Music Genre Analysis

Using clustering

By:

Aaryana Dutta

INDEX

Introduction

In this task, we aim to classify songs on the basis of textual data in the form of keywords and predict its most likely genre. This paper presents a methodology to use a clustering algorithm to classify songs solely using these keywords.

About the dataset:

The dataset consists of 147 songs. Each song has three keywords associated with it, corresponding to the instrument used, the mood of the song and its style. The final column represents the 'ground truth genre' of each song, i.e., the actual genre it belongs to. It has the following categories: Hip hop, classical, country, pop and rock.

Workflow implemented in this task:

1. Creating a 'dataframe' of the dataset
2. Vectorization using Bag Of Words
3. Dimensionality reduction using Principal Component Analysis
4. Combining the embeddings
5. Clustering
6. Analysis of the results

Prerequisites

All the necessary concepts required for comprehension and effective implementation of the task solution will be discussed below:

Vectorization of words

Vectorization is the process of converting textual data into numerical representations. This is crucial because it converts words into a format that computers can understand and allows machine learning algorithms to process text data effectively. These assigned vectors are not completely random. Similar words tend to have similar vectors, which allows the computer to understand the relationships between different words based on how ‘close’ or ‘far-away’ they are in the vector space.

The two main vectorization techniques considered in this task statement are described below.

Bag of Words

It is a simple and flexible text vectorization technique that captures the occurrences of a word in a document.

It consists of two key components:

1. A predefined vocabulary of words.
2. A way to measure the presence of these words in a document.

The term "bag" of words refers to the fact that this approach ignores the order and structure of words. Instead, it focuses solely on identifying which known words appear in the document, without considering their specific positions.

An example is given in the table below:

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

TF-IDF Vectorization

It is a statistical metric used to determine the how important a word is within a document compared to its overall occurrence across a collection of documents.

TF-IDF combines two components: Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency (TF) quantifies the number of times a word occurs within a document. A higher frequency generally indicates greater significance. If a term appears frequently in a document, it is likely to be essential to the document's

topic. The formula is:

$$TF(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

However certain common words such as ‘the’, ‘and’, ‘or’, etc may have a high frequency leading but might not be important to the document’s topic at all. This is where IDF comes in.

Inverse Document Frequency (IDF) adjusts the significance of words by decreasing the weight of commonly found words across multiple documents while increasing the importance of rare terms. A word that appears in fewer documents is considered more ‘important’. The formula is:

$$IDF(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

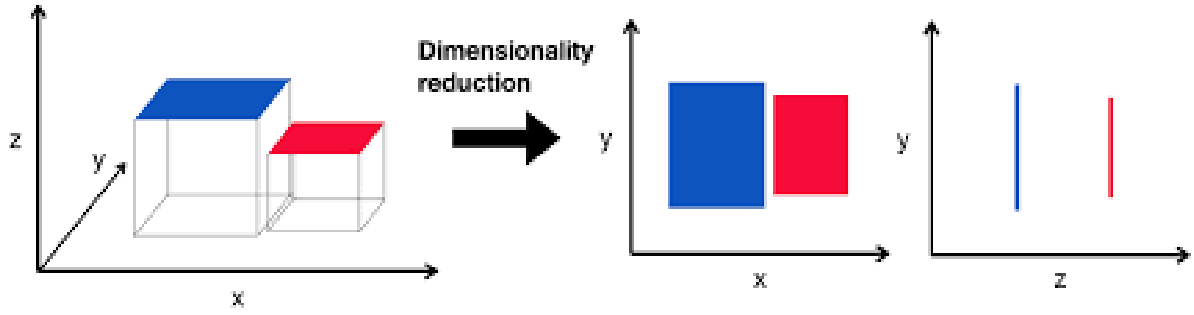
A combination tf and idf is used to to determine the weight of a word in the document.

