# Spotify Data Analysis and Music Recommendation System

Mitansh Maheshwari

2024-10-25

## Project Overview

In an age where music streaming has become a predominant mode of consuming music, understanding user listening habits is crucial for enhancing user experience and satisfaction. This project focuses on analyzing Spotify streaming data to develop a music recommendation system that personalizes music suggestions based on individual user preferences. By harnessing user listening history and audio feature data, the project aims to identify patterns in music consumption, ultimately creating a tailored recommendation engine that enhances the user's music discovery experience.

### Problem Statement

As the landscape of music streaming evolves, understanding user behavior and preferences has become increasingly essential for enhancing user engagement and satisfaction. Despite the vast amounts of data generated by listeners on platforms like Spotify, there is a lack of comprehensive analysis that explores the nuanced relationships between listening habits, emotional responses, and audio features.

The primary challenges this project aims to address include:

- **Understanding Listening Patterns**: Users exhibit diverse listening behaviors influenced by factors such as time of day, day of the week, and personal mood. Identifying and analyzing these patterns is crucial for tailoring music experiences to individual preferences.
- **Correlation with Audio Features**: While audio features like danceability, energy, and loudness are known to influence user engagement, there is insufficient research on how these attributes correlate with users' emotional states and overall listening satisfaction.
- **Enhancing Music Discovery**: Users often struggle to discover new music that aligns with their tastes, leading to a potential decline in engagement. A deeper understanding of user preferences and listening contexts is necessary to improve music recommendation systems.
- **User Retention Insights**: Understanding what drives user retention on music streaming platforms is essential for developing strategies that keep users engaged and satisfied. Identifying key factors that contribute to prolonged usage can inform platform improvements and content recommendations.

By addressing these challenges, this project seeks to provide valuable insights into user behavior, enhance music discovery through personalized recommendations, and ultimately improve user satisfaction and retention on the Spotify platform.

### Purpose

The primary purpose of this project is to conduct a comprehensive analysis of music listening behaviors and patterns among users. By leveraging extensive streaming data, this project aims to achieve the following objectives:

1. **In-depth Analysis of Listening Behavior**: Explore and analyze various factors influencing how users interact with music on the Spotify platform. This includes examining trends in listening habits over time, identifying the impact of different audio features (such as danceability, energy, and loudness), and understanding the role of contextual elements like time of day and day of the week in shaping user preferences.
2. **Enhancing Music Discovery**: Utilize insights gained from the data analysis to facilitate personalized music recommendations. By understanding user engagement patterns and preferences, the project will provide tailored suggestions that help users discover new tracks and artists aligned with their tastes, thereby enriching their overall music experience.
3. **Informing User Engagement Strategies**: By analyzing listening trends and user interactions, the project aims to derive actionable insights that can enhance user engagement. Understanding what drives user retention and satisfaction will allow for more effective marketing strategies and improvements to the Spotify platform, ultimately leading to a more enjoyable listening experience for users.
4. **Developing a Recommendation System**: As a secondary goal, this project will also focus on creating a recommendation system that suggests songs based on a user's mood and listening history. While this is a smaller component of the overall project, it aims to personalize the user experience further by adapting recommendations to individual preferences.

### Objective

The objectives of this project are designed to guide the analysis of music listening behaviors and inform the development of personalized experiences for users. Specifically, we aim to:

1. **Examine User Listening Patterns**: Analyze streaming data to identify trends and variations in user listening habits over time, focusing on factors such as time of day, day of the week, and genre preferences.
2. **Investigate Audio Features**: Explore how specific audio characteristics—such as danceability, energy, valence, loudness, and instrumentalness—affect user engagement and listening time. This will help to understand the attributes that contribute to users' enjoyment of music.
3. **Analyze Mood Correlations**: Assess how users' moods correlate with their music choices. By examining the emotional resonance of songs, we can identify patterns that can be leveraged to enhance music discovery.

4. **Identify High-Engagement Tracks and Artists**: Determine which tracks and artists exhibit the highest levels of user engagement. Understanding these metrics will provide valuable insights into popular content and user preferences.

5. **Develop a Mood-Based Recommendation System**: Create a recommendation system that suggests songs based on a user's mood and listening history, thereby improving music discovery and enhancing the overall user experience.

6. **Evaluate User Retention Factors**: Investigate the factors contributing to user retention based on their listening habits. By identifying key elements that keep users engaged with the platform, we can inform strategies for improving user satisfaction.

### Methodology

1. **Data Collection**: The data for this project was collected from Spotify through the Spotify API. I requested my listening history for the past year, which included key fields such as `endTime`, `artistName`, `trackName`, and `msPlayed`. Additionally, audio features like `danceability`, `energy`, `valence`, `loudness`, and `instrumentalness` were obtained using the Spotify API to enrich the analysis.

2. **Data Wrangling**: The collected data underwent a rigorous cleaning and wrangling process to ensure its quality and usability. This involved handling missing values by imputing them with mean values, ensuring appropriate data types for each variable, and filtering out irrelevant entries. The data was then combined into a unified dataset, which serves as the foundation for further analysis.

3. **Feature Engineering**: To gain deeper insights into listening habits, several new features were engineered from the dataset. This included extracting the day of the week, categorizing the time of day into periods, and grouping consecutive listening events into sessions. Additionally, a listener classification system was developed to identify users as Super Listeners, Moderate Listeners, Light Listeners, and Occasional Listeners based on their streaming frequency.

4. **Data Analysis**: The analysis was conducted using statistical methods and visualizations to uncover patterns in listening behavior. The relationships between different variables, such as audio features and user engagement, were explored to answer key research questions. The insights gained from this analysis aimed to inform the development of personalized music recommendations.

5. **Visualization and Reporting**: Visualizations were created to present the findings from the data analysis in an easily interpretable format. These included graphs and charts that illustrate user listening patterns, engagement metrics, and correlations between audio features and user behavior. The reporting process aimed to communicate the insights effectively, providing a comprehensive overview of the project findings and their implications for music discovery.

6. **Recommendation System Development**: a recommendation system was designed to suggest tracks based on user preferences. This system utilized the insights from the data analysis phase to provide tailored recommendations that align with users' musical tastes and moods.

### Conclusion

In summary, this project aims to provide a comprehensive analysis of music listening behaviors by leveraging Spotify's rich dataset. By examining user preferences and engagement patterns, we seek to uncover valuable insights that can enhance the music discovery experience for users. The methodology outlined above lays the groundwork for understanding the intricate relationships between various factors influencing listening habits, ultimately informing the development of a

recommendation system tailored to individual user needs. As we move forward, the findings from this analysis will contribute to a deeper understanding of how music impacts our daily lives and how personalized recommendations can foster a more engaging listening experience.

---

## Part 1: Data Source and Collection

The dataset utilized in this analysis is derived from Spotify's streaming history, which captures user listening behaviors over time. The data was personally requested from Spotify and is provided in two separate JSON files. Below is a detailed description of the data source, how it was collected, and the structure of the data files.

### Source of the Data

The primary source of the data is Spotify, a widely used music streaming service that offers users access to millions of tracks, albums, and playlists. Spotify allows users to track their listening history, which includes essential details about the songs they have played, the artists they have listened to, and the duration of each listening session.

### Data Collection Method

To gather the necessary data, I submitted a request through the Spotify website for my personal listening history. The data arrived within a week and contains the following key attributes:

1. **Received Data**:
   - The JSON files contain the following attributes for each listening event:
     - endTime
     - artistName
     - trackName
     - msPlayed
2. **Requested Audio Features:**:
   - In addition to the basic listening history data, I utilized the Spotify API to request specific audio features for each track. These features provide deeper insights into the characteristics of the music and include:
     - Danceability
     - Energy

  - Valence
  - Loudness
  - Instrumentalness

This dataset, derived from both direct requests to Spotify and the Spotify API, allows for a comprehensive examination of user listening history and music characteristics. The rich structure of the data makes it a valuable resource for building recommendation systems and understanding user engagement patterns within the Spotify ecosystem.

## Part 2: Data Cleaning/Wrangling

In this section, we focus on the crucial task of data cleaning and wrangling to prepare the dataset for analysis. This process ensures the data's quality and consistency, allowing for accurate insights to be drawn from it. The data cleaning and wrangling were carried out in three key steps:

### Step 1: Combining Streaming History Data from Multiple Sources

In order to analyze listening habits and build a music recommendation system, it's essential to have a comprehensive dataset. The data used in this project comes from two separate JSON files (Spotify limits the amount of streams per file), each containing records of tracks played over different periods. To ensure seamless analysis, we need to combine these files into a single dataset.

In this step, we will:

- Read the two JSON files containing streaming history.
- Combine the datasets into a unified structure.
- Save the combined dataset as a new JSON file for further analysis.
- Convert the JSON file to CSV as per Project 1 Requirements.

This approach simplifies the data processing workflow and ensures that we have all relevant information in one place.

```
# Load necessary libraries without startup messages
suppressPackageStartupMessages({
  library(jsonlite)
  library(dplyr)
  library(rmarkdown)
})

# Function to read JSON file and return as a list
```

```r
read_json_file <- function(file_path) {
  fromJSON(file_path)
}

# Read the two JSON files
data_0 <- read_json_file("StreamingHistory_music_0.json")
data_1 <- read_json_file("StreamingHistory_music_1.json")

# Combine the two datasets
combined_data <- rbind(data_0, data_1)

# Write the combined data to a CSV file
write.csv(combined_data, file = "combined_data.csv", row.names = FALSE)

# Display the first 5 rows of the combined data
head(combined_data, 5)
```

```
##            endTime   artistName                          trackName msPlayed
## 1 2023-10-11 23:26 Taylor Swift                       New Romantics   198489
## 2 2023-10-12 00:49 Taylor Swift                       New Romantics    31864
## 3 2023-10-12 00:54 Taylor Swift The Story Of Us (Taylor's Version)   267653
## 4 2023-10-12 00:58 Taylor Swift                            End Game   244826
## 5 2023-10-12 00:59 Taylor Swift         All You Had To Do Was Stay    34933
##   danceability energy valence loudness instrumentalness
## 1        0.653  0.848   0.724   -5.936         4.24e-06
## 2        0.653  0.848   0.724   -5.936         4.24e-06
## 3        0.516  0.784   0.552   -2.636         0.00e+00
## 4        0.649  0.589   0.151   -6.237         0.00e+00
## 5        0.589  0.713   0.540   -5.614         0.00e+00
```

**Step 2: Data Cleaning Process**

In this step, we will clean the combined dataset to ensure its quality and usability for analysis. This process involves several key actions:

1. **Handling Missing Values**: We will impute missing values in the columns for danceability, energy, valence, loudness, and instrumentalness using the mean of their respective columns. This will help maintain the integrity of our dataset.

2. **Ensuring Appropriate Data Types**: We will convert the `endTime` column to a proper date-time format and ensure that `msPlayed` is treated as a numeric value.

3. **Filtering Irrelevant Rows**: We will filter out tracks that were played for less than a specified duration (e.g., 10 seconds) to focus on meaningful listening experiences.

