What Makes a Rock Song Award-Winning?

Cristian Granchelli, Joan Orellana Rios, Cameron Kelahan

Can attributes and metrics of rock songs be used to predict their likelihood of winning the Grammy for "Best Rock Song?"

What common traits, if any, do award winning rock songs contain? Can intrinsic traits of songs, combined with metrics defined by Spotify, be looked at to determine award winning musical features? In this paper, we break down our data collection, data processing, and data analysis of a dataset containing roughly 1,000 popular songs, both award-winning and not.

The Data

Data Collection and Preparation

In order to investigate this, a list of every song nominated for the "Best Rock Song" Grammy was collected. The reason for keeping the scope of this project within one award, rather than include other rock-related Grammy's ("Best Rock Performance," "Best Rock Album," etc.), is because they are disparate by nature, and nominations are awarded for completely different reasons. The Best Rock Song Grammy competition is between individual songs, allowing us to compare them to each other and determine what attributes mold a work worthy of a nomination.

In order to examine what sets these nominations apart from other rock songs, the top 10 most streamed songs of each of these previously nominated artists on Spotify were collected so that the features and attributes of their popular non-nominated songs could be compared to songs that earned a nomination. This created a dataset of 1,016 popular rock songs. As a final step of processing the data, any songs that were released prior to the creation of this Grammy in 1992 were removed. Excluding all songs older than 1992 yielded a final dataset of 867 songs.

```r
# Set the working directory to this file's folder
library("rstudioapi")
setwd(dirname(getActiveDocumentContext()$path))
load("final_df_n_str.RData")

Sys.setenv(LANG = "en")

# Load necessary libraries
library(pROC)
library(MASS)
library(ROSE)
library(confintr)
library(ggplot2)
library(correlation)
library(corrplot)
library(class)
library(caret)
library(glmnet)
```

```r
# Selecting the relevant variables
data = final_df_n_str
data = data[,c("track_name", "artist_name", "IsWinner", "Year","year",
               "followers", "acousticness", "danceability", "duration_ms",
               "energy", "instrumentalness", "key", "liveness", "loudness",
               "mode", "tempo", "time_signature", "valence")]

# Merge the two year variable
data$Year[data$Year == "Undefined"] <- data$year[data$Year == "Undefined"]
data = data[,c("track_name","artist_name", "IsWinner", "Year", "followers",
               "acousticness", "danceability", "duration_ms",
               "energy", "instrumentalness", "key", "liveness", "loudness",
               "mode", "tempo", "time_signature", "valence")]

# Eliminating duplicates
data$track_name == "Closing Time"
data$track_name == "Smells Like Teen Spirit"
data$track_name == "Don't Wanna Fight"
data[914, ]
data[789,]
data[669,]

data = data[-c(669, 789, 914),]

sum(data$Year < 1992)
nrow(data)
data = data[!data$Year < 1992,]

# Creating row names

names = paste0(data$track_name, " - ", data$artist_name)

# Eliminating unusable variables
data = data[,c("IsWinner", "Year", "acousticness",
               "danceability", "duration_ms", "energy",
               "instrumentalness", "key", "loudness", "mode",
               "tempo", "time_signature", "valence")]
data = cbind(names = names, data)

# Casting variables
data$IsWinner[data$IsWinner == "Winner"] = 1
data$IsWinner[data$IsWinner == "Nominee"] = 1
data$IsWinner[data$IsWinner == "Nothing"] = 0
data$IsWinner = as.integer(data$IsWinner)
data$Year = as.integer(data$Year)
data$mode = as.factor(data$mode)
data$key = as.factor(data$key)
data$time_signature = as.factor(data$time_signature)

summary(data)
```

```r
# Checking balance between classes

# Non-Grammy nominated songs
```

```
length(data$IsWinner[data$IsWinner == 0]) /(length(data$IsWinner[data$IsWinner == 0]) +
                                            length(data$IsWinner[data$IsWinner == 1]))
```

```
## [1] 0.822376
```

```
# Grammy nominated songs
length(data$IsWinner[data$IsWinner == 1]) / (length(data$IsWinner[data$IsWinner == 0]) +
                                             length(data$IsWinner[data$IsWinner == 1]))
```

```
## [1] 0.177624
```

From the balance check between the number of nominated and non-nominated songs, it is seen that the classes are highly imbalanced: 82% non-nominated songs, 18% nominated songs. A note of this is made during the data processing stage so that measures may be taken to address this during the analysis.

Explanation of Variables

Metrics that are intrinsic to music as well as artificial metrics created and measured by the music streaming giant Spotify were collected in order to perform analysis of the songs. The intrinsic metrics used were: duration, musical key, modality/mode (major or minor key), tempo, and time signature. Spotify also uses what they call "audio features" (in the table below) to perform their own analysis of songs when creating playlists, suggesting music, etc. These professionally manufactured metrics were used to bolster the intrinsic metrics and increase the insight into what might make a song award-winning.

Audio Feature	Definition
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
Danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
Energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
Instrumentalness	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

Audio Feature	Definition
Loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.
Valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Although Spotify does not openly share the formulas behind how they determine these metrics, they were found suitable to assist in this analysis.

As a final processing step, the data was split into training and test datasets. The training dataset contains 80% of the original dataset, and the remaining 20% of the data in the test dataset will be used to test the models against after they have been trained. It is very important to test the model on never-before-seen data to determine not only how well the model performs, but also how well the model can generalize.

```r
# Splitting training and test set
training_size = floor(0.8 * nrow(data))
set.seed(42)
train_ind = sample(seq_len(nrow(data)), size = training_size)
training_set = data[train_ind,]
test_set = data[-train_ind,]

summary(training_set)
```

```
##     names              IsWinner           Year       acousticness
##  Length:693         Min.   :0.0000   Min.   :1992   Min.   :0.0000032
##  Class :character   1st Qu.:0.0000   1st Qu.:2001   1st Qu.:0.0016900
##  Mode  :character   Median :0.0000   Median :2010   Median :0.0278000
##                     Mean   :0.1876   Mean   :2009   Mean   :0.1553733
##                     3rd Qu.:0.0000   3rd Qu.:2018   3rd Qu.:0.2050000
##                     Max.   :1.0000   Max.   :2023   Max.   :0.9880000
##
##   danceability    duration_ms         energy       instrumentalness
##  Min.   :0.130   Min.   :  78591   Min.   :0.0975   Min.   :0.00e+00
##  1st Qu.:0.419   1st Qu.: 206413   1st Qu.:0.6040   1st Qu.:4.90e-06
##  Median :0.522   Median : 237800   Median :0.7570   Median :3.21e-04
##  Mean   :0.512   Mean   : 251635   Mean   :0.7182   Mean   :6.25e-02
##  3rd Qu.:0.607   3rd Qu.: 278267   3rd Qu.:0.8820   3rd Qu.:1.49e-02
##  Max.   :0.894   Max.   :1355938   Max.   :0.9960   Max.   :8.95e-01
##
##       key         loudness        mode         tempo        time_signature
##  9      : 95   Min.   :-18.148   0:203   Min.   : 48.58   1:  2
##  2      : 94   1st Qu.: -8.086   1:490   1st Qu.: 99.19   3: 37
##  7      : 84   Median : -6.253           Median :121.14   4:649
##  0      : 81   Mean   : -6.645           Mean   :123.28   5:  5
##  11     : 68   3rd Qu.: -4.767           3rd Qu.:141.93
##  4      : 58   Max.   : -1.574           Max.   :205.85
##  (Other):213
##     valence
##  Min.   :0.0494
##  1st Qu.:0.3050
##  Median :0.4640
##  Mean   :0.4725
##  3rd Qu.:0.6310
##  Max.   :0.9730
##
```

```
# Checking if the ratio is preserved

# Ratio of training set:
# - not-nominated
length(training_set$IsWinner[data$IsWinner == 0]) /
      (length(training_set$IsWinner[data$IsWinner == 0]) +
       length(training_set$IsWinner[data$IsWinner == 1]))
```

```
## [1] 0.822376
```

```
# - nominated
length(training_set$IsWinner[data$IsWinner == 1]) /
      (length(training_set$IsWinner[data$IsWinner == 0]) +
       length(training_set$IsWinner[data$IsWinner == 1]))
```

```
## [1] 0.177624
```

```
# Ratio of Test Set
# - not-nominated
length(test_set$IsWinner[data$IsWinner == 0]) /
      (length(test_set$IsWinner[data$IsWinner == 0]) +
       length(test_set$IsWinner[data$IsWinner == 1]))
```

```
## [1] 0.822376
```

```
# - nominated
length(test_set$IsWinner[data$IsWinner == 1]) /
      (length(test_set$IsWinner[data$IsWinner == 0]) +
       length(test_set$IsWinner[data$IsWinner == 1]))
```

```
## [1] 0.177624
```

Exploratory Data Analysis

Relationship Between Independent Variables

At first, correlations between the independent variables were examined, which is the measurement of association between two variables. This is essentially a measurement of similarity between the information learned from different variables. The larger the correlation between independent variables (negative or positive), the higher their collinearity, meaning the information they each carry is similar. This means their use together is less beneficial due to redundancy and should avoid the pairing. It is preferable to utilize pairings of variables that offer different information to assist the model.

It was important to understand the data better in these regards, so that it can be known which attribute pairings could give useful insight.

Correlation Between Continuous Variables

```r
attach(training_set)
# Correlations between continuous variables
cor_matrix = cor(training_set[,c(-1, -2, -9, -11, -13)])
```

```r
# Plot the correlation
corrplot.mixed(cor_matrix, tl.pos='lt')
```

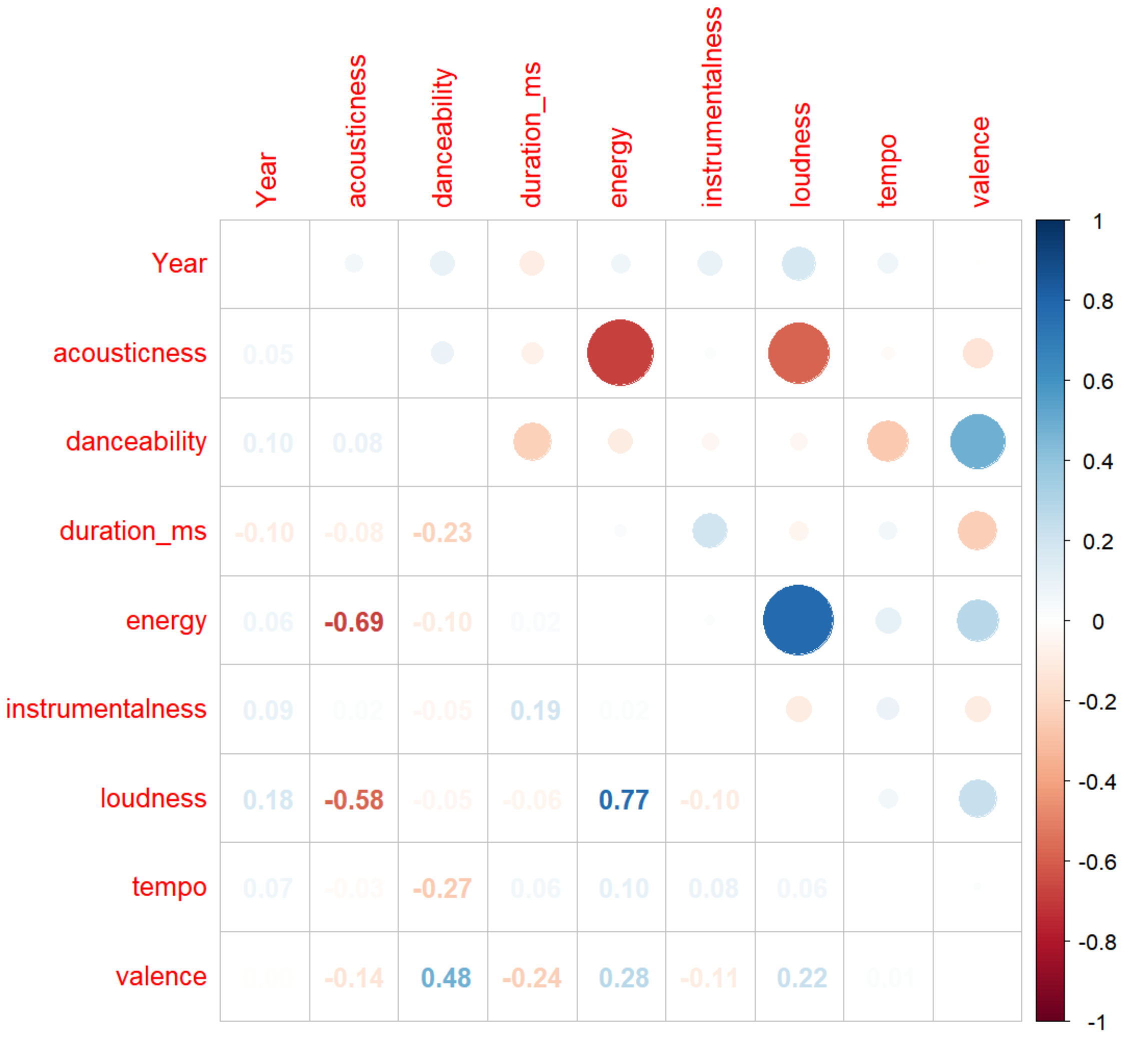

Figure 1: Correlation Matrix

The above plot visualizes the correlations between the continuous, independent variables. There are four pairings of variables that produce a large correlation: acousticness/energy, acousticness/loudness, danceability/valence, and loudness/energy. It was interesting to see that loudness and energy had such a high correlation, giving us some insight into how these metrics are defined.

```
pairs(training_set[,c(-1, -2, -9, -11, -13)], lower.panel = panel.smooth)
```

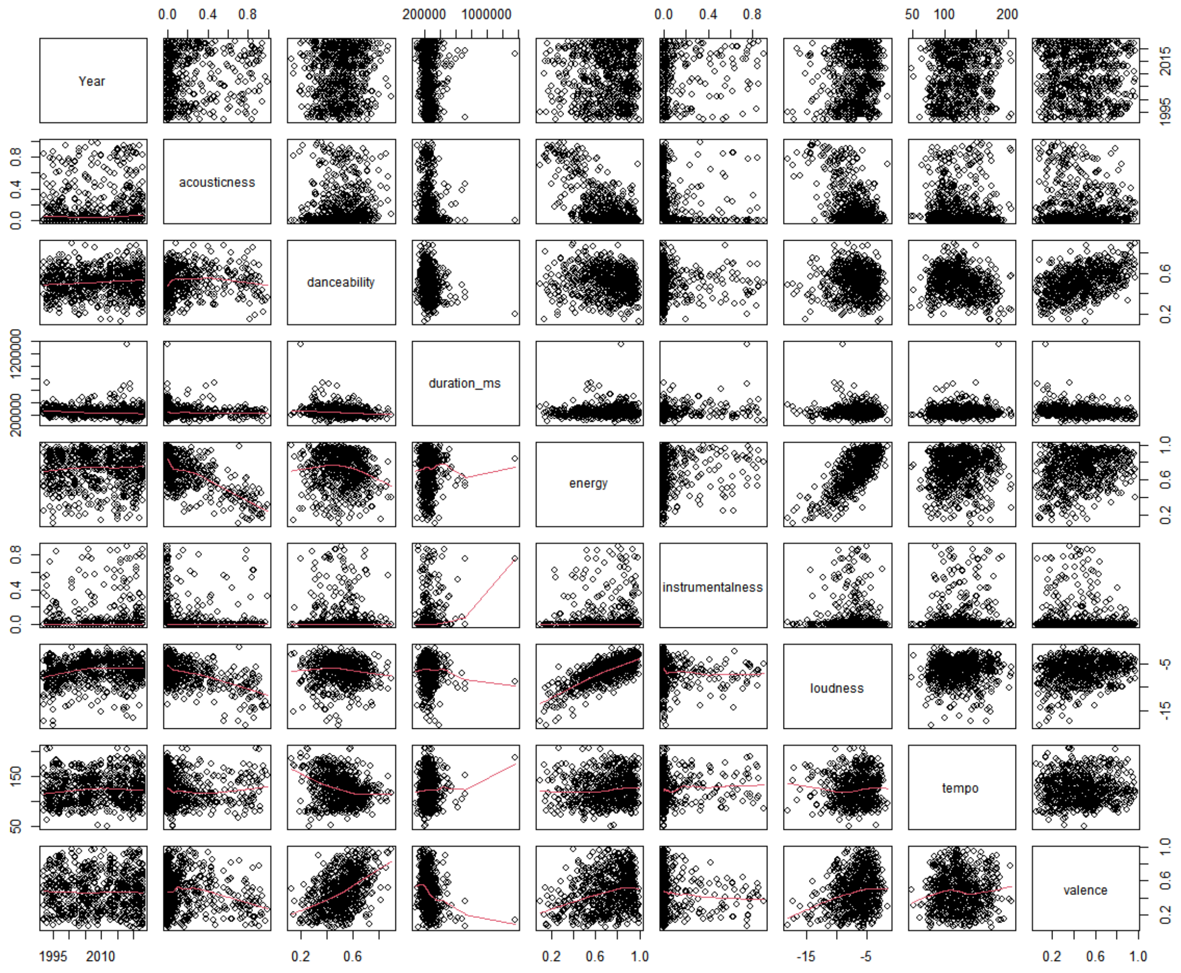

Figure 2: Pairs Correlation Plot with Line Fit

From evaluating this second type of plot showing the correlation between continuous, independent variables, one can understand the shape of the data and discover outliers as well as general trends. For example, almost the entire trend line of duration vs instrumentalness is caused by a single outlier. It also confirmed the high, positive correlation between loudness and energy, showing a nice upward direction of the trend line with the data points clustered around it. Furthermore, it showed the strong trend line between valence and danceability, although those data have higher variability and are less centered around the line. The highly negative correlation between acousticness and energy is also visualized, touting a firmly downward trend line.

Correlation Between Categorical Variables

The Cramer's V measurement was employed in order to determine the strength of similarity between the categorical variables. This measurement is a normalized version of the chi-square statistic, returning a value of 0 if there is no association between the categorical variables, and returning 1 if there is a perfect association in which one variable is completely determined by the other.

```
# Association measure for categorical variables (Cramer's V is a normalized
# version of the chi-square statistics)

# Key / Mode
cramersv(matrix(c(as.numeric(key), as.numeric(mode)), ncol = 2))
```

```
## [1] 0.3275984
```

```
# Key / Time Signature
cramersv(matrix(c(as.numeric(key), as.numeric(time_signature)), ncol = 2))
```

```
## [1] 0.305952
```

```
# Mode / Time Signature
cramersv(matrix(c(as.numeric(mode), as.numeric(time_signature)), ncol = 2))
```

```
## [1] 0.1425218
```

Although measurements of association between categorical variables are hard to interpret, the Cramer's V test still illuminates if there is full disassociation or full association between said variables. From the data, it can be seen that key/mode and key/time_signature are definitely not fully disassociated or fully associated. The same can be said about mode/time_signature, although the measured value was closer to 0.

ANOVA Test

Next, associations between each of the categorical variables and all of the continuous variables were examined utilizing ANOVA. ANOVA (or ANalysis Of VAriation) offers a way to compare continuous and categorical variables. If ANOVA points out some feature combinations are significant, then that suggests the statistical means of the groups of continuous variables are different. The chosen threshold for significance was 5%.

ANOVA makes some assumptions regarding the variances of the attributes which are not fully satisfied by all of the variables, meaning the obtained results should be considered carefully.

```
# Association between continuous and categorical variables

# Key
aco_key.aov <- aov(acousticness ~ key)
summary(aco_key.aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11   0.76 0.06888   1.154  0.316
## Residuals   681  40.65 0.05969
```

```
dan_key.aov <- aov(danceability ~ key)
summary(dan_key.aov) # SIGNIFICANT
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11  0.368 0.03343   1.911 0.0351 *
## Residuals   681 11.913 0.01749
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dur_key.aov <- aov(duration_ms ~ key)
summary(dur_key.aov)
```

```
##              Df    Sum Sq   Mean Sq F value Pr(>F)
## key          11 1.302e+11 1.184e+10   1.671 0.0758 .
## Residuals   681 4.825e+12 7.086e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ene_key.aov <- aov(energy ~ key)
summary(ene_key.aov) # SIGNIFICANT
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11  0.806 0.07330   1.926 0.0334 *
## Residuals   681 25.922 0.03806
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ins_key.aov <- aov(instrumentalness ~ key)
summary(ins_key.aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11  0.503 0.04574   1.696 0.0701 .
## Residuals   681 18.362 0.02696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
loud_key.aov <- aov(loudness ~ key)
summary(loud_key.aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11    109   9.871   1.467  0.139
## Residuals   681   4583   6.730
```

```
tem_key.aov <- aov(tempo ~ key)
summary(tem_key.aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11  12755  1159.5   1.384  0.176
## Residuals   681 570619   837.9
```

```
val_key.aov <- aov(valence ~ key)
summary(val_key.aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## key          11   0.77 0.07024   1.486  0.132
## Residuals   681  32.18 0.04725
```

```
# Mode
aco_mode.aov <- aov(acousticness ~ mode)
summary(aco_mode.aov) # SIGNIFICANT
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## mode          1   0.32  0.3200   5.382 0.0206 *
## Residuals   691  41.09  0.0595
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dan_mode.aov <- aov(danceability ~ mode)
summary(dan_mode.aov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## mode          1  0.012 0.01171    0.66  0.417
## Residuals   691 12.269 0.01775
```

```
dur_mode.aov <- aov(duration_ms ~ mode)
summary(dur_mode.aov)
```

```
##               Df    Sum Sq   Mean Sq F value Pr(>F)
## mode           1 9.401e+07 9.401e+07   0.013  0.909
## Residuals    691 4.955e+12 7.171e+09
```

```
ene_mode.aov <- aov(energy ~ mode)
summary(ene_mode.aov) # SIGNIFICANT
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)
## mode           1  0.374  0.3737   9.798 0.00182 **
## Residuals    691 26.354  0.0381
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ins_mode.aov <- aov(instrumentalness ~ mode)
summary(ins_mode.aov)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## mode           1  0.017 0.01729   0.634  0.426
## Residuals    691 18.848 0.02728
```

```
loud_mode.aov <- aov(loudness ~ mode)
summary(loud_mode.aov) # SIGNIFICANT
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)
## mode           1     59   59.19   8.828 0.00307 **
## Residuals    691   4633    6.70
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tem_mode.aov <- aov(tempo ~ mode)
summary(tem_mode.aov)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## mode           1    928   928.2   1.101  0.294
## Residuals    691 582445   842.9
```

```
val_mode.aov <- aov(valence ~ mode)
summary(val_mode.aov)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## mode           1   0.00 0.00049    0.01   0.92
## Residuals    691  32.95 0.04769
```

```
# Time signature
aco_time.aov <- aov(acousticness ~ time_signature)
summary(aco_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value   Pr(>F)
## time_signature    3   1.27  0.4238   7.275 8.28e-05 ***
## Residuals       689  40.14  0.0583
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dan_time.aov <- aov(danceability ~ time_signature)
summary(dan_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value  Pr(>F)
## time_signature    3  0.239 0.07964   4.557 0.00359 **
## Residuals      689 12.042 0.01748
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dur_time.aov <- aov(duration_ms ~ time_signature)
summary(dur_time.aov) # SIGNIFICANT
```

```
##                  Df    Sum Sq   Mean Sq F value   Pr(>F)
## time_signature   3 1.156e+11 3.855e+10   5.488 0.000993 ***
## Residuals      689 4.840e+12 7.024e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ene_time.aov <- aov(energy ~ time_signature)
summary(ene_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value   Pr(>F)
## time_signature   3   0.74 0.24678   6.543 0.000229 ***
## Residuals      689  25.99 0.03772
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ins_time.aov <- aov(instrumentalness ~ time_signature)
summary(ins_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value  Pr(>F)
## time_signature   3   0.34 0.11337   4.217 0.00573 **
## Residuals      689  18.52 0.02689
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
loud_time.aov <- aov(loudness ~ time_signature)
summary(loud_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value  Pr(>F)
## time_signature   3     78  25.887   3.865 0.00927 **
## Residuals      689   4614   6.697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tem_time.aov <- aov(tempo ~ time_signature)
summary(tem_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value Pr(>F)
## time_signature   3   7794  2598.1    3.11 0.0259 *
## Residuals      689 575579   835.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
val_time.aov <- aov(valence ~ time_signature)
summary(val_time.aov) # SIGNIFICANT
```

```
##                  Df Sum Sq Mean Sq F value Pr(>F)
## time_signature   3   0.52 0.17279   3.671 0.0121 *
## Residuals      689  32.43 0.04707
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The table below lists the categorical/continuous variable combinations that may be significant, according to ANOVA:

Categorical	Continuous
Key	Danceability; Energy
Mode	Acousticness; Energy; Loudness
Time signature	Acousticness; Danceability; Duration; Energy; Instrumentalness; Loudness; Tempo; Valence

The fact that these groups were significant suggests there may be a strong enough correlation between these pairings to use them in the models in a beneficial manner.

Interestingly, the time signature variable shows significance in combination with all other variables. This may be due to the fact that this variable is highly imbalanced, since 93.4% of the songs have a time signature of 4/4.

Partial Correlations

Because ANOVA does not convey which of the means were different from the others, it is necessary to perform a Multiple Comparison Procedure. Partial correlations was utilized in this case, which is an exhaustive search that compares each possible pairing of continuous variables.

```r
# Partial correlations
correlation(training_set[,c(-1, -2, -9, -11, -13)], partial = TRUE)
```

```
## # Correlation Matrix (pearson-method)
##
## Parameter1       |       Parameter2 |         r |         95% CI | t(691) |         p
## ------------------------------------------------------------------------------------
## Year             |     acousticness |      0.15 | [ 0.08,  0.22] |   4.06 | 0.001**
## Year             |     danceability |      0.15 | [ 0.07,  0.22] |   3.85 | 0.003**
## Year             |      duration_ms |     -0.08 | [-0.15, -0.01] |  -2.14 | 0.524
## Year             |           energy |     -0.02 | [-0.10,  0.05] |  -0.62 | > .999
## Year             | instrumentalness |      0.13 | [ 0.06,  0.20] |   3.51 | 0.010**
## Year             |         loudness |      0.24 | [ 0.16,  0.31] |   6.40 | < .001***
## Year             |            tempo |      0.10 | [ 0.02,  0.17] |   2.57 | 0.186
## Year             |          valence |     -0.11 | [-0.19, -0.04] |  -3.00 | 0.056
## acousticness     |     danceability |     -0.04 | [-0.11,  0.04] |  -1.03 | > .999
## acousticness     |      duration_ms |     -0.08 | [-0.15,  0.00] |  -2.06 | 0.572
## acousticness     |           energy |     -0.44 | [-0.50, -0.38] | -13.01 | < .001***
## acousticness     | instrumentalness |      0.02 | [-0.05,  0.10] |   0.60 | > .999
## acousticness     |         loudness |     -0.14 | [-0.21, -0.07] |  -3.78 | 0.004**
## acousticness     |            tempo |      0.03 | [-0.04,  0.11] |   0.85 | > .999
## acousticness     |          valence |      0.07 | [ 0.00,  0.15] |   1.94 | 0.688
## danceability     |      duration_ms |     -0.10 | [-0.17, -0.03] |  -2.65 | 0.157
## danceability     |           energy |     -0.17 | [-0.24, -0.10] |  -4.63 | < .001***
## danceability     | instrumentalness |      0.05 | [-0.03,  0.12] |   1.29 | > .999
## danceability     |         loudness | -2.81e-03 | [-0.08,  0.07] |  -0.07 | > .999
## danceability     |            tempo |     -0.30 | [-0.37, -0.23] |  -8.39 | < .001***
## danceability     |          valence |      0.53 | [ 0.47,  0.58] |  16.45 | < .001***
## duration_ms      |           energy |      0.05 | [-0.03,  0.12] |   1.30 | > .999
## duration_ms      | instrumentalness |      0.16 | [ 0.09,  0.24] |   4.39 | < .001***
## duration_ms      |         loudness |     -0.08 | [-0.15,  0.00] |  -2.08 | 0.572
## duration_ms      |            tempo |      0.01 | [-0.06,  0.09] |   0.36 | > .999
## duration_ms      |           valence |     -0.15 | [-0.22, -0.08] |  -3.96 | 0.002**
## energy           | instrumentalness |      0.16 | [ 0.09,  0.23] |   4.23 | < .001***
## energy           |          loudness |      0.61 | [ 0.56,  0.66] |  20.35 | < .001***
## energy           |             tempo |      0.03 | [-0.04,  0.11] |   0.87 | > .999
## energy           |           valence |      0.27 | [ 0.20,  0.34] |   7.33 | < .001***
## instrumentalness |          loudness |     -0.19 | [-0.26, -0.11] |  -4.97 | < .001***
## instrumentalness |             tempo |      0.07 | [-0.01,  0.14] |   1.77 | 0.916
## instrumentalness |           valence |     -0.09 | [-0.16, -0.01] |  -2.30 | 0.373
## loudness         |             tempo |     -0.02 | [-0.10,  0.05] |  -0.55 | > .999
## loudness         |           valence |     -0.01 | [-0.08,  0.06] |  -0.27 | > .999
## tempo            |           valence |      0.16 | [ 0.09,  0.23] |   4.30 | < .001***
##
## p-value adjustment method: Holm (1979)
## Observations: 693
```

There were three variable pairs that stood out as highly significantly different:

```
# Plots of variables with the largest partial correlation

# Danceability / Valence
ggplot(data = training_set, aes(danceability, valence)) + geom_jitter(color = "blue")
```

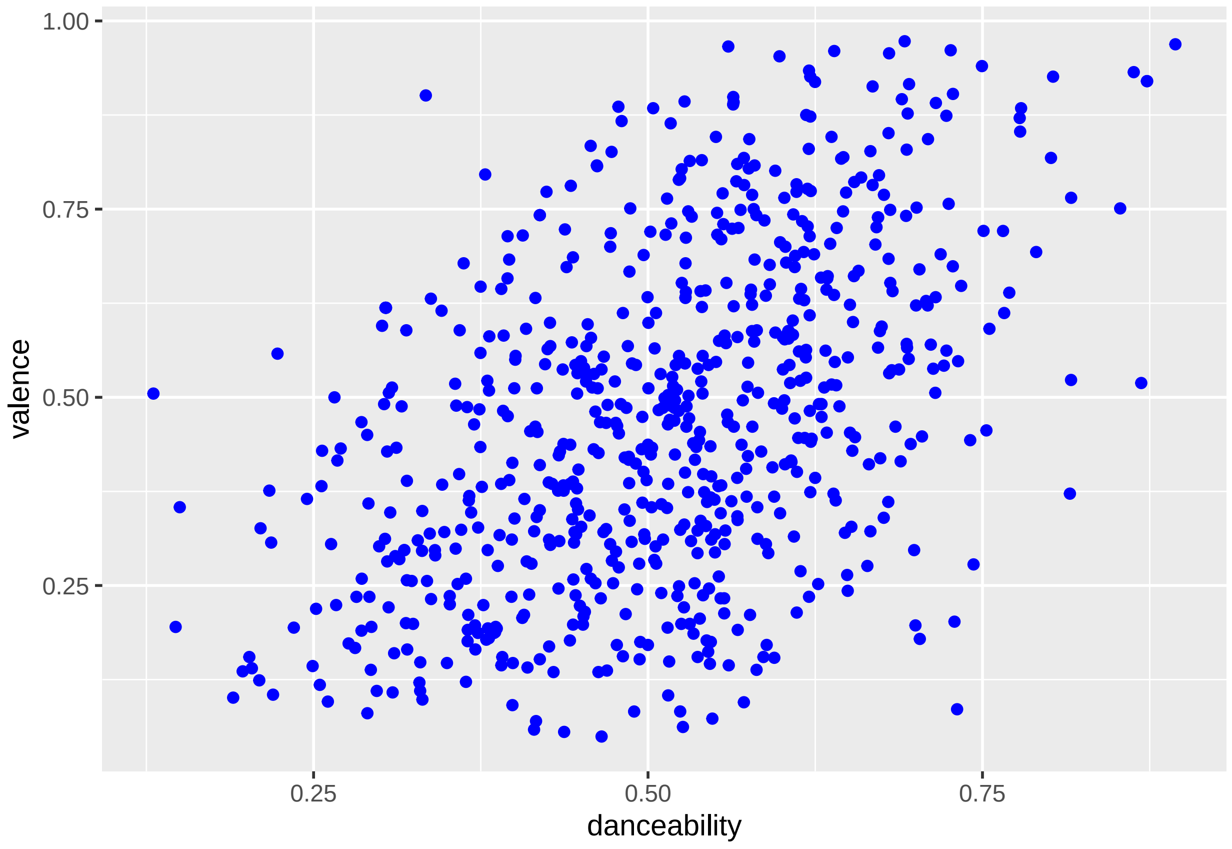

Figure 3: Partial Correlation between Danceability and Valence

