

# Origins: Textual Analysis of Internet Lyrics

Using to NLP on Transcribed Lyrics to Classify Artists

# We are all familiar with Shazam

Shazam works by analyzing the captured sound and seeking a match based on an acoustic fingerprint in a database of millions of songs. If it finds a match, it sends information such as the artist, song title, and album back to the user.



# SHAZAM

**keep in mind is that a song  
has a unique fingerprint.**

---

**Now imagine a scenario that makes that unique fingerprint a bit more blurry or in this case a lot more blurry.**

**Problem Statement:**  
Can Natural Language  
Processing predict one  
artist from another  
through a set of  
*transcribed lyrics?*

# Meet the Data:

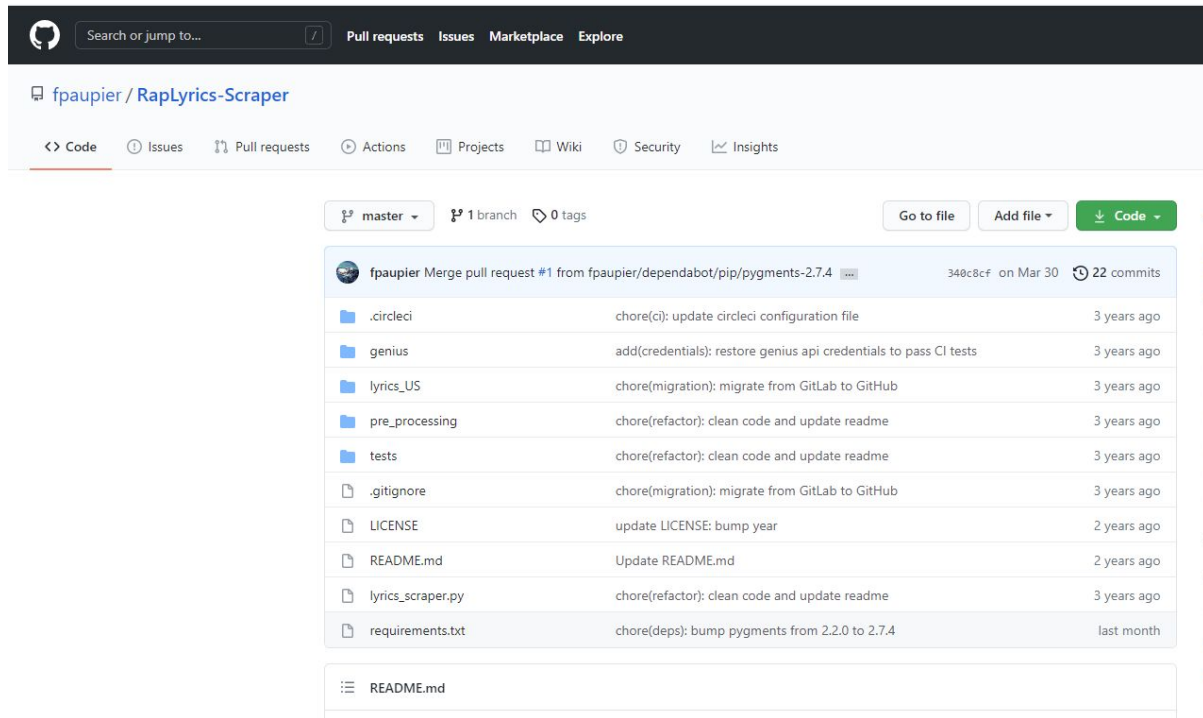
This data is mostly a contribution of the following Kaggle dataset that can be found here:

<https://www.kaggle.com/kerneler/s-tarter-rap-lyrics-8dd83bf6-f>

The source of most of these transcribed lyrics were acquired three years ago using the following api integration tool:

<https://github.com/fpaupier/RapLyrics-Scraper>

The lyrics have been partially cleaned and formatted.



Search or jump to...

Pull requests Issues Marketplace Explore

fpaupier / RapLyrics-Scraper

<> Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags

Go to file Add file Code

fpaupier Merge pull request #1 from fpaupier/dependabot/pip/pygments-2.7.4 348c8cf on Mar 30 22 commits

.circleci	chore(c): update circleci configuration file	3 years ago
genius	add(credentials): restore genius api credentials to pass CI tests	3 years ago
lyrics_US	chore(migration): migrate from GitLab to GitHub	3 years ago
pre_processing	chore(refactor): clean code and update readme	3 years ago
tests	chore(refactor): clean code and update readme	3 years ago
.gitignore	chore(migration): migrate from GitLab to GitHub	3 years ago
LICENSE	update LICENSE: bump year	2 years ago
README.md	Update README.md	2 years ago
lyrics_scraper.py	chore(refactor): clean code and update readme	3 years ago
requirements.txt	chore(deps): bump pygments from 2.2.0 to 2.7.4	last month

