MUSIC RECOMMENDATION SYSTEM — SPOTIFY CASE STUDY

Social Media - Final Project
Group 4

Francisco Miguel
Louane Angelloz-Nicoud
Yara Hossam
Jasmine Sams

INTRODUCTION & CONTEXT

Spotify enhances user experience with personalized music recommendations.

This project develops an intelligent system to suggest artists based on listening habits, inspired by **Netflix** and **Spotify**.

PROJECT OBJECTIVE

- Predict users' music preferences
- Generate personalized artist recommendations.
- Implementation of popularity-based, similarity-based, and hybrid systems

DATA OVERVIEW

1. 50000 interactions between 1889 users and 12054 artists.

2. Users listen to artists, and our system predicts what they might like next.

THE DATA

user_artists

	userID	artistID	weight
1	1879	188	256
2	792	230	4033
3	1900	3287	183
4	1791	318	507
5	1699	299	110
6	1926	10214	418
7	1277	154	7294
8	711	1360	37
9	32	220	327
10	255	1789	128
11	1754	812	277
12	1697	5353	217
13	1768	1041	118
14	605	1089	370

artists

	id	name	url	pictureURL	charid
1	8531	Argharus	http://www.last.fm/music/Argharus	http://userserve-ak.last.fm/serve/252/72690.jpg	I8531
2	11858	Do or Die	http://www.last.fm/music/Do+or+Die	http://userserve-ak.last.fm/serve/252/39456477.jpg	I11858
3	18642	Umek	http://www.last.fm/music/Umek	http://userserve-ak.last.fm/serve/252/2165619.jpg	I18642
4	2622	Pokémon	http://www.last.fm/music/Pok%C3%A9mon	http://userserve-ak.last.fm/serve/252/62176913.jpg	I2622
5	10527	Parque En El Espacio	http://www.last.fm/music/Parque+En+El+Espacio	http://userserve-ak.last.fm/serve/252/17635943.jpg	I10527
6	6475	Leões de Israel	http://www.last.fm/music/Le%C3%B5es+de+Israel	http://userserve-ak.last.fm/serve/252/14431969.jpg	I6475
7	9764	Autopilot Off	http://www.last.fm/music/Autopilot+Off	http://userserve-ak.last.fm/serve/252/132675.jpg	I9764
8	6294	Saves the Day	http://www.last.fm/music/Saves+the+Day	http://userserve-ak.last.fm/serve/252/34926095.jpg	I6294
9	5966	Circa Survive	http://www.last.fm/music/Circa+Survive	http://userserve-ak.last.fm/serve/252/45138161.jpg	I5966
10	17588	Chris Bondy	http://www.last.fm/music/Chris+Bondy		I17588
11	6017	One Last Hero	http://www.last.fm/music/One+Last+Hero	http://userserve-ak.last.fm/serve/252/29459971.jpg	I6017
12	11679	Andrew Hill	http://www.last.fm/music/Andrew+Hill	http://userserve-ak.last.fm/serve/252/29850069.jpg	I11679
13	601	Prince	http://www.last.fm/music/Prince	http://userserve-ak.last.fm/serve/252/3031469.jpg	I601
14	11696	2AM	http://www.last.fm/music/2AM	http://userserve-ak.last.fm/serve/252/60616519.png	I11696

THE CODE

```
## Load Data
artists = read.table("Data/artists_gp4.dat",sep="\t",stringsAsFactors = F,comment.cha
user_artists = read.table("Data/user_artists_gp4.dat",sep="\t",header = T)

### Data Transformation
# Convert to wide format where each row represents the listened count of a user
user_artists_wide <- spread(user_artists,key=artistID,value=weight)
dim(user_artists_wide)

## Create character Id
artists$charid=paste0("I",artists$id)
userids=user_artists_wide$userID
rownames(user_artists_wide) = paste0("U",userids)
colnames(user_artists_wide) = paste0("I",colnames(user_artists_wide))
user_artists_wide$IuserID = NULL
user_artists_wide[1:6,1:10]

# Select Top 1000
visits_byitem=colSums(user_artists_wide[,-1],na.rm = T)
visits_1k = user_artists_wide[,order(visits_byitem,decreasing = T)[1:1000]]

# Select users who has listened to at least 10 artists
num_visits=apply(visits_1k,1,function(x) return(sum(!is.na(x))))
visits_1k = visits_1k[num_visits>10,]
dim(visits_1k)

# Data is centered and scaled
visits_1k=t(scale(t(visits_1k))[,])

# Convert visits_1k into a recommanderlab sparse matrix
visits_1k_rrm=as(as.matrix(visits_1k),"realRatingMatrix")
r <- visits_1k_rrm
```

