

Name : Ng Ka Ho   StudentID: 22204822

Title: Exploring Music Genre Association

## I.        Introduction

This study investigates the relationship between music characteristics and popularity on Spotify and YouTube. We aim to answer research questions regarding the factors that contribute to a song's success on these platforms.

To achieve our objectives, we collected a dataset of music characteristics, streaming and video metrics, and other variables for a sample of songs.

We conducted regression analysis and clustering to identify common patterns in the data. By analyzing the Spotify and YouTube dataset, we hope to provide insights into music consumption and preferences in the digital age.

## II.       Data product

Our proposed data product includes a classification model to predict song genres based on their musical features, a regression model to analyze the relationship between musical features and views, and a recommendation system that uses Association Rule Mining and collaborative filtering techniques to generate personalized song recommendations.

The classification model can be trained on labeled data to predict new songs, while the regression model can provide valuable insights into the factors that contribute to a song's success on streaming platforms. The recommendation system can identify relationships between songs based on musical features and provide personalized recommendations to users.

This data product could be a powerful tool for music streaming platforms, offering valuable insights for artists and labels to inform marketing strategies, and providing users with a better music discovery experience. By predicting song genres and analyzing the relationship between musical features and views, the data product can help artists and labels understand what makes a song successful on streaming platforms. Additionally, the recommendation system can provide personalized recommendations to users based on their listening preferences, improving the overall user experience, and encouraging more engagement with the platform.

### III. Data Preprocessing

To prepare the data for analysis, we performed the following steps:

#### 1. Data cleaning, imputation, and feature selection:

- We checked for any missing values in the dataset and dropped the rows with missing values by using this command in R, `data <- data[complete.cases(data), ]`
- We selected relevant variables for this project and dropped the ones that were not useful. I drop variables such as: `Url_spotify`, `Uri`, `Description`. These are just some websites, code for the song and description is for the viewer instead of data mining.

#### 2. Normalization of numerical variables

- converting categorical variables to numerical format
  - `Licensed`
  - `official_video`

These two variables only store true/false so we can change it to Boolean value which 0 represents true and 1 represents false.

- standardizing the numerical variables  
This can lead to unbiased or accurate results since ensures that all variables are on the same scale and have equal weight in the analysis. It is useful when we do k-means clustering which is the algorithms based on distance of data.

#### 3. Dimension reduction:

The dataset before t-SNE uses all the numerical variables which have very high dimensions. High dimension data is difficult in visualization and causes redundancy.

High-dimensional data contain redundant variables that make us difficult to identify the most important variables for a given task. It increases training times but does not give us a better model.

I have tried to draw the plot before t-SNE using the features of song like key,

loudness, tempo but it failed. Since my dataset has many variables, it is hard to draw plots with that many variables. But I still want to show the effect of using t-SNE. So, I chose two variables to plot for example in next part.

We used t-SNE to reduce the dimensionality of the data to two components, which we added to the original data table.

#### 4. Visualization

I created two scatterplots to visualize the data before and after t-SNE. These plots helped us visualize the data before and after t-SNE and provided insights into the relationships between the variables.