README.md

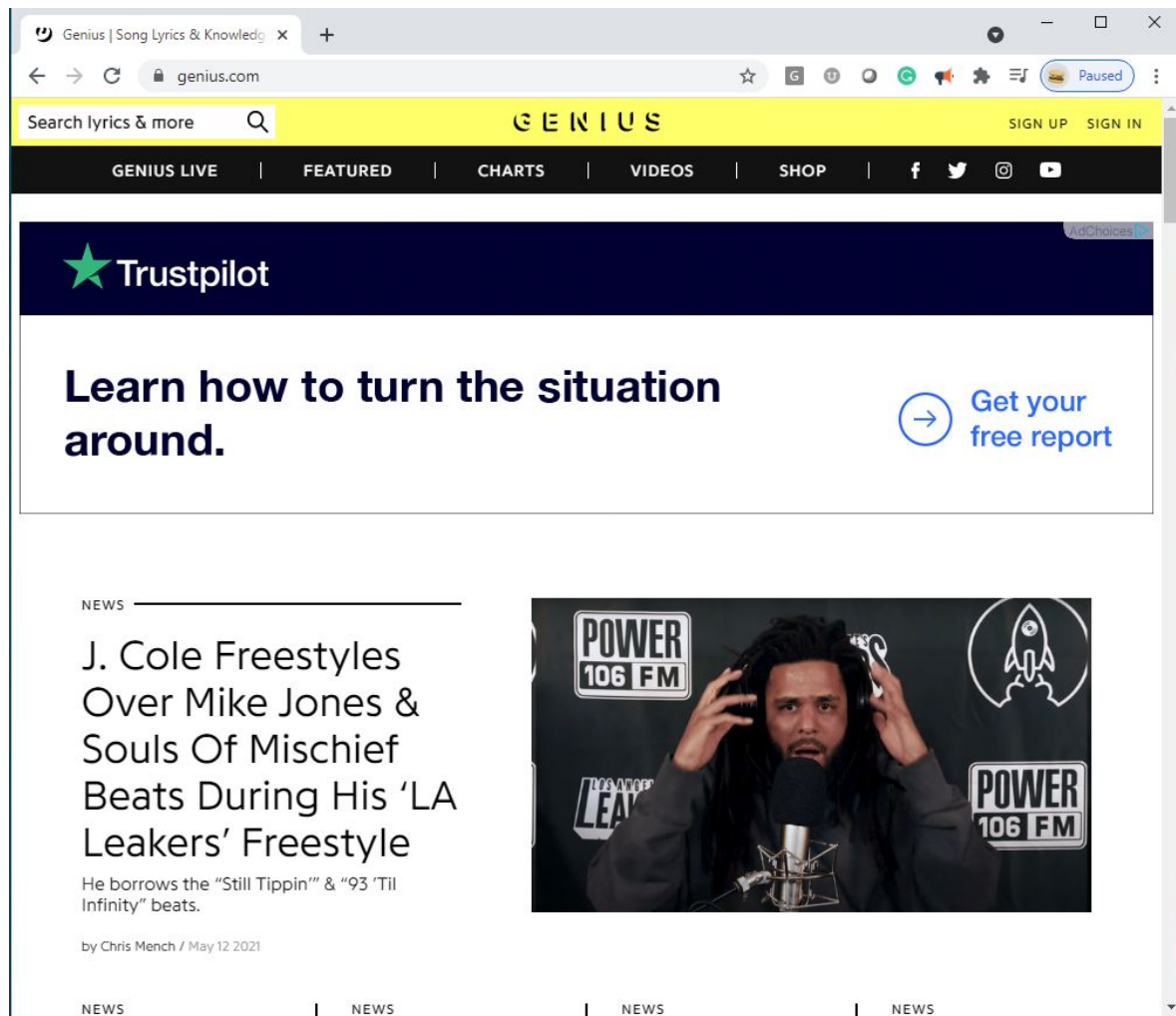
# To a lesser extent

— — —

A smaller portion of data has been gathered through the genius api directly and has been more extensively formatted.

This data is typically longer in both the number of characters and shorter in the number of lines this forms our control group.

**We will clarify their role later.**



The screenshot shows a web browser with the Genius website. The browser's address bar displays "genius.com". The website's header is yellow with the "GENIUS" logo and navigation links for "GENIUS LIVE", "FEATURED", "CHARTS", "VIDEOS", "SHOP", and social media icons. A search bar is located on the left of the header. Below the header, there is a dark blue banner for a Trustpilot advertisement. The advertisement text reads "Learn how to turn the situation around." and includes a button that says "Get your free report" with a right-pointing arrow icon. Below the advertisement, there is a news article section. The article is titled "J. Cole Freestyles Over Mike Jones & Souls Of Mischief Beats During His 'LA Leakers' Freestyle" and is categorized under "NEWS". The article text states: "He borrows the 'Still Tippin'' & '93 'Til Infinity' beats." and is attributed to "by Chris Mench / May 12 2021". To the right of the article text is a photograph of J. Cole wearing headphones and speaking into a microphone, with "POWER 106 FM" logos in the background. At the bottom of the page, there is a navigation bar with the word "NEWS" repeated four times, separated by vertical lines.

# Our objectives:

1. a) Classify an artist using a machine learning model  
b) Do so using both first a binary and then a multiclass classification



2. How can we account for where that success is taking place and what are the things that allow the model(s) to make an accurate prediction?



3. What makes these lyrics difficult to classify and how is this similar to NLP challenges in *real world* applications?

---

# The artists or classes

We have a total of 40 artists.

They all have varying numbers of songs. This also means that they have varying numbers of lyrics.

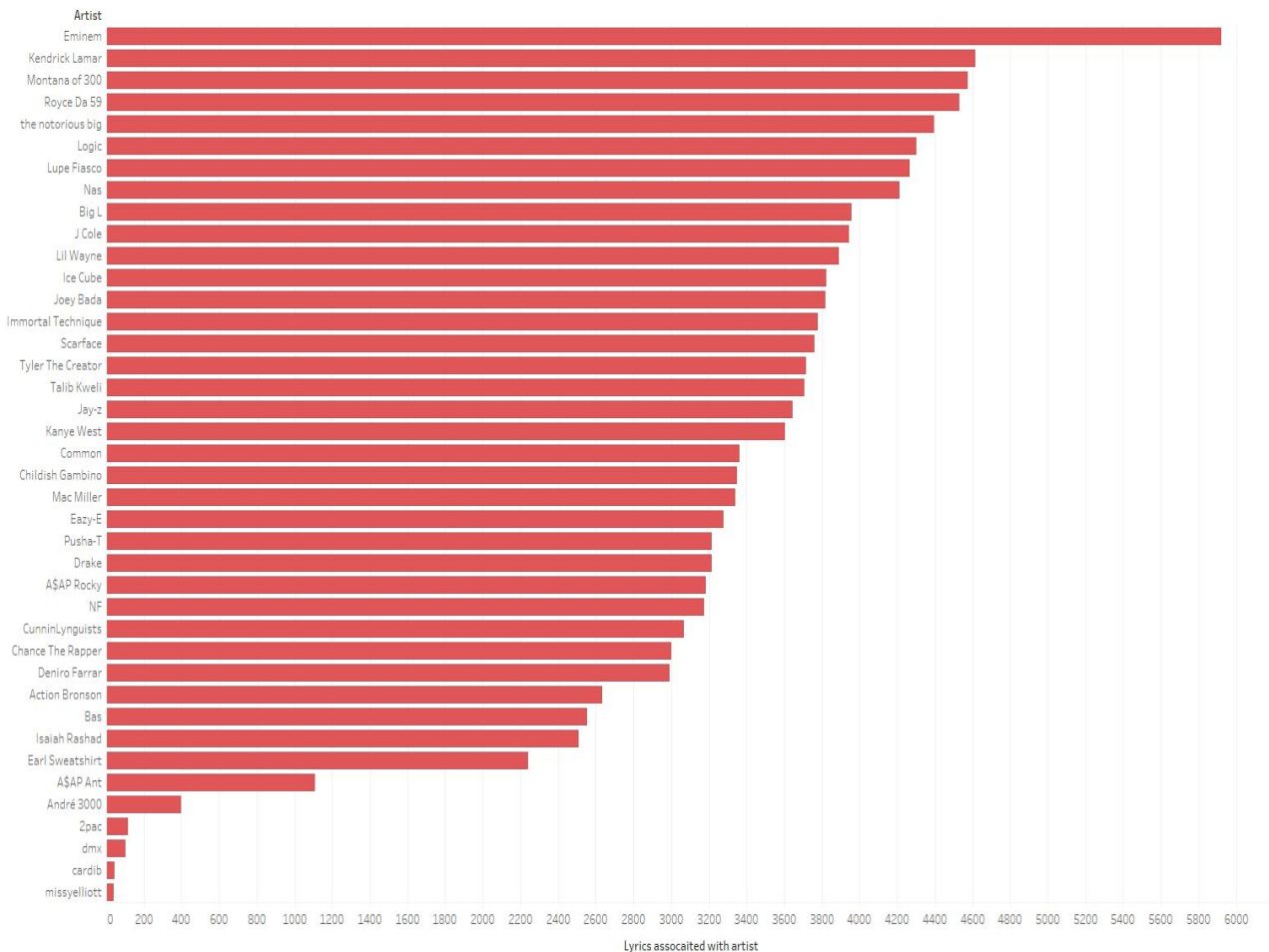
Our artist range from almost 6000 lyrics to less than 50 lines.

## Problem number 1

We are going to have imbalanced classes. Given the scenario, what can we do to improve the situation?

**Whatever we do we have to keep this in mind.**

Number of Lyrics associated with artists in dataset





# The text itself:

---

The lyrics provided broke songs down into verses, choruses, skits, intros and outros.

The immediate problem is that lyrics have been transcribed by different individuals so there is no exact uniformity

**Problem number 2 is how do we break down the texts?**

“As the world's biggest music encyclopedia with a passionate ***community of more than two million contributors***, Genius is a destination for artists, creatives, and superfans to discuss and deconstruct all things music.”

