(prediction) <- prediction$rn

  prediction  <- prediction[order(rownames(prediction)), -1:-2]

  prediction2 <- matrix(, nrow=nrow(prediction), ncol(prediction),
                        dimnames=list(rownames(prediction), colnames(prediction
   )))
```

```r
  colnames(prediction2) <- colnames(prediction)
  rownames(prediction2) <- rownames(prediction)

  for (u in rownames(prediction)){
    if (onlyNew == TRUE){
      unseen <- names(data[u, is.na(data[u,])])

      prediction2[u, ] <- as.numeric(t(ifelse(colnames(prediction) %in% unseen,
    prediction[u, ], as.numeric(NA))))
    }else{
      prediction2[u, ] <- prediction[u, ]
    }
  }

  # TopN
  TopN <- t(apply(prediction, 1, function(x) names(head(sort(x, decreasing=TRUE
   ), 5))))

  print("Prediction done")

  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}
```

Listing 2.5: Cluster Based Collaborative Filtering

2.5.6 Content Based Filtering

```r

#Content Based Filtering

ContentBased <- function(product_data, test_data, N, NN, onlyNew=TRUE){

  # Similarity calculation (copied from user-based collaborative filtering)
  similarity_matrix <- as.matrix(simil(product_data, method="cosine"))

  print("Similarity calculation done")

  # Set Nearest neighbors (copied from user-based collaborative filtering)
  similarity_matrix_NN <- similarity_matrix
```

```
  for (k in 1:nrow(similarity_matrix_NN)){
    crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
    similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val,
 similarity_matrix_NN[k,], 0)
  }

  similarity_matrix_NN[is.na(similarity_matrix_NN)] <- 0
  test_data2 <- test_data
  test_data2[is.na(test_data2)] <- 0

  print("Nearest neighbor selection done")

  ### Prediction (copied from item based collaborative filtering) ###
  prediction <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
                       dimnames=list(rownames(test_data), colnames(test_data)))
  prediction2 <- matrix(, nrow=nrow(test_data), ncol(test_data),
                        dimnames=list(rownames(test_data), colnames(test_data))
 )
  TopN <- matrix(, nrow=nrow(test_data), N, dimnames=list(rownames(test_data)))

  for (u in rownames(test_data)){
    # Numerator
    Num <-  test_data2[u, ] %*% similarity_matrix_NN

    # Denominator
    Denom <- colSums(similarity_matrix_NN, na.rm=TRUE)

    # Prediction
    prediction[u, ] <- Num/Denom

    if (onlyNew == TRUE){
      unseen <- names(test_data[u, is.na(test_data[u,])])
      prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
 , ], NA)
    }else{
      prediction2[u, ] <- prediction[u, ]
    }

    TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])

  }
```

```r

  print("Prediction done")

  res <- list(prediction, TopN)
  names(res) <- c('prediction', 'topN')

  return(res)
}
```

Listing 2.6: Content Based Filtering

2.5.7 Predictive Accuracy

```r

#####Evaluation Metrics

### Prediction Accuracy ###
###########################

RSME <- function(prediction, real){

  if (nrow(prediction) == nrow(real) & ncol(prediction) == ncol(real)){
    RSME = sqrt( sum( (prediction - real)^2 , na.rm = TRUE ) / (nrow(prediction
    ) * ncol(prediction)) )
    return(RSME)
  }else{
    return("Dimension of prediction are not equal to dimension of real")
  }
}

MAE <- function(prediction, real){

  if (nrow(prediction) == nrow(real) & ncol(prediction) == ncol(real)){
    RSME = sum( Mod(prediction - real) , na.rm = TRUE ) / (nrow(prediction) *
    ncol(prediction))
    return(RSME)
  }else{
    return("Dimension of prediction are not equal to dimension of real")
  }
}
```

Listing 2.7: Predictive Accuracy

2.5.8 Classification Accuracy

```


#########Classification accuracy#########

 Classification <- function(prediction, real, threshold=NA, TopN=NA){
   if (nrow(prediction) == nrow(real) & ncol(prediction) == ncol(real)){
     # Threshold #
     if (!is.na(threshold)){
       TP = sum(ifelse(prediction >= threshold & real >= threshold, 1, 0), na.rm
=T)
       FP = sum(ifelse(prediction >= threshold & real < threshold, 1, 0), na.rm=
T)
       FN = sum(ifelse(prediction < threshold & real >= threshold, 1, 0), na.rm=
T)
       Recall = TP/(TP+FN)
       Precision = TP/(TP+FP)
       F1 = 2 * ((Precision * Recall) / (Precision + Recall))
       Class_Thres = list(Recall, Precision, F1)
       names(Class_Thres) = c("Recall","Precision","F1")
     }
     if (!is.na(TopN)){
       TP = vector(, length = nrow(prediction))
       FP = vector(, length = nrow(prediction))
       FN = vector(, length = nrow(prediction))

