Spotify recommender system

Context

- Music, podcast and video service
- 400+ million monthly active users
- 50+ million tracks available
- Great discovery feature

Problem statement

Use Spotify’s API to create a 10-song playlist based on two ”seed” songs.

Description of data

Objective: gather a genre rich dataset → 27.859 unique tracks

Spotipy was used, which is a lightweight Python library for the Spotify Web API.

Genres:

- Country
- Pop
- Hip Hop
- R&B
- Jazz
- Blues
- Classical
- Latin
- Chill
- Workout
- Party
- Dance
- Disco

Data dictionary

Feature	Type	Description
id	string	Spotify track id
title	string	Track title
all_artists	string	Artist name
popularity	int	Score of tracks popularity. Ranges from 0-100, 100 being the most popular.
release_date	string	Track release date.
danceability	float	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.
energy	float	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
key	int	The key the track is in.
loudness	float	The overall loudness of a track in decibels (dB). Values typically range between -60 and 0 db.
mode	int	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.
accousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	float	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.
liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
valence	float	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
tempo	float	The overall estimated tempo of a track in beats per minute (BPM).
duration_ms	int	The duration of the track in milliseconds.
time_signature	int	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

Data cleaning and feature engineering

- Checked for missing values.
- Removed all rows with missing values.
- Checked and removed all duplicate tracks in the dataset.
- Created a column called 'release_year' where I only extracted the year of every value in the 'release_date’
- Removed the 'release_date' column
- Created a column called 'duration' where I transformed the duration in milliseconds to seconds
- Removed the 'duration_ms' column

EDA and visualization

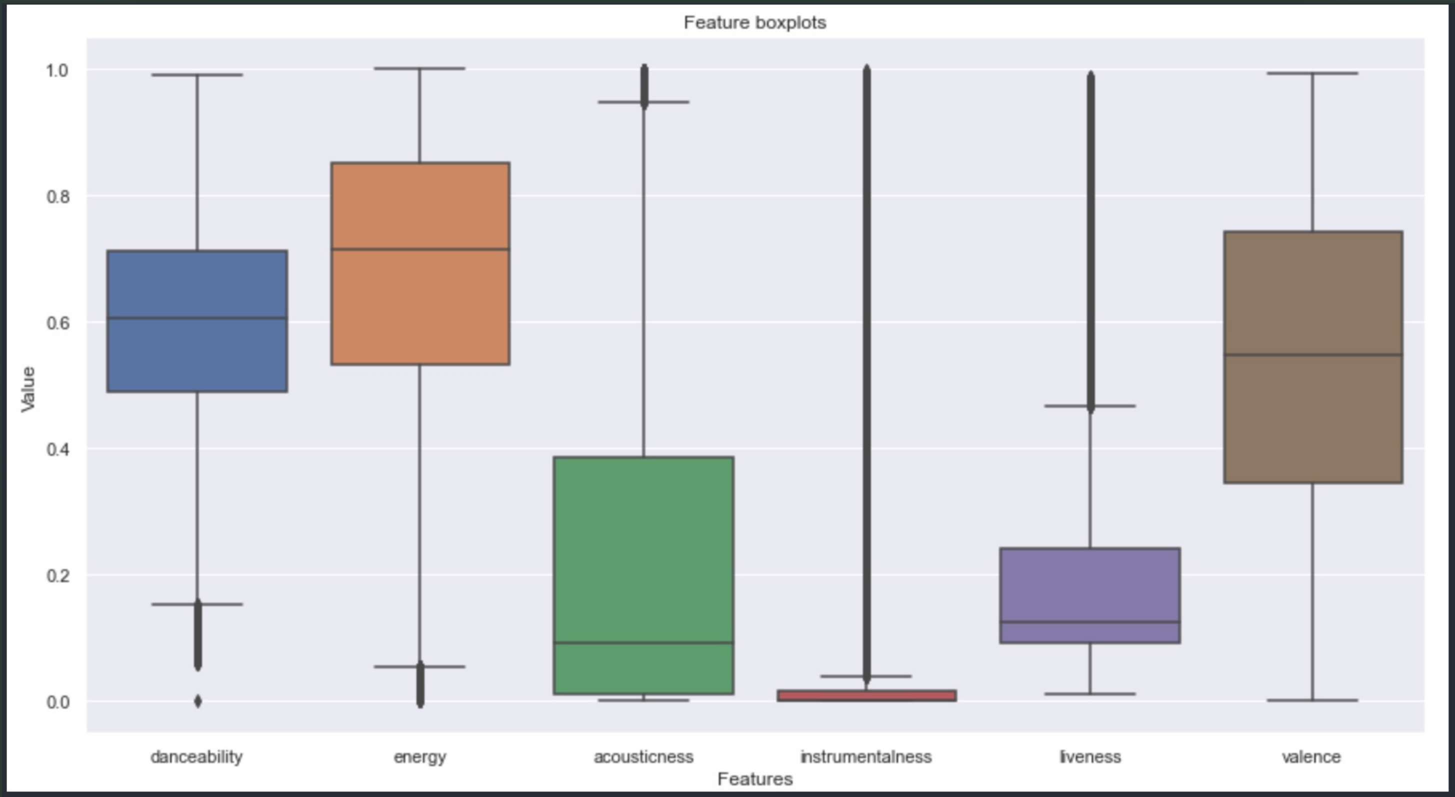
Wide variety of genres and artist is very important:

- **13.560 unique artists**
- **27.859 unique tracks**
- **No artist makes for more than 0.5% of the dataset**