```
# Loudness / Energy
ggplot(data = training_set, aes(loudness, energy)) + geom_jitter(color = "blue")
```

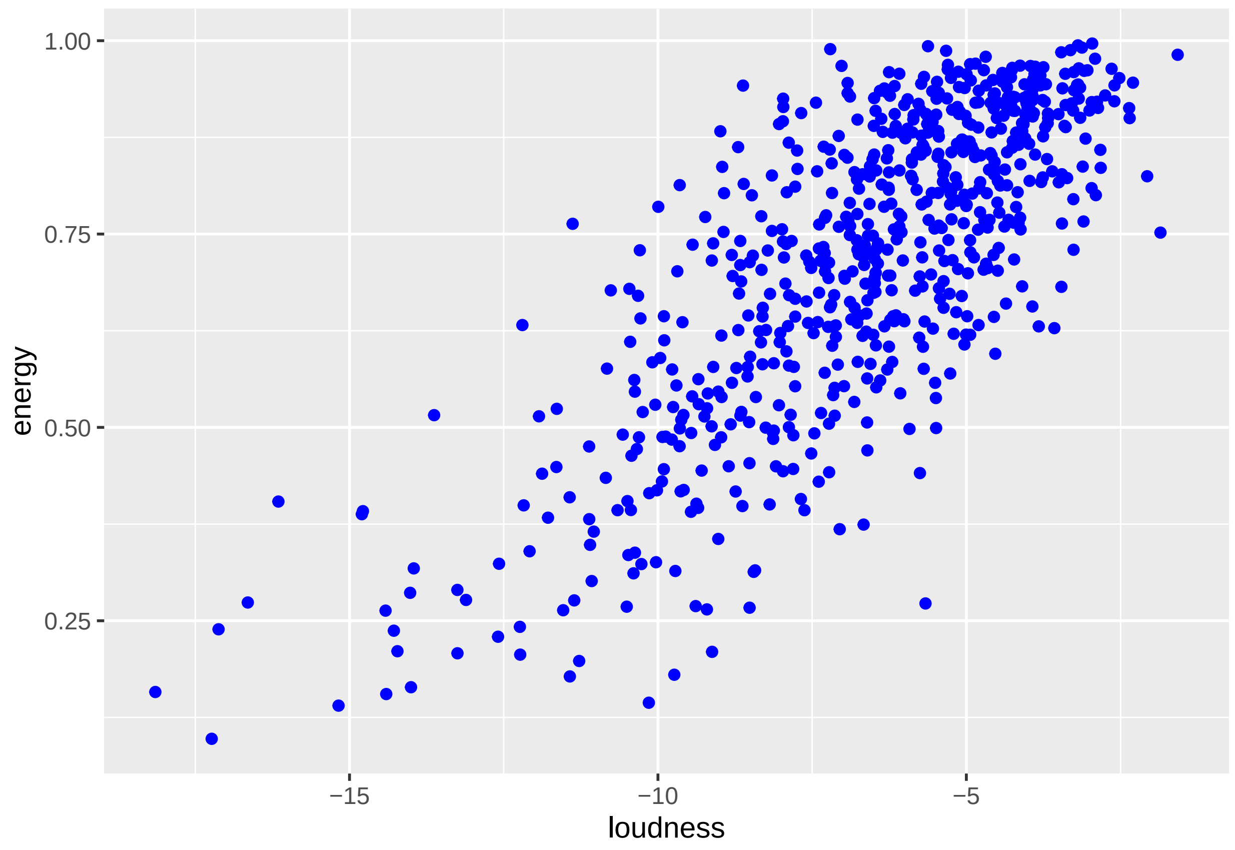

Figure 4: Partial Correlation between Loudness and Energy

```
# Acousticness / Energy
ggplot(data = training_set, aes(acousticness, energy)) + geom_jitter(color = "blue")
```

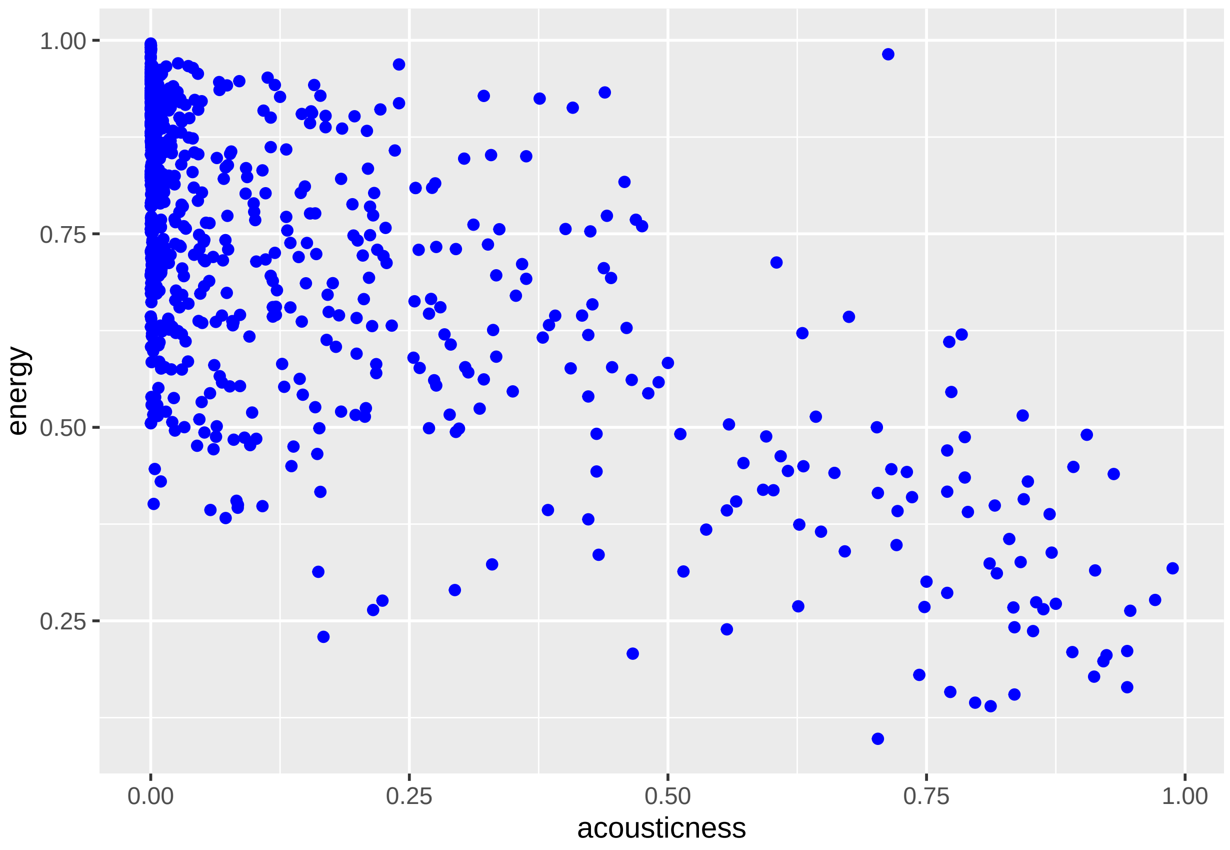

Figure 5: Partial Correlation between Acousticness and Energy

These three pairings are the same three largest correlations seen from the first correlation plots, confirming those findings. It is also interesting to see how the correlation value is lower once cleaned from the confounding effects.

By definition, correlation speaks to linear relationships. This information will directly guide how linear models of the data are developed, but non-linear relationships between the highly correlated data points could still exist.

```
# Possible Outlier
which.max(data$duration_ms)
```

```
## [1] 448
```

```
data[448, ]
```

```
## # A tibble: 1 x 14
##   names              IsWinner  Year acousticness danceability duration_ms energy
##   <chr>                 <int> <int>        <dbl>        <dbl>       <int>  <dbl>
## 1 Play - Dave Grohl         0  2018     0.000308        0.197     1355938  0.837
## # i 7 more variables: instrumentalness <dbl>, key <fct>, loudness <dbl>,
## #   mode <fct>, tempo <dbl>, time_signature <fct>, valence <dbl>
```

As a quick note, the song "Play" by Dave Grohl stood out during the analysis, being nearly 23 minutes long. Even though it could be considered an outlier, the song was not removed from the dataset since it can give meaningful insight on one of the reasons why a song may not gain popularity. Intuitively, a song that is very long is less likely to be played on the radio or used in movies, so it may not be heard by many people. This can later influence its likelihood of being considered for a Grammy nomination.

Distributions

An examination of the variables' distributions can give insight to better understand the data and how it can be interacted with. Some models that will be used in this project are based on assumptions that are related to distributions, often times assuming a normal distribution of data. From gauging how different variables are distributed, better models can be created by taking their distribution assumptions into account.

Continuous

```
# Checking distributions

par(mfrow= c(2, 4))

# Continuous variables
hist(acousticness)
hist(danceability)
hist(duration_ms)
hist(energy)
hist(instrumentalness)
hist(loudness)
hist(tempo)
hist(valence)
```

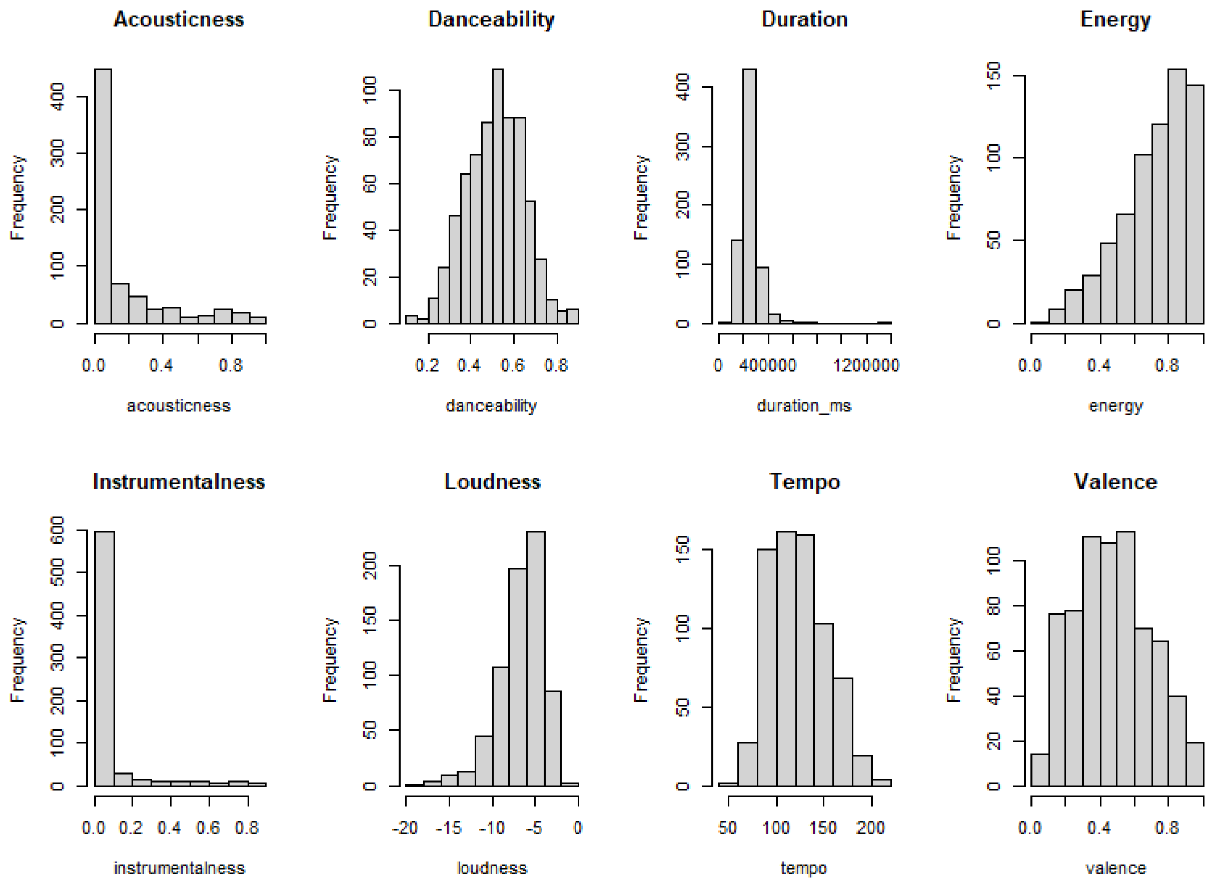

Figure 6: Histograms of all the continuous variables

From these histograms, it is clear that the danceability feature is the most normally distributed. All others have a small skew at best (valence) to larger skews (tempo, loudness, energy) to seemingly exponential skews or individual values (duration, acousticness, instrumentalness). These distributions will be considered as the models are constructed.

The distributions and means of each variable for non-nominated and nominated songs were also compared. Most were very similar, but highlighted below is the attribute with the greatest difference: Valence.

```
# Comparison IsWinner 0 vs 1 - Valence

par(mfrow = c(1, 2))

x <- data[data$IsWinner == 0,]$valence
hist(x, main = "Valence: Non-Nominated", xlab="Value")
abline(v = mean(x),                          # Add line for mean
       col = "red",
       lwd = 3)
text(x = mean(x) * 1.6,                      # Add text for mean
     y = 121,
     paste("Mean =", round(mean(x), digits=2)),
     col = "red",
     cex = 1)

x <- data[data$IsWinner == 1,]$valence
hist(x, main = "Valence: Nominated", xlab="Value")
abline(v = mean(x),                            # Add line for mean
       col = "red",
       lwd = 3)
text(x = mean(x) * 0.45,                        # Add text for mean
     y = 31,
     paste("Mean =", round(mean(x), digits=2)),
     col = "red",
     cex = 1)
```

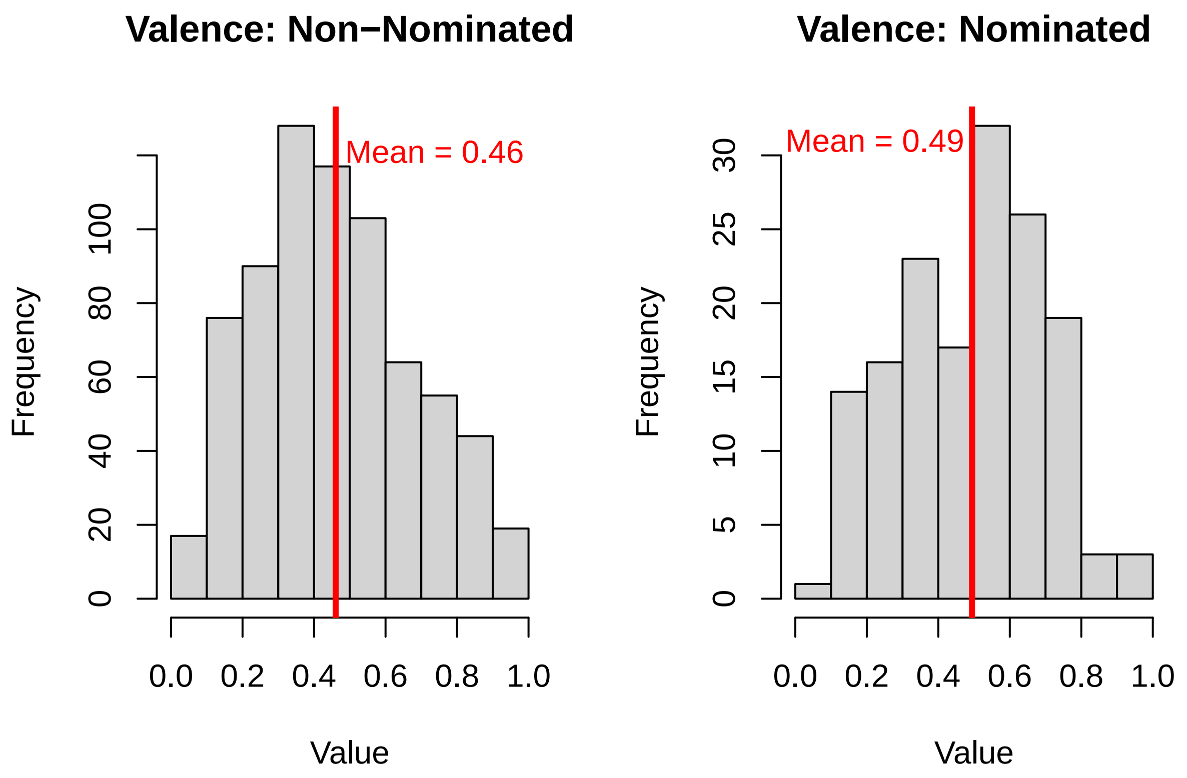

Figure 7: Distribution of the Valence variable conditioned to the dependent variable

The valence variable peaks at different locations for the two values of the dependent variable and has a different mean. This could point towards valence being an important factor for the models to consider.

Categorical

```
# Categorical variables

par(mfrow = c(1, 3))

barplot(table(key), main = "Key distribution")
barplot(table(mode), main = "Modality")
barplot(table(time_signature), main = "Time signature")
```

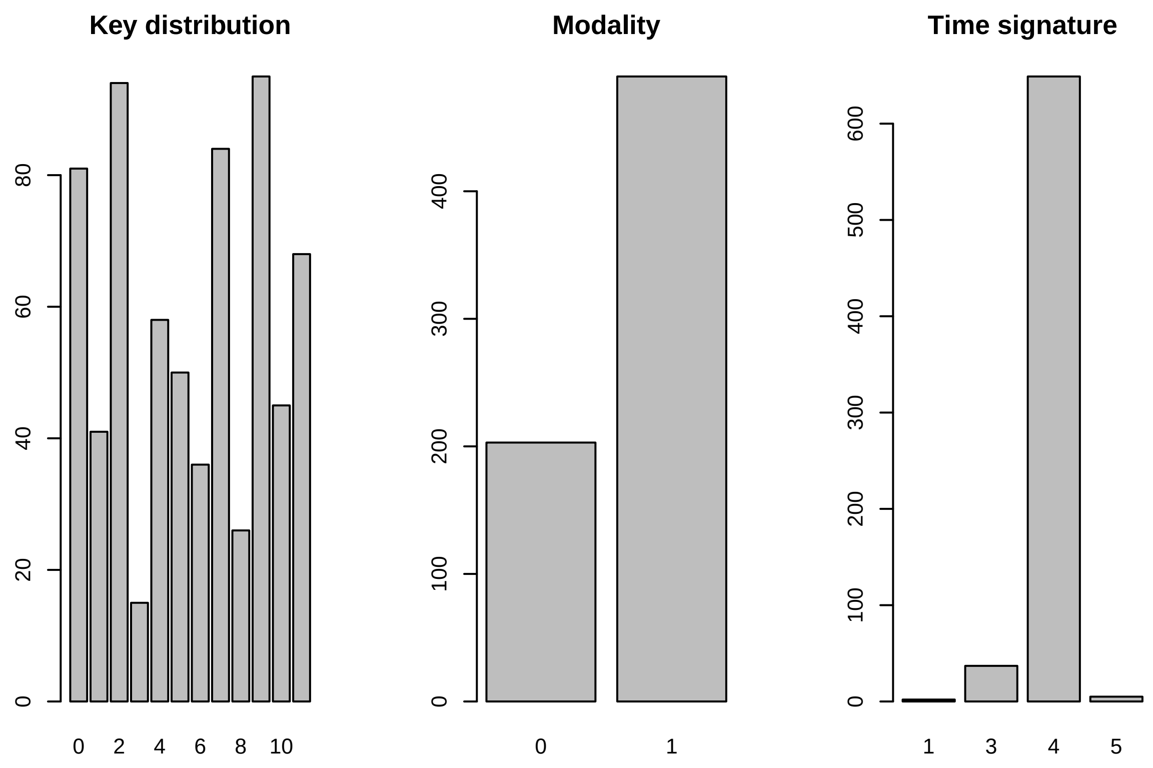

Figure 8: Bar plots of the categorical variables

Both mode and time_signature have an obvious statistical mode. The value of 1 for the variable mode represents songs in major keys, making up a super majority of the songs, and almost every song uses the time_signature of 4 with only a handful using 1, 3, or 5.

The distributions of non-nominated to nominated songs for the categorical variables was again compared, and highlighted below is the variable that demonstrated the greatest difference: Key.

```r
# Comparison IsWinner 0 vs 1 - Key

par(mfrow = c(1, 2))
x <- data[data$IsWinner == 0,]$key
barplot(table(x), main = "Key: Non-Nominated")

x <- data[data$IsWinner == 1,]$key
barplot(table(x), main = "Key: Nominated/Winner")
```

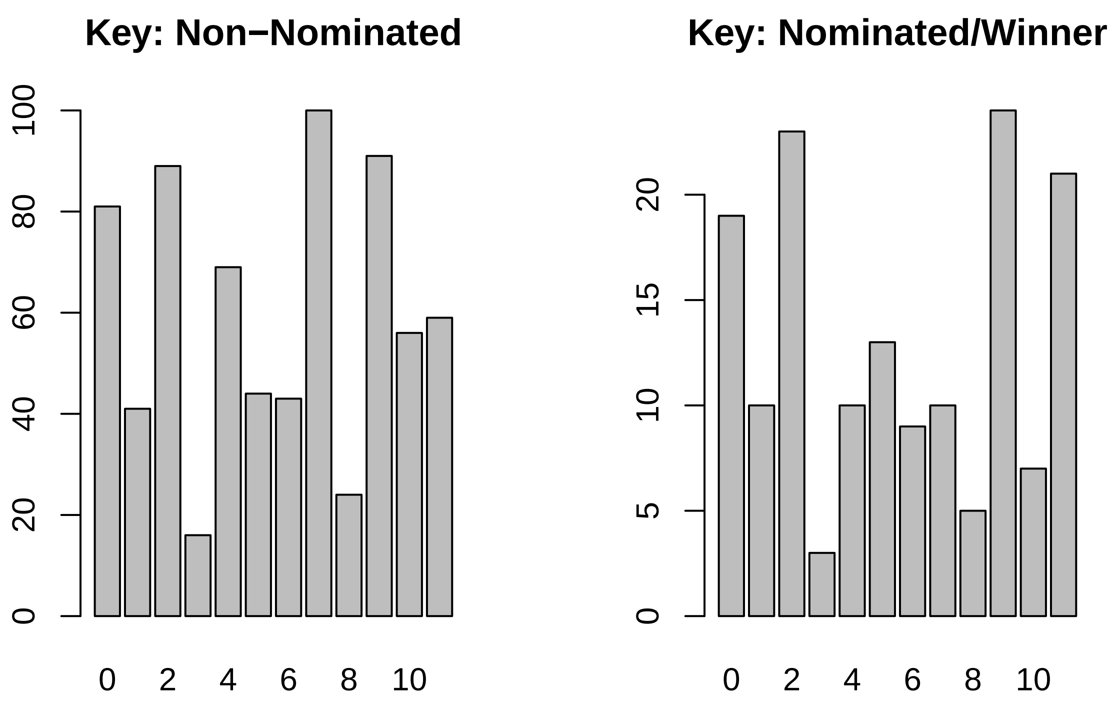

Figure 9: Distribution of the Key variable conditioned to the dependent variable

Although the distribution of key for both values of the dependent variable look very similar, there are some important differences. There is a much larger proportion of non-nominated songs written in key 4 (corresponding to the key of E) and key 7 (corresponding to the key of G) when compared to nominated songs. The opposite can be said about key 11 (key of B), with a larger proportion of nominated songs in that key when compared to non-nominated songs.

While these patterns are observable, the effect size may not be substantial enough to reach statistical significance, will be seen in the Chi-Squared tests performed. In other words, while key '11' may appear more often in nominated songs, the difference may not be large enough to confidently conclude that it has a significant impact on nominations.

Relationship Between Independent and Dependent Variables

Examining the association between independent and dependent variables can give insight into the data of independent variables which exhibit notable differences between the two possible values of the dependent variable.

Continuous

Box plots were utilized to visualize these relationships:

```r
# Relationships between dependent and independent variables

par(mfrow= c(2, 4))

boxplot(danceability ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(acousticness ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(duration_ms ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(energy ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(instrumentalness ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(loudness ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(tempo ~ training_set$IsWinner, xlab='Nominee Boolean')
boxplot(valence ~ training_set$IsWinner, xlab='Nominee Boolean')
```

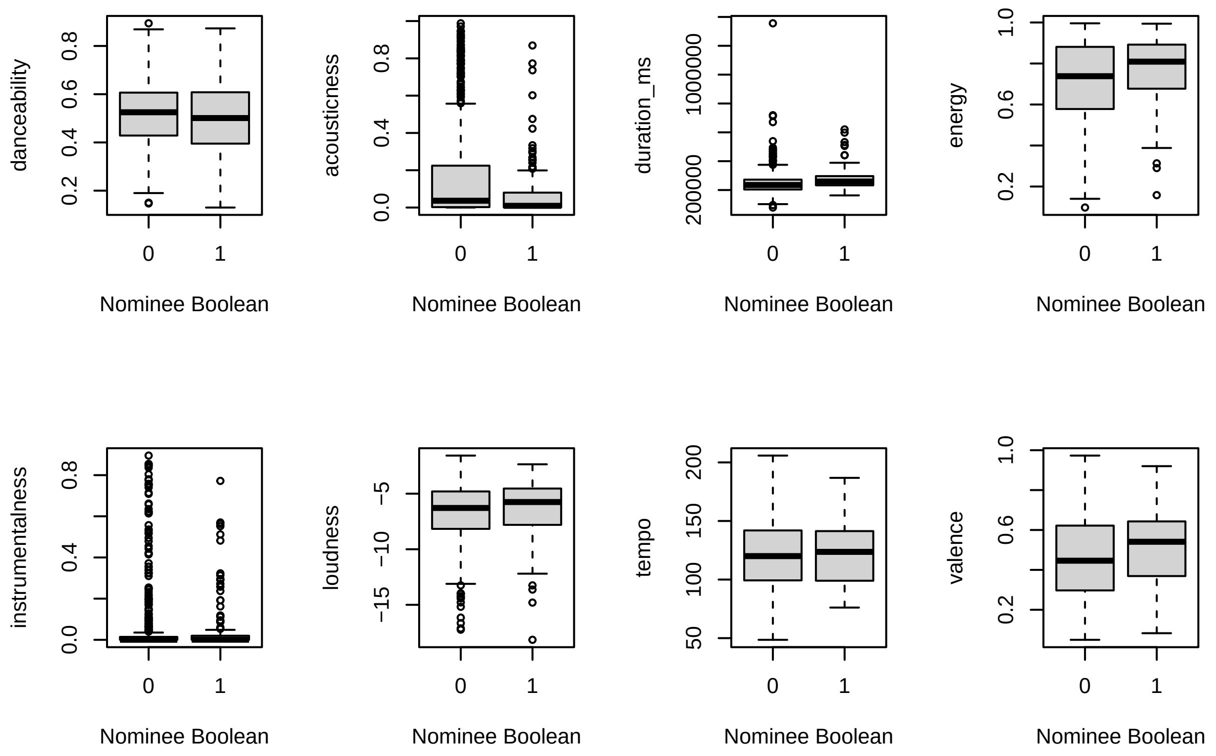

Figure 10: Boxplots of continuous independent variables conditioned to the dependent variable

The difference in median seen for the energy and valence variables points towards them being possibly valuable to the models. The difference in interquartile range for acousticness, energy, and valence is also interesting to see.

Categorical

The Chi Squared Test was utilized to measure the independent/dependent variables association for the categorical variables. It will pass judgement on the level of independence between the variables.

```
par(mfrow = c(1, 1))

chisq.test(key, training_set$IsWinner)
```

```
##
##  Pearson's Chi-squared test
##
## data:  key and training_set$IsWinner
## X-squared = 10.443, df = 11, p-value = 0.491
```

```
chisq.test(mode, training_set$IsWinner)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  mode and training_set$IsWinner
## X-squared = 0, df = 1, p-value = 1
```

```r
chisq.test(time_signature, training_set$IsWinner)
```

```
##
##  Pearson's Chi-squared test
##
## data:  time_signature and training_set$IsWinner
## X-squared = 3.3331, df = 3, p-value = 0.3431
```

The Chi-Squared Test revealed fairly high p-values for all of the categorical variables, and even produced a 1 for mode, suggesting complete independence between mode and the dependent variable.

The Cramer's V test was also applied and obtained values that suggest some kind of dependence between the independent and dependent variables.

```r
cramersv(matrix(c(as.numeric(key), as.numeric(training_set$IsWinner)), ncol = 2))
```

```
## Warning in stats::chisq.test(x, correct = correct): Chi-squared approximation
## may be incorrect
```

```
## [1] 0.4116833
```

```r
cramersv(matrix(c(as.numeric(mode), as.numeric(training_set$IsWinner)), ncol = 2))
```

```
## Warning in stats::chisq.test(x, correct = correct): Chi-squared approximation
## may be incorrect
```

```
## [1] 0.5604862
```

```r
cramersv(matrix(c(as.numeric(time_signature), as.numeric(training_set$IsWinner)), ncol = 2))
```

```
## Warning in stats::chisq.test(x, correct = correct): Chi-squared approximation
## may be incorrect
```

```
## [1] 0.4527729
```

Summary

From this exploratory data analysis, correlations between continuous variables were certainly found. Especially high correlations exist between the audio features acousticness/energy and loudness/energy. No categorical variables were shown to have any significant association with the dependent variable.

After examining the distribution for both continuous and categorical variables, most continuous variables showed non-normal distributions, which should be considered when building models. Categorical variables like mode and time signature had high frequencies of specific values.

For continuous variables, box plots visualized differences between nominated and non-nominated songs. Notably, energy and valence showed differences in medians. For categorical variables, Chi-Squared tests and Cramer's V indicated that these variables might not be strongly associated with the Grammy nomination status.

From this analysis, one can extract that features like energy and valence may play a role in distinguishing nominated from non-nominated songs. On the other hand, categorical variables did not show significant impact on nominations. This information is surely valuable for the next step, model development.

Model Development

In this project's model development strategy, a progressive approach was deployed, beginning with a logistic model and refining it through oversampling and variable selection techniques. Models such as ridge and lasso regression and K-nearest neighbors (KNN) were then explored to enhance predictive accuracy. Model performance is assessed through Type I Error, Sensitivity, and ROC curve analysis, while also considering discriminant analysis assuming normal distribution of key variables.

Oversampling

Because of the imbalanced nature of the dataset that was shown previously, an oversampled version of the dataset will be used in an attempt to further improve the models. This oversampling creates synthetic data of the minority class (nominated) with data values for all variables similar to and determined by the real data points of the minority class. This creates a dataset of balanced non-nominated and nominated songs.

