# Hot 100 Track Qualities

Iris Robedeaux

2024-12-31

# Introduction

In this study, I will analyze the features of a song that impact the charting of a song on the Billboard Hot 100. I'll delve into the specific qualities that impact charting, keying in specifically to if a song is a Top 10 hit or not. Billboard is the standard for charting in the United States, and provides a standard for if a song is popular or not. A charting song is factored through song purchases on physical media, digital formats, and a small portion of streaming. While they have several different charts, the Hot 100 is the top 100 charting songs (instead of albums) for a given week.

I chose this data because I follow the charts on my own time, and I wanted to find more information on what types of songs are more likely to chart, and possibly even create a checklist of qualities that could create a hit song. For the purposes of this data, I will focus on songs that are top 10 hits, to create a better analysis of what becomes a mega-hit as opposed to a song that charts.

Table 1: First five rows of the unprocessed data (continued below)

| ranking | song | band_singer | songurl | titletext | url |
|---|---|---|---|---|---|
| 1 | Breathe | Faith H... | /wiki/B... | Breathe | /wiki/F... |
| 2 | Smooth | Santana | /wiki/S... | Smooth | /wiki/S... |
| 2 | Smooth | Rob Tho... | /wiki/S... | Smooth | /wiki/R... |
| 3 | Maria M... | Santana | /wiki/M... | Maria M... | /wiki/S... |
| 3 | Maria M... | The Pro... | /wiki/M... | Maria M... | /wiki/T... |

Table 2: Table continues below

| year | lyrics | uri | danceability | energy | key | loudness | mode |
|---|---|---|---|---|---|---|---|
| 2000 | I can f... | spotify... | 0.529 | 0.496 | 7 | -9.007 | 1 |
| 2000 | Man, it... | spotify... | 0.609 | 0.923 | 9 | -3.908 | 1 |
| 2000 | Man, it... | spotify... | 0.59 | 0.637 | 9 | -9.171 | 1 |
| 2000 | Ladies ... | spotify... | 0.777 | 0.601 | 2 | -5.931 | 1 |
| 2000 | Turn up... | spotify... | NA | NA | NA | NA | NA |

Table 3: Table continues below

| speechiness | acousticness | instrumentalness | liveness | valence | tempo |
|---|---|---|---|---|---|
| 0.029 | 0.173 | 0 | 0.251 | 0.278 | 136.9 |
| 0.0338 | 0.16 | 4.73e-0... | 0.295 | 0.961 | 116 |
| 0.0301 | 0.00225 | 0.807 | 0.299 | 0.724 | 116 |
| 0.126 | 0.0406 | 0.00201 | 0.0348 | 0.68 | 97.91 |
| NA | NA | NA | NA | NA | NA |

| type | id | track_href | analysis_url | duration_ms | time_signature |
|---|---|---|---|---|---|
| audio_f... | 3y4LxiY... | https:/... | https:/... | 250547 | 4 |
| audio_f... | 0n2SEXB... | https:/... | https:/... | 294987 | 4 |
| audio_f... | 5IALWUY... | https:/... | https:/... | 244924 | 4 |
| audio_f... | 3XKIUb7... | https:/... | https:/... | 261973 | 4 |
| | | | | NA | NA |

As shown in the chart of the previous page, the given data has 26 variables. Namely, ranking, song, band_singer, and year, which shows the ranking of a song, the name of the song, and the year it was released. There are also the lyrics, which I won't be using within this analysis, as word processing would be out of the scop of this project. There are a couple different columns used to ID the song on Genius (the lyrics source), Billboard (the chart origin), and on Spotify (the source of the song qualities). Lastly, there are qualities of the song. These song qualities are examined and given by Spotify's AI model, which evaluates songs to better it's own algorithm.

The data has 3397 rows, or song instances, to begin with. This, however, does include duplicate songs that chart for multiple weeks, and there are a few rows that don't have the song qualities data.

# Data Evaluation

The data itself was quite clean, providing a great base for analysis. I did, however, make a couple tweaks to make it simpler. First, I concatenated the data so that only one instance of each song was included, just to avoid duplicates. This was done so that it was included at it's peak week, so we can see the song at it's most popular. I removed the duplicate columns for the song title, in addition to the ID columns, since they don't aid in my analysis. I also removed the lyrics because they aren't used. I then removed any rows that didn't have a "duration," thereby removing any rows without Spotify song qualities. These are necessary for my analysis, and so I won't be able to use rows without this data.

