Uncovering the Secrets of Song Success: A Statistical Analysis of Popular Spotify Music

DATA 606 - W2023 Final Project

Kane Smith, Rodrigo Rosales Alvarez, Arhur Trim, Jordan Keelan, Scott Bennett

2023-02-20

Contents

1 Introduction

1.1 Background

Spotify is the world's most popular subscription service for audio streaming. Spotify claims 489 million users, of which 205 million users are Spotify Premium subscribers(1). Spotify has an extensive library of music tracks and gathers data about the music in order to better recommend songs to users, and makes this data available through a web API(2). By analyzing this data, we hope to gain insights into how characteristics of different songs and the artists who created them affect the popularity of the songs. We also aim to explore other trends and relationships within the data, such as whether we can predict the genre of a song based on characteristics such as loudness and musical key. This analysis could potentially be used to identify what factors make songs popular and help artists create music that will be commercially successful (if that is their goal). It may also expose other findings that would be interesting to a general audience of music consumers.

1.2 Problem & Topic Importance

The music industry is in a constant state of evolution, and the popularity of a song can play a significant role in an artist's success. With an abundance of music being produced and released, it can be challenging to predict which songs will achieve popularity. Considering this environment, there is a growing need for a more scientifically informed understanding of the factors that contribute to a song's popularity. Although virality cannot always be predicted, it can be influenced, and record labels and streaming services can benefit greatly from affiliation with viral hits. By understanding the factors that contribute to popularity and identifying them in new songs and artists, they can make informed decisions on music production and marketing strategies.

"Data is becoming a primary way for labels and other tastemakers to find their next stars. Shav Garg is the co-founder of Indify, a company he calls a"music data platform." Music pros use his company to figure out who the next hot artists are, and they were extremely early in noticing artists like Khalid, who the company first featured in the fall of 2015, nearly a year and a half before his debut album." (SETARO, S.)

As data becomes an increasingly crucial tool in the music industry, many music producers and musicians are adopting a formulaic approach to music design. By breaking down successful or unsuccessful songs into their fundamental elements and analyzing patterns influencing listener emotions, they can leverage these findings in the creation of novel music. Understanding these song features and how they contribute to popularity can help artists stay ahead of musical trends and create more impactfull music.

This project's purpose is to analyze the features of songs (as listed in the "Summary of Variables" section above) and to develop multiple statistical models to determine the relevance of these features to a song's popularity. This analysis will provide valuable insights into the music industry and inform artists, music producers, and industry professionals on how to make informed decisions to increase the popularity of their music.

1.3 Data Source

For this project, we used a Kaggle dataset that offers consolidated data from the Spotify web APIs(3). The dataset is structured into two data tables, provided as CSV files. These two data tables are "Tracks" and "Artists"

The "Tracks" "csv" contains information for approximately 600,000 musical tracks available on Spotify. Features include "popularity" as well as a multitude of attributes to describe the character of the music itself i.e, "loudness" score and "danceability" score. The "Artists" csv contains additional data, specifically about the artist such as the list of genre's associated with that artist.

Usage of the dataset is governed by the Community Data License Agreement, which grants: "... a worldwide, non-exclusive, irrevocable (except as provided in Section 5) right to: (a) Use Data; and (b) Publish Data." (4)

1.4 Summary of Variables

One of the key variables of interest/response variables for our project is *popularity*. This is a score given to a track from 0-100, with the most popular track being given a score of 100. For some parts of the analysis, the *popularity* variable was used to classify each song as a "hit" or not. For the purposes of this project, a hit was considered a track in the upper quartile of *popularity*.

The independent variables available in the "tracks" data are:

1. **duration_ms** - the length of the track in ms
2. **explicit** - explicit lyrics
3. **artists** - artist names
4. **danceability** - score for how suited a track is for dancing, 0.0-1.0.
5. **energy** - score for how energetic a track is perceived, 0.0-1.0.
6. **key** - maps Pitch class notation (E.g. 0 = C, 1 = C sharp/D flat, 2 = D, and so on.)
7. **loudness** - decibel loudness of the track range from -60 to 0 dB
8. **mode** - modality of the track (0 is minor, 1 is major)
9. **speechiness** - score for how speech-like a track is, 0.0-1.0. Values close to one indicate something like a podcast (high speechiness).
10. **acousticness** - range of whether a track is acoustic (0.0-1.0)
11. **instrumentalness** - range of whether a track is instrumental (0.0-1.0)
12. **liveness** - range representing audience sounds in the track (0.0-1.0)
13. **valence** - represents how 'happy' a track is (0.0-1.0)
14. **tempo** - the temp of the track in beats per minute
15. **time_signature** - time signature of the track 3-7 (3 represents 3/4 time etc.)

```r
artists_df <- data.frame(read.csv("../spotify_dataset/artists.csv"))
tracks_df <- data.frame(read.csv("../spotify_dataset/tracks.csv"))
```

2 Preliminary Analyses

2.1 Data Cleaning

The dataset was already very clean when it was downloaded, but we still did validation to ensure that everything was good before we began an exploratory analysis and modeling. We started by looking for counts of NAs and duplicate values which there were none. Next, we checked the dimensions of our data frame; there are 22 variables and 529,958 rows.

The most important part of cleaning the data was changing variables that were meant to be factors (explicit, key, mode, time_signature) as factors instead of integers. The other important was to normalize our continuous features using mean normalization to potentially improve the performance of our models and to allow us to use linear discriminant analysis.

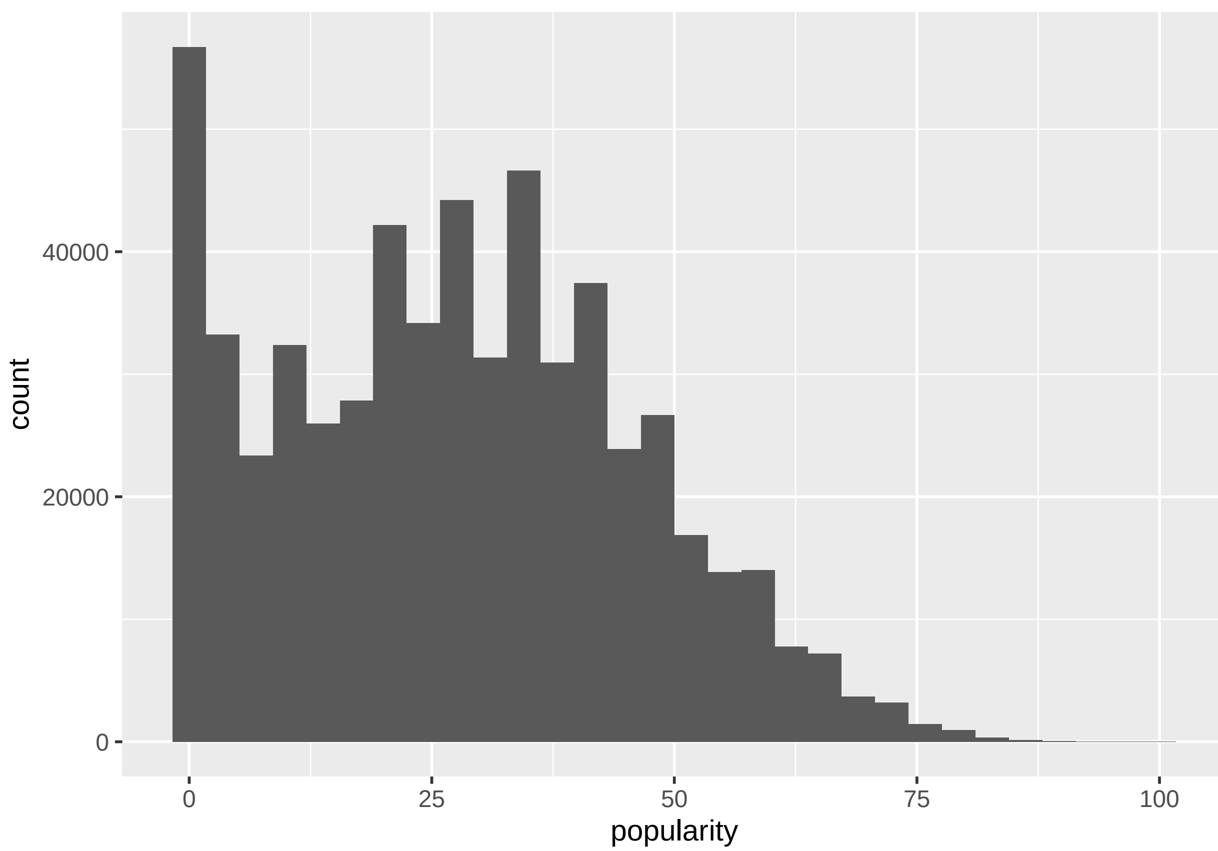
2.2 Visual Investigation

Histograms

Here below we are showing the distribution of our target variable, "popularity".

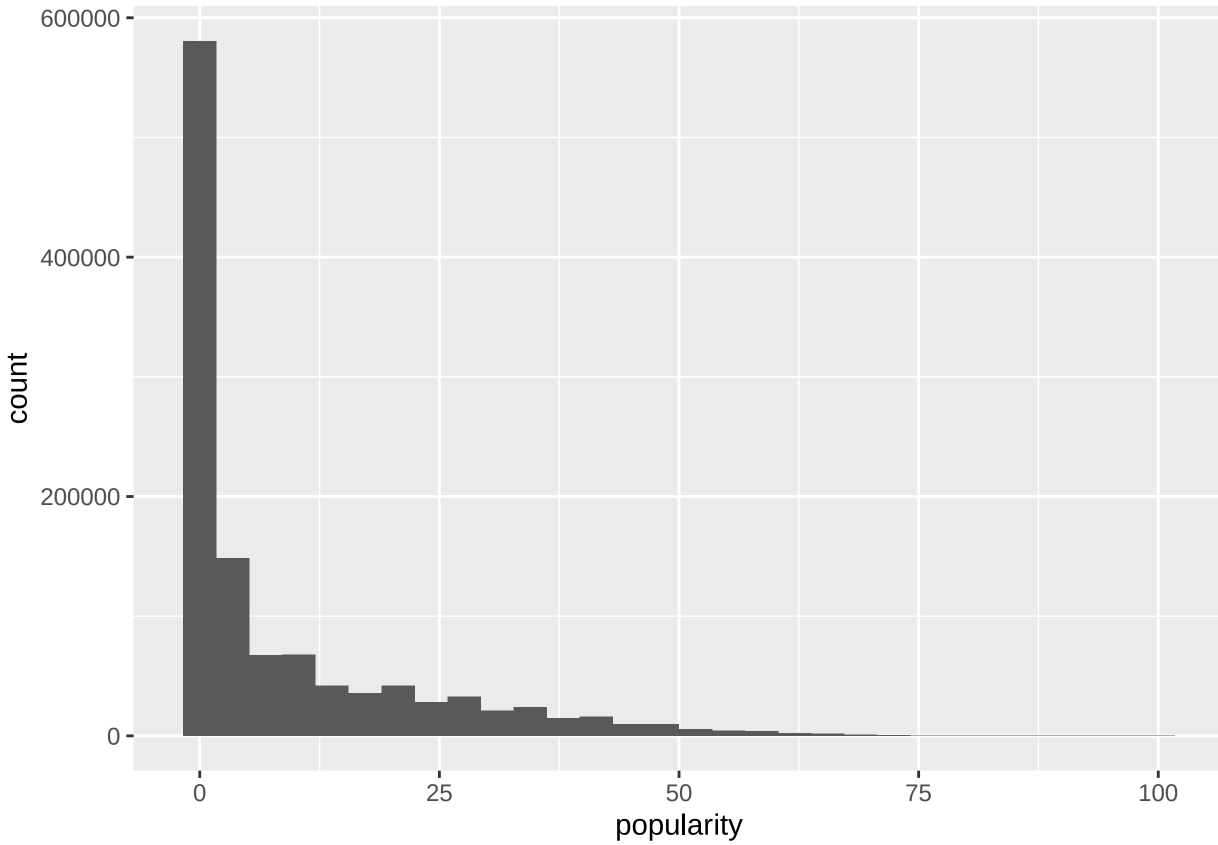
```
tracks_pop <- ggplot(data = tracks_df, aes(x = popularity), title = "Full Tracks Dataset Popularity His
    geom_histogram()
artists_pop <- ggplot(data = artists_df, aes(x = popularity),
    title = "Full Artists Dataset Popularity Histogram") + geom_histogram()

tracks_pop
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
artists_pop
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As we can see there are many data points with popularity=0, for that case we decided to eliminate those data points and try to normalize it.

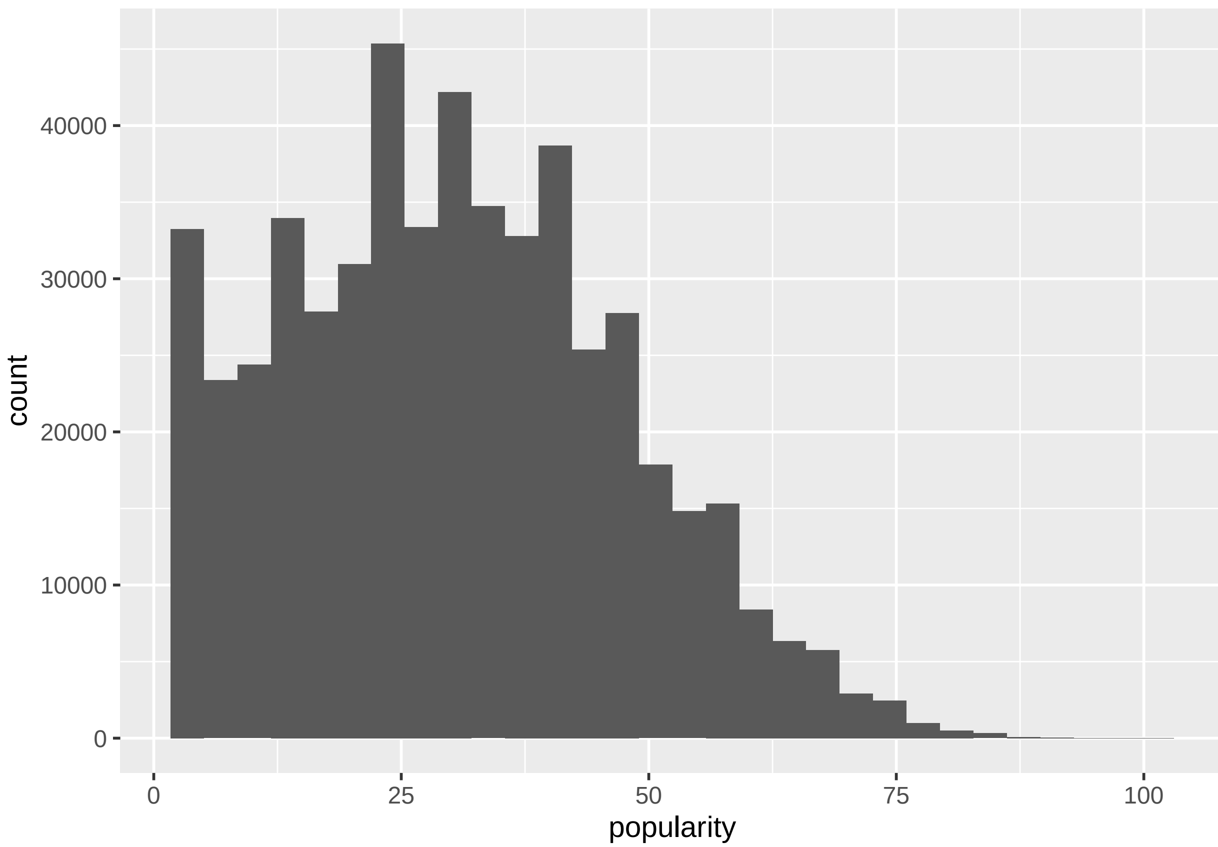
```
# Filter out tracks with popularity less than 1
popular_df <- tracks_df %>%
    filter(popularity > 1)

# normalize Data
popular_labels <- popular_df[, c("id", "name", "artists", "id_artists",
    "release_date")]
popular_factors <- popular_df[, c("explicit", "key", "mode",
    "time_signature")]
popular_numeric <- popular_df[, c("duration_ms", "popularity",
    "danceability", "energy", "loudness", "speechiness", "acousticness",
    "instrumentalness", "liveness", "valence", "tempo")]
popular_scaled <- scale(popular_numeric)
popular_factors <- lapply(popular_factors, factor)
popular_df2 <- data.frame(popular_labels, popular_factors, popular_scaled)
```

We can observe a chi-squared distribution for the filtered ans scaled value of "popularity".

