**KDAG TASK 2**

# MUSIC GENRE ANALYSIS

*By – Arnav Gawade 24CY10010*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Index :-**

1. Abstract
2. Introduction
3. Methodology
   1. Data Preprocessing
      1. Extracting Unique Keywords
   2. Vectorization Methods

3.2.1. Bag of Words (BoW)

3.2.2. TF-IDF

3.2.3. Binary Encoding (Bonus)

* 1. Clustering Algorithm: k-Means

3.3.1. Definition of k-Means

3.3.2. Steps of applying k-Means

3.3.3. Selection of value of k

* 1. Embedding Combination Strategy
     1. Types of Embedding Strategy we can use
     2. Reason for selecting Term Frequency Embedding method
  2. Dimensionality Reduction

3.5.1. PCA Application

3.6 Evaluation Metrics

3.6.1. Intrinsic Methods

3.6.2. Extrinsic Methods

* 1. Exploratory Data Analysis

3.7.1. Plots and Charts used

1. Result and Analysis

4.1. Comparision of TF-IDF and Bow

4.2. Choice of ‘K’

4.3. Reason for selecting the Term Frequency Embedding method

4.4. Binary Encoding vs other 2 vectorization methods

4.5. Analysis of Plots

4.6. Analysis of Intrinsic and Extrinsic indices

* 1. Assigning Genre to Keywords

1. Conclusion
2. References

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Abstract**

This study explores the effectiveness of different text vectorization techniques—**Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Binary Encoding**—for clustering music tracks based on keyword metadata. Using **k-means clustering**, we analyse the impact of these vectorization methods on clustering performance, evaluating results with **Silhouette Score** and some other intrinsic, extrinsic indices. We further investigate the influence of **different values of k**, justify the choice of **embedding combination method** and assess **genre distributions across clusters**. Additionally, we assess various plots provide further insight into feature separation and cluster composition and end results.

## **Introduction**

Music genre analysis is a fundamental task in various music industries having various applications such as music recommendation systems, playlist generation, etc. Traditionally, genres are assigned based on expert advice or data provided by artists and record labels. However, with the increasing availability of large-scale music datasets, **unsupervised learning techniques**, particularly **clustering**, offer a promising alternative for automatic music categorization. In this study, we explore **k-means clustering** as an unsupervised approach to grouping music tracks based on their data. Since traditional audio-based clustering is computationally expensive and requires extensive feature engineering, we focus on **text-based clustering** by extracting information from song data, specifically **keywords describing the music tracks**. These keywords capture essential stylistic and thematic aspects of each song, forming the basis for clustering. To transform the text-based data into numerical representations suitable for clustering, we employ three widely used **vectorization techniques**:

* **Bag of Words (BoW)** – Represents text as frequency-based feature vectors.
* **Term Frequency-Inverse Document Frequency (TF-IDF)** – Assigns importance weights to words based on their frequency in individual documents versus the entire dataset.
* **Binary Encoding** – Encodes the presence or absence of words in a track’s metadata without considering frequency(BONUS).

After applying these vectorization methods, we use **k-means clustering** to group similar tracks together and evaluate the clustering quality using **intrinsic and extrinsic validation metrics**, namely **Silhouette Score, Davies-Bouldin Index, and Adjusted Rand Index (ARI)**. These metrics provide insights into how well-defined the clusters are and how they align with actual genres. Additionally, we also examine **genre distributions across clusters**, **scatter plots of clustered data**, and **visualizations such as PCA pair plots, frequency occurrence plots, and cluster size pie charts**. This allows us to interpret the clustering results meaningfully. Moreover, we propose a simple but effective **genre assignment technique** that leverages keyword-based classification. Using this approach, we assign genres to new songs based on the presence of specific keywords(BONUS).

Through this report, we aim to address key research questions, including:

1. **Which vectorization method produces the best clustering results?**
2. **How does the choice of k affect clustering performance?**
3. **Does binary encoding perform well against BoW and TF-IDF?**
4. **What insights can be drawn from the clustering results, and how well do clusters align with actual genres?**

## **Methodology**

The methodology section provides a **step-by-step breakdown** of how we processed the data, applied different vectorization techniques, implemented clustering, and evaluated the quality of the clusters. This structured approach ensures the reliability and interpretability of our results.

* 1. **Data Preprocessing**

Data preprocessing is a crucial step in ensuring that the input data is structured correctly before applying clustering algorithms. The dataset used in this study comprises **song data** in the form of descriptive **keywords** associated with each track. Since these keywords are textual in nature, they need to be transformed into a suitable numerical representation.

The **key preprocessing steps** applied are as follows:

* + 1. Extracting Unique Words
* We created a **vocabulary of unique keywords** present across all songs.
* This vocabulary was used to construct vector representations in **BoW, TF-IDF, and Binary Encoding** formats.
* *unique\_keywords = np.unique(keywords.values) -* This was used for the above case.

There was not a need to standardize the text data as the entire data was in lower case. This would have to be implemented if both upper case and lower case characters were present. For example- ‘Rock’ and ‘rock’ would get treated separately instead of a single entity.

The entire data set was filled so there was also no need to handle the missing values.

* 1. **Vectorization Methods**

Vectorization in machine learning refers to the process of converting data—such as text, images, or categorical variables—into numerical representations (vectors) that can be processed by machine learning models.

Once the text data was preprocessed, it needed to be transformed into numerical representations so that clustering algorithms could process it.

We evaluated **three vectorization techniques**:-

* + 1. Bag of Words(BoW)
* **Definition:**

Bag of Words (BoW) is a simple yet effective text representation technique that creates a vector for each document (song) based on word occurrences. It **does not** consider the order of words but focuses solely on the **frequency** of each word in a document.

* **Implementation:**

A **vocabulary** of unique keywords was created. Each song was represented as a vector where each element in the vector corresponded to a word in the vocabulary.

