Application of K-Means Clustering Algorithm and Principal Component Analysis for Categorizing Music by Spotify Audio Features

Joshua Truong

Abstract

The ability to categorize music by similar audio characteristics is important to users who wish to have a personalized playlist based on their moods. In this paper, we will implement k-means clustering algorithm for analyzing users spotify playlist. Combined with principal component analysis, a dimensionality reduction technique, to reduce the dimensionality of the spotify dataset. With this model, we will analyze user preferences based on the cluster analysis.

Introduction

Music is arguably one of the most diverse ranges of auditory artworks that an artist can express. As such, classifying tracks into genres should be viewed as a broad or insufficient means of classification. One of the preferable approaches towards categorizing music is by clustering tracks based on similarity of audio features. For example, based on a track frequency whether it has a high or low frequency may evoke to users as a positive or negative sensation. By clustering tracks with similar audio features, users can discover commonalities that may benefit them in regards to having an organized playlist that can fit their moods. While users can organize their own playlists it is tedious, and more often than not there's so much features in regards to a track that a playlist is not optimized properly.
The reason for this project is for me to organized the list of musics in my liked folder for both amazon music and youtube playlist. Furthermore, I always want to explore for more music. Therefore, in order to make new discoveries I will need to look at my current playlist and analyzed the clusters to determine which playlist I find most interesting.

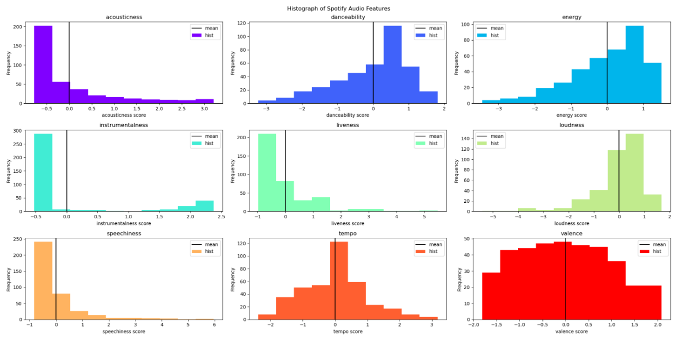
Data Acquisition

My current source of information for my dataset will consist of liked tracks from Amazon Music, Spotify, and Youtube. In order to conglomerate theses tracks from various platform I will use tunemymusic website to transfer all into Spotify. I choose Spotify as my main platform as Spotify has a developers api which allows for me to retrieve specific audio features from every music track I've ever listen to. Currently Spotify has rated each track based on these 9 audio features:

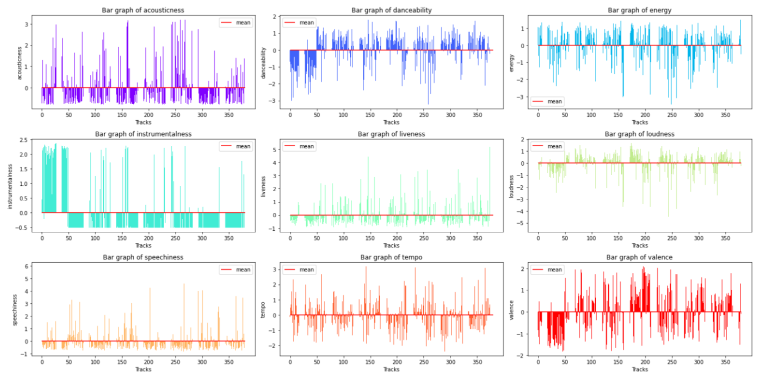
Feature	Description	Measure
Acousticness	A confidence measure for whether a track is acoustic	0.0: low confidence; 1.0: high confidence
Danceability	Is the track suitable for dancing?	0.0: least danceable; 1.0: most danceable
Energy	A perceptual measure of intensity and activity. High entropy.	0.0: least energetic; 1.0: most energetic
Instrumentalness	Predicts whether a track contains no vocals.	1.0: no vocal content; 0.5: instrumental tracks
Liveness	Detects the presence of an audience in the recording.	1.0: likely performed live
Loudness	The overall loudness of a track in decibels (dB).	Values typically range between -60 and 0 db.
Speechiness	Speechiness detects the presence of spoken words in a track.	1.0: exclusively speech-like; 0.33-0.66: made entirely of spoken words; <0.33: represent music and other non-speech-like tracks.
Tempo	The overall estimated tempo of a track in beats per minute (BPM).	
Valance	Describes the musical positiveness conveyed by a track.	1.0: Very positive; 0.0: Very negative

One thing to note when observing the features of the dataset is both loudness and tempo. Both of these features are not on a common scale with the rest which are in the scale between zero and one. This is important as we do not want the features to dominate our distance measurement. Therefore, we need to normalize our dataset using z-score which will put all features on a common scale.

