Spotipy Audio Analysis

APIs and Mean-Shift Clustering with Python by Liliko Uchida

Project Overview

Goal:

The goal of this project is to create a program which *uses APIs to find information about various music tracks and use this data to analyze the collection of songs*. I will be using APIs from last.fm/music and developer.spotify.com (the Spotify API will be refered to as “Spotipy”).

Why it matters:

This program is a very useful and interesting tool for frequent music listeners. Users can get a better insight of trends in what they listen to, find similar tracks to music they favor, and, from analytical data, discover ways in which they can vary their listening habits or choose music for different occasions.

Background information on APIs:

In its most condensed form, APIs allow us to grab data from various sites/locations on the web or Cloud and use these data in our own projects. For this project, because of my limited knowledge and practice with using APIs, this program will only function for music listened to through my personal computer. However, in the future, I hope to find a way for this program to be able to analyze any user’s music library.

Process

Finding Information

The first few days were dedicated to an intensive Google search on APIs: what exactly they are, how they work, and how to use them. This phase included a lot of patchworking from many resources (all noted in Github code).

Listening to Music

To best accomplish the randomness and variety of tracks, I sent out a request to friends to send me two or three songs they liked which were very different in genre. Using their recommendations, I was able to compile a set of 80 plus songs for a better data set and learned a lot of new great songs in the process!

Coding

The most difficult part of the process, the last large chunk of days were delegated towards actually beginning the coding process and making things work.

Design

Design choices:

The main method for this project is mean-shift clustering. In order to create a purpose for using mean-shift clustering, I arbitrarily selected the energy and valence (general positivity) levels of the tracks to plot. In selecting these two values as coordinates for each point, I hoped that the clustering method would cluster the tracks in groupings based on their overall mood. All of the data used for the mean-shift clustering algorithm were extracted from Spotify using their API.

Algorithms:

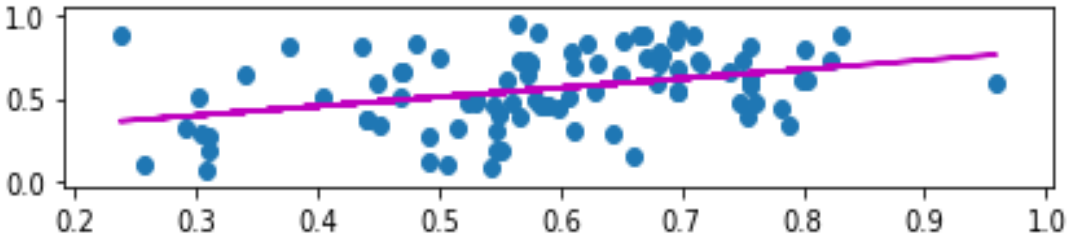
The main algorithm used in this project is mean-shift clustering.

1. initalize all data points as initial centroids
2. form bandwidths around every centroid with different radii values
3. for every centroid…
    a. calculate the weighted average of points which are within each bandwidth
    b. relocate that centroid to this weighted average
4. continue until evident clusters are formed by the centroids or other end condition is met.
5. Once evident clusters are formed by the centroids, centroids must be labelled based on which “clump” of centroids they belong to.
6. select a reference centroid.
7. if a centroid is within a threshhold distance of x, label that centroid as group 1.
8. else if a centroid is within a slightly larger threshold distance of y, label that centroid as group 2.
9. If the centroid is any further than either threshold distances, label that centroid as group 3.
10. To assign the datapoints to a group, calculate the distance between every centroid and every datapoint.
11. for every datapoint…
    a. calculate the distance between it and the centroid
    b. find the nearest centroid
    c. label the datapoint as the group to which the centroid belongs.

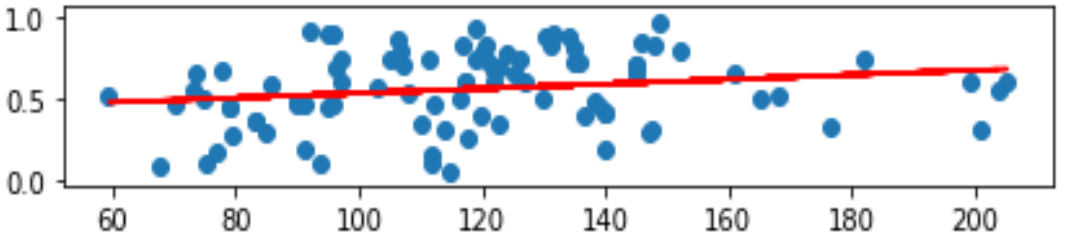
Part 1: Finding Correlation Between Audio Characteristics

To begin, I wanted to look into how different audio characteristics related to one another. Because there were so many options, I decide to focus on energy levels and graphed them against several other characteristics to see which one the energy was dependent on the most.

