Data Science-Project Report and Proposal

Layal Ghryani

2023-12-16

The development of music streaming services has revolutionized the music business by providing consumers with access to an extensive song collection. Spotify has emerged as the leading platform in the worldwide music streaming business, offering millions of tracks and securing a substantial market share. Due to its large user base and comprehensive dataset, Spotify offers a special chance to forecast and analyze track popularity. Music supporters, record labels, and musicians are all very interested in learning what makes a song popular. Precisely anticipating the level of popularity of songs can provide important information about the tastes of the audience, advertising tactics, and the general workings of the music business. This work intends to construct a prediction algorithm that can evaluate track popularity by utilizing Spotify’s Music Dataset.

Using Spotify’s dataset to forecast track popularity is the study’s research question/problem. This subject has in fact been investigated in the past, with a number of studies looking at the connection between track popularity, contextual factors, and auditory qualities. Certain aural characteristics, such as tempo, energy, and danceability, have been linked in certain studies to the popularity of a music. Track popularity has also been proven to be influenced by contextual factors, such as playlist inclusion, album release patterns, and artist popularity. But there’s still a lot to learn, especially when it comes to how audio and contextual information work together in comprehensive prediction models. Furthermore, research is still being done on the precise weight and interactions of these factors.

if (!require("tidyverse")) install.packages("tidyverse")

## Loading required package: tidyverse

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

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

## Loading required package: scales  
##   
## Attaching package: 'scales'  
##   
## The following object is masked from 'package:purrr':  
##   
## discard  
##   
## The following object is masked from 'package:readr':  
##   
## col\_factor

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

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

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

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

library(viridis)  
if (!require("treemap")) install.packages("treemap")

## Loading required package: treemap

library(treemap)  
if (!require("htmltools")) install.packages("htmltools")

## Loading required package: htmltools

library(htmltools)  
if (!require("tm")) install.packages("tm")

## Loading required package: tm  
## Loading required package: NLP  
##   
## Attaching package: 'NLP'  
##   
## The following object is masked from 'package:ggplot2':  
##   
## annotate

library(tm)  
if (!require("readr")) install.packages("readr")  
library(readr)  
if (!require("ggcorrplot")) install.packages("ggcorrplot")

## Loading required package: ggcorrplot

library(ggcorrplot)  
if (!require("nnet")) install.packages("nnet")

## Loading required package: nnet

library(nnet)  
if (!require("ISLR")) install.packages("ISLR")

## Loading required package: ISLR

library(ISLR)   
if (!require("dplyr")) install.packages("dplyr")  
library(dplyr)

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

Data Pre-Processing:

str(Spotify)

## 'data.frame': 32833 obs. of 23 variables:  
## $ track\_id : chr "6f807x0ima9a1j3VPbc7VN" "0r7CVbZTWZgbTCYdfa2P31" "1z1Hg7Vb0AhHDiEmnDE79l" "75FpbthrwQmzHlBJLuGdC7" ...  
## $ track\_name : chr "I Don't Care (with Justin Bieber) - Loud Luxury Remix" "Memories - Dillon Francis Remix" "All the Time - Don Diablo Remix" "Call You Mine - Keanu Silva Remix" ...  
## $ track\_artist : chr "Ed Sheeran" "Maroon 5" "Zara Larsson" "The Chainsmokers" ...  
## $ track\_popularity : int 66 67 70 60 69 67 62 69 68 67 ...  
## $ track\_album\_id : chr "2oCs0DGTsRO98Gh5ZSl2Cx" "63rPSO264uRjW1X5E6cWv6" "1HoSmj2eLcsrR0vE9gThr4" "1nqYsOef1yKKuGOVchbsk6" ...  
## $ track\_album\_name : chr "I Don't Care (with Justin Bieber) [Loud Luxury Remix]" "Memories (Dillon Francis Remix)" "All the Time (Don Diablo Remix)" "Call You Mine - The Remixes" ...  
## $ track\_album\_release\_date: chr "6/14/19" "12/13/19" "7/5/19" "7/19/19" ...  
## $ playlist\_name : chr "Pop Remix" "Pop Remix" "Pop Remix" "Pop Remix" ...  
## $ playlist\_id : chr "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" "37i9dQZF1DXcZDD7cfEKhW" ...  
## $ playlist\_genre : chr "pop" "pop" "pop" "pop" ...  
## $ playlist\_subgenre : chr "dance pop" "dance pop" "dance pop" "dance pop" ...  
## $ danceability : num 0.748 0.726 0.675 0.718 0.65 0.675 0.449 0.542 0.594 0.642 ...  
## $ energy : num 0.916 0.815 0.931 0.93 0.833 0.919 0.856 0.903 0.935 0.818 ...  
## $ key : int 6 11 1 7 1 8 5 4 8 2 ...  
## $ loudness : num -2.63 -4.97 -3.43 -3.78 -4.67 ...  
## $ mode : int 1 1 0 1 1 1 0 0 1 1 ...  
## $ speechiness : num 0.0583 0.0373 0.0742 0.102 0.0359 0.127 0.0623 0.0434 0.0565 0.032 ...  
## $ acousticness : num 0.102 0.0724 0.0794 0.0287 0.0803 0.0799 0.187 0.0335 0.0249 0.0567 ...  
## $ instrumentalness : num 0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.00 0.00 4.83e-06 3.97e-06 0.00 ...  
## $ liveness : num 0.0653 0.357 0.11 0.204 0.0833 0.143 0.176 0.111 0.637 0.0919 ...  
## $ valence : num 0.518 0.693 0.613 0.277 0.725 0.585 0.152 0.367 0.366 0.59 ...  
## $ tempo : num 122 100 124 122 124 ...  
## $ duration\_ms : int 194754 162600 176616 169093 189052 163049 187675 207619 193187 253040 ...

str(Spotify2)

## 'data.frame': 953 obs. of 25 variables:  
## $ track\_name : chr "Seven (feat. Latto) (Explicit Ver.)" "LALA" "vampire" "Cruel Summer" ...  
## $ artist.s.\_name : chr "Latto, Jung Kook" "Myke Towers" "Olivia Rodrigo" "Taylor Swift" ...  
## $ artist\_count : int 2 1 1 1 1 2 2 1 1 2 ...  
## $ released\_year : int 2023 2023 2023 2019 2023 2023 2023 2023 2023 2023 ...  
## $ released\_month : int 7 3 6 8 5 6 3 7 5 3 ...  
## $ released\_day : int 14 23 30 23 18 1 16 7 15 17 ...  
## $ in\_spotify\_playlists: int 553 1474 1397 7858 3133 2186 3090 714 1096 2953 ...  
## $ in\_spotify\_charts : int 147 48 113 100 50 91 50 43 83 44 ...  
## $ streams : num 1.41e+08 1.34e+08 1.40e+08 8.01e+08 3.03e+08 ...  
## $ in\_apple\_playlists : int 43 48 94 116 84 67 34 25 60 49 ...  
## $ in\_apple\_charts : int 263 126 207 207 133 213 222 89 210 110 ...  
## $ in\_deezer\_playlists : chr "45" "58" "91" "125" ...  
## $ in\_deezer\_charts : int 10 14 14 12 15 17 13 13 11 13 ...  
## $ in\_shazam\_charts : int 826 382 949 548 425 946 418 194 953 339 ...  
## $ bpm : int 125 92 138 170 144 141 148 100 130 170 ...  
## $ key : chr "B" "C#" "F" "A" ...  
## $ mode : chr "Major" "Major" "Major" "Major" ...  
## $ danceability\_. : int 80 71 51 55 65 92 67 67 85 81 ...  
## $ valence\_. : int 89 61 32 58 23 66 83 26 22 56 ...  
## $ energy\_. : int 83 74 53 72 80 58 76 71 62 48 ...  
## $ acousticness\_. : int 31 7 17 11 14 19 48 37 12 21 ...  
## $ instrumentalness\_. : int 0 0 0 0 63 0 0 0 0 0 ...  
## $ liveness\_. : int 8 10 31 11 11 8 8 11 28 8 ...  
## $ speechiness\_. : int 4 4 6 15 6 24 3 4 9 33 ...  
## $ popularity : num 5.66e+07 5.35e+07 5.60e+07 3.20e+08 1.21e+08 ...

