KDAG Selection Task 2

Arul Rana

February 15, 2025

1 Introduction

This report is for Task 2 for KDAG selections. The data set contains information on different songs classified by genre. The project first requires us to vectorize the information using methods like BoW or TF IDF, somehow combine the three different keywords, and use the Dimensionality reduction algorithm, specifically PCA, to visualize the data better and perform clustering.

2 Generate vectors for the three keywords

2.1 Why do we need this?

This technique converts textual data (like a paragraph or an essay) into vectors of numbers. This is required because computers understand numbers, not words. More specifically, most Machine Learning models work on numbers. Converting Natural Language into numbers is also challenging due to many factors. Some include the ambiguous nature of words and the unstructured nature of language.

2.2 Bag of Words

This method of vectorization works by finding the frequency of a single token in each document and forms vectors based on the frequency of each word.

Document 1	KDAG is the Data Analysis Group of the IIT Kharagpur
Document 2	KDAG is Kharagpur Data Analysis Group

Table 1: Examples of 2 Documents

	KDAG	is	the	Data	Analysis	Group	of	IIT	Kharagpur
Document 1	1	2	1	1	1	1	1	1	1
Document 2	1	1	0	1	1	1	0	0	1

Table 2: Vectors created by Bag of words.

Preprocessing the data before implementing Bow is essential for optimal results. Changing all characters to lowercase, removing Stopwords, etc., is necessary for clarity and ease of translation of textual data to vectors.

2.3 TF IDF

TF IDF works very similarly to Bag of words but replaces the frequency of a token with some other value defined by TF*IDF Now, what is TF? It is defined as

$$TF = \frac{\text{Number of times token t appears in a document}}{\text{Total number of tokens in the document}} = \frac{X_t^i}{\sum\limits_{v}^{S} X_v^i}$$

X_t^i represents the frequency of word t in document i, and S is the set of all tokens in the document.
IDF is defined as:

$$IDF = log(\frac{N}{df_i})$$

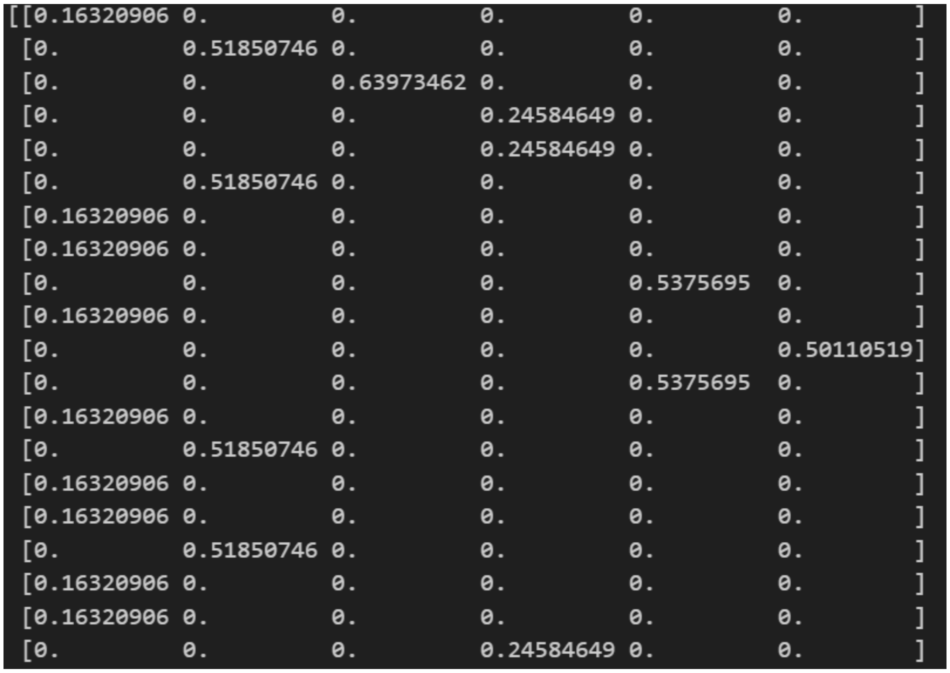
N is the total number of documents in the corpus, and df_t is the number of documents containing the term.

