Song Data and Lyrics Analysis and Visualization

By Eliot Shektman (ess239), Sophia Oguri (tso24), Keethu Ramalingam (kr439)

Github link: https://github.com/keethu-ram/song-lyrics

Introduction and Overview of Data

“Is that song by ___? It sounds like them.” “I don’t like ___, it all sounds the same to me.” “Oh this song is going to be a hit.” We often hear these comments about the music we listen to, as we unknowingly create correlations about certain artists, genres, and types of popular songs in our heads. But, are these correlations actually rooted in something quantifiable? Are we able to make predictions based on these patterns and correlations? As avid music lovers, we wanted to learn more about the possible patterns in the songs we listen to. We chose to analyze a Spotify top tracks dataset and took a look at various features associated with the top 50 tracks of each year from 2010 to 2019, which included values for beats per minute, popularity, loudness, etc. This dataset also included standard information such as artist, year, and genre. We were also interested in the actual contents of songs, and thus also found a Billboard top songs dataset that had lyrics of over 5000 popular songs. We combined these datasets by matching on the titles and artist with a Python script (in the linked Github repo). We used Python Pandas, collections, NLTK, SKLearn, BeautifulSoup, and Matplotlib libraries as well as Tableau.

Question 1: *Can Artists Be Identified By Their Lyrics?*

We sought to find out whether song lyrics could be used to classify songs as belonging to a certain artist. Firstly, we considered the average **number of unique lyrics per song.** When we plotted the results for top artists across several different genres, it was found that there were clear patterns based on the artists’ genre, with other trends among individual artists. Pop artists (Rihanna, Katy Perry, and Maroon 5) had the lowest counts of average lyrics, possibly due to the measured repetitiveness feature of many pop songs. Contrastingly, rappers (Iggy Azalea, Eminem, and Nicki Minaj) had the highest counts as their genre is typically characterized by its rapid speed. EDM (Calvin Harris) and Hip Hop artists (Taio Cruz and Black Eyed Peas) had counts in between those of rappers and pop singers.

Pitbull Word Cloud

Rihanna Word Cloud

We wanted to delve deeper by also analyzing the actual lyrics that comprised each artists’ songs. In particular,

we were interested in the most common words that appeared in each artists' songs. To do this, we aggregated the lyrics from all of the songs from an artist and removed standard stopwords. To display results, we visualized word clouds for two artists, Pitbull and Rihanna. We note distinctive feature words for each singer. Pitbull, whose songs are typically upbeat and classified as dance/ party pop, features the words "tonight", "time", and "baby". The word "international" is also featured on his word cloud, notable due to his reference of himself as Mr. Worldwide, suggesting that the theme makes its way into his songs. Rihanna, an R&B pop singer has many vocable fillers such as "ooh" and "na", and also words such as "love", "hard", and "diamond", which are also noted to be subjects/ themes of/in many of her songs.

From our initial exploratory data analysis and visualizations, we hoped to find out if lyrics in songs by different artists are distinctive enough to be able to be identified as their own. In order to do so, we utilized **model selection**, along with analysis of **confusion matrices** to select the best classifier to produce the results that form the basis of the answer to our question.

To preprocess the data for analysis, since our data is textual, we used the **bag-of-words** representation model, which describes the (frequency of) occurrence of words within a certain document (in our case, a song), essentially considering each word count as a feature. As it is good practice to train a model on a maximum of N-1 features (where N is the sample size), we wanted to isolate the most significant words. To do so, we used **TF-IDF statistics**. TF refers to term frequency and IDF, or Inverse Document Frequency, refers to a dictionary of words as keys and a list of documents, or in this case, which artists' songs, in which that word appears in as values) The purpose of TF-IDF is to reflect how important a certain word is to a document based on term frequency, and thus indicate words would be most useful to us (and are thus likely to make up artists' most common lyrics word clouds (shown above). We chose the words with the top 1000 TF-IDF scores as our feature words. Below is a sampling of some of these:-

```
print(my_terms)
```

```
'true', 'free', 'came', 'damn', 'close', 'dream', 'lyrics', 'sun', 'knew', 'beautiful', 'pain', 'goes', 'stand', 'low',
```

```
 gotta = True            Drake
  mind = True           Shawn
friends = True          Luke B
things = True              NF
  game = True           Kevin
  long = True           KIDS S
    oh = True           KIDS S
 heart = True           Charli
  talk = True              NF
 wrong = True              NF
  head = True              NF
   run = True          Lil Ba
 pussy = True          Lil Pu
broken = True          Keith
alright = True         Foster
 drive = True          Lil Pu
  gone = True          Russel
   ice = True            Zayn
  work = True          Blac Y
 brain = True          YBN Na
bought = True          Lil Pu
 party = True          Walker
 mouth = True          Ariana
```

