DataScience-Project

Layal Ghryani - Rayanah Alsubaie - Shaymaa Aldabbagh

2023-12-10

```
# Install and load necessary packages
if (!require("shiny")) install.packages("shiny")
```

```
## Loading required package: shiny
```

```
library(shiny)
if (!require("tidyverse")) install.packages("tidyverse")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidyverse)
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)
if (!require("scales")) install.packages("scales")
```

```
## Loading required package: scales
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(scales)
if (!require("caret")) install.packages("caret")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(caret)
if (!require("viridis")) install.packages("viridis")
```

```
## Loading required package: viridis
## Loading required package: viridisLite
##
## Attaching package: 'viridis'
##
## The following object is masked from 'package:scales':
##
##     viridis_pal
```

```r
library(viridis)
```

```r
# Read the CSV file
Spotify <- read.csv("spotify_songs.csv")
Spotify2 <- read.csv("spotify.csv")
```

```r
# Set the maximum file size limit to 30 MB
options(shiny.maxRequestSize = 30 * 1024^2)

# Load the required packages
library(shiny)
library(htmltools)

# Load datasets
Spotify <- read.csv("spotify_songs.csv")
Spotify2 <- read.csv("spotify.csv")

# Define the UI
ui <- fluidPage(
  tags$head(
    tags$title("CS 30721: Data Science - Shiny App"),
    tags$style(
      HTML("
        #title {
          text-align: center;
          margin-top: 20px;
        }
```

```
    ")
  )
),

wellPanel(
  tags$h1("Predicting Track Popularity: An Analysis of Spotify's Music Dataset", id = "title"),
  tags$hr(),
  HTML("<div style='text-align: center;'>
    <p>Layal Ghryani, Rayanah Alsubaie, Shaymaa Aldabbagh</p>
    <p>Instructor: Dr. Zain Balfagih</p>
    <p>CS 30721: Data Science</p>
  </div>"),
  tags$div(
    style = "text-align: center;",
    img(src = "Spot.png", width = "400px")
  )
),

mainPanel(
  tabsetPanel(
    tabPanel("Introduction",
      h2("Welcome to Our Data Science Final Project "),
      p("The development of music streaming services has revolutionized the music business by providi
      p("Navigate through different tabs to explore data pre-processing, exploratory data analysis, a
    ),
    tabPanel("Data Pre-processing",
      tabsetPanel(
        tabPanel("Data Source",
          fluidRow(
            column(
              width = 10,
              height = 20,
              h4("Importing Dataset"),
              verbatimTextOutput("importingDatasetOutput"),
              br(),
              h4("Dataset Dimensions"),
              verbatimTextOutput("datasetDimensionsOutput"),
              br()
            ),
            column(
              width = 10,
              height = 20
            )
          )
        ),
        tabPanel("Data Cleaning",
          fluidRow(
            column(
              width = 6,
              h4("DataSet Cleaning"),
              tableOutput("DatasetCleaning Table")
            )
          ),
```

```
      br(),
      h4("Analysis of Dataset Structure"),
      verbatimTextOutput("datasetStructureOutputSpotify"),
      br(),
      h4("Identifying Missing Values"),
      verbatimTextOutput("missingValuesOutputSpotify"),
      br(),
      h4("Checking for Duplicate Track IDs"),
      verbatimTextOutput("duplicateTrackIDsOutputSpotify"),
      br(),
      h4("Number of Duplicate Values"),
      verbatimTextOutput("numDuplicateValuesOutputSpotify"),
      br(),
      h4("Removing unnecessary columns"),
      verbatimTextOutput("cleanedDatasetTableSpotify"),
      br(),
      h4("Checking summary of numerical variables"),
      verbatimTextOutput("summaryNumericalVariablesSpotify"),
      br(),
      h4("Analysis of Dataset Structure (Spotify2)"),
      verbatimTextOutput("datasetStructureOutputSpotify2"),
      br(),
      h4("Identifying Missing Values (Spotify2)"),
      verbatimTextOutput("missingValuesOutputSpotify2"),
      br(),
      h4("Number of Duplicate Values (Spotify2)"),
      verbatimTextOutput("numDuplicateValuesOutputSpotify2"),
      br(),
      h4("Checking summary of numerical variables (Spotify2)"),
      verbatimTextOutput("summaryNumericalVariablesSpotify2")
    ),
    tabPanel("Cleaned Data Sets",
      fluidRow(
        column(
          width = 6,
          h4("Cleaned Dataset - Spotify"),
          tableOutput("cleanedDatasetTableSpotify"),
          h5("Dimensions:"),
          textOutput("dimensionsOutputSpotify")
        ),
        column(
          width = 6,
          h4("Cleaned Dataset - Spotify2"),
          tableOutput("cleanedDatasetTableSpotify2"),
          h5("Dimensions:"),