Dimensionality reduction

It is the process of transforming data from a high dimension space to a lower dimension space, while ensuring that this lower dimension space retains meaningful information about the original data.

Why is it necessary? It allows us to decrease the complexity of the data that we work with by reducing the number of features. This can help to prevent problems such as high computation time and overfitting.

How does it work?



In the first image, the data exists in a three-dimensional space. However, all three dimensions are not equally important. These dimensions which are ‘less important’ can be removed without losing any valuable data. Thus, ultimately, we end up in a one-dimensional space, which consists of the most significant feature, thereby making the data much easier to work with.

In the task, PCA has been used for dimensionality reduction.

Principal Component Analysis (PCA)

PCA works based on the following algorithm

1.**Standardize the Data** – Scale all features to have a mean of 0 and standard deviation of 1.

2.**Compute Covariance Matrix** – Measures how features vary together (positive, negative, or no correlation).

3.**Find Principal Components** – Identify new axes (PCs) that maximize variance using Eigenvalues and Eigenvectors.

Eigenvalues and eigenvectors the eigenvalues of matrix are scalars by which some vectors (eigenvectors) change when the matrix (transformation) is applied to it.

4.**Select Top PCs & Transform Data** – Keep the most significant principal components and project data onto them for dimensionality reduction.

Unsupervised Learning

Unsupervised learning is a machine learning approach where algorithms identify patterns in data without relying on labelled data.

Clustering

Clustering is an unsupervised machine learning algorithm that organizes and classifies different objects, data points, or observations into groups or clusters based on similarities or patterns. The goal is to ensure that items within the same

cluster are more similar to each other than to those in different clusters.

K means clustering

Algorithm :

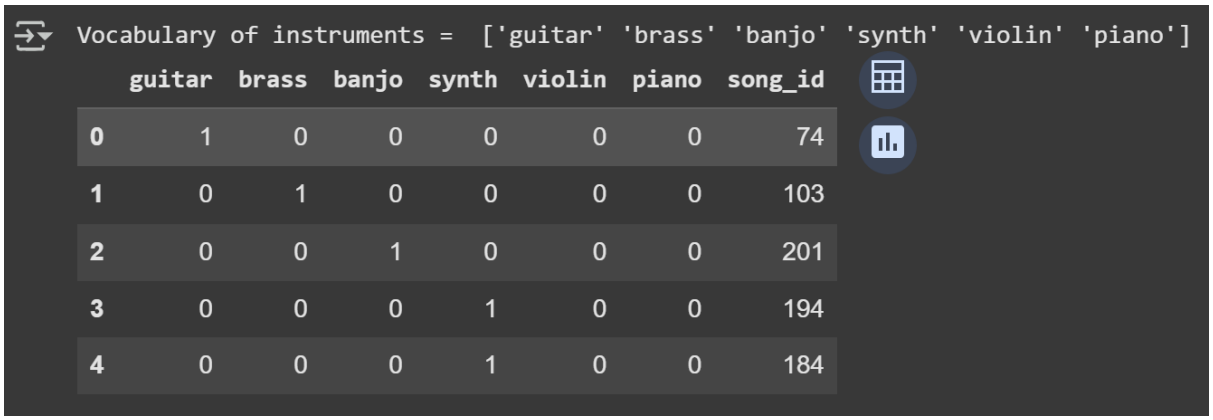
1. Decide the number of clusters, k.
2. Randomly select k points as initial cluster centres.
3. Each data point is assigned to the nearest centroid based on distance.
4. Compute the new centroid for each cluster as the mean of all assigned points.
5. Repeat until centroids don’t change much or max iterations have been done.

Methodology

Elaborated below are the steps that were undertaken to come up with an effective solution to the task statement.