TRANSFORMATIONS

user_artists_wide

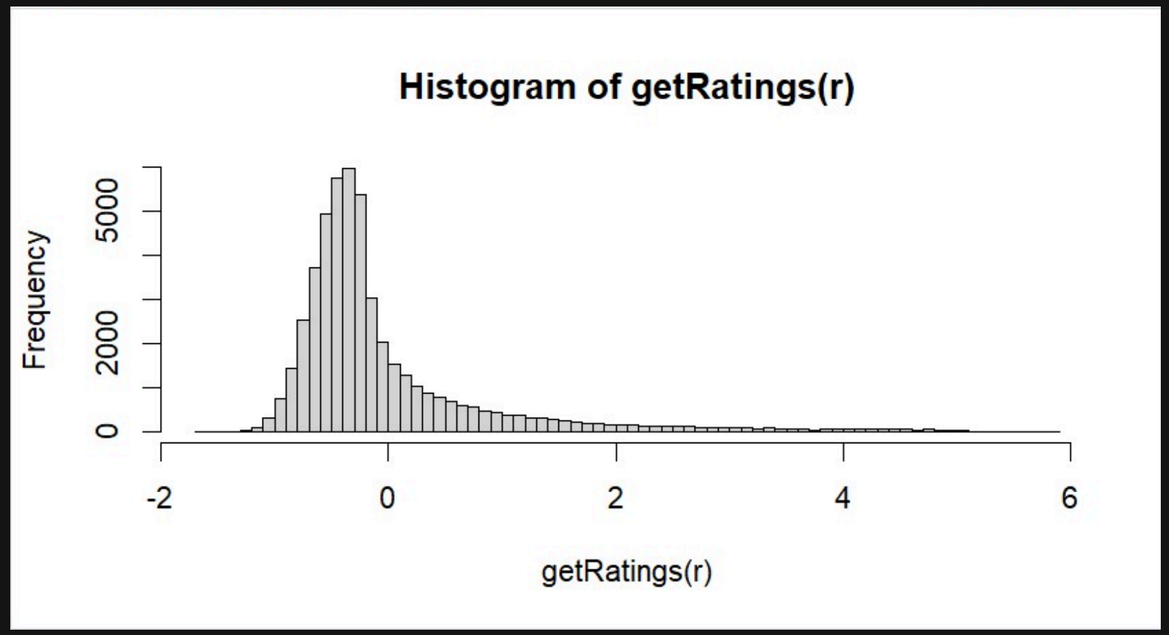
	I1	I2	I3	I4	I5	I6	I7	I8
U2	NA	NA	NA	NA	NA	NA	NA	NA
U3	NA	NA	NA	NA	NA	NA	NA	NA
U4	NA	NA	NA	NA	NA	NA	NA	NA
U5	NA	NA	NA	NA	NA	NA	NA	NA
U6	NA	NA	NA	NA	NA	NA	NA	NA
U7	NA	NA	NA	NA	NA	NA	NA	NA
U8	NA	NA	NA	NA	NA	NA	NA	NA
U9	NA	NA	NA	NA	NA	NA	NA	587

