

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-scrobble-45e52454-b/data?select=lastfm_user_scrobble.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

```
1 setwd('C:/Users/Student/Desktop/PROJECT')
2 #####Functions for user based Collaborative Filtering using pearson correlation
3 #####
4 meandiff <- function(data,i){
5   data[i,] - mean(data[i,],na.rm=TRUE)
6 }
7
8 #User Based Collaborative Filtering for multiple users
9 UserBasedCF_pearson <- function(train_data, test_data, N, NN, onlyNew=TRUE){
10
11   ### similarity ###
12   similarity_matrix <- matrix(, nrow = nrow(test_data), ncol = nrow(train_data)
13   ,
14   dimnames = list(rownames(test_data), rownames(
15   train_data)))
16
17   for (i in rownames(test_data)){
18     for (j in rownames(train_data)){
19       sim = sum(meandiff(test_data,i) * meandiff(train_data,j), na.rm = TRUE) /
20       (sqrt(sum(meandiff(test_data,i)^2,na.rm=TRUE)) *
21       sqrt(sum(meandiff(train_data,j)^2,na.rm=TRUE)))
22       similarity_matrix[i,j] <- sim
23     }
24   }
25   print("similarity calculation done")
26   ### Nearest Neighbors ###
27   similarity_matrix_NN <- similarity_matrix
28
29   for (k in 1:nrow(similarity_matrix_NN)){
30     crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
31     similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val,
32     similarity_matrix_NN[k,], NA)
33   }
34
35   print("Nearest Neighbor selection done")
36   ### Prediction ###
37   # Prepare
```

```

35 prediction <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data)),
36                     dimnames=list(rownames(test_data), colnames(test_data)))
37 prediction2 <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data)),
38                     dimnames=list(rownames(test_data), colnames(test_data))
39 )
40 TopN <- matrix(, nrow=nrow(test_data), ncol=N, dimnames=list(rownames(
41   test_data)))
42 ### Numerator ###
43 for (u in rownames(test_data)){
44   similarity_vector <- na.omit(similarity_matrix_NN[u, ])
45   NN_norm <- train_data[rownames(train_data) %in% names(similarity_vector),]
46
47   CM <- colMeans(train_data, na.rm=TRUE)
48   for (l in 1:ncol(NN_norm)){
49     NN_norm[,l] <- NN_norm[,l] - CM[l]
50   }
51   NN_norm[is.na(NN_norm)] <- 0
52
53   # Numerator
54   Num = similarity_vector %*% NN_norm
55
56   #Prediction
57   prediction[u, ] = mean(test_data[u, ], na.rm=TRUE) + (Num/sum(
58     similarity_vector, na.rm=TRUE))
59
60
61   if (onlyNew == TRUE){
62     unseen <- names(test_data[u, is.na(test_data[u,])])
63     prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
64   , ], NA)
65   }else{
66     prediction2[u, ] <- prediction[u, ]
67   }
68
69   TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])
70 }
71
72 print("Prediction done")

```

```

73
74   res <- list(prediction, TopN)
75   names(res) <- c('prediction', 'topN')
76
77   return(res)
78 }
79
80 #Pearson Correlation function for one user
81
82 UserBasedCFOneUser_Pearson <- function(dataet, user, N, NN, onlyNew=TRUE){
83
84   ### similarity ###
85   similarity_vect <- vector(), nrow(dataet))
86   names(similarity_vect) <- rownames(dataet)
87   for (i in rownames(dataet)){
88     if (i != user){
89       #sim <- sum(dataet[user, ]*dataet[i,], na.rm=TRUE)/sqrt(sum(dataet[user,
90       ]^2, na.rm=TRUE) * sum(dataet[i, ]^2, na.rm=TRUE))
91       sim = sum(meandiff(dataet,user) * meandiff(dataet,i), na.rm = TRUE) /
92         (sqrt(sum(meandiff(dataet,user)^2,na.rm=TRUE)) *
93          sqrt(sum(meandiff(dataet,i)^2,na.rm=TRUE)))
94       similarity_vect[i] <- sim
95     }
96   }
97
98   ### Nearest Neighbors ###
99   crit_val <- -sort(-similarity_vect)[NN]
100   similarity_vect <- na.omit(ifelse(similarity_vect >= crit_val,
101   similarity_vect, NA))
102
103   ### Prediction ###
104   # Prepare
105   NN_norm <- dataet[rownames(dataet) %in% names(similarity_vect),]
106   CM <- colMeans(dataet, na.rm=TRUE)
107   for (l in 1:ncol(NN_norm)){
108     NN_norm[,l] <- NN_norm[,l] - CM[l]
109   }
110   NN_norm[is.na(NN_norm)] <- 0
111
112   # Numerator
113   Num = similarity_vect %*% NN_norm

```

```

113 #Prediction
114 prediction = mean(dataet[user, ], na.rm=TRUE) + (Num/sum(similarity_vect, na.
    rm=TRUE))
115 names(prediction) = colnames(dataet)
116
117 if (onlyNew == TRUE){
118     unseen <- names(dataet[user, is.na(dataet[user,])])
119     prediction <- prediction[names(prediction) %in% unseen]
120 }
121 TopN <- head(-sort(-prediction), N)
122
123 return(TopN)
124 }

```

Listing 2.1: Pearson Correlation Functions

2.5.2 Cosine Similarity Functions

```

1 #####Functions for user based collaborative filtering using cosine similarity
   #####
2
3 #Cosine function for multiple users
4 UserBasedCF <- function(train_data, test_data, N, NN, onlyNew=TRUE){
5
6     ### similarity ###
7     similarity_matrix <- matrix(, nrow = nrow(test_data), ncol = nrow(train_data)
8
9     ,
10
11     dimnames = list(rownames(test_data), rownames(
12     train_data)))
13
14
15     for (i in rownames(test_data)){
16         for (j in rownames(train_data)){
17             sim <- sum(test_data[i, ]*train_data[j,], na.rm=TRUE)/sqrt(sum(test_data[
18             i, ]^2, na.rm=TRUE) * sum(train_data[j, ]^2, na.rm=TRUE))
19             similarity_matrix[i,j] <- sim
20         }
21     }
22
23     print("similarity calculation done")
24     ### Nearest Neighbors ###
25     similarity_matrix_NN <- similarity_matrix
26
27     for (k in 1:nrow(similarity_matrix_NN)){

```

```

21     crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
22     similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val,
similarity_matrix_NN[k,], NA)
23 }
24
25 print("Nearest Neighbor selection done")
26 ### Prediction ###
27 # Prepare
28 prediction <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data),
dimnames=list(rownames(test_data), colnames(test_data)))
29
30 prediction2 <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data),
dimnames=list(rownames(test_data), colnames(test_data))
31 )
32
33 TopN <- matrix(, nrow=nrow(test_data), ncol=N, dimnames=list(rownames(
test_data)))
34 ### Numerator ###
35 for (u in rownames(test_data)){
36     similarity_vector <- na.omit(similarity_matrix_NN[u, ])
37
38     NN_norm <- train_data[rownames(train_data) %in% names(similarity_vector),]
39
40     CM <- colMeans(train_data, na.rm=TRUE)
41     for (l in 1:ncol(NN_norm)){
42         NN_norm[,l] <- NN_norm[,l] - CM[l]
43     }
44     NN_norm[is.na(NN_norm)] <- 0
45
46     # Numerator
47     Num = similarity_vector %*% NN_norm
48
49     #Prediction
50     prediction[u, ] = mean(test_data[u, ], na.rm=TRUE) + (Num/sum(
similarity_vector, na.rm=TRUE))
51
52
53     if (onlyNew == TRUE){
54         unseen <- names(test_data[u, is.na(test_data[u,])])
55         prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
, ], NA)
56     }else{
57         prediction2[u, ] <- prediction[u, ]

