Final Report: Music Genre Clustering

CPS 803

David Nunez
500832868
djnunez@ryerson.ca

Paul Guevarra
500833529
paul.guevarra@ryerson.ca

Darryn Roopnarine
500876974
darryn.roopnarine@ryerson.ca

I. Introduction and Problem

In many applications such as Spotify, songs are grouped together into playlists based on their similarities to one another. The problem being tackled is to cluster a set of songs into "genres" based on their audio features (e.g. tempo, loudness, danceability).

Clustering/Classification of songs is a first step in the larger overall problem of recommending new songs to users based on their listening habits. If a listener decides that they like a song and indicates as such, they can be recommended other songs with similar audio features to said song.

The biggest challenge is the sheer size of the dataset, and the variance between songs. Having a large dataset can be limiting, as some algorithms cannot perform effectively. There may not be enough patterns for the model to pick up on to successfully perform clustering on them.

The following sections will detail the dataset used, the methods and models employed to solve the problem and the results of the model.

II. The Dataset

The dataset, obtained from Kaggle, consists of over 160,000 songs from Spotify.[1] There are numerous features including year released, artist, key, whether it begins on a major key, etc.

For the purposes of this model, we chose the floating point audio features that Spotify assigns each song automatically, which include: acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo and valence. Having numerous features would allow more opportunities to evaluate each song.

These features were chosen as they seem the best candidate for assessing the similarities between songs. All of the features chosen were in a numeric value. Moreover, these are the features that may easily be scaled during preprocessing.

The dataset does not include ground-truth labels. Spotify only assigns genres to artists, and not songs. Furthermore, artists are allowed to include more than one genre in their bio, making it impossible to create our own list of class labels. As such, we will only be employing unsupervised learning techniques (clustering instead of classification), and only using internal metrics to evaluate the model.

Because there are no class labels, it is impossible to assess the balance of the data.

III. Methods and Models

A. *Pre-processing*

All features were standardized on a scale between 0 and 1. This will ensure that all parameters are weighted equally. If this step was not performed, the model would put excess weight on the loudness and tempo parameters, as they are on average much larger than the other parameters.[2]

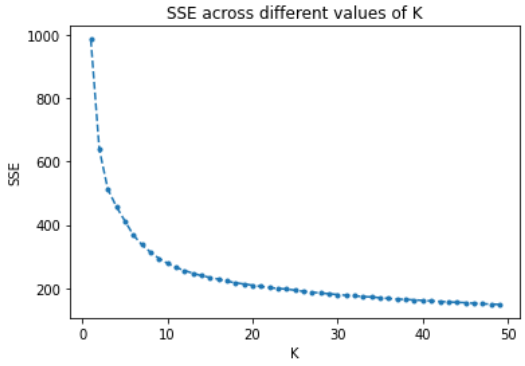
	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
0	0.995	0.708	0.1950	0.563	0.1510	-12.428	0.0506	118.469	0.7790
1	0.994	0.379	0.0135	0.901	0.0763	-28.454	0.0462	83.972	0.0767
2	0.604	0.749	0.2200	0.000	0.1190	-19.924	0.9290	107.177	0.8800
3	0.995	0.781	0.1300	0.887	0.1110	-14.734	0.0926	108.003	0.7200
4	0.990	0.210	0.2040	0.908	0.0980	-16.829	0.0424	62.149	0.0693

	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
0	0.998996	0.716599	0.1950	0.563	0.1510	0.745000	0.052219	0.485348	0.7790
1	0.997992	0.383603	0.0135	0.901	0.0763	0.494026	0.047678	0.344019	0.0767
2	0.606426	0.758097	0.2200	0.000	0.1190	0.627609	0.958720	0.439086	0.8800
3	0.998996	0.790486	0.1300	0.887	0.1110	0.708887	0.095562	0.442470	0.7200
4	0.993976	0.212551	0.2040	0.908	0.0980	0.676079	0.043756	0.254614	0.0693

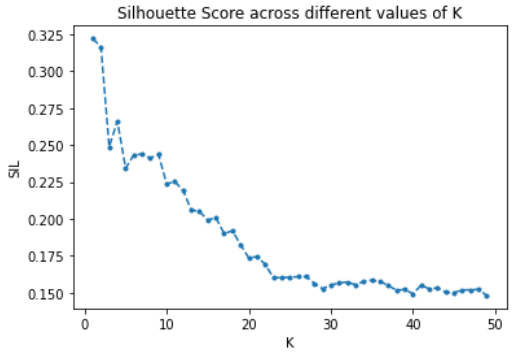
A sample of the extracted features before and after standardization

B. *Number of clusters*

The next step in the process is finding the optimal number of clusters to use for this problem. We used two different methods for finding this, as this is a somewhat subjective problem and we wanted multiple "takes" on it.