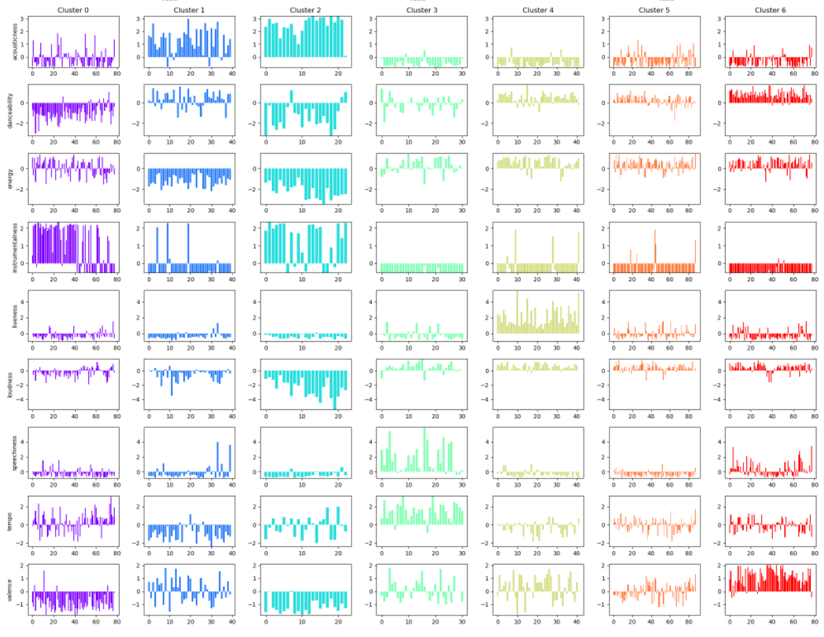
Exploratory Results



By mapping the features of the dataset onto a histogram chart, we notice certain charts are heavily skewed, like acouticness, instrumentalness, speechiness, etc. The skewness may indicate favortism among features, but could also indicate that a majority of popular tracks have features alike. For the valance graph we notice, it has a wide variance which could indicate a diverse set of tracks from sad to happy.

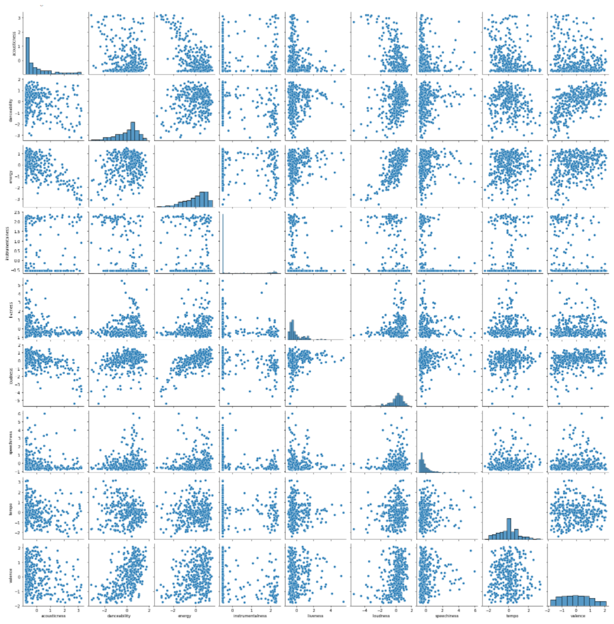


When observing the features of the dataset with a bar chart. We should take note that the goal of the k-means clustering algorithm is to cluster tracks in such a way that a the features will contain similar heights. As seen with the below clustered tracks.



Here we take notice that in every cluster of the instrumentalness feature, that most tracks have similar heights. Or that in every cluster the audio tracks have been clustered in such a way that it has mostly positive or negative values. Furthermore, the feature with the most uniform values may indicate its dominance in the cluster meaning there’s a specific feature that is associated with the majority of the track. For example, a cluster with high uniform distribution of height associated with valance, may indicate that the cluster has tracks that may give positive moods.

Out of curiosity, I thought it would be interesting to see if I could make a pair plot of each feature and determine whether a cluster exists. Sadly because the dataset contains 9 features, visualizing the clusters is very difficult as such a new approach must be made in order to reduce the dimensionality of the dataset.

