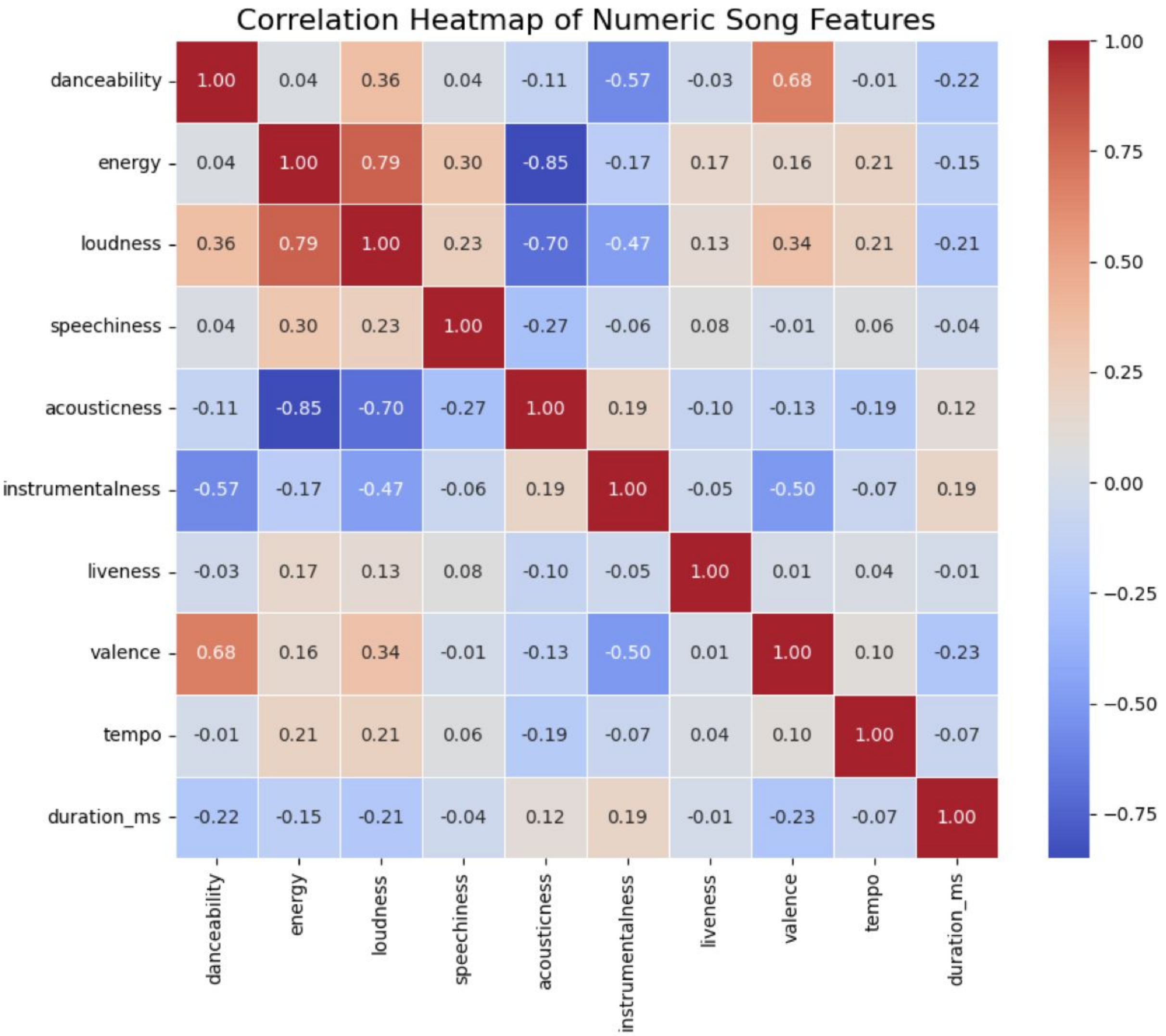
CAN A ROBOT FEEL THE GROOVE?