Vectorization using bag of words

For each song, a separate matrix was created corresponding to the bag of words representation for instrument, style and mood. In this representation, the unique keywords (which make up the vocabulary) are taken as the columns. A 1 in the matrix represents that the word has occurred whereas a 0 represents that it has not. An example is given below:



```
Vocabulary of instruments =  ['guitar' 'brass' 'banjo' 'synth' 'violin' 'piano']
```

	guitar	brass	banjo	synth	violin	piano	song_id
0	1	0	0	0	0	0	74
1	0	1	0	0	0	0	103
2	0	0	1	0	0	0	201
3	0	0	0	1	0	0	194
4	0	0	0	1	0	0	184

Justification for choosing bag of words over tf-idf:

The main difference between bag of words and tf-idf is that the former simply counts the frequency of words in a document whereas the latter also focuses on the importance

of each word in the document and ignores stop words such as ‘the’, ‘or’, etc.

Thus bag of words is more efficient in our task because

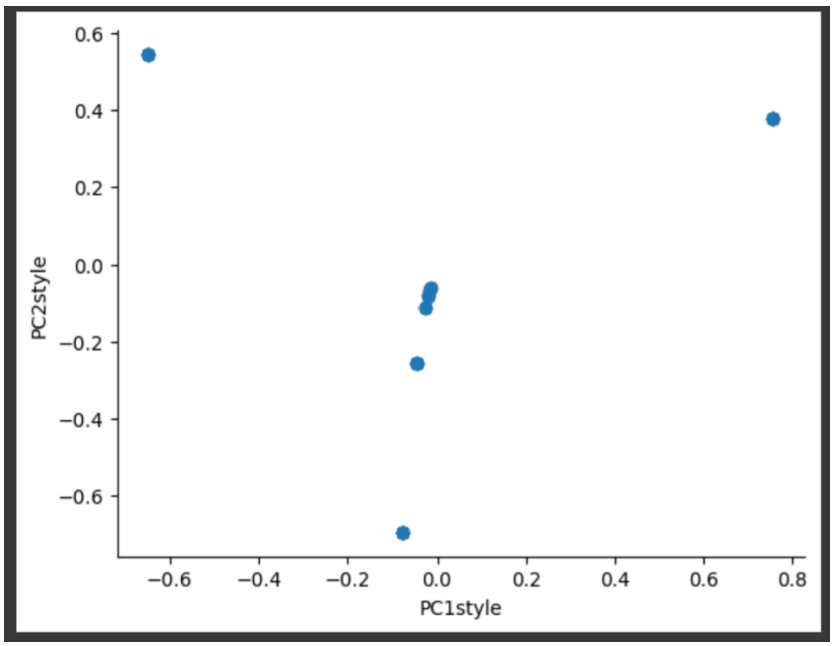
1. Keywords are of equal importance and thus there is no need for weighing certain words more than others.
2. There are no stop words present in the dataset.
3. BoW is much easier to implement and this simple frequency based representation is sufficient to capture the keyword presence.

Thus, since we have **limited keyword data**, TF-IDF's main advantage (reducing the weight of common words) doesn’t apply, making BoW the more efficient choice.

Dimensionality reduction using PCA

In this step, we applied Principal Component Analysis (PCA) to reduce the dimensionality of our song embeddings while preserving as much important information as possible. The algorithm has been discussed above.

This dimensionality reduction was necessary because each vector had a dimension equal to the number of keywords in the vocabulary. Using PCA, we transformed this into a two-dimensional space for easier and effective analysis. An example is given below



Combining the embeddings

In our analysis, we initially had **three separate embeddings** for each song, representing different aspects. Thus, each song had a very segmented representation. Combining the embeddings was necessary to create a **meaningful, and computationally efficient representation** of each song.