Artist	#
Arctic Monkeys	113
Johnny Cash	62
Taylor Swift	61
The Strokes	49
Kanye West	46
Red Hot Chili Peppers	45
blink-182	45
Cage The Elephant	45
The Rolling Stones	45
Eminem	43

EDA and visualization

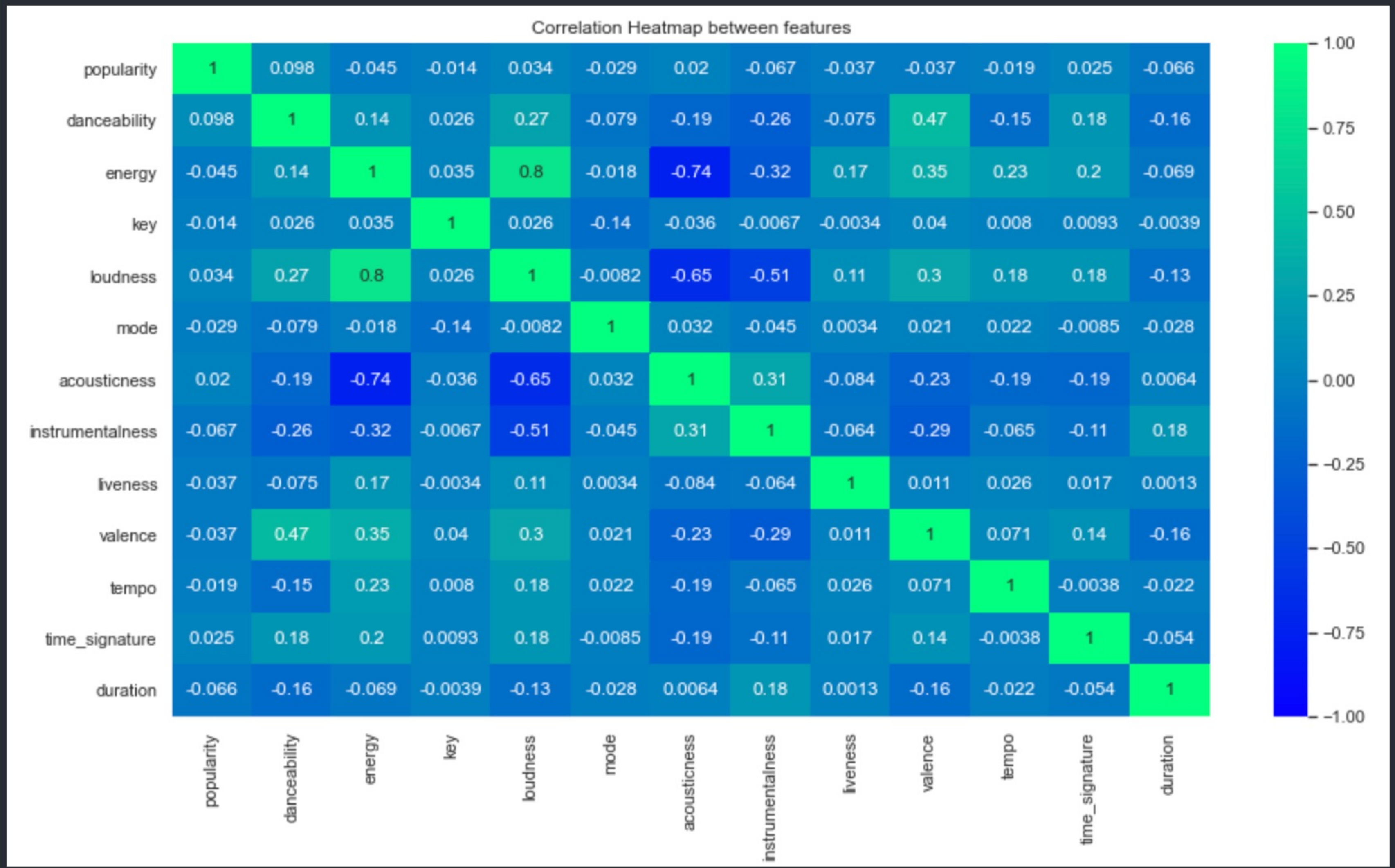
Boxplots for a couple of features to visualize some statistical metrics.