TF represents how often a token appears in a document, while IDF represents the importance of the token in the whole corpus of documents. So, $TF * IDF$ is a numerical value that is a factor in both of these properties of the token.

TF IDF differs from Bow as it considers the token's importance instead of just the frequency.

2.4 Which one did I use for tasks?

I used TF IDF instead of Bag of Words for this task. Due to the nature of the dataset, the value of TF was the same as the frequency of the Bow. But still, IDF was also adding the importance of the token to the data, so I felt it was better.



```
[[0.16320906 0.         0.         0.         0.         0.        ]
 [0.         0.51850746 0.         0.         0.         0.        ]
 [0.         0.         0.63973462 0.         0.         0.        ]
 [0.         0.         0.         0.24584649 0.         0.        ]
 [0.         0.         0.         0.24584649 0.         0.        ]
 [0.         0.51850746 0.         0.         0.         0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.5375695  0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.50110519]
 [0.         0.         0.         0.         0.5375695  0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.         0.51850746 0.         0.         0.         0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.         0.51850746 0.         0.         0.         0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.16320906 0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.24584649 0.         0.        ]
```

Figure 1: Some vectors formed from TF IDF method for keyword 1

3 Dimensionality reduction

After using vectorization techniques like Bow or TF IDF, we are left with high-dimensional, hard-to-visualize vectors. Dimensionality reduction uses algorithms to reduce the dimensions of vectors while preserving essential information.

3.1 Principal Component Analysis

Principal Component Analysis, or PCA for short, is a technique for dimensionality reduction and feature extraction. Let us talk about how it is implemented. First, we need to center the data, subtracting the data's average from all the vectors.

$$X_c = X - \frac{\sum X}{n}$$

Here, X is the matrix where each row corresponds to the vector of each dataset instance, anDatasetet is the centralized matrix. Now, we calculate the Correlation Matrix, which is defined as

$$C = \frac{X_c.X_c^T}{(N-1)}$$

Where N is the number of elements in the matrix, now we have to calculate the Eigenvalues and Eigenvectors.

$$|C - \gamma I| = 0$$

This equation will have many solutions for γ, and each solution will correspond to different Eigenvalues. To calculate Eigenvectors, we will use the relation

$$C.X = \gamma X$$

Here X is a matrix defined as

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Now, the Eigenvector is given as

$$Eigenvector = \frac{X}{\sqrt{\sum x_i^2}}$$

The eigenvector with the highest eigenvalue is the Principal Component of the dataset. Select the largest k eigenvalues and the corresponding eigenvectors depending on your desired output's dimension vector. Concatenate the eigenvectors and call the matrix V.

Now, to transform the original data, multiply it by V.

$$Z = X_c V$$

Z here represents the output data. While the mathematics in PCA is somewhat complex, the numpy library already has functions to multiply matrices, find eigenvalues, eigenvectors, etc.

4 Combining the embeddings into one

My first intuition was to combine all three keywords into a single vector from the start, as it would be easier. However, I quickly concluded that this was not optimal as the retention of information during PCA decreases when it is fed with higher dimension vectors. A long vector will lose much information when PCA is implemented on it. So, applying PCA to smaller vectors seemed more efficient. Hence, I used all three keywords separately. This left me with three arrays of 2d points. I had to combine them somehow, and I chose to add them together as no other method to retain them as 2D points and combine the three arrays seemed feasible.

5 Clustering

Clustering in data science is a technique used to group similar data points based on their characteristics. It's an unsupervised learning method that helps identify patterns and structures within datasets without predefined labels.

5.1 Why did I choose K-means clustering?

We were given the freedom to choose any clustering algorithm. I decided to use K-means clustering. This is due to the fact that you can select the number of clusters in this algorithm and due to the fact we already know the dataset contains songs of either of the five genres- ['rock', 'classical', 'country', 'hip-hop', 'pop'] So it makes sense to make 5 clusters. Many other clustering algorithms don't let you choose the number of clusters you want the data to be separated into.

