music_classification_unsupervised

September 25, 2022

1 Introduction

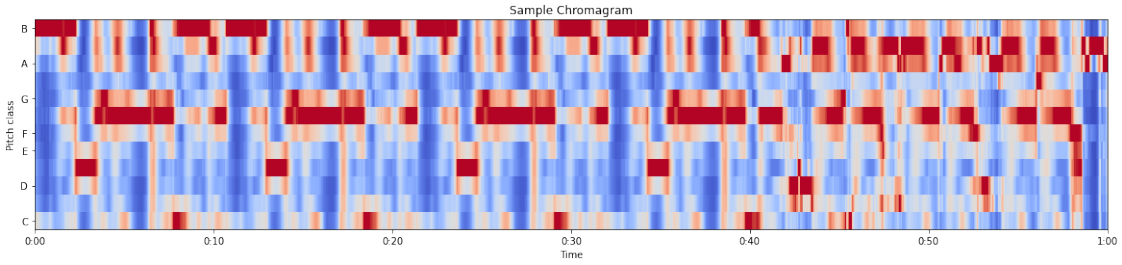
In this project I am interested in classifying different music genres based on a sample of 90 audio clips. The main goals includes extracting meaningful musical features, reducing dimensionality of the feature matrix for better visualization, perform classification via K-means and at last, evaluating the classification result based on Silhouette score. The audio data is collected from randomly selected songs in iTunes. The number of genres was pre-specified by the course instructor to ensure a meaningful result can be derived.

2 Feature Engineering

2.0.1 Frequency Domain Feature: Constant-Q Transform and Chroma

The main idea is to extract features related to 12 chroma masters. By applying constant-Q transformation, one is able to identify the sound frequency (chroma) out of 12 classes at each time point. Technically, the sound frequency defined in this manner ranges from 0 to 1. Below is a sample choromagram. In order to generate a useable feature vector for dissimilarity calculation, I use the following method: a sound signal is counted if its frequency is greater than 0.5, and ignored otherwise. As a result, we should get a 12-dimensional vector for each audio, where the rows give the number of strong signals corresponding to some pitch class.

Code: Take audio files as input, use "chroma_cqt" function from librosa to exract chromagram, and count the number of strong signals (>0.5) for each pitch class.



2.0.2 Time Domain Feature: Time Between Beats

To understand the general rhythm of a song, one may consider tempo feature like "number of beats per minute". Alternatively, "average time between beats" is used in our case, which measures

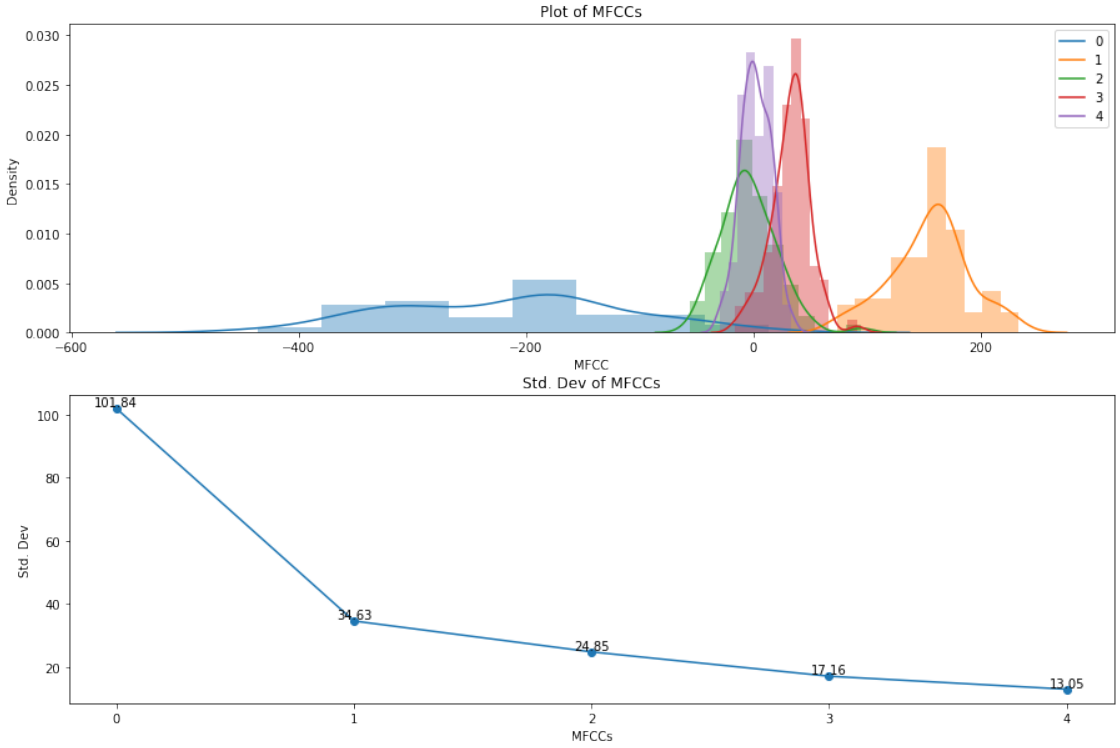
the same information as beats per minute. Calculation of this temp feature relies on the function provided by librosa. The resulting vector will be 1-dimensional.

