# Analysis of the Top 40 Streamed Songs of 2020

Gianna Barletta, Sophie Lefebvre, Grace Michael, Preston Reep & Dachuan Zhang

Northeastern University, Boston, MA, USA

## Introduction

The following report details the analysis of the Top 40 Streamed Songs of 2020 and how the components of these songs contributed to their positioning on the chart. The analysis will show if these songs carry numerous similarities, and if the #1 streamed song is the strongest in all areas of analysis.

We hypothesize that a song in the Top 5 Streamed Songs will score the highest, but we expect that it will not be the #1 most streamed song in our usable data, “Dance Monkey”. This is because we expect that the most popular songs will be successful all around in categories, but there could be external factors, not accounted for in the data, that influence a song’s popularity, such as celebrity endorsements or going viral on social media.

For organizational purposes, the analysis has been split into two categories: Spotify Analysis and Lyrical Analysis. Spotify Analysis will show the specific analysis of the songs’ attributes themselves, while Lyrical Analysis will focus on just the lyrics. Both aspects are important to the analysis of songs as the mood the music conveys may not always match the ideas the lyrics are trying to convey.

This project is significant because it quantifies what is deemed a popular song by society and it analyzes how popularity has changed over time.

## Data Sources and Methods

We started with a list of the Top 40 Streamed Songs of 2020 [4]. To get more details to use in our analysis, we then combined it with a Kaggle dataset that had Spotify audio features for over 160,000 tracks from 1921 to 2020 [2]. We merged this with our original list of 40 songs to get the stats for the Top 40 Streamed Songs. There were 6 songs not included in the Kaggle data set: “Blinding Lights”, “Own It”, “Lonely”, “Rover”, “Rain”, and “Flowers”. The lack of one or more null categories resulted in the song’s removal from our dataframe. These songs were exempt from our Spotify Analysis, but not the Lyrical Analysis. It is important to note that on the original Top

40 Streamed Songs chart, “Blinding Lights" was the overall most streamed song, so any references to the Original Top Streamed Song of 2020 will, infact, be to the 2nd Top Streamed Song, “Dance Monkey”.

Upon reflection, we concluded that songs have two major components: the music and the lyrics. This is why we separated the analysis into Spotify Analysis and Lyrical Analysis. All lyrics for all songs included in the Top 40 were found online and turned into txt files [5]. A list of explicit words was also found online and turned into a text file [3]. This gave us a bank of 43 total files, including the Spotify Data and Top 40 Data.

We created two functions, read_file and clean_df, that read in each CSV dataset as a Pandas dataframe, then turn the contents of the columns lowercase and remove punctuation and extra spaces. This standardizes the song and artist names across dataframes, which allows us to merge the two. We originally had it as one function, but cleaning the artist name no longer allows us to differentiate between whether it’s a single artist or multiple artists, which we wanted to use in our analysis. Instead we divided the function into two parts, allowing us to read in the CSV, duplicate the artist name column, then clean the original artist name column.

We merged the two datasets into a single dataframe by joining on a ‘key’ variable we created that combines the artist name and song title, since some songs have the same title. Some songs on Spotify have multiple versions, such as when an artist releases the song as a single and then again on their album, so we sorted by release date and dropped all duplicates except the first version.

The duration, danceability, valence, tempo, and loudness functions visualize the distribution of these attributes for the Top 40 Most Streamed Songs using histograms.

The statistics function takes in a dataframe and a column and outputs its respective z score. It filters out rows with NaN values, calculates the standard deviation and mean of the column, and then plug them into the z score formula (value-mean/standard deviation). This is iterated over each row, and these values are appended into a list that is outputted at the end.

The ranking most generic function takes in a dataframe and a list of values it is comparing and outputs its respective value. This value is determined by the sum of absolute z score values, and the lower the sum, the higher it is up the rankings. The function runs the statistics function over each column it is comparing, and then adds all of their absolute values together. The sum for each row is appended to a list which is outputted at the end.

To get the final output, this list is appended as a column onto a copy of the dataset, and that dataset is then sorted in ascending order.

For the lyrical analysis, to read in the text files, we created two functions: read_song and loop_thru_files. We created a folder with all our lyric text files and used the two functions to read in all files within the folder at once into one dictionary, with the song name as the key and the lyrics as the values. Our function clean_text does a basic cleaning that removes punctuation, new line characters, and turns all the words lowercase. We then ran a dictionary comprehension using

the clean_text function to clean all the lyrics at once.

Next, we had two more cleaning functions depending on which analysis we planned to perform on the lyrics. The first, remove_words, removes a given list of words from the lyrics if the words are an exact match. We use it to remove stop words, which allows us to only analyze the lyrical words that have more meaning rather than words that are used to make sentences flow.

We originally used this to remove explicit words as well, however, some explicit words still got through because they were not exact matches (i.e. the word “wetass” was included because it did not match “ass” exactly). We created a second function, remove_more_words, that matches substrings, utilizing the regex function findall. We needed this in addition to the previous function because the stop words were short, so this function would have eliminated a large chunk of the lyrics looking for substring matches. Removing explicits from the lyrics was done primarily to allow the WordCloud visualizations to have a level of appropriateness and inclusiveness by removing any vulgar language or slurs [3].

Within Lyrical Analysis, we looked at line length, common words between songs, and subjectivity and polarity. We accomplished this with the following functions:

The avg_line_length function returns the average number of words per line for the inputted song. The lyrics inputted into this function were cleaned using the clean_text function, however no words were removed to make sure the word count was accurate.

The score_songs function scores the polarity and subjectivity for each line in the inputted song and returns it as a dictionary.

The total_specific_words function counts the number of occurrences of a given list of words in the lyrics. Similar to the remove_more_words function, it uses the regex findall function to match substrings because we use the function to count the number of explicit words in each song.

We then created a function, big_dict, to create a dictionary of the above attributes for each song. The dictionary is then converted to a dataframe and then merged with the dataframe from the first part containing audio attributes and our new song ranking. Having all these values in one dataframe allows us to analyze all song features more efficiently.

The regression function takes in two lists of variables and uses the LinearRegression import from sklearn.linear_model and numpy arrays. These lists are converted into numpy arrays which are used to calculate the m and b in the point intercept line formula y=mx + b. The function takes these values and puts them into this form. After the line is constructed, the function also takes the list and calculates their r value using LinearRegression. Both the line equation and r value are outputted.

The cloud_generator function generates a WordCloud visualization for the inputted song with a music note as a mask. Words that occur more in the song are displayed larger in the WordCloud. The lyrics inputted into this function were cleaned using the clean_text function and the

remove_words function to remove stopwords and explicits. Explicits were removed so that the WordClouds would only display family friendly lyrics.

The polarity_vs_subjectivity function creates a KDE plot that displays the polarity and subjectivity scores for each line in a song. Multiple songs can be displayed on the KDE plot, each with their own color. The lyrics inputted into this function were cleaned using the clean_text function and the remove_words function to remove stopwords.

The comparing_most_common function compares two songs for common words, and outputs a tuple with the percentage of common words out of total words for song1 and a list of the common words. The lyrics inputted into this function were cleaned using the clean_text function and the remove_words function to remove stopwords.

The get_song_community function returns a list of lists where each list includes the two songs compared (using comparing_most_common) and the percentage of common words out of all words for song1.

The orgainze_dates function takes in a dataframe and column. It extracts the column as a series, and iterates over each row, splitting it on '-'. These items are then converted into integers using map() and all put into a list, which are each appended into another list. The output is a list of lists, that each contain a list containing the year, month, and day in that order.

From there, this function is run over the release dates. This list is then appended as a column onto the dataframe, where they can then be sorted into ascending order.

The dates_scaled function takes in a column and a dataframe that is already sorted by its release date. It takes the first entry and stores it as a variable. For each date, the function calculates how far away it is from an established origin date(which for our purposes is the earliest release date by calculating how many years, months, and days the date is from the first date, scaled by how many days are in the given period. 365 for a year, 30 for a month, and 1 for a day. These totals are then appended to a list which is, in end, appended to a dataframe. The function returns a list of these distances, which can be then added as a column in the dataset.

