Project 3: Song popularity on year-end charts

Tejit Pabari (tvp2107) & Canfer Akbulut (caa2175)

Introduction

Imagine that you are an incredibly ambitious artist looking to make a mark on musical history. Your goal is not to create a song that registers as popular for a few weeks before fading into obscurity; you want to create a hit that becomes so noteworthy that it defines popular culture at the time of its release. Certain accolades are indicative of this level of success. For instance, winning Song of the Year at a reputable award show such as the Grammy's would signify that your release was found to be of exceeding cultural significance. However, voting procedures for awards shows are often intransparent, and only one piece of music is awarded this honor on a yearly basis, making this metric unsuitable for data mining purposes. One measure of cultural relevance and popularity that does lend itself to large-scale data analysis is the Billboard Hot 100 Year-end Chart. Released annually to reflect performances of singles across a chart year within the United States, the Year-end Chart provides a glimpse into which songs and musical artists dominated the musical culture in any given year. Think back to the year 2011, when Adele's "Rolling in the Deep," LMFAO's "Party Rock Anthem," and Katy Perry's "Firework" blared from every radio station. Occupying the top 3 spots on the Year-end Chart list, these songs became so influential to the musical landscape of 2011 that even a decade later, music listeners are able to recall a time when these tracks were nearly inescapable. The aim of our project is to pinpoint the features that unite tracks that reach these high levels of cultural relevance and popularity. Though there is no foolproof method of creating a smash hit, we hope to help determined nascent artists identify approaches that have worked for artists in the past. This, in turn, may allow them to finetune their song-writing processes to achieve their desired result: a coveted top ten spot on the Year-end Chart list.

Measures

All variables denoted in *italics*.

We used Billboard API to scrape all songs that made the Top 100 Year-end Chart for all the years that were available (2006 - 2020). This provided us with the *song names*, *artist names, the chart positions*, and *the year in which they charted*. Now that we know the songs that we want to use to inform our analyses, we want to collect metrics that would allow us a better understanding of what distinguishes some songs from others. For example, how up-beat is a song? How much of its runtime is occupied by instrumental versus vocal composition? How positive or negative are the lyrics?

To assist us in doing so, we scraped the lyrics for all songs that made the Year-end Chart between 2006 and 2020 using Genius API. We were then able to compute *lyric valence* and *lyric repetitiveness*. Audio features of the songs were made available through Spotify API. Spotify offers developers several unique pre-engineered audio features which they use to classify songs, including *danceability*, *explicitness*, *loudness*, *acousticness, energy, liveness, valence, tempo, key,* and *speechiness* (for an explanation of each of these features, see Figure 1).

Feature	Description	Retrieved from
Song name	Name of song as it appears on the chart	Billboard
Artist name	Name of artist as it appears on the chart	Billboard
Chart position	Position at which song was placed on Year-end list	Billboard
Chart year	Year for which any particular song charted	Billboard
Lyric valence	Emotional valence of lyrics on a scale from -1 (most strongly negative) to 1 (most strongly positive)	Genius (engineered feature)
Lyric repetitiveness	Number of unique words in lyrics divided by total words	Genius (engineered feature)
Song duration	duration of the track in milliseconds*	Spotify
Song popularity	popularity of the track from 0 to 100 (0 = least popular, 100= most popular); popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are *	Spotify
Danceability	how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity (value of 0.0 is least danceable and 1.0 is most danceable)*	Spotify
Explicitness	whether song is marked as explicit (binary outcome with 0 = not explicit and 1 = explicit)	Spotify
Loudness	overall loudness of a track in decibels (typical range: -60 and 0	Spotify

	dB)*	
Acousticness	confidence measure from 0.0 to 1.0 of whether the track is acoustic (1.0 represents high confidence the track is acoustic)*	Spotify
Energy	measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity (death metal = high energy; Bach prelude = low energy)*	Spotify
Key	key of track in Pitch Class notation (0 = C, 1 = C♯/D ♭ , 2 = D...)*	Spotify
Mode	indicates the modality (major or minor) of a track (minor = 0; major = 1)*	Spotify
Speechiness	presence of spoken words in a track; more exclusively speech-like the recording (e.g. talk show, audio book, poetry) are closer to 1.0*	Spotify
Instrumentalness	Predicts whether a track contains no vocals; values closer to 1.0 represent greater likelihood the track contains no vocal content*	Spotify
Liveness	detects the presence of an audience in the recording from 0 to 1; higher liveness values = an increased probability that the track was performed live*	Spotify
Song valence	measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track (0 = most negative; 1 = most positive)*	Spotify
Tempo	Overall estimated tempo of a track in beats per minute (BPM)*	Spotify

*Table 1. Names of all features in analysis with descriptions and the source from which the metric was retrieved. *description quoted from Spotify documentation*

Some songs made the year-end lists for multiple years; to gauge the songs at their best performance (and to ensure that a song was not disadvantaged in its position because it had been released prior to other charting songs), we computed the *maximum rank* of each song - that is, the highest position a song occupied across Year-end Charts. Additionally, our chart raw ranking metrics were unsuitable for use in classification, as they present 100 labels with few data points corresponding to each label (for instance, with only one Number 1 spot per year, an aggregation of Number 1 Singles over 15 years would only provide 15 tracks for us to work off of). For ease of analysis, we assigned labels to our songs based on where they fall in three groups: 1 - 10, 11- 50, 51- 100. This

variable, called *ranking-3*, allows us to better implement classification algorithms further down the line.

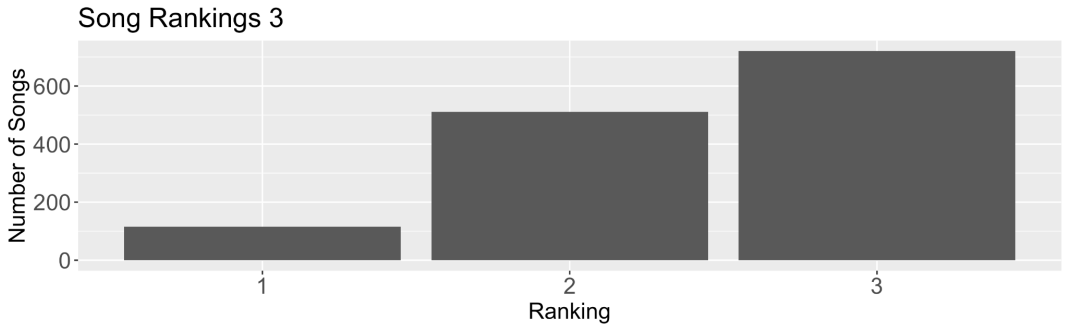


Figure 1: Histogram of the three classes of song ranking (Top Ten, 11-50, 51 - 100).

The above figure shows the increase in data per class, as opposed to 10-15 data points per 1-100 rank.

Data Exploration

Data Pre-processing

Much of our pre-processing concerned the data that was collected using Genius API, as the process of scraping supplied only the raw lyrics of the songs. The following steps were taken before the valence and repetitiveness of the lyrics were calculated:

1. We converted all space characters, including tabs and new lines, into single spaces.
2. We expanded contractions, so that words such as “I’m,” “I’d,” and “it’s” were turned into “I am,” “I would,” and “it is.” This was to make the data more uniform across the board; certain songs have a tendency to contract more than others, and we wanted to make sure that this did not introduce disparities in the word lengths of the songs as we calculated repetitiveness.
3. All punctuation marks, including apostrophes, were removed.
4. All text was changed to lowercase.

Before processing	After processing
Did a full one-eighty, crazy\n Thinking 'bout the way I was\n Did the heartbreak change me? Maybe\n But look at where I ended up\n	did a full one eighty crazy thinking bout the way i was did the heartbreak change me maybe but look at where i ended up

Table 2. An example of a raw and processed song lyric.

No stop words were removed from the lyrics as part of pre-processing. We made this decision because doing so would defeat the purpose of calculating repetitiveness, and because much of the time, and song-writers inclusion of stopwords is a deliberate artistic choice. Take the bridge of Dua Lipa's "Don't Stop Now," ranking in at Number 4 on the 2020 Year-end Chart: "Don't show up, don't come out /Don't start caring about me now / Walk away, you know how /Don't start caring about me now ('bout me now, 'bout me)." Fifty-six percent of the words in the bridge are stopwords, so it is easy to see how the removal of stop words would have rendered our engineered features meaningless.

We scaled the engineered lyric features and the audio features provided by Spotify to correct for 1) the differences in range and 2) differences in units across measures. By differences in range, we mean that while certain audio features that have the same theoretical maxima and minima (0-1) have different ranges in practice. For instance, *speechiness* (min: 0.023, mean: 0.104, max: 0.59, range: 0.568) covers much less area and has a less even spread than a song's *energy* (min: 0.056, mean: 0.67, max: 0.983, range: 0.9265). Some of our features are in different units. For instance, song loudness is given in decibels (min: -20.41, mean: -5.843, max: -1.19, range: 19.223), while song duration is given in milliseconds (min: 78200, mean: 225903, max: 688453, range: 610253). For the purposes of modeling, it was appropriate for us to scale our variables to reduce the great variability in the raw data.

