Scalable Music Data Analysis Using the Million Song Dataset

Team Name: The Hardy Hadoopers

Teammates:
Alec Pippas (awp251)
Bhavya Matam (bm3792)
Naman Vashishta (nv2375)
Saja Alsulami (sfa7673)
Zhuyuan Wang (zw4759)

1. Introduction

Music recommendation, playlist curation, and genre classification are critical challenges in the modern music industry, where data volumes are massive and diverse. This project analyzes large-scale music data to uncover patterns and groupings within songs using unsupervised techniques and scalable, Big Data processing tools.

We will use the **Million Song Dataset (MSD)**, a large-scale, industry-grade dataset, to explore several unsupervised clustering techniques that allow us to identify similar song groupings (clusters) that can be used for music recommendation or playlist creation.

Throughout this project we use the following clustering technique such as K-means.

1.1 Tools and Platforms

Overall the project utilizes scalable, Big Data tools and platforms in order for all related pipelines (data preprocessing, clustering analysis, and clustering visualization) to be fully scalable and able to process the entire 280 GB MSD dataset within a distributed computing environment.

Big Data tools used include:

- **Libraries:**
  - PySpark, Spark MLlib, h5py, PyMongo
- **Databases:**
  - MongoDB server deployed on MongoDB Atlas, a scalable, cloud-based database service
- **Cloud-based Notebook Environments:**
  - Kaggle Notebook with a Spark Session (running in local mode) with 24GB RAM, 4 Executor Cores
  - *The entire project pipeline is structured in a way that is fully portable to a genuine distributed computational environment with a Spark Execution Engine set up.* In order to run the project pipeline code end-to-end in a distributed environment, the Spark Session would have to be reconfigured to not run in local mode (and other environmental capacities would have to be taken into consideration).
- **Visualization Tools:**
  - Matplotlib, Seaborn, and Plotly

2. Dataset Overview

The Million Song Dataset is a **280 GB collection of metadata and audio features** for one million popular contemporary music tracks from 1920s- 2010. The dataset is created by Bertin-Mahieux et al.(2011). The dataset was published in the 12th International Society for Music Information Retrieval Conference (ISMIR 2011). Notably, the dataset consists of text-based metadata and features rather than raw audio files.

For practical purposes, and due to the momentary unavailability of the full 280 GB Million Song Dataset when this project was being put together, our clustering analysis relies on the approximately 2 GB, 10,000 song subset provided on the Million Song Dataset website. We do not believe that use of this subset hinders the demonstrability of our clustering analysis. Furthermore, given the scalable design of our project's code, utilizing the associated pipelines of the project would allow clustering analysis to be performed on the full MSD with accurate, meaningful clustering results.

2.1 Core Features

The fields of the MSD are extracted via **The Echo Nest Analyze API** and are well-suited for machine learning tasks. Each record, a song track, includes more than **50 fields**, such as:

- **High-level audio features**:
  - danceability, energy, tempo, loudness, key, mode, duration, time_signature, year, etc.
- **Time-series musical structure**:
  - bars_start, beats_confidence, segments_pitches (935×12), segments_timbre, tatums_start, etc.
- **Metadata**:
  - artist_name, title, release, song_hotttnesss, track_id, artist_terms, etc.

Our clustering analysis will rely on the *song_id*, *artist_name*, and *title* metadata fields, and the *segments_timbre* and *segmenents_pitches* audio feature fields.

2.2 Preliminary Exploration of Raw Dataset

In order to get a better understanding of the distribution and characteristics of the Million Song Dataset, the following exploratory visualizations have been produced.

Figure 1

Top 20 Genre Terms in the Dataset

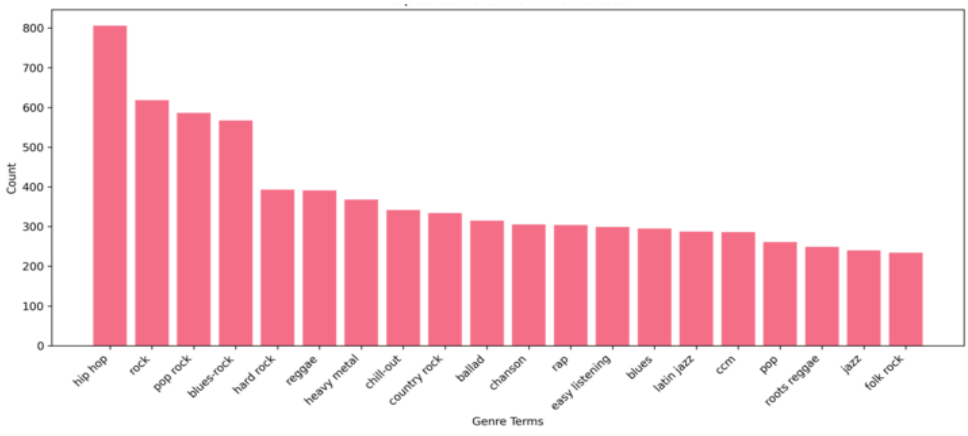


Figure 2
Distribution of Songs by Year

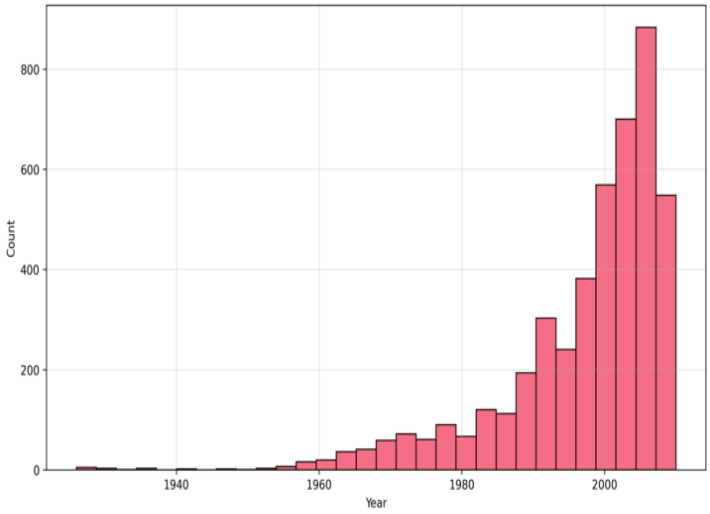


Figure 1, "Top 20 Genre Terms in the Dataset", depicts that our 10,000 song subset comprises a relatively high proportion of songs within the hip-hop and rock genres. Hip hop alone accounts for approximately 8% of the subset, meanwhile rock and subgenre derivatives, pop rock and blues rock accounts for about 24% of the songs in the subset. The other genres in the dataset are relatively uniformly distributed.

