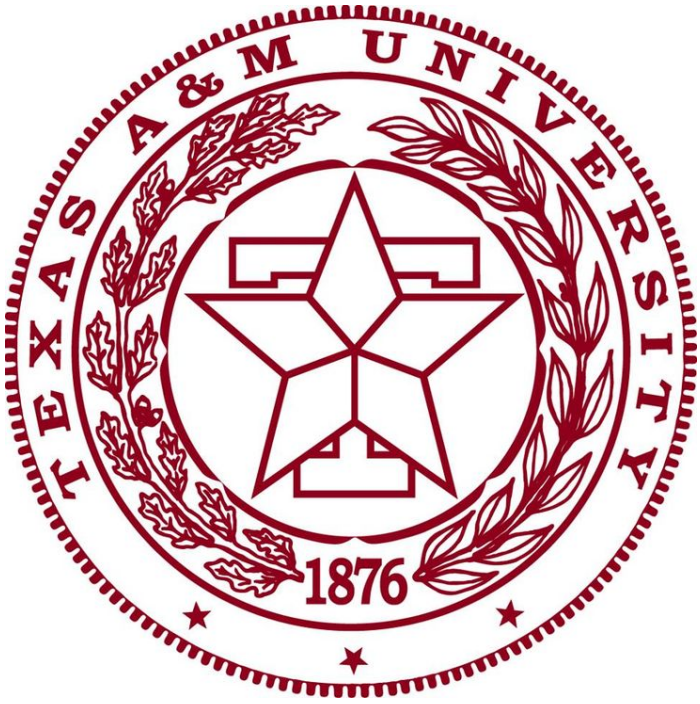
“Applying Clustering Methods to Categorize Songs Within A Playlist”

Fall 2022 MEEN 423 - Section 500



Luke Batteas

516000647

luke.batteas@tamu.edu

Executive Summary

The goal of the project is to cluster songs of a similar nature using 13 numeric features pulled from spotify API. Due to issues in precisely determining the boundaries of a genre, this problem was treated as an unsupervised learning problem. Linear and Kernel PCA were used on the features in an attempt to reduce the dimensionality or aid in the learning problem. No significant benefit was found from either Linear or Kernel PCA. A dendrogram was generated and it was found that the dataset likely has 8 eighters. Using this information, three clustering techniques were used, K-Means, Hierarchical Agglomerative Clustering, and DBSCAN. Of these techniques, no valid model was for the DBSCAN algorithm. A small portion of the data was hand labeled and used to validate the models and assign genres to the clusters. Using this, the HAC model was found to be superior to the K-Means model. A final portion of the dataset was hand labeled and used to confirm that the HAC model had identified clusters appropriately. Overall, the HAC model performed acceptably on both the validation and the testing set. However, since this problem was unsupervised, only a very small portion of the dataset was labeled. It is possible that the chosen data points to be labeled made the model appear better or worse that it actually is. Further analysis using a larger validation and training set is recommended to increase the confidence in the model.

Problem Definition

The goal of the problem is to group each of the songs in my playlist by genre. For example, all metal songs should be grouped together. Classifying music by genre is a non-trivial problem. Music may not fit cleanly into one genre, but may have two or more genres. Furthermore, the definition of a genre and where the boundaries of a genre lie are partially subjective. Finally, sub-genres within genres may be distinct from other songs still within the same primary genre. One potential method of solving this problem is to use machine learning to determine which songs should be categorized together. I created a playlist containing all of the songs I had liked from my spotify account. The playlist consists of 979 songs. For each of the songs, I utilized Spotify's developer API to download associated data with the song. Spotify has assigned 13 features to each song which is used by their algorithms to classify songs. The features, along with a brief description of each feature is shown in **Table 1**. As mentioned above, classification of music by genre is difficult and therefore the dataset is not labeled, nor easy to label. The problem type is therefore an unsupervised, clustering problem. The playlist itself is provided in the references [1].