2.0.3 Spectrum-based Feature: MFCCs

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients are coefficients that collectively make up an MFC.

- **Determining how many MFCCs to use**

All MFCCs for each piece of music sample are calculated, which yield 5 sample density distributions. One can notice that the coefficients tagged "0" and "1" in the plot exhibits higher volatility than the others. A screeplot of their standard deviations gives a numerical measure of such difference. In other words, coefficient "0" and "1" may be sufficient already in distinguishing different music genres. Therefore, only the first two MFCCs are adopted as features. The result is a 2-dimensional vector.



2.0.4 Spectrum-based Feature: Spectral Flatness

Spectral flatness (or tonality coefficient) is a measure to quantify how much noise-like a sound is, as opposed to being tone-like. A high spectral flatness (closer to 1.0) indicates the spectrum is similar to white noise. Each audio file will produce a time series of spectral flatness. For our purpose, To reduce dimensionality, the median spectral flatness is used instead. The output is a 1-dimensional

feature vector.

2.0.5 Time Domain Feature: Zero-Crossing Rate (ZCR)

The zero-crossing rate (ZCR) is the rate at which the sound signal changes from positive to zero to negative or from negative to zero to positive. It is helpful in detecting the general pattern of the music.

2.1 Features Summary

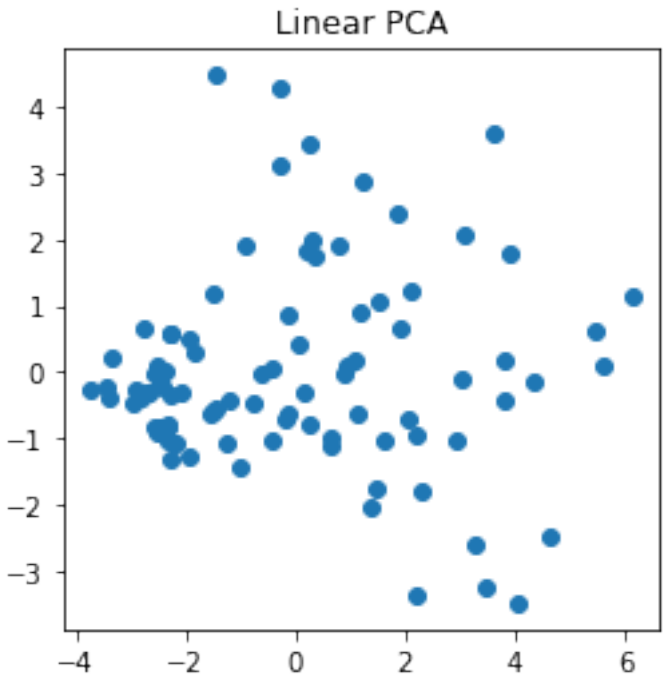
We build our feature matrix based on three classes of musical features, which are: time domain features, frequency domain features and spectrum shape based features. A brief summary below:

- `beat_time`: The average time difference between two concecutive beats.
- `Chroma_i_master`: The number of strong beats in the i-th row (representing one of the 12 pitches [C, C#, D, D# ,E, E#, F, F#, G, G#, A, A#, B]) of the chromagram matrix.
- `Median_Flatness`: The median of spectral flatness of an audio clip. (Mdian instead of mean is used to minimize the impact of outliers)
- `Median Zero-Crossing Rate`: The median of Zero-Crossing rate of an audio clip.
- `MFCCs (1,2)`: The first two Mel-frequency cepstral coefficients, which are also the most influencial among all.