       for (u in nrow(prediction)){
         threshold_pred = -sort(-prediction[u, ])[TopN]
         threshold_real = -sort(-real[u, ])[TopN]
         TP[u] = sum(ifelse(prediction[u, ] >= threshold_pred & real[u, ] >=
threshold_real, 1, 0), na.rm=T)
         FP[u] = sum(ifelse(prediction[u, ] >= threshold_pred & real[u, ] <
threshold_real, 1, 0), na.rm=T)
         FN[u] = sum(ifelse(prediction[u, ] < threshold_pred & real[u, ] >=
threshold_real, 1, 0), na.rm=T)
       }
       TP = sum(TP[u])
       FP = sum(FP[u])
       FN = sum(FN[])
       Recall = TP/(TP+FN)
       Precision = TP/(TP+FP)
```

```
      F1 = 2 * ((Precision * Recall) / (Precision + Recall))
      Class_TopN = list(Recall, Precision, F1)
      names(Class_TopN) = c("Recall", "Precision", "F1")
    }

    if (!is.na(threshold) & !is.na(TopN)){
      Class = list(Class_Thres, Class_TopN)
      names(Class) = c("Threshold", "TopN")
    }else if (!is.na(threshold) & is.na(TopN)) {
      Class = Class_Thres
    }else if (is.na(threshold) & !is.na(TopN)) {
      Class = Class_TopN
    }else{
      Class = "You have to specify the 'Threshold' or 'TopN' parameter!"
    }
    return(Class)
  }else{
    return("Dimension of prediction are not equal to dimension of real")
  }
}
```

Listing 2.8: Classification Accuracy

2.5.9 Ranking Accuracy

```


####### Ranking accuracy########

AUC <- function(real, prediction, threshold){

  pred <- ifelse(prediction >= threshold, 1, 0)
  real <- ifelse(real >= threshold, 1, 0)

  real[is.na(real)] <- 0
  pred[is.na(pred)] <- 0

  ROC <- roc(factor(prediction), factor(real))

  plot(ROC)

  AUC <- auc(ROC)
```

```
  return(AUC)
}


#produced errors when computing, excluded from report
NDCG <- function(real, prediction, TopN){
  for (u in rownames(real)){

    # compute ranking
    rank <- sort(-rank(prediction[u,]))[1:TopN]


    # Create NDCG vector
    if ( u == rownames(real)[1]){
      NDCG_vect <- Evaluation.NDCG(rank, real[u, names(rank)])
    }else{
      NDCG_vect <- rbind(NDCG_vect, Evaluation.NDCG(rank, real[u, names(rank)])
   )
    }
  }

  # Compute avarege NDCG
  NDCG_vect[is.na(NDCG_vect)] <- 0
  NDCG <- colMeans(NDCG_vect, na.rm=T)
  names(NDCG) <- "NDCG"
  return(NDCG)
}

#taken from https://github.com/mlr-org/mlr/
library(mlr)


#taken from https://github.com/mhahsler/recommenderlab
library(recommenderlab)

#taken from https://github.com/tidyverse/tidyr
library(tidyr)

#taken from https://github.com/tidyverse/dplyr
library(dplyr)
```

Listing 2.9: Ranking Accuracy

2.5.10 Data Manipulation For Collaborative Based Filtering

```

########################################Collaborative Filtering ######################################################################

#Reading the data
user_artists = read.table("user_artists.dat",header = TRUE,sep='\t')
mean(user_artists$weight)
sd(user_artists$weight)
{

#Checking the normalization and skewness of data

normaliseddata = dnorm(user_artists$weight,mean=745.2439,sd=3751.322)

hist(normaliseddata,prob = TRUE)

max(user_artists$weight)
min(user_artists$weight)


#Calculating a new column with the log values for optimization
user_artists$log_weight = log(user_artists$weight)

#Using linear model to find the coefficient
model = lm(log_weight ~ weight, user_artists)

summary(model)

#Adding the coefficient value to the log value to get a uniformed weight distribution
user_artists$log_weight = log(user_artists$weight) + 5.389e+00

#plotting histogram
hist(user_artists$log_weight)
str(user_artists)
}
#deleting the weight column
user_artists_updated = user_artists
user_artists_updated$weight = NULL

```