If a word appeared in the song’s metadata, its corresponding index in the vector was increased by 1.

* **Advantages:**
  1. Simple and computationally efficient.
  2. Works well when keywords are **distinctive** for different genres.
* **Limitations:**
  1. High dimensionality due to the large number of unique keywords.
  2. Does not consider word importance (e.g., common words may dominate representation).
     1. TF-IDF
* **Definition:**

TF-IDF improves upon BoW by weighting words based on their **importance** in the dataset. The **Term Frequency (TF)** measures how often a word appears in a document. The **Inverse Document Frequency (IDF)** reduces the impact of words that appear too frequently across many documents, making them less informative.

* **Implementation:**

Constructed a TF-IDF matrix where each row represented a song, and each column represented a keyword. I applied **log scaling** to reduce the dominance of extremely frequent words.

* **Advantages:**
  + 1. Gives **higher importance** to unique words that define genres
    2. Reduces the impact of overly common words that appear in most tracks.
* **Limitations:**

1. Still results in a sparse matrix.
2. Computationally more expensive than BoW.
   * 1. Binary Encoding

* **Definition:**

Unlike BoW and TF-IDF, Binary Encoding **only considers word presence or absence** rather than frequency. Each song is represented as a binary vector:

* + If a keyword appears in a song, the corresponding vector index is **1**.
  + If it does not appear, the index is **0**.
* **Implementation:**

Created a **binary matrix**, where each row represented a song and each column represented a keyword. Each entry in the matrix was either 0 (word absent) or 1 (word present).

* **Advantages:**

1. Reduces the **effect of word frequency bias**, ensuring that rare but meaningful words are not overshadowed.
2. Lowers the memory footprint compared to BoW.

* **Limitations:**
  + 1. **Ignores frequency information**, which may be crucial for distinguishing songs.
    2. Can lead to information loss, affecting clustering performance.
  1. **Clustering Algorithm: k-Means**

Clustering is an **unsupervised learning** technique used to group similar data points. together based on their features. It helps in identifying patterns and structures in data without predefined labels.

After vectorizing the text data, we applied **k-means clustering** to group songs into clusters based on feature similarity.

* + 1. **Definition of k-Means Clustering**

K-Means is a **centroid-based, unsupervised clustering algorithm** that partitions a dataset into **K distinct clusters** based on feature similarity. It minimizes the variance within each cluster by iteratively adjusting cluster centers (centroids). It works iteratively to assign each data point to the **nearest cluster centroid**.

* + 1. **Steps for Applying k-Means Clustering**
* **Choosing k (number of clusters):** The optimal k was chosen based on **Silhouette Score** and **Davies-Bouldin Index**.
* **Initialization of Cluster Centroids:** k centroids were randomly placed in the vector space.
* **Cluster Assignment:** Each song was assigned to the closest centroid based on **Euclidean Distance**.
* **Updating Centroids:** The new centroid positions were calculated based on the **average feature vector** of all points assigned to a cluster.
* **Repeat Until Convergence:** Steps 3 and 4 were repeated until **centroids stopped changing significantly**.
  + 1. **Selection of value of k**

We selected the best k value from the results we got from **Silhouette Score** and **Daves-Bouldin Index**.

I applied another method to find the best k value i.e. the **Elbow Plot Method**

* **Elbow Plot:**

**-Definition:**

An elbow plot is a graphical method used to determine the optimal number of clusters (*K*) in K-Means clustering. It plots the inertia (sum of squared distances from points to their cluster center) against different values of *K*.

* **How to Use an Elbow Plot?**

1. Run K-Means for multiple values of *K* (e.g., 1 to 10).
2. Compute Inertia (Within-Cluster Sum of Squares - WCSS) for each *K*.
3. Plot K vs. Inertia—the curve typically decreases as *K* increases.
4. Find the "Elbow Point"—the point where the decrease in inertia slows down significantly.
5. Choose K at the Elbow—this is the optimal number of clusters.
   1. **Embedding Combinations strategy**
      1. **Types of Embedding Strategies we can use**

In machine learning and natural language processing (NLP), an **embedding combination strategy** refers to how multiple embeddings (e.g., word embeddings, sentence embeddings) are combined to create a **single vector representation** of a document, paragraph, or any higher-level structure.

**Common Embedding Combination Strategies:-**

**1)Averaging (Mean Pooling)**

* **Definition:** Averaging technique computes the element-wise mean of all word embeddings in a sentence/document.
* **Pros:**

1. Simple and computationally efficient.
2. Works well when all words contribute equally.

* **Cons:**
  + Loses word order and ignores important context.

**2)Cross product of embedding**

* **Definition :-** The Cross Product of Embeddings is a technique where two or more embedding vectors are multiplied element-wise to generate a new vector.
* **Pros:** 
  + - 1. Useful in scenarios where different features **interact non-linearly** (e.g., user-item relationships in recommender systems).
      2. Works well for **pairwise comparisons** (e.g., similarity between two word embeddings).
* **Cons:** 
  + - 1. Since the cross product **multiplies elements**, it **destroys individual feature interpretability**.
      2. May not work well in contexts where features should be **combined additively** (e.g., simple word embeddings where averaging is better).
    1. **Reason for selecting Term Frequency Method**

A **Term Frequency (TF) Matrix** is a numerical representation of text data, where each row corresponds to a document (or song in your case), and each column represents a unique keyword. The values in the matrix represent how often each keyword appears in each document.

* A **Term Frequency Matrix (TFM)** is an **embedding technique** where text data (keywords, lyrics, etc.) is converted into a structured numerical format.
* For a dataset with **3** songs and **3** unique keywords, the **TF matrix** looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| * **Song ID** | * **Keyword 1** | * **Keyword 2** | * **Keyword 3** |
| * **Song 1** | * 1 | * 0 | * 2 |
| * **Song 2** | * 0 | * 3 | * 1 |
| * **Song 3** | * 2 | * 1 | * 0 |

* **Advantages**

- **Simple & Efficient** – Easy to compute and interpret.