From Figure 2, "Distribution of Songs by Year", despite containing a range of songs from 1920 - 2010, it is evident that a majority of the 10,000 song subset contain songs from after the year 1980. More notably, a substantial majority of the songs are from the early 2000s (2000 - 2010). It is likely that this concentration of songs in more recent years pulls songs from earlier years into clusters dominated by later-year songs. As such, the clustering of songs was largely based on the audio features of later year songs.

3. Project Goals

We aim to:

- Explore **unsupervised clustering** of songs using PCA and UMAP representations of audio features
- Develop **interpretable visualizations** of similar songs clusters in lower-dimensional feature space (2-D/3-D)
- Showcase a basic use case of clustered songs: music recommendation

This project emphasizes **scalable big data analysis**, using Spark and dimensional reduction to handle high-dimensional, large-volume input efficiently.

4. Methodology

4.1 Data Extraction and Preprocessing

Our clustering pipeline begins with the 10,000 song subset of the Million Song Dataset. This subset is distributed in a HDF5 format which consists of thousands of nested *.h5* files nested within a hierarchical directory structure. To begin the preprocessing pipeline, we first initiate a PySpark session. We then crawl the directory tree using standard Python functions, and stream each file into a Spark DataFrame, pulling only the fields that matter downstream: the identifiers (*song_id*, *artist_name*, *title*) and the two audio feature, time-series arrays, *segments_timbre* and *segments_pitches*. Reading through Spark keeps the I/O parallelised and avoids driver-side memory overflow issues, while the session configuration (24 GB executor and driver memory, four executor cores) is tuned to Kaggle's CPU notebook limits so that wide transformations never spill to disk.

Once selected fields of the raw data samples are in Spark dataframes, we normalize the textual metadata to eliminate join-time ambiguity. In this particular case, we only normalize artist names; each *artist_name* is stripped of diacritics with Unicode NFKD, lower-cased, and deduplicated; any record missing its primary key or the two audio arrays is dropped outright. This cleaning step is intentionally aggressive because duplicating *song_ids* or null *timbre* vectors would pervert later distance calculations biasing the cluster centroids at each iteration of the selected clustering technique, and ultimately resulting in clusters with centroids that are marginally shifted towards the duplicate data samples.

The most important part of the preprocessing pipeline is the aggregation of the variable-length audio sequences into fixed-length feature vectors. Each song's *segments_timbre* field and *segments_pitches* field are matrices with shape *(n_segments, 12)*, where n_segments varies from song to song (given the different lengths of each song). For every one of the 12 timbre and 12 pitch coefficients we compute four statistics, mean, max, min, and standard deviation across all *n_segments* segment susing PySpark UDFs that operate on the native NumPy arrays already present in each row. This results in a compact, 96-dimensional numerical fingerprint per song (12 x 4 = 48 from the *segments_timbre* fields, 12 x 4 =48 from the *segments_pitch* fields). Fixed dimensionality is essential because without it, no dimensionality reduction techniques or common distance metric during clustering could be applied across all samples within the dataset.

At the end of the aggregation step, the 10,000 data samples are cleaned and the *segments_timbre* and *segments_pitches* fields are engineered into the following features: *timbre_mean*, *timbre_max*, *timbre_min*, *timbre_std*, *pitch_mean*, *pitch_max*, *pitch_min*, and *pitch_std*. We then batch-insert them (in batches of 1,000 into MongoDB Atlas). Mongo provides two advantages over local parquet or CSV storage that matter for the upcoming experimental iterations. First, horizontal scaling lets us grow toward the full million-track dataset without refactoring the pipeline and running into unscalable local node memory constraint issues. Second, MongoDB's document storage model supports ad-hoc querying of both the metadata and the statistically aggregated pitch and timbre mean/max/min/standard-deviation audio features, which is useful when we want to segregate the data (e.g., by year, genre, artist, etc.) before rerunning new clustering jobs on a subset of the original dataset.

After the preprocessing pipeline the dataset is clean, normalized, and stored in MongoDB with a uniform feature representation all ready to feed into the dimensionality reduction and clustering pipelines in the subsequent project notebooks: *02-dimreduce.ipynb* and *03-kmeans-clustering.ipynb*.

4.2 Dimensionality Reduction

Dimensionality reduction is used to transfer the high dimensional space datasets into lower dimensional space and keep the important features of the original data. The reduction helps to reduce the complexity of

the dataset and make it a simple version so that we can use it to visualize the dataset. In this project we have a large dataset so we need to implement dimensionality reduction before we use the data in a cluster. This project used three methods of dimensional reduction:

- Principal Component Analysis (PCA) features (15 components)
- Uniform Manifold Approximation and Projection (UMAP 2D) features (UMAP_1, UMAP_2)
- Uniform Manifold Approximation and Projection(UMAP 3D) features (UMAP_1, UMAP_2, UMAP_3)

Using dimensional reduction can help us to mitigate noise and redundancy while keeping essential information. Also, decreases computational cost in clustering. When we have Lower-dimensional in our data is more suitable for clustering algorithms, reducing the risk of overfitting and enhancing interpretability [1].