```

```

58     }
59
60     TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])
61
62 }
63
64 print("Prediction done")
65
66 res <- list(prediction, TopN)
67 names(res) <- c('prediction', 'topN')
68
69 return(res)
70 }
71
72 #Cosine function for single user
73 UserBasedCFOneUser <- function(dataet, user, N, NN, onlyNew=TRUE){
74
75     ### similarity ###
76     similarity_vect <- vector(), nrow(dataet))
77     names(similarity_vect) <- rownames(dataet)
78     for (i in rownames(dataet)){
79         if (i != user){
80             sim <- sum(dataet[user, ]*dataet[i, ], na.rm=TRUE)/sqrt(sum(dataet[user,
81             ]^2, na.rm=TRUE) * sum(dataet[i, ]^2, na.rm=TRUE))
82             similarity_vect[i] <- sim
83         }
84     }
85
86     ### Nearest Neighbors ###
87     crit_val <- -sort(-similarity_vect)[NN]
88     similarity_vect <- na.omit(ifelse(similarity_vect >= crit_val,
89     similarity_vect, NA))
89
90     ### Prediction ###
91     # Prepare
92     NN_norm <- dataet[rownames(dataet) %in% names(similarity_vect),]
93     CM <- colMeans(dataet, na.rm=TRUE)
94     for (l in 1:ncol(NN_norm)){
95         NN_norm[,l] <- NN_norm[,l] - CM[l]
96     }
97     NN_norm[is.na(NN_norm)] <- 0

```



```

98 # Numerator
99 Num = similarity_vect %*% NN_norm
100
101 #Prediction
102 prediction = mean(dataet[user, ], na.rm=TRUE) + (Num/sum(similarity_vect, na.
    rm=TRUE))
103 names(prediction) = colnames(dataet)
104
105 if (onlyNew == TRUE){
106     unseen <- names(dataet[user, is.na(dataet[user,])])
107     prediction <- prediction[names(prediction) %in% unseen]
108 }
109 TopN <- head(-sort(-prediction), N)
110
111 return(TopN)
112 } ####Functions for user based collaborative filterinnng using cosine
    simillarity ####
113
114 #Cosine function for multiple users
115 UserBasedCF <- function(train_data, test_data, N, NN, onlyNew=TRUE){
116
117     ### similarity ###
118     similarity_matrix <- matrix(, nrow = nrow(test_data), ncol = nrow(train_data)
    ,
119                                     dimnames = list(rownames(test_data), rownames(
    train_data)))
120
121     for (i in rownames(test_data)){
122         for (j in rownames(train_data)){
123             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))
124             similarity_matrix[i,j] <- sim
125         }
126     }
127     print("similarity calculation done")
128     ### Nearest Neighbors ###
129     similarity_matrix_NN <- similarity_matrix
130
131     for (k in 1:nrow(similarity_matrix_NN)){
132         crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
133         similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val,
    similarity_matrix_NN[k,], NA)

```

```

134 }
135
136 print("Nearest Neighbor selection done")
137 ### Prediction ###
138 # Prepare
139 prediction <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data),
140                                     dimnames=list(rownames(test_data), colnames(test_data)))
141 prediction2 <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data),
142                                     dimnames=list(rownames(test_data), colnames(test_data))
143 )
144
145 TopN <- matrix(, nrow=nrow(test_data), ncol=N, dimnames=list(rownames(
146 test_data)))
147
148 ### Numerator ###
149 for (u in rownames(test_data)){
150     similarity_vector <- na.omit(similarity_matrix_NN[u, ])
151
152     NN_norm <- train_data[rownames(train_data) %in% names(similarity_vector),]
153
154     CM <- colMeans(train_data, na.rm=TRUE)
155     for (l in 1:ncol(NN_norm)){
156         NN_norm[,l] <- NN_norm[,l] - CM[l]
157     }
158     NN_norm[is.na(NN_norm)] <- 0
159
160     # Numerator
161     Num = similarity_vector %*% NN_norm
162
163     #Prediction
164     prediction[u, ] = mean(test_data[u, ], na.rm=TRUE) + (Num/sum(
165 similarity_vector, na.rm=TRUE))
166
167     if (onlyNew == TRUE){
168         unseen <- names(test_data[u, is.na(test_data[u,])])
169         prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
170 , ], NA)
171     }else{
172         prediction2[u, ] <- prediction[u, ]
173     }
174
175     TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])

```

```

172
173 }
174
175 print("Prediction done")
176
177 res <- list(prediction, TopN)
178 names(res) <- c('prediction', 'topN')
179
180 return(res)
181 }
182
183 #Cosine function for single user
184 UserBasedCFOneUser <- function(dataet, user, N, NN, onlyNew=TRUE){
185
186   ### similarity ###
187   similarity_vect <- vector(), nrow(dataet))
188   names(similarity_vect) <- rownames(dataet)
189   for (i in rownames(dataet)){
190     if (i != user){
191       sim <- sum(dataet[user, ]*dataet[i,], na.rm=TRUE)/sqrt(sum(dataet[user,
192         ]^2, na.rm=TRUE) * sum(dataet[i, ]^2, na.rm=TRUE))
193       similarity_vect[i] <- sim
194     }
195   }
196
197   ### Nearest Neighbors ###
198   crit_val <- -sort(-similarity_vect)[NN]
199   similarity_vect <- na.omit(iffelse(similarity_vect >= crit_val,
200     similarity_vect, NA))
201
202   ### Prediction ###
203   # Prepare
204   NN_norm <- dataet[rownames(dataet) %in% names(similarity_vect),]
205   CM <- colMeans(dataet, na.rm=TRUE)
206   for (l in 1:ncol(NN_norm)){
207     NN_norm[,l] <- NN_norm[,l] - CM[l]
208   }
209   NN_norm[is.na(NN_norm)] <- 0
210
211   # Numerator
212   Num = similarity_vect %*% NN_norm

```

```

212 #Prediction
213 prediction = mean(dataet[user, ], na.rm=TRUE) + (Num/sum(similarity_vect, na.
    rm=TRUE))
214 names(prediction) = colnames(dataet)
215
216 if (onlyNew == TRUE){
217     unseen <- names(dataet[user, is.na(dataet[user,])])
218     prediction <- prediction[names(prediction) %in% unseen]
219 }
220 TopN <- head(-sort(-prediction), N)
221
222 return(TopN)
223 }

```

Listing 2.2: Cosine Similarity Functions

2.5.3 Pearson Correlation for Item Based Collaborative Filtering

```

1 #Item based collaborative filtering using pearson correlation
2 meandiff2 <- function(data,i){
3     data[,i] - mean(data[,i],na.rm=TRUE)
4 }
5
6
7 ItemBasedCF_pearson <- function(train_data, test_data, N, NN, onlyNew=TRUE){
8     similarity_matrix = matrix(, ncol=ncol(test_data), nrow=ncol(train_data),
9         dimnames = list(colnames(test_data), colnames(train_data)))
10     for (i in colnames(test_data)){
11         for (j in colnames(train_data)){
12             sim = sum(meandiff2(test_data,i) * meandiff2(train_data,j), na.rm = TRUE)
13             /
14             (sqrt(sum(meandiff2(test_data,i)^2,na.rm=TRUE)) *
15                 sqrt(sum(meandiff2(train_data,j)^2,na.rm=TRUE)))
16             similarity_matrix[i,j] <- sim
17         }
18     }
19     print("Similarity calculation done")
20
21     # Nearest Neighbor
22     similarity_matrix_NN <- similarity_matrix
23
24     for (k in 1:ncol(similarity_matrix_NN)){