- **Captures Word Frequency** – Useful for keyword-based music clustering.

- **Works Well with Clustering**

**It gives a range of silhouette score, but it is greater than binary encoding and TF-IDF when used with bow.**

* 1. **Dimensionality Reduction**

**Dimensionality Reduction** is the process of **reducing the number of features (dimensions) in a dataset** while preserving its important patterns. It is widely used in **machine learning, deep learning, and data visualization** to simplify complex data without losing significant information. As the number of dimensions increases, data becomes **sparse**, making it harder for machine learning models to learn patterns. Thus this comes handy.

We used PCA for dimensionality reduction in this case

* + 1. **PCA Application**

PCA transforms high-dimensional data into a new coordinate system where:

* **Principal Component 1 (PC1):** Captures the direction of maximum variance in the data.
* **Principal Component 2 (PC2):** Captures the second most significant variance, orthogonal to PC1.

**Higher-order components** capture less variance and are discarded. The result is a **compressed representation** where the most important patterns remain.

**PCA** is implemented in the code in the following way:-

1. Computing the **mean** of all keyword vectors. This is used to **center** the data by subtracting the mean.
2. **Shifting the data** so that it is centered around zero. Essential for correct PCA calculations.
3. Computing the **covariance matrix**, which shows how features vary together.The covariance matrix helps find **correlations** between keywords.
4. Computing the Eigen values and Eigen vectors.
5. **Sorts eigenvalues in descending order** to keep the most important features.
6. **Finally**  projects data onto the **new PCA axes**.

* 1. **Evaluation Metrics**

Unlike supervised learning, where accuracy and precision can be directly measured using labeled data, clustering evaluation requires different metrics since there are often **no true labels**. Clustering evaluation is divided into two main categories:

1. **Intrinsic Evaluation Metrics** (Evaluate clustering quality without ground truth)
2. **Extrinsic Evaluation Metrics** (Compare clustering results with ground truth labels)
   * 1. **Intrinsic Evaluation Metrics**

These metrics assess clustering **without knowing the true labels** by analyzing **cohesion (compactness)** and **separation (distinction between clusters)**.

Some that we **used are:-**

1. **Silhouette Score :-** The Silhouette Score is a metric used to evaluate the quality of clustering. It measures how well-separated and cohesive the clusters are. A higher silhouette score indicates better clustering. It’s range lies between -1 and 1, -1 indicating wrong clustering and 1 indicating perfect clustering.
2. **Dave-Bouldin Score:-** The Davies-Bouldin Index (DBI) is a clustering evaluation metric that measures how well-separated and compact clusters are. A lower Davies-Bouldin score indicates better clustering quality.

Interpretation of DBI Score:

Lower DBI → Better clustering(clusters are well-separated and compact).

Higher DBI → Poor clustering (overlapping clusters or spread-out clusters).

* + 1. **Extrinsic Evaluation Metrics**

Extrinsic evaluation methods assess clustering **based on external ground truth labels** (e.g., actual genres for songs). These metrics help determine **how well clusters align with true categories**.

The one I have used is the Adjusted Rand Index.

* 1. **Adjusted Rand Index:-** It measures similarity between predicted clusters and true labels Accounts for chance grouping, making it more robust. If it gives 1 then it means perfect clustering. If it gives 0 then it means clustering is random.
  2. **Exploratory Data Analysis**
     1. **Plots and charts used**

To interpret the results, we used several visualization techniques:

* **Distribution of songs across clusters:** Helps in identifying if clusters are balanced.
* **Genre distribution within clusters:** Examines if certain clusters are genre-dominant.
* **Scatter plot of clusters:** Provides a visual representation of how distinct the clusters are.
* **PCA Pairplot:** Shows feature contribution to clustering.
* **Frequency occurrence plot:** Identifies the most commonly used keywords in the dataset.
* **Cluster size pie chart:** Shows data distribution across different clusters.

## **Result and Analysis**

* 1. **TF-IDF vs BoW Vectorization**

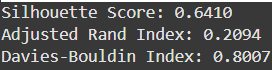
BoW outperformed both binary encoding and TF-IDF in terms of the Silhouette Score, indicating better intra-cluster cohesion and separation. This suggests that BoW’s simpler frequency-based representation effectively captured genre-related patterns without overemphasizing rare words, unlike TF-IDF.

**Some reasons could be:-**

1. TF-IDF downweights frequently occurring words, even if they are important for genre distinction. BoW **preserves all words equally**, allowing genre-specific terms to dominate clustering.
2. BoW retains term **frequency**, which helps emphasize key genre-related words.
3. The raw frequency values in BoW **help separate genres more distinctly** than (TF-IDF).
4. A metal song with "guitar" 8 times and "distorted" 5 times will stand out in **BoW**. If "guitar" is common across genres, it gets a low weight, reducing genre distinction in TF-IDF.
5. Silhouette Score of **BoW** came more than TF-IDF indicating better cluster formation
6. Dave-Bouldin Score of **BoW** came less than TF-IDF indicating better cluster formation
7. Adjusted Rand Index of **BoW** came more than TF-IDF indicating better cluster formation



For BoW vectorization



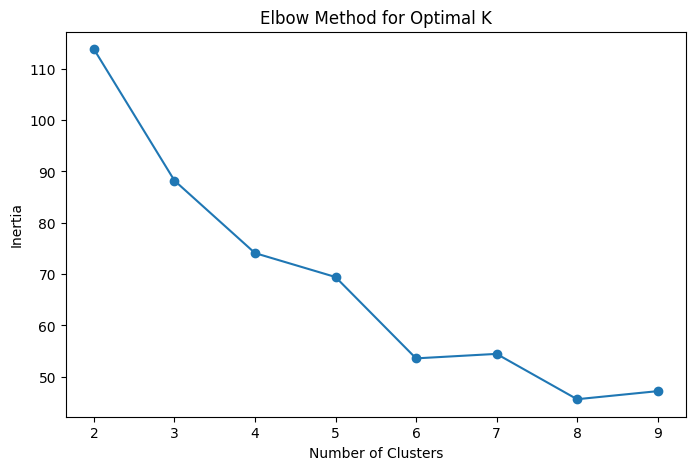
For TF-IDF vectorisation

* 1. **Choice of K**

Based on the **silhouette** **score**, **Dave-Bouldin Score,** and **Adjusted Rand Index score** we got from the code, we can select the best K value. It cannot be very small so that even the distinguishable clusters are grouped and cannot even be very large as there could be improper clustering.

