Music Genre Analysis

Rohan Kumar Sah

February 15, 2025

Abstract

This report was written to explain my thought process and methodology and to compile all the aiding visualisations and results in one place for KDAG 2025 Selections - Core Team Task 2. Please refer to the notebook as all the code is explained there first. I will avoid including any code in this report.

1 Introduction

Each sub-task is discussed in a separate section. The libraries used are as follows :

- pandas : for interacting with the dataset
- numpy : for making life easier
- seaborn + matplotlib : for visualisation
- random : for initialising centroids in clustering algorithm

2 Vectorisation

For generating embeddings, we were supposed to choose between BoW (Bag of Words) and TF-IDF. I went with BoW for the following reasons :

- BoW is extremely simple to implement and performs well for small datasets like this one.
- TF-IDF would have been a better choice if we were working with a text corpus but for this keyword analysis BoW does the trick.
- TF-IDF would recognise common words and reduce their impact, but here each keyword is equally important.

  It would be silly to discriminate between guitar and violin - see Table 1, because of frequency since there is no semantics involved here due to the dataset being three keywords, and TF-IDF doesn't serve its purpose well.

keyword_1	Frequency
banjo	6
brass	11
guitar	65
piano	12
synth	43
violin	10

Table 1: keyword_1 and frequency comparison

3 Dimensionality reduction

This involves standard PCA code using numpy. We simplified the sparse vectors while retaining helpful information.

4 Combining the embeddings into one

There are multiple ways of combining embeddings. Concatenation is out of the question since we are trying to preserve the dimensions. Since all keywords are equally important, we averaged the vectors. Simply summing up can cause larger magnitudes we do not want to deal with.

$$V_{\text{final}} = \frac{1}{3} \sum_{i=1}^{3} V_i$$

Cross Product would collapse the vector into a scalar, and we would lose information. For K-Means (which we will implement in the next section), average is the best choice.

