Analysis of Music Characteristics on Streaming Success

Samantha Juteram

STA 4102- Computational Methods for Applied Statistics

Professor Houston Sanders

28th April,2024

Abstract

In this project, the influence of musical attributes on the streaming success of songs on Spotify is investigated. Using a dataset of the top 10 songs of various Spotify artists, collected in February, 2023. An analysis was conducted on various musical characteristics to understand the role they play in driving streaming numbers. After data cleaning, exploratory data analysis, statistical data analysis, various models such as logistic regression, ridge regression, Support Vector Machines, and Random Forests were used to model streaming success. The analysis revealed that Loudness, Energy, and Instrumentalness are the top predictors of streaming success. The Random Forest model performed the best with 79% accuracy on the original dataset. While some multicollinearity was observed, it did not appear to have a significant impact on the Random Forest model performance. The results offer insights to music industry professionals which can be used to both understand and predict music's commercial use. The results can even potentially be used to provide data-driven recommendations to increase the chances of a song getting a high number of streams.

Introduction

“According to the Recording Industry Association of America, revenues from streaming music grew 26% to $5.9 billion in the first half of 2021, accounting for 84% of total music revenues for the period and a 4% climb from 2019 levels” (Wilson 1). With this success in this industry generating so much revenue, artists and music producers are eager to understand what factors influence streaming success of their music. If the connection between musical characteristics and streaming success can be better understood these music industry professionals can use this information to make decisions that will increase the chances of their music accumulating high streams. Through a thorough analysis, this project aims to investigate this area to answer the question: What are the key musical characteristics that influence the streaming success of songs on popular music platforms like Spotify?

Data Description

The dataset used in this project was obtained from Kaggle. It contained statistics for the top 10 songs of a variety of Spotify artists and also their corresponding music video on YouTube. It should be noted that none of the data related to YouTube was used in the project. The data was collected on February 7,2023 and included metrics of music characteristics and consumer interaction. In total, the dataset contained 26 variables with 20,718 observations. The following is a description of the variables present in the dataset which was provided by the author:

- "Track: name of the song, as visible on the Spotify platform.
- Artist: name of the artist.
- Url_spotify: the Url of the artist.
- Album: the album in wich the song is contained on Spotify.
- Album_type: indicates if the song is relesead on Spotify as a single or contained in an album.
- Uri: a spotify link used to find the song through the API.
- 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. A value of 0.0 is least danceable and 1.0 is most danceable.
- 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. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features

contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

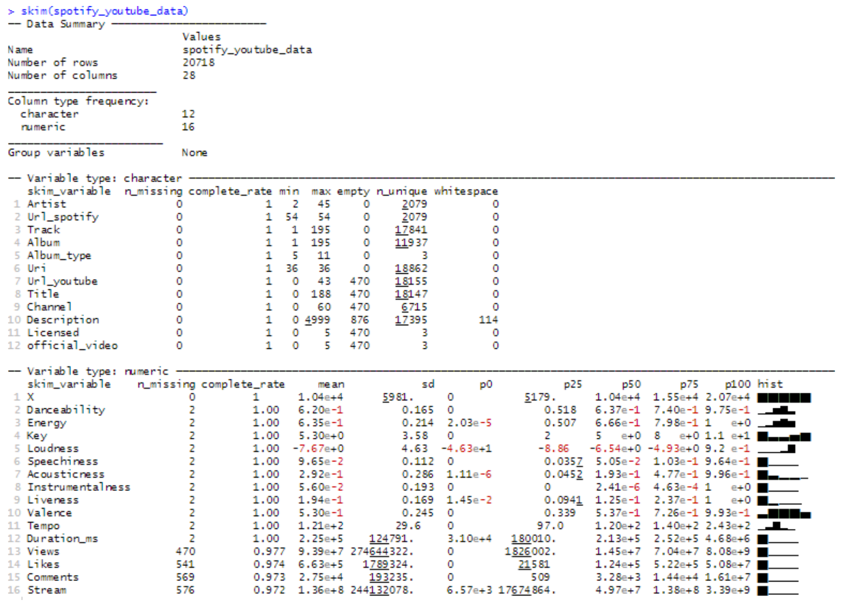
- Key: the key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
- Loudness: the overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.
- Speechiness: detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- 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. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

- Liveness: detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
- Valence: a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
- Tempo: the overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- Duration_ms: the duration of the track in milliseconds.
- Stream: number of streams of the song on Spotify.
- Url_youtube: url of the video linked to the song on Youtube, if it have any.
- Title: title of the videoclip on youtube.
- Channel: name of the channel that have published the video.
- Views: number of views.
- Likes: number of likes.
- Comments: number of comments.
- Description: description of the video on Youtube.

- Licensed: Indicates whether the video represents licensed content, which means that the content was uploaded to a channel linked to a YouTube content partner and then claimed by that partner.
- official_video: boolean value that indicates if the video found is the official video of the song." (Rastelli)

In the context of this project, there were limitations with this dataset. Firstly, the dataset only contained statistics for the top ten songs of each artist. This suggests a bias towards songs that are generally successful which may not represent all music on Spotify. By only looking at this data, the characteristics of less popular music are left out which could potentially affect the results of what factors affect streaming success. Also, the data is reflective of only a single point in time so it cannot account for changes in streams over time.

The first step of data exploration involved viewing a summary of the data using the skim function from the skimr package which yielded the following output:



```
> skim(spotify_youtube_data)
── Data Summary ────────────────────────
                           Values
Name                       spotify_youtube_data
Number of rows             20718
Number of columns          28
_______________________
Column type frequency:
  character                12
  numeric                  16
________________________
Group variables            None

── Variable type: character ────────────────────────────────────────────────────────────
   skim_variable  n_missing complete_rate min  max empty n_unique whitespace
 1 Artist                 0             1   2   45     0     2079          0
 2 Url_spotify            0             1  54   54     0     2079          0
 3 Track                  0             1   1  195     0    17841          0
 4 Album                  0             1   1  195     0    11937          0
 5 Album_type             0             1   5   11     0        3          0
 6 Uri                    0             1  36   36     0    18862          0
 7 Url_youtube            0             1   0   43   470    18155          0
 8 Title                  0             1   0  188   470    18147          0
 9 Channel                0             1   0   60   470     6715          0
10 Description            0             1   0 4999   876    17395        114
11 Licensed               0             1   0    5   470        3          0
12 official_video         0             1   0    5   470        3          0

── Variable type: numeric ──────────────────────────────────────────────────────────────
   skim_variable    n_missing complete_rate      mean          sd       p0         p25       p50       p75      p100 hist
 1 X                        0         1      1.04e+4       5981.  0             5179.   1.04e+4   1.55e+4   2.07e+4 ▇▇▇▇▇
 2 Danceability             2         1.00   6.20e-1      0.165  0             0.518   6.37e-1   7.40e-1   9.75e-1 ▁▂▆▇▂
 3 Energy                   2         1.00   6.35e-1      0.214  2.03e-5       0.507   6.66e-1   7.98e-1   1   e+0 ▁▂▆▇▆
 4 Key                      2         1.00   5.30e+0      3.58   0             2       5   e+0   8   e+0   1.1 e+1 ▇▂▂▃▇
 5 Loudness                 2         1.00  -7.67e+0      4.63  -4.63e+1      -8.86   -6.54e+0  -4.93e+0   9.2 e-1 ▁▁▁▂▇
 6 Speechiness              2         1.00   9.65e-2      0.112  0             0.0357  5.05e-2   1.03e-1   9.64e-1 ▇▁▁▁▁
 7 Acousticness             2         1.00   2.92e-1      0.286  1.11e-6       0.0452  1.93e-1   4.77e-1   9.96e-1 ▇▃▂▂▂
 8 Instrumentalness         2         1.00   5.60e-2      0.193  0             0       2.41e-6   4.63e-4   1   e+0 ▇▁▁▁▁
 9 Liveness                 2         1.00   1.94e-1      0.169  1.45e-2       0.0941  1.25e-1   2.37e-1   1   e+0 ▇▂▁▁▁
10 Valence                  2         1.00   5.30e-1      0.245  0             0.339   5.37e-1   7.26e-1   9.93e-1 ▃▇▇▇▆
11 Tempo                    2         1.00   1.21e+2     29.6    0            97.0     1.20e+2   1.40e+2   2.43e+2 ▁▃▇▂▁
12 Duration_ms              2         1.00   2.25e+5   124791.   3.10e+4    180010.    2.13e+5   2.52e+5   4.68e+6 ▇▁▁▁▁
13 Views                  470         0.977  9.39e+7 274644322.  0         1826002.    1.45e+7   7.04e+7   8.08e+9 ▇▁▁▁▁
14 Likes                  541         0.974  6.63e+5   1789324.  0           21581     1.24e+5   5.22e+5   5.08e+7 ▇▁▁▁▁
15 Comments               569         0.973  2.75e+4    193235.  0             509     3.28e+3   1.44e+4   1.61e+7 ▇▁▁▁▁
16 Stream                 576         0.972  1.36e+8 244132078.  6.57e+3 17674864.    4.97e+7   1.38e+8   3.39e+9 ▇▁▁▁▁
```

