Stage 3 report

Introduction

The best example to illustrate the commercialization of the arts is today's music industry. Being a multi-billion dollar industry, artists and labels have to work on finding a balance between making music that appeals to their fans and fulfills their artistic expression, while also makes money. In the digital landscape, the number of plays or streams a song gets is indicative of it's financial success. In other words, the more popular a song becomes, the more money it brings in. If this really is the case, then being able to idenfity how popular a song can become before it is even released would be of great interest to labels and record companies. So, this naturally raises the question that we will be focusing on in this report:

Given data on attributes of a song, can we predict how popular it will become?

Dataset

To answer this question, our group will be working with data on different audio-based features of a song to predict it's popularity when released on a music streaming platform. The dataset we will be using is from the Tidy Tuesday dataset collection [https://github.com/rfordatascience/tidytuesday/blob/master/data/2021/2021-09-14/readme.md]. We have cleaned and combined the `audio_features.csv` and `billboard.csv` datasets provided to obtain the dataset we will use for the rest of our analysis. The raw data files have 32 variables, of which we will be using 21 relevant variables, removing the variables that don't affect our dataset, such as URLs and id's for the songs. We also narrowed the billboard data by selecting only the rows which a song peaks on the chart, and ignoring other weeks, then removing the week specific variables. The full transformation can be found in the `Data_Transformation_script_Billboard.R` script. The remaining data represents different features of a song such as its key, tempo, loudness, danceability, etc. as well as metrics reflecting its popularity on billboard and spotify charts respectively.

Objective

Our goal with this project/report is to create a regression model to predict how "popular" a song may be based on it's audio-based features. This means we will be looking at each song independent of cultural trends, the associated artist's popularity, and other external influences. We will be looking into ideas such as what features popular songs have in common, if any specific feature is highly linked to a song performing well or being well recieved, and whether we can identify any subgroups with their own unique trends.

Motivation

We as a group agreed to working with this dataset and on this question because we are all avid music lovers. With varied interest and tastes amongst all of us, we thought it would be interesting to see how an emperical analysis of what is or isn't "popular music" would compare to our own personal preferences, and how much each of us may agree or disagree with the findings.

Exploratory Data Analysis

Exploration of Peak Position on Billboard as Response Variable

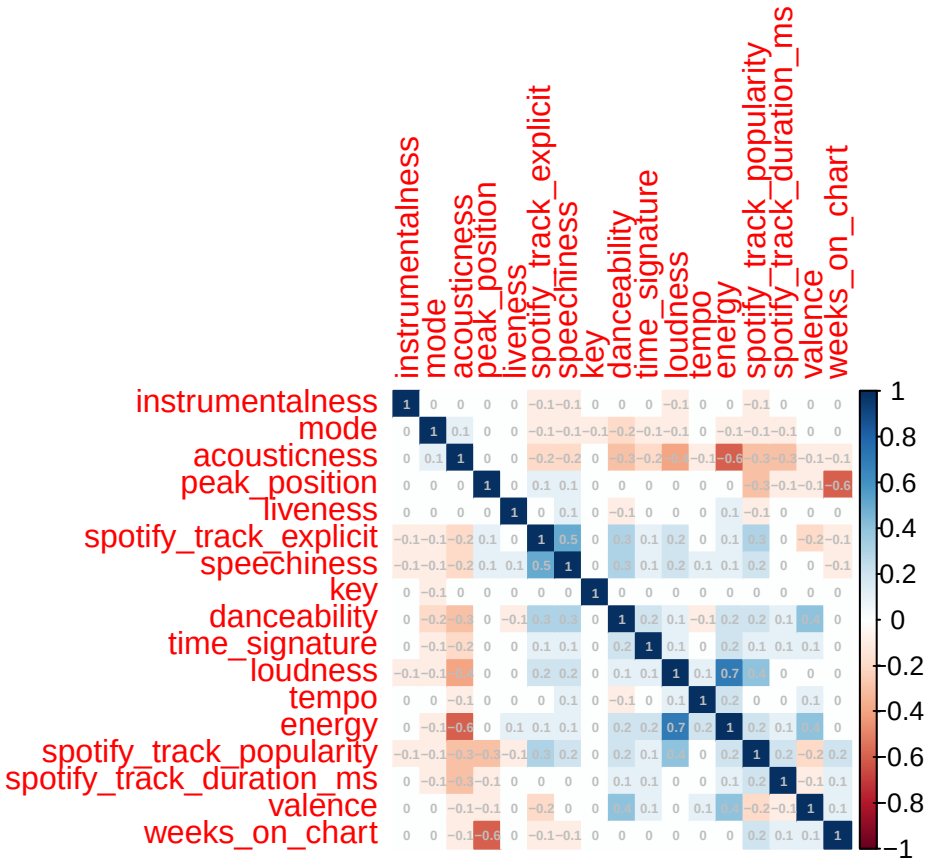
Originally, we had planned to use the peak position on the billboard as a response variable, and model that using features (covariates) of the actual song as described in the data. However, this response variable is not an optimal one to use, and through plots, logical reasoning, and exploratory data analysis we will show why and how we arose to this conclusion and led to our final decision of response variable.

We first cleaned our dataset by removing obviously non-useful variables like names. (However, while we recognize that different genres may have differing levels of popularity, for simplicity we will leave out this variable as the "genre" of certain songs may be ambiguous and there are too many categories to keep track of.)

```
bd_clean <- select(songs, -X, -performer, -song, -spotify_genre,
    -spotify_track_album, -date)
```