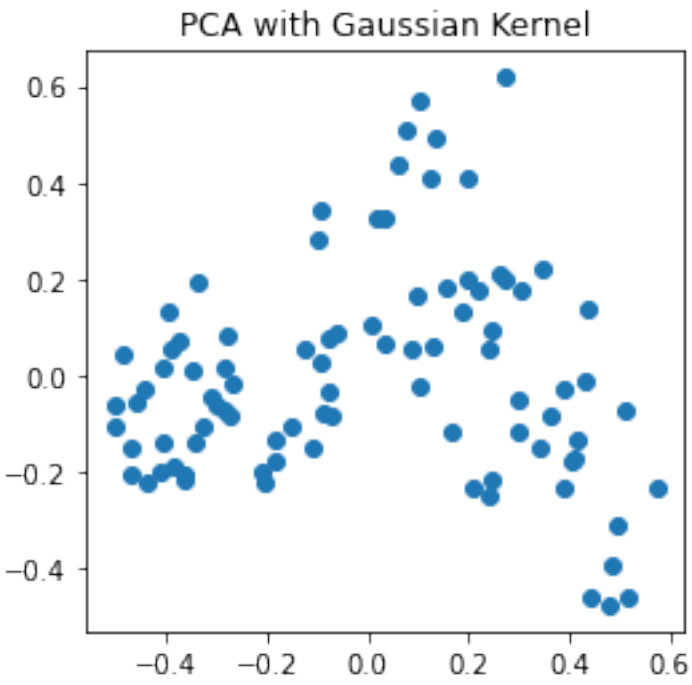
The resulting column space of the feature matrix is thus 17-dimensional

3 Dimension Reduction

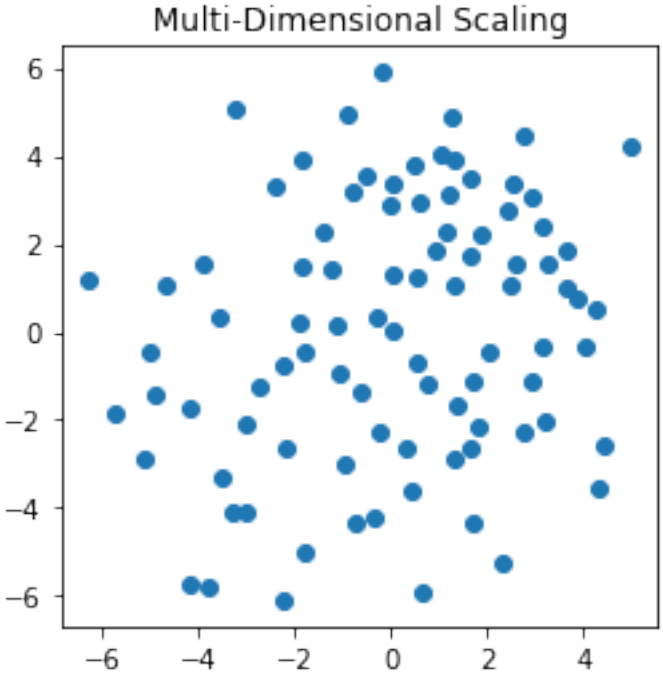
Linear PCA The principal component analysis gives a linear projection of our data onto orthogonal axis to possibly maximize the overall variability in a low dimensional space. From the plot below, which visualizes the projection of our data onto the first two eigenvectors of its covariance matrix, we can hardly find any meaningful cluster structures. It indicates that our data **may not be linearly seperable.**



Kernel PCA The kernel PCA may work better with non-linear structures by introducing a kernel function, which is to be applied to the standard-scaled dataset before projecting it onto a lower dimension space. In our case, the Gaussian kernel is used. From the result below, one can observe a **rough triangular pattern**, which **confirms the non-liearnity** of the input dataset. Though, it **still does not seperate our data very well.**