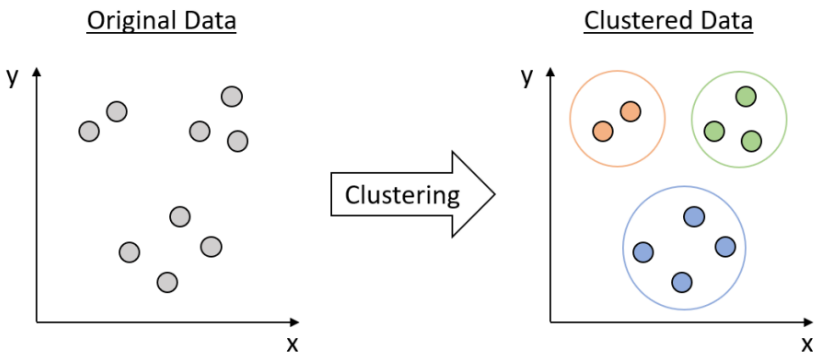


Figure 2: Example of how clustering works

5.2 K-means clustering

The algorithm is relatively simple. It first initializes k centroids randomly from the data. It then adds each point to a cluster of the centroid to which it is closest. The centroid coordinates are then updated to the average of their corresponding clusters.

$$x_c = \frac{\sum x}{n} \quad y_c = \frac{\sum y}{n}$$

This process continues until a maximum iteration or if each iteration is similar to the previous one.

This method has one problem, which is randomness. This will give different clusters every time you run it. This is due to the random selection of the initial centroids.

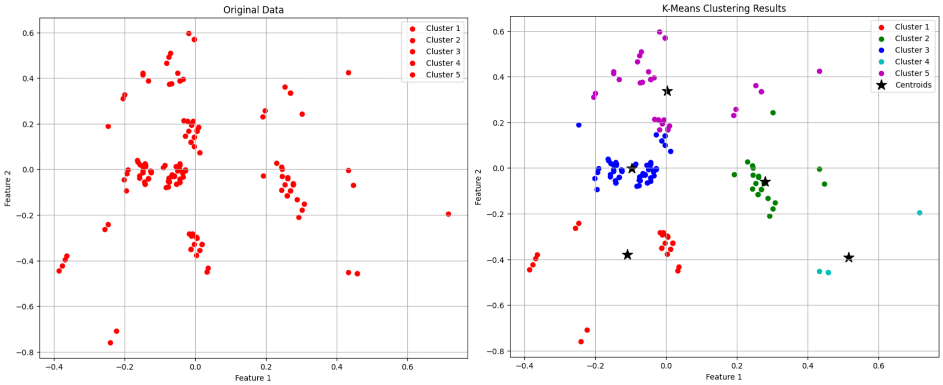


Figure 3: My own K-means algorithm used to cluster data

6 Analysis

6.1 Silhouette Score

The silhouette score is a metric used to evaluate the quality of clustering results, ranging from -1 to +1. It measures how well each data point fits into its assigned cluster compared to other clusters. This measures intra-cluster similarity.

Compute b: For the same point, find the smallest average distance to all points in any other cluster to which the point does not belong. This measures the nearest cluster distance. a is the average distance of the data point to all other data points in the same cluster. For each data point, the silhouette score is

$$Score = \frac{b - a}{\text{Max(a, b)}}$$

Now, the overall silhouette score is given by the average of $Score$ for all the points

$$\text{Silhouette Score} = \frac{\sum Score}{n}$$

n is the number of points.

6.1.1 Use

To find the best k for k-mean clustering, we can compare the silhouette score for different values, and the maximum score will denote the best k value for the data. Due to the randomness of k-means clustering, I got different best k values every time, but five was constantly in the top 3, which is consistent with our datset.