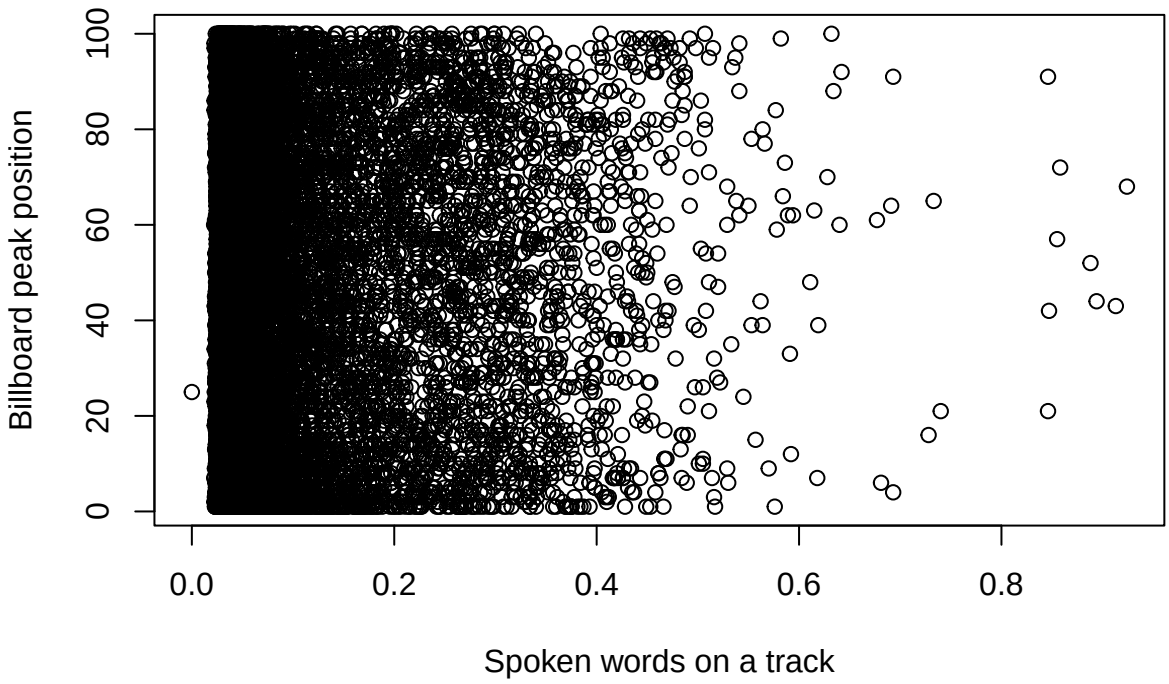
Next we make a correlation plot to have a broad overview of what variables may be connected to our response as well as with each other:

```
options(repr.plot.width = 15, repr.plot.height = 15)
cor_matrix <- round(cor(bd_clean), 1)
corrplot(cor_matrix, method = "color", addCoef.col = "grey",
    order = "AOE", number.cex = 0.4)
```



This is where we became very skeptical about our selected response variable: peak_position. The chart shows that a song's peak position on the billboards have little to no correlation to audio features. Even with actual audio features that our response variable had little correlation with, take "speechiness" for example, has a plot that looks like:

```
plot(x = bd_clean$speechiness, y = bd_clean$peak_position, xlab = "Spoken words on a track",
    ylab = "Billboard peak position")
```



Even a quick glance at these plots show that these audio features have little to no inherent effect on where the song ends up on the chart. Sure we had some correlated features like the weeks on the billboard and spotify song popularity, but those variables and their relation to peak_position is quite obvious and doesnt quite cover our goal of wanting to predict a songs position through mostly its AUDIO features.

We've seen here that because of the low correlation with the other features `peak_performance` may perform quite poorly as the response variable to our question. Hence we will continue to explore for better response variables to try and better predict a song's popularity.

Exploration of Spotify Track Populatiry on Billboard as Response Variable

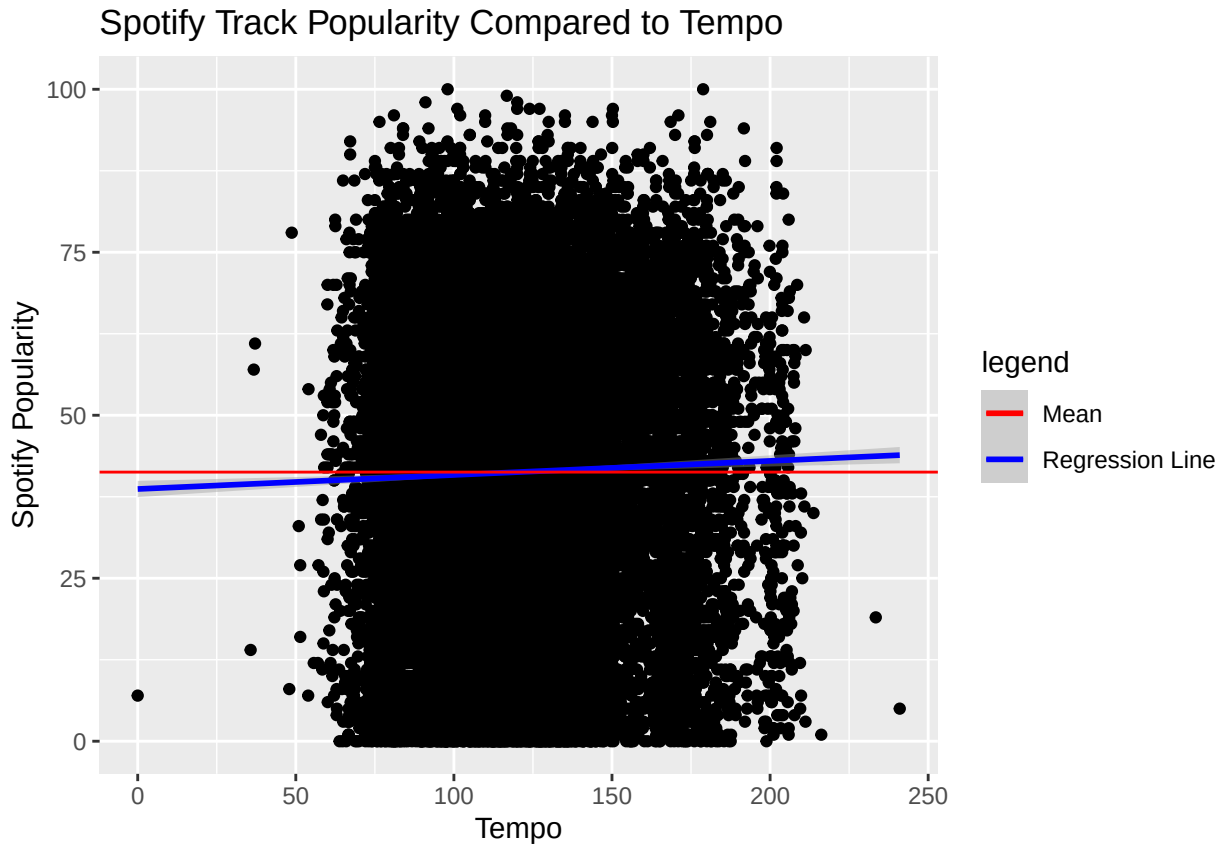
As we have seen above, `peak_performance` isn't the best response variable for popularity in our data set since it's not correlated with many of the audio features. In this section we will look at another measure of popularity of a song which our data set contains, this is `spotify_track_popularity`.

