Project 1: Exploratory Data Analysis

Shreyash Singh

2024-02-10

Background

In this exploratory data analysis, we will delve into the world of music using the Spotify Songs data set. This data set contains loads of information about various songs, including their features, popularity, and the playlists they belong to. Our goal is to uncover patterns, trends, and relationships within the data, offering insights into the factors that contribute to the success of a song.

Problem Definition

One of the key challenges in the music industry is understanding the factors that contribute to the popularity of songs. Artists, producers, and streaming platforms seek insights into the characteristics of successful songs to tailor their strategies. The problem is to identify patterns and trends within the Spotify Songs data set that can shed light on the features associated with highly popular songs.The main problem to be addressed: How can we enhance music discovery through song Similarity using the different song features.

Loading and Preparing Data: Performing Data Wrangling, Munging, and Cleaning from the original CSV File

```
# Loading necessary libraries
library(tidyverse)
library(ggplot2)
library(plotly)
library(dplyr)

# Loading the dataset
spotify_songs <- read.csv("spotify_songs.csv")

# Checking for rows with missing values
missing_values <- sum(!complete.cases(spotify_songs))

# Checking for duplicates
duplicates <- sum(duplicated(spotify_songs))

#printing the missing values
cat("Number of rows with some missing values:", missing_values, "\nNumber of duplicates:",
    duplicates, "\n")
```

```
## Number of rows with some missing values: 5
## Number of duplicates: 0
```

After inspecting the Spotify Songs dataset, we notice that only a few rows contain missing values(5), and there are no duplicate entries. Since the missing data is insignificant compared to the whole dataset, we can remove it to ensure that our dataset becomes more reliable for analysis, and that the Exploratory Data Analysis produces more accurate results.
Moreover, since the study is about song popularity I want to replace all songs with missing popularity to the average popularity of the data set.
I also found out that there were multiple entries of the same songs with different ids so I tried to remove any extra entries with same title and artist name.
I am also converting date to Date type and also extracting weekdays for potential use.
Since the Recommendar uses numeric id and the track ids are present in string format I have created unique conversions for each one as a numeric value.

```
# Removing the rows with missing values
spotify_songs <- spotify_songs[complete.cases(spotify_songs), ]

# Replacing missing values in track_popularity
average_track_popularity <- mean(spotify_songs$track_popularity, na.rm = TRUE)
spotify_songs$track_popularity[is.na(spotify_songs$track_popularity)] <-
  average_track_popularity

#Removing the duplicated data with the same artist and song name
#While keeping the highest popularity observation if found
spotify_songs <- spotify_songs %>%
  group_by(track_name, track_artist) %>%
  arrange(desc(track_popularity)) %>%
  distinct(track_name, track_artist, .keep_all = TRUE) %>%
  ungroup()

# Converting release date to Date type for chronological analysis.
spotify_songs$track_album_release_date <- as.Date(spotify_songs$track_album_release_date)

# Extracting day of the week for potential insights into variations over weekdays.
spotify_songs$release_day <- weekdays(spotify_songs$track_album_release_date)

# Include a numeric identifier for each track
track_ids <- as.character(spotify_songs$track_id)
track_id_indices <- as.numeric(factor(track_ids, levels = unique(track_ids)))
```