Data Statistics

In this section, we examine statistics of the variables to better understand the data we have collected.

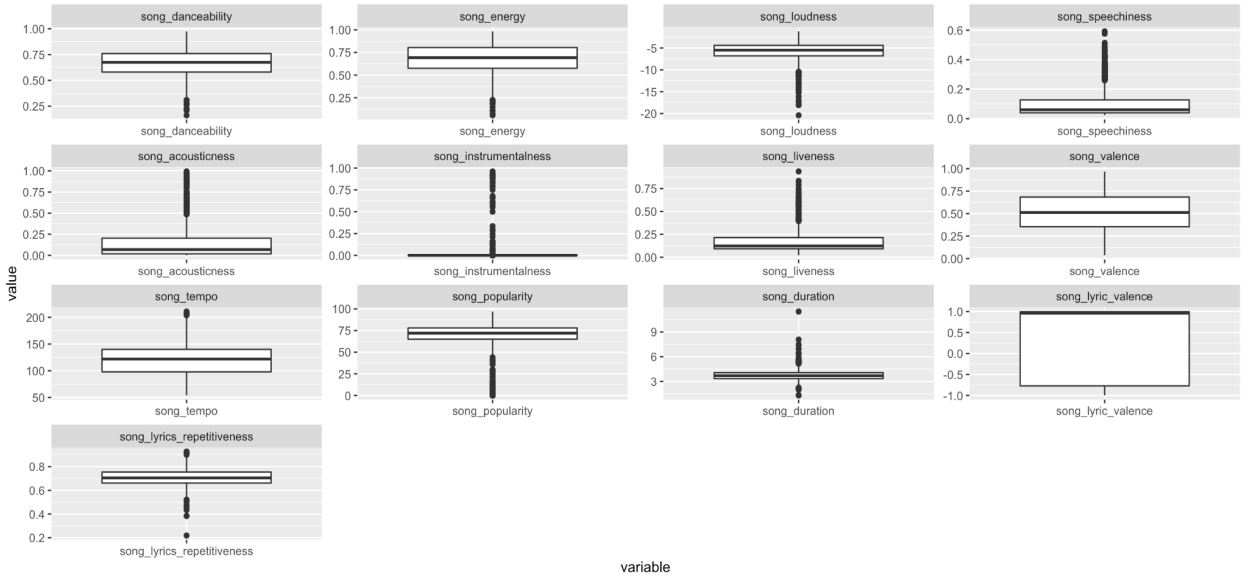


Figure 3: Histograms of all variables, showing the spread of all numeric variables.

As seen in the visualization, many features extracted from Spotify lie between 0 and 1, though many of them are not evenly distributed within the specified range. There is significant skewness to several variables in the data, which we later correct for with scaling. However, taken together, it seems as though most data points are concentrated within a restricted range.

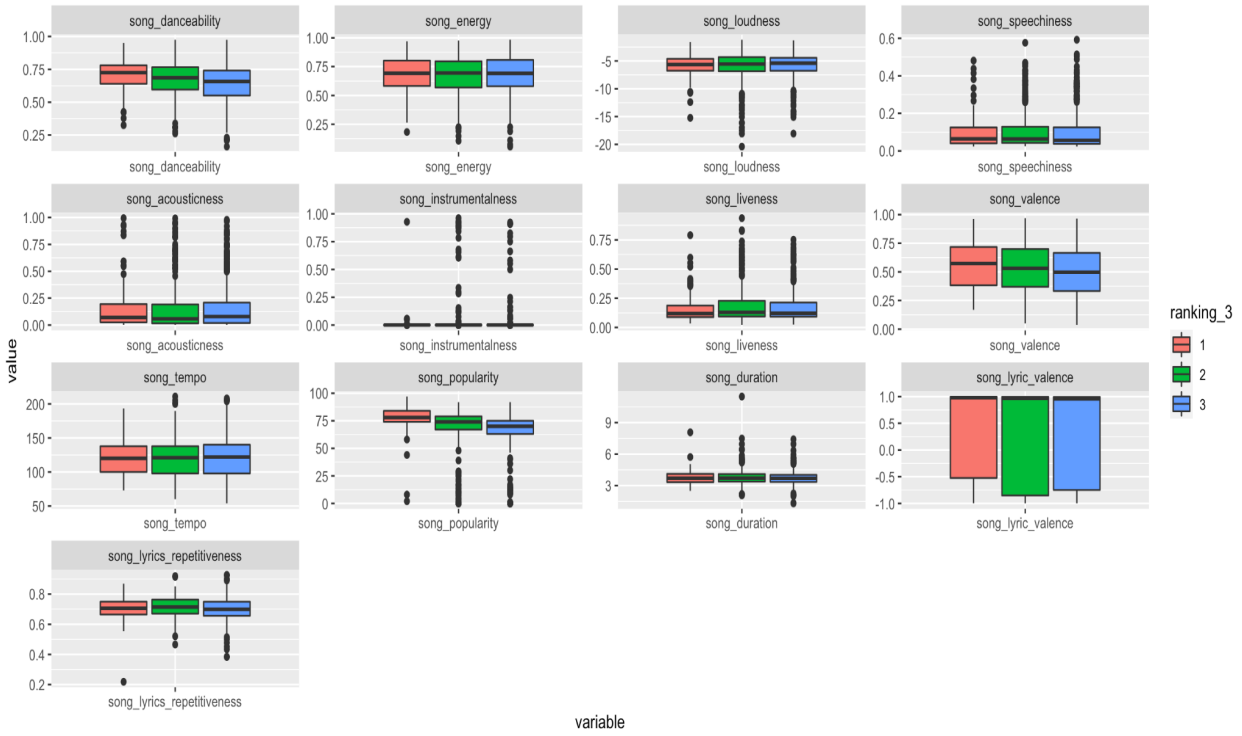


Figure 4: Histograms of all variables color coded by the three ranking classification labels (1 = Top Ten, 2 = 11 - 50, 3 = 51 - 100).

Here, we represent the distributions of the variables by classification group, where the red plots represent songs that make it into the coveted Top Ten list in the year-end charts, green plots represent songs that rank 11 - 50, and blue plots represent the bottom half of the year-end charts from 50 to one-hundred. This allows us to see which features differentiate songs in one group to songs in another. For instance, it looks as though danceability tends towards higher values for Top Ten songs than for songs ranking in at 21 - 50, which in turn have higher danceability than songs in the bottom 50 of the year-end charts. The same pattern can be observed for song valence, while for other features, such as song tempo, do not seem to vary significantly between our groups.

Data Limitations

One of our primary concerns in gathering data from Billboard API was that year-end chart data is only available from the year 2006 onwards, though Billboard

began to publish weekly chart data in 1958. What this means for our analyses is that we have somewhat limited data for classification purposes. Very few data points inform what makes a song go Number 1, for instance; in our early attempts at implementing a classification algorithm, having too many labels corresponding to chart rankings and too few points that belong to each label resulted in accuracy figures that were extremely low. Though new group designation labels (Top 10, 11 - 50, 51 - 100) remedied our accuracy problem somewhat, it would have been helpful to have access to more year-end data. This would have allowed us to provide answers for more specific questions about why songs chart at certain positions (why did this song go Number 1?), rather than having to widen our scope to the general position of a song within the charts (why did this song end up in the Top 10?). It would have been helpful to have more data points in general, as it would better inform prediction algorithms that we execute later on (for example, having more Top Ten entries in general would allow for a better sense of what similarities Top Ten entries over the years may share). Additionally, even when we create three classification categories, we are unable to create even data distribution across groups. The Top Ten, by definition, has less entries than the bottom 50 of the charts, and so our classification groups were imbalanced in their size. However, arbitrarily splitting our data into three categories (1-33) seemed like the less favorable option, as there would be no reason our wider audience would be interested in what distinguishes songs songs ranking 1-33 from songs that chart at 34 -66 .

More trouble arises from using different sources of information to convey similar concepts. Spotify and Billboard both provide metrics of song popularity: Billboard's *year-end ranking* uses the track's performance across the weekly charts to calculate where a song should be ranked, while Spotify's *song popularity* metric is mostly contingent on "the total number of plays the track has had and how recent those plays are." At a first glance, these two variables may seem as though they are reflections of the same concept, but with a correlation coefficient of $r = -0.16$ and an $R^2 = 0.02$, it seems that the relationship between these two variables is tenuous at best (see Figure 5). Similarly, we would think that song valence as retrieved from Spotify and lyric valence calculated by using Genius lyrics should be variables that have strong associations with one another, but the existence of a relationship between them is doubtful with $r = 0.08$ and $R^2 = 0.006541$. A limitation inherent in our data is that we are extracting information from different sources, which may each have drastically different ways of arriving at the same concept, be it popularity or valence. We must be mindful of the fact that our data does not come from a single source and maintain a critical eye while analyzing the results of our various models, as these discrepancies in our data can lead to misleading conclusions about what each of these features is indexing.