```
# Visualize filtered tracks popularity histogram
tracks_pop_filtered <- ggplot(data = popular_df, aes(x = popularity),
    title = "Full Tracks Dataset Popularity Histogram") + geom_histogram()
```

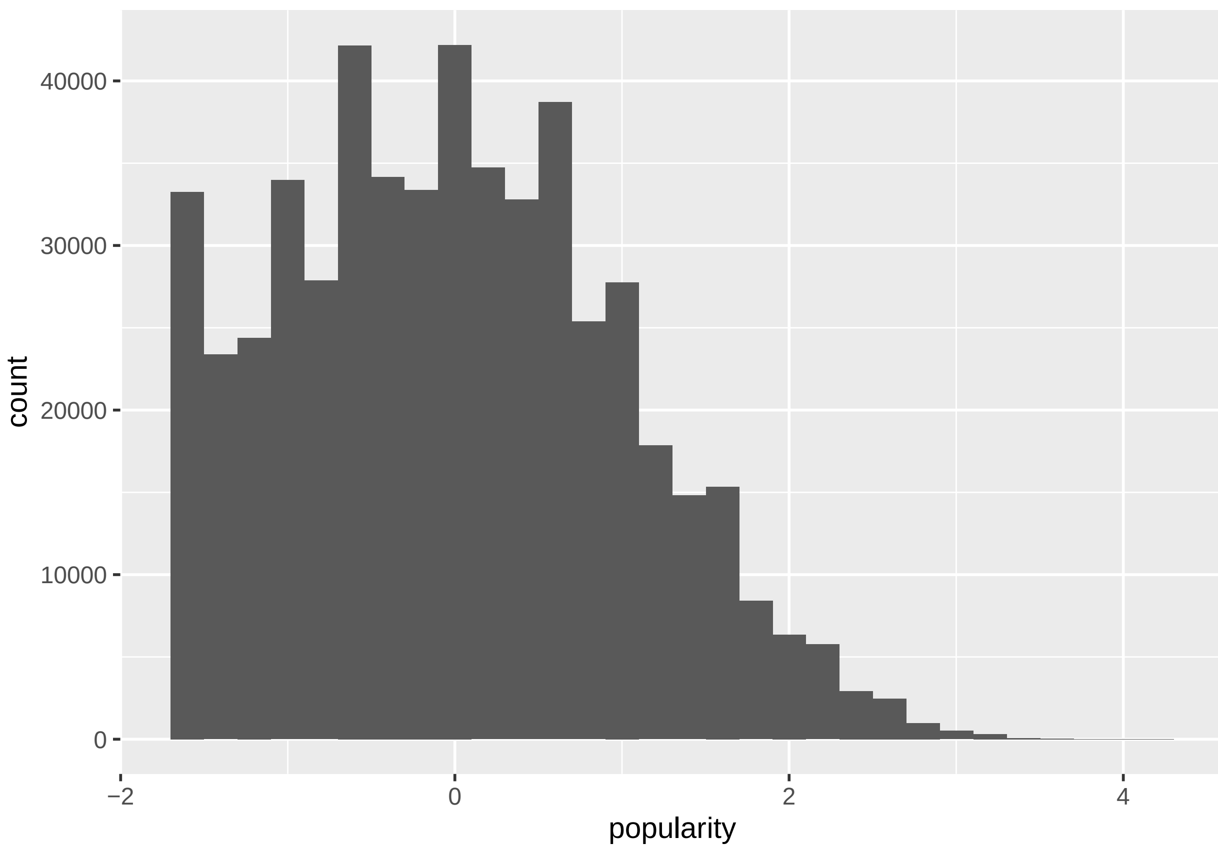
```
# artists_pop_filtered <- ggplot(data=artists_df,
# aes(x=popularity), title='Full Artists Dataset Popularity
# Histogram') + geom_histogram()
tracks_pop_scaled <- ggplot(data = popular_df2, aes(x = popularity),
    title = "Full Tracks Dataset Popularity Histogram") + geom_histogram()

tracks_pop_filtered
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
tracks_pop_scaled
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

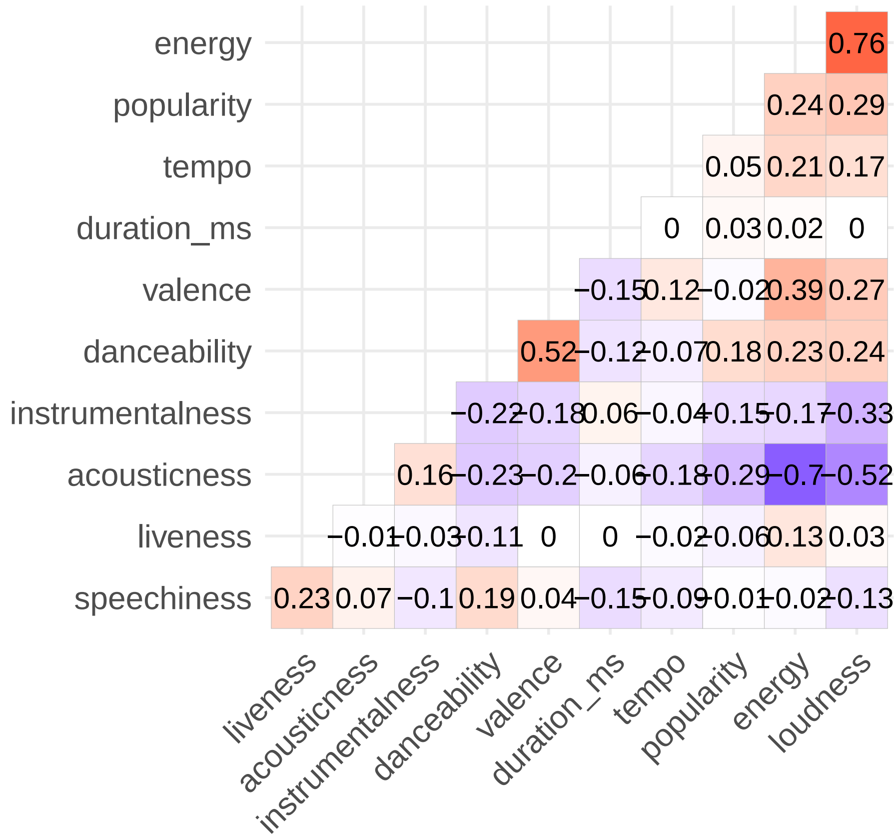



```r
# artists_pop_filtered
```

Correlation Matrix

- Strongest positive correlation between loudness and energy.
- Strongest negative correlation energy and acousticness

```r
# Calculate the correlation matrix
cor_matrix <- cor(popular_numeric)

# plot the correlation
ggcorrplot(cor_matrix, hc.order = TRUE, type = "lower", lab = TRUE,
    show.legend = FALSE)
```



Plots by Year and by Decade

To explore how the some of the variables in the dataset vary between different musical eras, we generated bar charts of the mean popularity, acoustic-ness, energy level, and song duration for each decade in the dataset by simplifying the 'date_published' value to a decade label. These plots show that newer music is generally more popular than older music, particularly music from decades before 1950. It also shows that the length of songs has stayed fairly steady over time, while the musical energy level has increased somewhat and the degree of acoustic instrumentation has decreased.

```
dim(tracks_df)
```

```
## [1] 586672     20
```

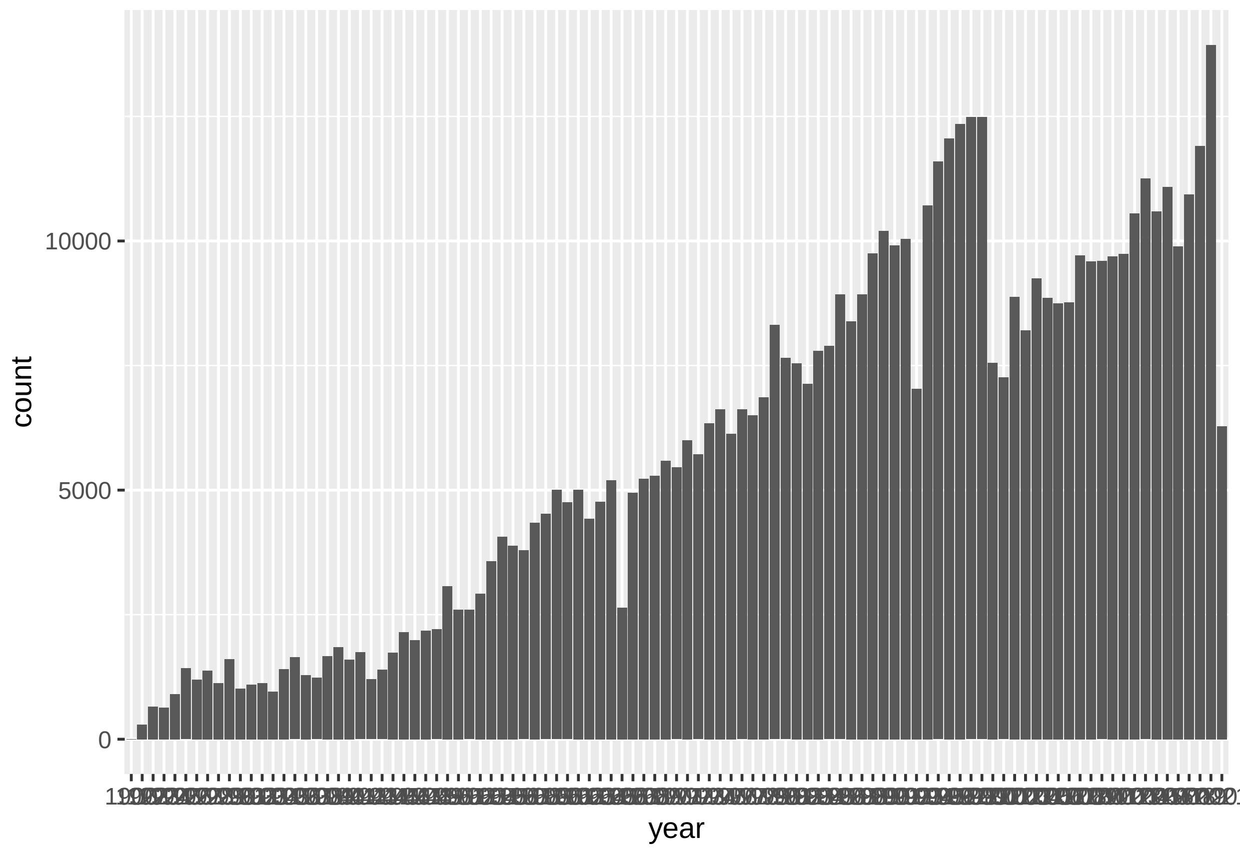
```
tracks_df$year <- eval(substr(tracks_df$release_date, 1, 4))
```

```
test_str <- "1922-02-22"
test_substr <- substr(test_str, 1, 4)
test_substr
```

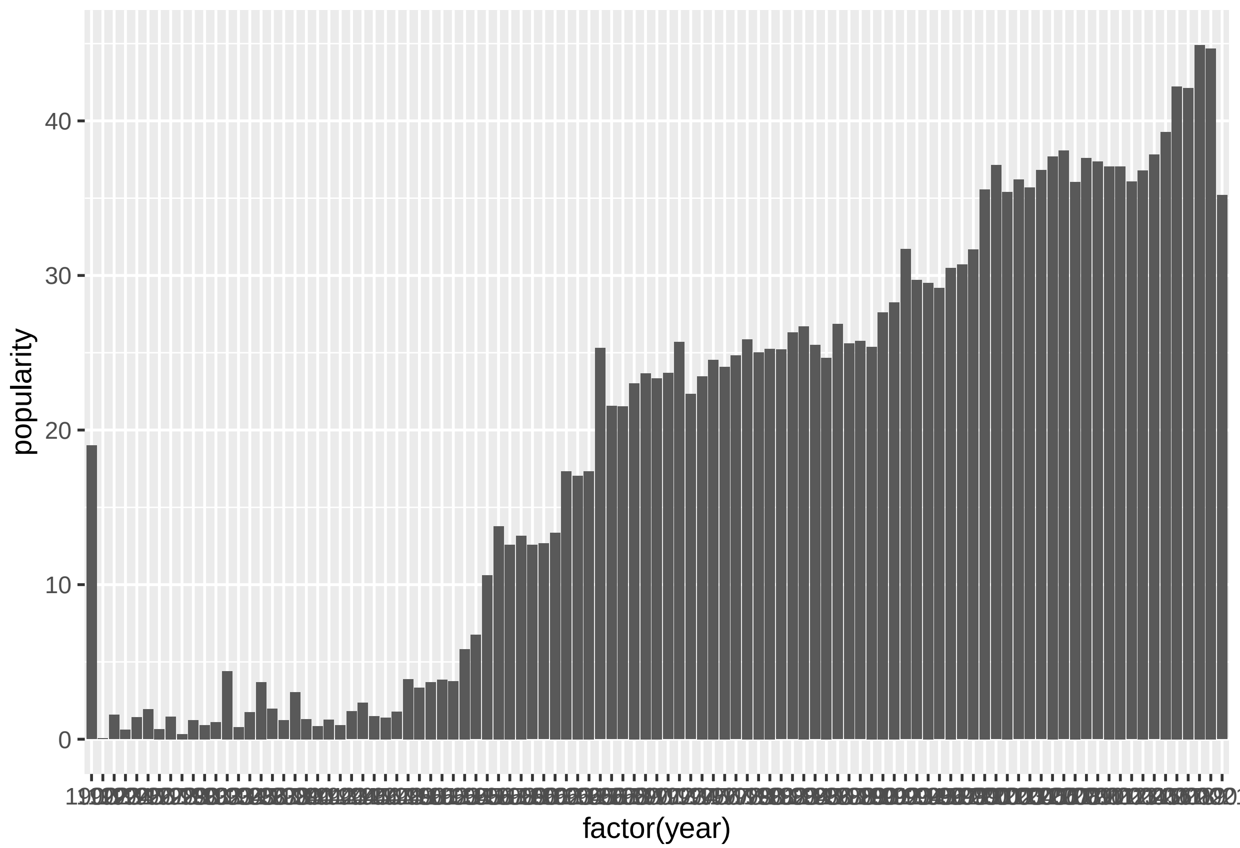
```
## [1] "1922"
```

```
tracks_df$decade <- eval(substr(tracks_df$release_date, 1, 3))
```

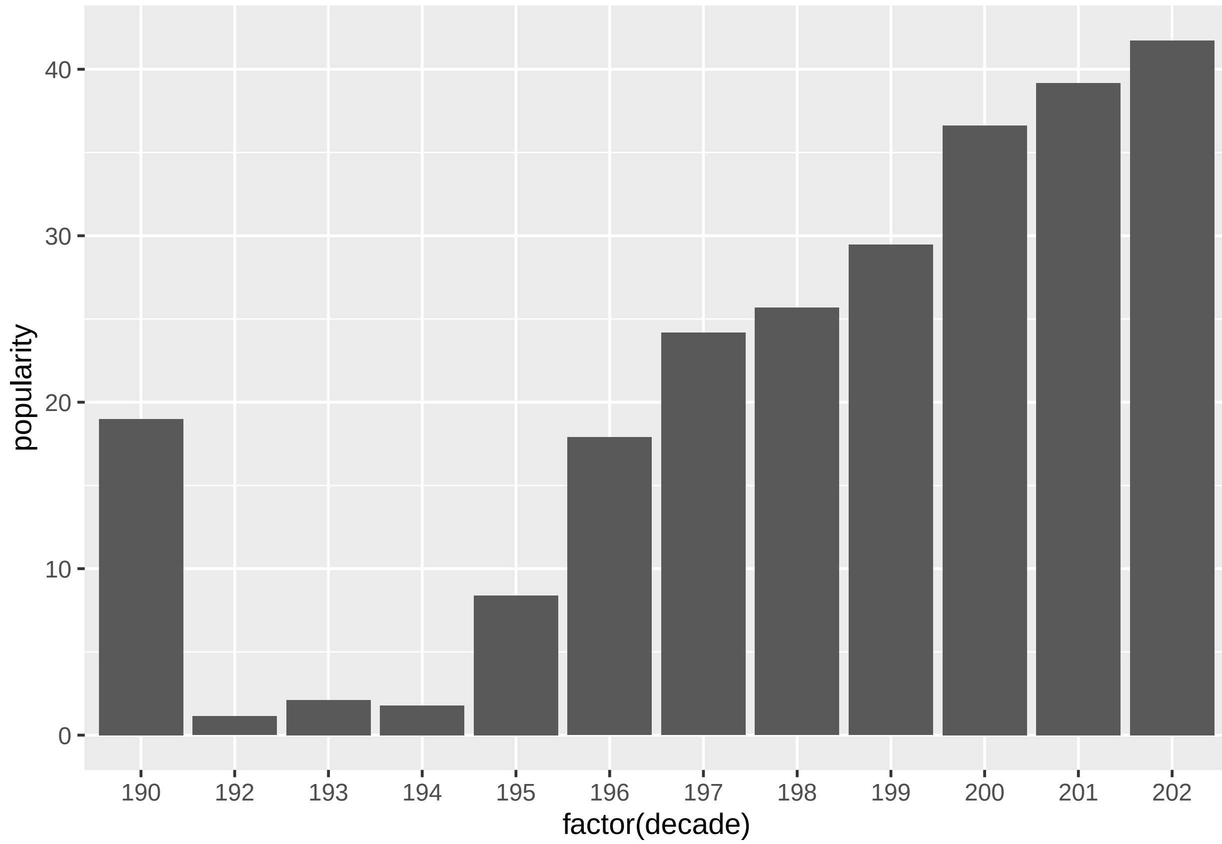
```
ggplot(data = tracks_df, mapping = aes(x = year)) + geom_bar()
```



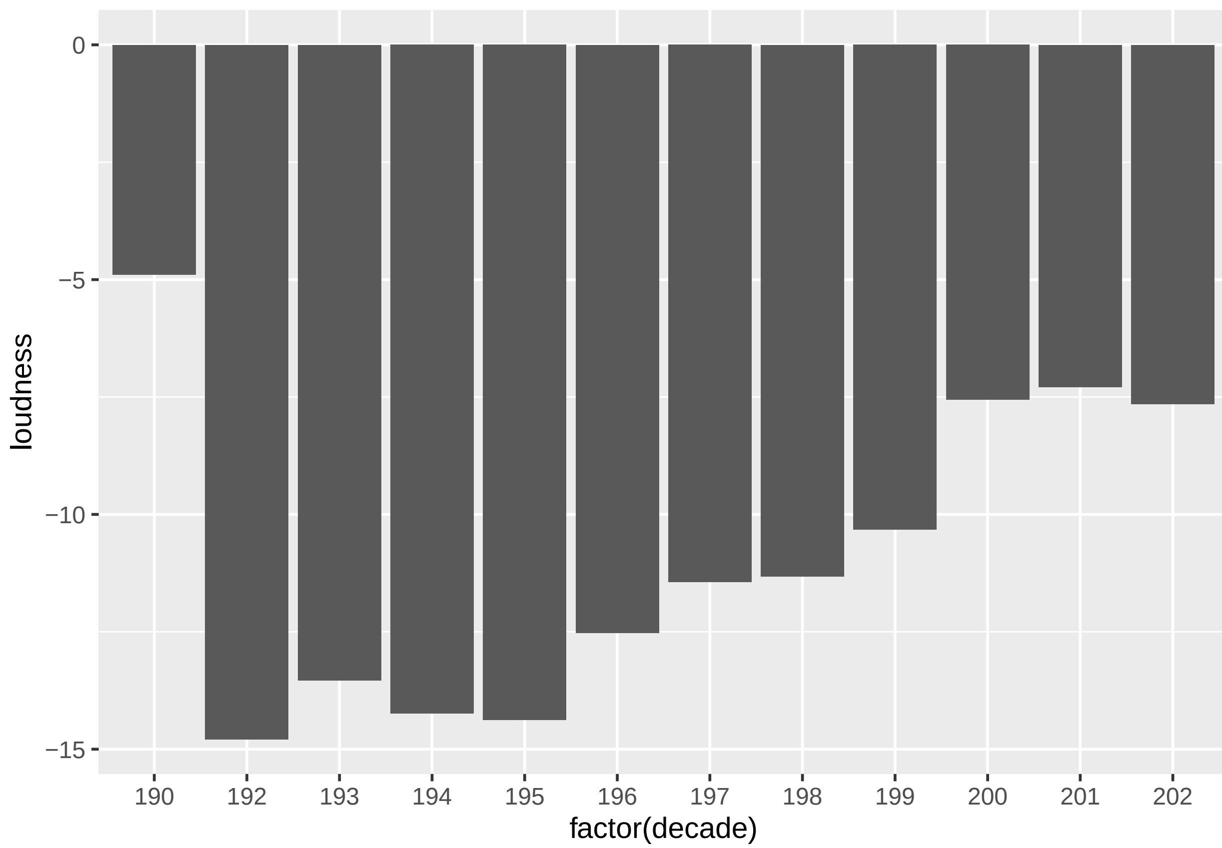

```
ggplot(tracks_df, aes(x = factor(year), y = popularity)) + geom_bar(stat = "summary",
    fun = "mean")
```