Overview of the Dataset

Before we dive into the analysis, get an initial understanding of its structure along with a few samples of information present in each section of the Dataset.

```
# Displaying the total number of observations
cat("The total number of observations in the data frame is:", nrow(spotify_songs), "\n")
```

```
## The total number of observations in the data frame is: 26229
```

```
# Displaing the structure of the dataset with 2 random instances for context(samples)
str(spotify_songs[13:14,])
```

```
## tibble [2 x 24] (S3: tbl_df/tbl/data.frame)
##  $ track_id                : chr [1:2] "41L3O37CECZt3N7ziG2z7l" "2tnVG71enUj33Ic2nFN6kZ"
##  $ track_name              : chr [1:2] "Yummy" "Ride It"
##  $ track_artist            : chr [1:2] "Justin Bieber" "Regard"
##  $ track_popularity        : num [1:2] 95 94
##  $ track_album_id          : chr [1:2] "1SN6N3fNkZk5oXQ9X46QZ3" "4zOhjJfe0dwqsNdDYk622E"
##  $ track_album_name        : chr [1:2] "Yummy" "Ride It"
##  $ track_album_release_date: Date[1:2], format: "2020-01-03" "2019-07-26"
##  $ playlist_name           : chr [1:2] "Todo Éxitos" "Pop / Dance"
##  $ playlist_id             : chr [1:2] "2ji5tRQVfnhaX1w9FhmSzk" "6mXh8CUBMBsBUu88a4eAQV"
##  $ playlist_genre          : chr [1:2] "pop" "pop"
##  $ playlist_subgenre       : chr [1:2] "dance pop" "dance pop"
##  $ danceability            : num [1:2] 0.662 0.88
##  $ energy                  : num [1:2] 0.519 0.751
##  $ key                     : int [1:2] 9 7
##  $ loudness                : num [1:2] -6.55 -4.26
##  $ mode                    : int [1:2] 0 0
##  $ speechiness             : num [1:2] 0.106 0.0874
##  $ acousticness            : num [1:2] 0.404 0.177
##  $ instrumentalness        : num [1:2] 0.00 6.43e-05
##  $ liveness                : num [1:2] 0.121 0.106
##  $ valence                 : num [1:2] 0.495 0.884
##  $ tempo                   : num [1:2] 146 118
##  $ duration_ms             : int [1:2] 210427 157606
##  $ release_day             : chr [1:2] "Friday" "Friday"
```

Exploratory Data Analysis

1. Check the distribution of songs in each genre

Understanding the distribution of the songs based Different genres and sub-genres is necessory to know abou the distribution and kinds of data observations present in the set.

```
# Count the number of songs in each genre and playlist
genre_counts <- table(spotify_songs$playlist_genre)
sub_counts <- table(spotify_songs$playlist_subgenre)

# Print the genre distribution and playlist distribution:
cat("Genre Distribution:\n")
```

```
## Genre Distribution:
```

```
print(genre_counts)
```

```
##
##   edm latin   pop   r&b   rap  rock
##  4631  3874  4711  4107  5133  3773
```

```
# Create a data frame for visualization
sub_genre_data <- data.frame(SubGenre = names(sub_counts), Count = as.numeric(sub_counts))

# Pie chart for Sub Genre Distribution
sub_genre_distribution_pie_chart <- ggplot(sub_genre_data, aes(x = "", y = Count, fill = SubGenre)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar(theta = "y") +
  labs(title = "Sub Genre Distribution of Songs",
       fill = "Sub Genre") +
  theme_minimal() +
  theme(axis.text = element_blank(),
        axis.title = element_blank(),
        panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 18)) +
  scale_fill_discrete(name = "Sub Genre") +
  theme(legend.position = "bottom")

# Display the pie chart
print(sub_genre_distribution_pie_chart)
```