Let us check for the dimensions of our spotify dataset:

# Output dataset dimensions  
cat("Dimensions of Spotify:", dim(Spotify), "\n")

## Dimensions of Spotify: 32833 23

cat("Dimensions of Spotify2:", dim(Spotify2), "\n")

## Dimensions of Spotify2: 953 25

# Identifying missing values across columns  
col\_miss\_Spotify <- colSums(is.na(Spotify))  
if (any(col\_miss\_Spotify > 0)) {  
 cat("Missing values in Spotify:", col\_miss\_Spotify[col\_miss\_Spotify > 0], "\n")  
} else {  
 cat("No missing values in Spotify\n")  
}

## Missing values in Spotify: 5 5 5

col\_miss\_Spotify2 <- colSums(is.na(Spotify2))  
if (any(col\_miss\_Spotify2 > 0)) {  
 cat("Missing values in Spotify2:", col\_miss\_Spotify2[col\_miss\_Spotify2 > 0], "\n")  
} else {  
 cat("No missing values in Spotify2\n")  
}

## Missing values in Spotify2: 1 57 58

# Find number of duplicate values  
duplicate\_obs\_Spotify <- duplicated(Spotify)  
cat("Number of duplicate observations in Spotify:", sum(duplicate\_obs\_Spotify), "\n")

## Number of duplicate observations in Spotify: 0

duplicate\_obs\_Spotify2 <- duplicated(Spotify2)  
cat("Number of duplicate observations in Spotify2:", sum(duplicate\_obs\_Spotify2), "\n")

## Number of duplicate observations in Spotify2: 0

# Check for duplicate track ID  
duplicate\_id\_Spotify <- duplicated(Spotify$track\_id)  
cat("Number of duplicate track IDs in Spotify:", sum(duplicate\_id\_Spotify), "\n")

## Number of duplicate track IDs in Spotify: 4477

# Checking summary of numerical variables  
Spotify\_num <- Spotify %>% select\_if(is.numeric)  
cat("Summary of numerical variables in Spotify:\n")

## Summary of numerical variables in Spotify:

print(summary(Spotify\_num))

## track\_popularity danceability energy key   
## Min. : 0.00 Min. :0.0000 Min. :0.000175 Min. : 0.000   
## 1st Qu.: 24.00 1st Qu.:0.5630 1st Qu.:0.581000 1st Qu.: 2.000   
## Median : 45.00 Median :0.6720 Median :0.721000 Median : 6.000   
## Mean : 42.48 Mean :0.6548 Mean :0.698619 Mean : 5.374   
## 3rd Qu.: 62.00 3rd Qu.:0.7610 3rd Qu.:0.840000 3rd Qu.: 9.000   
## Max. :100.00 Max. :0.9830 Max. :1.000000 Max. :11.000   
## loudness mode speechiness acousticness   
## Min. :-46.448 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.: -8.171 1st Qu.:0.0000 1st Qu.:0.0410 1st Qu.:0.0151   
## Median : -6.166 Median :1.0000 Median :0.0625 Median :0.0804   
## Mean : -6.720 Mean :0.5657 Mean :0.1071 Mean :0.1753   
## 3rd Qu.: -4.645 3rd Qu.:1.0000 3rd Qu.:0.1320 3rd Qu.:0.2550   
## Max. : 1.275 Max. :1.0000 Max. :0.9180 Max. :0.9940   
## instrumentalness liveness valence tempo   
## Min. :0.0000000 Min. :0.0000 Min. :0.0000 Min. : 0.00   
## 1st Qu.:0.0000000 1st Qu.:0.0927 1st Qu.:0.3310 1st Qu.: 99.96   
## Median :0.0000161 Median :0.1270 Median :0.5120 Median :121.98   
## Mean :0.0847472 Mean :0.1902 Mean :0.5106 Mean :120.88   
## 3rd Qu.:0.0048300 3rd Qu.:0.2480 3rd Qu.:0.6930 3rd Qu.:133.92   
## Max. :0.9940000 Max. :0.9960 Max. :0.9910 Max. :239.44   
## duration\_ms   
## Min. : 4000   
## 1st Qu.:187819   
## Median :216000   
## Mean :225800   
## 3rd Qu.:253585   
## Max. :517810

Spotify2\_num <- Spotify2 %>% select\_if(is.numeric)  
cat("Summary of numerical variables in Spotify2:\n")

## Summary of numerical variables in Spotify2:

print(summary(Spotify2\_num))

## artist\_count released\_year released\_month released\_day   
## Min. :1.000 Min. :1930 Min. : 1.000 Min. : 1.00   
## 1st Qu.:1.000 1st Qu.:2020 1st Qu.: 3.000 1st Qu.: 6.00   
## Median :1.000 Median :2022 Median : 6.000 Median :13.00   
## Mean :1.556 Mean :2018 Mean : 6.034 Mean :13.93   
## 3rd Qu.:2.000 3rd Qu.:2022 3rd Qu.: 9.000 3rd Qu.:22.00   
## Max. :8.000 Max. :2023 Max. :12.000 Max. :31.00   
##   
## in\_spotify\_playlists in\_spotify\_charts streams in\_apple\_playlists  
## Min. : 31 Min. : 0.00 Min. :2.762e+03 Min. : 0.00   
## 1st Qu.: 875 1st Qu.: 0.00 1st Qu.:1.416e+08 1st Qu.: 13.00   
## Median : 2224 Median : 3.00 Median :2.905e+08 Median : 34.00   
## Mean : 5200 Mean : 12.01 Mean :5.141e+08 Mean : 67.81   
## 3rd Qu.: 5542 3rd Qu.: 16.00 3rd Qu.:6.739e+08 3rd Qu.: 88.00   
## Max. :52898 Max. :147.00 Max. :3.704e+09 Max. :672.00   
## NA's :1   
## in\_apple\_charts in\_deezer\_charts in\_shazam\_charts bpm   
## Min. : 0.00 Min. : 0.000 Min. : 0.00 Min. : 65.0   
## 1st Qu.: 7.00 1st Qu.: 0.000 1st Qu.: 0.00 1st Qu.:100.0   
## Median : 38.00 Median : 0.000 Median : 2.00 Median :121.0   
## Mean : 51.91 Mean : 2.666 Mean : 51.18 Mean :122.5   
## 3rd Qu.: 87.00 3rd Qu.: 2.000 3rd Qu.: 36.00 3rd Qu.:140.0   
## Max. :275.00 Max. :58.000 Max. :953.00 Max. :206.0   
## NA's :57   
## danceability\_. valence\_. energy\_. acousticness\_.   
## Min. :23.00 Min. : 4.00 Min. : 9.00 Min. : 0.00   
## 1st Qu.:57.00 1st Qu.:32.00 1st Qu.:53.00 1st Qu.: 6.00   
## Median :69.00 Median :51.00 Median :66.00 Median :18.00   
## Mean :66.97 Mean :51.43 Mean :64.28 Mean :27.06   
## 3rd Qu.:78.00 3rd Qu.:70.00 3rd Qu.:77.00 3rd Qu.:43.00   
## Max. :96.00 Max. :97.00 Max. :97.00 Max. :97.00   
##   
## instrumentalness\_. liveness\_. speechiness\_. popularity   
## Min. : 0.000 Min. : 3.00 Min. : 2.00 Min. :1.346e+03   
## 1st Qu.: 0.000 1st Qu.:10.00 1st Qu.: 4.00 1st Qu.:5.484e+07   
## Median : 0.000 Median :12.00 Median : 6.00 Median :1.090e+08   
## Mean : 1.581 Mean :18.21 Mean :10.13 Mean :1.882e+08   
## 3rd Qu.: 0.000 3rd Qu.:24.00 3rd Qu.:11.00 3rd Qu.:2.402e+08   
## Max. :91.000 Max. :97.00 Max. :64.00 Max. :1.425e+09   
## NA's :58

Below is the detailed data dictionary to understand all the variables present in the dataset:

Track popularity is the outcome variable under investigation and prediction in this study. Track popularity is a metric that evaluates a song’s relative popularity within the Spotify ecosystem using data like the ones below:

track\_name - Song Name

track\_popularity - Song Popularity (0-100) where higher is better

playlist\_genre - Playlist genre

danceability - Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.

energy - Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.

key - The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation .

loudness -The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks.

mode - Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

speechiness - Speechiness detects the presence of spoken words in a track.

acousticness - A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

instrumentalness - Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context.

liveness - Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.

valence - A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.

tempo - The overall estimated tempo of a track in beats per minute (BPM).

duration\_ms - Duration of song in milliseconds

Exploratory Data Analysis (EDA): Let us start with the popularity analysis. For the purpose of this study, I am planning to classify track popularity attribute into different classes of low,medium and high popularity. As the dictionary mentions, track popularity is a value between 0 and 100. I am classifying the group as follows:

high - track popularity greater than 75 medium - track popularity between 30 and 75 low - track popularity less than 30

# Popularity Analysis  
Spotify <- Spotify %>%  
 mutate(popularity = case\_when(  
 track\_popularity <= 30 ~ "low",  
 track\_popularity > 30 & track\_popularity <= 75 ~ "medium",  
 track\_popularity > 75 ~ "high"  
 ))  
  
# Top tracks in the dataset  
popular\_track <- Spotify %>%  
 filter(popularity == "high") %>%  
 arrange(desc(track\_popularity)) %>%  
 distinct(track\_name, track\_popularity)  
  
cat("Top tracks in Spotify with high popularity:\n")

## Top tracks in Spotify with high popularity:

print(head(popular\_track, 10))

## track\_name track\_popularity  
## 1 Dance Monkey 100  
## 2 ROXANNE 99  
## 3 Tusa 98  
## 4 Memories 98  
## 5 Blinding Lights 98  
## 6 Circles 98  
## 7 The Box 98  
## 8 everything i wanted 97  
## 9 Don't Start Now 97  
## 10 Falling 97

# Create a summary of top artists within each playlist genre  
artist\_genre <- Spotify %>%  
 dplyr::select(playlist\_genre, track\_artist, track\_popularity) %>%  
 group\_by(playlist\_genre, track\_artist) %>%  
 summarise(n = n()) %>%  
 top\_n(10, n)

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

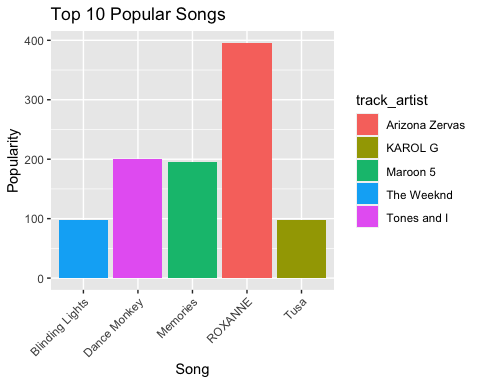
cat("Top 10 Track Artists within each Playlist Genre:\n")

## Top 10 Track Artists within each Playlist Genre:

print(artist\_genre)