After executing these steps, we will display the first five rows of the cleaned dataset to confirm the cleaning process's effectiveness.

```r
# Load necessary libraries
library(dplyr)

# Step 2: Read the combined data from the CSV
combined_data <- read.csv("combined_data.csv")

# Data Cleaning Process

# 1. Handle Missing Values by imputing with the mean
combined_data$danceability[is.na(combined_data$danceability)] <- mean(combined_data$danceability, na.rm = TRUE)
combined_data$energy[is.na(combined_data$energy)] <- mean(combined_data$energy, na.rm = TRUE)
combined_data$valence[is.na(combined_data$valence)] <- mean(combined_data$valence, na.rm = TRUE)
combined_data$loudness[is.na(combined_data$loudness)] <- mean(combined_data$loudness, na.rm = TRUE)
combined_data$instrumentalness[is.na(combined_data$instrumentalness)] <- mean(combined_data$instrumentalness, na.rm = TRUE)

# 2. Ensure appropriate data types
combined_data$endTime <- as.POSIXct(combined_data$endTime)
combined_data$msPlayed <- as.numeric(combined_data$msPlayed)

# 3. Filter out irrelevant rows (example: keep tracks played for more than 10 seconds)
combined_data <- combined_data %>% filter(msPlayed > 10000)

head(combined_data, 5)
```

```
##               endTime   artistName                          trackName msPlayed
## 1 2023-10-11 23:26:00 Taylor Swift                      New Romantics   198489
## 2 2023-10-12 00:49:00 Taylor Swift                      New Romantics    31864
## 3 2023-10-12 00:54:00 Taylor Swift The Story Of Us (Taylor's Version)   267653
## 4 2023-10-12 00:58:00 Taylor Swift                           End Game   244826
```

```
## 5 2023-10-12 00:59:00 Taylor Swift         All You Had To Do Was Stay    34933
##   danceability energy valence loudness instrumentalness
## 1        0.653  0.848   0.724   -5.936         4.24e-06
## 2        0.653  0.848   0.724   -5.936         4.24e-06
## 3        0.516  0.784   0.552   -2.636         0.00e+00
## 4        0.649  0.589   0.151   -6.237         0.00e+00
## 5        0.589  0.713   0.540   -5.614         0.00e+00
```

**Step 3: Feature Engineering**

In this step, we will enhance our dataset by creating new features that can provide deeper insights into listening habits and preferences. Feature engineering is a critical process in data analysis and machine learning, as it can significantly improve the performance of models by introducing informative attributes. Below are the newly created variables/features:

1. **Day of the Week**:
   - **Variable Name**: `day_of_week`
   - **Description**: This variable extracts the day of the week from the `endTime` column, allowing us to analyze patterns in listening behavior based on weekdays versus weekends. By knowing the day of the week, we can identify trends in music consumption, such as whether users listen more on weekends compared to weekdays.
   - **Values**: This variable can take on values such as "Monday," "Tuesday," "Wednesday," "Thursday," "Friday," "Saturday," and "Sunday."
2. **Time of Day**
   - **Variable Name:** `time_of_day`
   - **Description:** This variable categorizes the listening times into distinct periods of the day: Night, Morning, Afternoon, Evening, and Late Night. Understanding when users listen to music can help in creating personalized recommendations based on peak listening times.
   - **Values:**
     - Night: From 12:00 AM to 5:00 AM
     - Morning: From 6:00 AM to 11:00 AM
     - Afternoon: From 12:00 PM to 5:00 PM
     - Evening: From 6:00 PM to 9:00 PM
     - Late Night: From 10:00 PM to 11:59 PM
3. **Listening Session ID**:
   - **Variable Name**: `session_id`
   - **Description**: This variable groups consecutive listening events into sessions based on a defined time window (in this case, 30 minutes). By identifying listening sessions, we can better understand user engagement and the depth of their listening experiences. This feature helps to analyze how users consume music over a period rather than individual tracks.

- **Values**: Each session will be represented by a unique ID, with consecutive events grouped together. For example, if a user listens to several songs in one session, they will have the same session_id until there is a break of more than 30 minutes.

4. **Listener Categories**

- **Variable Name:** `listener_category`
- **Description:** This variable categorizes artists based on their play counts into four distinct groups: Super Listeners, Moderate Listeners, Light Listeners, and Occasional Listeners. This categorization helps to identify user engagement with different artists, providing insights into listening preferences and facilitating personalized recommendations.
- **Values:**
  - **Super Listeners:** More than 100 streams, indicating a high level of engagement and strong preference for the artist.
  - **Moderate Listeners:** Between 50 and 100 streams, reflecting considerable interest and occasional engagement with the artist.
  - **Light Listeners:** Between 10 and 50 streams, suggesting that the artist has some appeal but is not a primary choice for the listener.
  - **Occasional Listeners:** Fewer than 10 streams, indicating minimal engagement with the artist, possibly only sampled once or twice.

```r
# Feature Engineering Implementation
# 1. Day of the Week
combined_data$day_of_week <- weekdays(combined_data$endTime)

# 2. Time of Day
combined_data$time_of_day <- cut(as.numeric(format(combined_data$endTime, "%H")),
                                 breaks = c(-1, 5, 11, 17, 21, 24),
                                 labels = c("Night", "Morning", "Afternoon", "Evening", "LateNight"),
                                 right = TRUE)

# 3. Listening Session ID
combined_data <- combined_data %>%
  arrange(endTime) %>%
  mutate(session_id = cumsum(c(1, diff(endTime) > 1800)))

# 4. Listener Categories
artist_play_counts <- combined_data %>%
  group_by(artistName) %>%
  summarise(play_count = n()) %>%
  ungroup()

# Categorize listeners based on play counts
artist_play_counts <- artist_play_counts %>%
```

```r
  mutate(listener_category = case_when(
    play_count > 100 ~ "Super Listeners",
    play_count >= 50 & play_count <= 100 ~ "Moderate Listeners",
    play_count >= 10 & play_count < 50 ~ "Light Listeners",
    play_count < 10 ~ "Occasional Listeners"
  ))

# Join the listener category back to the combined data
combined_data <- combined_data %>%
  left_join(select(artist_play_counts, artistName, listener_category), by = "artistName")

# Display the updated dataset
head(combined_data, 5)
```

```
##              endTime   artistName                          trackName msPlayed
## 1 2023-10-11 23:26:00 Taylor Swift                      New Romantics   198489
## 2 2023-10-12 00:49:00 Taylor Swift                      New Romantics    31864
## 3 2023-10-12 00:54:00 Taylor Swift The Story Of Us (Taylor's Version)   267653
## 4 2023-10-12 00:58:00 Taylor Swift                           End Game   244826
## 5 2023-10-12 00:59:00 Taylor Swift         All You Had To Do Was Stay    34933
##   danceability energy valence loudness instrumentalness day_of_week time_of_day
## 1        0.653  0.848   0.724   -5.936         4.24e-06   Wednesday   LateNight
## 2        0.653  0.848   0.724   -5.936         4.24e-06    Thursday       Night
## 3        0.516  0.784   0.552   -2.636         0.00e+00    Thursday       Night
## 4        0.649  0.589   0.151   -6.237         0.00e+00    Thursday       Night
## 5        0.589  0.713   0.540   -5.614         0.00e+00    Thursday       Night
##   session_id listener_category
## 1          1   Super Listeners
## 2          2   Super Listeners
## 3          2   Super Listeners
## 4          2   Super Listeners
## 5          2   Super Listeners
```

## Part 3: Exploratory Data Analysis(EDA)

### Descriptive Statistics

This section of the analysis focuses on generating descriptive statistics for key audio features within the Spotify dataset. Descriptive statistics provide a summary of the central tendency, dispersion, and shape of the dataset's distribution, which is essential in exploratory data analysis (EDA) as it helps us understand the basic properties of the data we are working with.

Descriptive statistics play a crucial role in exploratory data analysis by providing insights into the characteristics of the data. They help identify trends, patterns, and anomalies within the dataset, allowing for informed decision-making in further analysis. Understanding the central tendency and variability of audio features such as danceability, energy, valence, loudness, and instrumentalness can lead to better insights into listener preferences and behavior.

```
# Load necessary libraries
library(dplyr)
library(knitr)

# Assuming the dataset is loaded as 'combined_data'

# Custom function to calculate mode
get_mode <- function(v) {
  uniq_v <- na.omit(unique(v))
  uniq_v[which.max(tabulate(match(v, uniq_v)))]
}

# Calculate mean, median, mode, and standard deviation for key columns
summary_stats <- combined_data %>%
  summarise(
    danceability_mean = mean(danceability, na.rm = TRUE),
    danceability_median = median(danceability, na.rm = TRUE),
    danceability_mode = get_mode(danceability),
    danceability_sd = sd(danceability, na.rm = TRUE),

    energy_mean = mean(energy, na.rm = TRUE),
    energy_median = median(energy, na.rm = TRUE),
    energy_mode = get_mode(energy),
    energy_sd = sd(energy, na.rm = TRUE),

    valence_mean = mean(valence, na.rm = TRUE),
```

```
    valence_median = median(valence, na.rm = TRUE),
    valence_mode = get_mode(valence),
    valence_sd = sd(valence, na.rm = TRUE),

    loudness_mean = mean(loudness, na.rm = TRUE),
    loudness_median = median(loudness, na.rm = TRUE),
    loudness_mode = get_mode(loudness),
    loudness_sd = sd(loudness, na.rm = TRUE),

    instrumentalness_mean = mean(instrumentalness, na.rm = TRUE),
    instrumentalness_median = median(instrumentalness, na.rm = TRUE),
    instrumentalness_mode = get_mode(instrumentalness),
    instrumentalness_sd = sd(instrumentalness, na.rm = TRUE)
  )

# Reformat the data to display factors as rows and statistics as columns
reformatted_stats <- data.frame(
  Factor = c("Danceability", "Energy", "Valence", "Loudness", "Instrumentalness"),
  Mean = c(summary_stats$danceability_mean, summary_stats$energy_mean, summary_stats$valence_mean,
           summary_stats$loudness_mean, summary_stats$instrumentalness_mean),
  Median = c(summary_stats$danceability_median, summary_stats$energy_median, summary_stats$valence_median,
             summary_stats$loudness_median, summary_stats$instrumentalness_median),
  Mode = c(summary_stats$danceability_mode, summary_stats$energy_mode, summary_stats$valence_mode,
           summary_stats$loudness_mode, summary_stats$instrumentalness_mode),
  SD = c(summary_stats$danceability_sd, summary_stats$energy_sd, summary_stats$valence_sd,
         summary_stats$loudness_sd, summary_stats$instrumentalness_sd)
)

# Display the summary table
kable(reformatted_stats, caption = "Descriptive Statistics for Key Audio Features")
```

Table 1: Descriptive Statistics for Key Audio Features

| Factor | Mean | Median | Mode | SD |
|---|---|---|---|---|
| Danceability | 0.6320172 | 0.632 | 0.632 | 0.1179142 |
| Energy | 0.6711195 | 0.683 | 0.800 | 0.1629690 |

| Factor | Mean | Median | Mode | SD |
|---|---|---|---|---|
| Valence | 0.5109430 | 0.535 | 0.629 | 0.2280846 |
| Loudness | -6.3737535 | -5.956 | -5.956 | 2.4428684 |
| Instrumentalness | 0.0066805 | 0.000 | 0.000 | 0.0511618 |

### Categorical Analysis

This section focuses on analyzing categorical data within the Spotify dataset. By examining categories such as listener category, day of the week, and time of day, we can gain insights into listening behaviors and preferences across different segments.

Categorical analysis is crucial in exploratory data analysis as it helps uncover patterns and trends in the data that may not be evident from numerical analysis alone. By examining how different listener categories, days of the week, and times of day affect listening behaviors, we can identify preferences and optimize recommendations or marketing strategies. Understanding these factors can aid in tailoring content to better meet the needs of different listener segments, ultimately enhancing user engagement and satisfaction.

```r
# Load necessary libraries
library(dplyr)
library(knitr)

# Check the most common categories in 'listener_category'
# Count unique artists per listener category
categorical_analysis <- combined_data %>%
  distinct(artistName, listener_category) %>%
  group_by(listener_category) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

# Display the listener category analysis
kable(categorical_analysis, caption = "Count of Unique Artists by Listener Category")
```

Table 2: Count of Unique Artists by Listener Category

| listener_category | count |
|---|---|
| Occasional Listeners | 273 |
| Light Listeners | 80 |

| listener_category | count |
|---|---|
| Super Listeners | 11 |
| Moderate Listeners | 10 |

```r
# Count occurrences for day_of_week
day_of_week_analysis <- combined_data %>%
  group_by(day_of_week) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

# Display the day of week analysis
kable(day_of_week_analysis, caption = "Total Listening Counts by Day of Week")
```

Table 3: Total Listening Counts by Day of Week

| day_of_week | count |
|---|---|
| Monday | 1546 |
| Thursday | 1465 |
| Tuesday | 1350 |
| Wednesday | 1195 |
| Friday | 1170 |
| Saturday | 1102 |
| Sunday | 586 |

```r
# Count occurrences for time_of_day
time_of_day_analysis <- combined_data %>%
  group_by(time_of_day) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

# Display the time of day analysis
kable(time_of_day_analysis, caption = "Total Listening Counts by Time of Day")
```

Table 4: Total Listening Counts by Time of Day

| time_of_day | count |
|---|---|
| Evening | 3174 |
| Night | 2552 |
| Afternoon | 1164 |
| LateNight | 906 |
| Morning | 618 |