Figure 1 Showing the Output of Using the skim function to view a summary of the Data Set

All rows with missing values, specifically for the variables that were to be explored, were removed. “Duplicates” were then removed meaning entries had the same track and album name due to the fact that some songs have more than one artist, and in this data, that song would appear for each artist. Variables that were not going to be used were also removed from the dataset. Outliers in each variable were then identified. However, it was decided to not to do anything to the outliers as the nature of music data is that it has high variability and altering/removing outliers would mean changing/leaving out important data. An additional category called “Stream_cat” was made which categorized songs into "High" streaming if the number of streams was above 10,000,000 or "Low" otherwise. The threshold was chosen based on a Forbes article that stated that for a song to be considered a hit, “you need at least 10 million for that” (Owsinski). This variable was made to facilitate logistic regression models later on. To get a deeper understanding of the data, histograms of each characteristic (Danceability, Energy, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, Duration_ms, Stream) were made to view their distributions and any skew or obscurities were noted using ggplot2. A barplot was made for Key as this is a discrete variable. They were patched together using the patchwork package.

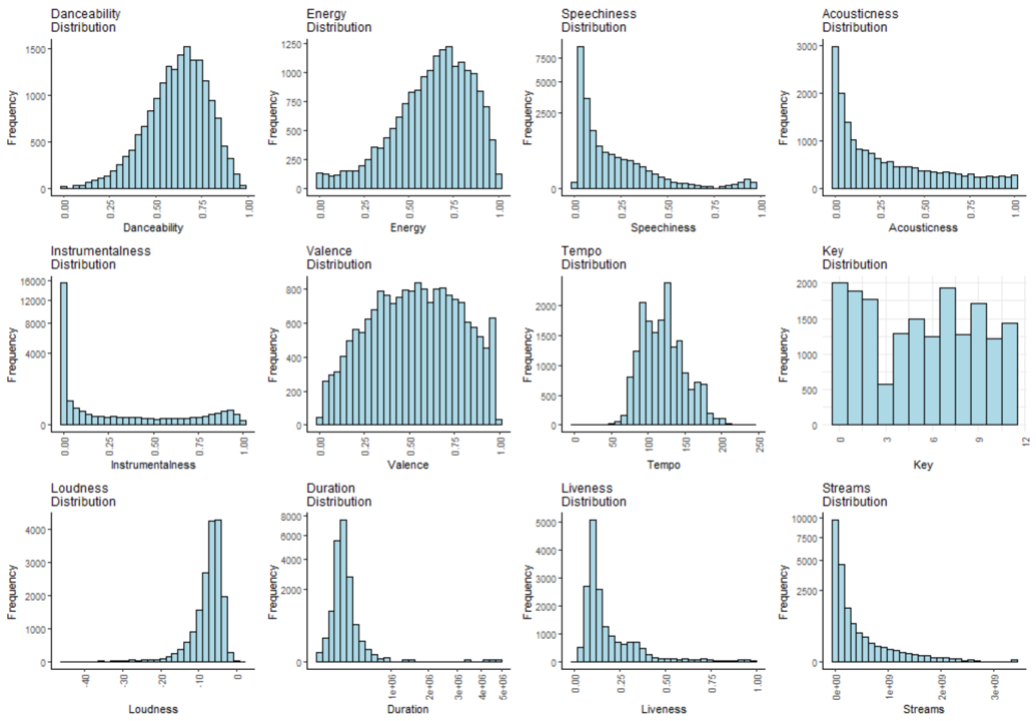
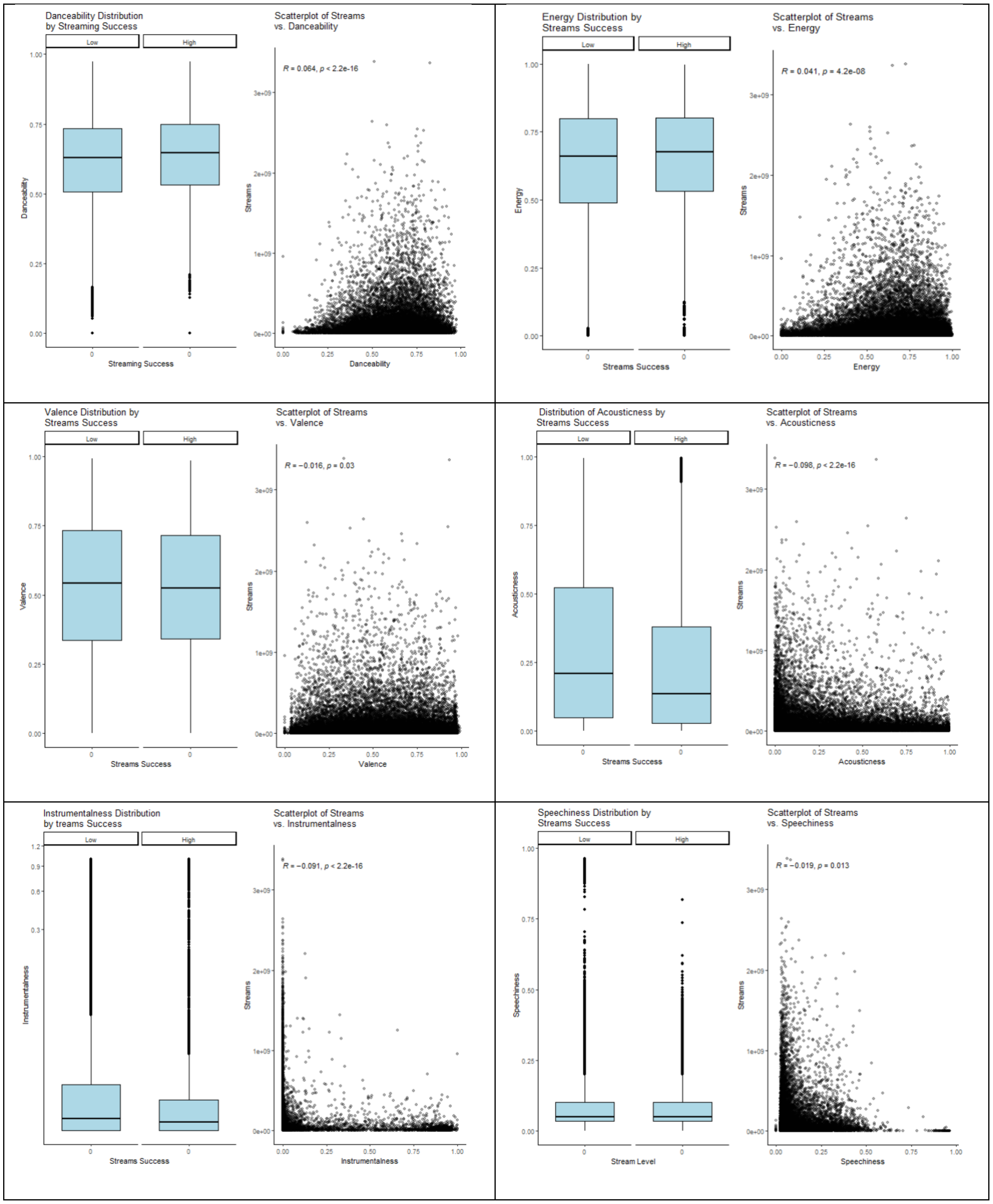
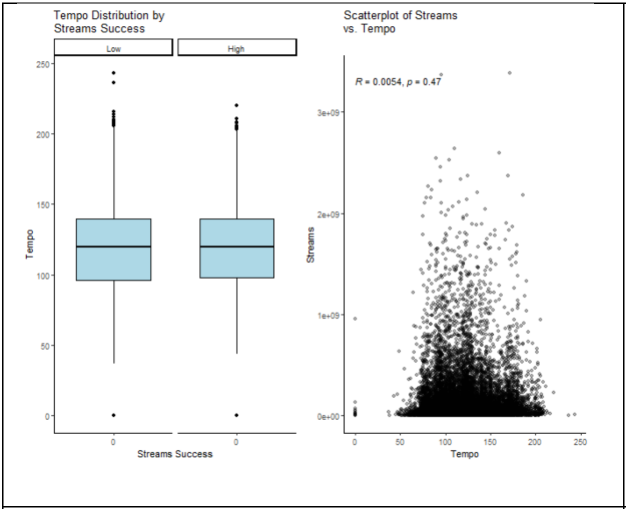
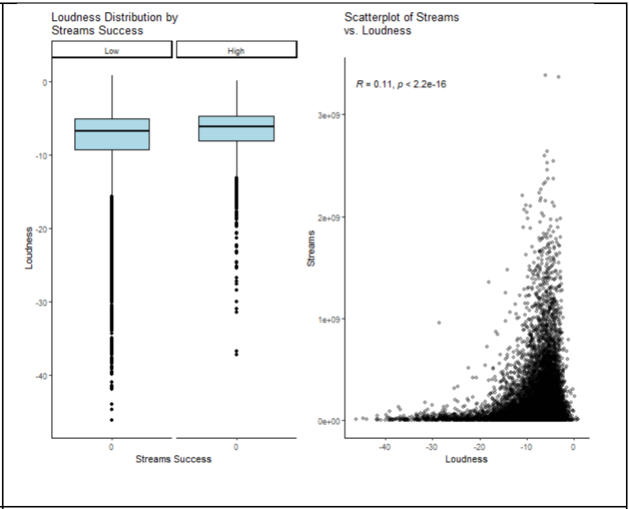


