Final project
Project 20: Song recommender

Group 16:
Seniv Volodymyr 309358 Oleg Kim 309025

Task description:

Develop a song recommender system from datasets of song attributes. Students may choose one or a combination of datasets from the references below. Feel free to choose any approach to develop a solution, however, it is encouraged to use more than one method for comparing performance.

The goal of this project is as follows:

-Perform exploratory data analysis (EDA) on the dataset to understand data distribution, statistical significance of each feature and observe trends. Develop hypotheses and reasoning that can help when building a model.

-Perform data preprocessing to prepare the data before feeding into the model, e.g.: feature cleaning, selection and rescale.-Split the dataset to create separate training and test datasets.

-Train and compare regression models (may use ML libraries such as Scikit-Learn). Evaluate the model's performance metric and assess the impact of preprocessing strategy (e.g., contribution of features).

Theoretical background:

Clustering model is the main concept around which are solution is based.

Changes from initial plan:

Initially we have chosen content-based filtering and matrix factorization to recommend songs to users. We concluded that it would be more effective to use clustering algorithms for the purpose of the task.

Clustering Model:

A clustering model is a type of unsupervised learning algorithm that aims to group similar data points together into clusters based on their inherent patterns or similarities. It does not rely on predefined labels or target variables but rather identifies patterns and structures in the data itself.

Algorithms:

<u>K-means clustering algorithm</u> – K-means clustering is an iterative algorithm used to partition a dataset into K clusters. It works by iteratively assigning data points to the nearest centroid and updating the centroids based on the mean of the assigned points. The algorithm requires specifying the number of clusters, K, in advance. It aims to minimize the sum of squared distances between data points and their assigned centroids. K-means clustering is computationally efficient and widely used for its simplicity and effectiveness in finding spherical-shaped clusters.

<u>Hierarchical clustering algorithm</u> – Hierarchical clustering, on the other hand, is an algorithm that creates a hierarchy of clusters. It starts with each data point as an individual cluster and gradually merges or splits clusters based on their similarity. The algorithm does not require specifying the number of clusters in advance and provides a dendrogram, a tree-like structure that visually represents the clustering hierarchy. Hierarchical clustering can be agglomerative (bottom-up) or divisive (top-down) in nature. Agglomerative clustering, which is commonly used, starts with individual data points as clusters and progressively merges the most similar clusters until all data points belong to a single cluster. Hierarchical clustering is advantageous when the underlying structure of the data is not well-defined or when exploring different levels of granularity in clustering.

Dataset Analysis

In the dataset we are provided with 16 columns where we have 14 of which are song attributes, 1 is a song name, 1 is for artist, and a "target" column which represents the preference of the author of the dataset, where "1" is preferred and "0" not preferred.

Continuous features: 'acousticness', 'danceability', 'duration_ms', 'energy', 'liveness', 'loudness', 'tempo', 'valence', 'speechiness', 'instrumentalness' **Discrete features:** 'key','mode','time_signature','target'

Here are the 14 track attributes:

Acousticness - A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic. (>= 0 ,<= 1)

Danceability - Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

Duration_ms - The duration of the track in milliseconds.

Energy - Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

Instrumentalness - Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

Key - The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1. (>= -1 , <= 11)

Liveness - Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

Loudness - The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.

Mode - Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

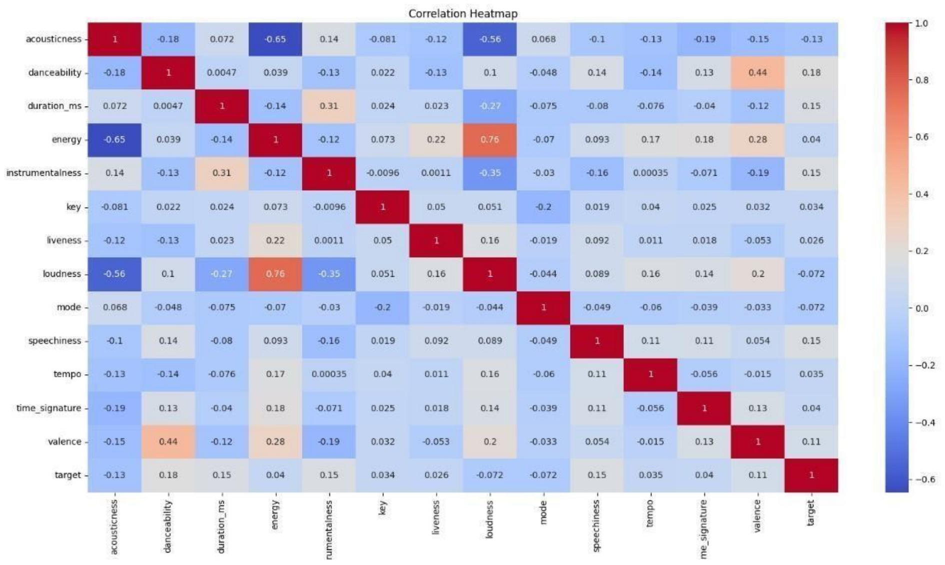
Speechiness - detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

Tempo – The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

Time_signature – An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4". (>= 3, <= 7)

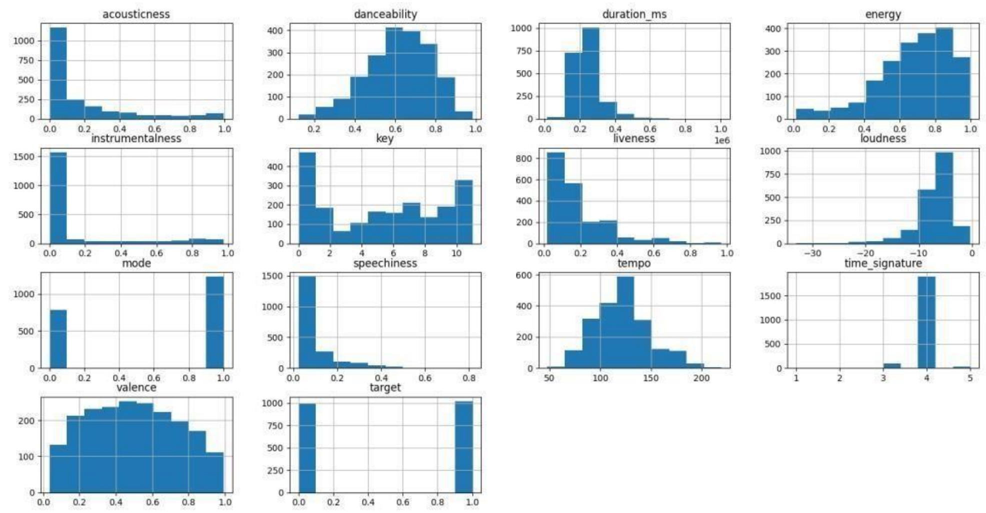
Valence - A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry). (>= 0, <= 1)

Heatmap:



Positive correlations are indicated by warm colors (reds) while negative correlations are indicated by cool colors (blues). The numerical values in the cells can also provide information about the magnitude of the correlations. For example, a value close to 1 or -1 indicates a strong correlation, while a value close to 0 indicates little to no correlation. Here we can see that energy and loudness have strong direct dependency, danceability and valence, instrumentalness and duration_ms, also have direct dependency while energy and acousticness, loudness and acousticness have strong inverse dependency.