```

```

23     crit_val <- -sort(-similarity_matrix_NN[,k])[NN]
24     similarity_matrix_NN[,k] <- ifelse(similarity_matrix_NN[,k] >= crit_val,
similarity_matrix_NN[,k], NA)
25 }
26 similarity_matrix_NN[is.na(similarity_matrix_NN)] <- 0
27
28 train_data[is.na(train_data)] <- 0
29
30 test_data2 <- test_data
31 test_data2[is.na(test_data2)] <- 0
32
33 print("Nearest neighbor selection done")
34
35 ### Prediction ###
36 prediction <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
dimnames=list(rownames(test_data), colnames(test_data)))
37 prediction2 <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
dimnames=list(rownames(test_data), colnames(test_data))
38 )
39 TopN <- matrix(, nrow=nrow(test_data), N, dimnames=list(rownames(test_data)))
40
41
42 for (u in rownames(test_data)){
43     # Numerator
44     Num <- test_data2[u, ] %*% similarity_matrix_NN
45
46     # Denominator
47     Denom <- colSums(similarity_matrix_NN, na.rm=TRUE)
48
49     # Prediction
50     prediction[u, ] <- Num/Denom
51
52     if (onlyNew == TRUE){
53         unseen <- names(test_data[u, is.na(test_data[u,])])
54         prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
, ], NA)
55     }else{
56         prediction2[u, ] <- prediction[u, ]
57     }
58
59     TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])
60
61 }

```

```

62
63 print("Prediction done")
64
65 res <- list(prediction, TopN)
66 names(res) <- c('prediction', 'topN')
67
68 return(res)
69 }

```

Listing 2.3: Pearson Correlation for Item Based Collaborative Filtering

2.5.4 Cosine Similarity for Item Based Collaborative Filtering

```

1  #Item based collaborative filtering using cosine similarity
2
3  ItemBasedCF <- function(train_data, test_data, N, NN, onlyNew=TRUE){
4      # Similarity
5
6      similarity_matrix <- as.matrix(simil(t(train_data), method="cosine"))
7
8      print("Similarity calculation done")
9      # Nearest Neighbor
10     similarity_matrix_NN <- similarity_matrix
11
12     for (k in 1:ncol(similarity_matrix_NN)){
13         crit_val <- -sort(-similarity_matrix_NN[,k])[NN]
14         similarity_matrix_NN[,k] <- ifelse(similarity_matrix_NN[,k] >= crit_val,
15             similarity_matrix_NN[,k], NA)
16     }
17
18     similarity_matrix_NN[is.na(similarity_matrix_NN)] <- 0
19
20     train_data[is.na(train_data)] <- 0
21
22     test_data2 <- test_data
23     test_data2[is.na(test_data2)] <- 0
24
25     print("Nearest neighbor selection done")
26
27     ### Prediction ###
28     prediction <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
29         dimnames=list(rownames(test_data), colnames(test_data)))
30     prediction2 <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data),

```

```

29         dimnames=list(rownames(test_data), colnames(test_data))
    )
30    TopN <- matrix(, nrow=nrow(test_data), N, dimnames=list(rownames(test_data)))
31
32    for (u in rownames(test_data)){
33        # Numerator
34        Num <- test_data2[u, ] %*% similarity_matrix_NN
35
36        # Denominator
37        Denom <- colSums(similarity_matrix_NN, na.rm=TRUE)
38
39        # Prediction
40        prediction[u, ] <- Num/Denom
41
42        if (onlyNew == TRUE){
43            unseen <- names(test_data[u, is.na(test_data[u,])])
44            prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
, ], NA)
45        }else{
46            prediction2[u, ] <- prediction[u, ]
47        }
48
49        TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])
50
51    }
52
53    print("Prediction done")
54
55    res <- list(prediction, TopN)
56    names(res) <- c('prediction', 'topN')
57
58    return(res)
59 }

```

Listing 2.4: Cosine Similarity for Item Based Collaborative Filtering

2.5.5 Demographic Based Filtering/Cluster Based Collaborative Filtering

```

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

```

```

2
3 DemographicBasedF <- function(data, N, centers, iter, onlyNew=TRUE){
4
5     data2 <- data
6
7     # fill with average product rating
8     colmeans <- colMeans(data2, na.rm=TRUE)
9
10    for (j in colnames(data2)){
11        data2[, j] <- ifelse(is.na(data2[,j]), colmeans[j], data2[, j])
12    }
13
14    km <- kmeans(data2, centers=centers, iter.max=iter)
15
16    head(km$cluster)
17    head(km$centers)
18
19
20    # Statistics of the groups
21    tab <- table(km$cluster)
22
23    # Assign users to groups
24    RES <- cbind(data, as.data.frame(km$cluster))
25
26    # Calculate average ratings for every cluster
27    aggregation <- aggregate(RES, list(RES$"km$cluster"), mean, na.rm=T)
28    aggregation <- aggregation[,-1]
29
30    # Make a prediction
31    users <- as.data.frame(RES$"km$cluster")
32    users <- cbind(users, rownames(RES))
33    colnames(users) <- c("km$cluster", 'rn')
34    rec()
35
36    prediction = merge(users, aggregation, by="km$cluster")
37    rownames(prediction) <- prediction$rn
38
39    prediction <- prediction[order(rownames(prediction)), -1:-2]
40
41    prediction2 <- matrix(, nrow=nrow(prediction), ncol(ncol(prediction),
42                                dimnames=list(rownames(prediction), colnames(prediction
    )))

```



```

43 colnames(prediction2) <- colnames(prediction)
44 rownames(prediction2) <- rownames(prediction)
45
46 for (u in rownames(prediction)){
47   if (onlyNew == TRUE){
48     unseen <- names(data[u, is.na(data[u,])])
49
50     prediction2[u, ] <- as.numeric(t(ifelse(colnames(prediction) %in% unseen,
51 prediction[u, ], as.numeric(NA))))
52   }else{
53     prediction2[u, ] <- prediction[u, ]
54   }
55 }
56
57 # TopN
58 TopN <- t(apply(prediction, 1, function(x) names(head(sort(x, decreasing=TRUE
59 ), 5))))
60
61 print("Prediction done")
62
63 res <- list(prediction, TopN)
64 names(res) <- c('prediction', 'topN')
65
66 return(res)
67 }

```

Listing 2.5: Cluster Based Collaborative Filtering

2.5.6 Content Based Filtering

```

1
2 #Content Based Filtering
3
4 ContentBased <- function(product_data, test_data, N, NN, onlyNew=TRUE){
5
6   # Similarity calculation (copied from user-based collaborative filtering)
7   similarity_matrix <- as.matrix(simil(product_data, method="cosine"))
8
9   print("Similarity calculation done")
10
11   # Set Nearest neighbors (copied from user-based collaborative filtering)
12   similarity_matrix_NN <- similarity_matrix