### Daily and Weekly Trends Analysis

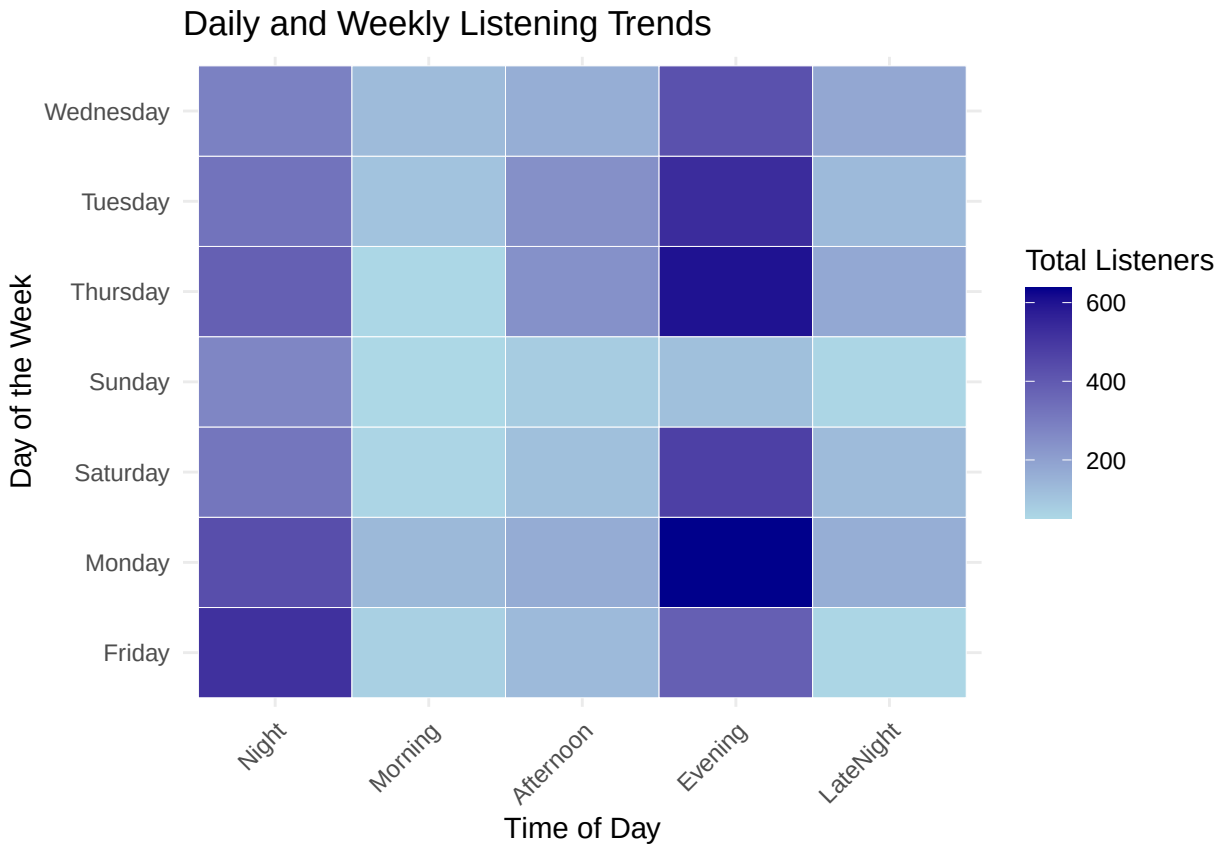
This section visualizes daily and weekly listening trends in the Spotify dataset using a heatmap. The heatmap effectively displays the number of listens across different days of the week and times of the day, helping identify peak listening times.

Analyzing daily and weekly trends is crucial for understanding user behavior and engagement patterns. By identifying peak listening times and preferred days, insights can be gained that inform marketing strategies, content scheduling, and recommendations. Heatmaps are particularly effective for this kind of analysis as they allow for quick visual assessments of large datasets, making it easier to spot trends and anomalies. This understanding can help in tailoring user experiences and maximizing listener satisfaction.

```
# Load necessary libraries
library(dplyr)
library(ggplot2)
library(reshape2) # For reshaping data

# Daily and Weekly Trends Analysis
daily_trends <- combined_data %>%
  group_by(day_of_week, time_of_day) %>%
  summarise(total_listens = n(), .groups = 'drop') %>%
  arrange(desc(total_listens))

# Reshape data for heatmap
daily_trends_reshaped <- dcast(daily_trends, day_of_week ~ time_of_day, value.var = "total_listens", fill = 0)

# Create heatmap
ggplot(melt(daily_trends_reshaped), aes(x = variable, y = day_of_week, fill = value)) +
  geom_tile(color = "white") +
```

```
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(x = "Time of Day", y = "Day of the Week", fill = "Total Listeners",
       title = "Daily and Weekly Listening Trends") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Using day_of_week as id variables
```



### Session Analysis

This section provides a detailed analysis of user listening sessions, summarizing key statistics such as the number of tracks listened to, session start and end times, and session durations. Session analysis is vital for understanding user engagement patterns, helping identify how long users are listening to music and how many tracks they typically listen to in one sitting. This information can inform strategies for content delivery, user experience enhancements, and marketing efforts. By analyzing session durations and track counts, insights can be gained into user preferences, peak usage times, and overall engagement trends. These insights can guide the development of features aimed at increasing user retention and satisfaction, such as personalized recommendations and session-based promotions.

```r
# Load necessary libraries
library(dplyr)
library(knitr)

# Session Analysis
session_analysis <- combined_data %>%
  group_by(session_id) %>%
  summarise(
    track_count = n(),                                # Number of tracks in each session
    session_start_time = min(endTime),                # Start time of session
    session_end_time = max(endTime),                  # End time of session
    session_duration = as.numeric(difftime(session_end_time, session_start_time, units = "mins"))  # Duration in minutes
  )

# Summarise overall session statistics
session_summary <- session_analysis %>%
  summarise(
    average_tracks_per_session = mean(track_count),
    average_session_duration = mean(session_duration),
    total_sessions = n(),
    max_session_duration = max(session_duration),
    min_session_duration = min(session_duration)
  )

# Display summary statistics with kable for better formatting
kable(session_summary, caption = "Overall Session Analysis", format = "markdown")
```

Table 5: Overall Session Analysis

| average_tracks_per_session | average_session_duration | total_sessions | max_session_duration | min_session_duration |
|---|---|---|---|---|
| 7.998099 | 25.31559 | 1052 | 258 | 0 |

**Frequency of Listening Analysis**

This analysis summarizes the most frequently listened-to tracks and also highlights the top 5 artists based on the total number of listens. Including the analysis of the top 5 most listened-to artists alongside track frequency analysis provides a broader understanding of user preferences and trends in music consumption. This information is valuable for identifying popular artists and tailoring recommendations, marketing strategies, and playlists to enhance user engagement. It can also inform partnerships and promotional opportunities with artists, driving user satisfaction and retention.

```
# Load necessary libraries
library(dplyr)      # For data manipulation
library(knitr)      # For rendering tables

# Frequency of Listening Analysis
track_frequency <- combined_data %>%
  group_by(trackName, artistName) %>%  # Group by track name and artist
  summarise(listen_count = n(), .groups = 'drop') %>%  # Count listens and drop grouping
  arrange(desc(listen_count)) %>%  # Sort by listen count in descending order
  top_n(10, listen_count)  # Show only the top 10 tracks by listen count

# Display frequency of listening with kable for better formatting
kable(track_frequency, caption = "Top Tracks by Frequency of Listening", format = "markdown",
      col.names = c("Track Name", "Artist Name", "Listen Count"))  # Custom column names
```

Table 6: Top Tracks by Frequency of Listening

| Track Name | Artist Name | Listen Count |
|---|---|---|
| Unwritten | Natasha Bedingfield | 157 |
| Diet Mountain Dew | Lana Del Rey | 121 |
| Is It Over Now? (Taylor's Version) (From The Vault) | Taylor Swift | 115 |
| Murder On The Dancefloor | Sophie Ellis-Bextor | 114 |
| Thank You | Dido | 100 |

| Track Name | Artist Name | Listen Count |
|---|---|---|
| Espresso | Sabrina Carpenter | 99 |
| How You Get The Girl (Taylor's Version) | Taylor Swift | 98 |
| Guilty as Sin? | Taylor Swift | 92 |
| All Too Well (Taylor's Version) | Taylor Swift | 87 |
| Say Don't Go (Taylor's Version) (From The Vault) | Taylor Swift | 82 |

```r
# Top 5 Most Listened-To Artists
artist_frequency <- combined_data %>%
  group_by(artistName) %>%  # Group by artist
  summarise(artist_listen_count = n(), .groups = 'drop') %>%  # Count listens per artist
  arrange(desc(artist_listen_count)) %>%  # Sort by listen count in descending order
  top_n(5, artist_listen_count)  # Show only the top 5 artists by listen count

# Display top 5 artists with kable for better formatting
kable(artist_frequency, caption = "Top 5 Most Listened-To Artists", format = "markdown",
      col.names = c("Artist Name", "Listen Count"))  # Custom column names
```

Table 7: Top 5 Most Listened-To Artists

| Artist Name | Listen Count |
|---|---|
| Taylor Swift | 4115 |
| Lana Del Rey | 226 |
| Sabrina Carpenter | 208 |
| One Direction | 180 |
| Natasha Bedingfield | 177 |

### Investigation of Audio Features

In this section, we explore the relationships between various audio features and their impact on user engagement through a correlation analysis and a statistical test (ANOVA).

**Correlation Matrix of Audio Features** The **correlation matrix** provides insights into how different audio features relate to one another. The features analyzed include:

- **danceability**: A measure of how suitable a track is for dancing, ranging from 0 to 1.
- **energy**: Represents the intensity and activity of a track, also scaled between 0 and 1.
- **valence**: Indicates the musical positiveness of a track, where higher values signify more positive feelings.
- **msPlayed**: The total milliseconds a track was played, representing user engagement.

The values in the correlation matrix show the degree of linear relationship between pairs of features:

- **Interpretation**:
  - A value close to **1** indicates a strong positive correlation, meaning that as one feature increases, the other tends to increase as well.
  - A value close to **-1** indicates a strong negative correlation, where an increase in one feature corresponds to a decrease in the other.
  - Values around **0** indicate little to no correlation.

**ANOVA Results for Energy Levels by Day of the Week** The **ANOVA (Analysis of Variance)** results help us understand whether there are statistically significant differences in energy levels across different days of the week. The output table includes the following columns:

| Column | Description |
|---|---|
| **Df** | Degrees of freedom, representing the number of levels in the factor (day_of_week) minus one for the between-group variability and the residuals. |
| **Sum Sq** | The total sum of squares for each factor, indicating the variability attributed to that factor. |
| **Mean Sq** | Mean square values, calculated as the sum of squares divided by the respective degrees of freedom. It gives an estimate of variance associated with each factor. |
| **F value** | The F-statistic, which is the ratio of the variance between groups to the variance within groups. A higher F value suggests a greater degree of variability between the group means relative to the variability within groups. |
| **Pr(>F)** | The p-value associated with the F-statistic. A lower p-value (typically $< 0.05$) indicates a statistically significant difference between the means of the groups being compared. |

- **Interpretation**:
  - The ANOVA results indicate that there are significant differences in energy levels across different days of the week (p-value = 0.0058), suggesting that the day of the week may influence the energy level of the music listened to.

**General Description of the Code** The provided R code performs an investigation into audio features within the dataset. It consists of the following steps:

1. **Correlation Analysis**:
   - A correlation matrix is calculated for the selected audio features: `danceability`, `energy`, `valence`, and `msPlayed`.
   - The correlation matrix is displayed in a neatly formatted table for easier interpretation.
2. **Statistical Testing (ANOVA)**:
   - An ANOVA test is conducted to determine if there are significant differences in energy levels based on the day of the week.
   - The results are summarized in a table that details the degrees of freedom, sum of squares, mean squares, F-statistic, and p-value.

Overall, this analysis helps in understanding the interplay between different audio features and user engagement, providing insights that can inform further music recommendations or analyses.

```r
# Load necessary libraries
suppressMessages(library(dplyr))
suppressMessages(library(knitr))
suppressMessages(library(kableExtra))

# 4. Investigation of Audio Features

## 4.1 Correlation Analysis
# Calculate the correlation matrix for selected features
correlation_matrix <- cor(combined_data[, c("danceability", "energy", "valence", "msPlayed")], use = "complete.obs")

# Create a kable for the correlation matrix
### Correlation Matrix:
kable(correlation_matrix, digits = 2, caption = "Correlation Matrix of Audio Features")
```

Table 9: Correlation Matrix of Audio Features

| | danceability | energy | valence | msPlayed |
|---|---|---|---|---|
| danceability | 1.00 | 0.10 | 0.42 | -0.09 |
| energy | 0.10 | 1.00 | 0.48 | -0.07 |
| valence | 0.42 | 0.48 | 1.00 | -0.13 |
| msPlayed | -0.09 | -0.07 | -0.13 | 1.00 |

```r
## 4.2 Statistical Tests
# Perform ANOVA to see if energy levels differ by day of the week
anova_results <- aov(energy ~ day_of_week, data = combined_data)

# Extract ANOVA summary table and convert to a data frame for kable
anova_summary <- as.data.frame(summary(anova_results)[[1]])

### ANOVA Results:
# Create a kable for the ANOVA results
kable(anova_summary, caption = "ANOVA Results for Energy Levels by Day of the Week")
```

Table 10: ANOVA Results for Energy Levels by Day of the Week

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| day_of_week | 6 | 0.4823289 | 0.0803881 | 3.031174 | 0.0058171 |
| Residuals | 8407 | 222.9575702 | 0.0265205 | NA | NA |

### Mood Labeling and Audio Features

This analysis focuses on the relationship between audio features and mood categorization of songs in the Spotify dataset. By examining features such as valence, energy, and danceability, we aim to classify each song into specific mood categories. This helps in understanding how musical elements influence listeners' emotions and preferences.

