Songs

December 11, 2022

[ ]:
```
In this journal, I work with the songs on my computer. I have a playlist on my␣
↪computer where I stored all of my favorite songs that I listened to over the␣
↪years. There is about 90 of these songs that are mp3 files on my computer.␣
↪The mp3 files for these songs are in the song_data folder. To do processing␣
↪on them, they needed to be converted to wav files. These wav files are␣
↪stored in the wav folder. My goal with the code below was to define the mood␣
↪of the song as relating to its tempo and loudness in which I would try to␣
↪group my songs together based on these two measurements. The groups that are␣
↪formed would be the recommended playlists that would be generated for my␣
↪group of songs.
```

[1]:
```
from IPython.display import Audio
from os import path
from pydub import AudioSegment
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import libfmp.b
import math
import sys
sys.path.append('/path/to/ffmpeg')
from tqdm import tqdm
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import pandas as pd
```

[2]:
```
song_path = "song_data"
```

[ ]:
```
The song_dict is where I store the name, tempo, and loudness for each of the␣
↪songs on my computer.
```

[3]:
```
song_dict = {"name": [], "tempo": [], "loudness": []}
```

[ ]:
```

```

```
The code below is where for each of my mp3 songs, I first convert them into wav␣
↪files and move them to the wav folder, and then do processing on them to␣
↪extract their tempo and loudness. I focused on tempo and loudness since␣
↪these are the two features of a song that most determine its mood and that I␣
↪was able to extract from the mp3 audio files. This extraction was done using␣
↪the librosa library. For the tempo, the librosa.beat.tempo function was able␣
↪to gather this data from the wav file. One parameter that I had to configure␣
↪was the start_bpm which is the baseline the function uses to indicate the␣
↪tempo of a song. I stated that this baseline should be 100 bpm since this is␣
↪about the average tempo for a song. For loudness, I used the librosa root␣
↪mean square method. It returns a list consisting of the energy of the song␣
↪measured for each frame of the song. Hence, it returns how loud a song is␣
↪for all parts of the song. To convert the loudness of the whole song, I␣
↪averaged these values together such that I would see in aggregate, how loud␣
↪a song is.
```

[4]:
```
files = librosa.util.find_files(song_path, ext=['mp3'])
files = np.asarray(files)
for file in tqdm(files):
    src = AudioSegment.from_mp3(file)
    name = "wav/" + file.split("/")[len(file.split("/")) - 1].split(".")[0]
    src.export(name, format="wav")
    y, sr = librosa.load(name)
    tempo = librosa.beat.tempo(y=y, sr=sr, start_bpm = 100)
    loudness = librosa.feature.rms(y=y).mean()
    song_dict["name"].append(name)
    song_dict["tempo"].append(tempo)
    song_dict["loudness"].append(loudness)
```

```
100%|                    | 89/89 [17:19<00:00, 11.68s/it]
```

[ ]:
```
The results of the data collection process is shown below.
```

[5]:
```
song_df = pd.DataFrame.from_dict(song_dict)
```

[6]:
```
song_df
```

[6]:

	name	tempo	loudness
0	wav/- Mekaku city Actors op	[99.38401442307692]	0.305389
1	wav/01 Guilty Crown op 1	[112.34714673913044]	0.310536
2	wav/01 HIGHSCHOOL OF THE DEAD	[95.703125]	0.295538
3	wav/01 Shuffle	[86.1328125]	0.244970
4	wav/01 only my railgun	[95.703125]	0.294948
..	…	…	…
84	wav/Yahari op 2	[112.34714673913044]	0.147142
85	wav/Yahari op 3	[92.28515625]	0.306154
86	wav/Yosuga No Sora	[99.38401442307692]	0.240719

```
87       wav/Your Lie in April op       [107.666015625]  0.111587
88  wav/tonari no kaibutsu kun op         [92.28515625]  0.255281

[89 rows x 3 columns]
```

[ ]:
```
I removed the wav part of the names such that they were more readable in the␣
↪table. Then, with the tempo, I converted the values from list format to␣
↪float format such that data processing could be done on them.
```

[15]:
```
song_df["name"] = song_df["name"].apply(lambda x: x.split("/")[1])
song_df
```