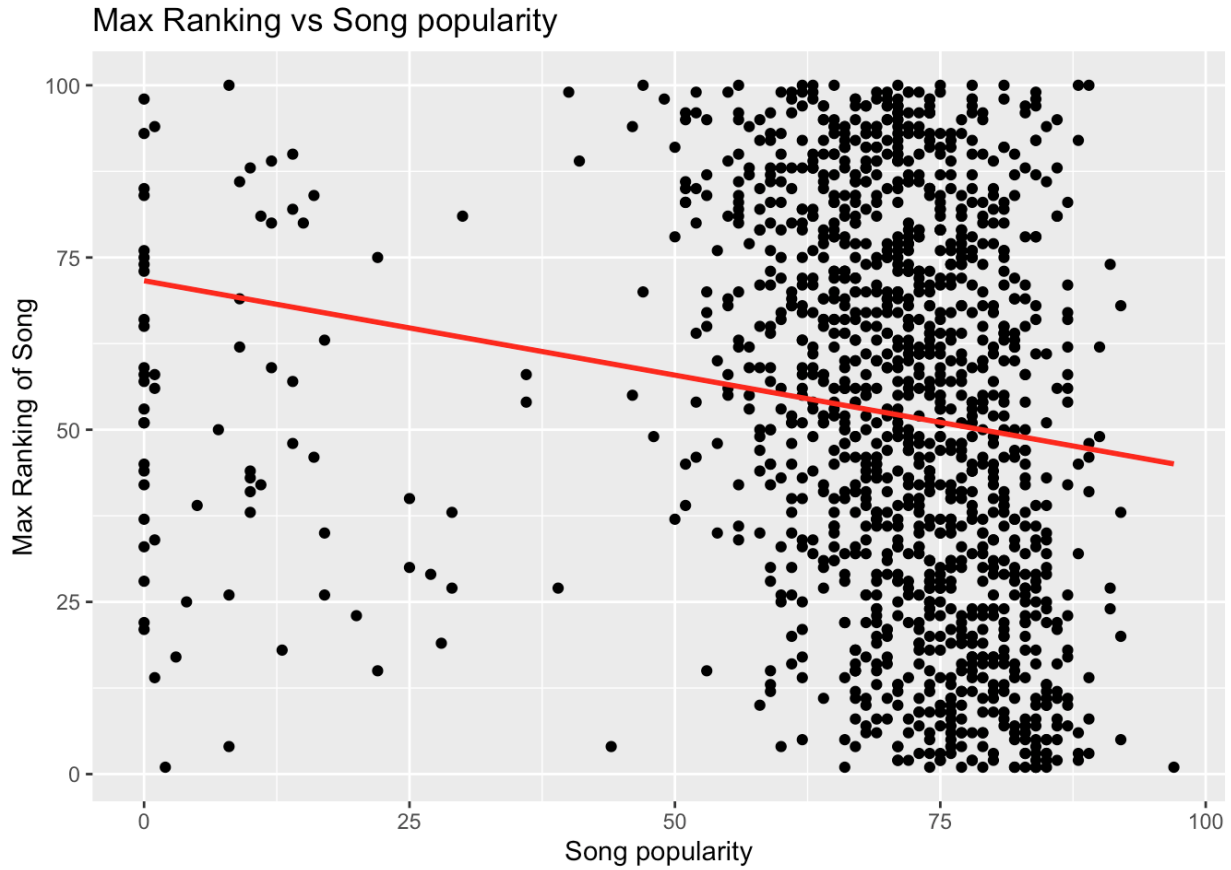


Figure 5. An ordinary least-squares regression plot of song popularity (Spotify metric) and max ranking of song (Billboard metric), demonstrating dubious association between two variables that, intuitively, should move together.

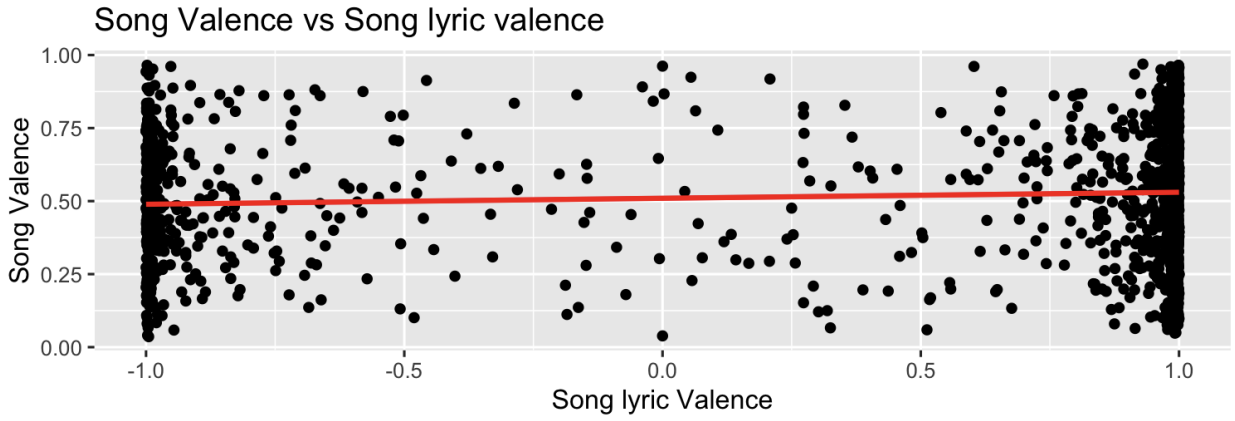


Figure 6. An ordinary least-squares regression plot of song lyric valence (engineered feature from Genius) and song valence (Spotify metric); though the two variables should have a strong relationship, the model has relatively poor fit.

An additional weakness introduced by the use of different APIs is that an identical track can be referenced in a number of different ways across services. Cardi B's 2018 smash hit is called "Bodak Yellow" on Spotify, but referred to as "Bodak Yellow (Money Moves)" on Billboard's charts. Imanbek's remix of SAINt JHN's "Roses" occupied a spot on the 2020 Year-end Chart, but it takes a lot of fine-tuning on Genius API to retrieve the remix rather than the original. Though we manually checked through problematic entries, there is always the possibility that an incorrectly identified song makes it into our dataset; this introduces extra variability that we are not able to account for in our analyses. Furthermore, this issue would present even more difficulty if we were to scale the size of our project upwards. While it was possible to check sort through the inconsistencies in a data frame of 1,300 rows, it would not be if we decided to apply the same procedure for Billboard's weekly charts from 1958 to 2020, for instance. Future projects should seek to remedy this issue by automatically accounting for covers, remixes, and alternate versions of songs.

Analyses

Feature Engineering

Numeric Features

We believe that lyrics can be just as influential to a song's popularity as its auditory features, and we wanted to capture certain elements of what makes a song stand out lyrically. We arrived at two features that we will be using to represent the lyrical content of Billboard Hot 100 Year-End hits: lyric valence and lyric repetitiveness.

Lyric Valence

In recent years, Spotify has rolled out a category of playlists that correspond to a specific moods, featuring emotions as varied as "my life is a movie" to "life sucks." However, in allocating certain songs to mood-based playlists, how does Spotify decide which songs are sad, and which are happy? It may be that they are leveraging the *song valence* feature, which indexes the musical valence of tracks. However, we believe that the emotional content of the lyrics may be in part responsible for how we designate emotion labels in reference to musical pieces. To compute lyric valence, we entered processed lyrics into VADER, a parsimonious, rule-based sentiment analysis tool. This allows us to represent the strength and direction of the emotional content in the lyrics of a song on a standardized scale between -1 and 1, with -1 being strongly negative and 1 being strongly positive. A sentiment score of zero represents a song whose lyrics are not inclined in either emotional direction.

Lyric Repetitiveness

In his parody song “Repeat Stuff,” comedian Bo Burnham satirizes the tendency for popular songs to have excessive repetition in their lyrics. Anecdotally, this trend can be observed in Top 100 singles across the years. The song “Baby” by Justin Bieber contains the word “baby” 56 separate times, constituting 13.2% of the lyrical content of the song. “Turn Down for What” by DJ Snake & Lil Jon is entirely composed of two phrases: “turn down for what” repeated 15 times, and “fire up that loud, another round of shots” repeated 10 times. Both songs scored a spot on the prestigious Billboard year-end lists, so perhaps it is not too far-fetched to propose the repetitiveness of lyrics, especially as a repetitive lyrical structure can be indicative of how catchy - and therefore how replayable - a song is (Modrzejewski, Szachewicz, & Rokita, 2020). To compute the lyrical repetitiveness of a song, we take number of unique words and divide them by the number of total words and subtract it from one. For reference, “Baby” has a repetitiveness score of 0.68, while “Turn Down for What” has a repetitiveness score of 0.91. Like all numeric variables, we scale our metric for lyric repetitiveness before analyses, but even in its unscaled form, it is a powerful tool that allows us to determine a song’s reliance on repetition over unique verses.

Word (Lyric) Features

TF-IDF

We decided to use term-frequency, inverse document frequency for our word engineered feature. TF-IDF was calculated by multiplying the frequency of words in the lyrics against the inverse frequency across a set of documents. We used the package text2vec, which creates a smooth idf that is defined as

$$idf = \frac{\log(1+(\#\ documents))}{(\#\ documents\ containing\ term)}$$

A log-scaled term frequency to calculate TF.

