Report on Techniques Used in Spotify Data Analysis and Shiny Song Recommendation App

1. Data Loading and Initial Exploration

The readr::read_csv() function is used to load a dataset containing information about Spotify songs.

The summary() function provides a quick overview of the dataset's structure, including basic statistics on the variables.

Libraries Used:

- tidyverse: Provides essential tools for data manipulation and analysis.
- e1071: Used for implementing Support Vector Machine (SVM) algorithms. caret: Used for model training, testing, and evaluation.
- dplyr: A data manipulation package, part of tidyverse.
- ggplot2: For data visualization.
- shiny: The primary package used for building interactive web applications.
- stringdist: Provides string distance algorithms for matching similar text, used for fuzzy matching of song and artist names.
- shinyWidgets: Adds enhanced UI elements such as a searchable dropdown for selecting artists.

2. Data Cleaning

a. The data cleaning process includes: Removing ID Columns: track_id, track_album_id, and playlist_id columns, useful in database contexts, are removed as they are not relevant for the analysis.
b. Removing Duplicates: The distinct() function is applied to remove duplicate songs based on the combination of track_name and track_artist. This ensures the analysis focuses on unique songs.
c. Converting Dates: The mutate() function, along with as.Date() and format(), is used to convert the track_album_release_date column into a new release_year column, allowing for time-based analysis.

3. Top Genres and Artists Analysis (2000s and 2010s)

The analysis is divided into two decades: the 2000s and the 2010s.

a. Top Genres: A loop iterates over each year within a decade (e.g., 2000–2010) to determine the most popular genres by year. Songs are grouped by playlist_genre, and a bar plot is generated using ggplot2 to visualize the top genres for each year.
b. Top Artists: For each year in a decade, the top 5 artists are identified using group_by() and summarize(), and the results are combined into a single data frame. The artists are ranked by their number of appearances in the dataset.
c. Top Songs: Similarly, for each year, the top 5 songs based on track_popularity are selected using the arrange() function. The songs are then visualized by year.

4. K-Means Clustering

K-Means clustering is performed on a subset of song features, including track_popularity, danceability, energy, tempo, acousticness, instrumentalness, and loudness. This method groups songs into 5 clusters based on their musical features.

Steps:

a. The kmeans() function is used to group songs into clusters, with 5 centers and 25 random starts.
b. A scatter plot is created to visualize the clusters, using ggplot2. The x-axis represents track_popularity, and the y-axis represents tempo.
c. Silhouette Score: The average silhouette score is calculated using the silhouette() function from the cluster package. This score measures how well each song fits within its assigned cluster.

5. Support Vector Machine (SVM) for Classification

An SVM model is trained to classify songs by their playlist_genre, using multiple song attributes.

Steps:

a. The spotify_data_cleaned dataset is further refined to include only relevant features for classification, such as track_popularity, danceability, and tempo.
b. The target variable (playlist_genre) is converted into a factor.
c. The data is split into training and testing sets using the createDataPartition() function from the caret package.
d. An SVM model with a radial kernel is trained using the svm() function from the e1071 package.
e. The model is set up for classification (C-classification), with a cost of 1 and a gamma value of 0.5.
f. The trained model makes predictions on the test set, and its performance is evaluated using a confusion matrix generated by the confusionMatrix() function.

Evaluation: The confusion matrix reports the model's performance, including precision and recall metrics. The overall accuracy of the model is printed to assess its classification ability.

6. Shiny UI Design

The user interface (UI) consists of a title panel, a sidebar panel, and a main panel.

UI Elements:

a. Artist Selection: A dropdown menu (pickerInput) allows users to select an artist from the available options, with live search enabled for convenience.
b. Song Selection: A dynamic dropdown (selectInput) updates based on the artist selected. This allows users to pick a song by the chosen artist.
c. Recommendation Button: An action button (actionButton) triggers the recommendation process.
d. Output: The selected song is displayed using (textOutput), and a table of recommended songs is displayed using (tableOutput).

7. Server-Side Logic

The server-side logic of the Shiny app performs various tasks, including dynamically updating song choices, handling user input validation, and generating song recommendations based on similarity.

Key Functionalities:

a. Dynamic Song Suggestions: The updateSelectInput() function is used to update the song list based on the selected artist. When a user selects an artist, the app retrieves the songs by that artist from the cleaned dataset.
b. Fuzzy Matching: The stringdist::stringdist() function computes string distances to identify the closest matching artist and song based on the user's input. This ensures that even if the user input isn't perfect, the app can still find similar options.
c. Euclidean Distance Calculation: To recommend similar songs, Euclidean distance is calculated between the selected song's features (e.g., danceability, energy, valence) and those of other songs in the same genre.
d. Recommendations: The top 20 most similar songs are displayed based on the Euclidean distance. These recommendations are shown in a table format.

8. Running the App

The shinyApp() function is used to launch the Shiny app, which takes the UI and server components as arguments.

Key Features:

a. Interactive UI: The app allows users to dynamically select an artist and song and receive personalized song recommendations.
b. Fuzzy Matching: Handles slight user input errors in artist and song names.
c. Recommendation System: Recommends similar songs based on musical features using Euclidean distance calculations.