Final Project SI206 - Spotify Sentiment Analysis

Jamie, Amy, Liam

Repository Link: https://github.com/jamiexpark/Song-Sentiment-Score-Analysis

1.The goals for your project including what APIs/websites you planned to work with and what data you planned to gather(10 points)

In our Spotfiy Sentiment score analysis project for the Final Project, we worked with Spotify/Spotipy, LyricsGenius with Genius, and Vader Sentiment score APIs to create databases with artist name and their streams, overall average sentiment score for an artist's songs, and comparing the top 10 songs with the highest sentiment scores, and the bottom 10 songs with the lowest sentiment scores. Our visualizations captured the relationship between these tables and gave us a clear visual representation of how the sentiment scores for top artists differ from artists below the ranking, how many streams top artists really get, and more.

2.The goals that were achieved including what APIs/websites you actually worked with and what data you did gather(10 points)

- Spotipy/Spotify Developer Tools: We used Spotify developer tools to utilize the keys to access the database, and Spotipy to find the top songs per artist.
- Chartmasters: We used BeautifulSoup to webscrape the top 100 artists and their respective number of streams.

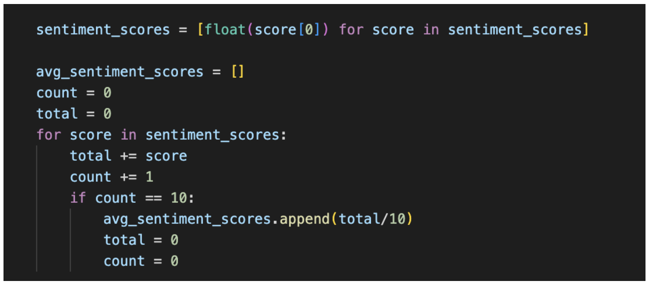
  LyricsGenius/Genius: We used this API to find and store the lyrics of the top 10 songs for each of the top 10 artists.
- vaderSentiment: We used this package to conduct sentiment analysis on song lyrics that were obtained from the LyricsGenius API. The sentiment scores produced told us which songs were positive, negative, or neutral.
- **Overall:** Through this project, we sought to apply and expand our knowledge of working with APIs, databases, and web scraping. We achieved this by using the aforementioned resources listed above. The process was complicated and there was lots of trial and error. We will describe some of the issues in the next question.

3.The problems that you faced (10 points)

- We were a bit stumped with figuring out how to put 25 (or in our case 10) data points each time and be able to run the data multiple times in this way, but we figured out how to keep continuity in the program. We were also having some trouble with the visualizations, but with all the resources available to matplotlib, we figured out how to make them more unique.
- A smaller issue we faced was working with a shared repository where we had to pull, push, and merge. Initially we were confused but we got the hang of it and it led to a very streamlined process of updating our files altogether.

4.The Calculations from the data in the database(i.e. a screenshot)(10 points)

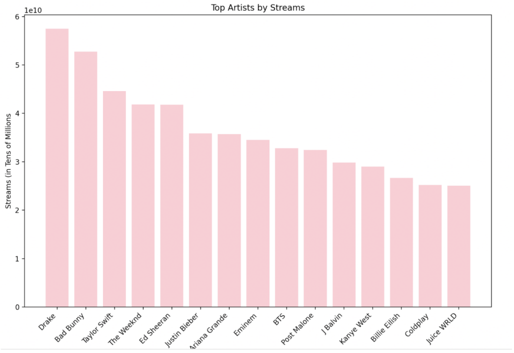
(calculations of calculating average sentiment score)