THE PRETZEL PATROL PLAYLIST CREATION ALGORITHM *PROTOTYPE

PREP-WORK

- ✓ CLEAN THE DATA
- ✓ CHOOSE DIMENSIONS
- ✓ CHOOSE SCALING
- ✓ EXPLORE CORRELATIONS
- ✓ REDUCE REDUNDANCY



CHOOSING AN ALGORITHM

K MEANS

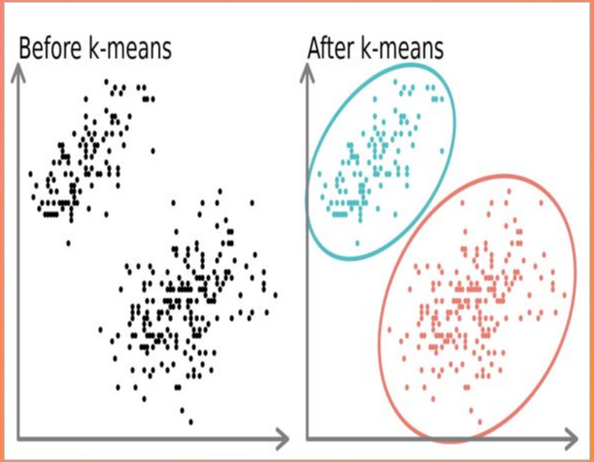
CREATES CLUSTERS BASED ON CENTER-POINTS

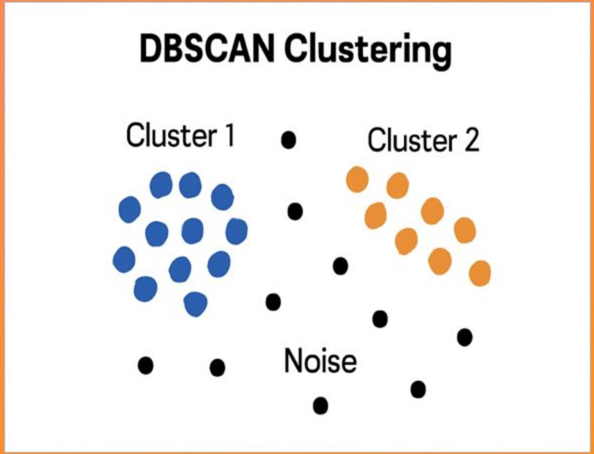
DB SCAN

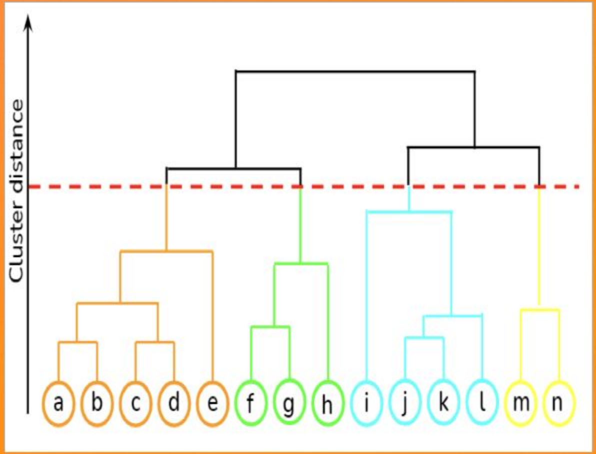
CREATES CLUSTERS BASED ON DENSITY

AGGLOMERATIVE

CREATES HEIRARCHIAL CLUSTERS







CHOOSING AN ALGORITHM

Clustering with K Means: 20 Playlists

Quantile transformed → PCA → KMeans (k=20)

- We decided to create **20 playlists**.
- Each playlist groups songs with **similar musical traits**.
- The algorithm doesn't know genres — it only sees the numbers (tempo, energy, danceability, etc.).
- But when we look at the groups, they make **musical sense**.

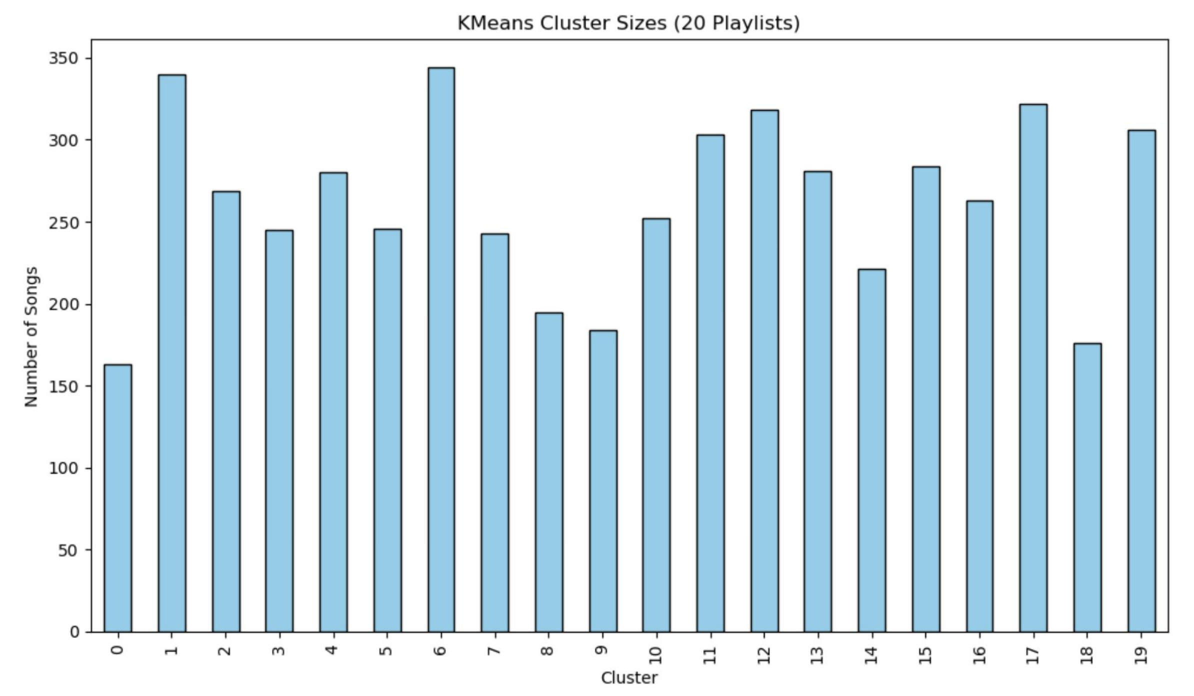
Clustering with DBSCAN:

Quantile transformed → UMAP → DBSCAN

- UMAP = **map-making → puts similar songs close together on a 2D/low-dimensional map**
- DBSCAN = **neighborhood-finding → detects dense song groups and leaves out outliers**
- Found **19 clusters + outliers**