-- From the Genius web site  
(<https://genius.com/Genius-about-genius-annotated>)

# Lastly are the lyrics transcriptions:

Besides the fact that some of these lyrics are actually incorrect. Some of the lyrics contributors take the liberty to misspell words *on purpose*. There is also no real validation even if they are actually correct.

For example the word dogs:

Dawgs

Dawgz

Doggs

Doggggs

We will get more into this when we discuss selecting pretrained pipelines vs bag of words.

— — —

# The lyrics themselves outside of transcription:

— — —

**Redundancy:** Rappers are notorious for using other artist's lines in fact there is even a term for it called biting. Will this affect the contents uniqueness that it will hinder classification?

**Limitations of vocabulary:** Rappers may create their own slang, jargon and terminology but they do subscribe to a community driven set of terms and how do those affect the results if they are using a similar lexicon of words?

**The absence of delivery:** Many have likened Tupac delivery to that of a church sermon and while some said Biggie emphasized less on the words but more on certain sounds in words. Either way the way deliver and style is a large part of how we can distinguish performers and that is also true of something like Shazam. How can that carry over the text?

# Solutions to problem #1

“How can we deal with imbalanced classes?”

1. The larger segment follow a semi uniform pattern  
(Will discuss this in the next slide)
2. The smaller dataset where they have larger spans of lyrics  
(will also be discussed after the above in two slides)
3. **Stratify:** In a classification setting, it is often chosen to ensure that the train and test sets have approximately the same percentage of samples of each target class as the complete set.

— — —

# Organizing the corpus --finding the patterns

— — —

## Pattern number one:

`[Chorus],[verse 1],[verse 2]`

These designate what the lyric is or if its something like a skit or an intro. Rappers like to put artistic concepts before songs. These are not lyrics so we successfully need to weed these out.

The chorus is often a duplicate and should be eliminated as should verses that reference the rappers name if they are featured or a guest in a song.

**So we break first on anything within the [brackets].**

## Pattern number two:

`"n\n\"`

This allows us to break the files down into the elements of the song.

**This gives us the verses.**

## Pattern three:

`"n\"`

The single line break is what will be our final designation.

**Lets see how this is applied.**

# Hip Hop rhyme structure

## 101

— — —

You may have heard rappers refer to their “bars”. What does that mean?

Since at least the 1650s, **a bar has meant a song’s time signature, or the number of beats in each measure.** It comes from the use of an actual line, or bar, to mark out musical measures.

**In Hip hop “the bar” is the rhyme.**

So this how a rapper is supposed to keep the beat.

Each line that we broke on is a bar or half of a bar. That granularity gives us our training and test data.

So our data on a line by line basis should look like this:

“`[Verse 1]` **--first break(turns these to white space)**

`n\n` **--Second break(turns these to white space)**

Freedom or jail, clips inserted, a baby's being born\n

Same time a man is murdered—the beginning and end\n

As far as rap go, it's only natural I explain\n

My plateau, and also, what defines my name\n

First it was Nasty, but times have changed\n

Ask me now, I'm the artist, but hardcore, my sign is for pain\n

I spent time in the game, kept my mind off fame\n”

**So each line in the dataframe would equal one bar or half a bar. Depending on the transcription.**

# Introducing our control group

---

Earlier I mentioned that there were two data sources. I initially sought to scrape the data myself. In doing so I structured it differently on purpose to see if it will have an impact on our binary classification and our multi-class classification.

So our smaller control group would look the same except with whole verses.

“**[Verse 1]** **--first break(turns these to white space)**  
**n\n\****--Second break(turns these to white space)**

Freedom or jail, clips inserted, a baby's  
being born  
Same time a man is murdered—the beginning and  
end  
As far as rap go, it's only natural I explain  
My plateau, and also, what defines my name  
First it was Nasty, but times have changed  
Ask me now, I'm the artist, but hardcore, my  
sign is for pain  
I spent time in the game, kept my mind off  
fame”

# Solutions to problem #2

“How can we navigate through a pile of text?”

1. Looking for “patterns” just as we did with newlines, tabs and returns.
2. Performing transformations of modified words to give them the correct spelling to reduce the number of words.
3. Taking advantage of Stopwords enhance differences between classes.