The time_between_release_dates takes in a column of dates generated by the above function and a dataset that is sorted in ascending order. For each entry, it finds out how its difference is from the previous entry. That subtraction is then appended into a list. The output is this list of differences.

The average_per_period takes in the list outputted by the previous function, and upper and lower bounds. These bounds are used to index the list based on a specified period of variables. The average of this period is then outputted.

Features_artist function takes in a list of unfiltered artists. It separates the artists by commas, and looks at their length. For each artist, if the length is larger than one, it will append a 1 into a list while if it is equal to or less than 1, it will append a 0. It outputs the list of 1's and 0's to represent

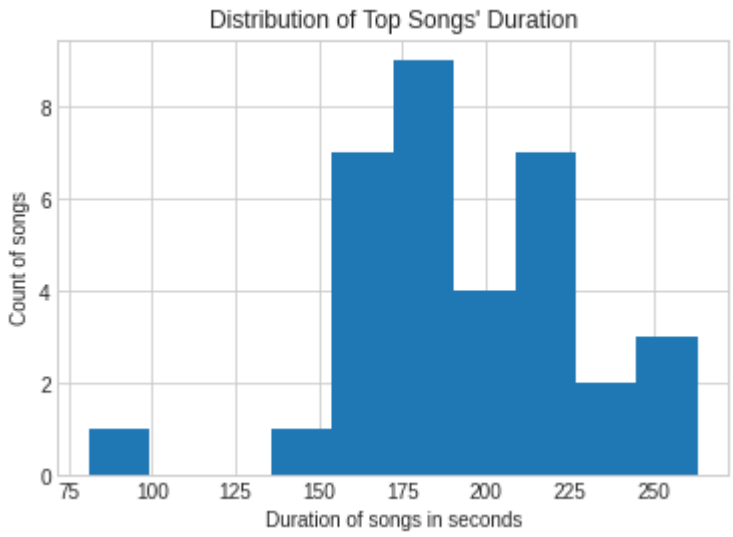
its featured status.

From this, two datasets are made. One dataset only contains songs without features and the other only contains songs with features. These are used on a bar chart, a box plot comparing its rankings for featured and solo artists separately, and represents them as different color dots in the scatterplot.

The comparison_scatter_plot takes two datasets, the solo and the featured, the larger dataset they both come from, the independent and dependant variables that are being compared,  each of the filtered data sets colors, a color for the regression line, the plot dimensions, and location on the subgraph. Using this data, a main plot is produced and different subplots produce scatter plots which compare the independent and dependent variables. A regression line is then produced, and it automatically fills out the necessary labels.

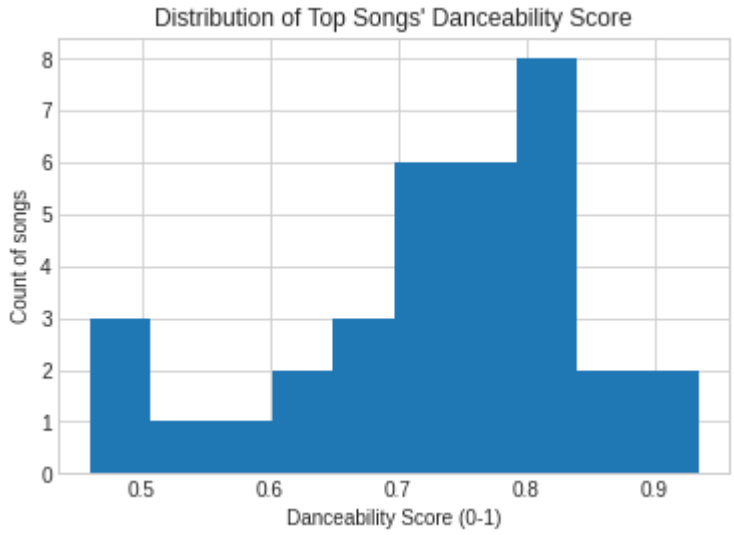
## Analysis and Results

**Figure 1: Distribution of duration of Top 40 Most Streamed Songs.**



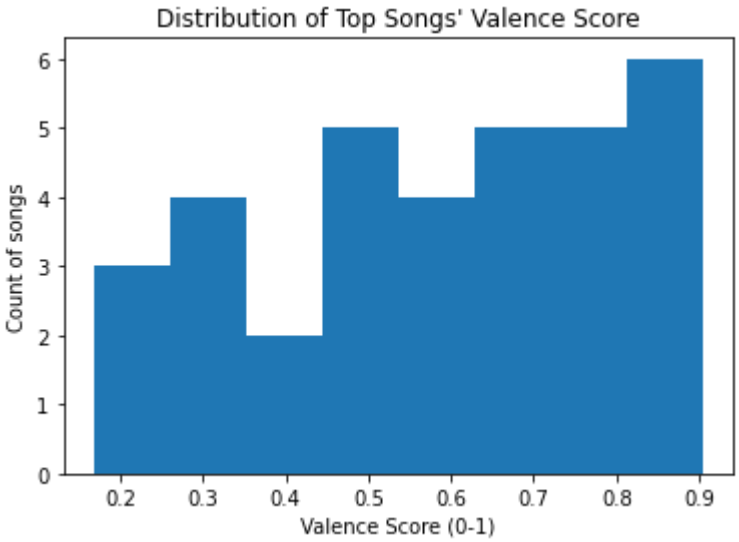

As we can see from Figure 1, it is clear that the majority of the top songs have a duration of around 175 seconds, with the lowest between 75 and 100 seconds and the highest around 250 seconds.

**Figure 2: Distribution of danceability of Top 40 Most Streamed Songs.**



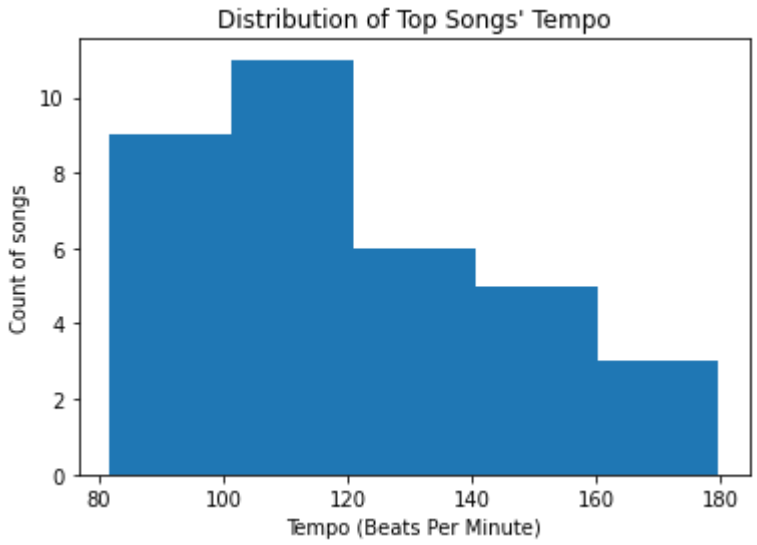

Danceability score measures a song's suitability for dancing. Figure 2 shows that most of the most popular songs have a relatively high danceability score of around 0.8 out of 1. None of the top 40 songs have a danceability score of lower than 0.5.

**Figure 3: Distribution of valence of Top 40 Most Streamed Songs.**



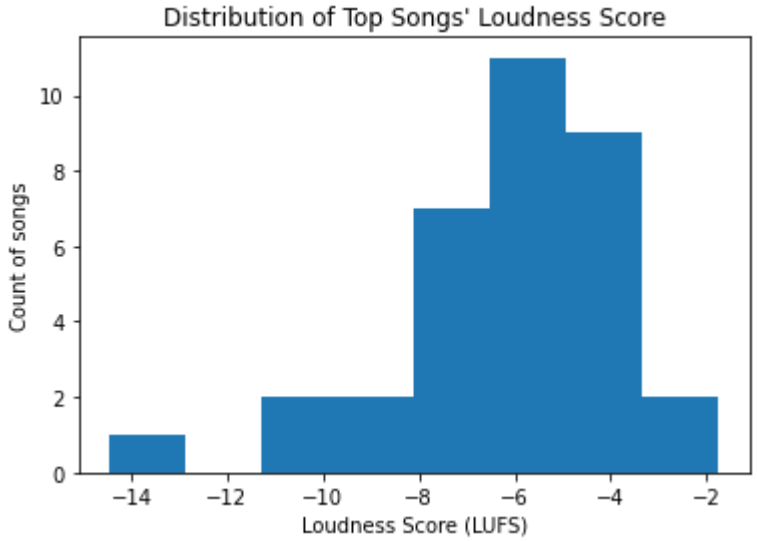

According to Spotify, Valence describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (happy, cheerful, euphoric), while tracks with low valence sound more negative (sad, depressed, angry). Figure 3 shows that the top 40 songs have a relatively uniform distribution of valence score, ranging from 0.2 to 0.9 out of 1.