visits_1k

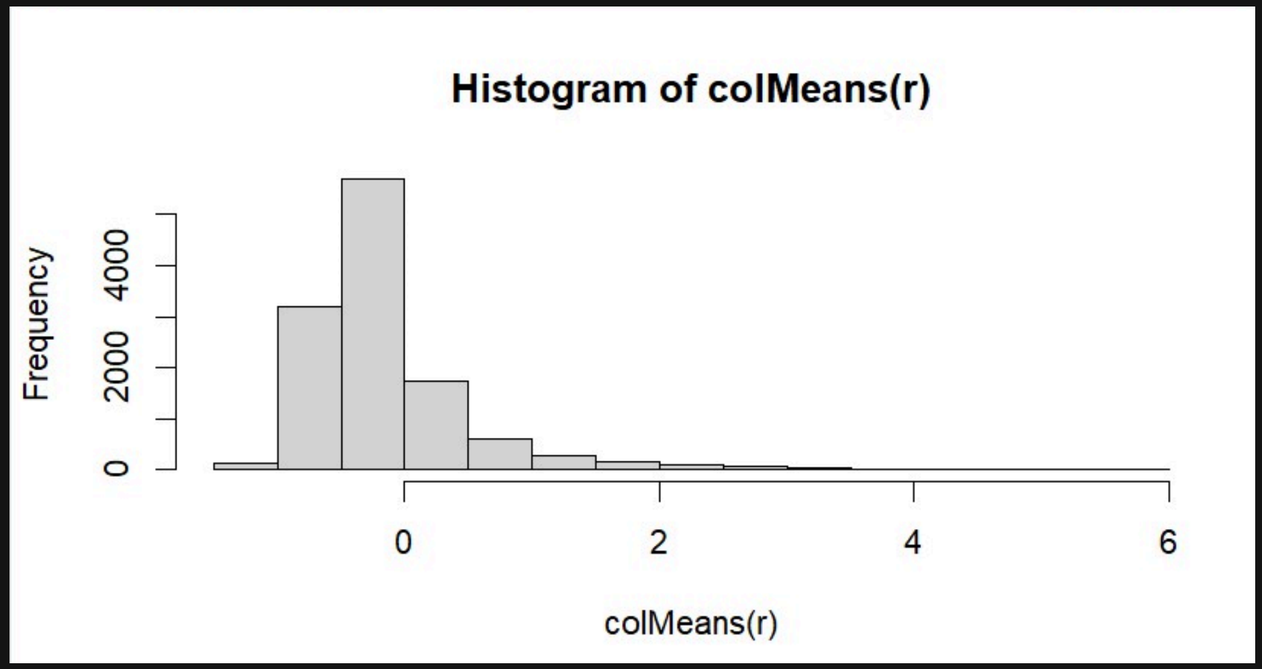
	I288	I88	I291	I497	I377	I66	I700	I287	I678	I71
U2	NA	-0.6768235	NA	NA	NA	NA	NA	NA	NA	-0.27628338
U4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
U5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
U7	NA	NA	2.13900569	NA	NA	NA	NA	NA	NA	NA
U8	-0.32807453	NA	NA	NA	NA	NA	NA	NA	NA	-1.06980824
U10	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
U11	0.37605659	NA	NA	NA	NA	NA	NA	NA	NA	NA
U12	-0.25204409	NA	NA	NA	NA	NA	NA	NA	NA	NA
U13	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
U15	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
U17	-0.31904567	NA	NA	NA	NA	NA	-0.3229524	NA	NA	NA
U18	NA	NA	NA	NA	NA	NA	NA	NA	NA	-0.31771051