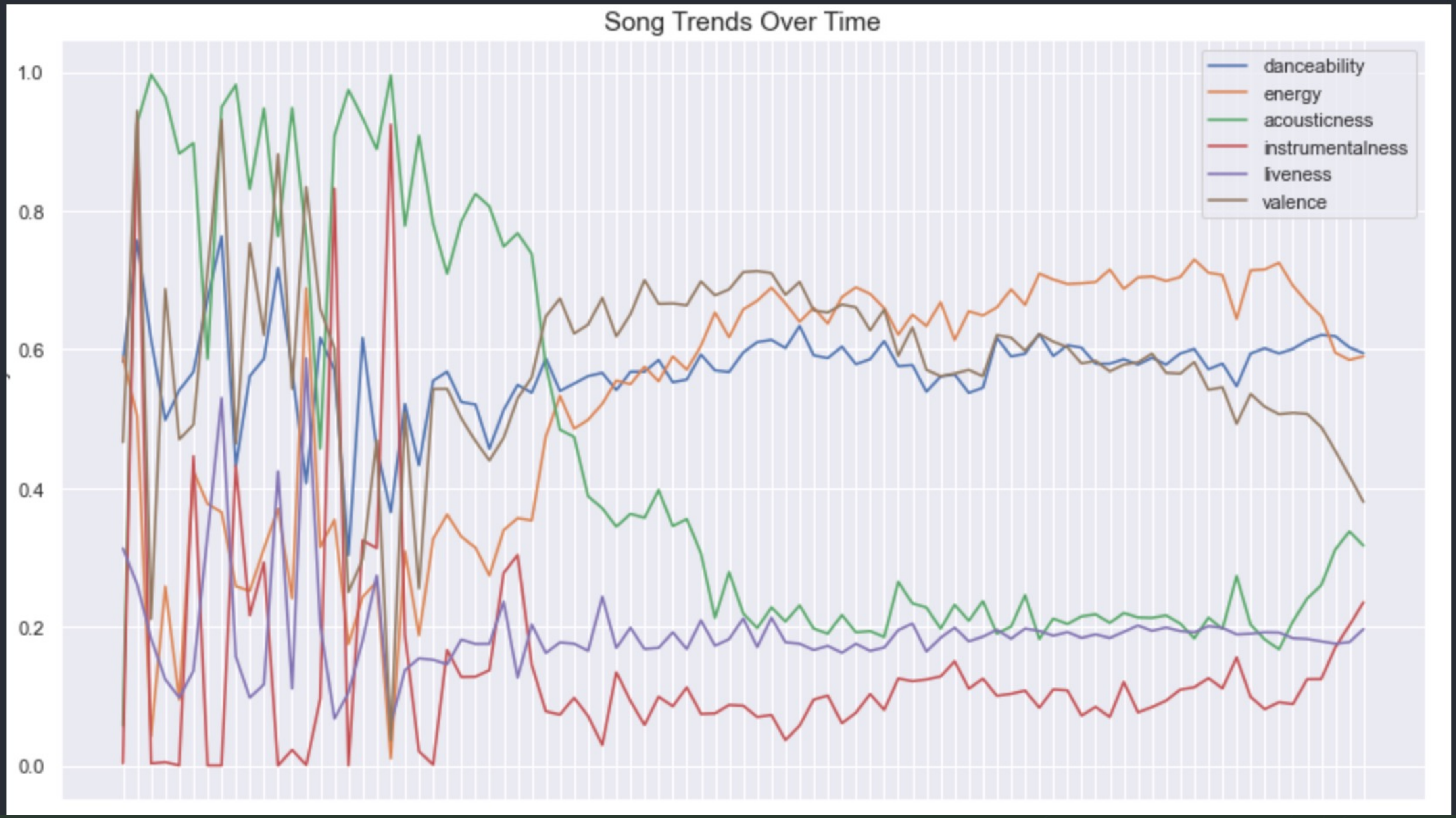
Relatively wide STD → Confirms various genres

EDA and visualization

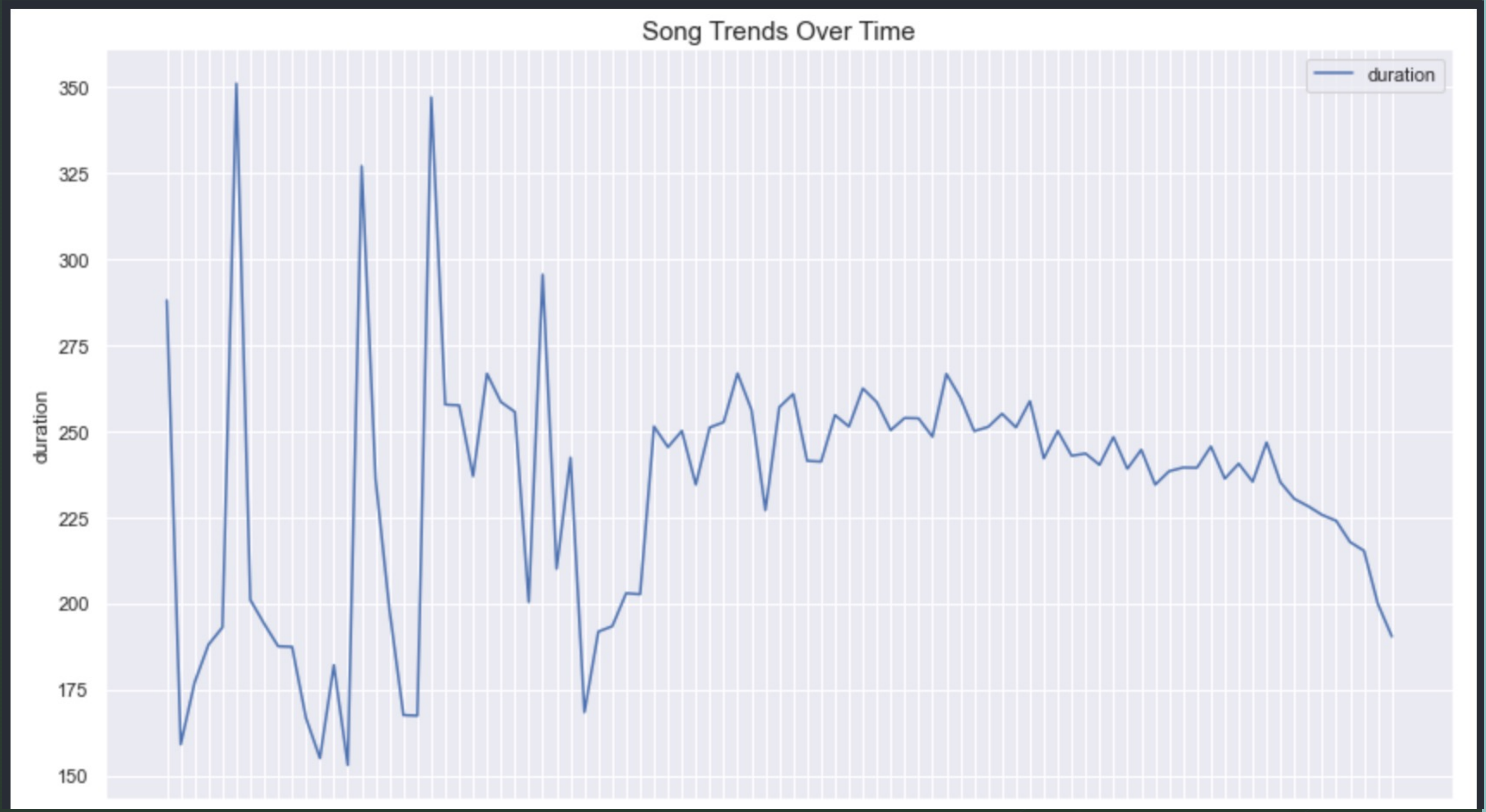
Correlation between audio features.



