Attributes Related to Song Popularity on Spotify

This notebook explores the different attributes of a song on Spotify to understand how each song attribute contributes to the overall popularity of a song on Spotify. It focuses on determining the impact of each attribute by using regression analysis methods.

Project Overview: For decades music has played an influential role in shaping various aspects of society, and it has always found a way to make us feel powerful emotions. While music is known to be subjective, the rise of the internet and streaming services have allowed us to get a better understanding of what kind of songs tend to appeal to the majority of the population. What is it about the songs that make them so popular that we hear it everywhere we go, and see it appear on our music streaming platforms? Spotify is one of the most used music platform with over 217 million monthly listeners around the world. As one of the biggest music streaming service, Spotify has developed its own metric to identify the popularity of a track by rating them out of 100. The goal is to understand the impact of different song attributes on the track popularity metric used by Spotify. This project will also explore the significance of certain words in a song title that may make a song more popular.

Project Relevance: Producers and writers can use the models and criteria to guide their process when creating a new song and/or evaluating older songs to understand trends. For instance, according to the models, producers can aim to include variables that are correlated with higher song popularity. Singers can choose which song on a new album is best to release as a single/teaser to generate excitement by first using the model to predict which song on their new album will be the most popular. Background music and sounds can play a significant role in people's mood and energy. The findings can also help marketing firms and event organizers refer to a criteria to decide on what music and sounds to play to create a specific kind of mood for ads or events.

Data: To explore the question above, a dataset was downloaded that includes the top 19,000 popular songs on Spotify. The full dataset can be found here: https://www.kaggle.com/edalrami/19000-spotify-songs (https://www.kaggle.com/edalrami/19000-spotify-songs)

Dataset: The original songs dataset contains 18,836 rows and 15 columns, and it is 2.11 MB. Each row is for each unique song, the first column is for the track title, and the remaining 14 columns are for different song attributes. A new dataset was created that is a modified version of the original dataset with 14,688 rows and 121 columns. Note 106 new columns are added with 103 columns for all unique words in a track title that appear on 50 or more songs, and the remaining 3 columns are different versions of the track title based on unique symbols and characters.

The following columns were used for analysis:

song_name1 - song_name4 - The title of the track; note there are 4 columns for song name with each column more cleaned up to accommodate for certain song titles with unique symbols and characters.

song_popularity - The popularity of a track is a value between 0 and 100, with 100 being the most popular. The popularity is calculated by an algorithm and is based, for the most part, on the total number of plays the track has had and how recent those plays are.

song_duration_ms - The duration of the track in milliseconds.

acousticness - How much natural acoustic sounds the track consists. 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 - 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 - 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.

instrumentalness - “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.

key - The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.

liveness - The presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

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 typical range between -60 and 0 db.

audio_mode - Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

speechiness - Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

tempo - The overall estimated tempo of a track in beats per minute (BPM). Tempo is the speed or pace of a given piece and derives directly from the average beat duration.

time_signature - The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

audio_valence - A measure from 0 to 1 describing the musical positiveness conveyed by a track. Songs with higher valence are more happy/positive.

Words in Song Titles - The one hundred most popular words in song titles, excluding English and Spanish stop words like “the” found here:https://www.ranks.nl/stopwords (https://www.ranks.nl/stopwords). Each column indicates a 0,1,2 etc. for how many times the word appears in each song’s title. An example of the top 100 words includes: “Me”, “Love”, “Baby”, “Amor” Remastered“,”Featuring“,”Paris“,”Tears", etc.

Data Cleaning - As mentioned above, the original dataset contained song name, song popularity and 13 variables related to the music and voices in the song. To create a more robust dataset, variables were created out of the unique words found in song titles. To create these new columns the following procedures were done in Excel. First, strip down the song names to exclude special characters like “?” and “,” so that each word in the song was isolated and separated by just a space. Then split each song title up so that each word was in an individual cell and de-duplicated the list so that one list of all of the unique words that appeared. The de-

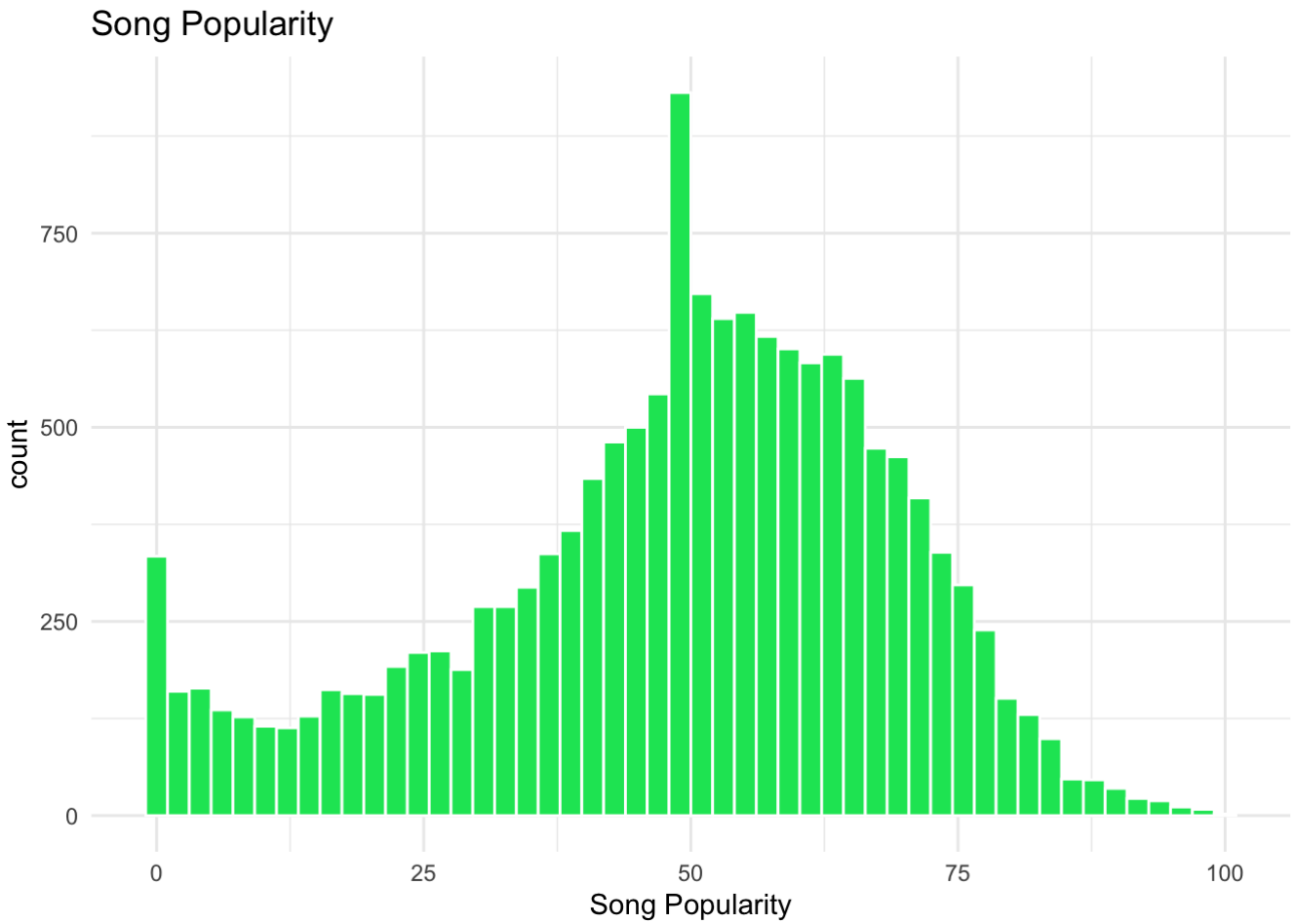
duplicated list contained over 10,000 words. Next, remove all of the English and Spanish “stop words” included in the list. Finally, use a count if function to fill in for each word-column a value, 0, 1, etc. for how many times each word appeared in a song title. The columns were limited to just include the top 100 words.

```
#install.packages(c("tidyverse","forecast","fpp"))
#install.packages("randomForest")
#install.packages("gbm")
library(tidyverse)
library(readxl)
library(ggplot2)
library(forecast)
library(fpp)
library(glmnet)
library(stringr)
library(magrittr)
library(ggthemes)
library(readr)
library(randomForest)
library(gbm)
theme_set(theme_bw())

spotify <- read.csv("song_data_no_outliers_100 col 10.9 better words.csv")
set.seed(1234)
# This will split into train and test 70-30
spotify$train <- sample(c(0, 1), nrow(spotify), replace = TRUE, prob = c(.3, .7))
test <- spotify %>% filter(train == 0)
train <- spotify %>% filter(train == 1)
xnames <- colnames(spotify)
xnames <- xnames[!xnames %in% c("song_name1", "song_name2",
                                "song_name3", "song_name4",
                                "song_popularity","train")]
```