The following figure shows the top 10 tf-idf features (sorted by the tf-idf value) for the dataset.

word	n	tf	idf	tf_idf	ranking_3
<chr>	<int>	<dbl>	<dbl>	<dbl>	<fct>
doo	162	0.659	6.11	4.02	3
tas	40	0.526	7.21	3.79	1
thunder	73	0.299	4.90	1.47	3
low	175	0.449	2.64	1.19	3
sail	25	0.187	6.11	1.14	2
na	144	0.257	4.26	1.09	3
animals	2	0.2	5.26	1.05	3
rack	84	0.178	5.82	1.04	2
ey	14	0.184	5.60	1.03	1
rompe	54	0.141	7.21	1.01	3

Figure 7. Top ten tf-idf features sorted by highest tf-idf values.

Most of these top words don't make sense in general usage, however, have a high tf-idf and hence could be used to distinguish between songs of different rankings.

Bag Of Words

The bag of words model takes into account the raw frequency of each word only. The following figure shows the frequency of the top 20 words extracted from all songs:

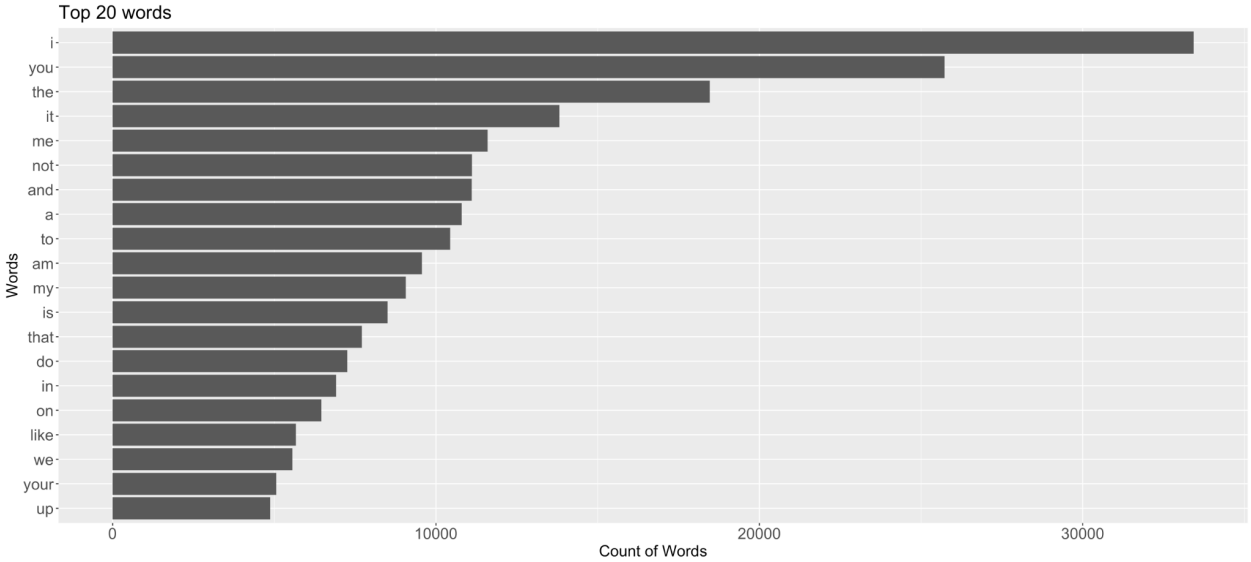


Figure 8. Histogram of word counts for the most popular words in song lyrics.

As seen from the above figure, most of these words occur across most of the songs and hence would not help with classification. Hence, we chose the TF-IDF model over this one. TF-IDF model weights words that occur uniquely across songs, which takes care of the fact that we haven't removed the stop words from songs, as explained before.

Train-Test Split

In order to evaluate our models, we split our data into train and test sets. We did a 20% test split. Since each ranking has a different number of data points, we used stratified splitting on it. We implemented this by splitting the data of each ranking separately by 20%, such that the train and test set contain approximately the same percentage of samples of each ranking as the complete set.

Ranking	# Songs in Complete Set	# Songs in Train Set	# Songs in Test Set
Top Ten (Rank 1)	115	92	23
11 - 50 (Rank 2)	511	409	102
51 - 100 (Rank 3)	721	577	144

Table 3. Number of data points by ranking within train-test split.

Numeric Features

All numeric features were scaled to account for the variation in ranges and differences in units across measures.

Regression

We considered regressing against all features from spotify and the two features engineered by us (song_lyrics_valence and song_lyrics_repetitiveness). The following figure summarizes our results.

```
Residuals:
    Min      1Q  Median      3Q     Max
-2.3668 -0.4251  0.2932  0.5375  1.1262

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                2.449889   0.017148 142.871  < 2e-16 ***
song_danceability         -0.059723   0.020063  -2.977  0.00297 **
song_energy                0.017996   0.029160   0.617  0.53725
song_loudness              0.031776   0.026988   1.177  0.23924
song_speechiness           0.012530   0.018065   0.694  0.48805
song_acousticness          0.014898   0.020298   0.734  0.46310
song_instrumentalness     -0.059496   0.019875  -2.994  0.00281 **
song_liveness             -0.022896   0.017529  -1.306  0.19172
song_valence              -0.048714   0.020681  -2.355  0.01864 *
song_tempo                -0.006644   0.017730  -0.375  0.70790
song_popularity           -0.136950   0.019053  -7.188 1.09e-12 ***
song_duration             -0.035700   0.017640  -2.024  0.04318 *
song_lyric_valence         0.003168   0.017759   0.178  0.85844
song_lyrics_repetitiveness -0.036204   0.017604  -2.057  0.03992 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6293 on 1333 degrees of freedom
Multiple R-squared:  0.06284,   Adjusted R-squared:  0.0537
F-statistic: 6.876 on 13 and 1333 DF,  p-value: 5.521e-13
```

Table 4. Regression results of all lyric and acoustic features as predictors.

A few significant features can be seen from the above table, such as song popularity, song danceability and song instrumentalness. However, the R^2 is very small (0.063), which shows that there isn't a significant relationship between the features and ranking.

We used the train and test split to predict on regression as well. We achieved an accuracy of 38%, which is slightly more than a random prediction. The confusion matrix of the prediction is shown.

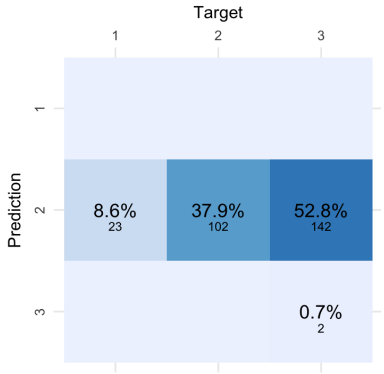


Figure 9. Confusion matrix of predictions for linear regression on numeric features

As can be seen from the above figure, almost all of the predictions are in rank-2 category and lm isn't able to predict rank-1 or rank-3 at all. Hence, regression analysis wasn't sufficient in explaining rankings of the songs.

K-Means

K-means clustering is a method that allows us to group together data points, based on a distance metric (euclidean in this case). The algorithm can then be used to assign each element to the closest cluster centroid from a total of k clusters. The cluster centroids are initially randomly assigned and are then calculated based on the means generated over multiple iterations.

We used K-Means to analyze the numeric features.

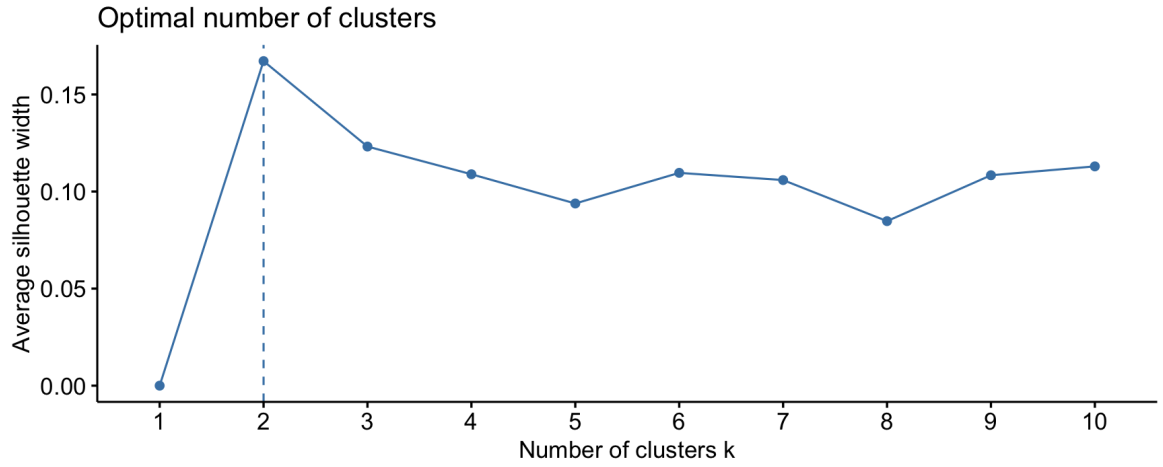


Figure 10. Silhouette plot of optimal k for k-means analysis, with a peak at k = 2.