Histogram:



All the graphs are asymmetrical but can be distributed by these categories:

Normal distribution: danceability, valence and tempo
Right-skewed distribution: liveness, acousticness, speechiness
Left-skewed distribution: energy, loudness
Symmetrical-bimodal distribution: target
Non-symmetrical-bimodal distribution: mode, time_sigbature, instrumentalness, key <u>df.isna().sum():</u>

Acousticness	0
danceability	0
Duration_ms	0
Energy	0
instrumentalness	0
key	0
liveness	0
loudness	0
mode	0
speechiness	0
tempo	0
Time_signature	0
valence	0
target	0

Song_title	0
artist	0

dtype: int64

df.info():

#	Column	Non-Null count	Dtype
0	Acousticness	2017 non-null	Float64
1	danceability	2017 non-null	Float64
2	Duration_ms	2017 non-null	Int64
3	Energy	2017 non-null	Float64
4	instrumentalness	2017 non-null	Float64
5	key	2017 non-null	Int64
6	liveness	2017 non-null	Float64
7	loudness	2017 non-null	Float64
8	mode	2017 non-null	Int64
9	speechiness	2017 non-null	Float64
10	tempo	2017 non-null	Float64
11	Time_signature	2017 non-null	Float64
12	valence	2017 non-null	Float64
13	target	2017 non-null	Int64
14	Song_title	2017 non-null	Object
15	artist	2017 non-null	Object

Dtypes: float64(10), int64(4), object(2)

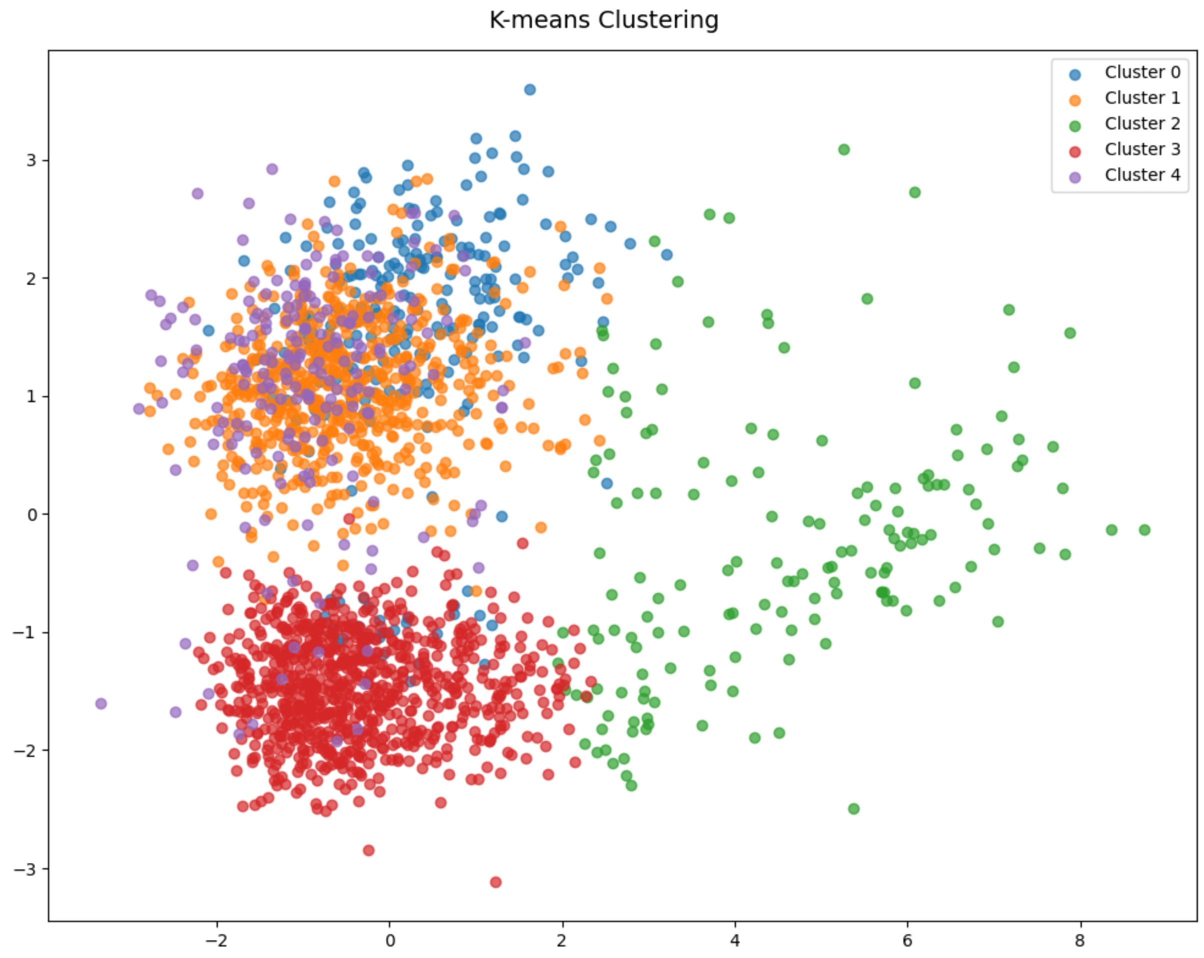
Memory usage: 252.2+ KB df.describe():

	mean	std	min	25%	50%	75%	max
acousticness	0.187590	0.259989	0.000003	0.009630	0.063300	0.265000	0.995000
danceability	0.618422	0.161029	0.122000	0.514000	0.631000	0.738000	0.984000
duration_ms	2.463062e +05	8.198181e +04	1.604200e +04	2.000150e +05	2.292610e +05	2.703330e +05	1.004627e +06
energy	0.681577	0.210273	0.014800	0.563000	0.715000	0.846000	0.998000
instrumental ness	0.133286	0.273162	0.000000	0.000000	0.000076	0.054000	0.976000
key	5.342588	3.648240	0.000000	2.000000	6.000000	9.000000	11.000000
liveness	0.190844	0.155453	0.018800	0.092300	0.127000	0.247000	0.969000