Data Exploration

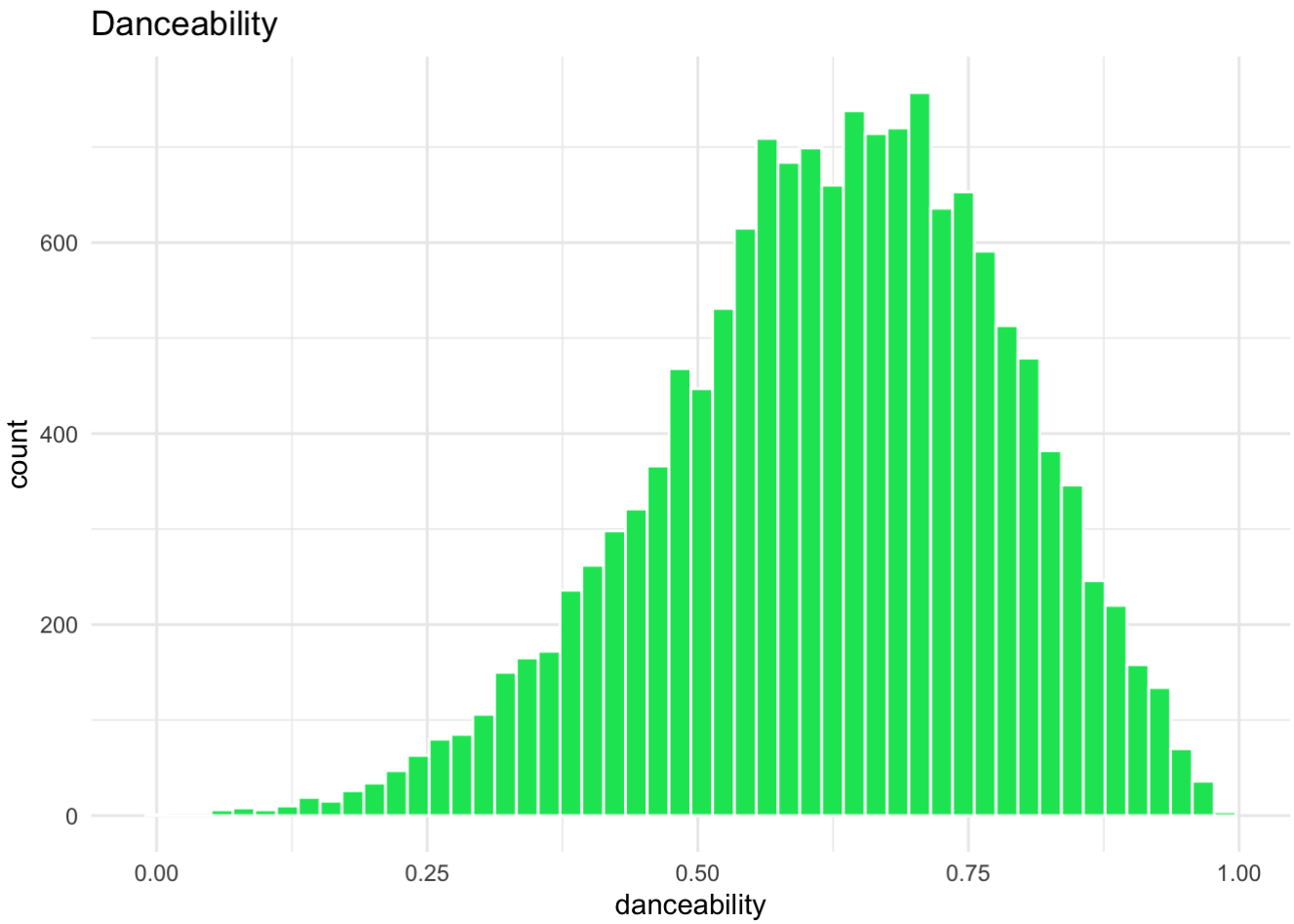
Song Popularity: Song Popularity is slightly skewed to the left with most song popularity ratings around 50 to 65, there are also more songs with a rating lower than 25 compared to songs with 75+ rating.

```
ggplot(spotify, aes(x = song_popularity)) +
  geom_histogram(fill ="#00E463", color = "white", bins = 50) +
  labs(title="Song Popularity",
       x = "Song Popularity") +
  theme_minimal()
```



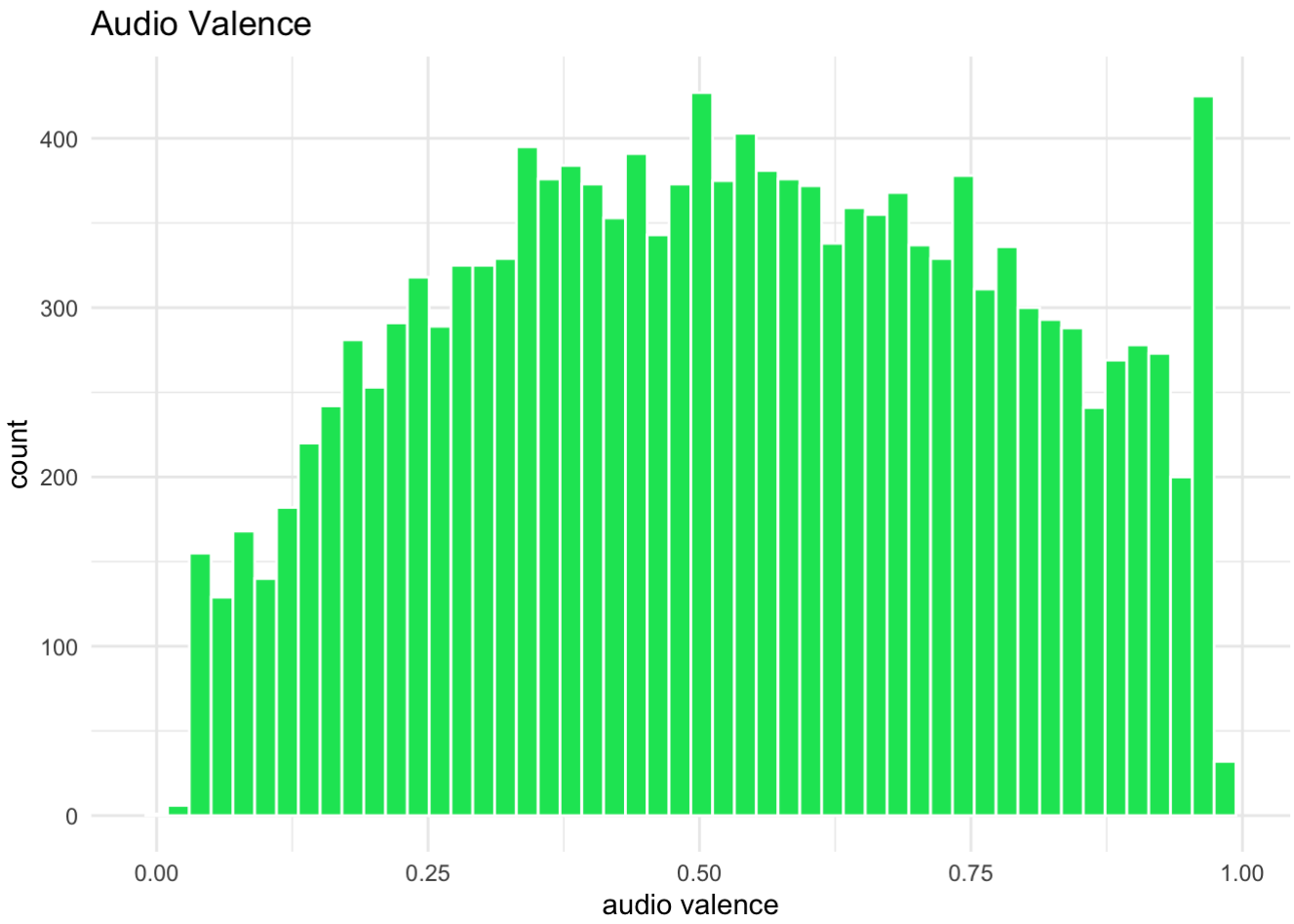
Danceability This attribute measures how danceable a song is within a scale of 0 to 1, the majority of the songs are between a danceability score of 0.5 to 0.75. There are more songs with a danceability of 0.75 or more compared to songs with a score of 0.25 or less.

```
ggplot(spotify, aes(x = danceability)) +
  geom_histogram(fill ="#00E463", color = "white", bins = 50) +
  labs(title="Danceability",
       x = "danceability") +
  theme_minimal()
```



