```
# Package library
#install.packages("rpart")
library(rpart)
#install.packages("kernlab")
library(kernlab)
#install.packages("caret")
library(caret)
#install.packages("lattice")
library(lattice)
#install.packages("tidyverse")
library(tidyverse)
#install.packages("dplyr")
library(dplyr)
#install.packages("ggplot2")
library(ggplot2)
#install.packages("corrplot")
library(corrplot)
#install.packages("quanteda")
library(quanteda)
#install.packages("quanteda.textplots")
library(quanteda.textplots)

# Convert Spotify CSV into dataframe
Spotify_data_raw <- read.csv("C:/Users/gsgro/OneDrive/Desktop/Syr_MSBA/Term 2/Data Science/Final
Project/Spotify Dataset.csv")

# Review Spotify raw data
summary(Spotify_data_raw)
str(Spotify_data_raw)
head(Spotify_data_raw)

# Update column names
spotify_data_clean <- Spotify_data_raw %>%
  rename(Song_Title = track_name,
         Artist_Name = artist.s._name,
         Artist_Count = artist_count,
         Song_Release_Year = released_year,
         Song_Release_Month = released_month,
         Song_Release_Day = released_day,
         Spotify_Playlist_Count = in_spotify_playlists,
         Spotify_Charts_Count = in_spotify_charts,
         Total_Streams = streams,
         Apple_Playlist_Count = in_apple_playlists,
         Apple_Charts_Count = in_apple_charts,
```

```
        Deezer_Playlist_Count = in_deezer_playlists,
        Deezer_Charts_Count = in_deezer_charts,
        Shazam_Charts_Count = in_shazam_charts,
        Beats_per_Minute = bpm,
        Key = key,
        Mode = mode,
        Danceability = danceability_.,
        Valence = valence_.,
        Energy = energy_.,
        Acousticness = acousticness_.,
        Instrumentalness = instrumentalness_.,
        Liveness = liveness_.,
        Speechiness = speechiness_.)

# Combine Release year, month, and day into one column
spotify_data_clean$Song_Release_Date <- as.Date(paste(spotify_data_clean$Song_Release_Year,
                                        spotify_data_clean$Song_Release_Month,
                                        spotify_data_clean$Song_Release_Day, sep = "-"))

# Delete Release year, month, and day from dataframe
spotify_data_clean <- spotify_data_clean %>%
  select(-Song_Release_Year, -Song_Release_Month, -Song_Release_Day)

# Convert from character to numeric
spotify_data_clean$Total_Streams <- as.numeric(spotify_data_clean$Total_Streams)
spotify_data_clean$Deezer_Playlist_Count <- as.numeric(spotify_data_clean$Deezer_Playlist_Count)
spotify_data_clean$Shazam_Charts_Count <- as.numeric(spotify_data_clean$Shazam_Charts_Count)

# Add column that calculates how many days its been since the song was released
spotify_data_clean$Days_Since_Today <- as.numeric(Sys.Date() -
spotify_data_clean$Song_Release_Date)

# Remove unnecessary columns
spotify_data_clean <- subset(spotify_data_clean, select = -c(Artist_Count))

# Rearrange column order
spotify_data_clean <- spotify_data_clean[, c("Song_Title",
                                    "Artist_Name",
                                    "Song_Release_Date",
                                    "Days_Since_Today",
                                    "Spotify_Playlist_Count",
                                    "Spotify_Charts_Count",
                                    "Total_Streams",
                                    "Apple_Playlist_Count",
```

```
                                "Apple_Charts_Count",
                                "Deezer_Playlist_Count",
                                "Deezer_Charts_Count",
                                "Shazam_Charts_Count",
                                "Mode",
                                "Key",
                                "Beats_per_Minute",
                                "Danceability",
                                "Valence",