```
## Oversampling

oversampled_train_data = ovun.sample(IsWinner ~.,
                                     data = training_set[,-1],
                                     method = "over",
                                     p = 0.5, seed = 42)$data

# Checking oversampled training set balance

sum(oversampled_train_data$IsWinner == 0)
```

```
## [1] 563
```

```
sum(oversampled_train_data$IsWinner == 1)
```

```
## [1] 538
```

Logistic Model

The logistic regression model was considered initially, a model that belongs to the category of the generalized linear models. The logistic regression model predicts the probability of an event to occur (in this case, of a song to be nominated) by having the log-odds of the event to become a linear combination of the independent variables.

```
## Simple logistic model

logistic = glm(IsWinner ~ ., data = training_set[,c(-1,-2)],
               family = "binomial")
summary(logistic)
```

```
##
## Call:
## glm(formula = IsWinner ~ ., family = "binomial", data = training_set[,
##     c(-1, -2)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4917  -0.6917  -0.5259  -0.2675   2.4856
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.316e+01  1.009e+03   0.013  0.98959
## Year              -1.573e-02  1.155e-02  -1.362  0.17310
## acousticness      -2.498e+00  7.920e-01  -3.153  0.00161 **
## danceability      -1.112e+00  1.009e+00  -1.102  0.27034
## duration_ms        4.193e-06  1.355e-06   3.095  0.00197 **
## energy             6.726e-01  1.013e+00   0.664  0.50665
## instrumentalness -6.925e-01  7.431e-01  -0.932  0.35140
## key1               1.318e-01  5.074e-01   0.260  0.79501
## key2              -9.187e-02  4.005e-01  -0.229  0.81856
## key3              -2.383e-01  7.202e-01  -0.331  0.74070
## key4              -9.570e-02  4.805e-01  -0.199  0.84213
## key5               4.763e-01  4.566e-01   1.043  0.29694
## key6              -1.639e-02  5.301e-01  -0.031  0.97534
## key7              -7.271e-01  4.545e-01  -1.600  0.10962
## key8              -6.845e-02  6.012e-01  -0.114  0.90936
## key9              -4.907e-02  4.041e-01  -0.121  0.90336
## key10             -7.789e-01  6.226e-01  -1.251  0.21087
## key11              5.632e-01  4.349e-01   1.295  0.19528
## loudness          -1.177e-01  6.954e-02  -1.693  0.09051 .
## mode1              7.803e-02  2.434e-01   0.321  0.74854
## tempo             -4.585e-04  3.857e-03  -0.119  0.90537
## time_signature3    1.452e+01  1.009e+03   0.014  0.98851
## time_signature4    1.497e+01  1.009e+03   0.015  0.98816
## time_signature5    1.038e+00  1.181e+03   0.001  0.99930
## valence            1.288e+00  5.951e-01   2.165  0.03038 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 669.04  on 692  degrees of freedom
## Residual deviance: 609.75  on 668  degrees of freedom
## AIC: 659.75
##
## Number of Fisher Scoring iterations: 14
```

From the creation of the model, it determined valence, duration, and acousticness to be significant variables.

Predictions for the full Logistic Model

```r
# Computing predictions

logistic_predictions_full = predict(logistic,
                                    newdata = test_set[,c(-1, -2)],
                                    type = "response")
```

Below are the predictions as computed by the logistic model created above, using different thresholds of classification. The threshold defines at which confidence level the model will classify a song as "nominated."

Threshold = 0.2

```r
# Threshold = 0.2

logistic_predictions_full_02 = ifelse(logistic_predictions_full > 0.2, 1, 0)
logistic_accuracy_full_02 = sum(logistic_predictions_full_02 == test_set[2]) /
```

```
                                 dim(test_set[2])[1]

table(test_set$IsWinner, logistic_predictions_full_02)
```

```
##    logistic_predictions_full_02
##       0   1
##   0 102  48
##   1  12  12
```

```
false_positive_logistic_full_02 = table(test_set$IsWinner, logistic_predictions_full_02)[3]
negative_logistic_full_02 = table(test_set$IsWinner, logistic_predictions_full_02)[1] +
                                  table(test_set$IsWinner, logistic_predictions_full_02)[3]
typeIerror_logistic_full_02 = false_positive_logistic_full_02 / negative_logistic_full_02

typeIerror_logistic_full_02
```

```
## [1] 0.32
```

```
true_positive_logistic_full_02 = table(test_set$IsWinner, logistic_predictions_full_02)[4]
positive_logistic_full_02 = table(test_set$IsWinner, logistic_predictions_full_02)[2] +
                                  table(test_set$IsWinner, logistic_predictions_full_02)[4]
sensitivity_logistic_full_02 = true_positive_logistic_full_02 / positive_logistic_full_02

sensitivity_logistic_full_02
```

```
## [1] 0.5
```

At a classification threshold of 0.2, the Type I Error (False Positive Rate), defined as False Positive classifications / Total Negative songs, was 0.32. That means around a third of the non-nominated songs were misclassified as nominated

The Sensitivity (True Positive Rate), defined as True Positive classifications / Total positive songs was 0.5. Contextually, this means that half of the nominated songs were classified correctly.

Ideally, the Type I error should be lower than this and the Sensitivity should be much higher.

Threshold = 0.3

In an attempt to improve the classification of songs from the model, the threshold was increased to 0.3.

```
# Threshold = 0.3

logistic_predictions_full_03 = ifelse(logistic_predictions_full > 0.3, 1, 0)
logistic_accuracy_full_03 = sum(logistic_predictions_full_03 == test_set[2]) /
                                 dim(test_set[2])[1]

table(test_set$IsWinner, logistic_predictions_full_03)
```

```
##    logistic_predictions_full_03
##       0   1
##   0 133  17
##   1  22   2
```

```
false_positive_logistic_full_03 = table(test_set$IsWinner, logistic_predictions_full_03)[3]
negative_logistic_full_03 = table(test_set$IsWinner, logistic_predictions_full_03)[1] +
                            table(test_set$IsWinner, logistic_predictions_full_03)[3]
typeIerror_logistic_full_03 = false_positive_logistic_full_03 / negative_logistic_full_03
```

```
typeIerror_logistic_full_03
```

```
## [1] 0.1133333
```

```
true_positive_logistic_full_03 = table(test_set$IsWinner, logistic_predictions_full_03)[4]
positive_logistic_full_03 = table(test_set$IsWinner, logistic_predictions_full_03)[2] +
                                  table(test_set$IsWinner, logistic_predictions_full_03)[4]
sensitivity_logistic_full_03 = true_positive_logistic_full_03 / positive_logistic_full_03

sensitivity_logistic_full_03
```

```
## [1] 0.08333333
```

With an increase in threshold to 0.3, the Type I Error decreased to 0.113. This was an improvement, as it means there were fewer non-nominated songs classified as nominated, but to get a full picture Sensitivity still needed to be examined.

The Sensitivity also decreased to 0.083, which is not optimal behavior. This means that at a classification threshold of 0.3, the model correctly predicts only 8% of the nominated songs.

The combined information from these two tests suggests the model is classifying more songs as non-nominated, avoiding false positives while also missing correct classifications of nominated songs.

Threshold = 0.4

For a final comparison, the threshold was set to 0.4.

```
# Threshold = 0.4

logistic_predictions_full_04 = ifelse(logistic_predictions_full > 0.4, 1, 0)
logistic_accuracy_full_04 = sum(logistic_predictions_full_04 == test_set[2]) /
                                dim(test_set[2])[1]

table(test_set$IsWinner, logistic_predictions_full_04)
```

```
##    logistic_predictions_full_04
##       0   1
##   0 148   2
##   1  24   0
```

```
false_positive_logistic_full_04 = table(test_set$IsWinner, logistic_predictions_full_04)[3]
negative_logistic_full_04 = table(test_set$IsWinner, logistic_predictions_full_04)[1] +
                            table(test_set$IsWinner, logistic_predictions_full_04)[3]
typeIerror_logistic_full_04 = false_positive_logistic_full_04 / negative_logistic_full_04

typeIerror_logistic_full_04
```

```
## [1] 0.01333333
```

```
true_positive_logistic_full_04 = table(test_set$IsWinner, logistic_predictions_full_04)[4]
positive_logistic_full_04 = table(test_set$IsWinner, logistic_predictions_full_04)[2] +
                            table(test_set$IsWinner, logistic_predictions_full_04)[4]
sensitivity_logistic_full_04 = true_positive_logistic_full_04 / positive_logistic_full_04

sensitivity_logistic_full_04
```

```
## [1] 0
```

With an increase in threshold to 0.4, the Type I Error decreased further to 0.013. This again was an improvement, but it may be detrimental to the overall model accuracy.

The Sensitivity reached 0. This means the model managed to properly classify 0 nominated songs

Overall, increasing the threshold was detrimental.

ROC Curve

An ROC Curve is meant to display the performance of a classification model. It plots the True Positive Rate (Sensitivity) against the False Positive Rate (Type I Error) across different confidence thresholds. This takes into account both sides of the classification accuracy discussed above and paints a comprehensive picture of how well the model can classify the song data.

```
# ROC curve

roc.out <- roc(test_set$IsWinner, logistic_predictions_full)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

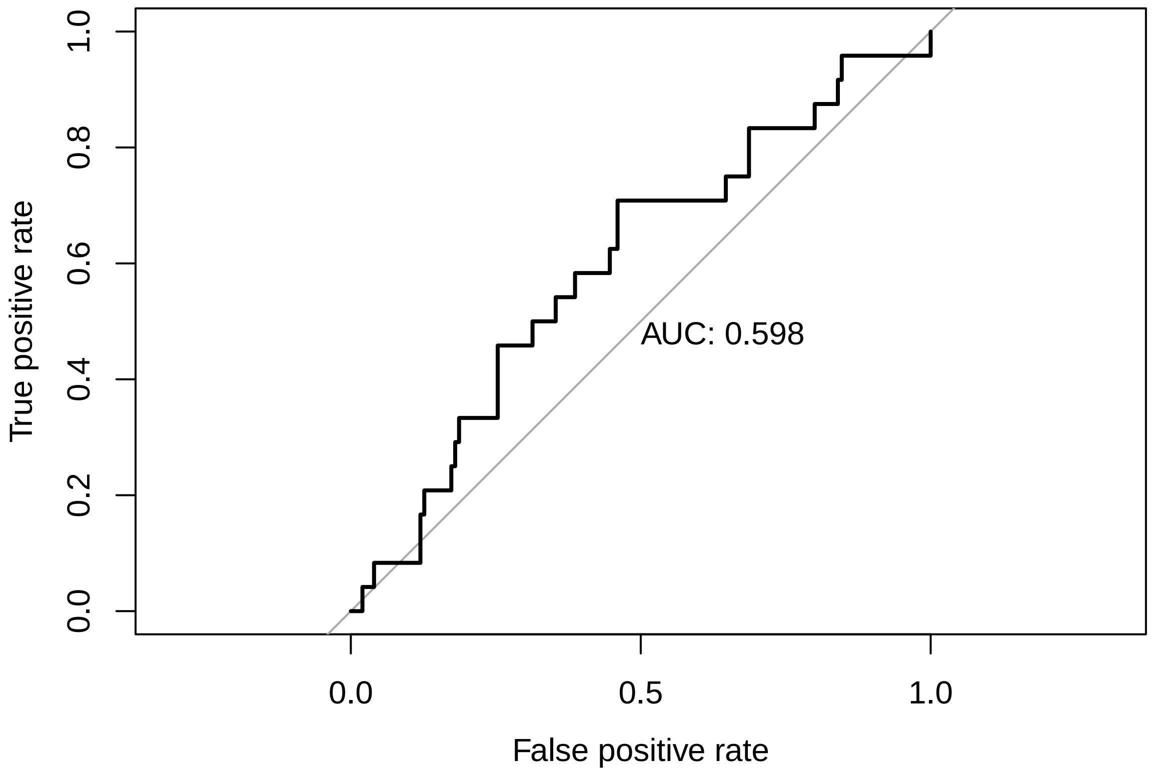

Figure 11: ROC Curve of the Full Logistic Model

```
auc(roc.out)
```

```
## Area under the curve: 0.5978
```

The ROC curve produced by the logistic model touts an Area Under the ROC Curve (AUR/AUC) of 0.598, meaning it achieves almost 60% diagnostic accuracy. An AUC larger than 0.5 or 50% is needed to make a model more efficient than a coin flip, so the basic logistic model does attain some predictive capabilities.

Stepwise Variable Selection and Reduced Logistic Model

Stepwise variable selection was performed in an attempt to further improve the logistic model. This process examines each variable in the model and removes the least beneficial one from the model's consideration. Once it removes one, it re-examines the remaining variables and repeats the process. As it removes more and more variables, it also goes back and checks if re-adding a previously removed variable would benefit the current state of the model at that time.

```
# Stepwise variable selection

log_both =  stepAIC(logistic, direction = "both")
```

```
## Start:  AIC=659.75
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     energy + instrumentalness + key + loudness + mode + tempo +
##     time_signature + valence
##
##                    Df Deviance    AIC
## - key              11   622.43 650.43
## - tempo             1   609.76 657.76
## - time_signature    3   613.77 657.77
## - mode              1   609.85 657.85
## - energy            1   610.19 658.19
## - instrumentalness  1   610.68 658.68
## - danceability      1   610.96 658.96
## - Year              1   611.62 659.62
## <none>                  609.75 659.75
## - loudness          1   612.61 660.61
## - valence           1   614.50 662.50
## - duration_ms       1   620.54 668.54
## - acousticness      1   621.35 669.35
##
## Step:  AIC=650.43
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     energy + instrumentalness + loudness + mode + tempo + time_signature +
##     valence
##
##                    Df Deviance    AIC
## - time_signature    3   626.10 648.10
## - mode              1   622.45 648.45
## - tempo             1   622.49 648.49
## - energy            1   622.96 648.96
## - instrumentalness  1   623.14 649.14
## - danceability      1   623.35 649.35
## - Year              1   623.63 649.63
## <none>                  622.43 650.43
## - loudness          1   624.85 650.85
## - valence           1   628.06 654.06
## - acousticness      1   632.83 658.83
## - duration_ms       1   633.35 659.35
## + key              11   609.75 659.75
```

```
##
## Step:  AIC=648.1
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     energy + instrumentalness + loudness + mode + tempo + valence
##
##                    Df Deviance    AIC
## - mode              1   626.11 646.11
## - tempo             1   626.16 646.16
## - danceability      1   626.85 646.85
## - energy            1   626.88 646.88
## - Year              1   627.35 647.35
## - instrumentalness  1   627.52 647.52
## <none>                  626.10 648.10
## - loudness          1   628.79 648.79
## + time_signature    3   622.43 650.43
## - valence           1   631.73 651.73
## - duration_ms       1   636.51 656.51
## - acousticness      1   636.75 656.75
## + key              11   613.77 657.77
##
## Step:  AIC=646.11
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     energy + instrumentalness + loudness + tempo + valence
##
##                    Df Deviance    AIC
## - tempo             1   626.16 644.16
## - danceability      1   626.85 644.85
## - energy            1   626.89 644.89
## - Year              1   627.36 645.36
## - instrumentalness  1   627.52 645.52
## <none>                  626.11 646.11
## - loudness          1   628.80 646.80
## + mode              1   626.10 648.10
## + time_signature    3   622.45 648.45
## - valence           1   631.74 649.74
## - duration_ms       1   636.51 654.51
## - acousticness      1   636.75 654.75
## + key              11   613.92 655.92
##
## Step:  AIC=644.16
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     energy + instrumentalness + loudness + valence
##
##                    Df Deviance    AIC
## - energy            1   626.97 642.97
## - danceability      1   627.15 643.15
## - Year              1   627.37 643.37
## - instrumentalness  1   627.56 643.56
## <none>                  626.16 644.16
## - loudness          1   628.87 644.87
## + tempo             1   626.11 646.11
## + mode              1   626.16 646.16
## + time_signature    3   622.50 646.50
## - valence           1   632.13 648.13
## - duration_ms       1   636.57 652.57
## - acousticness      1   636.77 652.77
## + key              11   613.93 653.93
##
## Step:  AIC=642.97
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     instrumentalness + loudness + valence
##
##                    Df Deviance    AIC
## - instrumentalness  1   628.09 642.09
## - Year              1   628.28 642.28
## - danceability      1   628.33 642.33
## - loudness          1   628.89 642.89
## <none>                  626.97 642.97
## + energy            1   626.16 644.16
## + tempo             1   626.89 644.89
## + mode              1   626.96 644.96
## + time_signature    3   623.07 645.07
## - valence           1   634.70 648.70
## - duration_ms       1   637.71 651.71
## + key              11   614.57 652.57
## - acousticness      1   642.14 656.14
##
## Step:  AIC=642.09
## IsWinner ~ Year + acousticness + danceability + duration_ms +
##     loudness + valence
##
##                    Df Deviance    AIC
## - danceability      1   629.35 641.35
## - loudness          1   629.63 641.63
## - Year              1   629.83 641.83
## <none>                  628.09 642.09
## + instrumentalness  1   626.97 642.97
## + energy            1   627.56 643.56
## + time_signature    3   623.59 643.59
## + tempo             1   628.04 644.04
## + mode              1   628.09 644.09
## - valence           1   635.88 647.88
## - duration_ms       1   637.81 649.81
## + key              11   616.08 652.08
## - acousticness      1   642.66 654.66
##
## Step:  AIC=641.35
## IsWinner ~ Year + acousticness + duration_ms + loudness + valence
##
##                    Df Deviance    AIC
## - loudness          1   630.46 640.46
## <none>                  629.35 641.35
## - Year              1   631.52 641.52
## + danceability      1   628.09 642.09
## + instrumentalness  1   628.33 642.33
## + energy            1   628.51 642.51
## + tempo             1   629.00 643.00
## + time_signature    3   625.01 643.01
## + mode              1   629.35 643.35
## - valence           1   635.94 645.94
## - duration_ms       1   640.36 650.36
## + key              11   617.43 651.43
## - acousticness      1   644.51 654.51
##
## Step:  AIC=640.46
## IsWinner ~ Year + acousticness + duration_ms + valence
##
##                    Df Deviance    AIC
## <none>                 630.46 640.46
## + loudness          1   629.35 641.35
## + danceability      1   629.63 641.63
## + instrumentalness  1   629.75 641.75
## - Year              1   633.93 641.93
## + tempo             1   630.19 642.19
## + time_signature    3   626.23 642.23
## + energy            1   630.46 642.46
## + mode              1   630.46 642.46
## - valence           1   636.42 644.42
## - duration_ms       1   642.21 650.21
## + key              11   619.02 651.02
## - acousticness      1   646.13 654.13
```

According to the stepwise selection, the most valuable variables for the logistic model are: year, valence, duration, and acousticness.

```
# Fitting the reduced model

logistic_reduced = glm(IsWinner ~  Year + valence + duration_ms + acousticness,
                       data = training_set,  family = "binomial")

summary(logistic_reduced)
```

```
##
## Call:
## glm(formula = IsWinner ~ Year + valence + duration_ms + acousticness,
##     family = "binomial", data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3010  -0.6878  -0.5905  -0.2982   2.5190
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.591e+01  2.084e+01   1.723 0.084923 .
## Year         -1.928e-02  1.037e-02  -1.860 0.062921 .
## valence       1.161e+00  4.771e-01   2.434 0.014943 *
## duration_ms   3.990e-06  1.231e-06   3.243 0.001183 **
## acousticness -2.177e+00  6.354e-01  -3.426 0.000613 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 669.04  on 692  degrees of freedom
## Residual deviance: 630.46  on 688  degrees of freedom
## AIC: 640.46
##
## Number of Fisher Scoring iterations: 5
```

Utilizing the information gained from the stepwise variable selection, a reduced version of the logistic model was created.

Predictions for the Reduced Logistic Model

```
# Computing predictions

logistic_predictions = predict(logistic_reduced, newdata = test_set[,c(-1, -2)],
                               type = "response")
```

As done with the full logistic model, predictions using different thresholds were examined to ascertain the performance of the reduced model.

Threshold = 0.2

```
# Threshold = 0.2

logistic_predictions_02 = ifelse(logistic_predictions > 0.2, 1, 0)
logistic_accuracy_02 = sum(logistic_predictions_02 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, logistic_predictions_02)
```

```
##    logistic_predictions_02
##      0  1
##   0 94 56
##   1 11 13
```

```
false_positive_logistic_02 = table(test_set$IsWinner, logistic_predictions_02)[3]
negative_logistic_02 = table(test_set$IsWinner, logistic_predictions_02)[1] +
                             table(test_set$IsWinner, logistic_predictions_02)[3]
typeIerror_logistic_02 = false_positive_logistic_02 / negative_logistic_02

typeIerror_logistic_02
```

```
## [1] 0.3733333
```

```
true_positive_logistic_02 = table(test_set$IsWinner, logistic_predictions_02)[4]
positive_logistic_02 = table(test_set$IsWinner, logistic_predictions_02)[2] +
                       table(test_set$IsWinner, logistic_predictions_02)[4]
sensitivity_logistic_02 = true_positive_logistic_02 / positive_logistic_02

sensitivity_logistic_02
```

```
## [1] 0.5416667
```

At a confidence threshold of 0.2, the model produces a Type I Error of 0.37, incorrectly classifying 37% of the non-nominated songs as nominated. This is slightly worse than the 0.32 seen from the full model.

It also produces a Sensitivity of 0.54, meaning it correctly classifies 54% of the nominated songs, a slight improvement to the 0.5 produced from the full model at this threshold.

Threshold = 0.3

```
# Threshold = 0.3

logistic_predictions_03 = ifelse(logistic_predictions > 0.3, 1, 0)
logistic_accuracy_03 = sum(logistic_predictions_03 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, logistic_predictions_03)
```

```
##    logistic_predictions_03
##       0   1
##   0 144   6
##   1  23   1
```

```
false_positive_logistic_03 = table(test_set$IsWinner, logistic_predictions_03)[3]
negative_logistic_03 = table(test_set$IsWinner, logistic_predictions_03)[1] +
                       table(test_set$IsWinner, logistic_predictions_03)[3]
typeIerror_logistic_03 = false_positive_logistic_03 / negative_logistic_03

typeIerror_logistic_03
```

```
## [1] 0.04
```

```
true_positive_logistic_03 = table(test_set$IsWinner, logistic_predictions_03)[4]
positive_logistic_03 = table(test_set$IsWinner, logistic_predictions_03)[2] +
                       table(test_set$IsWinner, logistic_predictions_03)[4]
sensitivity_logistic_03 = true_positive_logistic_03 / positive_logistic_03

sensitivity_logistic_03
```

```
## [1] 0.04166667
```

As seen before with the full model, increasing the threshold to 0.3 decreased the Type I Error, but to a more significant degree. It now produces a value of 0.04, which is a significant drop in misclassifications of non-nominated songs. Coupled with that, however, was a significant drop in Sensitivity, reaching 0.042.

Threshold = 0.4

```
# Threshold = 0.4

logistic_predictions_04 = ifelse(logistic_predictions > 0.4, 1, 0)
logistic_accuracy_04 = sum(logistic_predictions_04 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, logistic_predictions_04)
```

```
##    logistic_predictions_04
##       0
##   0 150
##   1  24
```

```
false_positive_logistic_04 = table(test_set$IsWinner, logistic_predictions_04)[3]
negative_logistic_04 = table(test_set$IsWinner, logistic_predictions_04)[1] +
                       table(test_set$IsWinner, logistic_predictions_04)[3]
typeIerror_logistic_04 = false_positive_logistic_04 / negative_logistic_04

typeIerror_logistic_04
```

```
## [1] NA
```

```
true_positive_logistic_04 = table(test_set$IsWinner, logistic_predictions_04)[4]
positive_logistic_04 = table(test_set$IsWinner, logistic_predictions_04)[2] +
                       table(test_set$IsWinner, logistic_predictions_04)[4]
sensitivity_logistic_04 = true_positive_logistic_04 / positive_logistic_04

sensitivity_logistic_04
```

```
## [1] NA
```

As was done with the full logistic model, the threshold was increased to 0.4. With this change, the model classified no songs as nominated, producing values of N/A from the Type I Error and Sensitivity calculations.

ROC Curve

```
# ROC curve

roc.out <- roc(test_set$IsWinner, logistic_predictions)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

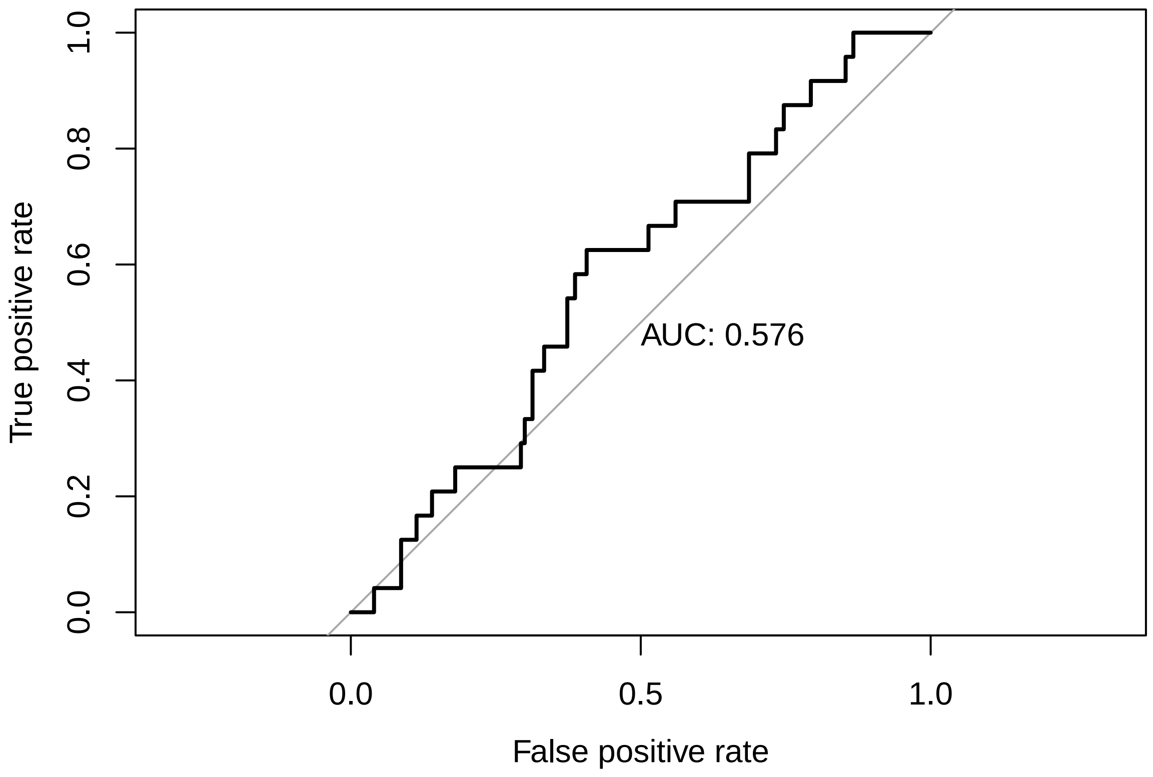

Figure 12: ROC Curve of the Reduced Logistic Model

```
auc(roc.out)
```

```
## Area under the curve: 0.5758
```

The ROC curve shows that the reduced logistic model produces a diagnostic accuracy of 0.576, slightly worse than the full logistic model.

Logistic Oversampled Model

In this section, a logistic model is created (same as seen above) using the previously created oversampled data set that was discussed at the beginning of the Model Development section as an attempt to improve the logistic model's accuracy.