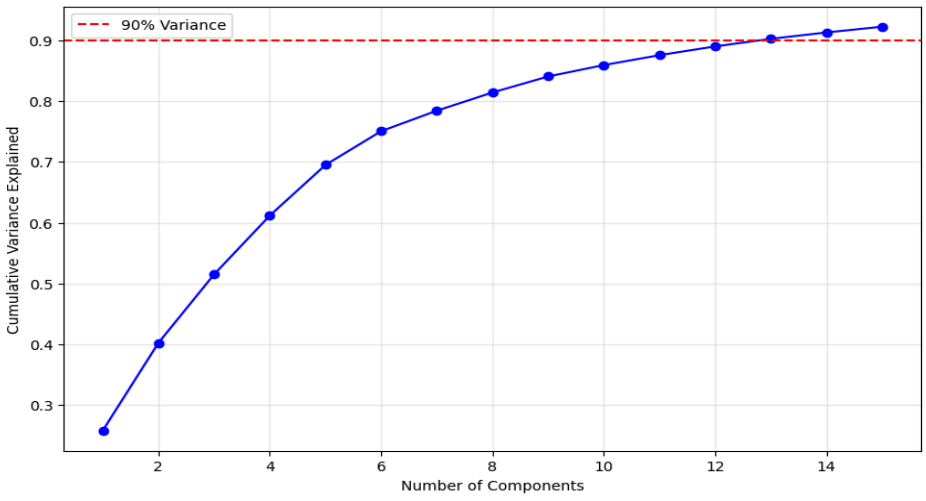
4.2.1 Principal Component Analysis (PCA):

PCA is a reduction that is used for linear dimensionality reduction, and it transforms the complex data into simpler data. PAC is transferring the data into a new coordinate system where the highest variance for the projection of the data is the first coordinate, and the second highest variance is the second coordinate, etc.

Figure 3 shows the cumulative variance which represents the variance. The variance is the main point of choosing the coordinate to reduce the data dimension [2].

Figure 3

Cumulative Explained Variance by PCA Components

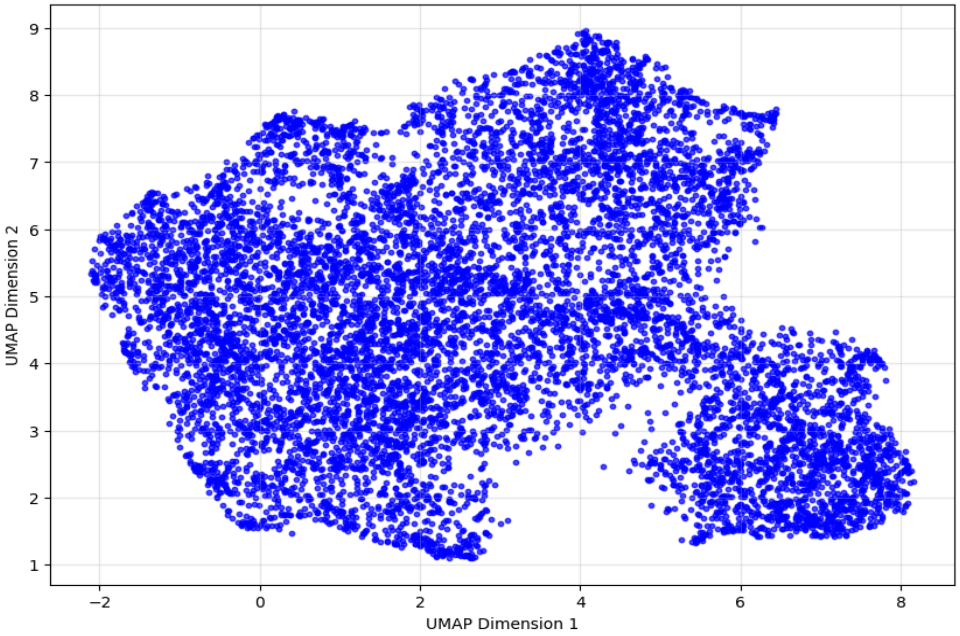


4.2.2 Uniform Manifold Approximation and Projection (UMAP) 2D:

UMAP is dimensional reduction used to visualize the high dimensional dataset into a simpler version that has the same main features. The UMAP 2D helps visualize the two dimensional dataset. Figure 4 below shows the result after we implement the UMAP 2D in Million Song dataset [3].

Figure 4

2D UMAP Projection

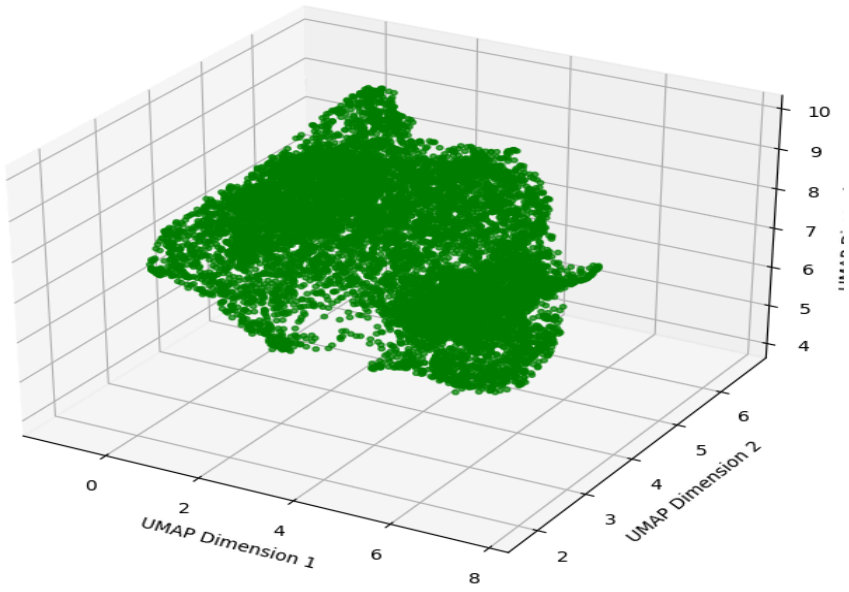


4.2.3 Uniform Manifold Approximation and Projection (UMAP) 3D:

UMAP 3D is a reduction dimension method that helps to visualize the dataset. This project used UMAP 3D to visualize the dataset into three dimensions. Figure 5 below shows the result after implementing the three dimension reduction on a million song dataset [3].