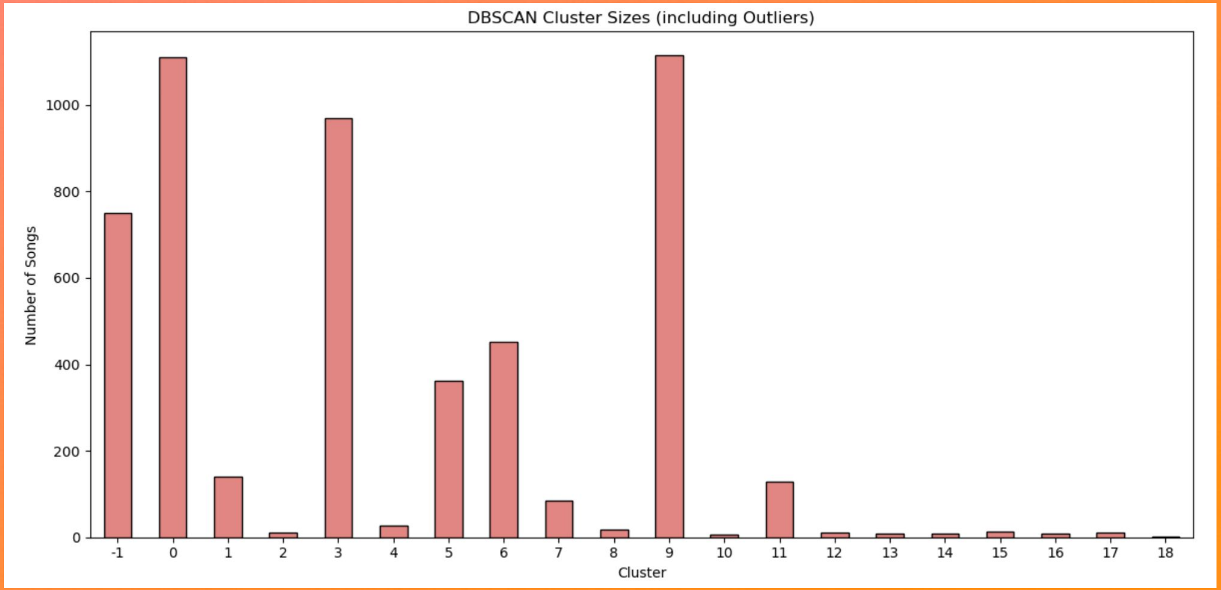
Playlist Names and Sizes - with K Means

0 – Chill Acoustic Instrumental (Sad/Moody)
1 – Dance Pop / Vocal EDM (Happy)
2 – Intense Instrumental Electronic / Rock (Dark)
3 – Indie / Singer-Songwriter (Mid-tempo)
4 – Happy Acoustic Pop (Upbeat, Short)
5 – Danceable & Happy (Vocal Pop)
6 – Energetic Electronic / Fast Beats
7 – Long Tracks / Mixed Styles
8 – Danceable Rap / Hip-Hop
9 – Happy Spoken / Rap-Pop
10 – Dark Energetic Electronic Instrumental
11 – Energetic with Live Feel (Mixed)
12 – Danceable Electronic Rap (Dark but Groovy)
13 – Speech-Heavy Rap
14 – Fast Rap (High Tempo)
15 – Short Happy Pop
16 – Chill Acoustic / Moody Slow
17 – Chill Acoustic Instrumental (Extended)
18 – Experimental / Instrumental Mix
19 – Energetic Electronic (Club/Live)

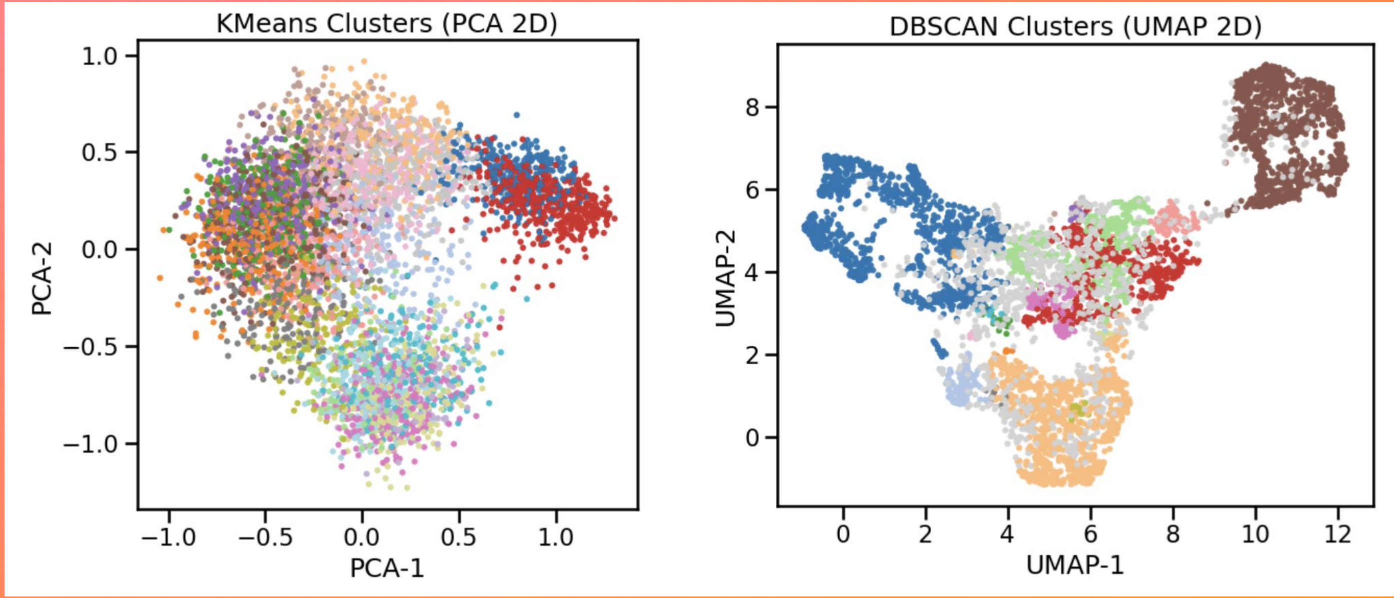


Playlist Names and Sizes - with DB SCAN

-1 – Happy / Vocal-Forward / Danceable (Outliers)
0 – Acoustic / Chill / Quiet/Soft / Instrumental
1 – Happy / Vocal-Forward / Acoustic / Short Tracks
2 – Fast / Happy / Acoustic / Vocal-Forward / Chill
3 – Danceable / Vocal-Forward / Loud / Happy / Spoken
4 – Fast / Happy / Short Tracks / Live / Acoustic
5 – Danceable / Electronic / Loud / Energetic / Happy
6 – Live / Loud / Danceable / Energetic / Electronic
7 – Live / Electronic / Energetic / Loud / Fast
8 – Slow / Dark/Moody / Vocal-Forward / Electronic
9 – Not Dancey / Energetic / Dark/Moody / Instrumental
10 – Electronic / Dark/Moody / Long Tracks
11 – Danceable / Happy / Slow / Vocal-Forward / Spoken
12 – Happy / Live / Slow / Acoustic / Short Tracks
13 – Happy / Danceable / Vocal-Forward / Slow / Chill
14 – Electronic / Long Tracks / Dark/Moody / Slow
15 – Loud / Vocal-Forward / Fast / Energetic / Electronic
16 – Spoken/Rap / Fast / Danceable / Happy / Vocal-Forward
17 – Happy / Danceable / Slow / Chill / Acoustic
18 – Happy / Danceable / Spoken/Rap / Electronic / Fast



Comparison



- **K Means = tidy but artificial**

  → 20 clusters always, even if the data doesn't naturally split that way. Overlaps show it's forcing boundaries.

- **DBSCAN = musically faithful but messy**

  → Finds real dense blobs (genres/substyles) and leaves outliers aside. Some clusters are big, some tiny, reflecting the real music distribution.