## # A tibble: 61 × 3  
## # Groups: playlist\_genre [6]  
## playlist\_genre track\_artist n  
## <chr> <chr> <int>  
## 1 edm Armin van Buuren 38  
## 2 edm Bassjackers 38  
## 3 edm Blasterjaxx 38  
## 4 edm Calvin Harris 40  
## 5 edm David Guetta 60  
## 6 edm Dimitri Vegas & Like Mike 79  
## 7 edm Hardwell 76  
## 8 edm Martin Garrix 125  
## 9 edm R3HAB 38  
## 10 edm The Chainsmokers 49  
## # ℹ 51 more rows

# Create a bar plot with varied colors  
top\_10\_popular\_songs <- head(Spotify[order(-Spotify$track\_popularity),  
 c("track\_name", "track\_artist",   
 "track\_popularity")], 10)  
  
ggplot(top\_10\_popular\_songs, aes(x = track\_name, y = track\_popularity,   
 fill = track\_artist)) +  
 geom\_bar(stat = "identity") +  
 labs(title = "Top 10 Popular Songs",  
 x = "Song",  
 y = "Popularity") +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))

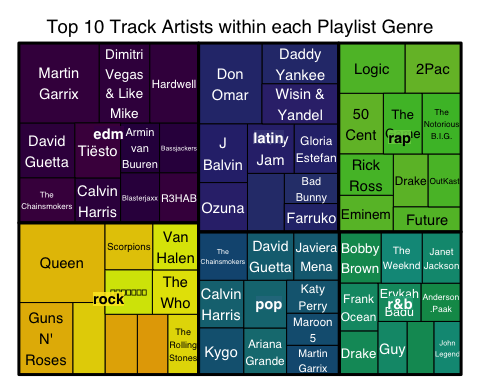
 Top artist by genre

The top artists list features many edm artists. This may be due to the high popularity of edm songs. So, what about the artists who creates songs in other genres.We will try to find out who are the top artists in each genre.We can use a tree map to analyze this.

library(ggplot2)  
library(dplyr)  
library(treemap)  
  
# Load the dataset  
Spotify <- read.csv("spotify\_songs.csv") # Replace "data.csv" with the actual file name and path  
  
# Create a summary of top artists within each playlist genre  
artist\_genre <- Spotify %>%  
 dplyr::select(playlist\_genre, track\_artist, track\_popularity) %>%  
 group\_by(playlist\_genre, track\_artist) %>%  
 summarise(n = n()) %>%  
 top\_n(10, n)

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

# Create a treemap visualization  
tm <- treemap(artist\_genre, index = c("playlist\_genre", "track\_artist"), vSize = "n", vColor = 'playlist\_genre', palette = viridisLite::viridis(6), title = "Top 10 Track Artists within each Playlist Genre")



# Display the treemap  
print(tm)

## $tm  
## playlist\_genre track\_artist vSize vColor stdErr vColorValue  
## 1 edm Armin van Buuren 38 1 38 NA  
## 2 edm Bassjackers 38 1 38 NA  
## 3 edm Blasterjaxx 38 1 38 NA  
## 4 edm Calvin Harris 40 1 40 NA  
## 5 edm David Guetta 60 1 60 NA  
## 6 edm Dimitri Vegas & Like Mike 79 1 79 NA  
## 7 edm Hardwell 76 1 76 NA  
## 8 edm Martin Garrix 125 1 125 NA  
## 9 edm <NA> 630 11 630 NA  
## 10 edm R3HAB 38 1 38 NA  
## 11 edm The Chainsmokers 49 1 49 NA  
## 12 edm Tiësto 49 1 49 NA  
## 13 latin Bad Bunny 32 1 32 NA  
## 14 latin Ballin Entertainment 42 1 42 NA  
## 15 latin Daddy Yankee 61 1 61 NA  
## 16 latin Don Omar 100 1 100 NA  
## 17 latin Farruko 30 1 30 NA  
## 18 latin Gloria Estefan 43 1 43 NA  
## 19 latin J Balvin 56 1 56 NA  
## 20 latin <NA> 516 10 516 NA  
## 21 latin Nicky Jam 45 1 45 NA  
## 22 latin Ozuna 47 1 47 NA  
## 23 latin Wisin & Yandel 60 1 60 NA  
## 24 pop Ariana Grande 39 1 39 NA  
## 25 pop Avicii 40 1 40 NA  
## 26 pop Calvin Harris 42 1 42 NA  
## 27 pop David Guetta 44 1 44 NA  
## 28 pop Javiera Mena 43 1 43 NA  
## 29 pop Katy Perry 34 1 34 NA  
## 30 pop Kygo 41 1 41 NA  
## 31 pop Maroon 5 33 1 33 NA  
## 32 pop Martin Garrix 30 1 30 NA  
## 33 pop <NA> 392 10 392 NA  
## 34 pop The Chainsmokers 46 1 46 NA  
## 35 r&b Anderson .Paak 30 1 30 NA  
## 36 r&b Bobby Brown 42 1 42 NA  
## 37 r&b D'Angelo 28 1 28 NA  
## 38 r&b Drake 33 1 33 NA  
## 39 r&b Erykah Badu 32 1 32 NA  
## 40 r&b Frank Ocean 36 1 36 NA  
## 41 r&b Guy 30 1 30 NA  
## 42 r&b Janet Jackson 39 1 39 NA  
## 43 r&b John Legend 28 1 28 NA  
## 44 r&b <NA> 339 10 339 NA  
## 45 r&b The Weeknd 41 1 41 NA  
## 46 rap 2Pac 55 1 55 NA  
## 47 rap 50 Cent 53 1 53 NA  
## 48 rap Drake 36 1 36 NA  
## 49 rap Eminem 39 1 39 NA  
## 50 rap Future 33 1 33 NA  
## 51 rap Logic 65 1 65 NA  
## 52 rap <NA> 451 10 451 NA  
## 53 rap OutKast 34 1 34 NA  
## 54 rap Rick Ross 44 1 44 NA  
## 55 rap The Game 46 1 46 NA  
## 56 rap The Notorious B.I.G. 46 1 46 NA  
## 57 rock Aerosmith 36 1 36 NA  
## 58 rock Creedence Clearwater Revival 38 1 38 NA  
## 59 rock Guns N' Roses 76 1 76 NA  
## 60 rock <NA> 532 10 532 NA  
## 61 rock Queen 134 1 134 NA  
## 62 rock Scorpions 44 1 44 NA  
## 63 rock The Cranberries 45 1 45 NA  
## 64 rock The Rolling Stones 36 1 36 NA  
## 65 rock The Who 40 1 40 NA  
## 66 rock Van Halen 42 1 42 NA  
## 67 rock オメガトライブ 41 1 41 NA  
## level x0 y0 w h color  
## 1 2 0.2298462 0.60843373 0.08822378 0.15060241 #30004C  
## 2 2 0.3180699 0.60843373 0.08822378 0.15060241 #33004C  
## 3 2 0.2298462 0.45783133 0.08822378 0.15060241 #35004C  
## 4 2 0.1265315 0.45783133 0.10331469 0.13537295 #38004C  
## 5 2 0.0000000 0.59323533 0.12653147 0.16580082 #3B004C  
## 6 2 0.1813811 0.75903614 0.11463287 0.24096386 #3D004C  
## 7 2 0.2960140 0.75903614 0.11027972 0.24096386 #40004C  
## 8 2 0.0000000 0.75903614 0.18138112 0.24096386 #42004C  
## 9 1 0.0000000 0.45783133 0.40629371 0.54216867 #440154  
## 10 2 0.3180699 0.45783133 0.08822378 0.15060241 #45004C  
## 11 2 0.0000000 0.45783133 0.12653147 0.13540400 #47004C  
## 12 2 0.1265315 0.59320428 0.10331469 0.16583187 #4A004C  
## 13 2 0.6001780 0.51584061 0.12292280 0.09102307 #2E3E7A  
## 14 2 0.5169077 0.43050648 0.08327028 0.17635721 #2E3B7A  
## 15 2 0.5496453 0.87703661 0.17345546 0.12296339 #2E387A  
## 16 2 0.4062937 0.75608901 0.14335162 0.24391099 #2E357A  
## 17 2 0.6001780 0.43050648 0.12292280 0.08533413 #2E337A  
## 18 2 0.6223473 0.60686369 0.10075344 0.14922533 #2E307A  
## 19 2 0.4062937 0.57907327 0.11061400 0.17701575 #2F2E7A  
## 20 1 0.4062937 0.43050648 0.31680708 0.56949352 #414487  
## 21 2 0.5169077 0.60686369 0.10543964 0.14922533 #322E7A  
## 22 2 0.4062937 0.43050648 0.11061400 0.14856679 #342E7A  
## 23 2 0.5496453 0.75608901 0.17345546 0.12094760 #372E7A  
## 24 2 0.5083215 0.00000000 0.09711083 0.14042063 #147A80  
## 25 2 0.5083215 0.14042063 0.09711083 0.14402115 #147680  
## 26 2 0.4062937 0.14050739 0.10202783 0.14393439 #147280  
## 27 2 0.5164087 0.28444178 0.10532740 0.14606470 #146E80  
## 28 2 0.6217361 0.28444178 0.10293360 0.14606470 #146A80  
## 29 2 0.6054324 0.18474054 0.11923735 0.09970124 #146680  
## 30 2 0.4062937 0.00000000 0.10202783 0.14050739 #146280  
## 31 2 0.6054324 0.08797168 0.11923735 0.09676885 #145E80  
## 32 2 0.6054324 0.00000000 0.11923735 0.08797168 #145A80  
## 33 1 0.4062937 0.00000000 0.31837602 0.43050648 #2A788E  
## 34 2 0.4062937 0.28444178 0.11011501 0.14606470 #145680  
## 35 2 0.9091373 0.16013139 0.09086268 0.11544356 #069758  
## 36 2 0.7246697 0.27557494 0.09478583 0.15493153 #06975E  
## 37 2 0.8777228 0.00000000 0.06113861 0.16013139 #069763  
## 38 2 0.7246697 0.00000000 0.08754742 0.13179671 #069768  
## 39 2 0.8122171 0.16013139 0.09692019 0.11544356 #06976E  
## 40 2 0.7246697 0.13179671 0.08754742 0.14377823 #069773  
## 41 2 0.8122171 0.00000000 0.06550565 0.16013139 #069778  
## 42 2 0.9119846 0.27557494 0.08801542 0.15493153 #06977E  
## 43 2 0.9388614 0.00000000 0.06113861 0.16013139 #069783  
## 44 1 0.7246697 0.00000000 0.27533028 0.43050648 #22A884  
## 45 2 0.8194556 0.27557494 0.09252903 0.15493153 #069788  
## 46 2 0.8730879 0.84847179 0.12691214 0.15152821 #75BC32  
## 47 2 0.7231008 0.66537520 0.10121144 0.18309659 #70BC32  
## 48 2 0.8466633 0.50575568 0.07885885 0.15961952 #6BBC32  
## 49 2 0.7231008 0.43050648 0.12356255 0.11036000 #66BC32  
## 50 2 0.8466633 0.43050648 0.15333666 0.07524920 #61BC32  
## 51 2 0.7231008 0.84847179 0.14998707 0.15152821 #5CBC32  
## 52 1 0.7231008 0.43050648 0.27689921 0.56949352 #7AD151  
## 53 2 0.9255222 0.50575568 0.07447781 0.15961952 #56BC32  
## 54 2 0.7231008 0.54086648 0.12356255 0.12450872 #51BC32  
## 55 2 0.8243122 0.66537520 0.08784389 0.18309659 #4CBC32  
## 56 2 0.9121561 0.66537520 0.08784389 0.18309659 #47BC32  
## 57 2 0.2678261 0.00000000 0.06923378 0.18181027 #E4A700  
## 58 2 0.1947460 0.00000000 0.07308010 0.18181027 #E4AF00  
## 59 2 0.0000000 0.00000000 0.12231983 0.21724545 #E4B700  
## 60 1 0.0000000 0.00000000 0.40629371 0.45783133 #FDE725  
## 61 2 0.0000000 0.21724545 0.19474604 0.24058587 #E4C000  
## 62 2 0.1947460 0.31568875 0.10823369 0.14214258 #E4C800  
## 63 2 0.1223198 0.00000000 0.07242621 0.21724545 #E4D100  
## 64 2 0.3370599 0.00000000 0.06923378 0.18181027 #E4D900  
## 65 2 0.3018257 0.18181027 0.10446798 0.13387847 #E4E200  
## 66 2 0.3029797 0.31568875 0.10331397 0.14214258 #DDE400  
## 67 2 0.1947460 0.18181027 0.10707968 0.13387847 #D5E400  
##   
## $type  
## [1] "index"  
##   
## $vSize  
## [1] "n"  
##   
## $vColor  
## [1] NA  
##   
## $stdErr  
## [1] "n"  
##   
## $algorithm  
## [1] "pivotSize"  
##   
## $vpCoorX  
## [1] 0.03937008 0.96062992  
##   
## $vpCoorY  
## [1] 0.0246063 0.8878937  
##   
## $aspRatio  
## [1] 1.333941  
##   
## $range  
## [1] NA  
##   
## $mapping  
## [1] NA NA NA  
##   
## $draw  
## [1] TRUE