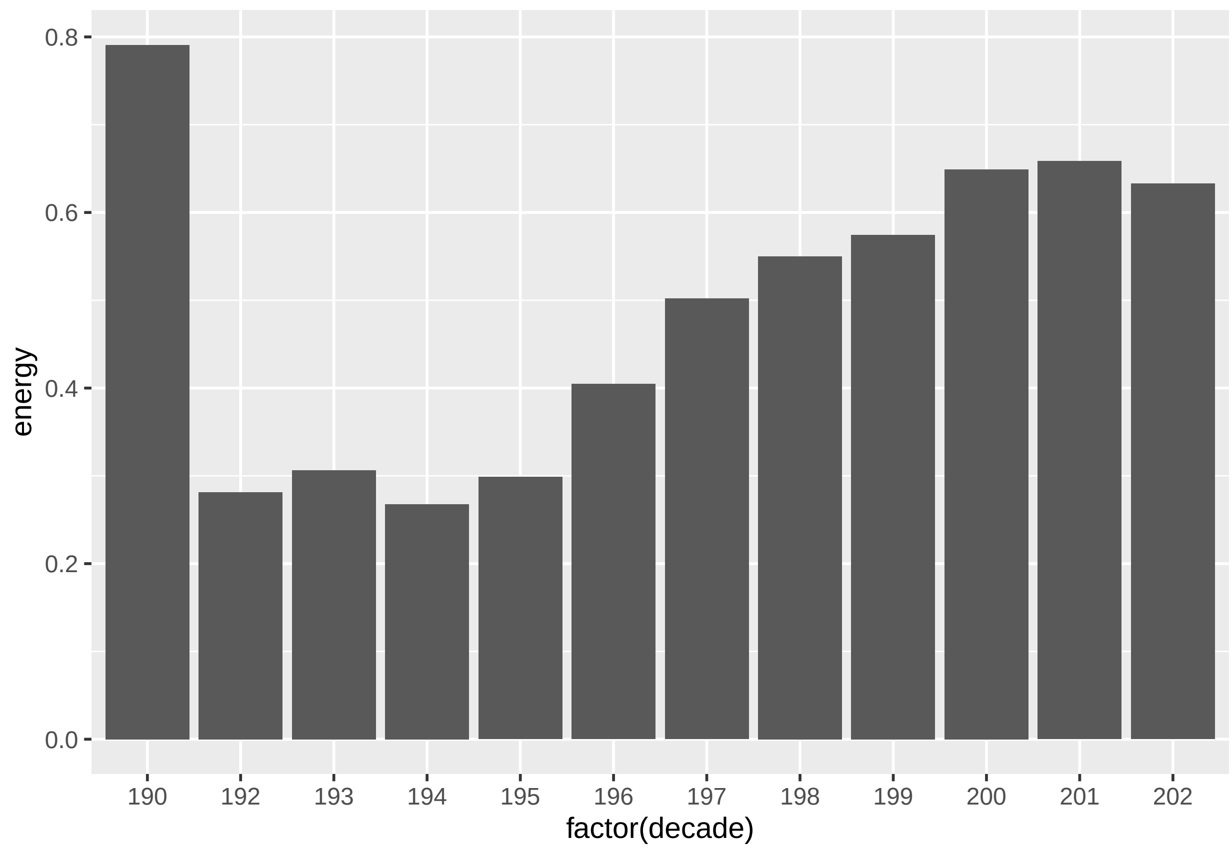

```
ggplot(tracks_df, aes(x = factor(decade), y = popularity)) +
    geom_bar(stat = "summary", fun = "mean")
```




```
ggplot(tracks_df, aes(x = factor(decade), y = loudness)) + geom_bar(stat = "summary",
    fun = "mean")
```



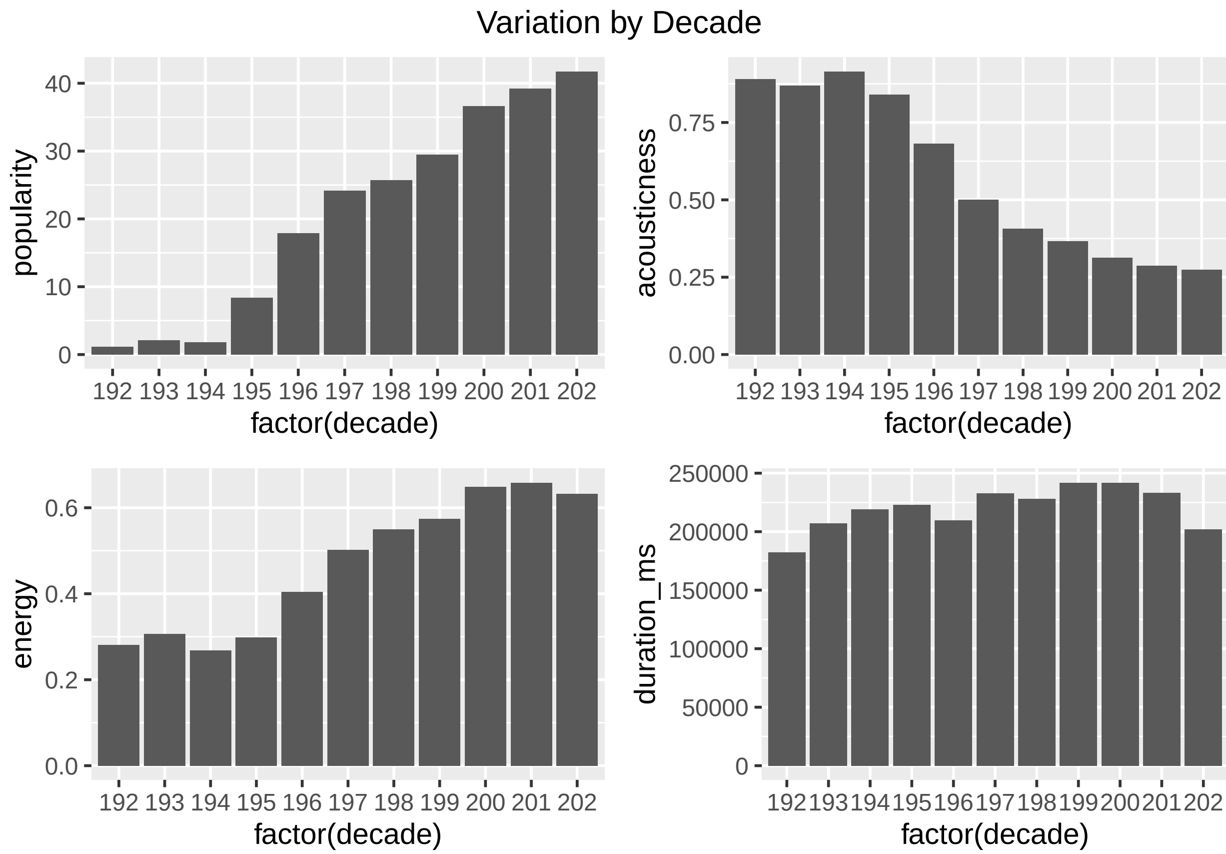

```r
ggplot(tracks_df, aes(x = factor(decade), y = energy)) + geom_bar(stat = "summary",
    fun = "mean")
```




```r
# Remove one record with date value of 1900-01-01 to clean
# up the decade plots
tracks_clean <- tracks_df[-478628, ]
```

```r
plt2 <- ggplot(tracks_clean, aes(x = factor(decade), y = acousticness)) +
    geom_bar(stat = "summary", fun = "mean")

plt3 <- ggplot(tracks_clean, aes(x = factor(decade), y = energy)) +
    geom_bar(stat = "summary", fun = "mean")

plt1 <- ggplot(tracks_clean, aes(x = factor(decade), y = popularity)) +
    geom_bar(stat = "summary", fun = "mean")

plt4 <- ggplot(tracks_clean, aes(x = factor(decade), y = duration_ms)) +
    geom_bar(stat = "summary", fun = "mean")
```

```r
grid.arrange(plt1, plt2, plt3, plt4, top = "Variation by Decade",
    ncol = 2)
```

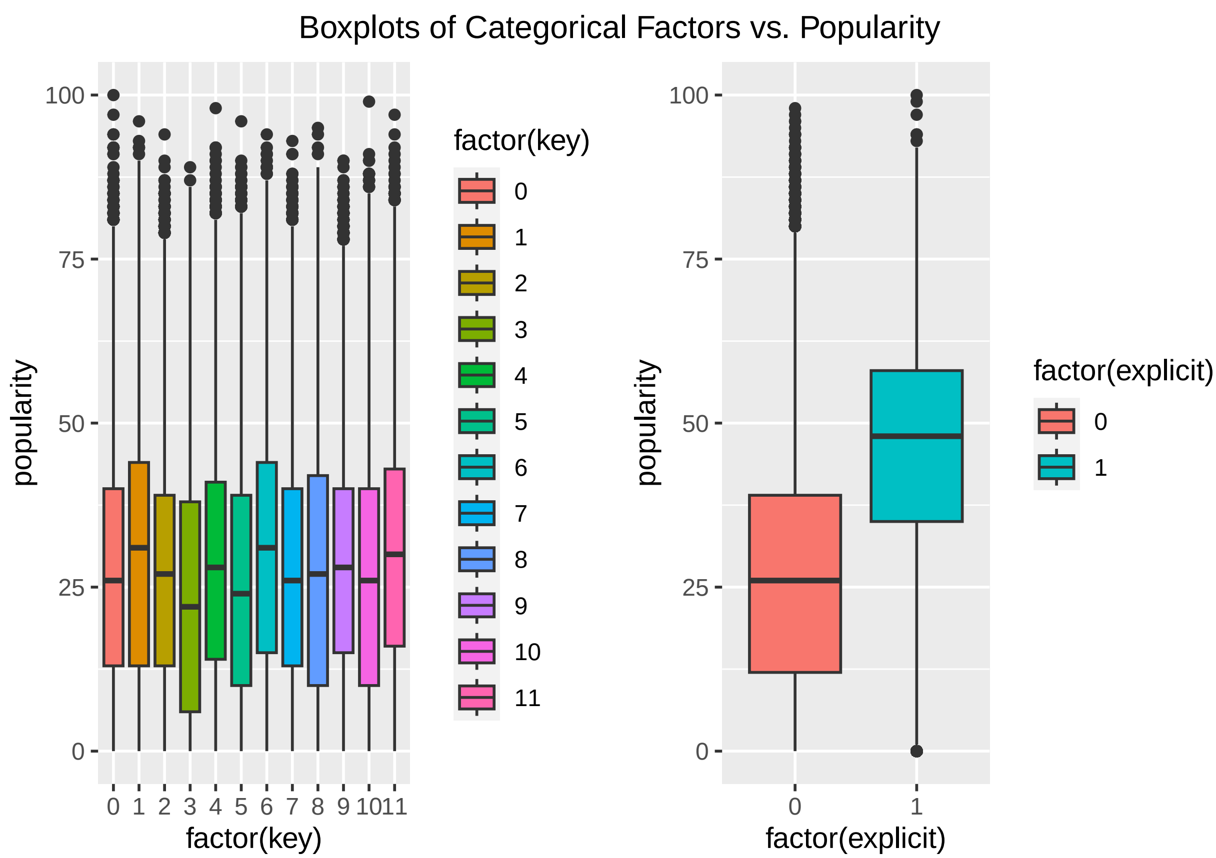


As well, we think its interesting to show the various "trends" of song features by time. Though, it is important to not that this should not constitute a time-series analysis. Though the x-axis is time, this is not a longitudinal data sample. Popularity is not changing over time, it is changing in response to the number of plays that song received and how long ago it received them at the time that this data was extracted. These "trends" cannot be projected forward, as they are all simply a representation of data at a cross-section in time.

Popularity and Hit Boxplots

To begin investigating the idea that *popularity* could be related to features which describe the character of the music, it would be useful to do some preliminary visual exploration of those features. Two of the available "factor" variables (*key* and *explicit*) can be analyzed using boxplots:

```
box1 <- ggplot(data = tracks_df, aes(x = factor(key), y = popularity,
    fill = factor(key))) + geom_boxplot()
box2 <- ggplot(data = tracks_df, aes(x = factor(explicit), y = popularity,
    fill = factor(explicit))) + geom_boxplot()

grid.arrange(box1, box2, ncol = 2, top = "Boxplots of Categorical Factors vs. Popularity")
```



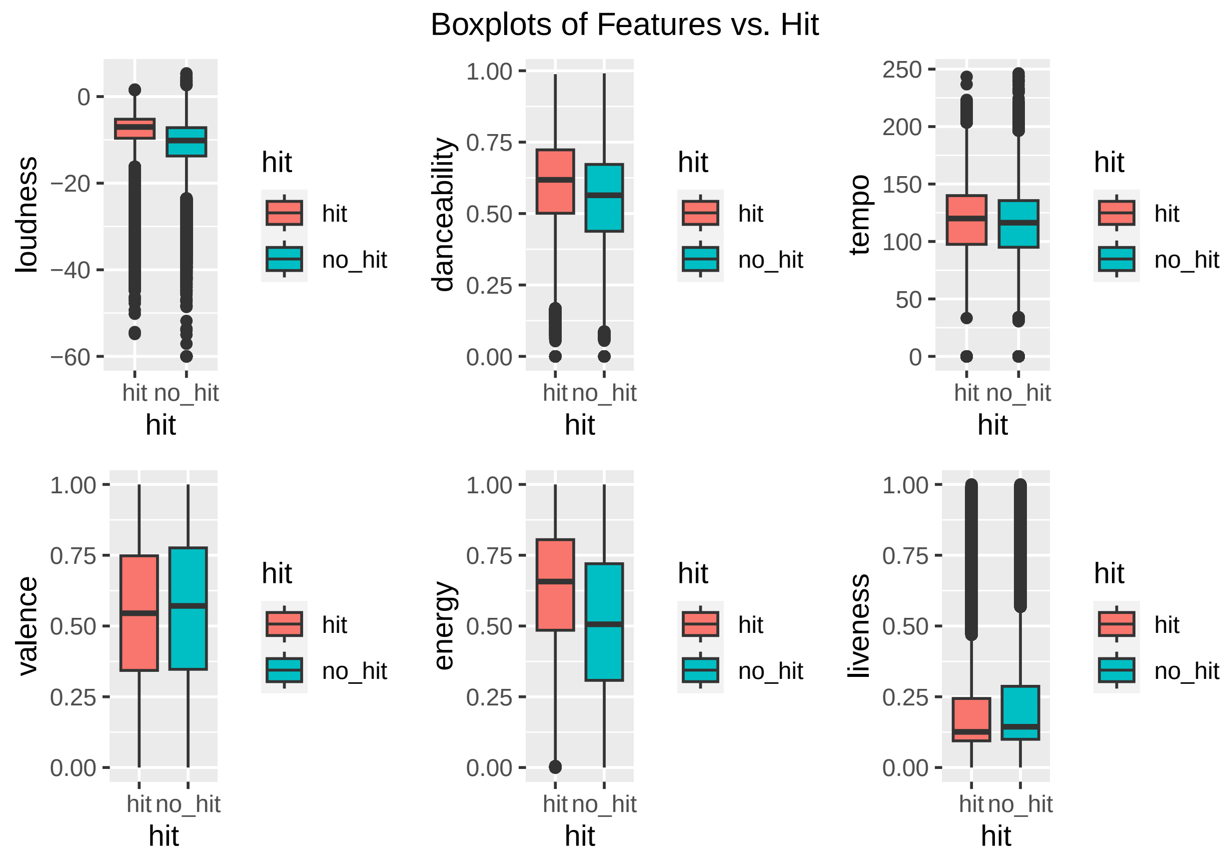
It can be seen that the interquartile range of "popularity" appears to be different for songs flagged as explicit. For the song "key", there may be differences but they are not as obvious in this visual.

In addition to predicting the numerical value of popularity, another opportunity is to predict if a song is a "hit" or not (i.e., predict if the song's popularity will be in the top quartile). By framing in the problem this binary way, we will have an opportunity to investigate more potential models (such as logistiic regression). Additionally, some people might find more meaning in a clear simple, classification such as "hit" and "no hit".