DATA UNDERSTANDING

Ratings per interactions

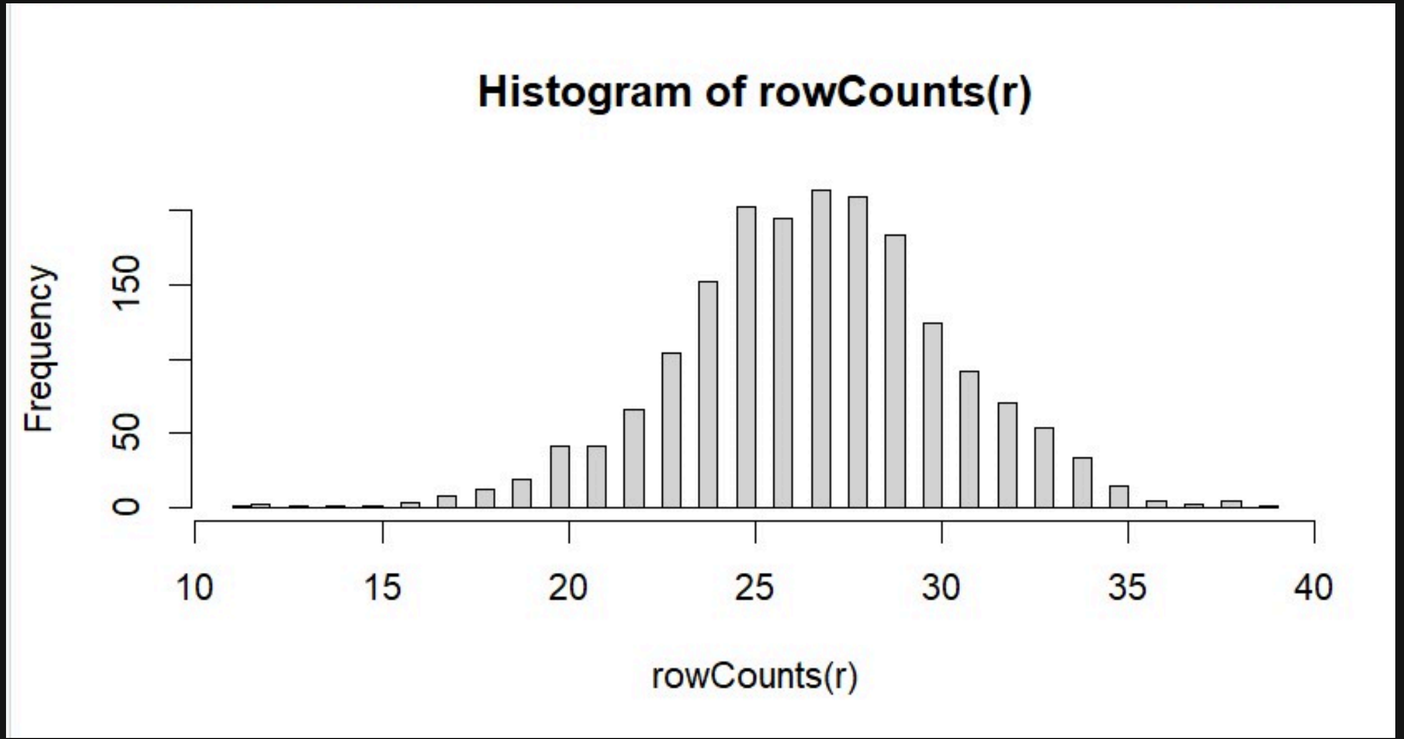


Ratings per artists

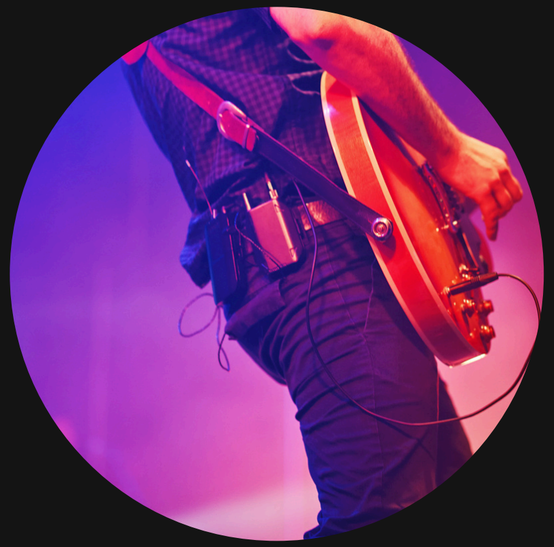


DATA UNDERSTANDING

Artists per user



RECOMMENDATION METHODS



- Popularity-Based (POPULAR): Popular artists based on global user interactions.
- User-Based Collaborative Filtering (UBCF): Identifies users with similar tastes.
- Singular Value Decomposition (SVD): A matrix factorization technique that uncovers latent patterns in user-item interactions.
- Singular Value Decomposition with Features (SVDF): An enhanced version of SVD that incorporates additional contextual features for improved recommendations.
- Alternating Least Squares (ALS): A matrix factorization algorithm that optimizes user-item preference predictions.
- Hybrid Approach: Combines recommendation techniques (UBCF + POPULAR).
- Cascade: A sequential recommendation approach where one method refines or filters the results of another, enhancing precision.

TRAINING THE MODELS

EVALUATION SCHEME

- Method: Cross Validation
- Folds: 4
- Given recommendations: 10
- Threshold: 1.2

```
set.seed(100)
e <- evaluationScheme(r, method="cross", k=4, given=10, goodRating=1.2)
```

TRAINING THE MODELS

CREATING RECOMMENDERS

```
train=getData(e, "train")
test=getData(e, "known")

ub_r <- Recommender(train, method="UBCF", param=list(nn=50, normalize="center"))
p_r <- Recommender(train, method="POPULAR")
svd_r <- Recommender(train, method="SVD")
svdf_r <- Recommender(train, method="SVDF")
als_r <- Recommender(train, method="ALS")
alsi_r <- Recommender(train, method="ALS_implicit")
```

```
set.seed(100)
# Create a hybrid recommender
hybrid_r <- HybridRecommender(
  Recommender(train, method = "POPULAR"),
  Recommender(train, method = "UBCF"),
  Recommender(train, method = "SVD"),
  weights = c(0.8, 0.2, 0.0)
)
hybrid_p <- predict(hybrid_r, test, type="ratings")

# Hybrid + cascade aproach of Popular, IBCF and UBCF + UBCF
hcas <- (0.8 * as(p_p, "matrix") + 0.2 * as(ub_p, "matrix") )
hcas_rrm <- as(hcas, "realRatingMatrix")
hcas_r <- Recommender(hcas_rrm, method="UBCF", param=list(nn=50))
hcas_p <- predict(hcas_r, test, type="ratings")
```

