Spotify “Mood” Playlist

Sandhika Surya Ardianto (5025211022)
Muhammad Zikri Ramadhan (5025211085)

Case study:

Music has a wide variety of types, which can differ based on the genre, the mood, or even the audio characteristics itself. Nowadays, music platforms like Spotify provide not only the genre of a song but also various audio features such as danceability, energy, valence, acousticness, and many others that. These features can be retrieved through the Spotify Web API which enables deeper analysis of patterns within music collection based on quantitative attributes. One of the promising approaches is to apply clustering algorithms to group similar songs based on their underlying audio characteristics. This technique can assist in creating automated, mood-based, or theme-consistent playlists which can enhances user experience without solely relying on manual curation or predefined genres. We aim to develop a playlist recommendation based on those features using clustering algorithms such as KMeans, DBSCAN, and Gaussian Mixtures Models.

Dataset:

Dataset used in this project can be obtained from Kaggle - Spotify Audio Features Dataset, which contains 2 csv files collected in November 2018 and April 2019 from the public web API that Spotify provides. The one we used is from April 2019 which includes 130.000 tracks along with various audio features. The features of dataset mostly is numerical either with an continuous or ratio, while some categorical features that represented as strings. The distribution of features is presented below:

1. Numerical
    - acousticness : likelihood of the track being acoustic (0 to 1)
    - danceability : how suitable the track is for dancing (0 to 1)
    - duration_ms : length of the track in milliseconds
    - energy : perceived intensity and activity level of the track (0 to 1)
    - instrumentalness : likelihood of whether a track contains no vocals (0 to 1)
    - liveness : detects the presence of a live audience sound (0 to 1)
    - loudness : the overall loudness of a track in decibels (dB)
    - speechiness : measure of the presence of spoken words in the track (0 to 1)
    - tempo : the overall estimated tempo of a track in beats per minute (BPM)
    - time_signature : an estimated time signature or how many beats are in each bar
    - valence : the musical positiveness conveyed by a track which is racks with high valence sound more positive (happy), while tracks with low valence sound more negative (sad)
    - popularity : popularity score of the track

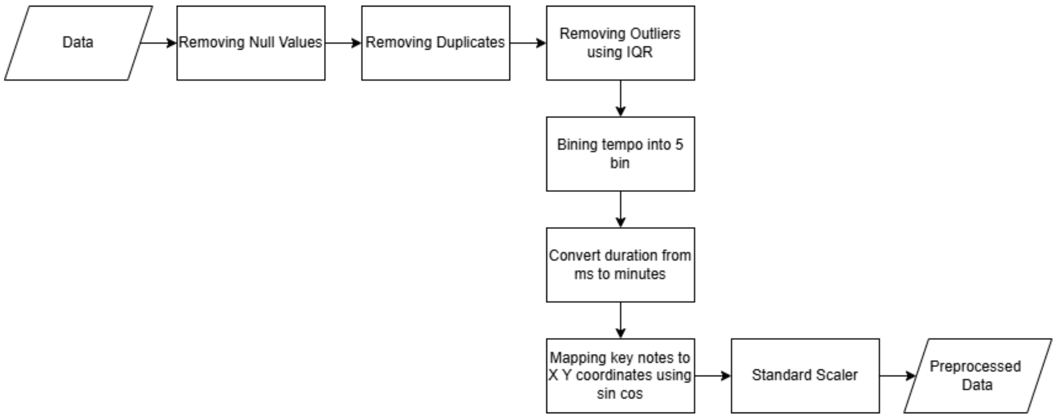
2. Categorical
    - track_id : unique identifier for each track in database
    - track_name : title of the song
    - artist_name : name of the performing artist or band
    - key : musical key of the track (integer from 0 to 11) like 0 = C, 1 = C♯/D♭, 2 = D
    - mode : the modality (major or minor) of a track which is major presented by 1 and minor is 0

Methodology:

1. Preprocessing

   For the first step, we performed EDA and preprocessing. EDA include a visualisation like shape of distribution using histogram for each numeric features, correlation matrix to measure covariance between numeric features, and box plot to display outliers and quartile range. We remove null values and duplicates from the record. Using lower bound as Q1 - 1,5 * IQR and upper bound as Q3 + 1,5 * IQR, we consider any data exceed that limits as outliers.