**Figure 4: Distribution of tempo of Top 40 Most Streamed Songs.**




There is a clear right-skewed distribution of top songs' tempo, measured in beats per minute. The highest number of top songs have a tempo of 100-120 beats per minute, with a range of 80 to 180 beats, as shown in the histogram above.

**Figure 5: Distribution of loudness of Top 40 Most Streamed Songs.**



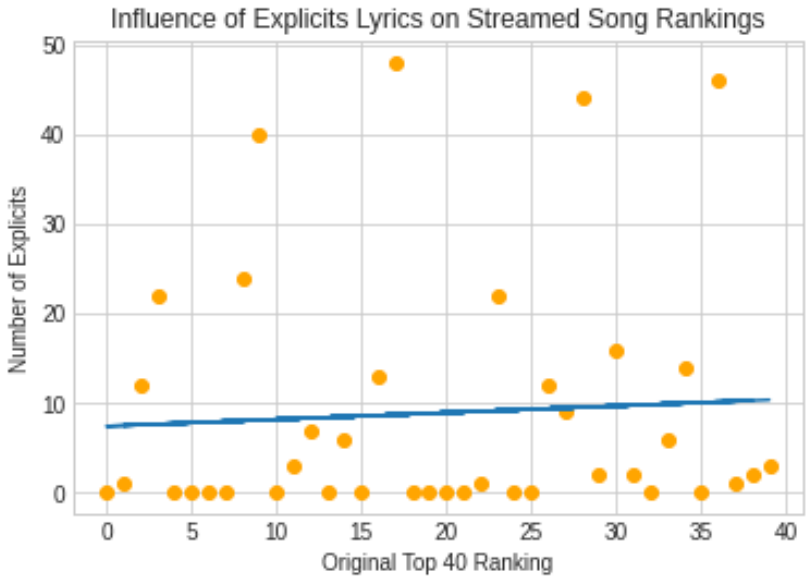

Loudness Unit Full Scale (LUFS) measures the perceived loudness of sound, in addition to the conventional dB measurement. Higher (less negative) score means a higher level of loudness. The largest number of top songs have a loudness score of around -6 LUFS.

**Figure 6: Top song of analyzed Top 40 (based on attributes) print statement.**

```
The song that should be the most successful based on our attribute
analysis is: perfect by ed sheeran .
```

Following the analysis of our Spotify data and the ranking of our songs via their z-scores in relation to each of the remaining attributes, the song with the smallest total of z-scores, aka the song who's attributes' z-scores were the shortest distance away from zero, was determined. This song was presented in a print statement displaying both the song title and the musician, as shown above.

**Figure 7: Scatterplot displaying number of explicit words for Top 40 Most Streamed Songs of 2020.**

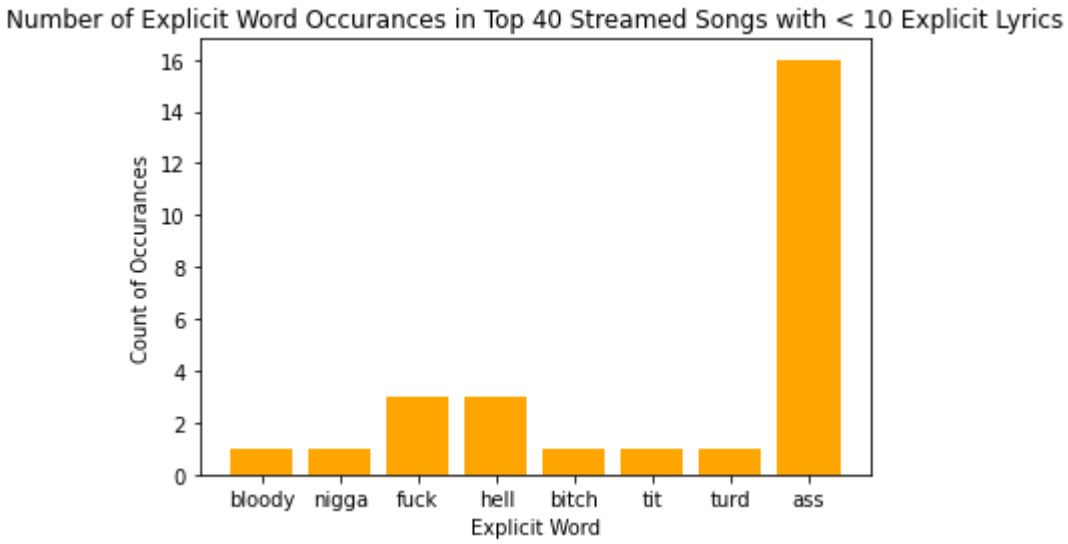



Following the processing of the Lyrical Analysis, we had numerous analyses to make. In regard to explicit content, we can see from the scatterplot (Figure 7), that there does not appear to be a pattern between the number of explicits words in a song and its ranking in the Top 40 Streamed Songs. The song with the most explicit words, "WAP", was ranked 17th. By analyzing the scatterplot, we can see that almost 75% of the songs have less than 10 explicit words, and the 25% that have more are spread fairly consistently across the chart. Because of these results, we can conclude that the number of explicit words in a song does not contribute to the popularity of the song.

After debate, a regression line was included to further prove the randomness of the data. As the line barely has a slope, it shows that there is practically no correlation between the number of explicit words and the song's stream amount. If anything, it's almost a negative effect as the rank of streaming decreases (the worse rankings), the regression line increases, making it appear as if

songs that are streamed more have less explicit lyrics.

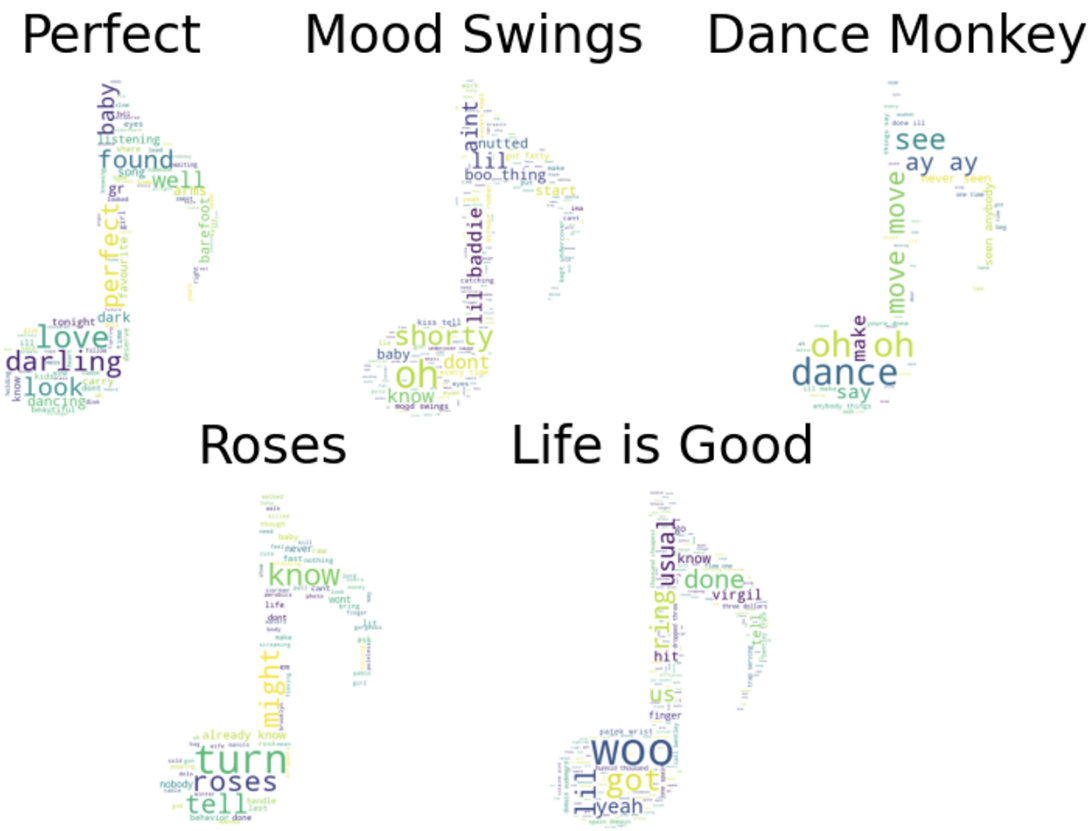
**Figure 8: Barchart displaying the frequency of explicit words for Top 40 Most Streamed Songs of 2020 with < 10 occurrences of explicit words.**