1. Mood Labeling

We begin by initializing mood labels for each song in the dataset with a default value of "Unknown Mood." The following criteria are then applied based on three key audio features:

- **Valence**: Indicates the musical positiveness conveyed by a track. High valence corresponds to a happy or positive mood, while low valence suggests a more negative or sad feeling.
- **Energy**: Reflects the intensity and activity level of a song. High energy is associated with fast, loud, and rhythmic tracks, while low energy is linked to calm or quiet music.
- **Danceability**: Represents how suitable a track is for dancing, influenced by tempo, rhythm stability, and overall musicality.

2. Mood Grouping

Once mood labels are assigned, we aggregate the data by mood category to calculate the average values of danceability, energy, and valence for each mood group. This provides insights into the characteristics of songs associated with different moods.

3. Heatmap Visualization

To further explore the relationships between audio features, a heatmap is generated to visualize the correlations among the various audio features (danceability, energy, valence, loudness, instrumentalness). This helps to identify patterns and trends in the dataset.

**Mood Categories and Descriptions** The following table summarizes the mood categories along with their descriptions and average audio feature values:

| Mood | Description | Average Valence | Average Energy | Average Danceability |
|---|---|---|---|---|
| Happy, Energetic, Uplifting | Joyful and lively songs | $> 0.5$ | $> 0.7$ | $> 0.5$ |
| Sad, Calm, Relaxing | Melancholic and soothing tracks | $< 0.3$ | $< 0.3$ | $< 0.3$ |
| Content, Peaceful, Reflective | Reflective and serene songs | $> 0.5$ | $< 0.5$ | 0.3 - 0.7 |
| Anxious, Intense, Agitated | High-energy but tense songs | $< 0.3$ | $> 0.7$ | 0.3 - 0.7 |
| Confident, Empowered, Motivational | Uplifting and empowering tracks | $> 0.5$ | $> 0.7$ | $> 0.5$ |
| Romantic, Warm, Nostalgic | Sentimental and warm songs | $> 0.5$ | $< 0.5$ | $> 0.7$ |
| Fun, Playful, Lively | Upbeat and entertaining tracks | $> 0.4$ | $> 0.4$ | $> 0.4$ |
| Somber, Reflective, Wistful | Thought-provoking and reflective songs | $< 0.3$ | $< 0.5$ | $> 0.5$ |
| Mellow, Serene, Dreamy | Calm and gentle tracks | $< 0.4$ | $< 0.4$ | $< 0.4$ |
| Relaxed, Positive, Easy-going | Laid-back and feel-good songs | $> 0.4$ | $< 0.4$ | $< 0.5$ |
| Gritty, Raw, Intense | Raw and intense tracks | $< 0.3$ | $> 0.7$ | $< 0.3$ |

Sources -

1. https://nycdatascience.com/blog/student-works/spotify-metrics-do-you-know-them/ 2, https://towardsdatascience.com/what-makes-a-song-likeable-dbfdb7abe404

The analysis demonstrates a clear relationship between audio features and mood categorization in the Spotify dataset. By understanding these relationships, we can develop better recommendation systems that align with users' emotional states and enhance their listening experiences.

```
# Load necessary libraries
library(dplyr)
library(kableExtra)
```

```r
library(ggplot2)
library(reshape2)

# 5.1 Refined Mood Labeling
# Initialize the mood_labels vector
mood_labels <- character(nrow(combined_data))

# Assign moods based on the revised conditions informed by music research
for (i in 1:nrow(combined_data)) {
  row <- combined_data[i, ]

  # Assigning moods based on research insights
  if (row['valence'] > 0.6 && row['energy'] > 0.7 && row['danceability'] > 0.5) {
    mood_labels[i] <- 'Happy, Energetic, Uplifting'  # Bright, fast-paced songs
  } else if (row['valence'] < 0.3 && row['energy'] < 0.4 && row['danceability'] < 0.4) {
    mood_labels[i] <- 'Sad, Calm, Relaxing'  # Slow, somber songs
  } else if (row['valence'] >= 0.4 && row['valence'] < 0.6 && row['energy'] < 0.6 && row['danceability'] >= 0.4) {
    mood_labels[i] <- 'Content, Peaceful, Reflective'  # Moderate energy, soothing
  } else if (row['valence'] < 0.3 && row['energy'] >= 0.5 && row['danceability'] >= 0.4) {
    mood_labels[i] <- 'Anxious, Intense, Agitated'  # Intense, faster songs
  } else if (row['valence'] >= 0.5 && row['energy'] >= 0.6 && row['danceability'] >= 0.4) {
    mood_labels[i] <- 'Confident, Empowered, Motivational'  # Uplifting, strong beats
  } else if (row['valence'] >= 0.5 && row['energy'] < 0.5 && row['danceability'] > 0.5) {
    mood_labels[i] <- 'Romantic, Warm, Nostalgic'  # Slow, sentimental
  } else if (row['valence'] >= 0.4 && row['energy'] >= 0.4 && row['danceability'] > 0.5) {
    mood_labels[i] <- 'Fun, Playful, Lively'  # Upbeat and catchy
  } else if (row['valence'] < 0.4 && row['energy'] < 0.5 && row['danceability'] >= 0.4) {
    mood_labels[i] <- 'Somber, Reflective, Wistful'  # Thoughtful and slow
  } else if (row['valence'] < 0.4 && row['energy'] < 0.4 && row['danceability'] < 0.4) {
    mood_labels[i] <- 'Mellow, Serene, Dreamy'  # Chill and laid-back
  } else if (row['valence'] >= 0.3 && row['energy'] < 0.6 && row['danceability'] < 0.5) {
    mood_labels[i] <- 'Relaxed, Positive, Easy-going'  # Calm but cheerful
  } else if (row['valence'] < 0.3 && row['energy'] >= 0.4 && row['danceability'] < 0.4) {
    mood_labels[i] <- 'Gritty, Raw, Intense'  # Aggressive or edgy
  } else {
    mood_labels[i] <- 'Balanced, Neutral, Easy-going'  # General category to cover remaining cases
  }
```

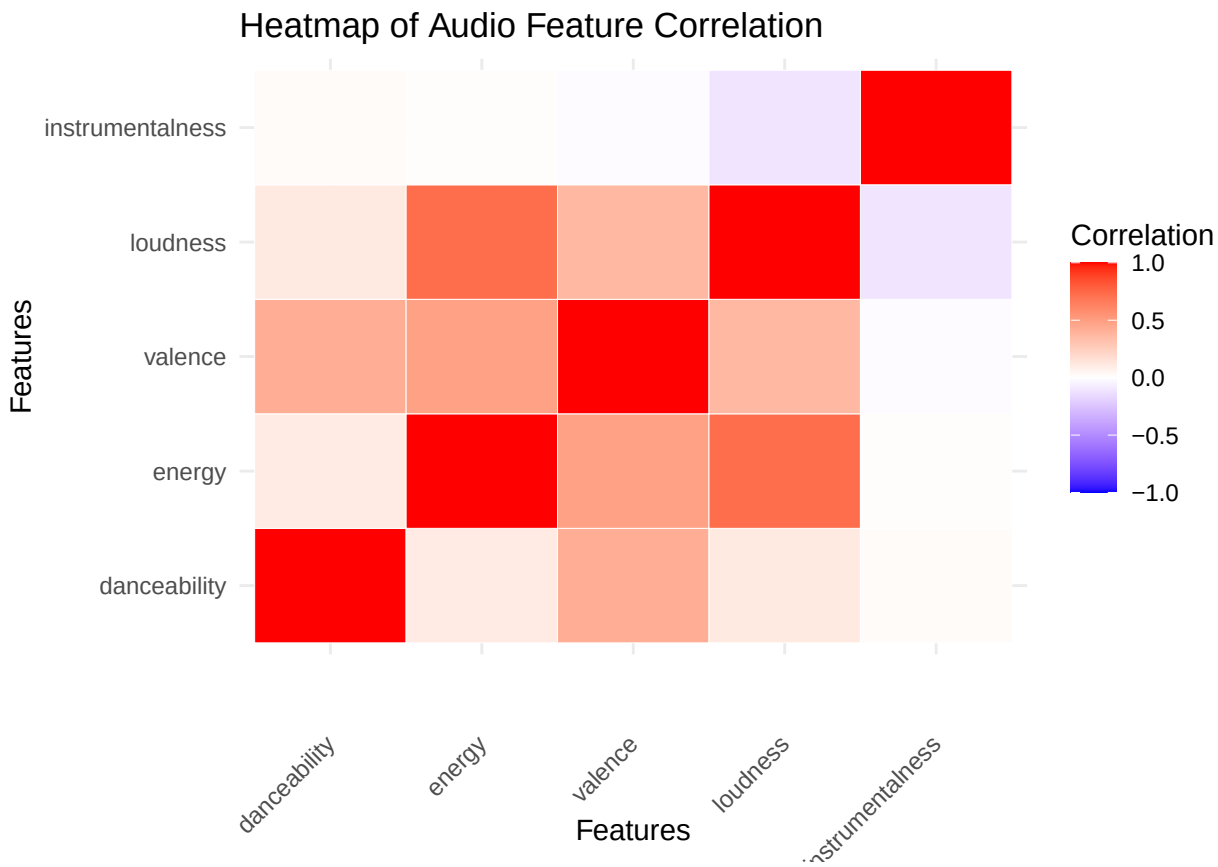
```
}

# Add mood labels to the dataset
combined_data$mood <- mood_labels

# 5.2 Mood Grouping
# Group songs based on mood and calculate mean values for audio features
mood_grouped <- combined_data %>%
  group_by(mood) %>%
  summarise(
    avg_danceability = mean(danceability, na.rm = TRUE),
    avg_energy = mean(energy, na.rm = TRUE),
    avg_valence = mean(valence, na.rm = TRUE),
    count = n()
  )

# Display mood grouping results with kable
mood_grouped %>%
  kable(digits = 2, caption = "Average Audio Features by Mood Group") %>%
  kable_styling(full_width = F, position = "left", bootstrap_options = c("striped", "hover", "condensed"))
```

Table 12: Average Audio Features by Mood Group

| mood | avg_danceability | avg_energy | avg_valence | count |
|---|---|---|---|---|
| Anxious, Intense, Agitated | 0.58 | 0.65 | 0.21 | 1209 |
| Balanced, Neutral, Easy-going | 0.56 | 0.68 | 0.39 | 746 |
| Confident, Empowered, Motivational | 0.66 | 0.73 | 0.61 | 2068 |
| Content, Peaceful, Reflective | 0.64 | 0.48 | 0.48 | 541 |
| Fun, Playful, Lively | 0.66 | 0.67 | 0.57 | 886 |
| Gritty, Raw, Intense | 0.36 | 0.57 | 0.18 | 85 |
| Happy, Energetic, Uplifting | 0.69 | 0.82 | 0.76 | 1926 |
| Mellow, Serene, Dreamy | 0.38 | 0.28 | 0.33 | 7 |
| Relaxed, Positive, Easy-going | 0.37 | 0.47 | 0.50 | 74 |
| Romantic, Warm, Nostalgic | 0.65 | 0.43 | 0.78 | 45 |
| Sad, Calm, Relaxing | 0.33 | 0.27 | 0.16 | 35 |

| | | | | |
|---|---|---|---|---|
| Somber, Reflective, Wistful | 0.60 | 0.39 | 0.21 | 792 |

```r
# 5.3 Heatmap of Audio Feature Correlation
# Calculate the correlation matrix of audio features
audio_features <- combined_data %>% select(danceability, energy, valence, loudness, instrumentalness)
correlation_matrix <- cor(audio_features, use = "complete.obs")

# Melt the correlation matrix for heatmap
correlation_melted <- melt(correlation_matrix)

# Create the heatmap
ggplot(correlation_melted, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0, limit = c(-1, 1), name = "Correlation") +
  theme_minimal() +
  labs(title = "Heatmap of Audio Feature Correlation", x = "Features", y = "Features") +
  theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1))
```




```
# 5.4 Display the First Few Rows of the Data
combined_data %>%
  head(5) %>%
  kable(caption = "First Five Rows of Combined Data", digits = 2) %>%
  kable_styling(full_width = F, position = "left", bootstrap_options = c("striped", "hover", "condensed"))
```