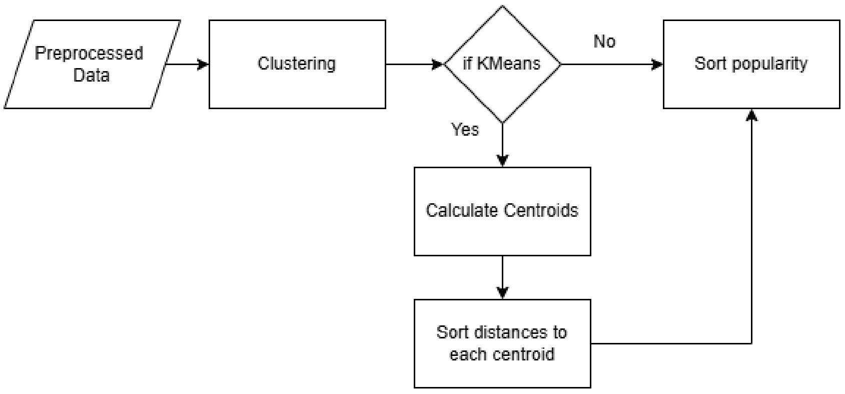
   After data cleaning, we would create some additional features like make tempo to more definite range by bining it into 5 bin, convert duration from ms to minutes by dividing it with 60000, and to make key notes more representative, mapping it to X and Y coordinates using sin and cos. Final step before data used for clustering, we would normalize it with Standard Scaler to make the distance between features comparable and avoid domination by features with larger numerical ranges. This step is important especially for algorithms like KMeans or DBSCAN that rely on distance calculation.


Data
Removing Null Values
Removing Duplicates
Removing Outliers using IQR
Bining tempo into 5 bin
Convert duration from ms to minutes
Mapping key notes to X Y coordinates using sin cos
Standard Scaler
Preprocessed Data

2. Clustering Method

   - KMeans is a clustering algorithm based on centroids which are calculated as the mean of the surrounding data points. The goal is to iteratively update the centroids' positions by recalculating the mean of each cluster. The process continues until the centroids no longer change significantly, at which point the iteration stops. In the first iteration, random data points are selected as the initial centroids, based on the desired number of clusters. This algorithm tends to form clusters with circular (spherical) shapes, which makes it struggle with overlapping data or clusters with irregular shapes.
   - DBSCAN is a clustering algorithm based on data density where data points that lie within a maximum distance parameter called *epsilon* (ε) are considered neighbors. A group of neighboring points are considered a cluster if it contains at least a minimum number of points. DBSCAN can identify points that lie alone in low-density regions as outliers or noise. However, this algorithm may perform poorly when dealing with datasets that have extremely varying densities because of setting the right ε and *MinPts* values is quite challenging.
   - GMM or Gaussian Mixtures Model is a clustering algorithm based on probability, where each data point is assigned to clusters based on likelihood. The algorithm starts by initializing the means and standard deviations through random sampling, along with weights representing the proportion of each cluster. During the E-step (Expectation step), the algorithm calculates the probability of each data point belonging to each

