Music Genre Analysis

By: Tadi Joshua Raj

Task 2 Report

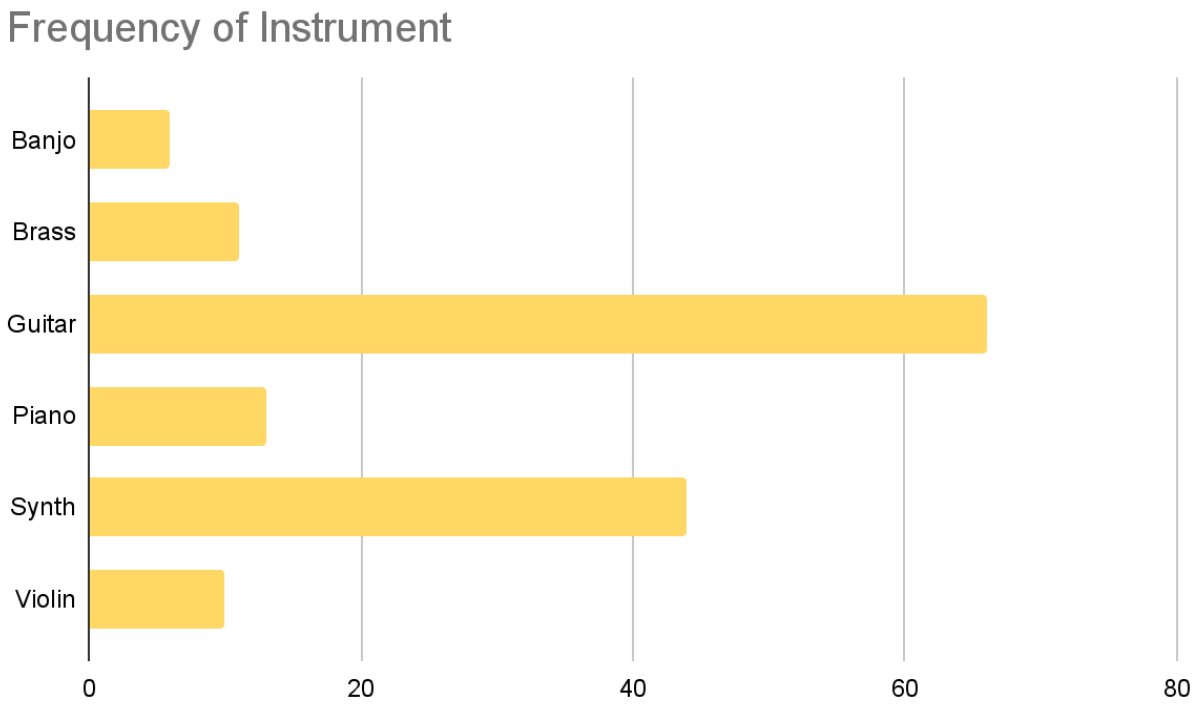
Executive Summary

In this task, I was given a **dataset** containing information on songs, which **contained keywords** describing the songs, **and** their **corresponding genre** labelled **as** the **ground truth**.
My task was to **find meaningful insights** in the correlation between the keywords and genre, and then **predict the genre of a few songs** which were associated with a certain combination of keywords.