Table 13: First Five Rows of Combined Data

| endTime | artistName | trackName | msPlayed | danceability | energy | valence | loudness | instrumentalness | day_of_week | time_o |
|---|---|---|---|---|---|---|---|---|---|---|
| 2023-10-11 23:26:00 | Taylor Swift | New Romantics | 198489 | 0.65 | 0.85 | 0.72 | -5.94 | 0 | Wednesday | LateNi |
| 2023-10-12 00:49:00 | Taylor Swift | New Romantics | 31864 | 0.65 | 0.85 | 0.72 | -5.94 | 0 | Thursday | Night |
| 2023-10-12 00:54:00 | Taylor Swift | The Story Of Us (Taylor's Version) | 267653 | 0.52 | 0.78 | 0.55 | -2.64 | 0 | Thursday | Night |
| 2023-10-12 00:58:00 | Taylor Swift | End Game | 244826 | 0.65 | 0.59 | 0.15 | -6.24 | 0 | Thursday | Night |
| 2023-10-12 00:59:00 | Taylor Swift | All You Had To Do Was Stay | 34933 | 0.59 | 0.71 | 0.54 | -5.61 | 0 | Thursday | Night |

```r
output_path <- "combined_data.csv"
write.csv(combined_data, file = output_path, row.names = FALSE)
```

## Part 4: Data Visualization

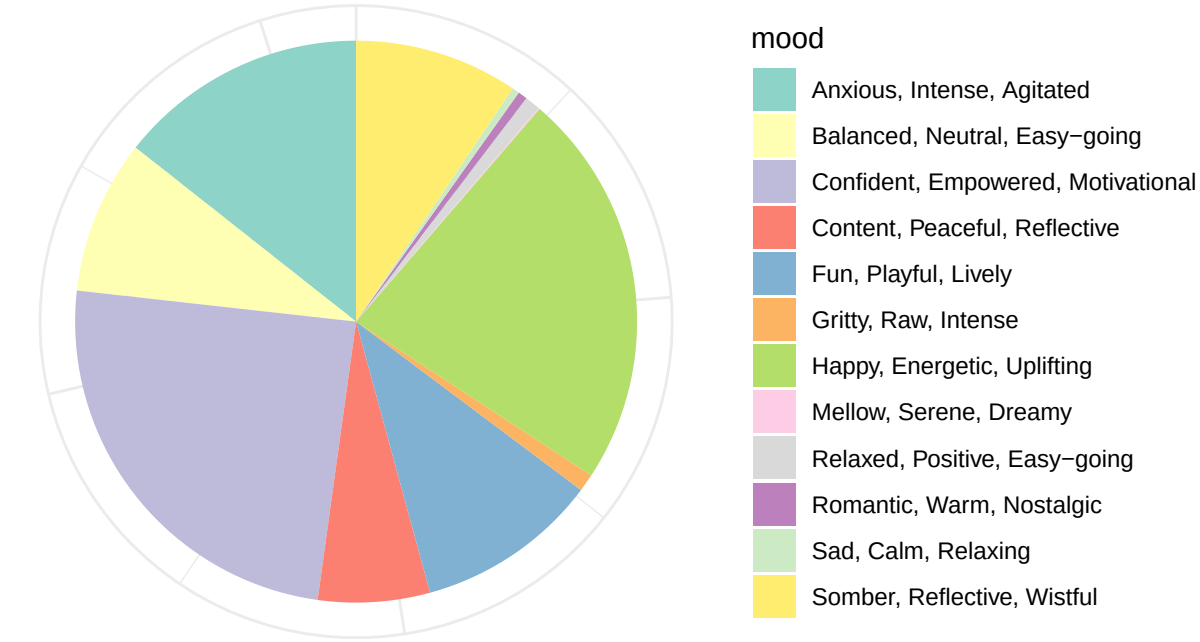
### Mood Distribution in Spotify Listening History

The pie chart represents the distribution of different mood categories in the Spotify listening history. Each slice corresponds to a specific mood label, showing how frequently tracks associated with that mood were played. The chart provides a clear visual of the relative proportions of each mood, offering an overview of the listener's preferences across various emotional tones.

The insights drawn from this chart can help identify which mood categories dominate the listening habits. A larger slice for a particular mood indicates a strong preference for that emotional tone, such as Happy, energetic, or uplifting music. Conversely, a more balanced distribution suggests a diverse taste across multiple moods. Understanding this distribution is useful for building mood-based music recommendations or exploring how the listener's preferences shift over different periods.

```r
# Create a summarized data frame for mood counts
mood_distribution <- combined_data %>%
  group_by(mood) %>%
  summarise(count = n())

# Generate a pie chart
ggplot(mood_distribution, aes(x = "", y = count, fill = mood)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +
  labs(
    title = "Mood Distribution in Spotify Listening History",
    x = NULL,
    y = NULL
```

```
) +
theme_minimal() +
theme(axis.text.x = element_blank(), axis.ticks = element_blank()) +
scale_fill_brewer(palette = "Set3")
```



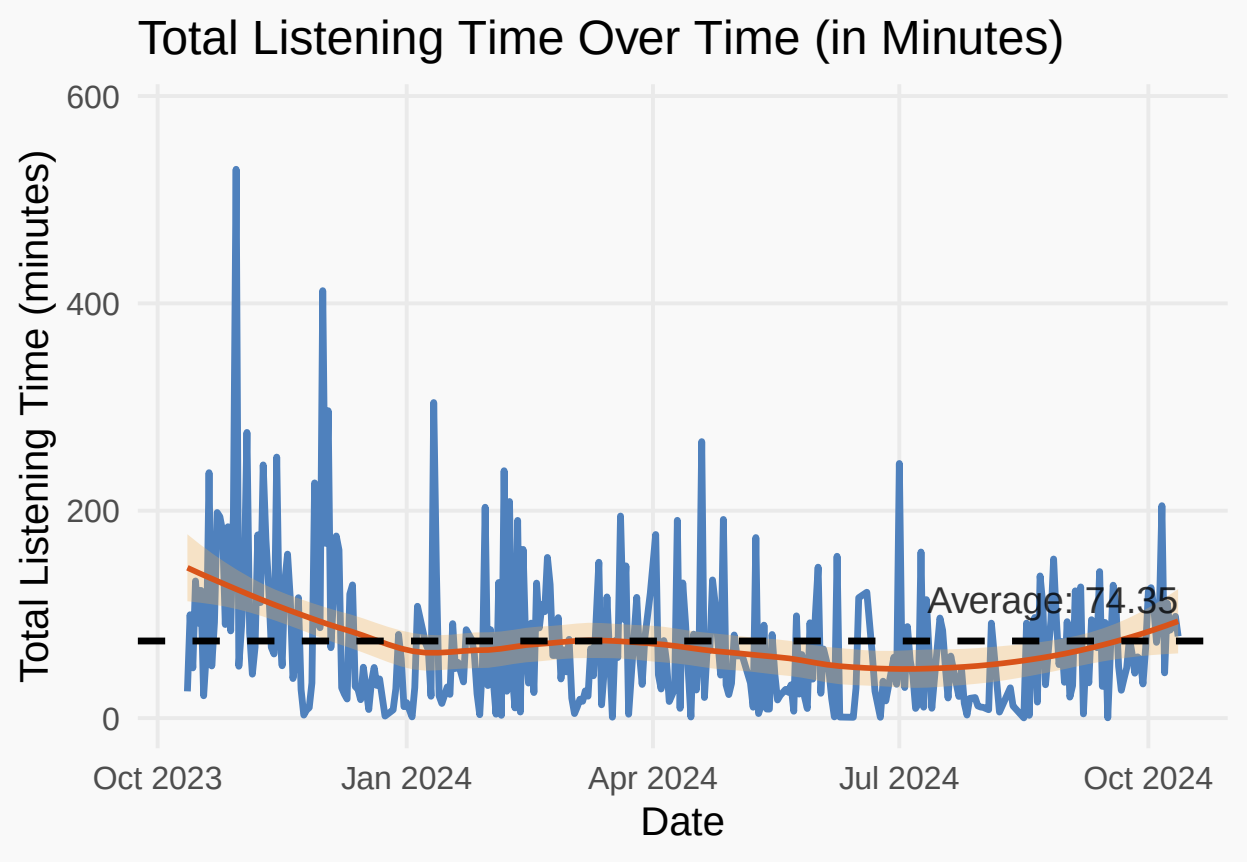
### Total Listening Time Over Time

The line graph visualizes the total listening time in minutes over a period, aggregated by day. Each point on the x-axis represents a specific date, while the y-axis indicates the total listening duration for that day. A smooth trend line has been added to highlight general patterns in listening habits, along with a horizontal dashed line to mark the average daily listening time. The graph's clean design, with a soft blue line and minimal background, enhances readability and focuses attention on key trends.

The insights from this graph help identify variations in daily listening behavior. Peaks in the graph suggest days with significantly higher engagement, possibly indicating special occasions or mood changes that led to more music consumption. Conversely, troughs might reflect less active listening periods. The smooth trend line allows for an easy observation of overall listening patterns, while the average line offers a benchmark to quickly assess how daily behavior compares to typical listening habits. This information can be useful for understanding shifts in music engagement over time, potentially uncovering seasonal or weekly trends.

```r
library(ggplot2)
library(dplyr)

# Aggregate data by day and convert msPlayed to minutes
daily_listening <- combined_data %>%
  mutate(endTime = as.Date(endTime)) %>%
  group_by(endTime) %>%
  summarise(Total_msPlayed = sum(msPlayed) / 60000) # Convert to minutes

# Calculate the average listening time
average_listening <- mean(daily_listening$Total_msPlayed)

# Determine the y-axis limits
y_limit <- c(0, max(daily_listening$Total_msPlayed) * 1.1)  # Set limit with a 10% increase for better view

# Create a unique and visually appealing line graph without points and adjusted scale
ggplot(daily_listening, aes(x = endTime, y = Total_msPlayed)) +
  geom_line(color = "#4F81BD", linewidth = 1.2) +  # Soft blue line
  geom_smooth(method = "loess", formula = y ~ x, span = 0.5, color = "#D95319",
              fill = "#F0AD4E", alpha = 0.3, se = TRUE) +  # Explicit formula
  geom_hline(yintercept = average_listening, linetype = "dashed", color = "black", linewidth = 1.2) +  # Average line
  labs(title = "Total Listening Time Over Time (in Minutes)",
       x = "Date", y = "Total Listening Time (minutes)") +
  annotate("text", x = max(daily_listening$endTime), y = average_listening + 40,  # Move annotation higher
           label = paste("Average:", round(average_listening, 2)),
           color = "black", hjust = 1, size = 5, alpha = 0.8) +  # Average annotation with transparency
```

```r
scale_y_continuous(limits = y_limit) +  # Set y-axis limits
theme_minimal(base_size = 15) +  # Increase base font size for readability
theme(panel.grid.major = element_line(color = "#EAEAEA"),  # Light grid lines
      panel.grid.minor = element_blank(),  # No minor grid lines
      plot.background = element_rect(fill = "#F9F9F9", color = NA),  # Soft background
      legend.position = "none")  # Remove legend if not needed
```