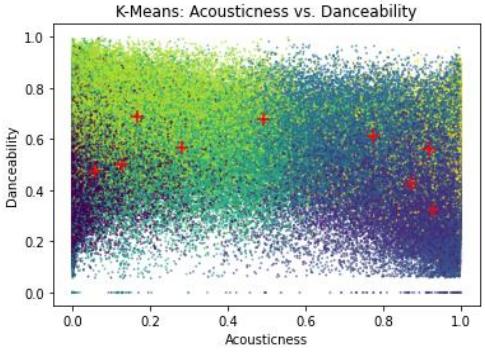


The first method is the elbow method. A small (2000 songs) random sample is taken from the dataset and the k-means algorithm is run over it many times with different amounts of clusters (1-50). The SSE (sum of squared distances) metric for each is plotted. The idea is to find the point where the SSE stops decreasing by much (the elbow). [3]



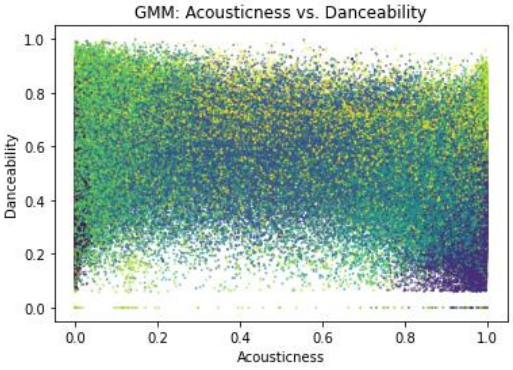
The second method was using the Euclidean silhouette scores method. Similar to the elbow method, it uses the same sample data set and the k-means algorithm is run over many times with different amounts of clusters (1-50). The SIL (Silhouette scores) metric is plotted for each point. The goal is to find the global maximum. However, having 2 clusters is not ideal for song clustering.

C. K-Means



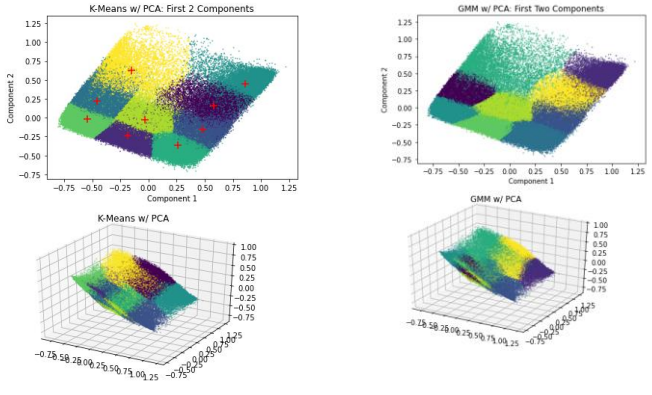
To train the model, we implemented the K-Means algorithm, which randomly selects a certain number (# of clusters k) of points to act as centroids, then iteratively updates the centroids based on the closest data points (songs) until a local optimum is reached.

D. Gaussian Mixture



We also implemented a Gaussian Mixture Model Algorithm. To train the model, we predetermine the amount of components (using the same amount of clusters in K-means). Unlike K-Means, Gaussian mixture takes variance into consideration. The variance is determined by the dimensions, as the features increase to more than size of one, the covariance equals #features squared. The Gaussian Mixture also uses an estimation algorithm, Expectation- Maximum. This takes a data point and measures the probability it belongs to each cluster. The cluster with the highest probability is then used these values to update the means and variance matrix proportionally. This allows the data points' probability to be recalculated iteratively with the new values, to increase the log-likelihood function. [4] This will always converge to a local optimum.