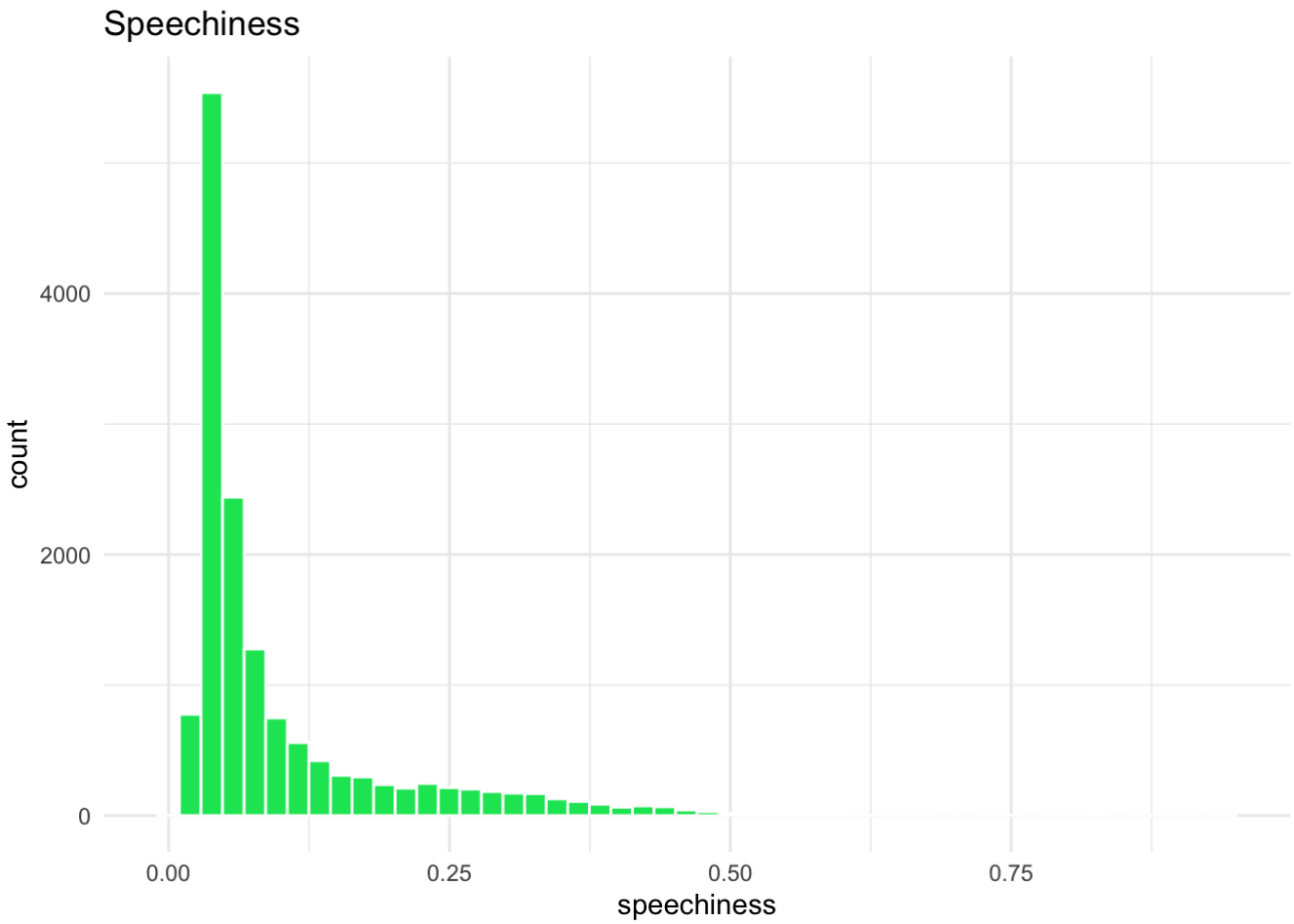
Audio Valence This attribute is also measured on a scale of 0 to 1, with 1 being a very happy song and 0 being a very sad song. There are more happy tracks in the dataset than sad songs, and there also appears to be a high number of really happy songs with a score of about 0.9.

```
ggplot(spotify, aes(x = audio_valence)) +
  geom_histogram(fill ="#00E463", color = "white", bins = 50) +
  labs(title="Audio Valence",
       x = "audio valence") +
  theme_minimal()
```



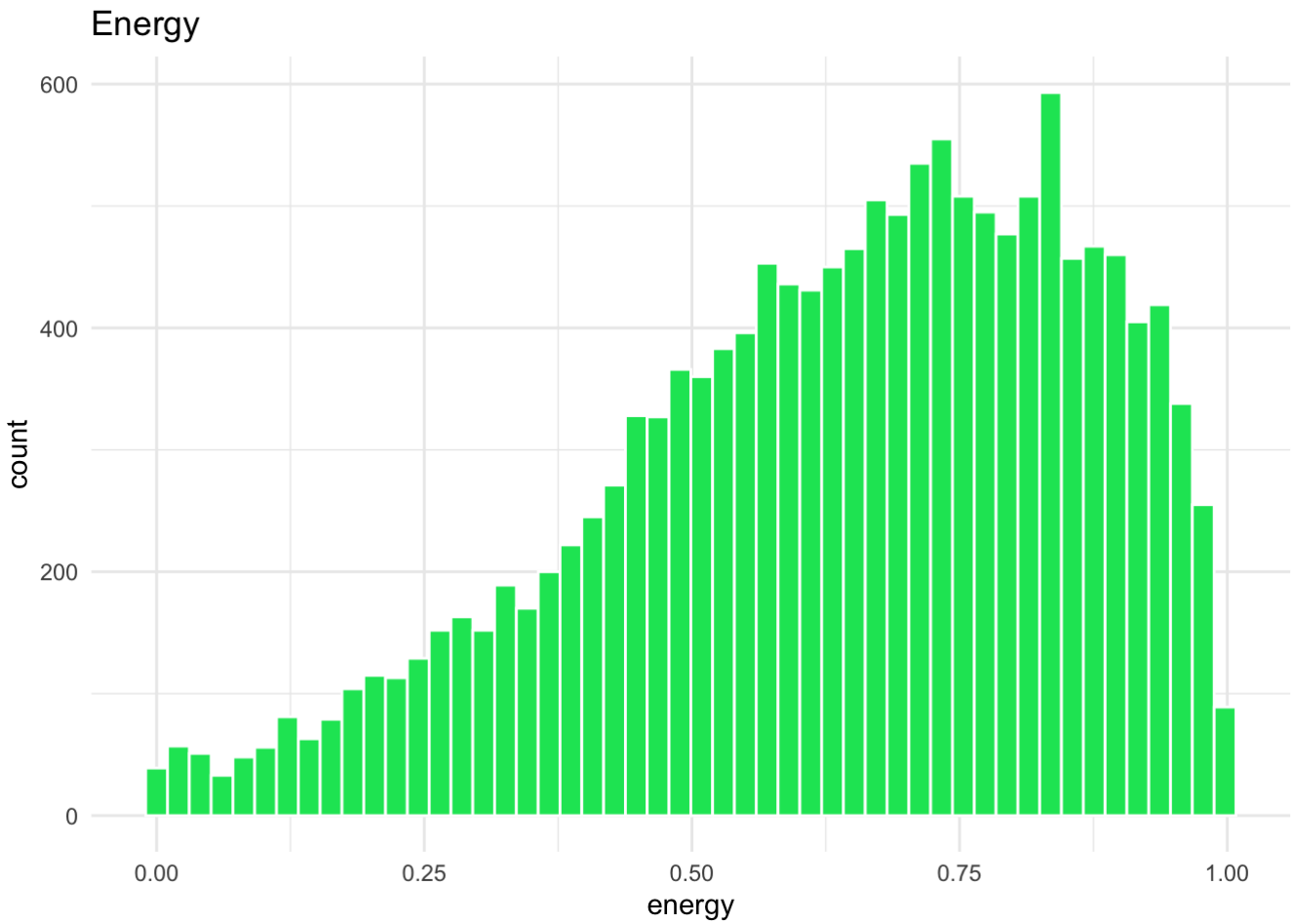
Speechiness Speechiness is on a scale of 0 to 1, and a very big portion of the dataset are tracks with low levels of speechiness, which means the distribution is highly skewed to the right and most songs contain more music than words.

```
ggplot(spotify, aes(x = speechiness)) +
  geom_histogram(fill ="#00E463", color = "white", bins = 50) +
  labs(title="Speechiness",
       x = "speechiness") +
  theme_minimal()
```