```

```

13
14 for (k in 1:nrow(similarity_matrix_NN)){
15     crit_val <- -sort(-similarity_matrix_NN[k,])[NN]
16     similarity_matrix_NN[k,] <- ifelse(similarity_matrix_NN[k,] >= crit_val,
17     similarity_matrix_NN[k,], 0)
18 }
19
20 similarity_matrix_NN[is.na(similarity_matrix_NN)] <- 0
21 test_data2 <- test_data
22 test_data2[is.na(test_data2)] <- 0
23
24 print("Nearest neighbor selection done")
25
26 ### Prediction (copied from item based collaborative filtering) ###
27 prediction <- matrix(, nrow=nrow(test_data), ncol=ncol(test_data),
28 dimnames=list(rownames(test_data), colnames(test_data)))
29 prediction2 <- matrix(, nrow=nrow(test_data), ncol(ncol(test_data),
30 dimnames=list(rownames(test_data), colnames(test_data))
31 )
32 TopN <- matrix(, nrow=nrow(test_data), N, dimnames=list(rownames(test_data)))
33
34 for (u in rownames(test_data)){
35     # Numerator
36     Num <- test_data2[u, ] %*% similarity_matrix_NN
37
38     # Denominator
39     Denom <- colSums(similarity_matrix_NN, na.rm=TRUE)
40
41     # Prediction
42     prediction[u, ] <- Num/Denom
43
44     if (onlyNew == TRUE){
45         unseen <- names(test_data[u, is.na(test_data[u,])])
46         prediction2[u, ] <- ifelse(colnames(prediction) %in% unseen, prediction[u
47 , ], NA)
48     }else{
49         prediction2[u, ] <- prediction[u, ]
50     }
51
52     TopN[u, ] <- names(-sort(-prediction2[u, ])[1:N])
53 }

```

```

52
53     print("Prediction done")
54
55     res <- list(prediction, TopN)
56     names(res) <- c('prediction', 'topN')
57
58     return(res)
59 }

```

Listing 2.6: Content Based Filtering

2.5.7 Predictive Accuracy

```

1
2     #####Evaluation Metrics
3
4     ### Prediction Accuracy ###
5     #####
6
7     RSME <- function(prediction, real){
8
9         if (nrow(prediction) == nrow(real) & ncol(prediction) == ncol(real)){
10             RSME = sqrt( sum( (prediction - real)^2 , na.rm = TRUE ) / (nrow(prediction)
11                             * ncol(prediction)) )
12             return(RSME)
13         }else{
14             return("Dimension of prediction are not equal to dimension of real")
15         }
16     }
17
18     MAE <- function(prediction, real){
19
20         if (nrow(prediction) == nrow(real) & ncol(prediction) == ncol(real)){
21             RSME = sum( Mod(prediction - real) , na.rm = TRUE ) / (nrow(prediction) *
22                             ncol(prediction))
23             return(RSME)
24         }else{
25             return("Dimension of prediction are not equal to dimension of real")
26         }
27     }

```

Listing 2.7: Predictive Accuracy

2.5.8 Classification Accuracy

```
1
2
3 #####Classification accuracy#####
4
5 Classification <- function(prediction, real, threshold=NA, TopN=NA){
6   if (nrow(prediction) == nrow(real) & ncol(prediction) == ncol(real)){
7     # Threshold #
8     if (!is.na(threshold)){
9       TP = sum(iffelse(prediction >= threshold & real >= threshold, 1, 0), na.rm
=T)
10      FP = sum(iffelse(prediction >= threshold & real < threshold, 1, 0), na.rm=
T)
11      FN = sum(iffelse(prediction < threshold & real >= threshold, 1, 0), na.rm=
T)
12      Recall = TP/(TP+FN)
13      Precision = TP/(TP+FP)
14      F1 = 2 * ((Precision * Recall) / (Precision + Recall))
15      Class_Thres = list(Recall, Precision, F1)
16      names(Class_Thres) = c("Recall", "Precision", "F1")
17    }
18    if (!is.na(TopN)){
19      TP = vector(, length = nrow(prediction))
20      FP = vector(, length = nrow(prediction))
21      FN = vector(, length = nrow(prediction))
22
23      for (u in nrow(prediction)){
24        threshold_pred = -sort(-prediction[u, ])[TopN]
25        threshold_real = -sort(-real[u, ])[TopN]
26        TP[u] = sum(iffelse(prediction[u, ] >= threshold_pred & real[u, ] >=
threshold_real, 1, 0), na.rm=T)
27        FP[u] = sum(iffelse(prediction[u, ] >= threshold_pred & real[u, ] <
threshold_real, 1, 0), na.rm=T)
28        FN[u] = sum(iffelse(prediction[u, ] < threshold_pred & real[u, ] >=
threshold_real, 1, 0), na.rm=T)
29      }
30      TP = sum(TP[u])
31      FP = sum(FP[u])
32      FN = sum(FN[])
33      Recall = TP/(TP+FN)
34      Precision = TP/(TP+FP)
```

```

35     F1 = 2 * ((Precision * Recall) / (Precision + Recall))
36     Class_TopN = list(Recall, Precision, F1)
37     names(Class_TopN) = c("Recall", "Precision", "F1")
38 }
39
40 if (!is.na(threshold) & !is.na(TopN)){
41     Class = list(Class_Thres, Class_TopN)
42     names(Class) = c("Threshold", "TopN")
43 }else if (!is.na(threshold) & is.na(TopN)) {
44     Class = Class_Thres
45 }else if (is.na(threshold) & !is.na(TopN)) {
46     Class = Class_TopN
47 }else{
48     Class = "You have to specify the 'Threshold' or 'TopN' parameter!"
49 }
50 return(Class)
51 }else{
52     return("Dimension of prediction are not equal to dimension of real")
53 }
54 }

```

Listing 2.8: Classification Accuracy

2.5.9 Ranking Accuracy

```

1
2
3 ##### Ranking accuracy#####
4
5 AUC <- function(real, prediction, threshold){
6
7     pred <- ifelse(prediction >= threshold, 1, 0)
8     real <- ifelse(real >= threshold, 1, 0)
9
10    real[is.na(real)] <- 0
11    pred[is.na(pred)] <- 0
12
13    ROC <- roc(factor(prediction), factor(real))
14
15    plot(ROC)
16
17    AUC <- auc(ROC)

```

```

18   return(AUC)
19 }
20
21
22 #produced errors when computing, excluded from report
23 NDCG <- function(real, prediction, TopN){
24   for (u in rownames(real)){
25
26     # compute ranking
27     rank <- sort(-rank(prediction[u,]))[1:TopN]
28
29
30     # Create NDCG vector
31     if ( u == rownames(real)[1]){
32       NDCG_vect <- Evaluation.NDCG(rank, real[u, names(rank)])
33     }else{
34       NDCG_vect <- rbind(NDCG_vect, Evaluation.NDCG(rank, real[u, names(rank)]))
35     }
36   }
37
38   # Compute avarege NDCG
39   NDCG_vect[is.na(NDCG_vect)] <- 0
40   NDCG <- colMeans(NDCG_vect, na.rm=T)
41   names(NDCG) <- "NDCG"
42   return(NDCG)
43 }
44
45 #taken from https://github.com/mlr-org/mlr/
46 library(mlr)
47
48
49 #taken from https://github.com/mhahsler/recommenderlab
50 library(recommenderlab)
51
52 #taken from https://github.com/tidyverse/tidyr
53 library(tidyr)
54
55 #taken from https://github.com/tidyverse/dplyr
56 library(dplyr)