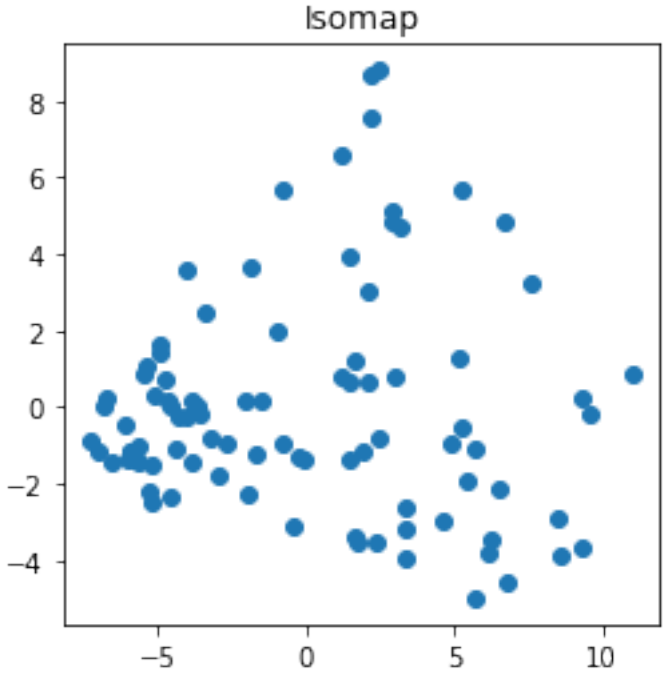


Multi-Dimensional Scaling The MDS is another non-linear method that attempts to reduce dimensionality such that the resulting distances well respect the pairwise distances of the data in its original higher-dimensional space. In our case, MDS **performs poorly** as its lower-dimensional projection looks very **close to white noise.**



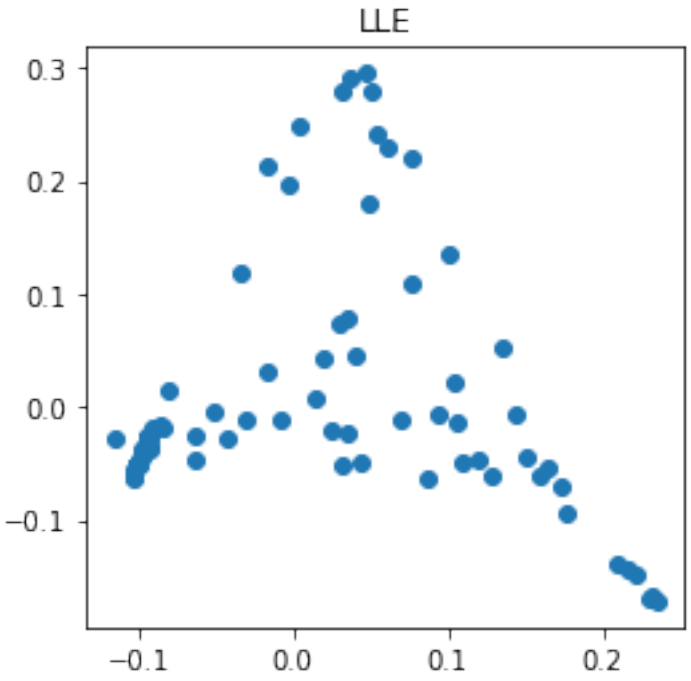
3.0.1 Isomap

The Isomap algorithm uses similar dimension reduction method to MDS. However, it aims to preserve geodesic distance on a manifold instead of euclidean distance that is used in MDS. Its advantage is that it takes into account the "general shape" of input dataset and finds pairwise distances in a manifold-based graph instead of on a hyperplane. From the plot below, one can see that, similar to kernel PCA, **a rough triangular pattern** is revealed. However, the result is **still not quite linearly seperable.**



3.0.2 Locally Linear Embedding

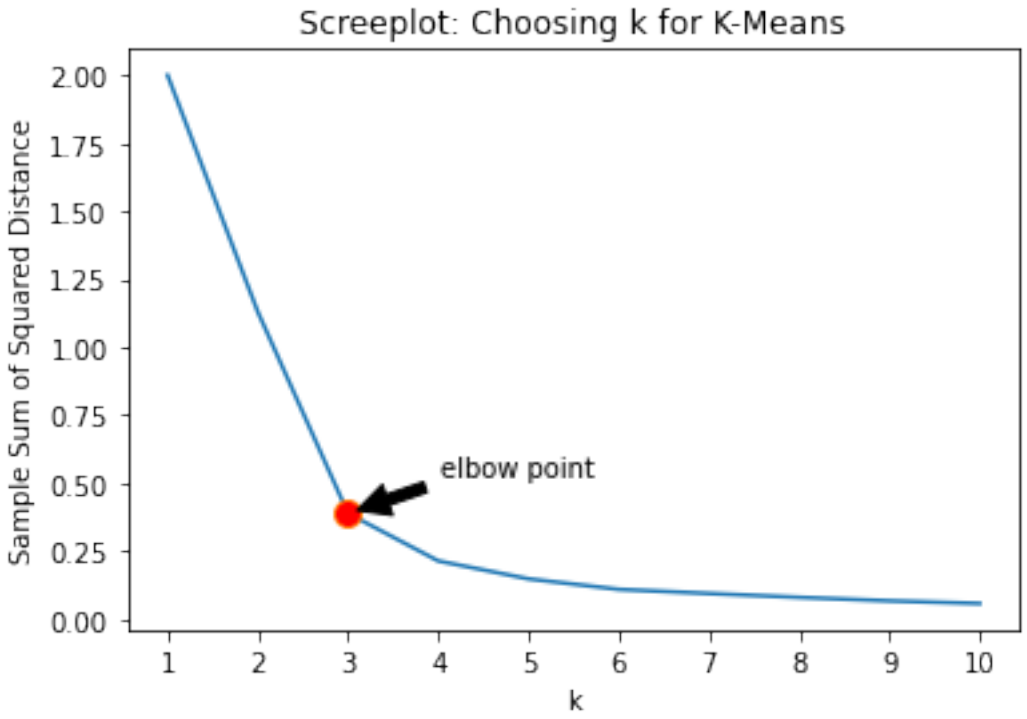
Locally linear embedding tries to reduce dimensionality while also preserving the local dissimilarity structure of each k-NN neighborhood corresponding to each point. By visually comparing LLE to previous methods, the low-dimensional representation it creates **gives the best result** as one can see a very clear triangular shape. Points are **very well seperated into three clusters**. Such visualization also gives us useful insight in choosing the number of clusters.