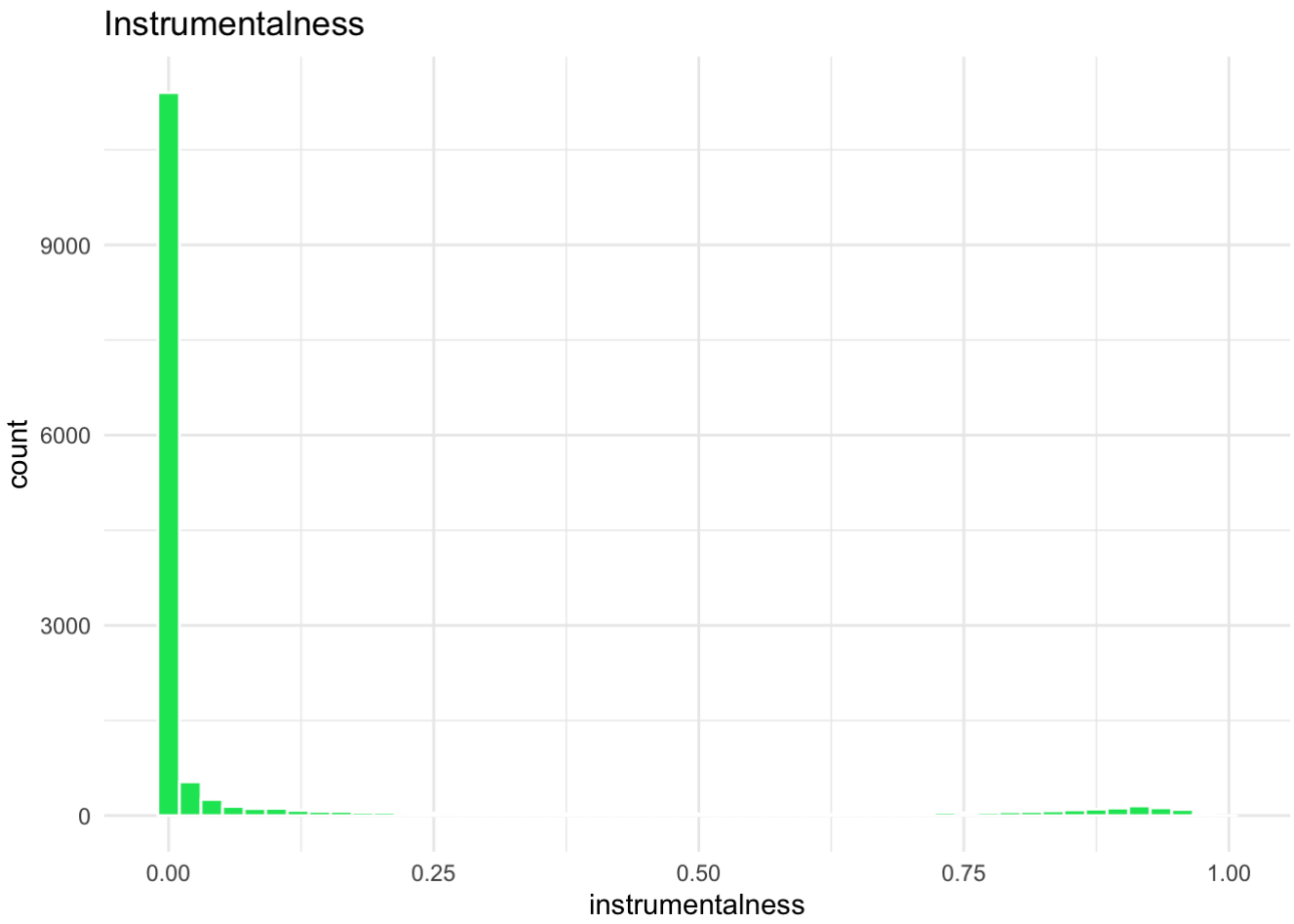
Energy This attribute appears to have a distribution highly skewed to the left with a majority of the tracks having an energy score of over 0.6 out of 1.

```
ggplot(spotify, aes(x = energy)) +
  geom_histogram(fill ="#00E463", color = "white", bins = 50) +
  labs(title="Energy",
       x = "energy") +
  theme_minimal()
```



InstrumentalnessOver 90% of the tracks in this dataset are not complete instrumental versions of songs and therefore, this attribute's distribution is all on the left towards an instrumentalness score of almost 0.

```
ggplot(spotify, aes(x = instrumentalness)) +
  geom_histogram(fill ="#00E463", color = "white", bins = 50) +
  labs(title="Instrumentalness",
       x = "instrumentalness") +
  theme_minimal()
```



Forward Stepwise Regression

```
# forward selection
fit_fw <- lm(song_popularity ~ 1, data = train)
yhat_train <- predict(fit_fw, train)
mse_train <- mean((train$song_popularity - yhat_train)^2)
yhat_test <- predict(fit_fw, test)
mse_test <- mean((test$song_popularity - yhat_test)^2)
mse_test
```

```
## [1] 419.2422
```

```
#log of results
log_fw <-
 tibble(
   xname = "xname",
   model = paste0(deparse(fit_fw$call), collapse = ""),
   mse_train = mse_train,
   mse_test = mse_test
 )

xnamesfw <- xnames

while (length(xnamesfw) > 0) {
 best_mse_train <- NA
 best_mse_test <- NA
 best_fit_fw <- NA
 best_xname <- NA
 # select the next best predictor
 for (xname in xnamesfw) {
   fit_fw_tmp <- update(fit_fw, as.formula(paste0(". ~ . + ", xname)))
   yhat_train_tmp <- predict(fit_fw_tmp, train)
   mse_train_tmp <- mean((train$song_popularity - yhat_train_tmp) ^ 2)
   yhat_test_tmp <- predict(fit_fw_tmp, test)
   mse_test_tmp <- mean((test$song_popularity - yhat_test_tmp) ^ 2)
   if (is.na(best_mse_test) | mse_test_tmp < best_mse_test) {
     best_xname <- xname
     best_fit_fw <- fit_fw_tmp
     best_mse_train <- mse_train_tmp
     best_mse_test <- mse_test_tmp
     }
 }
 log_fw <- log_fw %>% add_row(
   xname = best_xname,
   model = paste0(deparse(best_fit_fw$call), collapse = ""),
   mse_train = best_mse_train,
   mse_test = best_mse_test )
 fit_fw <- best_fit_fw
 xnamesfw <- xnamesfw[xnamesfw!=best_xname]
}

log_fw
```