The optimal number of clusters is 2. This is way too small and doesn't distribute the data properly.

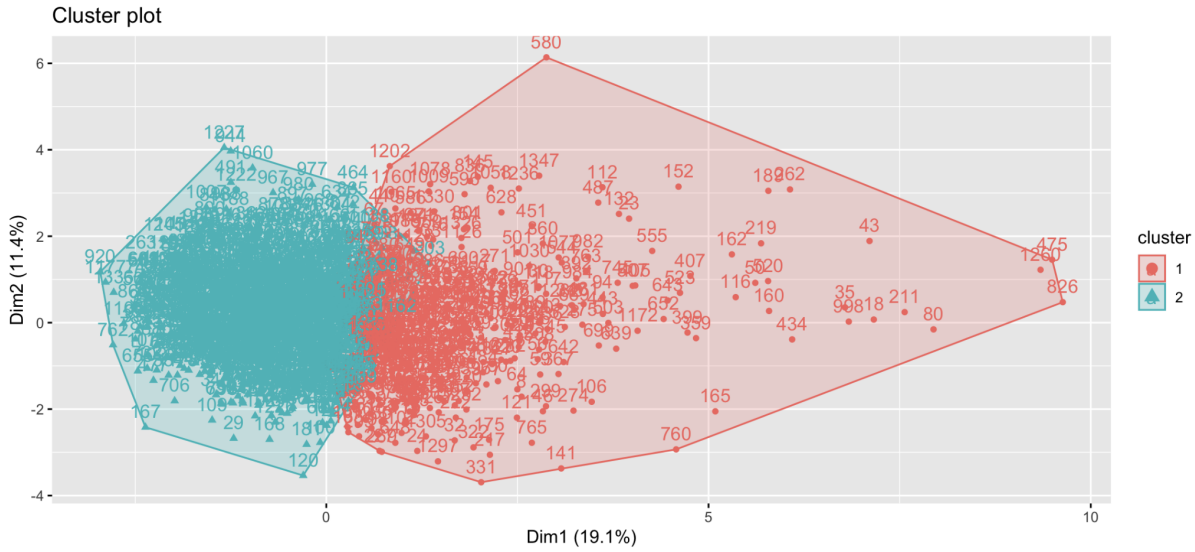


Figure 11. Cluster plot for k=2, with the area covered by each group denoted by a unique color.

Although k-means clusters the data quite well into two distinct clusters, this wasn't much help to us as the number of data points in each cluster is roughly the same, and hence, further analysis was required to determine the ranking of the songs.

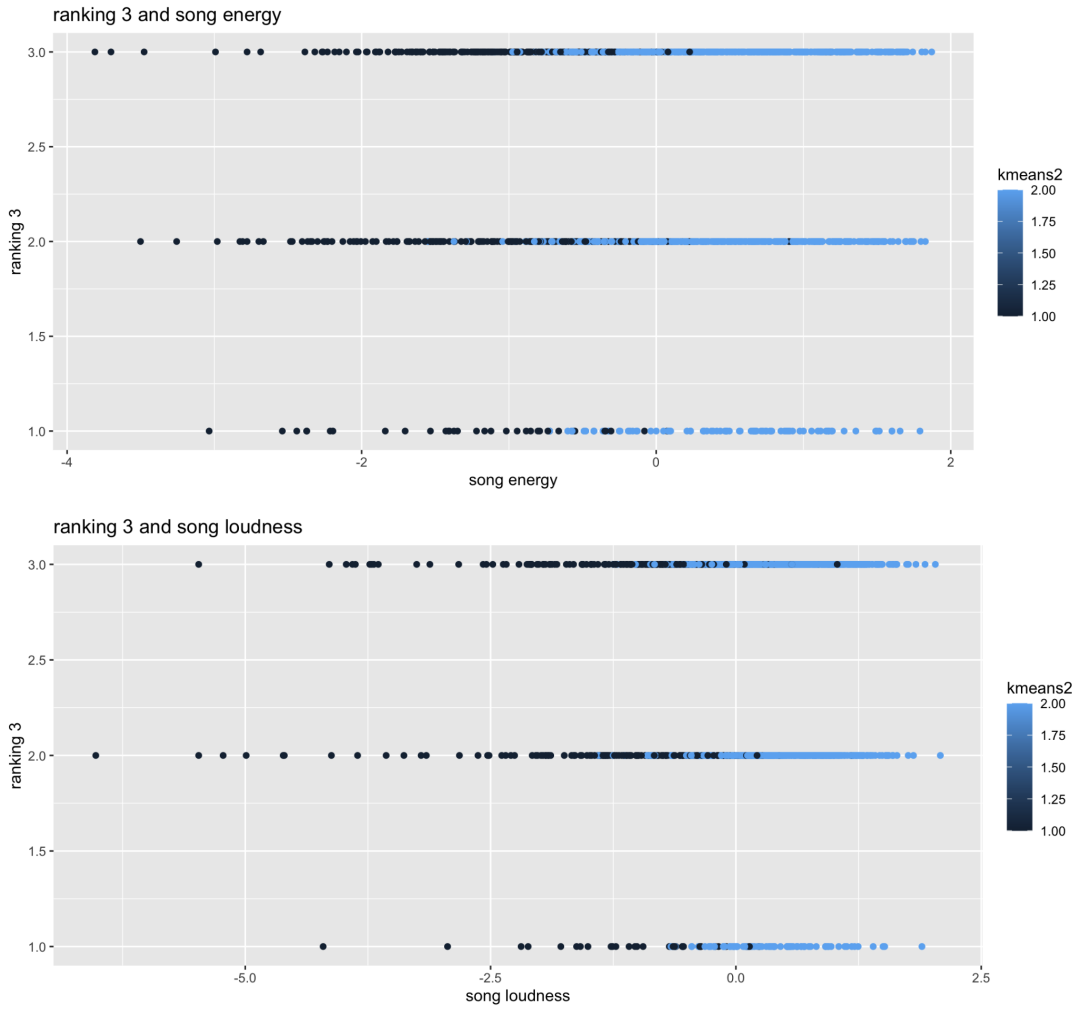


Figure 11. Plot of song acoustic features against ranking groups, displaying clear clustering when data is sorted by song loudness and song energy.

Interestingly, we saw that the song energy and song loudness data points, when coloured based on the cluster they belong to, and plotted against the ranking, show a clear divide between the two clusters. Hence, we believe that these two features could be contributing majorly in the clustering of the data. However, their distinction doesn't act across the ranking and thus it doesn't help us with the task.

We did not try to evaluate this method as we would have had to use other methods, alongside kmeans, to get a rank of the song (as it is an unsupervised method

and doesn't yield rank classes in itself). Since k-means itself didn't yield sufficient results, we did not see a need to implement this method.

DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) is an algorithm which groups together a given set of points with its neighbours, marking the outliers that lie alone in low-density regions. It is different from K-Means clustering as it allows for flexibility of cluster shapes and is also not dictated by the number of clusters. Instead it takes in the parameters: *epsilon*, which is the maximum distance between two data points to be classified as a neighbour and *minPoints*, which is the minimum number of points required to count a group as a cluster.

We ran DBSCAN on the numerical features.

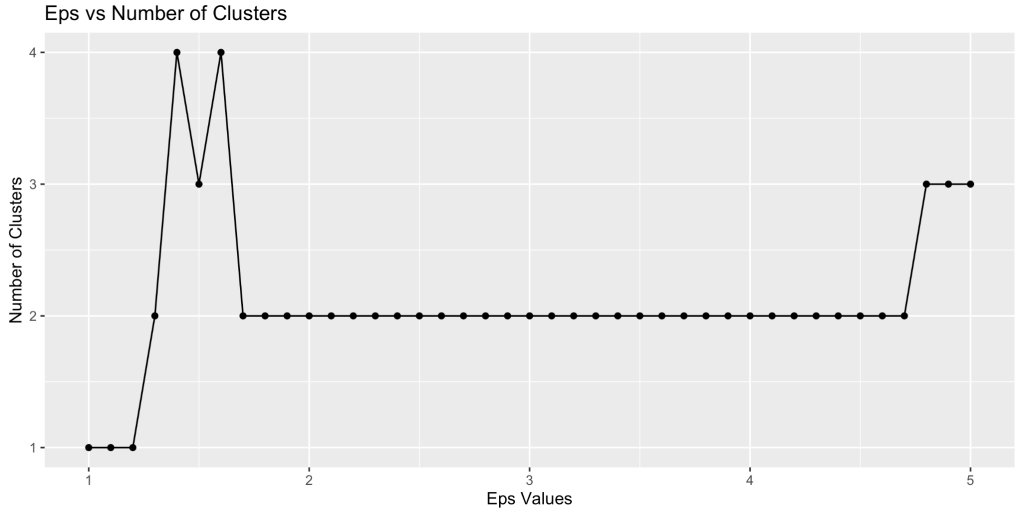


Figure 12. Epsilon values against the number of clusters to determine to determine the optimal number of clusters for DBSCAN.

The above figure shows the total number of clusters for different epsilon values. The maximum number of clusters obtained is 4. However, in each case, one cluster dominates as it contains the most number of data points. This shows that most of the data points are clustered together in one of the clusters, with a few outliers that might get clustered depending on the epsilon value.