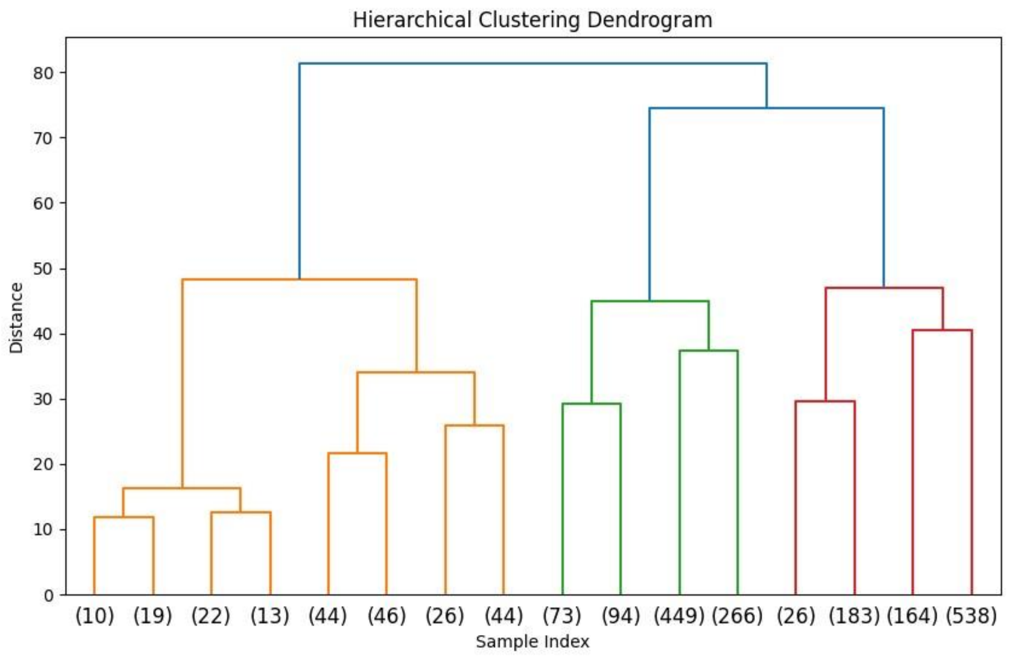
loudness	-7.085624	3.761684	-33.097000	-8.394000	-6.248000	-4.746000	-0.307000
mode	0.612295	0.487347	0.000000	0.000000	1.000000	1.000000	1.000000
speechiness	0.092664	0.089931	0.023100	0.037500	0.054900	0.108000	0.816000
tempo	121.603272	26.685604	47.859000	100.189000	121.427000	137.849000	219.331000
time_signature	3.968270	0.255853	1.000000	4.000000	4.000000	4.000000	5.000000
valence	0.496815	0.247195	0.034800	0.295000	0.492000	0.691000	0.992000
target	0.505702	0.500091	0.000000	0.000000	1.000000	1.000000	1.000000

Scatter plots:

K-means scatter plot:



Hierarchical dendrogram:



From these plots we can see how the algorithms group the data, where groups are represented by huge number of dots of different colors for k-means and the colored groups in a case for dendrogram.

Important: The dendrogram represents the hierarchical clustering structure by showing the merging of clusters at different levels. The height of each branch in the dendrogram corresponds to the distance or dissimilarity between the clusters being merged.

The specific method used for calculating the linkage in the current implementation is "ward," which employs the Ward's minimum variance method. Ward's method minimizes the total within-cluster variance when merging clusters. It does not directly consider the desired number of clusters but focuses on optimizing the variance-based criterion.

Steps of Implementation:

The whole project will be implemented on python.

Step 1) Data preprocessing:

- Read the Spotify dataset into a pandas DataFrame.
- Preprocess the data by normalizing or scaling the song attribute values. You can use the StandardScaler from the sklearn.preprocessing module to standardize the numerical attributes.

Step 2) Dataset splitting:

- Split the preprocessed dataset into a training set and a validation set using the train_test_split function from the sklearn.model_selection module.
- Use an 80-20 split, where 80% of the data is used for training and 20% for validation.

Step 3) K-means clustering

- Apply K-means clustering on the training dataset using the KMeans class from the sklearn.cluster module.
- Specify the desired number of clusters, K, and fit the K-means model to the training data.
- Obtain the cluster labels for the training and validation data using the predict method of the trained K-means model.

Step 4) Hierarchical clustering

- Apply Hierarchical clustering on the training dataset using the AgglomerativeClustering class from the sklearn.cluster module.
- Fit the Hierarchical clustering model to the training data.
- Obtain the cluster labels for the training and validation data using the fit_predict method of the trained Hierarchical clustering model.

Step 5) Silhouette Score calculation (Evaluation method):

- Calculate the silhouette score for both K-means and Hierarchical clustering using the silhouette_score() function from the sklearn.metrics module.
- Pass the feature matrix (song attributes) and the corresponding cluster labels (obtained from K-means and Hierarchical clustering) to the silhouette_score function.

Step 6) Euclidean distance (Evaluation method):

- Calculate the Euclidean distance for both K-means and Hierarchical clustering using the pairwise_distances() function from the sklearn.metrics module.
- Pass the feature matrix (song attributes) and the corresponding cluster labels (obtained from K-means and Hierarchical clustering) to the pairwise_distances() function. • Calculate the average distance

Step 7) Visual Inspection (Evaluation method):

- Plot the resulting clusters using data visualization libraries like Matplotlib or Seaborn.
- Use scatter plots or other visualization techniques to visually inspect the clusters and assess their quality, coherence, and separation.

How program works:

Brief program flow:

1. Program must work in the following way:
2. The user enters the song he prefers (this song should be from the data set we are working with).
3. User should also input the name of the algorithm he wants to use.
4. The data set file is opened and stored in the variable.

5. The program performs the appropriate preprocessing of the data before building a model.
6. The evaluation metric is calculated for the chosen algorithm and then displayed in the console.
7. Then the clustering is applied on the data set. The clustering type depends on the type of algorithm chosen.
8. Based on the clustering data, the model is build.
9. The array of ten songs is returned to the user and printed in the console.