To begin investigating "hit" and "no hit" visually, we first classified a tracks in the top quartile of popularity as "hit", and other songs and "no hit". From here, we produced boxplots of "hit" or "no_hit" for different features such as *loudness* and *danceability*:

```
quart_pop <- tracks_df %>%
    mutate(quartile = factor(ntile(popularity, 4)), hit = case_when(quartile ==
        4 ~ "hit", TRUE ~ "no_hit"))

box1 <- ggplot(data = quart_pop, aes(x = hit, y = loudness, fill = hit)) +
    geom_boxplot()
box2 <- ggplot(data = quart_pop, aes(x = hit, y = danceability,
    fill = hit)) + geom_boxplot()
box3 <- ggplot(data = quart_pop, aes(x = hit, y = tempo, fill = hit)) +
    geom_boxplot()
box4 <- ggplot(data = quart_pop, aes(x = hit, y = valence, fill = hit)) +
    geom_boxplot()
box5 <- ggplot(data = quart_pop, aes(x = hit, y = energy, fill = hit)) +
    geom_boxplot()
box6 <- ggplot(data = quart_pop, aes(x = hit, y = liveness, fill = hit)) +
    geom_boxplot()
```

```
grid.arrange(box1, box2, box3, box4, box5, box6, ncol = 3, top = "Boxplots of Features vs. Hit")
```



At least visually it appears that some of these features could be useful in predicting which tracks are a hit. For example, the mean of "loudness" and "energy" appear to be higher for "hit" than "no_hit". On the other hand, some of these features don't seem to have such an obvious relationship in this visual; the means and interquartile ranges for the tempo plot do not appear to be so different between "hit" and "no hit".

Sampling

To build some of our models, we decided to use random sampling to divide the data into training and testing sets.

```
set.seed(1)

N <- nrow(popular_df2)
n <- N * 0.8

idx = sample(1:N, size = n, replace = FALSE)
train = popular_df2[idx, ]
test = popular_df2[-idx, ]
```

3 Classification of Target Variable - Hit or Not?

As we have stated, the dependent variable that we are using to assess if a song will be classified as a hit is "popularity," a score from 0 to 100 that ranks how popular an artist is relative to other artists on the platform. To use this variable in classification problems, we decided to code it, "1" means that the song is a hit, "0" indicates that the song is not a hit; the threshold used to make the division was the 75th percentile in the popularity column, everything below the 75th percentile is considered "not hit", the rest is considered a "hit".

```
popular_df2 <- popular_df2 %>%
    mutate(popularity_coded = ifelse(popularity >= quantile(popular_df2$popularity,
        c(0.75))[1], 1, 0))

# Convert the outcome variable to a factor
popular_df2$popularity_coded <- as.factor(popular_df2$popularity_coded)

# Confusion matrix function to be added to the control
# object
cfm <- function(data, lev = NULL, model = NULL) {
    cm <- confusionMatrix(table(data$pred, data$obs))
    print(cm)
    cm$byClass
}

# control object for k-fold cross validation
ctrl <- trainControl(method = "cv", number = 10, summaryFunction = cfm)
```

3.1 LDA

Type of Supervised Learning algorithm to commonly used for either binary or multi-class classification problems or as a dimension reduction tool. Using K-fold Cross Validation to divide the normalized dataset in 10 folds, we train and test our model multiple times.

```
# fit a regression model and use k-fold CV to evaluate
# performance
lda_model <- train(popularity_coded ~ duration_ms + explicit +
    danceability + energy + key + loudness + mode + speechiness +
    acousticness + instrumentalness + liveness + valence + tempo +
    time_signature, data = popular_df2, method = "lda", trControl = ctrl)
```

```
## Confusion Matrix and Statistics
##
##
##     0 1
##   0 3 2
##   1 2 3
##
##                Accuracy : 0.6
##                  95% CI : (0.262, 0.878)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.377
##
##                   Kappa : 0.2
##
##  Mcnemar's Test P-Value : 1.000
##
```

```
##             Sensitivity : 0.6
##             Specificity : 0.6
##          Pos Pred Value : 0.6
##          Neg Pred Value : 0.6
##              Prevalence : 0.5
##          Detection Rate : 0.3
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 0.6
##
##        'Positive' Class : 0
##
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. Sensitivity will be used instead.
```

```
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37856 11530
##   1  1331  2279
##
##                Accuracy : 0.757
##                  95% CI : (0.754, 0.761)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.172
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.966
##             Specificity : 0.165
##          Pos Pred Value : 0.767
##          Neg Pred Value : 0.631
##              Prevalence : 0.739
##          Detection Rate : 0.714
##    Detection Prevalence : 0.932
##       Balanced Accuracy : 0.566
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37882 11548
##   1  1305  2261
##
##                Accuracy : 0.757
##                  95% CI : (0.754, 0.761)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.172
```

```
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.967
##             Specificity : 0.164
##          Pos Pred Value : 0.766
##          Neg Pred Value : 0.634
##              Prevalence : 0.739
##          Detection Rate : 0.715
##    Detection Prevalence : 0.933
##       Balanced Accuracy : 0.565
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37898 11565
##   1  1289  2243
##
##                Accuracy : 0.757
##                  95% CI : (0.754, 0.761)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.171
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.967
##             Specificity : 0.162
##          Pos Pred Value : 0.766
##          Neg Pred Value : 0.635
##              Prevalence : 0.739
##          Detection Rate : 0.715
##    Detection Prevalence : 0.933
##       Balanced Accuracy : 0.565
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37785 11506
##   1  1402  2302
##
##                Accuracy : 0.756
##                  95% CI : (0.753, 0.76)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.172
```

```
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.964
##             Specificity : 0.167
##          Pos Pred Value : 0.767
##          Neg Pred Value : 0.621
##              Prevalence : 0.739
##          Detection Rate : 0.713
##    Detection Prevalence : 0.930
##       Balanced Accuracy : 0.565
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37839 11462
##   1  1348  2346
##
##                Accuracy : 0.758
##                  95% CI : (0.755, 0.762)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.178
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.966
##             Specificity : 0.170
##          Pos Pred Value : 0.768
##          Neg Pred Value : 0.635
##              Prevalence : 0.739
##          Detection Rate : 0.714
##    Detection Prevalence : 0.930
##       Balanced Accuracy : 0.568
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37867 11600
##   1  1320  2209
##
##                Accuracy : 0.756
##                  95% CI : (0.753, 0.76)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.166
```

```
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.966
##             Specificity : 0.160
##          Pos Pred Value : 0.766
##          Neg Pred Value : 0.626
##              Prevalence : 0.739
##          Detection Rate : 0.715
##    Detection Prevalence : 0.933
##       Balanced Accuracy : 0.563
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37856 11497
##   1  1332  2312
##
##                Accuracy : 0.758
##                  95% CI : (0.754, 0.762)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.175
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.966
##             Specificity : 0.167
##          Pos Pred Value : 0.767
##          Neg Pred Value : 0.634
##              Prevalence : 0.739
##          Detection Rate : 0.714
##    Detection Prevalence : 0.931
##       Balanced Accuracy : 0.567
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37806 11537
##   1  1381  2272
##
##                Accuracy : 0.756
##                  95% CI : (0.753, 0.76)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.17
```

```
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.965
##             Specificity : 0.165
##          Pos Pred Value : 0.766
##          Neg Pred Value : 0.622
##              Prevalence : 0.739
##          Detection Rate : 0.713
##    Detection Prevalence : 0.931
##       Balanced Accuracy : 0.565
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37839 11507
##   1  1348  2302
##
##                Accuracy : 0.757
##                  95% CI : (0.754, 0.761)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.174
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.966
##             Specificity : 0.167
##          Pos Pred Value : 0.767
##          Neg Pred Value : 0.631
##              Prevalence : 0.739
##          Detection Rate : 0.714
##    Detection Prevalence : 0.931
##       Balanced Accuracy : 0.566
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37816 11511
##   1  1371  2298
##
##                Accuracy : 0.757
##                  95% CI : (0.753, 0.761)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.172
```

```
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.965
##             Specificity : 0.166
##          Pos Pred Value : 0.767
##          Neg Pred Value : 0.626
##              Prevalence : 0.739
##          Detection Rate : 0.714
##    Detection Prevalence : 0.931
##       Balanced Accuracy : 0.566
##
##        'Positive' Class : 0
##
```

```
print(lda_model)
```

```
## Linear Discriminant Analysis
##
## 529958 samples
##     14 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476962, 476962, 476963, 476963, 476963, 476962, ...
## Resampling results:
##
##   Sensitivity  Specificity  Pos Pred Value  Neg Pred Value  Precision  Recall
##   0.9657       0.1653       0.7665          0.6296          0.7665     0.9657
##   F1      Prevalence  Detection Rate  Detection Prevalence  Balanced Accuracy
##   0.8547  0.7394      0.7141          0.9316                0.5655
```

3.1.1 LDA Assumptions

Multivariate Normality - mulri.norm Test

H_0 (Null hypothesis): The variables follow a multivariate normal distribution.

H_A (Alternative hypothesis): The variables do not follow a multivariate normal distribution.

We will use an alpha value of 0.05.

```
# Multivariate Normality
N <- nrow(popular_scaled)
idx = sample(1:N, size = 1000, replace = FALSE)
popular_sample = popular_scaled[idx, ]
mult.norm(popular_sample)$mult.test
```

```
##           Beta-hat   kappa p-val
## Skewness     225.4 37560.7     0
## Kurtosis     472.0   307.6     0
```

Equality of Variance - Levene Test

H_0 (Null hypothesis): Sample variances are equal.

H_A (Alternative hypothesis): Samples variances are not equal.

We will use an alpha value of 0.05.

```
# Equality of Variance
popularity_var_test <- data.frame(popular_factors, popular_df2["popularity"])
N <- nrow(popularity_var_test)
idx = sample(1:N, size = 1000, replace = FALSE)
popular_sample2 = popularity_var_test[idx, ]
leveneTest(popularity ~ ., data = popular_sample2)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group  86    1.02   0.44
##       913
```

Despite that our model obtained an Accuracy value of 0.756, we decided not to use this model as it failed both assumptions needed, the Multivariate Normality and Equality of Variances. No QDA model used, as the Multivariate Normality assumption was not met.

3.2 Logistic Regression

```
# fit a regression model and use k-fold CV to evaluate
# performance
lr_model <- train(popularity_coded ~ duration_ms + explicit +
    danceability + energy + key + loudness + mode + speechiness +
    acousticness + instrumentalness + liveness + valence + tempo +
    time_signature, data = popular_df2, method = "glm", trControl = ctrl,
    family = "binomial")
```

```
## Confusion Matrix and Statistics
##
##
##    0 1
##  0 2 3
##  1 3 2
##
##                Accuracy : 0.4
##                  95% CI : (0.122, 0.738)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.828
##
##                   Kappa : -0.2
##
##  Mcnemar's Test P-Value : 1.000
##
##             Sensitivity : 0.4
##             Specificity : 0.4
##          Pos Pred Value : 0.4
##          Neg Pred Value : 0.4
##              Prevalence : 0.5
##          Detection Rate : 0.2
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 0.4
##
##        'Positive' Class : 0
##
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. Sensitivity will be used instead.
```

```
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37651 11317
##   1  1537  2492
##
##                Accuracy : 0.757
##                  95% CI : (0.754, 0.761)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.183
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.961
##             Specificity : 0.180
##          Pos Pred Value : 0.769
##          Neg Pred Value : 0.619
##              Prevalence : 0.739
##          Detection Rate : 0.710
##    Detection Prevalence : 0.924
##       Balanced Accuracy : 0.571
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37628 11263
##   1  1559  2546
##
##                Accuracy : 0.758
##                  95% CI : (0.754, 0.762)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.187
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.960
##             Specificity : 0.184
##          Pos Pred Value : 0.770
##          Neg Pred Value : 0.620
##              Prevalence : 0.739
##          Detection Rate : 0.710
##    Detection Prevalence : 0.923
##       Balanced Accuracy : 0.572
```

```
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37658 11294
##   1  1529  2515
##
##                Accuracy : 0.758
##                  95% CI : (0.754, 0.762)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.186
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.961
##             Specificity : 0.182
##          Pos Pred Value : 0.769
##          Neg Pred Value : 0.622
##              Prevalence : 0.739
##          Detection Rate : 0.711
##    Detection Prevalence : 0.924
##       Balanced Accuracy : 0.572
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37714 11318
##   1  1473  2491
##
##                Accuracy : 0.759
##                  95% CI : (0.755, 0.762)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.186
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.962
##             Specificity : 0.180
##          Pos Pred Value : 0.769
##          Neg Pred Value : 0.628
##              Prevalence : 0.739
##          Detection Rate : 0.712
##    Detection Prevalence : 0.925
##       Balanced Accuracy : 0.571
```

```
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37742 11236
##   1  1445  2573
##
##                Accuracy : 0.761
##                  95% CI : (0.757, 0.764)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.194
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.963
##             Specificity : 0.186
##          Pos Pred Value : 0.771
##          Neg Pred Value : 0.640
##              Prevalence : 0.739
##          Detection Rate : 0.712
##    Detection Prevalence : 0.924
##       Balanced Accuracy : 0.575
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37672 11181
##   1  1515  2628
##
##                Accuracy : 0.76
##                  95% CI : (0.757, 0.764)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.196
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.961
##             Specificity : 0.190
##          Pos Pred Value : 0.771
##          Neg Pred Value : 0.634
##              Prevalence : 0.739
##          Detection Rate : 0.711
##    Detection Prevalence : 0.922
##       Balanced Accuracy : 0.576
```

```
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37650 11225
##   1  1537  2584
##
##                Accuracy : 0.759
##                  95% CI : (0.756, 0.763)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.191
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.961
##             Specificity : 0.187
##          Pos Pred Value : 0.770
##          Neg Pred Value : 0.627
##              Prevalence : 0.739
##          Detection Rate : 0.710
##    Detection Prevalence : 0.922
##       Balanced Accuracy : 0.574
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##         0     1
##   0 37703 11265
##   1  1484  2543
##
##                Accuracy : 0.759
##                  95% CI : (0.756, 0.763)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.19
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.962
##             Specificity : 0.184
##          Pos Pred Value : 0.770
##          Neg Pred Value : 0.631
##              Prevalence : 0.739
##          Detection Rate : 0.711
##    Detection Prevalence : 0.924
##       Balanced Accuracy : 0.573
```

```
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##          0     1
##   0 37709 11197
##   1  1478  2611
##
##                Accuracy : 0.761
##                  95% CI : (0.757, 0.764)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.196
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.962
##             Specificity : 0.189
##          Pos Pred Value : 0.771
##          Neg Pred Value : 0.639
##              Prevalence : 0.739
##          Detection Rate : 0.712
##    Detection Prevalence : 0.923
##       Balanced Accuracy : 0.576
##
##        'Positive' Class : 0
##
## Confusion Matrix and Statistics
##
##
##          0     1
##   0 37696 11285
##   1  1491  2523
##
##                Accuracy : 0.759
##                  95% CI : (0.755, 0.763)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.188
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.962
##             Specificity : 0.183
##          Pos Pred Value : 0.770
##          Neg Pred Value : 0.629
##              Prevalence : 0.739
##          Detection Rate : 0.711
##    Detection Prevalence : 0.924
##       Balanced Accuracy : 0.572
```

```
##
##        'Positive' Class : 0
##
```

```
summary(lr_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.378  -0.782  -0.560   0.764   3.973
##
## Coefficients:
##                   Estimate Std. Error z value             Pr(>|z|)
## (Intercept)        1.55382    0.14363   10.82 < 0.0000000000000002 ***
## duration_ms        0.01789    0.00387    4.63     0.00000368242839 ***
## explicit1          1.26024    0.01499   84.07 < 0.0000000000000002 ***
## danceability       0.37619    0.00446   84.33 < 0.0000000000000002 ***
## energy            -0.18598    0.00720  -25.83 < 0.0000000000000002 ***
## key1               0.25819    0.01563   16.52 < 0.0000000000000002 ***
## key2              -0.04236    0.01395   -3.04              0.00240 **
## key3               0.12211    0.02106    5.80     0.00000000668318 ***
## key4               0.04076    0.01515    2.69              0.00712 **
## key5               0.05113    0.01502    3.40              0.00066 ***
## key6               0.26415    0.01700   15.54 < 0.0000000000000002 ***
## key7              -0.00958    0.01353   -0.71              0.47901
## key8               0.23179    0.01692   13.70 < 0.0000000000000002 ***
## key9               0.00189    0.01388    0.14              0.89159
## key10              0.12010    0.01658    7.24     0.00000000000044 ***
## key11              0.17803    0.01585   11.23 < 0.0000000000000002 ***
## loudness           0.68378    0.00680  100.62 < 0.0000000000000002 ***
## mode1              0.02835    0.00733    3.87              0.00011 ***
## speechiness       -0.21075    0.00544  -38.72 < 0.0000000000000002 ***
## acousticness      -0.26449    0.00503  -52.63 < 0.0000000000000002 ***
## instrumentalness -0.05438    0.00427  -12.73 < 0.0000000000000002 ***
## liveness          -0.07829    0.00380  -20.61 < 0.0000000000000002 ***
## valence           -0.39463    0.00443  -89.09 < 0.0000000000000002 ***
## tempo              0.07500    0.00366   20.51 < 0.0000000000000002 ***
## time_signature1  -2.99683    0.14944  -20.05 < 0.0000000000000002 ***
## time_signature3  -3.04856    0.14376  -21.21 < 0.0000000000000002 ***
## time_signature4  -2.93093    0.14340  -20.44 < 0.0000000000000002 ***
## time_signature5  -2.80910    0.14647  -19.18 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 608015  on 529957  degrees of freedom
## Residual deviance: 536462  on 529930  degrees of freedom
## AIC: 536518
##
## Number of Fisher Scoring iterations: 5
```

```r
# Average confusion matrix
lr_0_0 <- c(37651, 37628, 37658, 37714, 37742, 37672, 37650,
    37703, 37709, 37696)
lr_0_1 <- c(11317, 11263, 11294, 11318, 11236, 11181, 11225,
    11265, 11197, 11285)
lr_1_0 <- c(1537, 1559, 1529, 1473, 1445, 1515, 1537, 1484, 1478,
    1491)
lr_1_1 <- c(2492, 2546, 2515, 2491, 2573, 2628, 2584, 2543, 2611,
    2523)
lr_conf_matrix <- data.frame(`Predicted 0, Actual 0` = mean(lr_0_0),
    `Predicted 0, Actual 1` = mean(lr_0_1), `Predicted 1, Actual 0` = mean(lr_1_0),
    `Predicted 1, Actual 1` = mean(lr_1_1))
paste("Type 1 Error:", (lr_conf_matrix[1, 3]/sum(lr_conf_matrix[1,
    ])) * 100, "%")
```

```
## [1] "Type 1 Error: 2.83947029764623 %"
```

```r
paste("Type 2 Error:", (lr_conf_matrix[1, 2]/sum(lr_conf_matrix[1,
    ])) * 100, "%")
```

```
## [1] "Type 2 Error: 21.2433815509908 %"
```

We believe that in our case, type 1 error is the more important error, as that means our model predicts a song to be a hit but actually in not a hit. In this case, if someone were to use our model with the goal of making a hit song on Spotify, they would waste their time and error to make a song that fails. In other words, type 1 error results in wasted resources while type 2 error does not.