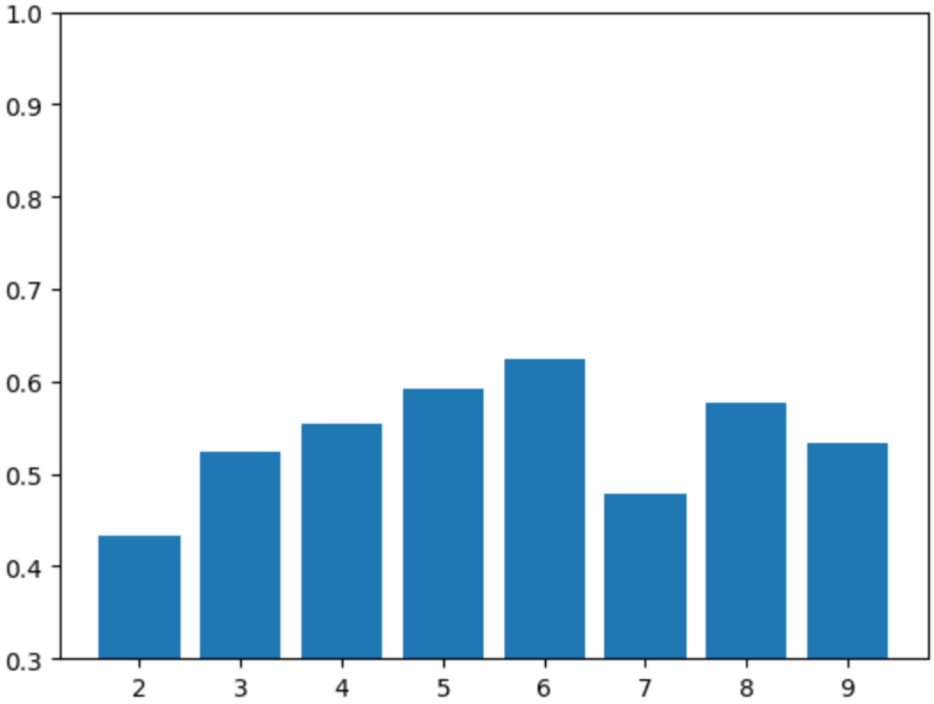


Figure 4: plot of silhouette scores vs k

Additionally, the silhouette score represents the quality of clusters formed in the dataset.

7 End Result

After finally doing the k-mean cluster, the silhouette score for $k = 5$ came as

$$Score = 0.6412589610752819$$

This score is decent but not the best score. I believe major improvements can be made by changing the way to combine the three vectors and doing it before PCA. I think this because PCA inherently causes a loss of information, and intuitively combining before PCA makes more sense. But Concatenating them before PCA directly would result in a higher loss of information due to higher dimentions being fed into the PCA algoritm, so I decided to discard this approach.

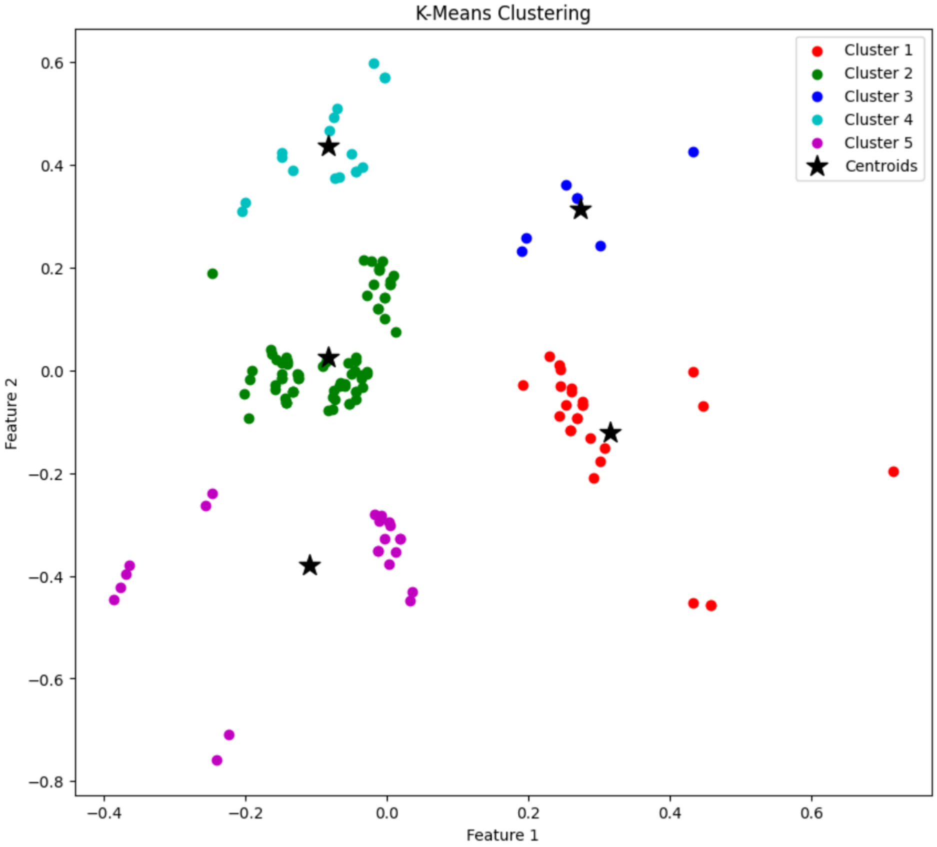
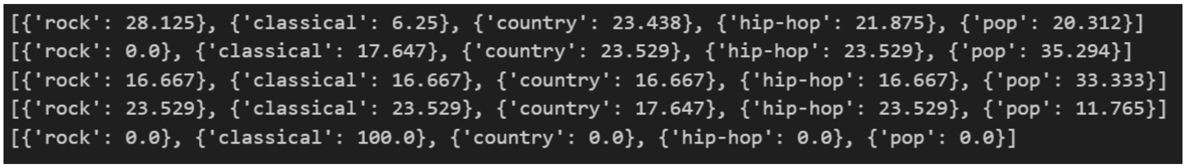