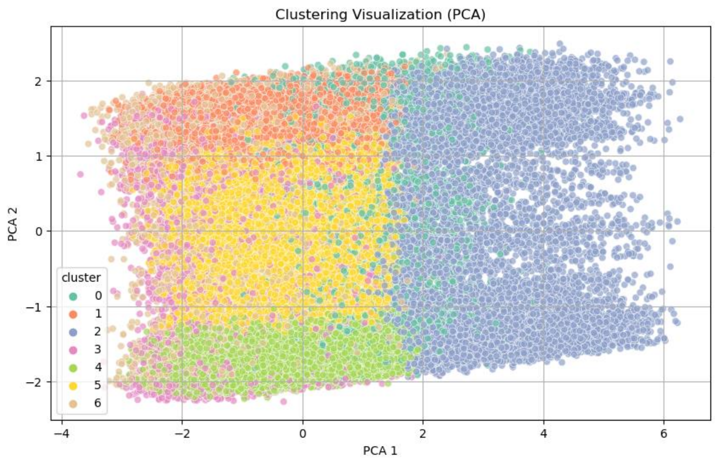
cluster using the Gaussian Probability Density Function (PDF) based on the mean, standard deviation, and weight. Each data point is assigned to the cluster with the highest likelihood from joint distribution. After the E-step, the M-step (Maximization step) updates the parameters (means, standard deviations, and weights) based on the calculated probabilities. This algorithm can handle overlapping data effectively because it is not limited to rigid cluster shapes, allowing data points in overlapping regions to be assigned appropriately based on the probabilistic model.

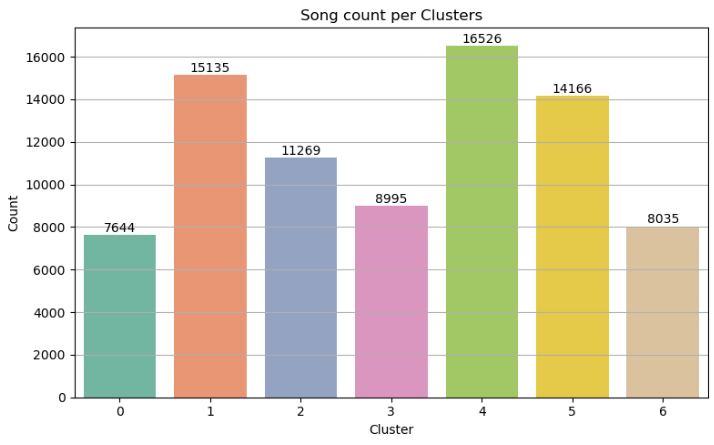


3. Clustering Results and Analysis

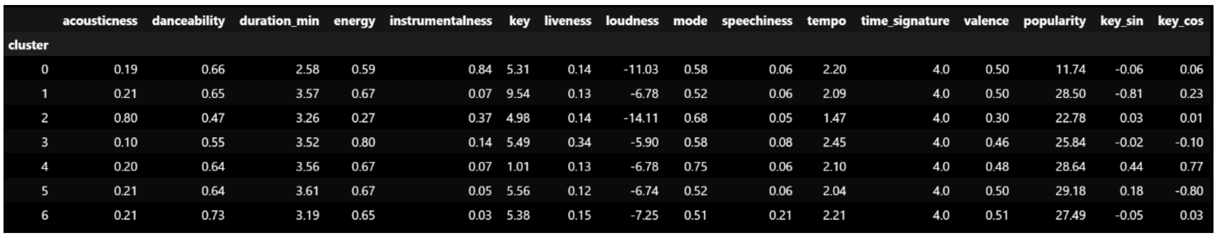
a. K-Means

We choose 7 as the number of the clusters. But it's all depends on how many playlists a user want to create. Here are the clusters and the distributions:





Once the clusters are created, we check the summary of each clusters. The way we do this is by taking the average of each numerical columns from each clusters. The summary can be seen in the picture below:



	acousticness	danceability	duration_min	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	popularity	key_sin	key_cos
cluster																
0	0.19	0.66	2.58	0.59	0.84	5.31	0.14	-11.03	0.58	0.06	2.20	4.0	0.50	11.74	-0.06	0.06
1	0.21	0.65	3.57	0.67	0.07	9.54	0.13	-6.78	0.52	0.06	2.09	4.0	0.50	28.50	-0.81	0.23
2	0.80	0.47	3.26	0.27	0.37	4.98	0.14	-14.11	0.68	0.05	1.47	4.0	0.30	22.78	0.03	0.01
3	0.10	0.55	3.52	0.80	0.14	5.49	0.34	-5.90	0.58	0.08	2.45	4.0	0.46	25.84	-0.02	-0.10
4	0.20	0.64	3.56	0.67	0.07	1.01	0.13	-6.78	0.75	0.06	2.10	4.0	0.48	28.64	0.44	0.77
5	0.21	0.64	3.61	0.67	0.05	5.56	0.12	-6.74	0.52	0.06	2.04	4.0	0.50	29.18	0.18	-0.80
6	0.21	0.73	3.19	0.65	0.03	5.38	0.15	-7.25	0.51	0.21	2.21	4.0	0.51	27.49	-0.05	0.03