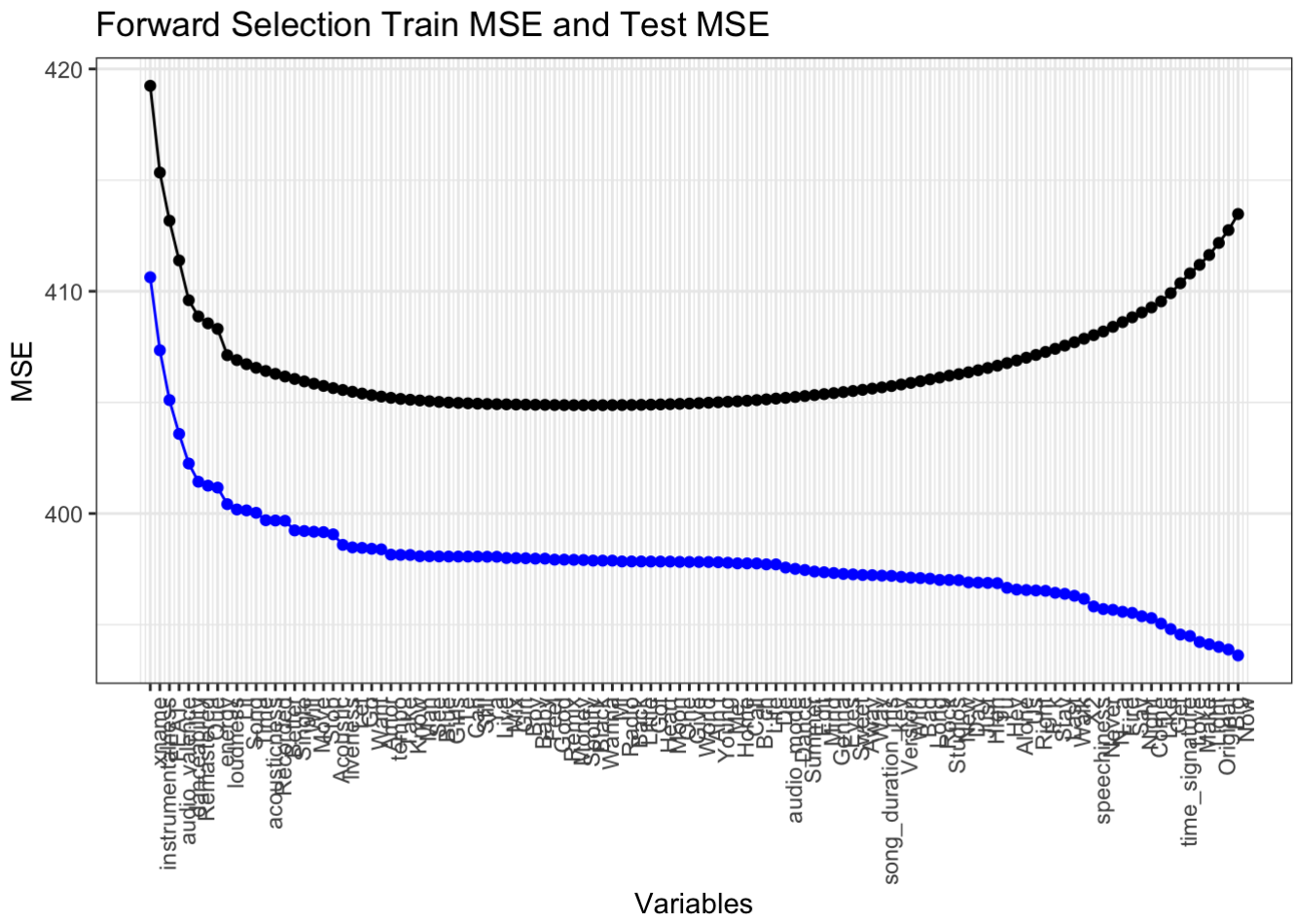
xname <chr>
xname
instrumentalness
FEAT.
audio_valence
danceability

xname <chr>
Remastered
Que
energy
loudness
Lil

1-10 of 114 rows | 1-1 of 4 columns Previous **1** 2 3 4 5 6 … 12 Next

Train vs. Test MSE for Forward Selection:

```
ggplot(log_fw, aes(seq_along(xname), mse_test)) +
  geom_point() +
  geom_line() +
  geom_point(aes(y=mse_train), color="blue") +
  geom_line(aes(y=mse_train), color="blue") +
  scale_x_continuous("Variables", labels = log_fw$xname, breaks = seq_along(log_fw$xnam
e)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Forward Selection Train MSE and Test MSE") +
  ylab("MSE")
```



Minimum test MSE for forward selection:

```
log_fw[which.min(log_fw$mse_test),]$mse_test
```

```
## [1] 404.8752
```

The full model for forward selection:

```
log_fw[which.min(log_fw$mse_test),]$model
```

```
## [1] "lm(formula = song_popularity ~ instrumentalness + FEAT. + audio_valence +     da
nceability + Remastered + Que + energy + loudness + Lil +     Song + Time + acousticness
+ Recorded + Better + Single +     Will + Move + Stop + Acoustic + liveness + Let + Go +
Want +     Amor + tempo + Take + Know + Man + See + Blue + Girls + Te +     Can + Still
+ Ya + Live + Mix + Girl + Boy + Baby + Feel +     Good + Remix + Money + Spotify + Blac
k, data = train)"
```

Forward selection coefficients:

```
best_fw_model <- as.formula("song_popularity ~ instrumentalness + FEAT. +
                            audio_valence + danceability + Remastered + Que + energy +
                            loudness + Lil + Song + Time + acousticness + Recorded + Bett
er + Single +
                            Will + Move + Stop + Acoustic + liveness + Let + Go + Want +
                            Amor + tempo + Take + Know + Man + See + Blue + Girls + Te +
                            Can + Still + Ya + Live + Mix + Girl + Boy + Baby + Feel +
                            Good + Remix + Money + Spotify + Black")
best_fw_model <- lm(best_fw_model, data = train)
coef(best_fw_model)
```

```
##      (Intercept) instrumentalness            FEAT.    audio_valence
##      59.85915269      -6.47137108       4.86980669      -5.03864043
##     danceability       Remastered              Que           energy
##       4.86386966       6.08675055      -6.88826413      -7.66534929
##         loudness              Lil             Song             Time
##       0.33873330       8.94312045       3.11588235       3.81288669
##     acousticness         Recorded           Better           Single
##      -2.87585030      -0.11136989       1.69961703      -7.06497297
##             Will             Move             Stop         Acoustic
##       3.56570863      -3.54599423       2.36951512       3.67473507
##         liveness              Let               Go             Want
##      -4.98526310       5.63582645      -1.78422384       2.46395529
##             Amor            tempo             Take             Know
##      -2.83547310      -0.01743994       1.52841962      -0.77062954
##              Man              See             Blue            Girls
##      -2.88728999      -0.64659685      -1.44038173       1.34768560
##               Te              Can             Still               Ya
##       0.49750749      -0.41047909       0.62431415      -1.62802852
##             Live              Mix             Girl              Boy
##       0.04920054      -2.67945301       1.03944135      -1.22858912
##             Baby             Feel             Good            Remix
##      -1.80773717       0.69630915      -2.61655446      -0.04286203
##            Money          Spotify            Black
##      -1.70433629       2.90539026       2.30167772
```

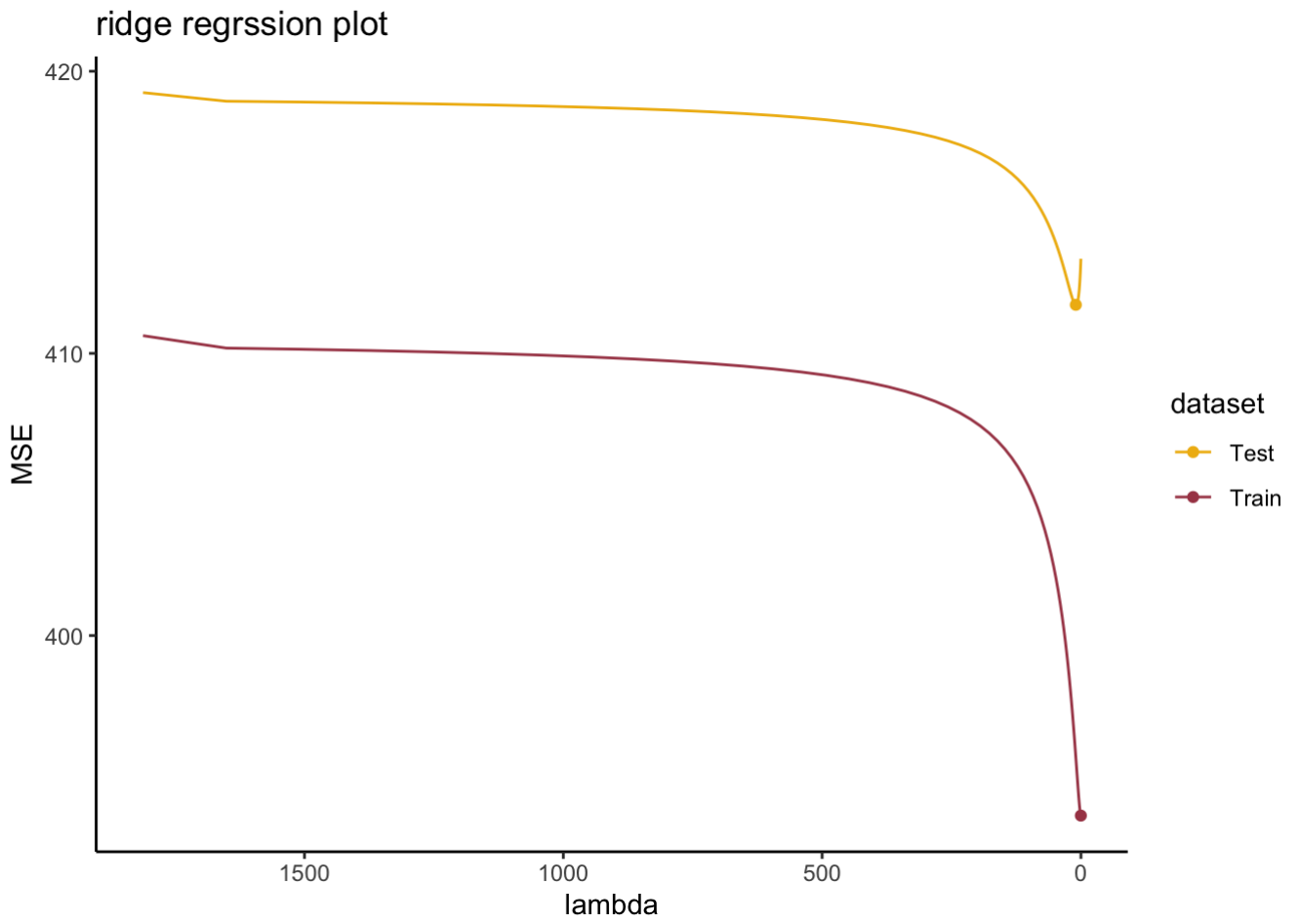
Forward Selection Summary:

The final forward selection model contained 46 variables. As expected, many of the variables describing the characteristics of the music including instrumentalness, audio valence, danceability, energy, loudness, acousticness, liveness and tempo were included in the model. Some of the first words from song titles to be added to the model included “Feat.”, “Remastered”, “Lil” and “Live”. It is interesting that these words were all added first because they all had positive coefficient values and because they all tell us something more about the song. Both “Feat.” and “Lil” tell us that more than one artist collaborated on the song while “Remastered” and “Live” tell us that this was not the first time this song was released. But, because “live” can refer to a live performance or living it can be difficult to infer which meaning of the word has a higher correlation with song popularity.

Ridge Regression

```
## linear regression ##
lf <- "song_popularity ~ song_duration_ms"
for (i in 2:length(xnames)) {
  as.character(xnames[i])
  lf <- paste(lf, "+", xnames[i], sep = " ")
}

f <- as.formula(lf)

x_train <- model.matrix(f, train)[, -1]
y_train <- train$song_popularity
x_test <- model.matrix(f, test)[, -1]
y_test <- test$song_popularity

## ridge ##
fit_ridge <- cv.glmnet(x_train,
                       y_train,
                       alpha = 0,
                       nfolds = 10)

mse_train <- vector()
for (i in 1:length(fit_ridge$lambda)) {
  yhat_train_ridge <- predict(fit_ridge, x_train, s = fit_ridge$lambda[i])
  mse_train_tmp <- mean((y_train - yhat_train_ridge)^2)
  mse_train <- append(mse_train, mse_train_tmp)
}

mse_test <- vector()
for(i in 1:length(fit_ridge$lambda)) {
  yhat_test_ridge <- predict(fit_ridge, x_test, s = fit_ridge$lambda[i])
  mse_test_tmp <- mean((y_test - yhat_test_ridge)^2)
  mse_test <- append(mse_test, mse_test_tmp)
}

rid_mse <- tibble(
  lambda_rid = fit_ridge$lambda,
  mse = mse_train,
  dataset = "Train"
)

rid_mse <- rbind(rid_mse, tibble(
  lambda_rid = fit_ridge$lambda,
  mse = mse_test,
  dataset = "Test"
))

## min MSE and related lambda ##
df <-spread(rid_mse, key = dataset, value = mse)
te <- df[which(df$Test== min(df$Test)), ]
ta <- df[which(df$Train== min(df$Train)), ]
sub <-rbind(ta, te)
sub %>%
  gather("Test", "Train", key = "dataset", value = "mse") -> sub
```

```
sub <- sub[c(2,3), ]

## plots ##
rid_mse %>%
  ggplot(aes(x = lambda_rid, y = mse)) +
  geom_line(aes(color = dataset)) +
  scale_colour_manual(values =c("#efb810", "#a84554")) +
  scale_x_reverse() -> p
p + geom_point(data = sub,aes(color = dataset)) +
  labs(title = "ridge regrssion plot", x = "lambda", y = "MSE") +
  theme_classic()
```



Minimum test MSE Ridge Regression:

```
sub
```

lambda_rid	dataset	mse
<dbl>	<chr>	<dbl>
9.8969417	Test	411.7218
0.1811871	Train	393.6199

2 rows

Coefficients for Ridge Regression with Minimum Test MSE:

```
## coefficients ##
lambda_min_mse_test <- as.numeric(sub[1,1])
coef(fit_ridge, s = lambda_min_mse_test)
```

```
## 114 x 1 sparse Matrix of class "dgCMatrix"
##                                1
## (Intercept)       4.945912e+01
## song_duration_ms -2.174548e-06
## acousticness     -1.297761e+00
## danceability      4.016249e+00
## energy           -2.557726e+00
## instrumentalness -4.653392e+00
## key              -2.132193e-02
## liveness         -3.457890e+00
## loudness          1.391377e-01
## audio_mode        3.917555e-01
## speechiness      -3.067482e+00
## tempo            -1.196785e-02
## time_signature    1.132877e+00
## audio_valence    -3.410661e+00
## FEAT.             3.710029e+00
## Love             -8.203063e-01
## Remix             1.189378e-02
## Version          -1.204105e+00
## Remastered        3.968936e+00
## Edit             -2.049558e+00
## Like              3.407415e+00
## N...a            -1.121974e+00
## Radio            -3.580082e-01
## De               -3.479495e-02
## F..k              1.453778e+00
## Single           -4.560850e+00
## One              -6.579204e-01
## Go               -1.038287e+00
## Mix              -2.658315e+00
## Back              2.943262e-01
## Acoustic          3.090235e+00
## Time              2.463087e+00
## Get              -3.681722e+00
## Good             -1.552583e+00
## Way               4.227848e-01
## Night            -2.948796e+00
## Girl              3.439341e-01
## Man              -2.187668e+00
## Life              2.064929e+00
## Want              1.812559e+00
## Know             -5.249770e-01
## Mi               -1.193138e-02
## Now               5.398727e+00
## Heart             2.261704e-01
## Baby             -8.433348e-01
## Take              1.120560e+00
## Just              6.528301e-01
## Got              -8.702507e-01
## Come             -2.606219e+00
## Let               3.613939e+00
## Never             3.339220e+00
```

```
## Feel              3.049369e-01
## Black             1.757547e+00
## Day              -3.338691e+00
## Song              2.051315e+00
## Live              7.861839e-02
## Little            3.569520e-01
## Home              1.395033e+00
## Better            1.248173e+00
## Me.               2.357586e+00
## Te               -2.037333e-02
## New               9.615943e-01
## Original          2.896360e+00
## X2               -3.465168e+00
## Make             -5.641476e+00
## World            -3.086389e-01
## Fire              2.970095e+00
## Mind             -1.609218e+00
## Young            -5.071209e-01
## Right             1.537073e+00
## Blue             -8.082380e-01
## Que              -4.910545e+00
## Away             -1.561169e+00
## Bad               1.896706e+00
## Amor             -2.237384e+00
## Dance             2.558293e+00
## Light             4.168162e+00
## Lil               7.249494e+00
## Ain.t            -1.078752e+00
## Say               3.215687e+00
## Tu                1.003059e+00
## Boy              -8.899345e-01
## Eyes             -2.475197e+00
## Summer            2.958677e+00
## Sweet             1.650780e+00
## Still             1.118598e+00
## Money            -5.885053e-01
## Big              -4.634682e+00
## Last              3.222601e+00
## Will              1.874899e+00
## Stay              4.091542e+00
## Hey               5.744248e+00
## Give             -1.265358e+00
## High              1.600851e+00
## Recorded          4.186506e-01
## See              -3.954276e-02
## Girls             1.137724e+00
## Call              4.203204e-01
## Wild             -2.254700e+00
## Moon             -6.268201e-01
## Alone             3.762517e+00
## Move             -2.440399e+00
## Stop              8.814542e-01
## Spotify           1.342772e+00
## B..ch            -7.627150e-01
```

```
## Us                2.622776e+00
## Rock             -3.284121e+00
## Gonna             2.039330e+00
## Studios           7.395127e-01
## Wanna            -7.047871e-01
## Walk              4.051468e+00
## Ya               -4.377847e-01
## Can              -6.467783e-01
## Long              2.164747e+00
```

Ridge Regression Summary

As expected, using ridge regression kept all of the 113 variables in the final model but with very small coefficients. To complete the ridge regression, models for 100 values of lambda were tested and the lambda that resulted in the smallest test MSE was chosen. According to this model, danceability [coefficient = 4.02] and time signature [coefficient = 1.13] have the largest positive correlations out of the Spotify attributes on the popularity of songs while audio valence [-3.41], liveness and instrumentalness had the largest negative relationship with song popularity. The words with the highest positive correlation were “Lil” [coefficient = 7.25] and “Hey” [coefficent = 5.74] while the words with the largest negative correlation were “Que” [-4.91] and “Make” [-5.64]. Overall, because of the large number of variables in the final model, it is difficult to interpret the results of Ridge Regression.