Functions description:

1. preprocessing_the_data(df: pd.DataFrame) -> np.ndarray: This function takes a pandas DataFrame (df) containing song attributes as input. It preprocesses the data by normalizing or scaling the attribute values using the StandardScaler from scikit-learn. The function returns the preprocessed attributes as a NumPy array.
2. kmeans_alg(attributes_scaled: np.ndarray) -> np.ndarray: This function applies K-means clustering on the preprocessed attributes. It creates a KMeans object with specified parameters, such as the number of clusters (n_clusters) and random state. The function fits the K-means model to the scaled attributes and returns the cluster labels and cluster centers as NumPy arrays.
3. hierarchical_alg(attributes_scaled: np.ndarray) -> np.ndarray: This function applies Hierarchical clustering on the preprocessed attributes using the AgglomerativeClustering class from scikit-learn. It specifies the number of clusters and fits the hierarchical model to the scaled attributes. The function returns the cluster labels as a NumPy array.
4. calculate_silhouette_score(attributes_scaled: np.ndarray, labels: np.ndarray) -> float: This function calculates the silhouette score for a given clustering result. It takes the scaled attributes and cluster labels as input and uses the silhouette_score function from scikit-learn to compute the score. The function returns the silhouette score as a float value.
5. calculate_euclidean_distance(attributes_scaled: np.ndarray, labels: np.ndarray, cluster_centers: np.ndarray) -> float: This function calculates the average Euclidean distance between samples and their cluster centers. It takes the scaled attributes, cluster labels, and cluster centers as input. The function uses the pairwise_distances function from scikit-learn to compute the distances and returns the average

   recommend_songs(input_song: str, labels: np.ndarray, algorithm: str, num_recommendations: int) -> List[str]: This function recommends songs based on an input song. It takes the input song name, clust
   distance as a float value.
6.

   labels, clustering algorithm, and the number of recommendations as input. The function finds the cluster label for the input song, filters songs with the same cluster label, and sorts them based on some criteria (e.g., popularity, similarity). It returns a list of recommended songs.

7. plot_kmeans(attributes_scaled: np.ndarray, kmeans_labels: np.ndarray, cluster_centers: np.ndarray, attribute_names: List[str]): This function plots the results of K-means clustering. It takes the scaled attributes, K-means cluster labels, cluster centers, and attribute names as input. It creates a scatter plot where each data point represents a song attribute. The clusters are visualized by different colors, and the cluster centers are marked with red crosses.
8. plot_hierarchical(attributes_scaled: np.ndarray, hierarchical_labels: np.ndarray): This function plots the results of Hierarchical clustering using a dendrogram. It takes the scaled attributes and hierarchical cluster labels as input. It uses the linkage and dendrogram functions from the scipy.cluster.hierarchy module to generate the dendrogram plot.

Libraries:

NumPy: NumPy is a popular library for scientific computing in Python and is often used for matrix factorization algorithms.

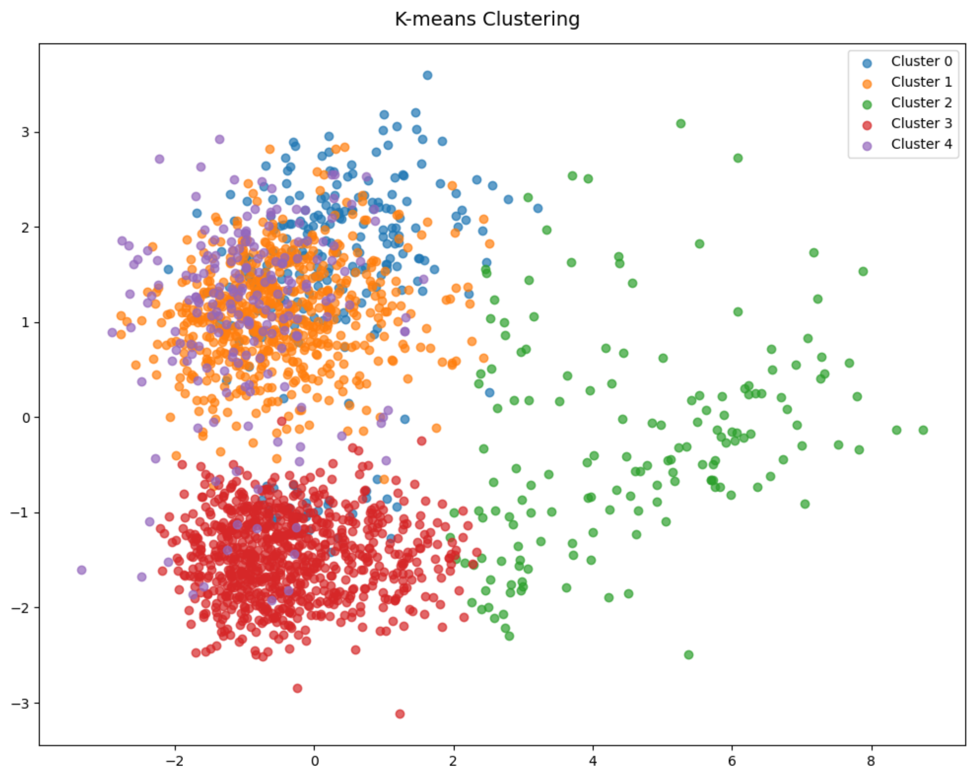
Pandas: pandas is a powerful data manipulation library that is widely used in Python for data analysis tasks, including data preprocessing for recommender systems. *scikit-learn:* scikitlearn is a popular machine learning library in Python that provides a range of algorithms for both supervised and unsupervised learning tasks, including matrix factorization.

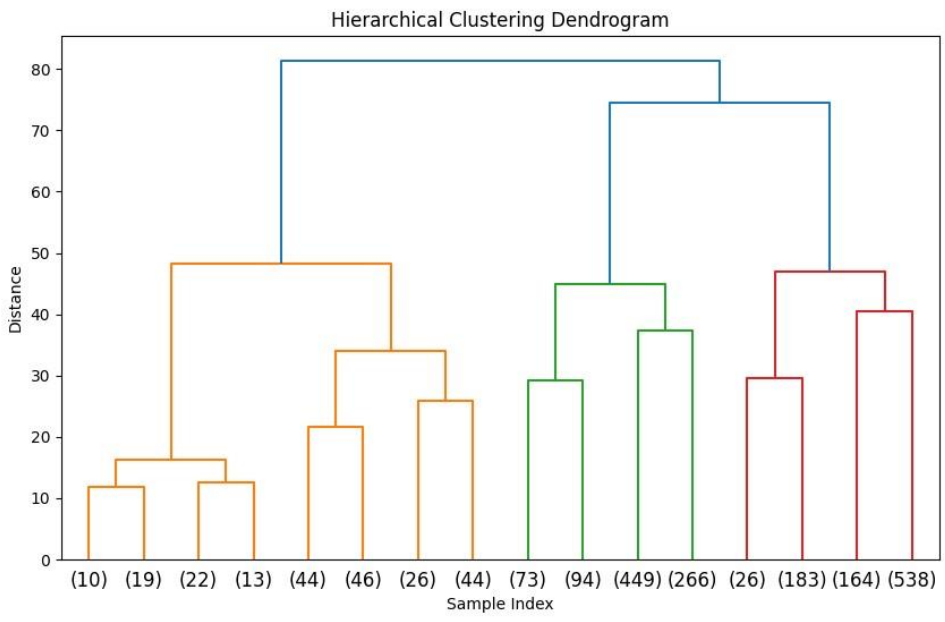
SciPy: is a powerful scientific computing library for Python. It provides a wide range of mathematical algorithms and functions for tasks such as numerical integration, optimization, signal processing, linear algebra, statistics, and more. SciPy is built on top of NumPy, another popular library for numerical computing in Python.

