Jack Leeper
CPTS 315
Project Report

Introduction:

The data mining task I decided to take on is a recommender system for songs. I wanted to make a program that could take a sample of user liked songs, and then based off of those songs, generate some similar songs that the user may also enjoy. This is a very common feature among almost every modern music playing application, I myself found Spotify's discover weekly playlist, or just recommended songs in general to be very accurate and an inspiration for me to want to recreate something similar to this.

Dataset:

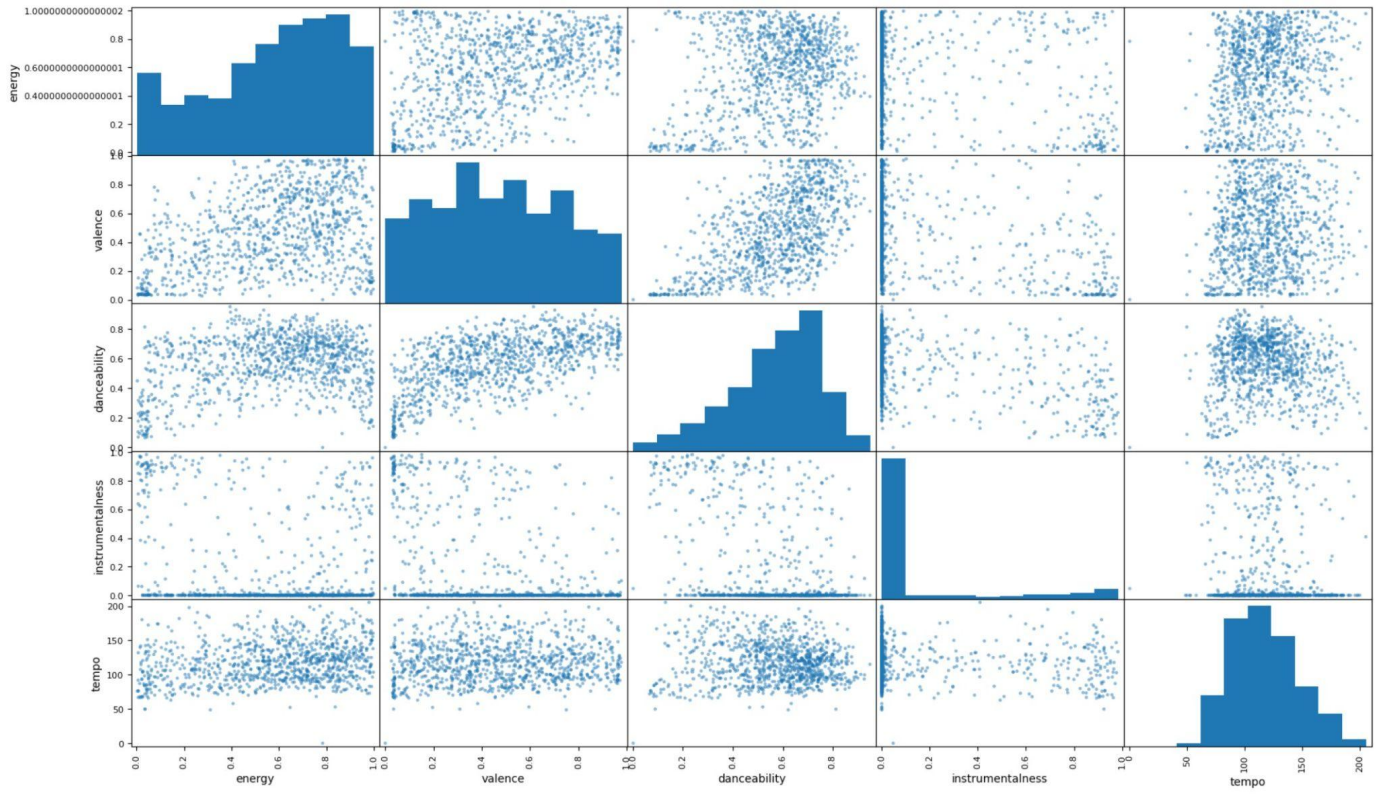
The data set I used (https://www.kaggle.com/datasets/saurabhshahane/spotgen-music-dataset/versions/122) which is also included in the source files, was a kaggle dataset containing a massive list of songs with very relevant attributes displayed for each song including valence, instrumentalness, danceability, popularity, energy, tempo, all sorts of attributes to compare and contrast with each other. This is key for the methodology I used to complete the task, but first I had to organize the dataset, so using ***pandas*** I read the csv file of all the Tracks in the dataset, making a Tracks dataframe that consisted of everything I needed and so afterwards I was ready to begin trying to use this data to complete my task.