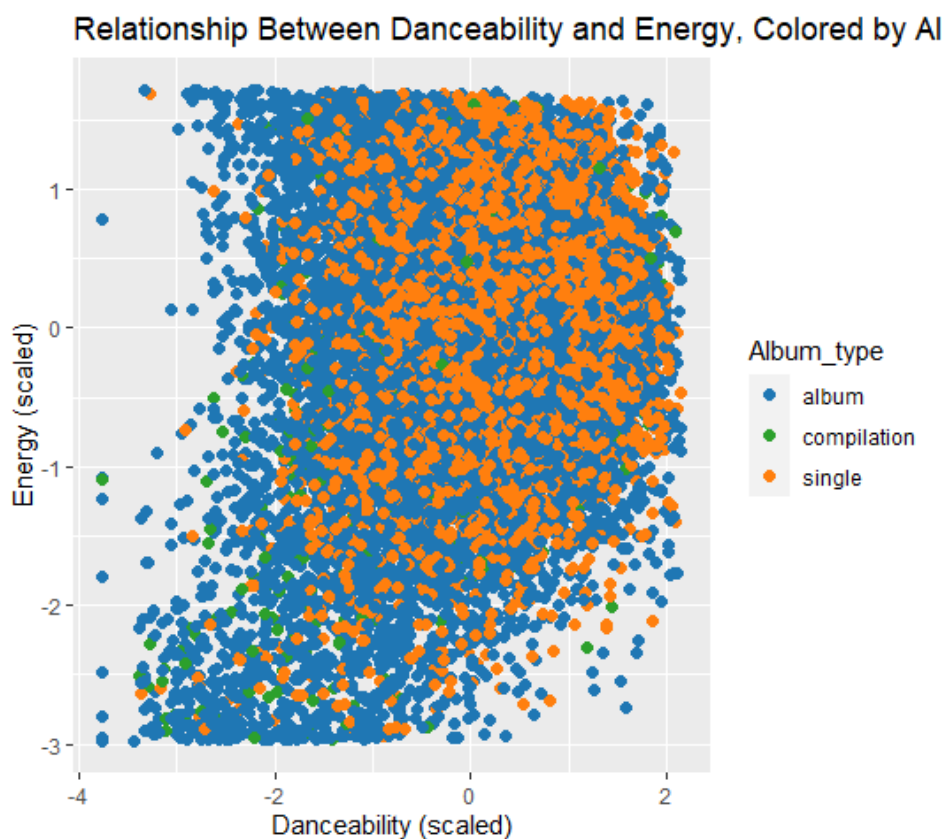


Figure 1. Relationship between danceability and Energy, colored by album type

The scatterplot before t-SNE shows that the "Danceability" and "Energy" variables are somewhat uncorrelated, with data points scattered across the plot. This means there may not be a strong relationship between these two variables, and that they may each be important in predicting the target variable.

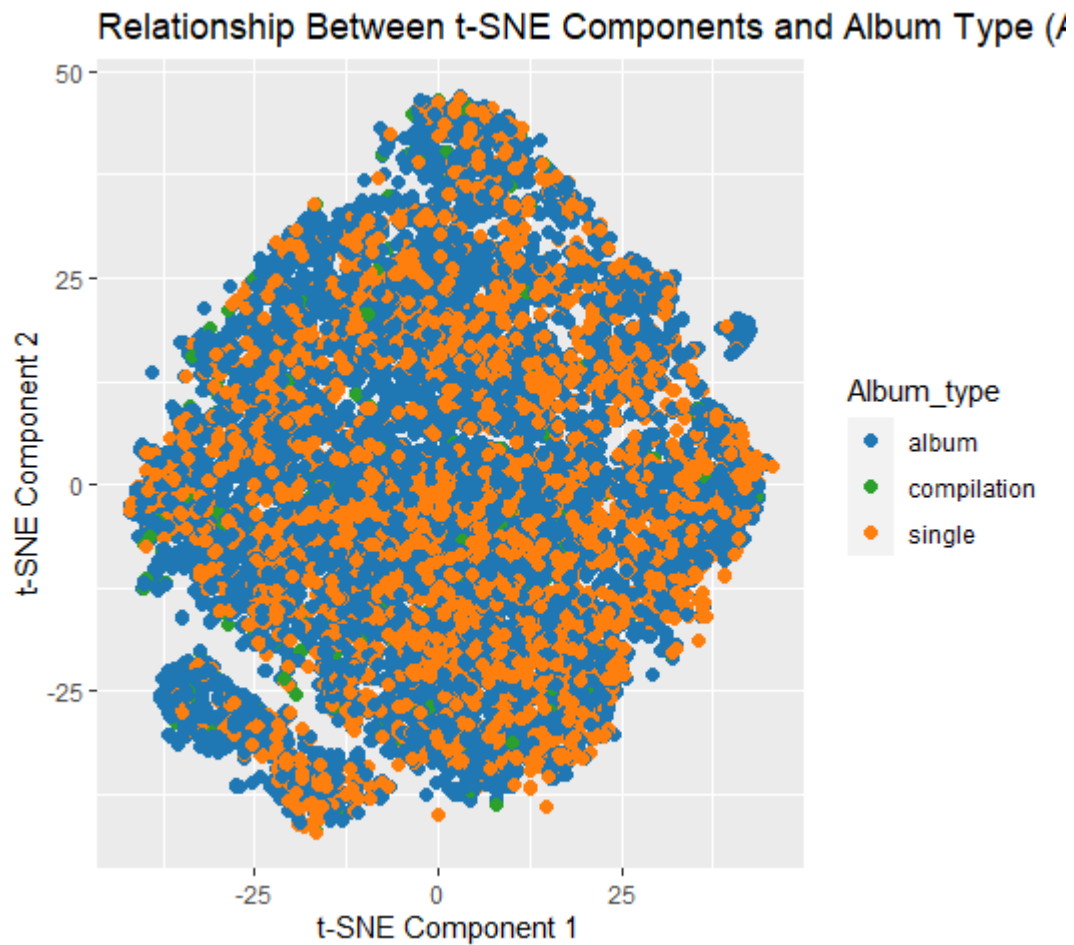


Figure 2. Relationship between t-SNE component1 and 2, colored by album type

t-SNE components 1 and 2 represent the two dimensions of the lower-dimensional space that t-SNE maps the high-dimensional data onto. Each data point is assigned a value for these components based on its position in the high-dimensional space and its similarity to other data points. The t-SNE components capture different aspects or patterns of the data and should be interpreted as a new representation of the data that captures its underlying structure or patterns.

The scatterplot after t-SNE shows that the data points are more concentrated along the t-SNE component 2 axis, with some separation between the different album types. This suggests that the t-SNE algorithm has identified some underlying structure or patterns in the data, and that the t-SNE components may be more useful for predicting the target variable than the original variables.

#### IV. Regression Analysis

We conducted regression analysis to investigate the relationship between music characteristics, views, and stream.

We divided the data into training and testing sets and fitted regression models to the training data. We split the data into training and testing sets and converted the variables to numeric data types. 20% of the data is being used for testing and 80% of the data is being used for training. We then used these models to make predictions on the testing data and computed R-squared values to evaluate the model performance.

