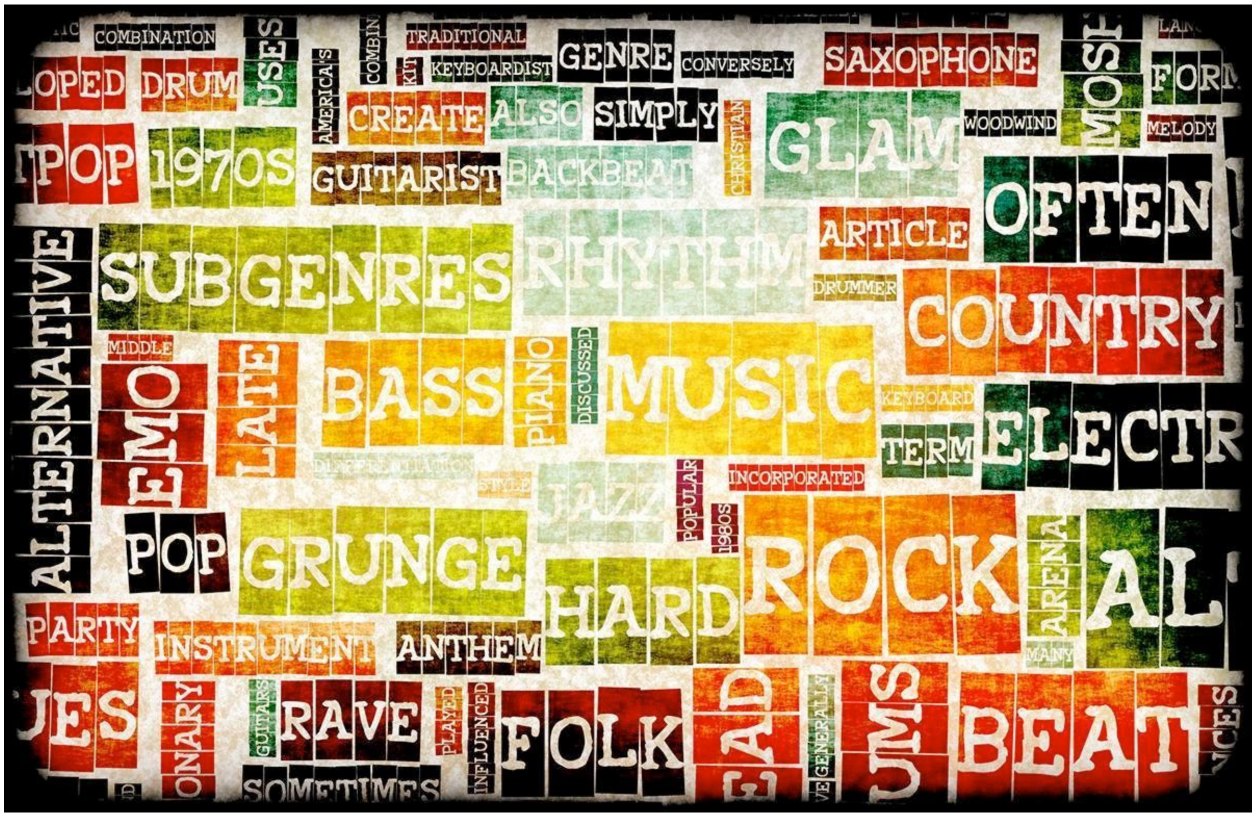
Record Labels

Classifying Musical Genres Using Song Lyrics



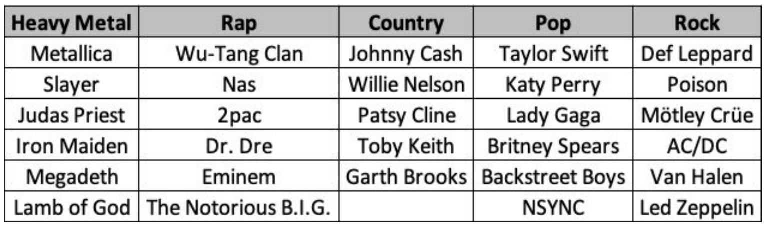
James Walsh

December 14, 2019

Springboard Data Science Career Track - Capstone 2 Report

We have all heard those debates that can go on endlessly amongst friends. LeBron or Jordan? Mac or PC? Five Guys or In-N-Out? (For my money: Jordan, Mac, Five Guys.) Musical preference is no different. Favored artists and styles can be topics of long discussions and arguments. A lot of time can be spent even disagreeing on what genre artists and songs fall under. While much of it boils down to arbitrary metrics, the question of what truly defines musical genres remains. Surly compositional aspects like instruments, tempo and key are important, but what about the lyrics themselves? Can a song's genre be accurately classified using only its lyrical text? Using over 3000 song lyrics gathered from LyricWiki by Fandom.com, I set out to explore songs' lyrical content, structure and themes to see just how well lyrics could define a genre.

The first step was identifying the data I wanted to gather. Since my primary focus was classifying the lyrics by genre, I wanted to choose a few genres that were fairly distinct from each other to avoid as much overlap as I could. To that end, I decided on heavy metal, rap, country, pop and rock since there are some clear distinctions between these five. Furthermore, within each genre, I deliberately selected artists that undoubtedly fall under their prescribed genre in order to maintain as much uniqueness to each genre as I could. I started with three artists from each genre, for which I was able to gather roughly 1400 lyrics. I thought more would help, so I increased to six artists in each genre except country. Because I was successfully gathering nearly twice as many lyrics from the country genre than the others, I used five artists instead of six.