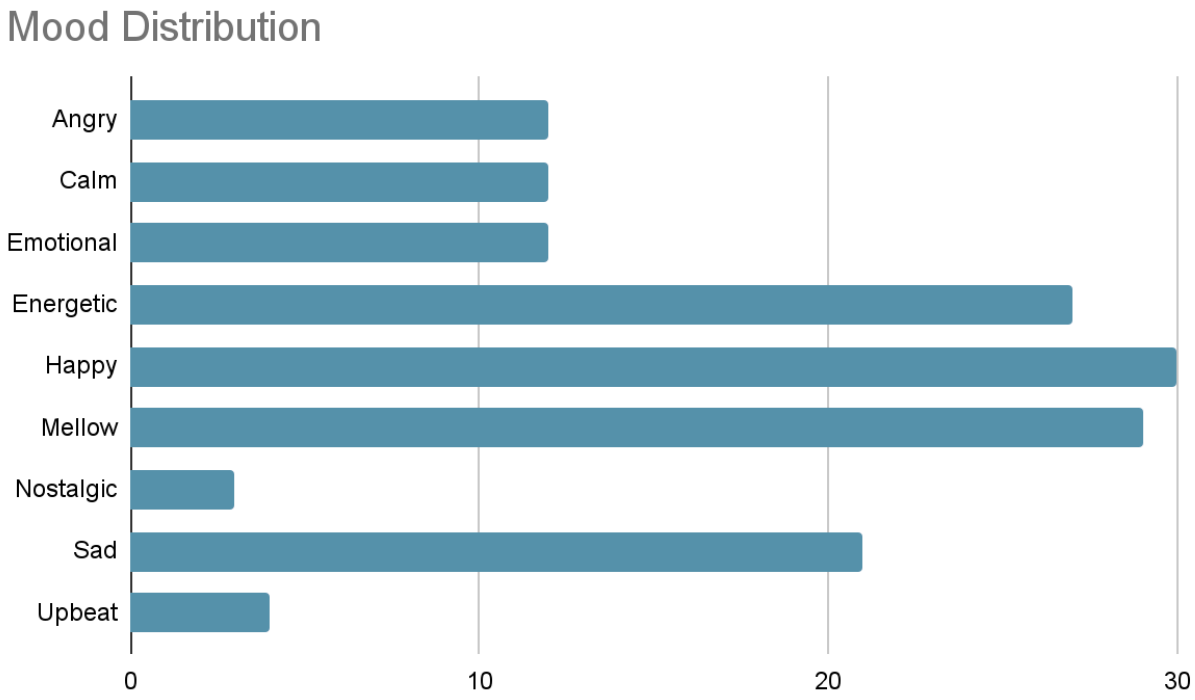
Introduction

Analyzing the dataset

The dataset provided had a total of **147 songs** and each song had **3 keywords** describing it and **a genre** associated with it.
The first keyword gives us information about the instrument used in the song, and there are a total of 6 instruments used, they are banjo, brass, guitar, piano, synth and violin.
Guitar is the most widely used instrument (used in 66 songs), and **banjo is the least** used instrument (used only in 6 songs).



Moving onto the second keyword, this keyword describes the mood of the song. Most of the songs seem to be either **energetic, happy, mellow or sad.**



The third keyword seems to describe the feel of the song, most of them are either **fast, melodic, slow or upbeat.**

Pre-Requisites

In my solution I have used some properties of eigenvectors, covariance of a matrix and euclidean distance.

Problem Statement

With the given dataset, apply the following machine learning techniques:

1) Word Vectorization
2) Principal Component Analysis (PCA)
3) Combine PCs to form a 2D plot of the songs
4) K-Means Clustering
5) Analysis

Under analysis identify the percentage ground truth in each cluster, get the silhouette score of the k-means clustering and assign genres to the following keywords:

- [piano, calm, slow]
- [guitar, emotional, distorted]
- [synth, mellow, distorted]

My Approach

Word Embeddings

Bag of words:

This method of word embedding is straightforward, we take all the keywords and make a dictionary using them, then we construct a vector of length equal to the length of the dictionary and populate it using the frequency of the keywords.
But, this method, as I feel, has a few drawbacks.

- It **does not give** us a **relation** between the keywords.
- It **does not show** the **importance** of a keyword compared to the others.

Now, there is another method of word embedding which tackles the above issues. This method is TF-IDF.

TF-IDF:

In this method we take multiple documents, here I have made **each song as a document,** now in each document we check the term frequency (also we can make many types of dictionaries, more on that below), and across all the documents we get the term frequency now denoted as document frequency. **To vectorize** a document (here song) we first **represent** the **document as a combination of words** (from our created dictionary), and we **multiply** the **term frequency** of that word with the **inverse of the document frequency** of that word.
I have made 5 different dictionaries:

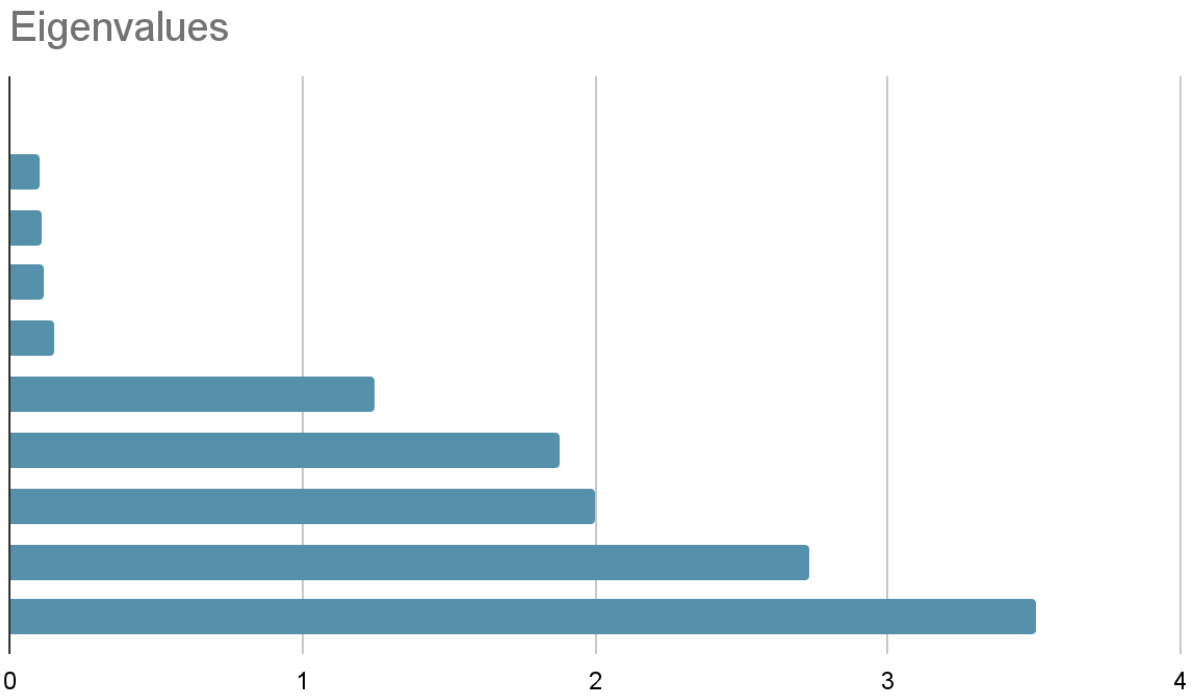
1. UniGram - Treating each unique word as an item - 24 elements
2. BiGram - Treating a unique set of 2 words as an item - 99 words
3. TriGram - Treating a unique set of 3 words as an item - 125 words
4. UniCharGram - Treating each character present in the dataset as an element - 22 words
5. BiCharGram - Treating a pair of characters as an element - 161 words

We shall see which method is the best later, for now I am applying the same process to all the different dictionaries.

Dimensionality Reduction

The algorithm I'm using for this is Principal Component Analysis.
In this, we identify the most important features describing the vectors. We simply make a covariance matrix of all vectors, then we find the eigenvalues of this matrix, these eigenvalues show us how much variation the vectors show corresponding to an axis in n dimension (here n is the size of the covariance matrix), this gives us n axes, and n eigenvalues corresponding to each axis, if we simply multiply the covariance matrix nxn with an eigenvector corresponding to an eigenvalue nx1 we get the coordinates of all song on that axis as a vector nx1.

And this is denoted as a principal component PC. Graph below here shows a few eigenvalues (with UniCharGram dictionary)



Combining Word Embeddings

As we can see from the eigenvalues from above, only a few show significant variation, hence we take only those PCs corresponding to them, we take 2 PCs from these and plot our songs on them, since PCs are just axes.
But since we get more than 2 PCs which are significant, we have to combine our significant pairs of PCs. One simple method of doing this is just taking the mean on them. And make plots
Once we do this we find that some dictionaries (the 5 which I defined earlier) show very little clustering, while others show decent clustering.