```r
#filtering the users for normalization and weights > 10 and number of users > 5

artists_subset =  user_artists_updated %>% filter(log_weight > 10) %>% group_by
    (artistID) %>% summarise(Totalusers = n_distinct(userID)) %>% filter(
    Totalusers > 5)


#sub-setting the data with the values from the filtered users
user_artists_updated1 = subset(user_artists_updated, artistID %in%
    artists_subset$artistID)

length(unique(user_artists_updated1$artistID))
length(unique(user_artists_updated1$userID))

#Spreading the data
user_artists_transform = spread(user_artists_updated1,artistID,log_weight)


#Converting the row names to indices
row.names(user_artists_transform) <- user_artists_transform$userID


#min(transpose_matrix)
user_artists_transform[,1] = NULL

row.names(user_artists_transform)
names(user_artists_transform)

#converting the data into matrix
user_artists_transform_matrix = as(user_artists_transform,"matrix")


# Number of users and artists
nrow(user_artists_transform_matrix)
ncol(user_artists_transform_matrix)

# Min, max and average rating for the artists
min(user_artists_transform_matrix, na.rm=TRUE)
max(user_artists_transform_matrix, na.rm=TRUE)
mean(user_artists_transform_matrix, na.rm=TRUE)
```

```
hist(user_artists_transform_matrix)
t = table(is.na(user_artists_transform_matrix))
t

# sparsity
t[2]/(t[1]+t[2])
#98.18503% sparse data


#Split the data into test and train

train_id = sample(1:nrow(user_artists_transform_matrix), 0.7 * nrow(
   user_artists_transform_matrix))
test_id <- setdiff(1:nrow(user_artists_transform_matrix), train_id)

train_data = user_artists_transform_matrix[train_id,]
test_data = user_artists_transform_matrix[test_id,]


#Cosine correlation for single user

UserBasedCFOneUser(user_artists_transform_matrix,'6',3,10,onlyNew = TRUE)

#Cosine correlation for multiple user

prediction_cosine = UserBasedCF(train_data,test_data,3,15,onlyNew = TRUE)
prediction_cosine_preddata = prediction_cosine$prediction
prediction_cosine_topN = prediction_cosine$topN

write.csv(prediction_cosine_preddata,file = "prediction_cosine_preddata.csv")
write.csv(prediction_cosine_topN,file= "prediction_cosine_topN.csv")

#Pearson correlation for single user
UserBasedCFOneUser_Pearson(user_artists_transform_matrix,'6',3,10,onlyNew =
   TRUE)

# takem from https://github.com/HenrikBengtsson/parallelly
library(doParallel)
k=detectCores()
cl <- makeCluster(k-1)
#Pearson correlation for multiple user
prediction_pearson = UserBasedCF_pearson(train_data,test_data,3,15,onlyNew =
```

```
    TRUE)
prediction_pearson_preddata <- as.data.frame(prediction_pearson$prediction)

TopN_pearson <- as.data.frame(prediction_pearson$topN)

write.csv(prediction_pearson_preddata,file ="prediction_pearson_preddata.csv")
write.csv(TopN_pearson,file = "TopN_pearson.csv")

#taken from https://github.com/mhahsler/recommenderlab
#Using "recommender lab" library

train_data = user_artists_transform_matrix[train_id,]
test_data = user_artists_transform_matrix[test_id,]
train = as(train_data,"realRatingMatrix")
test = as(test_data,"realRatingMatrix")

recommenderRegistry$get_entry("UBCF", dataType="realRatingMatrix")

recom_Userbased <- Recommender(train, method = "UBCF")
predd_recomm <- predict(recom_Userbased,newdata=test@data@p[66],n=10)


#########Item based collaborative filtering##########
k=detectCores()
cl <- makeCluster(k-1)
#Item based collaborative filtering using pearson
prediction_item_pearson = ItemBasedCF_pearson(train_data,test_data,3, 10,
    onlyNew=TRUE)
prediction_item_pearson_data = prediction_item_pearson$prediction
prediction_item_pearson_TopN = prediction_item_pearson$top
prediction_item_pearson_TopN

write.csv(prediction_item_pearson_data,file= "prediction_item_pearson_data.csv"
    )
write.csv(prediction_item_pearson_TopN,file="prediction_item_pearson_topN.csv")

#Item based collaborative filtering using cosine
prediction_item_cosine = ItemBasedCF(train_data,test_data,3, 15, onlyNew=TRUE)
prediction_item_cosine_data = prediction_item_cosine$prediction
prediction_item_cosine_TopN = prediction_item_cosine$topN
prediction_item_pearson_TopN

```