We fitted two linear regression models to the training data, with Spotify streams and YouTube views as the dependent variables, and music characteristics variables as independent variables.

Model on views:

Coefficients for views:

```
> print(coef(model1_views))
```

(Intercept)	Danceability	Energy	Key
95189468.50	17075263.99	-11584895.96	-85788.76
Loudness	Speechiness	Acousticness	Instrumentalness
33278512.55	-8931666.12	-5124551.05	-2310476.76
Liveness	Valence	Tempo	
-2958605.88	-4210316.31	-1709043.47	

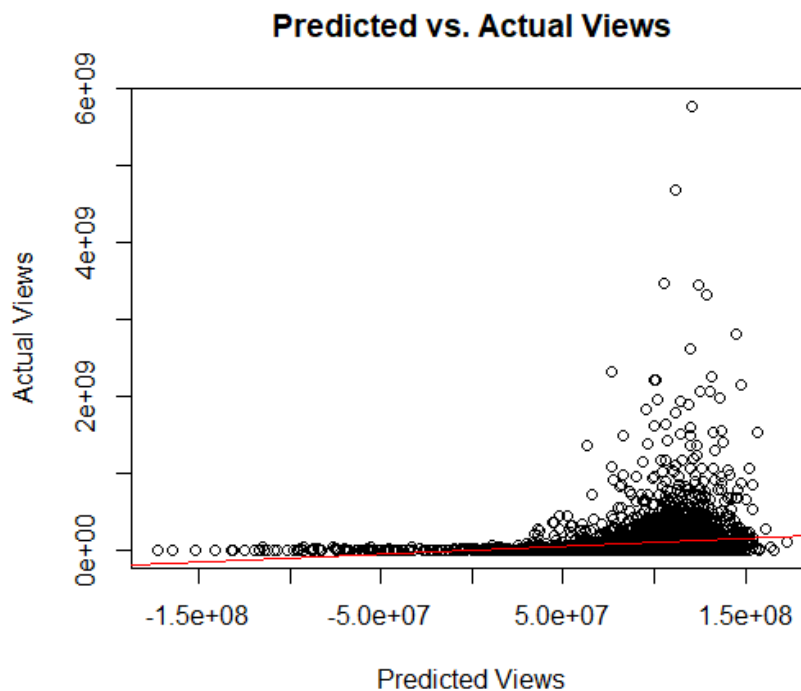


Figure 3. the relationship between the actual and predicted values for the Views model

The scatter plot shows that the model is reasonably accurate, as most of the points fall within a relatively tight band near the diagonal line. However, there are some outliers where the model's prediction is quite far off from the actual value. These outliers may be worth investigating further to see if there are any particular song features or other factors that could be contributing to the large prediction errors.

R-squared value for views: 0.01862232

The low R-squared value of 0.0186 for the Views model suggests that the included independent variables are not very effective at predicting the number of views a song will receive based on its audio features and other factors. This indicates that there may be other important variables in the relationship that are not accounted for by the current model. We all know artists are more important for predicting views.

An example using the views model:

There has been a user who uploaded a new music on YouTube. We have data about features of music. In this example we set feature of music is,  
 Danceability = 0.8, Energy = 0.1, Key = 7, Loudness = 5.2, Speechiness = 0.01,  
 Acousticness = 0.05, Instrumentalness = 0.1, Liveness = 0.1, Valence = 0.6,  
 Tempo = 60

By the coefficients of the model, I can predict how many views will this song have. By using predict function in R. We get.

Predicted number of views for the new song: 174197684

Model on stream:

Coefficients for Stream:

```
> print(coef(model1_stream))
```

(Intercept)	Danceability	Energy	Key
6.652202e-316	5.541992e-317	-1.399462e-316	-1.217984e-317
Loudness	Speechiness	Acousticness	Instrumentalness
1.786112e-316	-4.451867e-317	-1.139867e-316	-2.970428e-317
Liveness	Valence	Tempo	
-3.045391e-317	-6.533066e-317	-5.568130e-318	

R-squared value for Stream: NaN

In this model, the coefficients for the Stream model suggest that most of the variables included in the model have very small or near-zero values. This may indicate that the listener on Spotify is not choosing the music based on music characteristics. Maybe we should explore alternative models using other variables to improve the accuracy and usefulness of the predictions.

Also, we can't calculate the R-square value for this model. If the total sum of squares (SSTOT) is close to zero or the residual sum of squares (SSRES) is close to zero, R shows result NaN.

#### IV. Clustering Analysis

In this section, we aimed to group similar songs together based on their musical features using k-means clustering. We selected the numerical columns related to musical features, such as danceability, energy, loudness, and valence, for clustering.