As we can see in the correlation plot above in the above section, `spotify_track_popularity` has a correlation with almost all parameters in our data set. This means we are much more likely to be able to actually predict it's popularity based on the values of the other parameters, which will preform better than `peak_performance` did.

Here we see the two least correlated variables with respect to `spotify_track_popularity`:

```
mean_popularity <- mean(songs$spotify_track_popularity)
ggplot(data = songs, aes(tempo, spotify_track_popularity)) +
    geom_point() + geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Tempo", y = "Spotify Popularity", title = "Spotify Track Popularity Compared to Tempo")
```

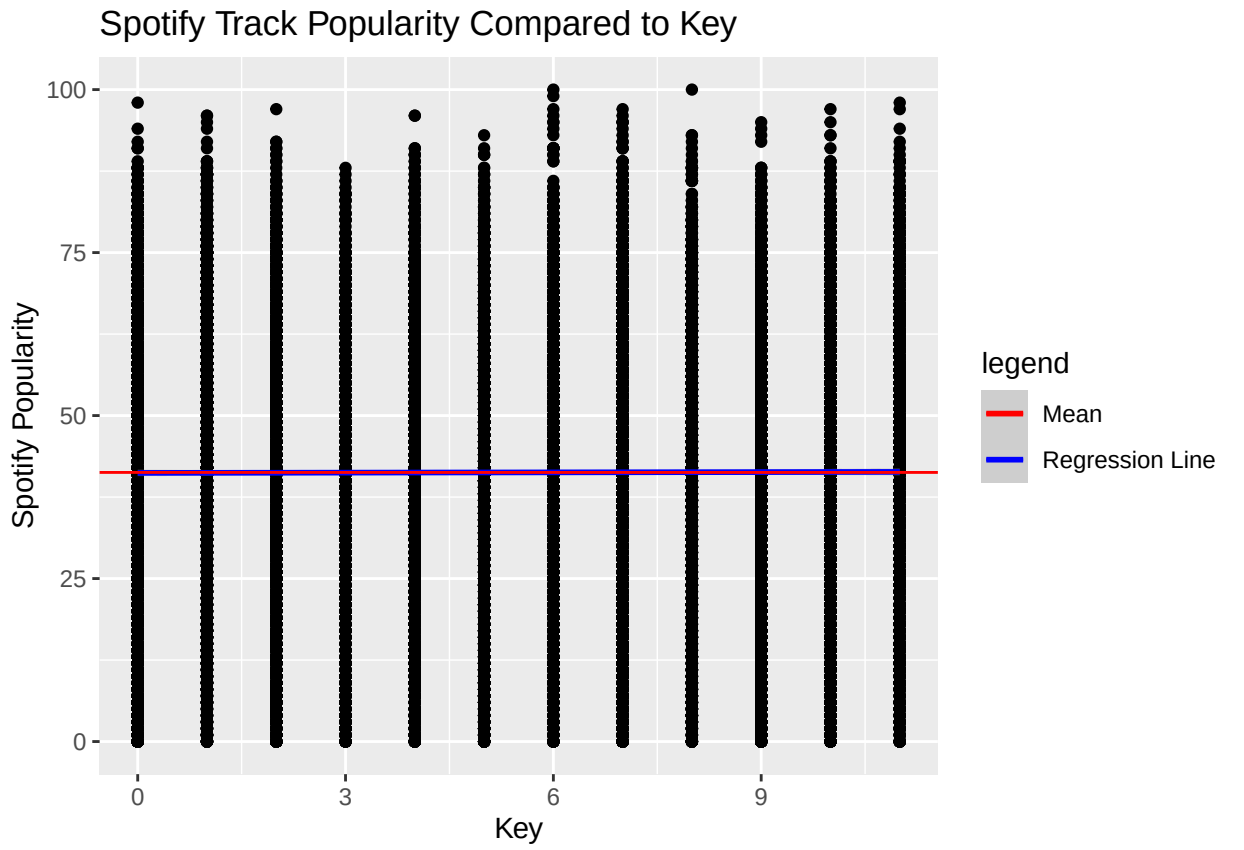
```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(data = songs, aes(key, spotify_track_popularity)) + geom_point() +
    geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
```

```
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Key", y = "Spotify Popularity", title = "Spotify Track Popularity Compared to Key")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



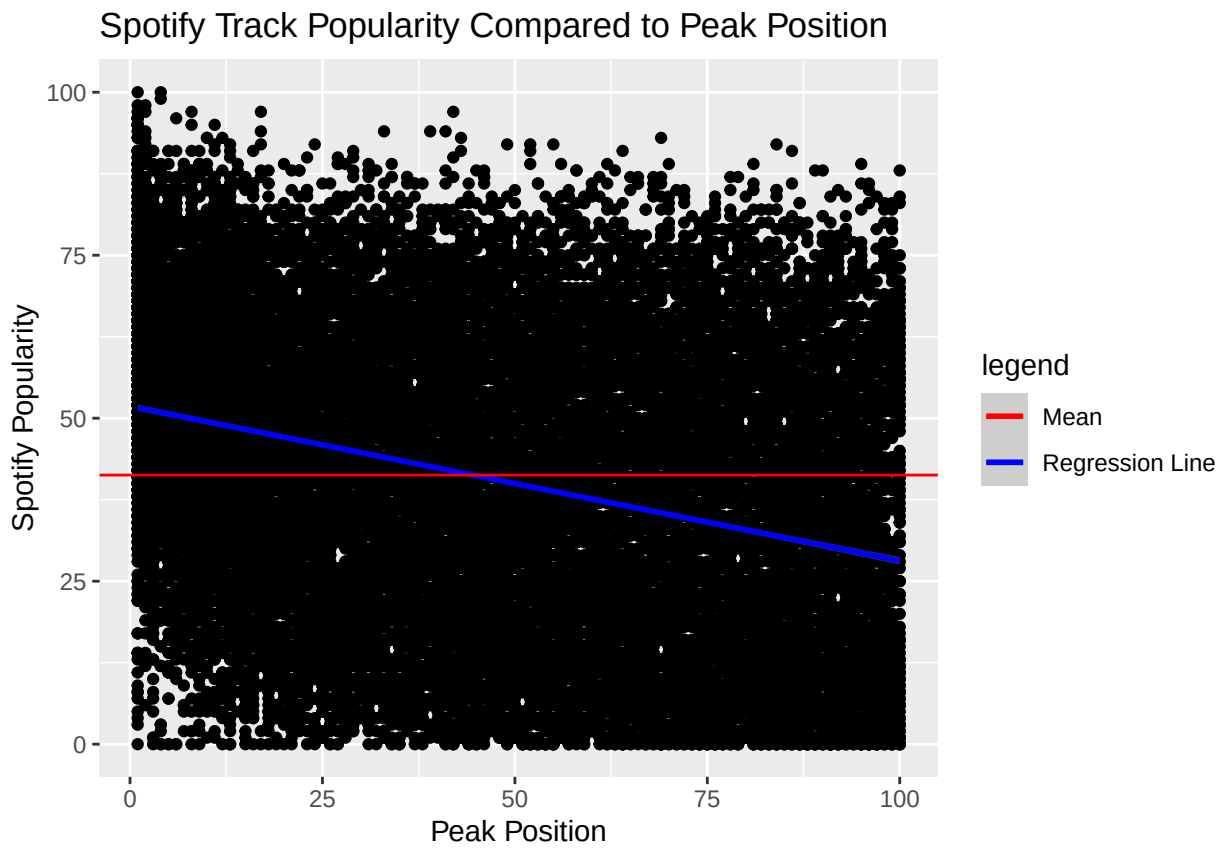
As we can see `key` is quite a unique variable, since it's discrete on a scale of 1-11. But on the plot here we see that the values are roughly equally distributed across the different keys, and all the keys have nearly full coverage of popularity. We see that the mean of the track popularity almost exactly matches the univariate regression line of the two variables, which means that there's almost no information gained about the popularity through looking at the `key`. We expect that using LASSO, this would be thrown out of the model quite quickly, or minimized to be near zero using Ridge, so I'd expect this to have very little weight in any final model we pick. The `key` is an integer mapping to the pitch of the song (ex. A, B, C, A# ...) which explains the lack of information we get from that variable, as popular songs often come in many different pitches.

With respect to popularity `tempo` is also a weird parameter. It's clear from looking at this plot that the vast majority of tempos fall between roughly 60-200 bpm. When looking at the data here it's easy to tell there's not a lot of information to gain here. We see that reinforced from the regression line that nearly matches the mean here too. Logically there's no magic tempo that makes songs into hits and almost all songs fall between this set range, so there's no way of determining which song of a specific tempo is a hit versus one that is unpopular, without using some other parameter as well, so intuitively tempo will not play a large role in predicting the popularity. If we use this for anything, it may make more sense to include tempo as a categorical variable to represent whether a song falls into this range of tempos or not.

The following plots are some of the higest correlation with `spotify_track_popularity`:

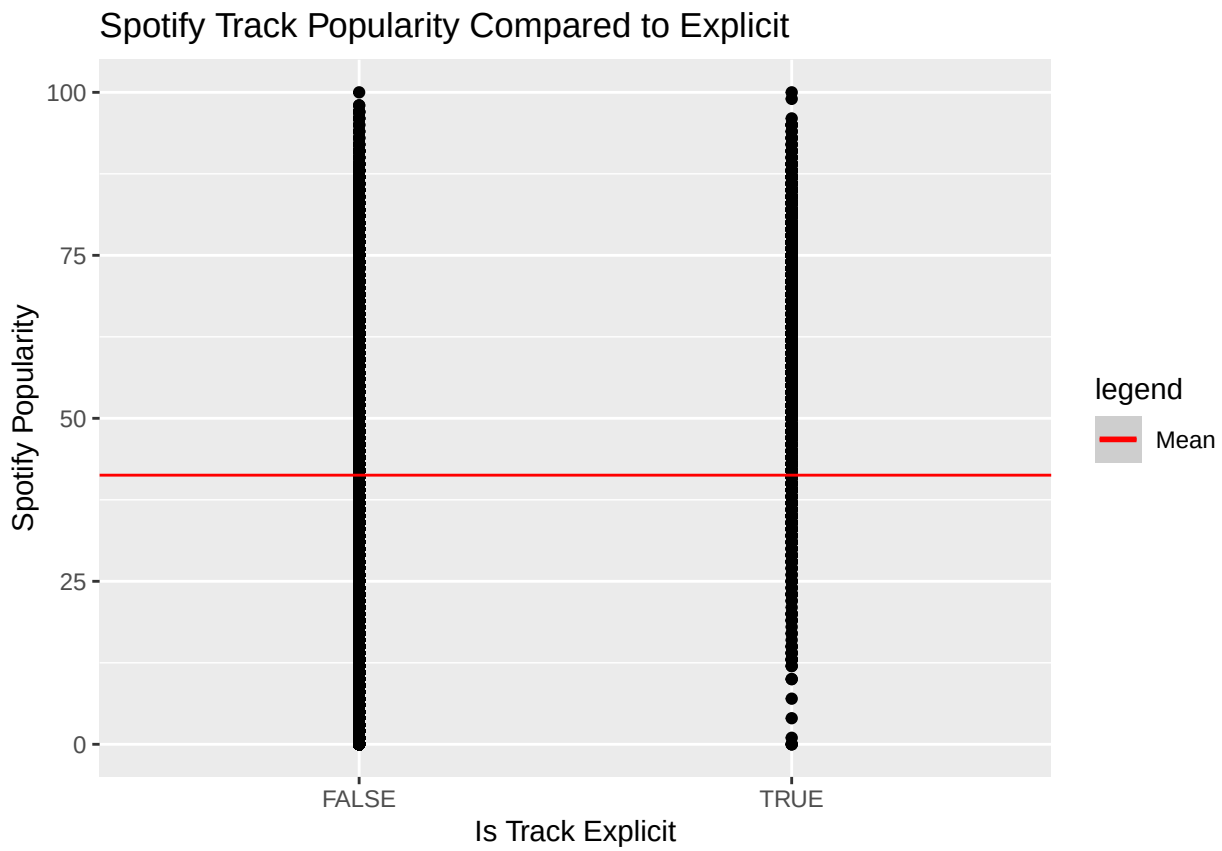
```
ggplot(data = songs, aes(peak_position, spotify_track_popularity)) +
    geom_point() + geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Peak Position", y = "Spotify Popularity", title = "Spotify Track Popularity Compared to P
```