```
write.csv(prediction_item_cosine_data,file="prediction_item_cosine_data.csv")
write.csv(prediction_item_cosine_TopN,file = "prediction_item_cosine_TopN.csv")
```

Listing 2.10: Data Manipulation For Collaborative Based Filtering

2.5.11 Performing Evaluation Metrics

```
#########################################Evaluation Metrics#################

######Userbased#######
##Prediction accuracy with the results from pearson
RSME(prediction_pearson$prediction,test_data)
MAE(prediction_pearson$prediction,test_data)

##Prediction accuracy with the results from cosine
RSME(prediction_cosine$prediction,test_data)
MAE(prediction_cosine$prediction,test_data)

######itembased#######
##Prediction accuracy with the results from pearson
RSME(prediction_item_pearson$prediction,test_data)
MAE(prediction_item_pearson$prediction,test_data)

##Prediction accuracy with the results from cosine
RSME(prediction_item_cosine$prediction,test_data)
MAE(prediction_item_cosine$prediction,test_data)


###userbased classification accuracy
##Classification accuracy with pearson
Classification(prediction_pearson$prediction, test_data, threshold=5, TopN=10)

##Classification accuracy with cosine
Classification(prediction_cosine$prediction, test_data, threshold=5, TopN=10)

###itembased classification accuracy
##Classification accuracy with pearson
Classification(prediction_item_pearson$prediction, test_data, threshold=5, TopN
   =20)

##Classification accuracy with cosine
```

```
Classification(prediction_item_cosine$prediction, test_data, threshold=6, TopN=10)

######Ranking accuracy
####Ranking accuracy user based
#Using pearson function

#taken from https://rdrr.io/cran/DescTools/src/R/StatsAndCIs.r
library(AUC)
#used for graphs

AUC(test_data, prediction_pearson$prediction, 5)

#Using cosine function
NDCG(test_data, prediction_cosine$prediction, 5)
AUC(test_data, prediction_cosine$prediction, 5)

####Ranking accuracy item based
#Using pearson function
NDCG(test_data, prediction_item_pearson$prediction, 5)
AUC(test_data, prediction_item_pearson$prediction, 5)

#Using cosine function
NDCG(test_data, prediction_item_cosine$prediction, 5)
AUC(test_data, prediction_item_cosine$prediction, 10)

# #########################################################################################
# DEMOGRAPHIC BASED FILTERING
# #########################################################################################
#getting user features. converting to matrix and feeding them into the demographic filter function
rec=function(){}
Prediction_cluster_matrix = DemographicBasedF(user_artists_transform_matrix, 3, 150, 75, onlyNew=TRUE)
Prediction_cluster_prediction_matrix = as.data.frame(Prediction_cluster_matrix$prediction)
Prediction_cluster_topN_matrix = as.data.frame(Prediction_cluster_matrix$topN)
Prediction_cluster_topN_matrix
```

```
Prediction_cluster_prediction_matrixtest = subset(Prediction_cluster_prediction_matrix, row.names(Prediction_cluster_prediction_matrix) %in% row.names(test_data))

dim(test_data)
row.names(test_data)
row.names(Prediction_cluster_prediction_matrix)

### Results for Cluster Based Filtering ###

#Evaluation metrics- Cluster Based

#Prediction accuracy2
RSME(Prediction_cluster_prediction_matrixtest,test_data)


#Classfication accuracy
Classification(Prediction_cluster_prediction_matrixtest,test_data,threshold = 10,TopN = NA)
```

Listing 2.11: Performing Evaluation Metrics

maybe recomender lab

2.5.12 Data Manipulation For Content Based Filtering

```
# #########################################################################################
# CONTENT BASED FILTERING
# #########################################################################################

#https://github.com/tidyverse/readr
library(readr)

#taken from https://github.com/tidyverse/dplyr
library(dplyr)

# taken from https://github.com/tidyverse
library(tidyverse)
```

```
#taken from https://github.com/tidyverse/tidyr
library(tidyr)

# import user_taggedartists file
user_taggedartists <-  read.table("user_taggedartists.dat", header=TRUE) %>%
   select(userID, artistID, tagID)

# import tags file
tags <-  read.delim("user_taggedartists.dat",header=TRUE)




# import arts file
art <- read_delim("artists.dat", delim = "\t") %>% select(id, name)
art$name <- iconv(art$name, from = "UTF-8", to = "ASCII//TRANSLIT")


# extract count of tags for each group of artists and tagID

tags_counts <- arrange(summarise(group_by(user_taggedartists, tagID),
                                 TotalUsers = length(unique(userID)) ), desc(
   TotalUsers) )