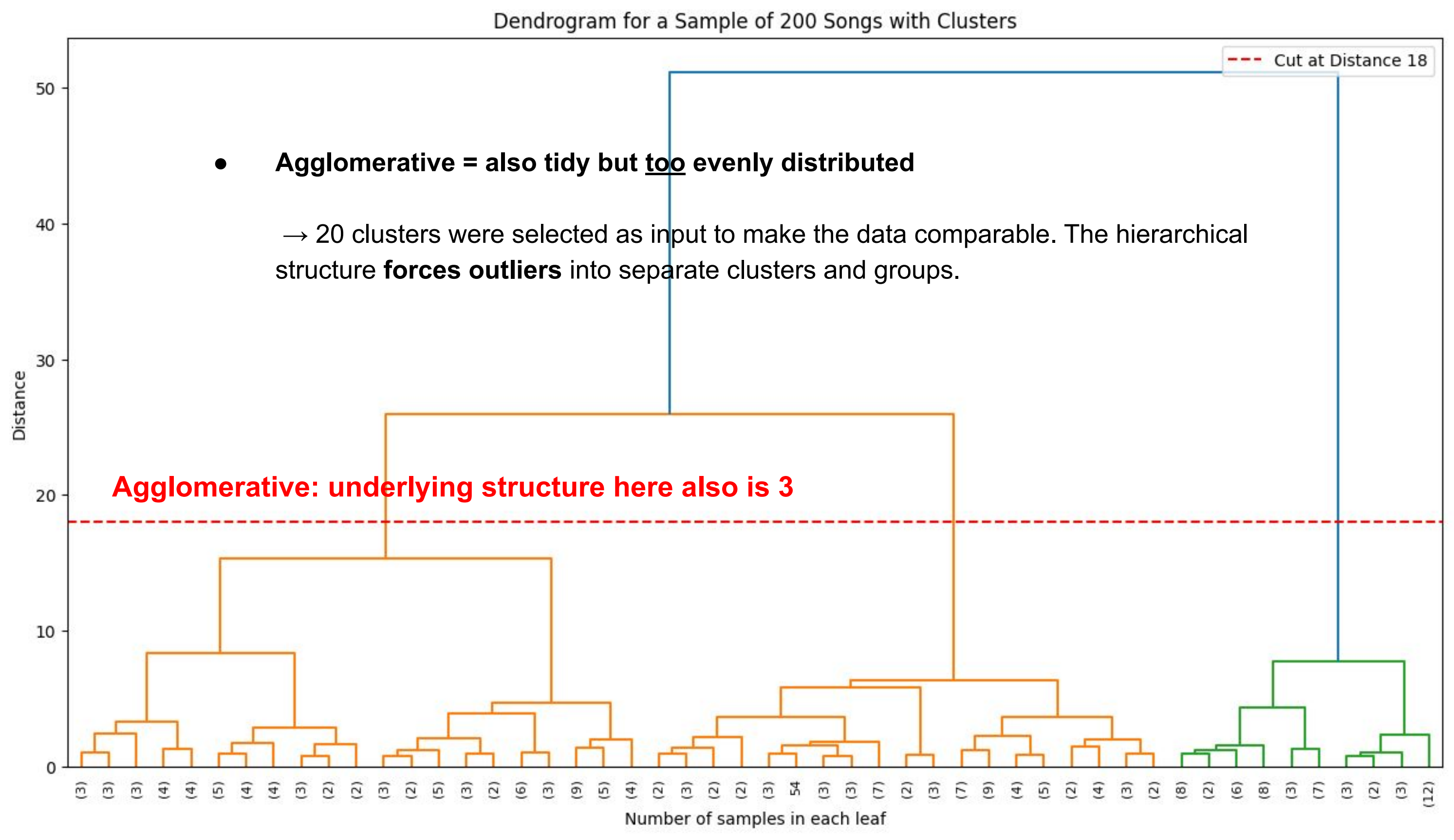
CHOOSING AN ALGORITHM

Clustering with AgglomerativeClustering: 20 Playlists

Quantile transformed → PCA → AGG-Clustering

- Minimise the **computing power**
- **3 clusters** are the **result** of the sample
- **Create 20 playlists**, as we did with **K-Means and DBSCAN** (19 playlists)
- **Distribution based on hierarchical merging**

Dendogram of a sample from 200 songs



- **Agglomerative = also tidy but <u>too</u> evenly distributed**

  → 20 clusters were selected as input to make the data comparable. The hierarchical structure **forces outliers** into separate clusters and groups.

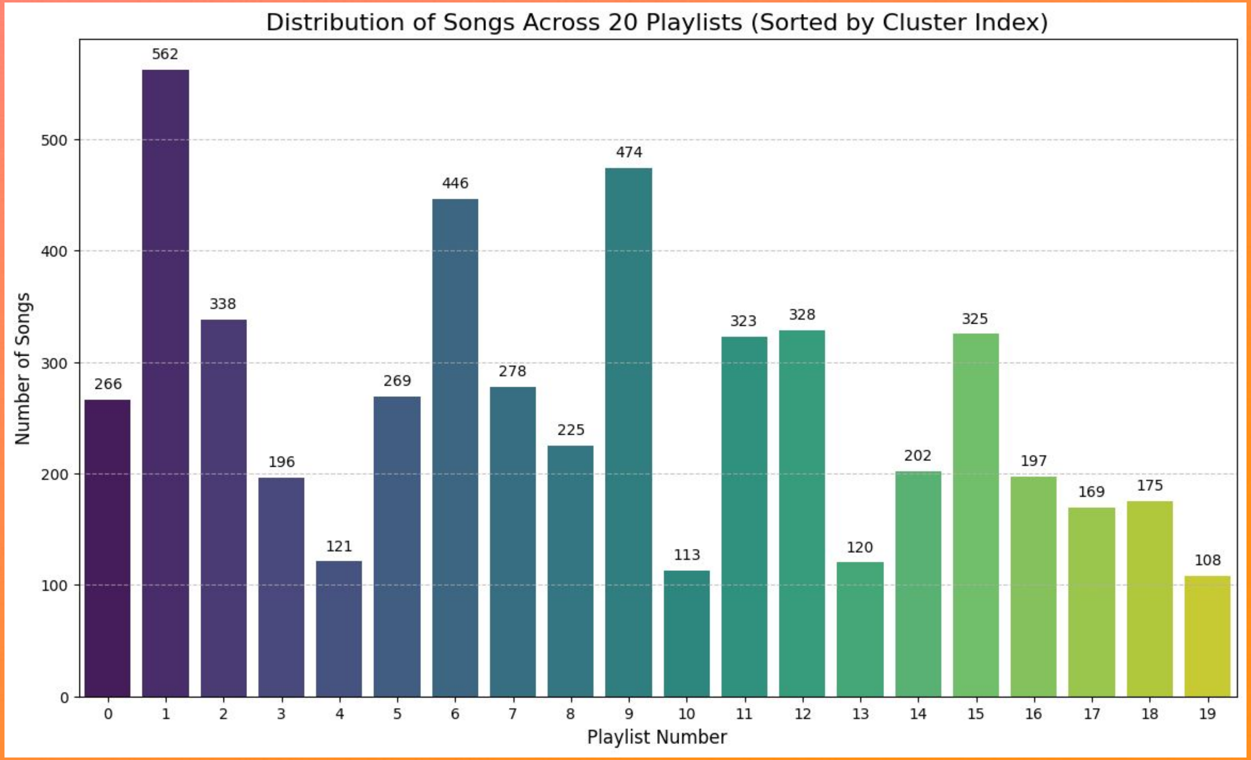
Agglomerative: underlying structure here also is 3