We will attempt to improve our model by balancing our classes in the training data.

3.2.1 Balanced Logistic Regression

Due to the way we classified "Hit" and "No Hit", we believe there may be an imbalance of the labels in our data which is affecting our model. We will correct this by adding another to the control object that will account for this.

```r
# need to define class levels with valid r variable names
levels(popular_df2$popularity_coded) <- c("flop", "hit")

ctrl_balanced <- trainControl(method = "cv", number = 10, summaryFunction = cfm,
    sampling = "down", classProbs = TRUE)
```

Running our balanced logistic regression:

```r
lr_model_balanced <- train(popularity_coded ~ duration_ms + explicit +
    danceability + energy + key + loudness + mode + speechiness +
    acousticness + instrumentalness + liveness + valence + tempo +
    time_signature, data = popular_df2, method = "glm", trControl = ctrl_balanced,
    family = "binomial")
```

```
## Confusion Matrix and Statistics
##
##
##        flop hit
##   flop    3   2
##   hit     2   3
##
##                Accuracy : 0.6
```

```
##                  95% CI : (0.262, 0.878)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.377
##
##                   Kappa : 0.2
##
##  Mcnemar's Test P-Value : 1.000
##
##             Sensitivity : 0.6
##             Specificity : 0.6
##          Pos Pred Value : 0.6
##          Neg Pred Value : 0.6
##              Prevalence : 0.5
##          Detection Rate : 0.3
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 0.6
##
##        'Positive' Class : flop
##
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. Sensitivity will be used instead.
```

```
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25612  4280
##   hit  13575  9529
##
##                Accuracy : 0.663
##                  95% CI : (0.659, 0.667)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.282
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.654
##             Specificity : 0.690
##          Pos Pred Value : 0.857
##          Neg Pred Value : 0.412
##              Prevalence : 0.739
##          Detection Rate : 0.483
##    Detection Prevalence : 0.564
##       Balanced Accuracy : 0.672
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25792  4201
```

```
##   hit  13395  9608
##
##                Accuracy : 0.668
##                  95% CI : (0.664, 0.672)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.291
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.658
##             Specificity : 0.696
##          Pos Pred Value : 0.860
##          Neg Pred Value : 0.418
##              Prevalence : 0.739
##          Detection Rate : 0.487
##    Detection Prevalence : 0.566
##       Balanced Accuracy : 0.677
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25919  4349
##   hit  13268  9459
##
##                Accuracy : 0.668
##                  95% CI : (0.664, 0.672)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.287
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.661
##             Specificity : 0.685
##          Pos Pred Value : 0.856
##          Neg Pred Value : 0.416
##              Prevalence : 0.739
##          Detection Rate : 0.489
##    Detection Prevalence : 0.571
##       Balanced Accuracy : 0.673
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25764  4084
```

```
##   hit  13423  9725
##
##                Accuracy : 0.67
##                  95% CI : (0.666, 0.674)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.297
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.657
##             Specificity : 0.704
##          Pos Pred Value : 0.863
##          Neg Pred Value : 0.420
##              Prevalence : 0.739
##          Detection Rate : 0.486
##    Detection Prevalence : 0.563
##       Balanced Accuracy : 0.681
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25677  4207
##   hit  13510  9602
##
##                Accuracy : 0.666
##                  95% CI : (0.662, 0.67)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.288
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.655
##             Specificity : 0.695
##          Pos Pred Value : 0.859
##          Neg Pred Value : 0.415
##              Prevalence : 0.739
##          Detection Rate : 0.485
##    Detection Prevalence : 0.564
##       Balanced Accuracy : 0.675
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25788  4159
```

```
##   hit  13399  9650
##
##                Accuracy : 0.669
##                  95% CI : (0.665, 0.673)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.293
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.658
##             Specificity : 0.699
##          Pos Pred Value : 0.861
##          Neg Pred Value : 0.419
##              Prevalence : 0.739
##          Detection Rate : 0.487
##    Detection Prevalence : 0.565
##       Balanced Accuracy : 0.678
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25582  4216
##   hit  13606  9593
##
##                Accuracy : 0.664
##                  95% CI : (0.66, 0.668)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.285
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.653
##             Specificity : 0.695
##          Pos Pred Value : 0.859
##          Neg Pred Value : 0.414
##              Prevalence : 0.739
##          Detection Rate : 0.483
##    Detection Prevalence : 0.562
##       Balanced Accuracy : 0.674
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25705  4211
```

```
##   hit  13482  9597
##
##                Accuracy : 0.666
##                  95% CI : (0.662, 0.67)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.288
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.656
##             Specificity : 0.695
##          Pos Pred Value : 0.859
##          Neg Pred Value : 0.416
##              Prevalence : 0.739
##          Detection Rate : 0.485
##    Detection Prevalence : 0.565
##       Balanced Accuracy : 0.675
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25674  4280
##   hit  13513  9529
##
##                Accuracy : 0.664
##                  95% CI : (0.66, 0.668)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.284
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.655
##             Specificity : 0.690
##          Pos Pred Value : 0.857
##          Neg Pred Value : 0.414
##              Prevalence : 0.739
##          Detection Rate : 0.484
##    Detection Prevalence : 0.565
##       Balanced Accuracy : 0.673
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 25871  4277
```

```
##   hit  13316  9531
##
##                Accuracy : 0.668
##                  95% CI : (0.664, 0.672)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.289
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.660
##             Specificity : 0.690
##          Pos Pred Value : 0.858
##          Neg Pred Value : 0.417
##              Prevalence : 0.739
##          Detection Rate : 0.488
##    Detection Prevalence : 0.569
##       Balanced Accuracy : 0.675
##
##        'Positive' Class : flop
##
```

```
summary(lr_model_balanced)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.723  -1.042   0.042   1.028   3.488
##
## Coefficients:
##                  Estimate Std. Error z value            Pr(>|z|)
## (Intercept)       2.64962    0.19202   13.80 < 0.0000000000000002 ***
## duration_ms       0.02496    0.00483    5.17        0.00000023367 ***
## explicit1         1.32523    0.02156   61.47 < 0.0000000000000002 ***
## danceability      0.37258    0.00551   67.68 < 0.0000000000000002 ***
## energy           -0.13561    0.00888  -15.27 < 0.0000000000000002 ***
## key1              0.28353    0.01962   14.45 < 0.0000000000000002 ***
## key2             -0.02251    0.01697   -1.33              0.18465
## key3              0.12430    0.02573    4.83        0.00000136467 ***
## key4              0.04514    0.01849    2.44              0.01461 *
## key5              0.04093    0.01825    2.24              0.02496 *
## key6              0.28463    0.02135   13.33 < 0.0000000000000002 ***
## key7              0.00932    0.01649    0.57              0.57169
## key8              0.22498    0.02099   10.72 < 0.0000000000000002 ***
## key9              0.02709    0.01694    1.60              0.10983
## key10             0.13083    0.02038    6.42        0.00000000014 ***
## key11             0.18296    0.01964    9.32 < 0.0000000000000002 ***
## loudness          0.60500    0.00799   75.72 < 0.0000000000000002 ***
## mode1             0.03380    0.00905    3.73              0.00019 ***
## speechiness      -0.24666    0.00649  -38.03 < 0.0000000000000002 ***
## acousticness     -0.26860    0.00614  -43.72 < 0.0000000000000002 ***
```

```
## instrumentalness -0.02826    0.00510   -5.54        0.00000002985 ***
## liveness         -0.08399    0.00452  -18.59 < 0.0000000000000002 ***
## valence          -0.41430    0.00555  -74.61 < 0.0000000000000002 ***
## tempo             0.07137    0.00438   16.28 < 0.0000000000000002 ***
## time_signature1  -3.06213    0.19776  -15.48 < 0.0000000000000002 ***
## time_signature3  -3.10785    0.19211  -16.18 < 0.0000000000000002 ***
## time_signature4  -2.99814    0.19175  -15.64 < 0.0000000000000002 ***
## time_signature5  -2.85705    0.19498  -14.65 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 382858  on 276173  degrees of freedom
## Residual deviance: 336071  on 276146  degrees of freedom
## AIC: 336127
##
## Number of Fisher Scoring iterations: 4
```

```r
lrb_0_0 <- c(25520, 25855, 25876, 25839, 25672, 25691, 25782,
    25639, 25763, 25748)
lrb_0_1 <- c(4202, 4309, 4197, 4231, 4215, 4156, 4281, 4196,
    4267, 4195)
lrb_1_0 <- c(13667, 13332, 13311, 13348, 13515, 13497, 13405,
    13548, 13424, 13439)
lrb_1_1 <- c(9606, 9499, 9612, 9578, 9594, 9652, 9528, 9613,
    9542, 9614)
lrb_conf_matrix <- data.frame(`Predicted 0, Actual 0` = mean(lrb_0_0),
    `Predicted 0, Actual 1` = mean(lrb_0_1), `Predicted 1, Actual 0` = mean(lrb_1_0),
    `Predicted 1, Actual 1` = mean(lrb_1_1))
paste("Type 1 Error:", (lrb_conf_matrix[1, 3]/sum(lrb_conf_matrix[1,
    ])) * 100, "%")
```

```
## [1] "Type 1 Error: 25.3767279671219 %"
```

```r
paste("Type 2 Error:", (lrb_conf_matrix[1, 2]/sum(lrb_conf_matrix[1,
    ])) * 100, "%")
```

```
## [1] "Type 2 Error: 7.97214118854702 %"
```

Using K-fold cross validation on our unbalanced model we obtained a Misclassification Rate of 24.09% Using K-fold cross validation on our balanced model we obtained a Misclassification Rate of 33.33%

In both cases, type 1 error is much greater than type 2 error. It would seem an unbalanced model performs more accurately for our goals.

3.2.2 Logistic Regression Assumptions

Absence of Multicolinearity - Variance Inflation Factor

```r
## VIF
model_fit <- lm(popularity ~ duration_ms + factor(explicit) +
    danceability + energy + factor(key) + loudness + factor(mode) +
    speechiness + acousticness + instrumentalness + liveness +
    valence + tempo + factor(time_signature), data = popular_df2)

vif(model_fit)
```

```
##                         GVIF Df GVIF^(1/(2*Df))
## duration_ms            1.063  1           1.031
## factor(explicit)       1.080  1           1.039
## danceability           1.727  1           1.314
## energy                 4.322  1           2.079
## factor(key)            1.121 11           1.005
## loudness               2.916  1           1.708
## factor(mode)           1.080  1           1.039
## speechiness            1.342  1           1.159
## acousticness           2.142  1           1.464
## instrumentalness       1.218  1           1.104
## liveness               1.138  1           1.067
## valence                1.724  1           1.313
## tempo                  1.110  1           1.054
## factor(time_signature) 1.241  4           1.027
```

Using the VID function we checked for Multicolinearity, no major issues found, the only variable that has a high VIF value is "energy", as its less than 5 no changes are needed.

Lack of Influential Outliers - Cook's Distance

```
# Influential Outliers
popular_df[cooks.distance(model_fit) > 1, ]
```

```
##  [1] id               name             popularity       duration_ms
##  [5] explicit         artists          id_artists       release_date
##  [9] danceability     energy           key              loudness
## [13] mode             speechiness      acousticness     instrumentalness
## [17] liveness         valence          tempo            time_signature
## <0 rows> (or 0-length row.names)
```

Using Cooks Distance we detected no influential outliers, no further action is needed.

3.3 Classification Tree

To compare to the previous classification models, a classification tree was also built with the same predictor variables. As with the other models, training was done with 80% of data, and testing with the remaining 20%.

```
set.seed(1)

N <- nrow(popular_df2)
n <- N * 0.8

idx = sample(1:N, size = n, replace = FALSE)
train = popular_df2[idx, ]
test = popular_df2[-idx, ]
```

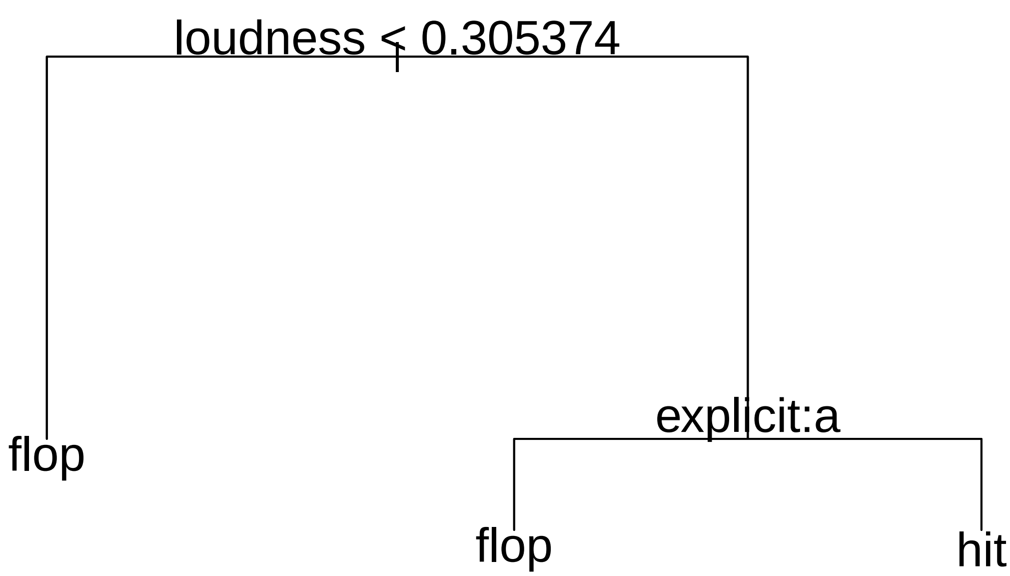
The classification tree was build as shown in the code below:

```
tree.class <- tree(popularity_coded ~ duration_ms + explicit +
    danceability + energy + key + loudness + mode + speechiness +
    acousticness + instrumentalness + liveness + valence + tempo +
    time_signature, data = train)
summary(tree.class)
```

```
##
## Classification tree:
## tree(formula = popularity_coded ~ duration_ms + explicit + danceability +
```

```
##     energy + key + loudness + mode + speechiness + acousticness +
##     instrumentalness + liveness + valence + tempo + time_signature,
##     data = train)
## Variables actually used in tree construction:
## [1] "loudness" "explicit"
## Number of terminal nodes:  3
## Residual mean deviance:  1.07 = 454000 / 424000
## Misclassification error rate: 0.247 = 104805 / 423966
```

```
plot(tree.class)
text(tree.class, cex = 1.5, col = "black")
```



As shown in the plot, the tree produced with default setting was very simple. It contained only 3 terminal nodes and only used "loudness" and "explicit" to perform the classification. The misclassification rate was determined with the 20% test data as shown below:

```
tree.pred <- predict(tree.class, test, type = "class")

tab <- table(tree.pred, test$popularity_coded)
tab
```

```
##
## tree.pred  flop   hit
##      flop 77109 25174
##      hit   1165  2544
```

```
mis = 1 - sum(diag(tab))/sum(tab)
mis
```

```
## [1] 0.2485
```

The misclassification rate was found to be 0.285. Since the tree was already very small, it was not expected that pruning would be necessary. Still, we confirmed this using cross-validation to choose tree complexity. The chart shown below confirms the suspicion that 3 terminal nodes is optimal for this case; reducing terminal nodes would reduce model performance.

```
cv.class = cv.tree(tree.class)
plot(cv.class$size, cv.class$dev, type = "b", cex.main = 1.5,
    cex.lab = 1.5, cex.axis = 1.5)
```

Now that the tree complexity has been determined, we used 10 k fold cross-validation to determine the mean misclassification. To perform this cross-validation, 10 folds were produced which were stratified on "hit" or "no hit" i.e, "popularity_coded".

```
set.seed(10)
strat_folds <- createFolds(factor(popular_df2$popularity_coded),
    k = 10)

for (i in 1:10) {

    idx <- strat_folds[[i]]
    fold <- popular_df2[idx, ]
    # print(table(fold$popularity_coded))

}
```

```r
mis_tree <- function(idx) {
    Train <- popular_df2[-idx, ]
    Test <- popular_df2[idx, ]
    tree.class <- tree(popularity_coded ~ duration_ms + explicit +
        danceability + energy + key + loudness + mode + speechiness +
        acousticness + instrumentalness + liveness + valence +
        tempo + time_signature, data = Train)
    tree_hat <- predict(tree.class, Test, type = "class")

    tab <- table(tree_hat, Test$popularity_coded)
    print(tab)

    misclass = 1 - sum(diag(tab))/sum(tab)
    type_1 = tab[1, 2]/sum(tab)
    type_2 = tab[2, 1]/sum(tab)
    print(misclass)
    print(type_1)
    print(type_2)
    return(misclass)

}

misclass_tree = lapply(strat_folds, mis_tree)
```

```
##
## tree_hat  flop   hit
##     flop 38532 12501
##     hit    655  1308
## [1] 0.2482
## [1] 0.2359
## [1] 0.01236
##
## tree_hat  flop   hit
##     flop 38574 12557
##     hit    613  1251
## [1] 0.2485
## [1] 0.2369
## [1] 0.01157
##
## tree_hat  flop   hit
##     flop 38573 12478
##     hit    614  1331
## [1] 0.247
## [1] 0.2355
## [1] 0.01159
##
## tree_hat  flop   hit
##     flop 38563 12486
##     hit    624  1323
## [1] 0.2474
## [1] 0.2356
## [1] 0.01177
##
## tree_hat  flop   hit
```

```
##     flop 38599 12458
##     hit    588  1350
## [1] 0.2462
## [1] 0.2351
## [1] 0.0111
##
## tree_hat  flop   hit
##     flop 38597 12475
##     hit    590  1334
## [1] 0.2465
## [1] 0.2354
## [1] 0.01113
##
## tree_hat  flop   hit
##     flop 38602 12459
##     hit    585  1350
## [1] 0.2461
## [1] 0.2351
## [1] 0.01104
##
## tree_hat  flop   hit
##     flop 38551 12540
##     hit    636  1268
## [1] 0.2486
## [1] 0.2366
## [1] 0.012
##
## tree_hat  flop   hit
##     flop 38599 12528
##     hit    589  1281
## [1] 0.2475
## [1] 0.2364
## [1] 0.01111
##
## tree_hat  flop   hit
##     flop 38583 12518
##     hit    604  1291
## [1] 0.2476
## [1] 0.2362
## [1] 0.0114
```

```r
paste("Mean Misclassification:", mean(as.numeric(misclass_tree)))
```

```
## [1] "Mean Misclassification: 0.247374322233435"
```

```r
paste("Mean Type 1 Error:", 0.23587)
```

```
## [1] "Mean Type 1 Error: 0.23587"
```

```r
paste("Mean Type 2 Error:", 0.011507)
```

```
## [1] "Mean Type 2 Error: 0.011507"
```