#length(unique(user_taggedartists$tagID))
tag_top200 <- tags_counts

# Take top 200 tags
tag_top200 <- arrange(tag_top200, tagID)

# subset tags which is having top 200
tag_top200$Names <- subset(tags, tagID %in% tag_top200$tagID)$tagValue

# Selecting the Top 200 Tags based on Maximum number of users

tag_top200 <- arrange(tag_top200, desc(TotalUsers))


toptags <- subset(user_taggedartists, tagID %in% tag_top200$tagID)

#Selecting only those Artists which are used by Collaborative based Filering
```

```
   Recommendation Systems
testart <- subset(user_taggedartists, artistID %in% user_artists$artistID)
testart1 <- subset(testart, artistID %in% user_artists_updated1$artistID)

# Deleting couple of records with issues.
user_artists_updated2 <- user_artists_updated1[!user_artists_updated1$artistID
    == "5533",]
user_artists_updated3 <- user_artists_updated2[!user_artists_updated2$artistID
    == "4941" ,]






#tag_top200 <- tags_counts[1:200,]

summarized_tag <- summarise(group_by(toptags, artistID, tagID ), Count = length
   (tagID) )

summarized_tag <- subset(summarized_tag, artistID  %in%
   user_artists_updated2$artistID)


# Creating the base Matrix

matrix <- spread(summarized_tag, tagID, Count)

row.names(matrix) <- matrix$artistID

matrix[,][is.na(matrix[,])] <- 0

ag_artistID <- as.vector(matrix$artistID)
ag_mat <- as.matrix(matrix[,2:ncol(matrix)])
rm(matrix)

ntags <- length(as.vector(ag_mat))
ntags

sum(!is.na(as.vector(ag_mat)) ) / ntags
1 - sum(!is.na(as.vector(ag_mat)) ) / ntags

```

```r
# Creating the Final Base Matrix for Content Based RS

fin_matrix <- ag_mat

fin_matrix[,][is.na(fin_matrix[,])] <- 0
fin_matrix[,][fin_matrix[,] > 0] <- 1


nrow(fin_matrix)
ncol(fin_matrix)

##########Updating original user based matrix
user_artists_updated2 <- user_artists_updated1[!user_artists_updated1$artistID
    == "5533",]
user_artists_updated3 <- user_artists_updated2[!user_artists_updated2$artistID
    == "4941" ,]



user_artists_new =spread(user_artists_updated3,artistID,log_weight)

#Converting the row names to indices
row.names(user_artists_new) <- user_artists_new$userID


#min(transpose_matrix)
user_artists_new[,1] = NULL

#row.names(user_artists_transform)
#names(user_artists_transform)

#converting the data into matrix
user_artists_transform_new = as(user_artists_new,"matrix")

write.csv(user_artists_transform_new , file = "user_artists_transform_new.csv")
write.csv(fin_matrix,file="content_matrix.csv")
set.seed(2)
train_rows = sample(1:nrow(user_artists_transform_new), 0.7*nrow(
   user_artists_transform_new))

train_content <- as(user_artists_transform_new, "matrix")[train_rows,]
test_content <- as(user_artists_transform_new, "matrix")[-train_rows,]
```

```

CB_updated <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=T)
CB_updated_pred = CB_updated

CB_updated$prediction
CB_updated$topN
```

Listing 2.12: Data Manipulation For Content Based Filtering

2.5.13 Quantitative Evaluations between systems

```


##############################################################
### Quantitative Evauation & comparison with item-based Collaborative Filtering
    ###
##############################################################

# Load Models

# Split train - Test



# Score Models

ptm <- proc.time()
CB <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=T)
Time <- (proc.time() - ptm)
Time

### Results for Content-Based Filtering


### Prediction Accuracy ###
###########################

# RSME Content-based
RSME(CB$prediction, test_content)

MAE(CB$prediction, test_content)

```

```
### Classification Accuracy ###
##############################

# Recall/precision Content-based
Classification(CB$prediction, test_content, threshold=3)
```

Listing 2.13: Quantitative Evaluations between systems

2.5.14 Qualitative Results

```
# ####################################################################################
# ####################################################################################
############### Qualitative Results :

#taken from https://github.com/mhahsler/recommenderlab
library(recommenderlab)
art_sim <- similarity(as(fin_matrix, "binaryRatingMatrix"), method = "cosine",
                      which = "users")

# convert to an R matrix
art_sim <- as(art_sim, "matrix")

# round to 3 digit precision
art_sim[][] <- round(art_sim[][],3)

# # name rows and collumns according to artistID for easy retrieval
colnames(art_sim) <- ag_artistID
rownames(art_sim) <- ag_artistID


