

Final Report

Discovery of the recent music trends based on Sentiment  
Analysis in Twitter or Spotify



Cardiff University School of Computer Science and  
Informatics

Author: Darren O'Callaghan

Supervisor: Dr. Daniela Tsaneva

Moderator: Víctor Gutiérrez Basulto

One Semester Individual Project – 40 credits

CM3203

## Abstract

This report investigates the relationship between the commercial performances of musical artists and genres, and the general sentiment around them on Twitter. This involved collecting data from a Spotify managed website; [spotifycharts.com](https://spotifycharts.com), from the Spotify API, and from Twitter using the open-source Twint tool.

My results show that sentiment cannot be used to predict the future performance of any given artist or identify which genres will be popular in the future. It did show that almost all break out commercial performances were achieved by those with positive sentiment. It also showed that large swings in sentiment after the release of a single or album are rare and those that do have them do not perform as well as those releases with more consistent sentiment on release. Where the change in sentiment is relatively low, it is not possible to predict success.

My project shows that tracking sentiment may be useful to those in the music industry to ensure that negative sentiment is not seen over the long term, but it is not particularly useful to measure it directly after a release, or to use it to predict the success of a particular genre- as both popular and less popular genres were seen to have similar average sentiment.

## Acknowledgements

I would like to thank my supervisor Daniela Tsaneva for her help and encouragement throughout this project. I left each of our meetings more confident and excited than when it started, and this project would not be as good without her help.

I would also like to thank my housemates George, Nick, Joe, Jamie, and Rob; for their support and friendship during my time working on this project during the COVID-19 pandemic.

## Contents

Abstract.....	2
Acknowledgements.....	3
Table of Figures .....	6
1. Introduction and Background .....	8
1.1 Introduction .....	8
1.2 Sentiment Analysis.....	8
1.2 Possible Applications of the Project .....	10
2. Project Objectives .....	11
2.1: Goal 1: Identify if positive sentiment leads to longer term success for a given artist. .....	11
2.2: Goal 2: Determine if Negative Sentiment is Truly Bad. ....	11
2.3: Goal 3: Determine the Effect of Neutral Sentiment. ....	11
2.4: Goal 4: Determine if differences exist in Twitter Sentiment around different genres of music.....	12
2.5 Goal 5: Investigate the differences in sentiment across platforms.....	12
2.6: Goal 6: Determine if the same relationships between sentiment and streaming performance hold for extremely popular artists, and those with a smaller following. ..	12
2.7 Goal 7: Determine if tweets that describe sentiment towards an artist but not their music, have the same relationship to commercial performance compared to tweets that are directly about the music. ....	13
3. Implementation/ Solution Overview .....	13
3.2 Spotify Data Collection.....	13
3.3 Twitter Data Collection .....	20
3.4 Natural Language Processing Work .....	22
3.5 Sentiment Analysis Models.....	23
3.6 Sentiment Classification via OpenAi .....	26
3.7 Flask Website .....	27
4. Results and Evaluation .....	29
4.1 Average Sentiment of Artists .....	29
4.2 Change in Sentiment after Musical Releases.....	31
4.3 Differences in Sentiment from Twitter and Spotify.....	33
4.4 Differences in Sentiment Between Musical Genres .....	35
4.5 Website Testing .....	36
4.6 Evaluation of Results.....	37

5. Future Work .....	39
5.1 Automatic Data Collection. ....	39
5.2 Additional Data Collection .....	40
5.3 Use of NLP Pipeline .....	40
6. Conclusion .....	40
7. Reflection on Learning .....	41
7.1 What I have Learned.....	41
References.....	42
Appendix .....	44

## Table of Figures

Figure 1: General NLP Process

Figure 2: Key Steps for Project

Figure 3: "InsertCsv" function created to collect data from [spotifychart.com](https://spotifychart.com)

Figure 4: Example "track" MongoDB document

Figure 5: "createArtistCollection" function: created to collect Artist initial artist information from Spotify API

Figure 6: "addReleases" function: used to add release collect release (album and singles) information from Spotify API

Figure 7: Example "Artist" MongoDB document

Figure 8: First part of "twintTest" function: Used to collect tweets about musical artists.

Figure 9: Second part of "twintTest" function. Used to collect tweets about musical artists.

Figure 10: Example "Tweet" MongoDB document

Figure 11: Confusion Matrix of first Naïve Bayes classifier.

Figure 12: Confusion Matrix of second Maive Bayes classifier.

Figure 13: Confusion Matrix of third Naïve Bayes classifier with reduced smoothing parameter applied.

Figure 14: Confusion matrix of test using OpenAI's "Advanced Tweet Classifier".

Figure 15: Screenshot showing chart of artists spotify streams against their average sentiment, as seen on created website.

Figure 16: Screenshot of graph of Billie Eilish's average sentiment against time as seen on the created website.

Figure 17: Screenshot of Stream Count of releases when in chart against the change in sentiment after the release, as seen on the created website.

Figure 18: Screenshot of website page showing aritst's Spotify Followers to their average sentiment chart, and additional statistics.

Figure 19: Graph of all Artist's respective stream counts against their average sentiment for 2019-2021

Figure 20: "totalReleasesStreams" function: Used to return the stream count of a given release of an artist.

Figure 21: Graph of various release's stream count against the % change in sentiment after the release is published.

Figure 22: Alternaitve view of Figure 21, with outliers removed.

Figure 23: Graph of all artist's Spotify follower totals against the artist's average sentiment

# 1. Introduction and Background

## 1.1 Introduction

The music industry has grown considerably in recent years [28] and has become a major part of many people's lives. One service in particular, Spotify, has been noted as one of the most popular ways to consume music in recent times [29] as it allows users to stream a vast library of songs. In addition, the business of developing and publishing music has become incredibly lucrative, and technology has become a major part in this process. Many artists publish and promote their music through social media, and like almost any topic, their music is discussed by millions of users. For example, at the time of writing, the popular band Coldplay has recently announced their new single "Higher Power", to their 23 million Twitter Followers [1], many of which will have then gone on to tweet about the band and the new single.

As this relationship between music publishing and social media has grown and social media has continued to make up a significant part of where many people spend their time online, the ability to market oneself on social media correctly has also gained greater importance. For example, it has been argued that "social media is an essential part of marketing strategy in online settings..." [2]. Therefore, it can be argued that for musical artists today, their social media presence is a vital part of their success. For many social media sites, these musical acts can track their social media following through several different statistics like followers or comments, both of which have been seen in varying degrees to lead to greater commercial sales of music. [3] My project aims to investigate if another measurement can be used by artists, and those promoting them, to gauge if their music will be a success: sentiment.

## 1.2 Sentiment Analysis

Sentiment is defined as "a thought, opinion or idea based on a feeling about a situation, or a way of thinking about something", [4] and the sentiment of the public or specific groups of people has been examined and used in a variety of different situations. For example, YouGov, the British market research and data analytics firm, tracks sentiment around a range of topics from football,[5] to politics. [6] When this sentiment is extracted from text via various forms of Natural Language Processing, computational linguistics, and text analysis, we refer to this as Sentiment Analysis. [7]

Approaches to Sentiment Analysis can be split into two categories:

1. **Lexicon-based Techniques:** A corpus or dictionary of words with known sentiment is used. New documents are scored depending on how many words they have that are deemed positive or negative by the corpus. This score then can be used to tell if the new document has a positive or negative sentiment.
2. **Machine Learning Techniques:** These are supervised learning techniques where different techniques like Neural Networks, Naïve Bayes Classifiers, Bayesian Networks, or others attempt to classify pieces of text as either positive or negative.



From looking at a couple of example articles presented to me by my supervisor Daniela, I decided to attempt to use a machine learning technique in my project.

As mentioned above, Natural Language Processing is also involved when performing sentiment analysis, particularly when preparing the data to be used by either of the techniques above. In general, an NLP process takes this general form [8]:

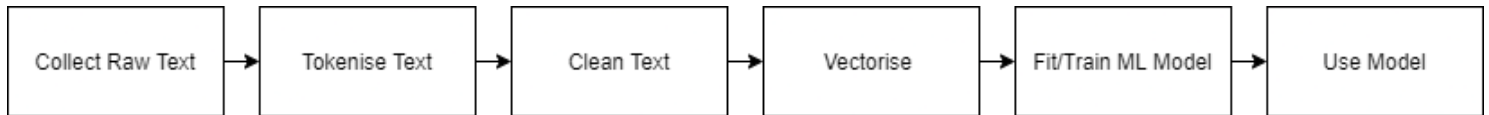


Figure 1: General NLP Process

1. Collect Raw text that will be processed.
2. Split the text into smaller units called “tokens”. These are usually individual words, characters or sub-words or any other way that can “tell” the eventual model what to “look at”.
3. Clean the text, removing unnecessary parts of the text that do not add meaning. These can be “stopwords” like “and”, “a” or “but”, or punctuation.
4. Vectorise the text, converting them into numeric form so that a computer can understand them.
5. Use this vectorised text, and their associated labels, to train a machine learning model.
6. When the model is correctly trained and working well, the model can then be used to apply labels to new text. In our case, with sentiment analysis, this would take the form of “positive” or “negative” classes.

The last two steps are why this process is a supervised learning technique. Supervised Learning is where a set of labelled datasets is first fed to a model, so it can “learn” characteristics of the data in each class. When it has been trained effectively, it should then be able to tell which class a new piece of data belongs to, to at least some degree of accuracy. [9]

It is also important to get a correct balance between correctly identifying which class different datum belong to in the labelled dataset (often called the “testing set” - a subset of the overall labelled dataset used to test the model works correctly), but also making sure that the model does not work so well on the training set that the model is then not effective when presented with new data. If this is the case, it is said that the model is “overfitted”. Of course, the opposite can also occur, called “underfitting”, where the model has not been trained enough and is therefore classifying both data in the testing set, and new data to a low degree of accuracy.

During my initial research, the use of the Bayes Theorem seemed like a promising technique to perform sentiment analysis on data from Twitter. A good example of this is from Goel,

Gautam and Kumar [10], who created a Naïve bayes Classifier, a machine learning method to sentiment analysis, to determine if tweets about the 2016 US Presidential election were positive or negative towards the main candidates in the election. Naïve Bayes is Naïve because it assumes that all features (In our case, a feature would be a unique word) of a class are independent of each other. For each word then in a class:

$$P(Class|Word) = \left( \frac{P(Word|Class) * P(Class)}{P(Word)} \right)$$

And then for a piece of text (like a tweet...):

$$\begin{aligned} P(Class|Word_1, Word_2, Word_3, Word_4) \\ = \left( \frac{P(Word_1, Word_2, Word_3, Word_4|Class) * P(Class)}{P(Word_1, Word_2, Word_3, Word_4)} \right) \end{aligned}$$

In this study, the classifier achieved an efficiency of 58.4% for determining if tweets have positive or negative sentiment, and it will be interesting to see if a similar classifier could be built and adapted to improve this.

Its also been shown by Pichl, Zangerle and Specht [30] that data from Spotify and Twitter can be compatible and used together. In their article, they build a new dataset combining data from both Twitter and Spotify. They used the Twitter Streaming API to search for “nowplaying”, “listento” and “listeningto” which presented many tweets containing information on different artists and songs, as well as Spotify links, which could be followed to extract the artist and song information for further analysis. This emphasised to me the possibility of creating a new dataset, that combined data from multiple sources, particularly from Spotify and Twitter. They were then able to create a limited recommender system from this data. This makes me believe that Spotify and Twitter data should be compared together, to identify musical trends, and see if those in the music industry could gain further insights from determining sentiment about artists and genres.

## 1.2 Possible Applications of the Project

Additionally, I also wanted to attempt to present the data that I collected in a pleasing way, that would emphasise how a tool or system that uses this data could be used by those in marketing within the music industry to better understand the effect of their work on the general sentiment on their artist. For example, in other industries like finance, sentiment is regularly measured and tracked. Some have even created specific indexes that adjust based on perceived sentiment.[11] I was therefore interested in creating a tool that could allow those in the music industry to view and understand sentiment in a similar way. I wanted to create a system that would allow general charts to be “drilled down into”, to allow for further analysis, and for those without a deep understanding of concepts like Sentiment Analysis and Natural Language Processing to be able to obtain insight from the calculated sentiment. This view came from my industrial placement, where I was tasked with creating similar visuals, for other domains.

## 2. Project Objectives

During the initial planning stage of my project, I broke down what I wished to achieve with this project into several specific aims. These aims are mainly focussed on my individual interests around the project and the general questions that I wanted to find answers to when I took on the project.

The initial goals are listed below:

### 2.1: Goal 1: Identify if positive sentiment leads to longer term success for a given artist.

This is the most basic question that I wanted to answer at the start of this project. In general, I wished to answer if we could predict how well an artist performs commercially, if we measure their sentiment on social media. Answering this question leads to additional opportunities going forward. If it is found that there is a relationship between these two variables, then those that market music could use this to understand and possibly predict how successful a musical release would be over a long period, very quickly. This would allow them additional methods to become more effective. For example, an artist could have the marketing resources allocated to them cut, if it can be accurately predicted that their commercial performance would not be worth using them.

Alternatively, if a relationship cannot be found, or is proven to not exist, an argument could be made against the importance of discussion that takes place on social media, and perhaps marketing resources could be spent more effectively elsewhere. For example, an artist may engage consumers that are not active on social media. Their sentiment then does not relate to their performance, and it could be argued that something like Twitter adverts for the artist would be less effective, than something like television adverts which might better engage the consumer. This would also be more effective than just measuring something like follower count- it is possible that someone can have a large social media following, but not engage them.

### 2.2: Goal 2: Determine if Negative Sentiment is Truly Bad.

This was another simple question that came to mind at the start of this project. One of my hypotheses when starting this project was that “all publicity was good publicity”. This goal is essentially a test of this view in today’s modern-day setting. My initial feeling was that if users on Twitter are talking about an artist, they are likely also listening to them. If we can say that negative sentiment can be avoided, this can be a major flag for advertisers, and then either actions can be taken to avoid this (different or more adverts for example), or the artist could be dropped altogether by their management, allowing them to focus on more successful artists.

### 2.3: Goal 3: Determine the Effect of Neutral Sentiment.

This was another interesting question I had, based on the hypothesis, that if an artist has neutral sentiment, I originally thought that it meant that nobody cared enough to talk about the artist. When linked to the idea that “all publicity is good publicity”, this line of thinking would then suggest that this is the worst possible sentiment that social media could have

towards an artist. Like the other goals, the main idea was to gain further insight into sentiment with the hope that those promoting, and marketing music could benefit from having these questions answered.

#### 2.4: Goal 4: Determine if differences exist in Twitter Sentiment around different genres of music.

For many reasons, like to categorise the huge amount of music that exists and help people find music that they will like, the world's music is broken into genres. In general, artists tend to create music in specific genres, and therefore are often associated with it. For example, many artists are labelled "rock bands", "pop stars" or "rappers". If sentiment could be used to greater understand which genre of artists will perform well commercially in the future, it would emphasise the need for the music industry to track it. It could also inform organisations like record labels, which genres could be seen as more "risky" or "safe", allowing them to make decisions (e.g. Which new artist to sign to the label) more effectively.

#### 2.5 Goal 5: Investigate the differences in sentiment across platforms.

Following discussions with my supervisor, we also wondered how sentiment differed between different technology platforms. Twitter's allows users to publish their thoughts on any topic through a "tweet" that can be a maximum of 280 characters. This text can obviously be broken down and sentiment can be extracted from it. Spotify however has no functionality that allows users to present their thoughts. A comment section or blog of some kind is not implemented, but it can be argued that sentiment is expressed in different ways. In Spotify's case, the ability to add songs to a personal "library" through clicking a "heart" icon, is a clear expression of positive sentiment towards an artist. Similarly, users can "follow" artists to keep informed about their latest releases. Since the idea is that users follow their favourite artists, we also argue that "following" an artist is an expression of positive sentiment. We believe it would be informative to investigate if a relationship exists between these different forms of sentiment, so we made this a goal for this project.

#### 2.6: Goal 6: Determine if the same relationships between sentiment and streaming performance hold for extremely popular artists, and those with a smaller following.