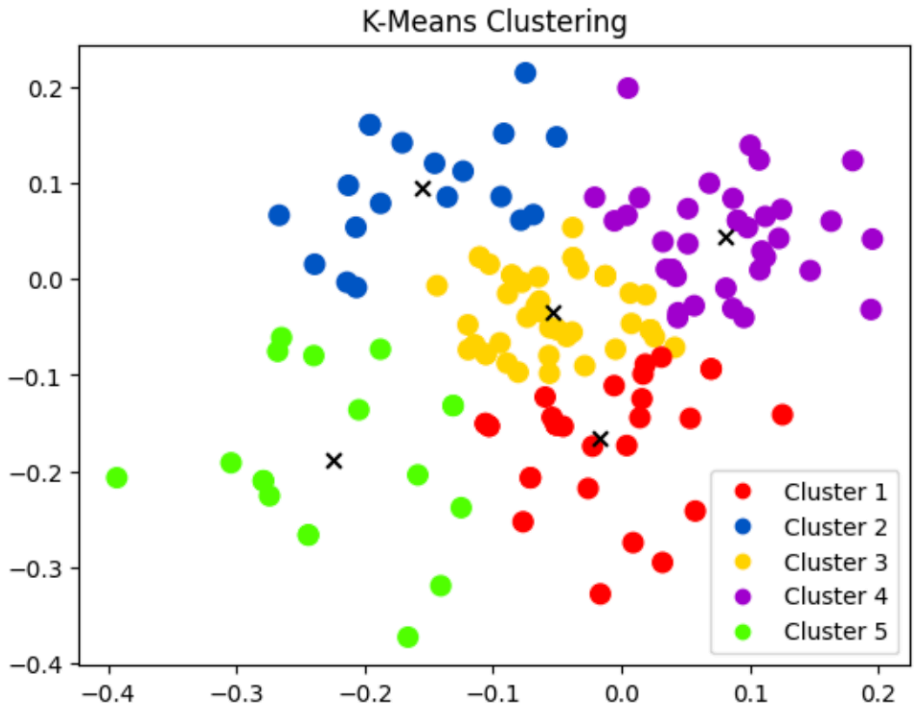
Clustering

I am applying k-meet clustering (recommended), in this we just randomly select k number of points and assign them as centroids of clusters. And assign the points close to the centroid to that cluster. Then we find the centroids of the newly formed clusters and reassign all the points again to the closest centroid respectively, once we iterate this many times, the centroids wont change anymore, hence we get nice clusters (supposedly).

Result

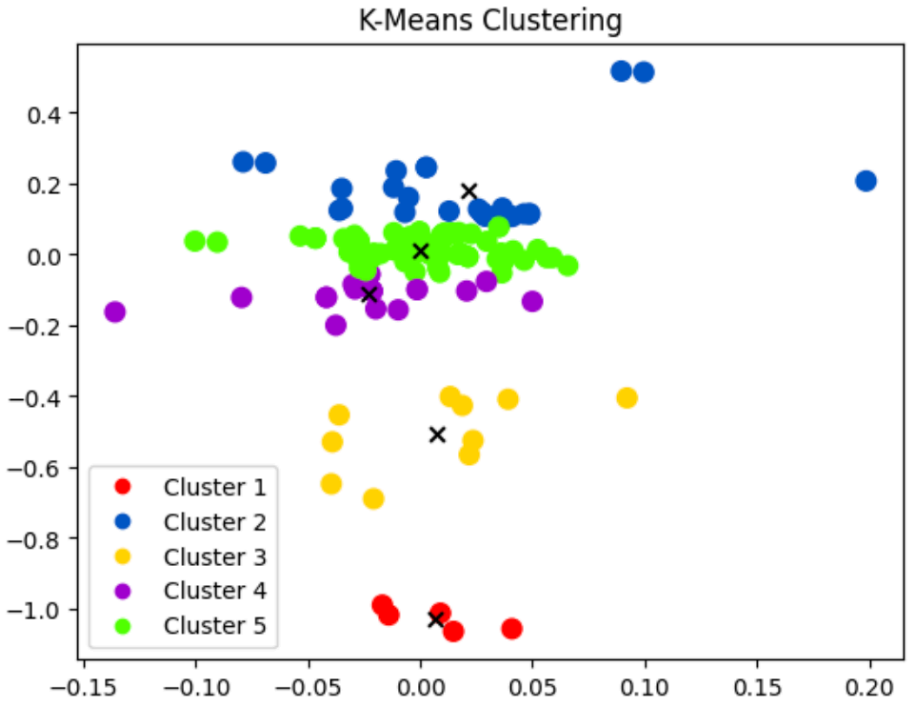
UniGram dictionary:

```
Cluster Genre Distribution:
    [Cluster 1] Classical: 44.83%, Country: 17.24%, Hip-Hop:   6.9%, Pop: 10.34%, Rock: 20.69%
    [Cluster 2] Classical:   0.0%, Country: 21.05%, Hip-Hop: 31.58%, Pop: 47.37%, Rock:   0.0%
    [Cluster 3] Classical: 10.87%, Country: 19.57%, Hip-Hop: 19.57%, Pop: 26.09%, Rock: 23.91%
    [Cluster 4] Classical:  25.0%, Country:  8.33%, Hip-Hop: 36.11%, Pop: 16.67%, Rock: 13.89%
    [Cluster 5] Classical:   0.0%, Country: 47.06%, Hip-Hop:   0.0%, Pop:  5.88%, Rock: 47.06%
The song with keywords ['piano' 'calm' 'slow'] might belong to cluster 1
The song with keywords ['guitar' 'emotional' 'distorted'] might belong to cluster 3
The song with keywords ['synth' 'mellow' 'distorted'] might belong to cluster 5
Silhouette Score: 0.3531
```