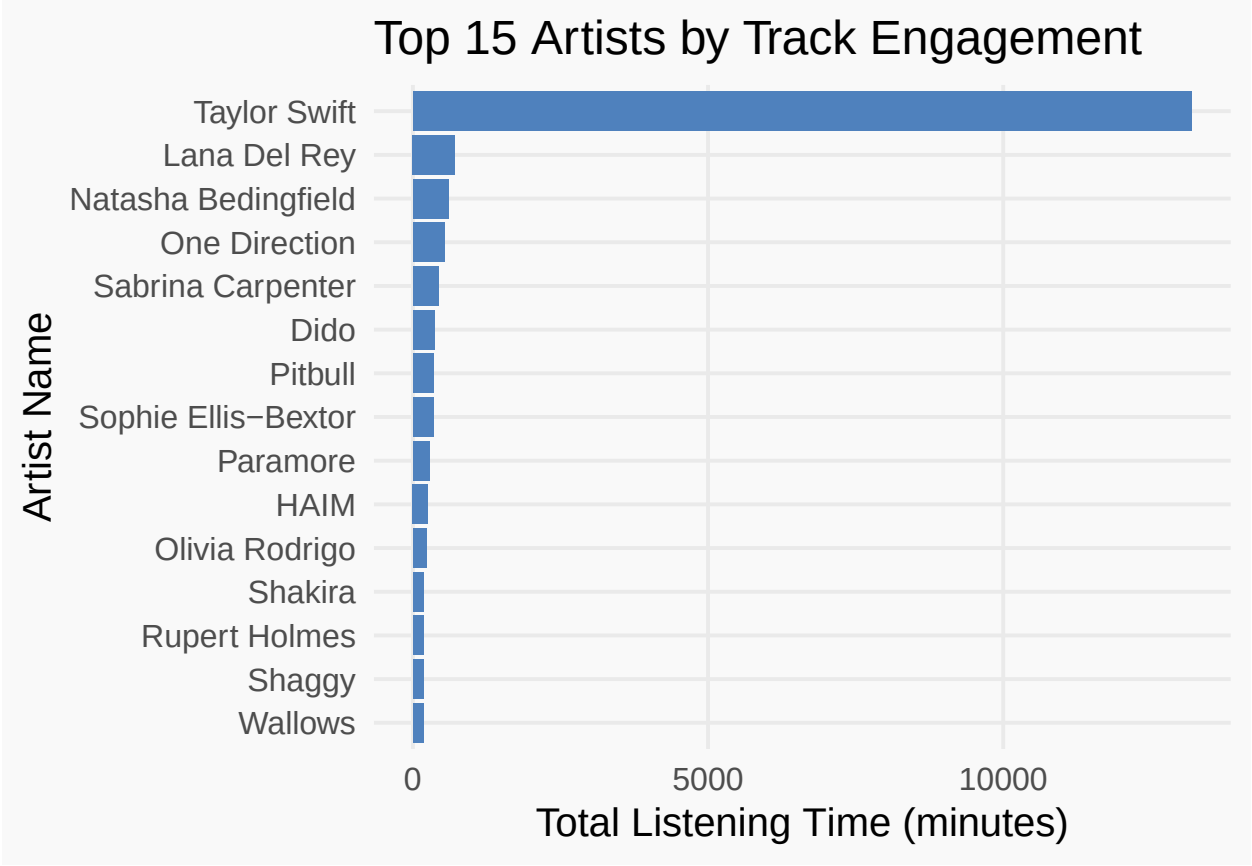


### Top 15 Artists by Track Engagement

The bar plot showcases the top 15 artists based on total listening time, measured in minutes. The x-axis represents the artist names, and the y-axis indicates the total engagement, with bars showing the cumulative time spent listening to each artist. The chart is oriented horizontally to improve readability, making it easier to compare engagement levels across different artists. A consistent soft blue color is used to maintain a clean and visually appealing design.

From this plot, we can gain insights into listening preferences by identifying the most frequently played artists. Artists with the longest bars represent those with the highest engagement, suggesting that their music resonates more or is preferred by the user. This information can reveal patterns in music taste, highlight favorite genres or artists, and help understand which musicians drive the most significant portion of listening activity. It can also serve as a basis for personalized recommendations or further exploration into why these particular artists stand out.

```r
library(ggplot2)
library(dplyr)

# Aggregate total listening time by artist and convert msPlayed to minutes
artist_engagement <- combined_data %>%
  group_by(artistName) %>%
  summarise(Total_msPlayed = sum(msPlayed) / 60000) %>%  # Convert to minutes
  arrange(desc(Total_msPlayed)) %>%  # Sort artists by total listening time
  slice_head(n = 15)  # Select top 15 artists

# Create the bar plot
ggplot(artist_engagement, aes(x = reorder(artistName, Total_msPlayed), y = Total_msPlayed)) +
  geom_bar(stat = "identity", fill = "#4F81BD") +  # Soft blue color for bars
  coord_flip() +  # Flip coordinates for better readability
  labs(title = "Top 15 Artists by Track Engagement",
       x = "Artist Name", y = "Total Listening Time (minutes)") +
  theme_minimal(base_size = 15) +  # Increase base font size for readability
  theme(panel.grid.major = element_line(color = "#EAEAEA"),  # Light grid lines
        panel.grid.minor = element_blank(),  # No minor grid lines
        plot.background = element_rect(fill = "#F9F9F9", color = NA),  # Soft background
        legend.position = "none")  # Remove legend if not needed
```




**Mood Correlations with Danceability, Energy, and Valence**

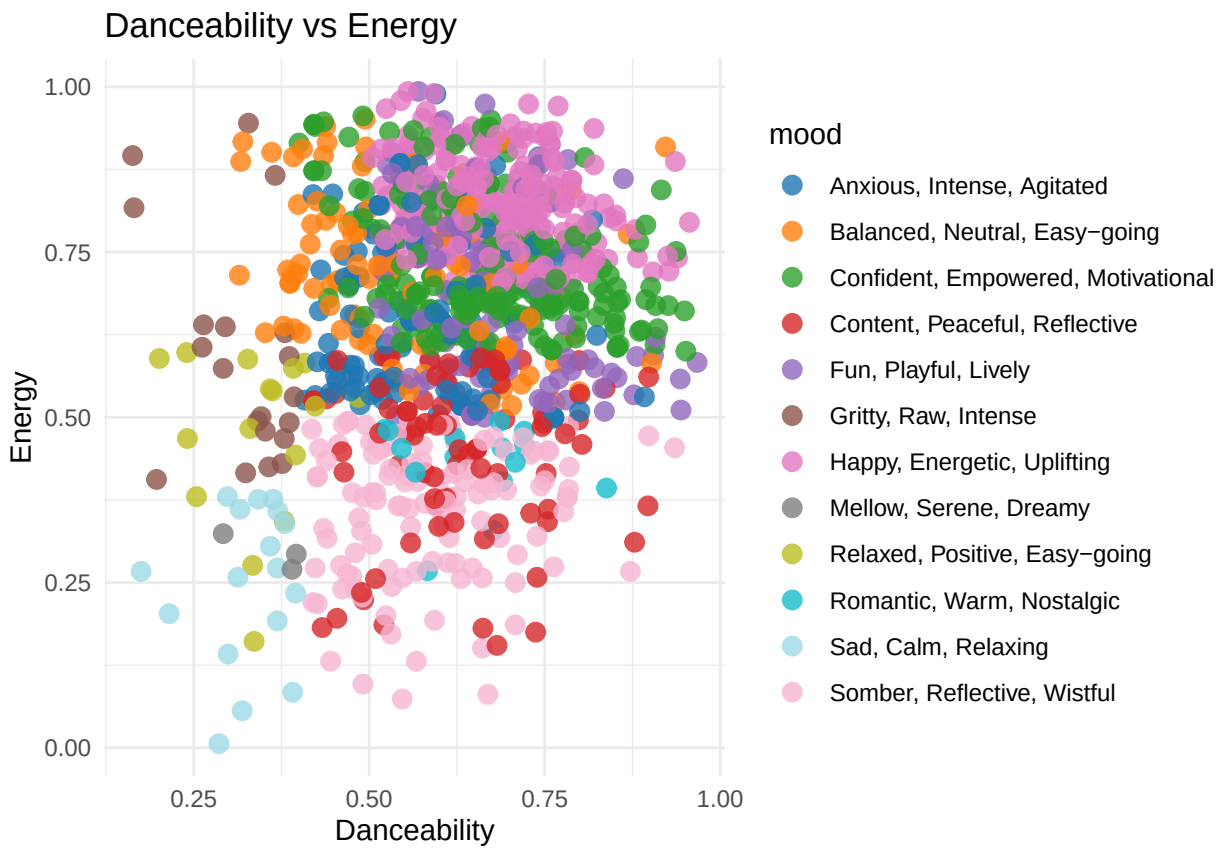
This analysis presents three 2D scatter plots that explore the relationships between the musical features of danceability, energy, and valence, while also considering the mood associated with each track. The first scatter plot illustrates the correlation between danceability and energy, providing insights into how these two features interact and affect the overall mood of the music. The second plot examines the relationship between danceability and valence, revealing how the rhythm and tempo of a track relate to its emotional tone. Lastly, the third plot analyzes energy against valence, further expanding our understanding of how the intensity and emotional quality of music influence listener mood. Each plot is color-coded according to mood categories, making it easier to identify patterns and trends within the dataset.

The scatter plots offer valuable insights into how specific musical attributes align with different moods. For example, tracks characterized by high danceability and energy may correlate with more upbeat and happy moods, while tracks with lower energy but higher valence might represent a more relaxed or melancholic state. The visual representation allows for easy identification of clusters where certain moods are concentrated, indicating a preference for particular musical qualities. This understanding can help in creating more tailored music recommendations based on mood, enhancing the listener's experience by aligning their emotional state with the most suitable tracks. Moreover, examining the relationships among these features could inform artists and producers about the musical elements that resonate most with audiences, guiding the creation of future music that better aligns with listeners' emotional needs and preferences.

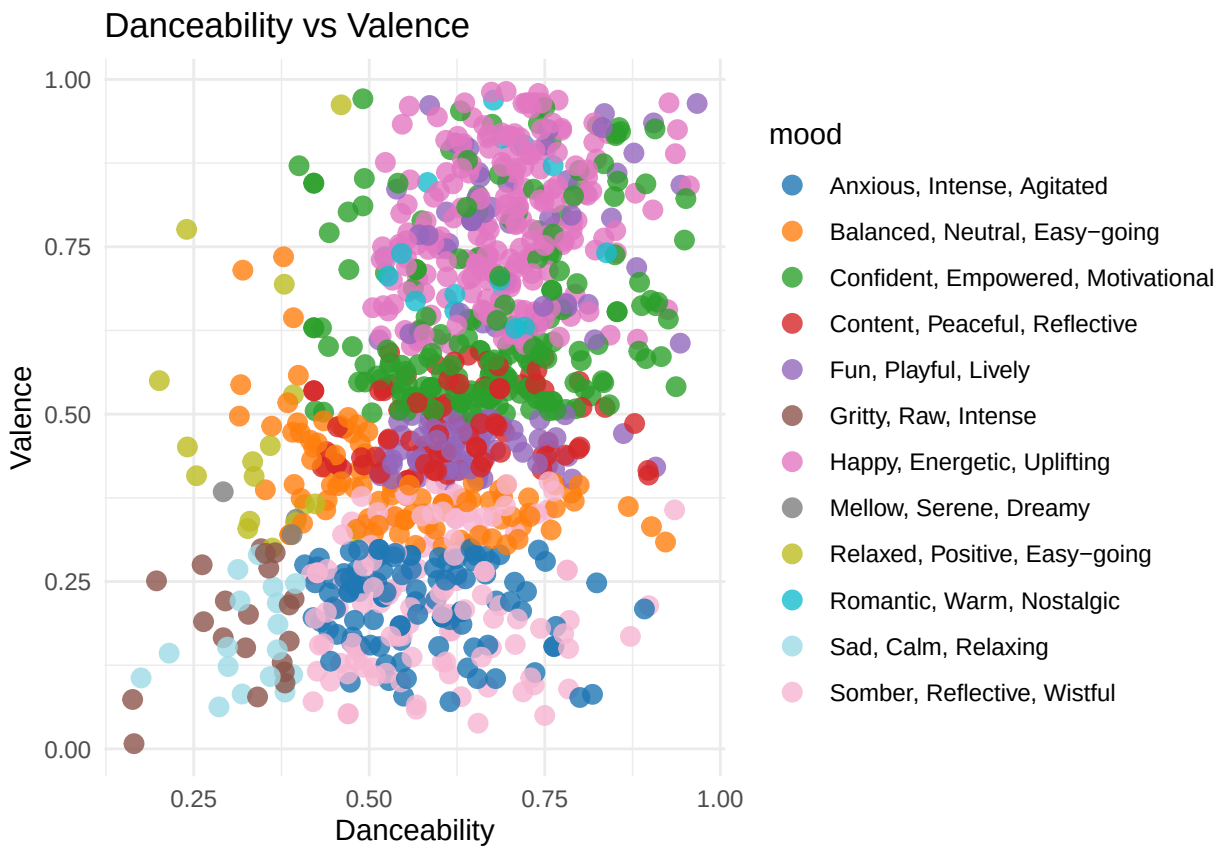
Ultimately, these plots not only serve as a means to visualize data but also provide a deeper understanding of the interplay between music and emotion, emphasizing how various musical characteristics can evoke or reflect different moods in listeners. This exploration lays the groundwork for further analysis and potential applications, such as developing mood-based playlists or music recommendation systems that leverage these insights to enhance user engagement and satisfaction.

```
# Load necessary libraries without displaying messages
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(ggplot2))

# Step 1: Data Preparation
# Select relevant features and ensure mood labels are assigned
mood_data <- combined_data %>%
  select(artistName, trackName, danceability, energy, valence, mood) %>%
  filter(!is.na(mood)) %>%
  na.omit()  # Remove any additional rows with NA values

# Group by artist and track to ensure multiple plays are considered
mood_data <- mood_data %>%
  group_by(artistName, trackName) %>%
  summarise(across(c(danceability, energy, valence), mean, .names = "avg_{col}"),
            mood = first(mood),
            .groups = 'drop')

# Define a custom color palette with more colors
custom_colors <- c('#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
                   '#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#9edae5', '#f7b6d2',
                   '#d9d9d9', '#ff9896')

# Step 2: Create 2D Scatter Plots separately using ggplot2

# Scatter plot for Danceability vs. Energy
```