```
## `geom_smooth()` using formula 'y ~ x'
```



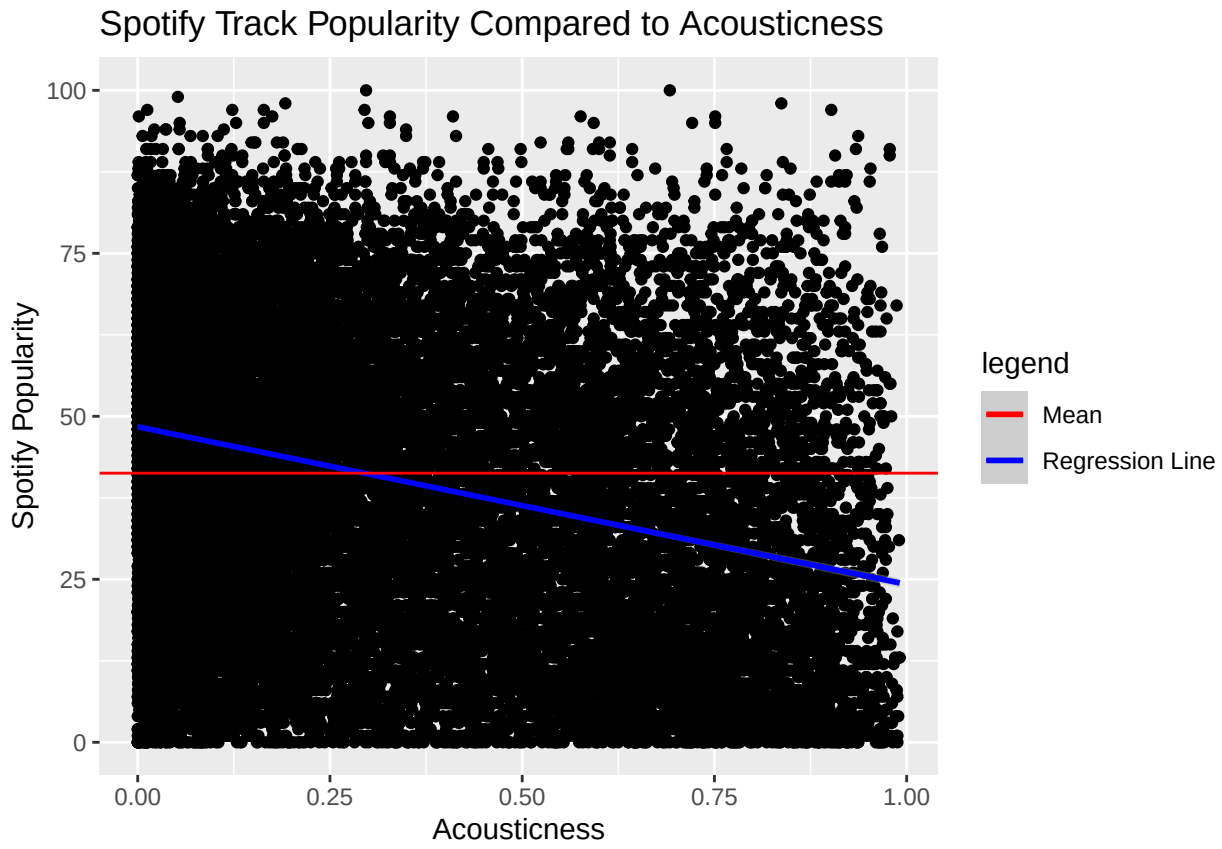