RESULTS AND VISUALIZATIONS

ERRORS

	RMSE	MSE	MAE
POPULAR	1.024162	1.048907	0.7308125
ub	1.096134	1.201509	0.7868232
svd	1.063455	1.130938	0.7569970
svdf	1.028155	1.057102	0.7489630
als	1.021738	1.043948	0.7333334
hybrid	1.022502	1.045511	0.7307811
hcas	1.027282	1.055307	0.7645216

RESULTS AND VISUALIZATIONS

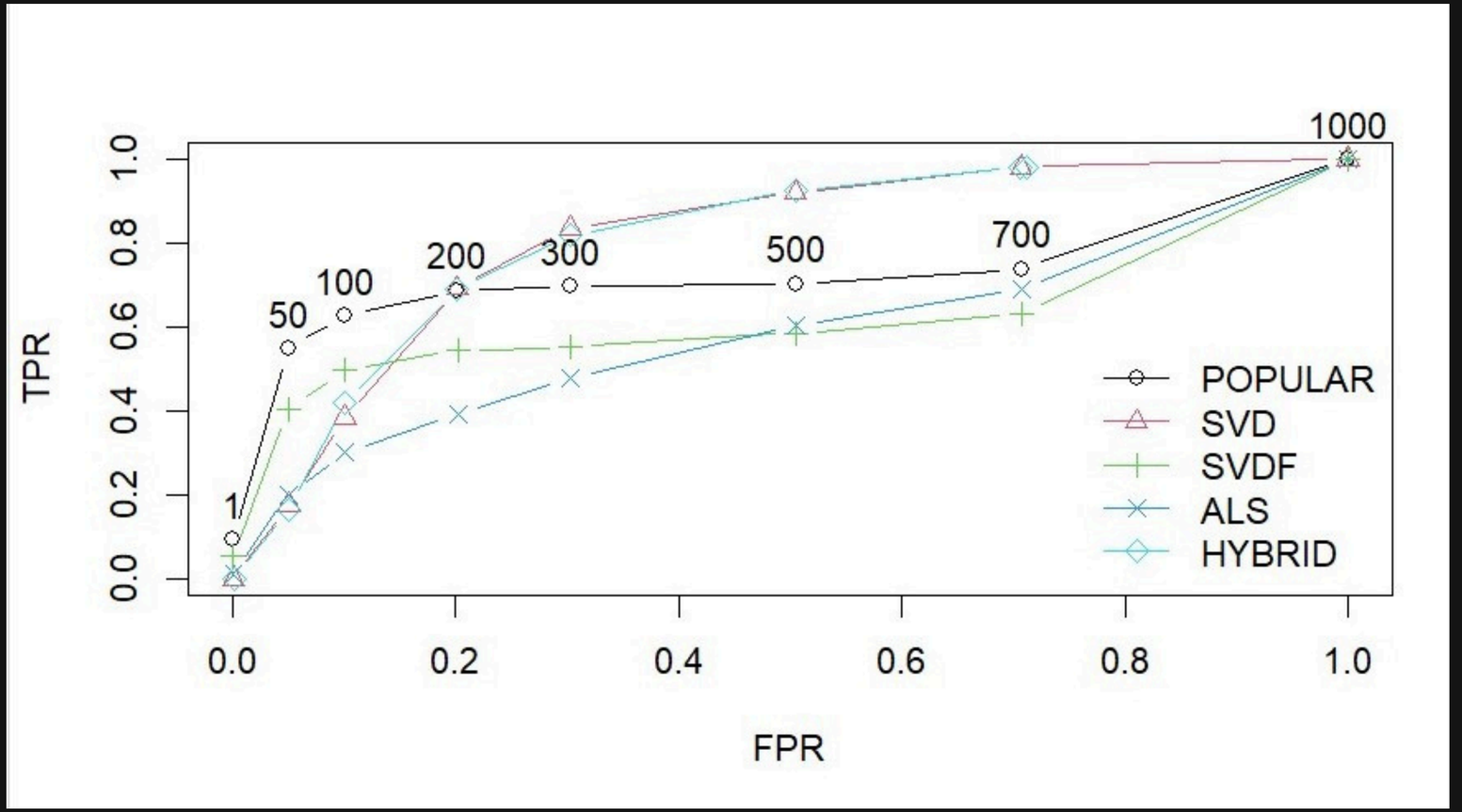
CONFUSION MATRIX

$SVD

	TP	FP	FN	TN	N	precision	recall	TPR	FPR	n
[1,]	0.00000000	1.00000	0.489847716	988.5102	990	0.0000000000	0.0000000	0.0000000	0.001010608	1
[2,]	0.08629442	49.91371	0.403553299	939.5964	990	0.0017258883	0.1756072	0.1756072	0.050443144	50
[3,]	0.18781726	99.81218	0.302030457	889.6980	990	0.0018781726	0.3844394	0.3844394	0.100870919	100
[4,]	0.33629442	199.66371	0.153553299	789.8464	990	0.0016814721	0.6910686	0.6910686	0.201781707	200
[5,]	0.40862944	299.59137	0.081218274	689.9188	990	0.0013620981	0.8352981	0.8352981	0.302769385	300
[6,]	0.45177665	499.54822	0.038071066	489.9619	990	0.0009035533	0.9198508	0.9198508	0.504847373	500
[7,]	0.48096447	699.51904	0.008883249	289.9911	990	0.0006870921	0.9813333	0.9813333	0.706939516	700
[8,]	0.48984772	989.51015	0.000000000	0.0000	990	0.0004944850	1.0000000	1.0000000	1.000000000	1000

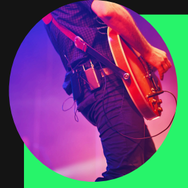
RESULTS AND VISUALIZATIONS

ROC CURVE



CONCLUSION

- The hybrid model of 80% Popularity based + 20% User Based is the best model overall.
- Even though SVD doesnt have the best performance it has the best ROC curve and is perfect for our user_artist_wide Matrix.
- After some difficulties implementing the hybrid model in the app we decided to use the SVD for the reasons stated above.



MODEL TESTING
VS
IMPLEMENTATION

+SCRAPPER

+ SERVER

Differences

- Selecting all artists vs Top 1000 artists
- Database (all artists) is using SVD model

Our goal: Select the best artists for each user by considering all necessary data (i.e. all artists).

```
# Select users who has listened to at least 10 artists
num_visits=apply(user_artists_wide,1,function(x) return(sum(!is.na(x))))
user_artists_10 = user_artists_wide[num_visits>10,]
dim(user_artists_10)
```

```
database=as(as.matrix(user_artists_wide),"realRatingMatrix")
model <- Recommender(r, method = "SVD")
```

Scrapper

- Used what we learned in class to do this function