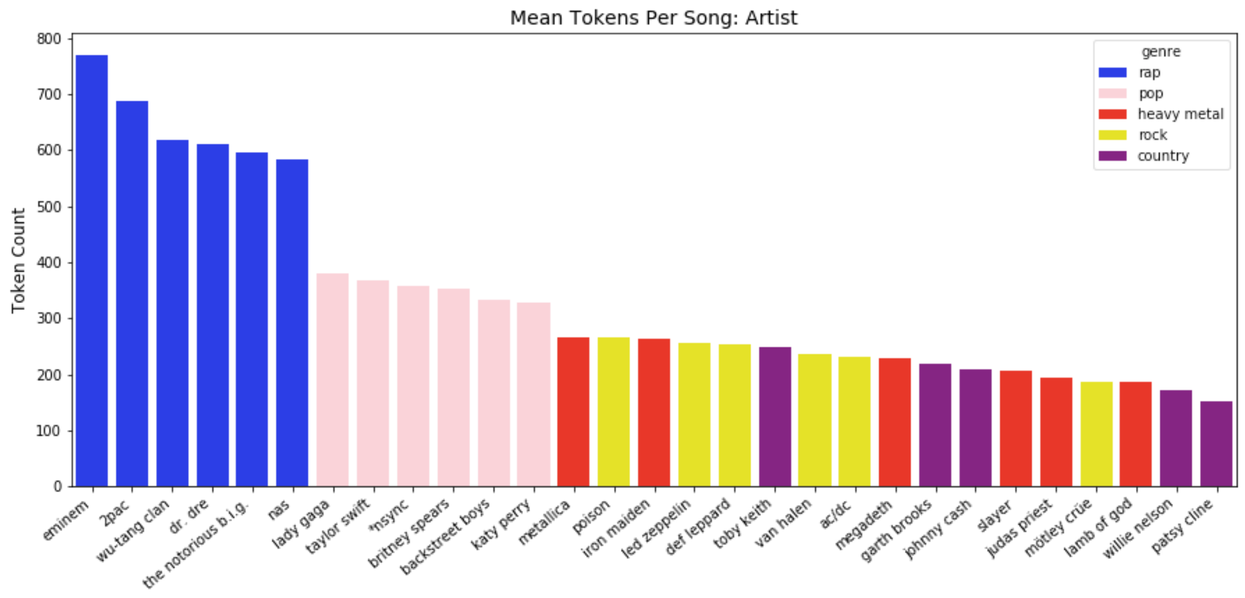


Heavy Metal	Rap	Country	Pop	Rock
Metallica	Wu-Tang Clan	Johnny Cash	Taylor Swift	Def Leppard
Slayer	Nas	Willie Nelson	Katy Perry	Poison
Judas Priest	2pac	Patsy Cline	Lady Gaga	Mötley Crüe
Iron Maiden	Dr. Dre	Toby Keith	Britney Spears	AC/DC
Megadeth	Eminem	Garth Brooks	Backstreet Boys	Van Halen
Lamb of God	The Notorious B.I.G.		NSYNC	Led Zeppelin

With my artists identified, I set out to collect as many lyrics from each artist as I could. I searched the internet for reputable sites that either were structured for easy scraping or had an efficient (and free) API. I came across Musixmatch.com which did indeed have an API which only required a short application process. After a few minutes, I had my key and started reviewing their API documentation. The site even had a nice "API playground" so I could try out my key and familiarize myself with the calls and procedures. The site claims to have "the world's largest lyrics database with more than 14 million lyrics" and their data retrieval gets very granular. Each artist has a unique ID, as does every album and individual song. This is all great...if you know the ID of the data you are looking for. To gather the song ID's required to pull out the lyrics, I wrote some code for API calls that produced the artist ID for each artist on my list. Using those artist ID's, I used the API to gather all albums under that artist ID. There were several duplicate albums under each artist, some with the full track list and others with only partial lists. Determining which specific album ID contained the correct song listing could be a bit of a mess, so I gathered all album / album ID pairs. Once I had all the album ID's, I was able to retrieve all track ID's for the songs on those albums and with the track ID's I was able to pull out the lyrics.