[15]:
```
                         name                  tempo  loudness
0     - Mekaku city Actors op   [99.38401442307692]  0.305389
1        01 Guilty Crown op 1  [112.34714673913044]  0.310536
2   01 HIGHSCHOOL OF THE DEAD           [95.703125]  0.295538
3                  01 Shuffle          [86.1328125]  0.244970
4          01 only my railgun           [95.703125]  0.294948
..                         …                   …       …
84               Yahari op 2  [112.34714673913044]  0.147142
85               Yahari op 3         [92.28515625]  0.306154
86            Yosuga No Sora   [99.38401442307692]  0.240719
87       Your Lie in April op       [107.666015625]  0.111587
88  tonari no kaibutsu kun op         [92.28515625]  0.255281

[89 rows x 3 columns]
```

[17]:
```
song_df["tempo"] = song_df["tempo"].apply(lambda x: x[0])
song_df
```

[17]:
```
                         name       tempo  loudness
0     - Mekaku city Actors op   99.384014  0.305389
1        01 Guilty Crown op 1  112.347147  0.310536
2   01 HIGHSCHOOL OF THE DEAD   95.703125  0.295538
3                  01 Shuffle   86.132812  0.244970
4          01 only my railgun   95.703125  0.294948
..                         …         …       …
84               Yahari op 2  112.347147  0.147142
85               Yahari op 3   92.285156  0.306154
86            Yosuga No Sora   99.384014  0.240719
87       Your Lie in April op  107.666016  0.111587
88  tonari no kaibutsu kun op   92.285156  0.255281

[89 rows x 3 columns]
```

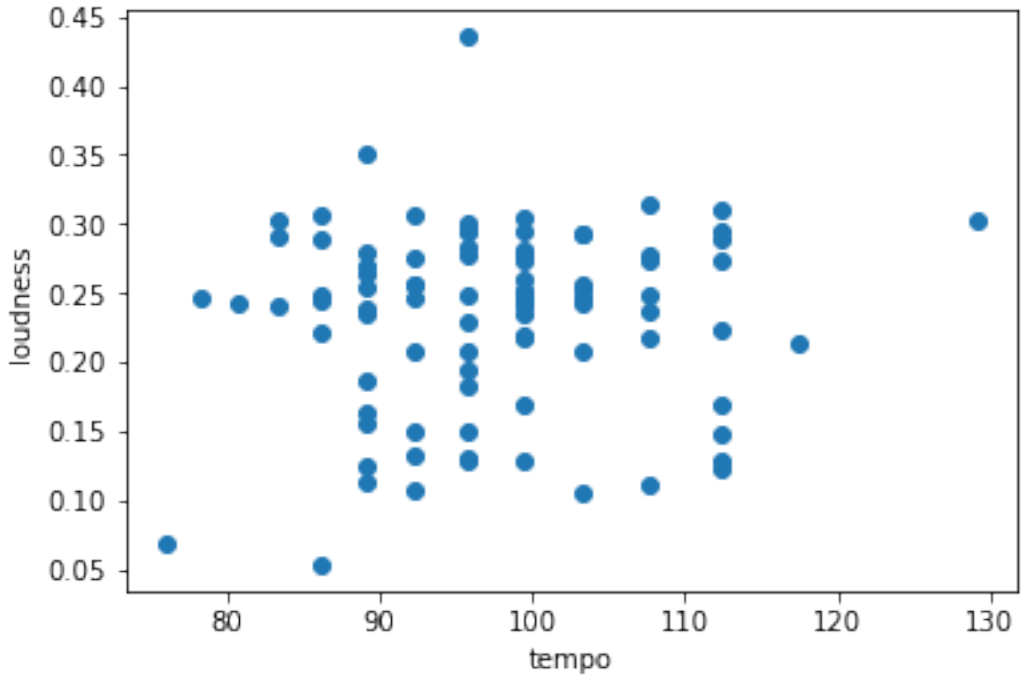
[ ]:
```

```

The scatterplot below shows how my songs are distributed in terms of tempo and loudness. We can see that the points are distributed evenly across the graph which means that meaningful clusters can be formed. We want these clusters to be based off of the moods that a song can be. Using the ideas of energy and valence in Spotify, we would want to create 4 clusters. This is because with the Spotify mood-based recommender system, the mood of a song was based on whether it had high or low energy and high or low valence creating 4 quadrants that a song can be in with respect to mood.

[19]:
```
plt.scatter(song_df["tempo"], song_df["loudness"])
plt.xlabel("tempo")
plt.ylabel("loudness")
```

[19]:
```
Text(0, 0.5, 'loudness')
```



[ ]:
In the code below, the points are clustered together into 4 groups. This is done using KMeans which applys centroids to the scatterplots in which the points closest to that centroid would be put into that specific group. The 4 centroids used in the clustering is shown by the scatterplot below. Based on the placement of the centroids, we can see that the clustering is largely occuring by the differences in song tempo since this is where the larger variation in the song data occurs.