Test cases:

The number of clusters has fixed value of 5, since it is an optimal value for both algorithms. With this value both algorithms have best performance.(it will be seen in test cases)

1. For 14 attributes (all except song_title and artist) and 5 clusters:





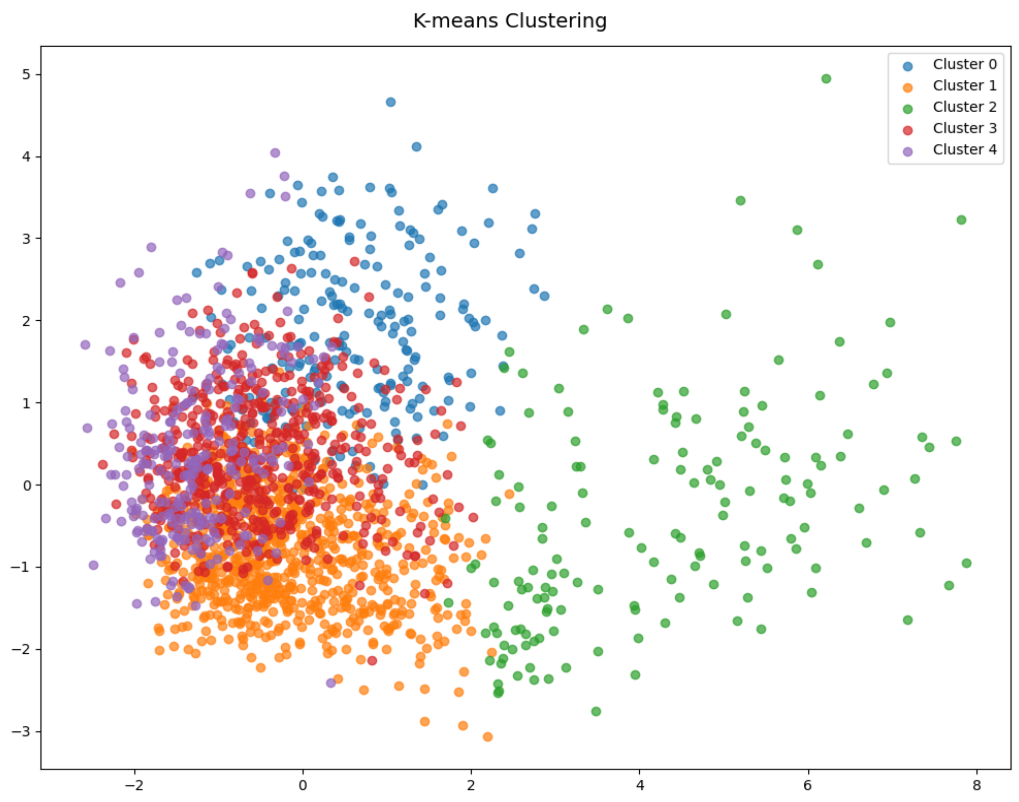
Enter a song name: ”Oh lala”

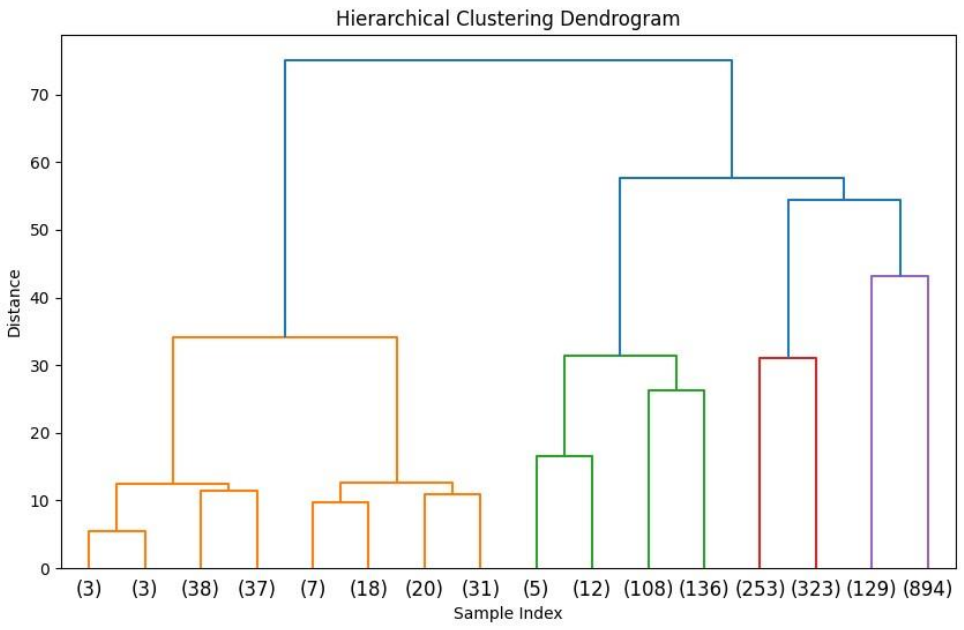
	K-means	Hierarchical
Silhouette score	0.147379856547866	0.13259244864596242
Euclidean distance	3.0588162450736642	6.041331170430666
Songs recommended	'Redbone' 'Sneakin’ 'Childs Play' 'Subways - In Flagranti Extended Edit' 'Donme Dolap - Baris K Edit' 'Cemalim' 'One Night' 'Char' 'World In Motion' 'One Nation Under a Groove'	'Redbone' 'Parallel Lines' 'Childs Play' 'Digital Animal' 'Donme Dolap - Baris K Edit' 'Cemalim' 'One Night' 'Char' 'World In Motion' 'One Nation Under a Groove'
User rating (subjective assesement)	10	10

Observation:

As can be observed from this experiment, the suggestions seems to be quite good and precise(subjectively and objectively)

2. For 9 attributes (acousticness, danceability, duration_ms, energy, instrumentalness, key, liveness, loudness, mode) and 5 clusters:





(The colors seen in the dendrogram are used to distinguish different merging events at various levels of the hierarchical tree but do not directly represent the clusters)