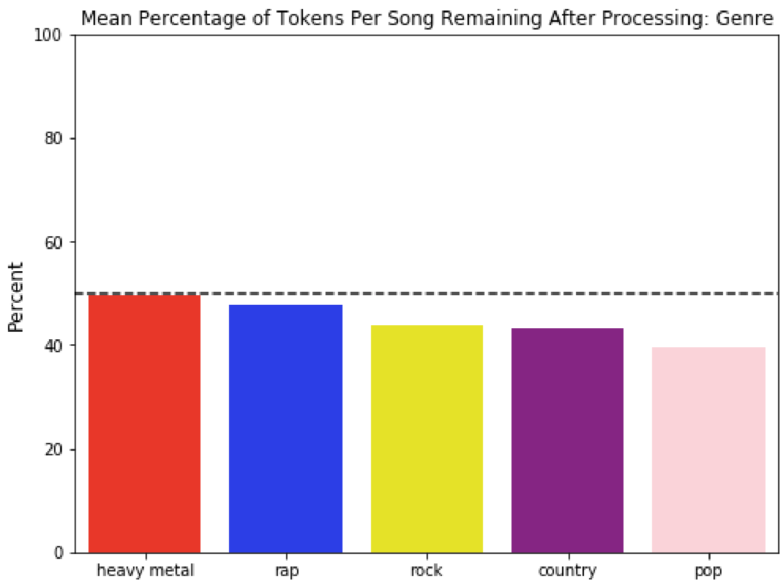
Unfortunately, I discovered that my free Musixmatch API key only allowed me to pull 30% of a song's lyrics. This would not satisfy my goals, so I returned to searching the internet for other options. I came across itunespy, a Python package with built in functions for retrieving song lyrics from LyricWiki by entering in an artist and song title. I tried it out and it did in fact produce the complete, accurate lyrics. It also had a function to return all albums by a given artist and all tracks from an album. However, the *get_tracks* function only took an album title as input, so there is no way of specifying the album you want if there are multiple with the same title which is more common than I had expected. This meant I would have to manually compile a list of songs for which I wanted to obtain lyrics. So, I had one method that produced the lyrics I sought, but only in partial form, and another that produced the complete lyrics but required that I sort out artist / song myself. To get around this, I set out to combine the two so as to use Musixmatch to acquire complete, accurate track listings and itunespy's *get_lyrics* function to retrieve the full lyrics. I found the code behind the *get_lyrics* function on Github and tweaked parts of it to work with the output from the Musixmatch API. By combining the two, I had my code set up to produce all the lyrics from the artists / albums in the nested dictionary I created at the start. If I wanted to add more, I could simply add them to the dictionary without having to do a lot of rewriting.