Above, treemap depicts top 10 track artists with in each of the playlist genre. The size of the boxes in treemap corresponds to the count tracks for the artists. For genre edm, rock, pop, rap, latin and r&b, the top track artist are Martin Garrix, Queen, The Chainsmoker, Logic, Don Omar and Bobby Brown respectively.

One of Spotify’s most popular features is its Discover Playlist, a playlist that is generated each week based on a user’s listening habits. As a Spotify user I have found these playlists to be extremely accurate and useful. I wanted to make a try to build a basic version of it, a song recommendation engine based on different attributes as follows:

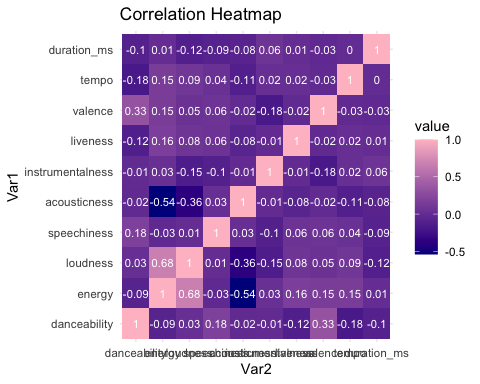
Based on Genre: Songs will be displayed as per the user preferred genre and rating scale. Based on Artists: Songs will be filtered as per the artist preference of the user and the rating scale. Based on Mood: Songs will be filtered as per the mood preference and rating scale specified by the user. For this purpose, songs have been classified into different groups like Gym(the songs with high energy),Cheerful(the songs with high valence),Party/Dance(the songs with high danceability) and Others.

# Select the variables for correlation  
variables <- c('danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration\_ms')  
  
# Compute the correlation matrix  
correlation\_matrix <- cor(Spotify[, variables])  
  
# Create a heatmap  
library(ggplot2)  
library(reshape2)

##   
## Attaching package: 'reshape2'

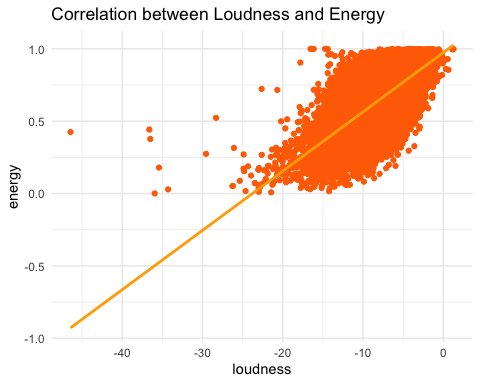
## The following object is masked from 'package:tidyr':  
##   
## smiths

melted\_correlation <- melt(correlation\_matrix)  
ggplot(data = melted\_correlation, aes(x = Var1, y = Var2, fill = value)) +  
 geom\_tile() +  
 scale\_fill\_gradient(low = "darkblue", high = "pink") +  
 theme\_minimal() +  
 labs(title = "Correlation Heatmap") +  
 geom\_text(aes(label = round(value, 2)), color = "white", size = 3) + coord\_flip()

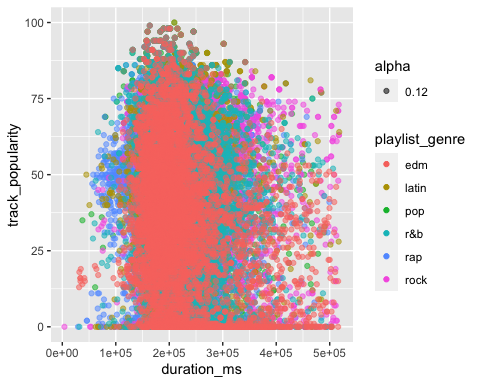


library(ggplot2)  
library(scales)  
  
# Assuming you have a Spotify frame called 'Spotify' with columns: loudness and energy  
  
ggplot(Spotify, aes(x = loudness, y = energy)) +  
 geom\_point(color = "#FF6F00") + # Set the color of the points to orange  
 geom\_smooth(method = "lm", color = "#FFAB00", se = FALSE) + # Set the color of the smoother line to a lighter shade of orange  
 scale\_color\_manual(values = c("#FF6F00", "#FFAB00")) + # Match the colors for points and line  
 labs(title = "Correlation between Loudness and Energy") +  
 theme\_minimal()

## `geom\_smooth()` using formula = 'y ~ x'

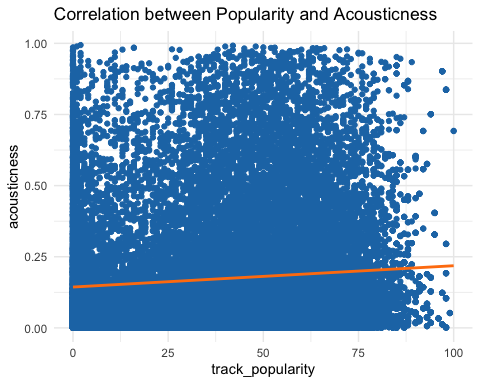


ggplot(data = Spotify) +   
 geom\_point(mapping = aes(x = duration\_ms, y = track\_popularity,  
 color = playlist\_genre, alpha = 0.12))

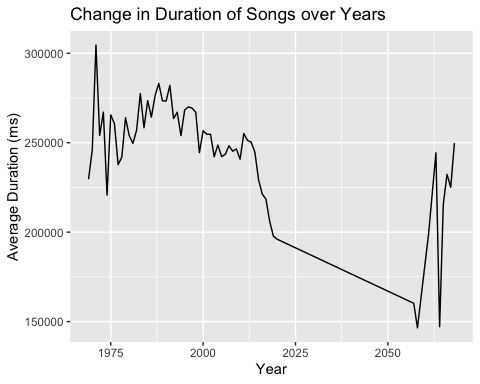


library(ggplot2)  
library(scales)  
  
