Beckett Dumo

Dr. Jason Zietz

INFO 3510

11/8/2023

The community around music: An exploration of the danceability and sentiment of popular songs

Background

The way we share and listen to music has evolved with the rise of social media. Music brings people together. Communities around genres and artists are everywhere, from social media like Twitter to forums like Reddit. The love of music transcends any language or culture. As people, we can connect through sounds and music as profoundly as we connect through words and touch. With social media growing larger by the day and the love of music unwavering, sharing music we like online is routine. One digital service does this better than the rest.

TikTok exploded as one of the largest social media services in 2018. It uses short videos as a means of content to be consumed by users. Songs can trend quickly on TikTok because, more often than not, the short videos are accompanied by music. TikTok trends heavily influence the popularity of songs used in their videos. Business Insider writes, “Songs that trend on TikTok often end up charting on the Billboard 100 or Spotify Viral 50.”[1] Intriguingly, it is not only new songs that trend on TikTok; songs multiple decades old have found their way into trends and charts. With TikTok shaping listening habits more than ever, I sought to learn if its effect can be measured. In an age of social media and trends affecting what music is popular, are popular songs danceable, happy songs? Or are they popular because they are catchy? I approached this task through the following analysis.

---

[1] Whateley, Dan. “How Tiktok Is Changing the Music Industry and the Way We Discover New, Popular Songs.” Business Insider, Business Insider, www.businessinsider.com/how-tiktok-is-changing-music-industry#:~:text=Songs%20that%20trend%20on%20TikTok,music%2Danalytics%20company%20MRC%20Data.

Libraries and APIs

I began by looking at the APIs I used in class and previous classes. I had used the Spotify API before for my final project in INFO 2201 and enjoyed the speed and data available on the API. I thought AZLyrics was a more consistent source for lyrics than Genius, so it was natural that I utilized the AZLyrics API. Furthermore, I found the Valence Aware Dictionary and Sentiment Reasoner we used in class the best way to approach the sentiment of lyrics. VADER worked well with large amounts of words and could handle nuances because of how it was trained. Lastly, during my preliminary research, I discovered the Billboard API. It was an API designed to pull the Billboard Top 100 songs from any date. It was easy to use and outputted a ChartData object of the top 100 songs wherein each entry had attributes of the title and artist. Calling the artist or song name directly for each entry proved very useful.

Getting popular songs

The first step in the analysis was to pull the current Billboard Top 100 songs into the Python notebook. After pulling the ChartData in, I ran two for-loops to separate the artists and song titles into their lists. Here is an example of the output:

```
hot-100 chart from 2023-10-28
-----------------------------
1. 'Cruel Summer' by Taylor Swift
2. 'Paint The Town Red' by Doja Cat
3. 'Snooze' by SZA
4. 'IDGAF' by Drake Featuring Yeat
5. 'Monaco' by Bad Bunny
6. 'I Remember Everything' by Zach Bryan Featuring Kacey Musgraves
7. 'Fast Car' by Luke Combs
8. 'First Person Shooter' by Drake Featuring J. Cole
9. 'Last Night' by Morgan Wallen
10. 'Thinkin' Bout Me' by Morgan Wallen
11. 'Fukumean' by Gunna
12. 'Vampire' by Olivia Rodrigo
13. 'Dance The Night' by Dua Lipa
14. 'Fina' by Bad Bunny & Young Miko
15. 'Virginia Beach' by Drake
```

Aggregate sales, streams, and radio plays of songs calculate the Billboard Top 100 songs. The list is updated weekly, so for consistency of the analysis, I specified the date for the list to be the Hot 100 chart from October 28, 2023. Based on the artists, there was a good diversity of genres, from Country with Luke Combs to Rap with Drake. Having a good diversity of genres

was essential to me in this analysis because if one artist dominated the list, then my analysis would be biased from the genre or style the artist belongs to. This was the case for the first week of November; over half the list was Taylor Swift. Taylor rereleased her famous 1989 album as Taylor's Versions; every song is trending, which changed my visualizations noticeably. The album separates from her folk style, using more synth and upbeat pace. My visualizations moved from a normal distribution to a primarily high level of danceability and positivity.