```
# Fitting logistic oversampled

logistic_over = glm(as.numeric(unlist(oversampled_train_data[1])) ~ .,
                    data = oversampled_train_data[-1],
                    family = "binomial")
summary(logistic_over)
```

```
##
## Call:
## glm(formula = as.numeric(unlist(oversampled_train_data[1])) ~
##     ., family = "binomial", data = oversampled_train_data[-1])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8605  -1.0536  -0.2199   1.0291   1.9388
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.333e+01  9.174e+02   0.015 0.988411
## Year              -1.681e-02  7.643e-03  -2.199 0.027843 *
## acousticness      -3.065e+00  5.064e-01  -6.053 1.42e-09 ***
## danceability      -1.312e-01  6.356e-01  -0.206 0.836464
## duration_ms        7.359e-06  1.057e-06   6.960 3.40e-12 ***
## energy             2.447e-01  6.750e-01   0.363 0.716935
## instrumentalness -9.579e-01  5.149e-01  -1.860 0.062865 .
## key1               2.466e-01  3.288e-01   0.750 0.453274
## key2               4.924e-02  2.674e-01   0.184 0.853877
## key3              -5.849e-01  5.038e-01  -1.161 0.245649
## key4              -6.562e-02  3.216e-01  -0.204 0.838306
## key5               6.072e-01  3.054e-01   1.988 0.046778 *
## key6               1.550e-01  3.479e-01   0.445 0.656044
## key7              -5.468e-01  2.850e-01  -1.919 0.055034 .
## key8               1.999e-01  3.852e-01   0.519 0.603903
## key9               6.610e-02  2.691e-01   0.246 0.805927
## key10             -6.165e-01  3.826e-01  -1.611 0.107113
## key11              6.273e-01  3.023e-01   2.075 0.038007 *
## loudness          -9.879e-02  4.489e-02  -2.200 0.027772 *
## mode1              7.647e-02  1.585e-01   0.483 0.629399
## tempo              1.764e-03  2.643e-03   0.667 0.504604
## time_signature3    1.641e+01  9.172e+02   0.018 0.985729
## time_signature4    1.705e+01  9.172e+02   0.019 0.985171
## time_signature5    2.038e+00  1.080e+03   0.002 0.998494
## valence           1.539e+00  3.993e-01   3.854 0.000116 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1525.7  on 1100  degrees of freedom
## Residual deviance: 1323.9  on 1076  degrees of freedom
## AIC: 1373.9
##
## Number of Fisher Scoring iterations: 14
```

Using the oversampled dataset, the full logistic model now determines the following variables to be significant: year, acousticness, duration, key5, key11, loudness, and valence. This is quite a difference compared to the previous logistic model based on the original dataset, which only listed 3 variables (acousticness, duration, and valence) as significant.

Predictions for the Oversampled Logistic Model

```
# Computing predictions

logistic_over_predictions_full = predict(logistic_over, newdata = test_set[,c(-1, -2)],
                                         type = "response")
```

Below are the predictions as computed by the oversampled, full logistic model created above, using different thresholds of classification. The same thresholds that were explored in the original-dataset full logistic model will he highlighted again: 0.2, 0.3, and 0.4

Threshold = 0.2

```
# Threshold = 0.2

logistic_over_predictions_full_02 = ifelse(logistic_over_predictions_full > 0.2, 1, 0)
logistic_over_accuracy_full_02 = sum(logistic_over_predictions_full_02 == test_set[2]) /
                                     dim(test_set[2])[1]

table(test_set$IsWinner, logistic_over_predictions_full_02)
```

```
##    logistic_over_predictions_full_02
##       0   1
##   0  22 128
##   1   1  23
```

```
false_positive_logistic_over_full_02 = table(test_set$IsWinner,
                                             logistic_over_predictions_full_02)[3]
negative_logistic_over_full_02 = table(test_set$IsWinner,
                                       logistic_over_predictions_full_02)[1] +
                                       table(test_set$IsWinner,
                                             logistic_over_predictions_full_02)[3]
typeIerror_logistic_over_full_02 = false_positive_logistic_over_full_02 /
                                   negative_logistic_over_full_02

typeIerror_logistic_over_full_02
```

```
## [1] 0.8533333
```

```
true_positive_logistic_over_full_02 = table(test_set$IsWinner,
                                            logistic_over_predictions_full_02)[4]
positive_logistic_over_full_02 = table(test_set$IsWinner,
                                       logistic_over_predictions_full_02)[2] +
                                       table(test_set$IsWinner,
                                             logistic_over_predictions_full_02)[4]
sensitivity_logistic_over_full_02 = true_positive_logistic_over_full_02 /
                                    positive_logistic_over_full_02

sensitivity_logistic_over_full_02
```

```
## [1] 0.9583333
```

At threshold 0.2, the Type I Error (False Positive Rate) was 0.853, meaning 85.3% of the non-nominated songs were misclassified as nominated. This is a fairly large error.

The Sensitivity (True Positive Rate) was .958, a fantastic value, showing the the oversampled full logistic model correctly classified 95.8% of the nominated songs. However, knowing the value of the Type I Error, it is clear that this value is so high because the model is classifying a vast majority of the data points as nominated. While it misses very few of the truly nominated songs, it misclassifies a large number of the non-nominated songs.

Threshold = 0.3

```
# Threshold = 0.3

logistic_over_predictions_full_03 = ifelse(logistic_over_predictions_full > 0.3, 1, 0)
logistic_over_accuracy_full_03 = sum(logistic_over_predictions_full_03 == test_set[2]) /
                                     dim(test_set[2])[1]

table(test_set$IsWinner, logistic_over_predictions_full_03)
```

```
##    logistic_over_predictions_full_03
##       0   1
##   0  46 104
##   1   5  19
```

```
false_positive_logistic_over_full_03 = table(test_set$IsWinner,
                                             logistic_over_predictions_full_03)[3]
negative_logistic_over_full_03 = table(test_set$IsWinner,
                                       logistic_over_predictions_full_03)[1] +
                                       table(test_set$IsWinner,
                                             logistic_over_predictions_full_03)[3]
typeIerror_logistic_over_full_03 = false_positive_logistic_over_full_03 /
                                   negative_logistic_over_full_03

typeIerror_logistic_over_full_03
```

```
## [1] 0.6933333
```

```
true_positive_logistic_over_full_03 = table(test_set$IsWinner,
                                            logistic_over_predictions_full_03)[4]
positive_logistic_over_full_03 = table(test_set$IsWinner,
                                       logistic_over_predictions_full_03)[2] +
                                       table(test_set$IsWinner,
                                             logistic_over_predictions_full_03)[4]
```

```r
sensitivity_logistic_over_full_03 = true_positive_logistic_over_full_03 /
                                    positive_logistic_over_full_03

sensitivity_logistic_over_full_03
```

```
## [1] 0.7916667
```

At a threshold of 0.3, the Type I Error improves to 0.693, misclassifying fewer non-nominated songs than at threshold 0.2, but the Sensitivity also decreases to 0.792 meaning the model is missing more of the nominated songs.

Threshold = 0.4

```r
# Threshold = 0.4

logistic_over_predictions_full_04 = ifelse(logistic_over_predictions_full > 0.4, 1, 0)
logistic_over_accuracy_full_04 = sum(logistic_over_predictions_full_04 == test_set[2]) /
                                     dim(test_set[2])[1]

table(test_set$IsWinner, logistic_over_predictions_full_04)
```

```
##    logistic_over_predictions_full_04
##      0  1
##   0 75 75
##   1  6 18
```

```r
false_positive_logistic_over_full_04 = table(test_set$IsWinner,
                                             logistic_over_predictions_full_04)[3]
negative_logistic_over_full_04 = table(test_set$IsWinner,
                                       logistic_over_predictions_full_04)[1] +
                                       table(test_set$IsWinner,
                                             logistic_over_predictions_full_04)[3]
typeIerror_logistic_over_full_04 = false_positive_logistic_over_full_04 /
                                   negative_logistic_over_full_04

typeIerror_logistic_over_full_04
```

```
## [1] 0.5
```

```r
true_positive_logistic_over_full_04 = table(test_set$IsWinner,
                                            logistic_over_predictions_full_04)[4]
positive_logistic_over_full_04 = table(test_set$IsWinner,
                                       logistic_over_predictions_full_04)[2] +
                                       table(test_set$IsWinner,
                                             logistic_over_predictions_full_04)[4]
sensitivity_logistic_over_full_04 = true_positive_logistic_over_full_04 /
                                    positive_logistic_over_full_04

sensitivity_logistic_over_full_04
```

```
## [1] 0.75
```

At a classification threshold of 0.4, the model's Type I Error decreases a fair amount to 0.5, but the Sensitivity also decreases slightly to 0.75. In comparison to the full logistic model using the original dataset, which at this threshold obtained the worst Sensitivity value of 0, the oversampled full logistic model performs much better (again, at this threshold). The ROC curve will exhibit how well the model performs overall.

ROC Curve

```
# ROC curve

roc.out <- roc(test_set$IsWinner, logistic_over_predictions_full)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

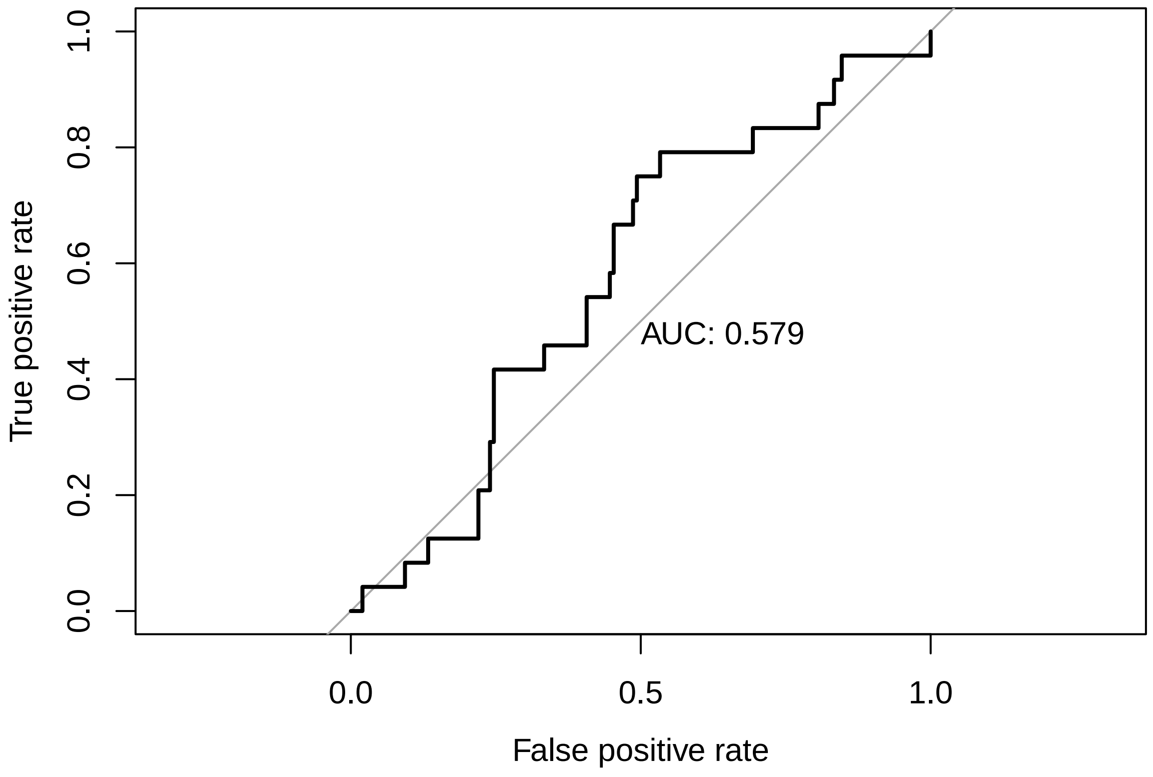

Figure 13: ROC Curve of the Full Oversampled Logistic Model

```
auc(roc.out)
```

```
## Area under the curve: 0.5792
```

Overall, the ROC curve shows that the full oversampled logistic model produces a diagnostic accuracy of 0.579, which is slightly lower than the ROC curve produced by the full logistic model using the original data.

Stepwise Variable Selection and Reduced Logistic Model of the Oversampled Dataset

Stepwise variable selection is again performed on the oversampled logistic model in an attempt to improve the model's classification accuracy by only utilizing the most insightful variables.

```
log_over_both =  stepAIC(logistic_over, direction = "both")
```

```
## Start:  AIC=1373.92
## as.numeric(unlist(oversampled_train_data[1])) ~ Year + acousticness +
##     danceability + duration_ms + energy + instrumentalness +
##     key + loudness + mode + tempo + time_signature + valence
##
##                    Df Deviance    AIC
## - danceability      1   1324.0 1372.0
## - energy            1   1324.0 1372.0
## - mode              1   1324.2 1372.2
## - tempo             1   1324.4 1372.4
## <none>                  1323.9 1373.9
## - instrumentalness  1   1327.5 1375.5
## - Year              1   1328.8 1376.8
## - loudness          1   1328.8 1376.8
## - key              11   1353.6 1381.6
## - time_signature    3   1337.7 1381.7
## - valence           1   1339.0 1387.0
## - acousticness      1   1366.4 1414.4
## - duration_ms       1   1381.4 1429.4
##
## Step:  AIC=1371.97
## as.numeric(unlist(oversampled_train_data[1])) ~ Year + acousticness +
##     duration_ms + energy + instrumentalness + key + loudness +
##     mode + tempo + time_signature + valence
##
##                    Df Deviance    AIC
## - energy            1   1324.1 1370.1
## - mode              1   1324.2 1370.2
## - tempo             1   1324.6 1370.6
## <none>                  1324.0 1372.0
## - instrumentalness  1   1327.5 1373.5
## + danceability      1   1323.9 1373.9
## - loudness          1   1328.8 1374.8
## - Year              1   1329.0 1375.0
## - key              11   1353.6 1379.6
## - time_signature    3   1337.7 1379.7
## - valence           1   1342.3 1388.3
## - acousticness      1   1366.5 1412.5
## - duration_ms       1   1382.5 1428.5
##
## Step:  AIC=1370.13
## as.numeric(unlist(oversampled_train_data[1])) ~ Year + acousticness +
##     duration_ms + instrumentalness + key + loudness + mode +
##     tempo + time_signature + valence
##
##                    Df Deviance    AIC
## - mode              1   1324.4 1368.4
## - tempo             1   1324.9 1368.9
## <none>                  1324.1 1370.1
## - instrumentalness  1   1327.6 1371.6
## + energy            1   1324.0 1372.0
## + danceability      1   1324.0 1372.0
## - Year              1   1329.4 1373.4
## - loudness          1   1330.5 1374.5
## - key              11   1353.7 1377.7
## - time_signature    3   1338.1 1378.1
## - valence           1   1344.8 1388.8
## - acousticness      1   1375.6 1419.6
## - duration_ms       1   1384.3 1428.3
##
## Step:  AIC=1368.36
## as.numeric(unlist(oversampled_train_data[1])) ~ Year + acousticness +
##     duration_ms + instrumentalness + key + loudness + tempo +
##     time_signature + valence
##
##                    Df Deviance    AIC
## - tempo             1   1325.2 1367.2
## <none>                  1324.4 1368.4
## - instrumentalness  1   1327.8 1369.8
## + mode              1   1324.1 1370.1
## + energy            1   1324.2 1370.2
## + danceability      1   1324.3 1370.3
## - Year              1   1329.8 1371.8
## - loudness          1   1330.7 1372.7
## - key              11   1353.7 1375.7
## - time_signature    3   1338.3 1376.3
## - valence           1   1345.0 1387.0
## - acousticness      1   1375.6 1417.6
## - duration_ms       1   1384.3 1426.3
##
## Step:  AIC=1367.17
## as.numeric(unlist(oversampled_train_data[1])) ~ Year + acousticness +
##     duration_ms + instrumentalness + key + loudness + time_signature +
##     valence
##
##                    Df Deviance    AIC
## <none>                  1325.2 1367.2
## + tempo             1   1324.4 1368.4
## - instrumentalness  1   1328.5 1368.5
## + danceability      1   1324.8 1368.8
## + mode              1   1324.9 1368.9
## + energy            1   1324.9 1368.9
## - Year              1   1330.3 1370.3
## - loudness          1   1331.2 1371.2
## - time_signature    3   1339.0 1375.0
## - key              11   1356.0 1376.0
## - valence           1   1345.9 1385.9
## - acousticness      1   1376.9 1416.9
## - duration_ms       1   1385.3 1425.3
```

```r
response_variable_over = as.numeric(unlist(oversampled_train_data[1]))

reduced_variables_over = as.matrix(oversampled_train_data[,c(2, 3, 5, 7, 8, 9, 12, 13)],
                                  ncol = 8)

reduced_train_data_over = matrix(c(
  response_variable_over,
  as.numeric(reduced_variables_over[,1]),
  as.numeric(reduced_variables_over[,2]),
  as.numeric(reduced_variables_over[,3]),
  as.numeric(reduced_variables_over[,4]),
  as.factor(reduced_variables_over[,5]),
  as.numeric(reduced_variables_over[,6]),
  as.factor(reduced_variables_over[,7]),
  as.numeric(reduced_variables_over[,8])
), ncol = 9)

head(reduced_variables_over)
```

```
##   Year   acousticness duration_ms instrumentalness key  loudness
## 1 "2010" "3.18e-04"   " 374453"   "9.98e-05"       "2"  " -4.928"
## 2 "1993" "4.65e-01"   " 718600"   "4.28e-06"       "2"  "-10.383"
## 3 "2020" "3.30e-01"   " 344693"   "2.29e-02"       "1"  "-10.268"
## 4 "2017" "1.18e-01"   " 201159"   "0.00e+00"       "5"  " -4.553"
## 5 "2012" "2.25e-03"   " 218190"   "4.12e-03"       "11" " -3.850"
## 6 "1995" "1.47e-02"   " 274000"   "3.91e-06"       "7"  "-10.247"
##   time_signature valence
## 1 "4"            "0.2330"
## 2 "4"            "0.3630"
## 3 "4"            "0.2560"
## 4 "4"            "0.7030"
## 5 "4"            "0.5880"
## 6 "4"            "0.3930"
```

The stepwise function determined the following variables to be the useful and cohesive: instrumentalness, year, loudness, time_signature, key, valence, acousticness, and duration.

```
colnames(reduced_train_data_over) = c("IsWinner", "Year", "acousticness",
                                      "duration_ms", "instrumentalness",
                                      "key", "loudness",
                                      "time_signature", "valence" )
logistic_reduced_over = glm(response_variable_over ~ Year + acousticness
                            + duration_ms + instrumentalness + key
                            + loudness
                            + time_signature + valence,
                            data = oversampled_train_data,
                            family = "binomial")
```

Predictions for the Oversampled Reduced Logistic Model

```
logistic_reduced_over_predictions = predict(logistic_reduced_over,
                                            newdata = test_set[,c(-1, -2)],
                                            type = "response")
```

In order to compare to the previously created models with ease, the reduced oversampled logistic model was tested on the same three thresholds that the other models were tested on.

Threshold = 0.2

```
# Threshold = 0.2

logistic_reduced_over_predictions_02 = ifelse(logistic_reduced_over_predictions >
                                              0.2, 1, 0)
logistic_reduced_over_accuracy_02 = sum(logistic_reduced_over_predictions_02 ==
                                          test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, logistic_reduced_over_predictions_02)
```

```
##    logistic_reduced_over_predictions_02
##       0   1
##   0  22 128
##   1   1  23
```

```
false_positive_logistic_reduced_over_02 = table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_02)[3]
negative_logistic_reduced_over_02 = table(test_set$IsWinner,
                                          logistic_reduced_over_predictions_02)[1] +
                                          table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_02)[3]
typeIerror_logistic_reduced_over_02 = false_positive_logistic_reduced_over_02 /
                                      negative_logistic_reduced_over_02

typeIerror_logistic_reduced_over_02
```

```
## [1] 0.8533333
```

```
true_positive_logistic_reduced_over_02 = table(test_set$IsWinner,
                                               logistic_reduced_over_predictions_02)[4]
positive_logistic_reduced_over_02 = table(test_set$IsWinner,
                                          logistic_reduced_over_predictions_02)[2] +
                                          table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_02)[4]
sensitivity_logistic_reduced_over_02 = true_positive_logistic_reduced_over_02 /
                                       positive_logistic_reduced_over_02

sensitivity_logistic_reduced_over_02
```

```
## [1] 0.9583333
```

At this threshold, the Type I Error was 0.853 and the Sensitivity was 0.958, which are the same exact values for the full oversampled logistic model at this threshold.

Threshold = 0.3

```
# Threshold = 0.3

logistic_reduced_over_predictions_03 = ifelse(logistic_reduced_over_predictions >
                                               0.3, 1, 0)
logistic_reduced_over_accuracy_03 = sum(logistic_reduced_over_predictions_03 ==
                                          test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, logistic_reduced_over_predictions_03)
```

```
##    logistic_reduced_over_predictions_03
##       0   1
##   0  48 102
##   1   5  19
```

```
false_positive_logistic_reduced_over_03 = table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_03)[3]
```

```
negative_logistic_reduced_over_03 = table(test_set$IsWinner,
                                          logistic_reduced_over_predictions_03)[1] +
                                          table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_03)[3]
typeIerror_logistic_reduced_over_03 = false_positive_logistic_reduced_over_03 /
                                      negative_logistic_reduced_over_03

typeIerror_logistic_reduced_over_03
```

```
## [1] 0.68
```

```
true_positive_logistic_reduced_over_03 = table(test_set$IsWinner,
                                               logistic_reduced_over_predictions_03)[4]
positive_logistic_reduced_over_03 = table(test_set$IsWinner,
                                          logistic_reduced_over_predictions_03)[2] +
                                          table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_03)[4]
sensitivity_logistic_reduced_over_03 = true_positive_logistic_reduced_over_03 /
                                       positive_logistic_reduced_over_03

sensitivity_logistic_reduced_over_03
```

```
## [1] 0.7916667
```

With a threshold of 0.3, when compared to threshold 0.2, the model again produced a lower Type I Error of 0.68 and a lower Sensitivity of 0.792. These values are very similar to the full oversampled logistic model at this threshold as well.

Threshold = 0.4

```
# Threshold = 0.4

logistic_reduced_over_predictions_04 = ifelse(logistic_reduced_over_predictions >
                                               0.4, 1, 0)
logistic_reduced_over_accuracy_04 = sum(logistic_reduced_over_predictions_04 ==
                                          test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, logistic_reduced_over_predictions_04)
```

```
##    logistic_reduced_over_predictions_04
##       0  1
##    0 75 75
##    1  6 18
```

```
false_positive_logistic_reduced_over_04 = table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_04)[3]
negative_logistic_reduced_over_04 = table(test_set$IsWinner,
                                          logistic_reduced_over_predictions_04)[1] +
                                          table(test_set$IsWinner,
                                                logistic_reduced_over_predictions_04)[3]
typeIerror_logistic_reduced_over_04 = false_positive_logistic_reduced_over_04 /
                                      negative_logistic_reduced_over_04

typeIerror_logistic_reduced_over_04
```

```
## [1] 0.5
```

```r
true_positive_logistic_reduced_over_04 = table(test_set$IsWinner,
                                              logistic_reduced_over_predictions_04)[4]
positive_logistic_reduced_over_04 = table(test_set$IsWinner,
                                         logistic_reduced_over_predictions_04)[2] +
                                         table(test_set$IsWinner,
                                               logistic_reduced_over_predictions_04)[4]
sensitivity_logistic_reduced_over_04 = true_positive_logistic_reduced_over_04 /
                                       positive_logistic_reduced_over_04

sensitivity_logistic_reduced_over_04
```

```
## [1] 0.75
```

Using the threshold of 0.4 again produced the same results to the full oversampled logistic model, with a Type I Error of 0.5 and a Sensitivity of 0.75. This suggests the models are extremely similar.

ROC Curve

```r
# ROC curve

roc.out <- roc(test_set$IsWinner, logistic_reduced_over_predictions)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

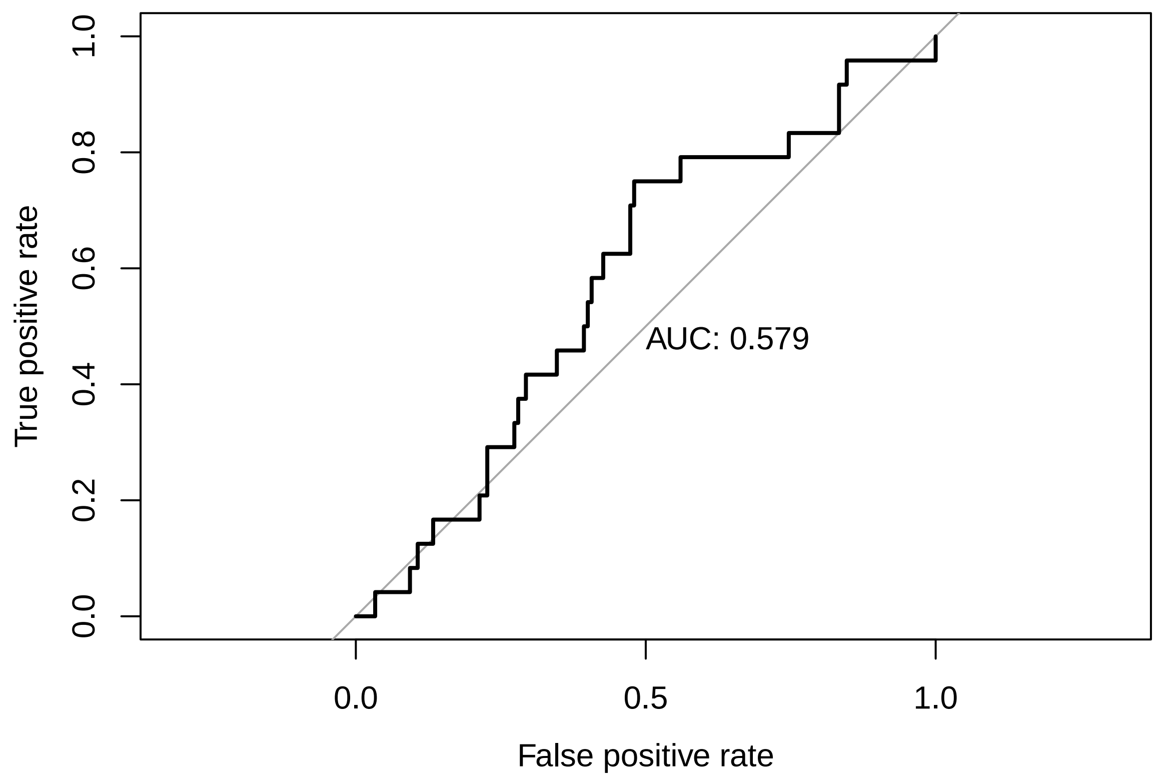

Figure 14: ROC Curve of the Reduced Oversampled Logistic Model

```
auc(roc.out)
```

```
## Area under the curve: 0.5792
```

The ROC curve adds to the evidence that the models are very similar, producing the exact same AUC value of the full oversampled logistic model: 0.579.

Discriminant Analysis

Discriminant analysis is a statistical technique based on the Bayes theorem used to solve classification problems. This technique assumes that the distributions of the independent variables conditioned to the dependent variable are normal. The Shapiro-Wilk test was computed in order to verify if the variables satisfied this condition. The null-hypothesis of this test is that a sample comes from a normal distribution. The null-hypothesis rejected if the p-value is less than 0.05, which means the probability of that sample to have been generated from a normal distribution is less that 5%.

```
shapiro.test(danceability[IsWinner == 0]) # Yes
```

```
##
##  Shapiro-Wilk normality test
##
## data:  danceability[IsWinner == 0]
## W = 0.99599, p-value = 0.1629
```

```
shapiro.test(danceability[IsWinner == 1]) # Yes
```

```
##
##  Shapiro-Wilk normality test
##
## data:  danceability[IsWinner == 1]
## W = 0.98983, p-value = 0.4586
```

```
shapiro.test(acousticness[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  acousticness[IsWinner == 0]
## W = 0.70527, p-value < 2.2e-16
```

```
shapiro.test(acousticness[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  acousticness[IsWinner == 1]
## W = 0.56506, p-value < 2.2e-16
```

```
shapiro.test(duration_ms[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  duration_ms[IsWinner == 0]
## W = 0.70806, p-value < 2.2e-16
```

```
shapiro.test(duration_ms[IsWinner == 1]) # No
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  duration_ms[IsWinner == 1]
## W = 0.84049, p-value = 1.513e-10
```

```r
shapiro.test(energy[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  energy[IsWinner == 0]
## W = 0.93579, p-value = 7.7e-15
```

```r
shapiro.test(energy[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  energy[IsWinner == 1]
## W = 0.91349, p-value = 4.31e-07
```

```r
shapiro.test(instrumentalness[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  instrumentalness[IsWinner == 0]
## W = 0.42748, p-value < 2.2e-16
```

```r
shapiro.test(instrumentalness[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  instrumentalness[IsWinner == 1]
## W = 0.48341, p-value < 2.2e-16
```

```r
shapiro.test(loudness[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  loudness[IsWinner == 0]
## W = 0.94738, p-value = 2.941e-13
```

```r
shapiro.test(loudness[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  loudness[IsWinner == 1]
## W = 0.90541, p-value = 1.513e-07
```

```r
shapiro.test(tempo[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  tempo[IsWinner == 0]
## W = 0.98439, p-value = 9.905e-06
```

```
shapiro.test(tempo[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  tempo[IsWinner == 1]
## W = 0.95867, p-value = 0.0005596
```

```
shapiro.test(valence[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  valence[IsWinner == 0]
## W = 0.97381, p-value = 1.724e-08
```

```
shapiro.test(valence[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  valence[IsWinner == 1]
## W = 0.97234, p-value = 0.009262
```

QQ plots of the conditioned independent variables

A visual tool was also used to check if the variables followed a normal distribution.

Danceability

```
par(mfrow = c(1, 2))

qqnorm(danceability[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(danceability[IsWinner == 0],lwd = 2, col = "red")

qqnorm(danceability[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(danceability[IsWinner == 1],lwd = 2, col = "red")
```

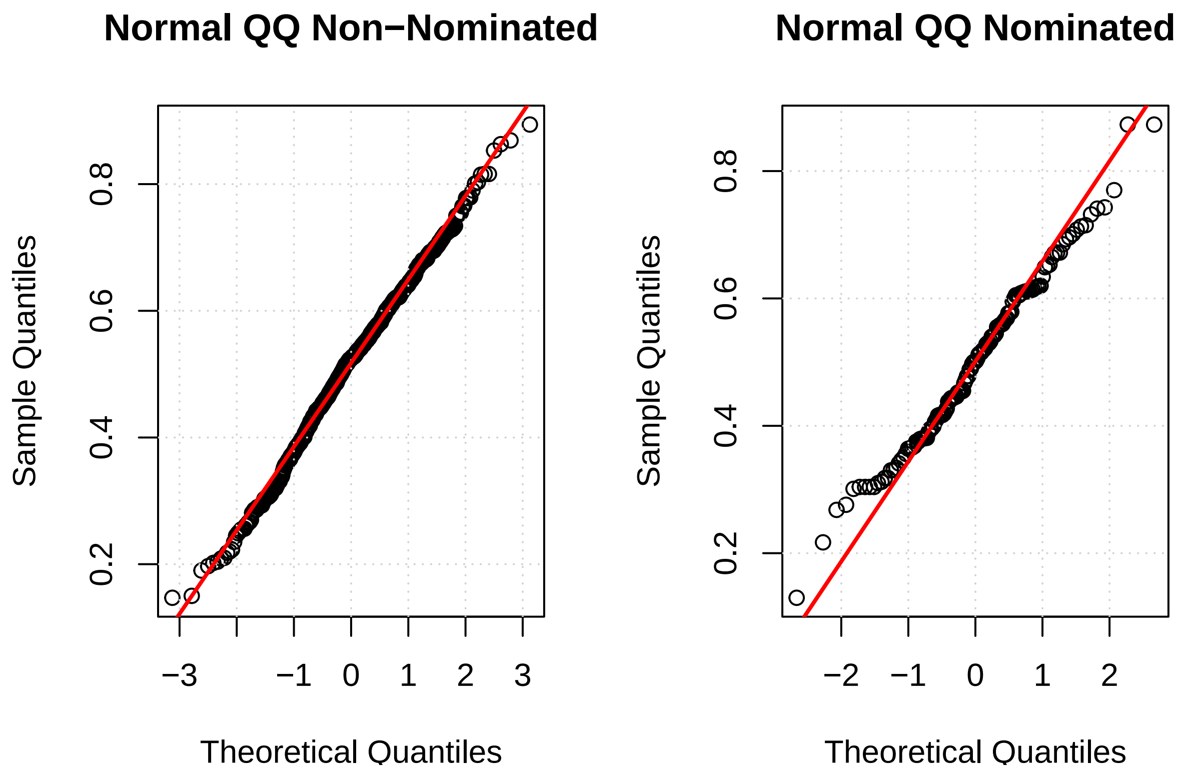

Figure 15: QQ Plots of Danceability conditioned to the dependent variable

Danceability looks normal confirming the result of the test.

Acousticness