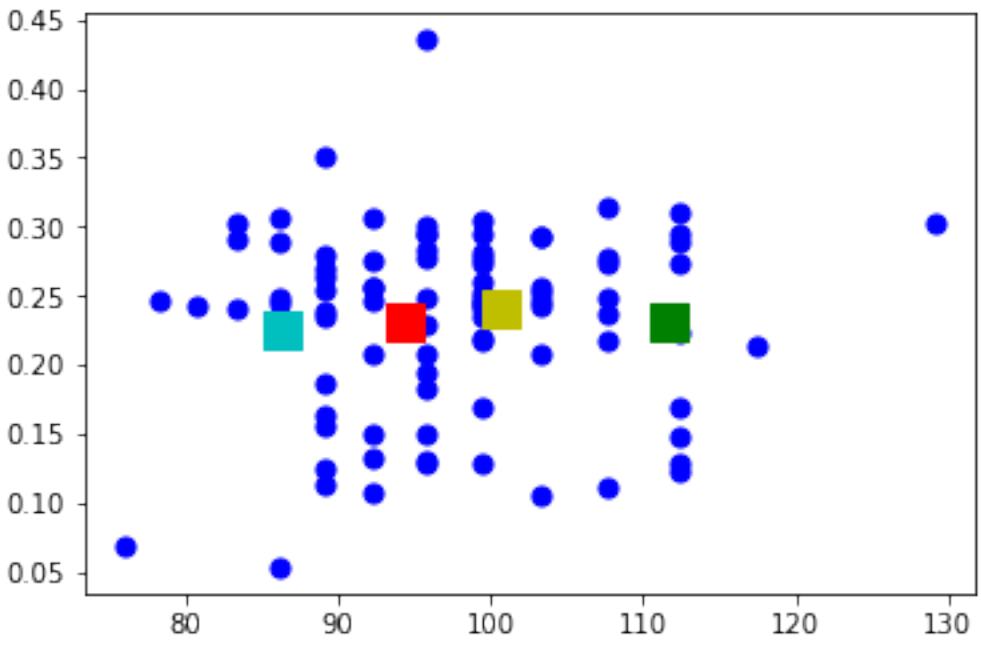
[24]:
```
X = song_df.drop(["name"], axis=1)
Kmean = KMeans(n_clusters=4)
Kmean.fit(X)
```

[24]:
```
KMeans(n_clusters=4)
```

[32]:
```
centroids = Kmean.cluster_centers_
centroids
```

[32]:
```
array([[111.746638  ,   0.23081718],
       [ 94.36565897,   0.23020894],
       [ 86.29776674,   0.22474839],
       [100.70913462,   0.24122443]])
```

[35]:
```
plt.scatter(X["tempo"], X["loudness"], s =50, c="b")
plt.scatter(centroids[0][0], centroids[0][1], s=200, c="g", marker="s")
plt.scatter(centroids[1][0], centroids[1][1], s=200, c="r", marker="s")
plt.scatter(centroids[2][0], centroids[2][1], s=200, c="c", marker="s")
plt.scatter(centroids[3][0], centroids[3][1], s=200, c="y", marker="s")
plt.show()
```



[ ]:
```
The Kmean.labels_ shows the cluster that each of the songs were grouped into.␣
↪These results can be inputted into our original song dataframe where we can␣
↪then analyze the characteristics of each cluster.
```

[36]:
```
Kmean.labels_
```

[36]:
```
array([3, 0, 1, 2, 1, 1, 2, 1, 3, 1, 2, 1, 0, 0, 0, 2, 2, 2, 2, 3, 3, 1,
       2, 1, 2, 1, 2, 3, 0, 2, 3, 2, 1, 0, 1, 3, 1, 3, 3, 2, 1, 0, 3, 0,
       3, 2, 2, 2, 2, 0, 3, 3, 1, 1, 0, 1, 3, 3, 0, 2, 1, 1, 3, 2, 2, 2,
       2, 3, 1, 1, 2, 3, 3, 3, 0, 0, 0, 3, 1, 0, 3, 3, 2, 0, 0, 1, 3, 0,
       1], dtype=int32)
```