Finally, I added a "top10" column, which is a binary value that indicates if a song charted within the top 10 of the Hot 100 at any point within it's charting lifetime. This separates the charting songs from the standout songs. I also renamed the "band_singer" column to "artist" in the interest of brevity.

As seen in the table on the next page, the columns now indicate the peak song ranking, the song name, artist, year, if the song was a top 10 hit, and the qualities of the song, which included features like the song's key, it's duration, and several variables quantifying aspects like loudness or dancability.

Table 5: First five rows of the processed data (continued below)

| ranking | song | artist | year | danceability | energy |
|---|---|---|---|---|---|
| 37 | '03 Bonnie & Clyde | Jay-Z | 2003 | 0.759 | 0.678 |
| 84 | 19 Somethin' | Mark Wills | 2003 | 0.58 | 0.816 |
| 14 | 21 Questions | 50 Cent | 2003 | 0.576 | 0.807 |
| 14 | 21 Questions | Nate Dogg | 2003 | 0.561 | 0.775 |
| 92 | 24/7 | Kevon Edmonds | 2000 | 0.696 | 0.627 |

Table 6: Table continues below

| key | loudness | mode | speechiness | acousticness | instrumentalness |
|---|---|---|---|---|---|
| 9 | -5.148 | 0 | 0.314 | 0.23 | 0 |
| 4 | -5.165 | 1 | 0.0726 | 0.132 | 0 |
| 6 | -3.908 | 0 | 0.317 | 0.353 | 0.000363 |
| 1 | -6.22 | 0 | 0.322 | 0.349 | 0.000303 |
| 4 | -5.629 | 1 | 0.0334 | 0.0685 | 0 |

| liveness | valence | tempo | duration_ms | time_signature | top10 |
|---|---|---|---|---|---|
| 0.15 | 0.327 | 89.64 | 205560 | 4 | FALSE |
| 0.0776 | 0.814 | 89.94 | 200467 | 4 | FALSE |
| 0.0512 | 0.908 | 90.01 | 224427 | 4 | FALSE |
| 0.267 | 0.85 | 90.93 | 134133 | 4 | FALSE |
| 0.0993 | 0.322 | 116 | 274267 | 4 | FALSE |

# Data Modeling

## Introduction

I chose three modeling strategies for this data. Logistic Regression, Linear Discriminant Analysis, and Classification Trees. The response variable, if a song is a Top 10 hit, is binary, a true or false value. Because of this, the options for possible models was constrained to those which could predict a binary response. In deciding in what to use, K-Nearest Neighbors was also considered, however this model proved ineffective for this data, and other models provided better predictions.

For a standardized comparison between models, the training and test sets were the same for all of them. In addition, all (but one) of the models takes most of the possible variables into account. The equation for these models is:

$$Top10 = \hat{f}(\cdot)$$

Where Top10 is a true or false value if a song is a top 10 hit, and $\hat{f}(\cdot)$ is a function of the predictors: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration (in ms), and time signature.