# Assuming you have a Spotify frame called 'Spotify' with columns: track\_popularity and acousticness  
  
ggplot(Spotify, aes(x = track\_popularity, y = acousticness)) +  
 geom\_point(color = "#1F77B4") + # Set the color of the points to a blue shade  
 geom\_smooth(method = "lm", color = "#FF7F0E", se = FALSE) + # Set the color of the smoother line to an orange shade  
 scale\_color\_manual(values = c("#1F77B4", "#FF7F0E")) + # Match the colors for points and line  
 labs(title = "Correlation between Popularity and Acousticness") +  
 theme\_minimal()

## `geom\_smooth()` using formula = 'y ~ x'



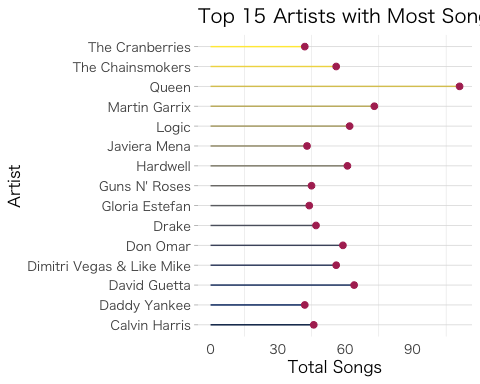
Spotify$track\_album\_release\_year <- as.numeric(format(as.Date(Spotify$track\_album\_release\_date, "%m/%d/%y"), "%Y"))  
  
# Calculate the average duration for each year  
average\_duration\_by\_year <- aggregate(duration\_ms ~ track\_album\_release\_year, Spotify, mean)  
  
# Plot the change in duration over years  
library(ggplot2)  
ggplot(data = average\_duration\_by\_year, aes(x = track\_album\_release\_year, y = duration\_ms)) +  
 geom\_line() +  
 labs(x = "Year", y = "Average Duration (ms)", title = "Change in Duration of Songs over Years")



# Calculate the average duration for each genre  
average\_duration\_by\_genre <- aggregate(duration\_ms ~ playlist\_genre, Spotify, mean)  
  
# Display the average duration for each genre  
print(average\_duration\_by\_genre)

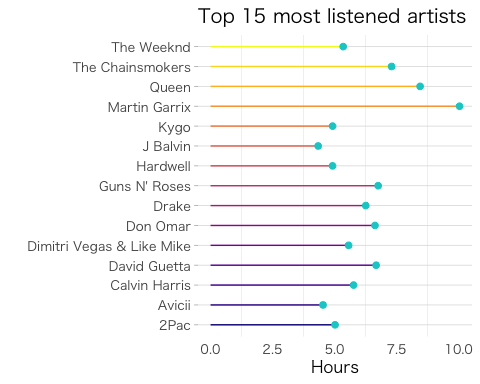
## playlist\_genre duration\_ms  
## 1 edm 222540.9  
## 2 latin 216863.4  
## 3 pop 217768.1  
## 4 r&b 237599.5  
## 5 rap 214163.9  
## 6 rock 248576.5

most\_songs <- Spotify %>%  
 group\_by(track\_artist) %>%  
 summarize(total\_songs = n\_distinct(track\_name)) %>%  
 arrange(desc(total\_songs)) %>%  
 slice(1:15) %>%  
 ggplot(aes(x = track\_artist, y = total\_songs, color = track\_artist)) +  
 geom\_segment(aes(x = track\_artist, xend = track\_artist, y = 0, yend = total\_songs)) +  
 geom\_point(size = 2, color = "maroon") +  
 scale\_color\_viridis(discrete = TRUE, guide = "none", option = "E") +  
 theme\_light(base\_size = 12, base\_family = "HiraKakuProN-W3") +  
 theme(  
 panel.grid.major.x = element\_blank(),  
 panel.border = element\_blank(),  
 axis.ticks.x = element\_blank()  
 ) +  
 labs(title = "Top 15 Artists with Most Songs",  
 x = "Artist",  
 y = "Total Songs") +  
 coord\_flip()  
  
most\_songs

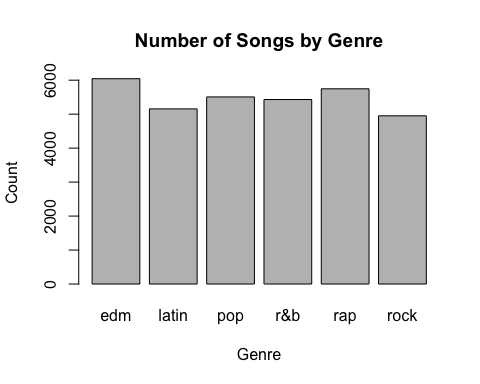


most\_listened <- Spotify %>%  
 group\_by(track\_artist) %>%  
 mutate(track\_artist = iconv(track\_artist, to = "UTF-8")) %>%  
 summarize(listenedHours = sum(duration\_ms) / 1000 / 3600) %>%  
 arrange(desc(listenedHours)) %>%  
 slice(1:15) %>%  
 ggplot(aes(x = track\_artist, y = listenedHours, color = track\_artist)) +  
 geom\_segment(aes(x = track\_artist, xend = track\_artist, y = 0, yend = listenedHours)) +  
 geom\_point(size = 2, color = "cyan3") +  
 scale\_color\_viridis(discrete = TRUE, guide = FALSE, option = "C") +  
 theme\_light(base\_size = 12, base\_family = "HiraKakuProN-W3") +  
 theme(  
 panel.grid.major.x = element\_blank(),  
 panel.border = element\_blank(),  
 axis.ticks.x = element\_blank()  
 ) +  
 labs(title = "Top 15 most listened artists") +  
 xlab("") +  
 ylab("Hours") +  
 coord\_flip()  
  
most\_listened

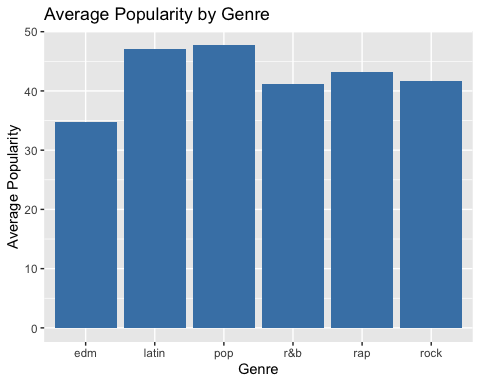
## Warning: The `guide` argument in `scale\_\*()` cannot be `FALSE`. This was deprecated in  
## ggplot2 3.3.4.  
## ℹ Please use "none" instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



# Count the number of songs for each genre  
genre\_counts <- table(Spotify$playlist\_genre)  
  
# Create a bar graph of the genre counts  
barplot(genre\_counts, main = "Number of Songs by Genre", xlab = "Genre", ylab = "Count")

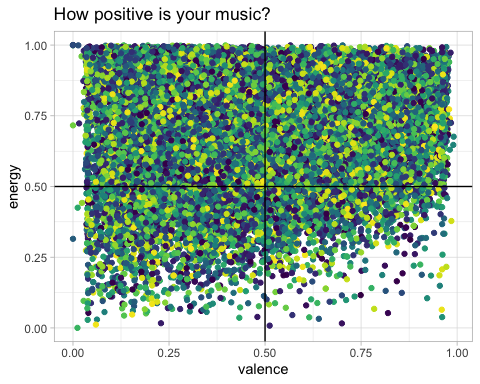


# Calculate the average popularity by genre  
avg\_popularity <- Spotify %>%  
 group\_by(playlist\_genre) %>%  
 summarise(avg\_popularity = mean(track\_popularity))  
  
# Plot the average popularity by genre  
ggplot(avg\_popularity, aes(x = playlist\_genre, y = avg\_popularity)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 labs(x = "Genre", y = "Average Popularity", title = "Average Popularity by Genre")



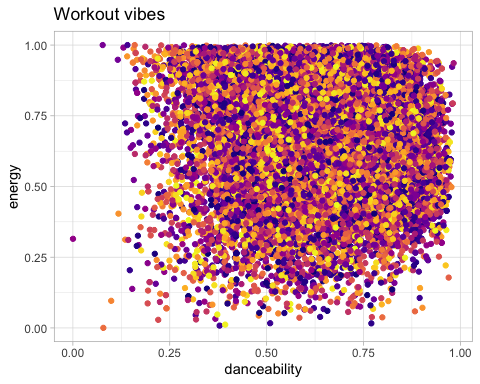
# Create the scatter plot  
ggplot(Spotify, aes(x = valence, y = energy, color = track\_name)) +  
 geom\_jitter(show.legend = FALSE) +  
 scale\_color\_viridis(discrete = TRUE, option = "D") +  
 geom\_vline(xintercept = 0.5) +  
 geom\_hline(yintercept = 0.5) +  
 scale\_x\_continuous(breaks = seq(0, 1, 0.25)) +  
 scale\_y\_continuous(breaks = seq(0, 1, 0.25)) +  
 labs(title = "How positive is your music?") +  
 theme\_light()

## Warning: Removed 5 rows containing missing values (`geom\_point()`).

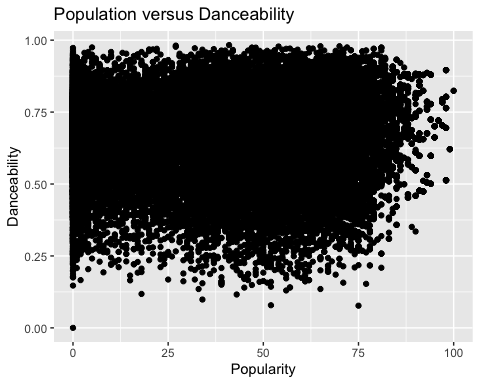


library(ggplot2)  
  