Lyrics

With the Billboard chart data and two lists of the songs and their artists in the notebook, I moved on to the lyrics. The AZLyrics API proved invaluable to the analysis. It allowed me to get the lyrics for the popular songs using the song name and artist. It was able to handle features and multiple artists. I set the accuracy of the API search to be 50%. This 10% decrease allowed me to get some song lyrics that were lost on higher scores. For the lyric search, I created a for-loop with try and except catches. I create a lyrics list for the lyrics to be appended to after they are found. I iterate through the ChartData entries within the for-loop, inputting each song artist and title into the AZLyrics API. If a song lyric is found on the website, the lyrics list adds the lyrics to the list. If not, the API searches for the lyrics on a search engine, the default being Google. If the song isn't there either, it will output that no lyrics are found, and a 0 is appended to the list. The except part of the try-except block is used to catch index errors that occur from too many missed songs by the API. The exception would allow the for loop to keep running without interrupting. The outputted lyrics list was full of newline characters and backslashes, hard for people but not hard for a computer. Here is an example of Cruel Summer by Taylor Swift:

```
lyrics
# Lot of lyrics
```

```
['(Yeah, yeah, yeah, yeah)\n\nFever dream high in the quiet of the night\nYou know that I caught it (Oh yeah, you\'re right, I
want it)\nBad, bad boy, shiny toy with a price\nYou know that I bought it (Oh yeah, you\'re right, I want it)\n\nKilling me slo
w, out the window\nI\'m always waiting for you to be waiting below\nDevils roll the dice, angels roll their eyes\nWhat doesn\'t
kill me makes me want you more\n\nAnd it\'s new, the shape of your body\nIt\'s blue, the feeling I\'ve got\nAnd it\'s ooh, whoa
oh\nIt\'s a cruel summer\nIt\'s cool, that\'s what I tell \'em\nNo rules in breakable heaven\nBut ooh, whoa oh\nIt\'s a cruel s
ummer\nWith you\n\nHang your head low in the glow of the vending machine\nI\'m not dying (Oh yeah, you\'re right, I want it)\nW
e say that we\'ll just screw it up in these trying times\nWe\'re not trying (Oh yeah, you\'re right, I want it)\n\nSo cut the h
eadlights, summer\'s a knife\nI\'m always waiting for you just to cut to the bone\nDevils roll the dice, angels roll their eyes
\nAnd if I bleed, you\'ll be the last to know\n\nOh, it\'s new, the shape of your body\nIt\'s blue, the feeling I\'ve got\nAnd
it\'s ooh, whoa oh\nIt\'s a cruel summer\nIt\'s cool, that\'s what I tell \'em\nNo rules in breakable heaven\nBut ooh, whoa oh
\nIt\'s a cruel summer\nWith you\n\nI\'m drunk in the back of the car\nAnd I cried like a baby coming home from the bar (Oh)\nS
aid, "I\'m fine," but it wasn\'t true\nI don\'t wanna keep secrets just to keep you\nAnd I snuck in through the garden gate\nEv
ery night that summer just to seal my fate (Oh)\nAnd I scream, "For whatever it\'s worth\nI love you, ain\'t that the worst thi
ng you ever heard?"\nHe looks up, grinning like a devil\n\nIt\'s new, the shape of your body\nIt\'s blue, the feeling I\'ve got
\nAnd it\'s ooh, whoa oh\nIt\'s a cruel summer\nIt\'s cool, that\'s what I tell \'em\nNo rules in breakable heaven\nBut ooh, wh
```