The results of the 10 k-fold cross validation echoed the results of the training set. The mean misclassification was 0.2474. This is a higher misclassification than the the logistic regression model, and therefore the classification tree is not recommended.

```r
tree_model_class <- train(popularity_coded ~ duration_ms + explicit +
    danceability + energy + key + loudness + mode + speechiness +
    acousticness + instrumentalness + liveness + valence + tempo +
    time_signature, data = popular_df2, trControl = ctrl, method = "rpart")
```

```
## Confusion Matrix and Statistics
## 
## 
##         flop hit
##   flop    2   3
##   hit     3   2
## 
##                Accuracy : 0.4
##                  95% CI : (0.122, 0.738)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.828
## 
##                   Kappa : -0.2
## 
##  Mcnemar's Test P-Value : 1.000
## 
##             Sensitivity : 0.4
##             Specificity : 0.4
##          Pos Pred Value : 0.4
##          Neg Pred Value : 0.4
##              Prevalence : 0.5
##          Detection Rate : 0.2
##    Detection Prevalence : 0.5
##       Balanced Accuracy : 0.4
## 
##        'Positive' Class : flop
## 
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. Sensitivity will be used instead.
```

```
## Confusion Matrix and Statistics
## 
## 
##          flop   hit
##   flop 37842 11428
##   hit   1345  2381
## 
##                Accuracy : 0.759
##                  95% CI : (0.755, 0.763)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
## 
##                   Kappa : 0.181
## 
##  Mcnemar's Test P-Value : <0.0000000000000002
## 
##             Sensitivity : 0.966
##             Specificity : 0.172
##          Pos Pred Value : 0.768
```

```
##          Neg Pred Value : 0.639
##              Prevalence : 0.739
##          Detection Rate : 0.714
##    Detection Prevalence : 0.930
##       Balanced Accuracy : 0.569
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38314 12161
##   hit    873  1648
##
##                Accuracy : 0.754
##                  95% CI : (0.75, 0.758)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.00000000000000621
##
##                   Kappa : 0.132
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.978
##             Specificity : 0.119
##          Pos Pred Value : 0.759
##          Neg Pred Value : 0.654
##              Prevalence : 0.739
##          Detection Rate : 0.723
##    Detection Prevalence : 0.952
##       Balanced Accuracy : 0.549
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 39187 13809
##   hit      0     0
##
##                Accuracy : 0.739
##                  95% CI : (0.736, 0.743)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.502
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.739
```

```
##          Neg Pred Value :   NaN
##              Prevalence : 0.739
##          Detection Rate : 0.739
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38349 12153
##   hit    838  1656
##
##                Accuracy : 0.755
##                  95% CI : (0.751, 0.759)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.134
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.979
##             Specificity : 0.120
##          Pos Pred Value : 0.759
##          Neg Pred Value : 0.664
##              Prevalence : 0.739
##          Detection Rate : 0.724
##    Detection Prevalence : 0.953
##       Balanced Accuracy : 0.549
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38619 12514
##   hit    568  1295
##
##                Accuracy : 0.753
##                  95% CI : (0.749, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.00000000000024
##
##                   Kappa : 0.11
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9855
##             Specificity : 0.0938
##          Pos Pred Value : 0.7553
```

```
##          Neg Pred Value : 0.6951
##              Prevalence : 0.7394
##          Detection Rate : 0.7287
##    Detection Prevalence : 0.9648
##       Balanced Accuracy : 0.5396
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 39187 13809
##   hit      0     0
##
##                Accuracy : 0.739
##                  95% CI : (0.736, 0.743)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.502
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.739
##          Neg Pred Value :   NaN
##              Prevalence : 0.739
##          Detection Rate : 0.739
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 37245 10800
##   hit   1942  3009
##
##                Accuracy : 0.76
##                  95% CI : (0.756, 0.763)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.212
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.950
##             Specificity : 0.218
##          Pos Pred Value : 0.775
```

```
##          Neg Pred Value : 0.608
##              Prevalence : 0.739
##          Detection Rate : 0.703
##    Detection Prevalence : 0.907
##       Balanced Accuracy : 0.584
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38588 12487
##   hit    599  1322
##
##                Accuracy : 0.753
##                  95% CI : (0.749, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.000000000000322
##
##                   Kappa : 0.112
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9847
##             Specificity : 0.0957
##          Pos Pred Value : 0.7555
##          Neg Pred Value : 0.6882
##              Prevalence : 0.7394
##          Detection Rate : 0.7281
##    Detection Prevalence : 0.9638
##       Balanced Accuracy : 0.5402
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38588 12487
##   hit    599  1322
##
##                Accuracy : 0.753
##                  95% CI : (0.749, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.000000000000322
##
##                   Kappa : 0.112
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9847
##             Specificity : 0.0957
##          Pos Pred Value : 0.7555
```

```
##          Neg Pred Value : 0.6882
##              Prevalence : 0.7394
##          Detection Rate : 0.7281
##    Detection Prevalence : 0.9638
##       Balanced Accuracy : 0.5402
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 37457 10994
##   hit   1731  2815
##
##                Accuracy : 0.76
##                  95% CI : (0.756, 0.764)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.204
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.956
##             Specificity : 0.204
##          Pos Pred Value : 0.773
##          Neg Pred Value : 0.619
##              Prevalence : 0.739
##          Detection Rate : 0.707
##    Detection Prevalence : 0.914
##       Balanced Accuracy : 0.580
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38619 12419
##   hit    569  1390
##
##                Accuracy : 0.755
##                  95% CI : (0.751, 0.759)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.119
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.985
##             Specificity : 0.101
##          Pos Pred Value : 0.757
```

```
##          Neg Pred Value : 0.710
##              Prevalence : 0.739
##          Detection Rate : 0.729
##    Detection Prevalence : 0.963
##       Balanced Accuracy : 0.543
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 39188 13809
##   hit      0     0
##
##                Accuracy : 0.739
##                  95% CI : (0.736, 0.743)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.502
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.739
##          Neg Pred Value :   NaN
##              Prevalence : 0.739
##          Detection Rate : 0.739
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38264 12162
##   hit    923  1646
##
##                Accuracy : 0.753
##                  95% CI : (0.749, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.000000000000322
##
##                   Kappa : 0.13
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.976
##             Specificity : 0.119
##          Pos Pred Value : 0.759
```

```
##          Neg Pred Value : 0.641
##              Prevalence : 0.739
##          Detection Rate : 0.722
##    Detection Prevalence : 0.952
##       Balanced Accuracy : 0.548
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38264 12162
##   hit    923  1646
##
##                Accuracy : 0.753
##                  95% CI : (0.749, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.000000000000322
##
##                   Kappa : 0.13
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.976
##             Specificity : 0.119
##          Pos Pred Value : 0.759
##          Neg Pred Value : 0.641
##              Prevalence : 0.739
##          Detection Rate : 0.722
##    Detection Prevalence : 0.952
##       Balanced Accuracy : 0.548
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38573 12527
##   hit    614  1281
##
##                Accuracy : 0.752
##                  95% CI : (0.748, 0.756)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000167
##
##                   Kappa : 0.107
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9843
##             Specificity : 0.0928
##          Pos Pred Value : 0.7549
```

```
##          Neg Pred Value : 0.6760
##              Prevalence : 0.7394
##          Detection Rate : 0.7279
##    Detection Prevalence : 0.9642
##       Balanced Accuracy : 0.5386
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38313 12179
##   hit    874  1629
##
##                Accuracy : 0.754
##                  95% CI : (0.75, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000000292
##
##                   Kappa : 0.13
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.978
##             Specificity : 0.118
##          Pos Pred Value : 0.759
##          Neg Pred Value : 0.651
##              Prevalence : 0.739
##          Detection Rate : 0.723
##    Detection Prevalence : 0.953
##       Balanced Accuracy : 0.548
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38313 12179
##   hit    874  1629
##
##                Accuracy : 0.754
##                  95% CI : (0.75, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000000292
##
##                   Kappa : 0.13
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.978
##             Specificity : 0.118
##          Pos Pred Value : 0.759
```

```
##          Neg Pred Value : 0.651
##              Prevalence : 0.739
##          Detection Rate : 0.723
##    Detection Prevalence : 0.953
##       Balanced Accuracy : 0.548
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38579 12531
##   hit    608  1277
##
##                Accuracy : 0.752
##                  95% CI : (0.748, 0.756)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000146
##
##                   Kappa : 0.107
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9845
##             Specificity : 0.0925
##          Pos Pred Value : 0.7548
##          Neg Pred Value : 0.6775
##              Prevalence : 0.7394
##          Detection Rate : 0.7280
##    Detection Prevalence : 0.9644
##       Balanced Accuracy : 0.5385
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 37440 11152
##   hit   1747  2656
##
##                Accuracy : 0.757
##                  95% CI : (0.753, 0.76)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.19
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.955
##             Specificity : 0.192
##          Pos Pred Value : 0.770
```

```
##          Neg Pred Value : 0.603
##              Prevalence : 0.739
##          Detection Rate : 0.706
##    Detection Prevalence : 0.917
##       Balanced Accuracy : 0.574
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38569 12523
##   hit    618  1285
##
##                Accuracy : 0.752
##                  95% CI : (0.748, 0.756)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000167
##
##                   Kappa : 0.107
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9842
##             Specificity : 0.0931
##          Pos Pred Value : 0.7549
##          Neg Pred Value : 0.6752
##              Prevalence : 0.7394
##          Detection Rate : 0.7278
##    Detection Prevalence : 0.9641
##       Balanced Accuracy : 0.5386
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38569 12523
##   hit    618  1285
##
##                Accuracy : 0.752
##                  95% CI : (0.748, 0.756)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000167
##
##                   Kappa : 0.107
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9842
##             Specificity : 0.0931
##          Pos Pred Value : 0.7549
```

```
##          Neg Pred Value : 0.6752
##              Prevalence : 0.7394
##          Detection Rate : 0.7278
##    Detection Prevalence : 0.9641
##       Balanced Accuracy : 0.5386
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38348 12228
##   hit    839  1581
##
##                Accuracy : 0.753
##                  95% CI : (0.75, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000000785
##
##                   Kappa : 0.127
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.979
##             Specificity : 0.114
##          Pos Pred Value : 0.758
##          Neg Pred Value : 0.653
##              Prevalence : 0.739
##          Detection Rate : 0.724
##    Detection Prevalence : 0.954
##       Balanced Accuracy : 0.547
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38578 12609
##   hit    609  1200
##
##                Accuracy : 0.751
##                  95% CI : (0.747, 0.754)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.00000000217
##
##                   Kappa : 0.099
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9845
##             Specificity : 0.0869
##          Pos Pred Value : 0.7537
```

```
##          Neg Pred Value : 0.6633
##              Prevalence : 0.7394
##          Detection Rate : 0.7279
##    Detection Prevalence : 0.9659
##       Balanced Accuracy : 0.5357
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38578 12609
##   hit    609  1200
##
##                Accuracy : 0.751
##                  95% CI : (0.747, 0.754)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.00000000217
##
##                   Kappa : 0.099
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9845
##             Specificity : 0.0869
##          Pos Pred Value : 0.7537
##          Neg Pred Value : 0.6633
##              Prevalence : 0.7394
##          Detection Rate : 0.7279
##    Detection Prevalence : 0.9659
##       Balanced Accuracy : 0.5357
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 37390 10997
##   hit   1797  2812
##
##                Accuracy : 0.759
##                  95% CI : (0.755, 0.762)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.201
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.954
##             Specificity : 0.204
##          Pos Pred Value : 0.773
```

```
##          Neg Pred Value : 0.610
##              Prevalence : 0.739
##          Detection Rate : 0.706
##    Detection Prevalence : 0.913
##       Balanced Accuracy : 0.579
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38602 12569
##   hit    585  1240
##
##                Accuracy : 0.752
##                  95% CI : (0.748, 0.755)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000375
##
##                   Kappa : 0.104
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9851
##             Specificity : 0.0898
##          Pos Pred Value : 0.7544
##          Neg Pred Value : 0.6795
##              Prevalence : 0.7394
##          Detection Rate : 0.7284
##    Detection Prevalence : 0.9656
##       Balanced Accuracy : 0.5374
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38602 12569
##   hit    585  1240
##
##                Accuracy : 0.752
##                  95% CI : (0.748, 0.755)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000375
##
##                   Kappa : 0.104
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9851
##             Specificity : 0.0898
##          Pos Pred Value : 0.7544
```

```
##          Neg Pred Value : 0.6795
##              Prevalence : 0.7394
##          Detection Rate : 0.7284
##    Detection Prevalence : 0.9656
##       Balanced Accuracy : 0.5374
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38277 12137
##   hit    910  1672
##
##                Accuracy : 0.754
##                  95% CI : (0.75, 0.757)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000000171
##
##                   Kappa : 0.133
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.977
##             Specificity : 0.121
##          Pos Pred Value : 0.759
##          Neg Pred Value : 0.648
##              Prevalence : 0.739
##          Detection Rate : 0.722
##    Detection Prevalence : 0.951
##       Balanced Accuracy : 0.549
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38548 12492
##   hit    639  1317
##
##                Accuracy : 0.752
##                  95% CI : (0.749, 0.756)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000079
##
##                   Kappa : 0.11
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9837
##             Specificity : 0.0954
##          Pos Pred Value : 0.7553
```

```
##          Neg Pred Value : 0.6733
##              Prevalence : 0.7394
##          Detection Rate : 0.7274
##    Detection Prevalence : 0.9631
##       Balanced Accuracy : 0.5395
##
##        'Positive' Class : flop
##
## Confusion Matrix and Statistics
##
##
##         flop   hit
##   flop 38548 12492
##   hit    639  1317
##
##                Accuracy : 0.752
##                  95% CI : (0.749, 0.756)
##     No Information Rate : 0.739
##     P-Value [Acc > NIR] : 0.0000000000079
##
##                   Kappa : 0.11
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.9837
##             Specificity : 0.0954
##          Pos Pred Value : 0.7553
##          Neg Pred Value : 0.6733
##              Prevalence : 0.7394
##          Detection Rate : 0.7274
##    Detection Prevalence : 0.9631
##       Balanced Accuracy : 0.5395
##
##        'Positive' Class : flop
##
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
print(tree_model_class)
```

```
## CART
##
## 529958 samples
##     14 predictor
##      2 classes: 'flop', 'hit'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476962, 476962, 476962, 476961, 476963, 476963, ...
## Resampling results across tuning parameters:
##
##   cp        Sensitivity  Specificity  Pos Pred Value  Neg Pred Value  Precision
##   0.002342  0.9670       0.15828      0.7654          0.6336          0.7654
##   0.006242  0.9825       0.10118      0.7562          0.6729          0.7562
```

```
##   0.024959  0.9891       0.06461      0.7502          0.6761          0.7502
##   Recall  F1      Prevalence  Detection Rate  Detection Prevalence
##   0.9670  0.8544  0.7394      0.7150          0.9343
##   0.9825  0.8546  0.7394      0.7265          0.9607
##   0.9891  0.8532  0.7394      0.7314          0.9751
##   Balanced Accuracy
##   0.5626
##   0.5418
##   0.5269
##
## Sensitivity was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.02496.
```

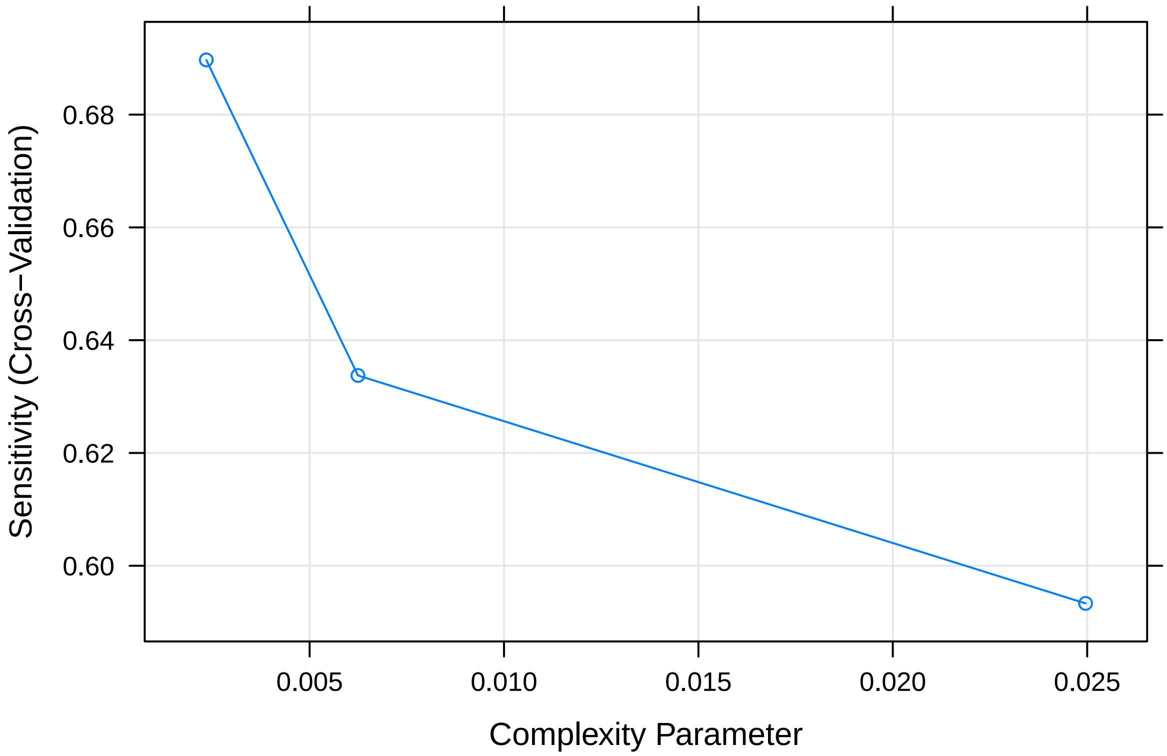
3.3.1 Balanced Classification Tree

Using the same logic as in the Logistic Regression, we decided to Balance our Tree.