- Select top 5 tracks of each artist
- Select the Image URL
  - Remove unnecessary characters

Our goal: Have a list of the Top 5 tracks from the recommended artists along with an image for each artist.

```
# Function to scrape the top tracks and background image URL from the Last.fm page
scrap_url <- function(artist_url) {
  page <- read_html(artist_url)

  # Get top tracks (first 5)
  top_tracks <- page %>%
    html_nodes(".chartlist-name a") %>%
    html_text() %>%
    head(5)

  # Scrape background image URL (from the background-image CSS property)
  img_url <- page %>%
    html_nodes(".header-new-background-image") %>%
    html_attr("style") %>%
    sub(".*url\\((.*)\\).*", "\\1", .)

  return(list(top_tracks = top_tracks, img_url = img_url))
}
```

CONT.

Server

- Listen for User interaction; “Show Recommendations” button
- Identify selected User

```
# 🔷 Server
server <- function(input, output, session) {

  # Reactive expression to hold recommendations
  recommendations <- reactiveVal(NULL)

  # Create a reactive expression for the user input
  selected_user <- reactive({
    req(input$user)  # Ensure user is selected
    input$user       # Return the selected user
  })

  # Observe the button click event to generate recommendations
  observeEvent(input$evaluate, {
    req(input$user)  # Ensure user is selected

#cat("user: ",input$user, "\n")
```

- Generates Recommendations (pre-trained model)
- Retrieves artists’ details (Name, ID, Web URL)

```
    # Generate recommendations for the selected user
    recomendations <- predict(model, database[selected_user(), drop = FALSE], n = 8)
    predicted_artists <- as(recomendations, "list")[[1]]

 #cat("predict: ",predicted_artists, "\n")

    if (length(predicted_artists) == 0) {
      recommendations(NULL)  # No recommendations found
    } else {
      # Filter artist details
      recommended_artists <- artists[artists$charid %in% predicted_artists, c("name", "charid", "url")]

      # Ensure recommended_artists has results
      if (nrow(recommended_artists) == 0) {
        recommendations(NULL)  # No matching artists found
      } else {
        # For each recommended artist, fetch their top tracks and image URL
        top_artist_info <- lapply(recommended_artists$url, scrap_url)
```