VADER sentiment analysis

Now that I have used the Billboard API to search for popular songs and AZLyrics to find the words to those popular songs, I need to find the sentiment of the lyrics. Instead of TextBlob, I went with VADER, another NLTK sentiment tool. VADER finds the sentiment of a text by gauging how many positive, negative, or neutral words it uses. Its lexicon defines these as a vocabulary of happy and sad words. I imported the library suite since the tool is built on NLTK, Natural Language Toolkit. I then downloaded vader_lexicon from the libraries. Next, I define a variable as the Sentiment Intensity Analyzer tool from VADER. I ran some tests to see how VADER would classify two sentences. After seeing the tool was working as intended, it was time for the lyrics. Initially, I tried using the outputted list data of the lyrics, but VADER struggled when it wasn't UTF-8 encoded. I then tried stripping out the integers and encoding the list in UTF-8, but I felt this was far too much work when I could just rerun the AZLyrics API.

To analyze the lyrics, I created a for-loop with try-except blocks. I defined a sentiment score and song name list for storing the data. I iterate through the ChartData entries within the for-loop, inputting each song artist and title into the AZLyrics API. The except block catches the index errors. If lyrics are found, it passes through to an if-else block. The if-else block checks to see if the lyrics are the zeros outputted when the lyrics aren't found. If it is a zero, the entry is passed. If the lyrics are indeed lyrics, I store the song name using the billboard data and store the sentiment score in the lists. VADER Sentiment Intensity Analyzer outputs four scores: percent negativity, percent neutrality, percent positivity, and the compound score, the sum of the previous three scores.

With the sentiment data stored in the list, I used pandas to create a DataFrame of the scores. As I mentioned, I stored each song's name and the scores. This is important for my visualization work later because I won't know which scores correlate to which song without it. I added a new column to the DataFrame and filled it with the song name data. Here is the outputted DataFrame that I created:

	neg	neu	pos	compound	name
0	0.173	0.660	0.166	-0.9416	Cruel Summer
1	0.213	0.712	0.075	-0.9982	Paint The Town Red
2	0.101	0.745	0.153	0.9669	Snooze
3	0.052	0.945	0.004	-0.9655	Monaco
4	0.044	0.843	0.113	0.9821	Fast Car
...	...	...	...	...	...
58	0.028	0.965	0.007	-0.6808	LaLa
59	0.267	0.666	0.067	-0.9991	SkeeYee
60	0.144	0.658	0.198	0.9814	Everything I Love
61	0.026	0.610	0.364	0.9995	Better Place
62	0.152	0.812	0.037	-0.9938	Truck Bed

What I found interesting about this was that a song like Better Place by *NSYNC, which has minimal swearing, got a very high compound sentiment score, indicating that it is very positive. Yet, a song like SkeeYee by Sexyy Red, about hollering at a girl in your expensive car with a lot of swearing, got such a negative score. This indicates that VADER, when it sees swearing, defines it as unfavorable despite not being used negatively in the song.

Song features using Spotify

With the popular song lyrics' sentiment found through VADER, it was time to find if popular songs are danceable, indicating they could be found in a TikTok trend. The Spotify API has a wonderful function that gets the audio feature information for a single track using its unique Spotify ID. It outputs metadata on a song's tempo, key, length, etc. It also outputs statistics developed by Spotify on danceability, energy, acoustics, liveness, etc. To access these features, you must get a Spotify API client ID and user secret ID from their developer page. Then, using requests, create the response features filled with the aforementioned credentials. After the credentials are sent into the API, an access token is generated, and you can begin accessing the API.

To begin the analysis, I first needed to get the unique Spotify ID for each song on the Billboard Top 100 list. I created two lists to store IDs and track names. In a for-loop, I iterate through the chart data and search for each track by inputting the song name in the middle of the API search URL. The ID is appended from the metadata by searching the "list literals" for the ID. After the ID is found, it is appended to the ID list, and the track name is taken from the ChartData. Initially, I also took the song name from Spotify by searching for the name, but within the Spotify data, the track names include the features of the song. This later ruined my DataFrame merge because it was trying to match song names with song names and features. With the unique IDs stored in a list, I can search for the audio features of the songs. I defined a list to store features in. I use a for-loop to iterate through the IDs list and request the audio features on the song. Then, the list adds the features to itself. Now that the features are stored in a list, I put the data in a DataFrame the same way I did with the sentiment scores. I then added the names of each song in a new column.