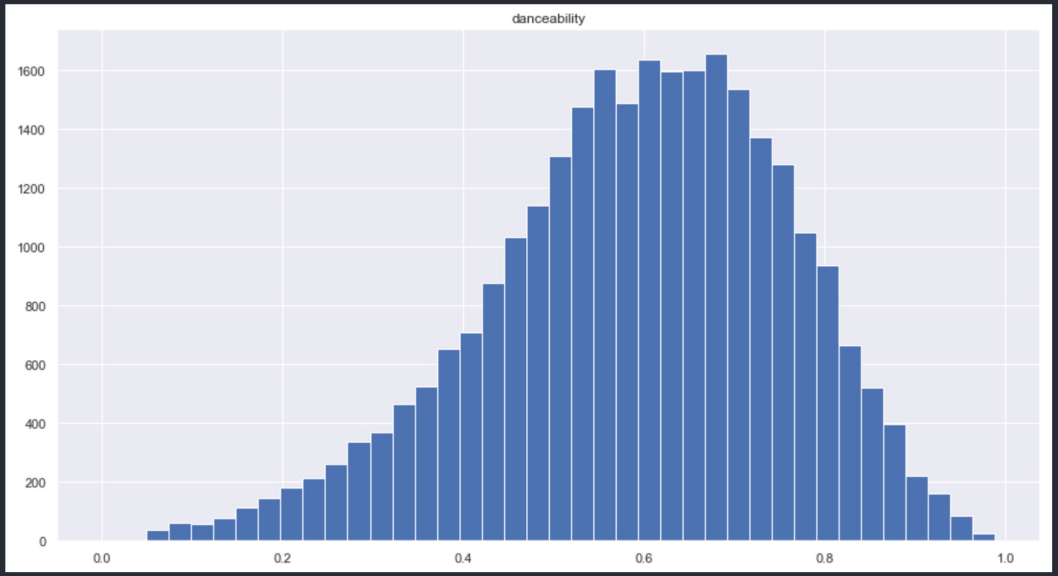
EDA and visualization

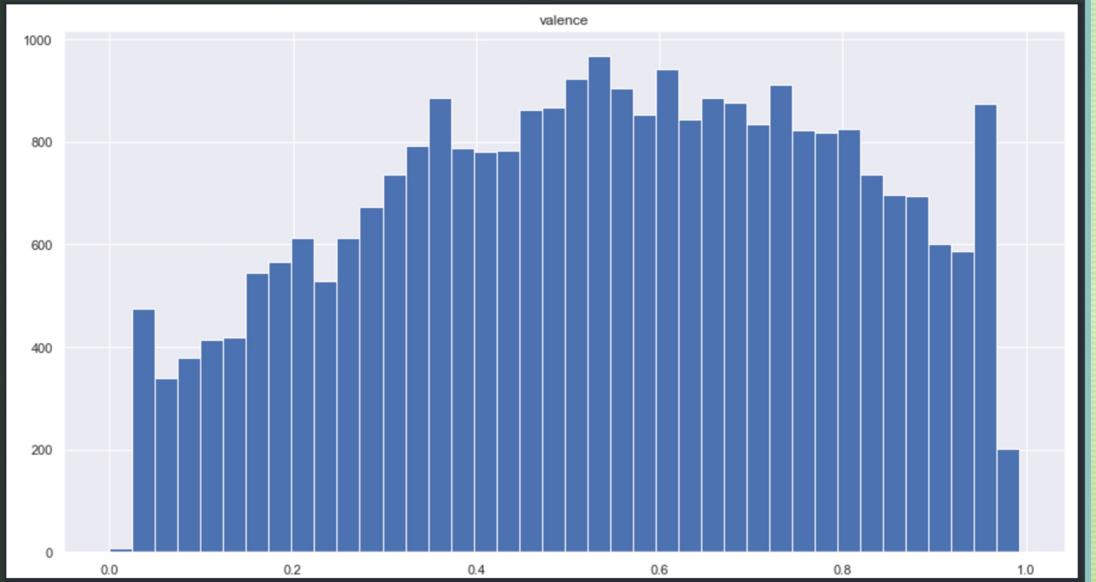


EDA and visualization



EDA and visualization





Model development

Data not labeled → Unsupervised approach

Two different types of clustering algorithms were chosen that can be split in 2 groups:

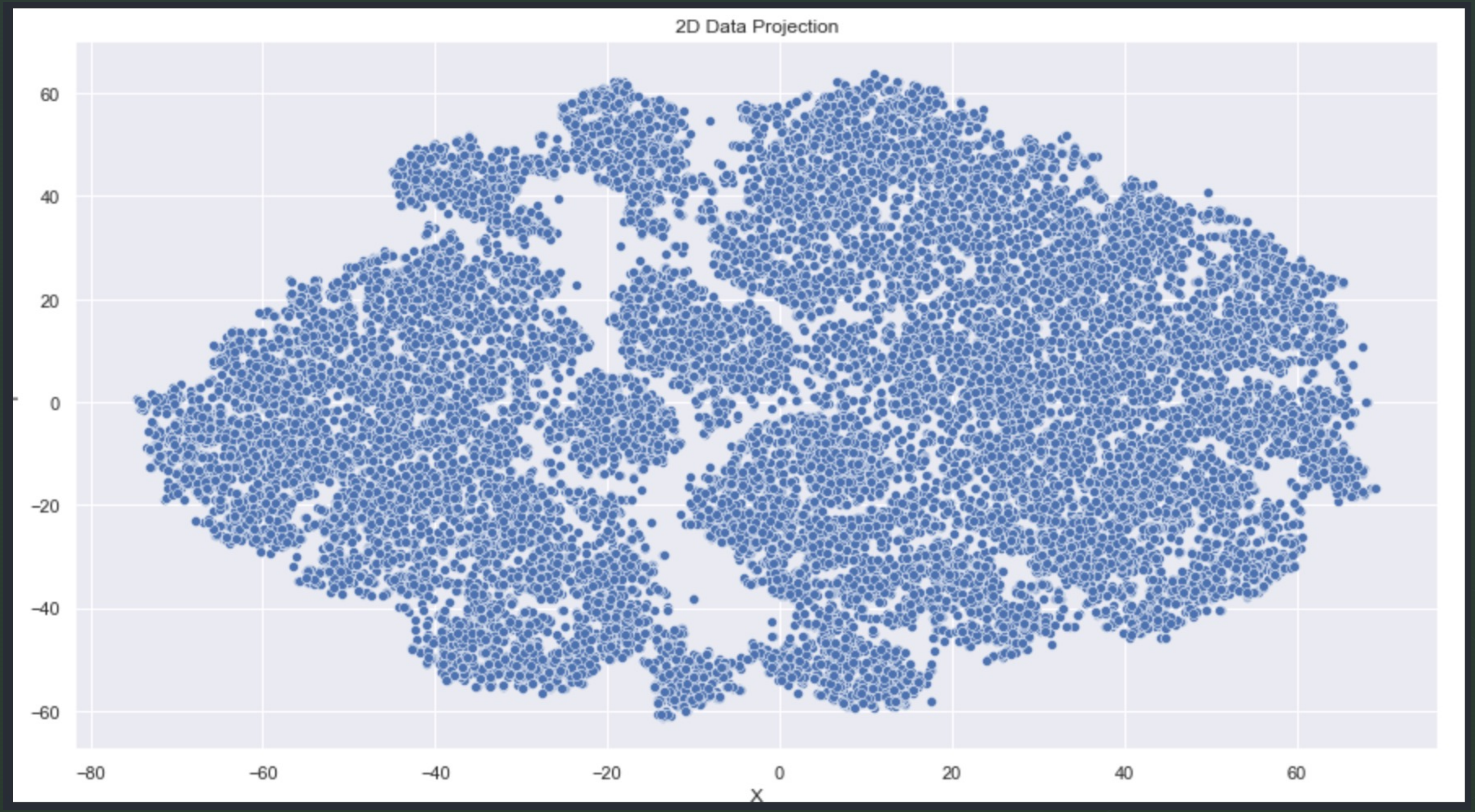
- Density model: DBSCAN
- Centroid based model: K-Means

Evaluation of the resulting clusters with 2 different metrics: Silhouette and Davis-Bouldin

The data is **scaled** before doing any processing.