Justification of the technique used:

'Averaging' was the technique used for combining because of the following reasons:

1. It provides a straightforward way to combine embeddings without introducing any unnecessary complexity.

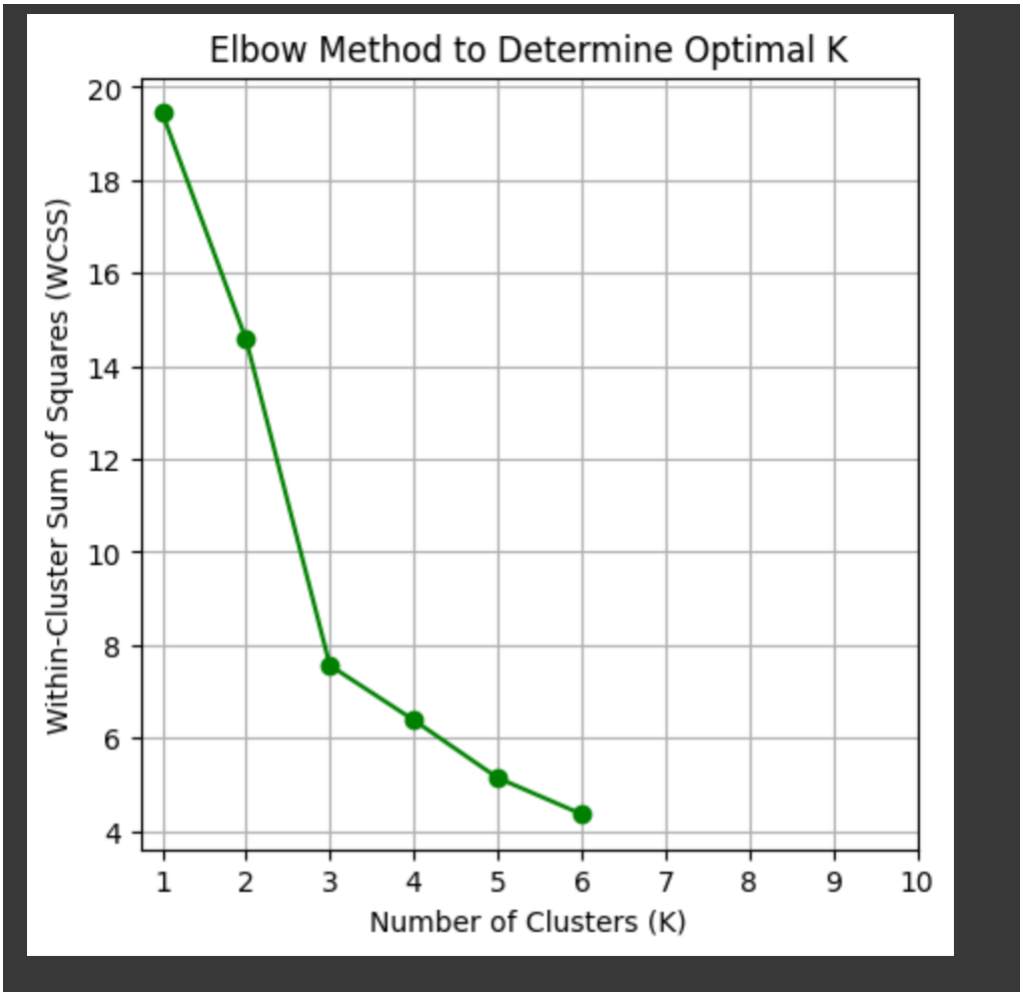
2. Since all three embeddings are **equally important**, averaging ensures that no single aspect is weighed more.
3. Techniques like concatenation would increase the dimension.
4. Faster and more computationally efficient.

Clustering

In this task, we implemented K-Means Clustering from scratch using only Pandas and NumPy. This algorithm was run multiple times with different values of k to finally end up with the most optimal value of k.