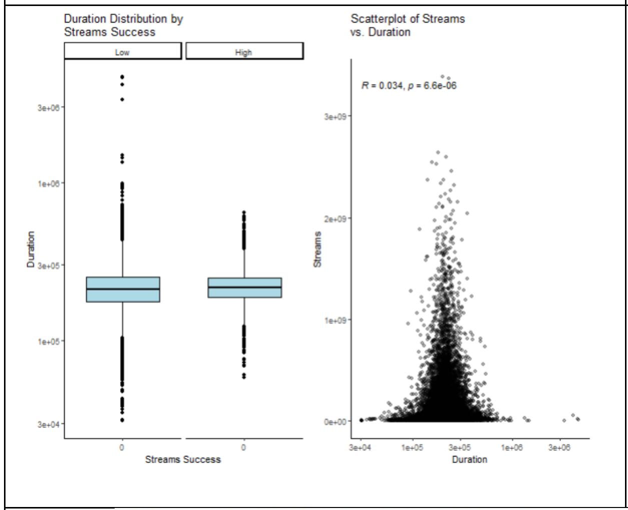
Figure 2 Showing the Distribution of Each Streaming Characteristic

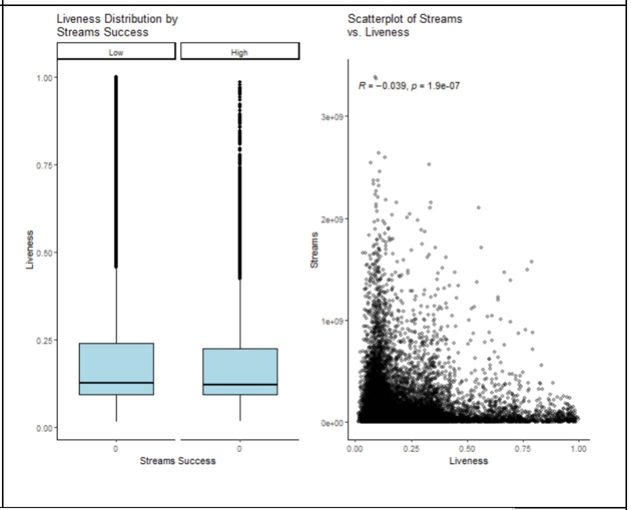
A scatter plot of each musical characteristic vs the streaming category was plotted to observe potential linear relationships. stat_cor(method= “pearson”) to display the correlation coefficient of for each scatter plot. Additionally, boxplots of each music characteristic by streaming category were plotted. Since Keys is discrete, piecharts for Keys by Stream_Cat was made for this variable.


Danceability Distribution by Streaming Success
Low
High
Danceability
Streaming Success
Scatterplot of Streams vs. Danceability
R = 0.064, p < 2.2e-16
Streams
Danceability
Energy Distribution by Streams Success
Energy
Streams Success
Scatterplot of Streams vs. Energy
R = 0.041, p = 4.2e-08
Energy
Valence Distribution by Streams Success
Valence
Streams Success
Scatterplot of Streams vs. Valence
R = −0.016, p = 0.03
Valence
Distribution of Acousticness by Streams Success
Acousticness
Streams Success
Scatterplot of Streams vs. Acousticness
R = −0.098, p < 2.2e-16
Acousticness
Instrumentalness Distribution by treams Success
Instrumentalness
Streams Success
Scatterplot of Streams vs. Instrumentalness
R = −0.091, p < 2.2e-16
Instrumentalness
Speechiness Distribution by Streams Success
Speechiness
Stream Level
Scatterplot of Streams vs. Speechiness
R = −0.019, p = 0.013
Speechiness









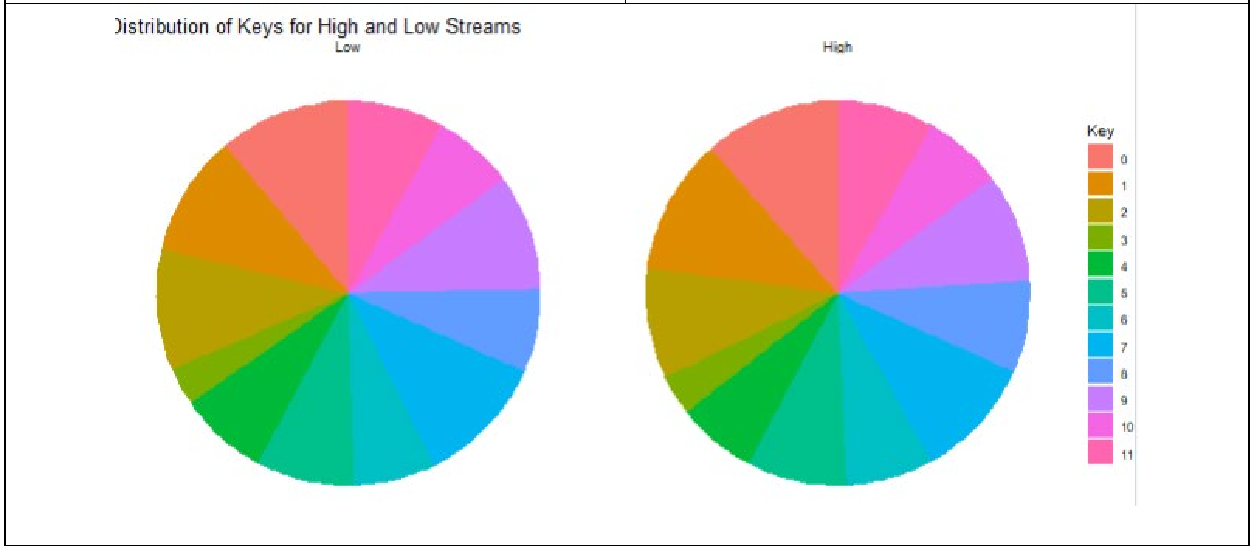


Table 1 Showing Scatter Plots for Each Music Characteristic vs Streams ,Boxplots of Each Music Characteristics by Stream_Cat and Pie Chart of Keys by Stream_Cat

There were no clear linear relationships, even after transforming the data. Differences amongst the plots were observed indicating that logistic regression would be a better avenue to explore.

Methodology

The results of the exploratory data analysis pointed towards investigating logistic regression. To begin, logistic regression models were created for each music characteristic as the predictor variable and Stream_cat as the y. Summaries of each of these models were viewed and their AIC values were compared. Instrumentalness, loudness and energy had the lowest AIC values. Next to see if any combination of these music characteristics could potentially create a model with even lower AIC, logistic regression models of all the possible combinations of these three were made. Again, there summaries were viewed and their AIC values were compared. Stream_cat ~ Loudness + Danceability * Instrumentalness had the lowest AIC and high significance for each variable. The vif function from the car package was used to check the multicollinearity of variables in this model.