(Most Important Features. Note: Artist names were cut-off and were thus presented to us as shown in the above screenshot)

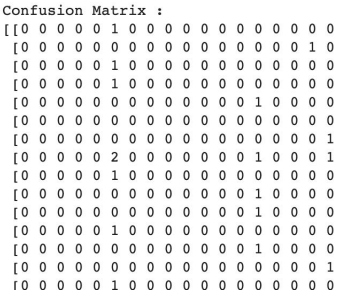
To assess the usefulness of the TF-IDF representation, we first used an NLTK Naive Bayes (NB) classifier to train and test data with features set as the 1000 most common words that were not stop words, only to obtain a 35.9% **AUC score** (indicating accuracy as it represents the area under the **ROC curve**) against the training set and a 2.2% AUC score against the test set. When we used a Bernoulli NB classifier, we obtained 64.1% accuracy against the test set and 0.0% accuracy against the test set.

Then, using the TF-IDF representation, scores improved, but not by as much as we had expected; with the NLTK NB model, a 46.6% training AUC score and a 24.2% test AUC score and with the Bernoulli NB model, 79.4% training AUC score and a 13.1% test AUC score. We then realized that this was likely due to the large number of artists that acted as our

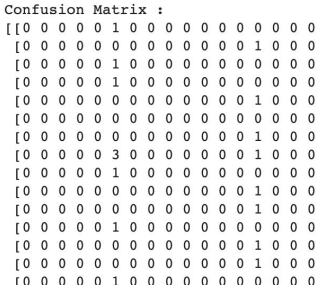
labels, or classes. To be precise, we had 1897 classes. To boost accuracy, we decided to limit the number of artists to predict to 20. This in fact worked, as our accuracy against the training set was boosted to 77.5% and the accuracy of the test was 69.1%. Out of curiosity, we decided to check out the most telling lyrics that associated a song with an artist with NLTK's `show_most_informative_features()` method. We found that many of Drake's songs were characterized by the use of the informal contraction of "got of", "gotta" , that NF, a rapper known for his emotional style, had words such as "talk", "wrong", and "head" associated with his music, and that country singers, Luke Bryans and Keith Urban seemed to write songs that featured themes of "friends" and being "broken", respectively (or at least the words themselves). We wanted to see if we could get better results still, and thus additionally used K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) classifiers. The results are recorded below:-

	NLTK NB	Bernoulli NB	KNN	SVM
Training	77.5%	79.5%	81.2%	75.6%
Test	49.1%	44.3%	48.7%	44.9%

Although we were impressed by the performance of the (however **overfitted**) models, particularly the NLTK NB and the K-Nearest Neighbors model, we sought to analyze their confusion matrices to further understand. Unfortunately, when viewing the matrices, we found that they were very sparsely populated (filled by a disproportionate number of 0s indicating low prediction success rates for many artists). To take a closer look at these artist-dependent prediction rates, we looked at **precision, recall and F1-scores** for each artist for the top-performing NLTK NB model. We then quickly realized our AUC scores were artificially boosted by some artists whose F1 scores (indicates success of prediction by combining both precision and recall results). Some artists with markedly high F1 scores include NF (F1: 0.84), YBN Namhir (F1: 0.71), Luke Bryan (F1: 0.63), KIDS SEE GHOSTS (F1: 0.54). Logically, this makes sense; all of these artists had features from their songs marked as one of the most important features of the model indicating that they had lyrics that were distinctive and were reproduced throughout their discography. Another interesting case of an artist with a non-zero F1 score is Korean boy band, BTS (F1: 0.20). Their Korean lyrics are transliterated to English, obviously making them very distinctive. We suppose their score is not higher because their lyrics are not repeated extensively in their songs.



```
Confusion Matrix :
[[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
  [0 0 0 0 0 2 0 0 0 0 0 0 0 0 1 0 0 0 1
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