We did not try to evaluate this method as we would have had to use other methods, alongside dbscan, to get a rank of the song (as it is an unsupervised method and doesn't yield rank classes in itself). Since dbscan itself didn't yield sufficient results, we did not see a need to implement this measure.

Random Forest

Random forest is an ensemble learning method constructs multiple decision trees and outputs the rank of the song that is the mode of the ranks of the individual trees. We used Random forest to classify the ranking of the songs.

	%IncMSE	IncNodePurity
song_danceability	11.59	46.81
song_energy	6.27	37.54
song_loudness	2.97	35.67
song_speechiness	4.41	39.60
song_acousticness	3.14	41.90
song_instrumentalness	2.23	17.41
song_liveness	1.69	38.03
song_valence	7.54	40.91
song_tempo	-0.63	36.97
song_popularity	35.95	76.08
song_duration	3.08	41.09
song_lyric_valence	1.47	34.37
song_lyrics_repetitiveness	3.63	41.01

Table 5. Acoustic and lyrical features as analyzed by random forest.

The above figure shows the features analysed by random forest. %IncMSE refers to the increase in mse of predictions as a result of random permutations of the value. The higher this value, the better the feature is at predicting. Song popularity and song danceability shows a high %IncMSE value showing that they are significant in predicting ranks.

We ran random forest on the train test split and got an accuracy of 37% on the test data, which is less than regression. The confusion matrix can be found in Figure 13. As seen in the visualization, the random forest algorithm is only able to predict rank-2 songs but not rank-3 or rank-1. Thus, we didn't use random forest as a predictor.

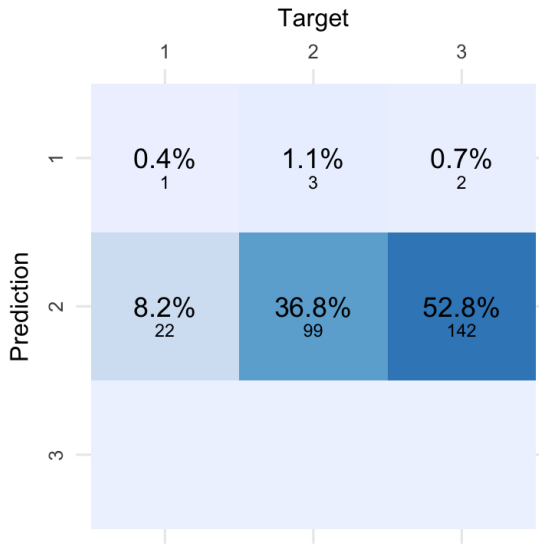


Figure 13. Confusion matrix for random forest on test-train split.

Word Features

We used the TF-IDF vectorizer to transform song lyrics to TF-IDF vectors. We did not remove stop words, for reasons explained before. We used these feature vectors to analyse the ranking of the songs, with different models.

K-Means

We ran K-Means on the TF-IDF feature vectors.

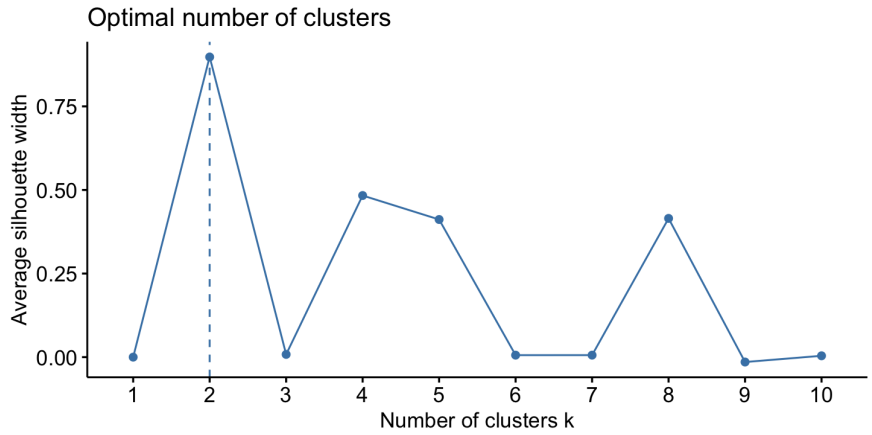


Figure 14. Silhouette analysis for the best number of clusters for TF-IDF feature vectors.

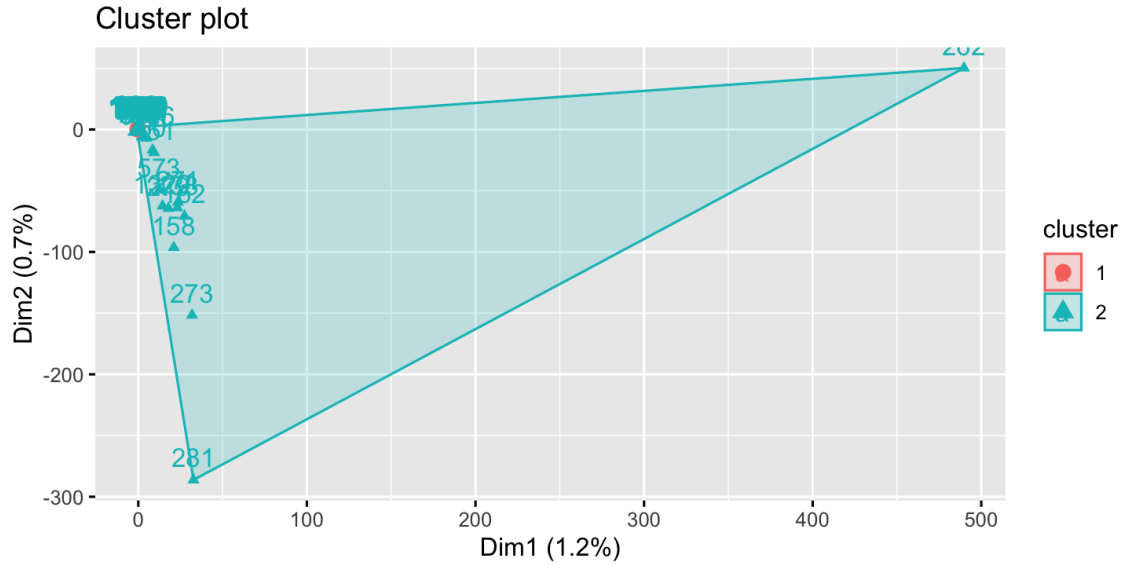


Figure 15: Representation of clusters using k = 2.

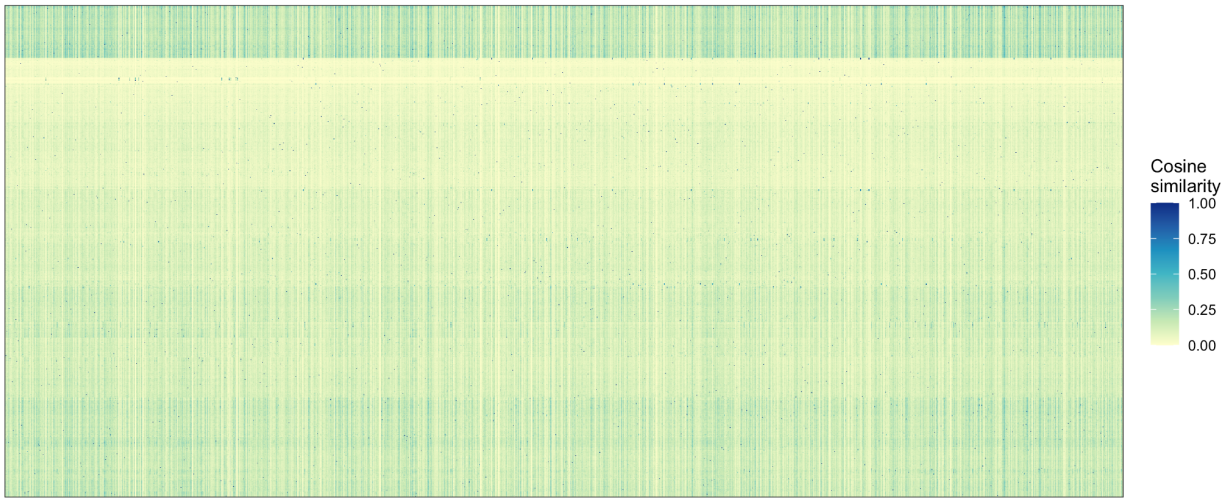
Because the clustering is highly imbalanced, it does not help us understand or predict the ranking of the song.