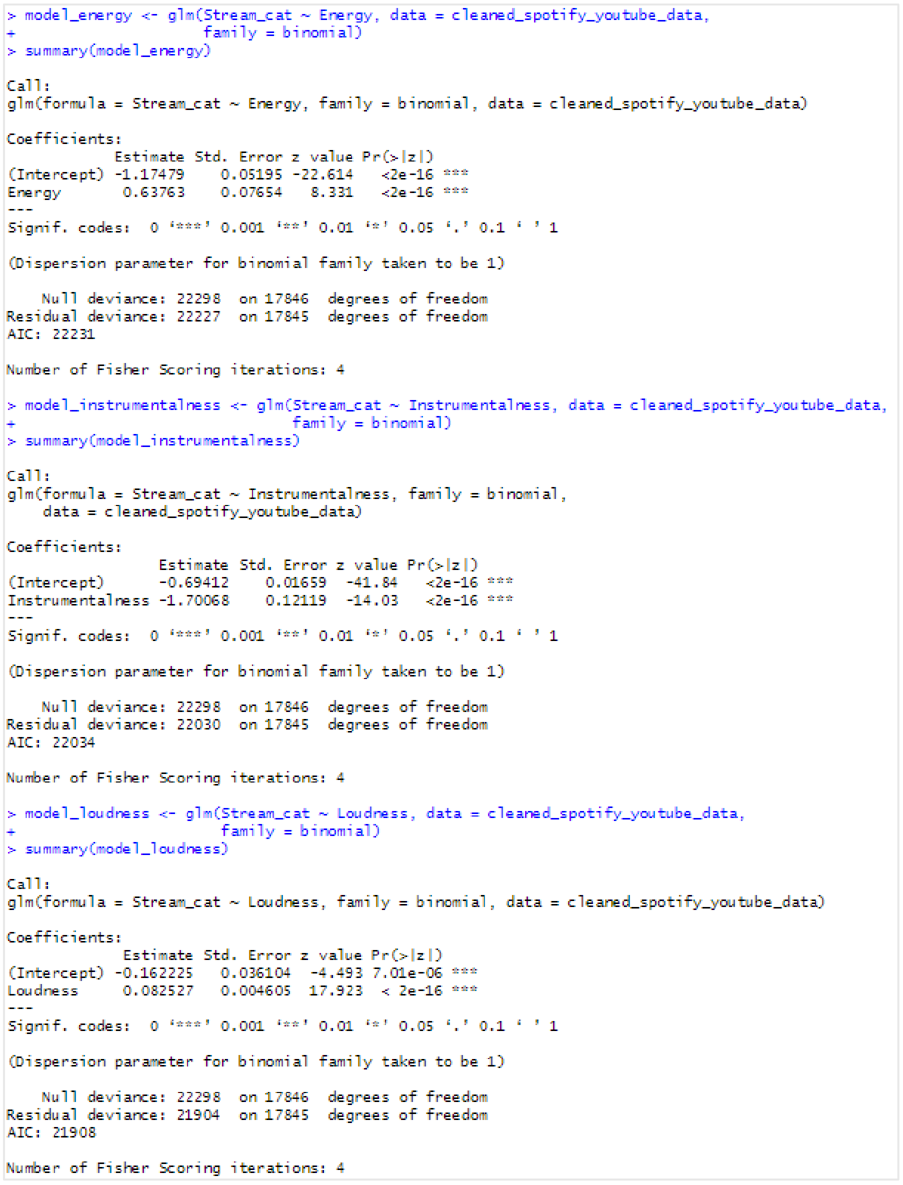
To address multicollinearity found, a ridge regression model was trained and tested as this model deals with the standard error caused by multicollinearity “by adding some bias in the estimates of the regression” (Team). The balanced dataset balanced_data was divided into a training set and a test set with a 75/25 split using the sample.split function from the caTools package. A model matrix was made for the training set but the intercept column was taken out. Stream_cat ~ Loudness + Danceability * Instrumentalness was used to specify the predictors for the model. The outcome variable Stream_cat was converted to a factor using as.factor. Cross-validation was used to determine the optimal value of lambda for ridge regression by using the cv.glmnet function from the glmnet package with alpha = 0 for ridge regression and family = “binomial” . The final ridge regression model was fitted to the training data using the glmnet function with the optimal lamda. Predictions were made on the test set using the model. The accuracy of the model was calculated by comparing the predicted classes to the actual classes in the test set. Then the confusionMatrix function from the caret package was used to create a confusion matrix. Due to predictions being biased towards low, a balanced data set was created. This was done by using To balance the ovun.sample function from the ROSE package with the formula Stream_cat ~ ., and method = "both" in order to over-sampling and under-sampling to address the class imbalance. A new train and test data sets were made from the new balanced datasets using the same methodology prior. The model was again tested and trained using the same methodology. However, there was no bias in the opposite direction. Due to the ridge regression model's poor performance and only moderate multicollinearity to start with, this model was abandoned.

Support vector machines were then explored as they do well in classification problems and more importantly are effective in high dimensional spaces. ("1.4. Support Vector Machines"). Using the original cleaned datasets the data was then split exactly as before. Using the svm function from the e1071 library, a model was trained on the training set with the same formula used prior. The SVM was configured for C-classification. Initially the kernel was set to linear. Predictions were made on the test data using the trained model, and a confusion matrix was made to evaluate the performance. The model was trained and tested for each of the kernels however the model performed the best with the radial kernel. The hyperparameters cost and gamma or adjusted to try to improve the model's accuracy. While the predictions were balanced, the model was not very accurate. Training and testing the support vector machine with the balance data set was also tried but there was little difference in the results. Due to the mediocre performance of the support vaccine machine, it was decided to explore another model.

Random Forest is another model that performs well for classification, but also they are generally not sensitive to outliers and not affected by multicollinearity due to the fact that it uses bootstrap sampling .(Raj) Using the original cleaned datasets the data split exactly as before, and a randomForest model was trained using the randomForest function from the randomForest package. The formula was the same as aforementioned. Predictions were then made on the test set, and the model's accuracy was calculated. The predictions were biased towards low so the model was trained on tested again with train and test data from the balanced datasets. The accuracy greatly increased. Hyperparameters min.samples.split, min.samples.leaf, and max.depth were fine-tuned. It was a concern that the model might not perform as well on the original data set with the unbalanced Stream_cat so the model was tested on the original data set and the accuracy remained high. To further evaluate the random forest model the F1 score which considers both precision on the recall was calculated. An ROC curve was then plotted and the AUC was calculated in order to find out more about the models ability to distinguish between high and low. Lastly the importance of each variable in the random forest model was found using the importance function and then visualized to see which variables were most influential in the model.

Results

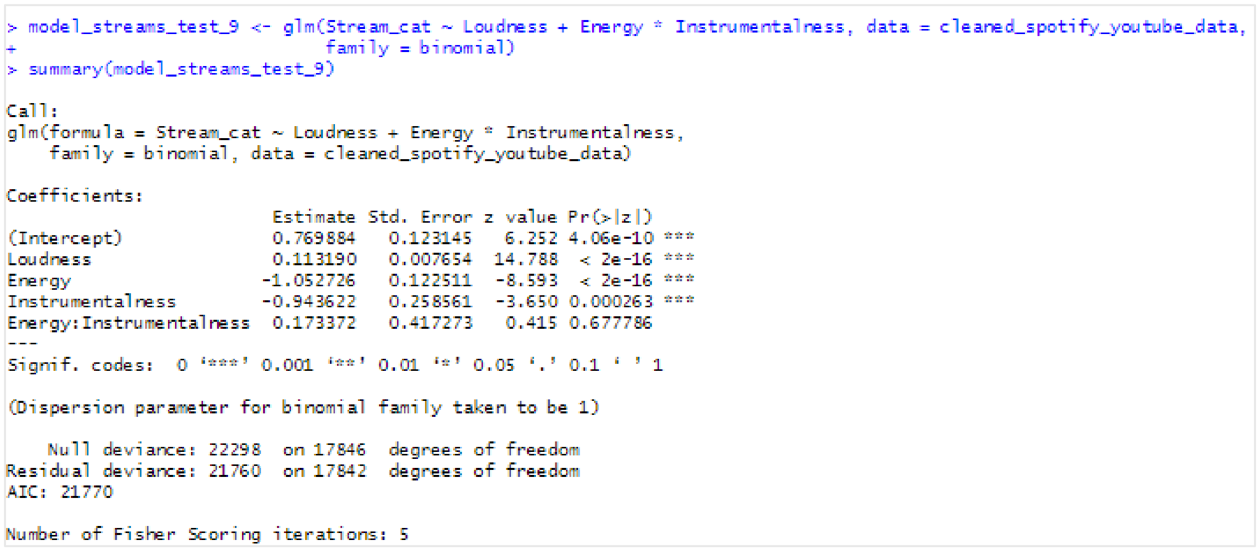
As aforementioned, the first part of the statistical analysis consisted of making logistic regression models for each music characteristic as a predictor variable and Stream_Cat as y. Instrumentalness, Energy and Loudness had the lowest AIC values with high significance. The following is the output of the summary of their related models:



```
> model_energy <- glm(Stream_cat ~ Energy, data = cleaned_spotify_youtube_data,
+                     family = binomial)
> summary(model_energy)

Call:
glm(formula = Stream_cat ~ Energy, family = binomial, data = cleaned_spotify_youtube_data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -1.17479    0.05195 -22.614   <2e-16 ***
Energy       0.63763    0.07654   8.331   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22298  on 17846  degrees of freedom
Residual deviance: 22227  on 17845  degrees of freedom
AIC: 22231

Number of Fisher Scoring iterations: 4

> model_instrumentalness <- glm(Stream_cat ~ Instrumentalness, data = cleaned_spotify_youtube_data,
+                               family = binomial)
> summary(model_instrumentalness)

Call:
glm(formula = Stream_cat ~ Instrumentalness, family = binomial,
    data = cleaned_spotify_youtube_data)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -0.69412    0.01659  -41.84   <2e-16 ***
Instrumentalness -1.70068    0.12119  -14.03   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22298  on 17846  degrees of freedom
Residual deviance: 22030  on 17845  degrees of freedom
AIC: 22034

Number of Fisher Scoring iterations: 4

> model_loudness <- glm(Stream_cat ~ Loudness, data = cleaned_spotify_youtube_data,
+                       family = binomial)
> summary(model_loudness)

Call:
glm(formula = Stream_cat ~ Loudness, family = binomial, data = cleaned_spotify_youtube_data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.162225   0.036104  -4.493 7.01e-06 ***
Loudness     0.082527   0.004605  17.923  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22298  on 17846  degrees of freedom
Residual deviance: 21904  on 17845  degrees of freedom
AIC: 21908

Number of Fisher Scoring iterations: 4
```