Of the 28 songs with less than 10 explicit words, only 10 actually has at least one explicit lyric. We decided to analyze these lyrics to see which explicit word was the most commonly occurring in the 'less explicit' explicit songs. We can see from the graph that the word 'ass' was by far the most popular.

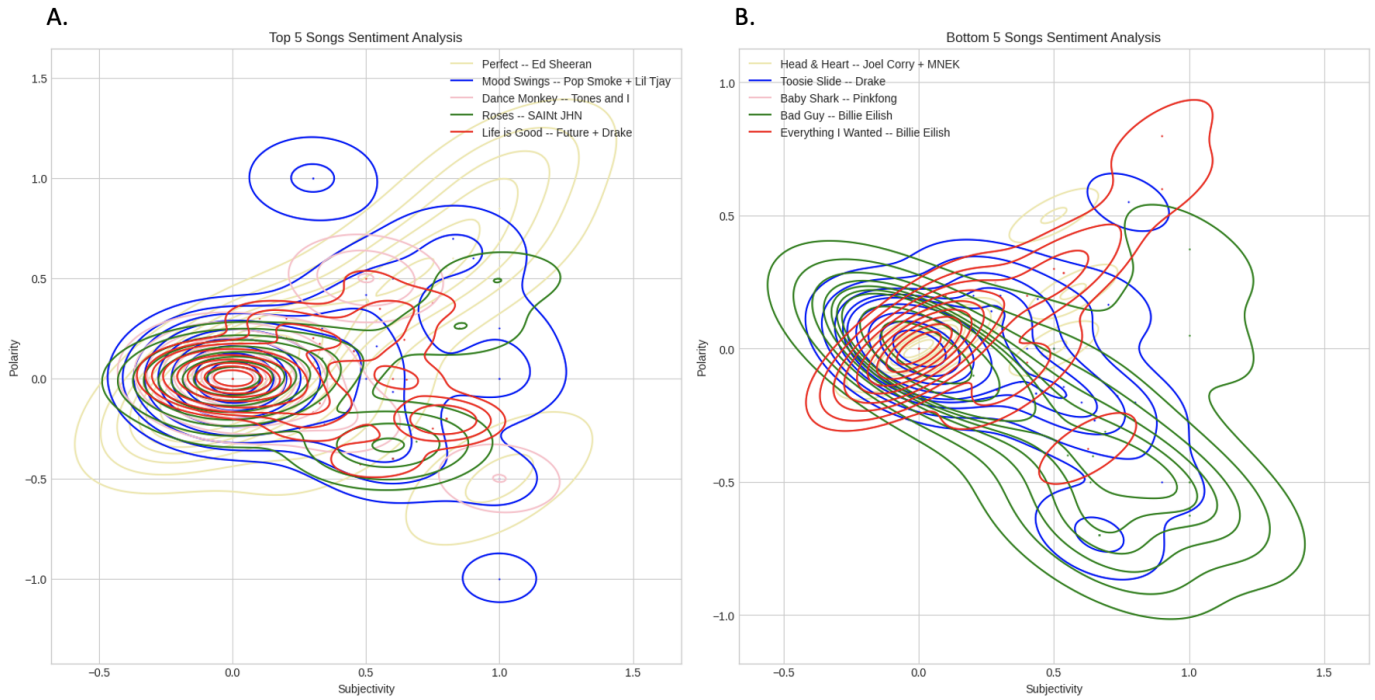
Despite the personal meaning of words is subjective, we believe that this would make sense as the song with the most occurrences in the least explicit songs as it is in no way derogatory. Rather, it is just a word that is typically deemed inappropriate.

**Figure 9: WordCloud visualizations for top 5 songs based on attribute ranking.**




We created WordClouds for the top 5 ranked songs based on our attribute ranking. As can be seen in Figure 9, the most used words in “Perfect” are perfect, darling, and love, while the words used the most in “Mood Swings” are shorty, oh, and lil baddie. For “Dance Monkey”, the most common words are dance, oh, move, and see, and for “Roses” the most used words are turn, roses, tell, and know. Finally, the most common words in “Life is Good” are woo, lil, and got.

**Figure 10: Kernel density estimate (KDE) plots displaying subjectivity and polarity for each line for top 5 songs (A) and the bottom 5 songs (B) based on attribute ranking.**