Because the free Musixmatch API key I was granted caps at 2000 calls daily, I gathered half the lyrics one day and the rest thereafter. Through the process of gathering the lyrics there were duplicate songs and those for which lyrics could not be found through the web scraping, so I ended up with just over 3000 lyrics. Some songs simply did not have lyrics published on the LyricWiki site, others song titles had odd additional terms like "edit", "live" or "remastered" and thus did not work with the web scraper. However, Most of those cases did not impede me from getting lyrics to the original track, so this was more a way of cutting down on duplicated than an actual issue. With these sets of lyrics as my text data, I started building out the NLP processes. I saved all the data into a pandas DataFrame with lyrics, song title, album, artist and genre as the columns for easier access and analysis. The goal for this phase was feature engineering so as to create a better dataset for NLP. From the raw text data, I calculated the length of the lyrics in characters, the total count of unaltered tokens, the average length of tokens per song and the count of contractions in each set of lyrics. After charting the data at this stage, it was clear that songs under the rap genre were by far the most verbose, followed by pop songs. Rock and heavy metal were fairly similar with country having the fewest words on average.



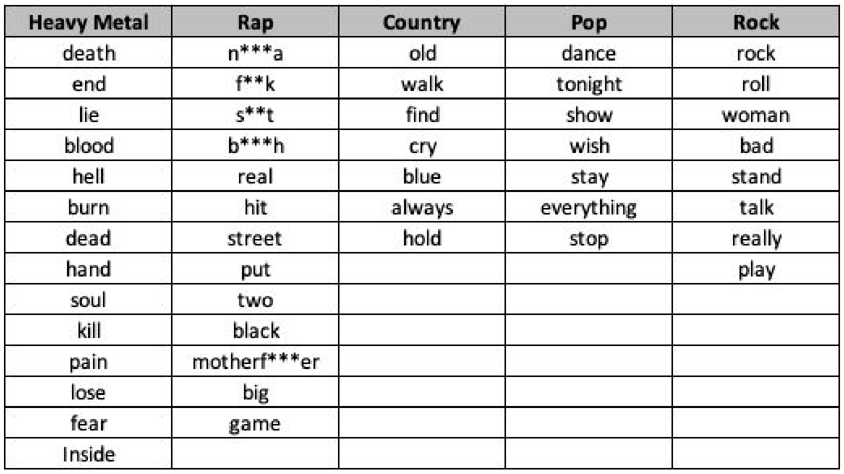
Next I began the phase of processing the data into manageable pieces for NLP. This stage went through several iterations as the project took shape and I began seeing more patterns in the data, like the use of common slang and misspellings. The first step I took was converting all tokens into lowercase to keep everything uniform, followed by tokenization using *regexp_tokenize* from the nltk library in order to isolate all words and essentially do away with any contractions. Once I had the tokens, I went over and edited common spelling issues like ending words with "in" instead of "ing" and "wanna" instead of "want to", but also some not-so common cases like "yaknahmsayin" into "you know what I'm saying". This list of word edits expanded several times as I continued to dig into the data and the lyric tokens got a little bit better each time. I also counted and recorded the total number of token edits made on each set of lyrics. Once I had the list of properly edited tokens, I used nltk's *WordNetLemmatizer* to remove suffixes and return the base form of the tokens. I also made a counter for each time a token was altered during lemmatizing and added those counts to data set.