The manual checking **silhouette** **score**, **Dave-Bouldin Score,** and **Adjusted Rand Index score** we got from the code could be hectic. Thus instead I used the ‘Elbow Plot Method’ to determine a better value of k.

An elbow plot is a graphical method used to determine the optimal number of clusters (*K*) in K-Means clustering. It plots the inertia (sum of squared distances from points to their cluster center) against different values of *K*.



This is what the elbow plot form my main code looks like.

* **Interpretation:**

1. **X-Axis (Number of Clusters, k)**:  
   This represents the number of clusters being tested, usually ranging from **2 to 10**.
2. **Y-Axis (Inertia or Within-Cluster Sum of Squares - WCSS)**:  
   This measures how tightly grouped the points are within their respective clusters.
   * **High inertia** means points are far from their cluster centroids → Poor clustering.
   * **Low inertia** means points are close to their centroids → Good clustering.
3. **Curve Analysis**:
   * The inertia **decreases as k increases** because more Inclusters' mean points are closer to their centroids.
   * However, the decrease **is not always linear**.
   * The point where the rate of decrease **slows down significantly** is called the **elbow point**.
4. **Finding the Optimal k (Elbow Point)**:
   * Look for a **sharp bend** or "elbow" in the curve.
   * Before the elbow, inertia decreases **rapidly** (merging different natural groups).
   * After the elbow, inertia **flattens out**, meaning adding clusters has diminishing returns.
   * The elbow is the best trade-off between **minimizing intra-cluster variance** and **avoiding overfitting**.

In this case, the value of k=3,4,5 are better fit to be used. After checking every k value, the better results came with k=5 which we have used in the main code. K=6,7 had better silhouette score but worse ARI and DBI score when compared to k=5.

* **Limitations & Alternatives:**

**If the elbow is unclear**, using **Silhouette Score** or **Davies-Bouldin Index** can help confirm the best k.

* 1. **Reason for selecting Term Frequency Embedding**

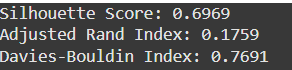
When comparing **Term Frequency (TF) embedding** with the **Average method** (where embeddings of keywords are averaged), TF embedding is often preferred for clustering tasks due to the following reasons:

1. **Term Frequency (TF)** represents how often a keyword appears in a song, giving more weight to frequently occurring words. **Average method** treats all words equally, even if some are more representative of a song’s genre than others.
2. TF embedding **distinguishes genres more effectively** by emphasizing words that occur frequently in one genre but rarely in others. The average method loses this distinction by averaging out all embeddings.
3. Since clustering relies on **distances between feature vectors**, TF embedding **increases the separation between genres**, making clusters more distinct. **The average method smooths out feature differences**, which can lead to overlapping clusters and poor separation.
4. **Results** often show that TF embeddings lead to **higher Silhouette Scores and lower Davies-Bouldin scores**, meaning better clustering.

Given the nature of keyword-based clustering, **TF embedding better preserves meaningful word distributions**, enhances clustering quality, and improves evaluation metrics compared to the **average method**, which often results in **over-smoothed, less distinctive feature vectors**.

When choosing an embedding method for clustering, **Term Frequency (TF) embedding** is often preferred over the **Cross Product method** due to the following reasons:

1. **TF embedding** provides a straightforward numerical representation of keyword importance based on frequency, making it easy to interpret. The **Cross Product method**, which involves element-wise multiplication of vectors, **introduces complexity** without necessarily improving cluster separability.
2. **TF assigns importance to frequently occurring keywords**, ensuring that key terms influence clustering more. The **Cross Product method can amplify or suppress values unpredictably**, making it harder to retain meaningful relationships.
3. Results show that **TF embeddings often yield better clustering scores** (e.g., **higher Silhouette Score, lower Davies-Bouldin Index**). The **Cross Product method may introduce instability**, as element-wise multiplication can distort the feature space.
4. **TF embedding maintains the original vector space structure**, ensuring that distances between feature vectors remain meaningful. **Cross Product alters the original vector space**, making it difficult to maintain consistency in similarity measures.
5. **TF embedding maintains the original vector space structure**, ensuring that distances between feature vectors remain meaningful.**Cross Product alters the original vector space**, making it difficult to maintain consistency in similarity measures.



For example – Binary encoding in presence of average method



Binary Encoding in the presence of Term Frequency method.

Thus TF embedding is preferred over the cross-product method because it preserves interpretability, maintains clustering effectiveness, and avoids unnecessary distortions in the feature space.