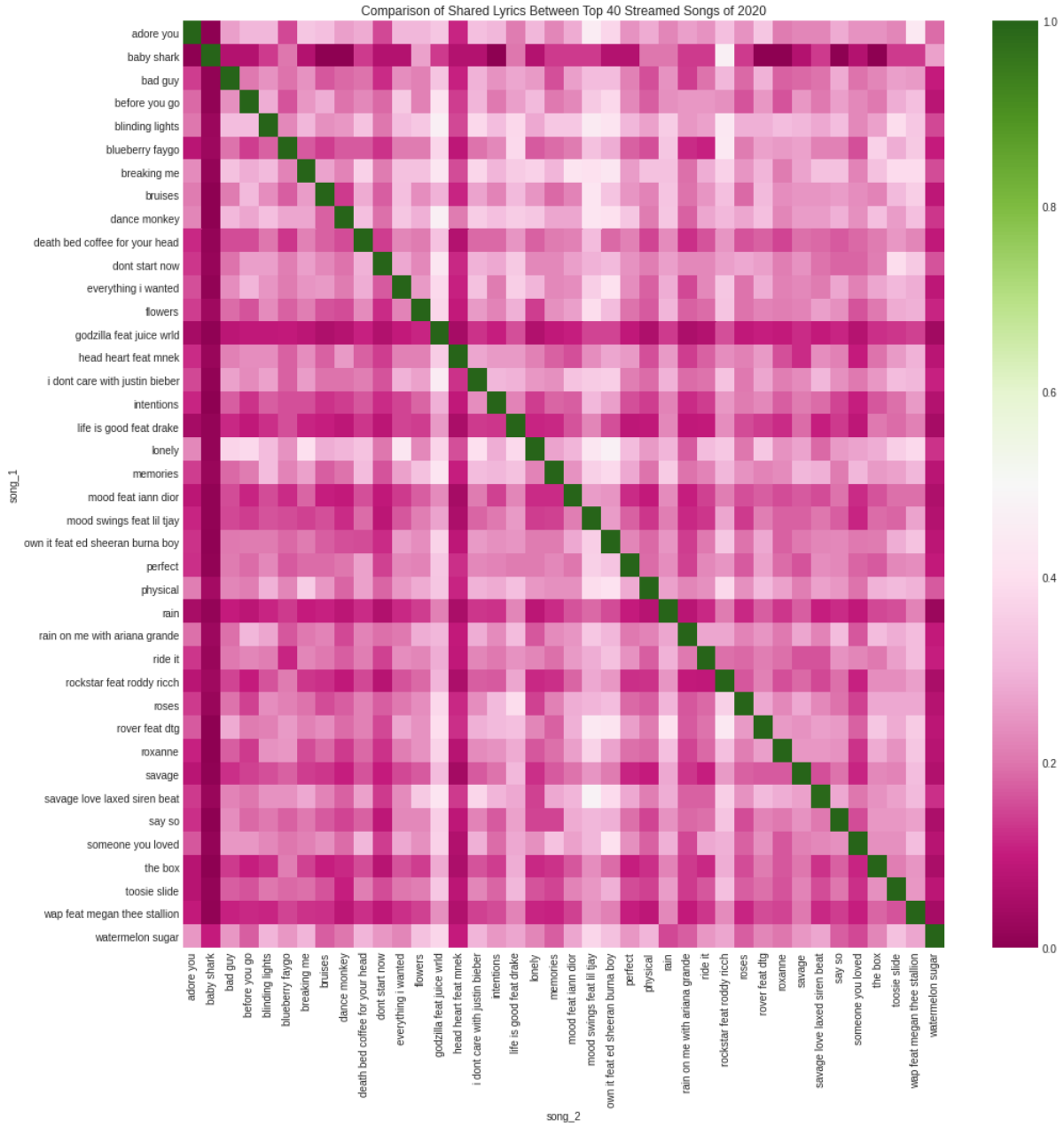

We plotted the subjectivity and polarity of each line for the top 5 ranked songs based on our attribute ranking on a KDE plot (Fig. 10A). As can be seen in Figure 10A, “Mood Swings” has the most negatively polar line as well as a few very positively polar lines. “Perfect” is the most subjective and positively polar of the top 5 songs. “Dance Monkey”, “Roses”, and “Life is Good” are all overall less polar and subjective than “Perfect” and “Mood Swings”.

In Figure 10B, “Everything I Wanted” is the most positively polar out of the 5 bottom ranked songs, and “Bad Guy” is the most subjective. “Bad Guy” also has the largest range of values out of the songs in Figure 10B. “Bad Guy” and “Toosie Slide” seem to be the most negatively polar songs, except for a few lines from “Toosie Slide” that are decently positively polar. “Head & Heart” and “Baby Shark” are overall less polar and subjective than the other songs displayed in the plot. Interestingly, “Baby Shark” only has two lines in the entire song where the subjectivity and polarity values do not equal zero, most likely because the majority of the lyrics consist of “doo doo doo”. It is for this reason that “Baby Shark”’s data cannot be seen on the graph.

We initially believed that more polar and subjective songs do better in our attribute ranking due to “Perfect” and “Mood Swings” being ranked first and second, respectively, and them being so polar and subjective. However, after visualizing the bottom 5 songs from our attribute ranking, it can be seen that “Bad Guy” has around the same subjectivity and it is just more negatively polar than “Perfect” and “Mood Swings”.

“Perfect” is ranked 40th in the Top 40 Most Streamed Songs list and “Mood Swings” is ranked 35th compared to “Dance Monkey” which is ranked 2nd, “Roses” which is ranked 3rd, and “Life is Good” which is ranked 10th. “Everything I Wanted” is ranked 33rd based on streams, and “Bad Guy” is ranked 30th. “Baby Shark” is ranked 26th, “Toosie Slide” is ranked 27th, and “Head & Heart” is ranked 6th. There does not seem to be a correlation between polarity/subjectivity and ranking based on number of streams.

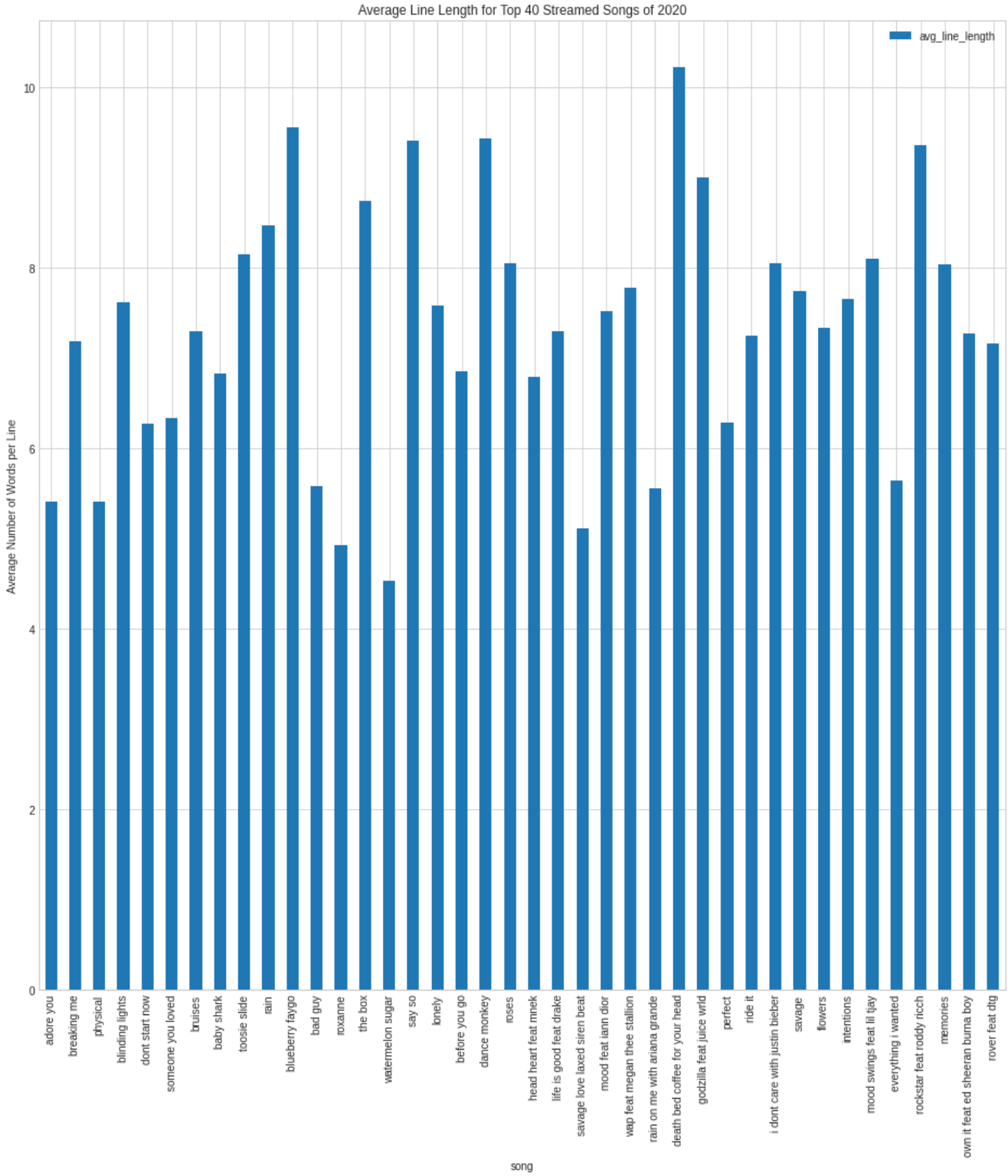
**Figure 11: Heatmap comparing ratio of shared to lyrics out of total lyrics between two songs for Top 40 Most Streamed Songs of 2020.**



We compared the Top 40 Most Streamed Songs of 2020 to determine how many words each two songs had in common. The heatmap's color gauge goes from 0.0 to 1.0, with the value representing the number of common words divided by the number of total words for song_1. As can be seen in Figure 11, when "Baby Shark", "Godzilla", and "Rain" are song_1 they seem to have very few common words with the other songs as most of their color blocks are a darker pink. The two songs that share the most words are "Savage Love" and "Mood Swings". 47.78% of "Savage Love"'s lyrics are shared with "Mood Swings", specifically 43 words. "Mood Swings" is ranked second based on our attribute ranking, whereas "Savage Love" is ranked 24th. In the Top 40 Most Streamed Songs list, "Mood Swings" is ranked 35th and "Savage Love" is 12th. The two songs with the lowest number of shared words over zero are "Baby Shark" and "Godzilla" with 0.5% of "Godzilla"'s lyrics being shared with "Baby Shark", specifically 2 words. "Baby Shark" is ranked 32nd while "Godzilla" is ranked 12th by our attribute ranking system. "Baby Shark" is ranked 26th and "Godzilla" is ranked 24th in the Top 40 Most Streamed Songs list. The pair of songs with the most common words and the pair with the least common words (over zero) demonstrate that there is most likely no correlation between number of common words and attribute ranking. Additionally, "Perfect", the top ranked song based on our attribute ranking, does not seem to have an usually high or low number of common words with other songs when it is song_1, further supporting the conclusion that there is no connection between number of common words and attribute ranking. "Baby Shark" and "Godzilla" are ranked closely based on streams, whereas "Savage Love" and "Mood Swings" are not ranked closely based on streams. This data does not show any clear correlation between the number of common words and ranking based on streams.