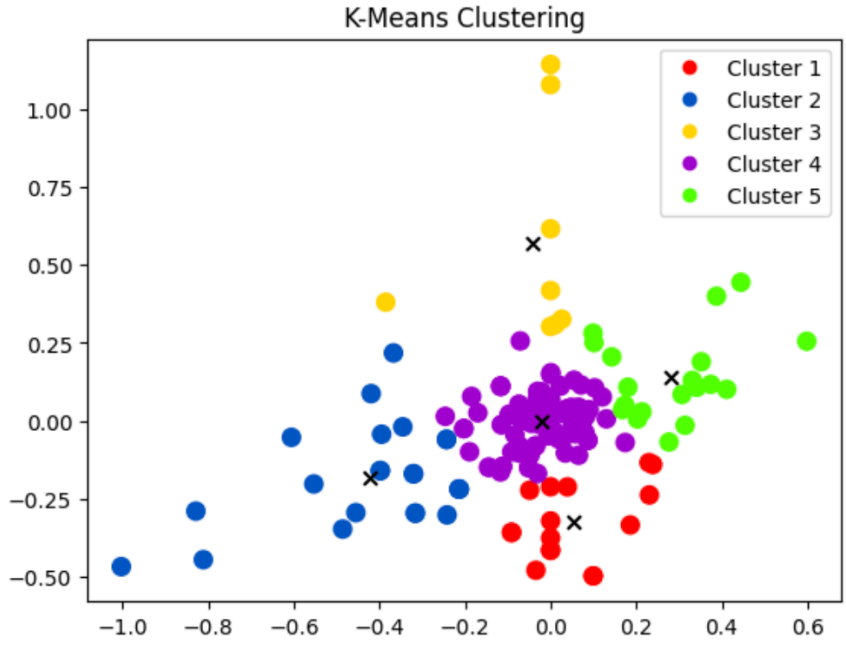
BiGram dictionary:

```
Cluster Genre Distribution:
    [Cluster 1] Classical:  40.0%, Country:   0.0%, Hip-Hop:   0.0%, Pop:  60.0%, Rock:   0.0%
    [Cluster 2] Classical: 14.81%, Country: 25.93%, Hip-Hop: 14.81%, Pop:  7.41%, Rock: 37.04%
    [Cluster 3] Classical:  10.0%, Country:  30.0%, Hip-Hop:  30.0%, Pop:  20.0%, Rock:  10.0%
    [Cluster 4] Classical: 15.79%, Country: 10.53%, Hip-Hop: 21.05%, Pop: 36.84%, Rock: 15.79%
    [Cluster 5] Classical: 19.77%, Country: 19.77%, Hip-Hop: 22.09%, Pop: 19.77%, Rock:  18.6%
The song with keywords ['piano' 'calm' 'slow'] might belong to cluster 2
The song with keywords ['guitar' 'emotional' 'distorted'] might belong to cluster 5
The song with keywords ['synth' 'mellow' 'distorted'] might belong to cluster 5
Silhouette Score: 0.5102
```