```
ggplot(data = songs, aes(spotify_track_explicit, spotify_track_popularity)) +
    geom_point() + geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Is Track Explicit", y = "Spotify Popularity", title = "Spotify Track Popularity Compared
```

```
## `geom_smooth()` using formula 'y ~ x'
```



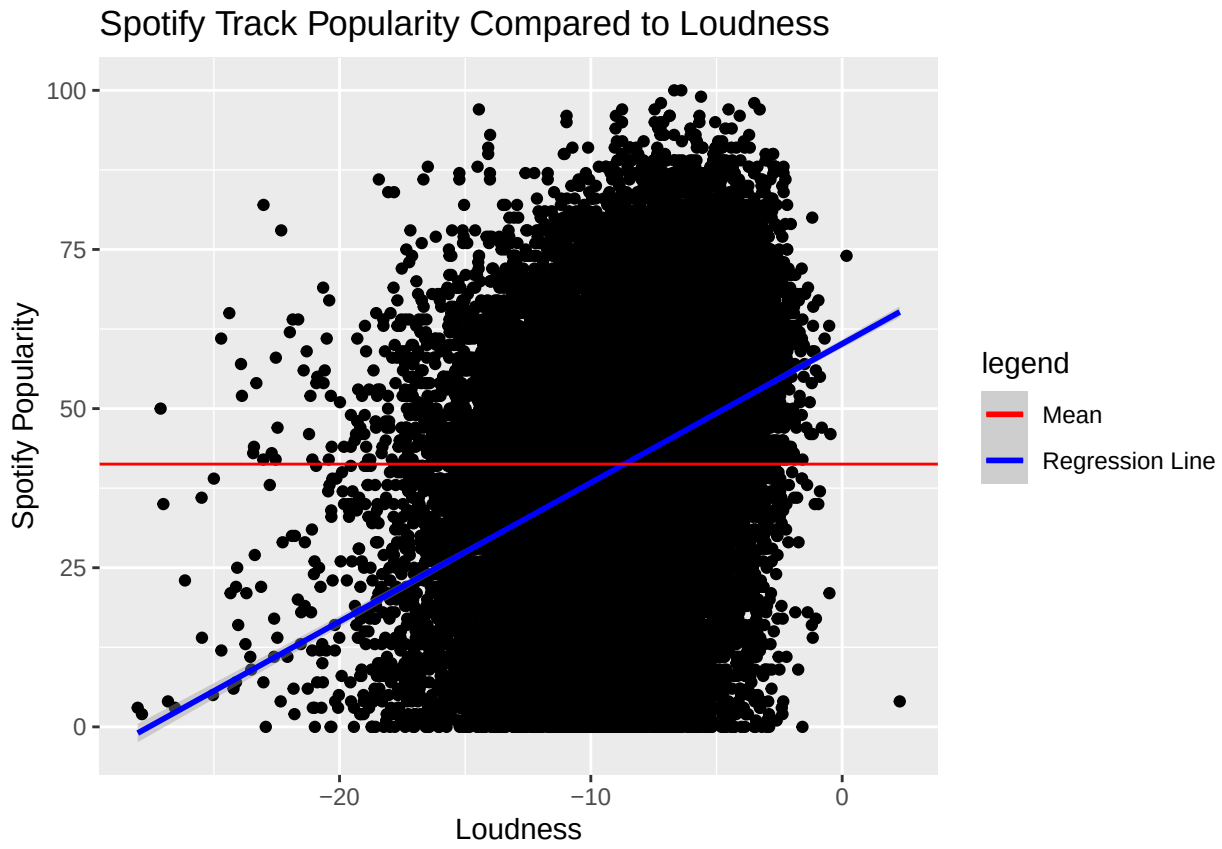