Table 1: Dataset Features

Feature	Range	Description
Danceability	0.0-1.0	A measure of how easy it is to dance to the song
Energy	0.0-1.0	Intensity of song
Key	Positive Integer	Key of the track
Loudness	Decimal Value	Averaged loudness value in dB
Mode	0 or 1	Modality of a track (Major = 1, Minor = 0)
Speechiness	0.0-1.0	Percent of Speech in the song
Acoustiness	0.0-1.0	The probability that the track is acoustic
Instrumentalness	0.0-1.0	Percent of Non-Speech in the song
Liveness	0.0-1.0	The Probability that the song was performed live
Valence	0.0-1.0	Emotion of Song (Negative emotions are lower value)
Tempo	Positive Integer	Beats per minute
Duration	Positive Integer	Duration of Song in milliseconds
Time Signature	Positive Integer	Meter of song

Learning Strategy

Scaling

In unsupervised clustering problems, the distance metric can be affected by the scale of the data. Between the two common options for scaling, standard scaling and min-max scaling, standard scaling was selected as it allows for PCA to be performed on the data without needing to rescale the features again. Standard scaling was done prior to any of the following techniques. The raw data prior to scaling is shown as a scatter matrix in **Figure 3** in **Appendix A**.

Number of Clusters

In unsupervised learning, the number of clusters is important to know as it is required for the K-Means algorithm and provides a stopping point for the Hierarchical Agglomerative Clustering algorithm. Although DBSCAN does not require the number of clusters to function, with knowledge on how many clusters are expected, it can be used to partially validate a model. It is therefore relevant to discover how many clusters are present in the data. From my intuition on the problem, I suspected around 5 clusters. To determine how many clusters appeared to be in the data, HAC with no cutoff was used to create a dendrogram and is shown in **Figure 1**. From visual inspection of the dendrogram, the number of clusters in the dataset appears to be 8 before the distance metric grows too large between cluster merges. Although the distance to cut off the clusters is somewhat subjective, the number of clusters is not vastly different from my intuition and is therefore selected as the number of clusters in the problem.

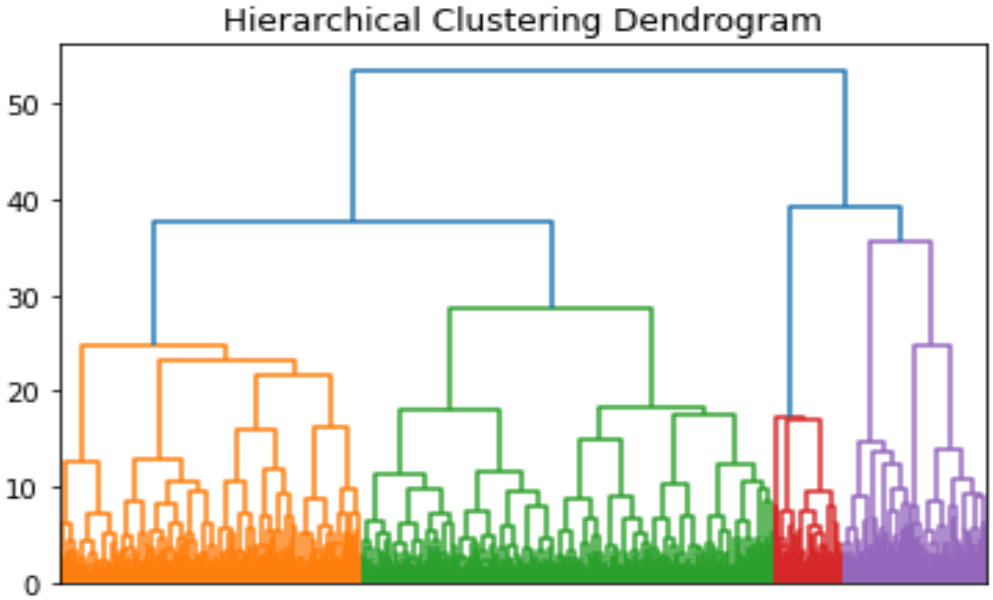


Figure #1: Dendrogram

Principal Components Analysis

Due to the high dimensionality of the data, Principal Component Analysis was attempted to reduce the complexity of the model and potentially offer better visualizations and performance. First, a linear PCA was used and a explained variance plot was generated and is shown below in **Figure 2**. As seen in the figure, 9 components are required to achieve 90% of the variance explained using a linear PCA. With 13 original features, this does not offer a significant advantage. Linear PCA was therefore not used due to only offering minimal dimensionality reduction. Kernel PCA was further used in an attempt to aid in the learning problem however no improvement was found. Therefore the 13 original features were used to train the model.