```
par(mfrow = c(1, 2))

qqnorm(acousticness[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(acousticness[IsWinner == 0],lwd = 2, col = "red")

qqnorm(acousticness[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(acousticness[IsWinner == 1],lwd = 2, col = "red")
```

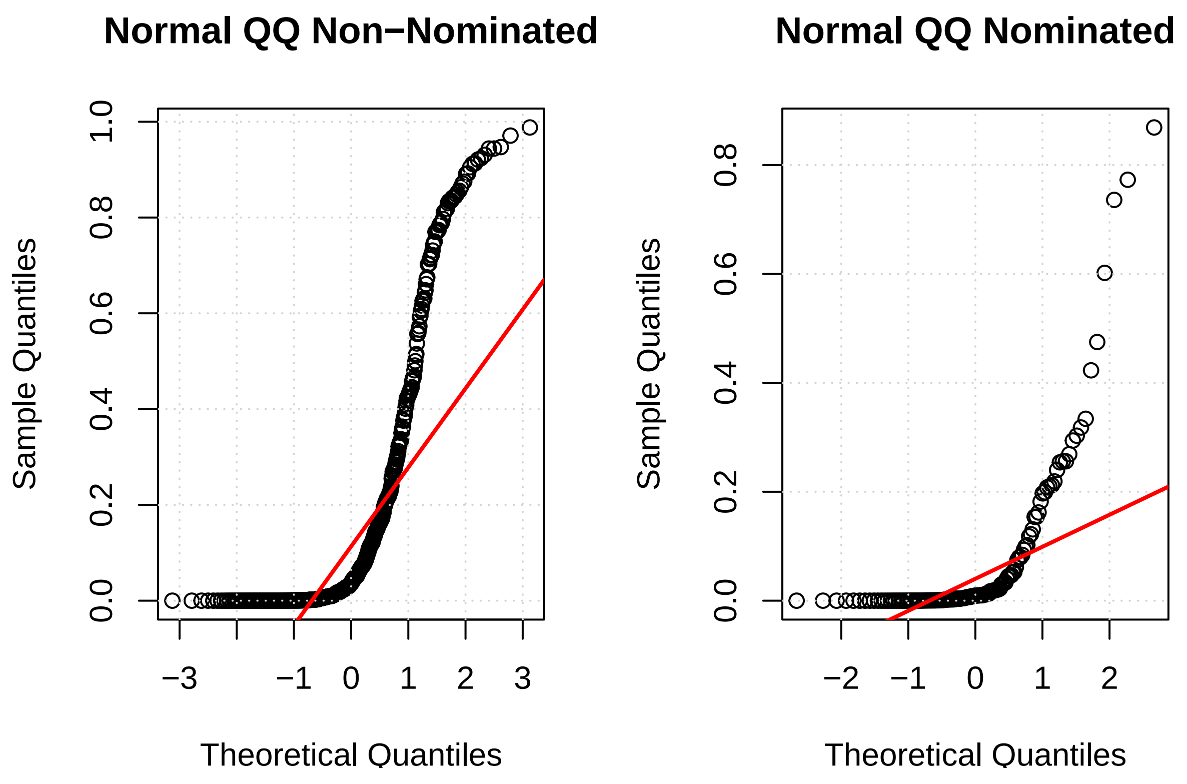

Figure 16: QQ Plots of Acousticness conditioned to the dependent variable

Acousticness is heavily not normal, following an S-shape.

Duration

```
par(mfrow = c(1, 2))

qqnorm(duration_ms[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(duration_ms[IsWinner == 0],lwd = 2, col = "red")

qqnorm(duration_ms[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(duration_ms[IsWinner == 1],lwd = 2, col = "red")
```

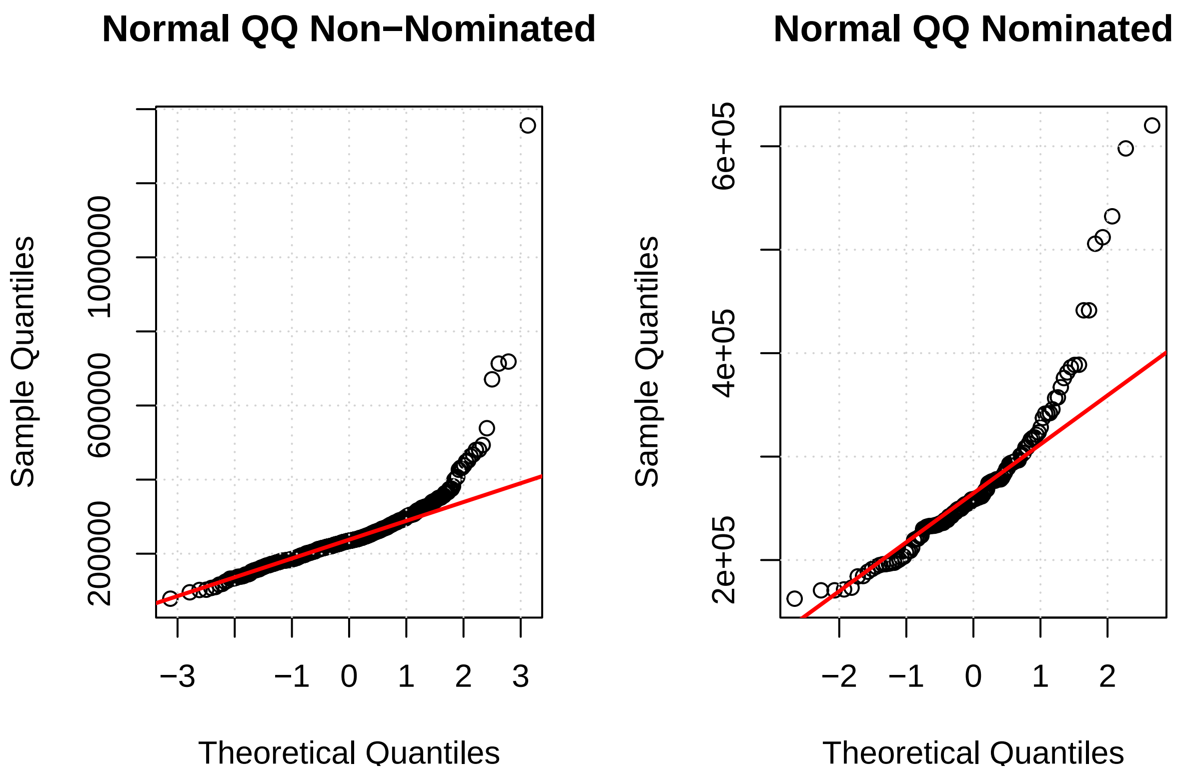

Figure 17: QQ Plots of Duration conditioned to the dependent variable

Duration shows a heavy right tail.

Energy

```
par(mfrow = c(1, 2))

qqnorm(energy[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(energy[IsWinner == 0],lwd = 2, col = "red")

qqnorm(energy[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(energy[IsWinner == 1],lwd = 2, col = "red")
```

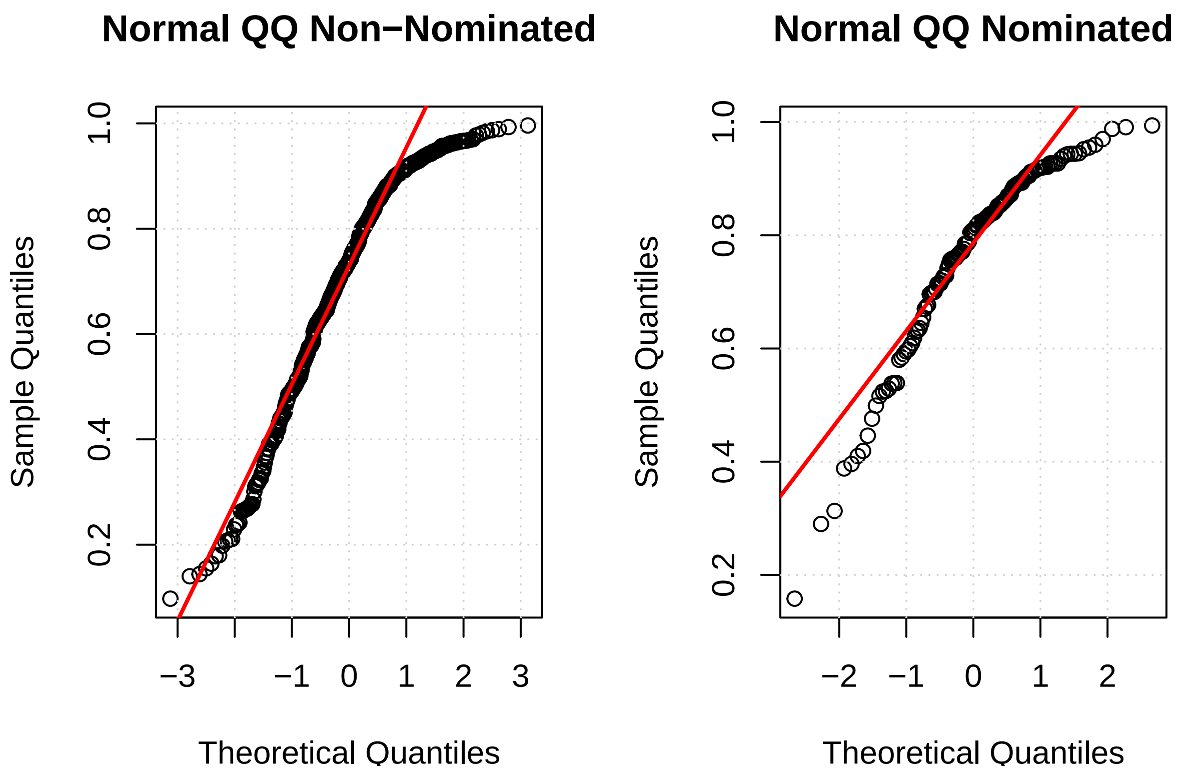

Figure 18: QQ Plots of Energy conditioned to the dependent variable

The energy feature shows not normal tails.

Instrumentalness

```
par(mfrow = c(1, 2))

qqnorm(instrumentalness[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(instrumentalness[IsWinner == 0],lwd = 2, col = "red")

qqnorm(instrumentalness[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(instrumentalness[IsWinner == 1],lwd = 2, col = "red")
```

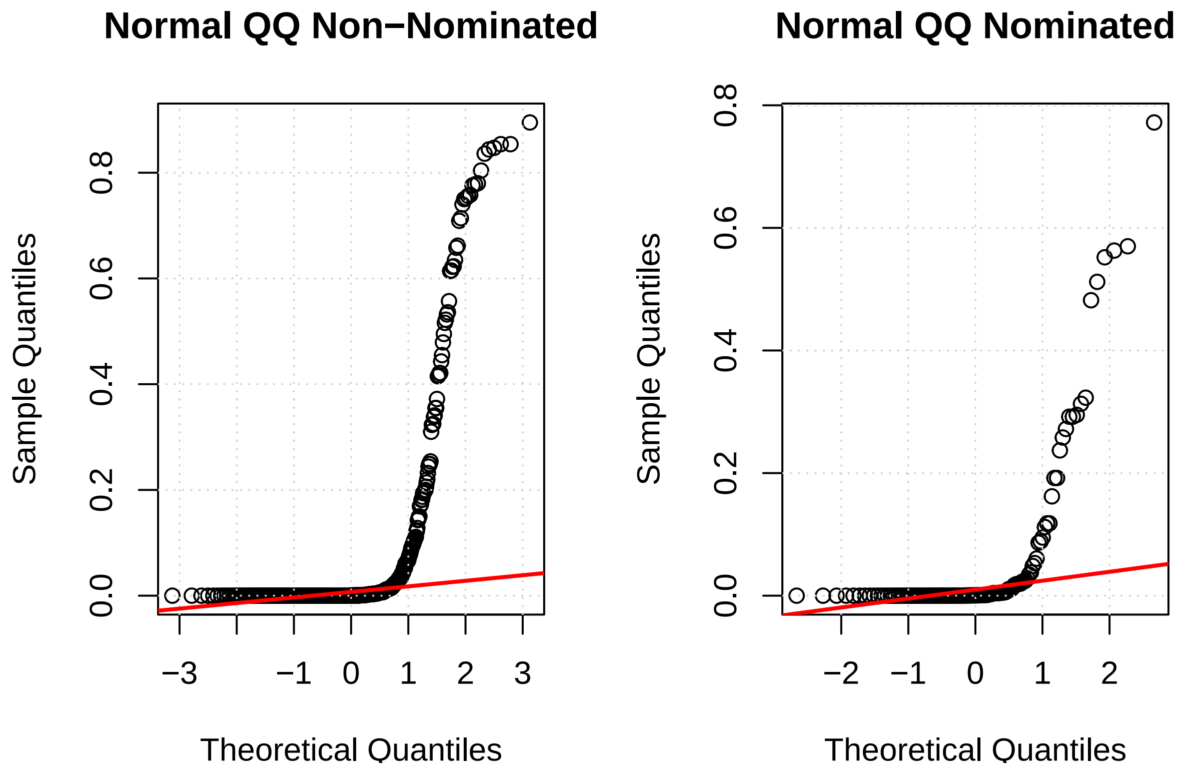

Figure 19: QQ Plots of Instrumentalness conditioned to the dependent variable

Instrumentalness shows a very heavy right tail.

Loudness

```
par(mfrow = c(1, 2))

qqnorm(loudness[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(loudness[IsWinner == 0],lwd = 2, col = "red")

qqnorm(loudness[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(loudness[IsWinner == 1],lwd = 2, col = "red")
```

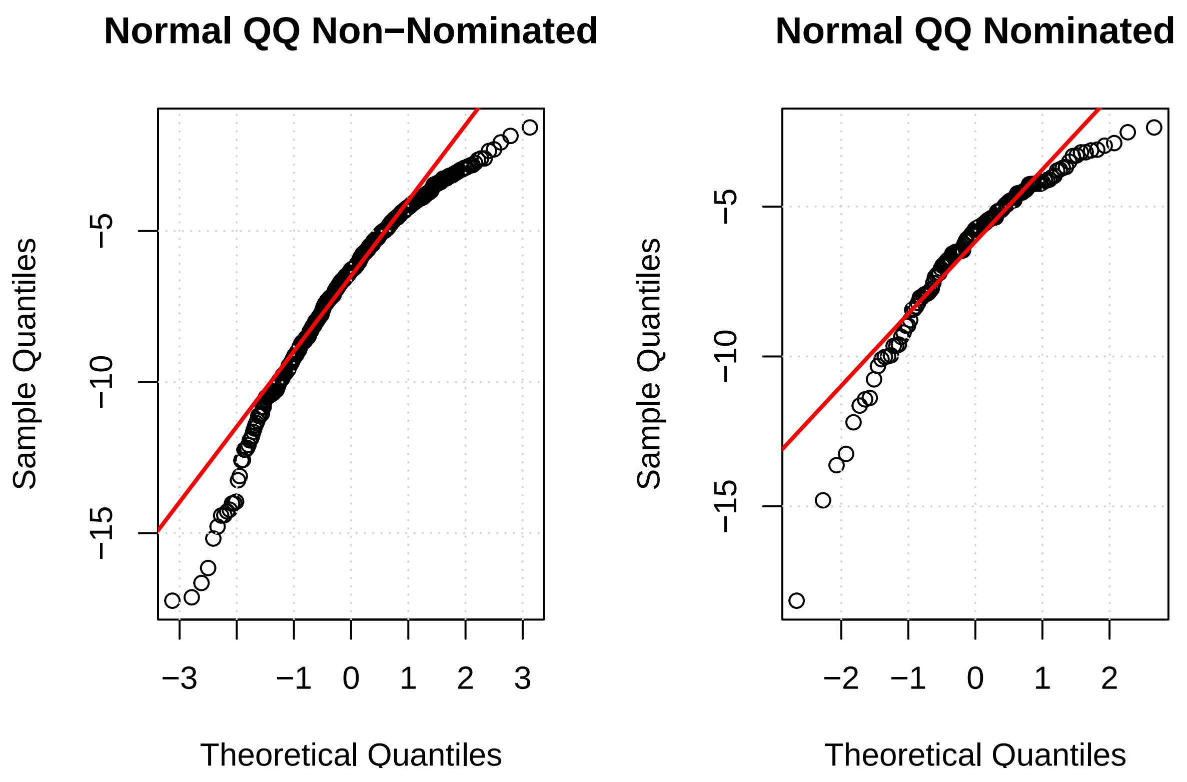

Figure 20: QQ Plots of Loudness conditioned to the dependent variable

Loudness shows not normal tails.

Tempo

```r
par(mfrow = c(1, 2))

qqnorm(tempo[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(tempo[IsWinner == 0],lwd = 2, col = "red")

qqnorm(tempo[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(tempo[IsWinner == 1],lwd = 2, col = "red")
```

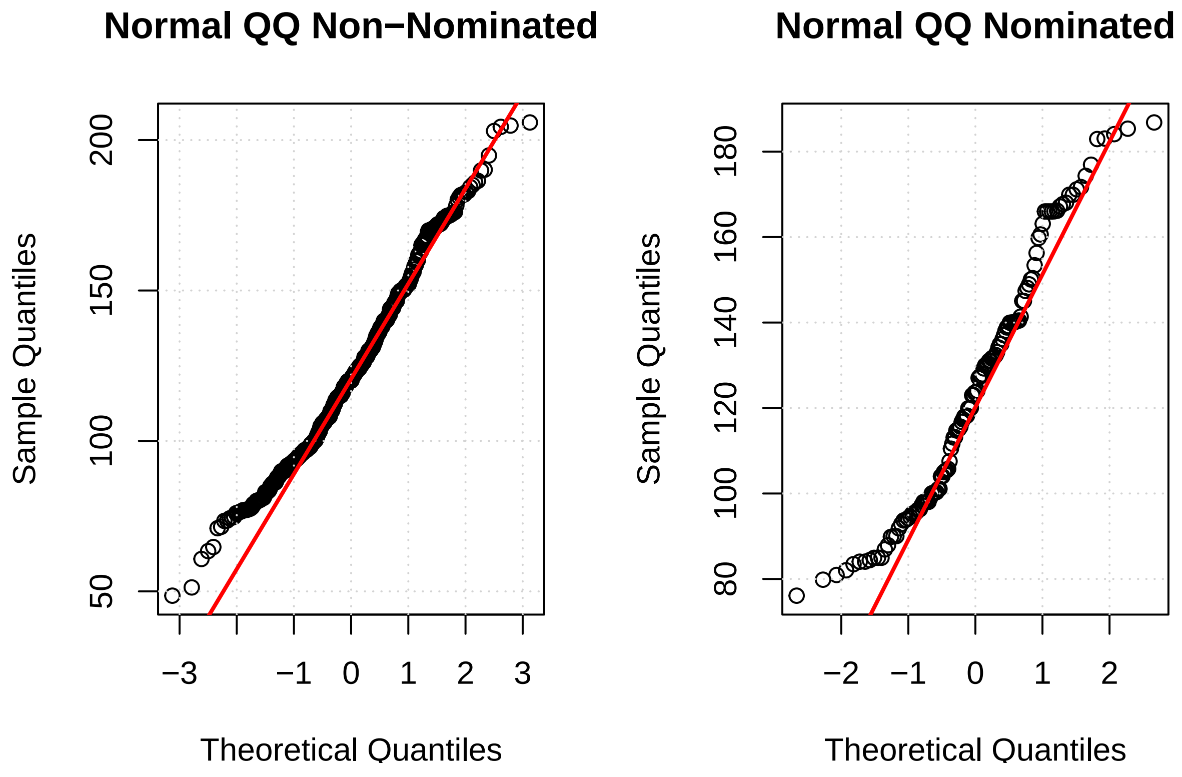

Figure 21: QQ Plots of Tempo conditioned to the dependent variable

The tempo variable seems close to a normal distribution.

Valence

```
# valence tails slightly not normal
par(mfrow = c(1, 2))

qqnorm(valence[IsWinner == 0], main="Normal QQ Non-Nominated")
grid()
qqline(valence[IsWinner == 0],lwd = 2, col = "red")

qqnorm(valence[IsWinner == 1], main="Normal QQ Nominated")
grid()
qqline(valence[IsWinner == 1],lwd = 2, col = "red")
```

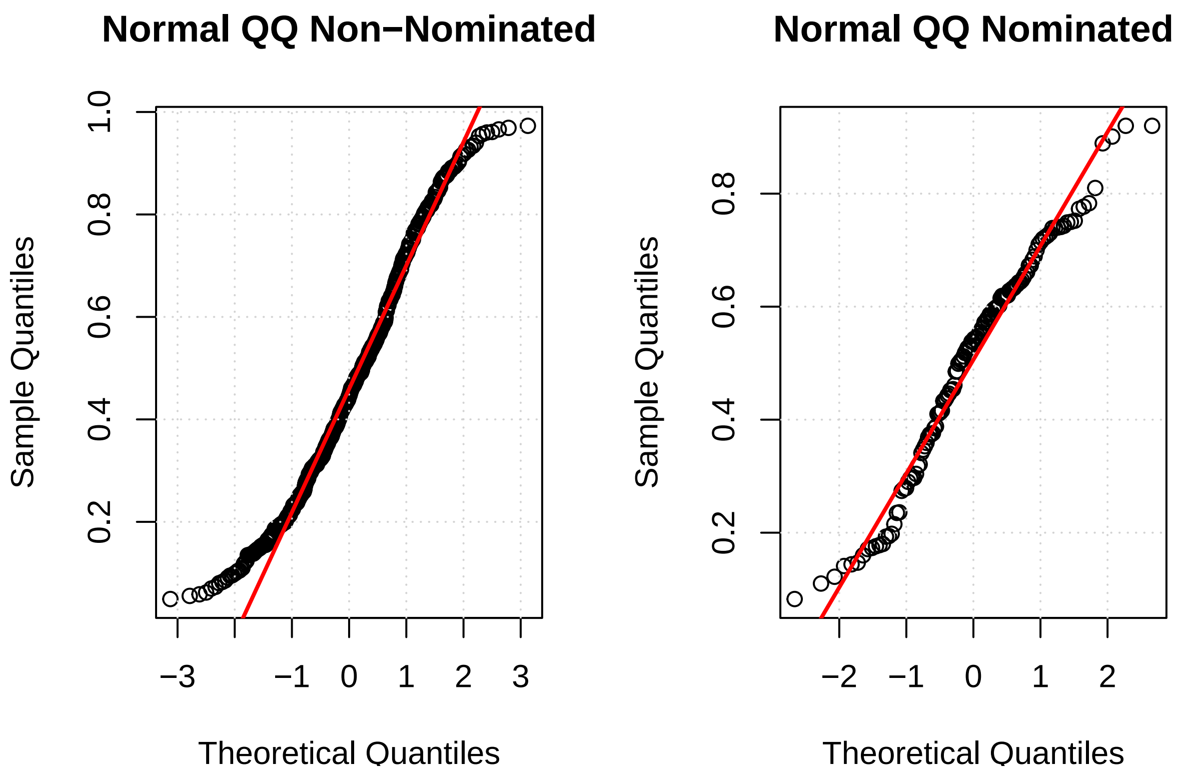

Figure 22: QQ Plots of Valence conditioned to the dependent variable

The valence variable seems close to a normal distribution, except for the tails.

Transformations

The only variable to have passed the test is danceability, so transformations were applied to the other variables to make them more normal-like. Specifically, the Box-Cox transformation was used to estimate the parameter.

The variables were then tested again. Examining the new output from the Shapiro test and the Q-Q plots, the variables were checked to see if the transformations succeeded in making the distributions more normal-like.

Acousticness

```
par(mfrow = c(1, 1))

b_acousticness <- boxcox(lm(acousticness ~ 1))
```

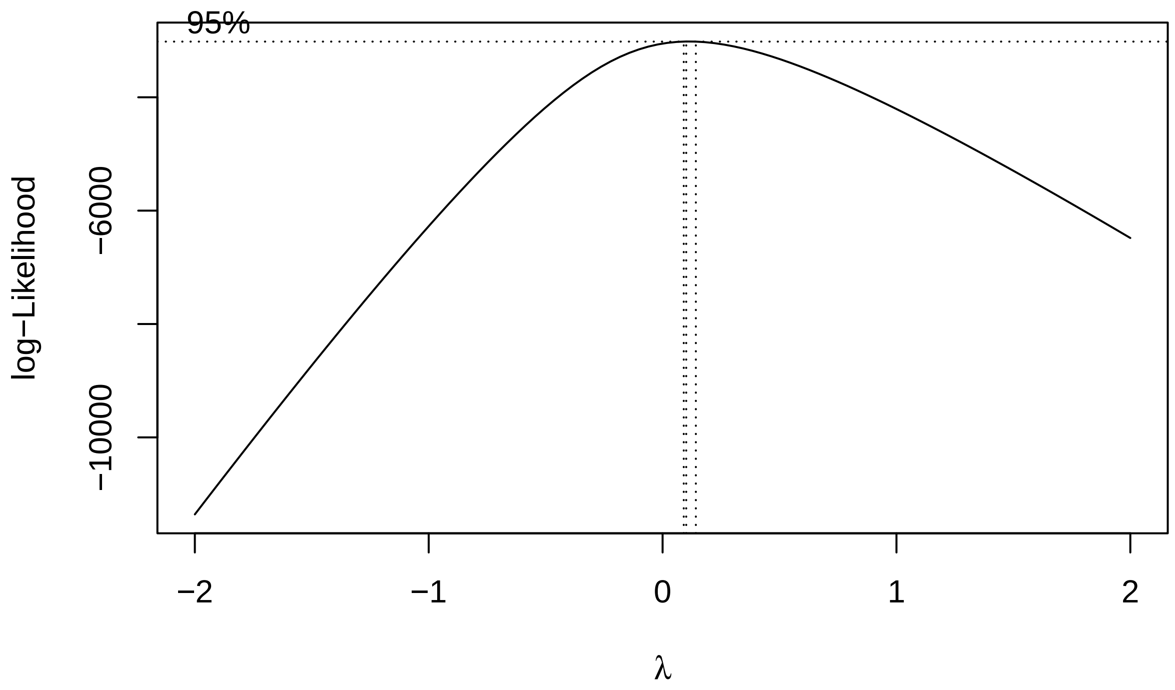

Figure 23: Distribution of the Acousticness Box-Cox Parameter

```
lambda <- b_acousticness$x[which.max(b_acousticness$y)]
acousticness_tran <- (acousticness ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

qqnorm(acousticness_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(acousticness_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(acousticness_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(acousticness_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(acousticness_tran[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  acousticness_tran[IsWinner == 0]
## W = 0.95483, p-value = 4.126e-12
```

```
shapiro.test(acousticness_tran[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  acousticness_tran[IsWinner == 1]
## W = 0.96627, p-value = 0.002542
```

```
# Original Q-Q Plots
qqnorm(acousticness[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(acousticness[IsWinner == 0],lwd = 2, col = "red")

qqnorm(acousticness[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(acousticness[IsWinner == 1],lwd = 2, col = "red")
```

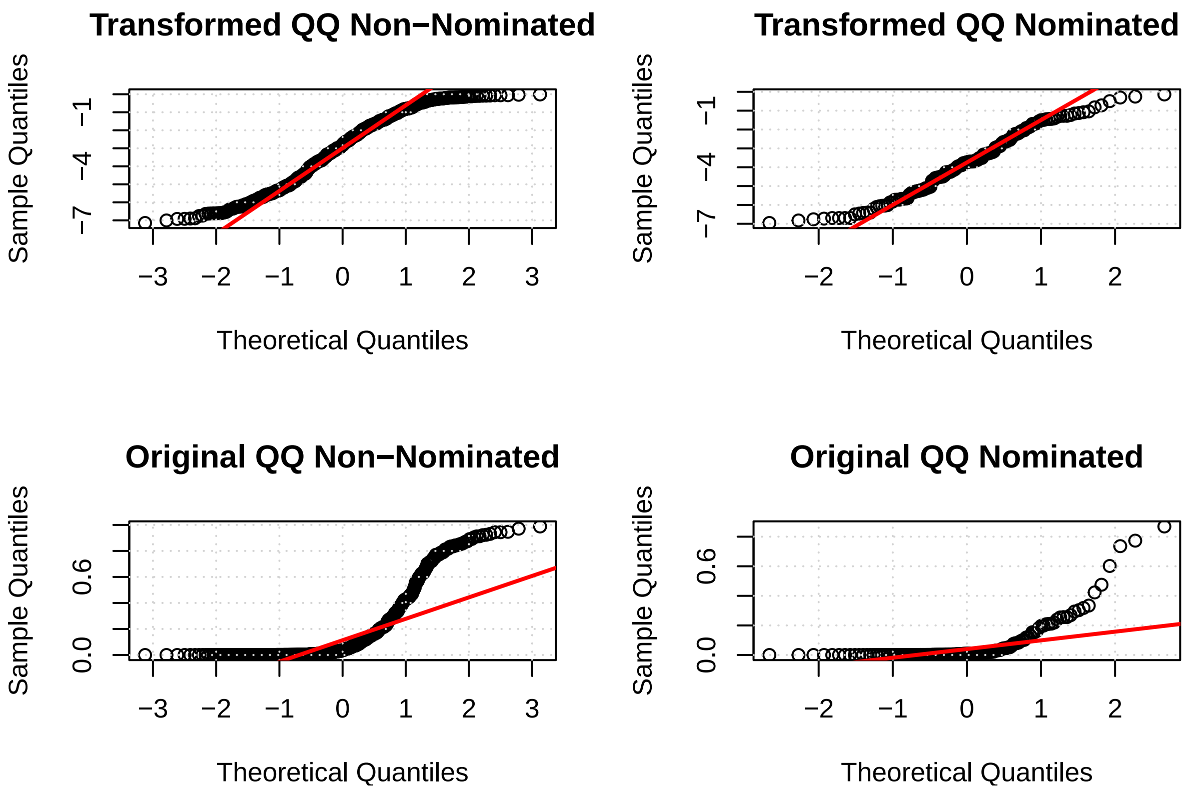

Figure 24: QQ Plots of Acousticness conditioned to the dependent variable (Original and Transformed)

The transformed variable did not pass the test, but the plots showed a good improvement, so the transformed variable will be used.

Duration

```
# duration_ms plots improved, test not passed

par(mfrow = c(1, 1))

b_duration_ms <- boxcox(lm(duration_ms ~ 1))
```

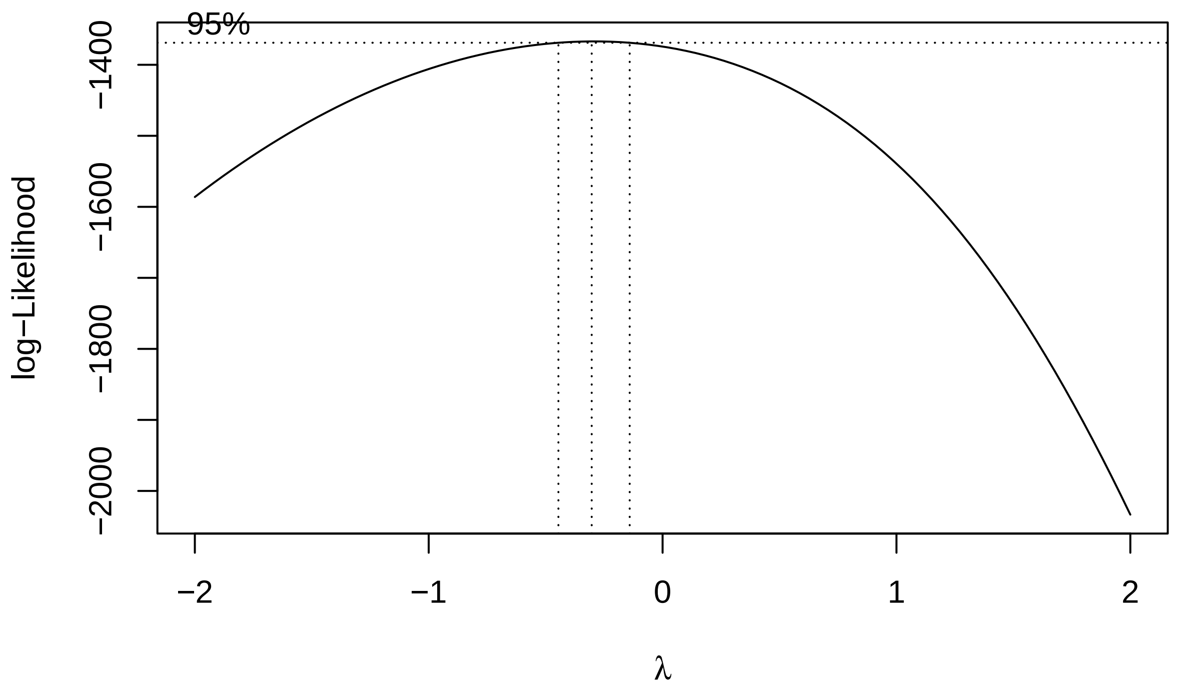

Figure 25: Distribution of the Duration Box-Cox Parameter