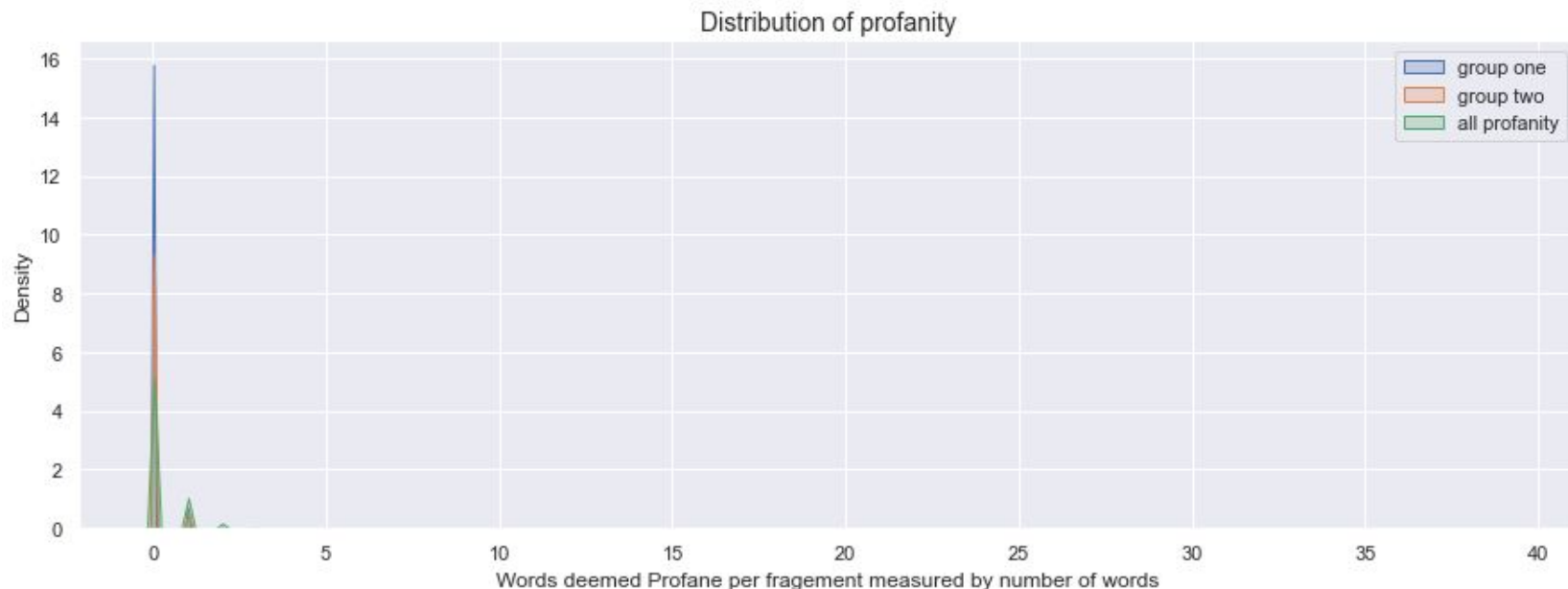
— — —



# The largest text pattern in the text was profanity

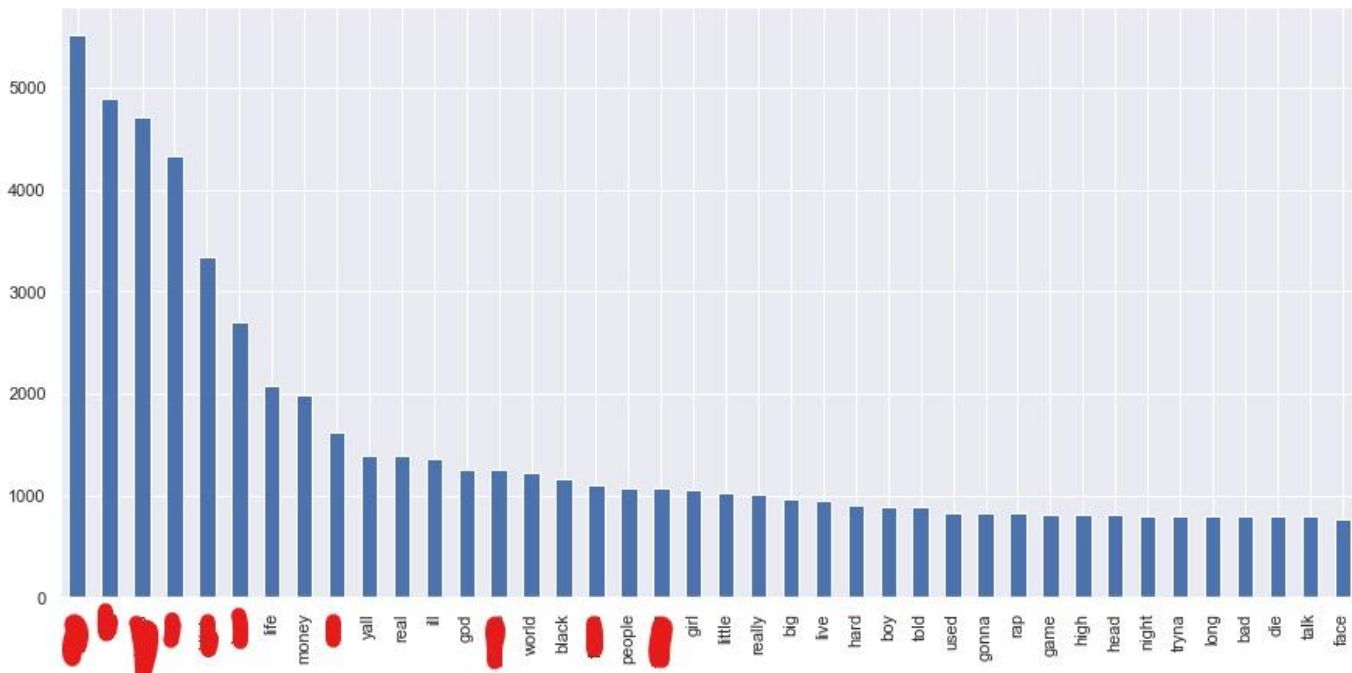
The number of **48,317** expletives and the total number of words in the corpus is **1,152,881**. These became a **primary component of my stop words.** (The point here is develop contrast to the next slide)

**4.19% are expletives. Here is what 4 percent looks like.**



Yes that does not look impressive but our lexicon analysis comes back to profanity over and over again.

Most frequent words in the corpus



Here is what a  
censored word  
cloud would like of  
the top 25 words.

This became the first group of words that seemed to span all the artists and also seemed that we could experiment with.



\_\_\_\_\_

# What if the number of expletives is tied to the number of words?

## Making sure that the number of expletives and the number of words were not related

The easiest way was to definitively determine this was a T-test between total of words and total expletives.

```
Ttest(statistic=453.6588014494813, pvalue=0.0)
```

And the low p-value speaks for itself.

# Solutions to problem #3

“These lyrics are kind of  
similar to each other how  
can anyone tell them  
apart?”

1. Are they really?  
Getting a sense of  
signs of individual  
fingerprints
2. Why I choose not to use  
an pretrained NLP  
pipeline.
3. General reservations

— — —

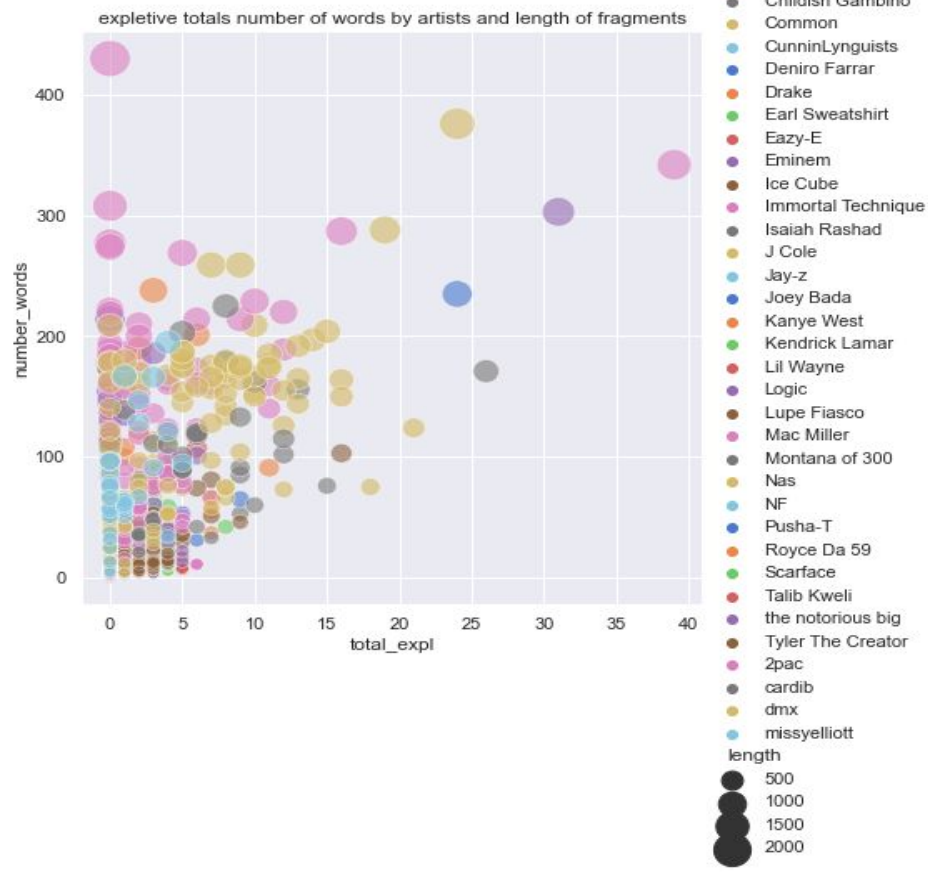
Our EDA could not conclusively say one way or another if they were visible clues to tell the artists apart

— — —

The difficulty with NLP EDA is that you are stuck creating your own features to try to analyze and often times it is a limited range.

The features were supplemented with sentiment analysis but it seemed fuzzy to me because at times it performed rather poorly. This goes both for Textblob and Vader.

Even with analysis applied on the expletives it seemed as if the vast majority of artists were mildly undistinguishable.



# What is a pre trained pipeline?

— — —

A pretrained pipeline is an example of **Transfer learning** often it would solve the problem understanding text however in this case it seemed to return mixed results. “Language model pre-training has been shown to be effective for improving many natural language processing tasks” (Dai and Le, 2015;). A research paper from Google highlighting BERT held up a dataset with problematic movie scripts results.

These lyrics span decades and are in a complex vernacular that is then transcribed by any of 2 million people in a community.