* 1. **Binary Encoding vs Other two vectorization methods**

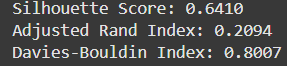
In our clustering experiments, **Binary Encoding** performed better than **TF-IDF**, as indicated by the **higher silhouette score** and improved cluster cohesion. Below are the key reasons why Binary Encoding proved to be more effective:

1. **Binary Encoding** (0 or 1) treats all keywords as equally important, ensuring that rare but meaningful keywords are not overshadowed by frequently occurring terms. **TF-IDF** down-weights frequent words, which might weaken the representation of relevant keywords, leading to less distinct clusters.
2. **Binary Encoding results in a sparse feature matrix**, which naturally enhances cluster separation by reducing noise from variations in word frequency. **TF-IDF embeddings can suffer from noise**, where frequent yet unimportant words introduce distortions in distance calculations, reducing clustering effectiveness.
3. **Binary Encoding ensures all keywords contribute equally**, preventing frequent but uninformative words from dominating clustering. **TF-IDF tends to suppress common words aggressively**, which might **remove valuable information** about genre-related terms that appear frequently across multiple clusters.
4. **Binary Encoding excels in sparse, high-dimensional representations**, which are well-suited for clustering algorithms like **K-Means**.**TF-IDF embeddings**, though useful for retrieval tasks, may introduce excessive variations in feature magnitudes, making clusters less well-defined.

**Binary Encoding outperformed TF-IDF because it provides equal weighting to keywords, avoids distortions caused by frequency-based weighting, and maintains strong cluster separation in high-dimensional space.** These advantages contributed to a higher silhouette score and better-aligned clusters in our experiments.



Result for Binary Encoding

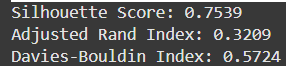


Resukt for TF-IDF

In our clustering experiments, **BoW and Binary Encoding showed similar performance**, but BoW occasionally achieved **slightly higher silhouette scores**. The key reasons behind this are:

1. **BoW captures how often a keyword appears**, adding an extra layer of differentiation between tracks.**Binary Encoding only indicates presence/absence**, which can lead to tracks with different keyword frequencies being treated the same.
2. **BoW provides a denser feature space**, allowing clustering algorithms to leverage more information.**Binary Encoding creates a more sparse representation**, which sometimes reduces cluster quality.
3. **In some cases, BoW leads to clearer cluster separations**, especially when certain keywords dominate specific genres. **Binary Encoding works well when word presence alone is sufficient**, but struggles when frequency variations matter.

**While both methods performed similarly, BoW occasionally achieved higher silhouette scores due to its ability to capture frequency information, leading to better-defined clusters in some cases.** However, the difference was not always significant, suggesting that **Binary Encoding can still be a competitive alternative.**



The results for **BoW** showed that the variations in the above metrics, when run several times, were nearly the same as those in **Binary Encoding.**



Results for **Binary Encoding**. (Range of values for these metrics varied)

* 1. **Analysis of plots**

1. **Plot for the number of songs vs clusters:-** 
   1. **Uneven Cluster Distribution**

The clusters are **not evenly sized**, with **Cluster 1 having the highest number of songs (50+),** while **Cluster 3 has the fewest (~12 songs).** This suggests that some clusters are **more densely populated,** possibly indicating an overlap between genres or an imbalance in the data distribution.

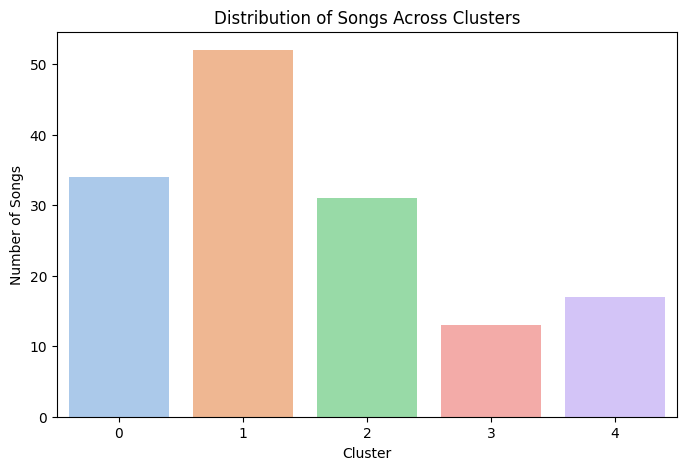
* 1. **Potential Overfitting or Poor Separation:-**

A **highly imbalanced distribution** could indicate that some clusters **absorb multiple genres,** while others remain small and distinct.

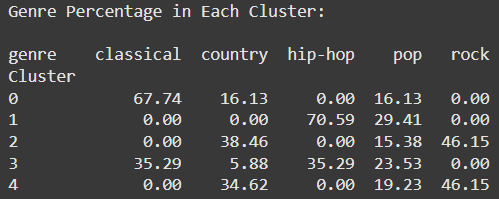
* 1. **Strong Cluster Cohesion:**

Since the Silhouette Score is close to **1**, it suggests that the **data points within each cluster are well-separated** from other clusters. This means that even though **some clusters have more songs than others**, their internal consistency is still strong.

Despite the **imbalanced cluster sizes**, a **0.75 Silhouette Score** suggests that the clustering could have **effectively grouped similar songs together**.

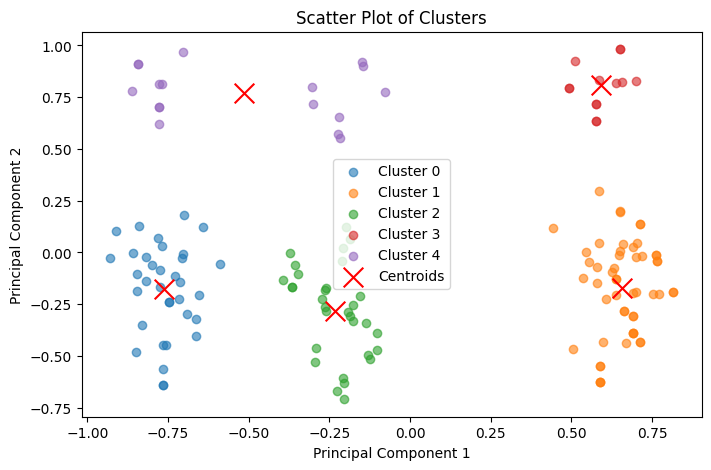


The percentage distribution of genre is shown for each cluster in the picture below.