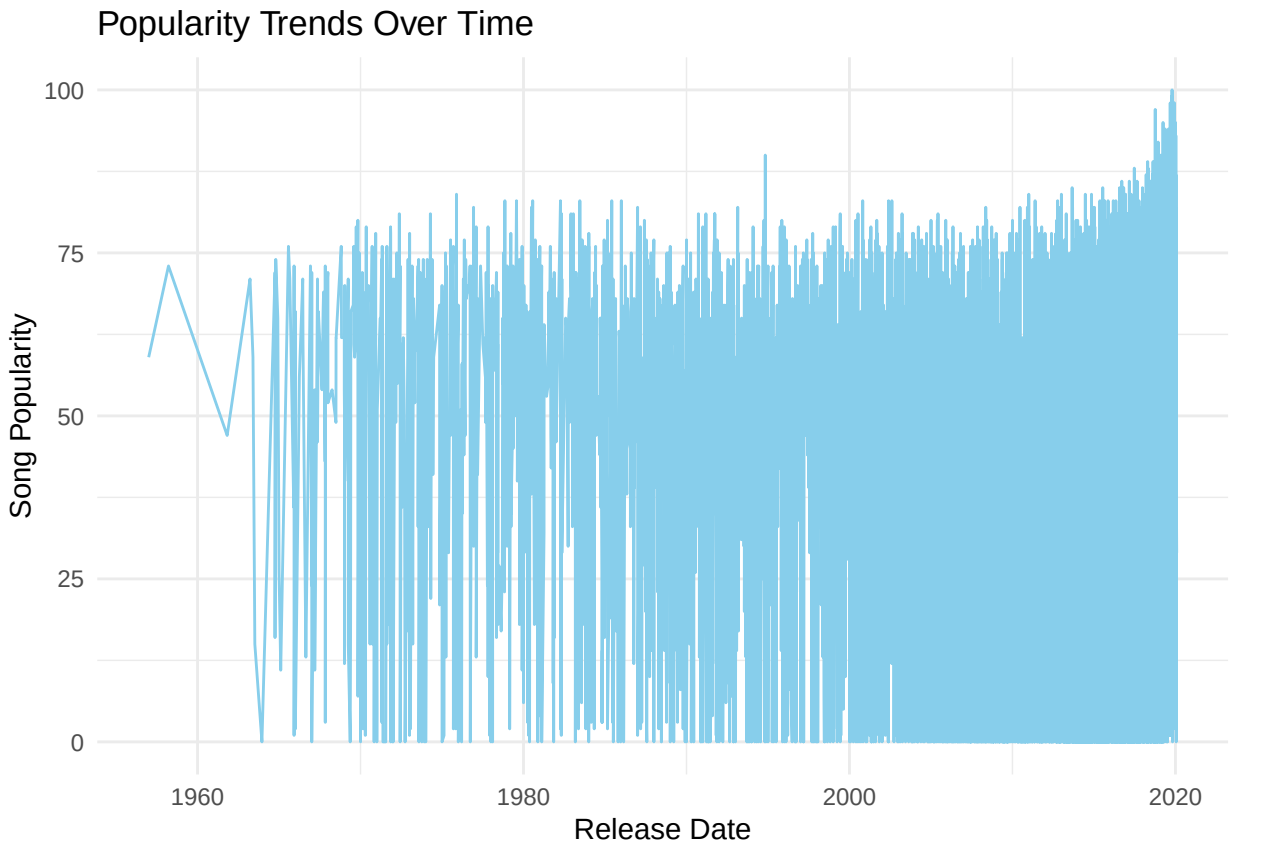
Sub Genre Distribution of Songs

It is evident that the data set contains diverse data sets of varied genres and kinds with almost equal distribution.
This makes the dataset more suitable to do research.

2. Popularity Trends Over Time:

This line plot illustrates how song popularity varies over time. Analyzing trends in popularity can reveal periods of heightened or reduced audience engagement, aiding in strategic decision-making for artists and producers.

```
# Popularity Trends Over Time
ggplot(spotify_songs, aes(x = track_album_release_date, y = track_popularity)) +
  geom_line(color = "skyblue") +
  labs(title = "Popularity Trends Over Time",
       x = "Release Date",
       y = "Song Popularity") +
  theme_minimal()
```



The majority of the timeline maintains a popularity range from 0 to 80. However, in the year 2020, there is a distinct shift, and song popularity extends to the full range of 0 to 100. This deviation suggests a noteworthy change in the distribution of popular songs during 2020, indicating a potential surge in highly popular releases or a shift in audience preferences.