Figure 3 Showing the Output of the Summary of the Models with Lowest AIC values

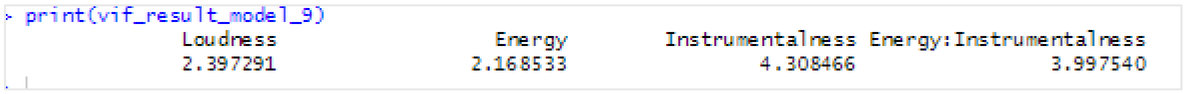
After making models with looking at models for all possible combinations of these charactersistics as the predictors, the Stream_cat ~ Loudness + Danceability * Instrumentalness had the lowest AIC with the following summary:



```
> model_streams_test_9 <- glm(Stream_cat ~ Loudness + Energy * Instrumentalness, data = cleaned_spotify_youtube_data,
+                              family = binomial)
> summary(model_streams_test_9)

Call:
glm(formula = Stream_cat ~ Loudness + Energy * Instrumentalness,
    family = binomial, data = cleaned_spotify_youtube_data)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)              0.769884   0.123145   6.252 4.06e-10 ***
Loudness                 0.113190   0.007654  14.788  < 2e-16 ***
Energy                  -1.052726   0.122511  -8.593  < 2e-16 ***
Instrumentalness        -0.943622   0.258561  -3.650 0.000263 ***
Energy:Instrumentalness  0.173372   0.417273   0.415 0.677786
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22298  on 17846  degrees of freedom
Residual deviance: 21760  on 17842  degrees of freedom
AIC: 21770

Number of Fisher Scoring iterations: 5
```

Figure 4 Showing the Output of the Summary of the Model with Lowest AIC

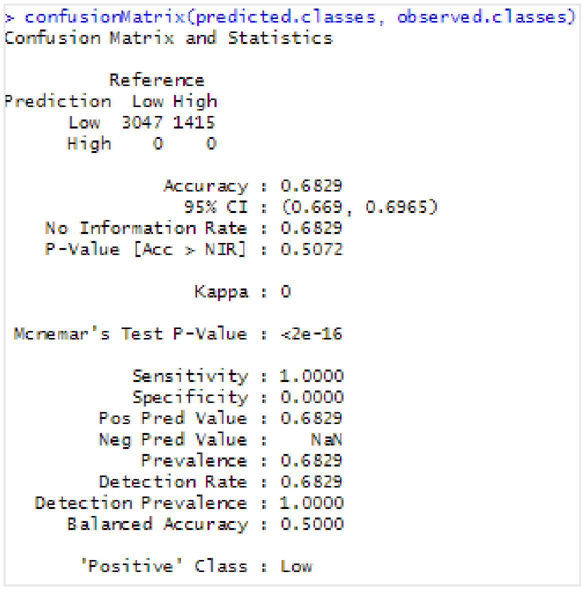
The results of using the vif() function to test the multicollinearity of the variables was as follows:



```
> print(vif_result_model_9)
               Loudness                  Energy        Instrumentalness Energy:Instrumentalness
               2.397291                2.168533                4.308466                3.997540
```

Figure 5 Showing VIF values of Model

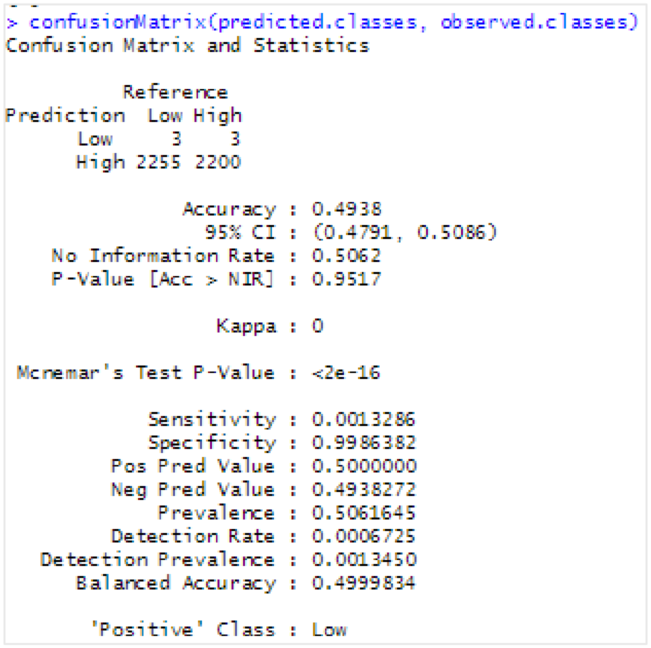
All of the terms had vif values between 1 and 5 which indicates moderate multicollinearity. Next a ridge regression model was trained and tested using the cleaned dataset and holdout validation which yielded the following Accuracy and confusion matrix:



```
> confusionMatrix(predicted.classes, observed.classes)
Confusion Matrix and Statistics

          Reference
Prediction  Low High
      Low  3047 1415
      High    0    0

               Accuracy : 0.6829
                 95% CI : (0.669, 0.6965)
    No Information Rate : 0.6829
    P-Value [Acc > NIR] : 0.5072

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.6829
         Neg Pred Value :    NaN
             Prevalence : 0.6829
         Detection Rate : 0.6829
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Low
```

Figure 6 Showing Confusion Matrix of Ridge Regression Model Trained Using Original Dataset

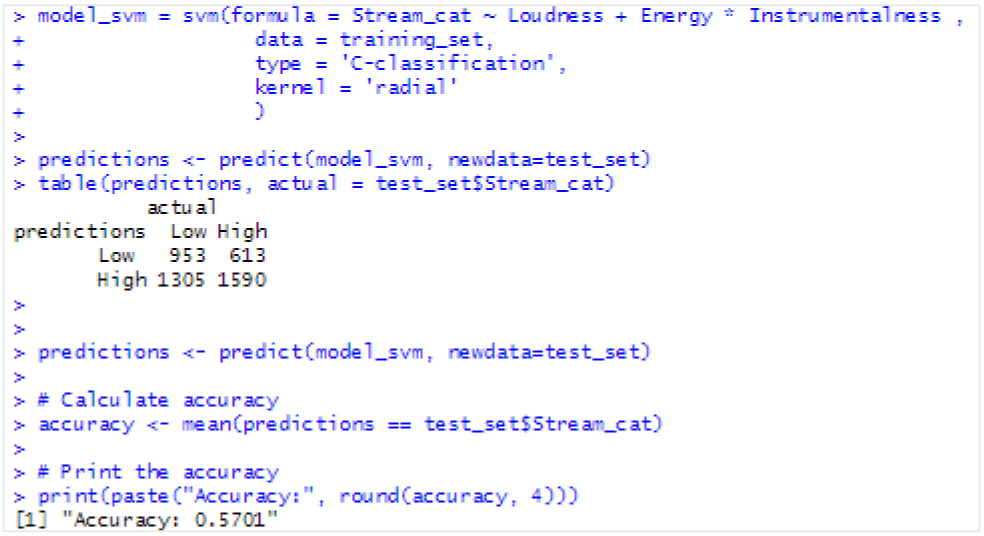
Although the accuracy was 68.3%, the confusion matrix revealed that the predictions were biased towards low. The ridge regression model was trained and tested using train and test sets from a balanced dataset. The following was the accuracy and confusion matrix:



```
> confusionMatrix(predicted.classes, observed.classes)
Confusion Matrix and Statistics

          Reference
Prediction  Low High
      Low     3    3
      High 2255 2200

               Accuracy : 0.4938
                 95% CI : (0.4791, 0.5086)
    No Information Rate : 0.5062
    P-Value [Acc > NIR] : 0.9517

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0013286
            Specificity : 0.9986382
         Pos Pred Value : 0.5000000
         Neg Pred Value : 0.4938272
             Prevalence : 0.5061645
         Detection Rate : 0.0006725
   Detection Prevalence : 0.0013450
      Balanced Accuracy : 0.4999834

       'Positive' Class : Low
```