```

Listing 2.9: Ranking Accuracy

2.5.10 Data Manipulation For Collaborative Based Filtering

```
1
2 #####Collaborative Filtering
3 #####
4 #Reading the data
5 user_artists = read.table("user_artists.dat",header = TRUE,sep='\t')
6 mean(user_artists$weight)
7 sd(user_artists$weight)
8 {
9
10 #Checking the normalization and skewness of data
11
12 normaliseddata = dnorm(user_artists$weight,mean=745.2439,sd=3751.322)
13
14 hist(normaliseddata,prob = TRUE)
15
16 max(user_artists$weight)
17 min(user_artists$weight)
18
19
20 #Calculating a new column with the log values for optimization
21 user_artists$log_weight = log(user_artists$weight)
22
23 #Using linear model to find the coefficient
24 model = lm(log_weight ~ weight, user_artists)
25
26 summary(model)
27
28 #Adding the coefficient value to the log value to get a uniformed weight
   distribution
29 user_artists$log_weight = log(user_artists$weight) + 5.389e+00
30
31 #plotting histogram
32 hist(user_artists$log_weight)
33 str(user_artists)
34 }
35 #deleting the weight column
36 user_artists_updated = user_artists
37 user_artists_updated$weight = NULL
38
```

```

39 #filtering the users for normalization and weights > 10 and number of users > 5
40
41 artists_subset = user_artists_updated %>% filter(log_weight > 10) %>% group_by
    (artistID) %>% summarise(Totalusers = n_distinct(userID)) %>% filter(
    Totalusers > 5)
42
43
44 #sub-setting the data with the values from the filtered users
45 user_artists_updated1 = subset(user_artists_updated, artistID %in%
    artists_subset$artistID)
46
47 length(unique(user_artists_updated1$artistID))
48 length(unique(user_artists_updated1$userID))
49
50 #Spreading the data
51 user_artists_transform = spread(user_artists_updated1,artistID,log_weight)
52
53
54 #Converting the row names to indices
55 row.names(user_artists_transform) <- user_artists_transform$userID
56
57
58 #min(transpose_matrix)
59 user_artists_transform[,1] = NULL
60
61 row.names(user_artists_transform)
62 names(user_artists_transform)
63
64 #converting the data into matrix
65 user_artists_transform_matrix = as(user_artists_transform,"matrix")
66
67
68 # Number of users and artists
69 nrow(user_artists_transform_matrix)
70 ncol(user_artists_transform_matrix)
71
72 # Min, max and average rating for the artists
73 min(user_artists_transform_matrix, na.rm=TRUE)
74 max(user_artists_transform_matrix, na.rm=TRUE)
75 mean(user_artists_transform_matrix, na.rm=TRUE)
76
77

```



```

78 hist(user_artists_transform_matrix)
79 t = table(is.na(user_artists_transform_matrix))
80 t
81
82 # sparsity
83 t[2]/(t[1]+t[2])
84 #98.18503% sparse data
85
86
87 #Split the data into test and train
88
89 train_id = sample(1:nrow(user_artists_transform_matrix), 0.7 * nrow(
    user_artists_transform_matrix))
90 test_id <- setdiff(1:nrow(user_artists_transform_matrix), train_id)
91
92 train_data = user_artists_transform_matrix[train_id,]
93 test_data = user_artists_transform_matrix[test_id,]
94
95
96 #Cosine correlation for single user
97
98 UserBasedCFOneUser(user_artists_transform_matrix, '6', 3, 10, onlyNew = TRUE)
99
100 #Cosine correlation for multiple user
101
102 prediction_cosine = UserBasedCF(train_data, test_data, 3, 15, onlyNew = TRUE)
103 prediction_cosine_preddata = prediction_cosine$prediction
104 prediction_cosine_topN = prediction_cosine$topN
105
106 write.csv(prediction_cosine_preddata, file = "prediction_cosine_preddata.csv")
107 write.csv(prediction_cosine_topN, file = "prediction_cosine_topN.csv")
108
109 #Pearson correlation for single user
110 UserBasedCFOneUser_Pearson(user_artists_transform_matrix, '6', 3, 10, onlyNew =
    TRUE)
111
112 # taken from https://github.com/HenrikBengtsson/parallelly
113 library(doParallel)
114 k=detectCores()
115 cl <- makeCluster(k-1)
116 #Pearson correlation for multiple user
117 prediction_pearson = UserBasedCF_pearson(train_data, test_data, 3, 15, onlyNew =

```

```

TRUE)
118 prediction_pearson_preddata <- as.data.frame(prediction_pearson$prediction)
119
120 TopN_pearson <- as.data.frame(prediction_pearson$topN)
121
122 write.csv(prediction_pearson_preddata, file = "prediction_pearson_preddata.csv")
123 write.csv(TopN_pearson, file = "TopN_pearson.csv")
124
125 #taken from https://github.com/mhahsler/recommenderlab
126 #Using "recommender lab" library
127
128 train_data = user_artists_transform_matrix[train_id,]
129 test_data = user_artists_transform_matrix[test_id,]
130 train = as(train_data, "realRatingMatrix")
131 test = as(test_data, "realRatingMatrix")
132
133 recommenderRegistry$get_entry("UBCF", dataType="realRatingMatrix")
134
135 recom_Userbased <- Recommender(train, method = "UBCF")
136 predd_recomm <- predict(recom_Userbased, newdata=test@data@p[66], n=10)
137
138
139 #####Item based collaborative filtering#####
140 k=detectCores()
141 cl <- makeCluster(k-1)
142 #Item based collaborative filtering using pearson
143 prediction_item_pearson = ItemBasedCF_pearson(train_data, test_data, 3, 10,
144 onlyNew=TRUE)
145 prediction_item_pearson_data = prediction_item_pearson$prediction
146 prediction_item_pearson_TopN = prediction_item_pearson$top
147 prediction_item_pearson_TopN
148 write.csv(prediction_item_pearson_data, file = "prediction_item_pearson_data.csv"
149 )
150 write.csv(prediction_item_pearson_TopN, file="prediction_item_pearson_topN.csv")
151
152 #Item based collaborative filtering using cosine
153 prediction_item_cosine = ItemBasedCF(train_data, test_data, 3, 15, onlyNew=TRUE)
154 prediction_item_cosine_data = prediction_item_cosine$prediction
155 prediction_item_cosine_TopN = prediction_item_cosine$topN
156 prediction_item_pearson_TopN

```

```

157 write.csv(prediction_item_cosine_data, file="prediction_item_cosine_data.csv")
158 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

```

1  #####Evaluation Metrics#####
2
3  #####Userbased#####
4  ##Prediction accuracy with the results from pearson
5  RSME(prediction_pearson$prediction, test_data)
6  MAE(prediction_pearson$prediction, test_data)
7
8  ##Prediction accuracy with the results from cosine
9  RSME(prediction_cosine$prediction, test_data)
10 MAE(prediction_cosine$prediction, test_data)
11
12 #####itembased#####
13 ##Prediction accuracy with the results from pearson
14 RSME(prediction_item_pearson$prediction, test_data)
15 MAE(prediction_item_pearson$prediction, test_data)
16
17 ##Prediction accuracy with the results from cosine
18 RSME(prediction_item_cosine$prediction, test_data)
19 MAE(prediction_item_cosine$prediction, test_data)
20
21
22 ###userbased classification accuracy
23 ##Classification accuracy with pearson
24 Classification(prediction_pearson$prediction, test_data, threshold=5, TopN=10)
25
26 ##Classification accuracy with cosine
27 Classification(prediction_cosine$prediction, test_data, threshold=5, TopN=10)
28
29 ###itembased classification accuracy
30 ##Classification accuracy with pearson
31 Classification(prediction_item_pearson$prediction, test_data, threshold=5, TopN
    =20)
32
33 ##Classification accuracy with cosine

```