```
Confusion Matrix :
[[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 3 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
  [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Left: part of Confusion matrix for NLTK NB, Right: part of Confusion matrix for Bernoulli NB

	precision	recall	f1-score	support
BTS	0.06	0.00	0.20	6
Cardi B	0.00	0.00	0.00	1
Daddy Yankee	0.00	0.00	0.00	1
Drake	0.00	0.00	0.00	1
Ella Mai	0.00	0.00	0.00	1
J. Cole	0.00	0.00	0.00	1
Juice WRLD	0.00	0.00	0.00	1
KIDS SEE GHOSTS	0.37	1.00	0.54	30
Kane Brown	0.00	0.00	0.00	2
Kanye West	0.00	0.00	0.00	1
Lil Pump	0.00	0.00	0.00	4
Luke Combs	0.00	0.00	0.00	4
Nicki Minaj	0.00	0.00	0.00	26
Post Malone	0.00	0.00	0.00	1
Selena Gomez	0.00	0.00	0.00	1
Taylor Swift	0.00	0.00	0.00	1

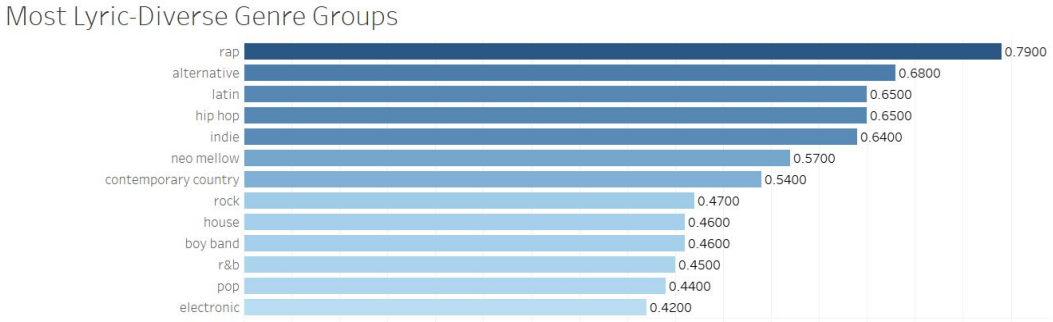
Scores for NLTK NB model

NOTE: We could not fit the entire screenshots, so these are parts of them to present an idea of what they looked like

We thus conclude that, of course qualified by the small size of our dataset, it is in fact possible to identify artists by their lyrics, but only given that the artists' songwriting is distinctive and makes use of these distinctive lyrics repeated frequently throughout their songs.

Question 2: ***Does the Genre Influence the Lyrics?***

We wanted to examine the extent of the relationship between genre and lyrics, if the genre would influence its unique lyrics, or if lyrics could be used to predict the genre. To do this, we first took the lyrics of all the songs and took every 3 consecutive words as a "trigram", noting which song they came from alongside the year for good measure. Unfortunately, we quickly realized that a direct relationship between trigrams and genres wouldn't prove productive in a visualization, as a direct visualization of trigrams that only appeared in one or two genres would result in mostly one-off lyrics found in one song. To combat this and the limited number of genres, we used these trigrams as values upon which to compare the genres. While it didn't make sense to directly compare genres by the sum of all trigrams in the songs, due to the limited number of songs per genre, comparing by the ratio of unique trigrams to all trigrams gave a metric for the lyric diversity per song which upon aggregating could be used to compare the genres themselves.