Methodology:

With the data formatted, and usable, now comes how do I take these attributes and do something useful with them. Given the data I have, with various attributes, I think it would be best to clusterize the data and generate recommendations via clustered groupings of songs. So that's the approach I took.
To illustrate the data points, I made a quick scatter matrix using pandas.

```
reduced_df = tracks_df[:1000]
pd.plotting.scatter_matrix(reduced_df[['energy', 'valence',
'danceability', 'instrumentalness', 'tempo']], figsize=(20,12))
```

I reduced the dataset to save time, but it's still effective in illustrating the plot.

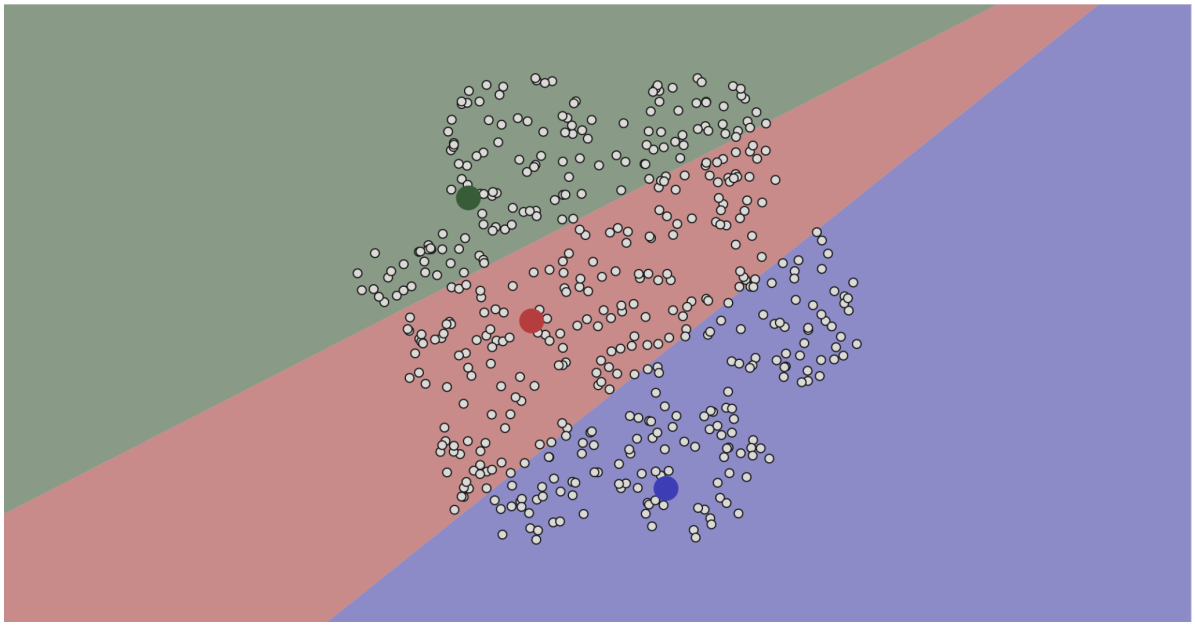


With the image above we can visually see some correlations between the attributes, some almost appear to make linear lines obviously with lots of variation but the correlation is still apparent.