```

34 Classification(prediction_item_cosine$prediction, test_data, threshold=6, TopN
    =10)
35
36 #####Ranking accuracy
37 #####Ranking accuracy user based
38 #Using pearson function
39
40 #taken from https://rdr.io/cran/DescTools/src/R/StatsAndCIs.r
41 library(AUC)
42 #used for graphs
43
44 AUC(test_data, prediction_pearson$prediction, 5)
45
46 #Using cosine function
47 NDCG(test_data, prediction_cosine$prediction, 5)
48 AUC(test_data, prediction_cosine$prediction, 5)
49
50 #####Ranking accuracy item based
51 #Using pearson function
52 NDCG(test_data, prediction_item_pearson$prediction, 5)
53 AUC(test_data, prediction_item_pearson$prediction, 5)
54
55 #Using cosine function
56 NDCG(test_data, prediction_item_cosine$prediction, 5)
57 AUC(test_data, prediction_item_cosine$prediction, 10)
58
59 #
    #####
60 # DEMOGRAPHIC BASED FILTERING
61 #
    #####
62 #getting user features. converting to matrix and feeding them into the
    demographic filter function
63 rec=function(){}
64 Prediction_cluster_matrix = DemographicBasedF(user_artists_transform_matrix, 3,
    150, 75, onlyNew=TRUE)
65 Prediction_cluster_prediction_matrix = as.data.frame(
    Prediction_cluster_matrix$prediction)
66 Prediction_cluster_topN_matrix = as.data.frame(Prediction_cluster_matrix$topN)
67 Prediction_cluster_topN_matrix

```

```

68
69 Prediction_cluster_prediction_matrixtest = subset(
      Prediction_cluster_prediction_matrix, row.names(
      Prediction_cluster_prediction_matrix) %in% row.names(test_data))
70
71 dim(test_data)
72 row.names(test_data)
73 row.names(Prediction_cluster_prediction_matrix)
74
75 ### Results for Cluster Based Filtering ###
76
77 #Evaluation metrics- Cluster Based
78
79 #Prediction accuracy2
80 RSME(Prediction_cluster_prediction_matrixtest, test_data)
81
82
83 #Classification accuracy
84 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

```

1  #
      #####
2  # CONTENT BASED FILTERING
3  #
      #####
4
5  #https://github.com/tidyverse/readr
6  library(readr)
7
8  #taken from https://github.com/tidyverse/dplyr
9  library(dplyr)
10
11 # taken from https://github.com/tidyverse
12 library(tidyverse)

```

```

13
14 #taken from https://github.com/tidyverse/tidyr
15 library(tidyr)
16
17 # import user_taggedartists file
18 user_taggedartists <- read.table("user_taggedartists.dat", header=TRUE) %>%
    select(userID, artistID, tagID)
19
20 # import tags file
21 tags <- read.delim("user_taggedartists.dat",header=TRUE)
22
23
24
25
26 # import arts file
27 art <- read_delim("artists.dat", delim = "\t") %>% select(id, name)
28 art$name <- iconv(art$name, from = "UTF-8", to = "ASCII//TRANSLIT")
29
30
31 # extract count of tags for each group of artists and tagID
32
33 tags_counts <- arrange(summarise(group_by(user_taggedartists, tagID),
34                                     TotalUsers = length(unique(userID)) ), desc(
    TotalUsers) )
35
36 #length(unique(user_taggedartists$tagID))
37 tag_top200 <- tags_counts
38
39 # Take top 200 tags
40 tag_top200 <- arrange(tag_top200, tagID)
41
42 # subset tags which is having top 200
43 tag_top200$Names <- subset(tags, tagID %in% tag_top200$tagID)$tagValue
44
45 # Selecting the Top 200 Tags based on Maximum number of users
46
47 tag_top200 <- arrange(tag_top200, desc(TotalUsers))
48
49
50 toptags <- subset(user_taggedartists, tagID %in% tag_top200$tagID)
51
52 #Selecting only those Artists which are used by Collaborative based Filerling

```

```

Recommendation Systems
53 teststart <- subset(user_taggedartists, artistID %in% user_artists$artistID)
54 teststart1 <- subset(teststart, artistID %in% user_artists_updated1$artistID)
55
56 # Deleting couple of records with issues.
57 user_artists_updated2 <- user_artists_updated1[!user_artists_updated1$artistID
  == "5533",]
58 user_artists_updated3 <- user_artists_updated2[!user_artists_updated2$artistID
  == "4941" ,]
59
60
61
62
63
64
65 #tag_top200 <- tags_counts[1:200,]
66
67 summarized_tag <- summarise(group_by(toptags, artistID, tagID ), Count = length
  (tagID) )
68
69 summarized_tag <- subset(summarized_tag, artistID %in%
  user_artists_updated2$artistID)
70
71
72 # Creating the base Matrix
73
74 matrix <- spread(summarized_tag, tagID, Count)
75
76 row.names(matrix) <- matrix$artistID
77
78 matrix[,][is.na(matrix[,])] <- 0
79
80 ag_artistID <- as.vector(matrix$artistID)
81 ag_mat <- as.matrix(matrix[,2:ncol(matrix)])
82 rm(matrix)
83
84 ntags <- length(as.vector(ag_mat))
85 ntags
86
87 sum(!is.na(as.vector(ag_mat)) ) / ntags
88 1 - sum(!is.na(as.vector(ag_mat)) ) / ntags
89

```

```

90 # Creating the Final Base Matrix for Content Based RS
91
92 fin_matrix <- ag_mat
93
94 fin_matrix[,][is.na(fin_matrix[,])] <- 0
95 fin_matrix[,][fin_matrix[,] > 0] <- 1
96
97
98 nrow(fin_matrix)
99 ncol(fin_matrix)
100
101 #####Updating original user based matrix
102 user_artists_updated2 <- user_artists_updated1[!user_artists_updated1$artistID
  == "5533",]
103 user_artists_updated3 <- user_artists_updated2[!user_artists_updated2$artistID
  == "4941" ,]
104
105
106
107 user_artists_new =spread(user_artists_updated3,artistID,log_weight)
108
109 #Converting the row names to indices
110 row.names(user_artists_new) <- user_artists_new$userID
111
112
113 #min(transpose_matrix)
114 user_artists_new[,1] = NULL
115
116 #row.names(user_artists_transform)
117 #names(user_artists_transform)
118
119 #converting the data into matrix
120 user_artists_transform_new = as(user_artists_new,"matrix")
121
122 write.csv(user_artists_transform_new , file = "user_artists_transform_new.csv")
123 write.csv(fin_matrix,file="content_matrix.csv")
124 set.seed(2)
125 train_rows = sample(1:nrow(user_artists_transform_new), 0.7*nrow(
  user_artists_transform_new))
126
127 train_content <- as(user_artists_transform_new, "matrix")[train_rows,]
128 test_content <- as(user_artists_transform_new, "matrix")[-train_rows,]

```



```

129
130 CB_updated <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=T)
131 CB_updated_pred = CB_updated
132
133 CB_updated$prediction
134 CB_updated$topN

```

Listing 2.12: Data Manipulation For Content Based Filtering

2.5.13 Quantitative Evaluations between systems

```

1
2
3 #####
4 ### Quantitative Evauation & comparison with item-based Collaborative Filtering
5     ###
6 #####
7 # Load Models
8
9 # Split train - Test
10
11
12
13 # Score Models
14
15 ptm <- proc.time()
16 CB <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=T)
17 Time <- (proc.time() - ptm)
18 Time
19
20 ### Results for Content-Based Filtering
21
22
23 ### Prediction Accuracy ###
24 #####
25
26 # RSME Content-based
27 RSME(CB$prediction, test_content)
28
29 MAE(CB$prediction, test_content)
30

```

```

31  ### Classification Accuracy ###
32  #####
33
34  # Recall/precision Content-based
35  Classification(CB$prediction, test_content, threshold=3)