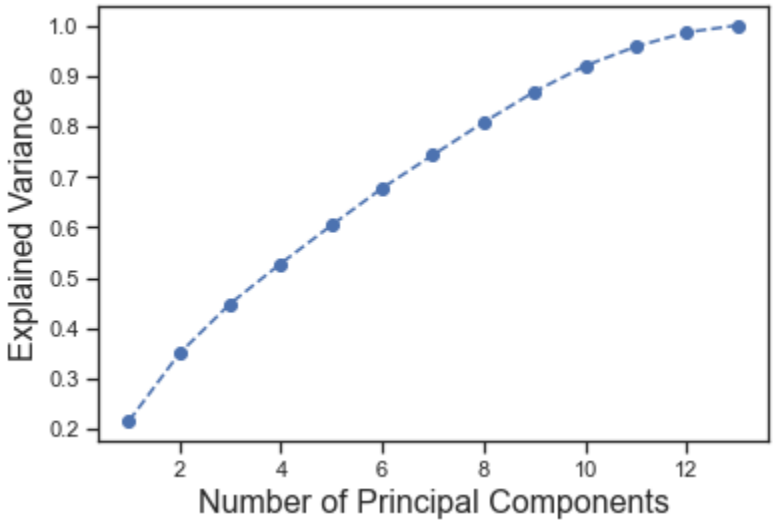


Figure #2: Explained Variance Plot for Linear PCA

Clustering Methods

Three different clustering techniques, K-Means, Hierarchical Agglomerative Clustering, and DBSCAN, were used to solve the unsupervised learning problem. For K-Means and HAC, the only hyperparameters set was the number of clusters which was selected as indicated from the dendrogram. For HAC, ward linkage was used based on the documentation provided from scikit-learn as it provides the most regular sized clusters.The DBSCAN technique has two hyperparameters that need to be tuned, the number of points to become a core point, and the radius of each point. Since the number of clusters was used to partially validate the model, the hyperparameters were tuned to achieve the correct number of clusters. If there were too many clusters either the number of points to become a cluster should be increased or the radius should be decreased. On the other hand, if there were too few clusters, the direction of change of the hyperparameters was reserved. Finally, the number of outliers determined was monitored. Some

outliers are expected but this number should be very small. A large percentage (>5%) of outliers suggests a poor fit of the data.

Model Validation and Testing

Once the model was generated, the model needed to be validated. However in unsupervised learning, the data lacks labels that could be used to measure the error. While it may be possible to use an external source to label each song with a genre, this is non-trivial. As discussed in the problem definition, there are issues where a song may not fit cleanly into one genre but may be a mixture of two or more. Furthermore, the definition for each genre may vary from person to person. Instead I will implement a simpler approach that selects two songs that I am personally confident are the same genre. This method will also be used to assign the genre title to each cluster using the first song. I will then assign a score to the model based on how many of the pairs are correctly clustered together. The paired songs and their associated genres are shown below in **Table 2**. Next, once the best model is selected, a similar test will be performed to ensure that the model is a good fit. The songs for the testing set are shown in **Table 3** and the songs should be within the genre they have assigned. This approach is taken because it is similar to holdout validation used in supervised learning and only requires labeling a few of the data points. Furthermore, my insight on the problem will be used to validate the model. From familiarity with the songs present in the playlist, I am sure that metal would be the largest genre, followed by either alternative or otacore. The smallest genres would be instrumental, fantasy, or vocals.

Table #2: Validation Table

Song 1	Song 2	Genre
Mortal Reminder	Sanctified with Dynamite	Metal
Leaving Caladan	Regicide	Instrumentals
Godzilla	My Darling	Rap
Animals	Chasing The Sun	Pop
Voracity	Kick Back (From "Chainsaw Man") - English Version	Otacore
Lullaby of Woe	Within	Fantasy
Misty Mountains	Goodbye	Vocals
Broken	Pop Cult Crucifixion	Alternative

Table #3: Testing Table

Song	Genre
Resist and Bite	Metal
Dirge From Dark Side	Instrumentals
STORY	Rap
Bad Romance	Pop
Deal With The Devil	Otacore
We All Lift Together	Fantasy
The Engineers Drinking Song	Vocals
Roentgen	Alternative

Model Training & Results

DBSCAN

No valid model was found for DBSCAN. Most of the fits had a very high outlier percentage, greater than 20%. This is far beyond the number of outliers expected. Furthermore, the remainder of the points were contained in two clusters with remainder of the data. Despite tuning the hyperparameters to obtain an appropriate number of clusters, this was not achieved. Tuning the hyper parameters did not show any significant improvement but instead created many very small clusters. This overall suggests the DBSCAN technique is a poor choice for the dataset. This can occur if the clusters present within the data have different variances.