Figure 5

3D UMAP Projection



4.3 Deducing Optimal Number of Clusters, *k*

Clustering is a type of unsupervised machine learning technique that uses an unlabeled dataset. All of the various clustering techniques are essentially ways of grouping data samples from a dataset into groups based on the similarities of each sample's features. This project uses clustering to group the similar song into the same cluster, which, in turn, can be used for song recommendation and playlist procurement [4].

In order to achieve optimal clustering results, the optimal *k* value(s), where *k* represents the number of clusters, must be chosen before conducting the actual clustering pipeline. As is evident in Figure 6 and Figure 7 below, the WCSS (within-cluster sum of squares) will always falls a k grows; however, the goal is not to minimize the WCSS, but rather it is to minimize the expected error on unseen data (i.e. the clustering model generalizes well to new unseen test data, achieving an acceptably low WCSS test error). Please refer to Annex 1 for a mathematical depiction of the WCSS.

If *k* is too small (too few clusters), heterogeneous data samples will all be crammed into the same cluster, meaning there will be a high intra-cluster variance (high WCSS) and low inter-cluster separation. Pragmatically, this would result in a poor music recommendation system because songs within clusters would not actually be substantially similar.

On the other hand, if *k* is too large (too many clusters), natural groupings of data samples (songs) are fractured into multiple mini- or micro-clusters. This means the centroids are chasing training set specific noise, and the generalizability of clusters for downstream cluster identification of new data samples is worsened. In effect, setting *k* too large causes clusters that are overfit, and potentially clusters that consist of only a single datapoint, prohibiting a cluster from resembling relational similarities to a group of datapoints, as it is only a single, idiosyncratic point. In a music recommendation context, having too many clusters would limit the amount of songs a particular query song could be related to, in other words the query song's neighborhood shrinks [9].

The project uses two different methods to decide the optimal *k* values of the categories:

- **Elbow method**
- **Silhouette analysis**

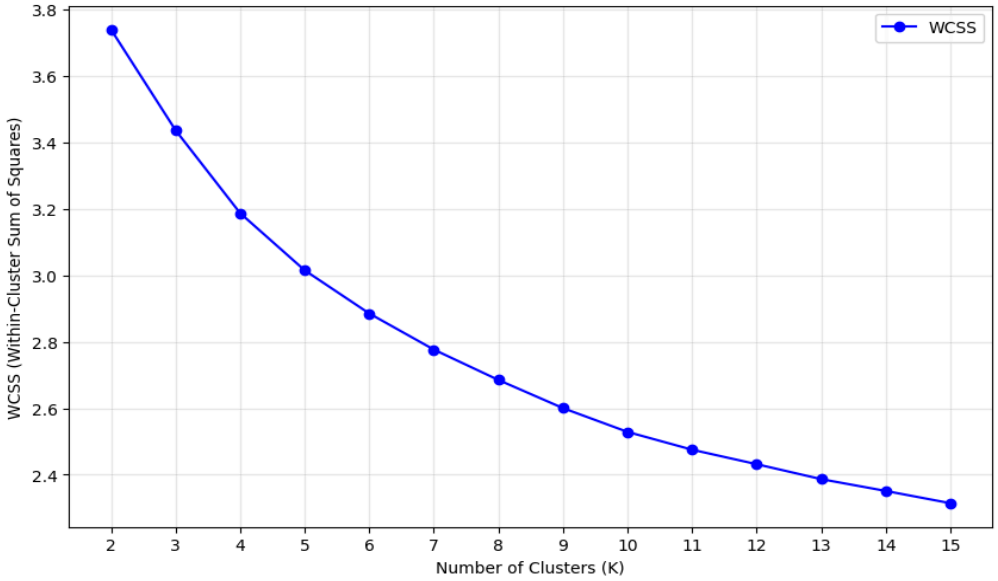
4.3.1 Elbow Method

The elbow method is a technique used in data analysis to determine the optimal number of clusters in a dataset. Elbow method which is a method to calculate the within-cluster sum of squares that is named as WCSS. Elbow point where the rate of decrease within-cluster sum of squares starts to level off, suggesting a suitable number of clusters [5]. The highest change of the slope at a specific point will be considered as the elbow point.

In PAC, using the elbow method results in an optimal cluster size: $k = 5$. As we can see in Figure 6, the elbow point is 5 because after k = 5, the slope begins because the rate at which the slope is decreasing becomes less significant.