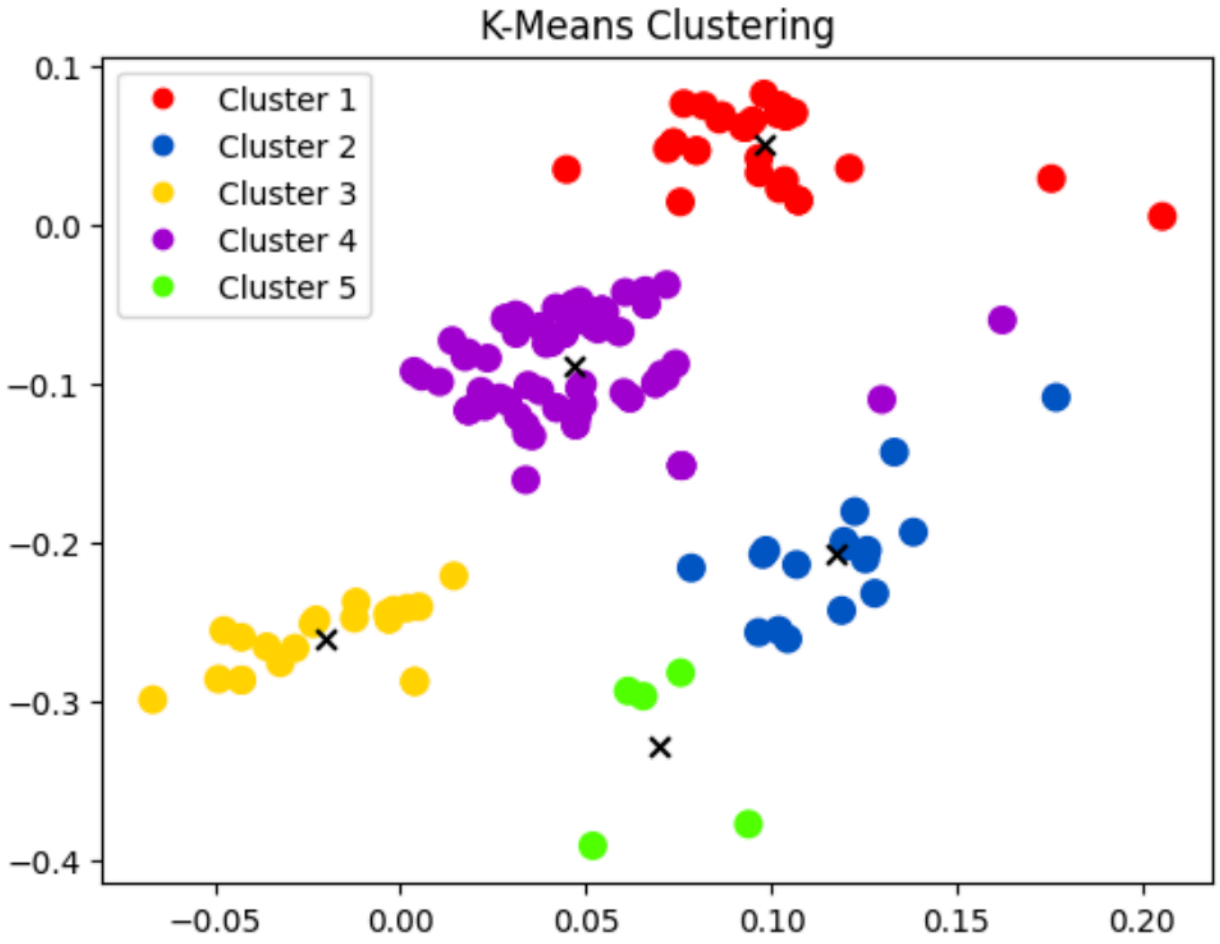
TriGram Dictionary:

```
Cluster Genre Distribution:
    [Cluster 1] Classical: 18.75%, Country:  12.5%, Hip-Hop: 31.25%, Pop: 18.75%, Rock: 18.75%
    [Cluster 2] Classical:  4.55%, Country: 27.27%, Hip-Hop: 22.73%, Pop: 27.27%, Rock: 18.18%
    [Cluster 3] Classical:  37.5%, Country:  12.5%, Hip-Hop:  25.0%, Pop:  25.0%, Rock:   0.0%
    [Cluster 4] Classical: 20.99%, Country: 18.52%, Hip-Hop: 12.35%, Pop: 22.22%, Rock: 25.93%
    [Cluster 5] Classical:  15.0%, Country:  25.0%, Hip-Hop:  40.0%, Pop:  10.0%, Rock:  10.0%
The song with keywords ['piano' 'calm' 'slow'] might belong to cluster 4
The song with keywords ['guitar' 'emotional' 'distorted'] might belong to cluster 4
The song with keywords ['synth' 'mellow' 'distorted'] might belong to cluster 2
Silhouette Score: 0.4078
```