Model development: data projection

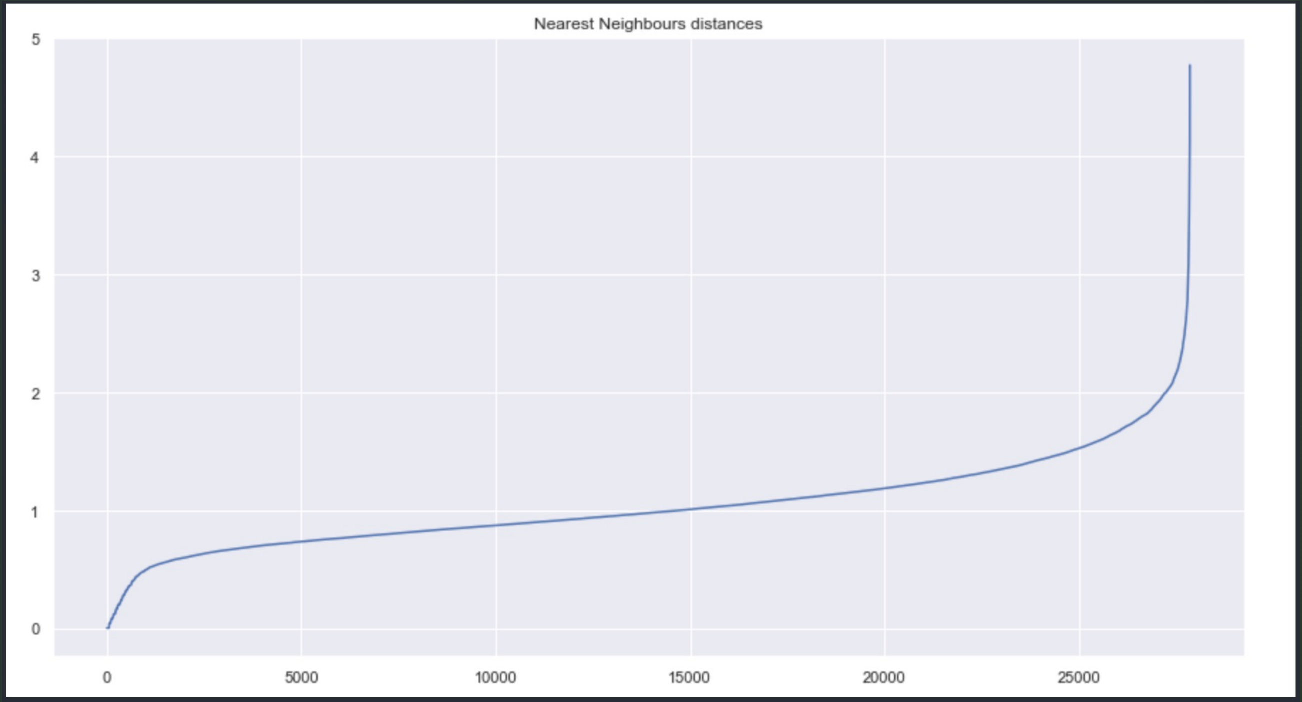
TSNE (T-distributed Stochastic Neighbor Embedding) is used to project the data in a 2D space. This big blob is a 2D representation of all the songs in the data set.



DBSCAN clustering algorithm

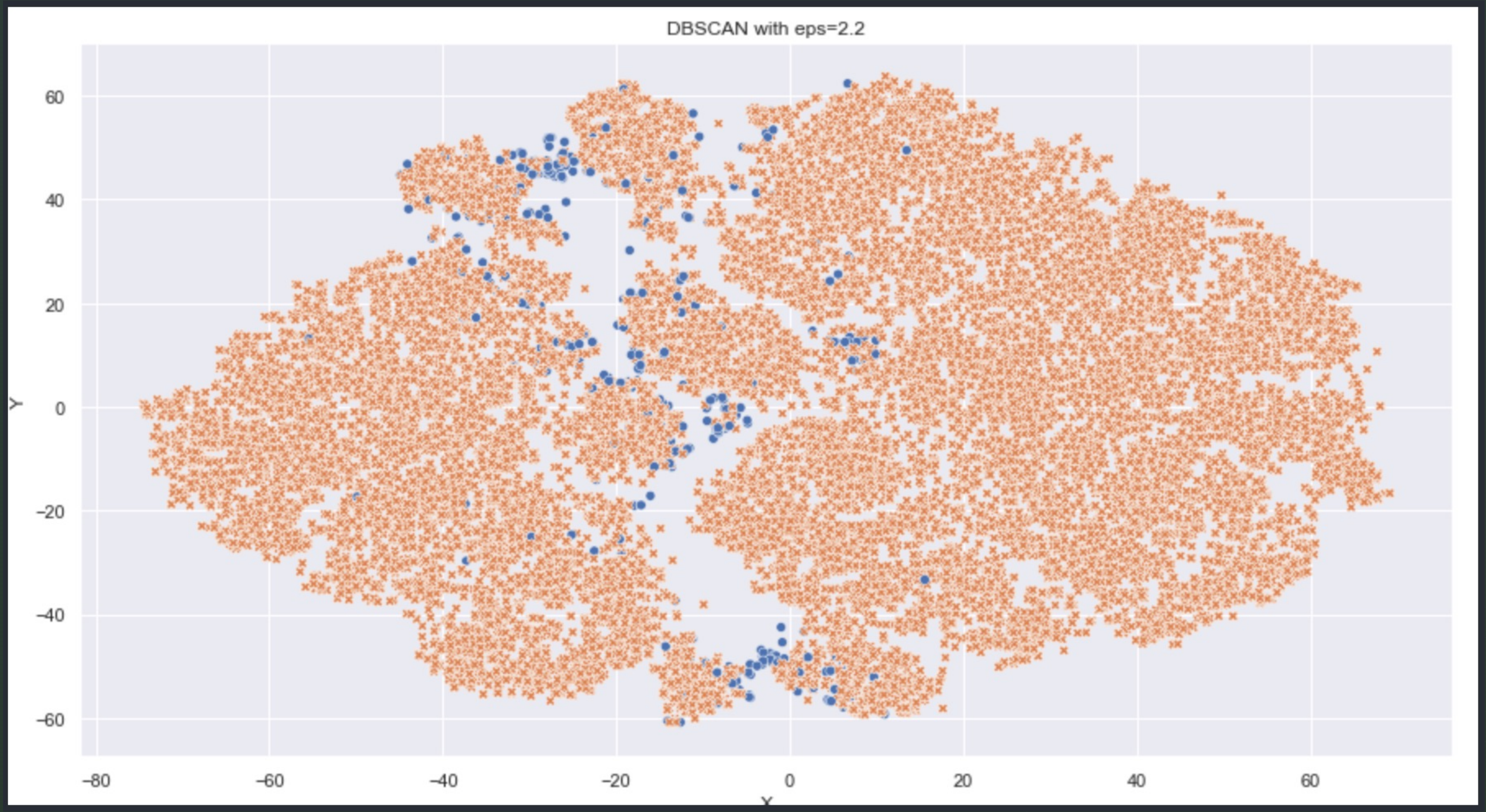
DBSCAN is a density-based clustering algorithm that forms clusters of dense regions of data points ignoring the low-density areas (considering them as noise).