Figure 6

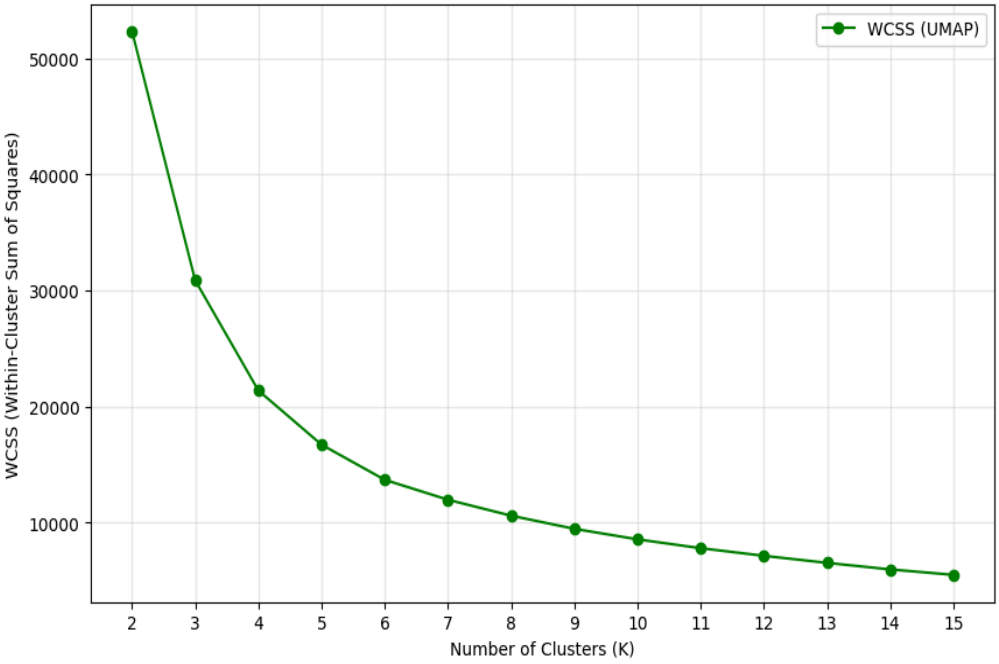
Elbow Method for PCA Data



The UMAP has an elbow point that represents the *k* value which is 3. In Figure 7 below the elbow point is 3 as it has the highest change in on the slope. For using the UMAP; the best k value of cluster is 3. At point 3 we can see the highest change in the slope.

Figure 7

Elbow Method for UMAP Data



4.3.2 Silhouette Analysis

A method used in cluster analysis to evaluate the quality of clustering results and help determine the optimal number of clusters. This method measures the distance between the clusters and it gives a score for the clustering.This method measures the distance between the clusters, and it gives a score for the clustering. It measures how each point is close to the cluster. The score of the silhouette between the -1 to +1. Each score has a different interpretation in the clustering [6].

- +1: represent the points away from the other clusters and inside its own cluster (ideal clustering).
- 0: represent the points that are very close to the decision boundary between two clusters (overlapping clusters).
- -1: represent the points that have been assigned to the wrong cluster as it closes to another cutler from the cluster that is assigned in.

This project used Silhouette analysis for PCA and the UMAP. In the PCA we assumed the *k* value was 5 and the silhouette analysis gave us a score equal to 0.1296. On the other hand for UMAP we assume that it has a value of $k = 3$ and silhouette analysis gives us a score of 0.4576.

4.4 K-means Clustering

This project implements K-means clustering which is an unsupervised machine learning algorithm. The K-means clustering algorithm groups similar data points into one cluster which makes each category have similar features. The K mean algorithm is clustering the data into K similar groups. The algorithm used Euclidean distance to calculate the similarity in the groups.

There are many steps to cluster the data in K mean clustering algorithm:

1. Initialize a random value of K value called centroid for the data.
2. Each data point is assigned to the closest centroid.
3. After assigning all the points, update the averages of the points in the same cluster and find the average point for each cluster.
4. Repeat the process till the center of the cluster stops changing [7] .

In our project we used the K-mean algorithm to group similar songs based on various features into one cluster. The *k* value of the PCA is 5 and in UMAP is 3 depending on the results from the elbow and silhouette method [8].

The result of implementing the elbow and silhouette method resulted in PCA having clustered into 5 categories and UMAP 2D and UMAP 3D clustered into 3 categories as it shows in the below figure 8 and figure 9.

Figure 8

Cluster Distribution using PCA Data (k=5)

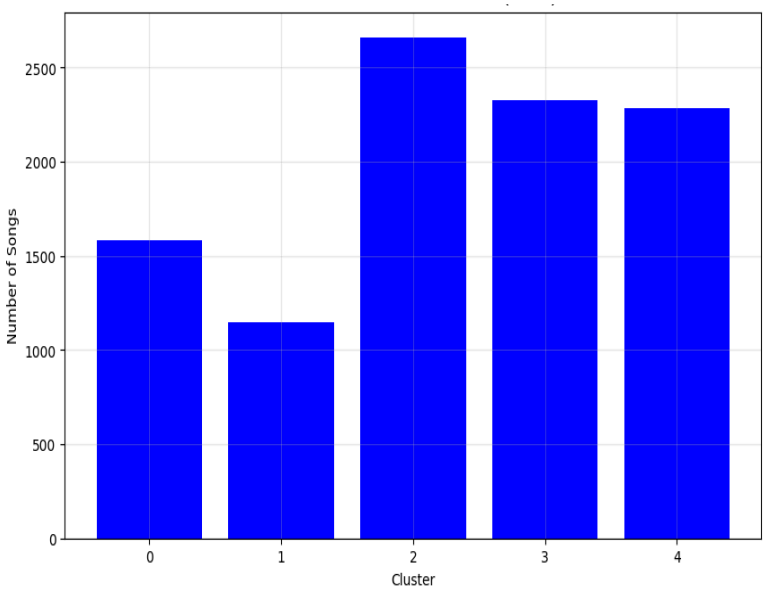


Figure 9

Cluster Distribution using UMAP 2D Data (k=3)