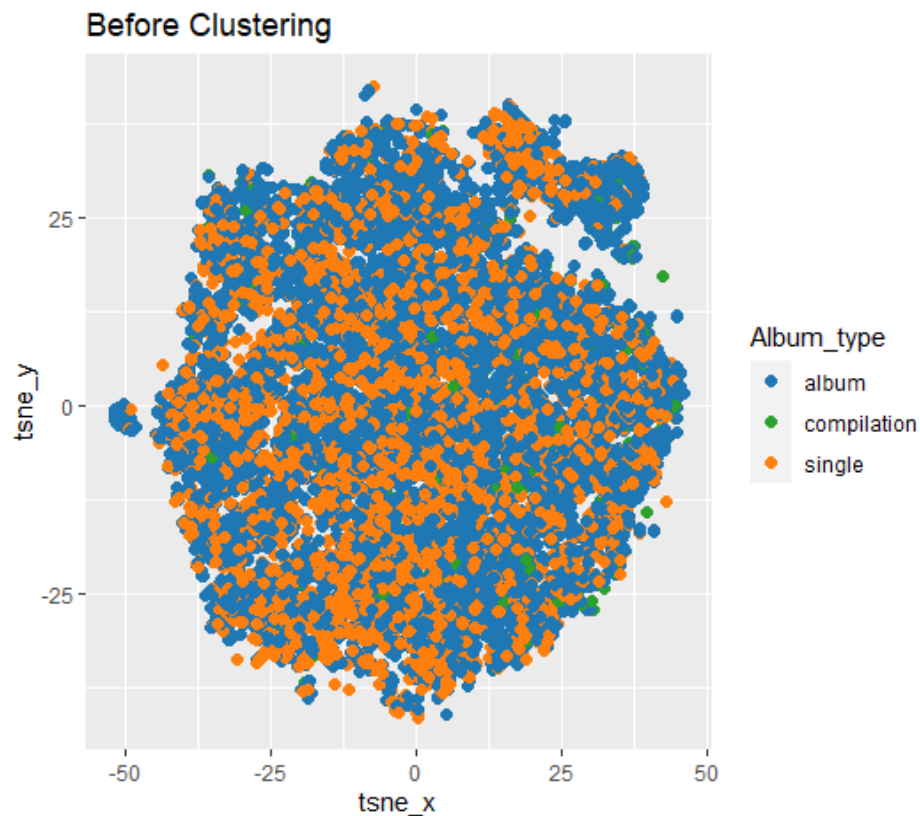


Figure 4. relationship between tsne\_x and tsne\_y before we use K-mean algorithms.

Before clustering, we created a scatterplot of the t-SNE result to visualize the data distribution. Each point represents one song. Then, we performed k-means clustering with  $k = 4$ , using 2 iterations with different starting centroids.



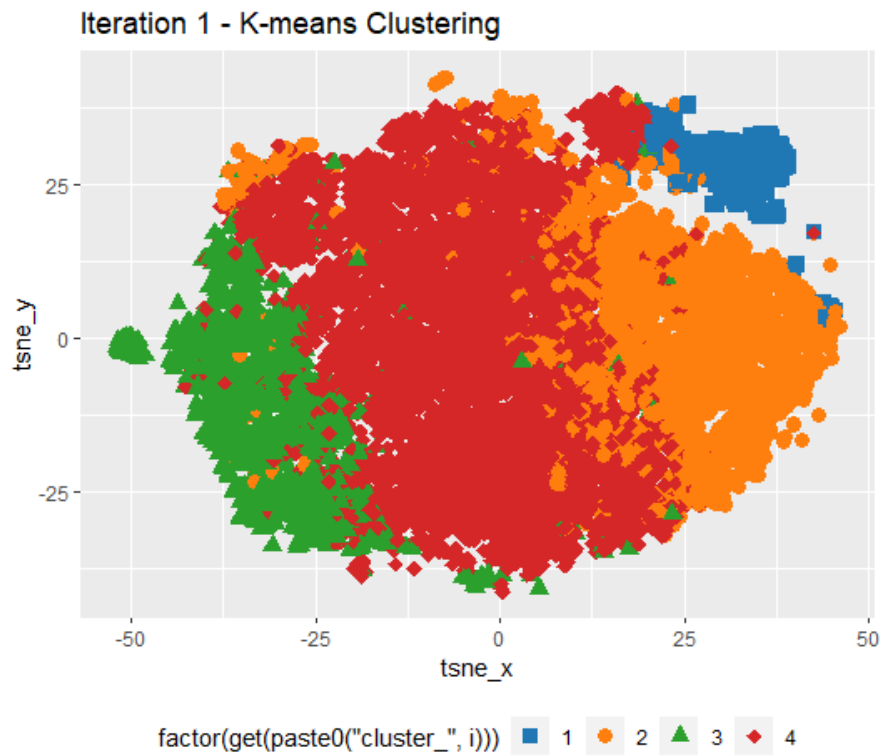


Figure 5. Iteration 1 on K-means Clustering.