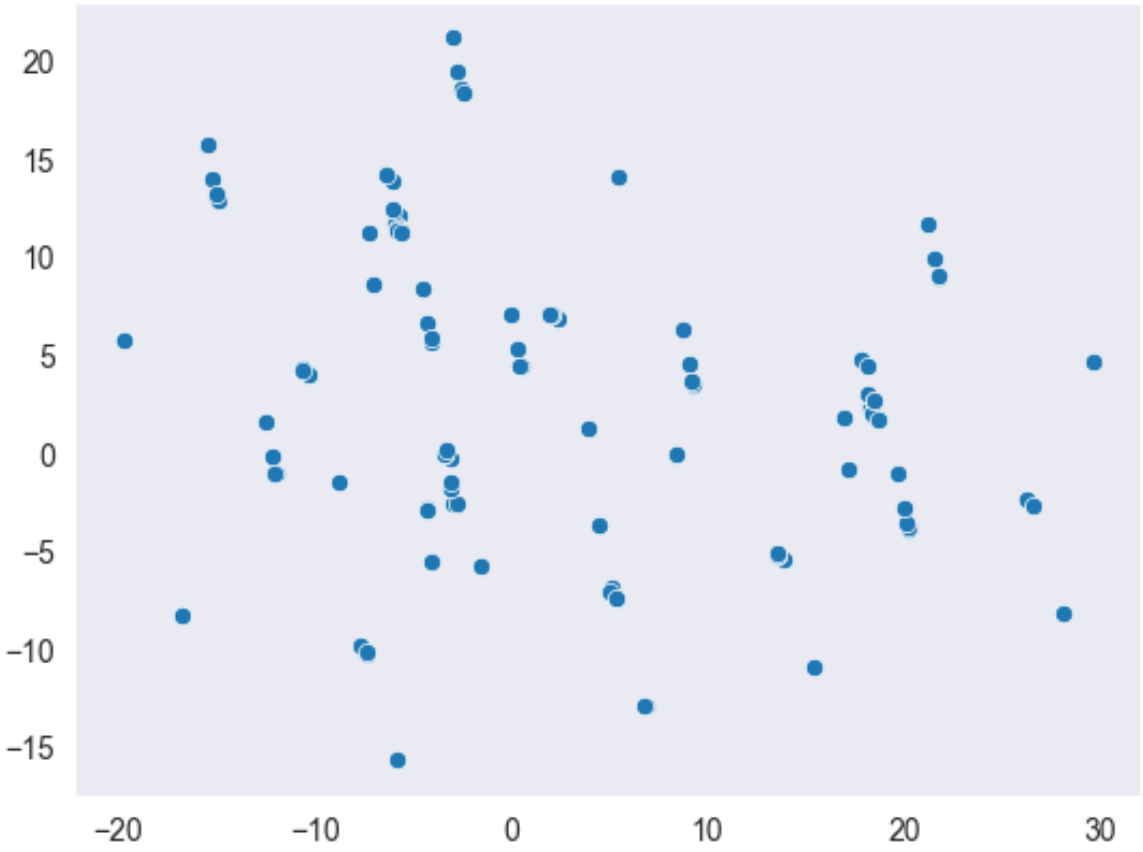

Figure 1: Vectors after combination

5 Clustering

I started with the K-Means algorithm for clustering; looking at the WCSS (Within-Cluster Sum of Square) and Silhouette Scores for different cluster numbers gives an idea of what value to take. Looking at the Elbow Point in the graph and considering that there are five genres means we should take k=5 for further analysis.

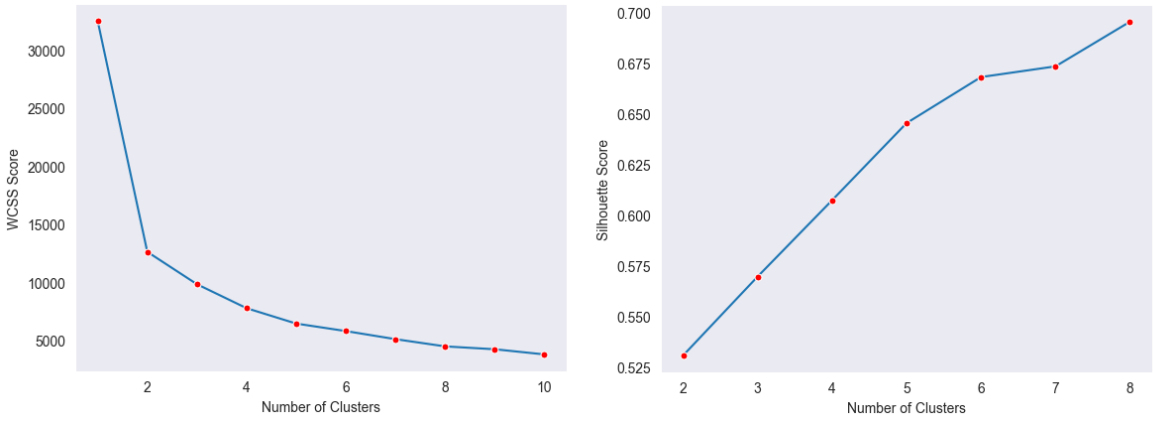

Figure 2: Metrics averaged over 100 iterations

However, soon, a problem arises: the K-Means algorithm can sometimes be inaccurate. Since the centroids are randomly initialised, if a centroid is initialised too far from the points, it won't shift, and if two centroids are initialised too close, they probably won't move.