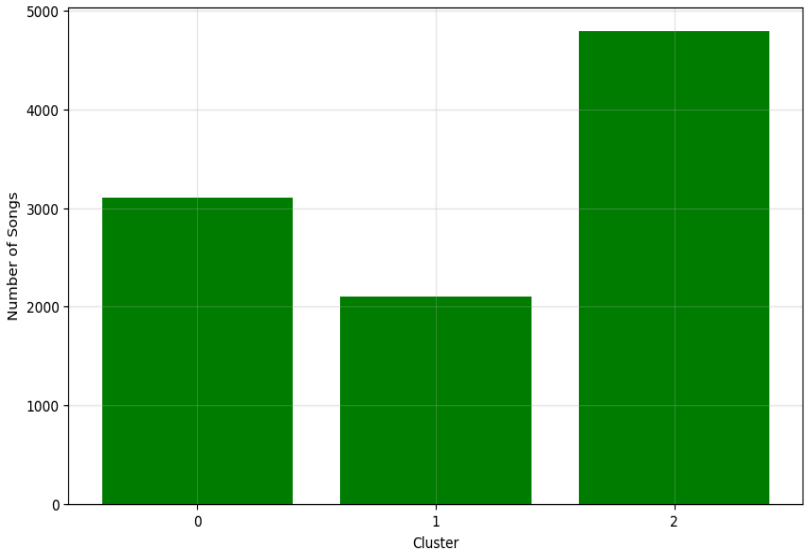
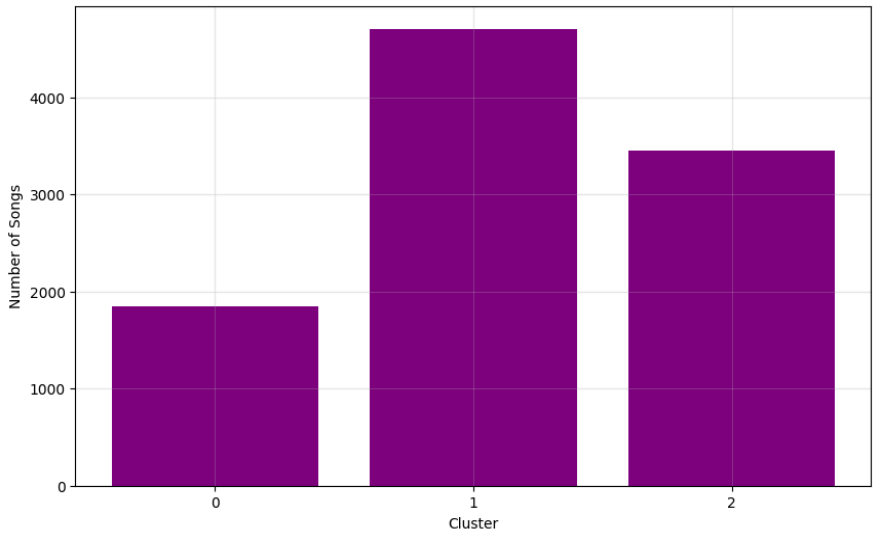


Figure 10

Cluster Distribution using UMAP 3D Data (k=3)



5. Clustering Visualizations and Results

5.1 Visualizing the Clustering Results

The following clustering visualizations have been produced so that clusters of higher-dimensional data samples can be visualized in a human-readable format: 2-dimensional and 3-dimensional.

Figure 11

PCA 2D Scatter Plot – K-Means Clustering Visualization (k=5)

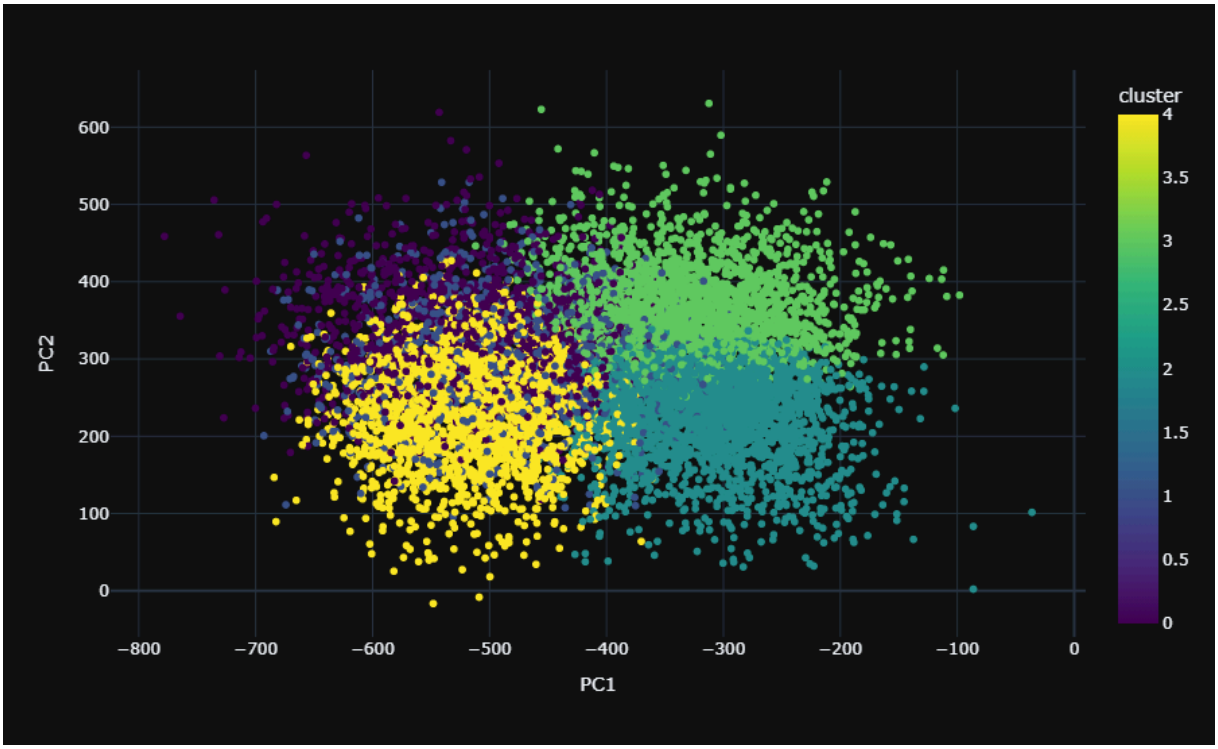


Figure 12

UMAP 2D – K-Means Clustering Visualization (k=3)