##############################################################
# set number of similar artists to recommend
n_recommended <- 5

# randomly select a user
artist <- sample(ag_artistID, 1)

# get name of artist from artist list
```

```
a_name <- art[art$id == artist,]$name

# fetch their recommendations: this returns a named vector sorted by similarity
# the names of the items are the artist IDs
arecs <- sort(art_sim[as.character(artist),], decreasing = TRUE)[1:
   n_recommended]

# extract the artist IDs and convert to numeric
arecs_IDs <- as.numeric(names(arecs))

# create list of artist names from artist ID's in list
arec_names <- art[art$id %in% arecs_IDs,]$name

# create a heading for the list of similar artists
table_head <- sprintf("Artists Similar to %s", a_name)

# display the list of similar artists

#taken from https://github.com/yihui/knitr
library(knitr)
kable(arec_names, col.names = table_head)

#
   ##################################################################################

# Generate a Top N Artist List by Genre

set.seed(42)

# set rownames = artistID's for easy retrieval - DON'T NEED THIS LINE OF CODE
   IN SHINY
rownames(ag_mat) <- ag_artistID

# extract the genre tagIDs from matrix and convert to numeric
tagIDs <- as.numeric(colnames(ag_mat))

# set number of artists to recommend
n_recommended <- 5

# randomly select a genre
tagID <- sample(tagIDs, 1)

```

```r
# get name of genre from tagID list
g_name <- tags[tags$tagID == tagID,]$tagValue

# fetch the top N artists:
# the names of the items are the artist IDs
g_arecs <- sort(ag_mat[,as.character(tagID)], decreasing = TRUE)[1:n_recommended]

# extract the artist IDs and convert to numeric
g_arecs_IDs <- as.numeric(names(g_arecs))

# create list of artist names from artist ID's in list
g_arec_names <- art[art$id %in% g_arecs_IDs,]$name

# create a heading for the list of similar artists
table_head <- sprintf("Top Artists in %s genre:", g_name)

# display the list of similar artists
kable(g_arec_names, col.names = table_head)
```

Listing 2.14: Qualitative Results

2.5.15 Hybrid Recommendation System Results

```r

# ###################################################################################
 # ##################################################################################

####Hybrid Recommendation Systems:cluster based and content based

#Load the data from content and cluster based

# Split train - Test data from cluster based
set.seed(2)
train_rows = sample(1:nrow(user_artists_transform_new), 0.7*nrow(user_artists_transform_new))

train_content <- as(user_artists_transform_new, "matrix")[train_rows,]
```

```r
test_content <- as(user_artists_transform_new, "matrix")[-train_rows,]

### Compute individual models ###
#Content based
Contentbased <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=F)
Contentbased$topN
content_pred = Contentbased$prediction

#Cluster based
clusterbased <- DemographicBasedF(user_artists_transform_new,  3, 150, 75,
   onlyNew=T)
clusterbased$topN
cluster_prediction = clusterbased$prediction

Cluster_pred = subset(cluster_prediction, row.names(cluster_prediction) %in%
   row.names(test_content))
Cluster_pred1 = as(Cluster_pred,"matrix")

### Transform results to lists (to be able to use the rowMeans function) ###
content_list <- as.list(content_pred)
cluster_list <- as.list(Cluster_pred1)

####################
### Compute Mean ###
####################
hybrid <- rowMeans(cbind(as.numeric(content_list), as.numeric(cluster_list)),
   na.rm=T)

### Transform list back to matrix with correct number of dimensions ###
Hybrid_prediction <- matrix(hybrid, nrow=nrow(test_content), ncol=ncol(
   test_content))
rownames(Hybrid_prediction) <- rownames(test_content)
colnames(Hybrid_prediction) <- colnames(test_content)

### Evaluate the Metrics for Prediction accuracy and Classification accuracy
   ###


### Results for content based and cluster based collaborative filtering ###

# Prediction accuracy
RSME(Hybrid_prediction, test_content)
```

```
MAE(Hybrid_prediction, test_content)

# Classification
Classification(Hybrid_prediction, test_content, threshold=6)


####################
### Hybrid RecSys ###
####################

#taken from https://github.com/mhahsler/recommenderlab
library("recommenderlab")


### Split train - Test ###
set.seed(2)


train_rows = sample(1:nrow(user_artists_transform_matrix), 0.7*nrow(
   user_artists_transform_matrix))

train <- as(user_artists_transform_matrix, "matrix")[train_rows,]
test <- as(user_artists_transform_matrix, "matrix")[-train_rows,]