Then, I removed the stop words and punctuation with nltk's *stopwords* and string's *punctuation* lists, calculated the average length in characters of the altered tokens and added all the above to the dataset as additional features. Like the previous steps, this went over several iterations as I discovered common words that were not included in nltk's *stopwords,* as well as junk words like "whoa" and "haha." I also counted unique tokens in each set of lyrics as well as listed the types of words (nouns, verbs, etc) the tokens were. After all the iterations of correcting spelling and removing junk words, rap songs were still the most verbose. Last, I calculated that each genre had half or less of the original token counts after processing. Heavy metal had retained the most at 49.5%, followed by rap (48%), rock (44%), country (43%) and pop (40%). This was interesting: rap averaged far more tokens per song than heavy metal, but the latter retained a greater percentage of their total tokens after processing. Apparently rap my be more verbose, but much of that text were either stop words or junk text that had no analytical value like "wooo" and "yeah." Pop songs retained the lowest percentage of their original tokens, indicating a heavy use of stop / junk words.



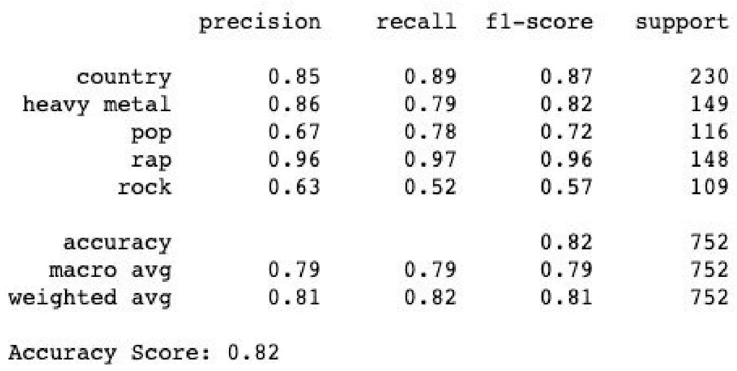
Aside from stop words and other meaningless tokens, I also wanted to remove as many tokens that are common across all genres as would be helpful to the modeling. This step would probably be repeated over several iterations as I tested and refined the model, so I made a simple list of the top 20 tokens across all genres from which I would select the most common and remove them from the complete corpus. I also wanted to get a feel for the most common tokens per genre to see where there were unique and overlapping values so I could continue to make each genre as unique as possible and easier to classify.

The best way to do this was to make a list of the top 50 most common tokens in each genre, and then remove any tokens from one genre list that were also found in any of the other genres, returning only tokens that were unique to that genre. For contrast, I also compiled a list of tokens that were not unique and found in each of the five genres, just incase I needed to remove them. The resulting token lists were very close to what I had expected for each genre. Heavy metal featured terms like “death”, “pain” and “fear” which are all common themes for metal. Rap contained a lot of profanity, which is common, as well as terms like “street” and “game”. Interestingly, rock’s most common unique tokens were “rock” and “roll”, so apparently rock’s favored topic is rock itself. Overall, these lists were pretty good summaries of common themes for these particular genres. So useful, in fact, that I added another feature in the primary DataFrame that would count the occurence of the terms in these lists to increase the uniqueness of each lyric under the different genres.