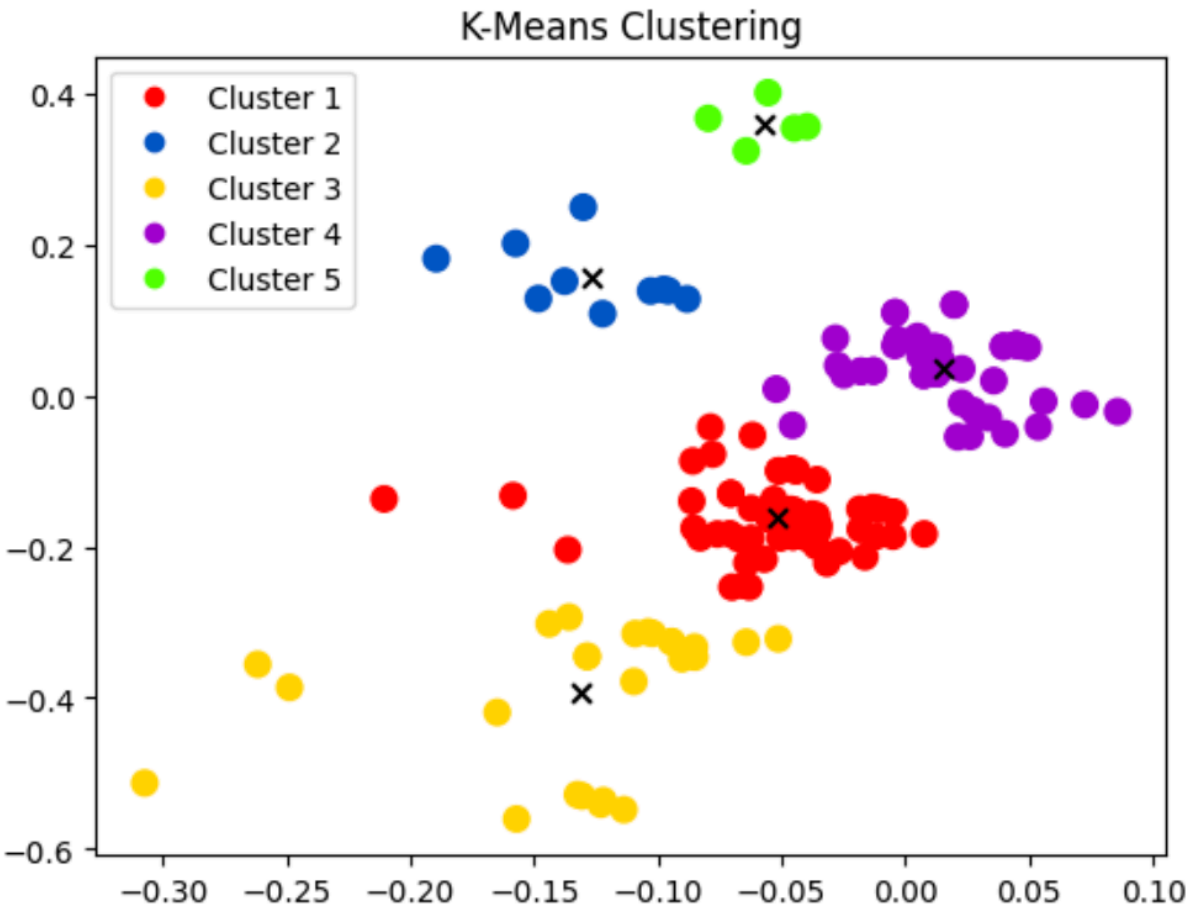
UniCharGram Dictionary:

```
Cluster Genre Distribution:
    [Cluster 1] Classical: 26.47%, Country: 29.41%, Hip-Hop:   0.0%, Pop: 14.71%, Rock: 29.41%
    [Cluster 2] Classical: 35.29%, Country: 17.65%, Hip-Hop: 23.53%, Pop:   0.0%, Rock: 23.53%
    [Cluster 3] Classical:  4.76%, Country: 23.81%, Hip-Hop: 42.86%, Pop: 28.57%, Rock:   0.0%
    [Cluster 4] Classical: 11.43%, Country: 15.71%, Hip-Hop: 22.86%, Pop: 28.57%, Rock: 21.43%
    [Cluster 5] Classical:  60.0%, Country:   0.0%, Hip-Hop:  20.0%, Pop:   0.0%, Rock:  20.0%
The song with keywords ['piano' 'calm' 'slow'] might belong to cluster 4
The song with keywords ['guitar' 'emotional' 'distorted'] might belong to cluster 1
The song with keywords ['synth' 'mellow' 'distorted'] might belong to cluster 3
Silhouette Score: 0.6282
```