### Compute individual models ###
set.seed(2)
train_rows = sample(1:nrow(user_artists_transform_new), 0.7*nrow(
   user_artists_transform_new))

train_content <- as(user_artists_transform_new, "matrix")[train_rows,]
test_content <- as(user_artists_transform_new, "matrix")[-train_rows,]

CBTFIDF <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=F)
IB <- UserBasedCF(train, test, 3, 10, onlyNew=F)

### Transform results to lists (to be able to use the rowMeans function) ###
CBTFIDF_list <- as.list(CBTFIDF$prediction)
IB_list <- as.list(IB$prediction)

###################
### Compute Mean ###
```

```
####################
hybrid <- rowMeans(cbind(as.numeric(CBTFIDF_list), as.numeric(IB_list)), na.rm=
   T)

### Transform list back to matrix with correct number of dimensions ###
Hybrid_prediction <- matrix(hybrid, nrow=nrow(test), ncol=ncol(test))
rownames(Hybrid_prediction) <- rownames(test)
colnames(Hybrid_prediction) <- colnames(test)

### Evaluate ###

### Results for content based and user based collaborative filtering ###



# MAE
MAE(Hybrid_prediction, test)

# RMSE
RSME(Hybrid_prediction, test)



# Classification

Classification(Hybrid_prediction, test, threshold=5)
```

Listing 2.15: Hybrid Recommendation System Results

Chapter 3

Appendix C

3.1 Graphs and Charts

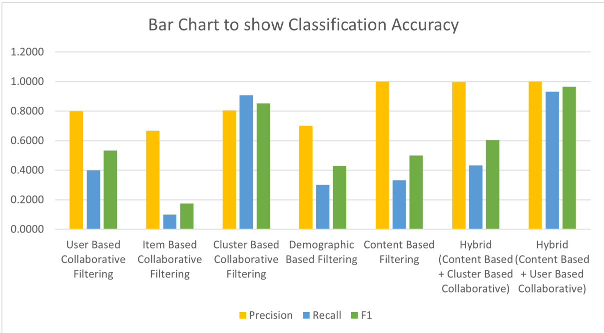