We analyzed this summary to know the characteristic of each clusters, what makes them different from the other clusters. From here, we conclude that each clusters consists of these type of music:

1. Cluster 0 – Instumentals
   This cluster is the highest in instrumentalness (0,84) with low speechiness (0,06), showing that this cluster filled with instrumental tracks. Low loudness and moderate energy shows that most of the tracks are chill and mellow.
2. Cluster 1 – Mainstream Pop
   With low instrumentalness (0,07), moderately high popularity (28,50), and balanced features, this playlist is filled with vocal-heavy mainstream songs. Its easy-listening nature contributes to strong commercial appeal.
3. Cluster 2 – Soft Acoustic
   Highest in acousticness (0,80) and very low energy (0,27), indicate that the songs in this playlist is strongly acoustic and relaxing.
4. Cluster 3 – Energetic Pop-Pock
   This playlist filled with high energy and low acoustic songs, resulting in highest energy score (0,80) and lowest in acousticness (0,10). Together with high loudness (-5,90 dB), it shows that this playlist is loud and vibrant.
5. Cluster 4 – Upbeat Pop
   High mode (0,75) and key sin and cos suggesting major keys contributes to bright sound and cheerful playlist.
6. Cluster 5 – Balanced Pop
   This playlist broadly appealing and have no strong genre leaning. This was shown by highest popularity (29,18), balanced energy (0,67), moderate mode and speechiness with notable artist such as Selena Gomez and Charlie Puth.
7. Cluster 6 – Dancefloor, Groovy
   Suitable for dance party, this palylist have the highest danceability (0,73) and lowest instrumentalness (0,03). This means that the playlist mostlikely consist of dance tracks with vocals.

Lastly, we create 2 sets of playlist from those clusters with each playlist contains top 100 songs in each clusters. The first set were made by picking top 100 songs closest to the centroid of its cluster. This gives us very similar songs in each cluster eventhough many mediocre artis or un-popular songs appears in the playlist.

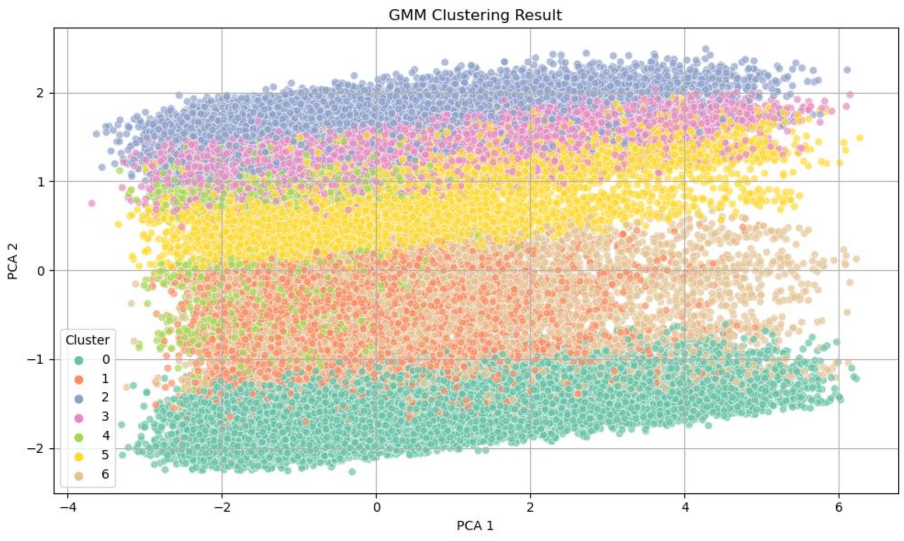
The second sets were made py choosing top 100 songs by popularity in each clusters. This method gives us much more related playlists with notable artist and songs in them. We can easily find well-known artist names like Dua Lipa and Black Pink, and also popular track such as Maze by Juice WRLD and I'll Never Love Again by Lady Gaga.