```
ggplot(data = songs, aes(acousticness, spotify_track_popularity)) +
    geom_point() + geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Acousticness", y = "Spotify Popularity", title = "Spotify Track Popularity Compared to Ac
```

```
## `geom_smooth()` using formula 'y ~ x'
```



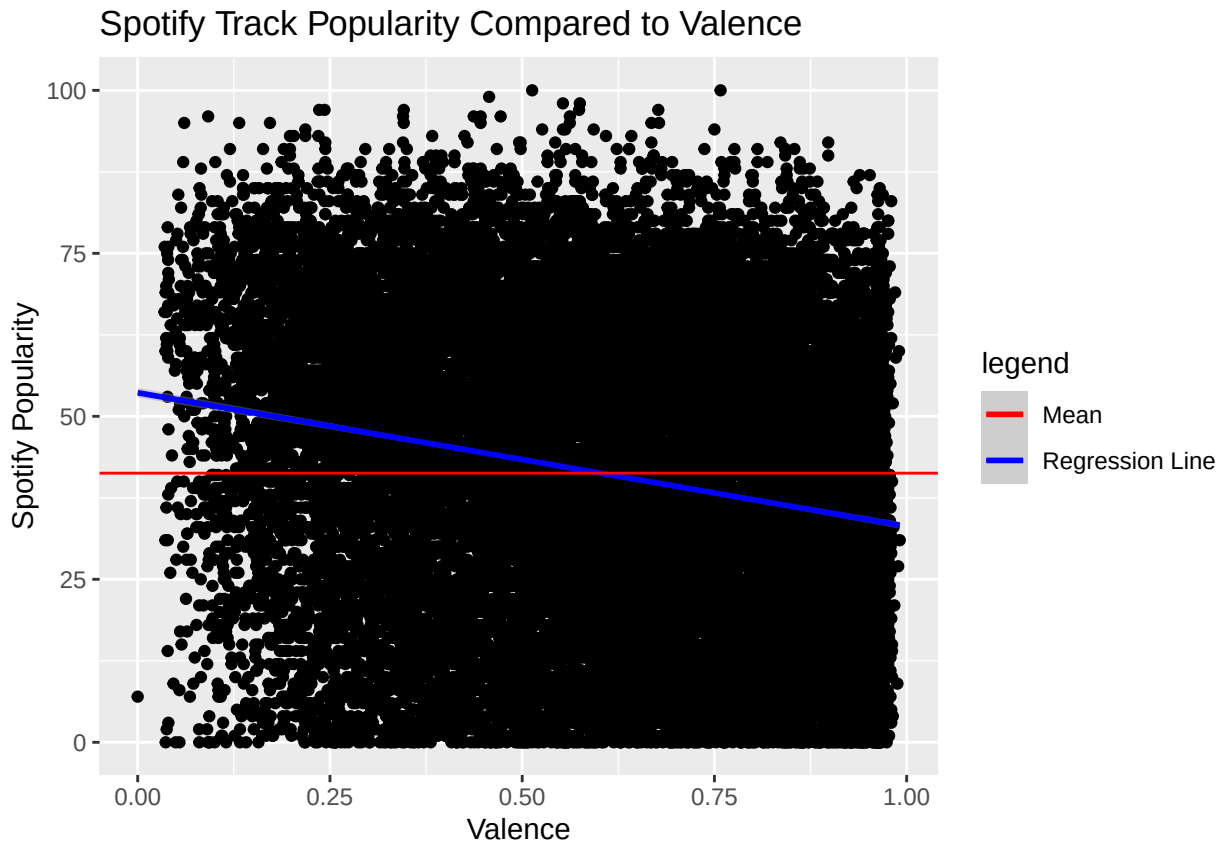

```
ggplot(data = songs, aes(loudness, spotify_track_popularity)) +
    geom_point() + geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Loudness", y = "Spotify Popularity", title = "Spotify Track Popularity Compared to Loudnes
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(data = songs, aes(valence, spotify_track_popularity)) +
    geom_point() + geom_smooth(method = "lm", aes(color = "Regression Line")) +
    geom_hline(aes(yintercept = mean_popularity, color = "Mean")) +
    scale_color_manual(name = "legend", values = c("red", "blue")) +
    labs(x = "Valence", y = "Spotify Popularity", title = "Spotify Track Popularity Compared to Valence
```