3. 15 of the most popular songs from the dataset were:

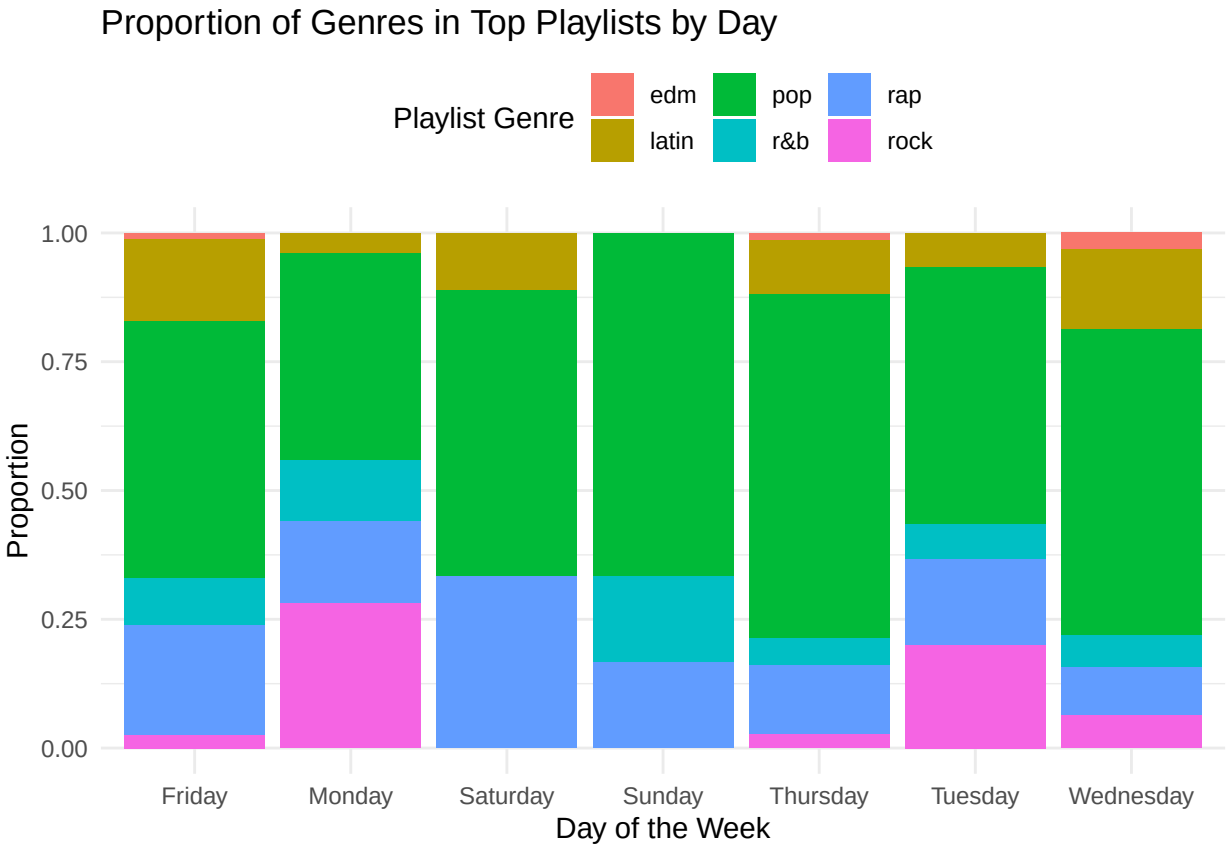
```
# Sort the dataset by popularity
top_15_popular_songs <- head(spotify_songs[order(-spotify_songs$track_popularity), ], 15)

# Displaying the 10 most popular songs
top_15_popular_songs[, c("track_name", "track_artist", "track_popularity")]
```

```
## # A tibble: 15 x 3
##    track_name                 track_artist        track_popularity
##    <chr>                      <chr>                          <dbl>
##  1 Dance Monkey               Tones and I                      100
##  2 ROXANNE                    Arizona Zervas                    99
##  3 Tusa                       KAROL G                           98
##  4 Memories                   Maroon 5                          98
##  5 Blinding Lights            The Weeknd                        98
##  6 Circles                    Post Malone                       98
##  7 The Box                    Roddy Ricch                       98
##  8 everything i wanted        Billie Eilish                     97
##  9 Don't Start Now            Dua Lipa                          97
## 10 Falling                    Trevor Daniel                     97
## 11 RITMO (Bad Boys For Life) The Black Eyed Peas               96
## 12 bad guy                    Billie Eilish                     95
## 13 Yummy                      Justin Bieber                     95
## 14 Ride It                    Regard                            94
## 15 Someone You Loved          Lewis Capaldi                     94
```