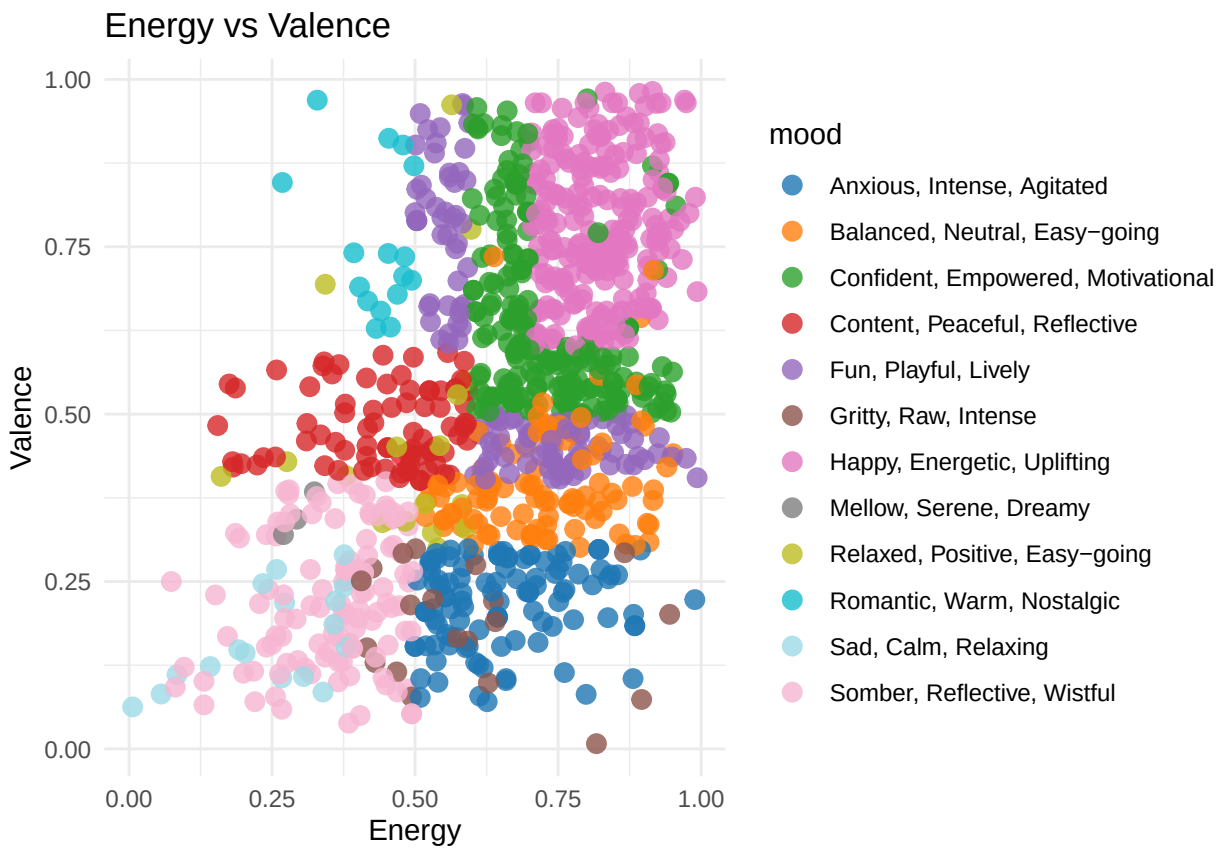
```r
p1 <- ggplot(mood_data, aes(x = avg_danceability, y = avg_energy, color = mood)) +
  geom_point(size = 3, alpha = 0.8) +
  labs(title = "Danceability vs Energy",
       x = "Danceability",
       y = "Energy") +
  theme_minimal() +
  scale_color_manual(values = custom_colors)

# Scatter plot for Danceability vs. Valence
p2 <- ggplot(mood_data, aes(x = avg_danceability, y = avg_valence, color = mood)) +
  geom_point(size = 3, alpha = 0.8) +
  labs(title = "Danceability vs Valence",
       x = "Danceability",
       y = "Valence") +
  theme_minimal() +
  scale_color_manual(values = custom_colors)

# Scatter plot for Energy vs. Valence
p3 <- ggplot(mood_data, aes(x = avg_energy, y = avg_valence, color = mood)) +
  geom_point(size = 3, alpha = 0.8) +
  labs(title = "Energy vs Valence",
       x = "Energy",
       y = "Valence") +
  theme_minimal() +
  scale_color_manual(values = custom_colors)

# Show the plots
print(p1)
```


Danceability vs Energy
Energy
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
Danceability
mood
Anxious, Intense, Agitated
Balanced, Neutral, Easy−going
Confident, Empowered, Motivational
Content, Peaceful, Reflective
Fun, Playful, Lively
Gritty, Raw, Intense
Happy, Energetic, Uplifting
Mellow, Serene, Dreamy
Relaxed, Positive, Easy−going
Romantic, Warm, Nostalgic
Sad, Calm, Relaxing
Somber, Reflective, Wistful

```
print(p2)
```


Danceability vs Valence
1.00
0.75
0.50
0.25
0.00
Valence
0.25
0.50
0.75
1.00
Danceability
mood
Anxious, Intense, Agitated
Balanced, Neutral, Easy−going
Confident, Empowered, Motivational
Content, Peaceful, Reflective
Fun, Playful, Lively
Gritty, Raw, Intense
Happy, Energetic, Uplifting
Mellow, Serene, Dreamy
Relaxed, Positive, Easy−going
Romantic, Warm, Nostalgic
Sad, Calm, Relaxing
Somber, Reflective, Wistful

```
print(p3)
```




**Normalized Listening Time for Top 5 Artists Over Time**

This analysis focuses on visualizing the normalized listening time for the top five artists based on total listening duration. The dataset is first prepared by converting the listening time from milliseconds to minutes, then aggregating the total listening time per artist for each date. The top five artists are identified based on their cumulative listening time. To enable meaningful comparisons across different artists, their listening times are normalized. Normalization is crucial as it adjusts the data to a common scale, allowing for a direct comparison of listening habits regardless of the absolute differences in total listening times between artists. This means that the values represent a proportion of each artist's maximum listening time, making it easier to observe trends and patterns over time.

In this visualization, a moving average is applied to the normalized listening times to smooth out short-term fluctuations and highlight longer-term trends. The

line chart, which displays the normalized listening time of the top five artists over time, provides insights into how listener engagement varies for each artist. The use of a custom color palette enhances the readability of the chart, allowing viewers to quickly identify trends for each artist. By representing normalized listening times, the analysis reveals not just the popularity of each artist, but also how their engagement levels change over time relative to each other, offering valuable insights into listener preferences and shifts in music consumption habits. This information can be particularly useful for artists and producers aiming to understand their audience better and tailor their marketing strategies accordingly.

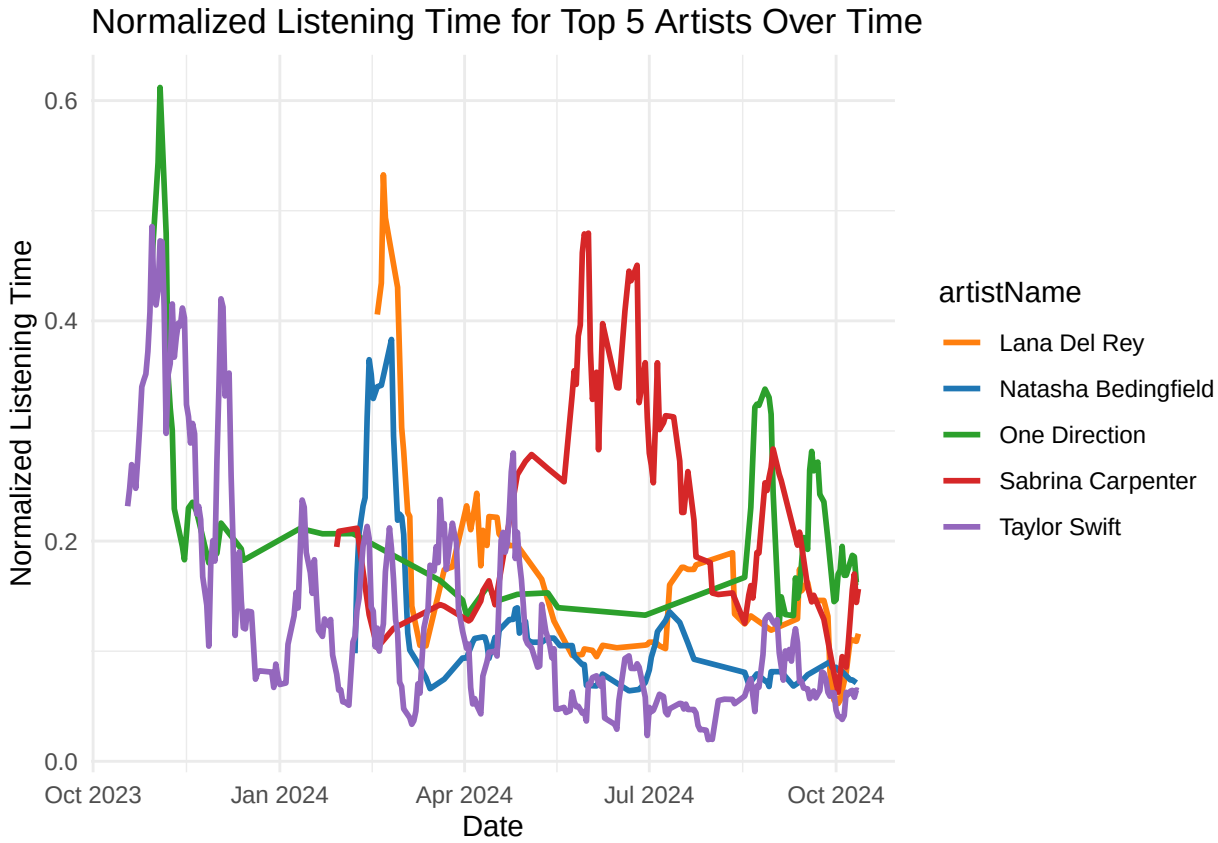
Furthermore, the smoothing process helps mitigate the impact of outliers or anomalies in daily listening behavior, thereby allowing for a clearer view of consistent patterns. By focusing on normalized values, the chart encourages a better understanding of listening dynamics among the top artists, illuminating moments of increased popularity or decline that may correlate with external factors such as new releases, media coverage, or social trends. This nuanced approach fosters a comprehensive analysis of listener engagement and can guide decisions for future content creation or promotional efforts within the music industry.

```r
# Load necessary libraries without displaying messages
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(lubridate))
suppressPackageStartupMessages(library(zoo))  # For the rollmean function

# Step 1: Data Preparation
# Assume combined_data has an 'end_time' column and 'milliseconds_played' for the listening time.
# Convert end_time to Date format if it's not already
combined_data$endTime <- as.Date(combined_data$endTime)

# Convert milliseconds played to minutes
combined_data$minutes_played <- combined_data$msPlayed / 60000  # Convert to minutes

# Aggregate data to get total listening time in minutes for each artist over time
artist_time_data <- combined_data %>%
  dplyr::group_by(endTime, artistName) %>%
  dplyr::summarise(total_minutes = sum(minutes_played), .groups = 'drop') %>%
  dplyr::arrange(endTime)

# Get the top 5 artists based on total listening time
top_artists <- artist_time_data %>%
  dplyr::group_by(artistName) %>%
  dplyr::summarise(total = sum(total_minutes)) %>%
  dplyr::top_n(5, total) %>%
  dplyr::pull(artistName)
```

```r
# Filter the data to keep only top 5 artists
top_artist_time_data <- artist_time_data %>%
  dplyr::filter(artistName %in% top_artists)

# Normalize the listening time for each artist
top_artist_time_data <- top_artist_time_data %>%
  dplyr::group_by(artistName) %>%
  dplyr::mutate(normalized_minutes = total_minutes / max(total_minutes)) %>%
  dplyr::ungroup()

# Smooth the normalized_minutes using a moving average and remove NAs
top_artist_time_data <- top_artist_time_data %>%
  dplyr::group_by(artistName) %>%
  dplyr::mutate(smooth_minutes = zoo::rollmean(normalized_minutes, k = 7, fill = NA, align = "right")) %>%
  dplyr::ungroup() %>%
  dplyr::filter(!is.na(smooth_minutes))  # Remove NAs

# Define a custom color palette with more colors
custom_colors <- c('#ff7f0e', '#1f77b4', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
                   '#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#9edae5', '#f7b6d2')

# Create Line Chart for Top 5 Artists Over Time using ggplot2
line_chart <- ggplot(top_artist_time_data, aes(x = endTime, y = smooth_minutes, color = artistName)) +
  geom_line(size = 1) +
  labs(title = "Normalized Listening Time for Top 5 Artists Over Time",
       x = "Date",
       y = "Normalized Listening Time") +
  theme_minimal() +
  scale_color_manual(values = custom_colors)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
# Print the line chart
print(line_chart)
```