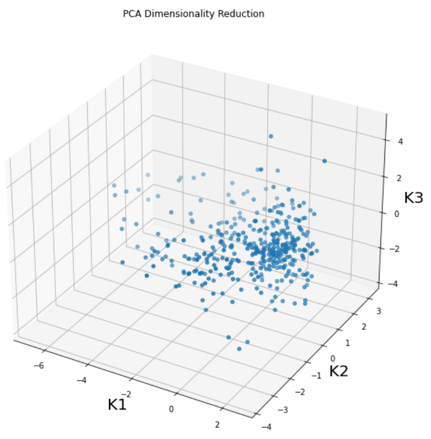


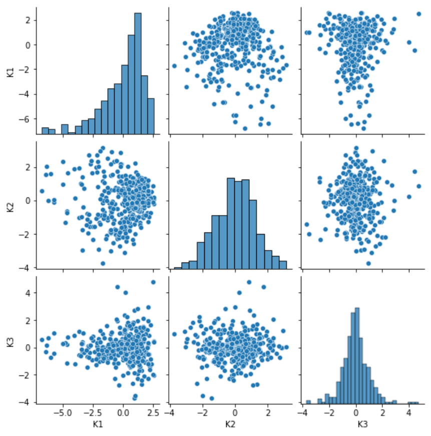
Methods

The first method applied to the dataset was a standard normalization, or the z-score. This step was necessary because the unit, or scale, of features loudness and tempo were not defined between zero and one like the rest of the features. It's important to have all features on a common scale as it will prevent these features from dominating the measurement scale.

The distance measurement used in my model is the L2-norm or euclidean distance as it's easier to scale with higher dimensional dataset. In addition, we only need to calculate the distance between data points and centroids. In order to minimize the error.

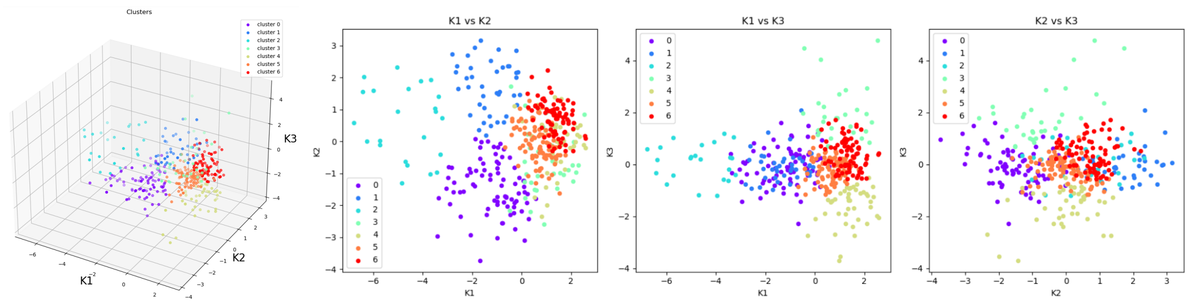
Before clustering the dataset, I also preprocessed the data by using principal component analysis (PCA) in order to reduce the dimensionality of the dataset. This process allows for two things to occur, the first is to enable better visualization of the dataset instead of imagining a higher dimension. This means that the dataset has 3 dimensions instead of the original 9.





In addition, to reduce dimensionality. PCA has also eliminated the worries of outliers. Because the PCA minimizes the L2 norm, or euclidean distance, this means that the algorithm is sensitive to outliers. Therefore, the algorithm will continue to rotate the dataset in such a way that minimizes the error. This is imperative because the k-means clustering algorithm is sensitive to outliers.

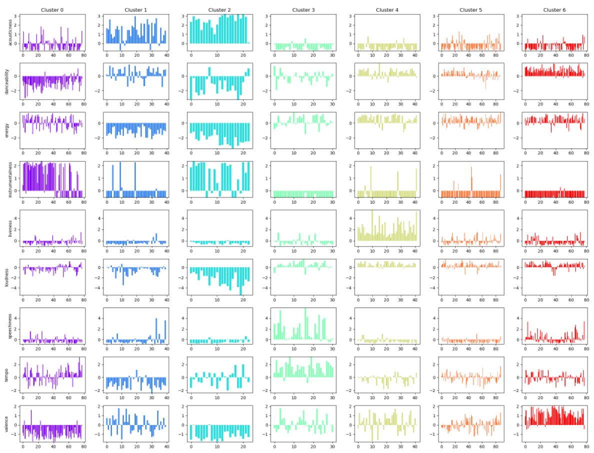
Finally, we use the k-means algorithm on our reduced dimensionality dataset.