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	type	
0	0.552	0.702	9.0	-5.707	1.0	0.1570	0.11700	0.000021	0.1050	0.564	169.994	audio_features	1BxfuPKGuaTgP7aM0Bb
1	0.868	0.538	5.0	-8.603	1.0	0.1740	0.26900	0.000003	0.0901	0.732	99.968	audio_features	2IGMVunIBsBLtEQyoI1M
2	0.559	0.551	5.0	-7.231	1.0	0.1320	0.14100	0.000000	0.1100	0.392	143.008	audio_features	4iZ4pt7kvcaH6Yo8UoZ
3	0.663	0.670	8.0	-8.399	1.0	0.2710	0.04640	0.000089	0.2050	0.138	136.952	audio_features	2YSzYUF3jWqb9YP9VXm
4	0.787	0.621	4.0	-5.009	0.0	0.0680	0.15000	0.000402	0.5800	0.130	139.056	audio_features	4MjDJD8cW7iVeWInc2E
...	...	...	...	...	...	...	...	...	...	...	...	...	
95	0.568	0.841	8.0	-3.802	1.0	0.0261	0.00514	0.000503	0.1470	0.668	103.983	audio_features	03fs6oV5JAlbytRYf33
96	0.675	0.546	4.0	-7.716	0.0	0.0511	0.55600	0.004410	0.1670	0.607	77.415	audio_features	7DswEZZthZ6piQpL25qG
97	0.544	0.496	0.0	-12.032	1.0	0.2100	0.13000	0.000000	0.1660	0.314	103.148	audio_features	3kfhS2L56Wj8fBcu7xE4
98	0.579	0.556	4.0	-6.277	0.0	0.0511	0.02140	0.000000	0.0904	0.558	116.943	audio_features	1bHnRc60O1N0l3PbHjaK
99	0.662	0.770	0.0	-4.887	1.0	0.0367	0.08750	0.000000	0.3500	0.769	132.057	audio_features	103u49mLoJUge83B1Cr

100 rows × 20 columns

l	uri	track_href	analysis_url	duration_ms	time_signature	error	name
r	spotify:track:1BxfuPKGuaTgP7aM0Bbdwr	https://api.spotify.com/v1/tracks/1BxfuPKGuaTg...	https://api.spotify.com/v1/audio-analysis/1Bxf...	178427.0	4.0	NaN	Cruel Summer
'	spotify:track:2IGMVunIBsBLtEQyoI1Mu7	https://api.spotify.com/v1/tracks/2IGMVunIBsBL...	https://api.spotify.com/v1/audio-analysis/2IGM...	231750.0	4.0	NaN	Paint The Town Red
!	spotify:track:4iZ4pt7kvcaH6Yo8UoZ4s2	https://api.spotify.com/v1/tracks/4iZ4pt7kvcaH...	https://api.spotify.com/v1/audio-analysis/4iZ4...	201800.0	4.0	NaN	Snooze
:	spotify:track:2YSzYUF3jWqb9YP9VXmpjE	https://api.spotify.com/v1/tracks/2YSzYUF3jWqb...	https://api.spotify.com/v1/audio-analysis/2YSz...	260111.0	4.0	NaN	IDGAF
j	spotify:track:4MjDJD8cW7iVeWInc2Bdyj	https://api.spotify.com/v1/tracks/4MjDJD8cW7iV...	https://api.spotify.com/v1/audio-analysis/4MjD...	267194.0	4.0	NaN	Monaco
.	...	...	...	...	...	...	...
;	spotify:track:03fs6oV5JAlbytRYf3371S	https://api.spotify.com/v1/tracks/03fs6oV5JAlb...	https://api.spotify.com/v1/audio-analysis/03fs...	187047.0	4.0	NaN	Everything I Love
l	spotify:track:7DswEZZthZ6piQpL25qGAM	https://api.spotify.com/v1/tracks/7DswEZZthZ6p...	https://api.spotify.com/v1/audio-analysis/7Dsw...	184258.0	4.0	NaN	On My Mama
l	spotify:track:3kfhS2L56Wj8fBcu7xE47H	https://api.spotify.com/v1/tracks/3kfhS2L56Wj8...	https://api.spotify.com/v1/audio-analysis/3kfh...	203442.0	5.0	NaN	Turks & Caicos
:	spotify:track:1bHnRc60O1N0l3PbHjaKyK	https://api.spotify.com/v1/tracks/1bHnRc60O1N0...	https://api.spotify.com/v1/audio-analysis/1bHn...	216667.0	4.0	NaN	Better Place
l	spotify:track:103u49mLoJUge83B1CnLxl	https://api.spotify.com/v1/tracks/103u49mLoJUg...	https://api.spotify.com/v1/audio-analysis/103u...	167535.0	4.0	NaN	Truck Bed