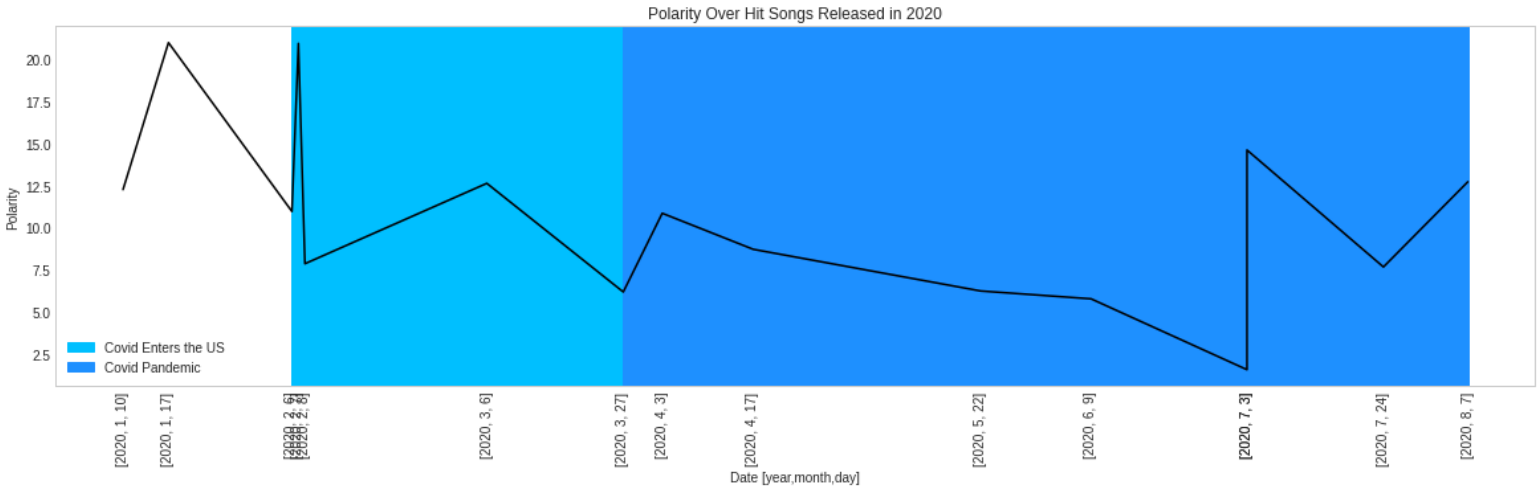
There are 16 instances where there are no common words between songs, and in all of these cases "Baby Shark" is one of the songs being compared. Out of the Top 40 Most Streamed Songs, "Baby Shark" is the outlier in that it is the only song with a target audience of young children. This is most likely the reason why it shares so few words with the other songs from the song list.

**Figure 12: Histogram of average line length for Top 40 Most Streamed Songs of 2020.**



We calculated the average number of words per line for each song in the Top 40 Most Streamed Songs of 2020 list. The song “Death Bed” has the longest lines, with an average of 10.23 words per line. The song with the shortest lines on average is “Watermelon Sugar”, with an average of 4.53 words per line. Most songs are between 6 and 9 words per line on average. “Death Bed” is ranked 28th and “Watermelon Sugar” is ranked 23rd. “Perfect” is ranked first in the attribute ranking and last in the streaming ranking and has an average of 6.28 words per line, and “Everything I Wanted”, which is ranked last in the attribute ranking, has an average of 5.65. There does not seem to be any correlation between attribute ranking and average line length. “Dance Monkey” is the highest ranked song in our dataframe based on the Top 40 Most Streamed Songs list as number 2, and its average is 9.43. “Roses” is ranked third based on streams and has an average of 8.05, and “Rockstar” is ranked fourth based on streams and has an average of 9.36. It is possible that songs that are streamed more have more words per line on average, as “Dance Monkey”, “Roses”, and “Rockstar” are highly ranked based on streams with higher averages, while “Perfect” is last on the streams ranking list with a much lower average.

**Figure 13: Polarity of songs released in 2020 by release date.**




Using the release date provided in the dataset and the polarity metric we calculated for each song, we showed the change in polarity of original Top 40 songs over the course of the year. The trend that this chart shows is that in the beginning of the year, the polarity of songs was higher, but as the Covid grew into being a pandemic, the polarity of popular music steadily declined until the July, where it began to pick up again.

One point of note is the nature of the available songs within this timeline itself. Out of 34 top 40 songs we were able to run polarity metrics over, only 15 of them were released in 2020. Out of 15 songs, 8 of them were released during the pandemic. It was also found that there was a great increase in average time between release dates during the pandemic. Before the pandemic(January 1st - January 24th)[1] , top 40 songs were released 7 days apart on average. In the period when the

first cases of Covid entered the United States (January 25th - March 10th)[1], average time between song releases increased to 12.25 days between each song. The largest gap between song releases was found during the time period where Covid was declared a global pandemic (March 11th - Present)[1], where the average time between release of top 40 songs was 16.77 days between top 40 song releases, with the longest stretch of time between releases also found in this time period with a 35 day stretch between the release of Rockstar by Dababy and Roddy Rich (4/17/20) and Rain On Me with Lady Gaga and Ariana Grande (5/22/20).

With the amount of lyrical data being used and analyzed, it seemed natural to implement a Lyric Generator. Lyrics were taken from the cleaned_lyrics dictionary and converted into a list of all song lines called phrases. Beginning and end words were noted, as well as their surrounding words to better understand the buildup of the phrases.

Once these words were established, a dictionary was created that contained each word from the list as a key, and the words that come after it as its values. Once the dictionary was created, now called ngrams,  the words were ready for lyric generation.

The generate_lyric function takes beginning words, its ngrams, and ending words. The function then adds words starting at a random beginning word and adding its ngrams until an ending word is reached.

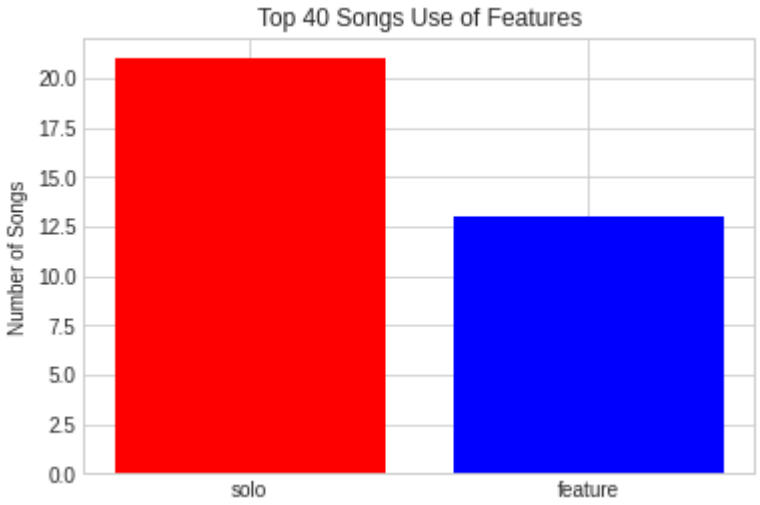
**Figure 14: Lyric generator example output.**

```
fuckin with

yeah

everybody hurts sometimes

id let you

big bands

im just hold

chest always in

im about you

got

everything look at
```

The repeat_function uses a for-loop to run the generate_lyric function an n determined number of times. Each time through the loop, a line is added to a list called song. The song is returned at the end of the function. The print statement presents the song to as lines in a list with each list item

being a line. In this example, there are 10 lines (n =10).