                                "Energy",
                                "Acousticness",
                                "Instrumentalness",
                                "Liveness",
                                "Speechiness")]

# NA Check and Removal
na_indices <- which(is.na(spotify_data_clean$Total_Streams))
na_indices

spotify_data_clean$Total_Streams[is.na(spotify_data_clean$Total_Streams)] <- 0

# Review Spotify clean data
summary(spotify_data_clean)
str(spotify_data_clean)
head(spotify_data_clean)

# Create numeric DF to user for correlation analysis
Spotify_data_correlation <- subset(spotify_data_clean, select = -c(Song_Title,
                                                                   Artist_Name,
                                                                   Song_Release_Date,
                                                                   Days_Since_Today,
                                                                   Spotify_Playlist_Count,
                                                                   Spotify_Charts_Count,
                                                                   Apple_Playlist_Count,
                                                                   Apple_Charts_Count,
                                                                   Deezer_Playlist_Count,
                                                                   Deezer_Charts_Count,
                                                                   Shazam_Charts_Count,
                                                                   Mode,
                                                                   Key))

#Review Metrics
```

```
hist(spotify_data_clean$Speechiness ,xlab='Speechiness ',main='Histogram of Spotify Top 1000 Songs Speechiness Metric', col="skyblue1",breaks = seq(min(spotify_data_clean$Speechiness), max(spotify_data_clean$Speechiness)), length.out = 7, probability = TRUE)
abline(v=mean(spotify_data_clean$Speechiness ), col="black", lwd=3)

hist(spotify_data_clean$Beats_per_Minute ,xlab='Beats per Minute',main='Histogram of Spotify Top 1000 Songs Beats per Minute Metric', col="skyblue1", probability = TRUE)
abline(v=mean(spotify_data_clean$Beats_per_Minute ), col="black", lwd=3)

hist(spotify_data_clean$Energy ,xlab='Energy',main='Histogram of Spotify Top 1000 Songs Energy Metric', col="skyblue1", probability = TRUE)
abline(v=mean(spotify_data_clean$Energy ), col="black", lwd=3)

hist(spotify_data_clean$Acousticness,xlab='Acousticness',main='Histogram of Spotify Top 1000 Songs Acousticness Metric', col="skyblue1", probability = TRUE)
abline(v=mean(spotify_data_clean$Acousticness ), col="black", lwd=3)

# Correlation analysis - all metrics
correlation_analysis1 <- cor(Spotify_data_correlation)
correlation_analysis1

# Correlation analysis - select metrics
Spotify_data_correlation_select <- subset(Spotify_data_correlation, select = -c(Danceability,
                                                                                Valence,
                                                                                Instrumentalness,
                                                                                Liveness))

correlation_analysis2 <- cor(Spotify_data_correlation_select)
correlation_analysis2

# Linear Regression with all variables
linear_regression1 <- lm(formula = Total_Streams ~ Beats_per_Minute + Danceability + Valence + Energy + Acousticness + Instrumentalness + Liveness + Speechiness, data = spotify_data_clean)
summary(linear_regression1)

# Graph Actual vs Predicted (linear regression 1)
predicted_values1 <- data.frame(Observed = spotify_data_clean$Total_Streams, Predicted = predict(linear_regression1))