4. Proportion of Genres in Top Playlists by Day:

```
## `summarise()` has grouped output by 'release_day'. You can override using the
## `.groups` argument.
```

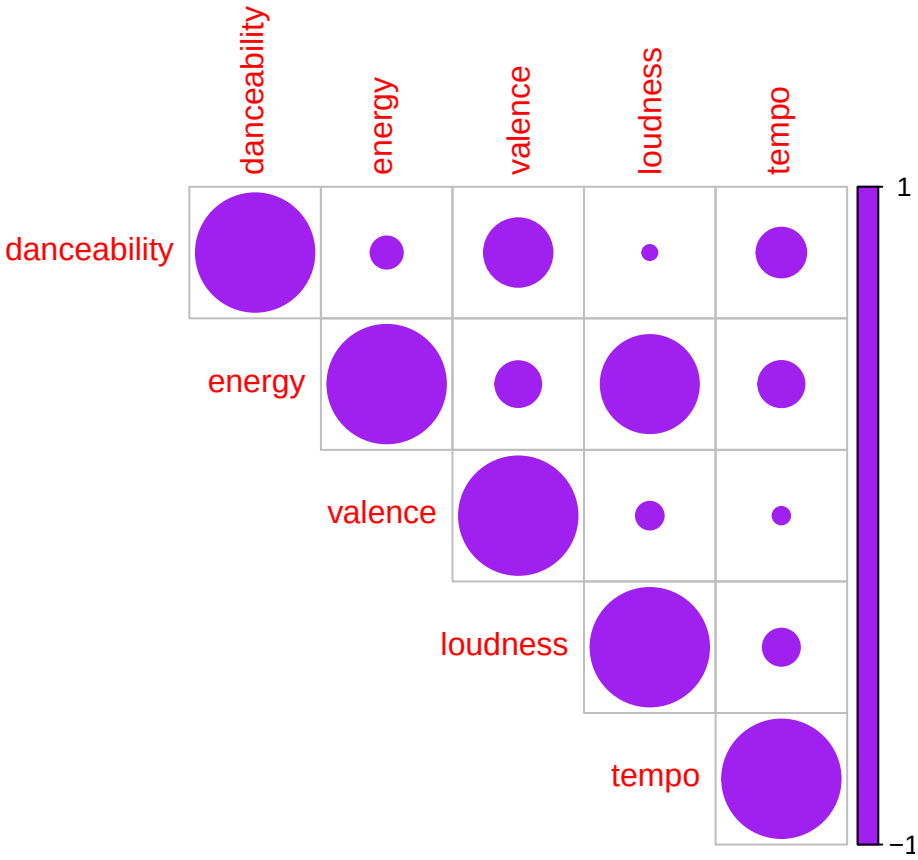


The distribution of releases is similar across most days of the week.
Notable differences are observed on Saturdays and Sundays.
Saturday has the highest proportion of rap releases compared to other genres.
Rock releases are absent on both Saturday and Sunday.

5. Correlation Heatmap

The correlation heatmap provides insights into the relationships between musical features. It helps identify strong correlations (positive or negative) between variables, guiding further analysis on influential musical attributes.

```
# Correlation Heatmap
cor_matrix <- cor(spotify_songs[, c("danceability", "energy", "valence", "loudness", "tempo")])
corrplot::corrplot(cor_matrix, method = "circle", type = "upper", col = "purple")
```



Analyzing this model we find that there are connections between the different characteristics related to the music data. Hence, we can say that different features also influence other features in the data set. We can use this understanding to create a model for Songs recommendation using these Features.