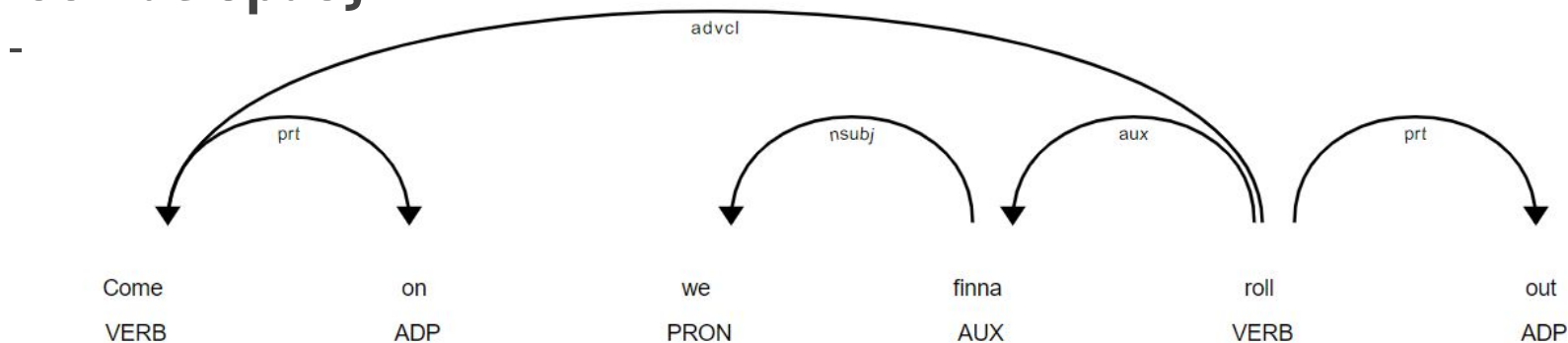
**After exploring several amazing and very cool examples of these**

**I came back to a very simple approach. Sorry BERT.**

**One language model performed better than CountVectorizer:**



# Let's take a quick look at Spacy.



## “Come on we finna roll out”

If you don't know what **finna** is it is contraction that means **“fixing to”**. Spacy recognizes the term as an auxiliary which is not bad thing actually in the same capacity as the word “is”. **This is impressive**. However it would take too long to deliver these results. The one language model that exhibited potential takes a long time.



# Designing an experiment.

---

**1. What are we predicting for?**

*What is our essential goal and how can we make it as simple as possible?*

**2. What are we neglecting as a result?**

*What are we sacrificing as a result. Is the trade off worth it?*

**3. Creating a plan.**

# What we are predicting for?

---

We have 40 artists the 36 with standard short length lyrical text and 4 with the longer sized text.

Our goal is fixated on we would call **True Positives and in the binary classification the True Negatives**).

The other measurement we are going to focus on is the **Accuracy**

In real world data applications that require quick turn around we often have to essentialize.

So here is what formulaically what Accuracy is equal to:

$$\frac{(TN + TP)}{(TN+TP+FN+FP)} = \frac{(\text{Number of correct assessments})}{\text{Number of all assessments}}$$

# Frank would not be happy with what we are doing.

The value of what Professor is discussing is actually **very real** if we were dealing with a clinical trials or where our negative results are important especially in improving our **true positives and minimizing false negatives**.

To be fair our experiment could be modified to include these values since we do have access to them.

**However**, we are largely neglecting sensitivity.



Frank Harrell

Professor of Biostatistics

[Vanderbilt University](#)

“One of the key elements in choosing a method **is having a sensitive accuracy scoring rule** with the correct statistical properties. **Experts in machine classification seldom have the background to understand this enormously important issue**, and choosing an improper accuracy score such as proportion classified correctly **will result in a bogus model.**”

Are we getting  
results better than  
a coin flip?

We are simply asking a yes  
or no question.

**However,**

When we do this we are  
limiting the scope of our  
experiment and we will  
soon see.

---

# Computational limitations

---

After the CountVectorizer we have 125219 rows  $\times$  6960 columns.

In fact Tfidf could not be run beyond 35% of the corpus without a memory error.

My machine wanted a divorce and Google Colab once again stole my 10 dollars.

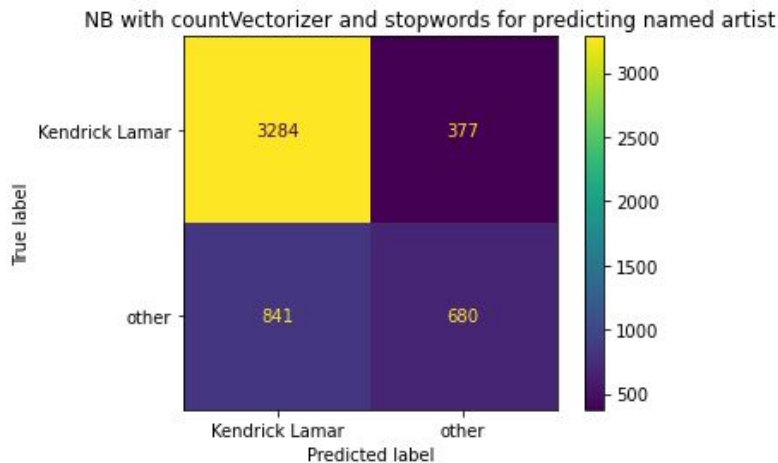
Similarly on the model selection front it seemed dismal in that the strain that my data was putting was hindering a simple prediction.



# A different approach...

— — —

In a classification our goal is to predict and output an explanation for that model (like our confusion matrix below). So in my case it would be something like this:



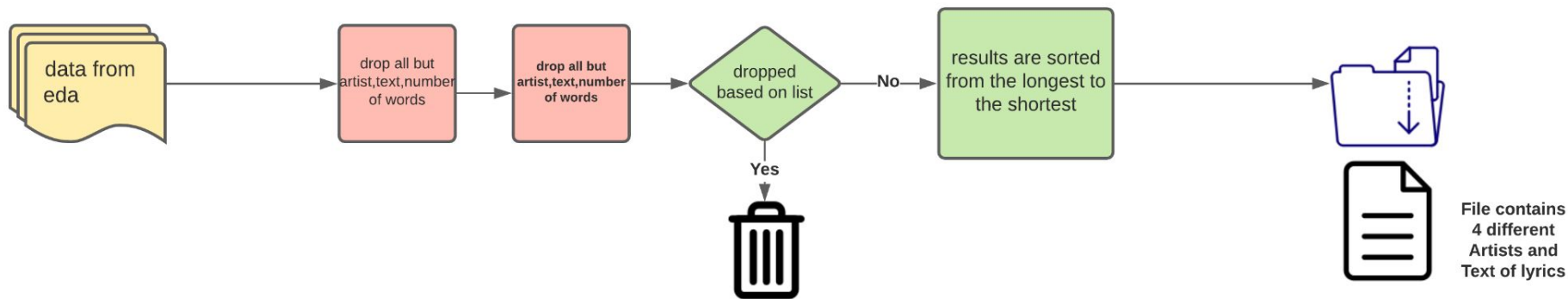
**What happens if lets we take a segment of our entire population and predict in small random groups and we limit our focus to just Accuracy and True Positives.**

# The experiment:

We will create a list of 4 artists out of the 40.

That means that there is 2,193,360 **permutations** that could be achieved or combinations where the artists would repeat

So we need a way to generate unique **combinations**.



Now all we need to do is to run this script a few hundred times. Which was easier to than I expected.

# Once we build a directory full of different artists and lyrics we can begin modeling.

— — —  
I created a set of functions that convert csv into a dataframe that consists of one target artist and others essentially creating our classifications.