E. Principal Component Analysis



To improve the models, we decided to implement a dimension reduction algorithm, specifically Principal Component Analysis. The goal of this algorithm is to reduce the number of dimensions to a smaller amount of components, which try to retain the information and important relationships between the original parameters.[5] We chose to use 3 components, to maintain around 80% variance in the dataset. Both the K-Means and Gaussian Mixture models were retrained using the obtained PCA components.

F. Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering is different from K-means clustering, as it calculates the dissimilarity between points. Hierarchical clustering treats each data point as its own cluster at first. HAC calculates the distance between two clusters, chooses the two most similar clusters, and then combines them into a larger cluster. Hierarchical Agglomerative Clustering uses a bottom up approach, iteratively combining clusters until there is only one large cluster with all the data points.[6] However, in order to decide how many clusters HAC should have, a dendrogram is used to visualize the merged clusters. A threshold line is drawn horizontally, and the amount of vertical lines intersected determine the amount of clusters.[7]

G. Metrics

To evaluate the models, we were forced to only use internal metrics rather than external metrics (such as purity index), which compare our clustered labels to ground-truth class labels, as we had none.

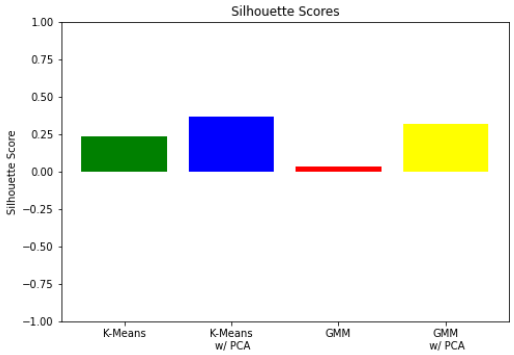
The first metric we used was the silhouette score, which essentially tries to evaluate the “distinctness” of the clusters; how similar data points in a certain cluster are compared to those of a different cluster. This is a bounded score on a scale from -1 to 1, with 1 being a perfect clustering and -1 being undesirable.[8]

The other metric used was the calinski-harabasz index, which measures the ratio of the variance between a datapoint and others in its cluster to the variance between it and data points in other clusters. A higher score is better.

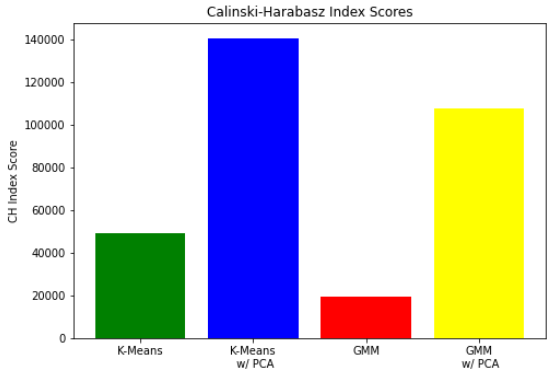
We considered using sum of squared differences (between data points and their respective centroid) to evaluate the models, however the sk-learn implementation of GMM does not have a parameter for centroids, making it impossible to calculate the SSE.[9]

IV. Results and Discussion

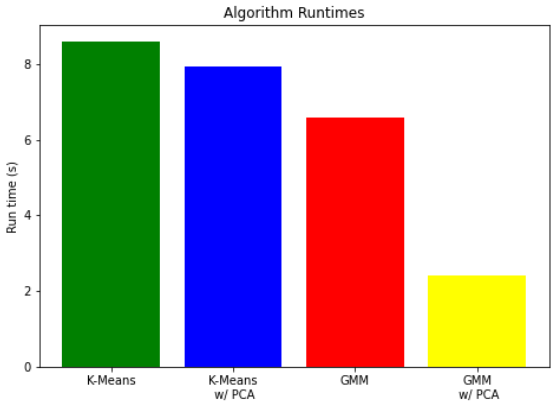
Trying to implement Hierarchical Agglomerative Clustering was not possible with our data set. As mentioned earlier, one of the challenges encountered was the sheer size of the data set. Hierarchical Agglomerative Clustering is known for not being able to handle large or even medium sized data sets. HAC computes many comparisons and combined with the large data set, uses a large amount of computing power. When trying to run HAC on our data set we encountered an error in which we had insufficient RAM.