RECOMMENDATION MODEL CREATION

1. Fuction Creation

I have implement a cosine similarity function to quantify the similarity between songs based on selected audio features and using the collaborative filtering approach. *Working: Let's say a user wants to get recommendations based on a particular song preference the Model will synthesize the dataset to find similar songs for the user*

```
# Function to calculate cosine similarity
cosine_similarity <- function(x, y) {
  sum(x * y) / (sqrt(sum(x^2)) * sqrt(sum(y^2)))
}
```

2. Getting Similar songs together

This part of the model introduces a custom R function, get_similar_songs, designed for music recommendation based on audio features. The function utilizes a subset of relevant features from the Spotify dataset, calculates cosine similarity against a specified song, and returns information about the most similar songs. This modular approach facilitates the integration of music recommendation functionality into broader projects, enhancing the efficiency and clarity of collaborative filtering processes.

```
# Function to get similar songs
get_similar_songs <- function(song_id, n = 5) {
  song_features <- spotify_songs[, c("danceability", "energy", "key", "loudness",
        "mode", "speechiness", "acousticness", "instrumentalness", "liveness", "valence", "tempo")]

  # Extracting features for the given song
  query_features <- song_features[spotify_songs$track_id == song_id, ]

  # Calculating the cosine similarity
  similarity_scores <- apply(song_features, 1, function(x) cosine_similarity(x, query_features))

  # Getting indices of most similar songs
  similar_song_indices <- order(similarity_scores, decreasing = TRUE)[1:(n + 1)]

  return(spotify_songs[similar_song_indices, c("track_id", "track_name", "track_artist")])
}
```

3. Testing our model on a random song

In this section, a random song is selected for testing the model. The seed is set for reproducibility, and the sample function is used to pick a random song from the dataset. The selected song is then used as the query to find and display similar songs.

```
# Selecting a random song for testing
set.seed(123)

random_song_index <- sample(1:nrow(spotify_songs), 1)
random_song_id <- spotify_songs$track_id[random_song_index]

# Get similar songs for the randomly selected song
similar_songs <- get_similar_songs(random_song_id, n = 5)

# Displaying similar songs for the random song
cat("Randomly Selected Song:\n")
```

```
## Randomly Selected Song:
```

```
print(spotify_songs[spotify_songs$track_id == random_song_id, c("track_name", "track_artist")])
```

```
## # A tibble: 1 x 2
##   track_name   track_artist
##   <chr>        <chr>
## 1 Disconnected Thilo
```

```
cat("\nSimilar Songs:\n")
```

```
##
## Similar Songs:
```