```
lambda <- b_duration_ms$x[which.max(b_duration_ms$y)]
duration_ms_tran <- (duration_ms ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

# Transformed data

qqnorm(duration_ms_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(duration_ms_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(duration_ms_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(duration_ms_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(duration_ms_tran[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  duration_ms_tran[IsWinner == 0]
## W = 0.96454, p-value = 2.059e-10
```

```r
shapiro.test(duration_ms_tran[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  duration_ms_tran[IsWinner == 1]
## W = 0.97209, p-value = 0.008766
```

```r
# Original data

qqnorm(duration_ms[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(duration_ms[IsWinner == 0],lwd = 2, col = "red")

qqnorm(duration_ms[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(duration_ms[IsWinner == 1],lwd = 2, col = "red")
```

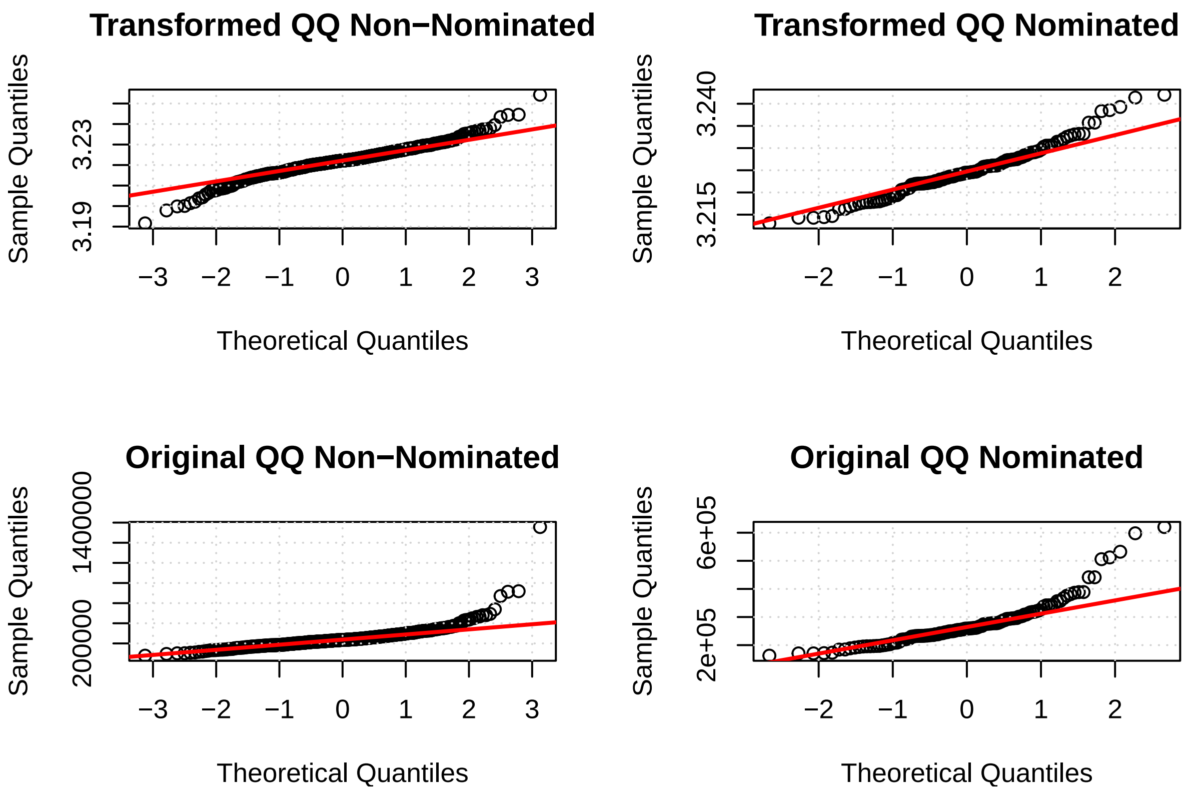

Figure 26: QQ Plots of Duration conditioned to the dependent variable (Original and Transformed)

The transformed variable did not pass the test but the plots show an improvement in the tails, so the transformed variable will be used.

Energy

```
par(mfrow = c(1, 1))

b_energy <- boxcox(lm(energy ~ 1))
```

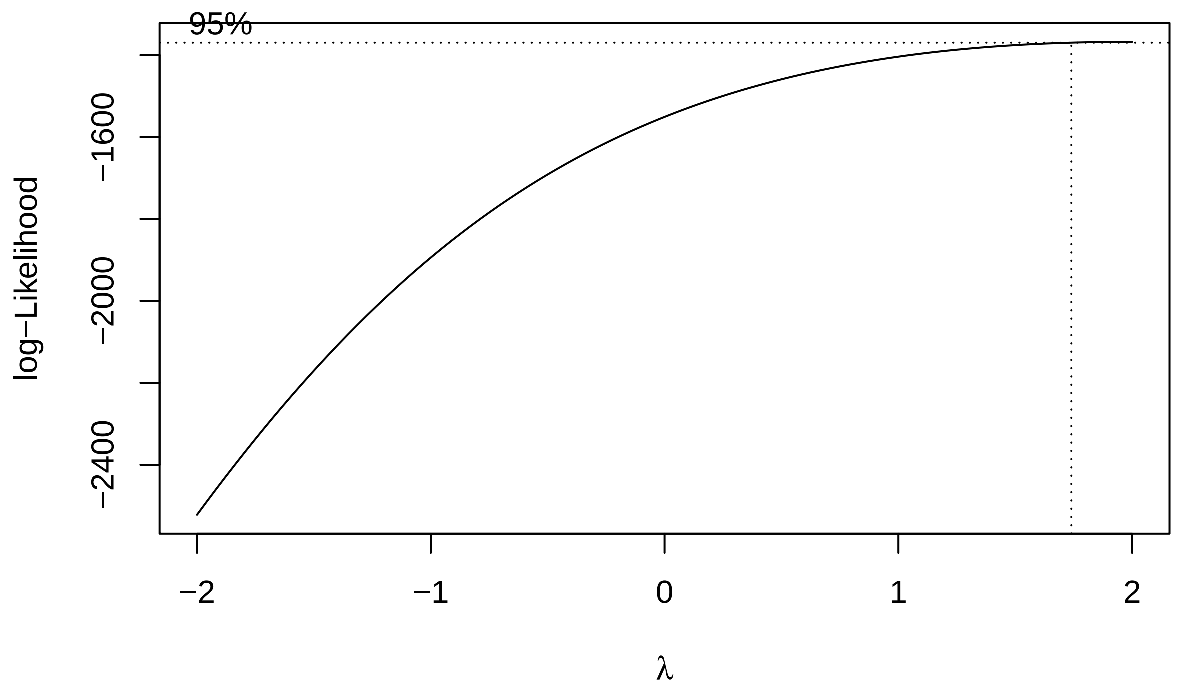

Figure 27: Distribution of the Energy Box-Cox Parameter

```
lambda <- b_energy$x[which.max(b_energy$y)]
energy_tran <- (energy ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

# Transformed data

qqnorm(energy_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(energy_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(energy_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(energy_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(energy_tran[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  energy_tran[IsWinner == 0]
## W = 0.96009, p-value = 3.178e-11
```

```
shapiro.test(energy_tran[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  energy_tran[IsWinner == 1]
## W = 0.96047, p-value = 0.0007932
```

```
# Original data

qqnorm(energy[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(energy[IsWinner == 0],lwd = 2, col = "red")

qqnorm(energy[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(energy[IsWinner == 1],lwd = 2, col = "red")
```

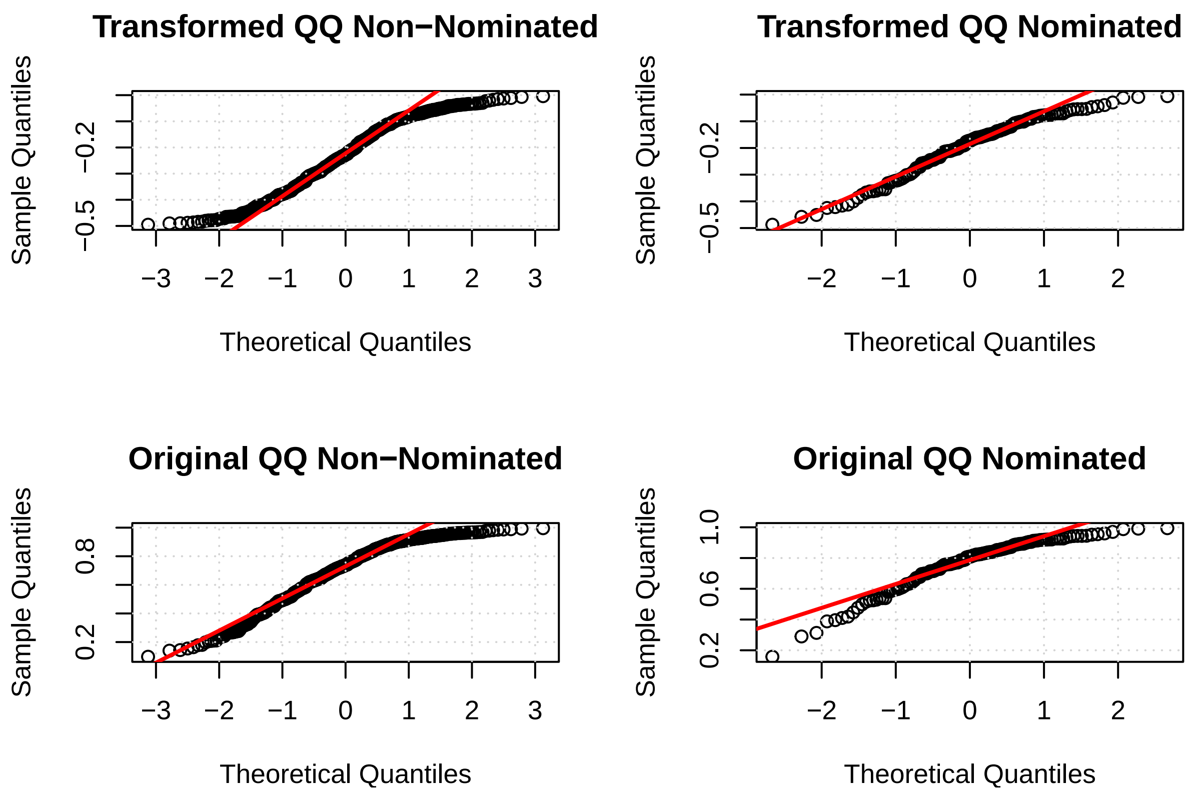

Figure 28: QQ Plots of Energy conditioned to the dependent variable (Original and Transformed)

The transformed variable did not pass the test and the plots don't show improvements, so the original values will be kept.

Instrumentalness

```
# instrumentalness can't apply the boxcox transformation because there are some 0's

par(mfrow = c(1, 1))

# Added a small value to overcome the issue of 0's
new_instrumentalness = instrumentalness + 1e-05

b_instrumentalness <- boxcox(lm(new_instrumentalness ~ 1))
```

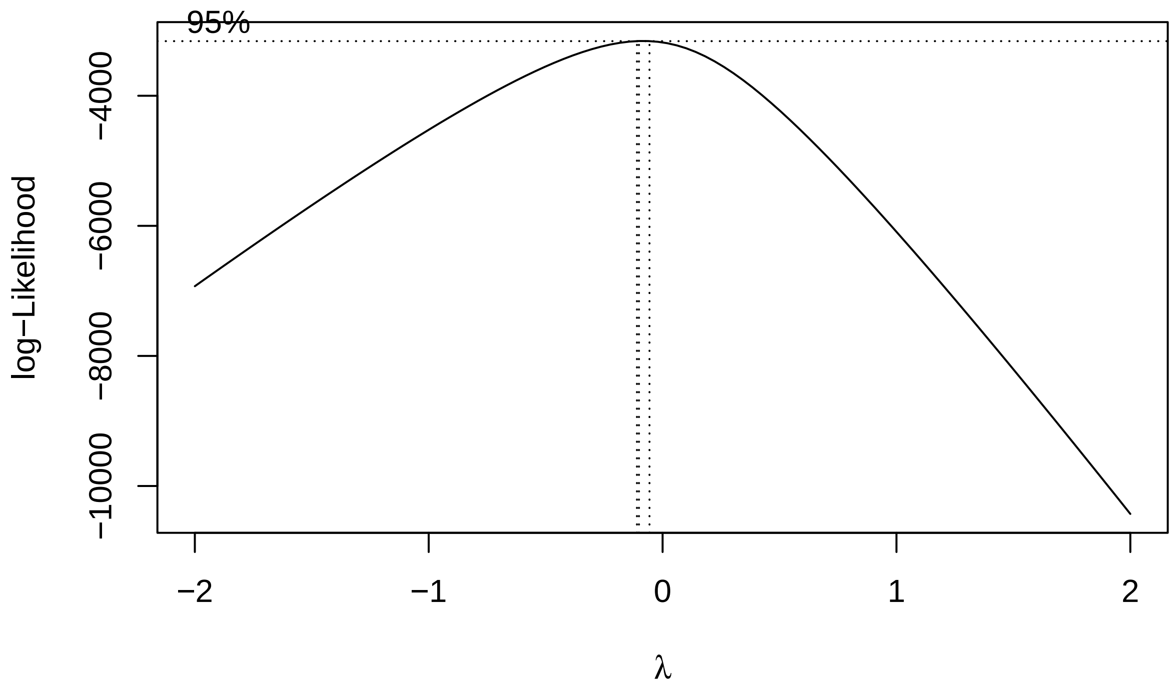

Figure 29: Distribution of the Instrumentalness Box-Cox Parameter

```
lambda <- b_instrumentalness$x[which.max(b_instrumentalness$y)]
instrumentalness_tran <- (new_instrumentalness ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

# Transformed data

qqnorm(instrumentalness_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(instrumentalness_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(instrumentalness_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(instrumentalness_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(instrumentalness_tran[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  instrumentalness_tran[IsWinner == 0]
## W = 0.88817, p-value < 2.2e-16
```

```
shapiro.test(instrumentalness_tran[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  instrumentalness_tran[IsWinner == 1]
## W = 0.90309, p-value = 1.132e-07
```

```
# Original data

qqnorm(instrumentalness[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(instrumentalness[IsWinner == 0],lwd = 2, col = "red")

qqnorm(instrumentalness[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(instrumentalness[IsWinner == 1],lwd = 2, col = "red")
```

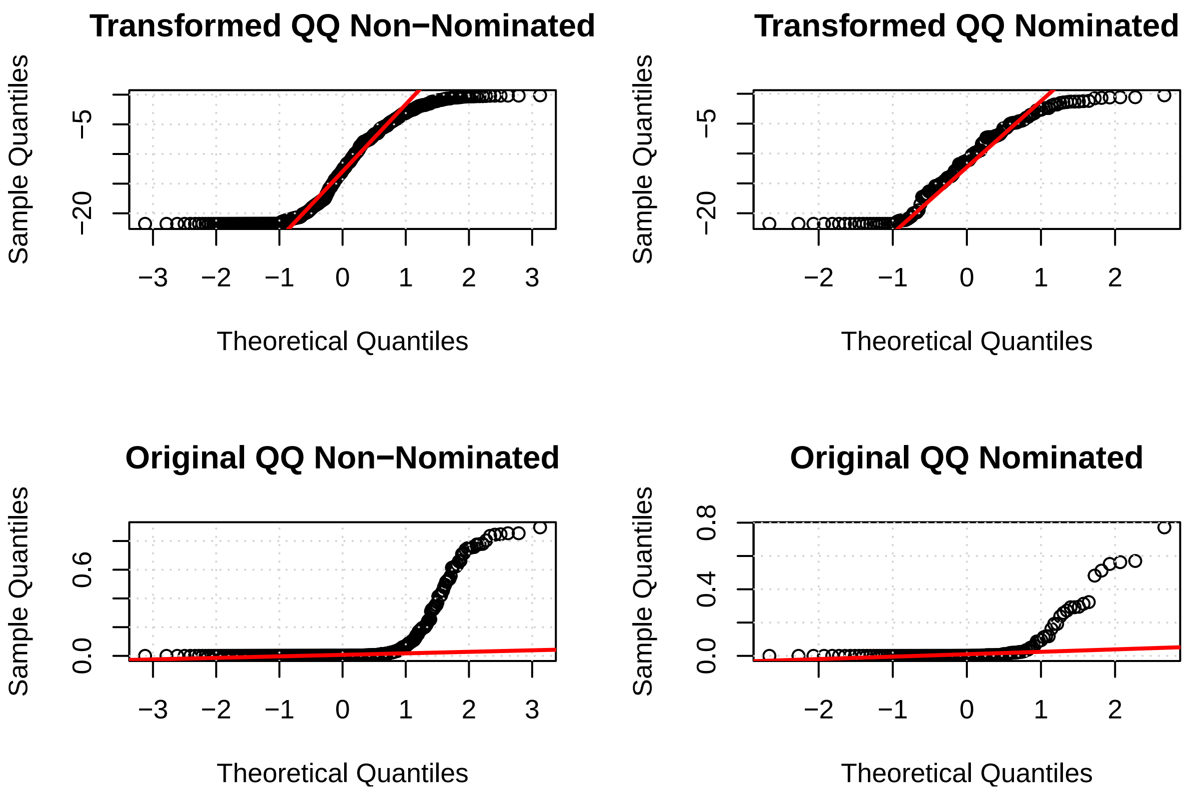

Figure 30: QQ Plots of Instrumentalness conditioned to the dependent variable (Original and Transformed)

The Box-Cox transformation could not be directly applied to the instrumentalness variable because of the presence of zeros, so a small value was added and then applied to it. The new variable did not pass the test and the plots did not improve, so the original data was kept.

Loudness

```
par(mfrow = c(1, 1))

new_loudness = loudness * (-1)

b_loudness <- boxcox(lm(new_loudness ~ 1))
```

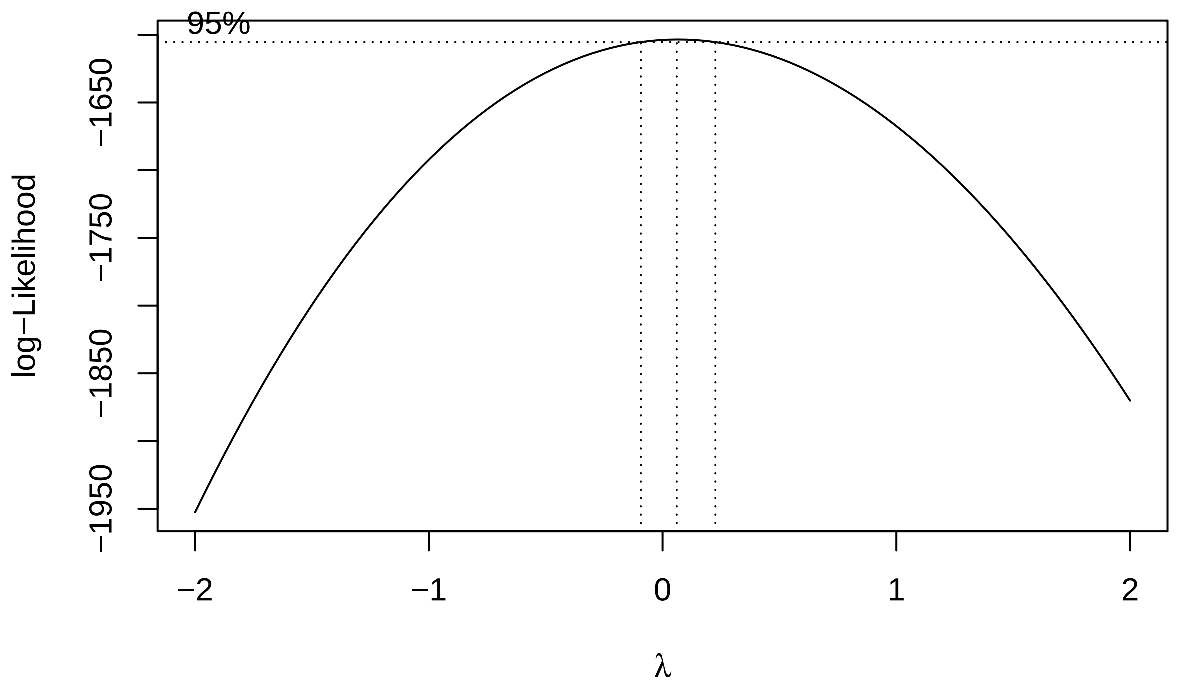

Figure 31: Distribution of the Loudness Box-Cox Parameter

```
lambda <- b_loudness$x[which.max(b_loudness$y)]
loudness_tran <- (new_loudness ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

# Transformed data

qqnorm(loudness_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(loudness_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(loudness_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(loudness_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(loudness_tran[IsWinner == 0]) # Yes
```

```
##
##  Shapiro-Wilk normality test
##
## data:  loudness_tran[IsWinner == 0]
## W = 0.99829, p-value = 0.8586
```

```
shapiro.test(loudness_tran[IsWinner == 1]) # Yes
```

```
##
##  Shapiro-Wilk normality test
##
## data:  loudness_tran[IsWinner == 1]
## W = 0.99337, p-value = 0.8045
```

```
# Original data

qqnorm(loudness[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(loudness[IsWinner == 0],lwd = 2, col = "red")

qqnorm(loudness[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(loudness[IsWinner == 1],lwd = 2, col = "red")
```

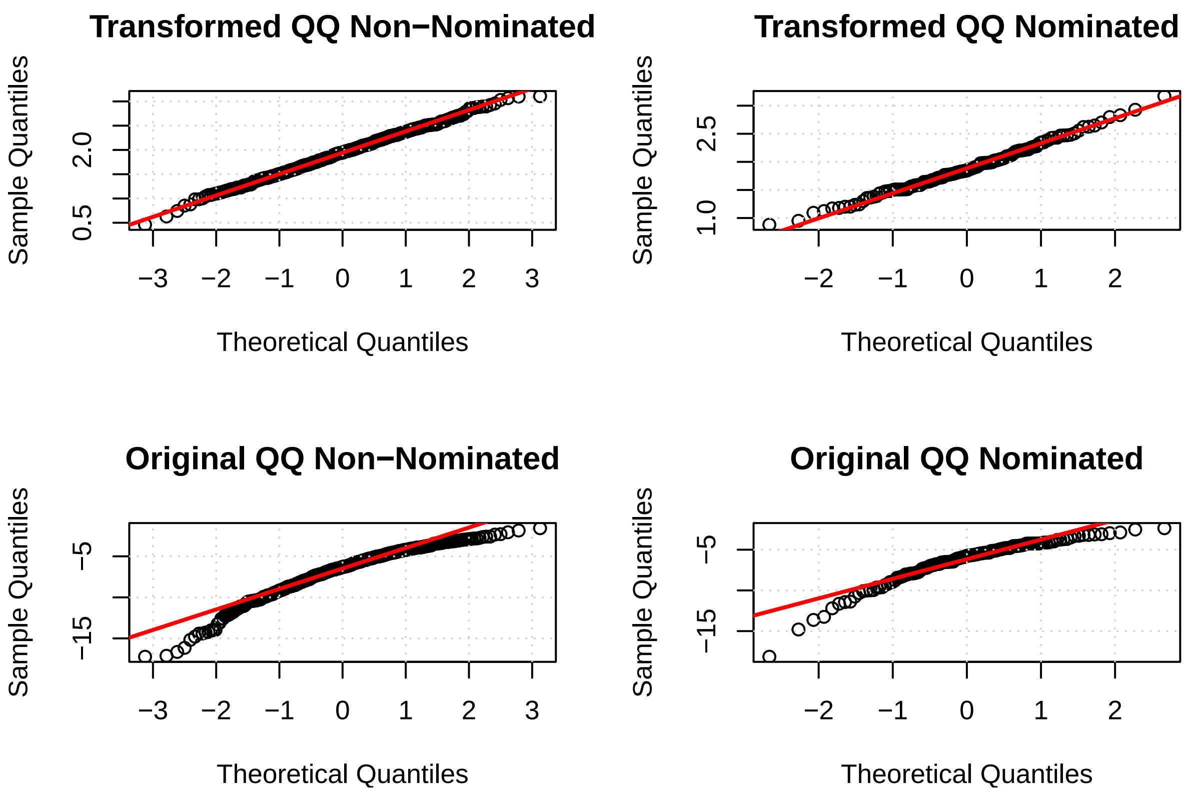

Figure 32: QQ Plots of Loudness conditioned to the dependent variable (Original and Transformed)

The Box-Cox transformation could also not be directly applied to the variable loudness because the variable takes negative values. As a workaround, the values were multiplied by -1 and then the transformation was applied. The new variable passed the test and the plots confirmed the result of the test. Since the variable was multiplied by -1, the interpretation of the variable will be mirrored.

Tempo

```
# tempo, slight improvement in the plots

par(mfrow = c(1, 1))

b_tempo <- boxcox(lm(tempo ~ 1))
```

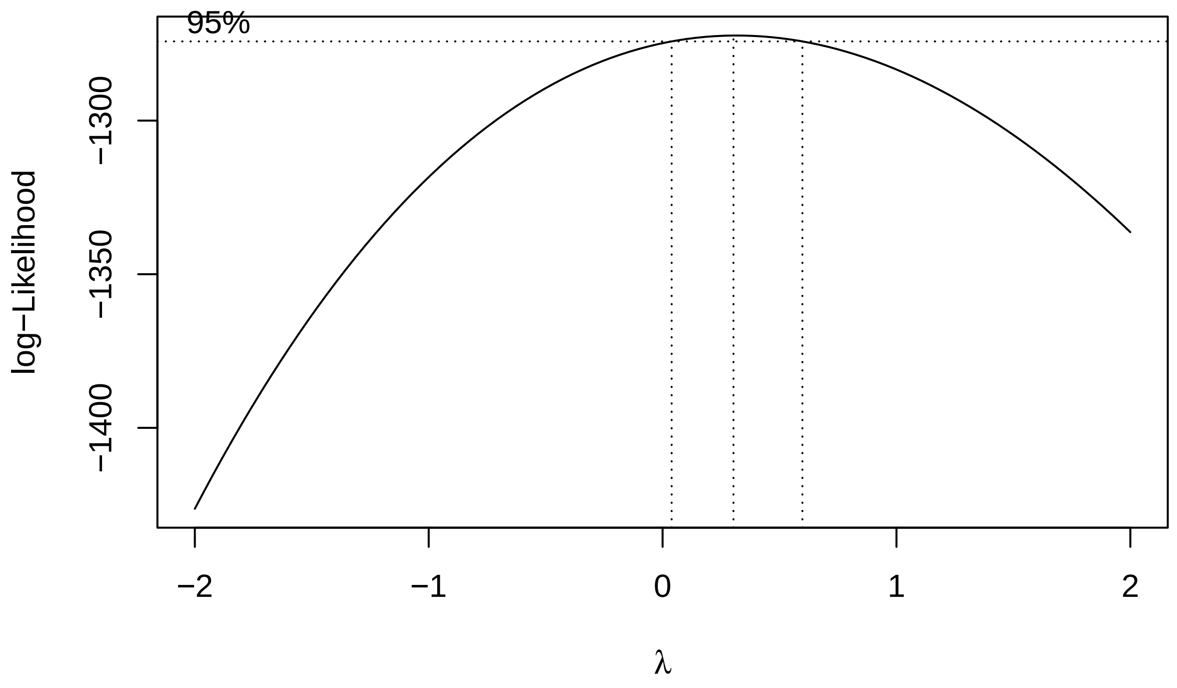

Figure 33: Distribution of the Tempo Box-Cox Parameter

```
lambda <- b_tempo$x[which.max(b_tempo$y)]
tempo_tran <- (tempo ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

# Transformed Data

qqnorm(tempo_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(tempo_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(tempo_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(tempo_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(tempo_tran[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  tempo_tran[IsWinner == 0]
## W = 0.99334, p-value = 0.01358
```

```
shapiro.test(tempo_tran[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  tempo_tran[IsWinner == 1]
## W = 0.96738, p-value = 0.003199
```

```
# Original data

qqnorm(tempo[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(tempo[IsWinner == 0],lwd = 2, col = "red")

qqnorm(tempo[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(tempo[IsWinner == 1],lwd = 2, col = "red")
```

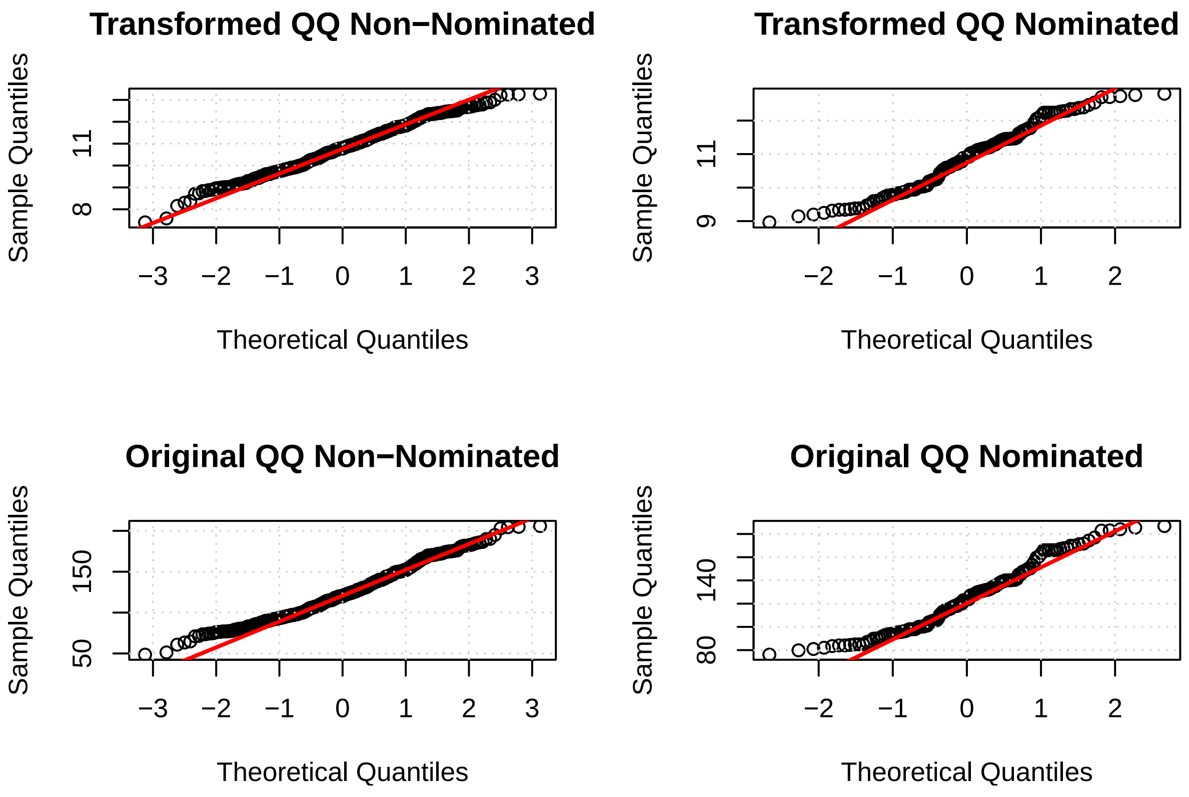

Figure 34: QQ Plots of Tempo conditioned to the dependent variable (Original and Transformed)

The transformed tempo variable showed slight improvement in the plots, but still did not pass the test, so we used the transformed variable.

Valence