ggplot(predicted_values1, aes(x = Observed, y = Predicted)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Scatter Plot of Observed vs Predicted Values", x = "Actual Total Streams", y = "Predicted Total Streams")
```

```
# Revised linear regression model with 0 intercept
linear_regression2 <- lm(formula = Total_Streams ~ 0 + Beats_per_Minute + Danceability + Valence +
Energy + Acousticness + Instrumentalness + Liveness + Speechiness, data = spotify_data_clean)
summary(linear_regression2)

predicted_values2 <- data.frame(Observed = spotify_data_clean$Total_Streams, Predicted =
predict(linear_regression2))

ggplot(predicted_values2, aes(x = Observed, y = Predicted)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Scatter Plot of Observed vs Predicted Values", x = "Actual Total Streams", y = "Predicted
Total Streams")

# Revised linear regression with selected factors that we find to be significant with 0 intercept
linear_regression3 <- lm(formula = Total_Streams ~ 0 + Beats_per_Minute + Energy + Acousticness +
Speechiness, data = spotify_data_clean)
summary(linear_regression3)

predicted_values3 <- data.frame(Observed = spotify_data_clean$Total_Streams, Predicted =
predict(linear_regression3))

ggplot(predicted_values3, aes(x = Observed, y = Predicted)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Scatter Plot of Observed vs Predicted Values", x = "Actual Total Streams", y = "Predicted
Total Streams")

# With a high stream amount and the variables in % form, the log of total streams was taken
linear_regression4 <- lm(formula = log1p(Total_Streams) ~ 0 + Beats_per_Minute + Energy +
Acousticness + Speechiness, data = spotify_data_clean)
summary(linear_regression4)

# Examine difference in charts vs streams and compare to other platforms
# Spotify
linear_regression5 <- lm(formula = Spotify_Charts_Count ~ 0 + Beats_per_Minute + Energy +
Acousticness + Speechiness, data = spotify_data_clean)
summary(linear_regression5)

# Apple
linear_regression6 <- lm(formula = Apple_Charts_Count ~ 0 + Beats_per_Minute + Energy +
Acousticness + Speechiness, data = spotify_data_clean)
summary(linear_regression6)
```

```
# Deezer
linear_regression7 <- lm(formula = Deezer_Charts_Count ~ 0 + Beats_per_Minute + Energy +
Acousticness + Speechiness, data = spotify_data_clean)
summary(linear_regression7)

# Review taking the log of charts vs streams and compare to other platforms
# Spotify
linear_regression8<- lm(formula = log1p(Spotify_Charts_Count) ~ 0 + Beats_per_Minute + Energy +
Acousticness +   Speechiness, data = spotify_data_clean)
summary(linear_regression8)

# Apple
linear_regression9 <- lm(formula = log1p(Apple_Charts_Count)~ 0+ Beats_per_Minute + Energy +
Acousticness +   Speechiness, data = spotify_data_clean)
summary(linear_regression9)

# Deezer
linear_regression10 <- lm(formula = log1p(Deezer_Charts_Count)~ 0+ Beats_per_Minute + Energy +
Acousticness +   Speechiness, data = spotify_data_clean)
summary(linear_regression10)

# Word cloud
spotify_corpus <- corpus(spotify_data_clean$Song_Title)
sotify_corpus <- tokens(spotify_corpus) %>%
  tokens_remove(stopwords("en"))
print(spotify_corpus)

spotify_matrix <- dfm(spotify_corpus)
head(spotify_matrix)

#Reviewed to remove additional words, Spanish stopword, errors, explicit
spotify_dtm <- dfm(spotify_corpus, stem = TRUE, remove_punct = TRUE,
remove=c(stopwords("english"),stopwords("spanish"), "que", "un", "la", "feat", "remix", "explicit", "ver",
"", "vol", "bzrp", "Taylor", "bts"))
spotify_dtm
spotify_dtm <- dfm_trim(spotify_dtm, min_termfreq = 2)
spotify_dtm

#Wordcloud
textplot_wordcloud(spotify_dtm)

#view(spotify_data_clean)
webWords <- as.matrix(spotify_dtm)
```

```
str(webWords)

# Word counts
wordCounts <- colSums(webWords)
wordCounts <- sort(wordCounts, decreasing = TRUE)
head(wordCounts, 10)

# Max streams
maxStreams <- which.max(spotify_data_clean$Total_Streams)
maxStreams
maxStreams_row <- spotify_data_clean[maxStreams, ]