# Assuming you have the Spotify frame 'spotify\_songs' with columns: track\_name, danceability, and energy  
  
track\_names <- Spotify$track\_name  
danceability <- Spotify$danceability  
energy <- Spotify$energy  
  
spotify\_data <- data.frame(  
 track\_name = track\_names,  
 danceability = danceability,  
 energy = energy  
)  
  
spotify\_data %>%  
 ggplot(aes(x = danceability, y = energy, color = track\_name)) +  
 geom\_jitter(show.legend = FALSE) +  
 scale\_color\_viridis(discrete = TRUE, option = "C") +  
 labs(title = "Workout vibes") +  
 theme\_light()

## Warning: Removed 5 rows containing missing values (`geom\_point()`).



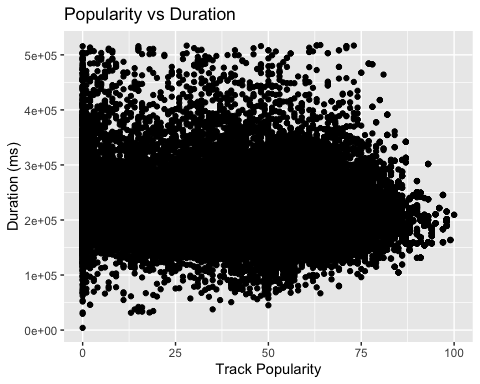
# Read the dataset  
Spotify <- read.csv("spotify\_songs.csv")  
  
# Create a scatter plot of population versus danceability  
ggplot(Spotify, aes(x = track\_popularity, y = danceability)) +  
 geom\_point() +  
 labs(x = "Popularity", y = "Danceability") +  
 ggtitle("Population versus Danceability")



# Load required packages  
library(ggplot2)  
library(dplyr)  
  
# Load the Spotifyset  
Spotify <- read.csv("spotify\_songs.csv") # Replace "Spotify.csv" with the actual file name and path  
  
# Explore the Spotifyset  
head(Spotify) # Check the structure and contents of the Spotifyset

## track\_id track\_name  
## 1 6f807x0ima9a1j3VPbc7VN I Don't Care (with Justin Bieber) - Loud Luxury Remix  
## 2 0r7CVbZTWZgbTCYdfa2P31 Memories - Dillon Francis Remix  
## 3 1z1Hg7Vb0AhHDiEmnDE79l All the Time - Don Diablo Remix  
## 4 75FpbthrwQmzHlBJLuGdC7 Call You Mine - Keanu Silva Remix  
## 5 1e8PAfcKUYoKkxPhrHqw4x Someone You Loved - Future Humans Remix  
## 6 7fvUMiyapMsRRxr07cU8Ef Beautiful People (feat. Khalid) - Jack Wins Remix  
## track\_artist track\_popularity track\_album\_id  
## 1 Ed Sheeran 66 2oCs0DGTsRO98Gh5ZSl2Cx  
## 2 Maroon 5 67 63rPSO264uRjW1X5E6cWv6  
## 3 Zara Larsson 70 1HoSmj2eLcsrR0vE9gThr4  
## 4 The Chainsmokers 60 1nqYsOef1yKKuGOVchbsk6  
## 5 Lewis Capaldi 69 7m7vv9wlQ4i0LFuJiE2zsQ  
## 6 Ed Sheeran 67 2yiy9cd2QktrNvWC2EUi0k  
## track\_album\_name  
## 1 I Don't Care (with Justin Bieber) [Loud Luxury Remix]  
## 2 Memories (Dillon Francis Remix)  
## 3 All the Time (Don Diablo Remix)  
## 4 Call You Mine - The Remixes  
## 5 Someone You Loved (Future Humans Remix)  
## 6 Beautiful People (feat. Khalid) [Jack Wins Remix]  
## track\_album\_release\_date playlist\_name playlist\_id playlist\_genre  
## 1 6/14/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 2 12/13/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 3 7/5/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 4 7/19/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 5 3/5/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 6 7/11/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## playlist\_subgenre danceability energy key loudness mode speechiness  
## 1 dance pop 0.748 0.916 6 -2.634 1 0.0583  
## 2 dance pop 0.726 0.815 11 -4.969 1 0.0373  
## 3 dance pop 0.675 0.931 1 -3.432 0 0.0742  
## 4 dance pop 0.718 0.930 7 -3.778 1 0.1020  
## 5 dance pop 0.650 0.833 1 -4.672 1 0.0359  
## 6 dance pop 0.675 0.919 8 -5.385 1 0.1270  
## acousticness instrumentalness liveness valence tempo duration\_ms  
## 1 0.1020 0.00e+00 0.0653 0.518 122.036 194754  
## 2 0.0724 4.21e-03 0.3570 0.693 99.972 162600  
## 3 0.0794 2.33e-05 0.1100 0.613 124.008 176616  
## 4 0.0287 9.43e-06 0.2040 0.277 121.956 169093  
## 5 0.0803 0.00e+00 0.0833 0.725 123.976 189052  
## 6 0.0799 0.00e+00 0.1430 0.585 124.982 163049

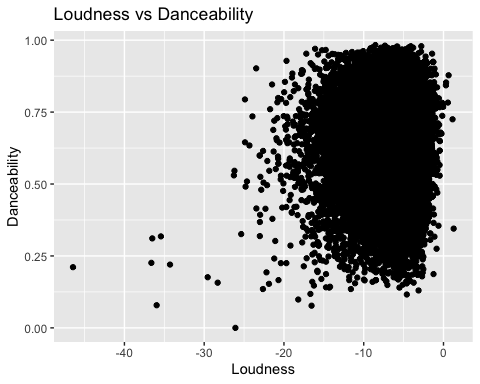
# Scatter plot: Popularity vs Duration  
ggplot(Spotify, aes(x = track\_popularity, y = duration\_ms)) +  
 geom\_point() +  
 labs(x = "Track Popularity", y = "Duration (ms)") +  
 ggtitle("Popularity vs Duration")



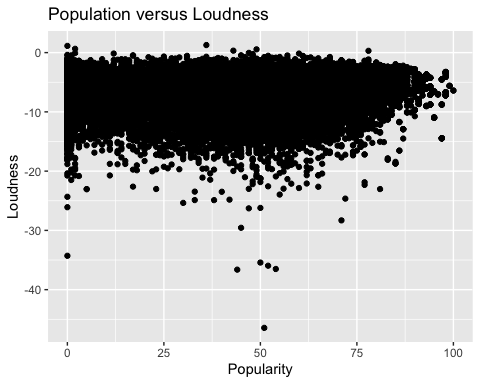
# Load required packages  
library(ggplot2)  
library(dplyr)  
  
# Load the Spotifyset  
Spotify <- read.csv("spotify\_songs.csv") # Replace "Spotify.csv" with the actual file name and path  
  
# Explore the Spotifyset  
head(Spotify) # Check the structure and contents of the Spotifyset

## track\_id track\_name  
## 1 6f807x0ima9a1j3VPbc7VN I Don't Care (with Justin Bieber) - Loud Luxury Remix  
## 2 0r7CVbZTWZgbTCYdfa2P31 Memories - Dillon Francis Remix  
## 3 1z1Hg7Vb0AhHDiEmnDE79l All the Time - Don Diablo Remix  
## 4 75FpbthrwQmzHlBJLuGdC7 Call You Mine - Keanu Silva Remix  
## 5 1e8PAfcKUYoKkxPhrHqw4x Someone You Loved - Future Humans Remix  
## 6 7fvUMiyapMsRRxr07cU8Ef Beautiful People (feat. Khalid) - Jack Wins Remix  
## track\_artist track\_popularity track\_album\_id  
## 1 Ed Sheeran 66 2oCs0DGTsRO98Gh5ZSl2Cx  
## 2 Maroon 5 67 63rPSO264uRjW1X5E6cWv6  
## 3 Zara Larsson 70 1HoSmj2eLcsrR0vE9gThr4  
## 4 The Chainsmokers 60 1nqYsOef1yKKuGOVchbsk6  
## 5 Lewis Capaldi 69 7m7vv9wlQ4i0LFuJiE2zsQ  
## 6 Ed Sheeran 67 2yiy9cd2QktrNvWC2EUi0k  
## track\_album\_name  
## 1 I Don't Care (with Justin Bieber) [Loud Luxury Remix]  
## 2 Memories (Dillon Francis Remix)  
## 3 All the Time (Don Diablo Remix)  
## 4 Call You Mine - The Remixes  
## 5 Someone You Loved (Future Humans Remix)  
## 6 Beautiful People (feat. Khalid) [Jack Wins Remix]  
## track\_album\_release\_date playlist\_name playlist\_id playlist\_genre  
## 1 6/14/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 2 12/13/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 3 7/5/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 4 7/19/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 5 3/5/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## 6 7/11/19 Pop Remix 37i9dQZF1DXcZDD7cfEKhW pop  
## playlist\_subgenre danceability energy key loudness mode speechiness  
## 1 dance pop 0.748 0.916 6 -2.634 1 0.0583  
## 2 dance pop 0.726 0.815 11 -4.969 1 0.0373  
## 3 dance pop 0.675 0.931 1 -3.432 0 0.0742  
## 4 dance pop 0.718 0.930 7 -3.778 1 0.1020  
## 5 dance pop 0.650 0.833 1 -4.672 1 0.0359  
## 6 dance pop 0.675 0.919 8 -5.385 1 0.1270  
## acousticness instrumentalness liveness valence tempo duration\_ms  
## 1 0.1020 0.00e+00 0.0653 0.518 122.036 194754  
## 2 0.0724 4.21e-03 0.3570 0.693 99.972 162600  
## 3 0.0794 2.33e-05 0.1100 0.613 124.008 176616  
## 4 0.0287 9.43e-06 0.2040 0.277 121.956 169093  
## 5 0.0803 0.00e+00 0.0833 0.725 123.976 189052  
## 6 0.0799 0.00e+00 0.1430 0.585 124.982 163049