Figure 7 Showing Confusion Matrix of Ridge Regression Model Trained Using Balanced Dataset

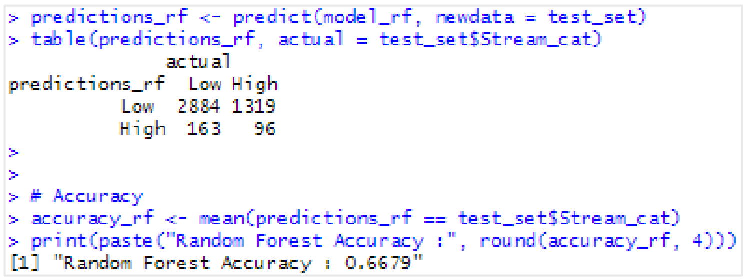
The confusion matrix revealed that now the predictions were balanced towards high with a 49.4% accuracy. Next the Support Vector Machine was trained and tested using the original dataset and holdout validation. The following are the accuracy and confusion matrix of the model after trying different kernels and hyperparameters:



```
> model_svm = svm(formula = Stream_cat ~ Loudness + Energy * Instrumentalness ,
+                  data = training_set,
+                  type = 'C-classification',
+                  kernel = 'radial'
+                  )
>
> predictions <- predict(model_svm, newdata=test_set)
> table(predictions, actual = test_set$Stream_cat)
           actual
predictions  Low High
       Low   953  613
       High 1305 1590
>
>
> predictions <- predict(model_svm, newdata=test_set)
>
> # Calculate accuracy
> accuracy <- mean(predictions == test_set$Stream_cat)
>
> # Print the accuracy
> print(paste("Accuracy:", round(accuracy, 4)))
[1] "Accuracy: 0.5701"
```

Figure 8 Showing Confusion Matrix and Accuracy of Support Vector Machine

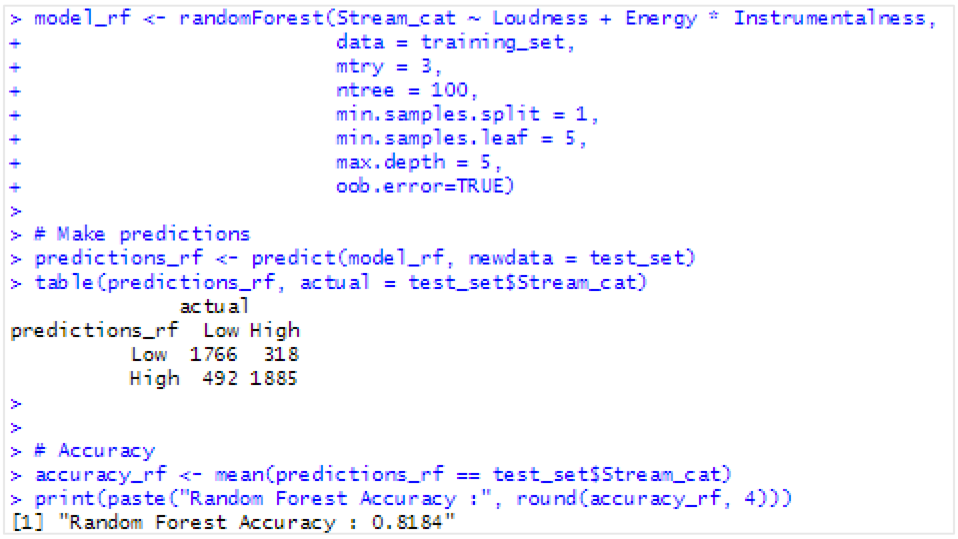
The model yielded the highest accuracy of 60.1% when the kernel was set to radial ,cost=1000 and gamma = 1 (the figure shown displays the model without hyperparameters as R would abort when trying to replicate). From the confusion matrix it can be seen that the predictions were balanced, however they were not very accurate. The random forest model was then explored. The Random Forest model was trained and tested using the original dataset and holdout validation. The following are the accuracy and confusion matrix of the model:



```
> predictions_rf <- predict(model_rf, newdata = test_set)
> table(predictions_rf, actual = test_set$Stream_cat)
              actual
predictions_rf  Low High
          Low  2884 1319
          High  163   96
>
>
> # Accuracy
> accuracy_rf <- mean(predictions_rf == test_set$Stream_cat)
> print(paste("Random Forest Accuracy :", round(accuracy_rf, 4)))
[1] "Random Forest Accuracy : 0.6679"
```

Figure 9 Showing Confusion Matrix and Accuracy of Random Forest Model Trained Using Original Dataset

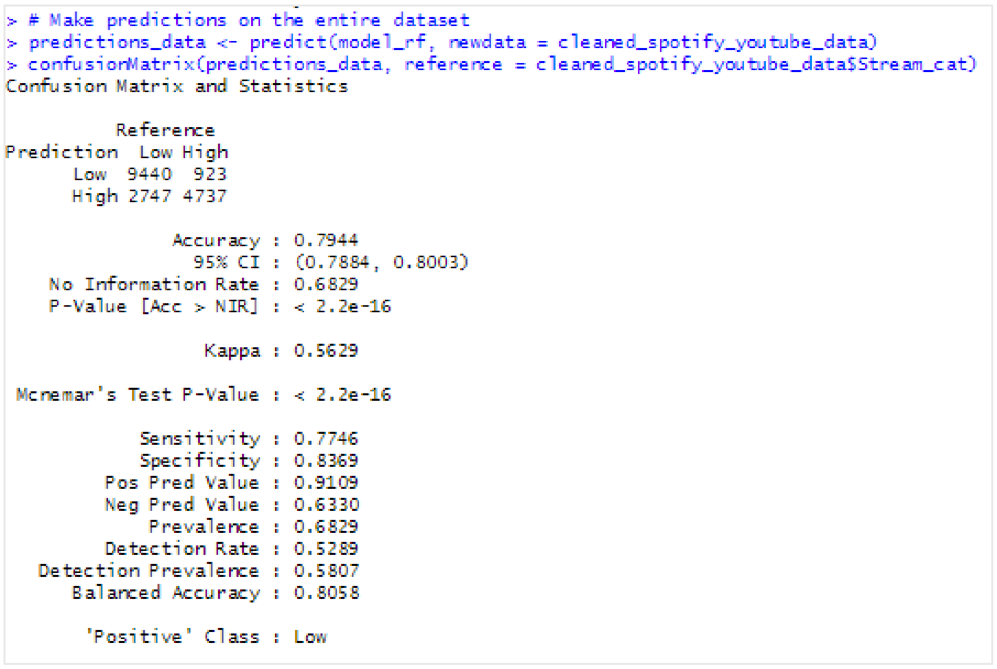
While the accuracy was 66.8%, the confusion matrix revealed that the predictions were biased towards low. The support vector machine model was then trained and tested using train and test sets from a balanced dataset. Hyperparameters were tuned as well. The following was the accuracy and confusion matrix:



```
> model_rf <- randomForest(Stream_cat ~ Loudness + Energy * Instrumentalness,
+                          data = training_set,
+                          mtry = 3,
+                          ntree = 100,
+                          min.samples.split = 1,
+                          min.samples.leaf = 5,
+                          max.depth = 5,
+                          oob.error=TRUE)
>
> # Make predictions
> predictions_rf <- predict(model_rf, newdata = test_set)
> table(predictions_rf, actual = test_set$Stream_cat)
              actual
predictions_rf  Low High
          Low  1766  318
          High  492 1885
>
>
> # Accuracy
> accuracy_rf <- mean(predictions_rf == test_set$Stream_cat)
> print(paste("Random Forest Accuracy :", round(accuracy_rf, 4)))
[1] "Random Forest Accuracy : 0.8184"
```

Figure 10 Showing Confusion Matrix and Accuracy of Random Forest Model Trained Using Balanced Dataset