```

Listing 2.13: Quantitative Evaluations between systems

2.5.14 Qualitative Results

```

1  #
   #####
2  #
   #####
3  ##### Qualitative Results :
4
5  #taken from https://github.com/mhahsler/recommenderlab
6  library(recommenderlab)
7  art_sim <- similarity(as(fin_matrix, "binaryRatingMatrix"), method = "cosine",
8                      which = "users")
9
10 # convert to an R matrix
11 art_sim <- as(art_sim, "matrix")
12
13 # round to 3 digit precision
14 art_sim[][] <- round(art_sim[],[],3)
15
16 # # name rows and columns according to artistID for easy retrieval
17 colnames(art_sim) <- ag_artistID
18 rownames(art_sim) <- ag_artistID
19
20
21 #####
22 # set number of similar artists to recommend
23 n_recommended <- 5
24
25 # randomly select a user
26 artist <- sample(ag_artistID, 1)
27
28 # get name of artist from artist list

```

```

29 a_name <- art[art$id == artist,]$name
30
31 # fetch their recommendations: this returns a named vector sorted by similarity
32 # the names of the items are the artist IDs
33 arecs <- sort(art_sim[as.character(artist),], decreasing = TRUE)[1:
    n_recommended]
34
35 # extract the artist IDs and convert to numeric
36 arecs_IDs <- as.numeric(names(arecs))
37
38 # create list of artist names from artist ID's in list
39 arec_names <- art[art$id %in% arecs_IDs,]$name
40
41 # create a heading for the list of similar artists
42 table_head <- sprintf("Artists Similar to %s", a_name)
43
44 # display the list of similar artists
45
46 #taken from https://github.com/yihui/knitr
47 library(knitr)
48 kable(arec_names, col.names = table_head)
49
50 #
    #####
51 # Generate a Top N Artist List by Genre
52
53 set.seed(42)
54
55 # set rownames = artistID's for easy retrieval - DON'T NEED THIS LINE OF CODE
    IN SHINY
56 rownames(ag_mat) <- ag_artistID
57
58 # extract the genre tagIDs from matrix and convert to numeric
59 tagIDs <- as.numeric(colnames(ag_mat))
60
61 # set number of artists to recommend
62 n_recommended <- 5
63
64 # randomly select a genre
65 tagID <- sample(tagIDs, 1)
66

```

```

67 # get name of genre from tagID list
68 g_name <- tags[tags$tagID == tagID,]$tagValue
69
70 # fetch the top N artists:
71 # the names of the items are the artist IDs
72 g_arecs <- sort(ag_mat[,as.character(tagID)], decreasing = TRUE)[1:
    n_recommended]
73
74 # extract the artist IDs and convert to numeric
75 g_arecs_IDs <- as.numeric(names(g_arecs))
76
77 # create list of artist names from artist ID's in list
78 g_arec_names <- art[art$id %in% g_arecs_IDs,]$name
79
80 # create a heading for the list of similar artists
81 table_head <- sprintf("Top Artists in %s genre:", g_name)
82
83 # display the list of similar artists
84 kable(g_arec_names, col.names = table_head)

```

Listing 2.14: Qualitative Results

2.5.15 Hybrid Recommendation System Results

```

1
2 #
    #####
3 #
    #####
4
5 ####Hybrid Recommendation Systems:cluster based and content based
6
7 #Load the data from content and cluster based
8
9 # Split train - Test data from cluster based
10 set.seed(2)
11 train_rows = sample(1:nrow(user_artists_transform_new), 0.7*nrow(
    user_artists_transform_new))
12
13 train_content <- as(user_artists_transform_new, "matrix")[train_rows,]

```

```

14 test_content <- as(user_artists_transform_new, "matrix")[-train_rows,]
15
16 ### Compute individual models ###
17 #Content based
18 Contentbased <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=F)
19 Contentbased$topN
20 content_pred = Contentbased$prediction
21
22 #Cluster based
23 clusterbased <- DemographicBasedF(user_artists_transform_new, 3, 150, 75,
    onlyNew=T)
24 clusterbased$topN
25 cluster_prediction = clusterbased$prediction
26
27 Cluster_pred = subset(cluster_prediction, row.names(cluster_prediction) %in%
    row.names(test_content))
28 Cluster_pred1 = as(Cluster_pred, "matrix")
29
30 ### Transform results to lists (to be able to use the rowMeans function) ###
31 content_list <- as.list(content_pred)
32 cluster_list <- as.list(Cluster_pred1)
33
34 #####
35 ### Compute Mean ###
36 #####
37 hybrid <- rowMeans(cbind(as.numeric(content_list), as.numeric(cluster_list)),
    na.rm=T)
38
39 ### Transform list back to matrix with correct number of dimensions ###
40 Hybrid_prediction <- matrix(hybrid, nrow=nrow(test_content), ncol=ncol(
    test_content))
41 rownames(Hybrid_prediction) <- rownames(test_content)
42 colnames(Hybrid_prediction) <- colnames(test_content)
43
44 ### Evaluate the Metrics for Prediction accuracy and Classification accuracy
    ###
45
46
47 ### Results for content based and cluster based collaborative filtering ###
48
49 # Prediction accuracy
50 RSME(Hybrid_prediction, test_content)

```

```

51 MAE(Hybrid_prediction, test_content)
52
53 # Classification
54 Classification(Hybrid_prediction, test_content, threshold=6)
55
56
57 #####
58 ### Hybrid RecSys ###
59 #####
60
61 #taken from https://github.com/mhahsler/recommenderlab
62 library("recommenderlab")
63
64
65 ### Split train - Test ###
66 set.seed(2)
67
68
69 train_rows = sample(1:nrow(user_artists_transform_matrix), 0.7*nrow(
    user_artists_transform_matrix))
70
71 train <- as(user_artists_transform_matrix, "matrix")[train_rows,]
72 test <- as(user_artists_transform_matrix, "matrix")[~train_rows,]
73
74
75 ### Compute individual models ###
76 set.seed(2)
77 train_rows = sample(1:nrow(user_artists_transform_new), 0.7*nrow(
    user_artists_transform_new))
78
79 train_content <- as(user_artists_transform_new, "matrix")[train_rows,]
80 test_content <- as(user_artists_transform_new, "matrix")[~train_rows,]
81
82 CBTFIDF <- ContentBased(fin_matrix, test_content, 3, 10, onlyNew=F)
83 IB <- UserBasedCF(train, test, 3, 10, onlyNew=F)
84
85 ### Transform results to lists (to be able to use the rowMeans function) ###
86 CBTFIDF_list <- as.list(CBTFIDF$prediction)
87 IB_list <- as.list(IB$prediction)
88
89 #####
90 ### Compute Mean ###

```

```

91 #####
92 hybrid <- rowMeans(cbind(as.numeric(CBTFIDF_list), as.numeric(IB_list)), na.rm=
    T)
93
94 ### Transform list back to matrix with correct number of dimensions ###
95 Hybrid_prediction <- matrix(hybrid, nrow=nrow(test), ncol=ncol(test))
96 rownames(Hybrid_prediction) <- rownames(test)
97 colnames(Hybrid_prediction) <- colnames(test)
98
99 ### Evaluate ###
100
101 ### Results for content based and user based collaborative filtering ###
102
103
104 # MAE
105 MAE(Hybrid_prediction, test)
106
107 # RMSE
108 RSME(Hybrid_prediction, test)
109
110
111 # Classification
112
113 Classification(Hybrid_prediction, test, threshold=5)