This made us realize that as people enjoy different genres, they have their reasons for listening to each, likely manifesting in a range of lyric-diversities based on each genre's expectations and the desires of their listeners. Pop music emphasizes catchiness at the expense of lyric diversity, and electronic music de-emphasizes lyrics as a whole, while other genres such as rap and hip hop focus on the words as the main vehicles of the song.

From here, we set out to see if we could predict genre from the lyrics using advanced methods. We primarily used **bag-of-words** for this section and classified on the bigrams of the raw lyrics, the bigrams of the lyrics after being cleaned of stopwords, and trigrams from the raw lyrics to see if word progression would have an impact as well. We also used the grouped genres as earlier but with slightly more merging (now only 9 genres) to ensure we had a plural number of songs in every genre; however, this unfortunately resulted in a dataset that was overwhelmingly pop music - overall, 76.6% of the dataset was one form of pop music or another. This makes sense considering the inherent bias in popularity resulting from the datasets we pulled from; however, proved to pose a significant challenge, especially with our limited data size. This resulted in classifiers tending to call everything pop music due to

underrepresentation or, even more dangerously, that some genres occasionally just wouldn't show up in the training set.

This was exemplified in our early classification attempts, including our first **Naive Bayes classifier**, which achieved 65% training accuracy, with a 3.3% accuracy rate for non-pop music. It would have achieved a better classification rate by simply calling everything pop music, and its non-pop classification accuracy wouldn't have suffered that much for it. However, even then, we couldn't in good faith call it a good classifier, despite the relatively high overall accuracy rate on paper. To this end, we subjected the datasets to a battery of classifiers, including some customized classifiers created with these issues in mind, and averaged the scores of each over 10 train-test data partitions using **K-fold cross validation principles**. We sought two objectives, not necessarily at the same time - to maximize overall accuracy, and to maximize non-pop music classification accuracy.

The classifiers used were Complement Naive Bayes, Support Vector Machine, Random Forest, K-Nearest Neighbors, and later SVM with subsets, CNB with subsets, and CNB, SVM, and RF with two-part classification. **Complement Naive Bayes** was used as a baseline classifier, generally considered one of the simpler methods of text classification, and we used the complement version as it's more suited for imbalanced datasets. Given a word, in the classification of a class, it would find the likelihood of a word appearing outside of that class, and in the prediction it would find the class with the lowest such score. Our **Support Vector Machine** used stochastic gradient descent to try and find linear classifiers to partition the data in high dimensions, intermittently hypothesizing a partition and testing it on a subset of the training data, improving the partition as it iterates. We used it because Support Vector Machines are generally considered to be one of the best text classification algorithms. We shall discuss what is meant by classifying with subsets and two-part classification later. We found that **K-Nearest Neighbors** gave the best results with only 1 nearest neighbor, even on the test data, and we applied **grid search** on as many classifiers as possible to try and optimize their parameters. The table below details the more interesting of the models trained, and each's average accuracy scores when predicting on the data they trained on and on the held-out validation set.

Classifier	**Complement Naïve Bayes**				**Support Vector Machine**				**Random Forest**			
Which dataset	*Not-pop*		*All*		*Not-pop*		*All*		*Not-pop*		*All*	
Train or test	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*
Raw Bigrams	100.0%	7.6%	100.0%	78.1%	98.9%	6.4%	99.8%	76.0%	98.5%	7.6%	99.7%	78.7%
Cleaned Bigrams	100.0%	10.7%	99.8%	78.4%	100.0%	8.2%	100.0%	78.2%	97.7%	10.0%	99.4%	79.7%
Raw Trigrams	100.0%	9.4%	100.0%	76.9%	99.1%	10.5%	99.8%	78.0%	97.3%	7.3%	99.4%	78.5%