Cosine Similarity

Cosine similarity is a metric that indicates how similar two documents are to each other. Unlike Euclidean distance, cosine similarity does not measure the exact distance between two vectors in a multidimensional space, but instead captures the orientation between them. This adjusts for the sensitivity that measures of Euclidean distances have for the length disparities between the two documents selected for comparison. Because

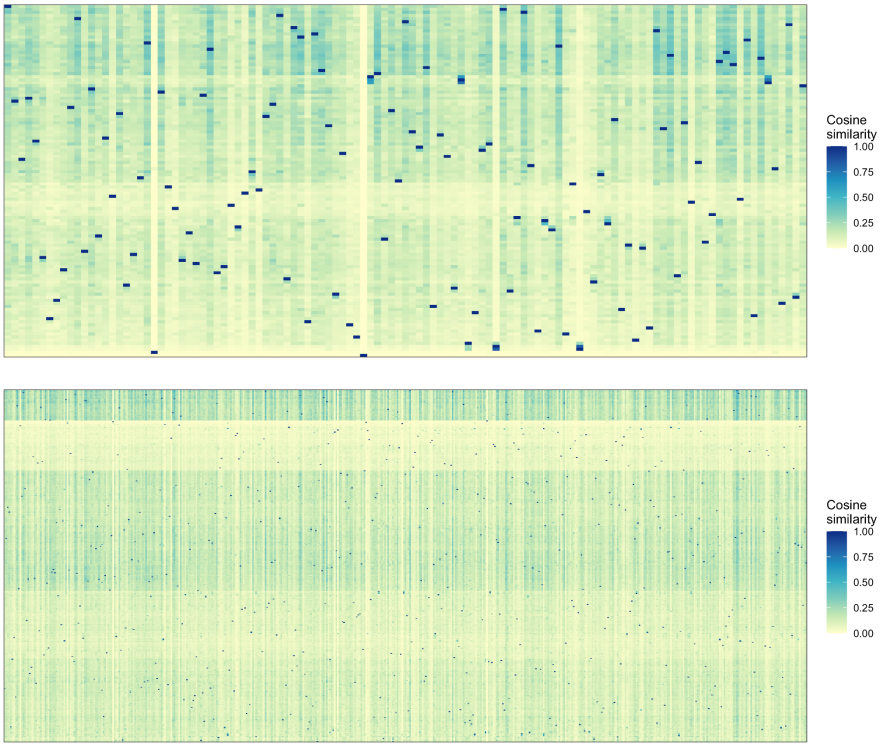
songs often differ in length and the number of words used across lyrics, we believe cosine similarity to be the measure of similarity that is most suited to our purposes.

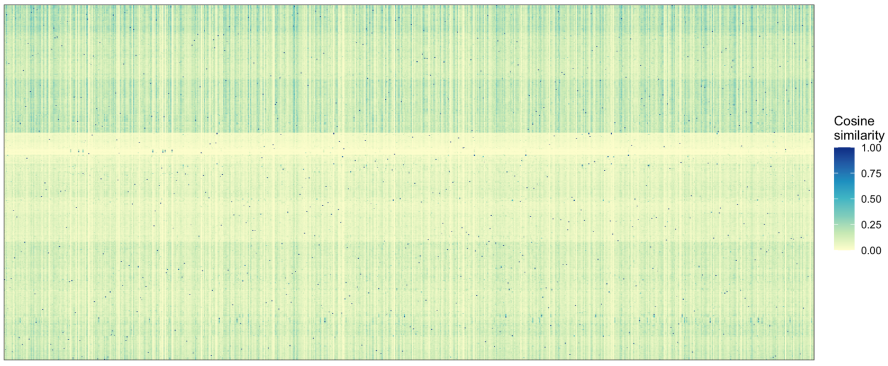
We ran cosine similarity on the TF-IDF features for all song lyrics.



The above figure shows the cosine similarity between all data. Each row is a song lyric and each column shows the cosine similarity with another song's lyrics. Shades of blue scattered in the figure shows that a lot of the songs have a high cosine similarity with other songs. However, there is no clear pattern seen.

Examining cosine similarity for each of the rankings:





The above figure shows the cosine similarity between songs of the same rankings. All figures have blue patches scattered across them, which shows that there are songs with high cosine similarity with other songs. Hence, we decided to use this method to predict the rankings of songs.

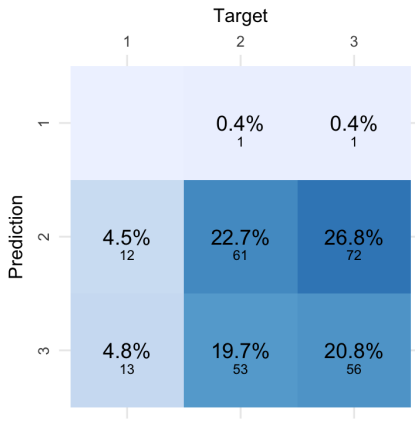
Algorithm Pipeline

We followed the following steps to identify the ranking of a song.

1. Calculate cosine similarity between the lyrics of the test song and the lyrics of all songs in training data.
2. Sort the vector and find the top 5 songs with the highest cosine similarity.
3. Use majority voting on the ranking of these songs to assign a rank to the test song.

Evaluation

We evaluated our algorithm on the test data and got an accuracy of 43.5%.



As seen in the confusion matrix, the algorithm was able to predict 22.7% of the songs in the second ranking category, and 20.8% of the songs in the third ranking category. It failed to predict any songs with rank 1. To understand the results better, we also calculated the precision, recall and f1 scores for each class using the confusion matrix.

Metric	Rank-1	Rank-2	Rank-3
Precision	0%	42.06%	45.9%
Recall	0%	53.0%	43.4%
F1	-	46.9%	44.6%

The precision for Rank-3 is the highest, which means that the algorithm correctly predicted 46% of Rank-3 songs, out of the total predicted Rank-3 songs. The recall for Rank-2 is the highest, which means that the algorithm correctly predicted 53% of actual rank-2 songs in the dataset.

Interpretation of the results shows that the algorithm isn't able to predict rank-1 songs since they are unique in their lyrics. It is best able to predict Rank-2 songs, as shown by the high recall, showing that a lot of the songs in the rank-2 category have TF-IDF feature vectors close to each other, which could mean that their lyrics have similar words. Similarly for rank-3, a lot of the song lyrics have similar words, but less so than rank-2 songs.

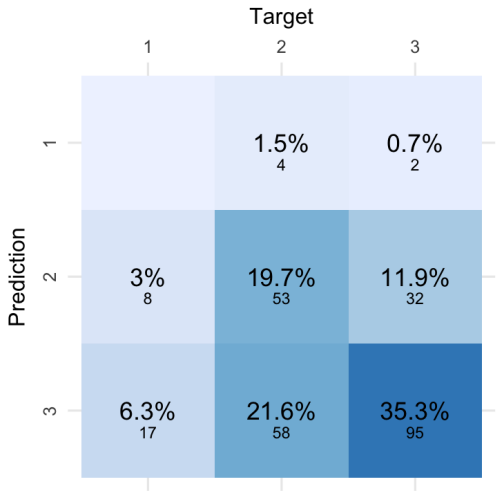
In order to improve our results, we decided to take this process a step further by combining the numeric and word features together and then analysing them with cosine similarity.

Numeric + Word Features

The scaled numeric features were combined with the TF-IDF features to produce new combined features. The process of evaluation with cosine similarity was identical to the procedure employed in the section above.

Evaluation

We evaluated the new features using cosine similarity on the test data and got an increased accuracy of 55%! Analysing the confusion matrix:



As seen from the figure above, there is a sharp increase in the predictions for rank-3 songs. 35% of the rank-3 songs were predicted correctly, which is more than 20.8% from the previous method, while only 19.7% of the rank-2 songs were predicted correctly, a decline from 22.7%. None of the rank-1 songs were predicted in this model as well. The overall accuracy rose, however, and hence this model is better than the previous one.

To understand the results better, we also calculated the precision, recall and f1 scores for each class using the confusion matrix.

Metric	Rank-1	Rank-2	Rank-3
Precision	0%	57.0%	55.9%
Recall	0%	46.1%	73.6%
F1	-	51.0%	63.5%

The precision of the rank-2 and rank-3 songs rose quite a lot, and is almost the same for both categories. The recall for rank-3 songs rose sharply, suggesting that this model is better at predicting rank-3 songs. The algorithm managed to predict 73% of the actual rank-3 songs, which we thought was an amazing achievement. The recall for rank-2 songs declined slightly from the previous model.

Interpreting the Results

This algorithm considers spotify and lyric features on top of the song TF-IDF features. The spotify and lyric features, such as song and lyric valence, song repetitiveness, liveliness, acoustic ness give us a lot of musical information about the songs. Hence, this algorithm helps us understand the music and rankings better than the previous one.

Interpretation of the results shows that the algorithm isn't able to predict rank-1 songs in this case as well since they are unique in their lyrics and in the features they employ. This shows that the top ranked songs might contain something different, something that distinguishes them from the rest of the music produced in the same time frame.