```

Listing 2.15: Hybrid Recommendation System Results

Chapter 3

Appendix C

3.1 Graphs and Charts

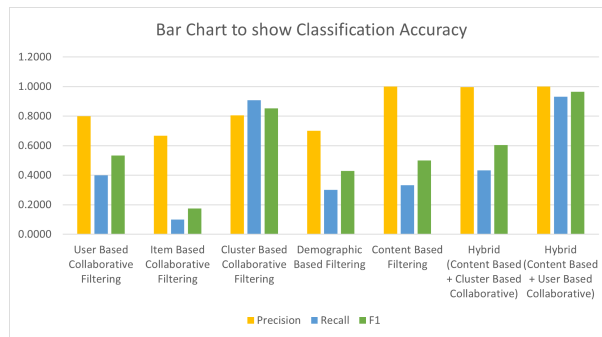


Figure 3.1: This bar chart shows how the the classification metrics compare with one another between systems

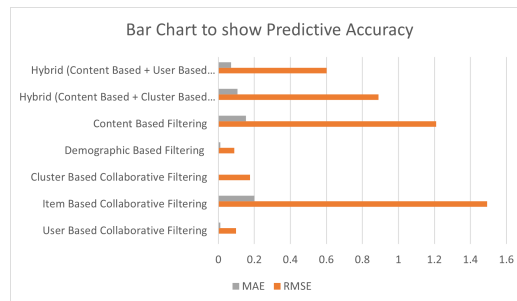


Figure 3.2: This bar chart shows how the the predictive accuracy metrics compare with one another between systems

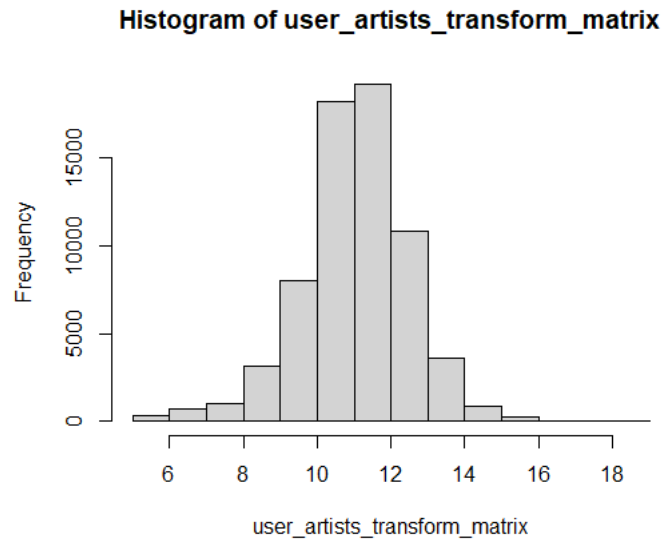


Figure 3.3: This histogram shows us how frequently an artist is chosen by users

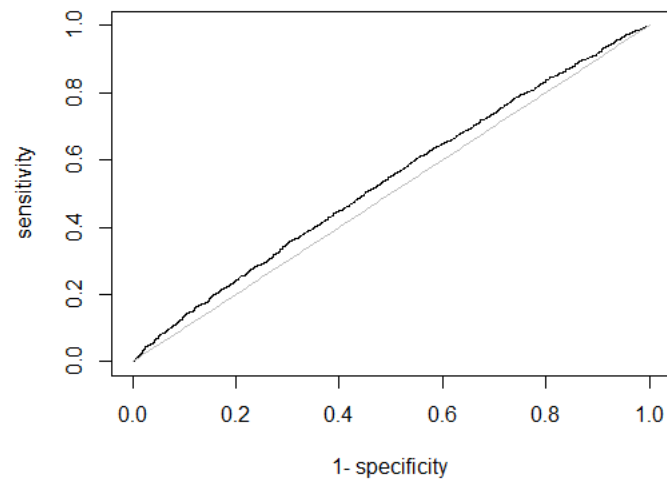


Figure 3.4: This is a graph to show the ROC Curve of how content based filtering changes in sensitivity with specificity. This is shown as the model is given more data

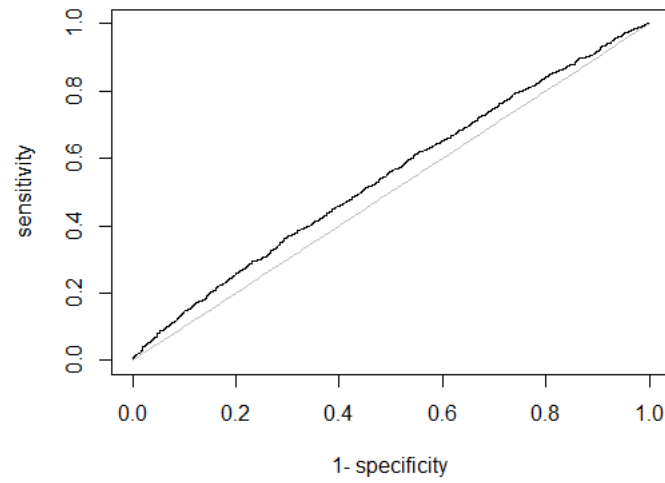


Figure 3.5: This is a graph to display the ROC Curve of how collaborative based filtering changes in sensitivity with specificity. This is shown as the model is given more data

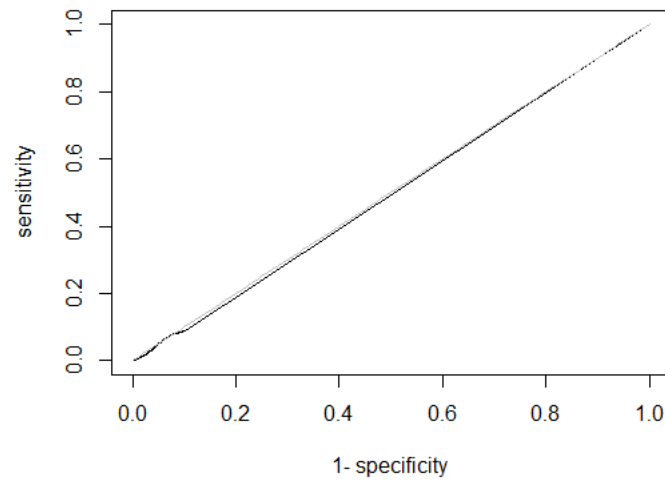


Figure 3.6: This is a graph to display the ROC Curve of how the hybrid system of content based filtering and collaborative based filtering changes in sensitivity with specificity. This is shown as the model is given more data

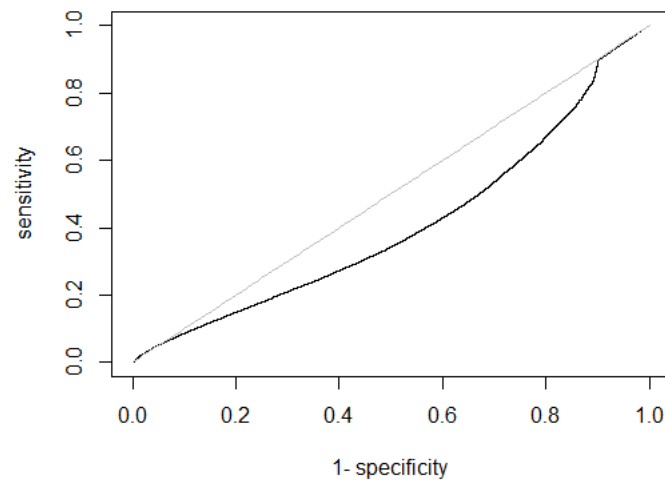


Figure 3.7: This is a graph to display the ROC Curve of how demographic based filtering changes in sensitivity with specificity. This is shown as the model is given more data