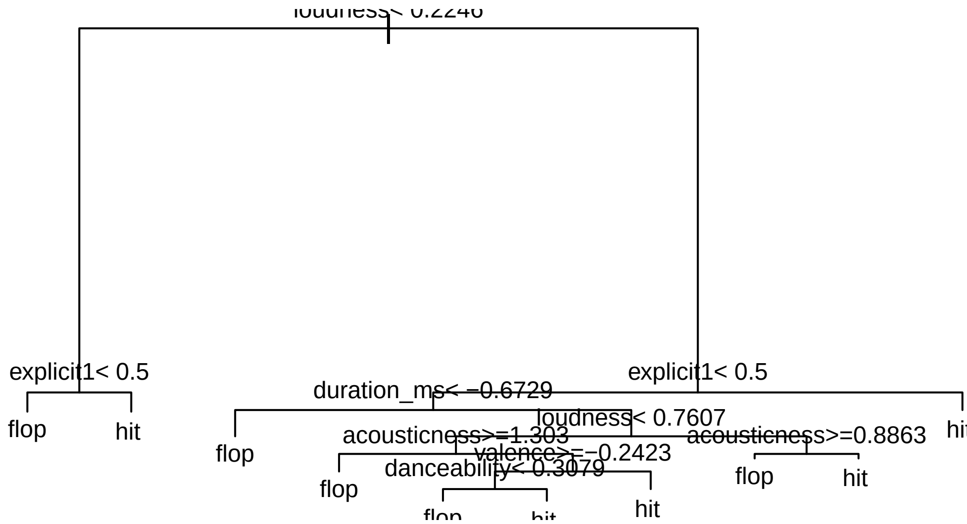
```
print(tree_model_class_balanced)
```

```
## CART
##
## 529958 samples
##     14 predictor
##      2 classes: 'flop', 'hit'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476963, 476962, 476961, 476962, 476962, 476962, ...
## Addtional sampling using down-sampling
##
## Resampling results across tuning parameters:
##
##   cp        Sensitivity  Specificity  Pos Pred Value  Neg Pred Value  Precision
##   0.002342  0.6897       0.6413       0.8452          0.4215          0.8452
##   0.006242  0.6337       0.6816       0.8497          0.3963          0.8497
##   0.024959  0.5933       0.6873       0.8434          0.3733          0.8434
##   Recall  F1      Prevalence  Detection Rate  Detection Prevalence
##   0.6897  0.7595  0.7394      0.5100          0.6035
##   0.6337  0.7258  0.7394      0.4686          0.5516
##   0.5933  0.6965  0.7394      0.4387          0.5202
##   Balanced Accuracy
##   0.6655
##   0.6577
##   0.6403
##
## Sensitivity was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002342.
```

```
plot(tree_model_class_balanced, what = "scree")
```




```
plot(tree_model_class_balanced$finalModel)
text(tree_model_class_balanced$finalModel, cex = 0.75)
```



This tree was pruned in the cross-validation according to sensitivity.

```
ctb_0_0 <- c(26717, 24768, 23068, 27574, 24605, 23072, 26974,
    24531, 22840, 27526, 25851)
ctb_0_1 <- c(5017, 4501, 4347, 5249, 4401, 4372, 4820, 4236,
    4133, 5114, 4643)
ctb_1_0 <- c(12470, 14419, 16119, 11613, 14582, 16115, 12213,
    14656, 16347, 11662, 13337)
ctb_1_1 <- c(8792, 9308, 9462, 8560, 9408, 9437, 8989, 9573,
    9676, 8695, 9166)
ctb_conf_matrix <- data.frame(`Predicted 0, Actual 0` = mean(ctb_0_0),
    `Predicted 0, Actual 1` = mean(ctb_0_1), `Predicted 1, Actual 0` = mean(ctb_1_0),
    `Predicted 1, Actual 1` = mean(ctb_1_1))
paste("Type 1 Error:", (ctb_conf_matrix[1, 3]/sum(ctb_conf_matrix[1,
    ])) * 100, "%")
```

```
## [1] "Type 1 Error: 26.3368887638561 %"
```

```
paste("Type 2 Error:", (ctb_conf_matrix[1, 2]/sum(ctb_conf_matrix[1,
    ])) * 100, "%")
```

```
## [1] "Type 2 Error: 8.71983916508565 %"
```

The balanced tree produced a misclassification rate of 35.06%, again as with Logistic regression, our model was not helped by balancing the training classes.

3.3.2 Classification Tree Assumptions

For tree methods, we do not have any assumptions to test besides our observations being independent of each other, which they are.

4.4 Regression Tree

Is a supervised learning statistical model where the target variable can take continuous values. We use the model to predict the popularity score for each of the songs, using all other variables as predictors, first we divide the data into training and testing using simple random sampling, we used 80% of the data as training and 20% as testing.

A regression Tree of 6 terminal nodes was created and using the variables loudness, acousticness, explicit and duration_ms as predictors.

```
popularity_tree_fit <- tree(popularity ~ duration_ms + explicit +
    danceability + energy + key + loudness + mode + speechiness +
    acousticness + instrumentalness + liveness + valence + tempo +
    time_signature, data = train)

summary(popularity_tree_fit)
```

```
##
## Regression tree:
## tree(formula = popularity ~ duration_ms + explicit + danceability +
##     energy + key + loudness + mode + speechiness + acousticness +
##     instrumentalness + liveness + valence + tempo + time_signature,
##     data = train)
## Variables actually used in tree construction:
## [1] "loudness"     "acousticness" "explicit"     "duration_ms"
## Number of terminal nodes:  6
## Residual mean deviance:  0.852 = 361000 / 424000
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -2.730  -0.694  -0.043   0.000   0.609   4.180
```

The RMSE obtained for our Regression Tree is 0.9239. RMSE is in the same units as our target value, in this case we have a very big RMSE if we compared to 0.00000000000000005539.

```
popularity_tree_predict <- predict(popularity_tree_fit, test)
print("RMSE:")
```

```
## [1] "RMSE:"
```

```
print(sqrt(mean((popularity_tree_predict - test$popularity)^2)))
```

```
## [1] 0.9239
```

```
print("Mean")
```

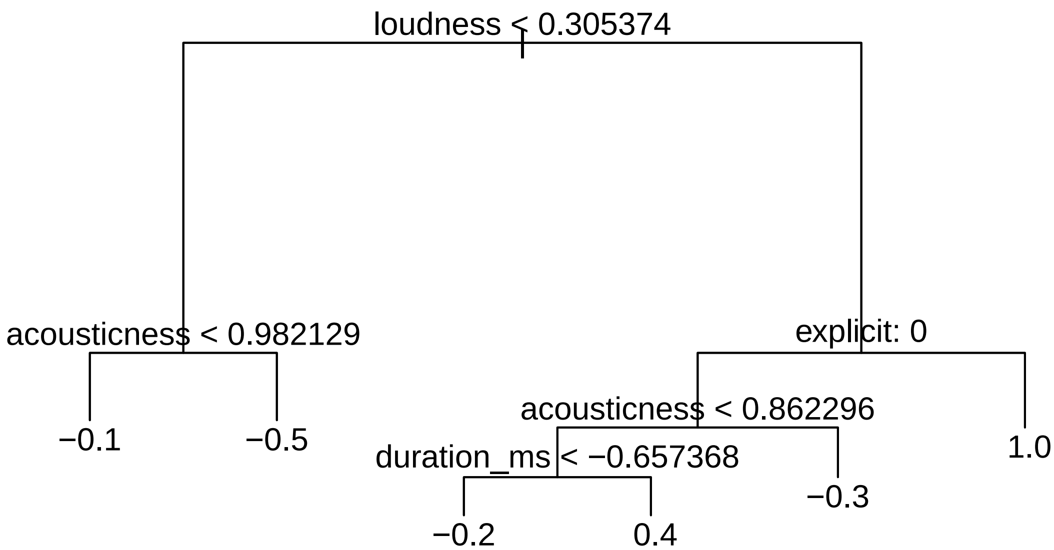
```
## [1] "Mean"
```

```
print(mean(popular_df2$popularity))
```

```
## [1] 0.00000000000000005539
```

Here below we can observe the regression tree plotted.

```
plot(popularity_tree_fit)
text(popularity_tree_fit, pretty = 0)
```



To see if tree pruning is needed, we plot the the cross-validation error and the size of the tree, we concluded that 6 terminal nodes is the best performing tree so no pruning is needed.

```
cv.popularity = cv.tree(popularity_tree_fit, K = 10)
plot(cv.popularity$size, cv.popularity$dev, type = "b")
```

Lastly, we did k-fold cross validation to see if we see an improvement. The RMSE obtained is very similar to the previous value, meaning that this model is not very accurate to obtain the real value of popularity.

```
ctrl <- trainControl(method = "cv", number = 10)

tree_model <- train(popularity ~ duration_ms + explicit + danceability +
    energy + key + loudness + mode + speechiness + acousticness +
    instrumentalness + liveness + valence + tempo + time_signature,
    data = popular_df, trControl = ctrl, method = "rpart")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
print(tree_model)
```

```
## CART
##
## 529958 samples
##     14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476963, 476962, 476962, 476963, 476962, 476962, ...
## Resampling results across tuning parameters:
##
##   cp       RMSE   Rsquared  MAE
##   0.01833  15.87  0.11562   12.91
##   0.02033  16.05  0.09629   13.08
```

```
##   0.08470  16.37  0.08361   13.39
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.01833.
```

3.4.1 Assumptions

For tree methods, we do not have any assumptions to test besides our observations being independant of each other, which they are.

4 Summary of Findings

Linear Discriminant Analysis to predict song popularity scores. While it showed equally promising misclassification rates to the Classification Tree and Logistic Regression, we discarded it due to failing to meet the Multivariate Normality and Equality of Variances assumptions.

Turning to a Regression Tree Model to predict popularity scores, but it was only capable of predicting scores of 20, 30, 40, and 50 with an RMSE of 15. Our analysis had a defined "hit" point of popularity score > 42, which was the 75th percentile, so we had hoped to see more terminal nodes above that range to feel that this model accomplished the goals of our investigation. Though we did balance the training data for the classification models, we did not for regression models. Looking back, we think that balancing for the regression models would have been beneficial. If we want a model that can predict popularity from 0-100, having more equality across that spectrum in the training data would likely have improved performance for our goals.

The classification models, Logistic Regression and Classification Tree, obtained similar misclassification results. Initially we trained the models with 10-fold cross validation on the unbalanced dataset, and we observed misclassification rates of 24.74% for the Classification Tree and 24.09% for Logistic Regression.

As we defined our distinction between a "hit" and "no-hit" to be the 75th percentile, we inherently created unbalanced classes from which to train our classification models. As only 25% of songs were considered "hits". To correct this, we implemented both up and down sampling techniques into our custom "TrainControl()" functions. As both yielded near identical results, we used only down sampling for the report. Once these classification models were performed on the balanced training folds, Classification Tree and Logistic Regression saw increased misclassification rates to 35.06% and 33.33%, respectively.

Balancing provided some interesting insight: with the unbalanced set, we saw that three of our methods produced misclassification rates near the proportion of hits in the dataset, 25%. Following balancing our classes in the training models, we saw an overall increase in misclassification rate, with an increase in both type 1 and type 2 error. As we used the caret library to conduct this cross-validation we were unable to determine if the method we used caused both training and testing to be balanced, or just training. Perhaps if only the training data was down-sampled and predicted on a test set that was heavily skewed to no-hits the worsening of the model could be explained.

In this investigation, a false positive should be considered worse than a false negative. As a false positive could cause a streaming service, or record label to invest in a song that is not going to be a hit. Alternatively, not investing money in a song could mean missed opportunity but does not directly cause loss of capital. So in the case of both classification models, type 1 error was worsened.

Balancing the training data for the Classification Tree improved the number of features used in modeling beyond just loudness and explicitness, and we increased from 2 to 9 internal nodes. However, it had poorer performance overall.

It's important to note that this dataset is produced by Spotify, with variables that it creates through its own process. As such, it is a descriptive dataset and not optimized to predict popularity with variables designed to cluster songs into playlists or reflect user preferences. We must remember that popularity is not just a product of plays, it is assigned by Spoitfy's algorithm. It's worth considering how much influence investment, business agreements/relationships, and artist image have on song ratings.

5 Conclusion

After evaluating the models, we found that Logistic Regression gave us the most confidence as a model. Through this model, we drew some interesting conclusions about the features of music that are related to its popularity. We found that explicitness, danceability, and loudness are the features that contribute most strongly and positively to a song being a hit, while speechiness and acousticness are the most strongly negative contributors to a song's "hit" status.

Though we believe it was the technical correct choice to balance the class data for our classification models, the outcome of worsening our model predictive performance was not expected.

Though this dataset is not ideal for the goals of this project, and at this level of analysis is likely not capable of informing music industry professionals on how to create or find the next hit, we did glean some interesting insights into the features of songs that succeed on Spotify. Perhaps the most important is not what makes a hit, but what features songs as less popular. It seems that today slow acoustic songs with too much talking or 'liveness' just aren't that popular. While high energy dance songs with explicitness seem to get more people interested.

Of course, music is often more a by-product of current social trends, so it's likely that a more robust analysis would include some form of trend analysis, both within the music itself, and in the greater social world that the music comes from. It's likely that this would not be so easily quantified and modeled, so this is a good example of a problem that still requires a significant degree of experience and domain knowledge to combine with the quantitative analytic results of these models.

6 References

1. https://investors.spotify.com/about/default.aspx
2. https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features
3. https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks
4. https://cdla.dev/sharing-1-0/
5. S. (n.d.). *Get-several-audio-features*. Spotify. https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features
6. Brownlee, J. (2020, July 31). *Cross-validation-for-imbalanced-classification*. Machine Learning Mastery. https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/

7 Appendix

7.1 Genre Classification

One future work item we considered in our project work and mentioned in our class presentation was trying to classify songs into genres based on their other classification attributes. The section below provides the code to prepare the data for genre-focused analysis and run classification methods.

In the code chunks below, the "genre" data which exists in the "artists" data needs to be merged with the "track" data so each track has a genre classification.