          textOutput("dimensionsOutputSpotify2")
        ),
        column(
          width = 12,
          h4("Frequencies Across Genres - Spotify"),
          tableOutput("genreFrequenciesTable")
        ),
        column(
```

```
            width = 6,
            h4("Show Dataset - Spotify"),
            dataTableOutput("showDatasetTableSpotify")
          ),
          column(
            width = 6,
            h4("Show Dataset - Spotify2"),
            dataTableOutput("showDatasetTableSpotify2")
          )
        )
      )
    )
  ),
  tabPanel("Exploratory Data Analysis",
    tabsetPanel(
      tabPanel("Popularity Analysis",
        tabsetPanel(
          tabPanel("Analysis",
          plotOutput("popularityAnalysisOutput")),
          tabPanel("Top 10 Popular Songs",
            plotOutput("top10PopularPlot")),
          tabPanel("Top 10 Least Popular Songs",
            tableOutput("top10LeastPopularTable")),
          tabPanel("Popularity vs. Acousticness",
            plotOutput("popularityAcousticnessPlot")),
          tabPanel("Top 15 Artists with Most Songs",
            plotOutput("top15ArtistsPlot")),
          tabPanel("Popularity vs. Duration",
            plotOutput("popularityDurationPlot")),
          tabPanel("Popularity vs. Danceability",
            plotOutput("popularityDanceabilityPlot"))
        )
      ),
      tabPanel("Correlation Between Attributes",
        tabsetPanel(
          tabPanel("Correlation Heatmap",
            plotOutput("correlationHeatmap")),
          tabPanel("Loudness vs. Energy",
            plotOutput("loudnessEnergyPlot")),
          tabPanel("Change in Duration Over Years",
            plotOutput("durationOverYearsPlot")),
          tabPanel("Average Duration by Genre",
            tableOutput("averageDurationByGenreTable")),
          tabPanel("Energy vs. Danceability",
            plotOutput("Workout vibes Plot"))
        )
      )
    )
  ),
  tabPanel("ML Model",
    tabsetPanel(
      tabPanel("Machine Learning Results",
        tableOutput("machineLearningResultsTable"))
```

```
        )
      )
    ),
    br(),
    h3("Additional Information"),
    p("This Shiny app provides insights into the analysis of Spotify's music dataset."),
    p("Explore different visualizations and tables to understand trends and correlations.")
  )
)

# Define the server
server <- function(input, output) {

  # Output importing dataset code
  output$importingDatasetOutput <- renderText({
    c("Spotify <- read.csv('spotify_songs.csv')",
      "\nSpotify2 <- read.csv('spotify.csv')")
  })

  # Output dataset dimensions
  output$datasetDimensionsOutput <- renderText({
    paste("\nDimensions of Spotify:", dim(Spotify),
          "\nDimensions of Spotify2:", dim(Spotify2))
  })

  # Check dataset dimensions
  observeEvent(input$checkDimensionsButton, {
    output$datasetDimensionsOutput <- renderText({
      dim(Spotify)  # or use dim(Spotify2) if that's the dataset you want to check
    })
  })

  top_10_popular_songs <- head(Spotify[order(-Spotify$track_popularity), c("track_name", "track_artist"

  # Create a bar plot with varied colors
  output$top10PopularPlot <- renderPlot({
    ggplot(top_10_popular_songs, aes(x = track_name, y = track_popularity, fill = track_artist)) +
      geom_bar(stat = "identity") +
      labs(title = "Top 10 Popular Songs",
           x = "Song",
           y = "Popularity") +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
  })

  # Sort the songs by popularity in ascending order
  top_10_least_popular_songs <- head(Spotify[order(Spotify$track_popularity), c("track_name", "track_ar

  # Display the top 10 least popular songs
  output$top10LeastPopularTable <- renderTable({
    top_10_least_popular_songs <- head(Spotify[order(Spotify$track_popularity), c("track_name", "track_
    top_10_least_popular_songs
  })
```

```
spotify2 <- reactive({
  req(input$dataset2)
  read.csv(input$dataset2$datapath)
})

# Data Cleaning
observe({
  # Analyzing the structure of the dataset
  output$datasetStructureOutputSpotify <- renderPrint({
    str(Spotify)
  })
  output$datasetStructureOutputSpotify2 <- renderPrint({
    str(Spotify2)
  })

  # Identifying missing values across columns
  output$missingValuesOutputSpotify <- renderPrint({
    col_miss <- colSums(is.na(Spotify))
    col_miss[col_miss > 0]
  })

  output$missingValuesOutputSpotify2 <- renderPrint({
    col_miss <- colSums(is.na(Spotify2))
    col_miss[col_miss > 0]
  })

  # Find number of duplicate values
  output$numDuplicateValuesOutputSpotify <- renderPrint({
    duplicate_obs <- duplicated(Spotify)
    paste("There are", sum(duplicate_obs), "duplicate observations in the data")