Rank-3 songs have a high recall, showing that if song lyric features and spotify features are both taken into account, then songs in the category share a lot of similar features. This helps us understand the nature of the songs as these share similar worded lyrics along with similar musical characteristics. The same could be said of rank-2 songs, however, their recall is much lower than rank-3, suggesting that they have a few factors that distinguish them from other songs produced in the same time frame

Conclusion

A variety of methods were applied to shed light on what differentiates Top Ten artists from others. For ambitious artists looking to make it big, this report may seem daunting; after all, presented with this many features and analyses, what should they strive to keep in mind the next time they go into the studio? What our project has shown is that there are certain features that - when suitable methods are applied - distinguish songs that rank higher than songs that place in less desirable positions along the Year-end chart. For instance, using the lyrical content of songs to compute cosine similarity allowed us to predict, with moderate accuracy, where a song placed in the Year-end charts. Audio features such as energy were spotlighted as important predictors of popularity across several methods, including random forest and k-means clustering. This project does not mean to propose definitive ways someone can craft a Top Ten Single; it merely suggests lyrical and audio features that are commonly found within tracks that have achieved a great deal of popularity. There are many ways in which aspiring artists can take advantage of our results. They may keep in mind that word-related features greatly informed our algorithms, and remember not to neglect the lyrical content of their songs. They may

Further Work

We identified a couple of areas where our study is lacking and could be strengthened. These could potentially be the next steps for the project

1. We can strengthen our predictions if we billboard rankings from previous years, or songs with much lower ranking (below 100). This would help us analyze the quality of songs further.

2. We can try other models to classify the data. This could help strengthen our predictions.
3. While going through the lyrics of the songs, we identified some words were elongated, such as “youuuuu” or “whoooo”. We could potentially reduce these words to their root form, thus improving our predictions.
4. Song lyrics often use colloquialisms. We can make a custom list of such words and replace them to enhance our model’s performance.
5. We didn’t try stemming and lamenting the words, as we prioritized other methods. These two pre-processing techniques could also help us achieve better results.

Critique of different group (Bernadette &

References

1. Pietroluongo, Silvio. "How We Chart The Year", Billboard.com, December 18, 2012
2. Modrzejewski, M., Szachewicz, J., & Rokita, P. (2020, October). Application of Neural Networks and Graphical Representations for Musical Genre Classification. In International Conference on Artificial Intelligence and Soft Computing (pp. 193-202). Springer, Cham.
3. “DBSCAN.” Wikipedia, 13 Feb. 2021. Wikipedia, https://en.wikipedia.org/w/index.php?title=DBSCAN&oldid=1006556084.
4. Features | Spotify for Developers. https://developer.spotify.com/discover/. Accessed 19 Apr. 2021.
5. Get Audio Features for a Track | Spotify for Developers. https://developer.spotify.com/console/get-audio-features-track/. Accessed 19 Apr. 2021.
6. Hot 100 Songs - Year-End | Billboard. https://www.billboard.com/charts/year-end/hot-100-songs. Accessed 19 Apr. 2021.
7. Kuhn, Max. The Caret Package. topepo.github.io, https://topepo.github.io/caret/. Accessed 19 Apr. 2021.
8. Plot_cosine_heatmap: Plot Cosine Similarity Heatmap in MutationalPatterns: Comprehensive Genome-Wide Analysis of Mutational Processes. https://rdrr.io/bioc/MutationalPatterns/man/plot_cosine_heatmap.html. Accessed 19 Apr. 2021.
9. “R - In a Random Forest, Is Larger %IncMSE Better or Worse?” Cross Validated, https://stats.stackexchange.com/questions/162465/in-a-random-forest-is-larger-incmse-better-or-worse. Accessed 19 Apr. 2021.
10. “R - Measures of Variable Importance in Random Forests.” Cross Validated, https://stats.stackexchange.com/questions/12605/measures-of-variable-importance-in-random-forests. Accessed 19 Apr. 2021.
11. “R Convert Matrix or Data Frame to SparseMatrix.” Stack Overflow, https://stackoverflow.com/questions/10555210/r-convert-matrix-or-data-frame-to-sparsematrix. Accessed 19 Apr. 2021.
12. “Random Forest.” Wikipedia, 17 Apr. 2021. Wikipedia, https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=1018242574.

13. RandomForest Package - RDocumentation. https://www.rdocumentation.org/packages/randomForest/versions/4.6-14. Accessed 19 Apr. 2021.
14. Replace_word_elongation: Replace Word Elongations in Textclean: Text Cleaning Tools. https://rdrr.io/cran/textclean/man/replace_word_elongation.html. Accessed 19 Apr. 2021.
15. Shung, Koo Ping. "Accuracy, Precision, Recall or F1?" Medium, 10 Apr. 2020, https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

Comments for the projects from Bernadette Paulyn Gostelow and Jiaxin Li:

At the beginning, the R squared score is quite low for the linear regression and there might be high multicollinearity within the variables or there is no linear relationship between dependent variables and independent variable.

For the data they have, the number of records that are classified as 1 is significantly less than the number of records that are classified as 2 or 3. For their further algorithm of using cosine similarity to predict rankings for a given records, all the predicted results are 2 and 3.Under this circumstance, it might be better to make this dataset balanced at first or to penalize classification error differently,

Project 3: Final Project

Overview

The final project in this class is an open-ended **data mining** project. Key deliverables are

- A **written report**, in a PDF format. There is no page limit but you must address all of the requirements below and produce a cohesive report.
- A critique written about **another group's** final project, your final project can be critiqued by multiple teams but there is no requirement that your project must be summarized.
- Your code on Github.
- You must have a partner for the final project.

Written report

- ~~The project must perform a form of data mining:~~
    - ~~Generate a non-obvious insights from the data that can be leads for further investigations (similar to the observation step in the scientific method). The most common way to do this is by merging datasets from different sources.~~
    - ~~A metric or algorithm that quickly filters or sorts large amounts of data for people.~~
    - ~~You should avoid projects where the end goal is a predictive tasks (too easy) or inferential statistics (too hard).~~
    - ~~Please talk to me if you're not sure that your project would be considered data mining or not.~~
- You must identify an audience and articulate the potential value for that audience in the report. Mention again in conclusion. - MOTIVATION

    - Mining implies value is discovered and so you must articulate what value is generated for what audience in your project. Your audience can be an individual or an institution.
        - You should have some level of detail like “recent college grad” or “think tank for public policy on housing” rather than a generic person/institution.
    - Value can be realized in the form of saved time, decreased uncertainty, or increased reward. I’m open to hearing other forms of value definition.
- ~~You should use at least 2 datasets or a single large dataset for your project.~~
    - ~~If you have two datasets you should articulate how the datasets complement each other, for example~~
        - ~~One dataset could provide complementing features (e.g. one dataset may have the activities and another may have the user demographic data).~~
        - ~~One dataset could provide more resolution (e.g. detailed surveys tend to be done less frequently so they’re often complemented with surveys that are less detailed but collected more frequently).~~
        - ~~One dataset could provide more data from a different population (e.g. a system change has made historical patent data to be stored in a different system).~~
        - ~~You should not call a dataset collected from a different time point but from the same source as a different dataset.~~
    - ~~For single datasets that are truly large or complex, you should articulate why it is considered large/complex relative to the standards your audience is used to.~~
    - ~~You must perform some sort of exploratory data analysis that checks the completion and/or quality of the data.~~
    - ~~You must identify the source of the data and summarize the datasets.~~
        - ~~Ideally, the data source should come from the entity that manages the data. Management is defined by the entity responsible if the data has quality issues. For example, data aggregators like Kaggle do not manage the data it posts on its challenges but Twitter manages Twitter data. Some exceptions exist like platforms like NYC OpenData allow you to report data quality issues and will be considered a manager of the data.~~
- You must use an algorithm to explore the relationship between different features in the data similar to Project 1.
    - ~~Please have at least one graphical summary that highlights how your algorithm is working (e.g. the correlation plot from HW1, the loadings plot from PCA, etc).~~

    - ~~You must quantitatively evaluate how your algorithm fits to the data. For example, correlations are meaningful if they're close to -1 or 1, clustering and supervised learning methods all have different metrics associated with them.~~
    - ~~Please engineer at least one feature and evaluate its usefulness in the context of your project.~~
- Please verify the results from your mining whether is due to chance or likely a real pattern.
    - For example, you may have to subset the data to see if the result is due to an outlier. You can also quote external sources to help understand the data.
    - Please make sure you quantitatively address the question "if you had a different dataset, how robust are the results from the algorithm?"
        - Please be sure to articulate how high uncertainty may affect your conclusions (even if your results are robust).
    - Please note that the algorithm above should be a quick filtering where the second step is a close examination of the result.
- Introduction and Conclusion (5 pts)
    - Please talk about any "iterations" that you had to perform from the start of the project to the end. Please have at least one paragraph that discusses the potential for data-snooping for your project.
    - Your introduction and conclusion should be written at the level so a peer outside of this class can understand your intent and findings.

Abstract requirements

To complete this portion, I recommend you talk to each other before the full project is finished.

- What was the initial motivation for tackling the project?
- What datasets were used?
- What aspect of the project is considered a **data-mining** and what is discovered?
- Is there anything you would have done differently? For example
    - Used different datasets
    - Used different models to explore the data
    - Generated a different feature
    - Arrived at a different conclusion from the given data. (These are simply suggestions, you team only has to come up with one thing)

Github Code

- You should have a README page
    - A at most 5 sentence summary of the project
    - An general explanation of the different files/folders if someone were to replicate your study.
- No data should be on Github but your README should explain how the data can be obtained.
- Your written report and code should both exist on Github.
    - You do not have to check in the PDF version of your report on Github but your Rmarkdown or LaTeX file should exist on Github.
    - If you use Word as a text editor. Make sure you have a link from your README to this document.