Classifier	**K-Nearest Neighbors**				**CNB-subsets**				**RF 2-step**			
Which dataset	*Not-pop*		*All*		*Not-pop*		*All*		*Not-pop*		*All*	
Train or test	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*
Raw Bigrams	100.0%	16.3%	100.0%	63.7%	100.0%	25.3%	58.0%	34.6%	100.0%	7.6%	100.0%	78.7%
Cleaned Bigrams	100.0%	13.0%	100.0%	75.0%	100.0%	21.5%	59.9%	37.2%	100.0%	10.7%	100.0%	79.8%
Raw Trigrams	100.0%	12.3%	100.0%	64.2%	100.0%	25.5%	44.5%	33.6%	99.7%	7.9%	99.9%	78.6%

After classification was run, we saw that the **cleaned bigrams performed better** than or comparably to the raw bigrams and the raw trigrams in almost every category, so we shall proceed to primarily analyze their results. The **Naive Bayes classifier** performed much better with the smoothing parameter set by grid

search, now having a 78.3% accuracy rate in general; however, while the non-pop music classification rate jumped to 10.7%, this was still below the rate expected from random chance, being 1/9 or 11.1%. The **Support Vector Machine,** contrary to expectation, performed slightly worse in both departments, while **Random Forest** had slightly better overall accuracy at 79.7% but lower non-pop accuracy at 10.0% and the **K-Nearest Neighbors** classifier did slightly worse in overall accuracy at 75.0% but better in non-pop accuracy at 13.0%. Finally, we found a classifier that had a mean non-pop accuracy slightly above random chance; however, it was unlikely to be statistically significant.

We realized that while these were fine classifiers in general, they didn't suffice for data as skewed as ours - so we had to do some customization. Our first thought was to either artificially bloat our non-pop data by duplicating songs or artificially shrinking the amount of pop data by only using some of the pop songs; however, we couldn't think of a safe way to use data augmentation for this type of data, and we worried that leaving out some pop songs might just drop pop music classification accuracy. Nevertheless, we had to do something so we attempted a middle-of-the-road strategy: we would **train the classifier on multiple subsets of the data**, each including all of the non-pop music and a random subset of the pop music data, and we would average the results of each to create our predictions. We called this **training on subsets**, and implemented it for Naive Bayes and for the Support Vector Machine. The Support Vector Machine was a bust and we only succeeded in reducing accuracy on pop music; however, this method greatly increased the non-pop accuracy for Naive Bayes, doubling it to 21.5%. This was out-done further by the classifiers trained on the raw bigrams and the raw trigrams, reaching 25.5% accuracy. Unfortunately, all of these classifiers suffered from greatly reduced pop music classification accuracy, dropping the overall accuracy of all the subset-trained Naive Bayes classifiers below 40%. As such, we made one more attempt at greater accuracy in both objectives through the two-step classifiers. Our reasoning was that if many classifiers were just calling everything pop music it was because there were so many examples of pop music and so few examples of the other eight genres. Thus, we tried fitting two classifiers to the data: the first would say if it was pop music or some other genre, and if the first said that the music wasn't pop then the second would attempt to be more specific, being trained only on non-pop music. Unfortunately, no iteration of these (of which we attempted Naive Bayes, Support Vector Machine, and Random Forest) achieved a significant non-pop accuracy rate, instead taking the top spot for overall accuracy through increased pop music classification accuracy, reaching 79.9% with the Random Forest two-step classifier.