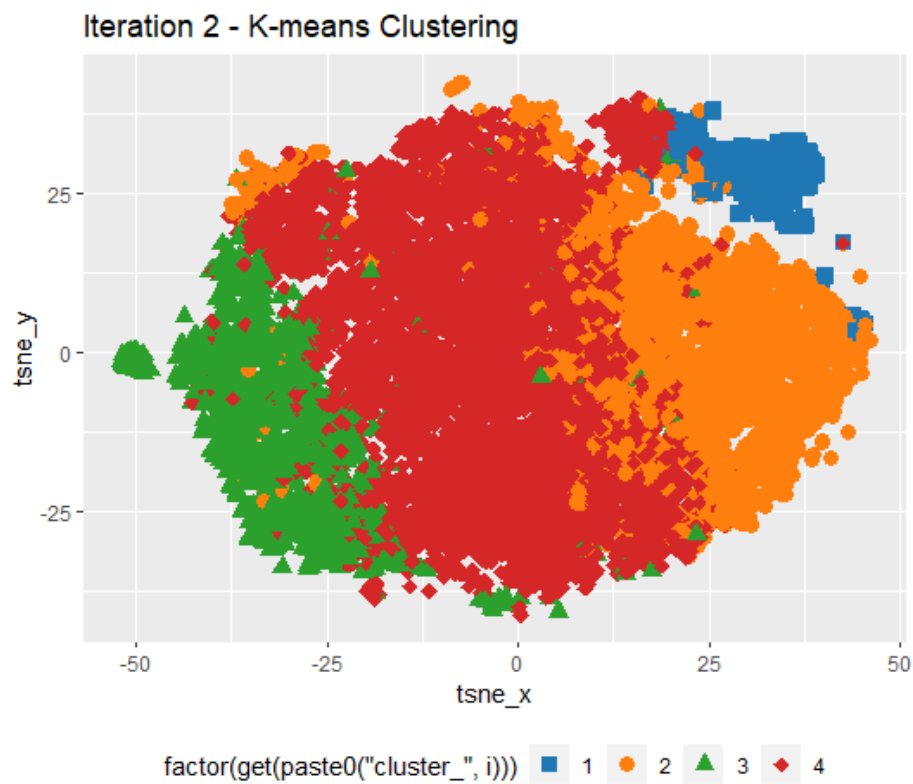


Figure 6. Iteration 2 on K-means Clustering.

The scatterplot in Figure 5. shows the results of k-means clustering for Iteration 1, with four clusters represented by different colors and shapes. The scatterplot shows that the four clusters are well-separated and distinct from each other, with relatively little overlap between them. This indicates that the k-means clustering algorithm was successful in grouping together songs that are similar in terms of their musical features.

The scatterplot in Figure 6. shows the results of k-means clustering for Iteration 2, with the same four clusters represented by the same colors and shapes as in Iteration 1. The scatterplot shows that the resulting clusters in Iteration 2 are almost identical to those in Iteration 1, with only a few points assigned to different clusters. This indicates that the algorithm converged to the same solution in both iterations, suggesting that the clusters are stable and consistent.

Through an analysis of the musical features of each cluster, we gained valuable insights into the overall characteristics of the songs in each group. Cluster 1 was characterized by high levels of energy, loudness, and valence, suggesting that these songs are more upbeat and energetic in nature. In contrast, Cluster 2 was characterized by low levels of energy and acousticness, indicating that these songs are more mellow and acoustic. Similarly, Cluster 3 was characterized by high levels of instrumentalness and low levels of speechiness, while Cluster 4 was characterized by high levels of danceability and speechiness.

Overall, these findings provide a deeper understanding of how different musical features contribute to the overall characteristics of each cluster.

## V. Association Rule Mining

Association rule mining was performed using the apriori algorithm with a support threshold of 0.0002 and a confidence threshold of 0.3. The output of the analysis was a set of four association rules that satisfied the specified support and confidence thresholds.

The scatterplot generated from the association rules mined from the dataset shows the relationship between support and confidence for each rule. The x-axis represents the support, which is the proportion of transactions that contain the items in the rule, while the y-axis represents the confidence, which is the proportion of transactions containing the antecedent that also contain the consequent.

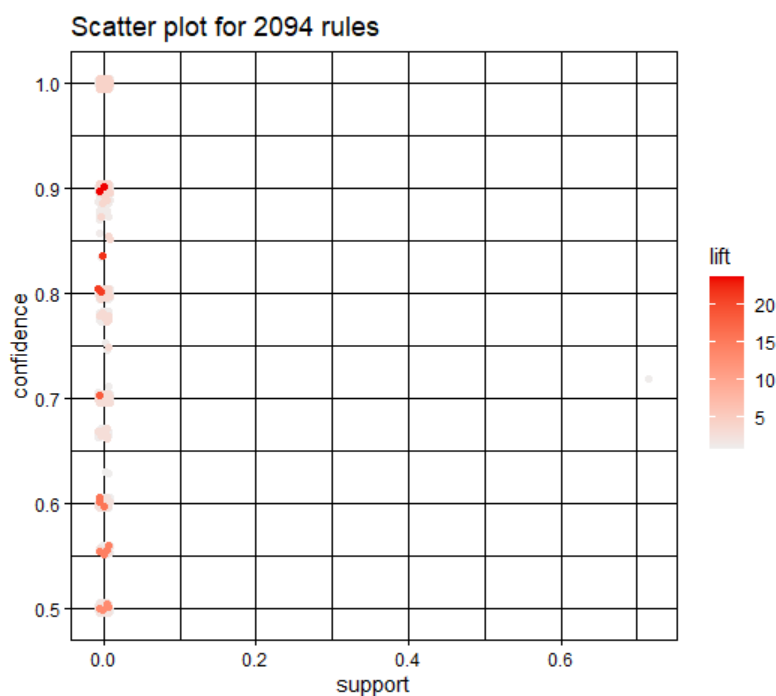


Figure 7. Scatter plot after exploring the mined rules.