Another aspect around twitter sentiment I thought was interesting to investigate was how the relationship between success and sentiment differs when an artist is extremely popular, compared to when they have a much smaller fan base. My hypothesis was that ultimately, there would still be musical artists that have fan bases able to "drum up" support and sentiment around the artist, but the limitations of having a small following would still appear when the commercial performance is found. Therefore, it could be argued that positive sentiment is "easier" to produce than excellent streaming figures. Additionally, if many artists have strong positive sentiment around them, but not the additional stream counts, an argument could be made against using sentiment as a key metric to measure these artists against, and it may not need to be determined after all.

2.7 Goal 7: Determine if tweets that describe sentiment towards an artist but not their music, have the same relationship to commercial performance compared to tweets that are directly about the music.

As stated previously, the sheer number of those that use social media mean that, for the most part, any topic is under discussion. The same can be said for Twitter. When this is viewed around the topic of musical artists, I think it's fair to say that for many artists, not all the tweets will have individual tracks, albums, or musical themes as their subjects. For example, a quick Twitter search for "Ariana Grande", a popular pop star with over 60 million monthly listeners on Spotify at the time of writing, [31] returns tweets about her recent COVID-19 vaccination [32], being spotted in Los Angeles [33], as well as about her music [34]. It would be interesting to identify if the subject of different tweets makes them more "important" to record their sentiment. My initial belief would be that the non-music focused tweets would effectively add "noise" to the sentiment analysis task, influencing the final average sentiment while not leading to a fresh set of streams. Therefore, the tweets about the artist's music are the more important to collect and analyse their sentiment.

### 3. Implementation/ Solution Overview

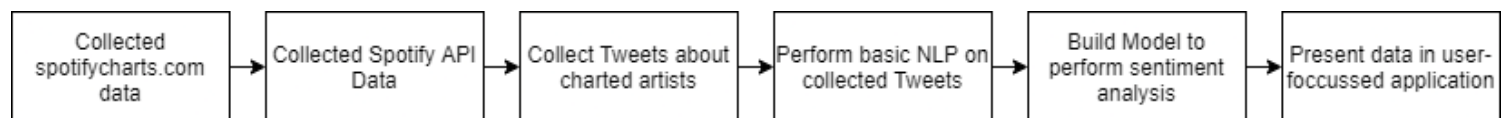


Figure 2: Key Steps for Project

The above diagram details the key steps that I took to complete this project and obtain the results that are presented in section 4 of this report. This section will explain in detail each of these steps.

#### 3.2 Spotify Data Collection

##### 3.2.1 Collecting Data from [spotifyCharts.com](https://spotifycharts.com)

My first task when starting my project was to collect Spotify data that could then be compared against any collected Twitter Data. The official Spotify API does not provide streaming statistics for individual tracks, albums, artists, or genres. However, Spotify does maintain an additional site: [spotifycharts.com](https://spotifycharts.com) [12] that contains the chart rankings for several of their charts. Crucially though, each track has an accompanying stream count for the relevant day or week that is selected. It also has a button that allows the selected chart to be downloaded in a csv format. Since the button is linked to a specific webpage that is written in a specific format, depending on the chart, it was straightforward to create a script that looped through each date in the time frame I selected for a given chart, collecting all the chart information that was provided by the website. The code that handled this is shown below.

```

def collectCsv():
    mongodb = testMongo()
    client = MongoClient('mongodb://127.0.0.1:27017/')

    date = datetime.datetime(2019, 1, 1) # Start at 01/01/2019
    date_str = str(date)[0:10]

    while date_str != "2021-01-01": # Continue the loop until 2021
        print("Pinging chart for date: ", date_str)
        # request the current date's corresponding csv document
        url = f"https://spotifycharts.com/regional/global/daily/{date_str}/download"

        r = requests.get(url, verify=False, stream=True)
        if r.status_code != 200:
            print("request failed")
            print(r.status_code)
            break
        decoded_csv = r.content.decode('utf-8')
        read_csv = csv.reader(decoded_csv.splitlines(), delimiter=',') # Read in the csv.
        csv_list = list(read_csv)
        for row in csv_list[2:]: # start from 2nd row, everything before is headers.
            artist = row[2]
            song = row[1]
            # Record the required artist information,
            # or update with additional chart info if the artist's document exists in the database.
            mongodb.tracksCollection.update_one({
                "title":row[1],
                "artist":row[2]},
                {"$setOnInsert": {
                    "title":row[1],
                    "artist":row[2]
                }},
                {"$push": {
                    "chart":{
                        "date": date,
                        "chartPositions": int(row[0]),
                        "chartStreams": int(row[3])
                    }
                }}, upsert=True)
            date += datetime.timedelta(days=1)
            # Increment the date to collect all the chart data in a daily fashion.
            date_str = str(date)[0:10]

```

Figure 3: "InsertCsv" function created to collect data from spotifycharts.com

The chart that I selected was the “Global 200” chart produced by Spotify, which shows the top 200 songs worldwide based on their stream count for the given day/week. This seemed like the most sensible chart to choose, at least for a first round of data collection.

This Spotify data was saved in a MongoDB Collection called “tracksCollection” and was made of documents that recorded each unique track name, its artist, and for every day that the track featured in the Spotify chart, another subdocument stored the date, chart position and the number of streams recorded for that given day. Each of these subdocuments were stored in a list called “chart”. These three major elements; “artist”, “track” and “chart” make up each unique track that was added to the chart. This was added to my local mongoDB instance using the “PyMongo” module. [35]

Heres an example document:

```

1 { "_id" : ObjectId("6029c7aae0f9d878eb3a53e1"),
2   "artist" : "Arctic Monkeys",
3   "title" : "Do I Wanna Know?",
4   "chart" : [ { "date" : ISODate("2020-11-16T00:00:00Z"), "chartPositions" : 199, "chartStreams" : 647982 },
5               { "date" : ISODate("2020-11-18T00:00:00Z"), "chartPositions" : 198, "chartStreams" : 667540 },
6               { "date" : ISODate("2020-11-19T00:00:00Z"), "chartPositions" : 192, "chartStreams" : 690416 },
7               { "date" : ISODate("2020-12-27T00:00:00Z"), "chartPositions" : 164, "chartStreams" : 720634 },
8               { "date" : ISODate("2020-12-28T00:00:00Z"), "chartPositions" : 161, "chartStreams" : 768351 },
9               { "date" : ISODate("2020-12-29T00:00:00Z"), "chartPositions" : 153, "chartStreams" : 786944 },
10              { "date" : ISODate("2020-12-30T00:00:00Z"), "chartPositions" : 156, "chartStreams" : 802084 } ] }

```

Figure 4: Example "track" MongoDB document

This document is for the Arctic Monkey's song "Do I Wanna Know?", which entered and then moved up the chart slightly towards the end of the time frame for my data collection.

This first data collection task allowed me to track the commercial performances of all of the artists that ever appeared in the chart during 2019-2021, over time. This performance could then be compared to the sentiment of Twitter users over the same period. Overall, this first data collection task led to chart information on 746 Artists who created 2891 different tracks.

I also completed some basic checks to ensure that the collection was created correctly:

Quality Check Description	Check Result	Screenshot
All tracks have an associated chart sub-document	Passed	<pre> &gt; db.tracksCollection.count() 2958 &gt; db.tracksCollection.count({"chart": {\$exists: true}}) 2958 </pre>
All tracks have an associated title	Passed	<pre> &gt; db.tracksCollection.count() 2958 &gt; db.tracksCollection.count({"title": {\$exists: true}}) 2958 </pre>
All tracks have an associated artist	Passed	<pre> &gt; db.tracksCollection.count() 2958 &gt; db.tracksCollection.count({"artist": {\$exists: true}}) 2958 </pre>

These checks made me confident that this process had been completed as expected, and the data collected was ready for use.

However, this data does not provide any information on when the artists released any new music, important for goal 1, and contained no information on the genre of the tracks or artists, which is important information for goal 4. This information would need to be collected from the Spotify API.



### 3.2.2 Collecting Data from Spotify API

I needed to collect three major pieces of information from the Spotify API. Firstly, I needed to be able to link any given track to a particular genre, so that comparisons between genre commercial performance and sentiment can be made. Secondly, I needed to have the release date of specific singles and albums available to determine how the initial sentiment of an artist when the release piece of music relates to its commercial performance over time. Finally, I needed the follower counts for the 746 artists that I had collected data on to this point, so that comparisons between sentiment shown on Twitter and Spotify could be made.

Spotify's API uses a conventional Server-to-Server authentication Code Flow (called a "Client Credentials Flow in their documentation [13]) in which a user obtains a "client ID" and "Client Secret" which is attached to their personal Spotify account. This pair of strings is then used in a POST request which, if the ID and Secret combination is valid, returns a "access token". If this access token is then included in an "Authorization" header to any future requests to the API, that request is accepted, until the access token expires. Finally, this authentication method does not give me access to any endpoints that may return the personal data of either myself, or any other Spotify user.

To complete this task, I first collected the list of artists that featured in the Global 200 chart over the time that I was measuring. Next, I used the Spotify API's "Search API" [14] to collect the Spotify specific Id of each given artist. This was required because every other endpoint of the Spotify API that I needed to use referred to artists through their respective ID, rather than their name. However, the "Search" endpoint did not always return the correct artist as the first result when the exact artist's name is given. To ensure that the correct artist was found, I took the artist that had the most Spotify Followers, and had a name that exactly matched the given search term. Here is an example to clarify this procedure:

For a search of "Arctic Monkeys":

Order in response JSON	Artist Name	Follower Count	Chosen?
1	Arctic Monkeys Tribute Band	110	False
2	Arctic Monkeys	11,433,279	True
3	The Monkeys	1,000	False
4	The Arctics	2,345	False

(Note: the "false" examples are not actual artists)

I think the assumption over taking the artist that has the most followers are a fair one. Since we were looking for artists that appear in the Global 200 Spotify chart- and so at some stage between 2019 and 2021 had a song in the top 200 in the world, I think it is extremely likely that they would have the most follower count of any search using their exact name as a search term.



Each of these artist “objects” that were returned by a given “search” contained the Spotify ID as mentioned, as well as a list of genres that the artist belonged to; and the number of “followers” that the artist has on Spotify. Therefore, I created a second MongoDB collection called “artistCollection” which was made of documents containing four elements: The artist “name”, the list of genres the artist belongs to, the artist’s Spotify ID, and the number of followers that the artist had. Here is the code that collected this initial data for the artist Collection:

```
def createArtistCollection():|
    """This builds the initial artistCollection in Mongo- containing artist name, genres and their spotify ID. """
    mongodb = testMongo("artistCollection2") # Connect to local Mongo ArtistCollection2 (artistCollection had errors)
    allArtists = getAllArtists() # Collect list of all the artists from the MongoDB Database.

    spotify_access_header = spotifyConnectionToken() # Connect to Spotify API

    for idx, artist in enumerate(allArtists):

        if "&" in artist:
            artistName = artist.replace("&", "") # API will exclude any part of an artists name that comes after a '&' in a search- which is bad.
        else:
            artistName = artist

        artist_object = requests.get(f"https://api.spotify.com/v1/search?q={artist}&type=artist", headers=spotify_access_header).json() # Search for the current artist.
        bestIdx = 0 # Uses the first result by default.
        for idx, result in enumerate(artist_object["artists"]["items"]):
            #print("Current best followers = ", artist_object["artists"]["items"][bestIdx]["followers"]["total"])
            #print("Current follower contender: ", result["followers"]["total"], " with name: ", result["name"])
            if result["followers"]["total"] > artist_object["artists"]["items"][bestIdx]["followers"]["total"] and result["name"] == artist:
                #Attempts to get the most popular artist that has a matching name (best assumption that it is the correct artist.)
                print("Idx changed: ", idx)
                bestIdx = idx #followers is the max, and the name is EXACTLY the same as the artist, we use that one.

        bestObject = artist_object["artists"]["items"][bestIdx]
        #print("Object picked: ")
        #print(bestObject)

        genres = bestObject["genres"]
        spotifyId = bestObject["id"]
        followers = bestObject["followers"]["total"]
        artistId = bestObject["id"]
        document = {"name": artist, "genres": genres, "spotifyId": artistId, "followers": followers} # Build the document that will be added to artistCollection.
        print(idx, ": Going into database: ", document)
        update_result = mongodb.insert_one(
            document
        )
```

Figure 5: "createArtistCollection" function: created to collect Artist initial artist information from Spotify API

Of course, this still leaves the information on an artist’s releases to still collect. For this, I used the Spotify API’s “Get an Artist’s Albums” endpoint. This endpoint would return a JSON that contained a list of album objects. I then went through each album, and if it was released in the time I was studying (2019 to 2021), then another “release” document would be added to the “releases” list of sub-documents). This list was part of the original “artist” document explained above. Also, the name of the release also could not be in a list of added release names. This was to prevent re-releases of albums and singles from being both added, for example a “deluxe” and standard release of the same album- I wanted to treat these as a single release. (Using my implementation, the “deluxe” version was likely the one added). The “release” sub-document was made up of a “releaseName”, a “releaseDate”, a “type” (I believed that this would distinguish between albums and singles, but I have only seen “album” types when looking through the database), and the “spotifyId” of the release. If any album had a release date that was not specific to a given day (eg. Was specific only to a given month of a year) then it was omitted. A final point is that I looked through releases

that were released in the United States- not specifying a region at this step was slightly confusing and sometimes came up with duplicates or other weird behaviour. In the end, I decided to take this step to simplify the release collection process; preferring that more releases be left out, rather than accept duplicate or adding releases with issues to my database, and therefore making them part of my analysis.

Here is the code that collected the releases information:

```
def addReleases():
    """Adds all releases by artists that were released between 2019-2021"""
    mongodb = testMongo("artistCollection2") # Connect to Mongo Database.
    allArtists = getAllArtists() # Get a list of all the artists
    spotify_access_header = spotifyConnectionToken() # Connect to the spotify api
    for artist_index, artist in enumerate(allArtists):

        spotifyId = getSpotifyId(artist)
        artist_releases_object = requests.get(f"https://api.spotify.com/v1/artists/{spotifyId}/albums?offset=0&include_groups=album,single&market=US", headers=spotify_acc
        # Using the Spotify ID for the artist collected previously, search up their list of releases.
        albums = artist_releases_object["items"]
        release_count = 0
        skipped_count = 0
        failed_count = 0
        added_releases = []
        for album in albums:
            # Create the 'release' sub-document.
            release = {"releaseName": album["name"], "releaseDate": album["release_date"], "type": album["type"], "spotifyId": album["id"]}
            try:
                release_date = datetime.datetime.strptime(release["releaseDate"], "%Y-%m-%d")
                if release_date > datetime.datetime(2019, 1, 1) and release_date < datetime.datetime(2021, 1, 1) and release["releaseName"] not in added_releases:
                    # Not interested in releases that were not published outside of our timeframe.
                    update_result = mongodb.update_one(
                        {"name": artist}, {
                            "$push": {
                                "releases": release
                            }
                        }, upsert=False)

                release_count += 1
                added_releases.append(release["releaseName"])
```

Figure 6: "addReleases" function: used to add release collect release (albums and singles) information from Spotify API

Here is an example document from my artist collection:

```
{ "_id" : ObjectId("6074cf9c95ebaf0bae5c4981"),
  "name" : "Arctic Monkeys",
  "genres" : [
    "garage rock",
    "modern rock",
    "permanent wave",
    "rock", "sheffield indie"],
  "spotifyId" : "7Ln80lUS6He07XvHI8qqHH",
  "followers" : 11433279,
  "releases" : [
    { "releaseName" : "Live at the Royal Albert Hall",
      "releaseDate" : "2020-12-04",
      "type" : "album",
      "spotifyId" : "7Heaa0B4KOxdWhSICTR2wE" },
    { "releaseName" : "Arabella (Live At The Royal Albert Hall)",
      "releaseDate" : "2020-12-02",
      "type" : "album",
      "spotifyId" : "2LV1P9aqTjSnQT3ELlyoxZ" },
    { "releaseName" : "505 (Live At The Royal Albert Hall)",
      "releaseDate" : "2020-11-19",
      "type" : "album",
      "spotifyId" : "0G37RbbfNQ4zDmM8nn5AeB" } ] }
```

Figure 7: Example "Artist" MongoDB document

This shows that Spotify considers the band “Arctic Monkeys” to be part of the genres “garage rock”, “modern rock”, “permanent wave”, “rock”, and “Sheffield indie”; they have 11,433,279 followers at the time the data was collected; and released 3 pieces of music between 2019 and 2021. Note how the 2<sup>nd</sup> and 3<sup>rd</sup> release are singles with a type of “album”- this wasn’t as useful as I thought it would be in the end.