Cluster Sizes - AGG-Clustering

<u>Cluster distortion:</u>

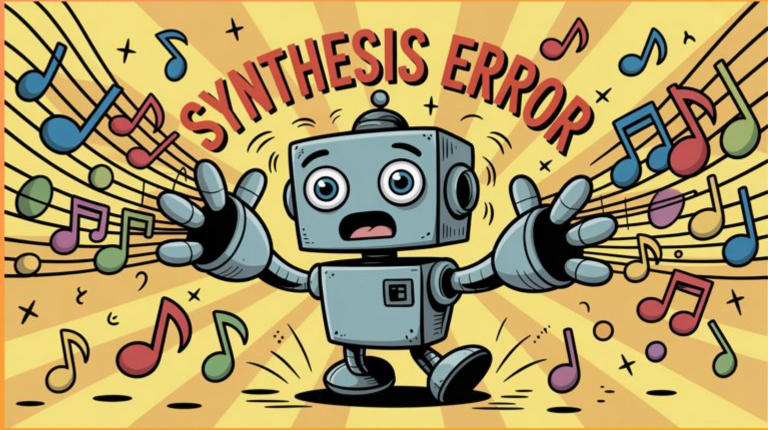
A outlier forced into a cluster distorts the averages and makes the cluster **less coherent**.

The algorithm **generates seemingly more balanced groups** instead of following the natural density of the data points, which is 3.



OVERALL CONCLUSIONS

ALGORITHMIC ADVANTAGES



ALGORITHMIC LIMITATIONS

DBSCAN

🎵 **MUSICALLY FAITHFUL**

⚠️ **LESS TIDY**

K-MEANS

📊 **ORGANIZED & BALANCED**

⚠️ **LESS GROOVY**

AGGLOMERATIVE

🌳 **GREAT FOR GENRES/ SUBGENRES**

⚠️ **LESS PRACTICAL**

LET'S KEEP EXPLORING!

NEXT STEPS:

EXPLORE SUPERVISED MACHINE LEARNING AND COMBINED ALGORITHMIC APPROACHES

Appendix

THE PRETZEL PATROL PLAYLIST CREATION ALGORITHM *PROTOTYPE

PRESENTATION AGENDA:

-How we created this prototype (UML)

-How our prototype creates a playlist (Clusters and Dimensions)

-Efficacy of prototype (Showcase clustering logic, Playlist Demo)

-Limitations of prototype (Musical Nuances, Data Complexity)

-Overall Pro/Con of UML

CHOOSING OUR DIMENSIONS

CATEGORICAL ?

RELEVANT ?

CONTINUOUS!

DANCEABILITY
ENERGY
~~KEY~~
LOUDNESS
~~MODE~~
SPEECHINESS
ACOUSTICNESS
INSTRUMENTALNESS
LIVENESS
VALENCE
TEMPO
~~TYPE~~
DURATION
~~TIME SIGNATURE~~
~~ID~~
~~HTML~~

THE IMPORTANCE OF SCALING DATA

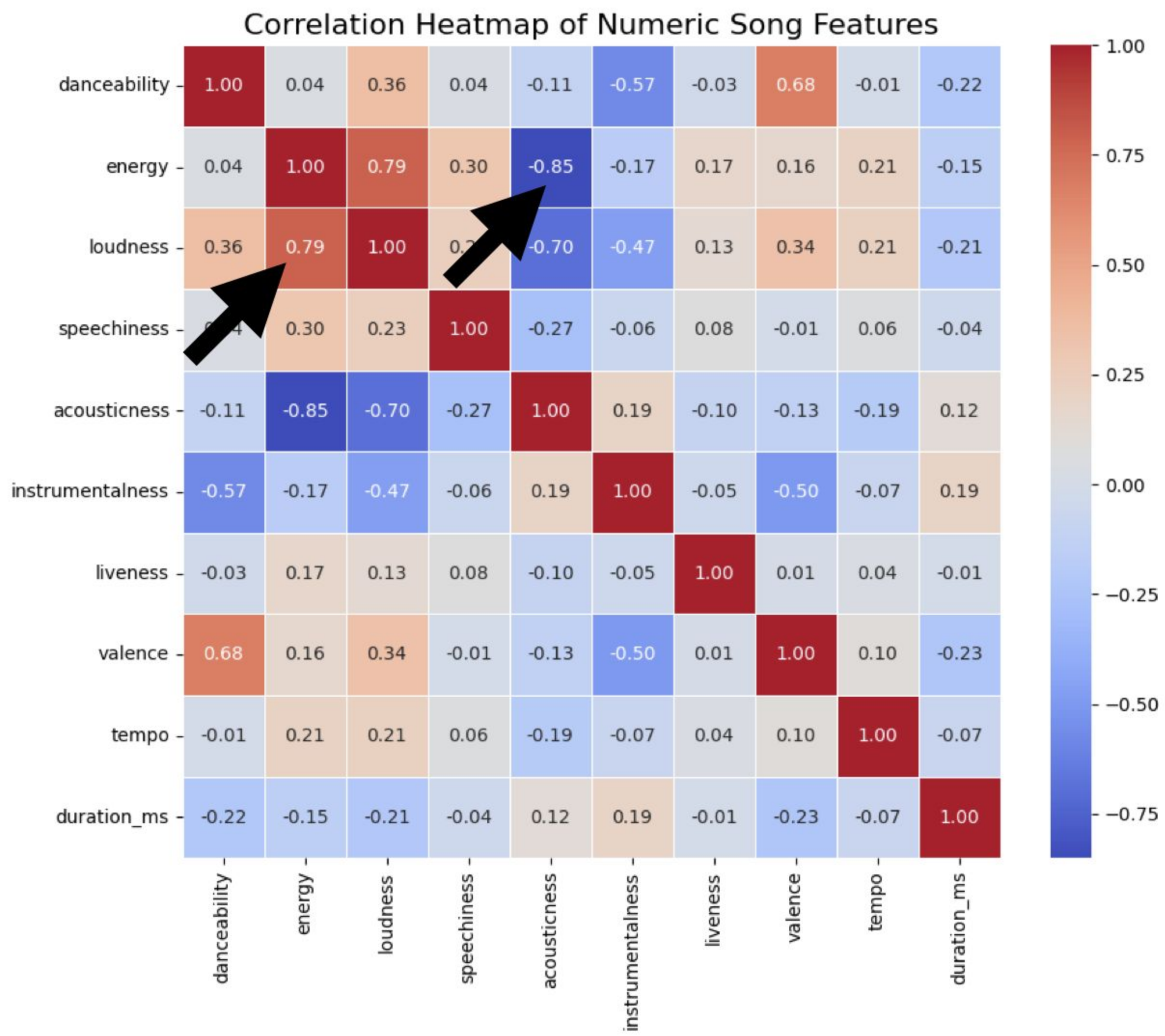
MACHINE LEARNING ALGORITHMS USE DISTANCE TO FIND SIMILARITIES.