Figure 3.1: This bar chart shows how the the classification metrics compare with one another between systems

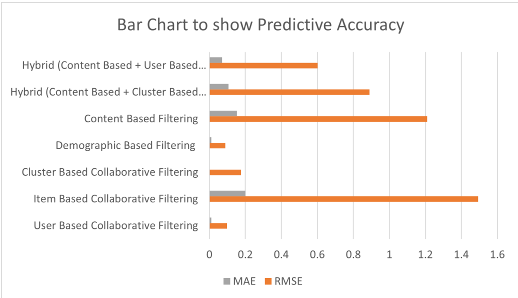

Figure 3.2: This bar chart shows how the the predictive accuracy metrics compare with one another between systems

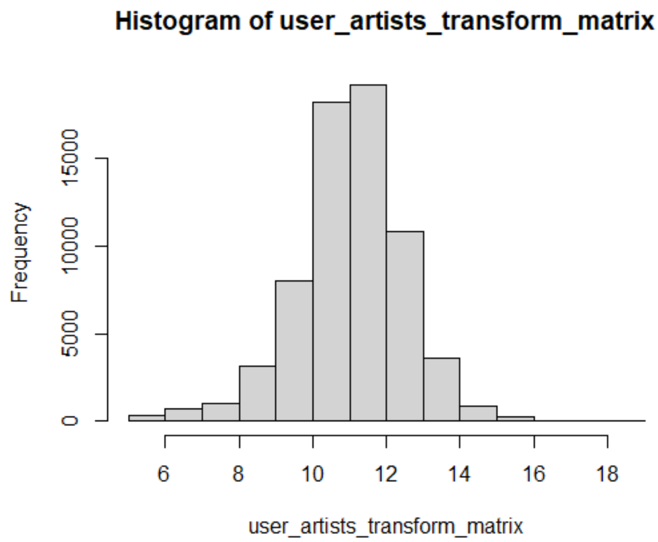

Figure 3.3: This histogram shows us how frequently an artist is chosen by users

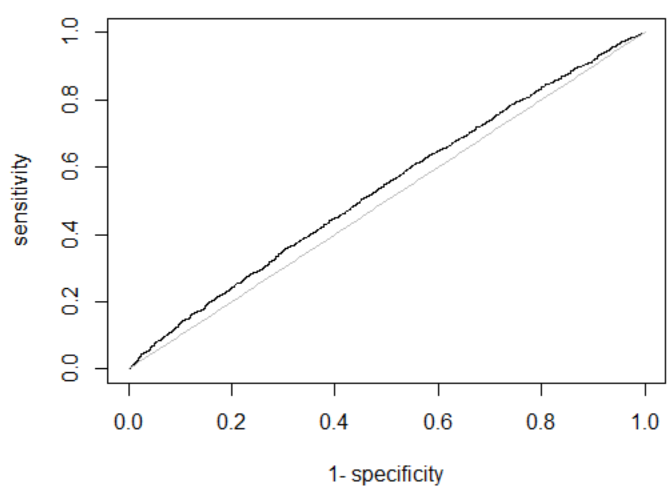

Figure 3.4: This is a graph to show the ROC Curve of how content based filtering changes in sensitivity with specificity.This is shown as the model is given more data

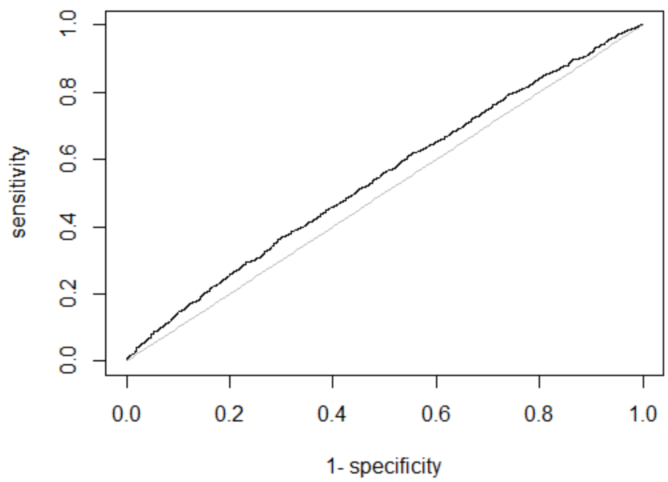

Figure 3.5: This is a graph to display the ROC Curve of how collaborative based filtering changes in sensitivity with specificity.This is shown as the model is given more data

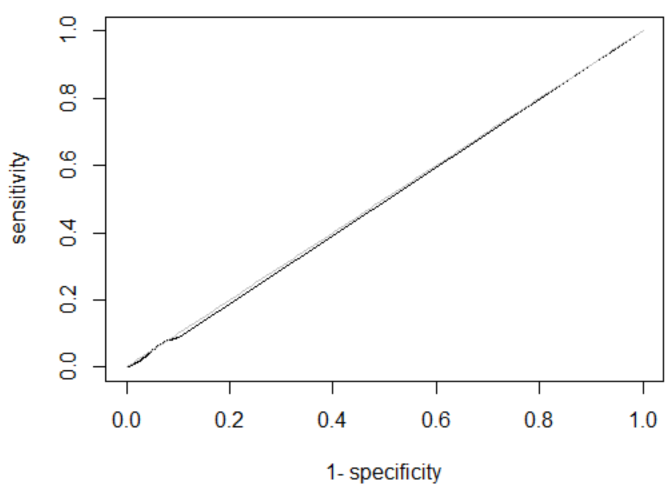

Figure 3.6: This is a graph to display the ROC Curve of how the hybrid system of content based filtering and collaborative based filtering changes in sensitivity with specificity.This is shown as the model is given more data

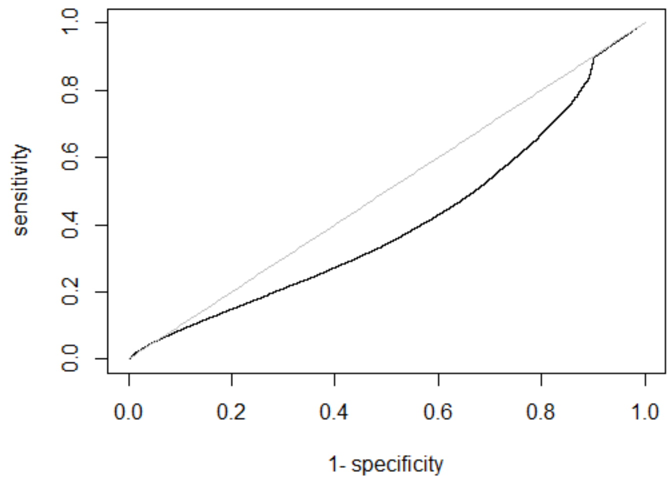

Figure 3.7: This is a graph to display the ROC Curve of how demographic based filtering changes in sensitivity with specificity.This is shown as the model is given more data