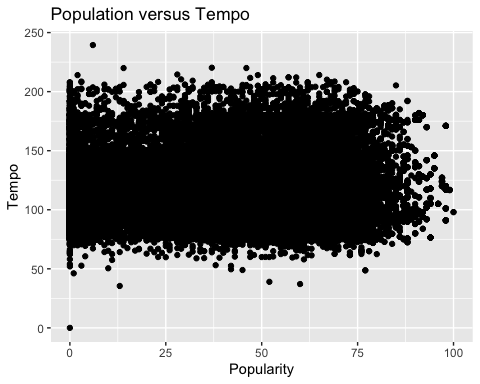
# Scatter plot: Popularity vs Duration  
ggplot(Spotify, aes(x = loudness, y = danceability)) +  
 geom\_point() +  
 labs(x = "Loudness", y = "Danceability") +  
 ggtitle("Loudness vs Danceability")



# Read the dataset  
Spotify <- read.csv("spotify\_songs.csv")  
  
# Create a scatter plot of population versus danceability  
ggplot(Spotify, aes(x = track\_popularity, y = loudness)) +  
 geom\_point() +  
 labs(x = "Popularity", y = "Loudness") +  
 ggtitle("Population versus Loudness")

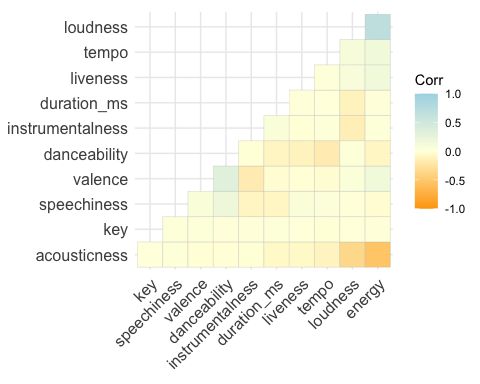


# Read the dataset  
Spotify <- read.csv("spotify\_songs.csv")  
  
# Create a scatter plot of population versus danceability  
ggplot(Spotify, aes(x = track\_popularity, y = tempo)) +  
 geom\_point() +  
 labs(x = "Popularity", y = "Tempo") +  
 ggtitle("Population versus Tempo")



Though the structure of each song is in some way unique, there are definitely some common threads happening. Let us check for the correlation between various attributes of a song.

# Extract relevant columns (attributes)  
attributes <- Spotify[c("acousticness", "loudness", "valence", "danceability", "liveness", "energy", "instrumentalness","key","tempo","duration\_ms","speechiness")]  
  
# Create a correlation matrix  
att\_cor <- cor(attributes)  
  
# Plot the correlation matrix using ggcorrplot  
ggcorrplot(att\_cor, type = "lower", hc.order = TRUE, colors = c("orange", "lightyellow", "lightblue"))

 From the correlation plot, we can observe that:

There exists a high positive correlation between energy and loudness.

There exists a high negative correlation between energy and acousticness.

There are moderate correlation between loudness and acousticness, and between valence and danceability.

We can also observe that speechiness, tempo and key have no strong correlation with track popularity. Thus, we can conclude that popularity is influenced by the following charateristics:

acousticness loudness valence danceability liveness energy instrumentalness This study can be helpful to us when we try to build a predictive model.

In this section, We trying to come up with a model which can predict the popularity of a song given all other attributes. More particulary, the model can help to predict in which popularity class: low,medium or high does the song feature by comparing its other attributes.

Logistic Regression with multinomial(NNET) variables

We can make use of a logistic regression with multinomial variables as there are three different popularity classes in our response variable. We have seen from the correlation plot during our exploratory data analysis that the track popularity has correlation with variables : acousticness, loudness, valence, danceability, liveness, energy and instrumentalness. So it is a good idea to build the model by fitting the popularity class with all these attributes. First step is to randomly split the whole dataset into training (75%) and testing (25%) set for model validation. I would train the model with the training set and then test the perdictive capability of the model using the testing set.

Spotify <- Spotify %>%mutate(popularity = case\_when(track\_popularity <= 30 ~ "low",  
 track\_popularity > 30 & track\_popularity <= 75 ~ "medium",  
 track\_popularity > 75 ~ "high"))

spotify\_train <- Spotify[c(12:15,17,18:21,22:24)]  
set.seed(123)  
train\_idx <- sample(nrow(spotify\_train), .70\*nrow(spotify\_train))  
  
train <- spotify\_train[train\_idx,]  
test <- spotify\_train[-train\_idx,]

Now , let us perform the model fitting and analysis: When we build logistic models we need to set one of the levels of the dependent variable as a baseline. We achieve this by using relevel() function.

# Setting the baseline   
train$popularity <- relevel(factor(train$popularity), ref = "low")

Once the baseline has been specified, we use multinom() function to fit the model and then use summary() function to explore the beta coefficients of the model.

# Fit multinomial logistic regression model using nnet  
nnet\_model <- multinom(popularity ~ ., data = train, MaxNWts = 10000)

## # weights: 39 (24 variable)  
## initial value 25249.406230   
## iter 10 value 19934.785195  
## iter 20 value 19317.181999  
## iter 30 value 18967.065686  
## final value 18966.931980   
## converged

# View the summary of the model  
summary(nnet\_model)

## Call:  
## multinom(formula = popularity ~ ., data = train, MaxNWts = 10000)  
##   
## Coefficients:  
## (Intercept) danceability energy key loudness speechiness  
## high 4.809515 1.04371496 -5.455723 0.013874612 0.33368610 -0.4953364  
## medium 2.776981 0.09018309 -1.366784 0.003423022 0.05690381 -0.5774267  
## acousticness instrumentalness liveness valence tempo  
## high 0.3634678 -3.4531041 -0.7616111 0.3151996 0.002773656  
## medium 0.4702569 -0.5508969 -0.2649272 0.1121809 0.001109072  
## duration\_ms  
## high -5.484978e-06  
## medium -3.892645e-06  
##   
## Std. Errors:  
## (Intercept) danceability energy key loudness  
## high 3.189888e-07 2.019264e-07 2.462579e-07 1.56557e-06 1.393468e-06  
## medium 1.469817e-06 8.936189e-07 1.187271e-06 7.35239e-06 6.289533e-06  
## speechiness acousticness instrumentalness liveness valence  
## high 5.950934e-08 4.809681e-08 5.547265e-09 5.777233e-08 1.662807e-07  
## medium 2.305388e-07 1.973743e-07 8.739571e-08 3.021510e-07 7.455947e-07  
## tempo duration\_ms  
## high 8.147455e-05 1.320738e-07  
## medium 3.183608e-04 1.687891e-07  
##   
## Residual Deviance: 37933.86   
## AIC: 37981.86

The output of summary contains the table for coefficients and a table for standard error. Each row in the coefficient table corresponds to the model equation. This ratio of the probability of choosing other popularity classes over the baseline class that is “low” is referred to as relative risk (often described as odds). However, the output of the model is the log of odds. To get the relative risk IE odds ratio, we need to exponentiate the coefficients.

# Extracting coefficients and exponentiating  
nnet\_coefficients <- coef(nnet\_model)  
nnet\_odds\_ratios <- exp(nnet\_coefficients)  
  
# Print the exponentiated coefficients  
print(nnet\_odds\_ratios)

## (Intercept) danceability energy key loudness speechiness  
## high 122.67216 2.839747 0.004271785 1.013971 1.396105 0.6093659  
## medium 16.07042 1.094375 0.254925360 1.003429 1.058554 0.5613410  
## acousticness instrumentalness liveness valence tempo duration\_ms  
## high 1.438309 0.03164725 0.4669136 1.370533 1.002778 0.9999945  
## medium 1.600405 0.57643258 0.7672618 1.118715 1.001110 0.9999961

The relative risk ratio for a one-unit increase in the variables for being in high and medium popularity classes vs. low popularity class is shown in the above output. Here a value of 1 represents that there is no change. However, a value greater than 1 represents an increase and value less than 1 represents a decrease. We can also use probabilities to understand our model.

# Assuming nnet\_model is your fitted multinomial logistic regression model using nnet  
predicted\_probs <- predict(nnet\_model, type = "probs", newdata = train)  
  
# Display the head of the probability table  
head(predicted\_probs)

## low high medium  
## 2986 0.2728531 0.08295117 0.6441958  
## 29925 0.3920592 0.01091166 0.5970291  
## 29710 0.2770265 0.07331618 0.6496573  
## 2757 0.4086572 0.01701446 0.5743283  
## 9642 0.1973675 0.25185298 0.5507795  
## 31313 0.4418558 0.01422519 0.5439190

The table above indicates that the probability of 2986th obviously being in the medium popularity is 64.41%, it being low popularity is 27.28% and it being high popularity is 0.08%. Thus we can conclude that the 2986th observation is medium popular. On a similar note – 29925th observation is medium popularity, 29710th observations is also medium popularity and so on. We will now check the model accuracy by building classification table. So let us first build the classification table for training dataset and calculate the model accuracy.

# Assuming nnet\_model is your fitted multinomial logistic regression model using nnet  
train$predicted <- predict(nnet\_model, newdata = train, type = "class")  
  
# Building the classification table  
ctable <- table(train$popularity, train$predicted)  
  
# Calculating accuracy - sum of diagonal elements divided by total observations  
accuracy <- sum(diag(ctable)) / sum(ctable)  
  
# Print accuracy (percentage)  
cat("Accuracy:", round(accuracy \* 100, 2), "%\n")

## Accuracy: 62.19 %

Accuracy in training dataset is 62.19%. We now repeat the above on the testing dataset.