As you can see, each row is a song, and each column is filled with statistics on the song. The error column is because a rate limit was hit on three songs near the end. In hindsight, I should have delayed the code with sleep() to give the API time to breathe.

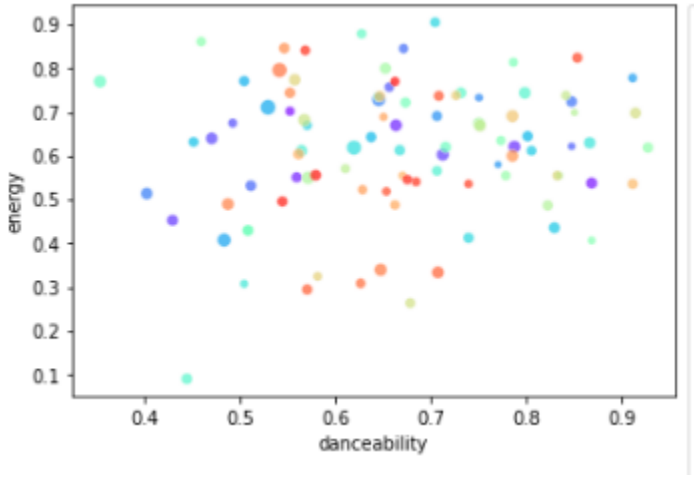
Analysis through Visualizations

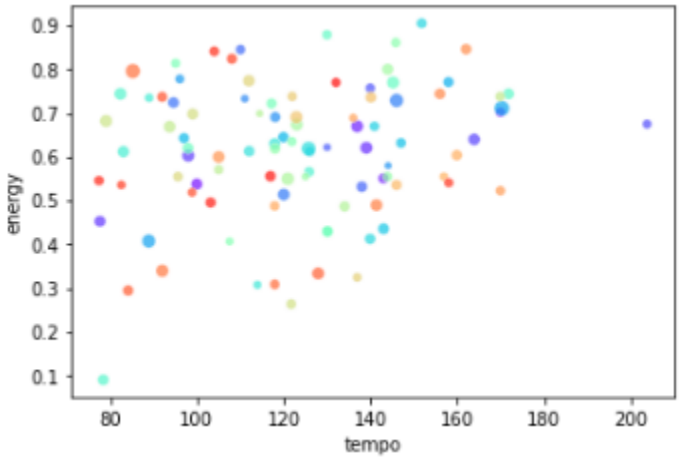
To look at both the sentiment data and the audio feature data in the same visualizations, I merged the DataFrames using pandas. I inner merged using the name column because Spotify error coded on three songs, and AZLyrics couldn't find lyrics on 37 songs. By merging the names, I wouldn't

```
songfeaturessentiment = pd.merge(
    left = df,
    right = sentimentdf,
    left_on = ['name'],
    right_on = ['name'],
    how = 'inner'
)
```