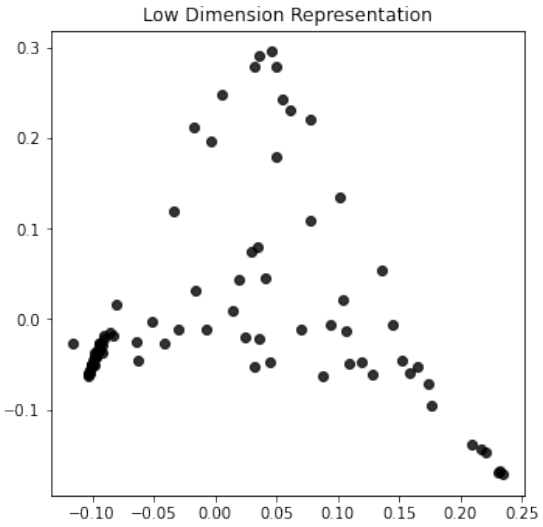


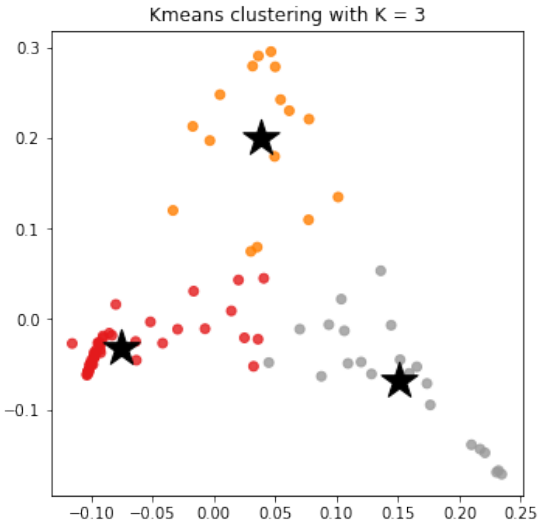
Conclusion: In total, the performance of five dimension reduction techniques are explored based on the 90 music samples. It seems that **Locally Linear Embedding gives the best result** in terms of exhibiting the 17-dimensional feature matrix in a 2-dimensional space and in a linearly seperable manner. Based on the visualization, we further hypothesize that the true number of clusters is equal to 3, or that the 90 music samples comes from **3 different music genres.**

4 Classification via K-Means

After applying Locally Linear Embedding, the low-dimensional representation of the sound features looks quite linearly seperable. We can thus apply K-Means to classify the audio data. We hypothesize that the 90 music sample consists of 3 different music genres as points in the lower-dimensional representation exhibits a clear triangular pattern. Also, further increasing the number of clusters from 3 to 4 does not reduce sample sum of squared distance significantly based on the screeplot. Therefore, k is set to be 3 in our case. A visualization of the initial clustering result is given below.