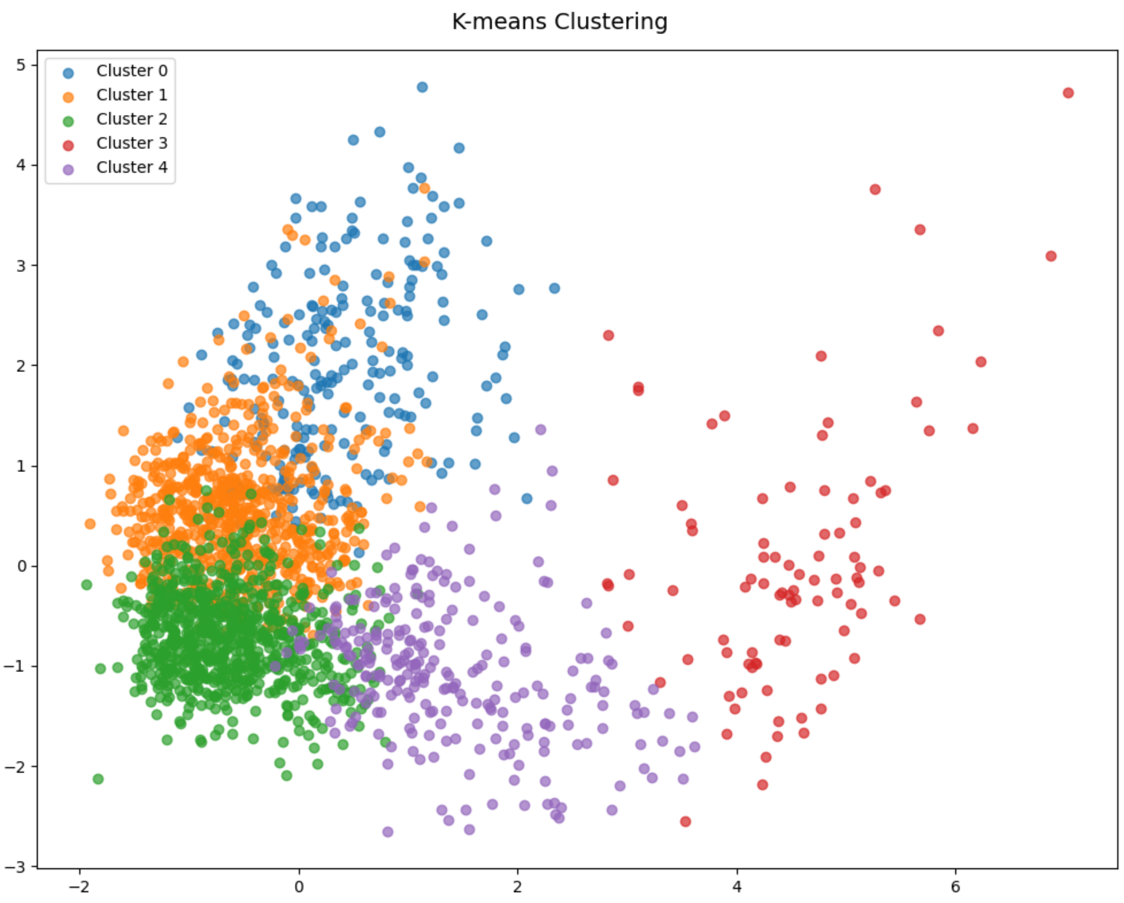
Enter a song name: "Oh lala"

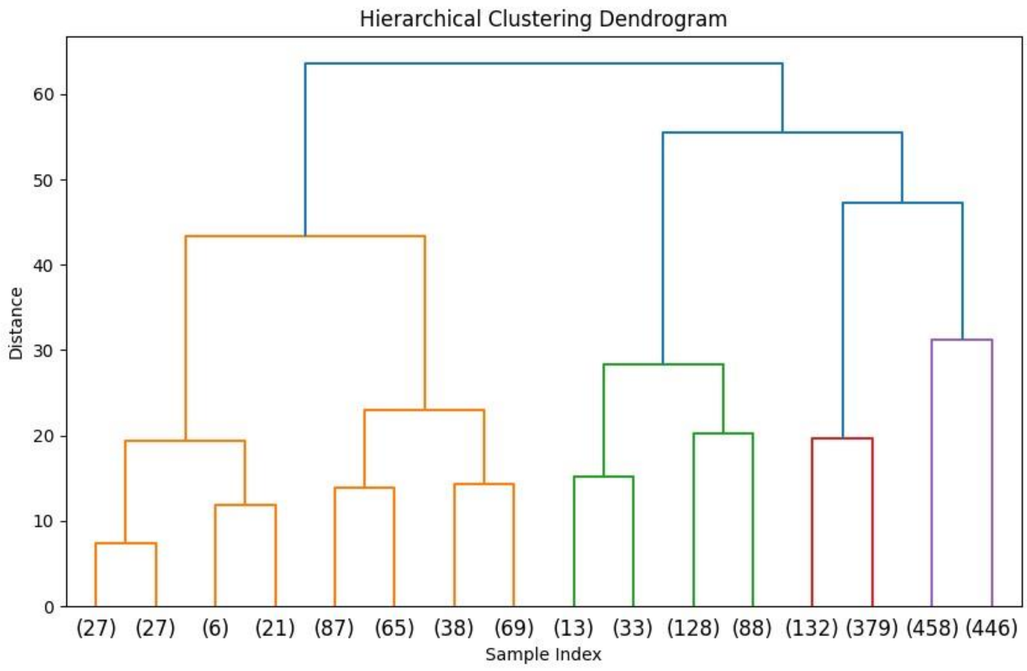
	K-means	Hierarchical
Silhouette score	0.1598402589594298	0.14503936951274463

Euclidean distance	2.3805277185414626	4.45150124881334
Songs recommended	"I've Seen Footage" 'Bouncin' 'C O O L - Radio Edit' 'Percolator (Jamie Jones Vault Mix) - mixed' 'Girlfriend' 'Lose My Mind' 'Look Alive' 'Check' '7/11' "Somebody's Watching Me"	"I've Seen Footage" 'One Nation Under a Groove' 'Bouncin' 'C O O L - Radio Edit' 'Percolator (Jamie Jones Vault Mix) - mixed' 'Lose My Mind' 'Look Alive' 'Check' '7/11' "Somebody's Watching Me"
User rating (subjective assesement)	6	5

Observation:
In this particular case the results are mediocre but not bad. (subjectively)
The silhouette score of both algorithms has better value than previously and Euclidean distance is smaller(objectively)