The difficulty with this dataset is that the "track" data did not have it's own genre. It is possible for an artist to produce songs in different genres, and since we did not have data at the track level, the decision was made to filter out artists who fall into multiple genres. For example, an artist who produces rock and rap will not be included in this data. This introduces some bias to the data, and for future work it would be optimal to find a data source which has genres already assigned on a track level, instead of relying on the genre of the artist.

```r
# clean the artists to those with only one genre and of the
# primary genre types

artist_genres <- artists_df %>%
    filter(genres != "[]") %>%
    mutate(genres = toupper(genres), rock = grepl("ROCK", genres,
        fixed = TRUE), rnb = grepl("R&B", genres, fixed = TRUE),
        pop = grepl("POP", genres, fixed = TRUE), country = grepl("COUNTRY",
            genres, fixed = TRUE), rap = grepl("RAP", genres,
            fixed = TRUE), jazz = grepl("JAZZ", genres, fixed = TRUE),
        classical = grepl("CLASSICAL", genres, fixed = TRUE),
        soul = grepl("SOUL", genres, fixed = TRUE), funk = grepl("FUNK",
            genres, fixed = TRUE), electronic = grepl("ELECTRONIC",
            genres, fixed = TRUE), disco = grepl("DISCO", genres,
            fixed = TRUE), num_genres = rock + rnb + pop + country +
            rap + jazz + classical + soul + funk + electronic +
            disco) %>%
    filter(num_genres == 1) %>%
    mutate(genre = case_when(rock == 1 ~ "rock", rnb == 1 ~ "rnb",
        pop == 1 ~ "pop", country == 1 ~ "country", rap == 1 ~
            "rap", jazz == 1 ~ "jazz", classical == 1 ~ "classical",
        soul == 1 ~ "soul", funk == 1 ~ "funk", electronic ==
            1 ~ "electronic", disco == 1 ~ "disco")) %>%
    dplyr::select(id, genre, name, popularity)
```

" After preparing the "artists" data as shown above, the next step is to clean up the "tracks" data so that the artist column can be read as a list.

```r
# clean the track artist lists
rep_str = c(`\\[` = "", `\\]` = "", `'` = "")

track_artists <- tracks_df
track_artists$id_artists <- str_replace_all(track_artists$id_artists,
    rep_str)
track_artists$id_artists <- as.list(track_artists$id_artists)
```

Finally, the artist data is merged with the tracks data. This is an inner join, resulting in tracks that only have one genre assigned.

```r
# inner join so only the artists with single genre remain
# alongsige the track data
genre_tracks <- merge(x = artist_genres, y = track_artists, by.x = "id",
    by.y = "id_artists") %>%
    dplyr::select(id:time_signature)

colnames(genre_tracks)
```

```
##  [1] "id"               "genre"            "name.x"           "popularity.x"
##  [5] "id.y"             "name.y"           "popularity.y"     "duration_ms"
##  [9] "explicit"         "artists"          "release_date"     "danceability"
## [13] "energy"           "key"              "loudness"         "mode"
## [17] "speechiness"      "acousticness"     "instrumentalness" "liveness"
## [21] "valence"          "tempo"            "time_signature"
```

```r
colnames(genre_tracks) <- c("artist_id", "genre", "artist_name",
    "artist_popularity", "song_id", "song_name", "song_popularity",
```

```
    "duration_ms", "explicit", "artists", "release_date", "danceability",
    "energy", "key", "loudness", "mode", "speechiness", "acousticness",
    "instrumentalness", "liveness", "valence", "tempo", "time_signature")
```

Next, the data is grouped by genre as a stata to facilitate splitting for train and test. 150000 observations were used for training to roughly target 75% (actually 73%). Below, the count of the strata can be seen.

```
# count the strata to understand what should be used to
# test/train
n = 150000
N = dim(genre_tracks)[1]

order <- unique(genre_tracks$genre)
order
```

```
##  [1] "pop"        "rock"       "disco"      "jazz"       "rap"
##  [6] "country"    "electronic" "rnb"        "soul"       "classical"
## [11] "funk"
```

```
strata <- genre_tracks %>%
    count(genre) %>%
    rename(all = n) %>%
    mutate(train = round(all * n/N)) %>%
    slice(match(order, genre))
strata
```

```
##         genre    all train
## 1         pop 112838 81023
## 2        rock  60051 43119
## 3       disco   1026   737
## 4        jazz  16592 11914
## 5         rap   8347  5994
## 6     country   3913  2810
## 7  electronic    985   707
## 8         rnb    336   241
## 9        soul   1502  1079
## 10  classical   2612  1876
## 11       funk    698   501
```

```
strata_sizes <- strata$train
```

The data is split into test and train:

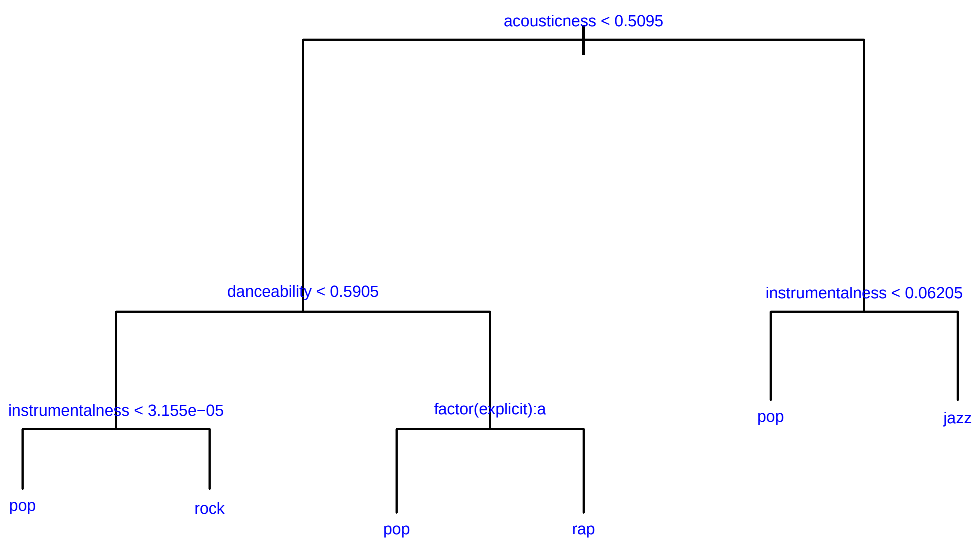
```
# perform the stratified sample
idx_strat <- sampling:::strata(genre_tracks, stratanames = c("genre"),
    size = strata_sizes, method = "srswor")
train_genre <- genre_tracks[idx_strat$ID_unit, ]
test_genre <- genre_tracks[-idx_strat$ID_unit, ]
```

Now the classification tree can be built with the training data:

```
# build tree genre
tree.genre <- tree(factor(genre) ~ duration_ms + factor(explicit) +
    danceability + energy + factor(key) + loudness + factor(mode) +
    speechiness + acousticness + instrumentalness + liveness +
    valence + tempo, train_genre)
summary(tree.genre)
```

```
##
## Classification tree:
## tree(formula = factor(genre) ~ duration_ms + factor(explicit) +
##     danceability + energy + factor(key) + loudness + factor(mode) +
##     speechiness + acousticness + instrumentalness + liveness +
##     valence + tempo, data = train_genre)
## Variables actually used in tree construction:
## [1] "acousticness"     "danceability"     "instrumentalness" "factor(explicit)"
## Number of terminal nodes:  6
## Residual mean deviance:  2.2 = 330000 / 150000
## Misclassification error rate: 0.391 = 58613 / 150001
```

```r
plot(tree.genre)
text(tree.genre, cex = 0.5, col = "blue")
```



The mislclassification rate for the tree on the test data is found as shown below:

```r
# test the tree misclass

tree.pred <- predict(tree.genre, test_genre, type = "class")

tab <- table(tree.pred, test_genre$genre)
tab
```

```
##
## tree.pred    classical country disco electronic  funk  jazz   pop   rap   rnb
##   classical          0       0     0          0     0     0     0     0     0
##   country            0       0     0          0     0     0     0     0     0
```

```
##   disco              0       0      0           0      0      0      0      0      0
##   electronic         0       0      0           0      0      0      0      0      0
##   funk               0       0      0           0      0      0      0      0      0
##   jazz             422      97      4          49      2   1742   1234     17      1
##   pop              212     954    259         159    165   2644  27625   1444     89
##   rap                0       1      1           2      2      1    383    786      2
##   rnb                0       0      0           0      0      0      0      0      0
##   rock             102      51     25          68     28    291   2573    106      3
##   soul               0       0      0           0      0      0      0      0      0
##
## tree.pred     rock  soul
##   classical      0     0
##   country        0     0
##   disco          0     0
##   electronic     0     0
##   funk           0     0
##   jazz         640    17
##   pop        10302   350
##   rap          157     6
##   rnb            0     0
##   rock        5833    50
##   soul           0     0
```

```
mis = 1 - sum(diag(tab))/sum(tab)
mis
```

```
## [1] 0.389
```

The performance of the tree is 0.4181 misclassification. It can be seen that the tree does not preduct many of the genres well. Genres such as "blues", "classicial" and country have no correct predictions. This could be related to the imbalanced dataset (high pop), or difficulty in drawing clear lines between the genres in general.

Although the assumptions for LDA were not met, as seen in the main body of the report, the LDA exercise was performed for completenes:

```
# build lda genre
lda.genre <- lda(factor(genre) ~ duration_ms + factor(explicit) +
    danceability + energy + factor(key) + loudness + factor(mode) +
    speechiness + acousticness + instrumentalness + liveness +
    valence + tempo, train_genre)
lda.genre
```

```
## Call:
## lda(factor(genre) ~ duration_ms + factor(explicit) + danceability +
##     energy + factor(key) + loudness + factor(mode) + speechiness +
##     acousticness + instrumentalness + liveness + valence + tempo,
##     data = train_genre)
##
## Prior probabilities of groups:
##   classical    country      disco electronic       funk       jazz        pop
##    0.012507   0.018733   0.004913   0.004713   0.003340   0.079426   0.540150
##         rap        rnb       rock       soul
##    0.039960   0.001607   0.287458   0.007193
##
## Group means:
##            duration_ms factor(explicit)1 danceability energy factor(key)1
```

```
## classical       257482         0.0005330       0.4118 0.3566      0.07516
## country         186637         0.0035587       0.5868 0.4549      0.05231
## disco           284307         0.0054274       0.7092 0.7177      0.05563
## electronic      294009         0.0240453       0.5816 0.6117      0.10891
## funk            253605         0.0459082       0.6617 0.7258      0.09581
## jazz            231901         0.0002518       0.5338 0.3670      0.05624
## pop             233022         0.0187478       0.5733 0.5669      0.06603
## rap             217864         0.4804805       0.7171 0.6841      0.14748
## rnb             264475         0.0373444       0.6385 0.5478      0.10788
## rock            242414         0.0306361       0.5168 0.6503      0.05506
## soul            218247         0.0213160       0.5904 0.5201      0.06673
##            factor(key)2 factor(key)3 factor(key)4 factor(key)5 factor(key)6
## classical       0.11194      0.07196      0.07463      0.08742      0.04904
## country         0.09858      0.04484      0.09359      0.10142      0.04199
## disco           0.08684      0.03664      0.05427      0.10583      0.06784
## electronic      0.06789      0.02970      0.09052      0.08204      0.07072
## funk            0.07385      0.02595      0.05389      0.06786      0.08184
## jazz            0.08855      0.05833      0.03995      0.16678      0.02694
## pop             0.11092      0.03700      0.08478      0.09155      0.05434
## rap             0.07441      0.02636      0.06557      0.07090      0.09226
## rnb             0.08299      0.04564      0.07054      0.07884      0.09544
## rock            0.13713      0.02217      0.10886      0.06134      0.04694
## soul            0.07785      0.03244      0.07600      0.11121      0.04727
##            factor(key)7 factor(key)8 factor(key)9 factor(key)10 factor(key)11
## classical       0.13006      0.06450      0.09595       0.06983       0.04371
## country         0.13737      0.06441      0.12028       0.09004       0.05125
## disco           0.10855      0.05563      0.12076       0.05834       0.08412
## electronic      0.11033      0.07214      0.10891       0.06365       0.08204
## funk            0.11976      0.06188      0.13772       0.07385       0.10778
## jazz            0.13564      0.09384      0.06857       0.08981       0.02644
## pop             0.12327      0.05477      0.11003       0.06269       0.07088
## rap             0.09510      0.08058      0.07641       0.08625       0.10244
## rnb             0.09959      0.11203      0.05809       0.04564       0.08299
## rock            0.13224      0.03664      0.15687       0.03850       0.07723
## soul            0.13994      0.06766      0.10380       0.08990       0.06673
##            loudness factor(mode)1 speechiness acousticness instrumentalness
## classical   -14.708        0.6269     0.07034       0.7456          0.42790
## country     -10.395        0.9060     0.06973       0.6478          0.04560
## disco        -9.108        0.5577     0.05965       0.1283          0.15703
## electronic  -11.063        0.5120     0.06382       0.2411          0.55977
## funk         -7.474        0.5908     0.11735       0.2926          0.12488
## jazz        -12.751        0.6358     0.06296       0.7636          0.26842
## pop          -8.966        0.6372     0.06285       0.4273          0.03884
## rap          -6.915        0.5163     0.19438       0.2242          0.01472
## rnb          -8.094        0.6763     0.05918       0.3284          0.01596
## rock         -9.208        0.6833     0.06656       0.2514          0.08487
## soul         -9.969        0.7118     0.06297       0.4193          0.04255
##            liveness valence tempo
## classical    0.2012  0.3785 112.3
## country      0.2052  0.6441 119.5
## disco        0.1815  0.7555 123.6
## electronic   0.1805  0.3912 121.9
## funk         0.3176  0.6476 127.4
## jazz         0.1928  0.5435 113.5
```

```
## pop          0.1996  0.5534 120.4
## rap          0.1963  0.5567 117.0
## rnb          0.1763  0.4950 115.4
## rock         0.2279  0.5401 123.8
## soul         0.1927  0.6243 117.8
##
## Coefficients of linear discriminants:
##                             LD1            LD2           LD3          LD4
## duration_ms       -0.0000002138  0.00000003761  0.0000006729 -0.000002834
## factor(explicit)1  3.8826418847  1.90868792802  1.5116843726  1.800943643
## danceability       1.8591524116  2.41219028442 -1.9610631932 -4.982847118
## energy            -0.0491795698 -1.65399930748  1.7982404590 -0.362623538
## factor(key)1       0.2126974473  0.23566349899 -0.0106707529  0.018191992
## factor(key)2       0.0891003041 -0.24310050854  0.0744728200  0.372680595
## factor(key)3       0.0221069062  0.26800182551  0.0279545151 -0.014344288
## factor(key)4       0.1462314750 -0.44347866258  0.0538918693  0.552084399
## factor(key)5      -0.0993172279  0.36879810632 -0.0324533454  0.111438572
## factor(key)6       0.2699110193 -0.05923779848 -0.0901194648 -0.003286987
## factor(key)7       0.0382367142 -0.04891038399  0.0770396312  0.299404990
## factor(key)8       0.0016336106  0.48721493788 -0.0001303109 -0.006536640
## factor(key)9       0.0747134510 -0.42157561055  0.1538608231  0.507277495
## factor(key)10      0.0736197287  0.35300286886 -0.1123223811  0.219393761
## factor(key)11      0.2169472523 -0.25080139841 -0.0764810757  0.258017758
## loudness           0.0394377080  0.07179123031 -0.1408638387 -0.042391061
## factor(mode)1     -0.0460755256 -0.26512912676 -0.0057939709  0.714809532
## speechiness        3.6897041851  1.75655388519  1.9451557769  2.034902265
## acousticness      -1.0934962512  2.15021710332 -0.8672544131  0.490439316
## instrumentalness  -1.1718826343  1.60361439242  3.3244460180 -1.744013833
## liveness          -0.0389625230 -0.11452591513 -0.1169121341  0.237343999
## valence           -0.9531389975  0.17305439695 -0.2331455944  1.923619830
## tempo             -0.0002047278 -0.00080910906 -0.0024610068 -0.004058750
##                             LD5          LD6          LD7          LD8
## duration_ms       -0.00000006209 -0.000002714 -0.000001225 -0.000002935
## factor(explicit)1 -0.22773162358 -1.187785430  0.731268288  0.183958154
## danceability       2.00945673954  2.352944294 -0.036572986 -0.935528994
## energy            -0.83663238109  0.455050390 -3.760971196 -3.278297803
## factor(key)1      -0.12093951350  1.063201092  0.274332242 -0.514968382
## factor(key)2      -0.06541156426  0.509788196 -0.027163798 -0.133566119
## factor(key)3       0.04955702285  0.493537131  0.124798296 -1.689814601
## factor(key)4       0.20880423190  1.532410979  0.695952156 -0.370757785
## factor(key)5       0.47351960321 -0.157215677  0.218663754  1.209250466
## factor(key)6       0.03975515452  1.427237540  0.273295568 -1.521818434
## factor(key)7       0.17019110518  0.543670405  0.102621587  0.228489961
## factor(key)8       0.35348406420  0.213132084  0.370505609  0.822219950
## factor(key)9       0.38828755468  1.134891428 -0.019716838 -0.085373756
## factor(key)10      0.48732348186  0.890916401  0.349068062 -0.066650424
## factor(key)11      0.21753708124  1.372306439  0.190141067 -0.738291504
## loudness          -0.05432684330  0.068220361  0.029681526  0.186169113
## factor(mode)1      0.73281495779  1.255221875  0.599846892 -0.200185414
## speechiness       -0.15920439713  1.753500723 -3.052665976 -0.765273617
## acousticness      -1.01655032216  0.756483983 -2.240084373 -1.185926829
## instrumentalness   0.76409697418  1.350582094  0.268257382  1.350001942
## liveness           1.03952525809  0.934925227 -2.641726005  2.061197482
## valence            2.88179892674 -2.190744905 -0.079190686 -0.004087116
```

```
## tempo               0.00252017100  0.007257494 -0.004894375  0.004174459
##                              LD9          LD10
## duration_ms         0.000002185 -0.000004313
## factor(explicit)1 -1.164648556 -0.861340631
## danceability      -0.940091322 -0.617194541
## energy            -4.123411543 -0.279459665
## factor(key)1       1.313297082  0.841800016
## factor(key)2      -0.232953125 -0.519140042
## factor(key)3       0.737328961 -0.208820090
## factor(key)4      -0.159253600  0.323770337
## factor(key)5       0.029535332 -0.001682381
## factor(key)6       1.308526031  0.202746690
## factor(key)7       0.396252358  0.667293996
## factor(key)8       1.414973368 -1.095111875
## factor(key)9       0.029726432  0.301256812
## factor(key)10      0.674617498  2.221389945
## factor(key)11      0.829583977  0.749686597
## loudness           0.051283174 -0.029586698
## factor(mode)1      0.046166250 -0.916525107
## speechiness        3.361996341  1.362113653
## acousticness      -1.926835347 -0.561357489
## instrumentalness  -0.398202797  0.052301666
## liveness           2.691246927 -0.402535022
## valence            0.588278690  0.328940527
## tempo             -0.007902902  0.000657561
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5    LD6    LD7    LD8    LD9   LD10
## 0.4649 0.3066 0.1681 0.0323 0.0155 0.0065 0.0035 0.0015 0.0008 0.0003
```

The misclassification for the LDA was 0.3894. This is in the same range as the tree; both models requires more work to improve the result.

```
# test the lda misclass

lda.predict <- predict(lda.genre, test_genre)
table <- table(lda.predict$class, test_genre$genre)
table
```

```
##
##                classical country disco electronic  funk  jazz   pop   rap   rnb
##   classical          233       4     0         23     1   369   195     1     0
##   country              0       0     0          0     0     0     0     0     0
##   disco                0       0     2          0     0     2    34     0     0
##   electronic          35       1    28        117    18   111   317     9     1
##   funk                 0       0     0          0     0     0     3     0     0
##   jazz               134      49     2          8     0   996   626     3     1
##   pop                246     959   201         76   148  3018 26819  1161    87
##   rap                  6      13     1          4     8    22   737  1093     5
##   rnb                  0       0     0          0     0     0     0     0     0
##   rock                82      77    55         50    22   160  3084    86     1
##   soul                 0       0     0          0     0     0     0     0     0
##
##                 rock  soul
##   classical      170     2
```

```
##   country        0     0
##   disco          3     0
##   electronic   425     9
##   funk           0     0
##   jazz         194     8
##   pop         8932   346
##   rap          671     9
##   rnb            0     0
##   rock        6537    49
##   soul           0     0
```

```r
misclass = 1 - sum(diag(table))/sum(table)
misclass
```

```
## [1] 0.3922
```