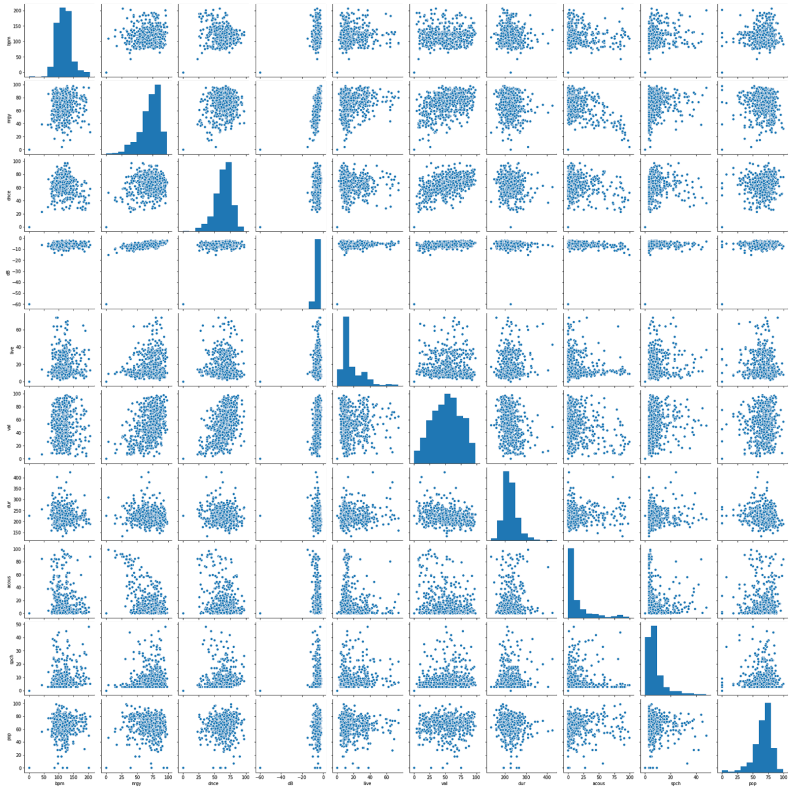
It must be noted that **overfitting** was an unfortunate reality of almost all our classifiers - the limited amount and nature of our data meant that combatting overfitting was difficult, and most attempts at doing so (ex. reducing max_depth for Random Forest) reduced validation accuracy as well, rendering our efforts pointless. The effect was most painfully obvious for the non-pop data: if one looked only at the results from predicting on the training data, all of the classifiers gave above 90% accuracy, in contrast to the abyssal scores that most received when predicting on the validation data.

In conclusion, our results are mixed: yes, but with an asterisk. Truly, while we have some stereotypes about some genre's lyric versus others, when you get to it it isn't all that easy to tell genres apart when there's nothing but text. For the data we had, we succeeded in creating classifiers that had almost 80% accuracy on the data and above 0.25% accuracy on non-pop music, albeit they were two different

classifiers, which points to greater accuracies being attainable on larger, more diverse datasets. If we were to continue this project and delve further into this question, we would use a larger dataset as such, and possibly attempt to fit using more classifiers denied to us such as neural networks, impossible on this dataset because of their obscenely large number of parameters to fit.

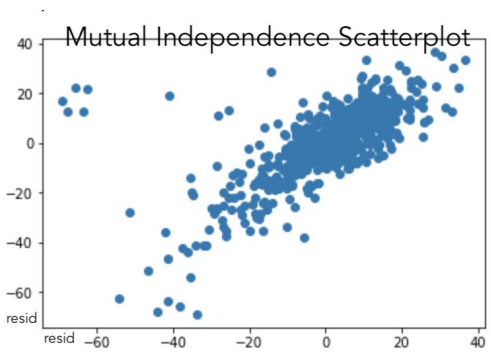
Question 3: ***Are We Able to Predict the Popularity of a Song From Its Features?***

Each year, there are a few songs that catch the attention of the world, appearing in commercials, Tik Tok dances, stores, and restaurants. We wanted to see if there was a specific formula for a song to become 'popular.' We identified the song's bpm, energy level, danceability, dB, liveliness, valence, duration, acousticness, speechiness, and popularity of its lyrics as characteristics that can be standardized across genres. These values were assigned to each song in the top 50 songs lists from 2010-2019 by Spotify-designated metrics. From these variables, we wanted to determine if it was possible to predict a song's popularity value before it gets released.



We began by analyzing possible trends between these characteristics, to see if any of these variables affected the other by plotting pairwise relationships. We found little visibly distinct relationships between the variables. We realized that with this pairwise plot we would not be able to see all of the correlations between the variables, due to the plots only showing clusters and specific ranges where the top 50 songs of each year usually resided. From there, to investigate how these variables are related and to determine the predictability of a song, we decided to apply regression tools to this dataset. As our definition of popularity is not defined on a binary scale, we attempted to implement linear regression rather than logistic regression.