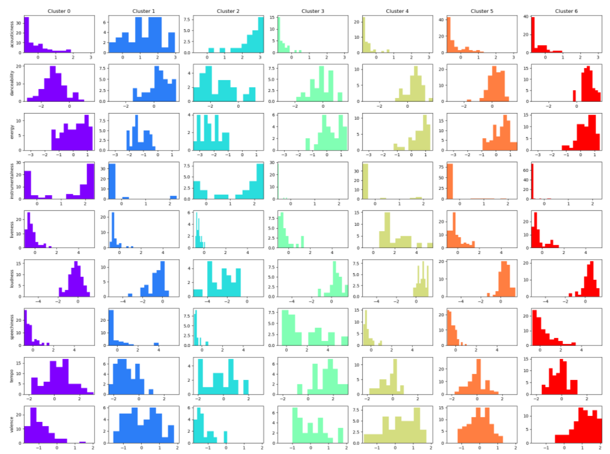


In order to find k, the number of clusters to use. I first ran from k =1 to 30. Then plotted an elbow chart in order to find the the elbow. Using a third-party library I found that k=7 seemed like the optimal number of clusters for the dataset. Since, k-means clustering algorithm does not guarantee a global minimum of mean distance error, the algorithm had to run several iterations. In addition, for each k I randomly selected a datapoint in order to avoid the same local minimum.

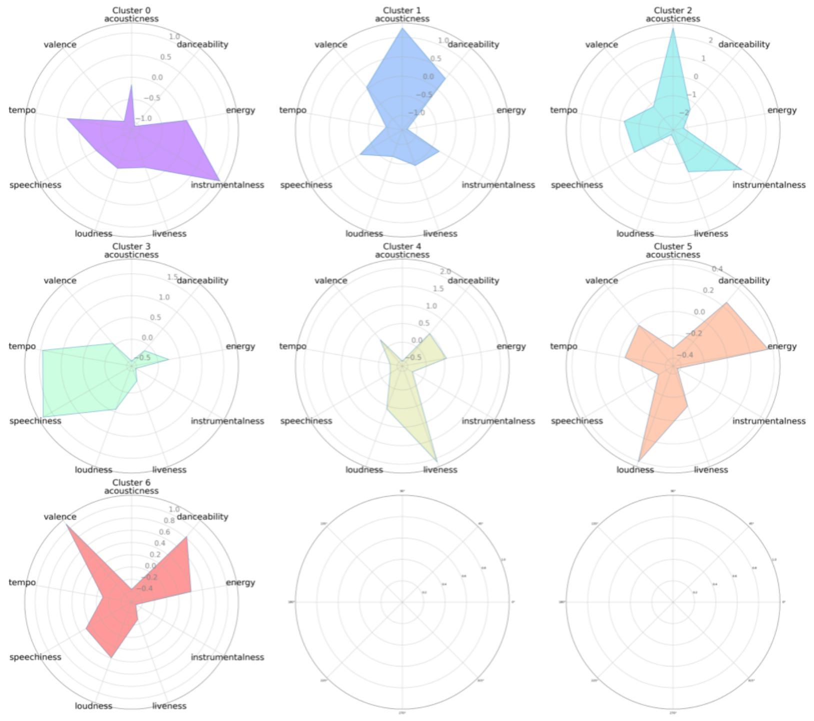
Results

After clustering our reduced dimensionality dataset, we'd like to see how each feature was clustered. Unfortunately, the principal component analysis is not great at allowing us to interpret the data. Therefore, we must take the data points in each cluster and map it back to their original values.





As a result of our k-means clustering algorithm, we see that we have indeed clustered most audio features in such a way that most will have matching heights. This is also clearly indicated with the histogram as we can see for a certain x value the frequency associated with x is very high.



In order to determine whether or not the clustering has worked. I decided to create a spider/radar chart of each cluster and the values would be the mean of each cluster audio features. The results should be that each radar chart should overlap too much with the rest and that no 2 charts should look similar to each other. For each cluster the chart displays a high value for certain audio features. This is the equivalent of a playlist with moods. As a user may want to play a playlist with high valance to feel positive or a playlist with high tempo for some upbeat music. Of course, there are more audio features associated with each cluster so a user can find a song and determine which is closer to a cluster.

Conclusion

While k-means clustering algorithm is a simple, yet effective method of clustering my spotify tracks audio features. There are still other ways to improve the model for example we can use k-median clustering in order to be absolutely sure that no outliers affect the clustering algorithm. Furthermore, if we want a more robust clustering technique perhaps implementing a gaussian mixture model. Since the GMM can do soft and hard clustering. However, with size of my current dataset using the GMM alone is futile as it is slow. Therefore, implementing PCA as a method to process the dataset before using the GMM can greatly improve the time complexity of the clustering.

While there are certainly better methods to cluster. I believe that the project has answered its initial question of whether or not we can organize our spotify playlist in such a way that can improve our mood.