3. For 5 attributes (acousticness, danceability, duration_ms, energy, instrumentalness) and 5 clusters:





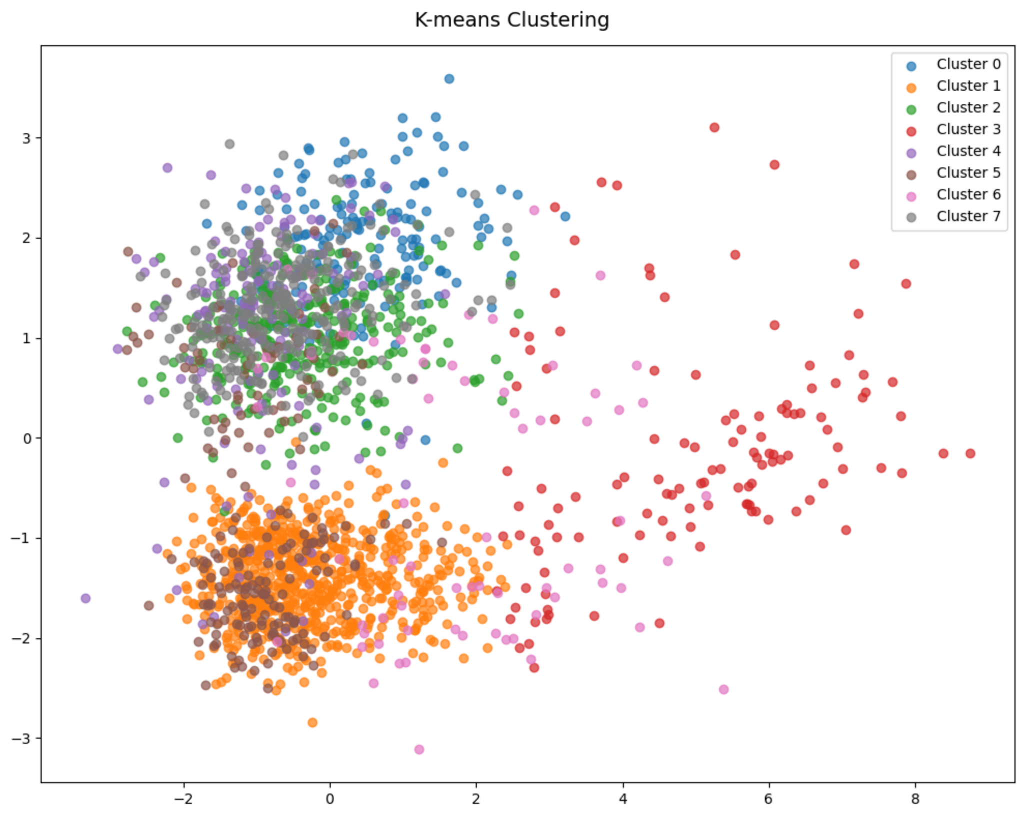
(The colors seen in the dendrogram are used to distinguish different merging events at various levels of the hierarchical tree but do not directly represent the clusters)

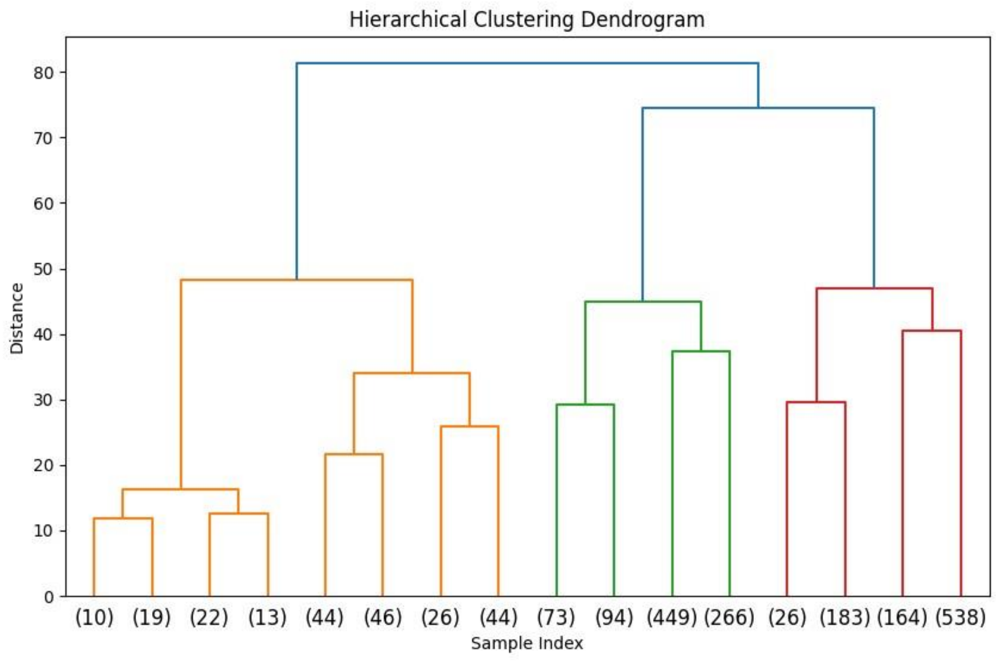
Enter a song name: “Oh lala”

	K-means	Hierarchical
Silhouette score	0.23461484391493775	0.19759835140213036
Euclidean distance	1.5830572145015152	3.768162204229671
Songs recommended	'Mask Off' 'Redbone' 'Xanny Family' 'Sneakin’' 'Childs Play' "I've Seen Footage" 'Digital Animal' 'Subways - In Flagranti Extended Edit' 'Donme Dolap - Baris K Edit' 'Cemalim'	'Mask Off', 'Redbone' 'Xanny Family' 'Sneakin’' 'Childs Play' "I've Seen Footage" 'Digital Animal' 'Subways - In Flagranti Extended Edit' 'Donme Dolap - Baris K Edit' 'Char'
User rating (subjective assesement)	7	8

Observation:
Surprisingly results for both algorithms are not so bad. (subjectively)
The silhouette for both algorithms score has best value among all cases and Euclidean distance is the smallest.(objectively)
It is because of the fact that the data gets smaller and is easier to be clustered and analyzed.