Figure 5: The final plot

7.1 Percentage Distribution in each cluster

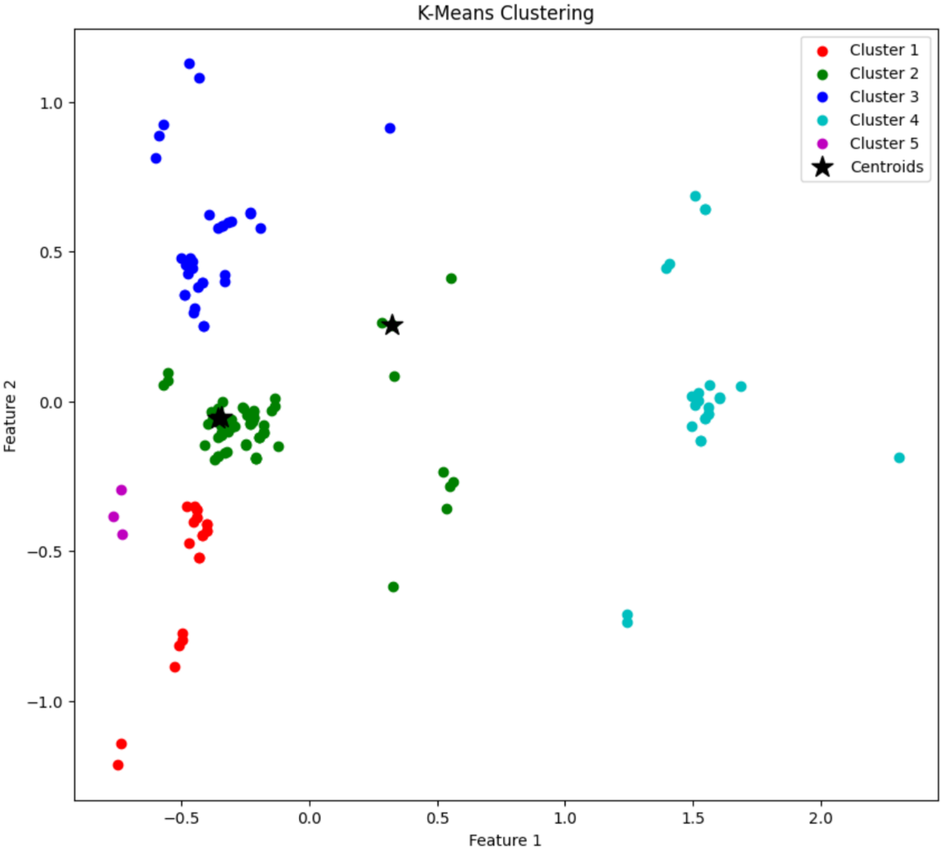


```
[{'rock': 28.125}, {'classical': 6.25}, {'country': 23.438}, {'hip-hop': 21.875}, {'pop': 20.312}]
[{'rock': 0.0}, {'classical': 17.647}, {'country': 23.529}, {'hip-hop': 23.529}, {'pop': 35.294}]
[{'rock': 16.667}, {'classical': 16.667}, {'country': 16.667}, {'hip-hop': 16.667}, {'pop': 33.333}]
[{'rock': 23.529}, {'classical': 23.529}, {'country': 17.647}, {'hip-hop': 23.529}, {'pop': 11.765}]
[{'rock': 0.0}, {'classical': 100.0}, {'country': 0.0}, {'hip-hop': 0.0}, {'pop': 0.0}]
```