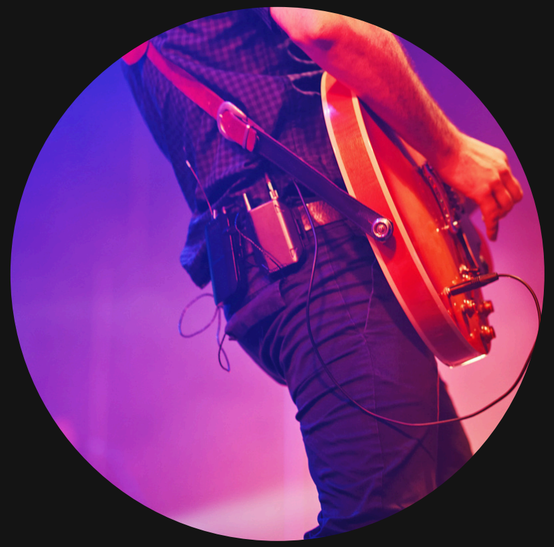
- Additional information gathered (Top Tracks, image URL)
- Provide Results in the User Interface

```
        # Add top tracks and background image URL to the data
        recommended_artists$top_tracks <- sapply(top_artist_info, function(info){
          # Flatten the list of top tracks to a single character vector
          if (length(info$top_tracks) == 0) {
            return("No top tracks found")
          }
          return(paste(info$top_tracks, collapse = ", "))
        })
        recommended_artists$img_url <- sapply(top_artist_info, function(info) info$img_url)

        data=recommended_artists[1:5,]
        recommendations(data) # Save the results to the reactive value
#print(data$name)
      }
    }
```

Our goal: Generate & display personalized artist recommendations to each User.

SHINNY APP



- Objective: Create an interactive music recommendation web application
- Selection of a user profile → Personalized suggestions of artists and songs
- 3 part presentation : UI, Recommendations, CSS

USER INTERFACE

Music Recommendation System

Select User:

U2

🔍 Show Recommendations

Click on "Show Recommendations" to see the recommendations

🎧 Recommended Artists

```
ui <- fluidPage(
  titlePanel(tags$strong("Music Recommendation System", style="text-align: center;")),
  hr(),

  sidebarLayout(
    sidebarPanel(
      selectizeInput("user", "Select User:", choices = rownames(r), multiple = FALSE, options = list(
        placeholder = 'Search for a user...',
        maxOptions = 100,  # Increase the number of displayed options
        create = FALSE,  # Prevent users from creating new options
        persist = FALSE  # Prevent persisting values not in the list
      )),
      actionButton("evaluate", "🔍 Show Recommendations"),
      h5('Click on "Show Recommendations" to see the recommendations', class="black-text")
    ),
```

```
    mainPanel(
      uiOutput("dynamic_css"),
      h3("🎧 Recommended Artists",
         style="text-align: center; font-size: 36px; font-weight: bold; color: #FFFFFF;"),
      uiOutput("recommendations_ui")
    )
  )
)
```

GENERATING RECOMMENDATIONS

Music Recommendation System

Select User:

U34

Show Recommendations

Click on "Show Recommendations" to see the recommendations

Recommended Artists

Pearl Jam

Top Tracks:

- Even Flow
- Black
- Alive
- Jeremy
- Yellow Ledbetter

Joy Division

Top Tracks:

- Love Will Tear Us Apart - 2020 Remaster
- Disorder - 2007 Remaster
- Shadowplay - 2007 Remaster
- Love Will Tear Us Apart
- She's Lost Control - 2007 Remaster

Avenged Sevenfold

Top Tracks:

```
# Render the recommendations UI based on the reactive value
output$recommendations_ui <- renderUI({
  rec_artists <- recommendations()
  #print(rec_artists$name)

  # Check if recommendations exist
  if (is.null(rec_artists)) {
    return(h4("❌ No recommendations found!", style="color: #FFFFFF; font-weight: bold;"))
  }
```

```
# Display recommended artists and their top tracks
lapply(1:nrow(rec_artists), function(i) {
  artist <- rec_artists[i, ]

  fluidRow(
    column(3,
           img(src = artist$img_url, height = "200px", style="border-radius:10px; max-width: 100%;")),
    #cat("img: ",artist$img_url, "\n"),
    column(9,
           h4(artist$name, style="font-weight: bold; color: #FFCC00; font-size: 24px;"),
           #cat("name: ",artist$name, "\n"),
           #cat("traks: "),
           #print(artist$top_tracks),
           strong("Top Tracks:", class = "top-song"),
           tags$ul(lapply(strsplit(artist$top_tracks, ",")[[1]], function(song) {
             tags$li(style="color: #FFFFFF; text-align: left; font-size: 16px;", trimws(song))
           }))
    ),
    hr()
```