[37]:
```
song_df["Cluster Group"] = Kmean.labels_
song_df
```

[37]:

	name	tempo	loudness	Cluster Group
0	- Mekaku city Actors op	99.384014	0.305389	3
1	01 Guilty Crown op 1	112.347147	0.310536	0
2	01 HIGHSCHOOL OF THE DEAD	95.703125	0.295538	1
3	01 Shuffle	86.132812	0.244970	2
4	01 only my railgun	95.703125	0.294948	1
..	…	…	…	…
84	Yahari op 2	112.347147	0.147142	0
85	Yahari op 3	92.285156	0.306154	1
86	Yosuga No Sora	99.384014	0.240719	3
87	Your Lie in April op	107.666016	0.111587	0
88	tonari no kaibutsu kun op	92.285156	0.255281	1

[89 rows x 4 columns]

[ ]:
```
Cluster zero is represented by the green square and features the fast songs␣
↪that are generally loud. Hence, these would be the happy and energetic pop␣
↪songs. Looking at the songs in the cluster, the songs generally match this␣
↪description but there do seem to be some exceptions with some more sadder␣
↪songs like Guilty Crown present.
```

[40]:
```
cluster_0 = song_df.loc[song_df["Cluster Group"] == 0]
print(len(cluster_0))
cluster_0
```

```
18
```

[40]:

	name	tempo	loudness	Cluster Group
1	01 Guilty Crown op 1	112.347147	0.310536	0
12	Ao no Kanata Rhythm op	112.347147	0.223902	0
13	Asterisk War op 1	107.666016	0.315073	0
14	Bakemonogatari Snake	117.453835	0.212813	0
28	Dorm opening	107.666016	0.216927	0
33	Future Diary op 1	112.347147	0.289750	0
41	HxH op	107.666016	0.277394	0
43	IS op 2	112.347147	0.293973	0

49	Mayo Chiki op	112.347147	0.128505	0
54	Naruto op 10	129.199219	0.301697	0
58	Naruto op 9	107.666016	0.248947	0
74	Shomin Sample op	112.347147	0.122309	0
75	Soul Eater op 2	107.666016	0.273981	0
76	Soul eater opening 1	107.666016	0.237215	0
79	TIAMHS op 2	112.347147	0.170019	0
83	Yahari op 1	112.347147	0.272939	0
84	Yahari op 2	112.347147	0.147142	0
87	Your Lie in April op	107.666016	0.111587	0

[ ]:
```
Looking at the songs in this cluster, it seems more Rock-heavy then compared to␣
↪the other cluster in which the songs are more intense in nature and not as␣
↪cheerful as the other. However, same as before, there are some exceptions in␣
↪which there are some songs in the playlist that I thought would make sense␣
↪to group in cluster 1. In terms of the scatterplot, this cluster is␣
↪represented as the red square.
```

[41]:
```
cluster_1 = song_df.loc[song_df["Cluster Group"] == 1]
print(len(cluster_1))
cluster_1
```

```
23
```

[41]:

	name	tempo	loudness	Cluster Group
2	01 HIGHSCHOOL OF THE DEAD	95.703125	0.295538	1
4	01 only my railgun	95.703125	0.294948	1
5	ANY op	92.285156	0.106582	1
7	Absolute Duo op	95.703125	0.276569	1
9	Akame Ga Kill op 1	92.285156	0.246980	1
11	Angel Beats op	95.703125	0.182298	1
21	C3 op	92.285156	0.131597	1
23	D Frag op	95.703125	0.300470	1
25	DAL op 2	95.703125	0.206858	1
32	Food Wars op	95.703125	0.282486	1
34	Gamers op	92.285156	0.208451	1
36	Golden Time op	95.703125	0.229785	1
40	Hi Score Girl op	95.703125	0.248953	1
52	Nanatsu No Taizai	95.703125	0.127815	1
53	Naruto Shippuden op 19	92.285156	0.275837	1
55	Naruto op 15	95.703125	0.435554	1
60	Noragami op 2	95.703125	0.131273	1
61	Nyaruko Op 1	95.703125	0.149608	1
68	SAO op 2	92.285156	0.257109	1
69	SAO op 3	92.285156	0.149893	1
78	TIAMHS op 1	95.703125	0.194768	1
85	Yahari op 3	92.285156	0.306154	1

```
88  tonari no kaibutsu kun op  92.285156  0.255281              1
```