Hyperparameter “eps”: value that deals with the radius of the clusters you are trying to find.



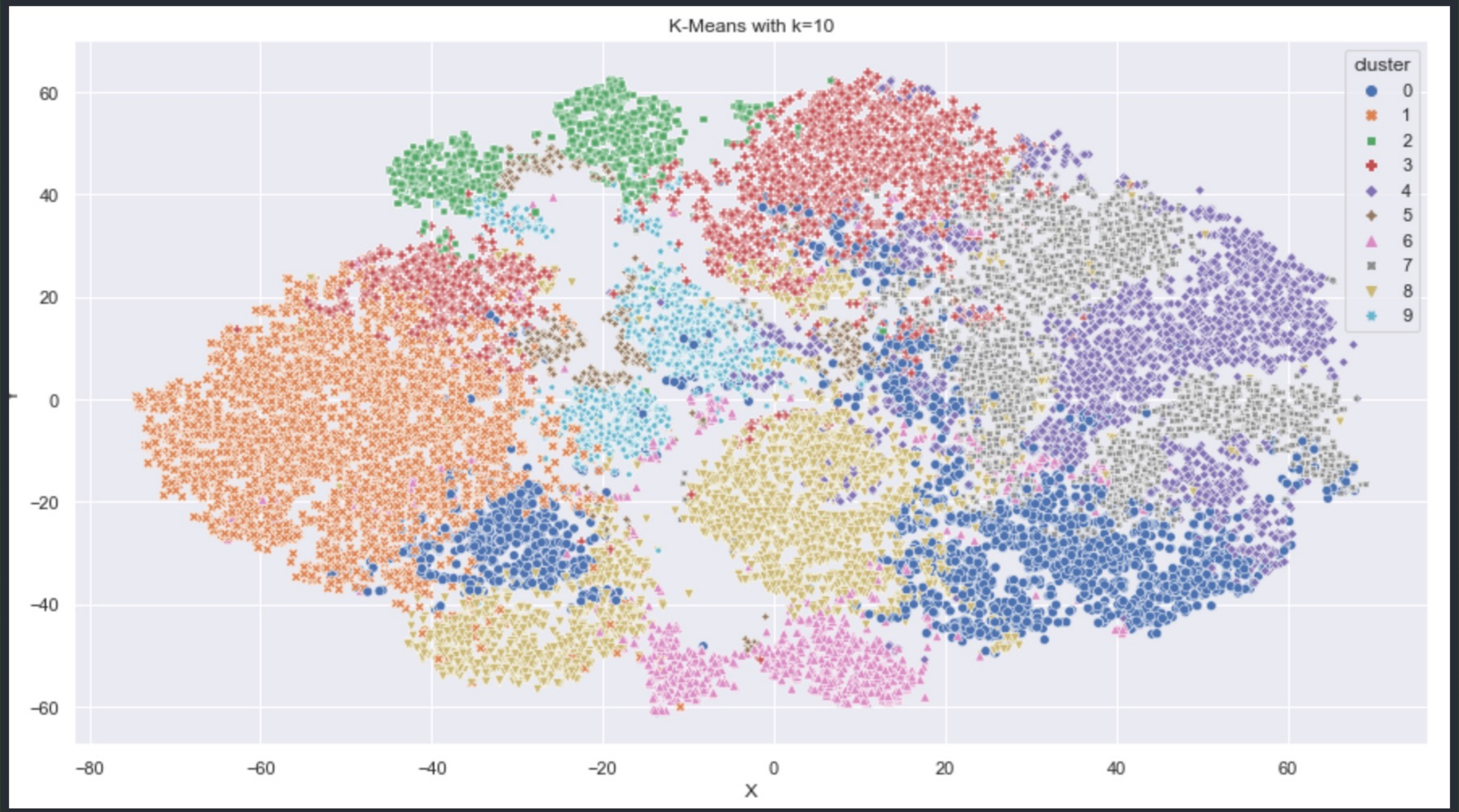
DBSCAN clustering algorithm

- “eps” = 2.2
- Estimated clusters = 2



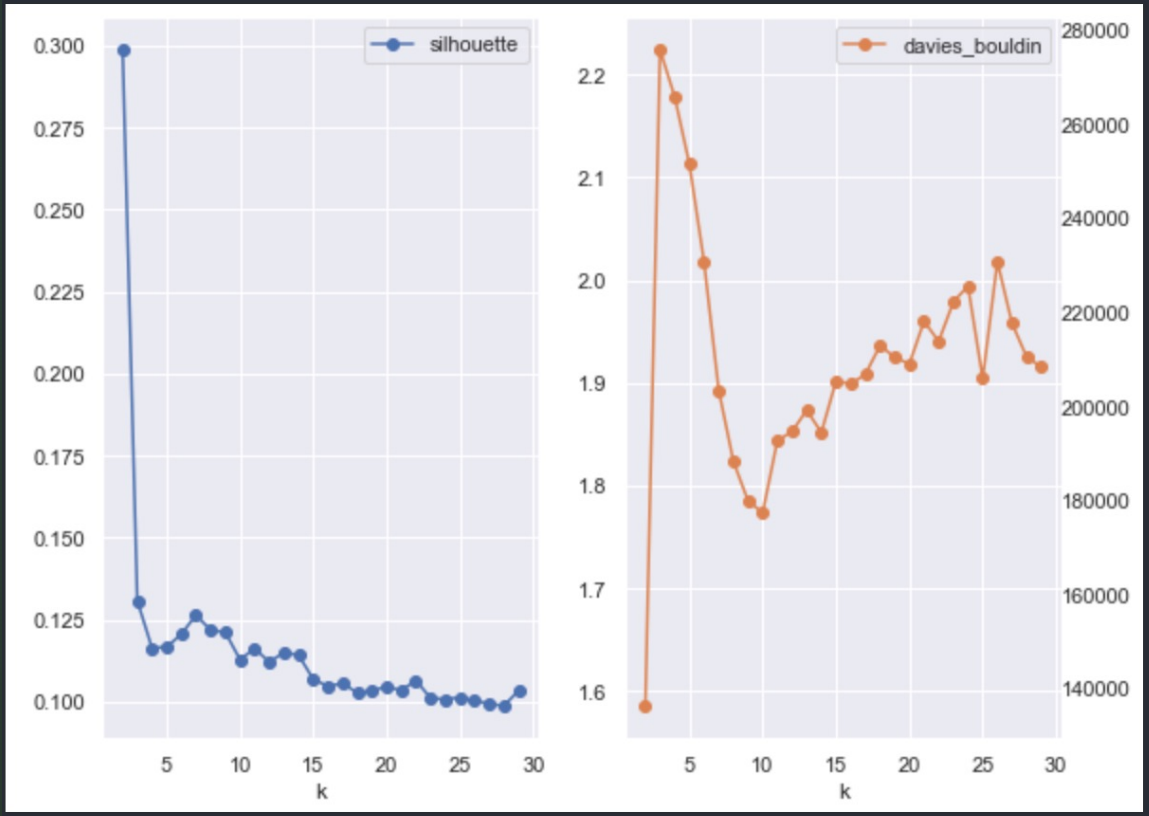
K-Means clustering algorithm

K-Means cluster is one of the most used unsupervised machine learning clustering techniques. It is a centroid based clustering technique that needs you decide the number of clusters (centroids) and randomly places the cluster centroids to begin the clustering process.



K-Means clustering algorithm

Prioritizing the quality of separation between clusters, as they represent types of similar songs, the Davies-Bouldin Index is going to be the final indicator for the k selection.



Model selection

Prioritizing Davies-Bouldin Index as the indicator for our model selection, K-Means is the best model to create significant clusters.

For the density model, the clustering hasn't worked as expected even after fine tuning the "eps" and the results weren't great either.

Davies-Bouldin Scores:

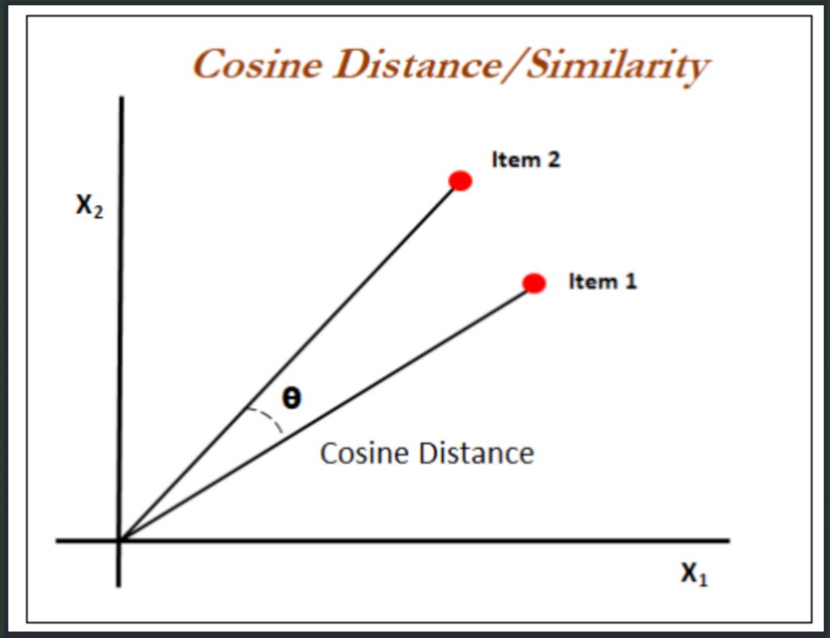
- DBSCAN: 2.22
- K-Means: 1.77

Recommender system

Music recommender system → based on clustering → serving as boundaries

User inputs two songs → 10-song playlist from the cluster recommended

Similarity metric to recommend songs → Cosine similarity



Streamlit app

Models can be integrated into product/applications and available to any user.

The complete deployment process involves three major steps:

- API which can easily access the machine learning models.
- Front-end application which will allow the users to access the predictions.
- Cloud/Server to deploy the application.

Conclusion

K-Means is the best model to create clusters (with the dataset used).

Recommendation systems can become complex, and the importance of data is again proven yet another time.

Spotify must have a hybrid recommender system that combines collaborative and content-based filtering.

As for next steps, I would like to try a hybrid recommender system combining content-based filtering with some NLP where lyrics come into play and the message of the song becomes an important feature. Also, I would like to gather more data and use a cloud/server to deploy the application.