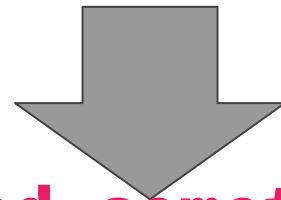
We use our first classification scenario to perform a gridsearch and use those best\_params.

Then pushing each file through a Binary Classifier. The results of the Accuracy, True Positive, True negative from the SKLEARN Classification report are then in a dictionary that appends to dataframe for the historic results.

So we have roughly 2,469 files. 2.7% of the total combinations we could make. Accounting for some of our combinations that do error out we are using only using a small segment of our data.

Again we are not predicting for every class or artist but rather seeing if we can predict for any class and how well.

```
15 {'artist': 'Childish Gambino', 'other_score': 0.8148827726809378,  
16 'target_score': 0.8200836820083682, 'accuracy': 0.8151244167962675}  
17 {'artist': 'Childish Gambino', 'other_score': 0.7974559686888454,  
18 'target_score': 0.7582417582417582, 'accuracy': 0.7942497753818508}  
19 {'artist': 'Childish Gambino', 'other_score': 0.7604311543810849,  
20 'target_score': 0.7531760435571688, 'accuracy': 0.7592646629705282}  
21 {'artist': 'Childish Gambino', 'other_score': 0.8093136173030236,  
22 'target_score': 0.7947019867549668, 'accuracy': 0.8084005793503001}  
23 {'artist': 'Childish Gambino', 'other_score': 0.7670588235294118,  
24 'target_score': 0.7667910447761194, 'accuracy': 0.7670179436058103}  
25 {'artist': 'Childish Gambino', 'other_score': 0.7928376025864213,  
26 'target_score': 0.7655367231638418, 'accuracy': 0.7906285714285715}  
27 {'artist': 'Childish Gambino', 'other_score': 0.7900048053820279,  
28 'target_score': 0.756578947368421, 'accuracy': 0.7877295118674429}
```



**I had something  
to work with!**



# Model selection: Go with whatever finishes less than 2 hours



Il mio nome è  
Bernoulli...

Models that took hours for what Bernoulli**nb** did in less than 5 minutes:

“They are extremely fast as compared to other classification models As in Bernoulli Naive Bayes each feature is treated independently with binary values only, it explicitly gives penalty to the model for non-occurrence of any of the features which are necessary for predicting the output  $y$ . And the other multinomial variant of Naive Bayes ignores this features instead of penalizing.”

They were all insanely slow and the scores were not much different as our friend over here.

So let's take a look at the results

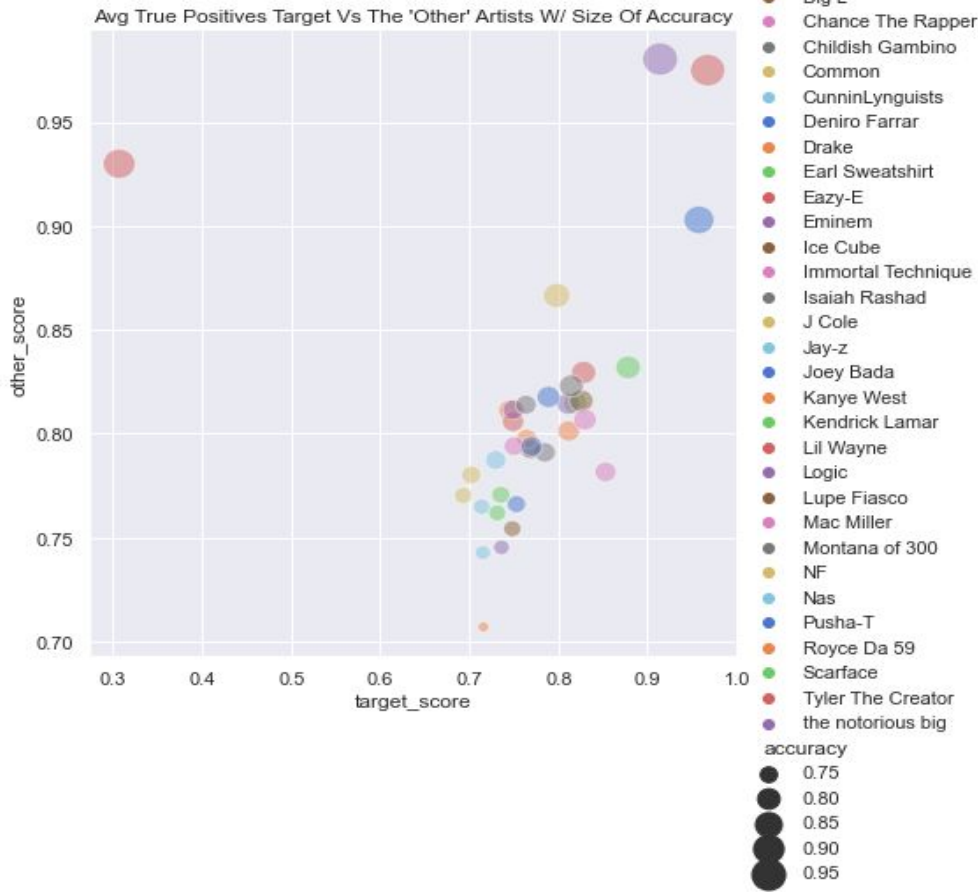
# Bernoli Naive Bayes:

— — —  
Average Accuracy of:  
0.810676(std 0.046174)

Average Target TP:  
0.812641(std 0.044469)

Average Other TN:  
0.807926(std 0.078665)

The advantages were clear it took minutes to run whereas the others take literally hours to go through the files.