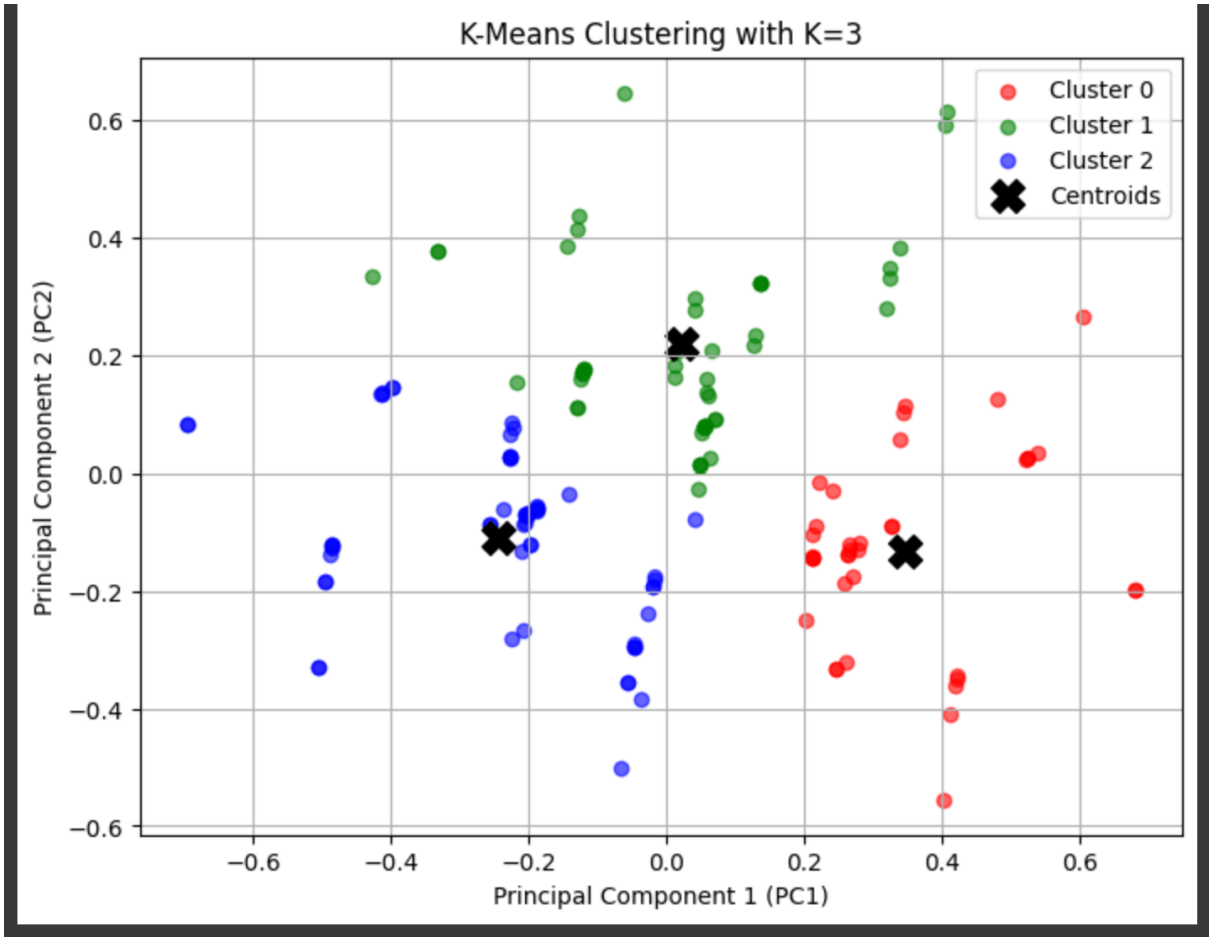
To check for this optimal value, we used a metric known as WCSS. Within Cluster Sum of Squares (WCSS) is a measure of how compact the clusters are. We plotted WCSS vs. K on a graph called the elbow curve. The Elbow Curve helps us find the best balance between small WCSS and 'not too many clusters.' The point in the graph which bends sharply like an

elbow gives us the value of the optimal k.



Thus, based on the graph, the optimal value of k obtained was **three.**

Using this value of k, we performed k means clustering on the combined embedding to get our final clustering result.