K-Means

The K-Means model was generated with 8 clusters as discussed above. The model was then validated using the strategy above. The K-Means model passed four of the eight checks. This suggests a poor model. The model failed on the Metal, Rap, Fantasy, and alternative genres. The model cluster statistics are shown in **Table 4** in **Appendix A**. These give further insight into how well the model clustered the songs. The largest genres were otacore and alternative, followed by metal. The smallest genre was instrumental and fantasy. This somewhat aligns with the predictions made earlier with the main difference being the metal genre is too small.

Hierarchical Agglomerative Clustering

The HAC model was generated with 8 clusters as discussed above. The model was then validated using the strategy above. The HAC model passed seven of the eight checks. This suggests a good fit. The model failed the metal genre. The model cluster statistics are shown in **Table 5** in **Appendix A**. These results provide some insight into how well the clustering was done. As predicted, metal was the largest category followed by otacore and then alternative. The smallest cluster was fantasy followed by instrumental. As this aligns with the predictions made earlier, this gives credence to the model.

Model Selection

The HAC model is selected as the best model through the validation strategy proposed using the validation songs and comparing the clustering statistics to the expected values. Next, to confirm a good fit, the testing songs were used. The HAC model succeeded in six of the eight training tests, failing the alternative and the fantasy checks. While lower than the validation check, this is well within random chance given such a small sample size. The HAC model is therefore selected as an appropriate model. The clustered scatter matrix is shown in **Figure 4** in **Appendix A**.

Summary

Due to lack of labels and the difficulty in acquiring labels, the problem of classifying songs into genres is relegated to that of an unsupervised learning problem. Using three different clustering methods, K-Means, HAC, and DBSCAN. Through a validation and testing procedure, the HAC model was selected as the best model going forward. However, due to the unsupervised data, validation and testing is difficult. 24 Songs were hand labeled and used to perform validation and testing. This only uses about 2% of the dataset for testing or validation however. In practice there could be significant errors within the model still present that were not detected. One possibility is to increase the number of songs used in the validation and testing sets. This however, would require labeling more of the data. One note of concern is that all models classified the metal section incorrectly during validation despite my full confidence that all of the songs are firmly in the metal genre. This indicates that there may be one or more key features missing from the dataset. One obvious example would be data about the lyrics. The sole relevant feature is the amount of speech in the song. That alone is likely insufficient and I believe the lyrics of a song may contain significant important information that may aid in clustering the songs. This poses a significant challenge however as lyrics would be a non-numeric data of varying amounts per song. My initial recommendation would be to use a natural language processing model prior to extract information from the lyrics and use this information as additional features. Overall, the HAC model performed acceptably well on the validation and testing sets but further testing would increase confidence in the model.

References

[1] "Meen 423 playlist," Spotify. [Online]. Available: https://open.spotify.com/playlist/6qeSBRzXBFqyGnOreDveCg?si=9dc732659da84b3e.

[2] "A machine learning deep dive into my Spotify data.," Open Data Science - Your News Source for AI, Machine Learning & more, 05-Apr-2018. [Online]. Available: https://opendatascience.com/a-machine-learning-deep-dive-into-my-spotify-data/.

[3] sfnxboy, RiveN. 2021. "Spotipy: How to read more than 100 tracks from a playlist" Available: https://stackoverflow.com/questions/39086287/spotipy-how-to-read-more-than-100-tracks-from-a-playlist