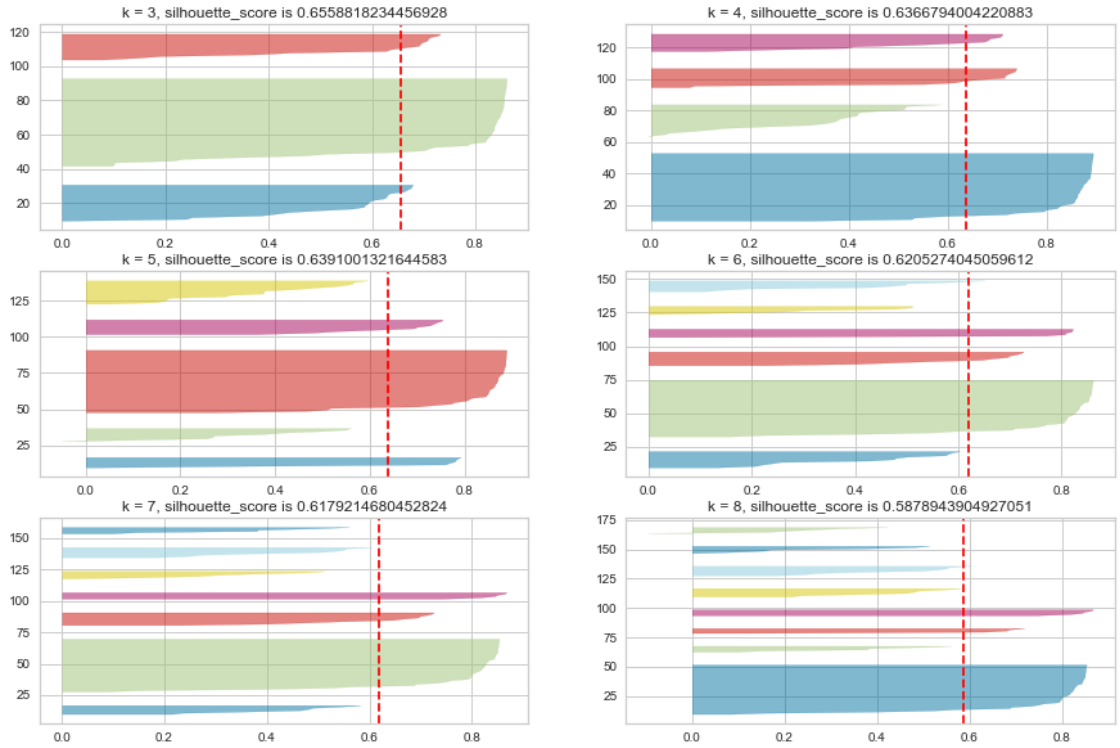


4.0.1 Evaluating Clustering results

The **average silhouette score** is used as a metric to evaluate our clustering result. A silhouette score is calculated using the formula (a-b)/max(a,b), where "a" is the mean intra-cluster distance and "b" is the mean inter-cluster distance from the nearest cluster. A silhouette score can range from 1 to -1. More specifically, a silhouette score of 1 indicates perfect seperation of a cluster to

its nearest neighbor; a silhouette score of 0 indicates significant overlapping and thus ineffective decision boundary; a silhouette score of -1 indicates incorrect cluster assignment. In short, **the closer the score is to 1, the better the quality of clustering**.

The silhouette scores for all clusters using different values of k have been calculated and shown below. We can thus compare several possible choices of "k" and also see the quality of clustering for each specific cluster. We say that a good clustering result should achieve: + Higher silhouette score on average + Minimal presence of clusters with below average silhouette scores + Minimal presence of negative silhouette scores + Small fluctuation in size and quality of clustering



Based on the above plots, it seems that setting k=3 is the optimal choice, which produces the highest average silhouette score of 0.656. The silhouette score for all 3 clusters are close to the average; no cluster has significantly lower clustering quality than the others; no unreasonable cluster assignment is found; the size of each cluster does not look unbalanced either.

5 Conclusion

To get a good feature matrix, I adopt 3 classes of musical features (time domain, frequency domain and spectrum-based features) in the hope to comprehensively grasp some structral differences across different music genres. After experimenting with several of the most commonly used dimension reduction techniques, I achieve the best result using LLE (Locally Linear Embedding), which gives a linearly seperable low-rank representation of the original high-dimensional data. With k=3, I am able to get a satisfactory classification result with an average silhouette score of 0.66.

```
[NbConvertApp] Converting notebook music_classification_unsupervised.ipynb to
pdf
[NbConvertApp] Support files will be in music_classification_unsupervised_files/
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Making directory ./music_classification_unsupervised_files
[NbConvertApp] Writing 32073 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 242983 bytes to music_classification_unsupervised.pdf
```