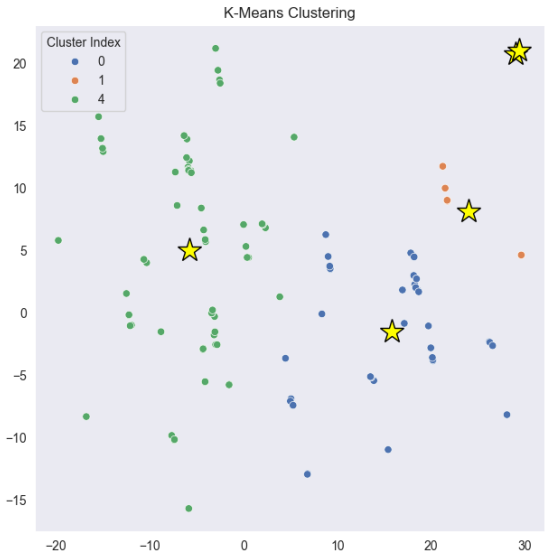

Figure 3: A failed K-Means example where two centroids are at a corner and belong to no cluster

The problem lies in centroid initialisation, so we must find a better way. We, therefore, shift to K-Means++, where centroids are initialised iteratively. Failures are almost eliminated by this method.

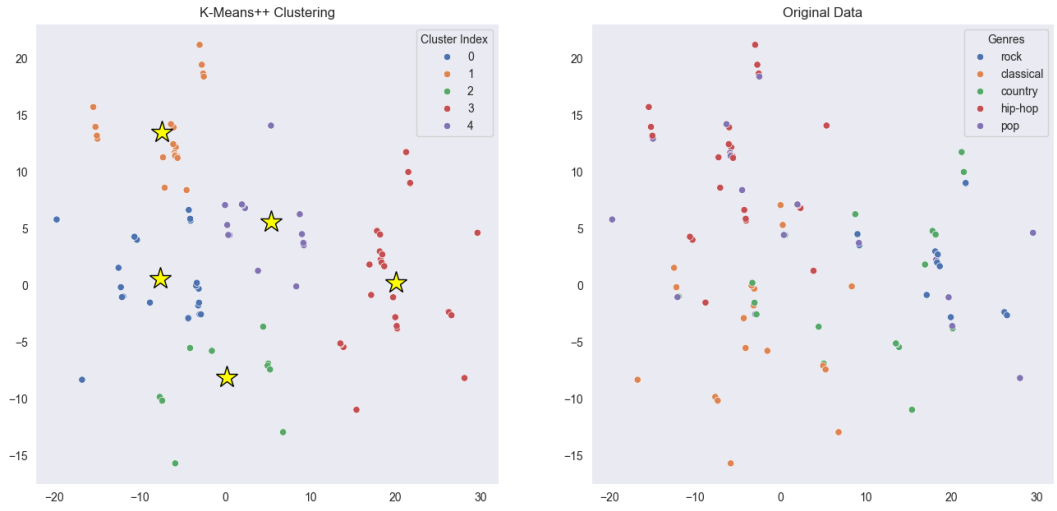

Figure 4: Final Clustering

6 Analysis

6.1 Percentage distribution of ground truth genres in each cluster

We plot a heatmap, comparing percentages of the genres within each cluster. We observe the following

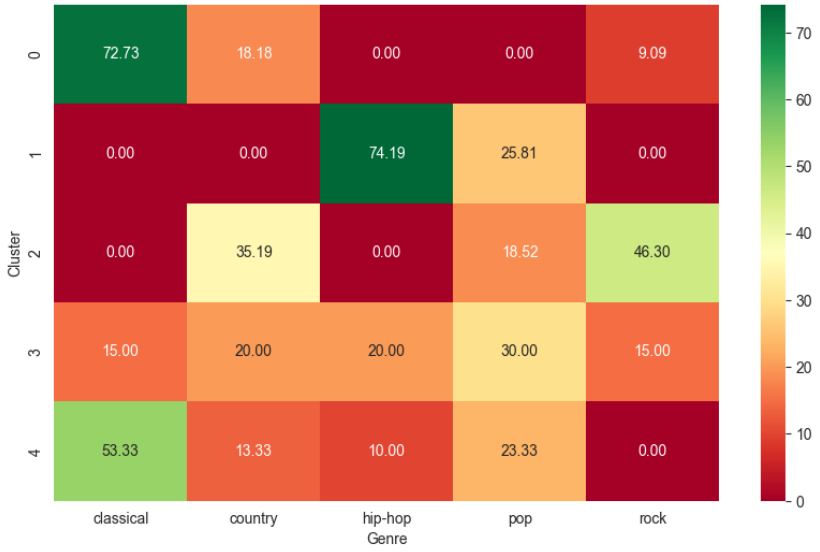

Figure 5: Heatmap showing the percentages of different genres within each cluster