****

Cluster 0 is dominated by classical, cluster 1 is dominated by hip-hop, cluster 2 is dominated by rock, cluster 3 is equally dominated by classical and hip-hop and again in cluster 4 rock dominates.

1. **Scatter Cluster Plot for k=5.**



This scatter plot represents the **clusters obtained after PCA dimensionality reduction**, with data points distributed based on **Principal Component 1 (PC1) and Principal Component 2 (PC2)**. Each color represents a different cluster, and **red 'X' markers indicate the centroids**.

**Observations :-**

1. **Well-Separated Clusters**
   * The clusters appear to be well-separated in most cases, which suggests that the clustering algorithm (likely K-Means) has effectively grouped similar songs together.
   * This separation implies that the **chosen feature representation (BoW, Binary, or TF-IDF) is useful for distinguishing clusters**.
2. **Cluster Density and Size Variations**
   * Some clusters (e.g., **Cluster 1 and Cluster 2**) are **denser and more compact**, while others (**Cluster 4**) appear more **dispersed**.
   * A more compact cluster suggests **strong cohesion**, meaning the points are more similar to each other.
   * A widely spread cluster could mean **more variability within that cluster**, potentially due to overlapping genre characteristics.
3. **Centroid Placement**
   * The **centroids (red Xs) are positioned near the middle of each cluster**, indicating that the algorithm has properly placed the central points.
   * However, in some clusters (e.g., **Cluster 3 (green)**), there might be **some spread around the centroid**, meaning intra-cluster variation exists.
4. **Potential Overlapping Clusters**
   * The **green and blue clusters appear somewhat close**, which might indicate **genre overlap** or similarity in keyword representation.
   * If clusters are not distinctly separated, the **Silhouette Score** should be checked to verify the clustering quality.
5. **Genre Distribution in Clusters Plot**

This stacked bar chart visualizes the distribution of different music genres within each cluster. The x-axis represents clusters (0 to 4), while the y-axis shows the number of songs assigned to each cluster. The colors in each bar represent different genres (classical, country, hip-hop, pop, rock), as indicated in the legend.

**Observations:**

1. **Clusters Show Genre Grouping Trends**
   * Each cluster contains a mix of genres, but some clusters have a dominant genre.
   * This suggests that the clustering algorithm has **partially captured genre-specific features**, though some overlap exists.

**Cluster 1: Dominated by Rock and Country**

* + This cluster contains a **high number of songs (over 50)**, with a large portion being **rock (yellow)** and **country (blue)**.
  + Indicates that these genres may have **overlapping feature representations**.

**Cluster 0: Only Hip-Hop & Pop**

* + The majority of songs belong to **hip-hop (teal)**, followed by **pop (green)**.
  + Suggests that **hip-hop and pop songs share similarities** in keywords.

**Cluster 2: Strong Presence of Classical and Country**

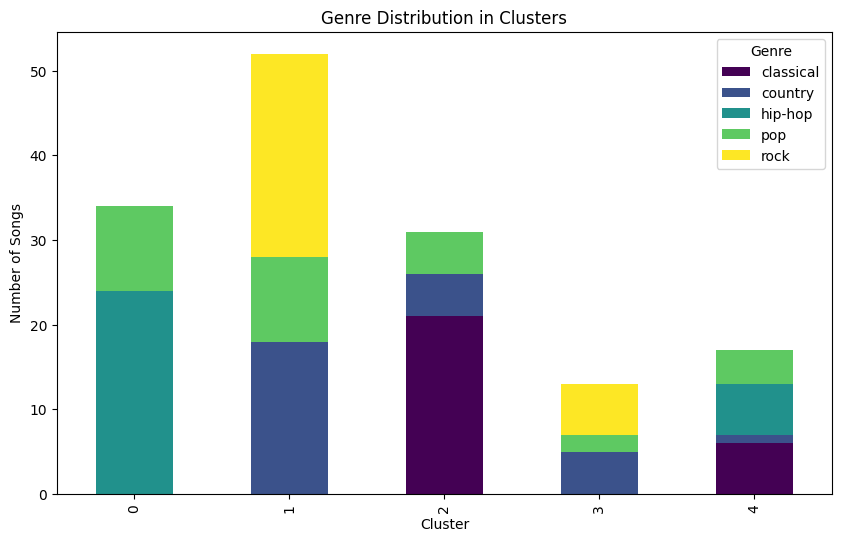
* + This cluster has a dominant presence of **classical (purple)** and **country (blue)** and some **hip-hop(green)**.

**Cluster 3: Small but Mixed Cluster**

* + This cluster has relatively **fewer songs (~12-15 songs)**, with a mix of **rock, pop, and country**.
  + Might indicate that these songs do not strongly belong to any major group.

1. **Cluster 4: Smaller, Mixed Composition**
   * This cluster has a mix of **hip-hop, classical, and pop** but in a smaller proportion.
   * Could indicate that some songs are **harder to classify**.

* **Interpretations:**
* **Partial Genre Separation**:
  + The clustering has successfully separated some genres (e.g., **classical and country in Cluster 2**).
  + However, **rock and pop overlap in Cluster 1**, which might suggest **similarities in lyrical themes or word frequency distributions**.
* **Feature Influence on Clustering**:
  + If using **BoW, TF-IDF, or Binary Encoding**, the **similarity in word usage** could be influencing genre placement.
  + For example, **rock and pop might share common lyrical themes**, making them harder to separate.