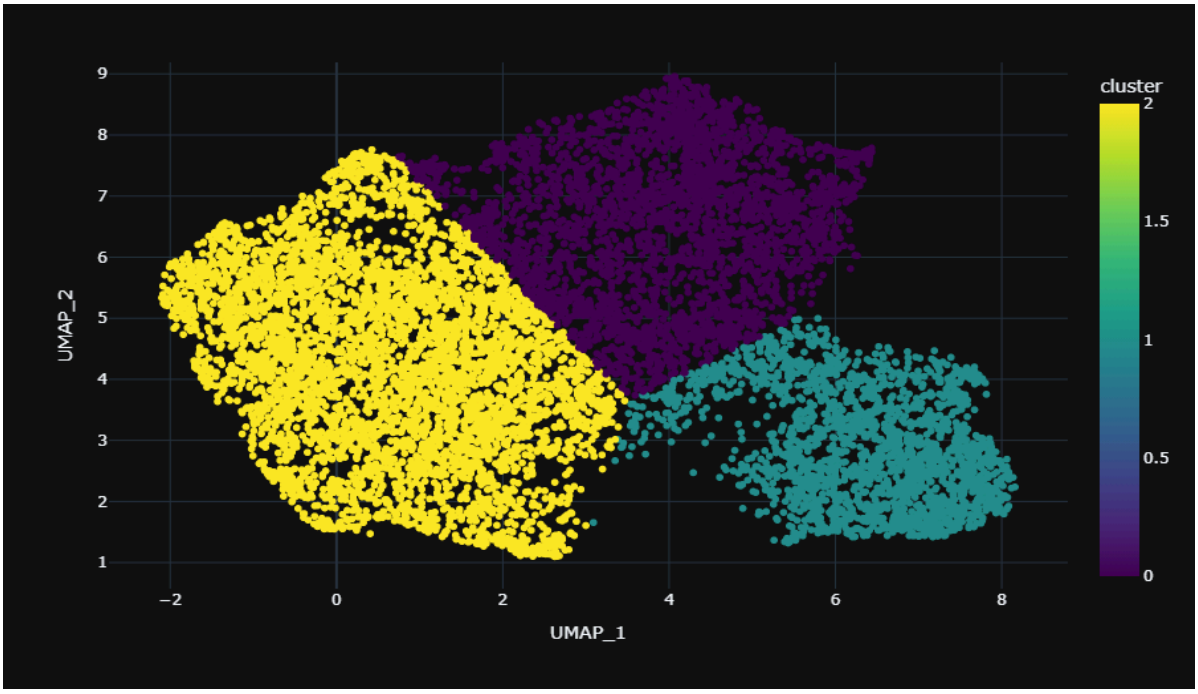


Figure 13

UMAP 3D – K-Means Clustering Visualization (k=3) – 1st Angle

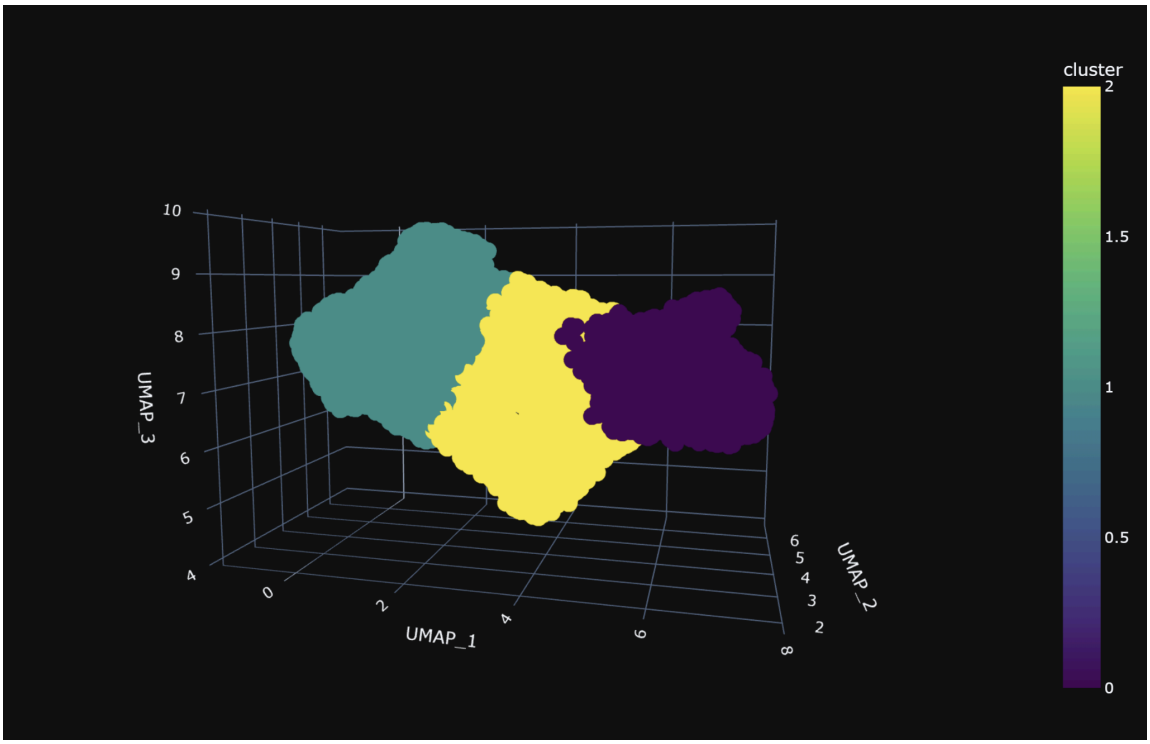
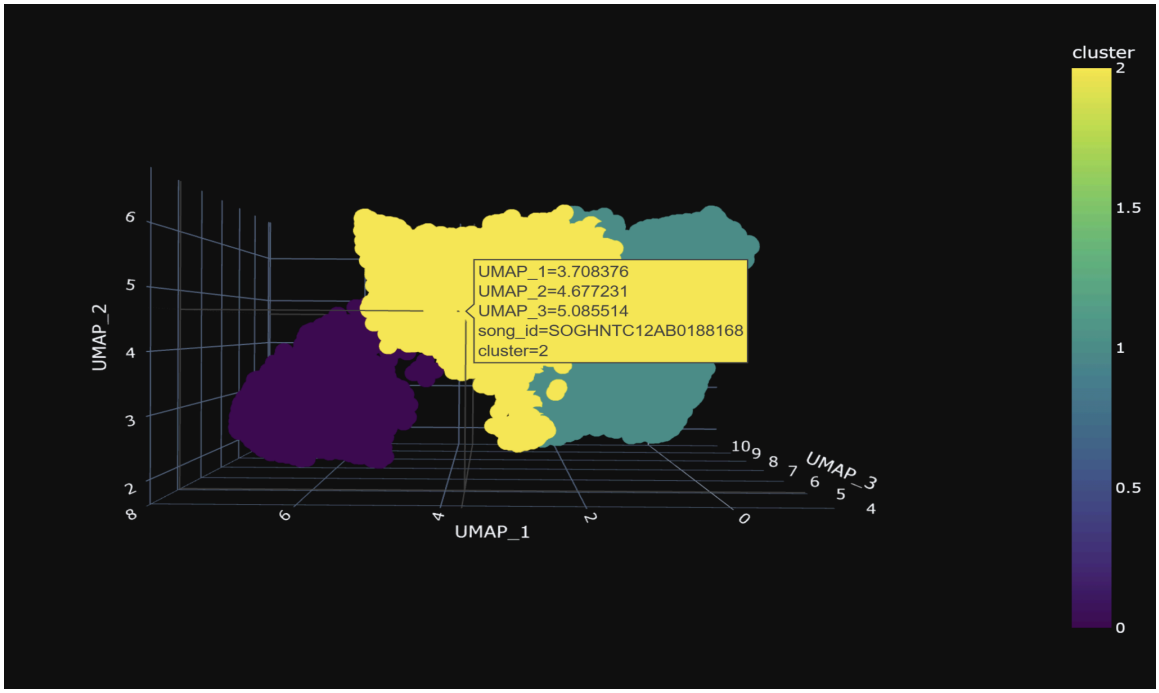