BiCharGram Dictionary:

```
Cluster Genre Distribution:
    [Cluster 1] Classical: 26.23%, Country: 13.11%, Hip-Hop: 29.51%, Pop: 18.03%, Rock: 13.11%
    [Cluster 2] Classical:   0.0%, Country:  30.0%, Hip-Hop:  60.0%, Pop:   0.0%, Rock:  10.0%
    [Cluster 3] Classical: 13.04%, Country:  4.35%, Hip-Hop: 26.09%, Pop: 56.52%, Rock:   0.0%
    [Cluster 4] Classical: 16.67%, Country: 35.42%, Hip-Hop:   0.0%, Pop: 14.58%, Rock: 33.33%
    [Cluster 5] Classical:   0.0%, Country:   0.0%, Hip-Hop:   0.0%, Pop:   0.0%, Rock: 100.0%
The song with keywords ['piano' 'calm' 'slow'] might belong to cluster 1
The song with keywords ['guitar' 'emotional' 'distorted'] might belong to cluster 3
The song with keywords ['synth' 'mellow' 'distorted'] might belong to cluster 3
Silhouette Score: 0.5987
```



Conclusion

As we can clearly see when we define the dictionary as bi character grams we get the best results.

Also, we can see that unigram has the worst result, with all the songs scattered, when we move to bigram, it becomes better, a possible reason for this could be that the keywords are getting mixed now or in other words the machine is making a relation between the keywords, this can be further verified as in trigram, we see worser clustering (comparing silhouette score), in trigram we see that the keywords are not mixing.

Now stemming from this we can predict that the uni character gram and bi character gram methods will be even better, since even more mixing of keywords happens in them, and from the results it's clear that they **are** better.

We also notice that uni char gram has higher silhouette score than the bi char gram, but the genre distribution of bi char gram is much better, this shows that bi char grams are able to represent the data better (since it has 161 words in its vocabulary) than uni char gram (which has only 22 words in its dictionary)

Also, my model (biCharGram, since it gives best results) predicts that the song with genres:

- [piano, calm, slow] = classical or hip-hop (put in cluster 1)
- [guitar, emotional, distorted] = hip-hop (put in cluster 3)
- [synth, mellow, distorted] = hip-hop (put in cluster 3)