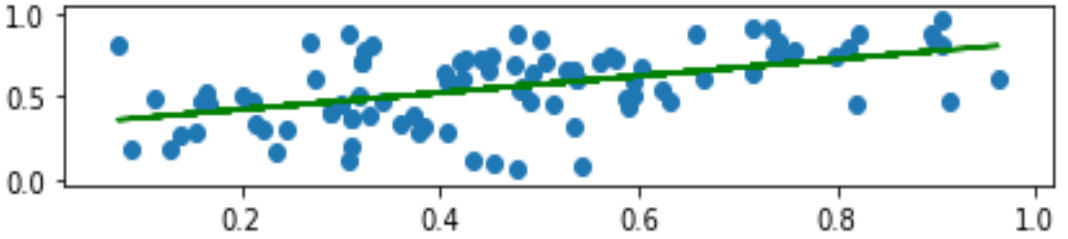
Danceability vs. Energy (slope = 0.5598054534387223)



Tempo vs. Energy (slope = 0.0013959296796774217)



Valence vs. Energy (slope = 0.5074938786919403)



Characteristic against Energy	**Slope**
Danceability	0.5598054534387223
Tempo	0.0013959296796774217
Valence	0.5074938786919403

From the slopes of the graphs, it can be concluded the dancebility of a track has the strongest correlation to the energy and tempo has the weakest correlation.

Part 2: Grouping Tracks

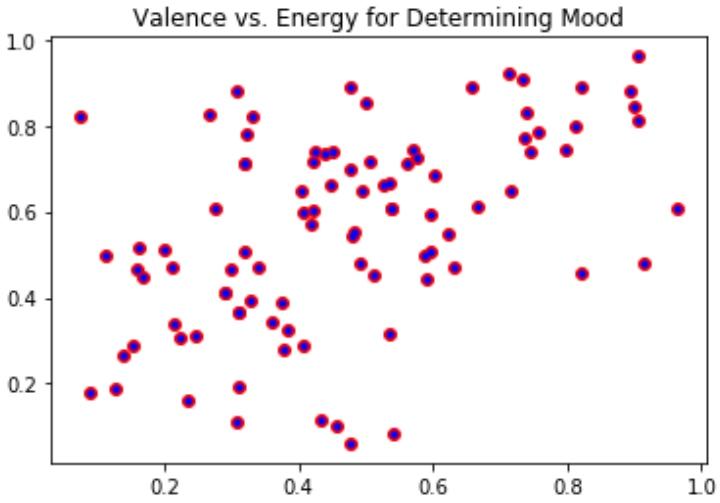


Figure 1: Demonstrates how every datapoint begins as an initial centroid.