SCALING:

BALANCES DOMINATING FEATURES.

REDUCES BIAS.

ENSURES EQUAL CONTRIBUTION.



EXPLORING CORRELATIONS

-LOUDNESS AND ENERGY

HIGH CORRELATION

-ACOUSTICNESS AND ENERGY

LOW CORRELATION

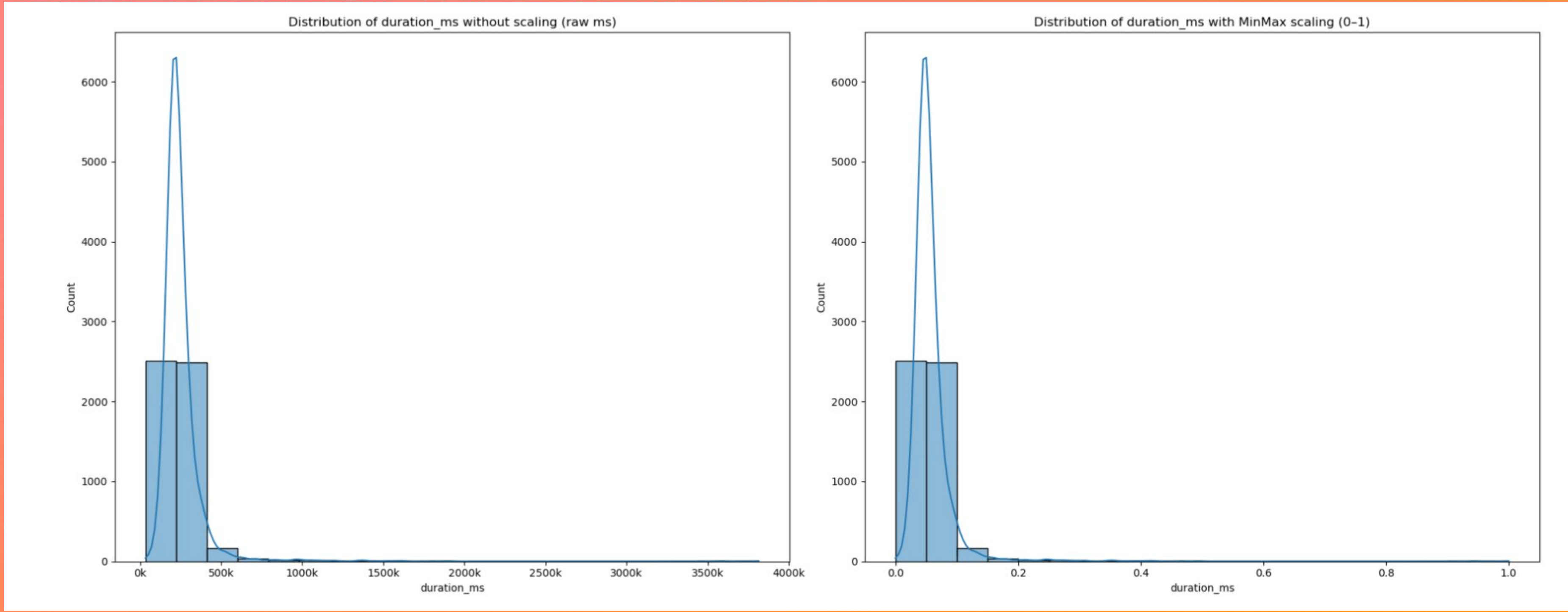
PRINCIPAL COMPONENT ANALYSIS

REDUCES DEMENSIONALITY (MIRRORED VALUES)