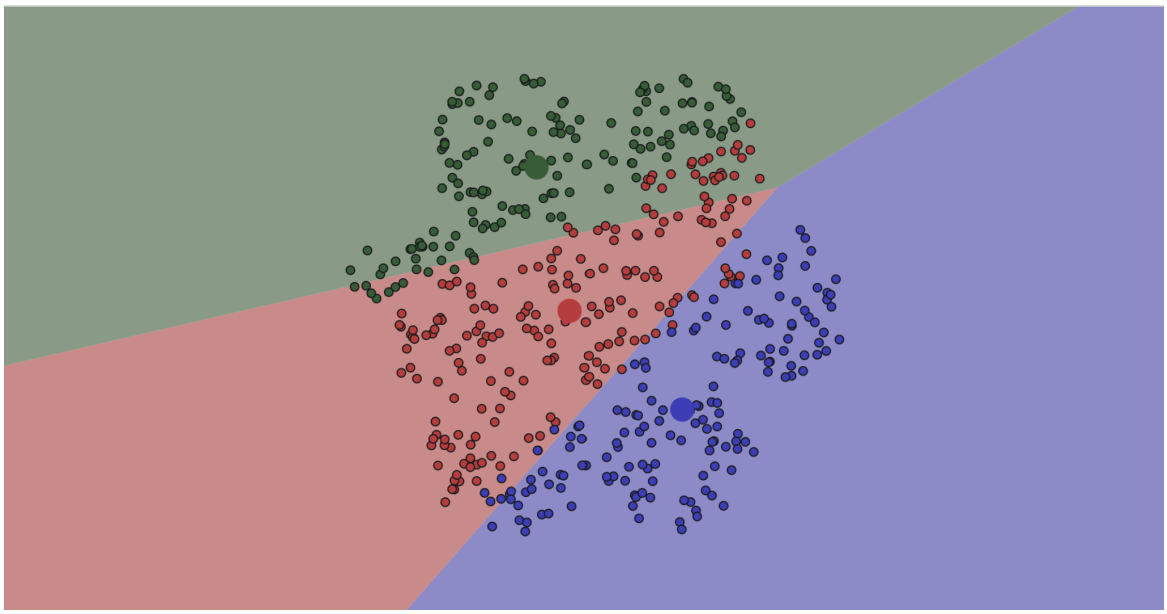
These various data points are what I need to clusterize, in order to sort the songs into similar categories.

(In the “makeResultFile” program) Using the *Sci-kit/sklearn* library, I implemented a *KMeans* model out of the data grouping tracks into separate clusters based on their attributes, the idea being songs of the same cluster are going to be similar.
The exact attributes that were used to make these clusters were 'energy', 'valence', 'danceability', 'instrumentalness', and 'tempo'. The amount of clusters to make was tricky to determine, too few and multiple different genres are grouped together, while if you make too many then songs of the same genre will be found in various other clusters. I somewhat arbitrarily chose 5 clusters and got results that seemed adequate, so 5 clusters is the amount I stuck with.
To illustrate how KMeans clusters the data I used this website that helps me with doing exactly that. https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

To start off, let’s say you have a scatter plot with many varying data points. I added three different centroids to this illustration which were placed randomly.