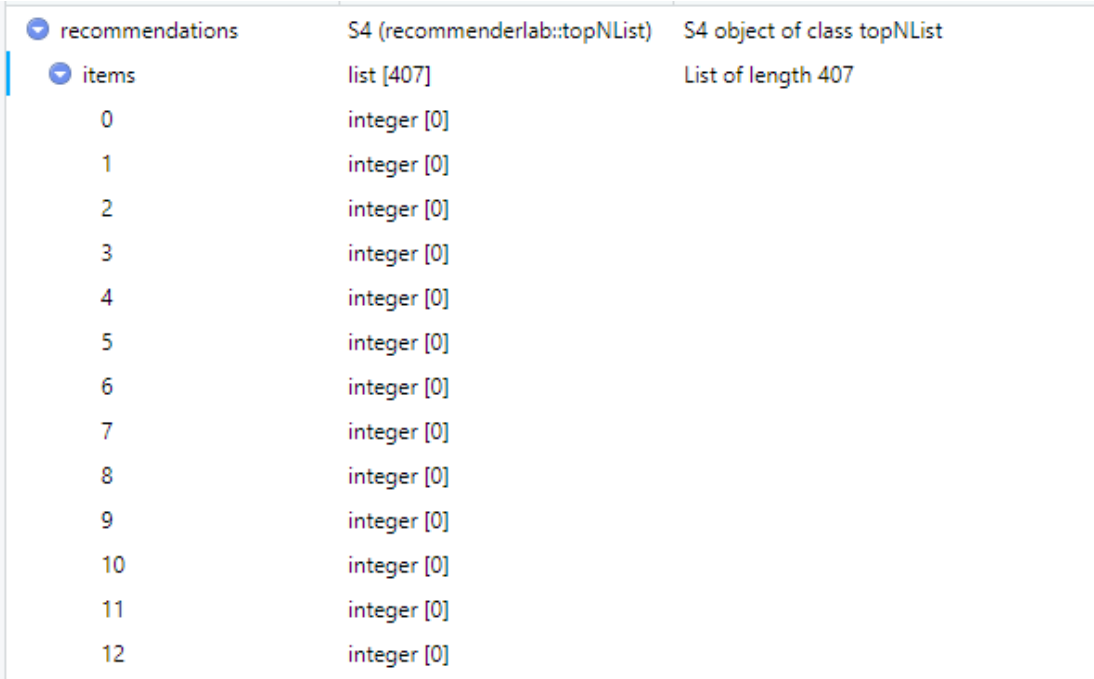
Since it is a big dataset, low support is acceptable.

The scatterplot Figure7. shows that there are many rules with low support and low confidence, which are located in the lower left corner of the plot. These rules may not be very interesting or useful. However, there are also some rules with high support and high confidence, which are located in the upper right corner of the plot. These rules may represent strong associations between the items in the dataset and could be useful for applications such as recommendation systems.

## VI. Recommendation System

In this section, we aimed to build a collaborative filtering recommendation model based on user ratings and generate recommendations for the test set. We converted the data to a sparse matrix format and split it into training and testing sets. We used collaborative filtering to build the recommendation model, and then generated recommendations for the test set using the prediction function with the parameter  $n=5$  to generate the top 5 recommendations for each user.

However, the recommendation system failed to generate any recommendations for the test set, and the printed output of `recommendations@items` was all zeros.



The screenshot shows the RStudio environment with the following structure:

Object	Class	Description
recommendations	S4 (recommenderlab::topNList)	S4 object of class topNList
items	list [407]	List of length 407
0	integer [0]	
1	integer [0]	
2	integer [0]	
3	integer [0]	
4	integer [0]	
5	integer [0]	
6	integer [0]	
7	integer [0]	
8	integer [0]	
9	integer [0]	
10	integer [0]	
11	integer [0]	
12	integer [0]	

This could be due to several reasons, such as sparsity of the data or lack of user-item interactions in the test set. It might be necessary to further explore the data and experiment with different parameters for the recommendation model to improve its performance.

## VII. Future step

Based on the findings of this report, several potential areas for future research and analysis could be explored. One potential area is to investigate the relationships between music genres and other attributes, such as lyrics or album release dates.

Additionally, further research could be conducted to explore how the findings of this report could be applied to music recommendation systems or marketing strategies.

Another potential area for future research is to compare the results of association rule mining with other data mining techniques to determine which approach is most effective for analyzing music data.

## VIII. conclusion

In this study, we investigated the relationship between music characteristics and popularity on Spotify and YouTube. We analyzed a dataset of music characteristics, streaming and video metrics, and other variables for a sample of songs using regression analysis, clustering, association rule mining, and recommendation system building. Our findings provided insights into music consumption and preferences in the digital age. However, our recommendation system failed to generate any recommendations for the test set, and further exploration and experimentation would be necessary to improve its performance.