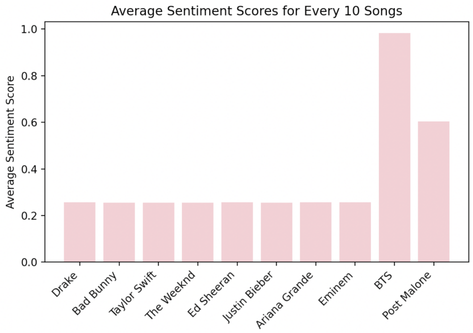


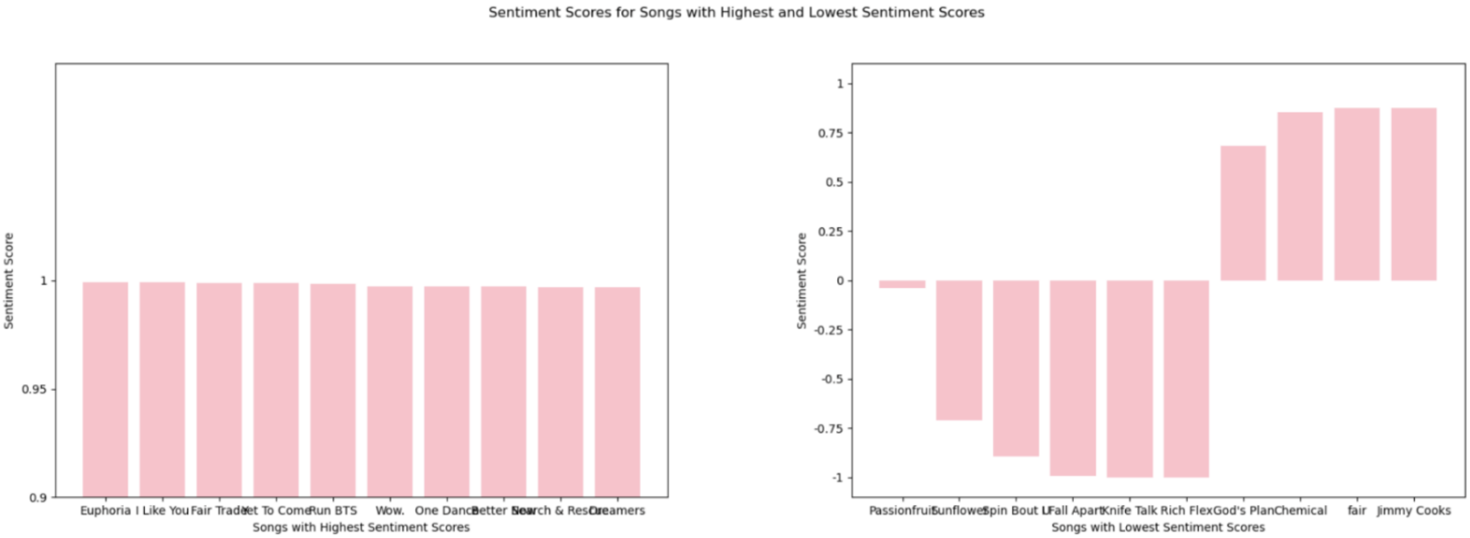
```
sentiment_scores = [float(score[0]) for score in sentiment_scores]

avg_sentiment_scores = []
count = 0
total = 0
for score in sentiment_scores:
    total += score
    count += 1
    if count == 10:
        avg_sentiment_scores.append(total/10)
        total = 0
        count = 0
```

5.The visualization that you created (i.e. screen shot or image file) (10 points)







6.Instructions for running your code (10 points)

- Install the proper resources using the following code if on Python3
    - pip install spotipy
    - pip install lyricsgenius
    - pip install vaderSentiment
    - Import these to your file
        - Use these guides to help

        - https://lyricsgenius.readthedocs.io/en/master/
        - https://spotipy.readthedocs.io/en/2.22.1/
        - https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/
    - You will need to get API credentials from Spotify and Genius in order to properly run the code
- Once these packages have been installed, the code will run
- Run the file 10 times in order to fully populate the database
- The second file with automatically update after the database is populated.

7.Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)

Def parse_web_with_soup(website):

- This function takes in a website(we use chartmasters for webscraping)
- The output of the function is a tuple of “artist_streams” that matches the artist name, and the number of streams they have on spotify.
- The function also creates a table of the artist and the number of streams they have.

Def top_ten_songs(artist_streams):

- The function takes the tuples made from parse_web_with_soup and then uses the name of the artists to search up the top ten songs per artist. It then stores the top 10 songs as a tuple with the name of the artist and their top 10 songs in top10_by_artist.

Def top_song_verses(top_10_songs):

- The function takes in the top_10_songs list that is returned in the top_ten_songs() function
- Call the genius API through LyricsGenius package to get lyrics of the top 10 songs for each artist
- Perform sentiment analysis of each of the songs
- Create a list of tuples that contain each song name and corresponding sentiment score
- In groups of 10, commit the names of the songs with their sentiment scores to the music database
- No output

Def run_visualization(connection):

- The function takes in “connection” or “conn” that connects the databases created in earlier functions
- Then, it calls visualtime.visualizations(connection), which is the function that runs all of the other visualizations.
- It then makes sure to close the connection when done.
- No output

Def visualizations(conn):

- The function is called from the main file to run all the visualizations one by one (out of 3 total). The purpose of this helper function was for debugging and showing each visualization individually.

Def visual_one(conn):

- The function takes in a conn/connection, and takes in data from the artist table on the SQL Database and creates a bar graph visualizing the artist name vs streams in tens of millions.

Def visual_two(conn):

- The function takes in the conn/connection, and takes in the data in sentiments table and the artists, and calculates the average for each 10 sentiment scores. Then, it matches up those averages to the artist name and prints a bar graph.

Def visual_three(conn):

- The function takes in the conn/connection, and takes in the data from the table, “sentiments”. It selects the top 10 distinct songs, as well as the bottom 10 distinct songs, to visually represent the difference between the two.

8.You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	Issue Description	Location	Result (did it solve the issue?)
4/19	Needed to know how to make graphs more unique and create multiple graphs	https://www.tutorialspoint.com/how-to-plot-multiple-graphs-in-matplotlib	Learned how to create multiple graphs!
4/13	Needed to know how to use Spotipy (features and authorization). This documentation gave all the instructions on how to get started from downloading to using the features	https://spotipy.readthedocs.io/en/2.22.1/	Learned how to use the Spotipy API - How to authorize spotipy API - Features and references
4/17/23	Needed to know how to use lyricsgenius. This documentation gave a step by step of how to set up and use the functions provided in the package.	https://lyricsgenius.readthedocs.io/en/master/	Learned how to use the lyricsgenius/genius API - Functions - Authorize and

			get Genius API token
4/17/23	Needed to know how to conduct a sentiment score analysis of the lyrics taken from the lyricsgenius API. This blog post gave a step by step process of how to install the package and utilize it properly.	https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/	Learned how to conduct sentiment analysis on strings
4/18/23	Needed to know how to only insert unique values into a table. This stackoverflow post taught a simple way to do so.	https://stackoverflow.com/questions/19337029/insert-if-not-exists-statement-in-sqlite	Learned how to insert only unique values into a sqlite database table