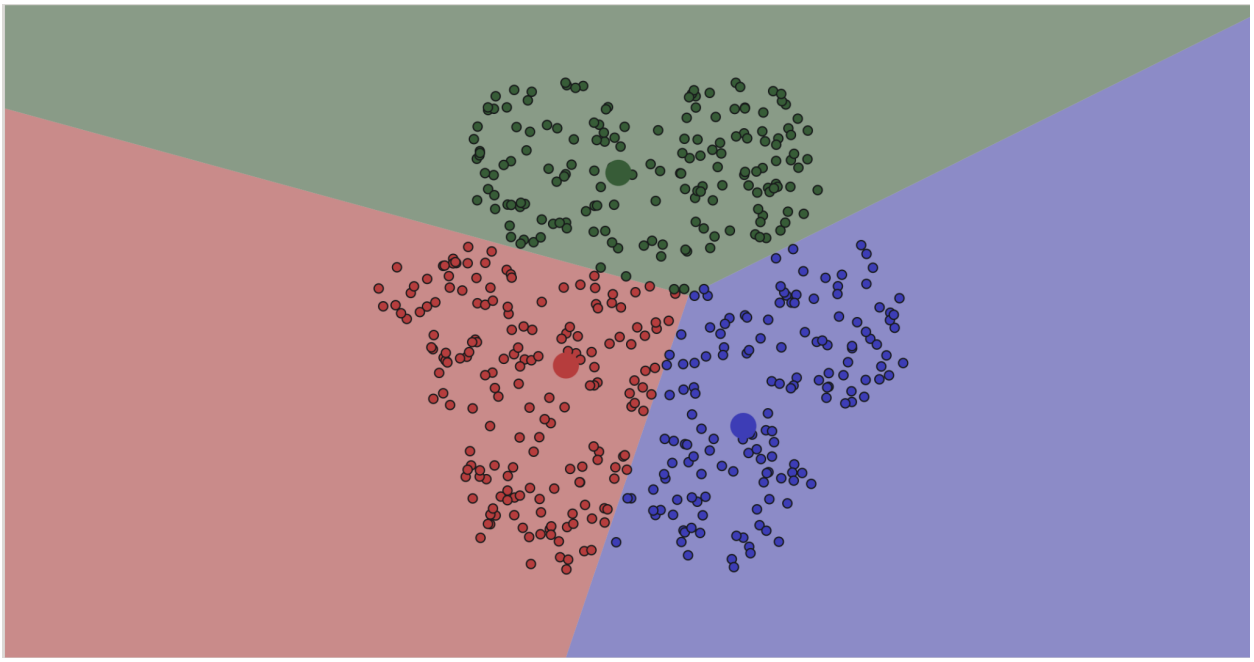


The scatter plot has now been split into three different clusters, whichever centroid is nearest to a datapoint determines what cluster that datapoint belongs to. However the algorithm is not finished, the centroids were placed randomly, so the clusters are not properly spaced. Now the algorithm will calculate the center of each cluster, and move the centroids accordingly,

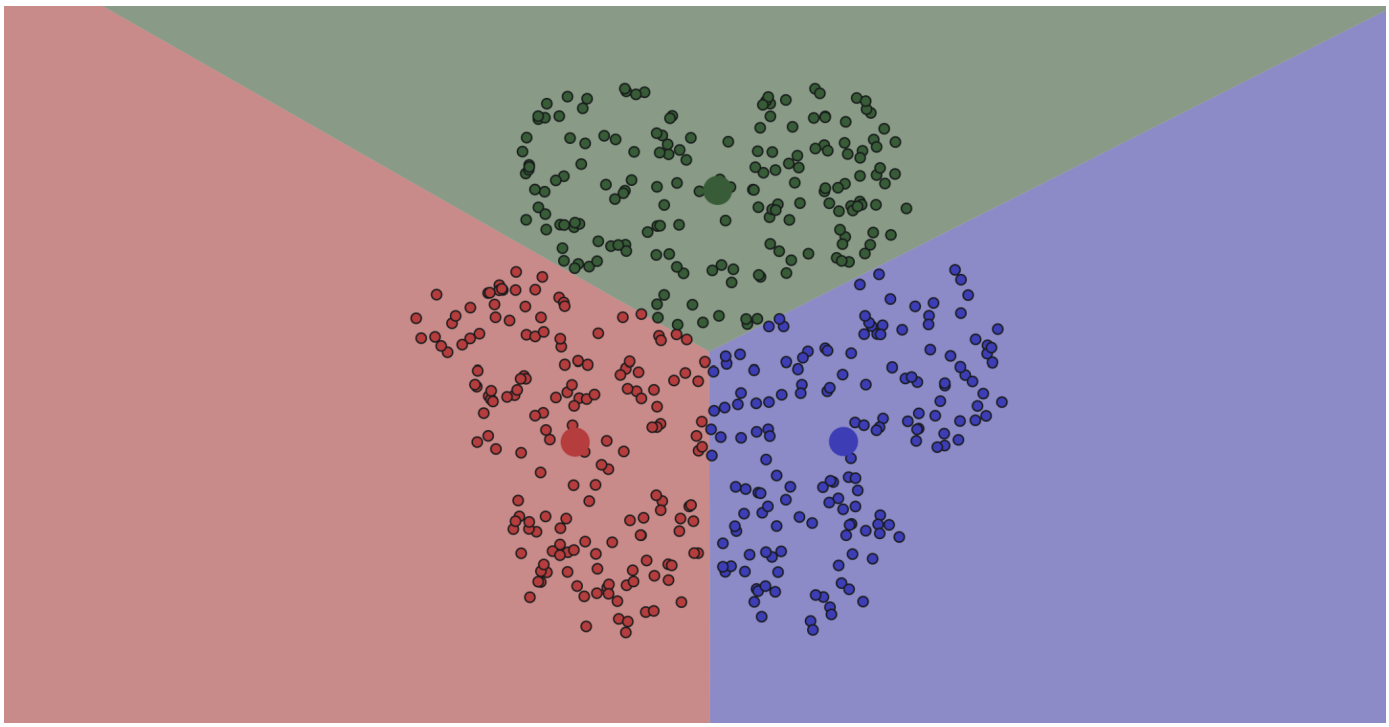


then it will remap each datapoint to a cluster depending on the distance of it to the nearest centroid.

After another iteration of moving the centroids and reassigning the data points now the clusters look like this.



Then after a few more iterations it finally stops readjusting the centroids, meaning the clusters have been properly divided, and so the final three divided clusters look like this.