# Assuming nnet\_model is your fitted multinomial logistic regression model using nnet  
test$predicted <- predict(nnet\_model, newdata = test, type = "class")  
  
# Building the classification table  
ctable <- table(test$popularity, test$predicted)  
  
# Calculating accuracy - sum of diagonal elements divided by total observations  
accuracy <- sum(diag(ctable)) / sum(ctable)  
  
# Print accuracy (percentage)  
cat("Accuracy:", round(accuracy \* 100, 2), "%\n")

## Accuracy: 59.36 %

We were able to find out a model which predicts the popularity class with a 59.36% accuracy.

k-NN

model\_data <- Spotify %>%  
 mutate(popularity\_gp = case\_when(  
 track\_popularity >= 0 & track\_popularity <= 31 ~ "Least\_Popularity",  
 track\_popularity >= 32 & track\_popularity <= 52 ~ "Average\_Popularity",  
 TRUE ~ "Highest\_Popularity"  
 )) %>%  
 select(where(is.numeric), -c( playlist\_genre, track\_popularity,duration\_ms), popularity\_gp)  
  
model\_data$popularity\_gp = as.factor(model\_data$popularity\_gp)  
str(model\_data)

## 'data.frame': 32833 obs. of 12 variables:  
## $ danceability : num 0.748 0.726 0.675 0.718 0.65 0.675 0.449 0.542 0.594 0.642 ...  
## $ energy : num 0.916 0.815 0.931 0.93 0.833 0.919 0.856 0.903 0.935 0.818 ...  
## $ key : int 6 11 1 7 1 8 5 4 8 2 ...  
## $ loudness : num -2.63 -4.97 -3.43 -3.78 -4.67 ...  
## $ mode : int 1 1 0 1 1 1 0 0 1 1 ...  
## $ speechiness : num 0.0583 0.0373 0.0742 0.102 0.0359 0.127 0.0623 0.0434 0.0565 0.032 ...  
## $ acousticness : num 0.102 0.0724 0.0794 0.0287 0.0803 0.0799 0.187 0.0335 0.0249 0.0567 ...  
## $ instrumentalness: num 0.00 4.21e-03 2.33e-05 9.43e-06 0.00 0.00 0.00 4.83e-06 3.97e-06 0.00 ...  
## $ liveness : num 0.0653 0.357 0.11 0.204 0.0833 0.143 0.176 0.111 0.637 0.0919 ...  
## $ valence : num 0.518 0.693 0.613 0.277 0.725 0.585 0.152 0.367 0.366 0.59 ...  
## $ tempo : num 122 100 124 122 124 ...  
## $ popularity\_gp : Factor w/ 3 levels "Average\_Popularity",..: 2 2 2 2 2 2 2 2 2 2 ...

table(model\_data$popularity\_gp)

##   
## Average\_Popularity Highest\_Popularity Least\_Popularity   
## 9560 12942 10331

set.seed(3245)  
gp <- runif(nrow(model\_data))  
model\_data <- model\_data[order(gp),]  
head(model\_data,5)

## danceability energy key loudness mode speechiness acousticness  
## 1031 0.703 0.885 9 -6.712 1 0.0322 0.02750  
## 12523 0.581 0.854 1 -8.485 0 0.0428 0.00197  
## 22966 0.598 0.526 10 -8.659 0 0.0415 0.12900  
## 12458 0.608 0.768 1 -9.911 1 0.0364 0.10100  
## 13070 0.730 0.785 2 -7.201 1 0.0456 0.04180  
## instrumentalness liveness valence tempo popularity\_gp  
## 1031 2.84e-03 0.2550 0.939 123.997 Average\_Popularity  
## 12523 1.30e-03 0.1110 0.788 131.180 Highest\_Popularity  
## 22966 0.00e+00 0.1400 0.529 123.935 Highest\_Popularity  
## 12458 1.41e-06 0.0942 0.748 132.699 Highest\_Popularity  
## 13070 6.69e-03 0.1230 0.724 137.639 Average\_Popularity

summary(model\_data[,-11])

## danceability energy key loudness   
## Min. :0.0000 Min. :0.000175 Min. : 0.000 Min. :-46.448   
## 1st Qu.:0.5630 1st Qu.:0.581000 1st Qu.: 2.000 1st Qu.: -8.171   
## Median :0.6720 Median :0.721000 Median : 6.000 Median : -6.166   
## Mean :0.6548 Mean :0.698619 Mean : 5.374 Mean : -6.720   
## 3rd Qu.:0.7610 3rd Qu.:0.840000 3rd Qu.: 9.000 3rd Qu.: -4.645   
## Max. :0.9830 Max. :1.000000 Max. :11.000 Max. : 1.275   
## mode speechiness acousticness instrumentalness   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000000   
## 1st Qu.:0.0000 1st Qu.:0.0410 1st Qu.:0.0151 1st Qu.:0.0000000   
## Median :1.0000 Median :0.0625 Median :0.0804 Median :0.0000161   
## Mean :0.5657 Mean :0.1071 Mean :0.1753 Mean :0.0847472   
## 3rd Qu.:1.0000 3rd Qu.:0.1320 3rd Qu.:0.2550 3rd Qu.:0.0048300   
## Max. :1.0000 Max. :0.9180 Max. :0.9940 Max. :0.9940000   
## liveness valence popularity\_gp   
## Min. :0.0000 Min. :0.0000 Average\_Popularity: 9560   
## 1st Qu.:0.0927 1st Qu.:0.3310 Highest\_Popularity:12942   
## Median :0.1270 Median :0.5120 Least\_Popularity :10331   
## Mean :0.1902 Mean :0.5106   
## 3rd Qu.:0.2480 3rd Qu.:0.6930   
## Max. :0.9960 Max. :0.9910

normalize <- function(x) {  
 return((x - min(x))/(max(x) - min(x)))  
}  
  
model\_norm <- model\_data  
model\_norm$popularity\_gp <- NULL  
model\_norm <- as.data.frame(lapply(model\_norm,normalize))  
summary(model\_norm)

## danceability energy key loudness   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.5727 1st Qu.:0.5809 1st Qu.:0.1818 1st Qu.:0.8021   
## Median :0.6836 Median :0.7210 Median :0.5455 Median :0.8441   
## Mean :0.6662 Mean :0.6986 Mean :0.4886 Mean :0.8325   
## 3rd Qu.:0.7742 3rd Qu.:0.8400 3rd Qu.:0.8182 3rd Qu.:0.8760   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## mode speechiness acousticness instrumentalness   
## Min. :0.0000 Min. :0.00000 Min. :0.00000 Min. :0.0000000   
## 1st Qu.:0.0000 1st Qu.:0.04466 1st Qu.:0.01519 1st Qu.:0.0000000   
## Median :1.0000 Median :0.06808 Median :0.08089 Median :0.0000162   
## Mean :0.5657 Mean :0.11663 Mean :0.17639 Mean :0.0852587   
## 3rd Qu.:1.0000 3rd Qu.:0.14379 3rd Qu.:0.25654 3rd Qu.:0.0048592   
## Max. :1.0000 Max. :1.00000 Max. :1.00000 Max. :1.0000000   
## liveness valence tempo   
## Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.09307 1st Qu.:0.3340 1st Qu.:0.4175   
## Median :0.12751 Median :0.5166 Median :0.5095   
## Mean :0.19094 Mean :0.5152 Mean :0.5048   
## 3rd Qu.:0.24900 3rd Qu.:0.6993 3rd Qu.:0.5593   
## Max. :1.00000 Max. :1.0000 Max. :1.0000

set.seed(123)  
train\_idx <- sample(nrow(model\_norm), .80\*nrow(model\_norm))  
  
model\_train <- model\_norm[train\_idx,]  
model\_test <- model\_norm[-train\_idx,]  
model\_train\_target <- model\_data[train\_idx,11]  
model\_test\_target <- model\_data[-train\_idx,11]  
  
sqrt(nrow(model\_data))

## [1] 181.1988

sum(diag(cm))/length(model\_test\_target)

## Error in as.integer(x): cannot coerce type 'closure' to vector of type 'integer'

Implications

We considered this project to be helpful for artists to understand what their audience is looking for and help them improve the popularity of their tracks. It was also meant to help music distributors to streamline their music library. The observations we found out from analysis can be used by an artist to improve the popularity of their songs. Creating songs with shorter duration or highly danceable songs have more chance to gain popularity.Maybe even the title of a song might affect the popularity of a song.Artists can try including common words like “Love”,“Like” etc which we found in most of the popular song titles. Maybe those words can help them to be featured in popular playlists. Music distributors could focus more on the genres which are popular among spotify users of current generation.Also, the genre R&B looks to gain popularity over the years. Hence, R&B artists can be collaborated for more works. Also more playlists related to danceable songs can also be included considering the popularity of danceable songs. Users can make use of our Song Recommendation Engine to get recommendations as per their preferences. Limitations

Even though spotify features over a 50 million songs, we are performing our analysis on a dataset with around 32k records.Using a dynamic dataset can improve the results of the analysis. Additional attributes can also be considered which can help our analysis like including the number of times a particular song has played or the most downloaded playlists. The dataset doesnot include any demographic attribute. Popularity of songs can be affected by the demography of the listeners. People in different countries might have different music tastes. A demographic data can provide more insights. We have tried a linear regression model here.A clustering or neural network analysis can also be used and tried to develop a better model. We have not considered multicollinearity of the variables while developing the model as the correlation is not that high .But if we can work with much larger dataset and find considerable collinearity between variables , we can take into account multicollinearity effect and try to remove it while building the model.