When we began applying **the least squares model of linear regression,** we checked the **regression assumptions** of mutual independence, nonlinearity, normal distribution of noise, and constant variance, to make sure our model held. The first model that we evaluated was a multivariate linear regression consisting of all 9 characteristics to see if popularity could be determined from a combination of all variables. However, from the get go,



the model failed the assumption of mutually independent residuals. Since our data was not in time or spatial order, we were initially confused as to how this could be. However, we realized that some of the characteristics could have qualities where two values of the same variable could be correlated, hence being autocorrelated. We hypothesized that these dependent residuals were coming from variables where the data was consolidated in a smaller range, such as the dB and duration. This smaller range would allow for a greater degree of correlation between values from the same variable.

In order to test this theory, we chose a limited number of characteristics to do our multivariate linear regression on. We were taking a risk by picking and choosing these characteristics as our variables at this stage. Going along with our assumption, we tested the linear model with characteristics excluding the potentially autocorrelated variables of dB and duration. After multiple iterations, we found that our hypothesis did indeed stand; when the variables of dB or duration were included, the auto-correlation plot appeared more linear and dependent. However, we ultimately knew that this method of picking and choosing variables would result in unreliable p-values. When taking a look at the model summary of this severely limited model, this was exhibited. In this model, we had an R-squared value of 0.93, an F-statistic of 2833, a probability F-statistic of 0, and a mean squared error of 376.72. The R-squared value close to 1 indicated that the linear model was most likely a good fit. Similarly, the probability F statistic of was 0 and the F-statistic value was very large. This indicated that all of our variables were statistically significant. However, our mean squared error was extremely high and the probability F statistic was exactly zero, which immediately raised many red flags for us. This indicated our assumptions were not nearly as strong as expected, that we had hastily eliminated too many variables and features in our process. Similarly, we had eliminated the bpm, and song duration from the model, which one could argue are the most prominent song characteristics.

Thus, we went back to our original linear model to determine a better linear fit and split our data into **testing and training sets** to accurately test our model. By running code to **minimize the AIC value to improve our model fit**, we found a new model with a total of 6 variables: the bpm, energy value, danceability, liveliness, duration, and acoustimass. The AIC of this new model was lower than that of the severely restricted model with cherry picked variables, reinforcing our understanding that the cherry-picked model was unreliable. However, when we analyzed the residual assumptions, this new model failed to be mutually independent and normally distributed in its residuals. Thus, the p-values for this linear model were invalidated. After failing the residual assumptions across models and highlighting the unreliability of limiting variables, this dataset has shown to not be compatible with a linear regression model.

In future analysis, we would like to attempt fitting a non-linear model to this dataset, in order to determine the predictability of a song using its characteristics. As non-linear models can flexibly adjust variables to weigh them differently, our next hypothesis is that the predictability of songs can be determined and a specific recipe for the basic features of a ‘popular’ song can be characterized. However, this would only be within the limits of our 9 defined variables and not to any other influences, which could limit its application. As we know, human attraction to music can be influenced and characterized by many outside forces including weather, mood, cultural events, etc.. Perhaps music and human behavior are not meant to be easily predicted and this unpredictability and fleeting attraction is what makes art so special.

Bibliography

Spotify Dataset:
Description: The top songs by year in the world by spotify. This dataset has several variables about the songs and is based on Billboard rankings.
https://www.kaggle.com/leonardopena/top-spotify-songs-from-20102019-by-year

Billboard Lyrics Dataset:
Description: Top 100 songs from 2000 to 2018 with lyrics acquired from Spotify's Genius feature.
https://data.world/typhon/billboard-hot-100-songs-2000-2018-w-spotify-data-lyrics/workspace/file?filename=billboard_2000_2018_spotify_lyrics.csv