SCALING THE DATA

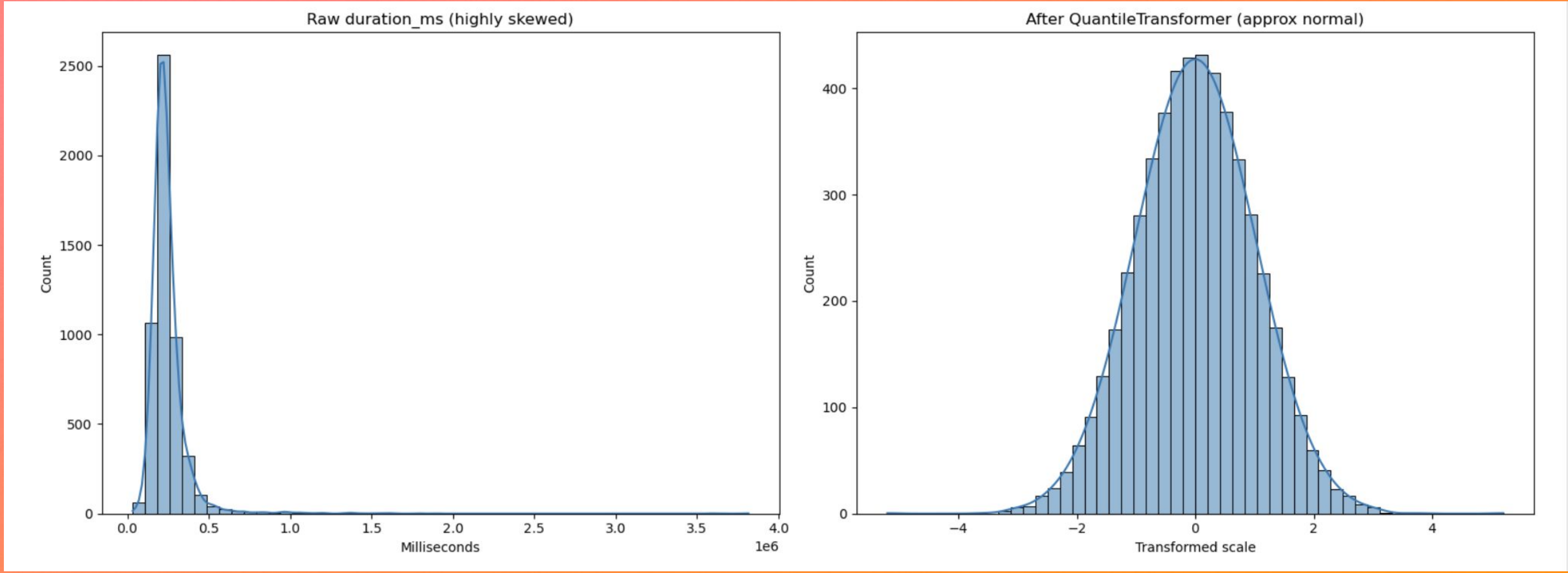
SCALING OPTION	WHAT IT DOES
MIN-MAX SCALER	SCALES TO A FIXED RANGE
STANDARD SCALER	SCALES CENTERING AROUND ZERO
ROBUST SCALER	IGNORES OUTLIERS AND SCALES THE REST
QUANTILE TRANSFORMER	RE-MAPS DATA, NORMALIZES BASED ON QUANTILE
POWER TRANSFORMER	RE-MAPS SKEWED DATA TO A NORMAL DISTRIBUTION

SCALING THE DATA



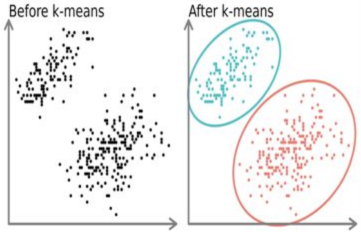
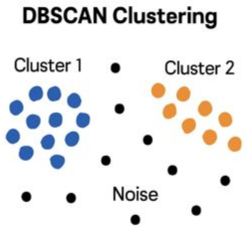
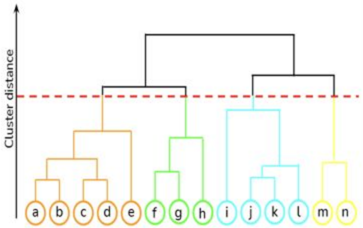
- The raw data (like song duration) is very **skewed** — most songs are short, but a few very long tracks pull the scale out of balance.
- Even after **MinMax scaling (0–1)**, the data is still **squeezed into one corner** — so clustering would not work fairly.

SCALING THE DATA

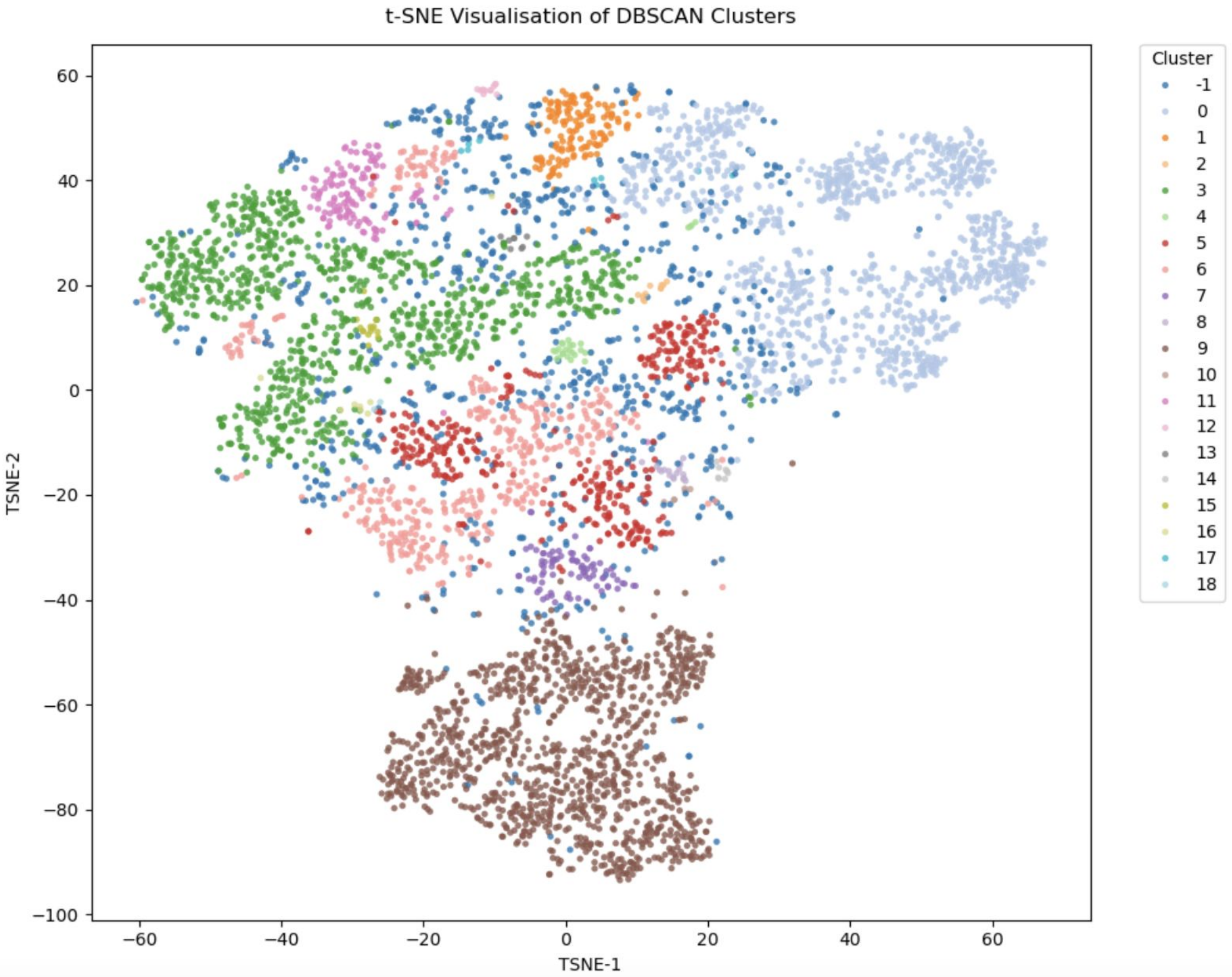


- To make the data more **balanced** and fair, we used a **Quantile Transformer.**
- This gives each feature an equal chance to influence the clustering.