To recap, after my Spotify Data Collection was complete, I had a MongoDB database with two Collections: tracksCollection: a list of tracks that contained their name, artist, and chart history; and artistCollection, which contained the artist’s name, their music’s genres, and the name and release date of the releases they published during 2019-2021.

One possible question you may have (or at least I thought about after having completed this work) is: why not have a single collection? It was initially simpler to keep these two data collection tasks slightly atomic (I did not want to make a mistake in the second task that led to me having to restart both tasks), but I do think the separate collections make sense. Ultimately, its key to remember that a release is not a track, but a collection of tracks. And due to the limitations of the API, I had access to streaming numbers for individual tracks, but not for whole releases, and I had good metadata on releases, but not tracks. This would mean that, in a single collection, the same release and track information would need to be present, so a single collection would not lead to a similar database- but instead one with larger and more complex documents.

### 3.3 Twitter Data Collection

At the start of my project, I was confident that collecting tweets on a given subject would be straightforward. After all, the studies that I researched and presented in section 2 of this report often made use of the Twitter API to collect their data. However, when looking into the API I decided that it would not be suitable. Its “free” setup would only allow 900 requests in a 15-minute period, and a total of 500,000 tweets per month. This was okay, but I also researched alternatives and found an open-source alternative: Twint.

#### 4.3.1 Collect Twitter Data from Twint

Twint is “an advanced Twitter scraping & OSINT tool written in Python that doesn’t use Twitter’s API...” [15] Instead, it uses Twitter’s search operations to scrape Tweets in a quick and easy way. Twint has also been used to collect data from Twitter in other cases. [36, 37] For my use, I decided that a simple search of each artist’s name and collecting the tweets that this returned would be a good enough starting point for my project. However, Twint is, by default, set up to return the latest tweets. While a time frame can be specified for the tweets, they also come back chronologically and rapidly. For example, specifying a search for 100 tweets over the time frame of 2019-2021 would return lots of tweets from near midnight on 1<sup>st</sup> January 2019. To create a relatively even spread of tweets, I decided to make lots of repeated searches for the same artist, while incrementing the dates after every search. This led to me requesting 50 tweets for each 14-day period across 2019 through to 2021. This created a good spread of tweets across the entire time frame and allowed an average sentiment at any point to be taken.

Again, I decided to store the tweets that I was collecting in a fresh MongoDB collection. However, Twint has no way to directly connect to a MongoDB instance, so my solution had to work around this. Twint did have a setting that would directly output the results of a search to a csv file, so I decided to use this. My script that completed the data collection would loop through the 746 artists that appear in the Global 200 Spotify chart; complete searches for the artist through Twint at regular time intervals; and save the results of each search by appending to a single csv per artist. When the searches reached 2021, the contents of the csv would then be reread by the script, and added to the Mongo Collection, the csv for that artist would then be deleted, and the process would begin again for the next artist in the list.

Here is the code that completes this loop:

```
def twintTest():
    mongodb = testMongo()
    artistList = getAllArtists()
    print(len(artistList))
    startDate = "2019-01-01"
    finishDate = "2021-01-01"

    # Twint Setup.
    c = twint.Config()

    c.Store_csv = True # This setting puts any Twint results automatically in a local
    c.Lang = "en"
    c.Hide_output = True
    c.Email = False
    c.Phone = False
    c.Output = "twint3.csv"
    c.Limit = 50
    c.Count = True
    stopwords_list = set(stopwords.words("english"))
```

Figure 8: First part of "twintTest" function: Used to collect tweets about musical artists

```
stopwords_list = set(stopwords.words("english"))
for artist in artistList: # Loop through the artists.
    print("Searching tweets for: ", artist)
    c.Search = artist # Set the artist's name as the search term for Twint
    cleaned_artist_name = re.sub('[^a-zA-Z\s]', '', artist) # Just used for csv name, NOT search
    c.Output = f"twint-{cleaned_artist_name}.csv"
    currentSinceDate = startDate
    currentUntilDate = "2019-01-14"
    while datetime.datetime.strptime(currentSinceDate, "%Y-%m-%d") < datetime.datetime.strptime(finishDate, "%Y-%m-%d"):
        c.Since = currentSinceDate # Adjusting the date range for the twint search in every loop.
        c.Until = currentUntilDate
        print("Searching for artist: ", artist, " between the dates: ", currentSinceDate, " and ", currentUntilDate)
        twint.run.Search(c) # Actually running the search here.
        currentSinceDate, currentUntilDate = incrementDates(currentSinceDate, currentUntilDate)
        # Incrementing the dates here. (Just uses a delta of 2 weeks and 1 day.)

    with open(f'twint-{cleaned_artist_name}.csv', 'r', encoding="utf8") as csv_file:
        csv_reader = list(csv.reader(csv_file, delimiter=','))
        # Read in the newly created csv.
        for row in csv_reader[2:]: # start from 2nd row, everything before is headers.
            subject = artist
            song = row[1]
            tweet = row[10].lower()
            # NLP performed before the tweet is added to the database.
            cleaned_tweet = re.sub("[^a-z\s]", "", tweet)
            tweet_no_stopwords = " ".join([i for i in cleaned_tweet.split() if i not in stopwords_list])
            tweet_no_links = " ".join([i for i in tweet_no_stopwords.split() if "http" not in i])
            # Add to the database.
            mongodb.tweetsCollection2.insert_one({
                "subject": subject,
                "text": tweet_no_links,
                "Date": datetime.datetime.strptime(row[3], "%Y-%m-%d")})
        # Remove the current artist's csv- its data is now stored in the mongo.
        os.remove(f"twint-{cleaned_artist_name}.csv")
```

Figure 9: Second part of "twintTest" function. Used to collect tweets about musical artists

The new MongoDB collection, "tweetsCollection" contains documents that contain a "subject" (the artist that was searched to find the tweet), the "text" of the tweet, and the date that the tweet was published. After collecting all the tweets for the 746 artists in this way, I had collected 3,361,961 tweets, that were ready to put through a sentiment classifier.

Here is an example document from this collection:

```
{ "_id" : ObjectId("60366e04fba237940dbb6295"),  
  "subject" : "Arctic Monkeys",  
  "text" : "imagine harry styles covering arctic monkeysi think would die",  
  "Date" : ISODate("2019-01-13T00:00:00Z"),  
  "sentiment" : "4" }
```

Figure 10: Example "Tweet" MongoDB document

Here we can see a tweet that has been collected about the band “Arctic Monkeys”, that was published on Twitter on 13<sup>th</sup> January 2019, and it seems the author wants Harry Styles to sing some Arctic Monkey’s songs. (This document also contains the sentiment, which was calculated later, and is explained below.)

With these three MongoDB collections, tracksCollection, artistCollection and tweetsCollection, I now had the data required to attempt to answer some of the questions that I had strived to answer at the start of my project.

### 3.4 Natural Language Processing Work

To effectively derive sentiment from the over 3 million tweets that I had collected, some Natural Language Processing would need to be completed on the tweets. I also decided to keep my Natural Language processing as simple as possible. My main Natural Language Processing took place when writing the tweets collected from a search from the temporary csv file to the MongoDB database. Firstly, I “cleaned” the tweet by matching any non-alphabetical characters with a regular expression and replacing them with an empty string. Next, I used the popular python package nltk [16] to import a standard set of stopwords, using this against each tweet in the csv file, and removing any that were found in this tweet. Finally, after some initial testing on this simple NLP procedure, I decided to add a final step that would attempt to remove any links that had been added to a tweet, since like stopwords, they do not add any sentiment. To do this, I removed any words in each tweet that contained the substring “http”. I think it is fair to assume that in the vast number of cases that these words would be a link of some form.

During my work on collecting data from Twitter and performing this NLP, I did investigate using the Spacy python package [21] which enables the creation of NLP pipelines that can tag certain parts of sentences, detect and label specific entities, perform lemmatisation and more. Ultimately, this was too costly in time to create, however I do discuss how creating and using a pipeline which includes functionality that Spacy provides could be a next step to improving my project in section 5 of this report. Ultimately, I think the simple steps that I did take to reduce the noise of the data that I was collecting was worthwhile, especially since they were relatively simple to add to my data collection script. Of course, it is hard to tell the exact affect these steps would have had without going through the sentiment analysis creation process with both the untouched noisier data, and the data that I went on to use. However, I think it is fair to say that the process I used reduced the total tokens that would be present across my dataset, speeding up the sentiment labelling process of an eventual model. Additionally, steps like stopword and removal of weblinks prevent the



possibility that those tokens would be mistakenly seen as implying some form of sentiment, when in truth they suggest none, which led to more accurate sentiment labelling.

### 3.5 Sentiment Analysis Models

With a clean set of tweets collected on all the artists present in the Spotify Global 200 chart throughout 2019 to 2021, it was now time to create a model that could assign a level of sentiment to each tweet. During my research into Sentiment Analysis described in section 2, some of the work that I studied [10] trained models that were used for classifying tweets used the Sentiment140 dataset [18]. This dataset contains 1.6 million tweets that have been labelled as either positive (labelled with a “4” for sentiment), or negative (labelled with a “0”) for sentiment. Contrary to what is noted in the dataset’s description [19], no tweets were labelled to have “neutral” sentiment.

With this dataset downloaded, I then split it into two, creating a training set which would contain tweets that would be initially “given” to the model to allow it to understand the characteristics of positive and negative tweets; and a testing set, which would then be passed to the created classifier. The quality of the classifier could then be assessed, by comparing the sentiment label that the classifier assigns to these tweets, and their “true” label, that was downloaded as part of the dataset. I decided to make the training set be composed of 80 percent of the whole dataset, with the testing set made up of the remaining 20 percent. The splitting of the dataset into these two new sets was completed using the “train\_test\_split” function [20] that is part of the sklearn model selection python module. This split was also completed randomly, by setting the function’s “shuffle” parameter to “True”.

Next, I had to decide which type of model to use. Again, similarly to the article mentioned previously, [10] I decided to first use a Naïve Bayes classifier as a first attempt, to see if I could match what that report had produced.

I then investigated the best way to implement a Naïve bayes classifier, and the implementation available from sklearn looked relatively straightforward to use since it had an option to repeatedly feed a classifier object chunks of data, using its “partial\_fit” function. [22] It was also obvious, even with just basic understanding of machine learning that it was unlikely that the model was going to be able to receive 1.28 training set tweets in one go (a quick test also confirmed this). I had also come across sklearn during my industry placement, which helped to quickly identify a starting point for training a model.

To feed a classifier training data, the tweets had to first be vectorised. To do this I used sklearn’s “CountVectorizer” object [23]. When a group of documents is passed to this object, it returns a matrix that describes how many of each token is in each document. Here is an example:

Tweet 1: “Arctic Monkeys are the best”.

Tweet 2: “Arctic Monkeys are talented”.

When we vectorise, these tweets using the “CountVectorizer” module, we get a vector like this:

Words (Features)	Arctic	Monkeys	are	the	best	talented
Tweet 1	1	1	1	1	1	0
Tweet 2	1	1	1	1	0	1

This was then converted to a standard python array with the standard “toArray” function, and then passed to the classifier, through the partial\_fit method mentioned previously. The vectoriser was also passed a vocabulary of words that contained all the words that were in all the entire Sentiment140 set of tweets, minus any stopwords. This would mean these stopwords would not be represented in the resulting matrix, so they would have no effect. My script then does this for all the tweets in the training set, in batches of 1000 tweets. Finally, the finished model is “saved” by being serialised by the “pickle” module [24], which allowed me to keep the classifier object on disk for later use, in the form of a .pkl file.

For my first attempt, I used the Gaussian Naïve Bayes Classifier, which works best for features that are real values along a continuous distribution [38]. At this point, I was trying to get my head around creating Sklearn models, its only in hindsight that I realise that using this type of model, on data like mine which is not continuous was ineffective. This was shown in my results, which achieved a poor accuracy. Here is the confusion matrix for that classifier:

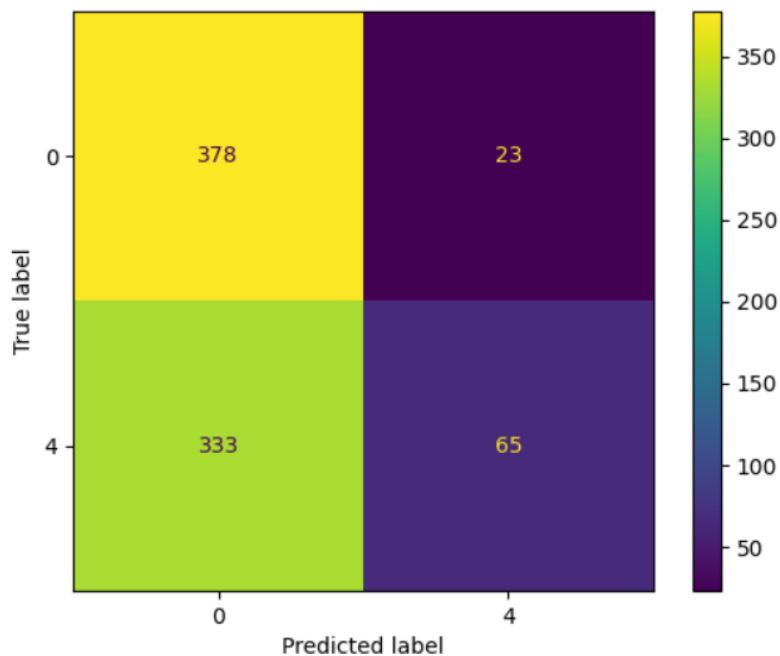


Figure 11: Confusion Matrix of first Naïve Bayes classifier

We can see that this classifier does not perform very well, as it heavily favours classifying text as negative (“predicted label” of 0- the left side of the confusion matrix), which leads



many tweets that have been labelled as positive in sentiment to be misread by the classifier as negative. Therefore, I was not confident in this classifier for use on my collected tweets.

Next, I decided to try the Bernoulli variant of the Naïve Bayes classifier that is also part of the sklearn package [25]. This assumes that the features presented to it are labelled with binary labels, which works perfectly with our use case, since our labels are either positive or negative (referring to the sentiment implied by the tweet). This second classifier worked for better, as seen by this confusion matrix.

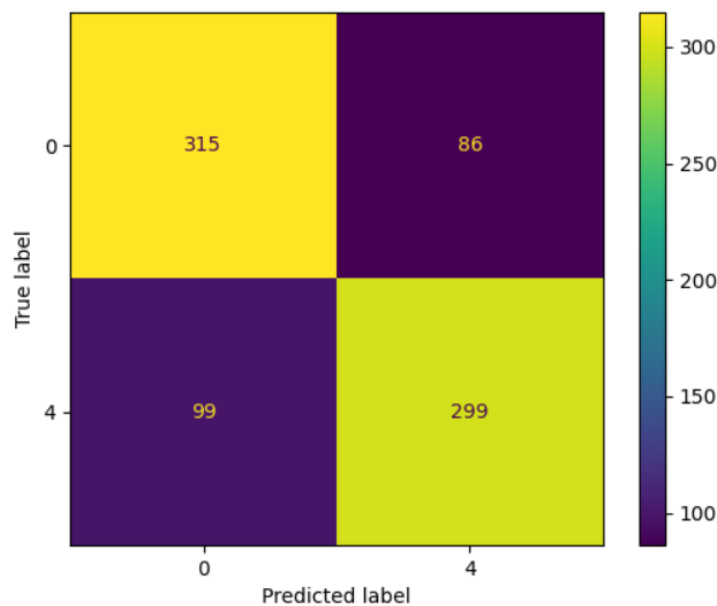


Figure 12: Confusion Matrix of second Naïve Bayes classifier

This classifier is a clear improvement on the first classifier, with an accuracy of 76.8%. Importantly, it also seems like the incorrectly classified tweets are well balanced between positively and negatively labelled tweets.