The model performed best when mtry=3, ntree=100, min.sample.split=1, min.sample.leaf = 5 and max.depth=5. The accuracy was 81.8% with balanced predictions as seen from the confusion matrix. The random forest model was also tested on the original cleaned dataset due to concerns about its performance on unbalanced data.



```
> # Make predictions on the entire dataset
> predictions_data <- predict(model_rf, newdata = cleaned_spotify_youtube_data)
> confusionMatrix(predictions_data, reference = cleaned_spotify_youtube_data$Stream_cat)
Confusion Matrix and Statistics

          Reference
Prediction  Low High
      Low  9440  923
      High 2747 4737

               Accuracy : 0.7944
                 95% CI : (0.7884, 0.8003)
    No Information Rate : 0.6829
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5629

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.7746
            Specificity : 0.8369
         Pos Pred Value : 0.9109
         Neg Pred Value : 0.6330
             Prevalence : 0.6829
         Detection Rate : 0.5289
   Detection Prevalence : 0.5807
      Balanced Accuracy : 0.8058

       'Positive' Class : Low
```

Figure 11 Showing Confusion Matrix and Accuracy of Random Forest Model Making Predictions on Original Dataset

The model made accurate predictions on the entire data set with a 79.4% accurate. To further evaluate the random forest model, the F1 score, AUC score and importance of the model were found and an ROC curve was plotted. The following are the results:

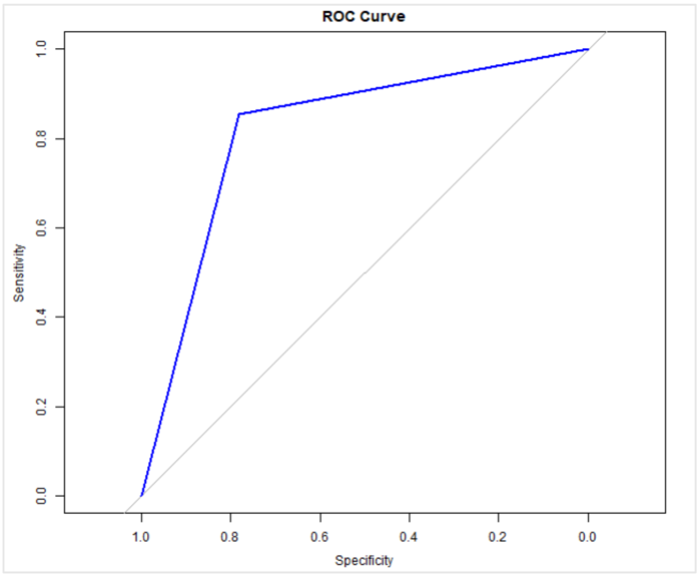
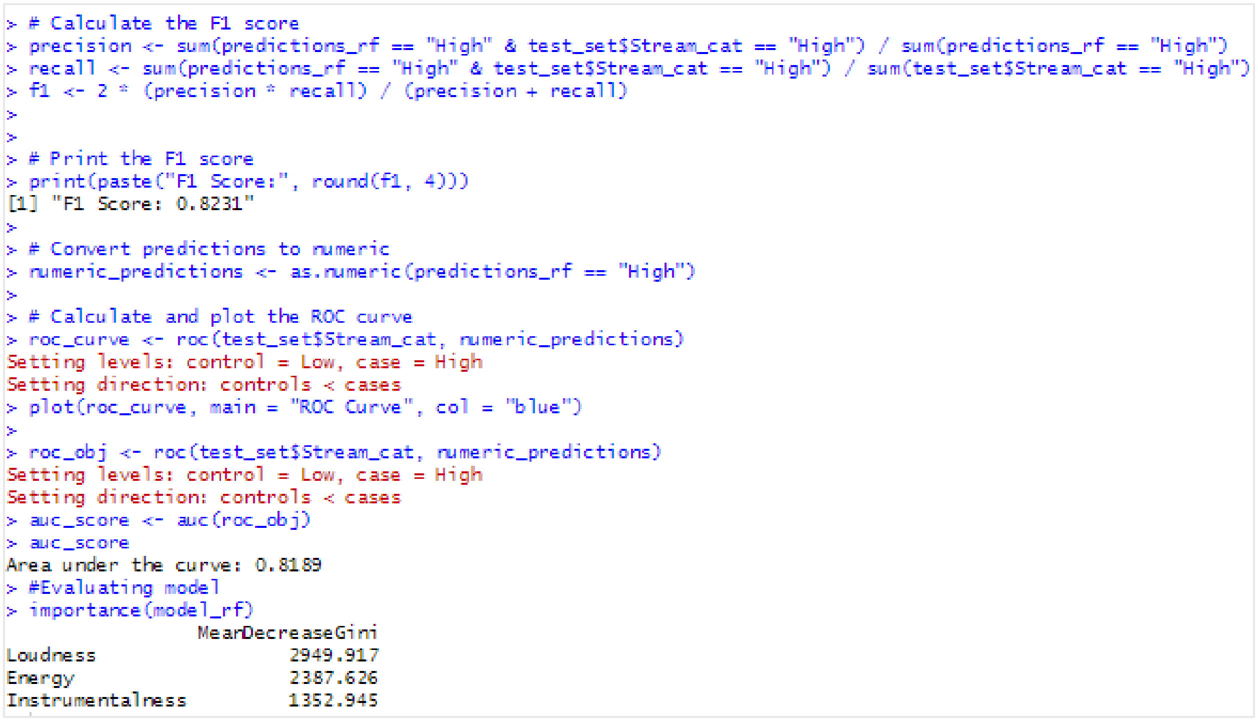


Figure 12 Showing ROC Curve for Random Forest Model



```
> # Calculate the F1 score
> precision <- sum(predictions_rf == "High" & test_set$Stream_cat == "High") / sum(predictions_rf == "High")
> recall <- sum(predictions_rf == "High" & test_set$Stream_cat == "High") / sum(test_set$Stream_cat == "High")
> f1 <- 2 * (precision * recall) / (precision + recall)
>
>
> # Print the F1 score
> print(paste("F1 Score:", round(f1, 4)))
[1] "F1 Score: 0.8231"
>
> # Convert predictions to numeric
> numeric_predictions <- as.numeric(predictions_rf == "High")
>
> # Calculate and plot the ROC curve
> roc_curve <- roc(test_set$Stream_cat, numeric_predictions)
Setting levels: control = Low, case = High
Setting direction: controls < cases
> plot(roc_curve, main = "ROC Curve", col = "blue")
>
> roc_obj <- roc(test_set$Stream_cat, numeric_predictions)
Setting levels: control = Low, case = High
Setting direction: controls < cases
> auc_score <- auc(roc_obj)
> auc_score
Area under the curve: 0.8189
> #Evaluating model
> importance(model_rf)
                  MeanDecreaseGini
Loudness                  2949.917
Energy                    2387.626
Instrumentalness          1352.945
```

Figure 13 Showing F1 Score, AUC and Importance of Random Forest Model

Looking at the ROC Curve, since the curve is close to the top-left corner which represents an ideal performance of a classifier, it indicates that the random forest model is a reasonably good classifier. Next, the F1 score was calculated to be 0.8231 indicating that the model has a good balance between precision and recall. Also, since the AUC score was found to be 0.8189 it suggests that the model is good at discriminating between the high and low class. Finally looking at the variable importance, Loudness had the highest Mean Decrease in Gini at 2949.917, followed by Energy at 2387.626 and then Instrumentalness at 1352.945. These scores indicate that the Loudness characteristic is the most influential in the model followed by Energy and Instrumentalness. The importance features were visualized to further grasp the importance of the features in comparison to one another.

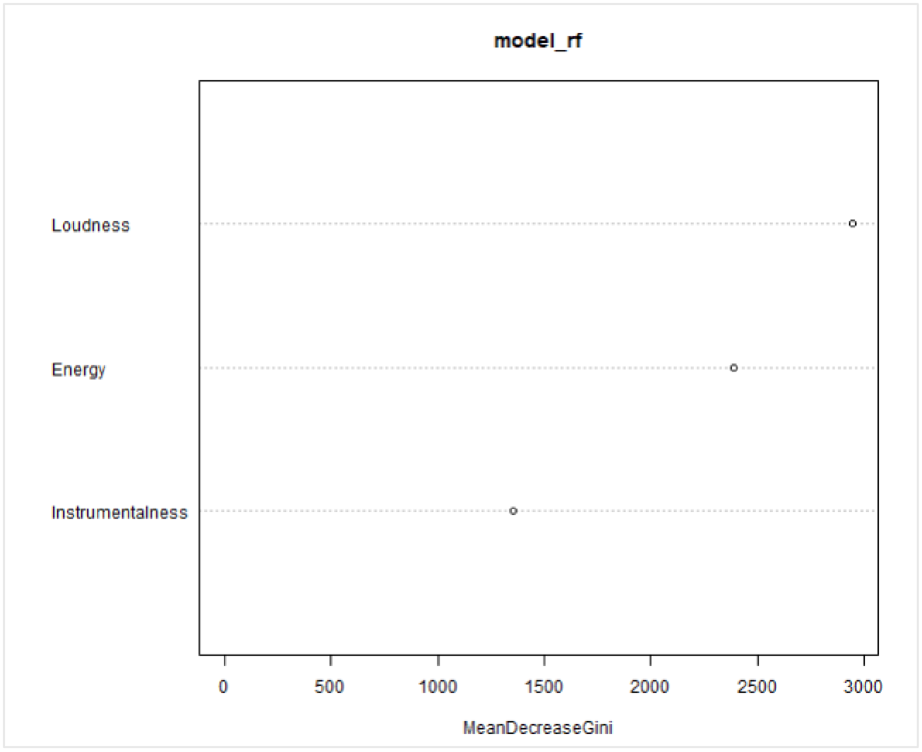


Figure 14 Showing Importance Plot for Random Forest Model

All in all, the results from the analysis address the research question, revealing that Loudness, Energy and Instrumentalness are music characteristics that significantly influence the streaming success of a song. Loudness particularly emerged as the most influential in the random forest model. The random forest model performed the best with high accuracy and balanced predictions. The results not only answer the research question, but also provide insightful implications for music production.