The silhouette score's goal is to calculate the discreteness of the clusters. The data gathered shows that K-Means with or without PCA scores are more distinct vs their Gaussian Mixture Model counterparts. The Gaussian Mixture Model alone scored the worst out of all the algorithms. This is due to differences in the type of clustering K-Means and Gaussian Mixture Model are. K-Means clustering is non-probabilistic and hard assigns data, having stricter boundaries when clustering data. Gaussian Mixture Model clustering is probabilistic, having more flexibility with its shape and what cluster a data point belongs to. [10]The non-probabilistic nature of K-Means and the reduction of dimensions by PCA, is what makes it the most distinct algorithm.



The calinski-harabasz index scores similarly to the silhouette scores, in which K-Means performs better than GMM overall and PCA drastically improves performance in both models. This makes sense, as similarly to silhouette score, calinski-harabasz rewards denser and more separated clusters.



When calculating runtime of each algorithm, we didn't take into account the calculation of applying PCA to the dataset. Using the PCA dataset with K-Means and GMM decreased their runtimes respectively, with GMM with PCA having the most significant decrease. The dimensions of the dataset is lowered, allowing less computations for both K-Means and Gaussian Mixture Model. With or without PCA GMM runs faster than K-Means. This is due to the Gaussian Mixture Model algorithm being more efficient and the speed in which this algorithm finds the local minimum. .[10]

Our solution has a very low bias and high variance approach. Because we didn't do a train/test split, the model pays attention entirely to training data. This makes the model perform well on the training set, but makes it susceptible to overfitting, and it may not generalize to other datasets well.[11]

Continuing with this project, we might have tested more algorithms or combinations of the ones we tested now. One of the models we would've implemented if we continued to work on this project is deep learning. With the new clustered data set we would create a song recommendation system for users. Users would choose songs within our dataset they would like and the recommender system would return songs with similar features (songs within the cluster). Another practical application that would suit this project if we continued on is to create playlists for users. Similarly to how the recommendation system would work, the user would choose songs they like. Then the program would pick songs with similar feature values and the same cluster.

V. Implementation and Code

We made heavy use of the sk-learn library to implement our solution. To standardize our data, we used the MinMaxScaler package. To implement K-Means, GMM and PCA we used the cluster, mixture and decomposition packages respectively. The metrics package was used for silhouette score and calinski-harabasz index. We used pandas to import the dataset into a DataFrame, then used NumPy for its n-dimensional array object once the data was preprocessed. Matplotlib was used for general plotting, and after PCA reduced the number of dimensions to 3, we used mpl_toolkits to plot 3d graphs of the data. Finally, we used the time package to measure the runtime of the algorithms.

References

[1] "🎶Spotify | Data Visualization", *Kaggle.com*, 2020. [Online]. Available: https://www.kaggle.com/tanujdhiman/spotify-data-visualiation. [Accessed: 02- Oct- 2020].

[2] D. Kumar. "Introduction to Data Preprocessing in Machine Learning", *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d

[3] K. Mahendru. “How to Determine the Optimal K for K-Means?”, *Medium*, 2019. [Online]. Available: https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb

[4] A. Singh.“Build Better and Accurate Clusters with Gaussian Mixture Models”, *Analytic Vidhya*, 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/

[5] “How to Combine PCA and K-means Clustering in Python?”, *365 Data Science*. [Online]. Available: https://365datascience.com/pca-k-means/

[6] Maklin.“Hierarchical Agglomerative Clustering Algorithm Example In Python”, *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

[7] P. Sharma.“A Beginner’s Guide to Hierarchical Clustering and how to Perform it in Python”, *Analytic Vidhya*, 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/

[8] H.Wei.“How to measure clustering performances when there are no ground truth?”, *Medium*, 2020. [Online]. Available: https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c

[9] C. Ballard. “How To Evaluate Unsupervised Learning Models”, *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/how-to-evaluate-unsupervised-learning-models-3aa85bd98aa2

[10] K. Kubara. “Gaussian Mixture Models vs K-Means. Which One to Choose?”, *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/gaussian-mixture-models-vs-k-means-which-one-to-choose-62f2736025f0

[11] S. Singh. “Understanding the Bias-Variance Tradeoff”, *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229