Appendix

Appendix A: Supplementary Material

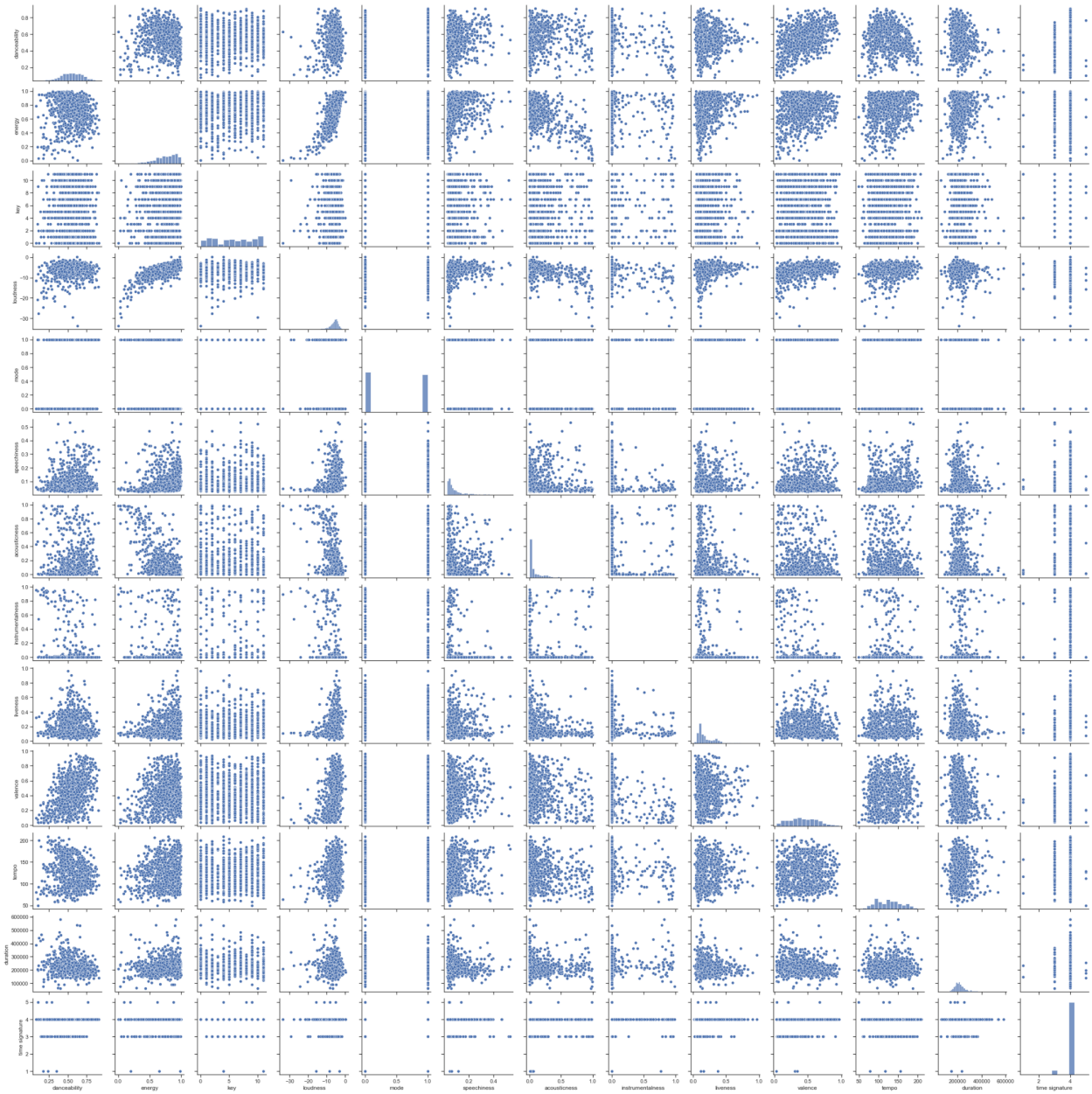


Figure #3: Raw Feature Scatter Matrix

Table #4 K-Means Model Clustering Statistics

Cluster	Genre	Count	Percentage
0	Alternative	202	21%
1	Metal	154	16%
2	Instrumental	42	4%
3	Rap	136	14%
4	Pop	124	13%
5	Vocals	69	7%
6	Otacore	208	21%
7	Fantasy	44	4%
	Total	979	100%

Table #5: HAC Model Clustering Statistics

Cluster	Genre	Count	Percentage
0	Instrumental	62	6%
1	Fantasy	47	5%
2	Metal	229	23%
3	Vocals	72	7%
4	Rap	93	9%
5	Alternative	182	19%
6	Otacore	184	19%
7	Pop	110	11%
	Total	979	100%