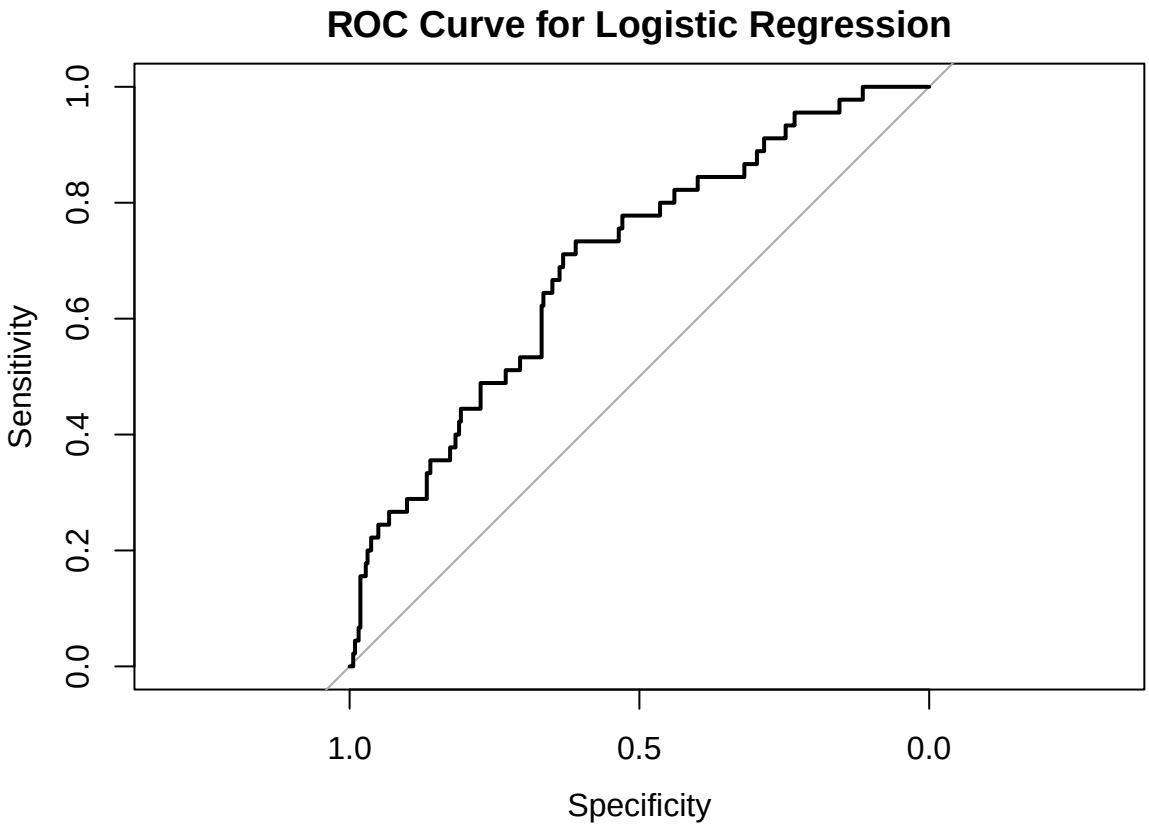
## Logistic Regression

The Logistic Regression model was the first to be considered for this data. It relies on a linear model converted through a link equation to provide the log odds of the response. As seen in Appendix A.4, I used the caret package's "train" function to model this, using 10 fold cross validation to form the model. In addition, this model was used on the testing set in order to get a classification accuracy estimate to compare to the other models.

The equation for the logistic regression follows, where $\hat{f}(\cdot)$ is a linear combination of the predictors, and $P(Top10)$ is the log-likelihood of a Top 10 hit:

$$P(Top10) = \frac{e^{\hat{f}(\cdot)}}{1 + e^{\hat{f}(\cdot)}}$$

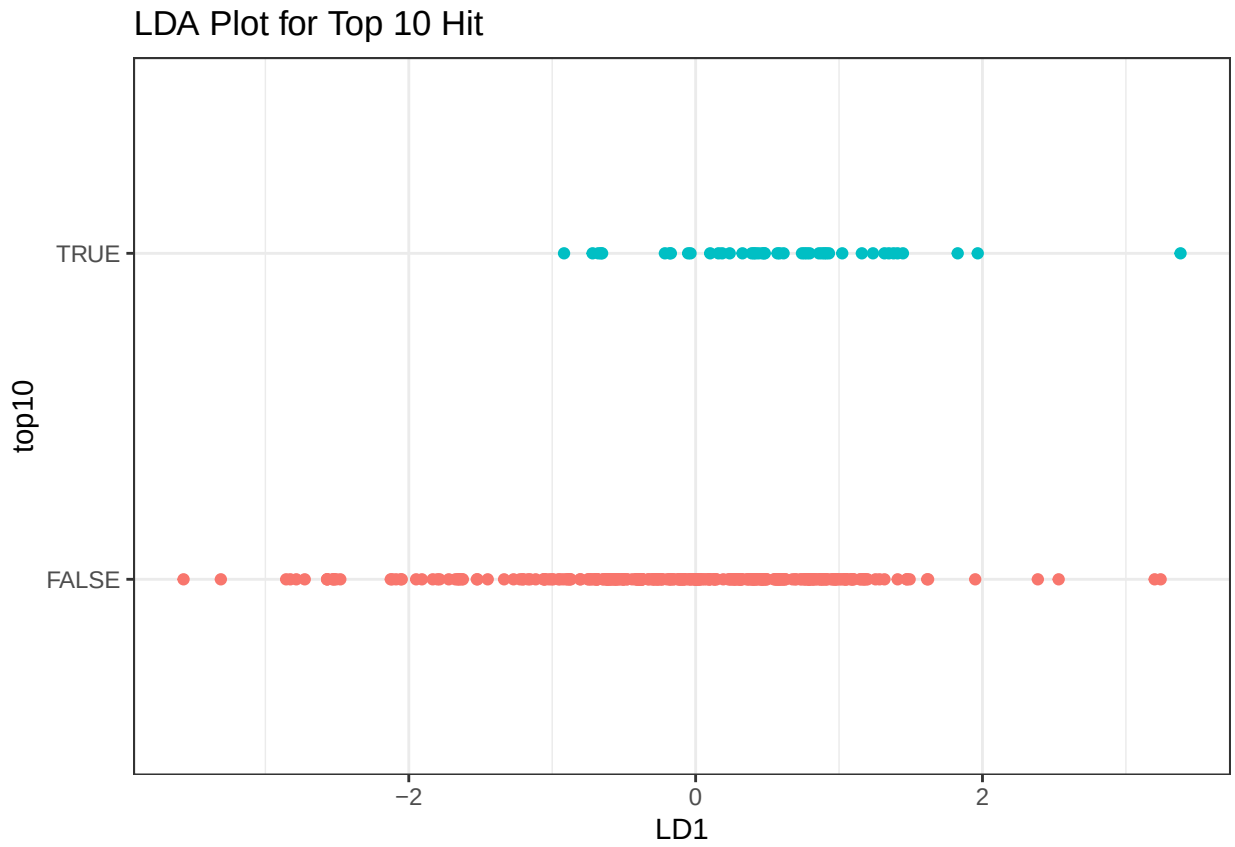
The ROC Curve for this model is on the next page. As the plot shows, the model has moderate predictive power of if a charting song lands in the Top 10. For Logistic Regression, as the trained model was cross validated on the testing set, the classification accuracy was 90%. This is a strong beginning to further analysis, as the Logistic Model does account for a large portion of the variance within the data.