And so that is a simplified illustration of what I was trying to do with all these various tracks and their varying attributes. Using KMeans I fitted the data into a model with 5 clusters. Each song

was assigned a cluster number, I added this cluster number as a column to the dataset, labeled “type”, then I saved this newly appended dataset to a new csv file named “result.csv”.

So all of this was to assign cluster numbers to each track, tracks in the same cluster are likely to be similar so if a user happens to like songs from a particular cluster, they’re likely to like other songs from that cluster. So the first thing to do is determine the user’s preferred cluster.
In my “getRecommendationsScript” the user needs to enter several songs they enjoy, and then the majority of the code is dedicated to simply finding which cluster number appears the most in their songs, then once the cluster has been determined it will give the user a sample of songs from that cluster and if all works well those songs will be categorized similarly and therefore the user is likely to enjoy the recommended songs.

Results:

I ran into some issues with the dataset when trying to input songs ID’s, perhaps it’s because the data has multiple different sources, but when I entered spotify song ID’s sometimes it wouldn’t work, perhaps if I used a different dataset I would have been fine, but in any case I simply chose 19 random songs from the dataset itself as the user inputted songs, then, having already assigned cluster numbers to the tracks in “result.csv” the program then determined the cluster number out of my 19 inputted songs, and then gave me back some songs of the same cluster, the following are my results from the terminal.

```
Favorite cluster: 3
```

The third cluster (or technically 4th since it’s 0-4) was most prominent in my grouping of songs. So now we will be getting recommendations of songs that are of cluster 3.

5 is enough songs to recommend so I only had it display me 5 suggested songs.

```
                                name       playlist  popularity  \
56           Nose On The Grindstone  Acoustic Grit        64.0
115                      Blasphemy        Rise Up        63.0
127                      Hypnotize        Rise Up        69.0
148                Hun Rider På Den         Ja dak        54.0
158  Velsignet - Har Det Godt pt. 2         Ja dak        53.0
```

Some of these songs are in the same playlist so that is a good sign for similarity.

	preview_url	speechiness	tempo	\
56	https://p.scdn.co/mp3-preview/5ee58b539a818b34...	0.0555	150.028	
115	https://p.scdn.co/mp3-preview/a0b64ecaff0fd3ce...	0.0367	154.023	
127	https://p.scdn.co/mp3-preview/134b723e412747c9...	0.0489	153.986	
148	https://p.scdn.co/mp3-preview/0754b66f191fedf6...	0.0507	145.957	
158	https://p.scdn.co/mp3-preview/e94a8769ad40d0e5...	0.1090	134.245	

The songs seem to have a very similar tempo, which makes sense as that was one of the fields for clustering. Also they have similar degrees of speechiness.

	available_markets	country	danceability
56	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.488
115	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.409
127	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.274
148	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.810
158	['AD', 'AE', 'AR', 'AT', 'AU', 'BE', 'BG', 'BH...	BE	0.622

The suggested songs all originate from the same countries, and they are all seemingly undanceable.

	valence	type
56	0.4020	3
115	0.4730	3
127	0.0397	3
148	0.8890	3
158	0.5750	3

The valence is also mostly matching among the songs, except for numbers 127, and 148.

	id	instrumentalness	key	liveness	loudness
56	4a2uqVlpRChHj32EjJLu7G	0.000000	10.0	0.1300	-11.043
115	6r7bDk4J0Okw4ezTBUQXo3	0.000000	10.0	0.3690	-4.098
127	6oO7WMjD6kEvCITLbVj0mu	0.000412	6.0	0.1150	-4.096
148	59xCJ6lThdTNnZaUVubKzB	0.000135	6.0	0.3520	-4.707
158	2xnJK0zZoSCbxIW4RYNYQv	0.000000	2.0	0.0449	-4.946

All of the songs lack instrumentalness, and interestingly also have similar levels of loudness as well as liveness.

	disc_number	duration_ms	energy
56	1.0	204000.0	0.281
115	1.0	275013.0	0.859
127	1.0	189440.0	0.920
148	1.0	151233.0	0.948
158	1.0	185504.0	0.897

The energy levels among the songs appear to be equally low.

Seeing these correlating traits among the songs is good, it tells me that the clustering has effectively categorized the songs, and while this method definitely isn't perfect or as in depth as real implemented spotify recommending algorithms or the like, I think the program effectively does what it needs to do, and that's to suggest to users songs they may like, based upon pre-existing user liked songs.