Lasso Regression

```
lf <- "song_popularity ~ song_duration_ms"
for (i in 2:length(xnames)) {
  as.character(xnames[i])
  lf <- paste(lf, "+", xnames[i], sep = " ")
}
f <- as.formula(lf)
x_train <- model.matrix(f,train)[ ,-1]
y_train <- train$song_popularity
x_test <- model.matrix(f,test)[ , -1]
y_test <- test[ ,"song_popularity"]
fit_lasso <- glmnet(x_train,y_train,alpha=1,nlambda = 100)
yhat_train_lasso <- predict(fit_lasso, x_train, s = fit_lasso$lambda)
mse_train_lasso <- (y_train - yhat_train_lasso)^2 %>%
  apply(2,FUN=mean)
yhat_test_lasso <- predict(fit_lasso, x_test, s = fit_lasso$lambda)
mse_test_lasso<- (y_test - yhat_test_lasso)^2 %>%
  apply(2,FUN = mean)
lasso_mse<- tibble(
  lambda = fit_lasso$lambda,
  mse = mse_train_lasso,
  dataset = "Train"
)
lasso_mse <- rbind(lasso_mse, tibble(
  lambda = fit_lasso$lambda,
  mse = mse_test_lasso,
  dataset = "Test"
))
mse_min <- lasso_mse %>%group_by(dataset) %>%filter(mse == min(mse))
lambda_min_mse_train <-as.numeric(mse_min[1,1])
lambda_min_mse_train
```

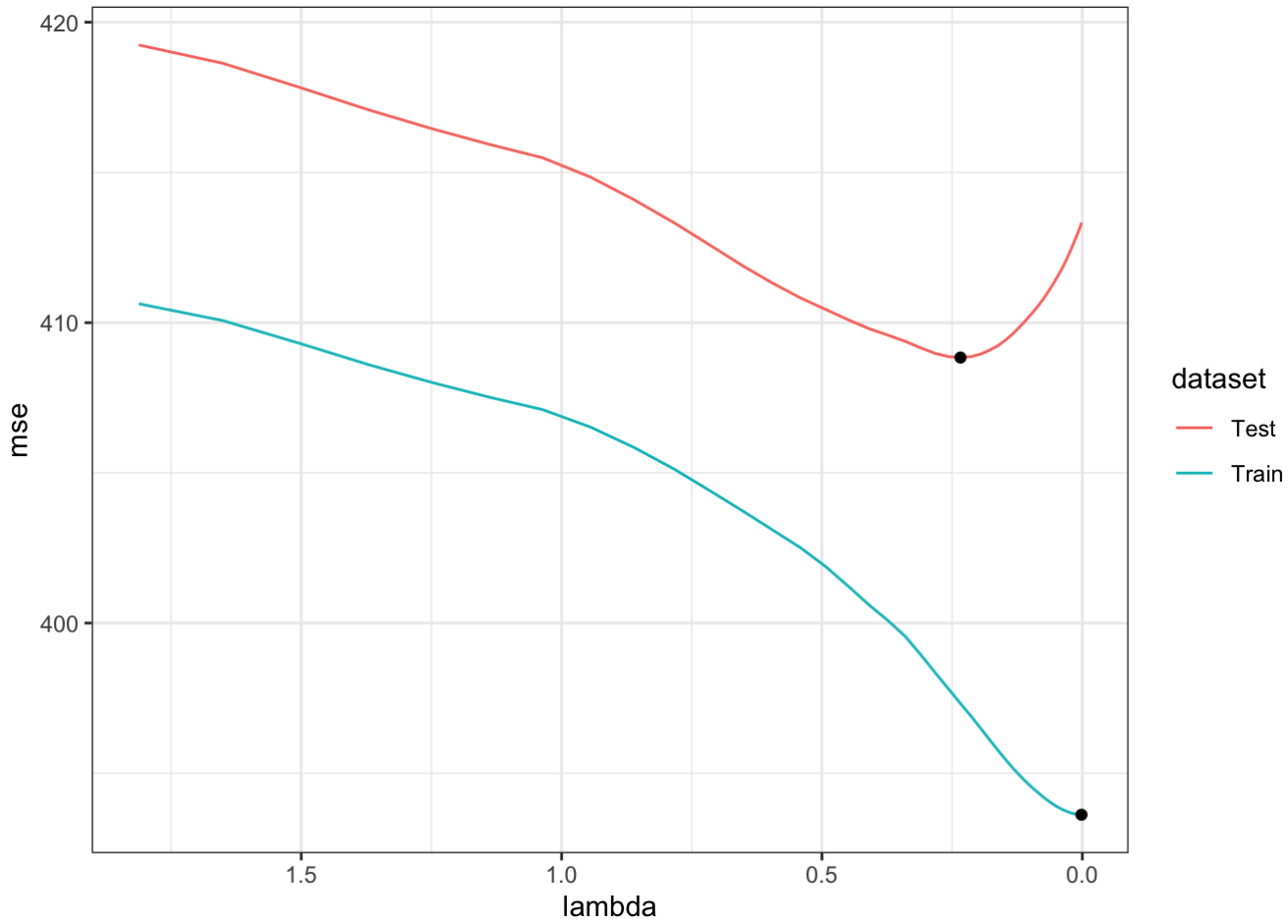
```
## [1] 0.001689757
```

```
lambda_min_mse_test <- as.numeric(mse_min[2,1])
lambda_min_mse_test
```

```
## [1] 0.2340121
```

```
lasso_mse %>% ggplot() +
  geom_line(aes(lambda, mse, color = dataset))+
  scale_x_reverse()+
  geom_point(data=mse_min, aes(lambda, mse))+
  scale_x_reverse()
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```



The minimum training MSE for lasso is:

```
mse_min <- lasso_mse %>%group_by(dataset) %>%filter(mse == min(mse))
mse_min
```

lambda <dbl>	mse <dbl>	dataset <chr>
0.001689757	393.6184	Train
0.234012138	408.8377	Test

2 rows

The coefficients for the lasso model with the minimum MSE are:

```
coef(fit_lasso,s=lambda_min_mse_test)
```

```
## 114 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)      49.420034769
## song_duration_ms  .
## acousticness     -0.779020790
## danceability      4.949221492
## energy           -2.384926275
## instrumentalness -6.327406338
## key               .
## liveness         -3.493434239
## loudness          0.129864623
## audio_mode        0.175275065
## speechiness      -3.201489016
## tempo            -0.008859174
## time_signature    0.926225495
## audio_valence    -4.798709660
## FEAT.             4.574575385
## Love              .
## Remix             .
## Version           .
## Remastered        4.013914337
## Edit             -0.992052808
## Like              2.830244239
## N...a             .
## Radio             .
## De                .
## F..k              .
## Single           -5.158341856
## One               .
## Go                .
## Mix              -0.471526230
## Back              .
## Acoustic          1.224389979
## Time              0.925985370
## Get              -2.882249760
## Good              .
## Way               .
## Night            -1.466977986
## Girl              .
## Man              -0.281844306
## Life              .
## Want              .
## Know              .
## Mi                .
## Now               4.841558006
## Heart             .
## Baby              .
## Take              .
## Just              .
## Got               .
## Come             -0.258674770
## Let               1.469486621
## Never             1.485004994
```