4. For 14 attributes (acousticness, danceability, duration_ms, energy, instrumentalness) and 8 clusters:





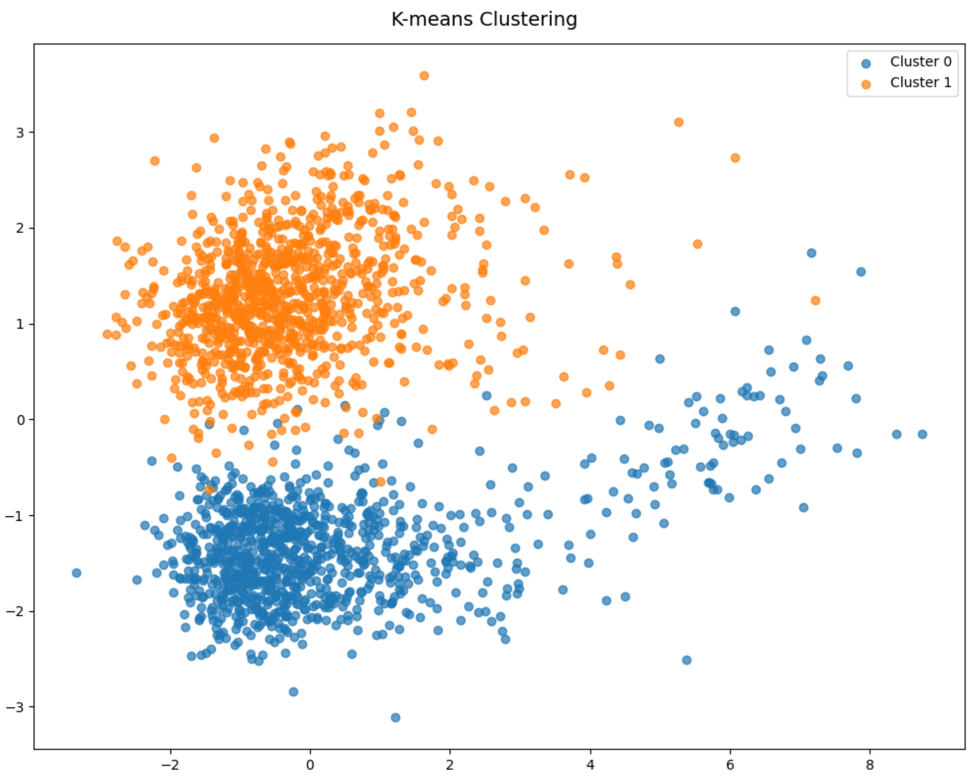
(The colors seen in the dendrogram are used to distinguish different merging events at various levels of the hierarchical tree but do not directly represent the clusters)

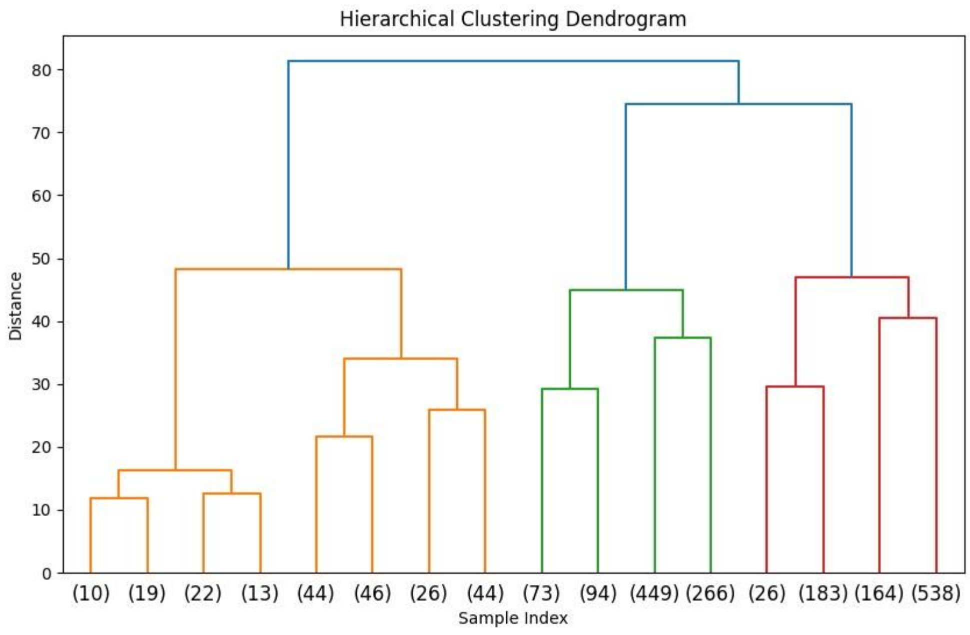
Enter a song name: “Oh lala”

	K-means	Hierarchical
Silhouette score	0.12898038413767765	0.09356861324048693
Euclidean distance	2.8773737263030137	5.774311023851943
Songs recommended	'One Nation Under a Groove', 'C O O L - Radio Edit' 'Percolator (Jamie Jones Vault Mix) - mixed' 'Girlfriend' 'Look Alive' '7/11' "Somebody's Watching Me" 'Versace Python' 'Monster' 'The Chase'	'Redbone' 'Parallel Lines' 'Childs Play' 'Digital Animal' 'Donme Dolap - Baris K Edit' 'Cemalim' 'One Night' 'Char' 'World In Motion' 'One Nation Under a Groove'
User rating (subjective assesement)	7	3

Observation:
Both algorithms this time performed differently. (subjectively)
The silhouette score worsened for both algorithms as well as Euclidean distance.(objectively)

5. For 14 attributes (acousticness, danceability, duration_ms, energy, instrumentalness) and 2 clusters:





(The colors seen in the dendrogram are used to distinguish different merging events at various levels of the hierarchical tree but do not directly represent the clusters) Enter a song name: “Oh lala”

	K-means	Hierarchical
Silhouette score	0.14126556266429527	0.27665089133355786
Euclidean distance	3.4044165504208723	6.1061133332779365
Songs recommended	'Mask Off' 'Redbone' 'Xanny Family' 'Master Of None' 'Parallel Lines' 'Sneakin’' 'Childs Play' 'Gyöngyhajú lány' "I've Seen Footage" 'Digital Animal'	'Mask Off', 'Redbone', 'Xanny Family', 'Parallel Lines', 'Sneakin’', 'Childs Play', "I've Seen Footage", 'Digital Animal', 'Donme Dolap - Baris K Edit', 'Cemalim'
User rating (subjective assesement)	8	7

Observation:
Surprisingly both algorithms performed quite well. (subjectively)
It can be observed that the silhouette score worsened for Hierarchical algorithm and became better for K-means algorithm as well as Euclidean distance.(objectively)

Summary:

Attributes/clusters	K-means			Hierarchical		
	Silhouette	Euclidean	R	Silhouette	Euclidean	R
14/5	0.147379856547866	3.0588162450736642	10	0.13259244864596242	6.041331170430666	10
9/5	0.1598402589594298	2.3805277185414626	6	0.14503936951274463	4.45150124881334	5
5/5	0.23461484391493775	1.5830572145015152	7	0.19759835140213036	3.768162204229671	8
14/8	0.12898038413767765	2.8773737263030137	7	0.09356861324048693	5.774311023851943	3
14/2	0.14126556266429527	3.4044165504208723	8	0.27665089133355786	6.1061133332779365	7

Conclusion:

K-means Clustering:

K-means is computationally efficient and works well with large datasets. It assigns each song to a specific cluster, allowing for straightforward recommendations within each cluster.

Hierarchical Clustering:

Hierarchical clustering captures hierarchical relationships between songs, allowing for more nuanced recommendations. It can handle different shapes and sizes of clusters and does not require specifying the number of clusters upfront. It also provides a dendrogram visualization, which can aid in understanding the clustering structure.

Hierarchical clustering can be computationally intensive, especially for large datasets.