For another attempt, I also made tweaks to my second Bernoulli classifier, by changing smoothing parameter, reducing it slightly. However, as shown below, this made very little difference in another small test:

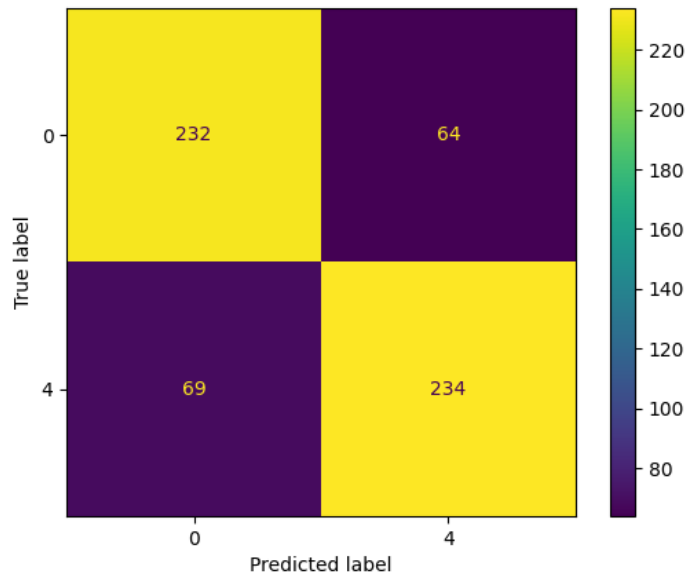


Figure 13: Confusion matrix of third Naïve Bayes classifier with reduced smoothing parameter applied. "

Ultimately, I was happy with the first Bernoulli Classifier that I had trained, particularly because the accuracy that it reported back was higher than some of the classifiers created in the articles that helped inspire this project. I think continuing to work on more classifiers would have been detrimental to the project. There was the possibility that striving for greater accuracy and precision regarding the sentiment140 would lead to the classifier overfitting for that dataset, and being less effective for my tweet collection, which was likely to be a little messier (see section 5.3), but the main reason was the sheer time that it would take to train each of these classifiers. At this stage, I also had the labelling of my far larger dataset to complete as well, so I decided to continue with this classifier.

### 3.6 Sentiment Classification via OpenAi

Finally, I also wanted to spend some time looking into how the new OpenAi gpt-3 machine learning engine [39] could be used to determine sentiment of tweets. This was possible as I had been allowed access to the OpenAi API while working on this project. To use this API, my account was given an "organisation key" and an API key, which I had to provide when using the OpenAi Python package [40].

At the time I was investigating OpenAi, they had an "advanced tweet classifier" preset, that would return sentiment labels for different tweets, when some tweets and their sentiment labels were previously provided. The OpenAi playground also shows how to use this in a python script. I then adapted this, so that I could read in tweets from my MongoDB collection, and piece them into the preset that OpenAi provided.

Therefore, I ran the preset with 10 chunks of 10 tweets, giving me a testing set of 100 tweets. Ultimately the results were much more disappointing:

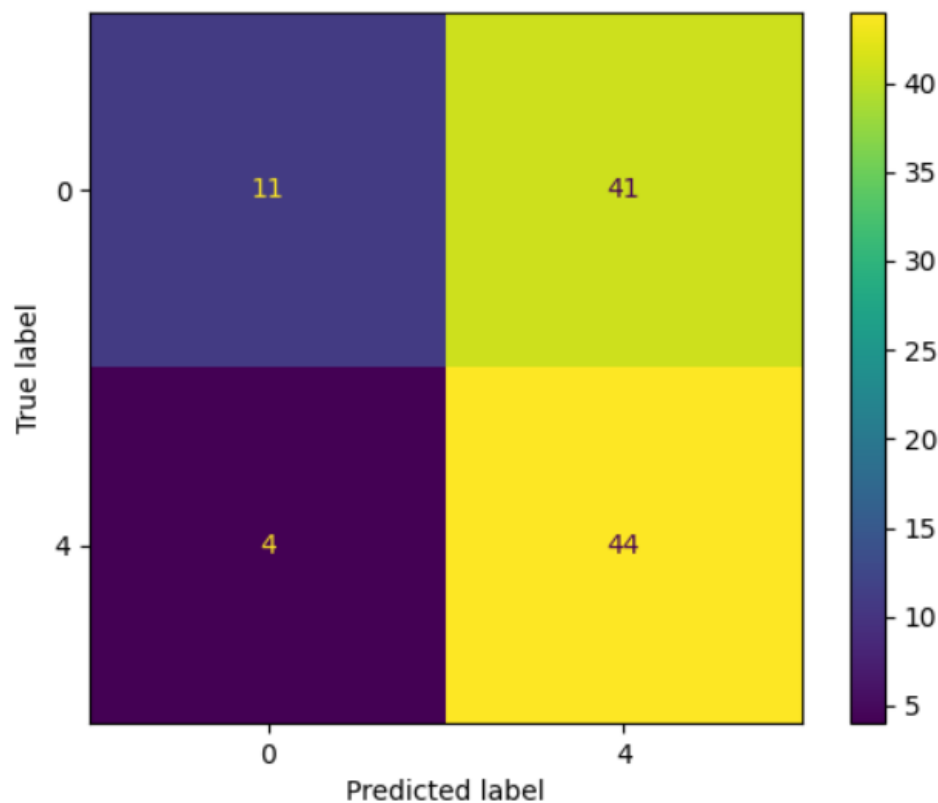


Figure 14: Confusion matrix of test using OpenAI's "Advanced Tweet Classifier"

Interestingly, this seemed to perform in an “opposite” fashion to my first classifier, as the OpenAI prefers to classify tweets as Positive, even when this is incorrect. This shows that a “general” solution with little setup and specific training seems to not be very effective when the subject is around musical artists. However, due to a lack of time, and because of the costs of using this classifier for the number of requests that I required, I didn’t investigate this method further, preferring to work with the ski-learn package instead. With more investigation, it could be shown that this method is a viable method to classify tweets.

### 3.7 Flask Website

A final piece of work that, after discussions with my supervisor, we decided would be interesting, was to build a website that could be used to present the finding of my work. Originally, the plan was to just present the charts and findings that I discuss below in section 4. However, I decided to expand this vision, to show that the data that I have collected and created can be used by social media managers and those in the music industry to gain insight into popular artists, and trends that are playing out on Twitter.

I decided to use create a Flask server, that would contain some of the code that I had wrote during my previous work, as well as html templates that would present my results and work in a pleasing way. For the full Here is a breakdown of the website:

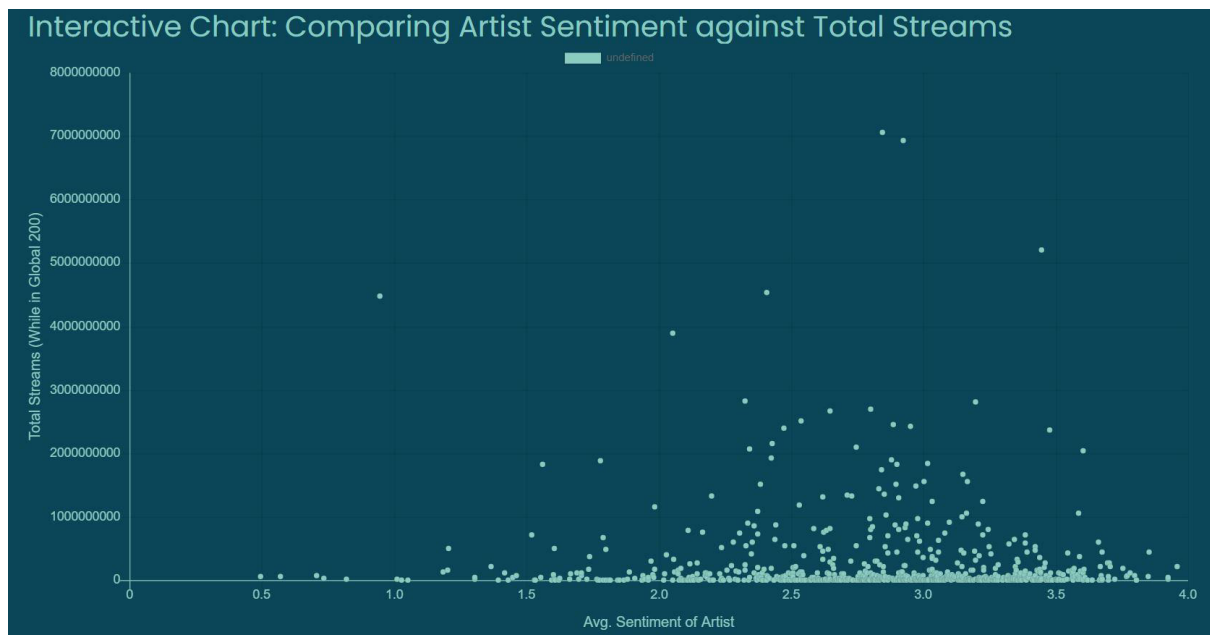


Figure 15: Screenshot showing chart of Artist Spotify Streams against their average sentiment, as seen on created website.

The home page (after a brief introduction) shows the chart of each artist's total stream count (while in the Global 200 chart) against their average sentiment across the entire time investigated (These charts are detailed further below). I also added functionality that allows each data point to be clicked to show the full sentiment chart for that artist. For example, here is the chart for Billie Eilish (the datapoint with the greatest stream count) which is presented when her point in the chart above is clicked:

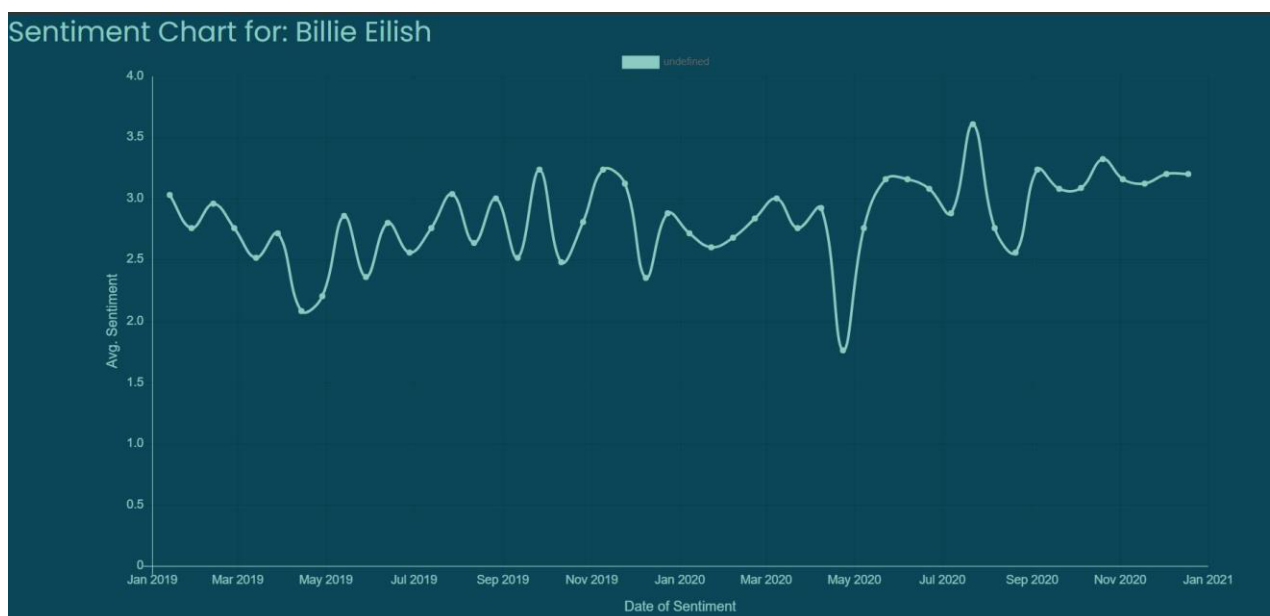


Figure 16: Screenshot of graph of Billie Eilish's average sentiment against time as seen on the created website.

I also included the charts that show the change in sentiment after releases, and the charts for various genres (again, the process for creating these are detailed in section 4):

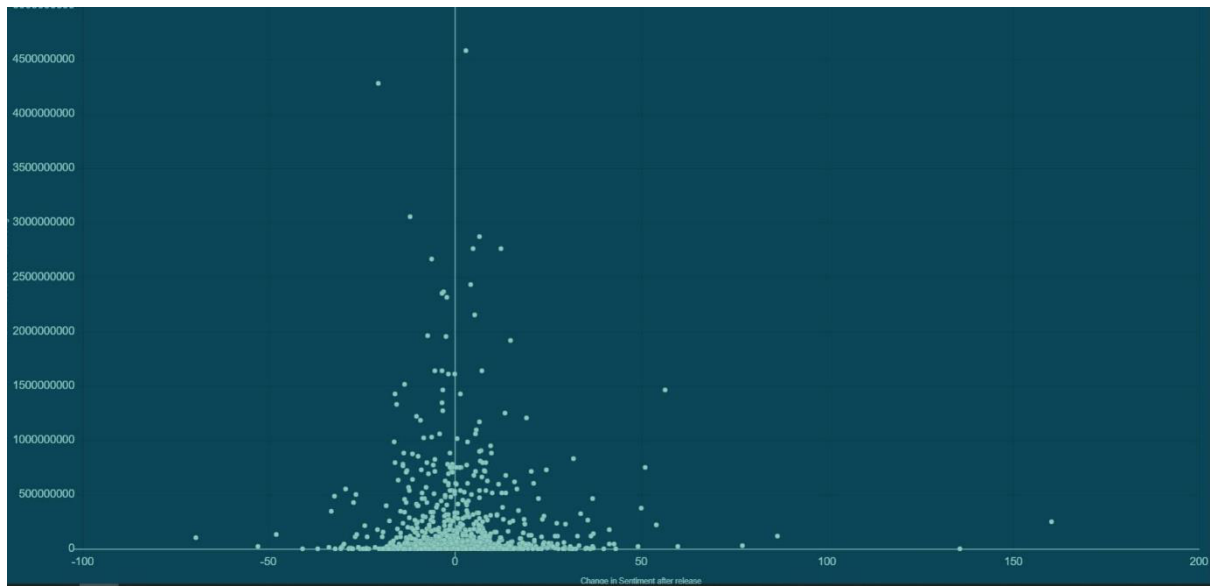


Figure 17: Screenshot of Stream Count of releases when in chart against the change in sentiment after the release, as seen on the created website.

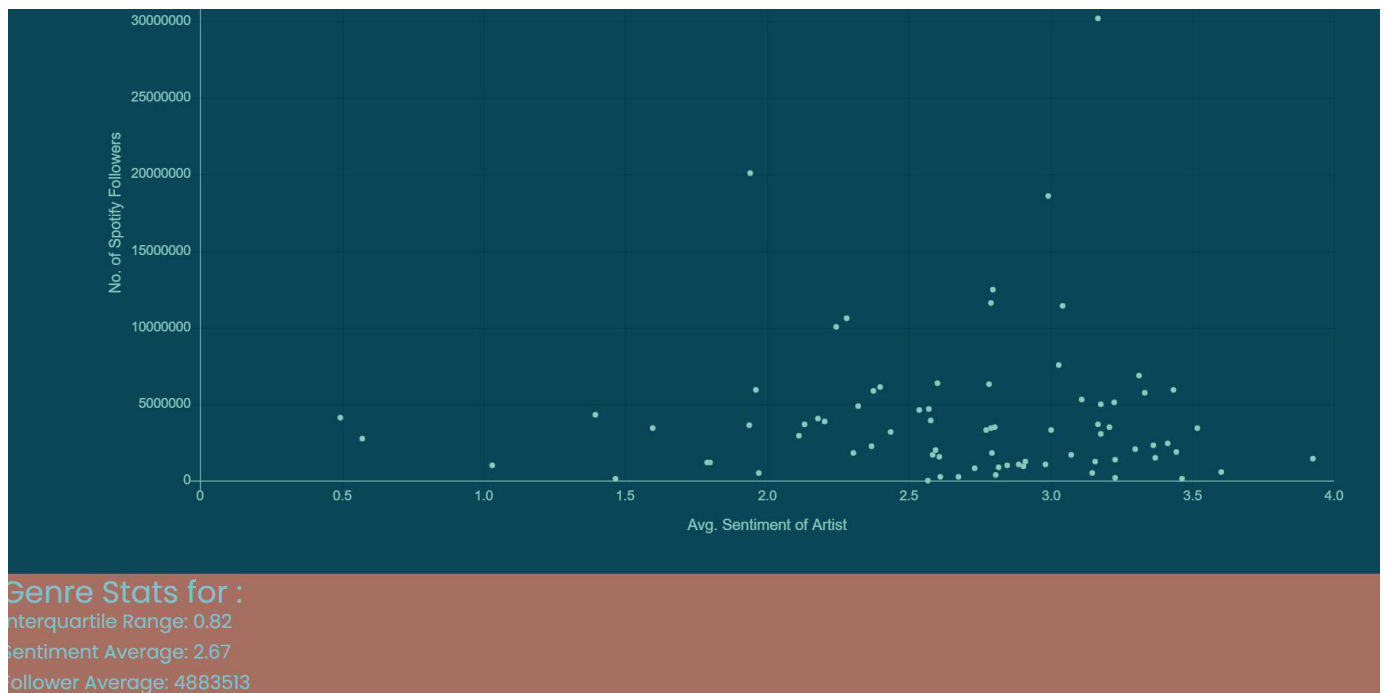


Figure 18: Screenshot of website page showing artist's Spotify Followers to their average sentiment chart, and additional statistics.