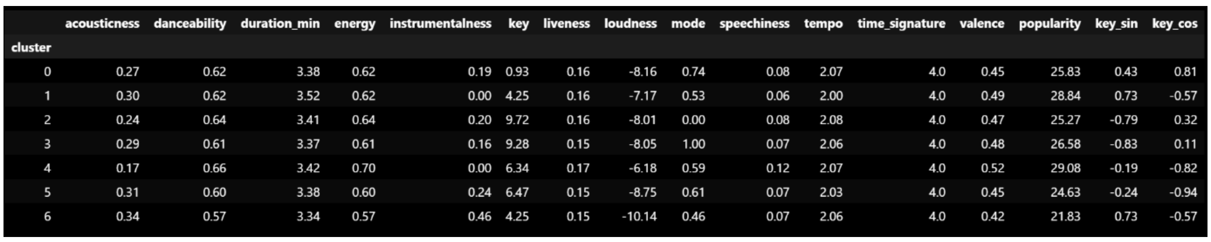
b. DBScan

With DBScan, we do not need to state the number of cluster that we were aiming to create. Instead, it will give us the best number of cluster according to our data. But with our data, the algorithm only produce one main cluster and one very small cluster for the outliars. This happens due to lack of clear density separation in our data. The datasets lacks dense regions that are clearly seperated one another by lower-density areas. Since DBSCAN relies heavily on such separation, it may treat all points as part of a single dense area or fail to find any dense core points at all.

c. Gaussian Mixture Model (GMM)

Similar to KMeans, we attempted to create 7 clusters using this method. Although the resulting clusters appear to be more clearly separated compared to KMeans, their characteristics are quite similar to one another. Some clusters differ significantly in one particular aspect, but remain very similar to the other six clusters in most other aspects. Even, some features do not show any distinct patterns across clusters, with values being relatively uniform for all playlists. The clustering results and summary of those playlists can be seen below:





cluster	acousticness	danceability	duration_min	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	popularity	key_sin	key_cos
0	0.27	0.62	3.38	0.62	0.19	0.93	0.16	-8.16	0.74	0.08	2.07	4.0	0.45	25.83	0.43	0.81
1	0.30	0.62	3.52	0.62	0.00	4.25	0.16	-7.17	0.53	0.06	2.00	4.0	0.49	28.84	0.73	-0.57
2	0.24	0.64	3.41	0.64	0.20	9.72	0.16	-8.01	0.00	0.08	2.08	4.0	0.47	25.27	-0.79	0.32
3	0.29	0.61	3.37	0.61	0.16	9.28	0.15	-8.05	1.00	0.07	2.06	4.0	0.48	26.58	-0.83	0.11
4	0.17	0.66	3.42	0.70	0.00	6.34	0.17	-6.18	0.59	0.12	2.07	4.0	0.52	29.08	-0.19	-0.82
5	0.31	0.60	3.38	0.60	0.24	6.47	0.15	-8.75	0.61	0.07	2.03	4.0	0.45	24.63	-0.24	-0.94
6	0.34	0.57	3.34	0.57	0.46	4.25	0.15	-10.14	0.46	0.07	2.06	4.0	0.42	21.83	0.73	-0.57

This is bad due to the intial problem that we want to address, which is creating playlist of songs. We want to make each playlist to be unique from the other, so the listener can have different set of songs for different mood and conditions.

4. Conclusions

From the analysis that we have done across those three methods, we conclude that Kmeans produced the best playlists for users. Although some clusters are overlapping, but the playlist created were different from one another. This covers the need of different palylist for different moods, conditions, and occassions