Heavy Metal	Rap	Country	Pop	Rock
death	n***a	old	dance	rock
end	f**k	walk	tonight	roll
lie	s**t	find	show	woman
blood	b***h	cry	wish	bad
hell	real	blue	stay	stand
burn	hit	always	everything	talk
dead	street	hold	stop	really
hand	put			play
soul	two			
kill	black			
pain	motherf***er			
lose	big			
fear	game			
Inside				

After all the processing, it was time to try a classifier model and see what kind of results I got. This was likely to be an iterative process and call for me to refine the text data and token processing even further. First, I produced a word vector using the processed tokens for each song, which resulted in a vector of a little over 17,000 tokens. I also created similar word vectors for the grammatical type of word a token is (noun, verb, etc.) as well as the top tokens that were unique to each genre as previously described. I then removed the original lyrics, the albums and the genre labels, leaving behind the song and artist. I left those two so I could explore which song lyrics were misclassified after running the model. After concatenating all these vectors into one single array, I selected a Multinomial Naive Bayes model from scikit-learn to perform the classifying of genres. After prepping the training and testing sets, the model yielded an overall accuracy score of 78%, which was not too bad for a first pass. Having exhausted any ideas for additional features to add, I returned to the common tokens found in all genres and removed a few more of them. This increased the model performance up to 82% accuracy. I repeated the step of removing some common words, but then the accuracy dipped to 80%, so I reverted back to the previous list as it seemed to produce the best results.



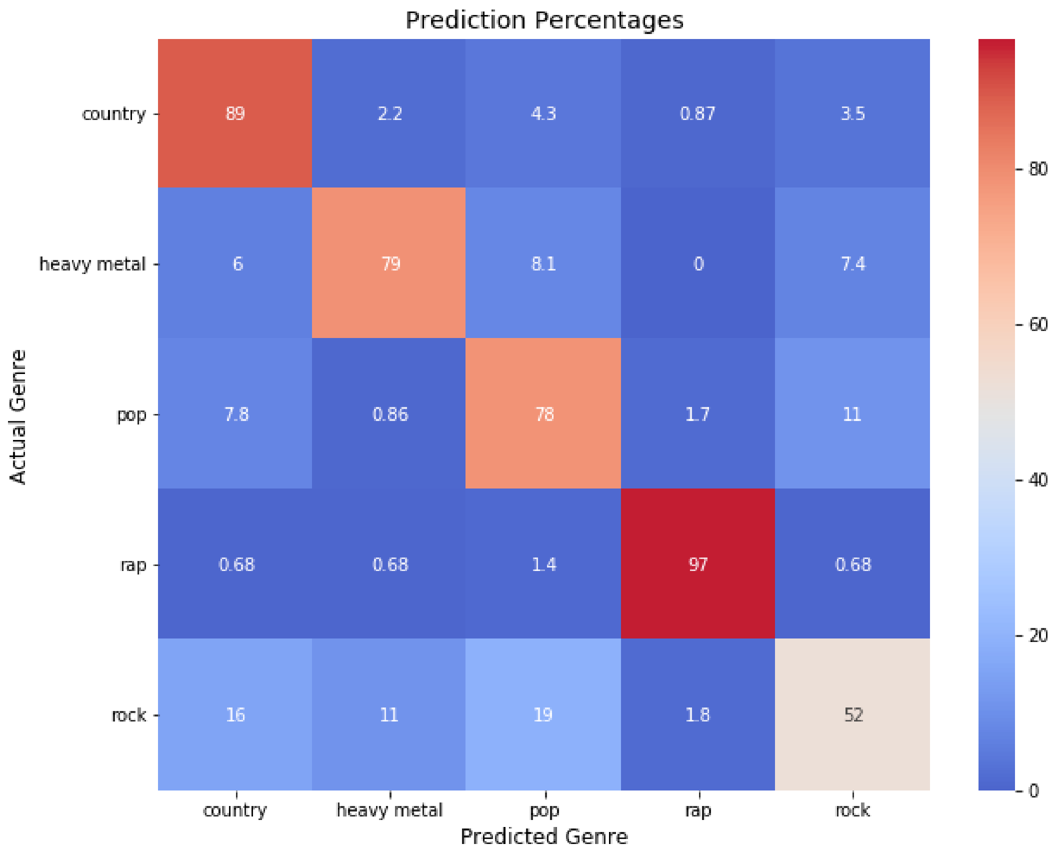
```
              precision    recall  f1-score   support

     country       0.85      0.89      0.87       230
 heavy metal       0.86      0.79      0.82       149
         pop       0.67      0.78      0.72       116
         rap       0.96      0.97      0.96       148
        rock       0.63      0.52      0.57       109

    accuracy                           0.82       752
   macro avg       0.79      0.79      0.79       752
weighted avg       0.81      0.82      0.81       752

Accuracy Score: 0.82
```