## 4. Results and Evaluation

### 4.1 Average Sentiment of Artists

Firstly, I decided to look at the average sentiment of each artist that appeared in the Global 200 chart between 2019 and 2021, with the hope being to find some patterns that would help to gain more understanding around Goal 1 and 2 of my project. Firstly, I used a MongoDB aggregation to make an ordered list of the artists based on the average sentiment of all the tweets about them that I collected:

Sentiment Ranking	Artist Name	Avg. Sentiment (3sf)
1	Rich Music LTD	3.96
2	Daryl Hall & John Oates	3.93
3	Banda MS de Sergio Lizárraga	3.92
4	Nio Garcia	3.90
5	Ufo361	3.85
6	Otis Redding	3.80
7	Nat King Cole Trio	3.80
8	Vedo	3.80
9	Bing Crosby	3.78
10	Peach Tree Rascals	3.78
11	Darlene Love	3.77
12	Sam Feldt	3.75
13	Ella Fitzgerald	3.72
14	Robin Schulz	3.71
....	....	.....
737	Band Aid	1.18
738	True Damage	1.05
739	Pretenders	1.03
740	Bryson Tiller	1.01
741	Bad Bunny	0.946
742	The Cratez	0.819
743	YoungBoy Never Broke Again	0.734
744	Why Don't We	0.708
745	Passenger	0.570
746	The Police	0.495

(For the full list, see Appendix 1)

Immediately, a listing like this tells us a few things. Firstly, it is likely that those artists on the very end of either side of sentiment like “The Police” or “Rich Music LTD” are almost certainly outliers. These may have occurred due to not collecting a large enough sample of tweets for these artists, but in some cases I would argue that it is because searching some artists will return tweets that are not necessarily about the musical artist in question or their music, but other topics instead. “The Police” is a good example of this, as I would suggest that searching this term would have produced tweets discussing topics around law enforcement, rather than the English Rock Band formed in 1977. [26] Another example, “Bad Bunny” has a quite negative sounding word (“bad”) in their name, which may have influenced the classifier to classify tweets incorrectly.

Next, I decided to compare these average sentiment values with the total stream count of the artists while they were in the Global 200 chart:

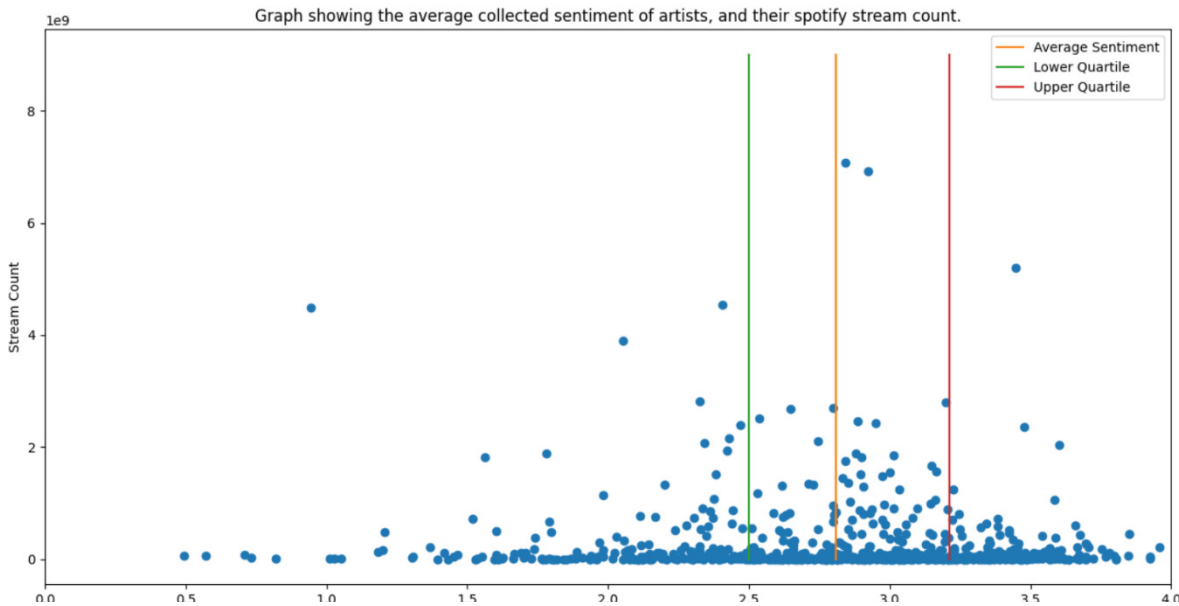


Figure 19: Graph of all Artists' respective stream counts against their average sentiment for 2019-2021

This graph shows quite a mixed outlook between average sentiment and musical performance. From just looking at the graph, it would be easy to suggest that those artists that went on to have breakaway musical success also tended to have positive sentiment on twitter during that time (at least greater than neutral sentiment at 2). However, the average sentiment of those artists that have over 1 billion streams during the period studied was 2.715. The overall average however was 2.810. The median of the whole dataset is also larger than the more commercial successful subset, with medians of 2.897 and 2.842, respectively. This would suggest that sentiment cannot accurately predict the commercial performance of an artist on its own. This does make sense, since a small artist could create a positive sentiment from twitter. My collected data does suggest that sentiment on Twitter is largely positive, which was a surprise given the discourse around Twitter and Social Media over the last few years. [27] This graph does show that those artists with lower sentiment (although far smaller in number) very rarely achieve major commercial success.

## 4.2 Change in Sentiment after Musical Releases

Sentiment over a long period of time may not have a relationship between the number of streams over the same period, but what about when sentiment changes directly after a release? To investigate this, I took the average of the sentiment around an artist 30 days before the release date of their releases during 2019-2021, and then again for the 30 days after the release. I then calculated the percentage change in sentiment using the following calculation:

$$\text{change in Sentiment} = \left( \frac{\text{30 days after sentiment}}{\text{30 days before sentiment}} * 100 \right) - 100$$

I also had to do some additional work to work out the streaming totals for the releases, since the spotifyCharts website only presents streaming count at an individual song level. To

do this, I went back to the Spotify API, and used their “Albums API” to get a track listing for each release that each artist published during 2019- 2021 (these releases being the ones we previously collected and stored in the Artist MongoDB Collection). After collecting the track listing for a given release, we could then go through each track and search for it in our tracks MongoDB Collection. If I found it, I recorded its stream count, if not, its stream count was recorded at zero. All the stream counts for all the tracks in a release were added together, to give us the release’s streaming count. Here the code for this process:

```
def totalReleaseStreams(artistName, albumName):
    """Returns the total streams for all the tracks by a given album,
    by collecting all the tracks by the api, and adding their stream counts together """

    spotify_access_header = spotifyConnectionToken() # Connect to Spotify API
    artist_id = getSpotifyId(artistName)
    track_listing = []
    stream_count = 0

    artist_albums_object = requests.get(f"https://api.spotify.com/v1/artists/{artist_id}/albums", headers=spotify_access_header).json()
    # 1. We get the releases that the artist has released.
    for idx, album in enumerate(artist_albums_object["items"]):
        current_album_name = album["name"]
        if current_album_name == albumName: # 2. If the current album were looking at matches the name given we get that track listing
            album_id = album["id"]
            tracks_object = requests.get(f"https://api.spotify.com/v1/albums/{album_id}/tracks", headers=spotify_access_header).json()
            for track in tracks_object["items"]:
                track_listing.append(track["name"])
            break

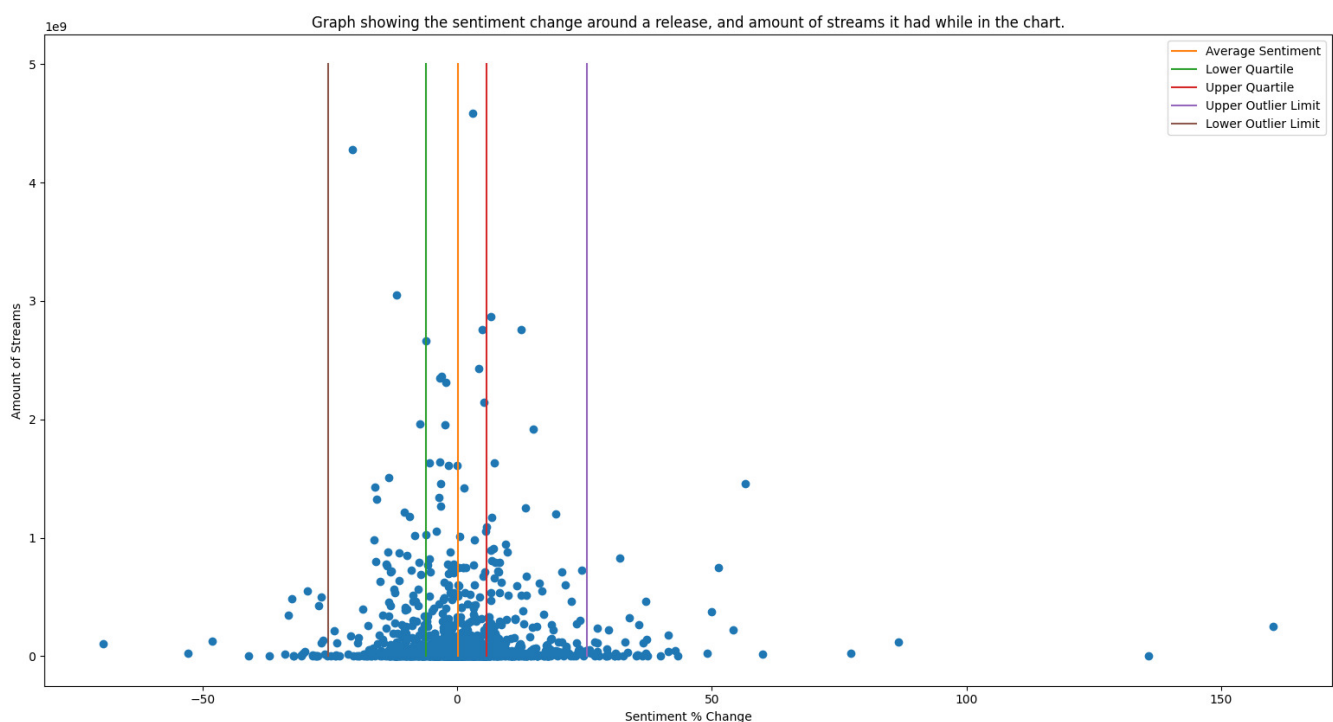
    for track in track_listing: # We then look for each track in our database and add the total streams for it to the total for the release.
        stream_count += totalTrackStreams(artistName, track)

    return stream_count
```

Figure 20: “totalReleaseStreams” function: Used to return the stream count of a given release of an artist

Additionally, unlike the other charts in this report, I saved the x and y values in a csv file, as it takes quite a bit longer to generate this chart compared to the others I have produced. The use of csv file is also used by the website detailed earlier in the report.

Using this process gave me these results:





These results go against my initial hypothesis (that high performing releases would cause a bit of a sentimental wave on social media, either negative, controversial fashion, or in a positive way). The results instead show that big swings in sentiment are very rare and almost always are some of the poorest performers on the entire chart. In this case, lots of releases were very far away from the mean sentiment change (either  $LQ - 1.5 * IQR$  or  $HQ + 1.5 * IQR$  away) and could be considered outliers in this dataset. 77 releases, making up 6.63% of all releases were past the limits, which are present on the graph above. If we ignore these releases, and effectively “zoom” in on the main bulk of data...

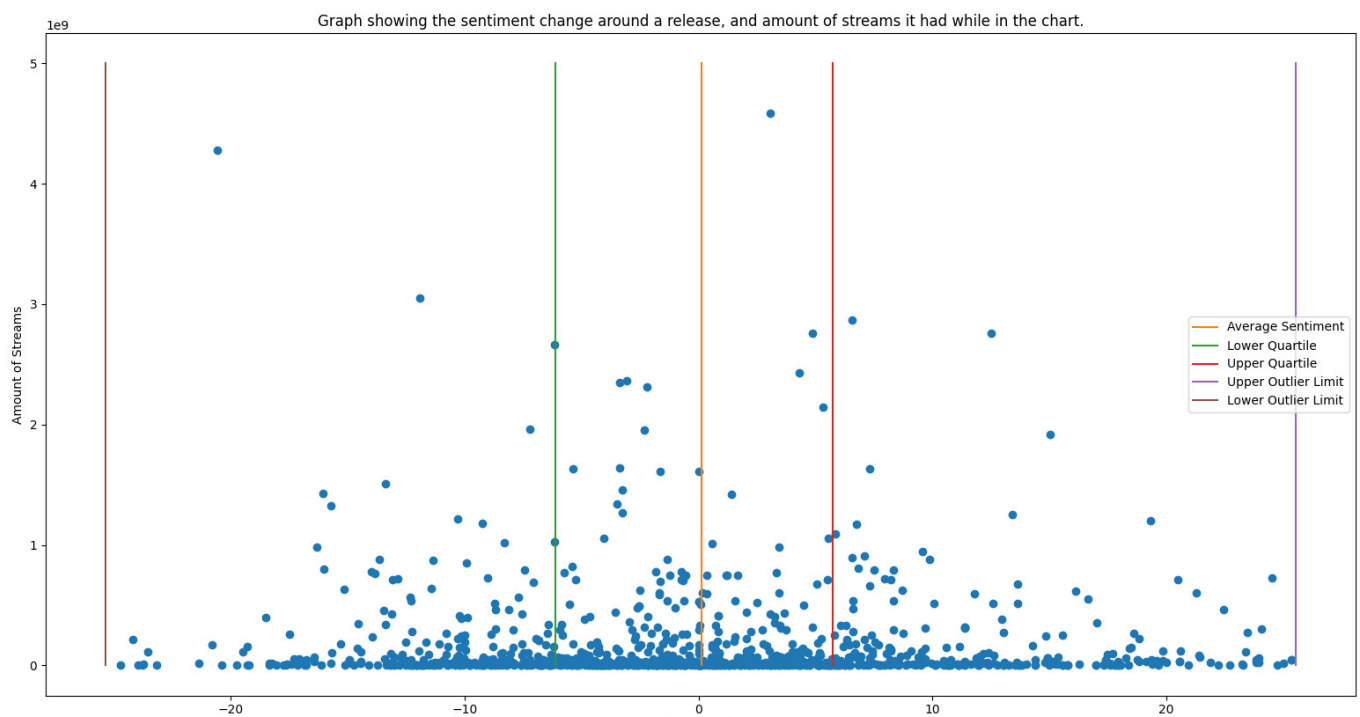


Figure 22: Alternative view of Figure 21, with outliers removed.

It is much harder to make out a major pattern here. For those release that break above the 1 billion mark, there is little pattern to their sentiment as 16 had an increase in their sentiment after a release, and 23 had a decrease in sentiment after their release.

This leads me to think that either sentiment change after a release does not indicate the possibility of strong commercial performance, or that the tweets collected were too “noisy”- not about the artist’s actual release, and therefore artist sentiment changed independently of how the release itself was received by fans.

#### 4.3 Differences in Sentiment from Twitter and Spotify

As discussed previously, sentiment is not just presented within text that is found on Twitter. While Spotify has no forum as such to express views in the same way, users can still “follow” their favourite artists, to get updates on them like when they release new music or announce new tour dates; or they can save songs to their “library” by clicking a small heart. While the latter is in my opinion a greater expression of sentiment (literally clicking a heart), the Spotify API provides no endpoint to see how many user libraries a given song is a part of.