  })

  # Find number of duplicate values
  output$numDuplicateValuesOutputSpotify2 <- renderPrint({
    duplicate_obs2 <- duplicated(Spotify2)
    paste("There are", sum(duplicate_obs2), "duplicate observations in the data")
  })

  # Check for duplicate track ID
  output$duplicateTrackIDsOutputSpotify <- renderPrint({
    duplicate_id <- duplicated(Spotify$track_id)
    sum(duplicate_id)
  })

  # Removing unnecessary columns
 #() output$cleanedDatasetTableSpotify <- renderTable({
   # Spotify <- Spotify %>% dplyr::select(-track_id, -track_album_id, -playlist_id)
 #   Spotify
#  })

  # Checking summary of numerical variables
  output$summaryNumericalVariablesSpotify <- renderPrint({
    Spotify_num <- Spotify %>% select_if(is.numeric)
```

```
    summary(Spotify_num)
  })

  # Checking summary of numerical variables
  output$summaryNumericalVariablesSpotify2 <- renderPrint({
    Spotify2_num <- Spotify2 %>% select_if(is.numeric)
    summary(Spotify2_num)
  })

})

# Inside your server function

# Output for cleaned dataset table - Spotify
output$cleanedDatasetTableSpotify <- renderTable({
  Spotify <- Spotify %>% dplyr::select(-track_id, -track_album_id, -playlist_id)
  Spotify
})

# Output for cleaned dataset table - Spotify2
output$cleanedDatasetTableSpotify2 <- renderTable({
  Spotify2 <- Spotify2 %>% dplyr::select(-track_id, -track_album_id, -playlist_id)
  Spotify2
})

# Output for dimensions - Spotify
output$dimensionsOutputSpotify <- renderText({
  paste("There are ", dim(Spotify)[1], "observations and", dim(Spotify)[2],
        "columns in our cleaned dataset")
})

# Output for dimensions - Spotify2
output$dimensionsOutputSpotify2 <- renderText({
  paste("There are ", dim(Spotify2)[1], "observations and", dim(Spotify2)[2],
        "columns in our cleaned dataset")
})

# Output for frequencies across genres
output$genreFrequenciesTable <- renderTable({
  spotify_genre_frequencies <- Spotify %>%
    group_by(playlist_genre) %>%
    summarise(total = n())
  kable(spotify_genre_frequencies) %>%
    kable_material(c("striped", "hover"))
})

# Output for showing dataset - Spotify
output$showDatasetTableSpotify <- renderDataTable({
  datatable(
    head(Spotify, 100),
    class = 'row-border stripe hover compact',
    rownames = FALSE,
    autoHideNavigation = TRUE,
```

```
      escape = FALSE
    )
  })

  # Output for showing dataset - Spotify2
  output$showDatasetTableSpotify2 <- renderDataTable({
    datatable(
      head(Spotify2, 100),
      class = 'row-border stripe hover compact',
      rownames = FALSE,
      autoHideNavigation = TRUE,
      escape = FALSE
    )
  })
  # Output for Popularity Analysis
output$popularityAnalysisOutput <- renderUI({
  Spotify <- Spotify %>%
    mutate(popularity = case_when(
      track_popularity <= 30 ~ "low",
      track_popularity > 30 & track_popularity <= 75 ~ "medium",
      track_popularity > 75 ~ "high"
    ))

  # Top tracks in the dataset
  popular_track <- Spotify %>%
    filter(popularity == "high") %>%
    arrange(desc(track_popularity)) %>%
    distinct(track_name, track_popularity)

  # Display the top tracks in a DataTable
  datatable(
    head(popular_track, 10),
    extensions = 'FixedColumns',
    options = list(
      scrollY = "400px",
      scrollX = TRUE,
      fixedColumns = TRUE
    )
  )
# Create a summary of top artists within each playlist genre
  artist_genre <- spotify %>%
    dplyr::select(playlist_genre, track_artist, track_popularity) %>%
    group_by(playlist_genre, track_artist) %>%
    summarise(n = n()) %>%
    top_n(10, n)

  # Create a treemap visualization
  tm <- treemap(artist_genre, index = c("playlist_genre", "track_artist"), vSize = "n", vColor = 'playli

  # Display the treemap
  tm_plot <- htmltools::tags$img(src = paste("data:image/svg+xml;utf8,", URLencode(print(tm))), style =

  # Combine the DataTable and treemap
```

```
  fluidRow(
    column(6, popular_track_table),
    column(6, tm_plot)
  )
})

# Output for Correlation Heatmap
output$correlationHeatmap <- renderPlot({
  # Select the variables for correlation
  variables <- c('danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness

  # Compute the correlation matrix
  correlation_matrix <- cor(Spotify[, variables])

  # Create a heatmap
  melted_correlation <- reshape2::melt(correlation_matrix)
  ggplot(data = melted_correlation, aes(x = V1, y = V2, fill = value)) +
    geom_tile() +
    scale_fill_gradient(low = "darkblue", high = "pink") +
    theme_minimal() +
    labs(title = "Correlation Heatmap") +
    geom_text(aes(label = round(value, 2)), color = "white", size = 3) + coord_flip()
})

}

# Run the Shiny app
shinyApp(ui, server)
```