**Figure 15: Histogram of number of Top 40 Most Streamed Songs with featured artists.**




Using the features_artist function we were able to find whether or not a song contains a features artist. The solo column represents songs that do not contain a feature and feature represents songs that do contain a feature. This data is plotted in this bar chart, showing that out of 34 songs on the top 40, 13 (38%) contain a featured artist while 21 (62%) do not.

**Figure 16A: Distribution of Top 40 Most Streamed Songs ranking based on solo and featured artists.**

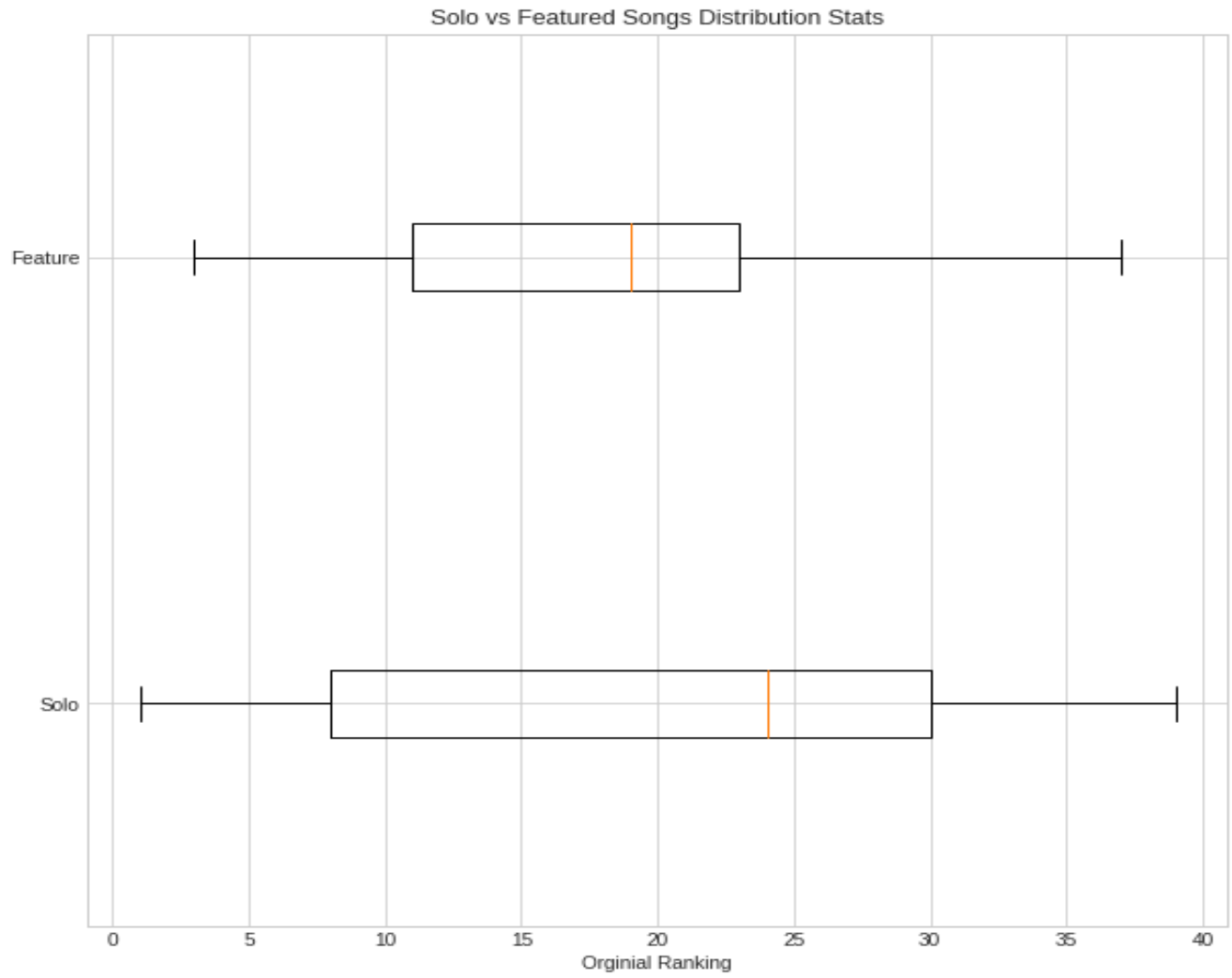
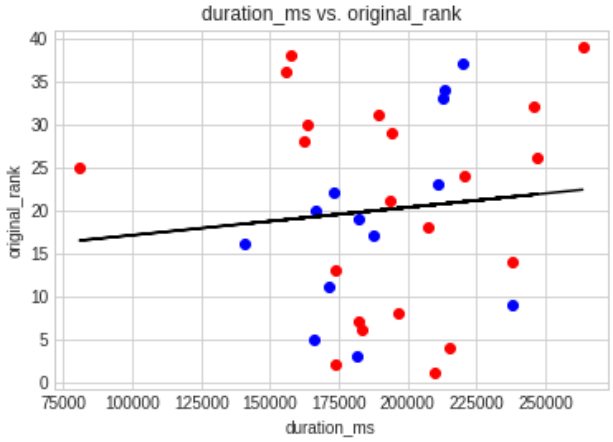


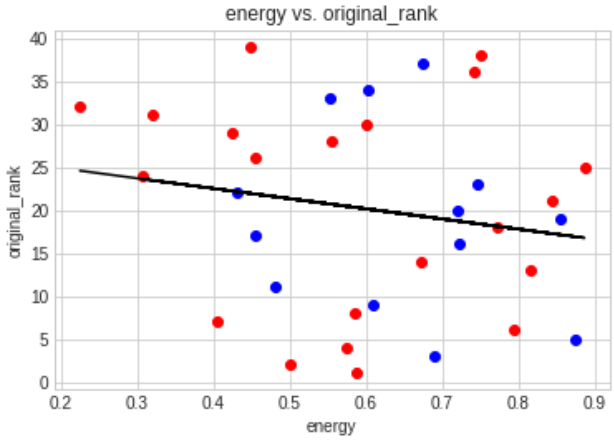
**Figure 16B: Descriptive statistics.**

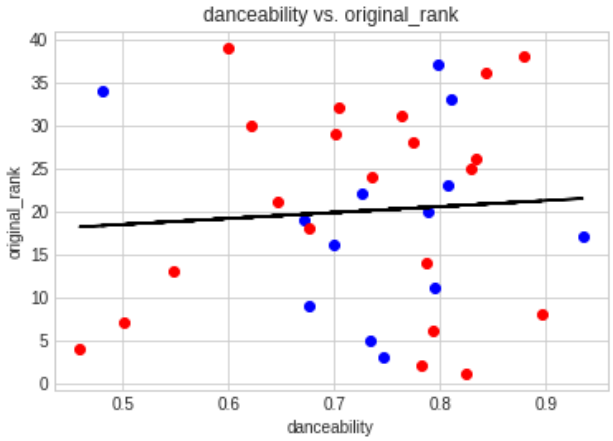
| | Featured | Solo |
|---|---|---|
| count | 13.000000 | 21.000000 |
| mean | 19.153846 | 20.571429 |
| std | 10.800047 | 12.371627 |
| min | 3.000000 | 1.000000 |
| 25% | 11.000000 | 8.000000 |
| 50% | 19.000000 | 24.000000 |
| 75% | 23.000000 | 30.000000 |
| max | 37.000000 | 39.000000 |

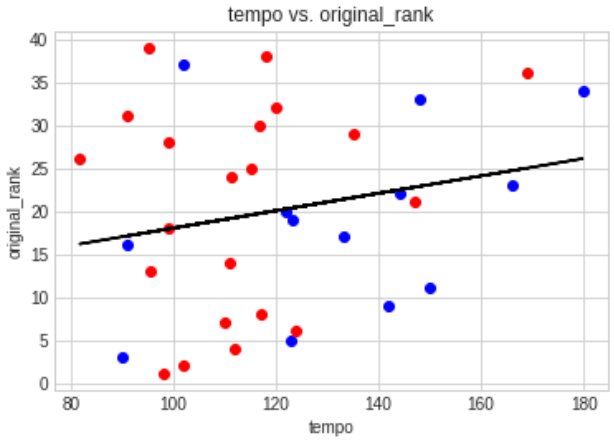
When it comes to the differences in the songs with features, on average, songs with features performed very slightly better (average ranking of 19.15) than songs without features (average ranking of 20.5). With the featured interquartile range being 13 and and solo interquartile range being 28, as well as the standard deviation for featured being 10.8 while the standard deviation for solo songs being 12.8, songs without a feature tend to have a larger degree of variation than those who contain a featured artist. From this, we can conclude that although the difference is slight, the rankings of solo songs are less consistent, and worse than those of songs with featured artists.