(Mainly because a given users library is considered private and request to access it requires them to login with their Spotify account beforehand). However, as noted before, I had noted down the follower count of each artist when I created my artist MongoDB database. Therefore, I decided to chart out the follower counts against the average sentiment:

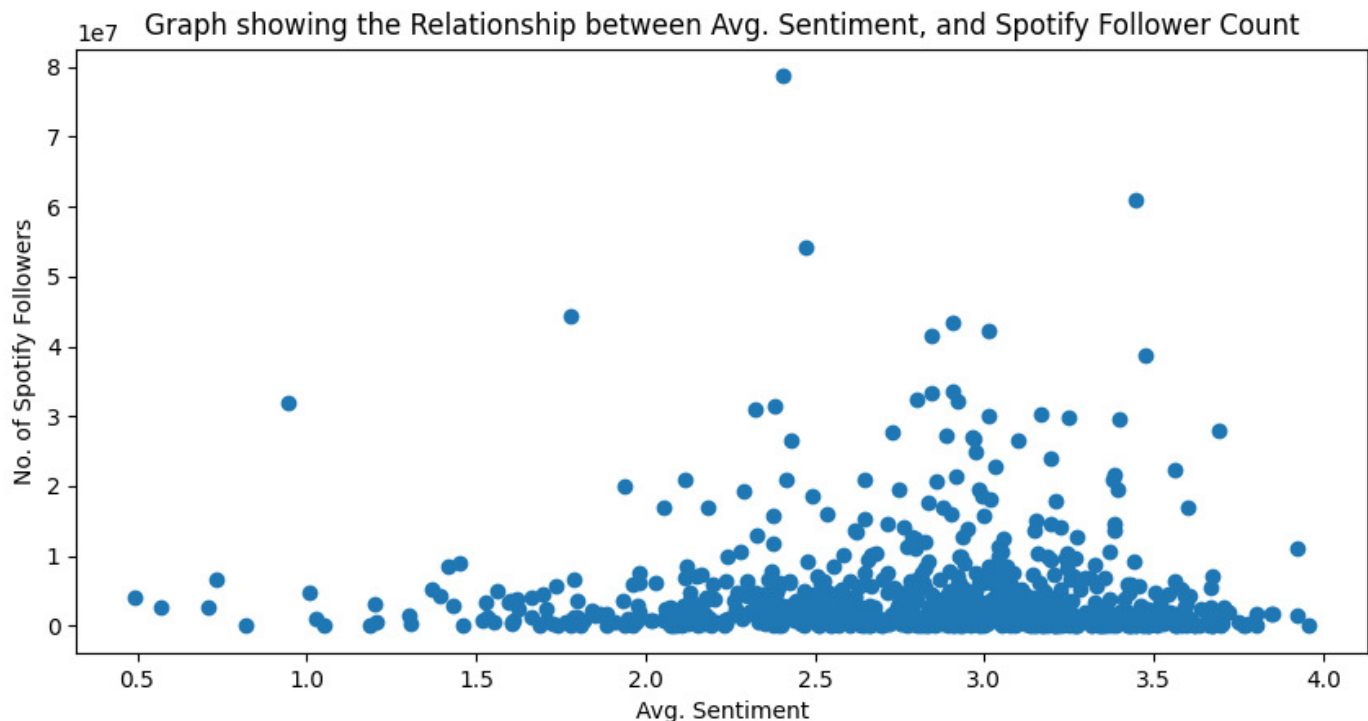


Figure 23: Graph of all artist's Spotify follower totals against the artist's average sentiment

This chart shows that those artists that can maintain a consistently good sentiment over time tend to be the most popular artists on Spotify.

Additionally, I think it could even be argued that this positive sentiment over time is usually required from artists to become extremely popular on Spotify, and therefore there is an argument for tracking the Twitter sentiment around artists over a medium to long time frame.

However, there is also lots of artists shown in the chart that have strong positive sentiment that do not have a large following on Spotify, which highlights the difference in how sentiment is shown by users on both platforms. Especially in my study, it could be argued that it's relatively easier to have a handful of Twitter users say something nice about you on that platform, than it is to build a following on Spotify that reaches into the millions.

Therefore, for me, those artists present with the top follower counts are key. For those over 3 million followers, all but two artists (Justin Bieber, and Bad Bunny) have positive sentiment (2 or greater on the axes in this report), indicating it as a characteristic present with popular artists. However, we cannot predict a current follower count from Twitter sentiment- perhaps if it were possible to obtain the number of followers gained over a period of time, we could compare that to the sentiment, and see if there is a pattern.

#### 4.4 Differences in Sentiment Between Musical Genres

Finally, I decided to investigate how sentiment changes across different genres of music, as part of goal 4 of my project. Spotify categorises music across a huge range of granular genres. A project called “Every Noise at Once” that is “an ongoing attempt at an algorithmically-generated, readability-adjusted scatter-plot of the musical genre-space...” has recorded 5415 different genres that Spotify has label songs on the platform to be a part of. [17] For example, a quick browser search of their list reveals there are 389 different rock genres. To simplify things slightly and prevent this report from having an additional 5000 or so graphs- 1 for each genre, I decided to focus on some core genres like “rock” and “pop” and include any artists who had at least one genre that included the core genre word. For example, for my “rock” genre chart below, artists that had been linked to “indie rock”, “latin rock”, “soft rock”, “modern rock” or any other genre with the word “rock” in it were included. The genres I made graphs for were: Rock, Pop, Hip Hop, House, Country and Electro. I also attempted to make an “R&B”/ “Rhythm & Blues” graph, but I found no artists with my method. Here is a summary of the charts, with each chart being available in the appendix:

Genre Name	Number of Artists (Percentage of all artists in chart)	Mean Sentiment	Interquartile Range	Average Spotify Follower Count
Rock	79 (10.6%)	2.67	0.817	4,883,513
Pop	444 (59.5%)	2.83	0.664	5,360,835
Hip Hop	189 (25.3%)	2.78	0.837	3,969,168
House	56 (7.51%)	3.034	0.785	4,540,627
Country	16 (2.14%)	2.64	0.403	2,381,588
Electro	66 (8.85%)	2.8	0.512	5,491,163

Looking at these results and the associated charts, pop is the most popular major genre on Spotify, with 444 artists appearing in the chart (59.9% of all the artists in the chart over 2019-2021). It also has the second highest sentiment across the genres and has a smaller spread of sentiment than other genres, which is interesting given how many artists have some form of pop genre associated to them, and how the other genres of comparable artists in them (Rock, and Hip Hop) have a much larger sentiment spread than Pop.

Country artists were the least popular on average according to the Spotify follower count, but also in average sentiment. It is also the only genre not to have an artist with 3.5 average sentiment or higher (though this may be due to the lack of artists that were found in the charts). It is hard to argue that the low sentiment value hints at this- the Rock genre has far more popular artists- and more artists total- yet, has just 0.03 higher sentiment on average than the Country artists. This indicates that sentiment cannot be used as a metric that clearly indicates that one genre is becoming more successful in the music industry than another.

On the other hand, the House genre has a clear lead in sentiment, and a strong average follower count. This could suggest that a very strong sentiment (3+) is no bad thing, and that therefore sentiment could be useful to track. Of course, more genres would have to be shown to have this level of sentiment and popularity to prove this for certain.

#### 4.5 Website Testing

To confirm that my flask website was working as intended, I recorded the results of some testing. Due to lack of time, I did not automate any of these, completing them manually instead:

Test No.	Test Description	Expected Result	Actual Result	Test Passed?
1	Test of "Drill Down" functionality (Main Page Graph)	Clicking an artist on the table on the main page brings the user to a full chart of that artist's sentiment.	Same as expected	Passed
2	Test of the "New Releases Effect" button on main page	Clicking this button takes the user to a page showing the interactive "releases" chart	Same as expected.	Passed
3	Test of the "follower relationship" button on main page.	Clicking this button takes the user to a page showing the interactive "followers" chart	Same as expected	Passed
4	Test of the Genre dropdown on main page.	Clicking a genre in this dropdown takes the user to a chart the correct chart for this genre.	Same as expected	Passed
5	Test of "Drill Down" functionality (Releases Graph)	Clicking an artist on the table on the main page brings the user to a full chart of	Same as expected	Passed

		that artist's sentiment.		
6	Test of "Drill Down" functionality (Followers Graph)	Clicking an artist on the table on the main page brings the user to a full chart of that artist's sentiment.	Same as expected	Passed
7	"Home" Button works in navbar (all pages)	For the Follower, Genre, Artist and release pages, the "home" navbar takes the user to the homepage.	Same as expected	Passed
8	"Followers Chart" button works on navbar (all pages)	For the home, Genre, Artist and release pages, the "Followers Chart" navbar takes the user to the homepage.	Same as expected.	Passed
9	"New Releases" button works on navbar (all pages)	For the Follower, Genre, Artist and home, the "New Releases" navbar takes the user to the homepage.	Same as expected	Passed

## 4.6 Evaluation of Results

Let us now look back to the goals that we stated at the start of the project, and see how well we can answer the questions they posed:

### 4.6.1 Goal 1 Evaluation (Positive Sentiment to long term success)

For this, I believe that we can mainly look to the average sentiment chart. It is not certain that artist with positive sentiment will have good commercial success. It is also clear that there is a "sweet spot" for sentiment between 3 and 3.5, as shown by the lower and upper quartiles.

#### 4.6.2 Goal 2 Evaluation (Role of Negative Sentiment)

Extending from the last point, negative sentiment should be avoided by musical artists. With just a handful of exceptions, only artists that have positive sentiment around them on Twitter over time have major commercial success. Additionally, as the follower chart shows, those that are successful on Spotify tend to have good sentiment. If an artist is found to have bad sentiment, social media managers should take action to improve it.

#### 4.6.3 Goal 3 Evaluation (Neutral Sentiment)

I think this question is mainly answered through the previous two sections, as neutral is effectively the middle ground of positive and negative sentiment in this project- there was no “neutral” class that tweets could be classified as. This ends up making it difficult to answer this since the data I used to train my model did not include a “neutral” class. There is an argument that using a “neutral” class in a future project could build on the work in this project, as it would be interesting to see what proportion of tweets from each of the other two classes are more neutral in nature.

#### 4.6.4 Goal 4 Evaluation (Differences in Genres)

Breaking the follower chart down into certain genres did lead to interesting findings. “Pop” is far and away the most popular genre if looking at the sheer number of artists in the chart associated with this genre. It also has quite a low spread to other genres.

My results show that there is similar sentiment shown towards artists of different genres. In my work this is shown when comparing “Rock” and “Country” genres. This indicates again that sentiment cannot necessarily be used to predict the future popularity of a genre. Of course, a follower count generally accumulates over time- looking at “change in followers or stream count” may have given more insight, but I ran out of time to investigate if it was possible to produce these results.

The genre with the highest sentiment was clearly the “House” genre. Since it has a 0.204 difference to the nearest genre that I looked at, looking at more genres that have a similar average sentiment would have been useful to see if there are any characteristics common to genres with sentiment greater than 3.

#### 4.6.5 Goal 5 Evaluation (Sentiment across platforms)

The chart I produced that compares Spotify Followers and average Twitter sentiment shows that the most loved artists on Spotify tend to have positive sentiment- though the spread is large: some artists with over 2 million followers have average sentiment around 2 (basically equal positive and negative), while others have sentiment over 3.5 (very positive). Few artists have managed to build high Spotify follower counts into the millions and have poor sentiment in the timeframe I investigated. In a similar way to what we’ve identified previously with stream totals, we cannot predict which artists will have the largest follower counts in the future, but we can say that if the artist has poor sentiment, over a long period, it is unlikely to be them.

#### 4.6.6 Goal 6 Evaluation (relationships between very popular and less popular artists and their sentiment)

This was difficult to answer during my project, as some data (particularly stream counts) were only publicly available through the spotifyCharts website, and so by definition the artists that I've investigated must be popular to a degree or else they would not have got on the chart. Therefore, if more data were to be publicly accessible, this goal could be worked on further.

However, in the work I did complete, the follower to sentiment chart does show that for the small proportion of artists that have a low sentiment, their follower count tends to be far lower. There are also artists with all kinds of follower counts at the more positive end of the sentiment spectrum.

#### 4.6.7 Goal 7 Evaluation (Around a Tweet's Subject)

Since this was my first attempt at sentiment analysis, and due to a lack of time, I did not investigate looking into how to break down my tweet database into those tweets that contain specific references to an artist's music. Therefore, this is a goal that I did not achieve in this project. I do discuss how this work could be completed in section 5.3 of this report.

## 5. Future Work

This section details some of the work I would complete if I were to continue with this project without the deadline. The aim of this work is to make the project more complete and to gain more insight from the project.

### 5.1 Automatic Data Collection.

The data used throughout this report was collected towards the start of my project, in a couple of major steps, and over a specific piece of time. To make this project more useful to those with an interest in the sentiment around music, it would be useful to collect this data in a real time fashion. This would mean that the data collected would continue well into 2021 to at least around the present day rather than the set 2019-2021 period investigated in this report.

For the most part, a lot of my work can be adapted so that the specific scripts used in the initial data collection are set to run regularly and close the delta between the time passed between the last "run" of the scripts, and the present time. For example, the python scripts that collect the Spotify data could be setup to run daily at midnight. They would first collect the most up to date chart published to spotifycharts.com (assuming they publish this new chart this quickly- the actual job time may need to be changed slightly otherwise). Once this data is collected, additional automation could be set up to collect the Spotify API data (genres, release data, follower counts) for any "new" artists to the Mongo Database.

The hardest part of this automated data collection would be collecting the Twitter data. Firstly, all the artists need tweets collected about them each new day. At 50 tweets every 14

days, this process would not be too strenuous to complete- though of course the number of artists that this process will need to be completed for will grow naturally over time. The tweets also must be classified for their sentiment, which could grow to be a costly process.

## 5.2 Additional Data Collection

In addition to collecting the most up to date data regularly, the data I have collected could be enhanced in several ways to allow for more insight to be gained about music artists and genre's sentiment. Naturally, with the data collection being the first task I completed for this project; these steps are far more obvious to me than they were at the beginning.

Firstly, during my project, I use the "Global 200" chart produced by Spotify, which shows the top 200 artists and their stream counts from across the globe. [Spotifycharts.com](https://spotifycharts.com) also allows for the top artists and their stream counts to be filtered by country. It would be interesting to break the artist's commercial performances down by country to see how they differ. Sentiment, meanwhile, would be more difficult to filter down to be country specific, as Twint does not offer its search to be focused on tweets from a specified country. It would still be interesting to see which country's streaming habits align most with the global sentiment though. Perhaps the use of the Twitter API may be able to allow collection of tweets by country, but this needs further investigation.

## 5.3 Use of NLP Pipeline

Finally, I decided to conduct basic Natural Language Processing when collecting my tweets through Twint by adding steps like removing stopwords, and webpage links. I noted previously that I decided against using the "Spacy" NLP Python package to complete my NLP, as it seemed too costly in time to learn how to use and implement in my project.

However, it could be used in several ways to get more insight from the tweets I collected. One way is that a Spacy model could be used to detect custom entities in the Tweets. In our case, these Entities could be artist names and release names. This would allow us to identify tweets that specifically refer to a given artist's music, and those that are about a different topic. These two subsets of tweets could then be compared, to see how both of their average sentiments relate to the artists music.

## 6. Conclusion

This report details the collection of data from Spotify and Twitter and then the use of that data to investigate the relationship between the sentiment around musical artists and genres and how they perform commercially. Overall, the results produced for this project show that it would be unwise to use sentiment to make predictions about the long-term success of an artist. Ultimately the range of fortunes for those with generally positive sentiment make this impossible. However, my findings also show that having negative sentiment is not something that any artist should want. Breakout commercial performances tend to occur to artists with good sentiment and many of the artists with the best Spotify follower counts have good average sentiment over the timeframe investigated.