:

- Cluster 0 is mostly populated by Classical (72.73%)
- Cluster 1 is mostly populated by Hip-Hop (74.19%)
- Cluster 2 is not dominated by a single genre
- Cluster 3 has a more even distribution then Cluster 2
- Cluster 4 is also mostly classical

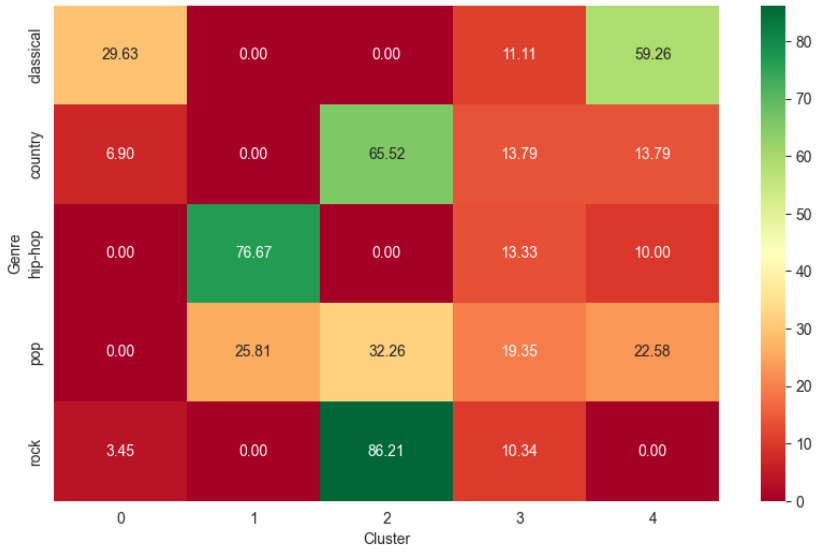

Figure 6: Heatmap showing the percentage of songs from each genre in each cluster

If we switch our calculation a bit we observe something else as well. Rock is the best clustered genre because most of its points are within the same genre. The Clustering of Hip-Hop and Country is also not that bad. Pop is the most poorly clustered of all.

6.2 Silhouette Score

We already have the general idea of the Silhouette Score for different number of clusters - see Figure 2. The silhouette score for a single sample is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where:

- $a(i)$ is the average intra-cluster distance (mean distance to other points in the same cluster).
- $b(i)$ is the average nearest-cluster distance (mean distance to points in the nearest neighboring cluster).

The average silhouette score for 5 clusters over 100 iterations is:

$$\text{Average Silhouette Score} = 0.65460999$$

For most iterations with k=5, the silhouette score remains within the range:

$$0.6 \leq s \leq 0.7$$

where s represents the silhouette score. This is a good clustering score considering that we used the most simplistic approaches to the problems in this task.

6.3 Genre Prediction

On converting the keywords to vectors and finding the nearest centroid we obtain the following results

- `['piano', 'calm', 'slow']` belongs to cluster 2
- `['guitar', 'emotional', 'distorted']` belongs to cluster 0
- `['synth', 'mellow', 'distorted']` belongs to cluster 1

We can refer to Figure 5 and make the following observations :

- Cluster 2 is not really dominated by a single genre but we can sort `['piano', 'calm', 'slow']` into Country
- Cluster 0 is dominated by Classical so we can sort `['guitar', 'emotional', 'distorted']` into Classical
- Cluster 1 is dominated by hip-Hop so we can sort `['synth', 'mellow', 'distorted']` into Hip-Hop