```
## `geom_smooth()` using formula 'y ~ x'
```

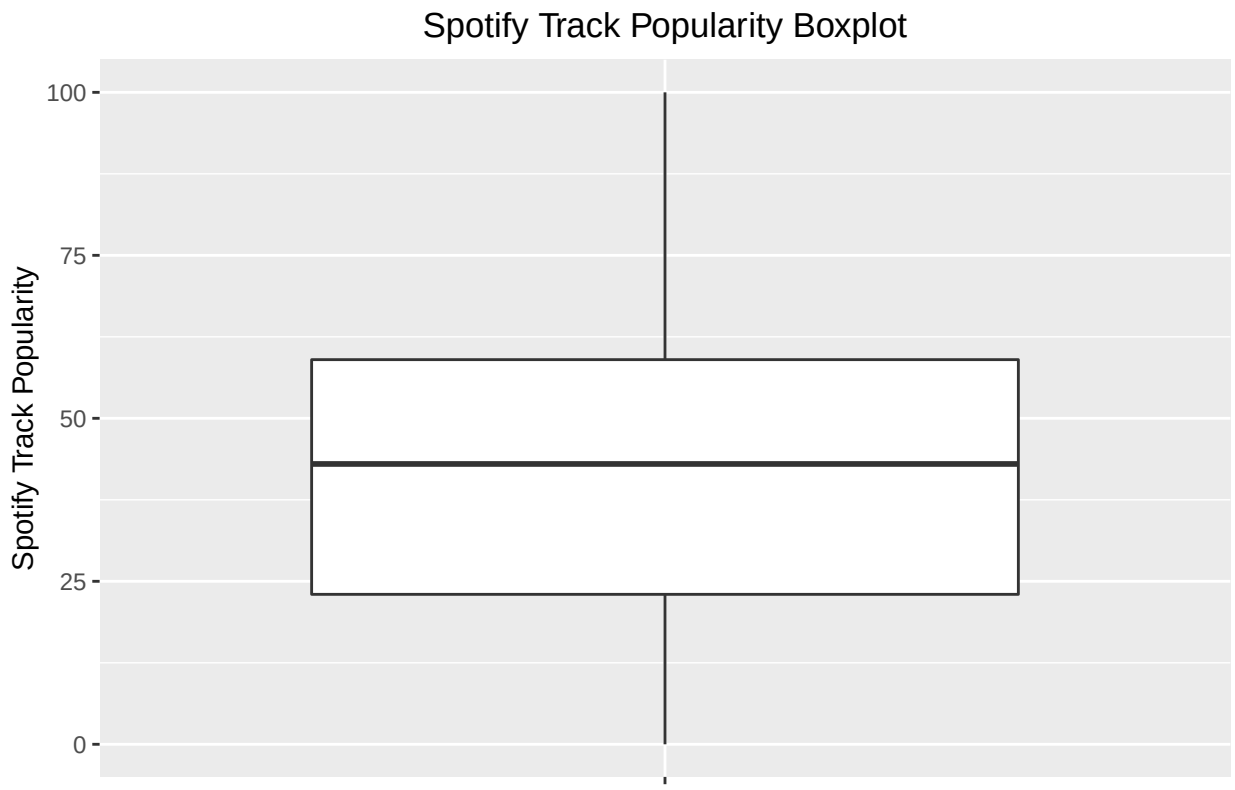


From the plots above, we see that while these are the "strongest" correlated parameters to our response variable, they are all quite weakly correlated with them. While `peak_position` which is the song's peak on the billboard charts (with 1 being the best) does show a negative correlation with the track popularity, it is quite interestingly not a fantastic predictor, we see a number of number one songs on the billboard charts with low popularity on Spotify. The binary variables `spotify_track_explicit` also shows that there's popular songs that are both explicit and not explicit, so it's strong correlation must come from the fact that there are fewer unpopular (less that 25 popularity) explicit songs. `acousticness` and `valence` are also both audio features that are negatively correlated to the popularity, showing that people must not like highly acoustic or valeant (highly positive) music. While `loudness` is quite a dramatic regression line, in some ways it is similar to `tempo` where most values are centered in a certain range, but this likely does have more effect on the popularity.

As we've seen with all these variables we will likely need many in tandem in order to create a model to predict the popularity of the songs, but we do have correlation in some form across many variables, so `spotify_track_popularity` should be a better response variable than `peak_performance`, so we will use this as our response variable.

Further Exploration Of Data

```
popularity_boxplot = songs %>%
    ggplot(aes(x = "", y = spotify_track_popularity)) + geom_boxplot() +
    ggtitle("Spotify Track Popularity Boxplot") + labs(x = "",
    y = "Spotify Track Popularity") + theme(plot.title = element_text(hjust = 0.5))
popularity_boxplot
```



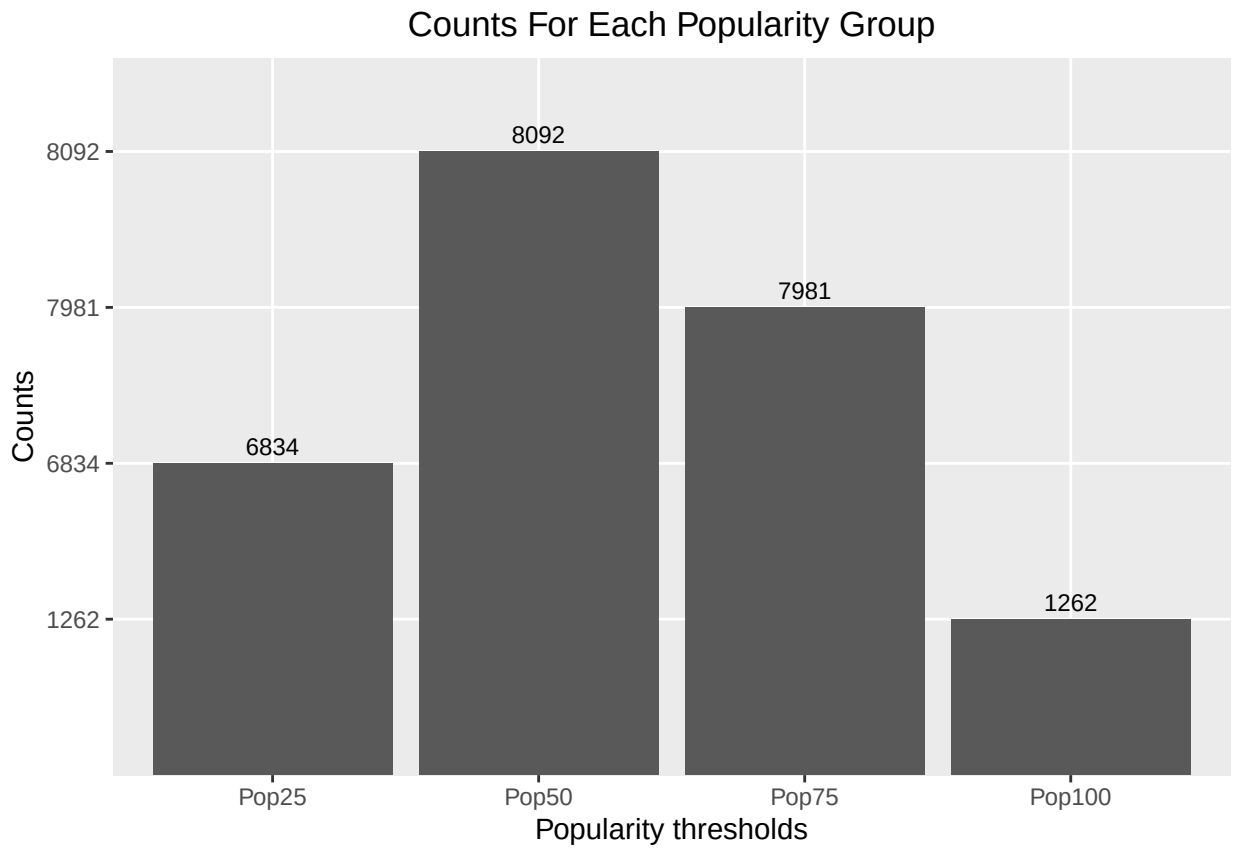

```
summary(songs$spotify_track_popularity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   23.00   43.00   41.28   59.00  100.00
```

While it was expected that spotify popoularities cover the entire range from 0 to 100, the boxplot, alongside the five number summary helps understand that the median popularity is 43. This suggests that a lot of songs do not rank highly in popularity. Hence, the higher popularity ranks are coveted. We will further explore how many songs are rated in four categories, as described below.