Additionally, my analysis found that big swings in sentiment directly after the release of an album or single by an artist were generally very rare, and almost always followed by weaker



commercial performances compared to those that had more consistent sentiment during the time before and after a release. I also found that for those with the smaller swings in sentiment, strong performances were seen for those that had a sentiment drop after a release, as well as those that had their sentiment rise. Again, this makes it hard to identify a common pattern between changes in sentiment directly after a release, and the eventual performance of that release. Therefore, it can be argued that it is useful to measure and track sentiment directly after a release- except to ensure that a large, unwanted swing in sentiment has not occurred.

In terms of musical genres, similar sentiment can be seen for genres that are more popular (as measured by average Spotify followers for the artists included in the genre, and in the number of artists linked to the genre). Despite being the most popular genre, pop had a much smaller spread in sentiment compared to the other genres investigated. Country music was the least popular and had the lowest sentiment. House music had by far the highest sentiment- leading me to wanting to find more similarly talked about genres to compare it to. In the same way as most of the artists were, all the genres investigated in depth had positive sentiment on average.

I do believe that it is beneficial for those in the music industry to track sentiment around artists to ensure they are not regularly seen in a very negative light as it is far less likely that these artists will achieve significant commercial success. I also believe that sentiment around the release of an album or single could be useful to track, but only to ensure that large swings do not occur- in general there is no benefit to tracking this if the change is relatively small. Finally, I would argue that for those wishing to predict which next musical genre will grow in popularity, sentiment is not a useful metric, as I have shown that both popular and unpopular genres can have similar sentiment.

## 7. Reflection on Learning

### 7.1 What I have Learned.

I have really enjoyed this project and working with my supervisor, Daniela Tsaneva, during this semester. Having never delved into Sentiment Analysis previously, it was fascinating to learn about the different methods available to conduct this task, investigate other domains in which it has previously been used, and use it myself in a subject area that I love: music.

I also learned a lot more about Natural Language Processing during working on this project and the complexities of doing it with a large dataset. For the benefit of completing the project on time and having results to show, I kept it to as simple as I could, though I look forward to continuing to investigating the subject and applying what I have learned in other domains after my time at Cardiff University is complete (and hopefully avoiding the mistakes I made in the process!)

Another major learning point was the difficulties of working with large amounts of data. I collected over 3 million tweets that required their sentiment to be classified during this project, and a further 1.6 million were used in creating my classifier to perform this task. I did not foresee the sheer amount of time that would be required to both collect the Spotify and Twitter data and use the smaller labelled dataset to build these classifiers. This led to a

lot of time being spent waiting for my laptop to complete these tasks locally (and a lot of times having to leave it to number crunch overnight). Inevitably, this led to me having to scale back certain parts of my project or move on to the next stage of it all together due to this additional time used. I am looking forward to spending some time learning how these types of tasks could be completed in the cloud as I think that will help me avoid this kind of issues in the future (as well as giving my laptop a well-deserved rest).

## References

[1]	Jem Aswad. 2021. <i>Coldplay Return With Max Martin-Produced New Single 'Higher Power', Next Week</i> . Available at: <a href="https://uk.finance.yahoo.com/news/coldplay-return-single-higher-power-150114437.html">https://uk.finance.yahoo.com/news/coldplay-return-single-higher-power-150114437.html</a> [Accessed: 8 May 2021]
[2]	Vinerean, S., 2017. Importance of Strategic Social Media Marketing. <i>Expert Journal of Marketing</i> , 5(1), pp. 29
[3]	Saboo, A.R., Kumar, V. and Ramani, G., 2016. Evaluating the impact of social media activities on human brand sales. <i>International Journal of Research in Marketing</i> , 33(3), pp.524-541.
[4]	Cambridge University Press. 2021. <i>SENTIMENT  Meaning in the Cambridge English Dictionary</i> . Available at: <a href="https://dictionary.cambridge.org/dictionary/english/sentiment">https://dictionary.cambridge.org/dictionary/english/sentiment</a> [Accessed: 8 May 2021]
[5]	Dominic Prince. 2021. <i>YouGov Signal: social media sentiment towards 'big six' tanks following European Super League new</i>   YouGov. Available at: <a href="https://yougov.co.uk/topics/sport/articles-reports/2021/04/20/yougov-signal-social-media-sentiment-towards-big-s">https://yougov.co.uk/topics/sport/articles-reports/2021/04/20/yougov-signal-social-media-sentiment-towards-big-s</a>
[6]	Nausheen, F. and Begum, S.H., 2018, January. Sentiment analysis to predict election results using Python. In <i>2018 2nd international conference on inventive systems and control (ICISC)</i> (pp. 1259-1262).
[7]	Janalta Interactive. 2021. <i>What is Sentiment Analysis? – Definition from Technopedia</i> . Available at: <a href="https://www.techopedia.com/definition/29695/sentiment-analysis">https://www.techopedia.com/definition/29695/sentiment-analysis</a>
[8]	Introducing Vectorising. <i>NLP With Python for Machine Learning Essential Training</i> . 2021. LinkedIn Learning Video added by LinkedIn Learning. Available at: <a href="https://www.linkedin.com/learning/nlp-with-python-for-machine-learning-essential-training/introducing-vectorizing?u=78047178">https://www.linkedin.com/learning/nlp-with-python-for-machine-learning-essential-training/introducing-vectorizing?u=78047178</a> [Accessed at: 8 May 2021]
[9]	IBM Cloud Education. 2020. <i>What is Supervised Learning?</i>   IBM. Available at: <a href="https://www.ibm.com/cloud/learn/supervised-learning">https://www.ibm.com/cloud/learn/supervised-learning</a> [Accessed: 8 May 2021]
[10]	Goel, A., Gautam, J. and Kumar, S., 2016, October. Real time sentiment analysis of tweets using Naive Bayes. In <i>2016 2nd International Conference on Next Generation Computing Technologies (NGCT)</i> (pp. 257-261).
[11]	Tammy Da Costa. 2020. <i>Using Sentiment Analysis to Analyse Stocks</i> . Available at: <a href="https://www.dailyfx.com/education/understanding-the-stock-market/stock-market-sentiment-analysis.html">https://www.dailyfx.com/education/understanding-the-stock-market/stock-market-sentiment-analysis.html</a> [Accessed at: 8 May 2021]
[12]	Spotify AB. 2021. <i>Spotify Charts</i> . Available at: <a href="https://spotifycharts.com/regional">https://spotifycharts.com/regional</a> [Accessed at: 8 May 2021]
[13]	Spotify AB. 2021. <i>Authorization Guide   Spotify For Developers</i> . Available at: <a href="https://developer.spotify.com/documentation/general/guides/authorization-guide/#client-credentials-flow">https://developer.spotify.com/documentation/general/guides/authorization-guide/#client-credentials-flow</a> [Accessed: 8 May 2021]
[14]	Spotify AB. 2021. <i>Web API Reference   Spotify For Developers</i> . Available at: <a href="https://developer.spotify.com/documentation/web-api/reference/">https://developer.spotify.com/documentation/web-api/reference/</a>

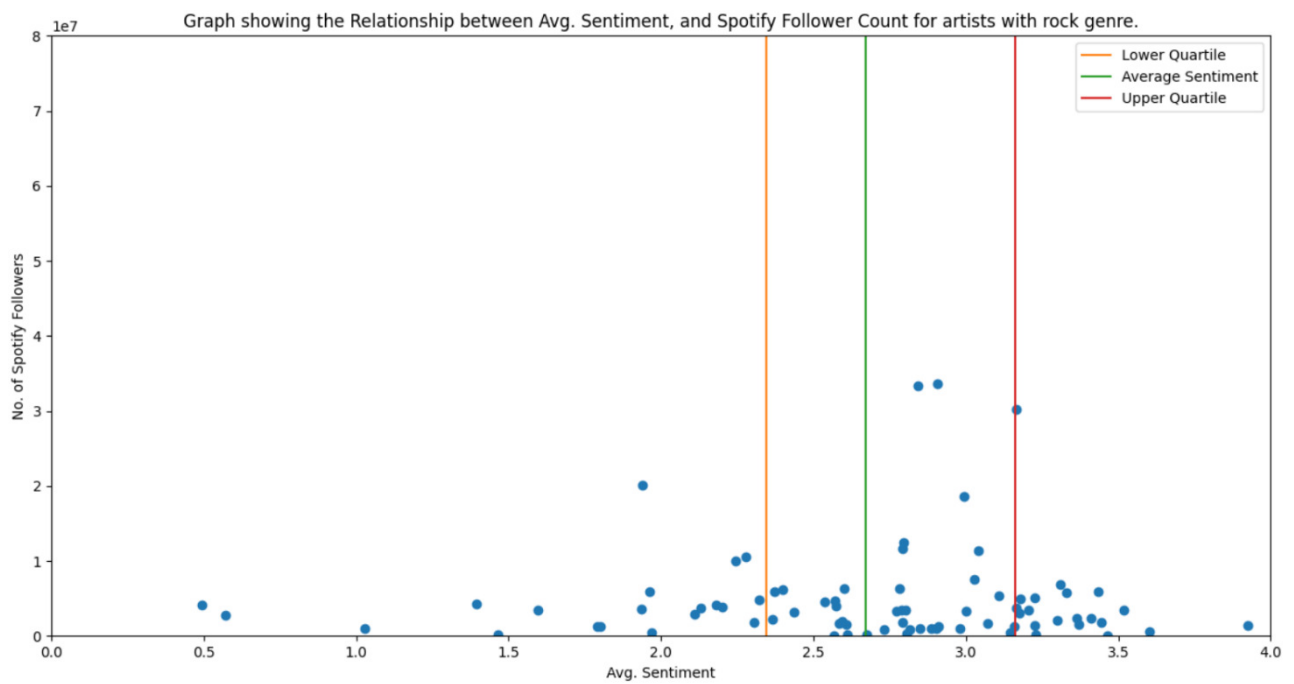
[15]	Francesco Poldi et al. 2020. <i>Twint Source Code</i> . Version 2.1.21. [Source Code] Available at: <a href="https://github.com/twintproject/twint/tree/master">https://github.com/twintproject/twint/tree/master</a> [Accessed: 6 February 2021].
[16]	Bird, S., Klein, E. and Loper, E., 2009. <i>Natural language processing with Python: analyzing text with the natural language toolkit</i> . " O'Reilly Media, Inc."
[17]	Glenn McDonald. 2021. <i>Every Noise at Once</i> . Available at: <a href="https://everynoise.com/everynoise1d.cgi?scope=all">https://everynoise.com/everynoise1d.cgi?scope=all</a> [Accessed at: 8 May 2021]
[18]	Go, A., Bhayani, R. and Huang, L., 2009. Twitter sentiment classification using distant supervision. <i>CS224N project report, Stanford</i> , 1(12)
[19]	Kaggle Inc. 2021. <i>Sentiment140 dataset with 1.6 million tweets</i> . Available at: <a href="https://www.kaggle.com/kazanova/sentiment140">https://www.kaggle.com/kazanova/sentiment140</a> [Accessed at: 8 May 2021]
[20]	Scikit-learn developers. 2020. <i>sklearn.model.selection.train_test_split – scikit-learn 0.24.2 documentation</i> . Available at: <a href="https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html">https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html</a> [Accessed at: 8 May 2021]
[21]	ExplosionAI GmbH et al. 2021. <i>spaCy Source Code</i> . Version 3.0.1. [Source Code] Available at: <a href="https://github.com/explosion/spaCy">https://github.com/explosion/spaCy</a> [Accessed at: 6 February 2021]
[22]	Scikit-learn developers. 2020. <i>Sklearn.naive_bayes.GaussianNB – scikit-learn 0.24.2 documentation</i> . Available at: <a href="https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html">https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html</a> [Accessed at: 8 May 2021]
[23]	Scikit-learn developers. 2020. <i>Sklearn.feature_extraction.text.CountVectorizer – scikit-learn 0.24.2 documentation</i> . Available at: <a href="https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer.fit_transform">https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer.fit_transform</a> [Accessed at: 8 May 2021]
[24]	Van Rossum, G., 2020. <i>The Python Library Reference, release 3.8.2</i> , Python Software Foundation.
[25]	Scikit-learn developers. 2020. <i>Sklearn.naive_bayes.BernoulliNB – scikit-learn 0.24.2 documentation</i> . Available at: <a href="https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB">https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB</a> [Accessed at: 8 May 2021]
[26]	Wikimedia Foundation, inc. 2021. <i>The Police – Wikipedia</i> . Available at: <a href="https://en.wikipedia.org/wiki/The_Police">https://en.wikipedia.org/wiki/The_Police</a> [Accessed at 8 May 2021]
[27]	Danny Goodwin. 2019. <i>How to Deal With All the Negativity on Twitter</i> . Available at: <a href="https://www.searchenginejournal.com/twitter-negativity/292245/#close">https://www.searchenginejournal.com/twitter-negativity/292245/#close</a> [Accessed at: 8 May 2021]
[28]	Stuart Dredge. 2019. <i>Music Industry enters 2020 on a wave of growth – and optimism</i> . Available at: <a href="https://musically.com/2020/01/03/analysis-music-industry-2020-growth/">https://musically.com/2020/01/03/analysis-music-industry-2020-growth/</a> [Accessed at: 8 May 2021]
[29]	Liikkanen, L.A. and Åman, P., 2016. Shuffling services: Current trends in interacting with digital music. <i>Interacting with Computers</i> , 28(3), pp.352-371.
[30]	Pichl, M., Zangerle, E. and Specht, G., 2014. Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation. In <i>Grundlagen von Datenbanken</i> (pp. 35-40).
[31]	Spotify AB. 2021. <i>Ariana Grande   Spotify</i> . Available at: <a href="https://open.spotify.com/artist/66CXWjxzNUsdJxJ2Jdwvnr">https://open.spotify.com/artist/66CXWjxzNUsdJxJ2Jdwvnr</a> [Accessed at: 9 May 2021]

[32]	Twitter, inc. 2021. "Grande updtte on Twitter: "ariana grande says she is fully vaccinated against covid-19 on Instagram <a href="https://t.co/cyDqiBSpdy">https://t.co/cyDqiBSpdy</a> " / Twitter" Available at: <a href="https://twitter.com/grandeupdtte/status/1391107023572455426">https://twitter.com/grandeupdtte/status/1391107023572455426</a> [Accessed at: 9 May 2021]
[33]	Twitter, inc. 2021. "Buzzing Pop on Twitter: "Ariana Grande was spotted out and about with friends last night in Los Angeles. <a href="https://t.co/21U3Y1ET36">https://t.co/21U3Y1ET36</a> " / Twitter". Available at: <a href="https://twitter.com/BuzzingPop/status/1390901005127196675">https://twitter.com/BuzzingPop/status/1390901005127196675</a> [Accessed at: 9 May 2021]
[34]	Twitter, inc. 2021. "Buzzing Pop on Twitter: "Lorde is revealed to be a fan of Ariana Grande's smash hit, "Into You", in resurfaced tweets. <a href="https://t.co/SJpVmlJkDJ">https://t.co/SJpVmlJkDJ</a> " / Twitter". Available at: <a href="https://twitter.com/BuzzingPop/status/1391014835987615754">https://twitter.com/BuzzingPop/status/1391014835987615754</a> [Accessed at: 9 May 2021]
[35]	Mike Dirolf et al. 2021. PyMongo. 3.11.3. [Source Code]. Available at: <a href="https://github.com/mongodb/mongo-python-driver">https://github.com/mongodb/mongo-python-driver</a> [Accessed: 9 May 2021]
[36]	Dutta, U., Hanscom, R., Zhang, J.S., Han, R., Lehman, T., Lv, Q. and Mishra, S., 2020. Analyzing Twitter Users' Behavior Before and After Contact by the Internet Research Agency. <i>arXiv preprint arXiv:2008.01273</i> .
[37]	Freelon, D. and Lokot, T., 2020. Russian Twitter disinformation campaigns reach across the American political spectrum. <i>Misinformation Review</i> .
[38]	Dr. Robert Kübler. 2021. <i>Learning by Implementing: Gaussian Naïve Bayes</i> . Available at: <a href="https://towardsdatascience.com/learning-by-implementing-gaussian-naive-bayes-3f0e3d2c01b2">https://towardsdatascience.com/learning-by-implementing-gaussian-naive-bayes-3f0e3d2c01b2</a> [Accessed: 12 May 2021]
[39]	Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language models are few-shot learners. <i>arXiv preprint arXiv:2005.14165</i> .
[40]	OpenAI. 2021. <i>Openai-python</i> . 0.3.0. [Source Code]. Available at: <a href="https://github.com/openai/openai-python">https://github.com/openai/openai-python</a> [Accessed at: 13 May 2021]