Observation of obtained clustering results:

The clusters appear to be reasonably well-separated, suggesting that the algorithm successfully identified distinct groups in the data.

However, there are some overlapping points between clusters, especially around the centre, which might indicate some similarities between different clusters.

Justification of using clustering:

1. K-Means is an unsupervised learning algorithm, making it ideal for discovering natural clusters in the dataset without prior knowledge of song genres.

2. K means works well with numerical data when we have to cluster points based on Euclidean distance.
3. It is computationally efficient.
4. It is relatively easy to implement.
5. The produces clusters are easy to interpret and visualize.

Thus, this algorithm is best suited for clustering our structured dataset.

Analysis

Percentage distribution of true genres in each cluster

This is necessary in order to understand the distribution of ground truth genres in each cluster. If each cluster contains a high percentage of a **single genre**, it suggests that the clustering model has effectively separated songs based on their true genres.

Our results:

Cluster	true genre	
0	hip-hop	57.894737
	pop	26.315789
	classical	7.894737
	country	5.263158
	rock	2.631579
1	classical	41.176471
	pop	21.568627
	country	15.686275
	rock	15.686275
	hip-hop	5.882353
2	rock	36.206897
	country	32.758621
	pop	17.241379
	hip-hop	8.620690
	classical	5.172414

In cluster 0, there is a strong dominance of hip-hop. Similarly, there is a strong dominance of classical in cluster 1. However, in cluster 2, the percentage distribution of rock and country is approximately the same. This suggests that there is some degree of overlap in this cluster.

Silhouette Score

What is it? It is a metric that helps us to assess the quality of clustering. The silhouette value quantifies how well an object fits within its assigned cluster (cohesion) relative to its separation from other clusters. It ranges from -1 to +1, with higher values indicating a strong association with its own cluster and weak similarity to neighbouring clusters.

Why is it necessary? Since clustering is an unsupervised learning task, there are no predefined labels to assess the accuracy of the generated clusters. Thus, evaluating the performance of our clustering algorithm requires internal evaluation metrics like silhouette score.

Formula:

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

Where ‘a’ is the average distance between a sample and other data points in the same cluster, and ‘b’ is the average distance between a sample and other data points in the nearest cluster.

Our results:

	song id	PC1	PC2	Cluster	true genre	silhouette score
0	74	-0.485078	-0.119380	2	rock	0.520042
1	103	-0.062615	0.647452	1	classical	0.401733
2	201	-0.223757	0.075885	2	country	0.220657
3	194	0.345979	0.114004	0	hip-hop	0.243539
4	184	0.326985	-0.090043	0	hip-hop	0.600320

```
Average Silhouette Score: 0.4239373889317584
```

The silhouette score was computed for each data point and finally an average silhouette score was obtained to judge the quality of our generated cluster.

What does this mean?

An average silhouette score of approximately 0.42 suggests moderate clustering quality. The clusters are separated but they are not perfectly distinct. Moreover, there are some genre overlaps because certain genres share similar features. A reason for this overlap could be that the reduced dimension space of the dataset might not be able to adequately distinguish between such genres.

Assigning genres to a new song based on keywords

A fairly intuitive technique was developed to assign a genre to a new song based on its keywords. The already generated clusters were used for this purpose

Methodology:

1. First, we find the cluster that it "fits into" the best. This could be the cluster from which its Euclidean distance is minimum.

2. Now that we have assigned a cluster, to assign a genre we find the genre having the **highest frequency** in **that** cluster.