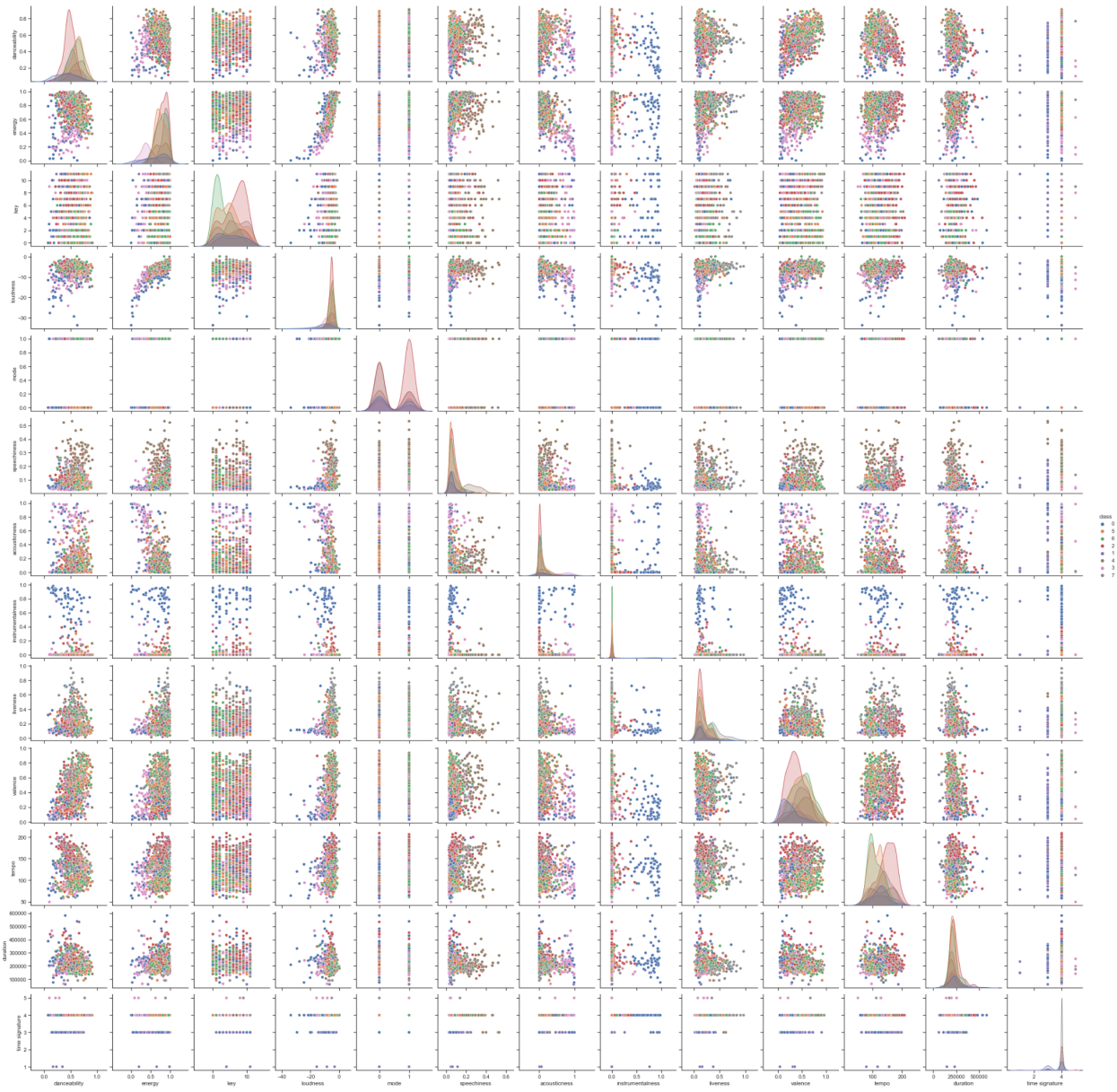


Figure #4: Classified Scatter Matrix for HAC Model

Appendix B: Files

getTracks.py

This file pings the spotify API using a developer account to collect the feature. In the interest of security, I have removed my client_ID and client_secret prior to submission. This code was taken from [3] to allow for collecting more than 100 songs at a time.

import_data.py

This file calls the getTracks file to collect the data and save it to a numpy file.

PCA.py

> This file runs Linear and Kernel PCA on a dataset, allows for generation of a dendrogram, a scatter matrix and can save results to a csv.

FitDBSCAN.py

> This file fits a DBSCAN algorithm to a dataset, allows for generation of a dendrogram, a scatter matrix and can save results to a csv.

FitHAC.py

> This file fits a HAC algorithm to a dataset, allows for generation of a dendrogram, a scatter matrix and can save results to a csv.

FitKMEANS.py

> This file fits a K-Means algorithm to a dataset

playlist_Data.npy

> This is a numpy storage file, similar to pickle, for the song title and 13 features for each song. This was used so I would not have to download a playlist each time I wanted to run the code.

plot_dendrogram.py

This file places all the code required to plot a dendrogram into one file that can be called into other files

K_Means_Model_Results.csv

A csv file with the song titles and class along with the table shown in **Table 4**. Generated from the K-Means model in the report.

HAC_Model_Result.csv

A csv file with the song titles and class along with the table shown in **Table 5**. Generated from the HAC model in the report.

ScatterPlot.png

Figure 3, if viewing is difficult on the pdf.

ClassifiedScatterPlotHAC.png

Figure 4, if viewing is difficult on the pdf.