Figure 14

UMAP 3D – K-Means Clustering Visualization (k=3) – 2nd Angle



5.2 Interpretation of Results

The clustering results obtained through PCA, UMAP 2D, and UMAP 3D embeddings provided compelling insights into the underlying structure of the music data. Using audio features such as timbre and pitch, we applied K-Means clustering to identify song groupings that share similar acoustic properties. The PCA-based clustering, which used 15 principal components and a cluster count of k=5, produced moderately distinct clusters. However, some overlap was observed, and the silhouette score of 0.1296 indicated that the clusters were not well-separated. On the other hand, UMAP both in 2D and 3D revealed clearer and more cohesive groupings, with a significantly higher silhouette score of 0.4576, suggesting more well-defined cluster boundaries. The UMAP embeddings preserved local structures more effectively, leading to tighter and more meaningful clusters in the reduced-dimensional space.

6. Conclusion & Further Discussions

Based on the analysis of the Million Song Dataset using big data tools and unsupervised clustering techniques, we drew several key results and conclusions.. Through data preprocessing and dimensional reduction techniques including PCA and UMAP, and K-Means clustering, songs of the subset were successfully grouped based on their audio features. Evaluation metrics such as the Elbow method and Silhouette analysis determined the effectiveness of the cluster algorithm and parameters on this dataset. Analysis of these clusters revealed distinct song profiles, categorizing music based on characteristics like tempo, timbre, and pitch (e.g., upbeat, cinematic, acoustic, vocally intense tracks). These findings demonstrated that clustering algorithms in organizing music based on intrinsic audio attributes can be meaningful.

Our work on the Million Song Dataset provides foundations for future research and development. Several avenues can be utilized to enhance the analysis and potential applications. This includes exploring more advanced audio features like MFCC and spectral contrast to refine clustering accuracy. We also aim to conduct temporal analysis to identify evolving music trends over time and implement additional clustering techniques such as DBSCAN and hierarchical clustering to uncover deeper relationships within the data. A significant next step could be to scale the analysis to the complete Million Song Dataset, optimizing Spark configurations and exploring GPU acceleration to handle the larger volume efficiently. Furthermore, we intend to develop a real-time recommendation engine, which will involve migrating the well-formed results to a vector database like Qdrant to enable faster and more dynamic suggestions in realistic applications.

Appendix

Annex 1

Within-Cluster Sum of Squares (WCSS) Formula

$$\mathrm{WCSS}(k) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2,$$

Million Song Dataset Citations:

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)* (pp. 591–596). https://www.ee.columbia.edu/~dpwe/pubs/BertEWL11-msd.pdf

Dataset available at:

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). The Million Song Dataset. http://millionsongdataset.com

References:

[1] Carreira-Perpiñán, M. A. (1997). *A review of dimension reduction techniques* (Technical Report CS–96–09). Department of Computer Science, University of Sheffield. https://faculty.ucmerced.edu/mcarreira-perpinan/papers/cs-96-09.pdf

[2] Calvert, W. (2014). PAC learning, VC dimension, and the arithmetic hierarchy. *arXiv*. https://arxiv.org/abs/1406.1111

[3] B. Li, Y. Zheng and R. Ran, "2DUMAP: Two-Dimensional Uniform Manifold Approximation and Projection for Fault Diagnosis," in IEEE Access, vol. 13, pp. 12819-12831, 2025, doi: 10.1109/ACCESS.2025.3531712. https://ieeexplore.ieee.org/document/10845752

[4] Rokach, L., & Maimon, O. (2005). Clustering methods. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook* (pp. 321–352). Springer. https://www.researchgate.net/profile/Lior-Rokach/publication/226748490_Clustering_Methods/links/02e7e536dcb9b70ea3000000/Clustering-Methods.pdf

[5] Bholowalia, P., & Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and k-means in WSN. International Journal of Computer Applications, 105(9). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=5771aa21b2e151f3d93ba0a5f12d023a0bfcf28b

[6] Shahapure, K. R., & Nicholas, C. (2020, October). Cluster quality analysis using silhouette score. In 2020 IEEE 7th international conference on data science and advanced analytics (DSAA) (pp. 747-748). IEEE. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9260048

[7] Arthur, D., & Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Stanford. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9260048

[8] Yadav, J., & Sharma, M. (2013). A Review of K-mean Algorithm. Int. J. Eng. Trends Technol, 4(7), 2972-2976.

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=65f1232434c5eeddd9e658db7ae0dd5c47b6e20d

[9] Chen, P., Dong, L., & Liu, Y. (2024). Design of music recommendation system based on EDA and K-means cluster analysis. *Proceedings of the 4th International Conference on Signal Processing and Machine Learning*.
https://doi.org/10.54254/2755-2721/50/20241695