## Appendix – Dataset and R Coding

### Spotify and YouTube dataset

<https://www.kaggle.com/datasets/salvatorerastelli/spotify-and-youtube>

#### R coding

```
#loading libraries
```

```
library(data.table)
```

```
library(Rtsne)
```

```
library(ggplot2)
```

```
library(gridExtra)
```

```
library(caret)
```

```
library(glmnet)
```

```
library(dplyr)
```

```
library(arules)
```

```
library(arulesViz)
```

```
library(recommenderlab)
```

```
library(biglm)
```

```
#read the data set
```

```
data <- fread("D:/Downloads/Study/data mining  
project/archive/Spotify_Youtube.csv")
```

```
#Data preprocessing
```

```
#check if there have any missing value
```

```
sum(is.na(data))
```

```
#drop the row with missing value
```

```
data <- data[complete.cases(data), ]
```

```
#drop the variable is not useful for this project
```

```
data <- subset(data, select = c(Artist, Track, Album, Album_type, Danceability, Energy,  
Key, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence,  
Tempo, Duration_ms, Stream, Title, Channel, Views, Likes, Comments, Licensed,  
official_video ))
```

```
#Convert categorical variables to numerical format
```

```

data$Licensed <- ifelse(data$Licensed == "No", 0, 1)
data$official_video <- as.integer(data$official_video)

#Normalize the numerical variables (for k-means clustering)
num_cols <- c("Danceability", "Energy", "Key", "Loudness", "Speechiness",
"Acousticness", "Instrumentalness", "Liveness", "Valence", "Tempo", "Duration_ms")
data[, (num_cols) := lapply(.SD, scale), .SDcols = num_cols]

# Fit t-SNE to the selected columns
tsne_obj <- Rtsne(as.matrix(data[, ..num_cols]), dims = 2, perplexity = 30, verbose =
TRUE, check_duplicates = FALSE)

# Convert the t-SNE result to a data table and add it to the original data table
tsne_data <- data.table(tsne_obj$Y)
setnames(tsne_data, c("tsne_x", "tsne_y"))
data <- cbind(data, tsne_data)

# Define colors for each album type
colors <- c("album" = "#1f77b4", "single" = "#ff7f0e", "compilation" = "#2ca02c")

# Create a scatterplot of the preprocessed data before t-SNE (using the first two
numerical columns)
p1 <- ggplot(data, aes(x = Danceability, y = Energy, color = Album_type)) +
  geom_point(size = 2) +
  scale_color_manual(values = colors) +
  ggtitle("Relationship Between Danceability and Energy, Colored by Album Type
(Before t-SNE)") +
  xlab("Danceability (scaled)") +
  ylab("Energy (scaled)")

# Create a scatterplot of the t-SNE result
p2 <- ggplot(data, aes(x = tsne_x, y = tsne_y, color = Album_type)) +
  geom_point(size = 2) +
  scale_color_manual(values = colors) +
  ggtitle("Relationship Between t-SNE Components and Album Type (After t-SNE)") +
  xlab("t-SNE Component 1") +
  ylab("t-SNE Component 2")

```

```

# Combine the two scatterplots into a single figure using the gridExtra package
grid.arrange(p1, p2, ncol = 2)

# Split the data into training and testing sets
set.seed(123)
trainIndex <- sample(nrow(data), floor(0.8 * nrow(data)))
training <- data[trainIndex, c("Danceability", "Energy", "Key", "Loudness",
"Speechiness", "Acousticness", "Instrumentalness", "Liveness", "Valence", "Tempo",
"Stream", "Views", "Likes", "Licensed", "official_video", "Artist", "Album",
"Album_type", "Duration_ms")]
testing <- data[-trainIndex, c("Danceability", "Energy", "Key", "Loudness",
"Speechiness", "Acousticness", "Instrumentalness", "Liveness", "Valence", "Tempo",
"Stream", "Views", "Likes", "Licensed", "official_video", "Artist", "Album",
"Album_type", "Duration_ms")]

# Convert integer64 data to numeric
training$Stream <- as.numeric(training$Stream)
training$Views <- as.numeric(training$Views)
training$Likes <- as.numeric(training$Likes)
testing$Stream <- as.numeric(testing$Stream)
testing$Views <- as.numeric(testing$Views)
testing$Likes <- as.numeric(testing$Likes)

# Fit linear regression models separately
model1_views <- lm(Views ~ Danceability + Energy + Key + Loudness + Speechiness +
Acousticness + Instrumentalness + Liveness + Valence + Tempo, data = training)
model1_stream <- lm(Stream ~ Danceability + Energy + Key + Loudness +
Speechiness + Acousticness + Instrumentalness + Liveness + Valence + Tempo, data =
training)

# Use models to make predictions on testing data
pred1_views <- predict(model1_views, newdata = testing)
pred1_stream <- predict(model1_stream, newdata = testing)

# Compute R-squared values for Model 1
r_squared1_views <- summary(model1_views)$r.squared
r_squared1_stream <- summary(model1_stream)$r.squared