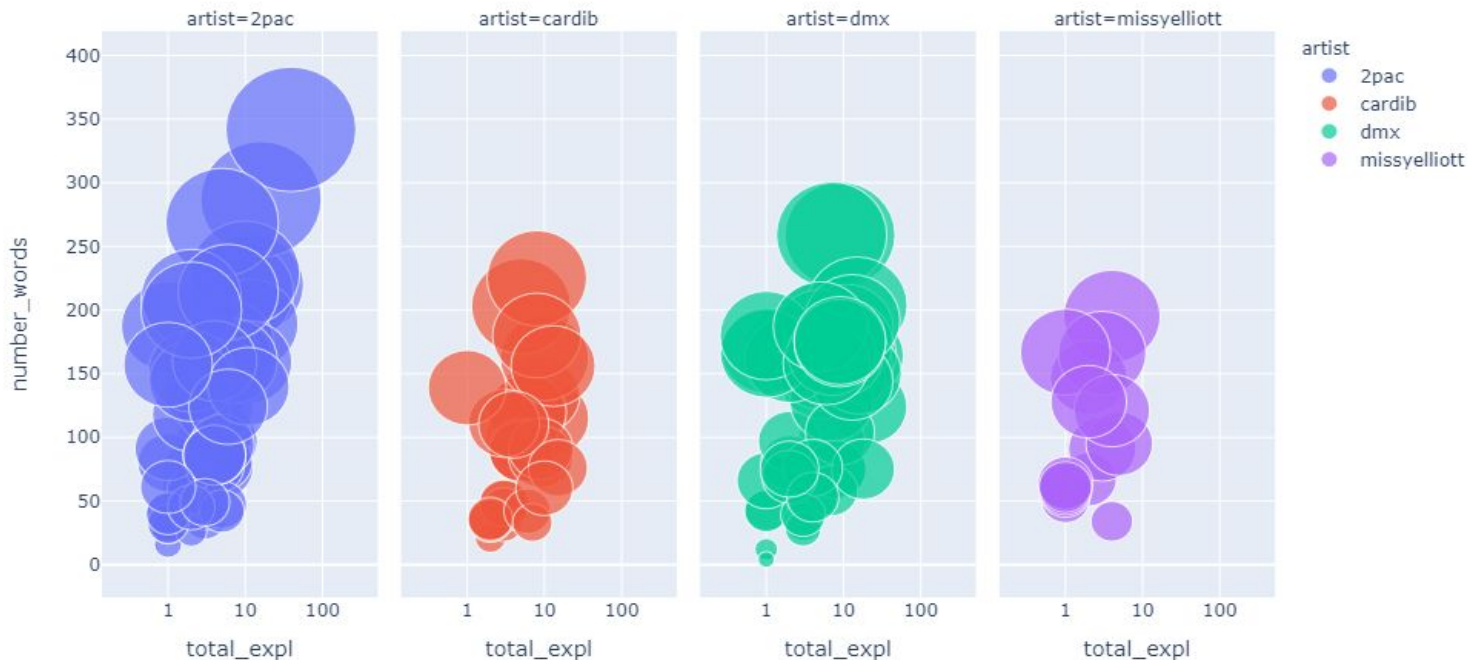


# Observations

## Control group :

— — —

There was little to no effect that they offered in the binary classification however they were immediately visible in the **Multinomial Model** results so much that they skewed the entire thing.

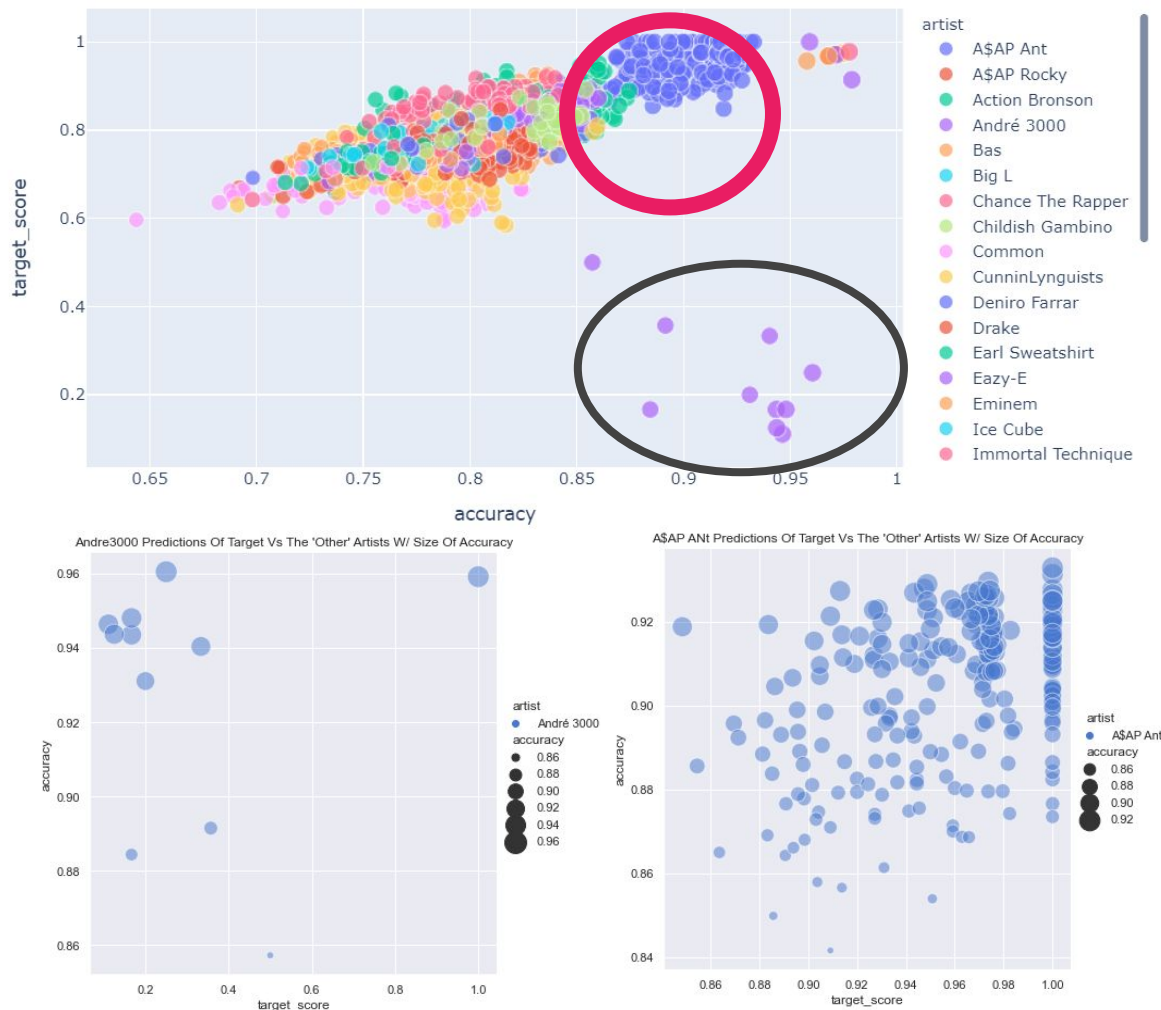


# Observation Clustering: Looking at good results and bad results

It seems that while there was a vast majority of classifications that would support our general question can we predict one artist out of many.

Looking at correlation data that was matched then merged with our earlier EDA was not giving any insight into what were the primary reasons that performance increasing or decreasing.

We have an answer we just did not have a why.

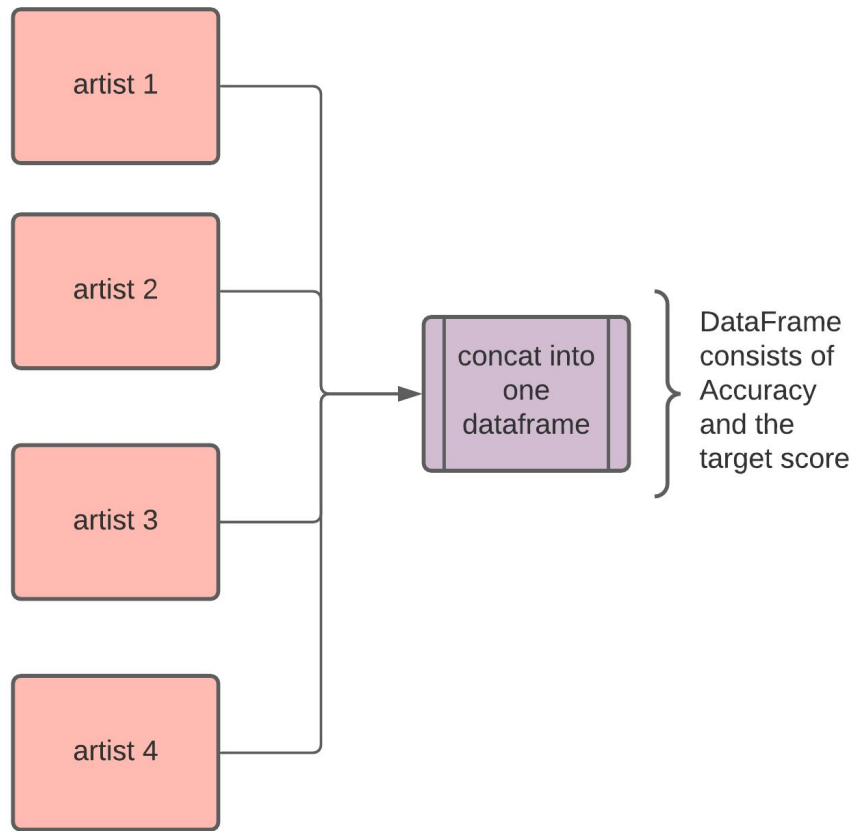


## Second Model: Multinomial NB

---

My goal was to stick with Binary classifications however running into our of slower models. I was looking for more results. So what I did with this Multinomial Naive Bayes is to treat it convert its results to something **slightly unnatural**. Run the model predict for Accuracy and the classifications score.