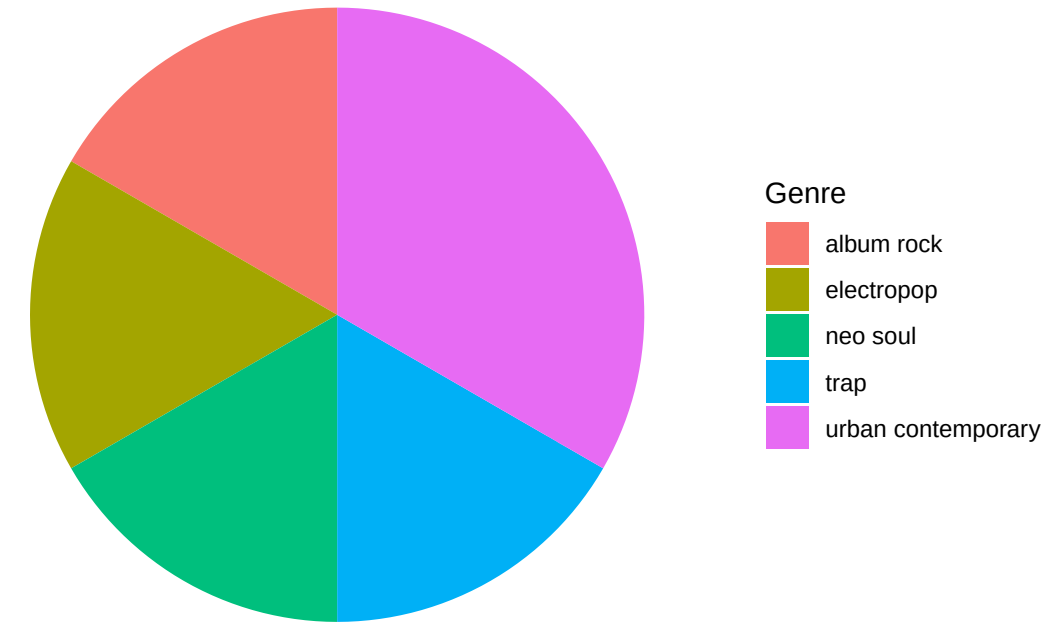
```
# Displaying information about similar songs with the first one
print(similar_songs[2:6, c("track_id", "track_name", "track_artist")])
```

```
## # A tibble: 5 x 3
##   track_id               track_name                                  track_artist
##   <chr>                  <chr>                                       <chr>
## 1 0nyrltZrQGAJMBZc1bYvuQ "Get Up Offa That Thing"                     James Brown
## 2 51wQovDO0hf05pkZYvu1GI "On the Road Again - Live"                   Willie Nels~
## 3 7CAhxeaTYXn1qWNRkZl9ad "Everybody's Talkin' - From \"Midnight Co~ Harry Nilss~
## 4 4ziiAuFyXiY1HNSLGM1xv8 "DOLLAR"                                     Becky G
## 5 6tgEc2O1uFHcZDKPoo6PC8 "ALREADY"                                    Beyoncé
```

Printing the Genre Distribution for the Recommendations:

```
# 1. Extract track IDs of similar songs
similar_song_ids <- similar_songs$track_id[1:6]
genres_data <- spotify_songs[spotify_songs$track_id %in% similar_song_ids,
                             c("track_id", "playlist_subgenre")]
genre_distribution_pie_chart <- ggplot(genres_data, aes(x = "", fill = playlist_subgenre)) +
  geom_bar(width = 1, stat = "count") +
  coord_polar(theta = "y") +
  labs(title = "Genre Distribution of the Songs",
       fill = "Genre") +
  theme_minimal() +
  theme(axis.text = element_blank(),axis.title = element_blank(),panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 18)) +
  scale_fill_discrete(name = "Genre")

print(genre_distribution_pie_chart)
```



The Pie Chart verifies that the Model does not simply work to choose similar genre based songs as recommendations however uses the cosine similarity function on the different features of the songs to recommend similar choices.

Conclusion

In conclusion, the analysis of the Spotify Songs data set has provided valuable insights into the dynamics of music popularity and song characteristics. Through exploratory data analysis, we have uncovered trends, distributions, and top-performing songs, contributing to a better understanding of the factors influencing song success. The development and testing of our song recommending model further demonstrated its potential to enhance music discovery by providing diverse recommendations beyond genre boundaries. The genre distribution of recommendations highlighted the versatility and effectiveness of the model in offering varied song choices.

This project serves as a valuable resource for artists, producers, and streaming platforms but also contributes to the broader field of data-driven music analysis. The combination of visualizations, statistical analysis, and machine learning techniques has helped me to navigate through the rich data set and draw meaningful conclusions.

In future developments, the song recommending system can be enriched by incorporating additional song features, such as sentiment analysis of lyrics or audio characteristics, to offer more refined and diverse recommendations. Implementing real-time model updates will enable the system to adapt dynamically to evolving user preferences and changing music trends.These enhancements collectively aim to elevate the recommending system's performance and user experience.