[ ]:
```
Looking at cluster 2, In the context of the scatterplot, this cluster is the␣
↪cyan square with the lower tempos. In terms of the mood of the songs, there␣
↪seem to be a mix between the chill pop and more sadder songs. This is likely␣
↪due to the fact that both of these types of songs would have a lower tempo.
```

[42]:
```
cluster_2 = song_df.loc[song_df["Cluster Group"] == 2]
print(len(cluster_2))
cluster_2
```

```
24
```

[42]:

	name	tempo	loudness	Cluster Group
3	01 Shuffle	86.132812	0.244970	2
6	AOT op 1	89.102909	0.239603	2
10	Akame Ga Kill op 2	89.102909	0.113813	2
15	Bakemonogatari main	86.132812	0.247472	2
16	Beatless op	86.132812	0.289109	2
17	Benkyou Dekinai op 1	89.102909	0.253351	2
18	Benkyou Dekinai op 2	80.749512	0.243105	2
22	Crossing Field	89.102909	0.164160	2
24	DAL op 1	89.102909	0.155754	2
26	DAL op 3	83.354335	0.241058	2
29	Dungeon ni	89.102909	0.185579	2
31	Fate Zero op 1	75.999540	0.068741	2
39	Hentai Ouji op	86.132812	0.306851	2
45	Jojo op 2	89.102909	0.279179	2
46	Kanokari op 2	83.354335	0.303363	2
47	Masamune op	89.102909	0.125102	2
48	Mashiro Symphony op	89.102909	0.264547	2
59	Nisekoi op 1	89.102909	0.270292	2
63	OPM Op	86.132812	0.053228	2
64	Oreshura	83.354335	0.290509	2
65	Plastic Memories op	78.302557	0.247116	2
66	Romeo and Juliet op	86.132812	0.221297	2
70	Sakurasou op 1	89.102909	0.235534	2
82	Unravel	89.102909	0.350230	2

[ ]:
```
Cluster 3 is represented in the scatterplot as the yellow square. Based on the␣
↪mood of the songs below, I see songs that seem more neutral in mood compared␣
↪to the other clusters. The songs most features general pop and rock songs␣
↪that would be like the songs that would play in a store while you are␣
↪shopping. Thus, this can be considered an extraneous cluster for songs that␣
↪may not have been a perfect fit for the other categories.
```

[44]:
```
cluster_3 = song_df.loc[song_df["Cluster Group"] == 3]
print(len(cluster_3))
cluster_3
```

```
24
```

[44]:

	name	tempo	loudness	Cluster Group
0	- Mekaku city Actors op	99.384014	0.305389	3
8	Accel World op	99.384014	0.247538	3
19	Bleach op 3	99.384014	0.294392	3
20	Blend S	103.359375	0.245582	3
27	Danganronpa Op	99.384014	0.277578	3
30	Familiar of Zero	103.359375	0.208721	3
35	Gate op	99.384014	0.169347	3
37	Gotoubun Op	103.359375	0.293748	3
38	Haruhi op	99.384014	0.217101	3
42	IS op 1	103.359375	0.243472	3
44	Jojo op 1	99.384014	0.259829	3
50	My Hero Academia	99.384014	0.235716	3
51	NGNL Op	99.384014	0.129106	3
56	Naruto op 2	99.384014	0.244666	3
57	Naruto op 5	103.359375	0.105599	3
62	Nyaruko op 2	103.359375	0.255352	3
67	SAO II OP Courage Full	99.384014	0.245472	3
71	Senko Op	103.359375	0.251692	3
72	Shana op	99.384014	0.219822	3
73	Shin no Nakama Opening	99.384014	0.273935	3
77	Sousei op 1	99.384014	0.281086	3
80	Trinity Seven	103.359375	0.292037	3
81	Undefeated Bahamut Chronicles op	99.384014	0.251487	3
86	Yosuga No Sora	99.384014	0.240719	3

[ ]:
```
In conclusion, song recommendation is a difficult issue to solve as it requires␣
↪accurate measurements and selection of features in the song to use for the␣
↪recommendation. In the case of the recommendation in this journal, I felt␣
↪that I could identify some patterns in some of the clusters that were formed␣
↪but I that there were songs in the clusters that were distinct from the␣
↪patterns that I recognized. In general, I believe that tempo was a good way␣
↪to differentiate the songs by mood, but I feel that loudness did not do␣
↪enough to do the same. To improve this, I would need to consider more␣
↪features in music that help in explaining the mood of a song. This will␣
↪require more research in the field of music theory which can then be applied␣
↪to this project in the future.
```