**Mood Distribution by Day of the Week**

This analysis examines how listeners' moods vary throughout the week, providing insights into potential trends in music consumption based on emotional states. The first step involves preparing the data by converting the endTime column to a date format and extracting the day of the week from these dates. This allows

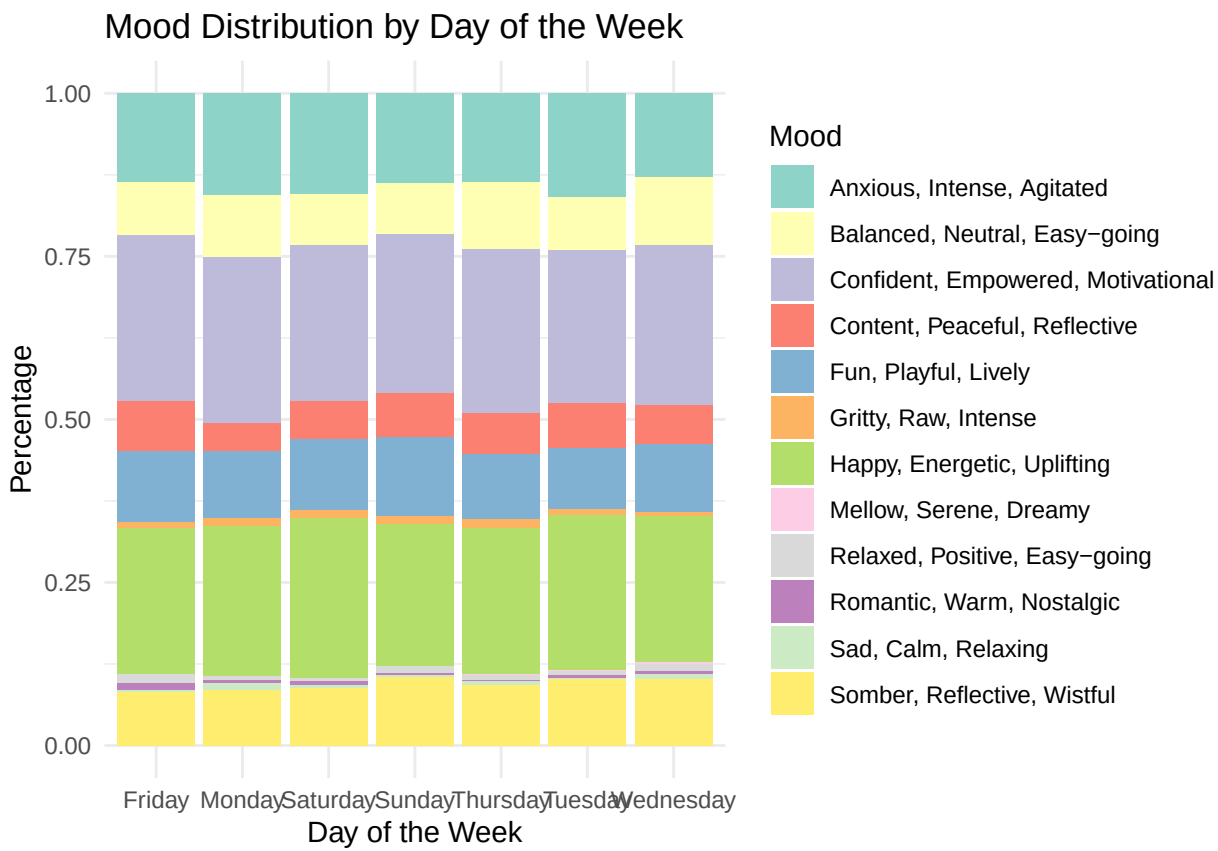
for a categorical analysis of moods across different days, facilitating a clearer understanding of how mood distribution may fluctuate in relation to the weekly cycle.

Next, the mood distribution is summarized by counting occurrences of each mood for each day of the week. This summary includes a calculation of the percentage of each mood relative to the total number of entries for that day, enabling a normalized view of mood representation. By filtering out any missing mood values, the analysis ensures that only complete cases are considered, providing a more accurate portrayal of the data.

The visualization employs a stacked bar chart to represent the mood distribution, where each bar corresponds to a day of the week, and the segments within each bar indicate the proportion of different moods. This method highlights not only the predominant moods for each day but also how they compare across the week. The use of the "Set3" color palette from the scale_fill_brewer function enhances the readability of the chart, ensuring that different moods are easily distinguishable. By visualizing the mood distribution in this manner, the analysis sheds light on patterns in listener emotions and how they might correlate with specific days, which can be particularly useful for understanding the psychological aspects of music consumption and for tailoring music recommendations or playlists based on the day of the week.

```r
# Load necessary libraries
library(dplyr)
library(ggplot2)

# Step 1: Data Preparation
# Assume combined_data has an 'endTime' column and 'mood' column.

# Convert endTime to Date format if it's not already
combined_data$endTime <- as.Date(combined_data$endTime)

# Add a new column for the day of the week
combined_data$day_of_week <- weekdays(combined_data$endTime)

# Step 2: Summarize mood distribution by day of the week
mood_distribution <- combined_data %>%
  select(day_of_week, mood) %>%
  filter(!is.na(mood)) %>%
  count(day_of_week, mood) %>%
  group_by(day_of_week) %>%
  mutate(percentage = n / sum(n) * 100)

# Step 3: Plot the mood distribution by day of the week
ggplot(mood_distribution, aes(x = day_of_week, y = percentage, fill = mood)) +
  geom_bar(stat = "identity", position = "fill") +
  labs(title = "Mood Distribution by Day of the Week",
```

```
      x = "Day of the Week",
      y = "Percentage",
      fill = "Mood") +
theme_minimal() +
scale_fill_brewer(palette = "Set3")  # Choose a suitable color palette
```



### Conclusion of the Data Visualization Section

The data visualization section of the analysis highlights significant insights into the emotional dynamics of music consumption through various visual representations. By utilizing scatter plots, we examined the relationships between key audio features—danceability, energy, and valence—and how they correspond to different mood classifications. The findings suggest clear trends in how these attributes interplay, offering a nuanced understanding of listener preferences and emotional responses.

Furthermore, the line charts depicting normalized listening times for the top artists over time revealed patterns in engagement and popularity, demonstrating how listener habits fluctuate across different periods. These visualizations not only enhance the interpretability of the data but also facilitate deeper insights into user behavior, informing future music recommendations and targeted engagement strategies.

Overall, this section underscores the importance of effective data visualization in deriving actionable conclusions from complex datasets, allowing for a richer understanding of the multifaceted relationships between music characteristics and listener emotions.

## Part 5: Music Recommendation System based on user mood

In today's digital age, music streaming services provide vast libraries of songs, making it essential to develop personalized recommendation systems that enhance user experience. This section outlines the methodology used to create a personalized music recommendation system based on user preferences, including mood, favorite artists, and disliked artists. The system utilizes a dataset of listening habits and audio features to generate tailored song recommendations.

### Methodology

1. **Data Loading**: The first step involves loading the dataset containing music information, including artist names, track names, mood indicators, and user listening habits. The read_csv function from the readr library is utilized to import the data efficiently while suppressing type messages for a cleaner output.
2. **User Profile Creation**: A user profile is constructed to encapsulate the individual's preferences. This profile includes:
    - **Preferred Moods**: The moods that the user enjoys in music, such as "Happy," "Energetic," "Uplifting," "Confident," "Empowered," and "Motivational."
    - **Favorite Artists**: A list of artists the user particularly enjoys, e.g., "Taylor Swift" and "JAY-Z."
    - **Disliked Artists**: Artists that the user prefers to avoid, e.g., "Kenya Grace."
3. **Recommendation Function**: A dedicated function, recommend_songs_personalized, processes the user profile and dataset to generate song recommendations. The function operates as follows:
    - **Initialize Recommendations**: It starts by creating an empty data frame to store the recommended songs.

- **Mood-Based Filtering**: The function iterates through the user's preferred moods. For each mood, it filters the dataset to find songs that match the mood criteria, using the str_detect function to ensure case-insensitive matching. The filtered results are appended to the recommendations data frame.
- **Disliked Artists Filtering**: After gathering mood-based recommendations, the function checks for any disliked artists specified in the user profile. It removes any recommendations that include these artists.
- **Favorite Artists Inclusion**: The function then looks for songs by the user's favorite artists and adds them to the recommendations list.
- **Finalizing Recommendations**: Duplicates are removed from the recommendations, and a random sample of songs is selected based on the number specified (defaulting to 10). This ensures variety in the recommendations.

4. **Displaying Recommendations**: Finally, the system checks whether any recommendations were generated. If there are results, they are formatted into a visually appealing table using the kable and kableExtra libraries. This enhances readability, making it easier for users to explore their personalized music options.

**Benefits of the Methodology**

- **Personalization**: By considering individual preferences for mood, favorite artists, and dislikes, the recommendation system provides a more relevant and enjoyable listening experience.
- **Flexibility**: The system allows for easy updates to the user profile, enabling dynamic recommendations that can adapt to changing musical tastes.
- **User Engagement**: Presenting recommendations in an organized and aesthetically pleasing format encourages users to explore new music, increasing overall engagement with the platform.
- **Scalability**: The methodology can be applied to larger datasets and expanded to include more complex recommendation algorithms, such as collaborative filtering or machine learning techniques.

In summary, this personalized music recommendation system not only enhances user satisfaction but also fosters a deeper connection to music by catering to individual preferences and moods.

```
# Load necessary libraries
library(dplyr)
library(readr)
library(knitr)
library(kableExtra)
library(stringr)

# Step 1: Load the combined data
```

```r
combined_data <- read_csv("combined_data.csv", show_col_types = FALSE)

# Step 2: Create a user profile
user_profile <- list(
  preferred_moods = c("Happy, Energetic, Uplifting", "Confident, Empowered, Motivational"),
  favorite_artists = c("Taylor Swift", "JAY-Z"),
  disliked_artists = c("Kenya Grace")
)

# Step 3: Function to recommend songs based on user profile
recommend_songs_personalized <- function(user_profile, data, num_recommendations = 10) {
  # Initialize an empty data frame for recommendations
  recommendations <- data.frame(artistName=character(), trackName=character(), mood=character(), stringsAsFactors=FALSE)

  # Filter based on preferred moods
  for (mood in user_profile$preferred_moods) {
    mood_recommendations <- data %>%
      filter(str_detect(tolower(mood), tolower(trimws(mood)))) %>%
      select(artistName, trackName, mood) %>%
      distinct()
    recommendations <- rbind(recommendations, mood_recommendations)
  }

  # Filter out disliked artists
  if (!is.null(user_profile$disliked_artists)) {
    recommendations <- recommendations %>%
      filter(!artistName %in% user_profile$disliked_artists)
  }

  # Include favorite artists if specified
  if (!is.null(user_profile$favorite_artists)) {
    favorite_recommendations <- data %>%
      filter(artistName %in% user_profile$favorite_artists) %>%
      select(artistName, trackName, mood) %>%
      distinct()
    recommendations <- rbind(recommendations, favorite_recommendations)
  }
```

```r
  # Remove duplicates and sample recommendations
  recommendations <- recommendations %>%
    distinct() %>%
    sample_n(min(num_recommendations, nrow(.)))

  return(recommendations)
}

# Step 4: Get personalized recommendations
personalized_recommendations <- recommend_songs_personalized(user_profile, combined_data)

# Step 5: Display personalized recommended songs
if (nrow(personalized_recommendations) > 0) {
  personalized_recommendations %>%
    kable() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
} else {
  print("No personalized recommendations could be found.")
}
```

| artistName | trackName | mood |
|---|---|---|
| Taylor Swift | End Game | Anxious, Intense, Agitated |
| Taylor Swift | Come In With The Rain (Taylor's Version) | Anxious, Intense, Agitated |
| Calvin Harris | Pray to God (feat. HAIM) | Fun, Playful, Lively |
| Taylor Swift | Soon You'll Get Better (feat. The Chicks) | Content, Peaceful, Reflective |
| Taylor Swift | Babe (Taylor's Version) (From The Vault) | Happy, Energetic, Uplifting |
| Rachel Platten | Girls | Content, Peaceful, Reflective |
| Taylor Swift | This Love | Somber, Reflective, Wistful |
| Harry Styles | Falling | Somber, Reflective, Wistful |
| ABBA | Mamma Mia | Happy, Energetic, Uplifting |
| harry uwu | Dont Let Me Go | Confident, Empowered, Motivational |