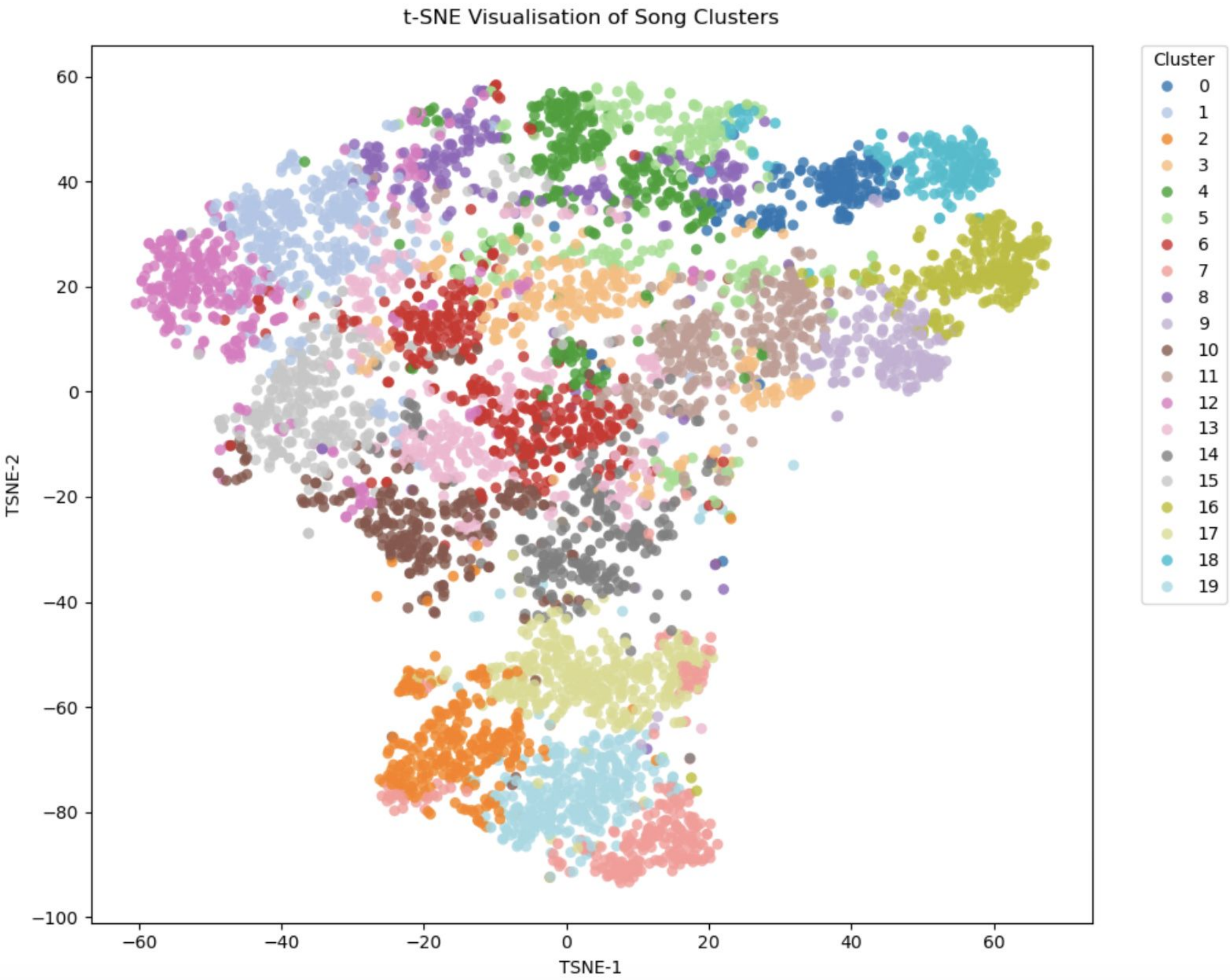
TYPES OF MACHINE LEARNING ALGORITHMS WE EXPLORED:

	WHAT IS IT?	PROS	CONS
K-Means	CREATES CLUSTERS BASED ON CENTER-POINTS 	FAST. SIMPLE.	NEEDS TO BE TOLD HOW MANY CLUSTERS TO CREATE.
DBscan	CREATES CLUSTERS BASED ON DENSITY 	GREAT FOR FINDING OUTLIERS. FINDS NUMBER OF CLUSTERS.	TIME CONSUMING CALCULATIONS.
Agg	CREATES HEIRARCHIAL CLUSTERS 	VISUAL. FINDS NUMBER OF CLUSTERS.	TOO SLOW FOR LARGE DATA. SENSITIVE TO OUTLIERS.

- Each dot is a song.
- The colors are the **natural clusters** found by DBSCAN. (***natural playlists*** *discovered by DBSCAN)*
- Songs close together sound more alike.
- Some dots are gray/extra (cluster **-1**) → these are **outliers** that don't fit into any cluster.



- Each dot is a song.
- The colors are the playlists.
- Songs that are close together sound more alike.



Final Comparison

DBSCAN

🎵 More *musically faithful*

- Respects natural structure
- Finds big genres + tiny niche groups
- Keeps outliers separate

⚠️ Less tidy → uneven cluster sizes

K Means

📊 More *organized & tidy*

- Exactly 20 playlists
- Balanced group sizes
- Easy to manage & explain

⚠️ Less faithful → forces songs into clusters

Agglomerative (Hierarchical)

🌳 Tree-like structure

- Builds a “family tree” of songs, merging step by step
- Can reveal relationships between clusters (genres → subgenres)

⚠️ **Less practical for 5,000+ songs** → too slow, memory-heavy

⚠️ Hard to decide where to “cut” the tree → playlists become arbitrary