A confusion matrix of the resulting labels also provided some interesting insight. Rap was by far the easiest to classify at 97% accuracy. Of the five songs that were mislabeled, two were spoken skits, which are common on rap albums and could have thrown off the model. Two more were labeled pop, which is not egregious, especially considering the songs “You Won’t See Me Tonight” by Nas and “Space Bound” by Eminem are particularly radio-friendly. “You Wouldn’t Understand” by Nas was labeled as country, which is odd, but it is also a particularly short song which may have lead the model to liken it to country rather than rap. Country was also labeled rather successfully at 89% accuracy. 4.3% were labeled as pop and 3.5% as rock, both of which are understandable given some similarities between those genres. However, two songs, “Ballad of the Harp Weaver” by Johnny Cash and “Jacky Don Tucker” by Toby Keith were labeled as rap. This is odd, but both songs have a relatively high amount of tokens for the



country genre. It lyrics length and overall token counts are two major factors the model uses for classifying. Rock was the most difficult to label with an accuracy of only 52%. 16% of rock lyrics were labeled as country, 11% as heavy metal and 19% as pop. A review of the mislabeled songs did not reveal any obvious patterns, but my takeaway is that rock as I defined by the lyrics in this exercise simply has too much in common with the other genres (except rap) and it is more difficult for the model to label those songs than the other genres.

Overall, the model performed better than expected, likely due to the amount of token processing performed prior. These were good results, but I wanted to see if any other model would work better. I used a Random Forest Classifier from scikit-learn on the exact same data arrays as my Naive Bayes model and got some fairly similar yet notably inferior results. Model accuracy dipped to 80%, but accuracy of rap labeling increased to over 99% with one song labeled as country. Country received a bump as well up to 93%, heavy metal stayed at 79% and pop dipped slightly to 72%. Rock remained the most difficult to classify and suffered the largest drop in accuracy; down to 38%. With this model, 35% of rock songs were labeled as country, up from 16% in the previous model. I also tried to model using a scikit-learn's SGD (Stochastic Gradient Descent) Classifier which performed even worse than the Random Forest. Model accuracy dropped to 53%, but rap remained at 97%. Heavy metal rose to 90% accuracy, but it the rest of the genres' accuracy dropped dramatically. 53% of country lyrics were labeled as heavy metal as were 63% of rock lyrics. In fact, less than 1% of pop songs were labeled as pop, 58% were labeled as heavy metal and 31% as rock. The SGDClassifier seemed to label rap accurately, but most every other genre as heavy metal. Of the three, Naive Bayes clearly performed the best.

In the end, using Naive Bayes, rap was the most successfully classified genre (97%), which is understandable given the volume of tokens per song and the relatively unique use of language. Country also has a fairly unique lyrical syntax as well as the very least amount of tokens which would bolster the success of predictions there (89%). Rock was difficult to label, which could illustrate that the genre is more generic in terms of content and construction. It may be more common for elements of rock to spill into country, pop and metal and vice versa than I expected and there are not enough unique tokens to truly set it apart. More data (songs / lyrics) would likely boost the performance of the models and that is an opportunity to grow the project. There is also the potential to factor in other aspects of the songs, like track runtime, key and tempo that would serve as useful features labeling, but because I wanted to on the lyrics specifically, I'd say the model works better than expected. So, even if people disagree on which genre labels apply to what artists, computers can provide some clarity, at least with 82% accuracy.