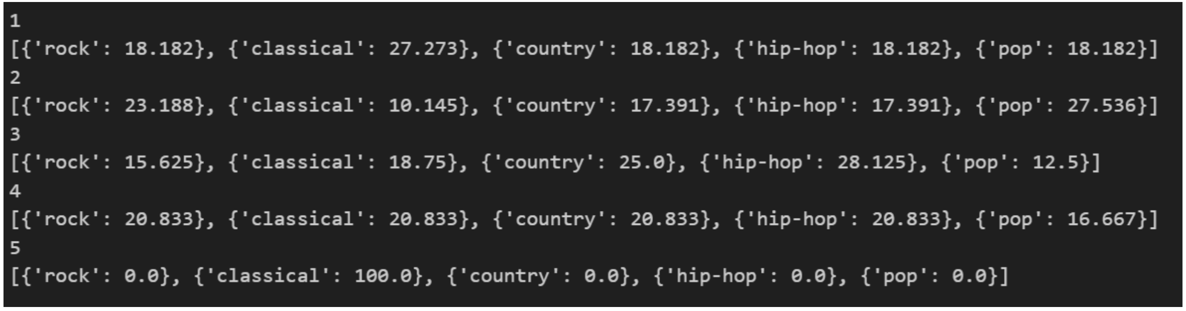
This is the percent distribution for one of the instances of k-mean cluster that I ran. Here, most clusters are uniformly filled but the last cluster has 100 percent classical songs in it. This tells us that the algorithm has found a way to identify a fraction of classical songs as a cluster. Though this is not full proof identification, as classical songs are also present in other groups. This also tells us that the model couldnt really find any other identification for any other genre, which is concerning.

7.2 Predicting genre

The task demands us to predict the genre of these three songs, [piano, calm, slow], [guitar, emotional, distorted], and [synth, mellow, distorted]. I added these to the points and plotted them, and this



was the result. The three points representing the songs are labelled centroids. interestingly, they are in the same cluster. Now lets see the Probabilty distribution inside these clusters.



```
1
[{'rock': 18.182}, {'classical': 27.273}, {'country': 18.182}, {'hip-hop': 18.182}, {'pop': 18.182}]
2
[{'rock': 23.188}, {'classical': 10.145}, {'country': 17.391}, {'hip-hop': 17.391}, {'pop': 27.536}]
3
[{'rock': 15.625}, {'classical': 18.75}, {'country': 25.0}, {'hip-hop': 28.125}, {'pop': 12.5}]
4
[{'rock': 20.833}, {'classical': 20.833}, {'country': 20.833}, {'hip-hop': 20.833}, {'pop': 16.667}]
5
[{'rock': 0.0}, {'classical': 100.0}, {'country': 0.0}, {'hip-hop': 0.0}, {'pop': 0.0}]
```

All 3 are in cluster 2, and the cluster 2 has majorly rock and pop songs, combined occupying more than 50 percent of the cluster. So i would predict that all 3 of these songs are either rock or pop.

8 Conclusion

In this Task, I explored various techniques for processing and analyzing textual data in the context of song classification by genre. I began by converting textual data into numerical vectors using the TF-IDF method, which effectively captured the importance of words within the Datasetet. To address the challenge of high-dimensional data, I applied Principal Component Analysis (PCA) for dimensionality reduction, enabling better visualization and interpretability of the dataset.

For clustering, I opted for the K-means algorithm, as it allowed for predefining the number of clusters, aligning well with the five genres in the Dataset. The silhouette score was used to assess the clustering performance, confirming the effectiveness of the chosen approach.