maxStreams_row

# Top 10 songs
top_songs <- head(spotify_data_clean[order(-spotify_data_clean$Total_Streams), ], 10)
top_songs

# Bar chart for the top 10 songs
ggplot(top_songs, aes(x = Song_Title, y = Total_Streams, fill = Song_Title)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  ggtitle("Top 10 Songs Based on Total Streams") +
  xlab("Song") +
  ylab("Total Streams")

#Top 10 songs by Song count
top_artists <- spotify_data_clean %>% count(Artist_Name) %>% arrange(desc(n)) %>% top_n(10)
ggplot(top_artists, aes(x = reorder(Artist_Name, -n), y = n)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 10 Artists in the Dataset", x = "Artist_Name", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


# Top 10 Artists based on Streams
top_artists2 <- spotify_data_clean %>%
  group_by(Artist_Name) %>%
  summarise(Total_Streams = sum(Total_Streams)) %>%
  arrange(desc(Total_Streams)) %>%
  head(10)
top_artists2


# Create a ggplot bar plot
ggplot(top_artists2, aes(x = reorder(Artist_Name, -Total_Streams), y = Total_Streams)) +
```

```
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 10 Artists by Total Streams", x = "Artist", y = "Total Streams") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


##Histograms of the 4 attributes
par(mfrow = c(2, 2))
hist(spotify_data_clean$Beats_per_Minute, main = "Beats_per_Minute", xlab = "BPM", col =
"lightblue")
hist(spotify_data_clean$Energy, main = "Energy", xlab = "Energy_%", col = "lightgreen")
hist(spotify_data_clean$Speechiness, main = "Speech", xlab = "Speech_%", col = "lightcoral")
hist(spotify_data_clean$Acousticness, main = "Acoustics", xlab = "Acoustics_%", col = "lightyellow")
par(mfrow = c(1, 1))

#Mode comparison
Top_mode <-spotify_data_clean %>% count(Mode) %>% arrange(desc(n)) %>% top_n(10)
ggplot(Top_mode, aes(x = reorder(Mode, -n), y = n)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Mode Type ", x = "Mode", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Bar Plots
# Bar plot total stream v Danceability
# Total streams are higher with higher percentage of danceability
ggplot(spotify_data_clean, aes(x = Danceability, y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Streams vs. Danceability", x = "Danceability", y = "Total Streams")


# Barplot  total stream v valence
# Total streams then to fluctuate with mood levels. We have a high peak at 25% and other high peaks
between 50 and 75%
ggplot(spotify_data_clean, aes(x = Valence, y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +

  labs(title = "Total Streams vs. Valence", x = "Valence", y = "Total Streams")

# Barplot total stream v energy
# Total streams increase when energy level increases. The highest streams we between 50 to80 percent in
energy % in a song.

ggplot(spotify_data_clean, aes(x = Energy,
                               y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
```

```
  labs(title = "Total Streams vs. Energy", x = "Energy", y = "Total Streams")

# Barplot total stream v acousticness
# Total streams are higher when Acousticness is at the lowest percentage.
ggplot(spotify_data_clean, aes(x = Acousticness, y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Streams vs.Acousticness ", x = "Acousticness", y = "Total Streams")

# Barplot total streams v Instrumentalness
# Total streams are at its higher when Instrumentalness is at its lowest.
ggplot(spotify_data_clean, aes(x = Instrumentalness , y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Streams vs. Instrumentalness ", x = "Instrumentalness", y = "Total Streams")

# Barplot total streams v liveness
# Total streams are  higher when liveness is lower.
ggplot(spotify_data_clean, aes(x = Liveness , y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Streams vs. Liveness ", x = "Liveness", y = "Total Streams")