```



```

# Print results for Model 1
cat("Model 1 (song features):\n")
cat("Coefficients for Views:\n")
print(coef(model1_views))
cat("R-squared value for Views:", r_squared1_views, "\n\n")
cat("Coefficients for Stream:\n")
print(coef(model1_stream))
cat("R-squared value for Stream:", r_squared1_stream, "\n\n")

new_song <- data.frame(Danceability = 0.8,
                      Energy = 0.1,
                      Key = 7,
                      Loudness = 5.2,
                      Speechiness = 0.01,
                      Acousticness = 0.05,
                      Instrumentalness = 0.1,
                      Liveness = 0.1,
                      Valence = 0.6,
                      Tempo = 60)

# Use Model 1 to make a prediction for the new song
new_song_pred <- predict(model1_views, newdata = new_song)

# Print the predicted number of views for the new song
cat("Predicted number of views for the new song:", new_song_pred, "\n")

# Create a scatter plot of predicted versus actual values for Views
plot(x = pred1_views, y = testing$Views, xlab = "Predicted Views", ylab = "Actual Views", main = "Predicted vs. Actual Views")

# Add a diagonal line to the plot to show perfect correlation
abline(a = 0, b = 1, col = "red")

# Clustering Analysis: K-means clustering to group similar songs together based on
their musical features

# Select the numerical columns for clustering

```

```
num_cols <- c("Danceability", "Energy", "Loudness", "Speechiness", "Acousticness",  
"Instrumentalness", "Liveness", "Valence")
```

```
# Create a scatterplot of the t-SNE result before clustering  
p0 <- ggplot(data, aes(x = tsne_x, y = tsne_y, color = Album_type)) +  
  geom_point(size = 2) +  
  scale_color_manual(values = colors) +  
  ggtitle("Before Clustering")
```

```
# Print the scatterplot  
print(p0)
```

```
# Perform k-means clustering with k = 4, using 2 iterations with different starting  
centroids  
set.seed(123)  
k <- 4  
nstart <- 2  
kmeans_objs <- list()  
for (i in 1:nstart) {  
  kmeans_objs[[i]] <- kmeans(data[, ..num_cols], centers = k, nstart = nstart)  
}
```

```
# Add the cluster assignments to the original data table for each iteration  
for (i in 1:nstart) {  
  data[, paste0("cluster_", i)] := kmeans_objs[[i]]$cluster  
}
```

```
# Define colors and symbols for each cluster  
colors <- c("#1f77b4", "#ff7f0e", "#2ca02c", "#d62728")  
symbols <- c(15, 16, 17, 18)
```

```
# Create a scatterplot of the t-SNE result, colored by cluster for each iteration  
p_list <- list()  
for (i in 1:nstart) {  
  p_list[[i]] <- ggplot(data, aes(x = tsne_x, y = tsne_y, color =  
factor(get(paste0("cluster_", i))), shape = factor(get(paste0("cluster_", i))))) +  
    geom_point(size = 3) +  
    scale_color_manual(values = colors) +
```

```

    scale_shape_manual(values = symbols) +
    ggtitle(paste0("Iteration ", i, " - K-means Clustering")) +
    theme(legend.position = "bottom")
}

# Print the scatterplots for each iteration
for (p in p_list) {
  print(p)
}

# Convert the relevant variables to factors
data$Artist <- as.factor(data$Artist)
data$Album_type <- as.factor(data$Album_type)
data$Genre <- as.factor(data$Genre)

# Create a transaction dataset
trans_data <- as(data[, c("Artist", "Album_type", "Genre"), with = FALSE],
"transactions")

# Explore the transaction dataset
summary(trans_data)

# Mine association rules
rules <- apriori(trans_data, parameter = list(support = 0.0002, confidence = 0.3))

# Explore the mined rules
summary(rules)
inspect(rules)

# Visualize the association rules using a scatter plot
plot(rules, method = "scatterplot")

# Convert the data to a sparse matrix format
rating_matrix <- as(data, "realRatingMatrix")

# Split the data into training and testing sets
set.seed(123)

```

```
sampled_ratings <- sample(x = c(TRUE, FALSE), size = nrow(rating_matrix), replace =  
TRUE, prob = c(0.8, 0.2))  
train_ratings <- rating_matrix[sampled_ratings]  
test_ratings <- rating_matrix[!sampled_ratings]
```

```
# Build the recommendation model using collaborative filtering  
model <- Recommender(train_ratings, method = "UBCF")
```

```
# Generate recommendations for the test set  
recommendations <- predict(model, test_ratings, n = 5)
```

```
# Print the top 5 recommended items for each user in the test set  
print(recommendations@items)
```

```
ggplot(training_df, aes(x = Danceability, y = Views)) +  
  geom_point() +  
  labs(title = "Relationship between Danceability and Views", x = "Danceability", y =  
"Views")
```

```
cluster_sizes <- data[, .N, by = cluster_1]  
ggplot(cluster_sizes, aes(x = factor(cluster_1), y = N)) +  
  geom_bar(stat = "identity", fill = colors) +  
  labs(title = "Cluster Sizes", x = "Cluster", y = "Size")
```