**Figure 17A: Ranking attributes vs. original ranking.**



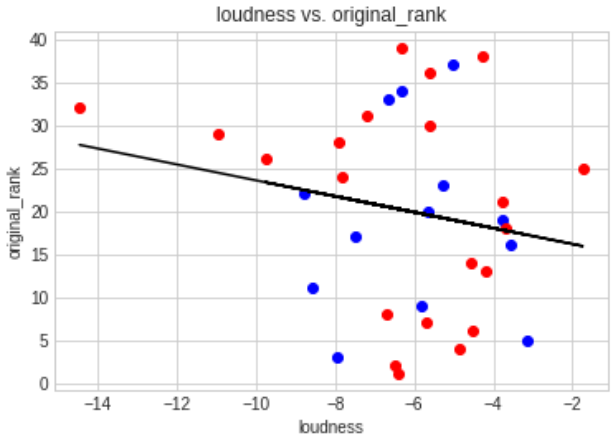



**Figure 17B: R values for regressions.**

| Category | R-Value |
|---|---|
| Duration (MS) | 0.09851714277097408 |
| Energy | -0.17687315643448473 |
| Danceability | 0.06877807461208926 |
| Tempo | 0.211811899540341 |
| Loudness | -0.19588524224827764 |

Figure 17A shows scatter plots comparing each variable that was compared in the ranking function done previously and in their original top 40 ranking. One the scatterplot, the red points represent songs without features(solo) while blue points songs with features. reprentFigure 17B shows the R values generated by regression models on the scatterplots generated in Figure 17A.

Although there is not a strong correlation between these variables and their original ranking, we have found that tempo has the strongest correlation with an R value of 0.2118.

## Conclusions

In conclusion, streamed songs successes are not determined by both musical and lyrical attributes. We hypothesized that one of the top 5 songs in the Top 40 Most-Streamed Songs list would be most successful in all musical attributes, however we were incorrect. It was determined that the song “Perfect” by Ed Sheeran, which was ranked 40th in Top Streamed Songs, is most successful.

From our Lyrical Analysis we found that there may be a correlation between longer lines in songs and more streams as songs with larger average line lengths seemed to be ranked higher on the Top 40 Most Streamed Songs list. Otherwise, there are no other correlations between the stream ranking and subjectivity/polarity, average line length, or common words between songs. We did not find any correlations between these lyrical characteristics and our attribute ranking. The number of explicits in a song also did not appear to affect the Top 40 Streamed ranking of a song.

There are a few limitations to our project. Namely, some of the songs listed on the Top 40 Streamed Songs of 2020 list were not included in the Spotify dataset or had NaNs, and so they could not be included in our analysis. “Blinding Lights”, “Own It”, “Lonely”, “Rover”, “Rain”,

and “Flowers” all had NaNs and were not included in the Spotify Analysis. Finding the data or taking the time to complete the dataset ourselves would allow us to use them in future analysis.

Another limitation that we had involved filtering out explicit words. We originally filtered out only exact matches for explicit words, but it left behind some words that had explicits within the word. We then switched to using a function that matched substrings. However, this captures a broader set of words than just explicit words. For instance, it will catch a word like “pass” and consider it explicit because it contains the substring “ass.” We saw no real good way to reconcile this issue of being overly strict or not strict enough in catching explicit words. We decided to go with the overly strict filter when counting the number of explicit words and removing words for the WordCloud visualizations, but this could bias the explicit word analysis.

While the Lyric Generator does in fact generate lyrics, we would like to work with it more in the future. For starters, the available possible lyrics are limited to the words used in the 40 songs used throughout the project. By increasing the library of words, the lyrics generated will create even more possibilities. In addition, we would also like to work to make the outcome fit more properly with both grammar and lyrical song structure. In order to do this properly and increase the likelihood of positive machine learning, some lyrics, such as features in songs that are in a different language, may need to be filtered out. We would also like to work to be able to adjust individual line lengths.

With that being said, we accomplished a number of things, for both Spotify and Lyrical Analyses. One of our biggest successes with the project was our ability to synthesize data across multiple datasets to analyze songs across a variety of attributes.

In regard to Spotify Analysis, we were able to take a variety of attributes for a list of songs, analyze their distances from the average, and determine which song best met the averages. We also analyzed the relationship between the original rankings and the song attributes to determine how much of an effect the attributes have on song popularity. We were able to determine, out of the variables we took into our ranking function, which factor has the greatest correlation, which was its tempo.

Aligning a timeline of the covid pandemic with release date analysis, we can conclude that the output of Top 40 singles decreased as the pandemic evolved, and during the initial quarantine the polarity of singles released decreased before picking up again in the summer. Another interesting observation is that a majority of the 34 songs we were able to calculate polarity metrics were released before 2020, with only 15 of them being released within the year of 2020. This could also speak to a general slowing down of output during that year, or potential nostalgia for a pre-pandemic world.

We were also able to look at the influence of that having a featured artist on the song had on its Top 40 ranking. 38% of the songs in the top 40 contained a featured artist while 62% had no feature. Although solo artists held a majority, it can be acknowledged that collaboration between established artists is a common and successful practice. We found that, on average, songs with features tended to have a higher rating (given that rankings are assigned in ascending order) and

there was a lot more variation in performance for songs that had a solo artist than songs with a featured artist. A possible conclusion that can be drawn from this is that solo artists can be more varied because it rides mainly on their personal brands, while songs with features artists rely on cross appeal between fanbases, which is seen as a safer bet for doing well in the charts.

In regard to Lyrical Analysis, we developed functions to look at subjectivity and polarity, average line length, and common words between songs. We also developed functions to analyze explicits. We created WordClouds, KDE plots, heatmaps, scatterplots, and histograms to display our data. We believe these visualizations communicate our findings clearly in a visually appealing way.

Ultimately, we concluded that both Spotify Analysis and Lyrical Analysis are important factors to take into account when analyzing music. While the instrumentals may follow patterns for success, the lyrical build-up may not, or vice-versa. Because of this, both must be considered in order to predict or analyze the popularity of a song.

## Author Contributions

Gianna Barletta: Reading in data and cleaning for both halves, creating dictionaries for various stages of cleaning, formatting data into dataframe with all lyrical analysis functions outputs together, remove_more_words function, total_specific_words function

Sophie Lefebvre: clean_text function, remove_words function (for stop words and explicits), subjectivity/polarity function and KDE visualization, common words function/heatmap, WordCloud function/visualizations, average line length function/histogram

Grace Michael: Code, PowerPoint and paper outline and organization, winner print, explicits organization, analysis (explicit and ex_word_count functions), and visualizations, Lyric Generator

Preston Reep: Statistics function, Ranking function, Release date analysis, Regression model, Feature analysis

Dachuan Zhang: Attribute histogram functions, generated preliminary dataframe for analysis, hosted GitHub Repository for the project

*GitHub Link*: https://github.com/zhang-dachuan/ds2500_project

## References


1. *A Timeline of COVID-19 Developments in 2020*. www.ajmc.com/view/a-timeline-of-covid19-developments-in-2020

2. . Ay, Y. E. (2021, January 24). Spotify dataset 1921-2020, 160K+ Tracks. Retrieved March 29, 2021, from

https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks?select=data.csv

3. *BannedWordList.com - a Resource for Web Administrators*, www.bannedwordlist.com/lists/swearWords.txt.

4. Copsy, R. (2021, January 04). The official Top 40 biggest songs of 2020. Retrieved March 29, 2021, from https://www.officialcharts.com/chart-news/the-official-top-40-biggest-songs-of-2020__29264/

5. Musixmatch. (n.d.). Retrieved March 29, 2021, from https://www.musixmatch.com/