ROC Curve for Logistic Regression
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity

## Linear Discriminant Analysis

The next model I chose to implement was a Linear Discriminant Analysis (LDA). LDA works by minimizing the variance within each class, the response in this case, and maximizing the distance between the means of the classes. In this data, the LDA works to classify the data into a Top 10 hit or not. For this model, I trained it on the training data, and then cross validated it with the testing data.

This model is a moderate predictor of a Top 10 hit. The graph on the following page shows the classification for the response variable, top10. While it does show some correlation for predicting a Top 10 hit, there isn't an obvious separation between the two, and the Top 10 group almost entirely overlaps the other observations. The classification accuracy for this model is 89%, which improves on the Logistic Regression model, but still doesn't provide a formitable model.

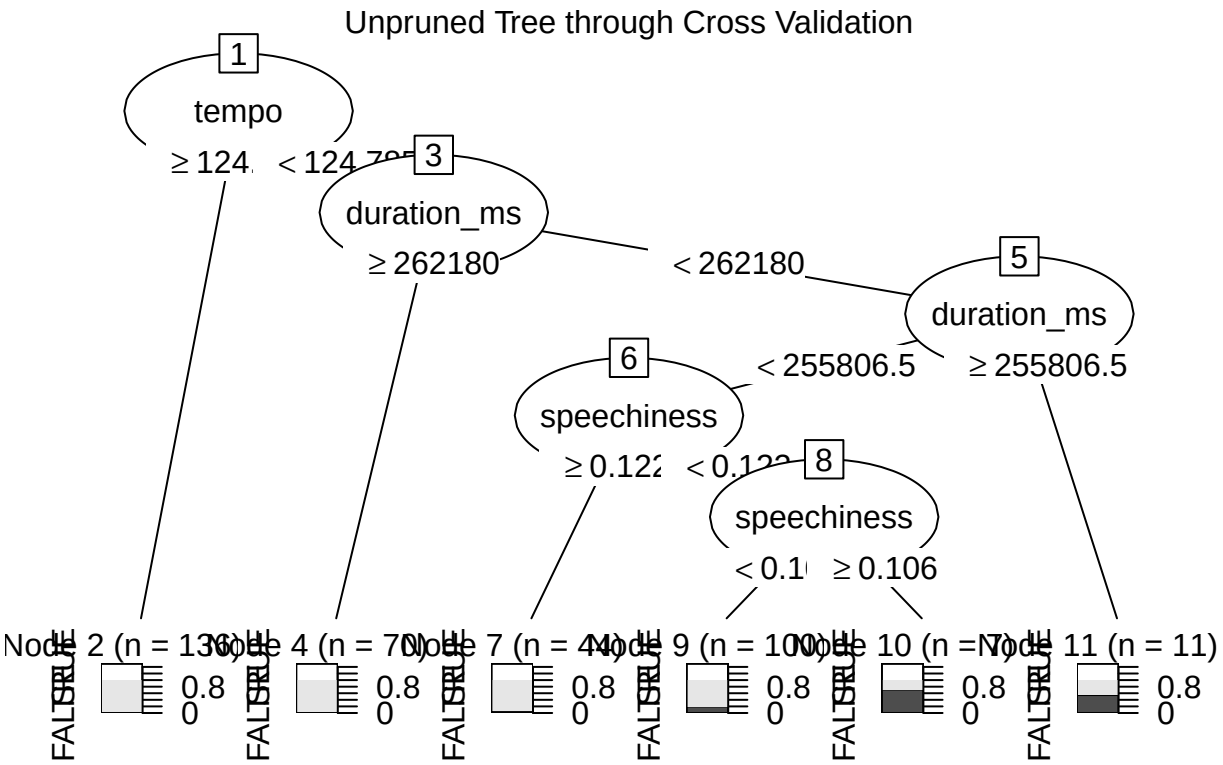

LDA Plot for Top 10 Hit
top10
TRUE
FALSE
−2
0
2
LD1

## Classification Trees

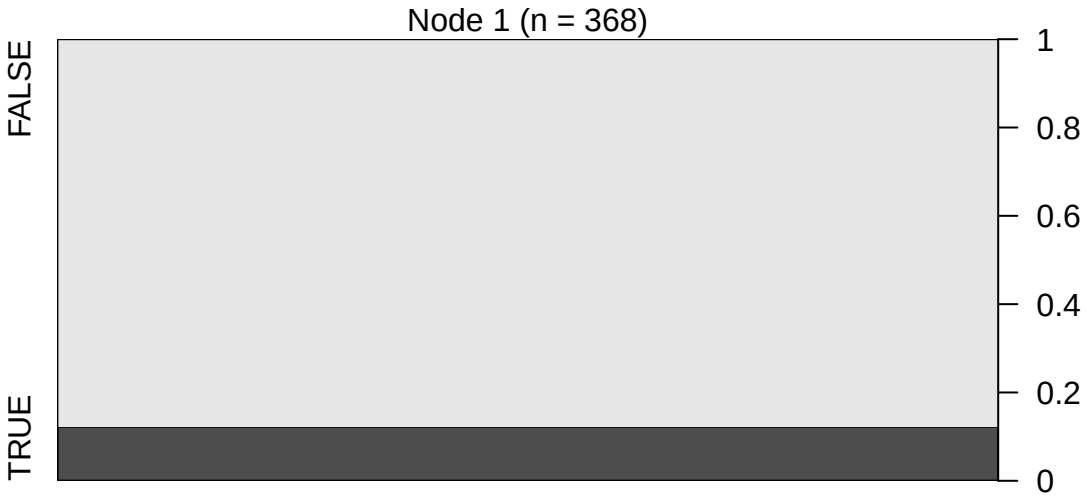
### Tree One

Lastly, for the models, I considered a classification tree. These algorithms create models that predict the response based on splitting the data through "leaves" on the tree that come to an outcome. These models work neatly for a binary response, and so I thought these would provide fair predictions for the data.

I initially formed a tree that used every predictor, which is graphed below. While the model does provide a simplified method of classification, the classification accuracy for the both this tree is 86%. The pruning process removed all of the predictors from the model, leaving just the bare probability of how many of the songs are Top 10 hits. The classification accuracy for this model is also 86%.


Unpruned Tree through Cross Validation
1
tempo
≥ 124
3
duration_ms
≥ 262180
< 262180
5
duration_ms
< 255806.5
≥ 255806.5
6
speechiness
≥ 0.122
8
speechiness
< 0.1
≥ 0.106
Node 2 (n = 136)
Node 4 (n = 70)
Node 7 (n = 44)
Node 9 (n = 100)
Node 10 (n = 7)
Node 11 (n = 11)
FALSE
TRUE
0.8
0