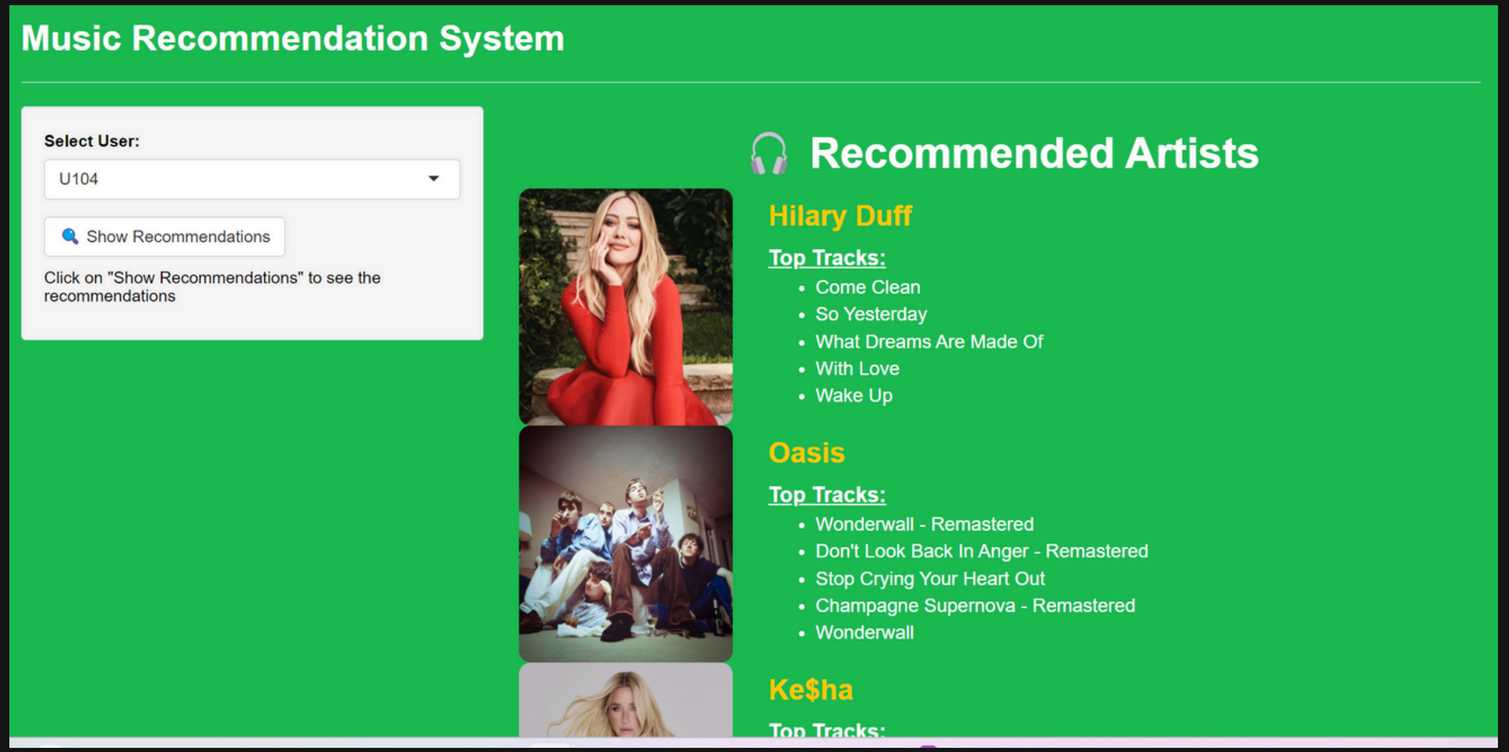
CUSTOMIZATION WITH CSS

```
  # Apply CSS styles
  output$dynamic_css <- renderUI({
    tags$style(HTML("
      .sidebar {
        background-color: #1DB954;
      }

      body {
        background-color: #1DB954;
        color: white;
      }

      .top-song {
        color: white;
        text-decoration: underline;
        text-align: left;
        font-size: 18px;
      }
```

```
        .black-text {
          color: black !important;
        }
        .control-label {
          color: black !important;
          font-weight: bold;
        }
      "))
    })
}
shinyApp(ui = ui, server = server)
```



THANK'S FOR LISTENING