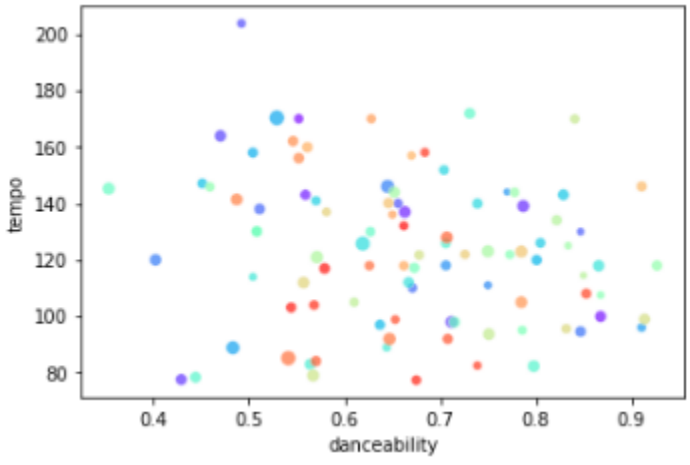
have any null data of songs found using Spotify but not AZLyrics.

Using Seaborn, I created a variety of scatterplots using the data collected. This visualization looks at each song's danceability compared to its energy. Danceability is an aggregate measure of each song's tempo, rhythm stability, beat strength, and regularity. A score is calculated through these features. Energy measures how intense a song is, using dynamic range, loudness, and timbre as metrics. Essentially, danceability measures how repetitive, and energy measures how active a song is. The plot shows that most popular songs are high-energy, but danceability doesn't affect popularity.

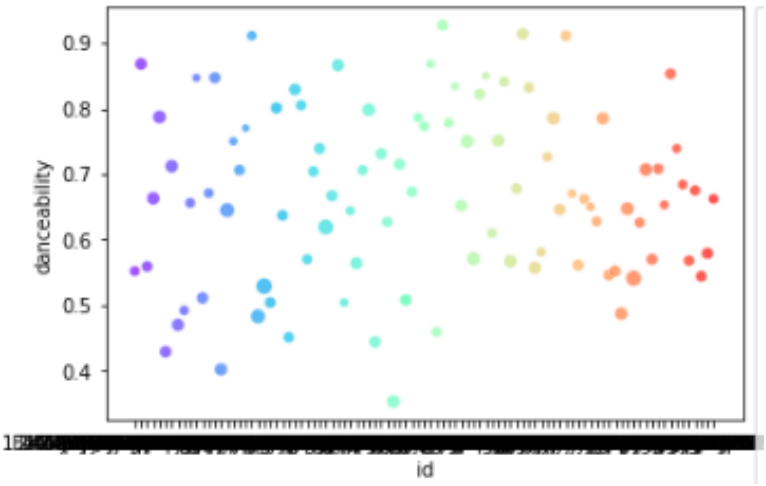




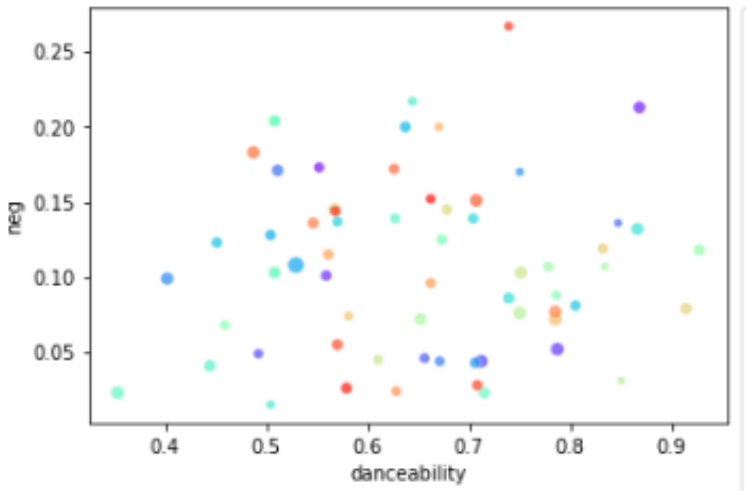
This plot explores energy and tempo. Most popular songs don't go past a tempo of 180 bpm.