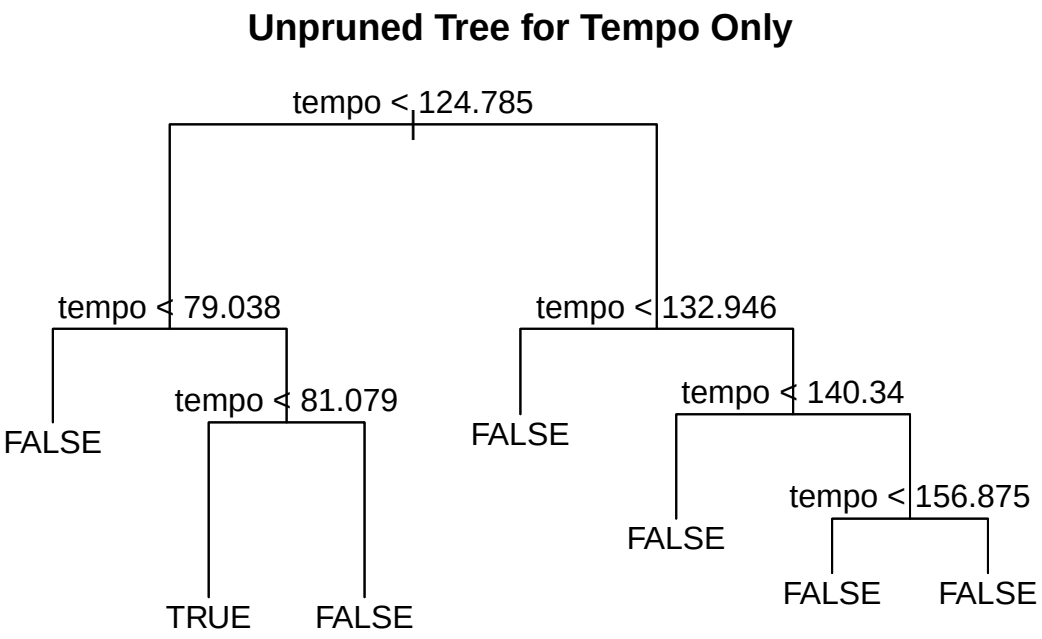
Pruned Tree Found Through Cross Validation



**Tree Two**

Lastly, I modified the Tree model. I noticed that the *Tempo* attribute was the root node, and none of the tree models produced considered this predictor on its own. I created a model based on this predictor alone.

This model provided a stronger correlation than the original tree models. The classification accuracy for this model is 90%. While not the strongest model, there is some predictive power within this tree model. As seen on the next page, a majority of the leaves of this tree lead to a "FALSE" prediction, and this could be further simplified.


Unpruned Tree for Tempo Only
tempo < 124.785
tempo < 79.038
tempo < 132.946
FALSE
tempo < 81.079
FALSE
tempo < 140.34
TRUE
FALSE
FALSE
tempo < 156.875
FALSE
FALSE

# Model Discussion & Analysis

Overall, the models did provide some valuable models for the data. The Logistic Regression and final Tree Model were both the strongest, with identical classification accuracies. The Linear Discriminant Anlysis (LDA) was the secdonary to these models, with a classification accuracy of 89%. Finally, the other tree models were the weakest, with a classification accuracy of 86%. There was generally a lot of noise in the data, which contributed to these models.

The Logistic Regression model is the best model in this instance. It's simplicity combined with it's predictive power means that it's a better choice than a tree that could be obscured or possibly confusing under further scrutiny. The summary information below, *Tempo* and *Speechiness* have the lowest p-values, showing that these variables contribute the most to the model.

```
##
## Call:
## NULL
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       4.461e+00  3.483e+00   1.281   0.2003
## danceability      5.335e-01  1.564e+00   0.341   0.7330
## energy           -2.034e-02  1.622e+00  -0.013   0.9900
## key               3.736e-02  4.815e-02   0.776   0.4377
## loudness          9.068e-02  1.058e-01   0.857   0.3913
## mode              4.545e-02  3.598e-01   0.126   0.8995
## speechiness      -6.011e+00  2.618e+00  -2.296   0.0217 *
## acousticness     -3.351e-01  9.988e-01  -0.336   0.7372
## instrumentalness  2.069e+00  1.611e+00   1.284   0.1990
## liveness         -2.077e+00  1.521e+00  -1.365   0.1721
## valence           2.665e-01  9.460e-01   0.282   0.7781
## tempo            -1.659e-02  7.135e-03  -2.325   0.0201 *
## duration_ms      -3.168e-06  4.468e-06  -0.709   0.4784
## time_signature   -7.827e-01  5.681e-01  -1.378   0.1683
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 273.39  on 367  degrees of freedom
## Residual deviance: 252.65  on 354  degrees of freedom
## AIC: 280.65
##
## Number of Fisher Scoring iterations: 6
```

Given that Logistic Regression is a transformation of Linear Regression, it makes sense that this model would be chosen, since often the simplest models, such as a Linear Model, are the optimal model.