```
## Feel             .
## Black            .
## Day             -1.367402304
## Song             .
## Live             .
## Little           .
## Home             .
## Better           .
## Me.              .
## Te               .
## New              .
## Original         .
## X2              -1.657502204
## Make            -5.035411843
## World            .
## Fire             0.790340187
## Mind             .
## Young            .
## Right            .
## Blue             .
## Que             -3.504525505
## Away             .
## Bad              .
## Amor             .
## Dance            .
## Light            1.971438798
## Lil              6.009470834
## Ain.t            .
## Say              0.235594833
## Tu               .
## Boy              .
## Eyes             .
## Summer           .
## Sweet            .
## Still            .
## Money            .
## Big             -2.392014847
## Last             0.464671359
## Will             .
## Stay             1.222490023
## Hey              3.771933121
## Give             .
## High             .
## Recorded         .
## See              .
## Girls            .
## Call             .
## Wild             .
## Moon             .
## Alone            0.916167076
## Move             .
## Stop             .
## Spotify          .
## B..ch            .
```

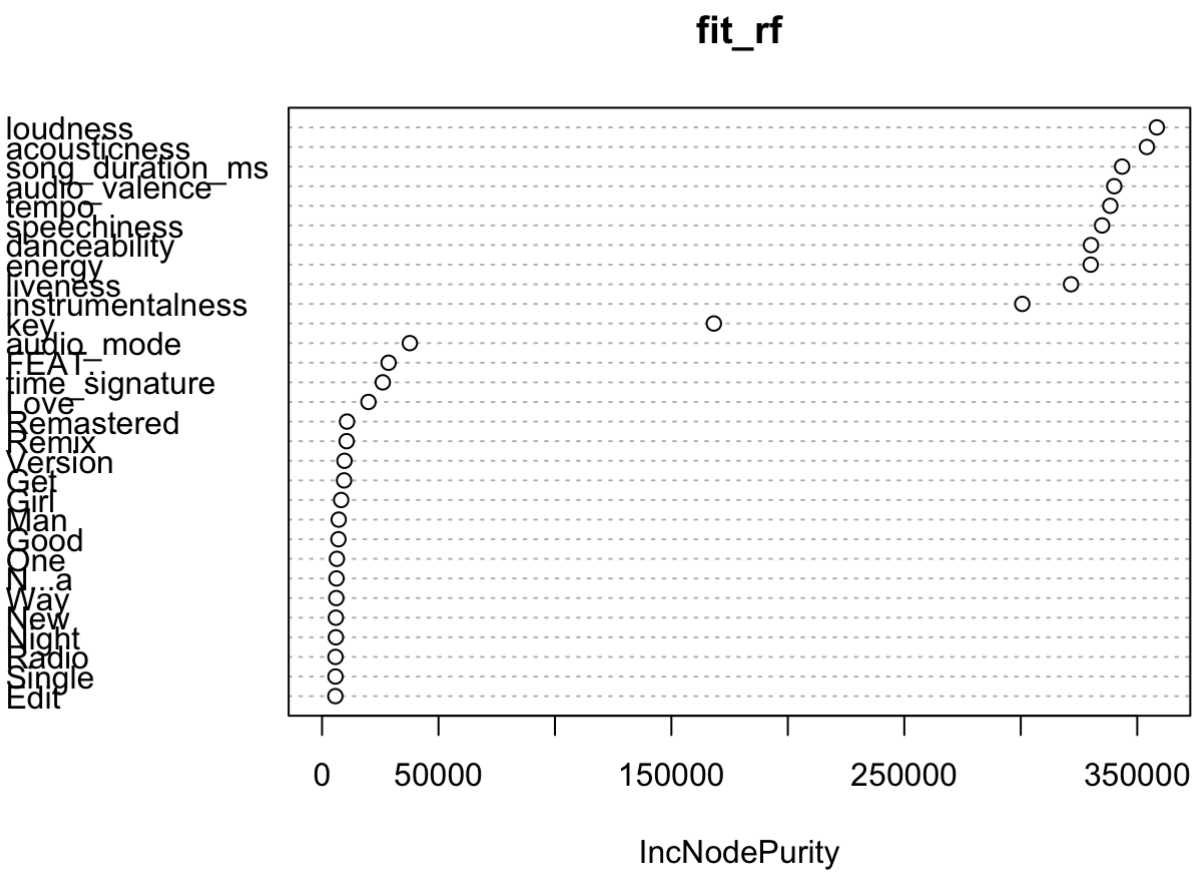
```
## Us                .
## Rock             -0.101021558
## Gonna             .
## Studios           .
## Wanna             .
## Walk              0.980779515
## Ya                .
## Can               .
## Long              .
```

Lasso Regression Summary

Lasso regression sets coefficients to zero for variables that do not sufficiently decrease the train MSE when added to the model. Lasso regression for 100 different values of lambda were executed and the lambda that resulted in the lowest test MSE was chosen. The final model includes 58 variables, reducing the coefficients of 55 variables to 0. The coefficient with the largest, positive coefficients are “Lil” [coefficient = 6.01], danceability [coefficient=4.95] and “Feat.” [coefficient = 4.57]. The variables with the largest negative coefficients were “Make” [coefficient = -5.04] and “Single” [coefficient = -5.16].

Random Forest

```
lf <- "song_popularity ~ song_duration_ms"
for (i in 2:length(xnames)) {
 as.character(xnames[i])
 lf <- paste(lf, "+", xnames[i], sep = " ")
}
f <- as.formula(lf)
x_train <- model.matrix(f,train)[ ,-1]
y_train <- train$song_popularity
x_test <- model.matrix(f,test)[ , -1]
y_test <- test[ ,"song_popularity"]
#################### Random Forest ##########################
fit_rf <- randomForest(f,
                       train,
                       ntree=500,
                       do.trace=F)
varImpPlot(fit_rf)
```




```
yhat_train_rf <- predict(fit_rf,train)
```

Train MSE for Random Forest:

```
mse_rf_train <- mean((yhat_train_rf-y_train)^2)
mse_rf_train
```

```
## [1] 70.08118
```

Test MSE for Random Forest:

```
yhat_test_rf <- predict(fit_rf,test)
mse_rf_test <- mean((yhat_test_rf-y_test)^2)
mse_rf_test
```

```
## [1] 397.5851
```

Random Forest Summary

The final random forest model included 30 variables. Using the variable importance plot we can see that the Spotify variables related to the attributes of the music are the most predictive. On the plot we can see “loundness”, “accousticness”, “song_duration_ms”, “audio_valence”, “tempo”, “speechiness”, “danceability”,

“energy” and “liveness” are the most predictive with IncNodepurity all higher than 250000. But we got the mse of train was small and mse of test was large indicating that we may be overfitting.

Conclusion

Through the majority of the models above similar trends were observed. The first was that the Spotify variables related to the attributes of a song seemed to overall be better predictors for song popularity than the words in a song title. However, repeatedly we did see the words “Feat.” and “Lil” coming through as words that were highly correlated with higher song popularity. These results are overall not surprising because 1.) we would expect the attributes of a song to better predict song popularity than a song’s title and 2.) Feat. and Lil are both words that appear more often when two or more artist have collaborated together on a track which may mean that the tracks attract a larger audience because followers of both artists will be interest.

After analyzing the seven models above, the random forest and forward selection models are the best fit to answer the question “What attributes contribute to song popularity?” but in two very different ways.

If you are an artist or producer that has already recorded an album of songs and would like to predict which song will be the most popular so that you can release it as a single or teaser for the album, the Random Forest model is the most helpful because it had the smallest test MSE at 396. This shows that the Random Forest model is better at predicting song popularity than any other model and would be best for comparing which new songs will be the most popular. However, the Random Forest model is always the most interpretable, so it is crucial to closely examine each variable.

If you are an artist or song-writer, that has yet to record a song but is trying to write the next big hit, the most helpful model would be Forward Selection. The forward selection model had the second smallest test MSE at 404.88 which tells us that it is the second best model out of the set at predicting song popularity. In addition, this model gives better insight into which variables are positively or negatively related to a song’s popularity through the coefficients for each variable. Because forward selection starts by adding in variables that reduce MSE the most, we also know which variables are the strongest predictors. Using this model we know that danceability, “Feat.”, “Remastered are correlated with higher popularity and instumentalness and audio valence are associated with lower popularity. Song-writers or producers could use this information to write or create songs that are better for dancing and feature a guest artist and avoid songs that are mostly instrumental or are overly happy.