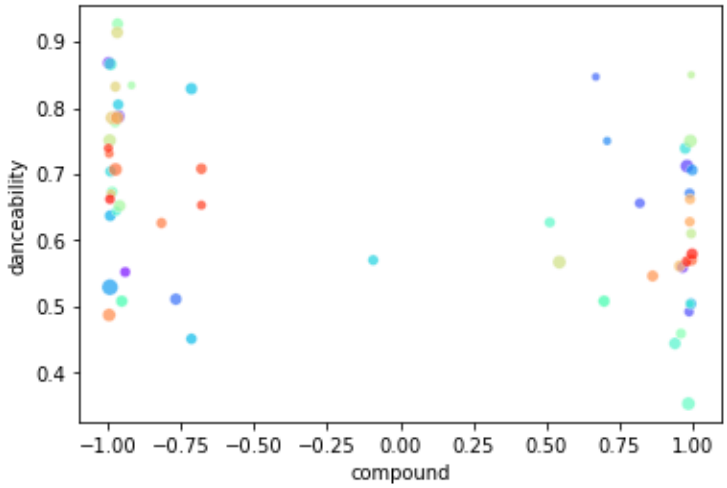
This plot explores tempo and danceability. Interestingly enough, a 200 bpm song is below average danceability.



This plot explores Danceability across songs, showing the variety of danceability in popular songs. The distribution is relatively normal, indicating that danceability and popularity aren't related.