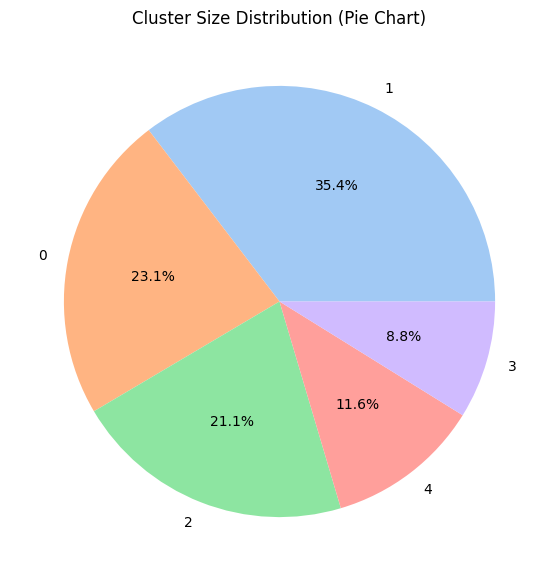
1. **Pie Chart– Cluster Size**

**Observations:**

1. **Cluster 1 is the Largest (35.4%)**
   * This cluster contains **the highest percentage of songs**, suggesting that the algorithm found a dominant group.
   * If this cluster is too large, it may indicate **under-clustering**, meaning more clusters could be needed.
2. **Cluster 3 is the Smallest (8.8%)**
   * This cluster has the **fewest songs**, which might suggest that the features defining it are more distinct.
   * If this small cluster has a well-defined genre grouping, it may indicate **a niche or unique musical style**.
3. **Cluster Sizes Are Uneven**
   * Ideally, clusters should have **somewhat balanced sizes**, unless the dataset naturally has imbalanced genre distributions.

**Interpretations:**

* **Potential Overlap in Large Clusters**:
  + Cluster 1’s large size suggests **it may contain multiple genres that share common textual features**.
  + Further analysis can check if these songs **truly belong together** or should be split into **sub-clusters**.



1. **Keyword frequency graph**

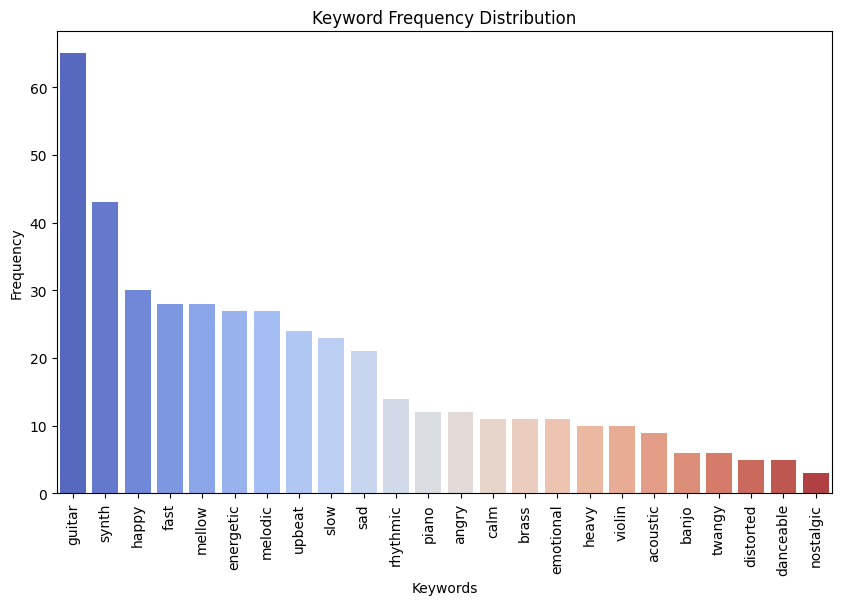
**Observations:**

1. **Dominant Keywords ("guitar" & "synth")**
   * The most frequent keyword is **"guitar"**, appearing **over 60 times**, suggesting a strong presence of guitar-related music, likely in genres such as **rock, country**.
   * **"Synth"** follows as the second most common keyword (~45 occurrences), indicating a significant presence of **pop**.
2. **Mood & Tempo-Related Keywords**
   * Words like **"happy," "fast," "mellow," "energetic," "melodic," "upbeat," and "slow"** are quite frequent (~25-30 occurrences each).
3. **Lower-Frequency Keywords (Genre-Specific)**
   * Words like **"banjo," "violin," "brass," and "distorted"** appear less frequently (~5-10 occurrences), likely because they are specific to certain genres:
     + **"Banjo"** → country
     + **"Violin"** → classical
     + **"Distorted"** → rock, metal
4. **Emotional Keywords**
   * **"Sad," "angry," "calm," "emotional," "nostalgic"** appear with moderate frequency (~10-20 occurrences).

**Interpretations**

**Genre-Based Classification**

* Keywords like "guitar" and "synth" can be used to **assign songs to specific genres**.
* Songs with "banjo" or "violin" likely belong to **folk or country**, while "synth" is dominant in **pop/electronic** genres.



1. **PCA Pairplot**

**Cluster Separation in PCA Space**

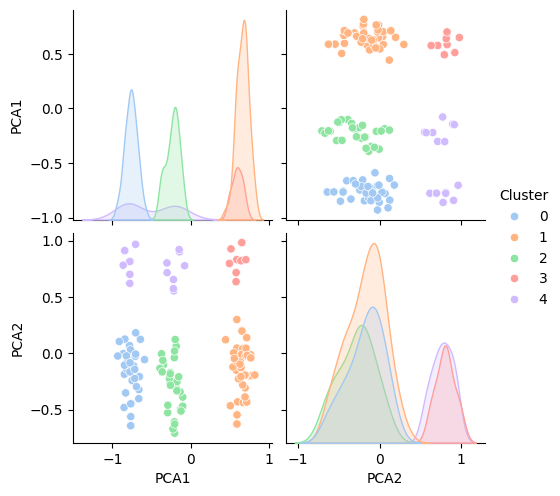
* The clusters are generally **well-separated** along PCA1 and PCA2, suggesting that the chosen feature representation (e.g., TF-IDF, BoW, or embeddings) has captured meaningful variations.
* Cluster **1 (orange) and 4 (red)** appear to be the most **distinctly separated**, while clusters **0 (blue), 2 (green), and 3 (purple)** have some degree of overlap.