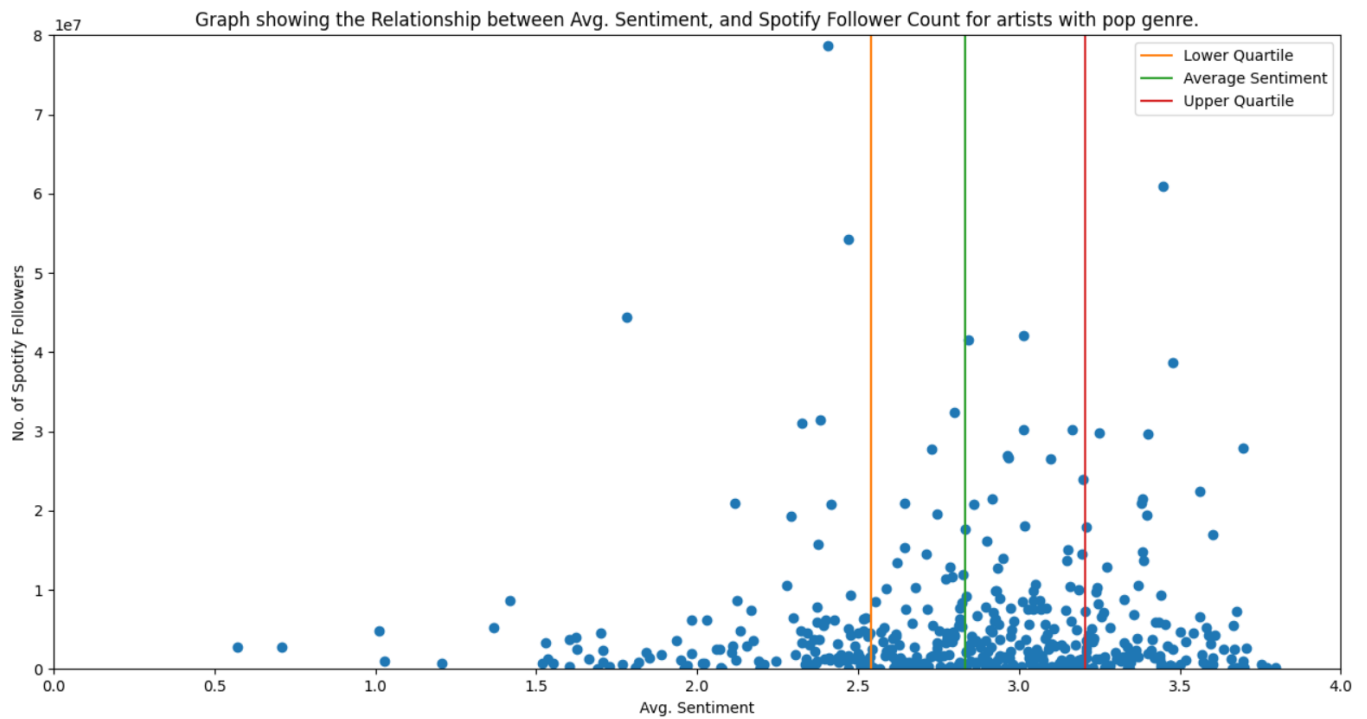
## Appendix

Appendix 1: Full list of artists and their recorded average sentiment  
See 'fullSentimentList.csv' in archive.zip

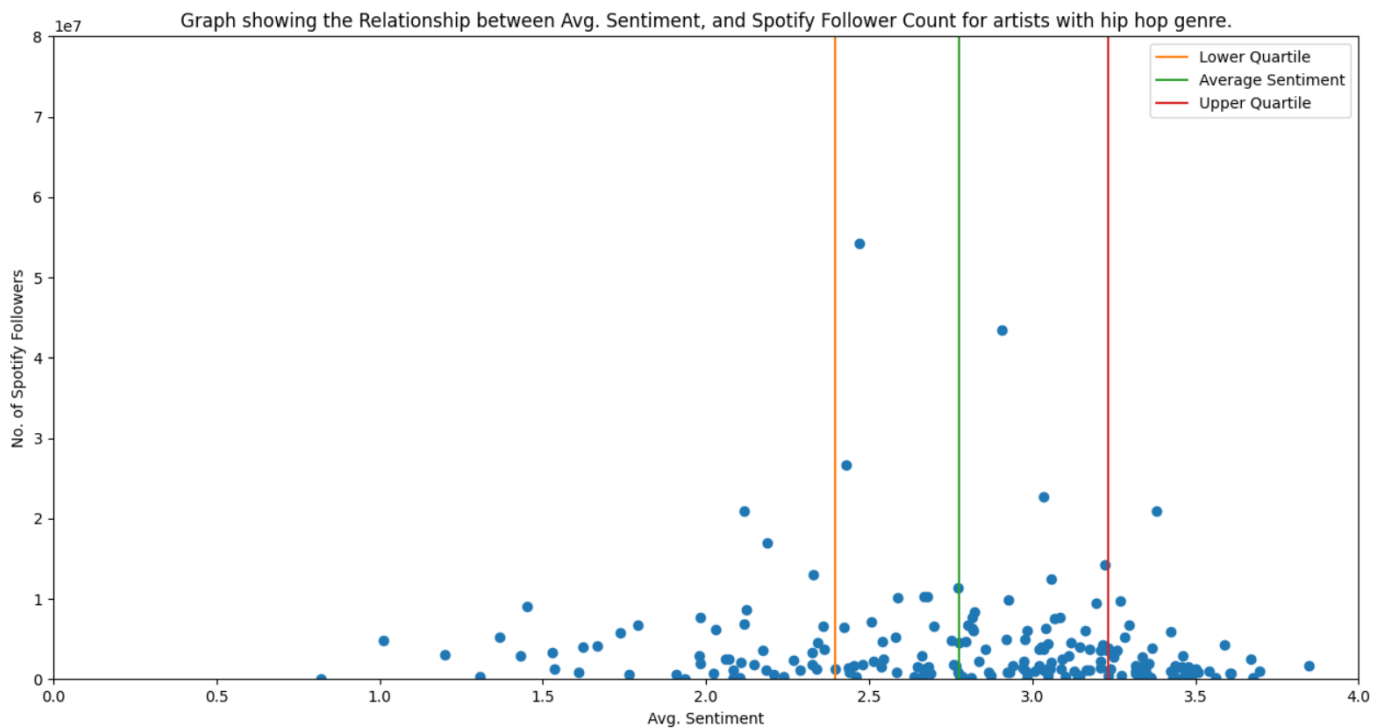
Appendix 2: Graph showing Spotify follower counts against average sentiment for “Rock” artists:



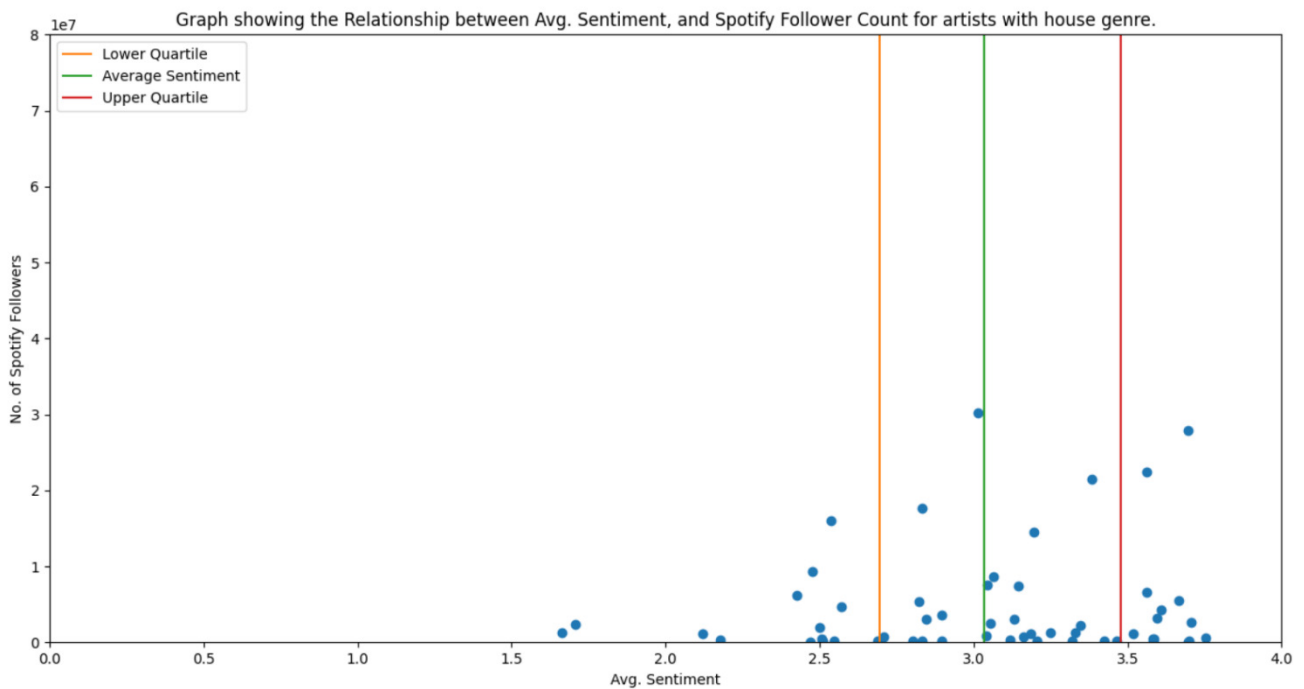
Appendix 3: Graph showing Spotify follower counts against average sentiment for “Pop” artists:



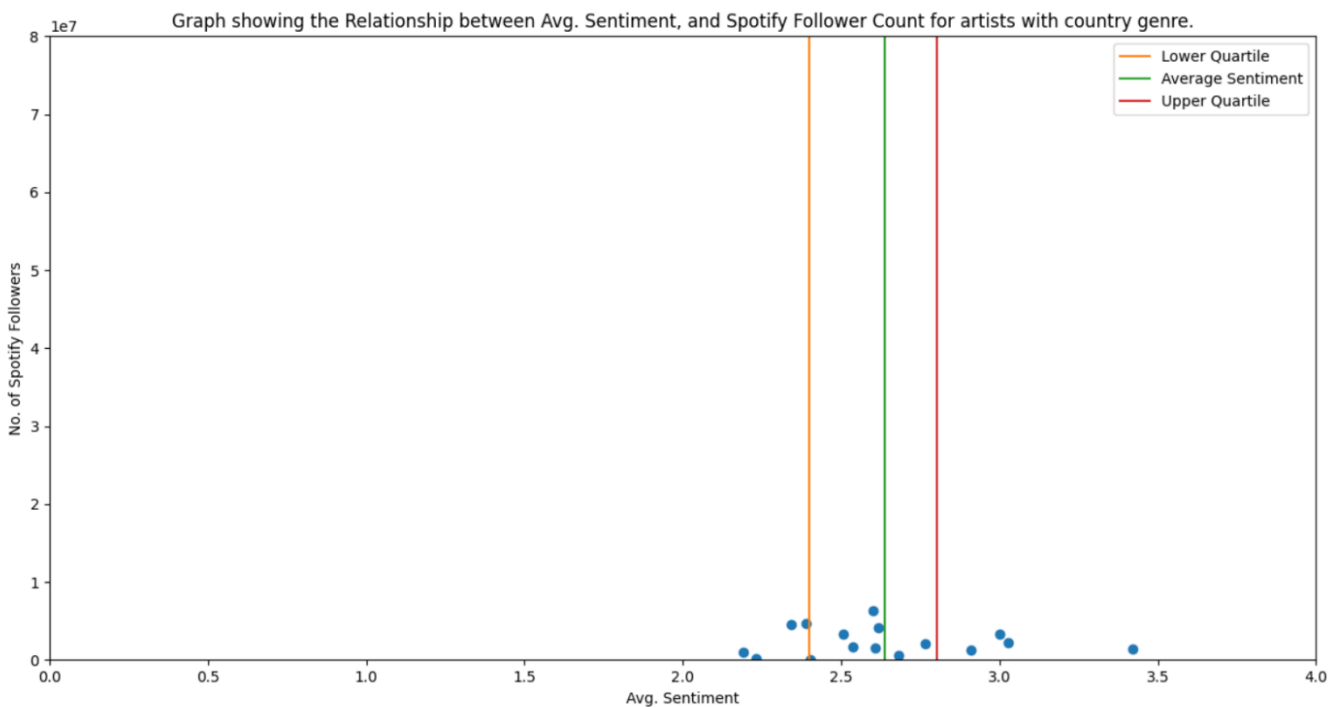
Appendix 4: Graph showing Spotify follower counts against average sentiment for “Hip Hop” artists:



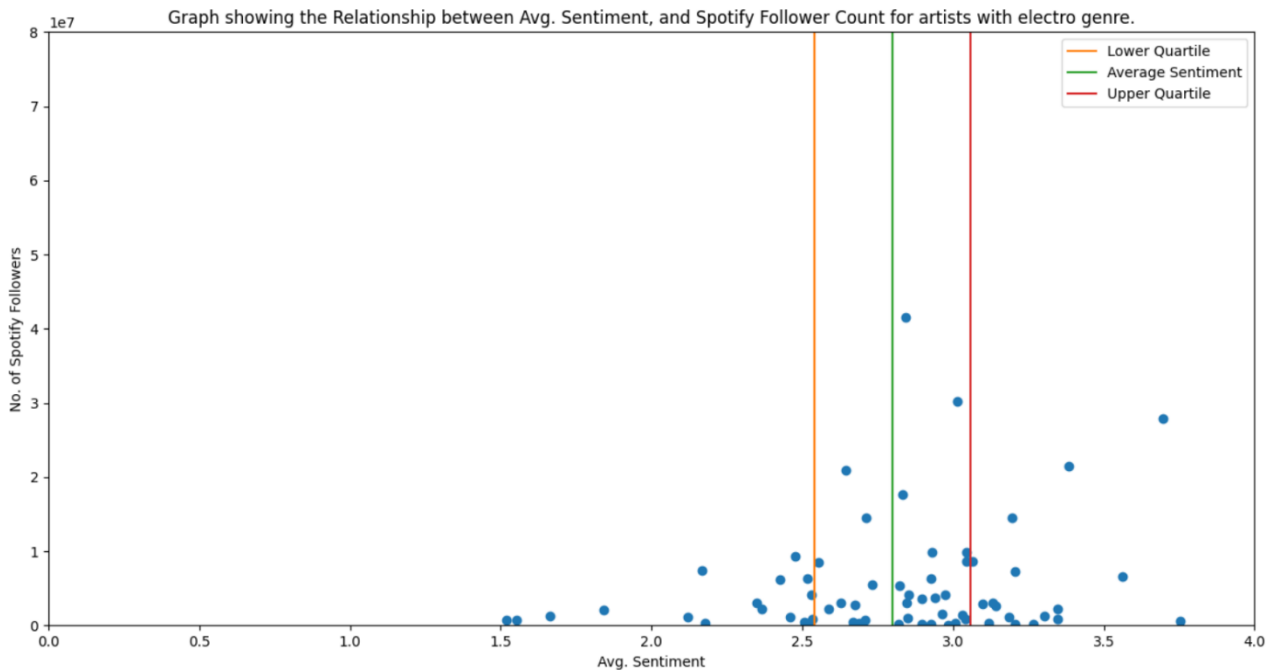
Appendix 5: Graph showing Spotify follower counts against average sentiment for “House” artists:



Appendix 6: Graph showing Spotify follower counts against average sentiment for “Country” artists:



Appendix 7: Graph showing Spotify follower counts against average sentiment for “Electro” artists:



Appendix 8: scaper.py – contains code used to complete Spotify and Twitter data collection tasks and analysing the MongoDB database I created. Attached in archive.zip.

Appendix 9: openAiWork.py – contains the code that was used to assess the OpenAi classifier. Attached in archive.zip.

Appendix 10: classifier\_work.py – contains the code used to train and test my sklearn classifiers, and then use them on my collected Twitter data. Attached in archive.zip.

Appendix 11: sentimentWork.py – contains the code that analyses the results of my sentiment analysis. Attached in archive.zip.