```
# valence, pretty much the same

par(mfrow = c(1, 1))

b_valence <- boxcox(lm(valence ~ 1))
```

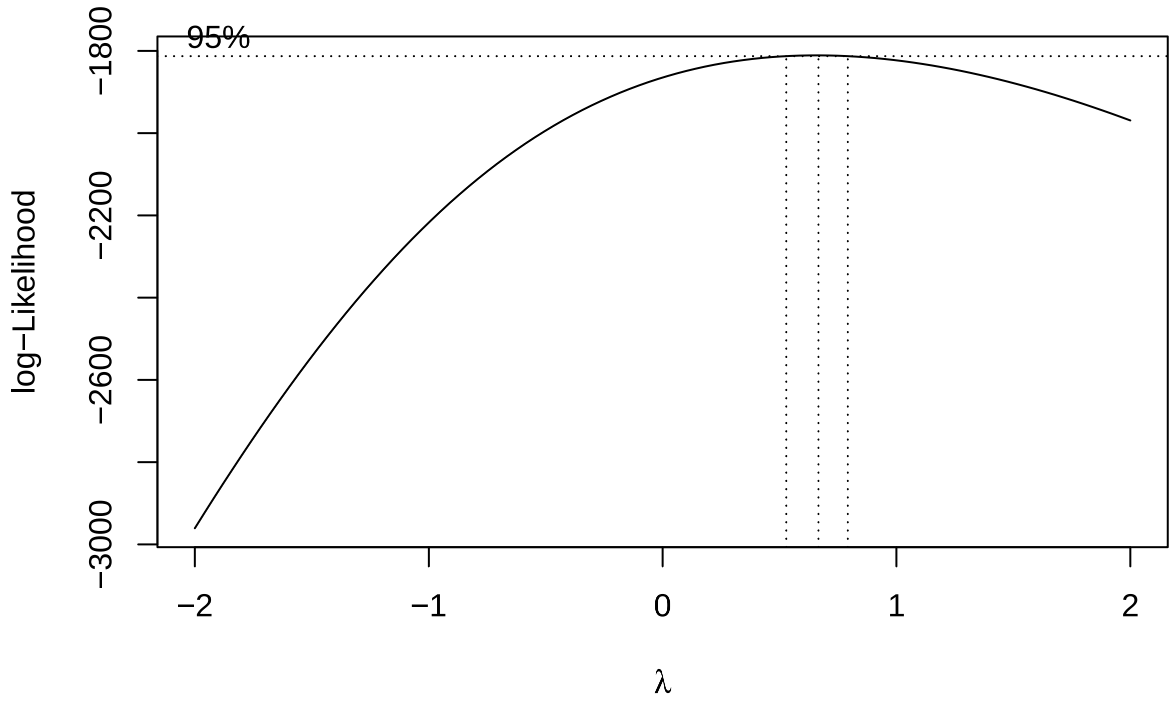

Figure 35: Distribution of the Valence Box-Cox Parameter

```
lambda <- b_valence$x[which.max(b_valence$y)]
valence_tran <- (valence ^ lambda - 1) / lambda
```

```
par(mfrow = c(2, 2))

# Transformed data

qqnorm(valence_tran[IsWinner == 0], main="Transformed QQ Non-Nominated")
grid()
qqline(valence_tran[IsWinner == 0],lwd = 2, col = "red")

qqnorm(valence_tran[IsWinner == 1], main="Transformed QQ Nominated")
grid()
qqline(valence_tran[IsWinner == 1],lwd = 2, col = "red")

shapiro.test(valence_tran[IsWinner == 0]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  valence_tran[IsWinner == 0]
## W = 0.98604, p-value = 3.22e-05
```

```
shapiro.test(valence_tran[IsWinner == 1]) # No
```

```
##
##  Shapiro-Wilk normality test
##
## data:  valence_tran[IsWinner == 1]
## W = 0.95626, p-value = 0.0003542
```

```
# Original data

qqnorm(valence[IsWinner == 0], main="Original QQ Non-Nominated")
grid()
qqline(valence[IsWinner == 0],lwd = 2, col = "red")

qqnorm(valence[IsWinner == 1], main="Original QQ Nominated")
grid()
qqline(valence[IsWinner == 1],lwd = 2, col = "red")
```

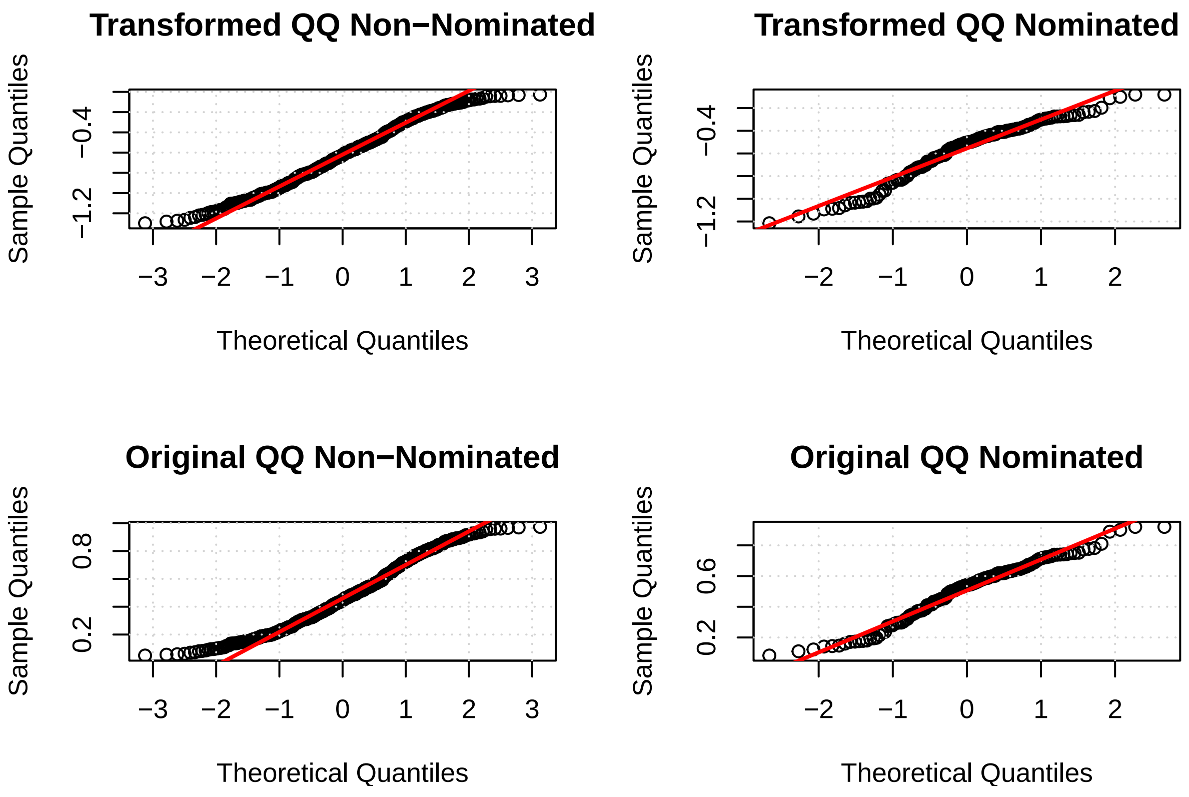

Figure 36: QQ Plots of Valence conditioned to the dependent variable (Original and Transformed)

The transformed variable did not change much, so the original data was kept.

Data transformation

```r
par(mfrow = c(1, 1))

b_data_acousticness <- boxcox(lm(data$acousticness ~ 1))
```

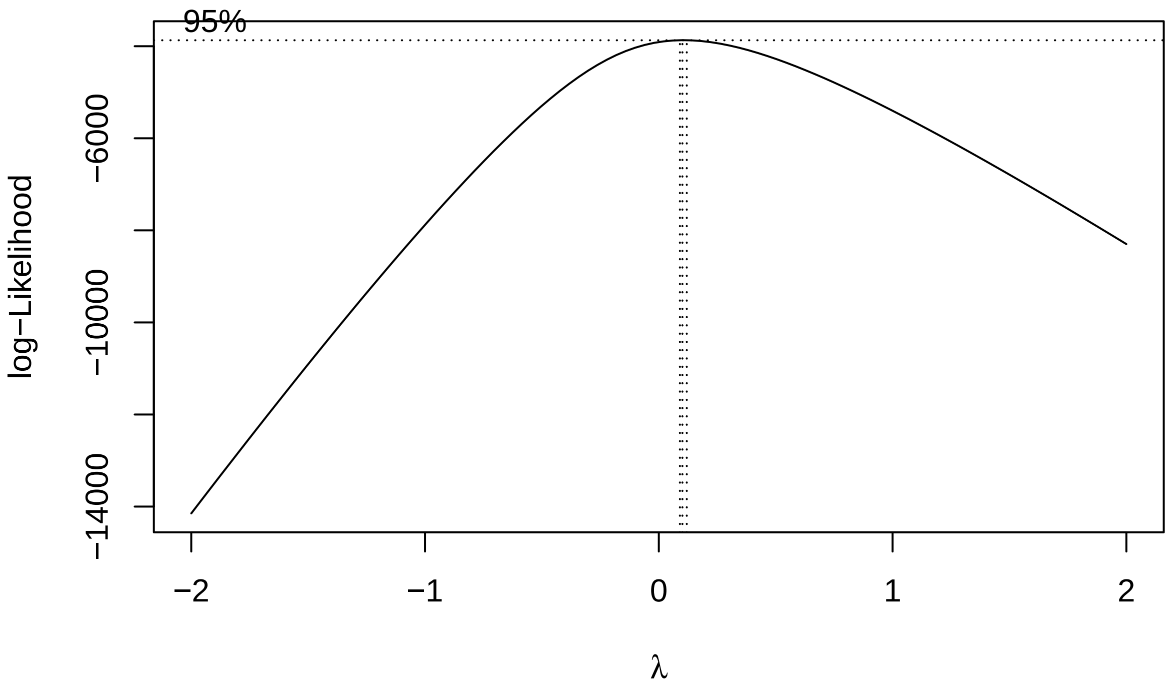

Figure 37: Distribution of the Acousticness Box-Cox Parameter

```r
lambda <- b_data_acousticness$x[which.max(b_data_acousticness$y)]
data_acousticness_tran <- (data$acousticness ^ lambda - 1) / lambda

b_data_duration_ms <- boxcox(lm(data$duration_ms ~ 1))
```

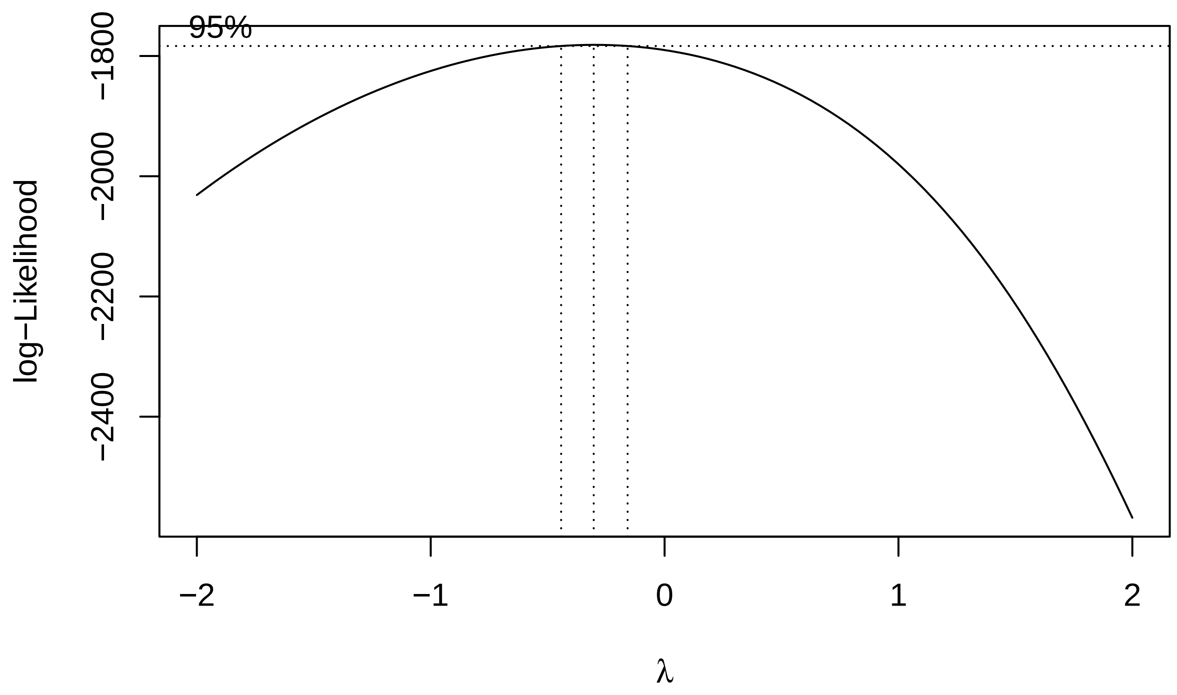

Figure 38: Distribution of the Duration Box-Cox Parameter

```
lambda <- b_data_duration_ms$x[which.max(b_data_duration_ms$y)]
data_duration_ms_tran <- (data$duration_ms ^ lambda - 1) / lambda

neg_loudness = data$loudness * (-1)
b_data_loudness <- boxcox(lm(neg_loudness ~ 1))
```

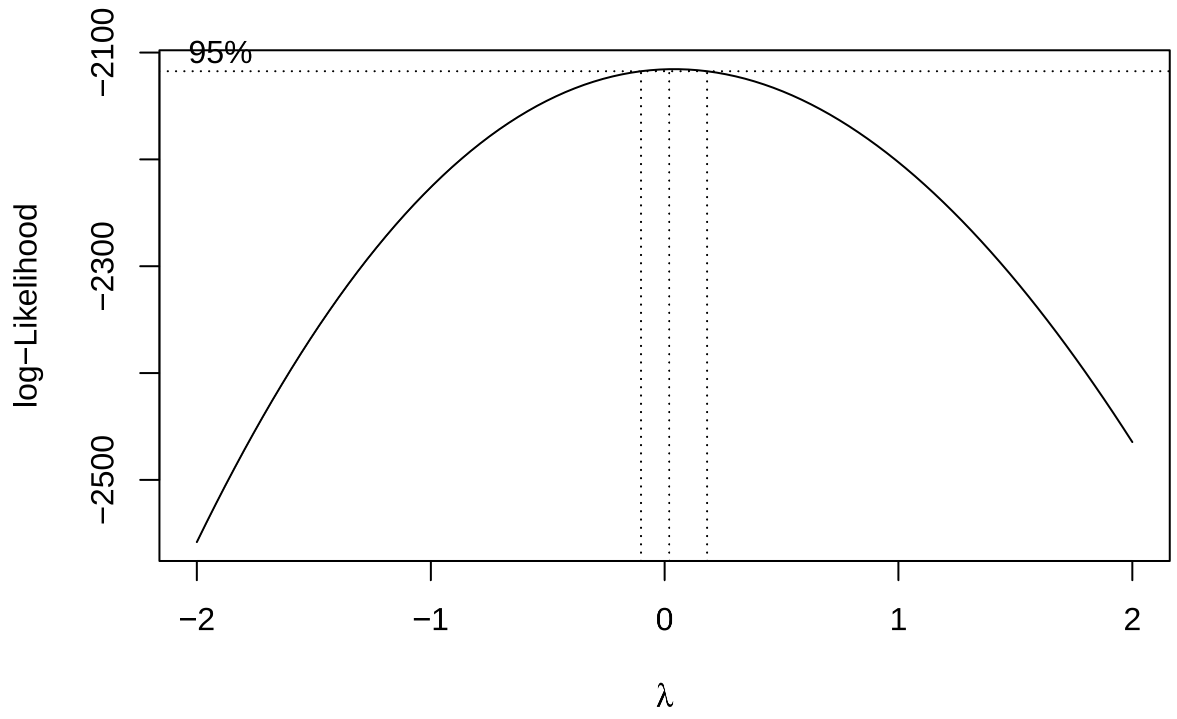

Figure 39: Distribution of the Loudness Box-Cox Parameter

```r
lambda <- b_data_loudness$x[which.max(b_data_loudness$y)]
data_loudness_tran <- (neg_loudness ^ lambda - 1) / lambda

b_data_tempo <- boxcox(lm(data$tempo ~ 1))
```

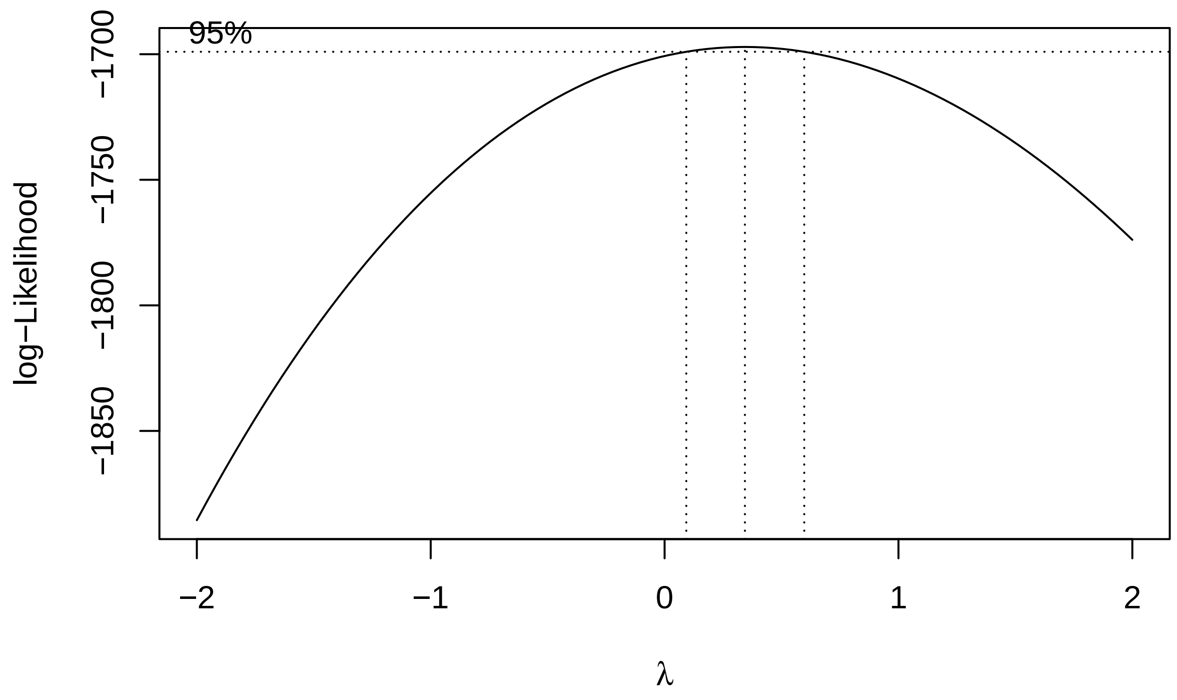

Figure 40: Distribution of the Tempo Box-Cox Parameter

```r
lambda <- b_data_tempo$x[which.max(b_data_tempo$y)]
data_tempo_tran <- (data$tempo ^ lambda - 1) / lambda

tran_data = matrix(c(data$IsWinner, data_acousticness_tran,
                     data_duration_ms_tran,
                     data$energy, data$instrumentalness, data_loudness_tran,
                     data_tempo_tran, data$valence), ncol = 8)

training_tran_data = tran_data[train_ind,]

test_tran_data = tran_data[-train_ind,]

colnames_tran_data = c("IsWinner", "acousticness_tran","duration_ms_tran",
                       "energy","instrumentalness", "loudness_tran",
                       "tempo_tran", "valence")

colnames(training_tran_data) = colnames_tran_data
colnames(test_tran_data) = colnames_tran_data
colnames(tran_data) = colnames_tran_data


training_tran_data = as.data.frame(training_tran_data)
test_tran_data = as.data.frame(test_tran_data)
tran_data = as.data.frame(tran_data)
```

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a technique based on the Bayes theorem, and finds a linear combination of variables that best separates the classes of the dependent variable.

```r
## LDA

simple_lda = lda(IsWinner ~ acousticness_tran + duration_ms_tran +
                energy + instrumentalness + loudness_tran + tempo_tran +
                valence, data = training_tran_data, family = "binomial")

pred_simple_lda = predict(simple_lda, newdata = test_tran_data,
                          type = "Response")
```

```r
# ROC curve

roc.out <- roc(test_set$IsWinner, as.numeric(pred_simple_lda$class) - 1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

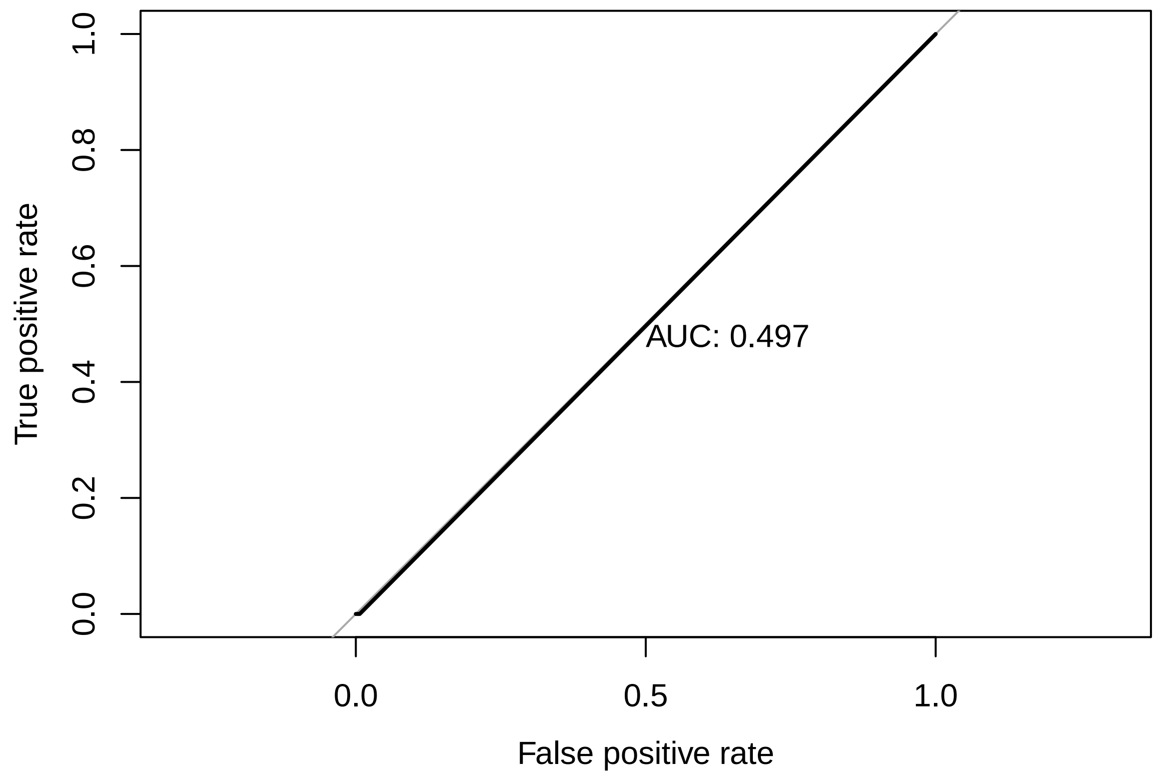

Figure 41: ROC Curve of LDA

```
auc(roc.out)
```

```
## Area under the curve: 0.4967
```

The ROC curve tells us that the predictions computed with this method are worse than a coin toss. In hope of improving this, the model will also be fit to the oversampled dataset.

Oversampled LDA

```
# Oversampled LDA

oversampled_train_tran_data = ovun.sample(IsWinner ~., data = training_tran_data,
                                          method = "over", p = 0.5, seed = 42)$data

simple_over_lda = lda(IsWinner ~ acousticness_tran + duration_ms_tran +
                   energy + instrumentalness + loudness_tran + tempo_tran +
                   valence, data = oversampled_train_tran_data,
                   family = "binomial")


pred_simple_over_lda = predict(simple_over_lda, newdata = test_tran_data,
                               type = "Response")
```

```
# ROC curve

roc.out <- roc(test_set$IsWinner, as.numeric(pred_simple_over_lda$class) - 1)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

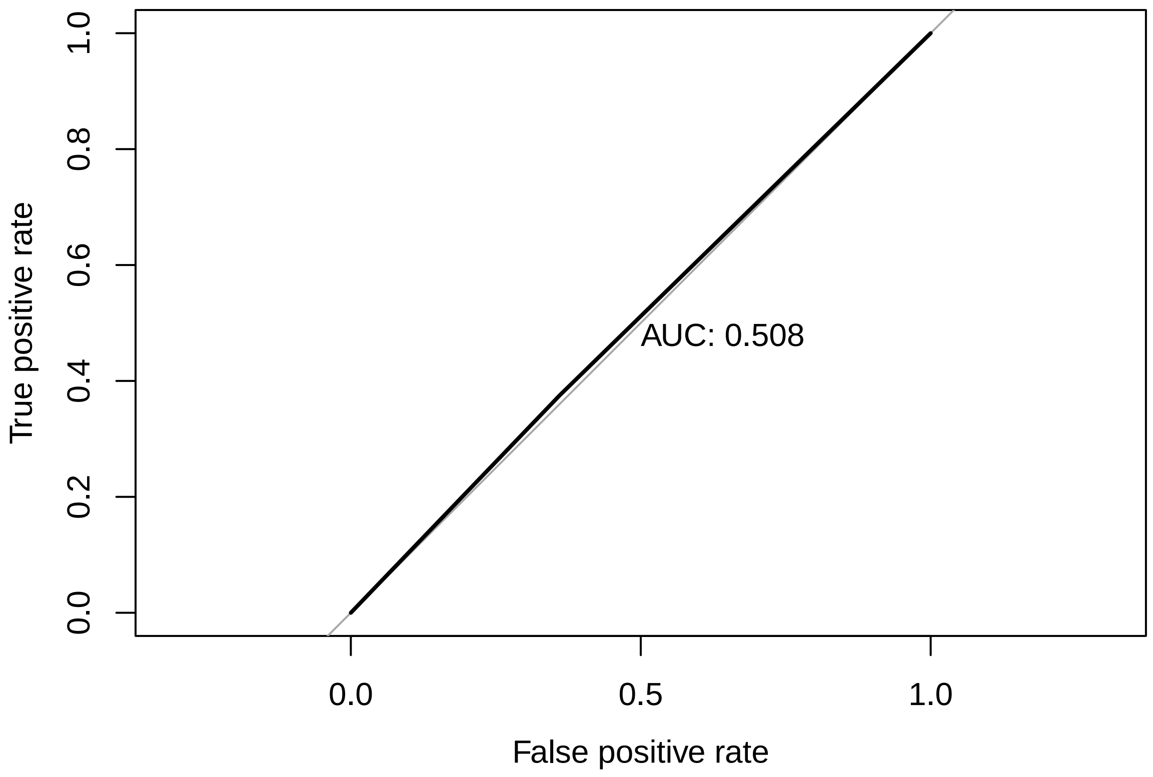

Figure 42: ROC Curve of Oversampled LDA

```
auc(roc.out)
```

```
## Area under the curve: 0.5075
```

The ROC curve shows a slight improvement, but the value is just above 0.5.

Predictions for the Oversampled LDA

Below, predictions are made on the oversampled LDA.

Threshold = 0.4

```
lda_over_predictions_04 = ifelse(pred_simple_over_lda$posterior[,1] > 0.4, 1, 0)
lda_over_accuracy_04 = sum(lda_over_predictions_04 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, lda_over_predictions_04)
```

```
##    lda_over_predictions_04
##       0   1
##   0  25 125
##   1   1  23
```

```
false_positive_lda_over_04 = table(test_set$IsWinner, lda_over_predictions_04)[3]
negative_lda_over_04 = table(test_set$IsWinner, lda_over_predictions_04)[1] +
                             table(test_set$IsWinner, lda_over_predictions_04)[3]
typeIerror_lda_over_04 = false_positive_lda_over_04 / negative_lda_over_04
typeIerror_lda_over_04
```

```
## [1] 0.8333333
```

```
true_positive_lda_over_04 = table(test_set$IsWinner, lda_over_predictions_04)[4]
positive_lda_over_04 = table(test_set$IsWinner, lda_over_predictions_04)[2] +
                             table(test_set$IsWinner, lda_over_predictions_04)[4]
sensitivity_lda_over_04 = true_positive_lda_over_04 / positive_lda_over_04
sensitivity_lda_over_04
```

```
## [1] 0.9583333
```

Threshold = 0.5

```
lda_over_predictions_05 = list(pred_simple_over_lda$class)
lda_over_accuracy_05 = sum(lda_over_predictions_05 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, lda_over_predictions_05[[1]])
```

```
##
##      0  1
##   0 96 54
##   1 15  9
```

```
false_positive_lda_over_05 = table(test_set$IsWinner, lda_over_predictions_05[[1]])[3]
negative_lda_over_05 = table(test_set$IsWinner, lda_over_predictions_05[[1]])[1] +
                             table(test_set$IsWinner, lda_over_predictions_05[[1]])[3]
typeIerror_lda_over_05 = false_positive_lda_over_05 / negative_lda_over_05
typeIerror_lda_over_05
```

```
## [1] 0.36
```

```
true_positive_lda_over_05 = table(test_set$IsWinner, lda_over_predictions_05[[1]])[4]
positive_lda_over_05 = table(test_set$IsWinner, lda_over_predictions_05[[1]])[2] +
                             table(test_set$IsWinner, lda_over_predictions_05[[1]])[4]
sensitivity_lda_over_05 = true_positive_lda_over_05 / positive_lda_over_05
sensitivity_lda_over_05
```

```
## [1] 0.375
```

Setting the threshold to 0.4 leads to high Type I Error and Sensitivity. This means that the model is able to detect the nominated songs, but classifies many non-nominated songs as nominated. Setting the threshold to 0.5 leads to low Type I Error and Sensitivity, which means that many nominated songs are classified as non-nominated, but most non-nominated songs are correctly classified.

Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is a generalization of LDA and does not assume that dependent variable have the same variance.