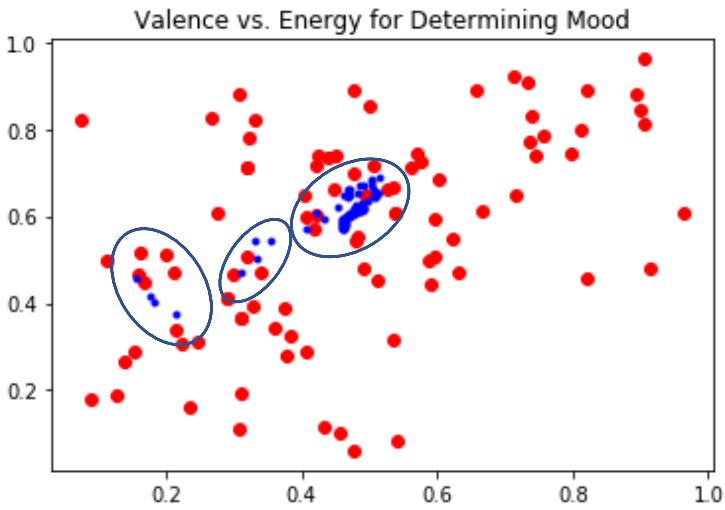
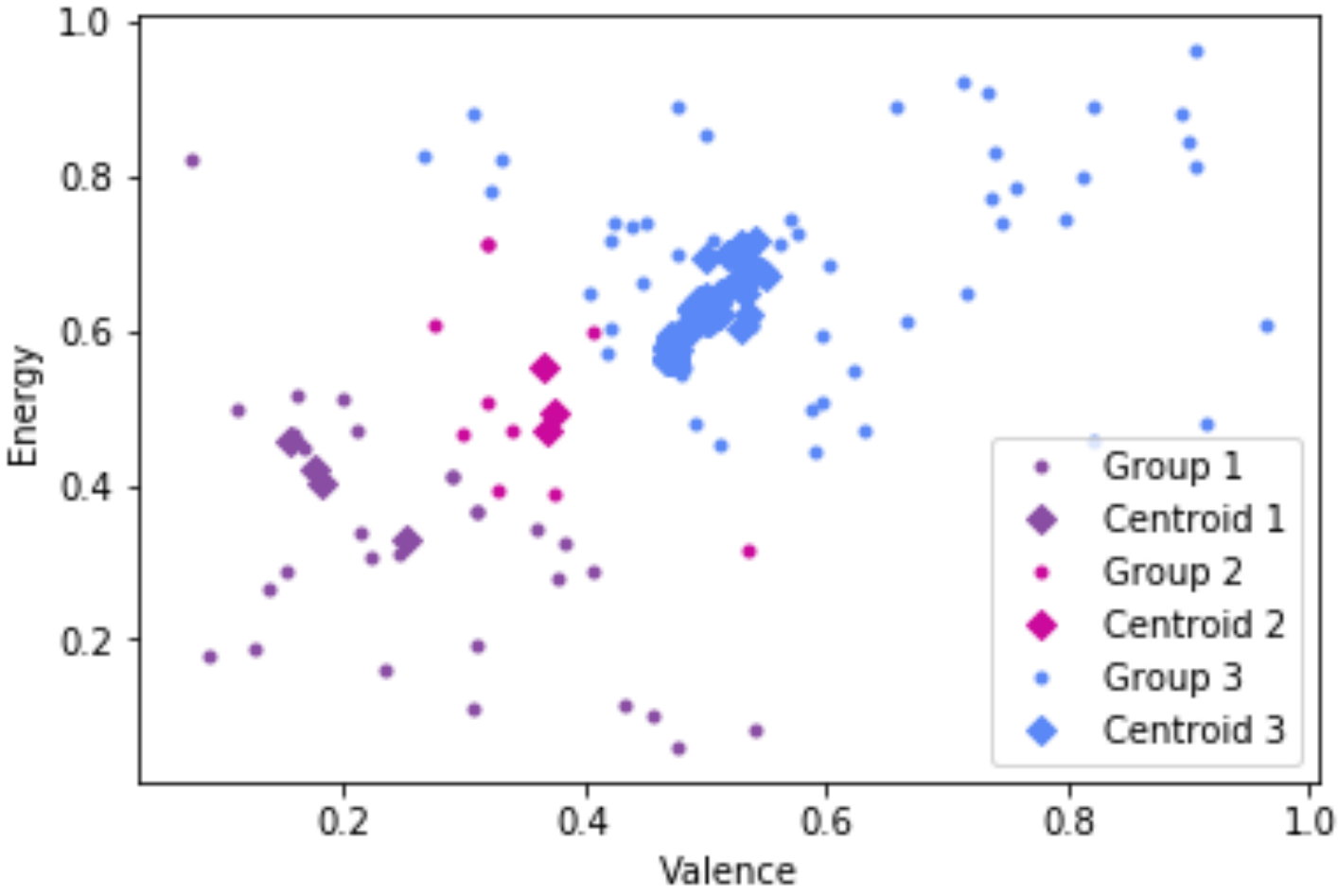


Figure 2: Shows “clumps” of centroids are formed.

Figure 3: A plot of all final centroids, clusters, and datapoints color-coded based on which cluster they belong to.



Conclusions

Mini result restated from part 1:

From the slopes of the graphs, it can be concluded the dancebility of a track has the strongest correlation to the energy and tempo has the weakest correlation.

Results of part 2:

Recalling that mean-shift clustering determines the number of clusters for you (vs. k-means in which the user had to choose a k value), so from the plot, it can b`e determined that the user's music preferences can be divided into three main categories. When running the code, an array of all track names associated with each group are also produced, allowing the user to look and see which tracks are similar to which and which are different. This can be helpful in creating playlists if similar songs are desired, if varied music is desired, etc.

What I learned:

Through this project, I learned how to build off of ideas which I am already familiar with and use this as a basis to expand and experiment with many different things. Particularly with APIs, coming into this project, the most I knew were that they existed and very vaguely what they were supposed to do. However, by taking a deeper look into APIs for this project, I gained a far better understanding of how they actually functioned as well as improved my web searching skills in desperate times of need. I also learned a new type of machine learning less supervised that k-means which was very interesting and required hard thinking to settle on an algorithm from scratch.

Future planning:

If I were to continue this project, I would like to adjust the program so that it worked with anybody's music library. I would also like to change it so that the Spotipy API could grab URIs directly from Spotify playlists instead of having to grab through last.fm which can be an inconvenience since it is through a separate website.

Acknowledgements/ Sources:

I received plenty of generous help on this project from Jenn and all of the TAs. Other resources used for the API project include:

https://spotipy.readthedocs.io/en/2.11.2/
https://developer.spotify.com/
https://geoffboeing.com/2016/05/analyzing-lastfm-history/
https://www.last.fm/api/
https://www.geeksforgeeks.org/node-url-formaturlobject-api/
All YouTube videos on LEARN SPOTIPY by Ian Annase
https://github.com/plamere/spotipy/tree/master/examples