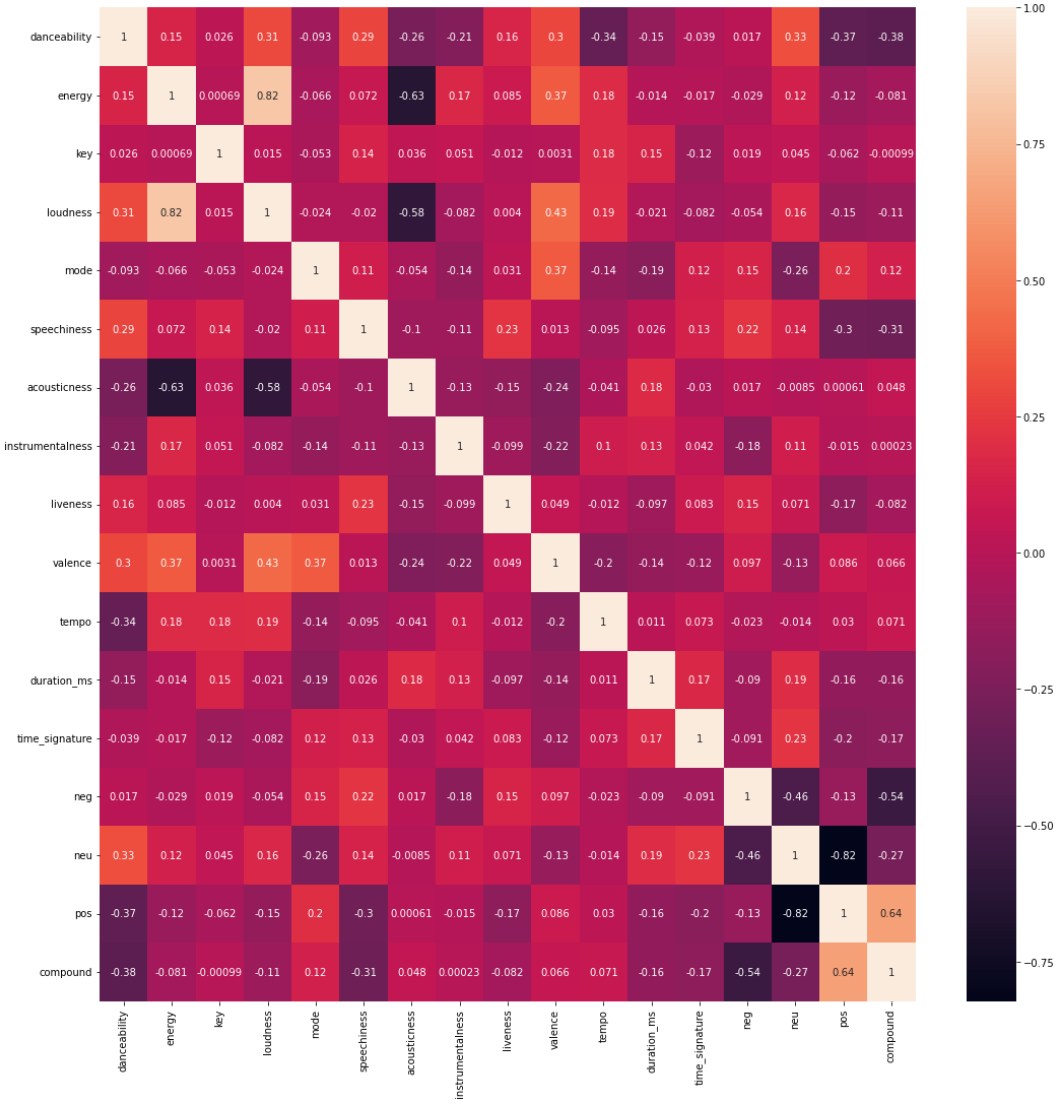


This plot explores the negativity of lyrics and danceability. There is no correlation between the lyrics' negativity and the song's danceability.



This plot explores the lyrics' compound sentiment and the beat's danceability. A score of -1 means the lyrics are overall negative, 0 being neutral, and +1 being overall positive. As you can see, plenty of positive and negative lyrical songs are very danceable.

The best way to understand if there is a correlation between any of the features is to create a correlation heatmap. Here is the heatmap of correlations generated using Seaborn.



There is a correlation for some features like loudness and energy because one is used to calculate the other. There is a minor correlation between the neutrality of lyrics and danceability. There is a negative correlation between the positivity of lyrics and danceability. This is likely due to genres like folk and pop often telling a story with an evolving beat, but genres like rap have swearing and a consistent beat.

In conclusion, there is little correlation between popular songs having a danceable beat. Songs with danceable beats don't always have positive lyrics. Through this data journey, I have learned that popular songs are popular for more reasons than danceability and TikTok. The depth of music we have today, with new subgenres being developed and bands filling the niches, is

fascinating. A popular song isn't always pop or folk hits by prominent artists. The top 100 is filled with many different genres, encapsulating how excellent listening taste is today. TikTok has expanded people's listening genres and allowed them to enjoy music released in the past.

Work for the Future

If I continue working on this topic, I would look into TikTok's API and see if they collect data on audio creators' use. I might also look at TikTok trends and how, over time, the artists featured in the trends changed in popularity. I could use the Spotify API, which assigns popularity scores based on monthly streams. This would be an exciting look at how artists featured in trends change because of their popularity on social media sites.

Sources and a thank you

Thank you so much for reading my write-up. I had a great time working on this project. Overall, I am proud of what I accomplished.

Resources

Whateley, Dan. "How Tiktok Is Changing the Music Industry and the Way We Discover New, Popular Songs." Business Insider, Business Insider, www.businessinsider.com/how-tiktok-is-changing-music-industry#:~:text=Songs%20that%20trend%20on%20TikTok,music%2Danalytics%20company%20MRC%20Data.

Python

Billboard API: https://github.com/guoguo12/billboard-charts

Spotify API: https://developer.spotify.com/

AZlyrics API: https://github.com/elmoiv/azapi

VADER: https://github.com/cjhutto/vaderSentiment

Matplotlib: https://matplotlib.org/

Seaborn: https://seaborn.pydata.org/

Numpy: https://numpy.org/