Results obtained:

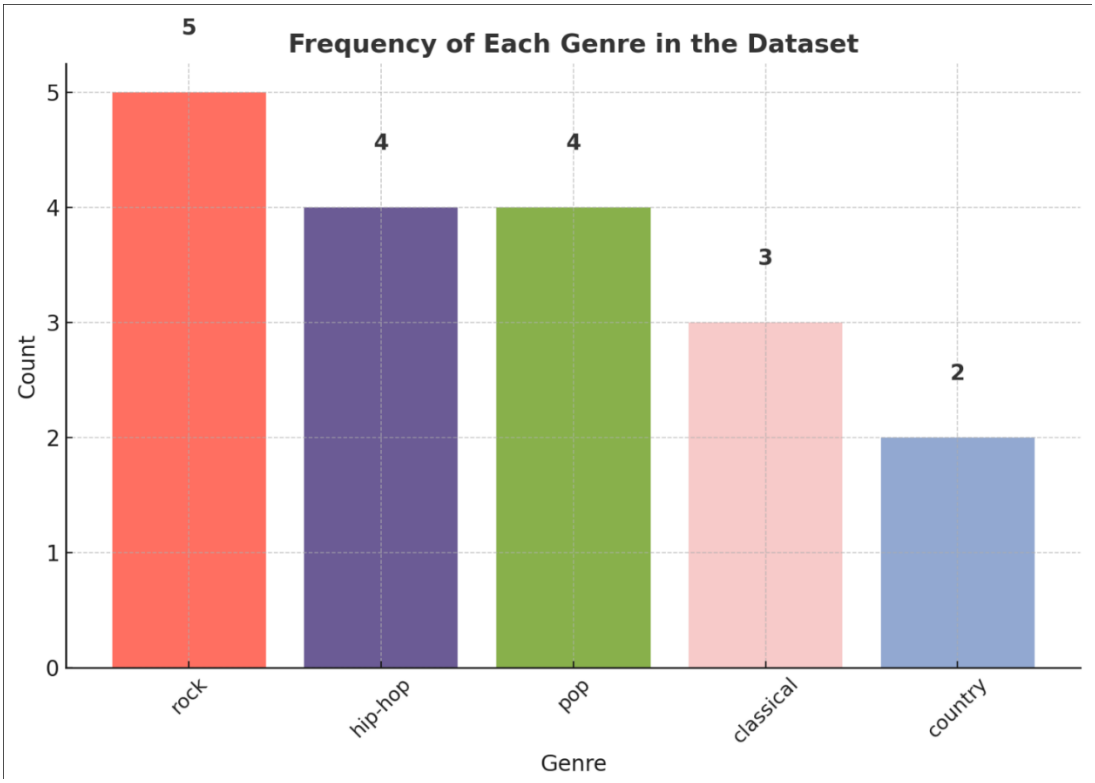
```
['classical', 'hip-hop', 'rock']
```

	predicted genre	Instrument	Mood	Style
new_0	classical	piano	calm	slow
new_1	hip-hop	guitar	emotional	distorted
new_2	rock	synth	mellow	distorted