```
## QDA

qda = qda(IsWinner ~ acousticness_tran + duration_ms_tran +
            energy + instrumentalness + loudness_tran + tempo_tran +
            valence, data = training_tran_data, family = "binomial")

pred_qda = predict(qda, newdata = test_tran_data, type = "Response")
```

```
# ROC

roc.out <- roc(test_set$IsWinner, as.numeric(pred_qda$class) - 1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

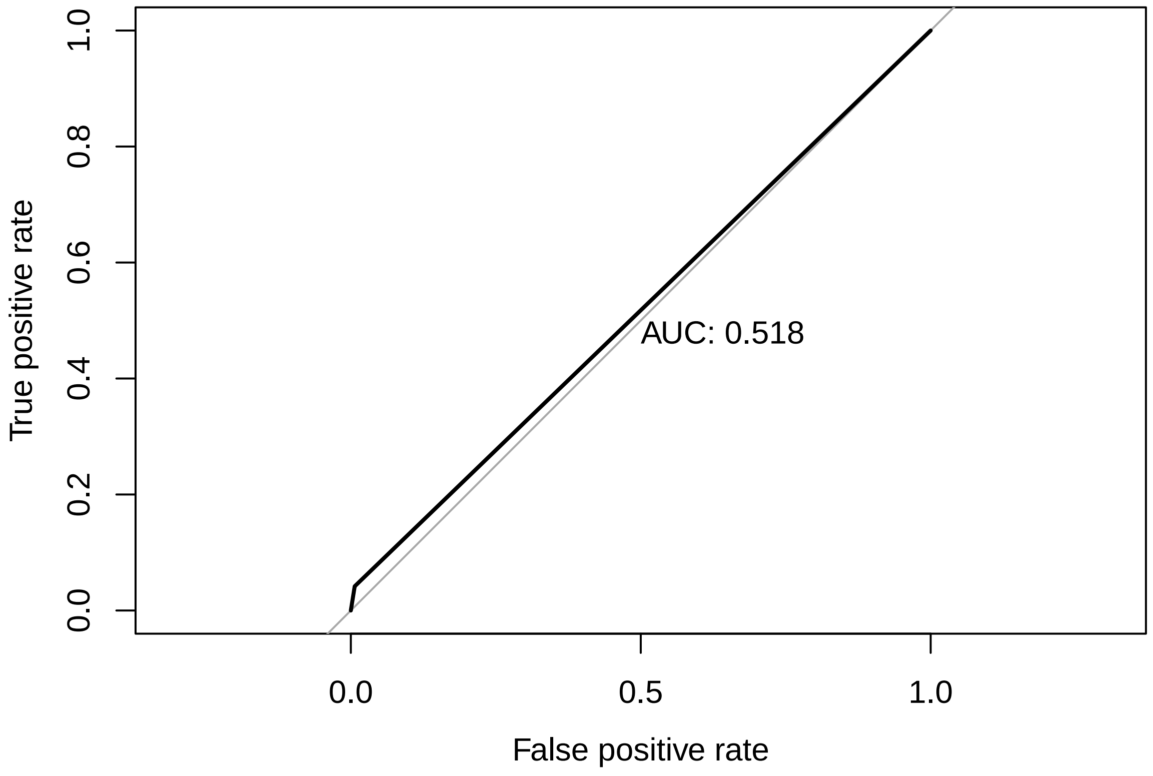

Figure 43: ROC Curve of QDA

```r
auc(roc.out)
```

```
## Area under the curve: 0.5175
```

The ROC curve shows a slight improvement compared to the linear discriminant analysis, but the area under the curve is still just above 0.5, so again the model will be fit to the oversampled dataset in an attempt to improve it.

```r
# QDA oversampled

qda_over = qda(IsWinner ~ acousticness_tran + duration_ms_tran +
            energy + instrumentalness + loudness_tran + tempo_tran +
            valence, data = oversampled_train_tran_data, family = "binomial")

pred_qda_over = predict(qda_over, newdata = test_tran_data, type = "Response")
```

```r
# ROC

roc.out <- roc(test_set$IsWinner, as.numeric(pred_qda_over$class) - 1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

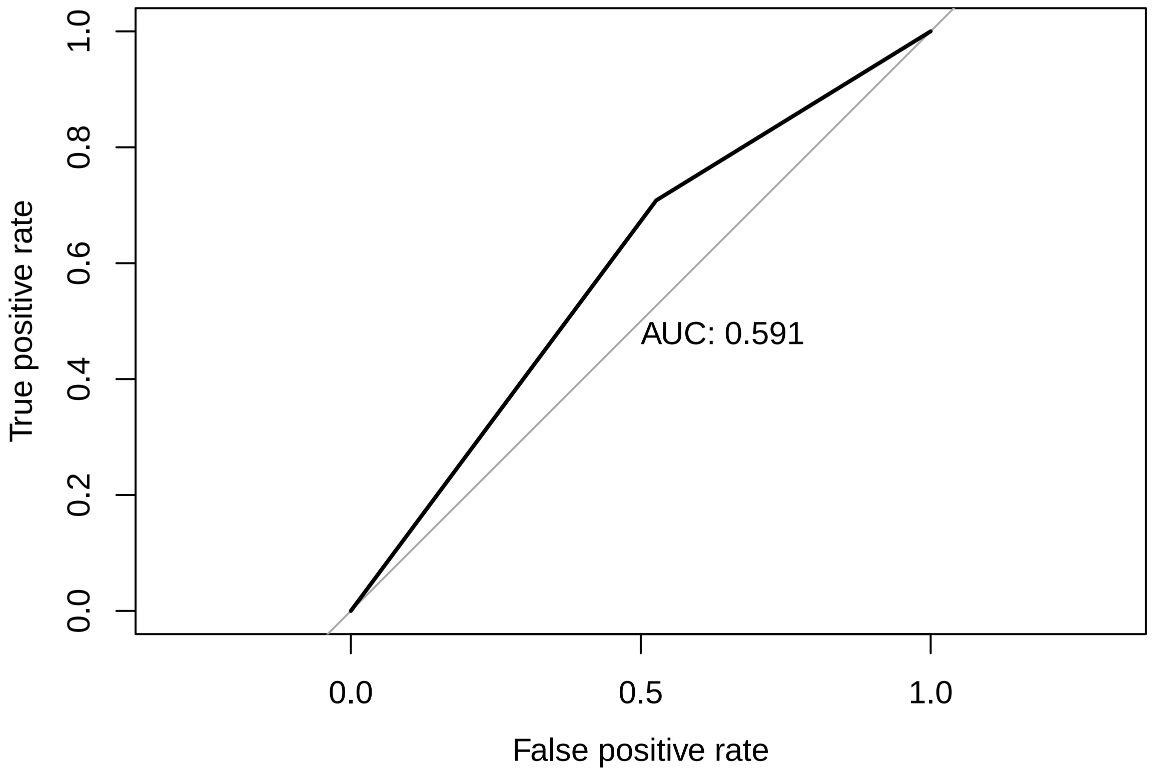

Figure 44: ROC Curve of Oversampled QDA

```r
auc(roc.out)
```

```
## Area under the curve: 0.5908
```

The ROC curve shows a significant improvement.

Predictions for the QDA

Threshold = 0.4

```
# Predictions

# Threshold 0.4

qda_over_predictions_04 = ifelse(pred_qda_over$posterior[,1] > 0.4, 1, 0)
qda_over_accuracy_04 = sum(qda_over_predictions_04 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, qda_over_predictions_04)
```

```
##    qda_over_predictions_04
##      0  1
##   0 54 96
##   1 12 12
```

```
false_positive_qda_over_04 = table(test_set$IsWinner, qda_over_predictions_04)[3]
negative_qda_over_04 = table(test_set$IsWinner, qda_over_predictions_04)[1] +
                             table(test_set$IsWinner, qda_over_predictions_04)[3]
typeIerror_qda_over_04 = false_positive_qda_over_04 / negative_qda_over_04
typeIerror_qda_over_04
```

```
## [1] 0.64
```

```
true_positive_qda_over_04 = table(test_set$IsWinner, qda_over_predictions_04)[4]
positive_qda_over_04 = table(test_set$IsWinner, qda_over_predictions_04)[2] +
                             table(test_set$IsWinner, qda_over_predictions_04)[4]
sensitivity_qda_over_04 = true_positive_qda_over_04 / positive_qda_over_04
sensitivity_qda_over_04
```

```
## [1] 0.5
```

Threshold = 0.5

```
# Threshold 0.5

qda_over_predictions_05 = list(pred_qda_over$class)
qda_over_accuracy_05 = sum(qda_over_predictions_05 == test_set[2]) / dim(test_set[2])[1]

table(test_set$IsWinner, qda_over_predictions_05[[1]])
```

```
##
##      0  1
##   0 71 79
##   1  7 17
```

```
false_positive_qda_over_05 = table(test_set$IsWinner, qda_over_predictions_05[[1]])[3]
negative_qda_over_05 = table(test_set$IsWinner, qda_over_predictions_05[[1]])[1] +
                             table(test_set$IsWinner, qda_over_predictions_05[[1]])[3]
typeIerror_qda_over_05 = false_positive_qda_over_05 / negative_qda_over_05
typeIerror_qda_over_05
```

```
## [1] 0.5266667
```

```
true_positive_qda_over_05 = table(test_set$IsWinner, qda_over_predictions_05[[1]])[4]
positive_qda_over_05 = table(test_set$IsWinner, qda_over_predictions_05[[1]])[2] +
                             table(test_set$IsWinner, qda_over_predictions_05[[1]])[4]
sensitivity_qda_over_05 = true_positive_qda_over_05 / positive_qda_over_05
sensitivity_qda_over_05
```

```
## [1] 0.7083333
```

Setting the threshold to 0.5 gave better results both in terms of Type I Error and Sensitivity. In particular, the model correctly classifies songs that weren't nominated as non-nominated 53% of the time and correctly classifies 71% of the songs that were nominated as nominated.

Regularized Regression

Regularized regression models are a type of regression with an additional constraint on the parameter designed to prevent overfitting and deal with multicollinearity.

Ridge Regression

Ridge regression exploits L2 regularization to shrink the estimated parameters towards zero.

```
## Ridge regression

ridge_cv = cv.glmnet(data.matrix(oversampled_train_data[,-1]),
                     oversampled_train_data$IsWinner, alpha = 0, family = "binomial",
                     type.measure = "class")
plot(ridge_cv)
```

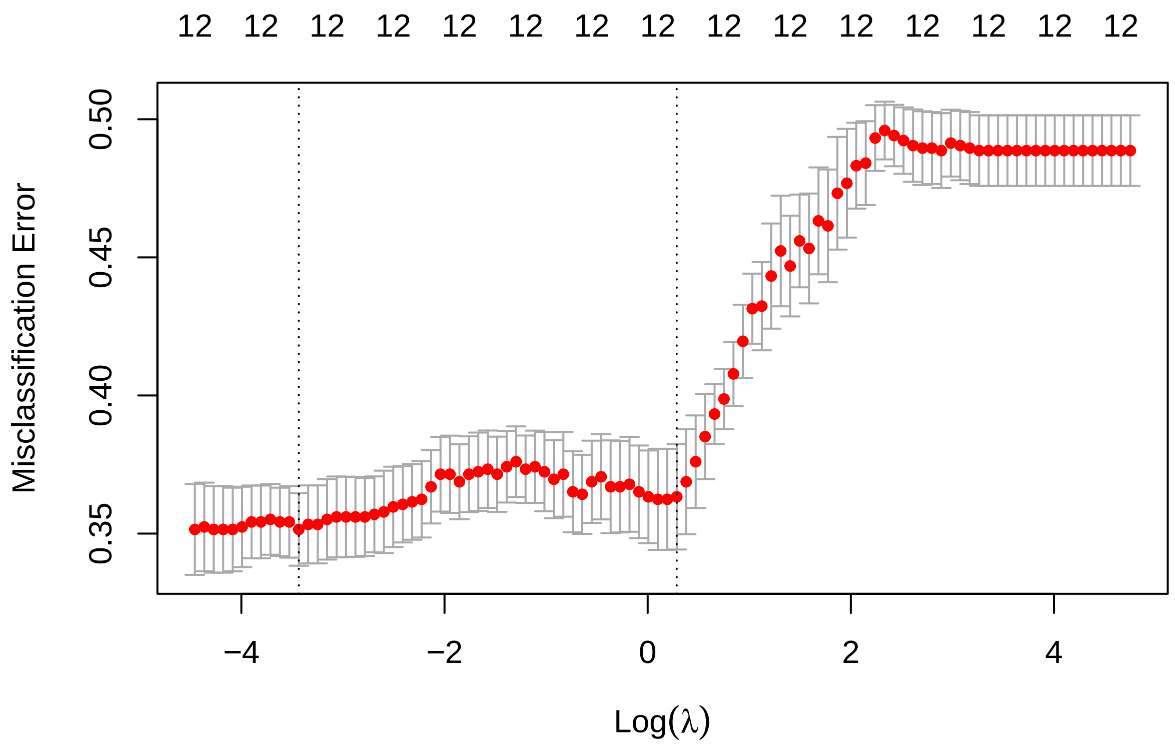

Figure 45: Misclassification Error of the Ridge Regression

```r
lambda_opt_ridge = ridge_cv$lambda.min

pred_ridge = predict(ridge_cv, data.matrix(test_set[, c(-1, -2)]), type = "class",
                     s = lambda_opt_ridge)

table(test_set$IsWinner, pred_ridge)
```

```
##    pred_ridge
##      0  1
##   0 97 53
##   1 14 10
```

```r
false_positive_ridge = table(test_set$IsWinner, pred_ridge)[3]
negative_ridge = table(test_set$IsWinner, pred_ridge)[1] + table(test_set$IsWinner,
                                                                 pred_ridge)[3]
typeIerror_ridge = false_positive_ridge / negative_ridge
typeIerror_ridge
```

```
## [1] 0.3533333
```

```r
true_positive_ridge = table(test_set$IsWinner, pred_ridge)[4]
positive_ridge = table(test_set$IsWinner, pred_ridge)[2] + table(test_set$IsWinner,
                                                                 pred_ridge)[4]
sensitivity_ridge = true_positive_ridge / positive_ridge
sensitivity_ridge
```

```
## [1] 0.4166667
```

Ridge regression predictions showed a low Type I Error and Sensitivity, meaning that the model is able to identify non-nominated songs, but struggles to correctly classify nominated songs.

Lasso Regression

Lasso regression uses L1 regularization to force some of the estimated parameters to be exactly zero.

```r
## Lasso regression

lasso_cv = cv.glmnet(data.matrix(oversampled_train_data[,-1]),
                     oversampled_train_data$IsWinner,
                     alpha = 1, family = "binomial", type.measure = "class")

plot(lasso_cv)
```

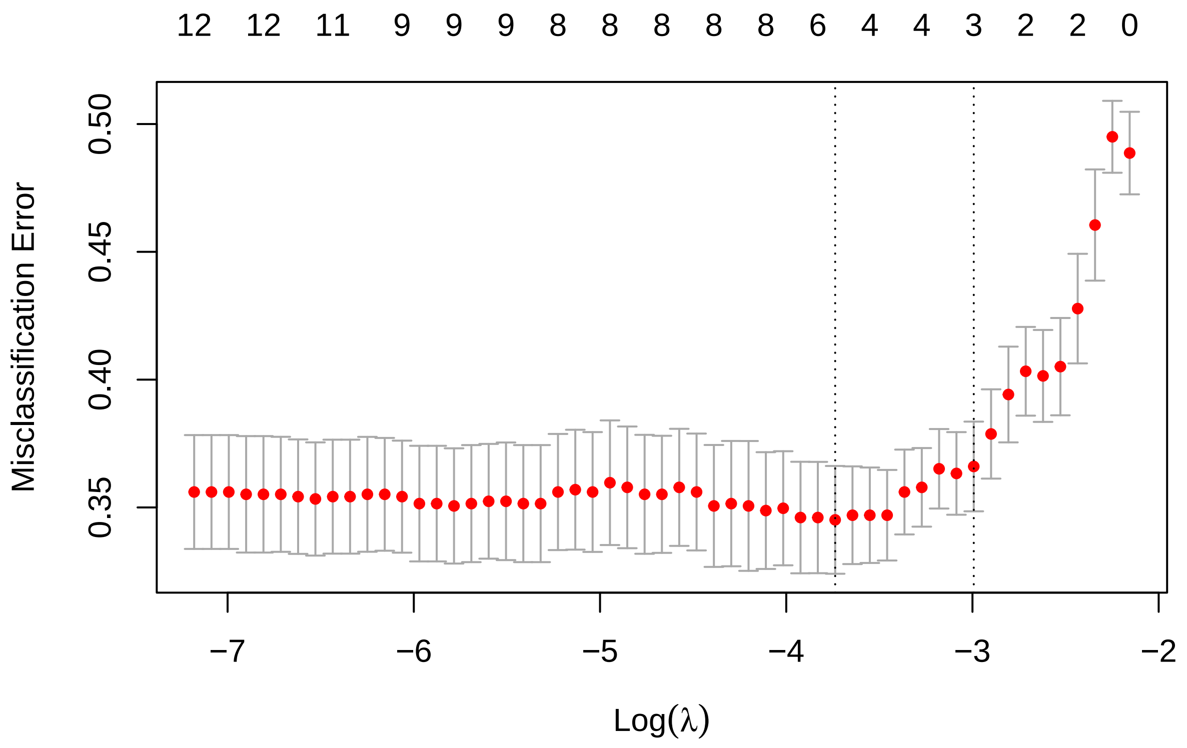

Figure 46: Misclassification Error of the Lasso Regression

```
lambda_opt_lasso = lasso_cv$lambda.min

pred_lasso = predict(lasso_cv, data.matrix(test_set[, c(-1, -2)]), type = "class",
                     s = lambda_opt_ridge)

table(test_set$IsWinner, pred_lasso)
```

```
##    pred_lasso
##      0  1
##   0 97 53
##   1 14 10
```

```
false_positive_lasso = table(test_set$IsWinner, pred_lasso)[3]
negative_lasso = table(test_set$IsWinner, pred_lasso)[1] + table(test_set$IsWinner,
                                                                  pred_lasso)[3]
typeIerror_lasso = false_positive_lasso / negative_lasso
typeIerror_lasso
```

```
## [1] 0.3533333
```

```
true_positive_lasso = table(test_set$IsWinner, pred_lasso)[4]
positive_lasso = table(test_set$IsWinner, pred_lasso)[2] + table(test_set$IsWinner,
                                                                  pred_lasso)[4]
sensitivity_lasso = true_positive_lasso / positive_lasso
sensitivity_lasso
```

```
## [1] 0.4166667
```

Lasso regression gave similar results when compared to ridge regression in terms of the Type I Error, and slightly worse results in terms of Sensitivity, therefore the same comments apply.

KNN

KNN (K-Nearest Neighbors) models are non-parametric and offer a flexible decision boundary for classification problems. The model will classify a song based on the "k" nearest neighbors (measured by Euclidean distance) of said song in the parameter space.

```r
# K-NN

min_max_norm = function(x) {
  (x - min(x)) / (max(x) - min(x))
}

normalized_data = as.data.frame(lapply(data[,c(-1, -2, -9, -11, -13)], min_max_norm))

IsWinner_norm = data$IsWinner

normalized_data = cbind(IsWinner_norm, normalized_data)

training_norm_data = normalized_data[train_ind,]

test_norm_data = normalized_data[-train_ind,]
```

After normalizing the data, the selection of the previously mentioned "k" value is performed to determine the most efficient number of "neighbors" the model will consider when classifying a song.

```r
# Selecting k

kmax = 100

test_error = numeric(kmax)

for (k in 1:kmax) {
  knn_pred = as.factor(knn(training_norm_data[,-1], test_norm_data[,-1],
                           cl = training_norm_data$IsWinner_norm, k = k))
  cm = confusionMatrix(data = knn_pred,
                       reference = as.factor(test_norm_data$IsWinner_norm))
  test_error[k] = 1 - cm$overall[1]
}
k_min = which.min(test_error)
k_min
```

```
## [1] 25
```

After analyzing the original training data set, the KNN model determined 25 to be the most optimal number of neighbors to consider.

```r
knn = knn(training_norm_data[,-1], test_norm_data[,-1],
          cl = training_norm_data$IsWinner_norm, k = k_min)

knn_pred_min = as.factor(knn)

table(test_norm_data$IsWinner_norm, knn)
```

```
##    knn
##       0   1
##   0 150   0
##   1  23   1
```

```
roc.out <- roc(test_set$IsWinner, as.numeric(knn) - 1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

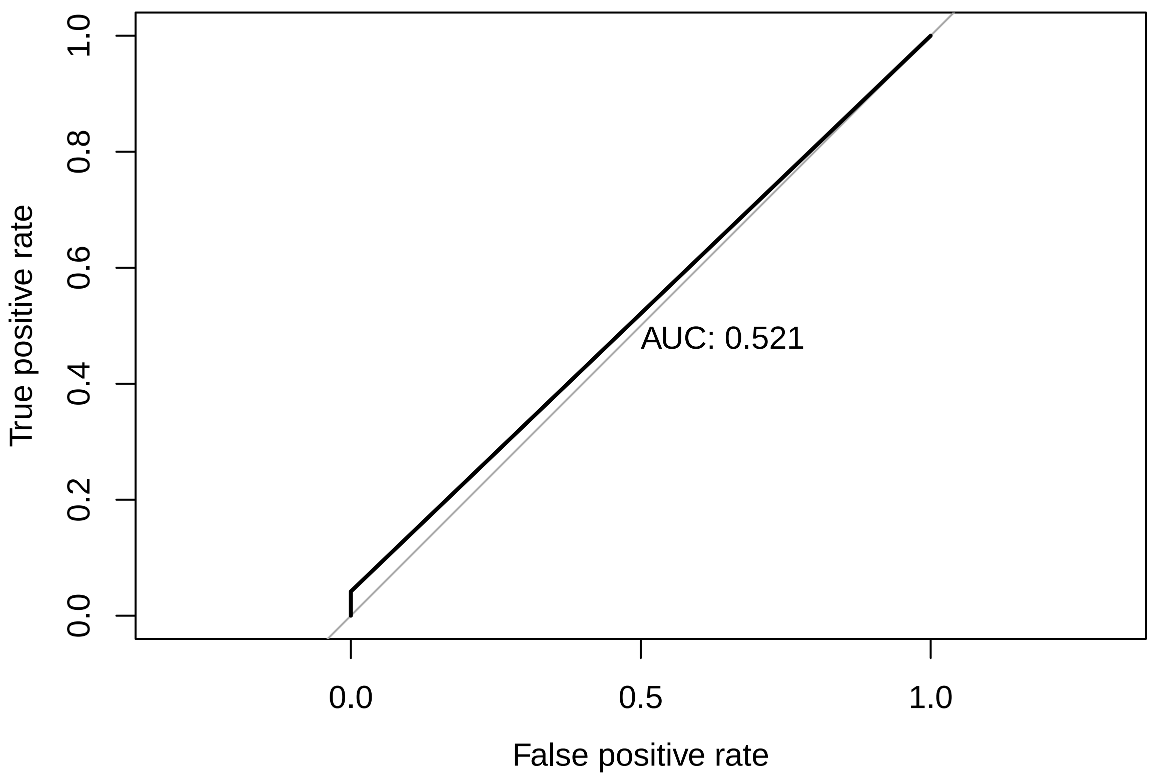

Figure 47: ROC Curve of the KNN Model

```
auc(roc.out)
```

```
## Area under the curve: 0.5208
```

```
false_positive_knn = table(test_set$IsWinner, as.numeric(knn) - 1)[3]
negative_knn = table(test_set$IsWinner, as.numeric(knn) - 1)[1] +
                     table(test_set$IsWinner, as.numeric(knn) - 1)[3]
typeIerror_knn = false_positive_knn / negative_knn
typeIerror_knn
```

```
## [1] 0
```

```
true_positive_knn = table(test_set$IsWinner, as.numeric(knn) - 1)[4]
positive_knn = table(test_set$IsWinner, as.numeric(knn) - 1)[2] +
                     table(test_set$IsWinner, as.numeric(knn) - 1)[4]
sensitivity_knn = true_positive_knn / positive_knn
sensitivity_knn
```

```
## [1] 0.04166667
```

Utilizing the chosen K value, the model yields an accuracy of 86.8% and obtains a Type I Error of 0, predicting all of the non-nominated songs correctly. However, it also yields a Sensitivity of 0.0417, barely classifying any nominated songs correctly. The AUC of the ROC curve is only 0.521.

Oversampling

Now the same process as above is repeated, but by building a model using the oversampled data.

```r
# oversampled

test_over_error = numeric(kmax)

normalized_over_data = as.data.frame(lapply(oversampled_train_data[,c(-8, -10, -12)],
                                            min_max_norm))

training_norm_data_over = normalized_over_data[train_ind,]

for (k in 1:kmax) {
  knn_over_pred = as.factor(knn(training_norm_data_over[,-1],
                                test_norm_data[,-1],
                                cl = training_norm_data$IsWinner_norm, k = k))
  cm_over = confusionMatrix(data = knn_over_pred,
                            reference = as.factor(test_norm_data$IsWinner_norm))
  test_over_error[k] = 1 - cm_over$overall[1]
}


k_min_over = which.min(test_over_error)
k_min_over
```

```
## [1] 14
```

The chosen value of “k” for the oversampled dataset is 14.

```r
knn_over = knn(training_norm_data_over[,-1], test_norm_data[,-1],
          cl = training_norm_data$IsWinner_norm, k = k_min_over)

knn_pred_min_over = as.factor(knn_over)

table(test_norm_data$IsWinner_norm, knn_over)
```

```
##    knn_over
##       0   1
##   0 146   4
##   1  23   1
```

```r
roc.out <- roc(test_set$IsWinner, as.numeric(knn_over) - 1)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(roc.out, print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate")
```

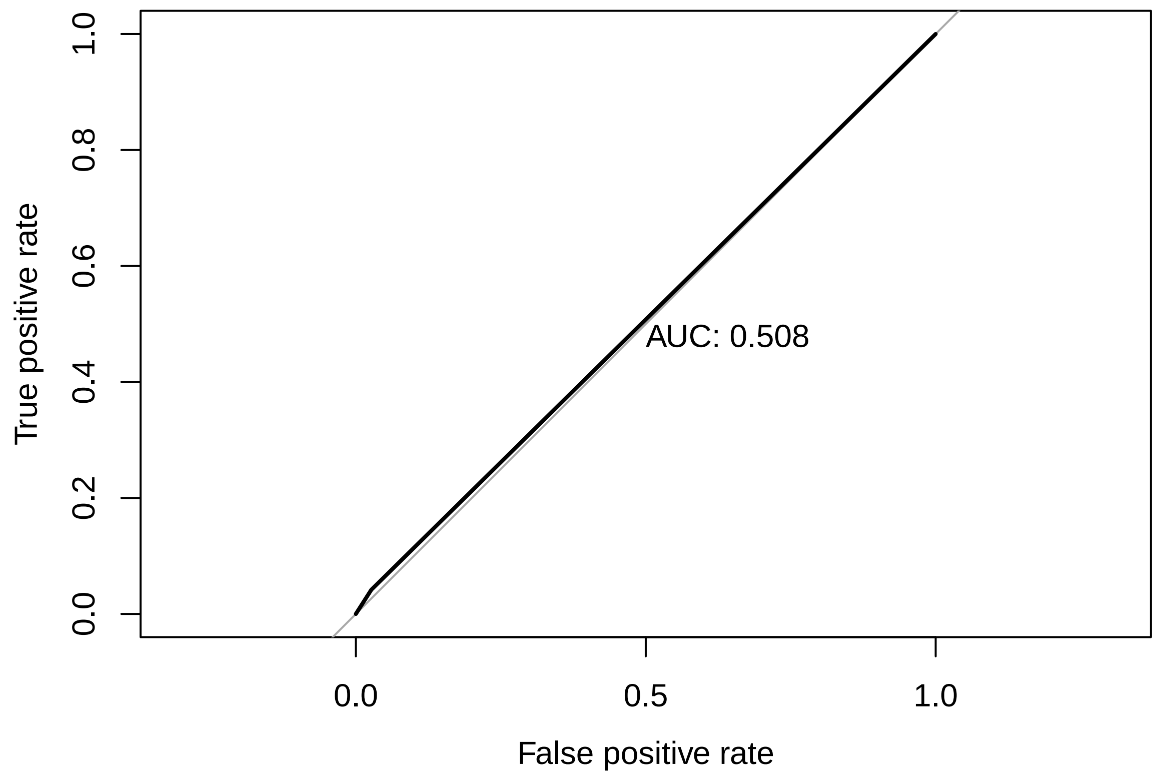

Figure 48: ROC Curve of the Oversampled KNN Model

```
auc(roc.out)
```

```
## Area under the curve: 0.5075
```

```
false_positive_knn_over = table(test_set$IsWinner, as.numeric(knn_over) - 1)[3]
negative_knn_over = table(test_set$IsWinner, as.numeric(knn_over) - 1)[1] +
                          table(test_set$IsWinner, as.numeric(knn_over) - 1)[3]
typeIerror_knn_over = false_positive_knn_over / negative_knn_over
typeIerror_knn_over
```

```
## [1] 0.02666667
```

```
true_positive_knn_over = table(test_set$IsWinner, as.numeric(knn_over) - 1)[4]
positive_knn_over = table(test_set$IsWinner, as.numeric(knn_over) - 1)[2] +
                    table(test_set$IsWinner, as.numeric(knn_over) - 1)[4]
sensitivity_knn_over = true_positive_knn_over / positive_knn_over
sensitivity_knn_over
```

```
## [1] 0.04166667
```

Utilizing the chosen "k" value, the KNN model yields a classification accuracy of 84.5%, but again produces a very low Type I Error (0.027) and very low Sensitivity (0.0417), correctly classifying only one nominated song.

Results and Conclusions

The below table makes a simple comparison of the predictive abilities of all the models created throughout this analysis:

Model Type	ROC
Logistic Full	0.598
Logistic Reduced	0.576
Logistic Oversampled Full	0.579
Logistic Oversampled Reduced	0.579
Linear Discriminant Analysis	0.497
LDA Oversampled	0.508
Quadratic Discriminant Analysis	0.518
QDA Oversampled	0.591
K-Nearest Neighbors	0.521
KNN Oversampled	0.508

Model Type	Type I Error	Sensitivity
Ridge Regression	0.353	0.417
Lasso Regression	0.353	0.417
K-Nearest Neighbors	0	0.0417
KNN Oversampled	0.027	0.417

Overall, the models that produced the highest diagnostic accuracy were the Logistic Full, Logistic Oversampled Reduced, and QDA Oversampled. As a reminder, although the AUC of the ROC for the Logistic Full model is the largest, it's sensitivity at the thresholds tested were poor. Since Sensitivity is the metric this research question values most highly (properly classifying the nominated songs), the fact that the QDA Oversampled and the Logistic Oversampled Reduced models maintained high values across multiple classification thresholds suggests that they are the best models for the job.

Future work could expand upon this research by consulting experts in the music industry, such as musicologists, to gain insight into other possible features. Adding new variables could allo the models to discover new connections it did not previously had and lead to higher classification abilities. Different types of models could also be considered, or modifications to the models considered here. Changing the accuracy metrics during the training and hyperparameter tuning stages of the KNN and/or early stopping could be implemented. Incorporating a lager data set with more recent data (if available) could also improve the models' abilities to classify nominated songs. Employing alternate, more advanced strategies to handle the imbalanced dataset could improve the output as well. Inspecting analysis over time may be a useful perspecive, offering insight into how trends over time may influence the likelihood of a type of song to be nominated.