# Barplot total streams v speechiness
# Total streams are higher when speech is lower.
ggplot(spotify_data_clean, aes(x = Speechiness , y = Total_Streams)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Total Streams vs. Speechiness ", x = "Speechiness", y = "Total Streams")

# Max Total Streams
which.max(spotify_data_clean$Total_Streams)
spotify_data_clean[56,]
spotify_data_clean [56, 13:20]

# Min in Total Streams
which.min(spotify_data_clean$Total_Streams)
spotify_data_clean[575,]
spotify_data_clean[575, 13:20 ]
spotify_data_clean [575,3]

# Max in Chart counts
which.max(spotify_data_clean$Spotify_Charts_Count)
spotify_data_clean [1,1:2]
spotify_data_clean[1,13:20]

# Min in chart counts
which.min(spotify_data_clean$Spotify_Charts_Count)
```

```
spotify_data_clean [207,1:2]
spotify_data_clean[207,13:20]

# Max in Spotify playlist counts
which.max(spotify_data_clean$Spotify_Playlist_Count)
spotify_data_clean [758, 1:2]
spotify_data_clean [758,  13:20]

which.min(spotify_data_clean$Spotify_Playlist_Count)
spotify_data_clean [95, 1:2]
spotify_data_clean [95,  13:20]

# Max in total streams song is  Blinding Lights  Artist name The Weeknd  release 2019
# BMP 171, Energy is 80 Speechiness is 7 Acoustics is 0
# Max in Spotify Charts count- Seven  by latto, Jung Kook
# BMP 125 Energy 83 Speech 4 Acoustics 31
# Max Spotify playlist counts- Get Lucky by Pharrell Williams
# BMP 116 energy 81 Acoustic 4 Speech 4
# mean based on Max output
mean_bpm<-mean(171,125,116)
mean_energy<-mean(80,83,781)
mean_speech<- mean(7,4,4)
mean_speech
mean_acoustics<- mean (0,31,4)
mean_acoustics
# Linear regression with only significant variables
linear_regression4 <- lm(formula = Total_Streams ~ 0+Beats_per_Minute+Energy+Acousticness+
Speechiness,data=spotify_data_clean)
summary(linear_regression4)
# Prediction
est <- data.frame(Beats_per_Minute = 171, Energy=80,Speechiness = 7, Acousticness =0)
predict(linear_regression4, est, type="response")
# Prediction output 593425615  v current spotify highest stream song of 3703895074
# Min in Total Streams is Love Grows  by Edison Lighthouse  relase date 1970
# BPM 110, Energy 69 Speech is 3 Acoustics 7

# Min in Spotify Chart Counts  -  Hits Different  by Taylor Swift
# BMP 106 energy 78 Speech 4 Acoustics 15
# Min Spotify Playlist – Still with You by Jung Kook
# BMP 88, energy 47 Acoustics 9 Speech 4

# Histograms
hist(spotify_data_clean$Spotify_Charts_Count)
hist(spotify_data_clean$Apple_Charts_Count)
```

```
hist(spotify_data_clean$Deezer_Charts_Count)
hist(spotify_data_clean$Shazam_Charts_Count)

# 4 Chart visual hist on Streaming platforms chart counts
par(mfrow = c(2, 2))
hist(spotify_data_clean$Spotify_Charts_Count, main = "Spotify Charts", xlab = "Charts_Count", col =
"lightblue")
hist(spotify_data_clean$Apple_Charts_Count, main = "Apple Charts", xlab = "Charts_Count", col =
"lightgreen")
hist(spotify_data_clean$Deezer_Charts_Count, main = "Deezer Charts", xlab = "Charts_Count", col =
"lightcoral")
hist(spotify_data_clean$Shazam_Charts_Count, main = "Shazam Charts", xlab = "Charts_Count", col =
"lightyellow")
par(mfrow = c(1, 1))

# Scatterplot with significant attributes
# Sort the data by 'bpm'
sorted_data <- spotify_data_clean %>%arrange(Beats_per_Minute)
#create a new column that combines song title and artist name
sorted_data <- sorted_data %>%
  mutate(Song_Title_and_Artist_Name = paste(Song_Title, ' - ', Artist_Name, sep = ' '))

# Create a scatter plot using ggplot2 Total streams v. Enery v BPM
ggplot(sorted_data, aes(x = Energy, y = Total_Streams, color = Beats_per_Minute)) +
  geom_point(alpha = 0.7) +
  scale_color_gradientn(colors = c("blue", "orange","yellow"), guide = "colorbar") +
  labs(title = 'Streams vs Energy vs BPM',
       x = 'Energy',
       y = 'Total_Streams',
       color = 'Beats_per_minute') +
  theme_minimal()
# Scatterplot for Total streams v Speechiness v Acoustics


sorted_data<- spotify_data_clean %>%arrange(Acousticness)
ggplot(sorted_data, aes(x=Speechiness,y =Total_Streams, color = Acousticness)) +
  geom_point(alpha = 0.7) +
  scale_color_gradientn(colors = c("blue", "orange", "yellow"), guide = "colorbar") +
  labs(title = 'Streams vs Acousticness vs Speech',
       x = 'Speechniess',
       y = 'Total_Streams',
       color = 'Acousticness') +
  theme_minimal()
```