Bonus

Analysis of the dataset

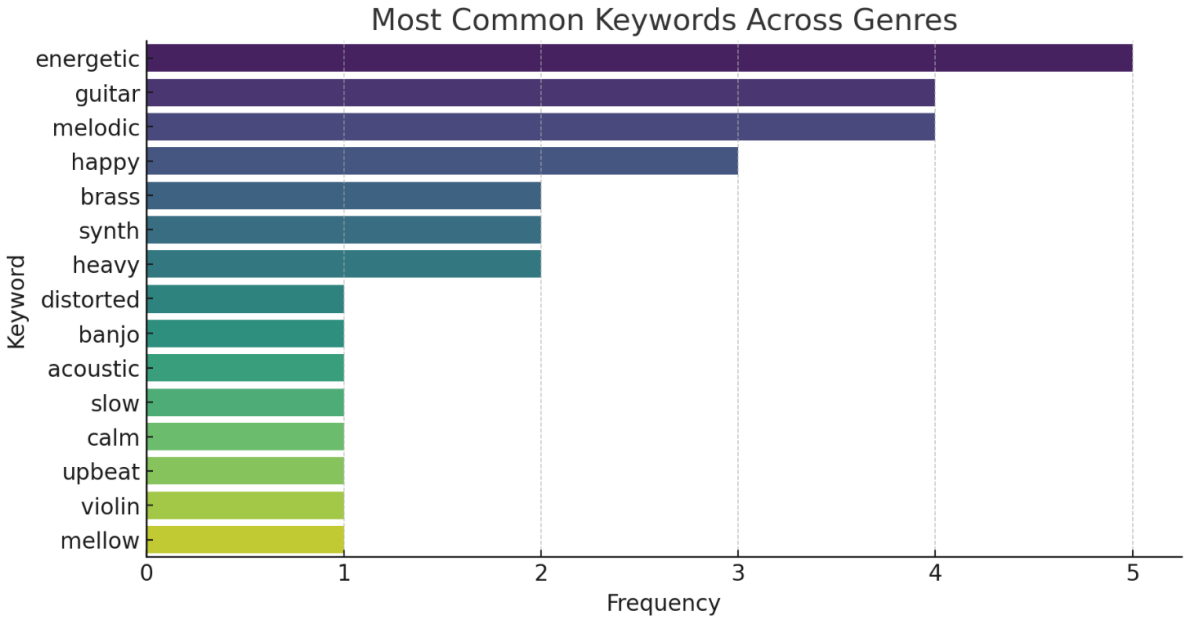
Frequency of each genre in the dataset:



Analysis of the relationship of keywords and true genres

Certain genres share some keywords and this could also lead to the overlap observed in our clustering results. Eg: Pop & Rock share "energetic" and "melodic" keywords. Hip-Hop & Synth-Pop share "synth" and "rhythmic" keywords.

Analysis of the most common keywords:



Extrinsic metrics

Extrinsic metrics require ground truth labels. They compare the result obtained from the clustering algorithm to what the result actually should be based (its true genre).

Some proposed extrinsic metrics for this task:

1. ADJUSTED RAND INDEX (ARI)
   It ranges from -1 to 1 and measures the similarity between predicted clusters and true genres.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

   n is the overlap between predicted and true labels.

2. NORMALIZED MUTUAL INFORMATION (NMI)
   It measures the mutual correlation between predicted clusters and true labels. It ranges from 0 (no correlation) to 1 (perfect correlation).

Normalized Mutual Information

- Normalized Mutual Information:

$$NMI(Y, C) = \frac{2 \times I(Y; C)}{[H(Y) + H(C)]}$$

where,

1) Y = class labels
2) C = cluster labels
3) H(.) = Entropy
4) I(Y;C) = Mutual Information b/w Y and C

Note: All logs are base-2.

Conclusion

In this task we delved into the application of k means, an unsupervised clustering algorithm for assigning genres to songs based on their keywords. At every step of the way, we compared methods and justified why the specific method used is the most efficient in the context of this task. This comparative analysis helped us to come up with the most optimized solution. Moreover, we also checked the quality of our clustering using intrinsic metrics like silhouette scores.

Sources:

- https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/
- https://www.geeksforgeeks.org/silhouette-algorithm-to-determine-the-optimal-value-of-k/
- https://medium.com/@tiami.abiola/clustering-wcss-and-elbow-method-427db8968ba1
- https://www.geeksforgeeks.org/bag-of-words-vs-tf-idf/
- https://www.techtarget.com/searchenterpriseai/definition/clustering-in-machine-learning
- https://www.geeksforgeeks.org/dimensionality-reduction/
- https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/
- https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d/
- https://machinelearningmastery.com/gentle-introduction-bag-words-model/
- https://www.geeksforgeeks.org/clustering-metrics/
- https://www.youtube.com/watch?v=FgakZw6K1QQ
- https://www.youtube.com/watch?v=4b5d3muPQmA

- https://www.youtube.com/watch?v=DG8XP1Ac0_g