```
pop25 = songs %>%
    filter(spotify_track_popularity <= 25)
pop50 = songs %>%
    filter(spotify_track_popularity > 25 & spotify_track_popularity <=
        50)
pop75 = songs %>%
    filter(spotify_track_popularity > 50 & spotify_track_popularity <=
        75)
pop100 = songs %>%
    filter(spotify_track_popularity > 75)


pop25_df = cbind("Pop25", nrow(pop25))
pop50_df = cbind("Pop50", nrow(pop50))
pop75_df = cbind("Pop75", nrow(pop75))
```

```
pop100_df = cbind("Pop100", nrow(pop100))

pop_all_df = as.data.frame(rbind(pop25_df, pop50_df, pop75_df,
    pop100_df))
colnames(pop_all_df) = c("Pops", "n")
ordered_pops = c("Pop25", "Pop50", "Pop75", "Pop100")

pop_n_plot = pop_all_df %>%
    ggplot(aes(x = Pops, y = n)) + geom_col() + scale_x_discrete(limits = ordered_pops) +
    labs(x = "Popularity thresholds", y = "Counts") + ggtitle("Counts For Each Popularity Group") +
    theme(plot.title = element_text(hjust = 0.5)) + geom_text(aes(label = n),
    vjust = -0.5, size = 3, position = position_dodge(0.9))

pop_n_plot
```



To better understand the spread of the data, four groups have been constructed, `Pop25`, which contains data for all values of `spotify_track_popularity` $<$ 25; `Pop50` for 25 $\leq$ `spotify_track_popularity` $<$ 50; `Pop75` for 50 $\leq$ `spotify_track_popularity` $<$ 75; and finally, `Pop100` for `spotify_track_popularity` $>$ 75. It is evident that a large portion of the songs get ranked between 25 $\leq$ `spotify_track_popularity` $<$ 75, while only 1262 out of the 24169 entries are ranked equal to or above 75.

Results and Analysis

Training models

Steve Model

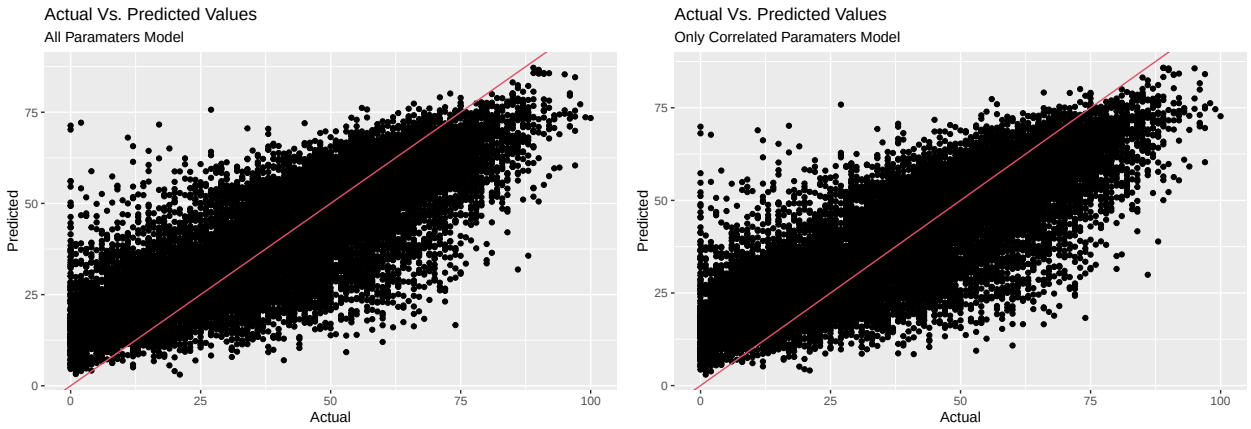
Aishwarya Model

Anirudh Model

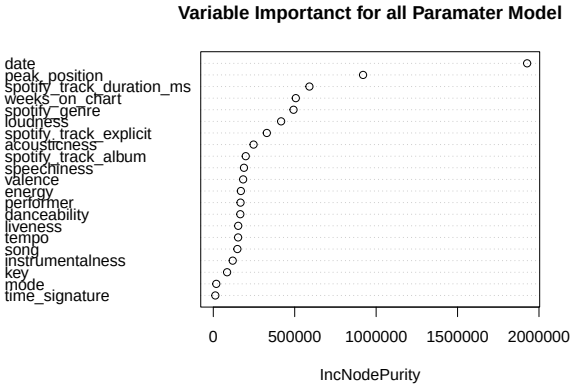
Zack Model

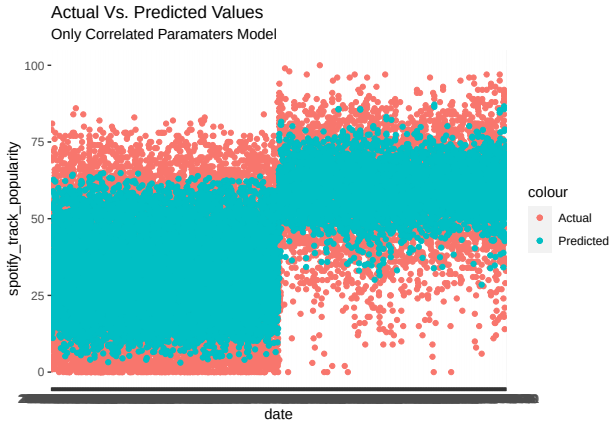
As we've seen in the EDA across the scatterplots, it is not immediately clear that a linear model fits our dataset very well. Since random forests are one of the best "out of the box" models, we will attempt to use random forests to predict the popularity of our songs. We will compare two different models, one with all fields and one with those with 0 correlation to our response removed.

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```



As we can see in the above graphics, where we plotted the expected values versus the predicted values against one another, we get a roughly linear line along the y = x line shown in red. The models do not differ greatly between one another. We do however see that at the very high/low values our random forest predicts poorer than it does for values in the center. The mean squared error for the model with all parameters is 182.8507916 and for the model removing the uncorrelated parameters is 183.6636607.





We can see from the above plots that date is by far the biggest contributor to the prediction. This is followed by the peak position on the billboard chard, genre, duration, and weeks on the billboard charts. These all make a lot of sense as the large contributors, the largest surprise to me was date beating billboard peak position, as it would make sense that billboard performance would indicate the Spotify popularity. In the next plot, date, which is clearly not a linear parameter with respect to the popularity is decently well predicted by our random forest. Though as expected from the previous plots, the data is more centered rather than covering the high/low values very well.

Comparing the models

Putting them all together

Results and Discussion

Results

Discussion

Conclusion