# Conclusions

For the Billboard Hot 100, my conclusion is that there are no aspects of a song that necessarily indicate it will be a Top 10 hit. The one quality that seemed to stand out within the Tree model and Logistic Model was the *Tempo* of a song. It indicated that a tempo between 79.038 and 81.079 would predict a Top 10 hit. This is a fairly slow tempo, which I'm surprised at. While this model isn't a great predictor, it would be interesting to explore more the affect that *Tempo* specifically has on the possibility of a song being a Top 10 Hit.

There are some considerations to my analysis. First, the dataset I used was made up of all charting songs, even if they didn't make it to the Top 10. These songs are unequivocally popular despite their position, so it would be interesting to see, with a wider data set, if including non-charting songs would provide better models.

Through this project, I've learned more of the broad strokes of how our classroom models are implemented in a real world context. The homework assignments entirely ask for a specific model within a question, and for good reason, to test our knowledge from that week. However, this project allowed me to pick for myself which models to use, which helped me learn more of the requirements for each model and why picking them could be advantageous.

Further looking into this topic could be improved in the future. Further narrowing of the model formula could be beneficial, as it would allow for a more interpretable result. It could also be interesting to see the relationship between artists and the charts. Popular artists have a much larger audience, leading to higher charting. Looking into specific charting placement could be valuable as well. Would it be possible to predict the specific position on the chart from the given information?

All in all, this project allowed me to learn more about the course material through a topic I was interested in. It showed me the ways that data can be manipulated and modeled, giving me insight to the course that I am able to take into my futher classes and career.

# Citations

Billboard Hot-100[2000-2023] data with features. (2024). [Dataset; Csv]. In Kaggle (Version 1). https://www.kaggle.com/datasets/suparnabiswas/billboard-hot-1002000-2023-data-with-features

# Code Appendix

## Appendix A - Computing Code

### Code A.1 - Data Import

```
#import the data file
musicData <- read.csv("C:[FILE PATH]\\billboard_24years_lyrics_spotify.csv")
```

### Code A.2 - Data Processing

```
#change the band/singer column to "artist"
musicData <- musicData %>% rename_at('band_singer', ~'artist')

#remove duplicate songs, save peak week
musicData <- musicData %>% group_by(song, artist) %>%
  arrange(song, ranking) %>%
  filter(row_number() == which.min(ranking)) %>% ungroup()

#remove unnecessary columns
musicData <- musicData %>% dplyr::select(-titletext, -url,
                                -lyrics, -uri, -type, -id,
                                -track_href, -analysis_url)

#remove rows without data
musicData <- musicData %>% drop_na(duration_ms)

#add top 10 column
musicData <- musicData %>% mutate( top10 = ifelse( ranking <= 10, TRUE, FALSE  ) )

#make it a t/f a factor
musicData$top10 <- as.factor(musicData$top10)
```

### Code A.3 - Make Training/Test Sets & Create Formula Variable

```r
#create training & test data
train <- sample( 1:nrow(musicData),  0.8*nrow(musicData), replace = FALSE,  )
trainData <- newMusic[train, ]
testData <- newMusic[-train, ]

# define the formula
formula <- top10 ~ danceability+energy+key+loudness+mode+speechiness +
                          acousticness+instrumentalness+liveness+valence+
                          tempo+duration_ms+time_signature
```

**Code A.4 - Logistic Regression**

```r
set.seed(2024)

# Define the train control with scaling
trainControl <- trainControl(method = "cv", number = 10)

# Train the KNN model with scaling and centering
logisticReg <- train(formula, data = trainData,
                          method = "glm",
                          family = binomial,
                          trControl = trainControl)

# predict for the test data
logPreds <- predict(logisticReg, newdata = testData)

#make the confusion matrix
logConfMatrix <- confusionMatrix(logPreds, testData$top10)

logClassAcc <- (logConfMatrix$table[1,1] + logConfMatrix$table[2,2]) /
                  sum(logConfMatrix$table)
```