Again all we want to know is can we predict an artist apart from the others.

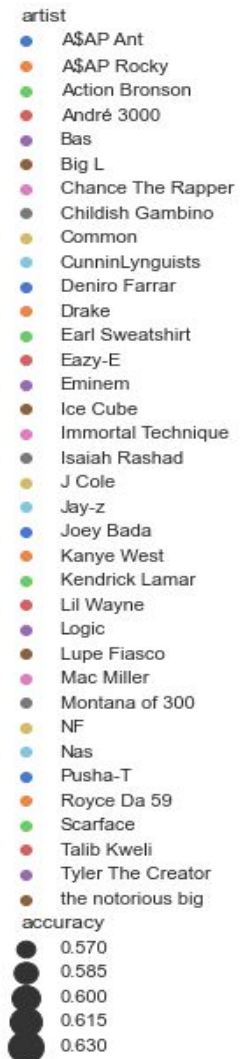
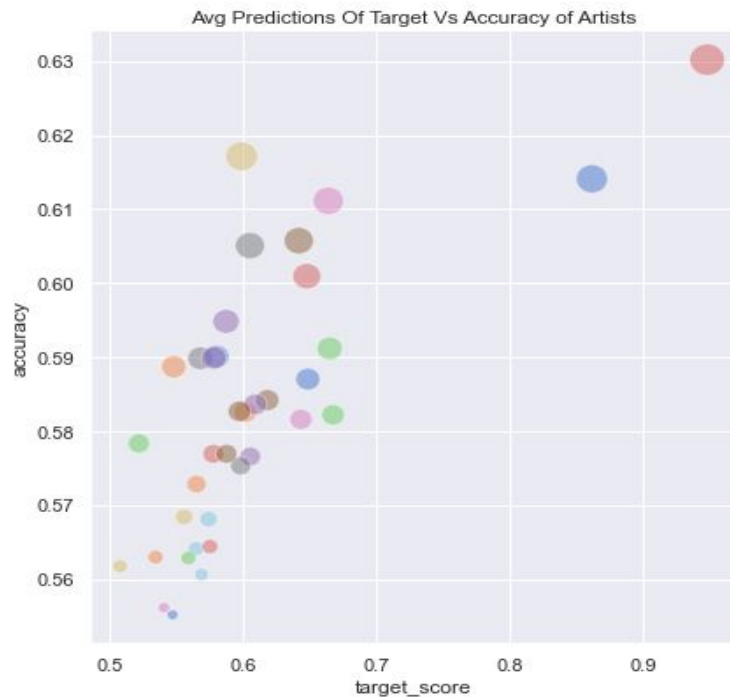


# This was not an apples to apples Comparison

— — —

There were results and considering all things we can see vast reduction in the Average Accuracy(which was expected) and also our under performer had now become our top performer.

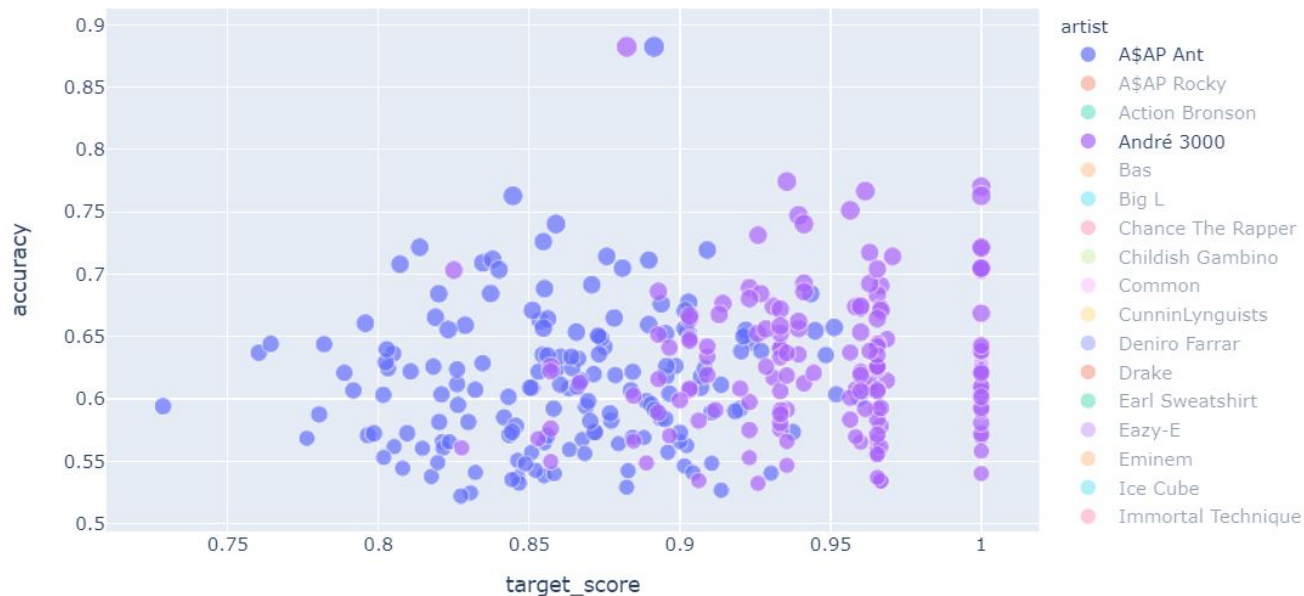
Which was very interesting to say the least.



**Our top performer last time is 2nd and our guy who was dead last is now number one.**

— — —

**We have our initial question answered but the foundation by which it has taken place is not evident nor is it giving us a larger hint as to why in this test?**



# recommendations:

- 1) **Explore a better way storing information:** MongoDB is a great tool — — that works as a great alternative to a relational database and also CSV files. Instead of outputting a hundred files you would execute a queries which are relatively fast depending on your internet connection.
- 2) **Explore different metrics:** Instead of focusing on accuracy explore the **True Positive Rate** and include a deeper analysis with more **Sensitivity**.  
  
**Maybe a more conventional method to try to tie the loose ends.**
- 3) **Explore alternative sources** of text that have a stronger uniformity and less of a variety in dealing with the same “words”.

## 4) Models:

Be warned: **Logistic Regression, Random Forest and Decision Trees** will take a **very long time** not for one iteration of this experiment both for grid search and turning out results. Actually run the models with gridsearch and let them run for a few hours.

One to try:

XGBoost and Tune the Class Weighting Hyperparameter

5) Running each individual sequence through RNN is not far fetched for maybe 20-30 epochs. I would recommend if you do this that you **use MongoDB as it offers a better alternative to clunky csvs**

**The code is written in away where all of this is extremely simple to scale.**



# Conclusion:

- a) We were able to classify artists with at times higher of level of accuracy using both first a binary and then with some modest minimal accuracy with our multiclass classifications
- b) Still can not account for why there is success or lack of success that is taking place. However we are closer to understanding how certain modifications can foster better results.

c) Some of my reservations were justified and in terms of a *real world* applications: I think it is important to have a solid eda and understanding of distinguishes your classes from another. Sometimes you have to take an unorthodox approach.

— — —