Discussion

The analysis conducted through logistic regression, ridge regression, SVM, and random forest models provided insights to interpret the influence of various musical characteristics on streaming success. Starting with logistic regression, this analysis highlighted that Instrumentalness, Energy and Loudness are significant predictors of streaming success due to their low AIC values and high significance levels. This indicates that these characteristics in particular play a substantial role in determining whether a song will be high streaming. Furthermore, the analysis also explored combinations of these characteristics. The combination Loudness + Danceability * Instrumentalness showed the lowest AIC which goes to show how complex streaming success is. That is, it depends on the relationship of multiple variables rather than single attributes. The vif values obtained for this model were between 1 and 5 which meant that there was some multicollinearity among the predictors. While multicollinearity was moderate, it suggests that there were some interdependencies among features.

To address multicollinearity, it was decided to use a ridge regression model because this technique is often used for analyzing "multiple regression data that suffer from multicollinearity" (Ridge Regression). However, the model was biased in its predictions even after balancing the data set. The poor performance could be attributed to the fact that there were many outliers in the data and ridge regression is strongly influenced by outliers. ("HuberRegressor vs Ridge on Dataset with Strong Outliers"). As mentioned in methodology, Support Vector Machines were then trued due to the fact that they work well with multiple variables. While the predictions were more balanced, the accuracy was poor. This can be attributed to the fact that SVMs generally do not do well with large data sets and are also prone to errors caused by the presence of outliers. This being due to the model trying to "tries to create a sharper decision boundary for our inputs by embedding the data into a hyperplane in the higher dimensional space and maximizing the distance of all the classes from your boundary.” (“How Do Outliers and Missing Values Impact These Classifiers?”) It was clear that a model less sensitive to outliers was needed.

Random Forests combine various decision trees and in doing so it is less sensitive to outliers in the data. (“Random Forest Explained! (Regression and Classification Tasks)”). However, each tree is built of a bootstrap sample so in an unbalanced dataset these samples are more likely to contain more of the majority class due to probability. So, the trees are more likely to learn the characteristics of the majority class leading to a bias in predictions. (Chen and Liaw). These factors can be the reasons that this model performed better, but only when the dataset was balanced achieving an accuracy of 81.8% on the test set, an F1 score of 0.8231 and an AUC score of 0.8189.

This project did have its limitations. Firstly, the definition of how many streams is considered a hit is not clear. Different sources have varying thresholds of what is considered a hit which can lead to inconsistencies when analyzing streaming success across diverse music platforms. This variation in definition could have introduced subjectivity into the analysis which complicates the interpretation of the results. Additionally, the genre of the songs was not considered in modeling. Genres that are more mainstream may have a different definition of how many streams is a hit versus more niche genres which further complicates the analysis. For example, a pop song which has been labeled as successful by having a certain number of streams that would be extraordinarily high for a death metal song due to death metal being less popular. Another limitation of this project is that the models did not consider the length of time a song had been released for. Songs that have been out for longer have had more time to accumulate streams. Without adjusting for release date, the models may have favored older songs and misrepresented the streaming success of newer tracks.

In any future analyses regarding this topic, there are a few recommendations. Firstly, it is recommended to include genre as a variable in the analysis to give a better understanding of what constitutes streaming success in different musical contexts. Also, since the definition of how many hits is considered high streaming is subjective, using anomaly detection techniques can make the analysis better by identifying songs that perform exceptionally well and poorly and then comparing typical patterns within their genre. By using characteristics of outliers for analysis, factors contributing to their unusual performance can be identified. Lastly, it would make the analysis even better in the measurement period of all songs could be standardized. By doing so, the streaming success is measured over a consistent time frame which removes any misrepresentation of the streaming success of older and newer songs.

Conclusion

This project aimed to explore the relationship between various musical characteristics on the streaming success of songs on music platforms like Spotify. Many statistical models and machine learning techniques were implemented and in doing so several key findings emerged.

At first, logistic regression models identified Instrumentalness, energy and loudness as the best predictors of streaming success with a combination of these three showing the most promising results. Due to moderate multicollinearity Ridge Regression was explored in the beginning. However, the Ridge regression model failed to deliver accurate predictions even after the data set was balanced. This was further confirmed by the random forest model which performance poorly initially on the unbalanced data but then performed well on the balanced data set. The support vector machine showed promise as it was not biased in its predictions however it was not accurate. In the end the random forest model performed the best with balance predictions and accuracy of 81.8% on the balanced data set.

This project had several limitations including the subjective definition of high streams on the failure to time a song had been available on streaming platforms. To address these limitations, future research in this area could focus on incorporating analysis specific to genre, utilize anomaly detection to identify extraordinary performances and also standardize the measurement period for number of streams.

All in all, this study was able to provide valuable insights into what factors influence streaming success although the findings are limited by the constraints of the dataset.

References

Chen, Chao, and Andy Liaw. Using Random Forest to Learn Imbalanced Data. statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

"How Do Outliers and Missing Values Impact These Classifiers?" Data Science Stack Exchange, 13 Apr. 2022, datascience.stackexchange.com/questions/109989/how-do-outliers-and-missing-values-impact-these-classifiers#:~:text=Support%20Vector%20Machines%20are%20prone,the%20classes%20from%20your%20boundary. Accessed 21 Apr. 2024.

"HuberRegressor vs Ridge on Dataset with Strong Outliers." Scikit-Learn, 2024, scikit-learn.org/stable/auto_examples/linear_model/plot_huber_vs_ridge.html. Accessed 21 Apr. 2024

Owsinski, Bobby. "How May Streams Make a Hit Song? It's a Few Zeros More than You Think." Forbes, 4 July 2017, www.forbes.com/sites/bobbyowsinski/2017/07/04/streams-hit-song/?sh=1707734711e1. Accessed 18 Apr. 2024.

Raj, Shivam. "Effects of Multi-Collinearity in Logistic Regression, SVM, Random Forest(RF)." Medium, Medium, 9 Aug. 2019, medium.com/@raj5287/effects-of-multi-collinearity-in-logistic-regression-svm-rf-af6766d91f1b#:~:text=Random%20Forest%20uses%20bootstrap%20sampling,different%20set%20of%20data%20points. Accessed 20 Apr. 2024.

"Random Forest Explained! (Regression and Classification Tasks)." Linkedin.com, 2024, www.linkedin.com/pulse/random-forest-explained-regression-classification-tasks-sellahewa/. Accessed 21 Apr. 2024.

Rastelli, Salvatore. “Spotify and Youtube.” Kaggle.com, 2023, www.kaggle.com/datasets/salvatorerastelli/spotify-and-youtube. Accessed 18 Apr. 2024.

Ridge Regression. www.ncss.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf.

Team, CFI. “Ridge.” Corporate Finance Institute, Corporate Finance Institute, 22 Nov. 2023, corporatefinanceinstitute.com/resources/data-science/ridge/#:~:text=Multicollinearity%20happens%20when%20predictor%20variables,the%20reliability%20of%20the%20estimates. Accessed 20 Apr. 2024.

Wilson, Josh. “The Age of Digital; Music Executive Reacts to the Impact of Digitalization in the Music Industry.” Forbes, 8 Nov. 2022, www.forbes.com/sites/joshwilson/2022/09/14/the-age-of-digital-music-executive-reacts-to-the-impact-of-digitalization-in-the-music-industry/?sh=c1a68ef537b1. Accessed 18 Apr. 2024.

“1.4. Support Vector Machines.” Scikit-Learn, 2024, scikit-learn.org/stable/modules/svm.html#:~:text=Support%20vector%20machines%20(SVMs)%20are,than%20the%20number%20of%20samples. Accessed 20 Apr. 2024.