**Code A.5 - Linear Discriminant Analysis**

```r
#fit the LDA model
lda.fit <- MASS::lda(formula, data = musicData, subset = train)

#predict on the test set
lda.pred <- predict(lda.fit, musicData[-train,])

#get the classification accuracy
ldaClassAcc <- mean(lda.pred$class==musicData[-train,]$top10)
```

**Code A.6 - Tree One**

```
# Fit a decision tree using rpart
rpartFit <- rpart( formula, trainData, method="class")

# Identify the value of the complexity parameter that produces
# the lowest CV error
cpMin <- rpartFit$cptable[which.min(rpartFit$cptable[,"xerror"]),"CP"]

# Prune using the CV error minimizing choice of the complexity parameter cp
rpartFitPruned <- prune(rpartFit, cp = cpMin)

# Convert pruned tree to a party object
originalFitPartyObj <- as.party( rpartFit )
prunedFitPartyObj <- as.party(rpartFitPruned)

#calculate the classification accuracy
unpTreeCVPreds <- predict(rpartFit, testData, type="class")
unpConfusionPreCV <- table(unpTreeCVPreds, testData$top10)
unpAcc <- (unpConfusionPreCV[1,1] + unpConfusionPreCV[2,2]) /
                        sum(unpConfusionPreCV)

#calculate the classification accuracy
prunedTreeCVPreds <- predict(rpartFitPruned, testData, type="class")
prunedConfusionPreCV <- table(prunedTreeCVPreds, testData$top10)
prunedAcc <- (unpConfusionPreCV[1,1] + unpConfusionPreCV[2,2]) /
                        sum(unpConfusionPreCV)
```

**Code A.7 - Tree Two**

```
#make a tree model with just tempo
myTree <- tree( top10 ~ tempo, trainData, method="class" )

#calculate the classification accuracy
myTreeCVPreds <- predict(myTree, testData, type="class")
myTreeConfusionPredCV <- table(myTreeCVPreds, testData$top10)

myTreeAcc <- (myTreeConfusionPredCV[1,1] + myTreeConfusionPredCV[2,2]) /
                        sum(myTreeConfusionPredCV)
```

## Appendix B - Figure and Table Composition

**Code B.1 - Print Unprocessed Data**

```r
#create a dataframe that has 7 max characters
## makes the printed table shorter (fits in one page now)
briefData <- musicData %>% mutate(across(everything(),
                            ~ ifelse(nchar(.) > 7,
                                    paste0(substr(., 1, 7), "..."), .)))

#print the first 5 rows of table (wrapped)
pander(head(briefData, 5), caption = "First five rows of the unprocessed data")

#remove the brief data obj
rm(briefData)
```

**Code B.2 - Print Processed Data**

```r
#print the first 5 rows of the processed data
pander(head(musicData, 5), caption = "First five rows of the processed data")
```

**Code B.3 - ROC Plot for Logistic Regression Model**

```r
#get the predicted values
predictedProbs <- predict(logisticReg, newdata = trainData, type = "prob")[,2]

# Generate ROC curve
rocCurve <- roc(trainData$top10, predictedProbs)

# Plot ROC curve
plot(rocCurve, main = "ROC Curve for Logistic Regression")

#remove the objects to declutter
rm(predictedProbs,rocCurve)
```

**Code B.4 - Linear Discriminant Analysis Plot**

```r
#define data to plot
lda_plot <- cbind(trainData, predict(lda.fit)$x)

#create plot
ggplot(lda_plot, aes(LD1, top10)) +
  geom_point( aes(color = top10) ) +
  labs( title = "LDA Plot for Top 10 Hit" ) +
  theme_bw() +
  theme(legend.position="none")
```

**Code B.5 - Unpruned Tree One Graph**

```
#plot the unpruned tree
plot(originalFitPartyObj, tp_args = list(text = "vertical", ymax = 1.5),
     main = "Unpruned Tree through Cross Validation")
```

### Code B.6 - Pruned Tree One Graph

```
#plot the pruned tree
plot(prunedFitPartyObj, main = "Pruned Tree Found Through Cross Validation")
```

### Code B.7 - Tree Two Graph

```
# Convert the object to a party obj and plot it
plot(myTree)
title("Unpruned Tree for Tempo Only")
text(myTree, pretty=0)
```

### Code B.8 - Summary of Logistic Regression

```
#get the summary information for the logistic model
summary(logisticReg)
```