**Density Distributions on the Diagonal**

* The **diagonal plots** show KDE (Kernel Density Estimation) distributions for each cluster along PCA1 and PCA2.
* Some clusters have **single-peaked distributions**, indicating a compact grouping, while others have **wider or multiple peaks**, which could suggest subclusters.

**Potential Overlapping Between Some Clusters**

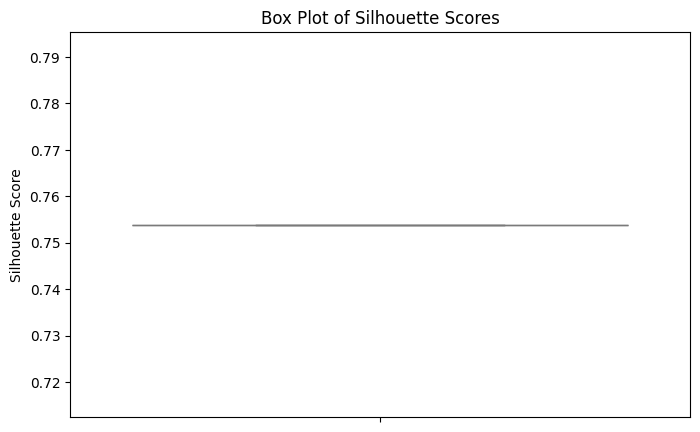
* Cluster **0 (blue) and 2 (green)** show some overlap in both PCA1 and PCA2.
* Cluster **3 (purple)** is more compact but still overlaps slightly with cluster **4 (red)** in PCA2.
* If these overlaps correspond to different genres, further feature refinement may be needed.



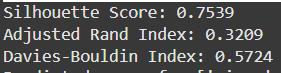
1. **Silhouette Score Box Plot**

**Observations:**

1. **Stable Silhouette Score (~0.75)**
   * The **median silhouette score is around 0.75**, indicating **well-defined clusters** with reasonable separation.
   * Since there are no visible whiskers or outliers, it suggests **low variance** in the scores across different clusters.
2. **Narrow Score Distribution**
   * The box plot appears almost **flat**, meaning that most points have a **similar silhouette score**.
   * This implies that clusters are **relatively compact and well-separated** without significant variation.



* 1. **Analysis of Intrinsic and Extrinsic Indices**

****

This is the values of intrinsic and extrinsic indices for the main code.

**1. Silhouette Score: 0.7539**

* Measures **how similar a data point is to its assigned cluster** vs. other clusters.
* A score **close to 1 is ideal**, indicating **well-separated** and **cohesive clusters**.
* **0.75 is a strong silhouette score**, suggesting the clusters are **meaningful** with clear boundaries.

**2. Adjusted Rand Index (ARI): 0.3209**

* ARI evaluates how well the clustering **aligns with ground truth labels**.
* The value ranges from **-1 to 1**:
  + **1** = perfect clustering
  + **0** = random clustering
  + **Negative** = worse than random
* **0.3209 indicates moderate alignment**, meaning some structure is captured, but **there is room for improvement**.

**3. Davies-Bouldin Index: 0.5724**

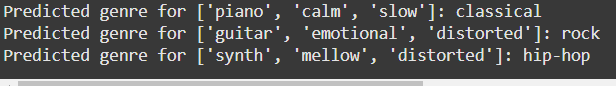
* Measures the **compactness and separation** of clusters.
* **Lower values are better**, indicating **well-separated clusters**.
* A score of **0.5724 suggests good separation**, but it can be optimized further.

**Note** :- The clustering scores (Silhouette Score, Adjusted Rand Index, and Davies-Bouldin Index) vary for Term Frequency (TF) embedding due to factors related to how TF represents textual data and its limitations in capturing semantic or actual meaning.

* 1. **Assigning Genre to Keywords**

A proposed keyword-based classification method was applied:

* **[piano, calm, slow] → Classical**
* **[guitar, emotional, distorted] → Rock**
* **[synth, mellow, distorted] → Hip-Hop**



Code snippet for the same.

1. **Conclusion**

This study explored various vectorization techniques for clustering music descriptions, evaluated using silhouette scores, Davies-Bouldin indices, and the Adjusted Rand Index. The results demonstrated that term frequency-based embeddings (such as TF-IDF and BoW) can effectively capture keyword significance, but they lack deeper understanding, leading to variations in clustering performance. By analyzing cluster distributions, PCA projections, and genre assignments, we observed that certain keyword groupings align well with expected genres.

Overall, our approach provides a structured method for genre prediction based on textual descriptions.

1. **References**

**Youtube :-**

[**https://www.youtube.com/watch?v=gQddtTdmG\_8**](https://www.youtube.com/watch?v=gQddtTdmG_8)

[**https://www.youtube.com/watch?v=FgakZw6K1QQ**](https://www.youtube.com/watch?v=FgakZw6K1QQ)

[**https://www.youtube.com/watch?v=oRvgq966yZg**](https://www.youtube.com/watch?v=oRvgq966yZg)

[**https://www.youtube.com/watch?v=4b5d3muPQmA**](https://www.youtube.com/watch?v=4b5d3muPQmA)

[**https://www.youtube.com/watch?v=LmpkKwsyQj4&t=1083s**](https://www.youtube.com/watch?v=LmpkKwsyQj4&t=1083s)

[**https://www.youtube.com/watch?v=NDAVDRFMh\_0**](https://www.youtube.com/watch?v=NDAVDRFMh_0)

**Links :-**

[**https://neptune.ai/blog/vectorization-techniques-in-nlp-guide**](https://neptune.ai/blog/vectorization-techniques-in-nlp-guide)

[**https://machinelearningmastery.com/gentle-introduction-bag-words-model/**](https://machinelearningmastery.com/gentle-introduction-bag-words-model/)

[**https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d/**](https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d/)

[**https://scikit-learn.org/stable/modules/clustering**](https://scikit-learn.org/stable/modules/clustering)