Colab links:
Face_Emotion_Music_Recommendation.ipynb
ML-01.ipynb

Dataset: kaggle-278k-songs
Paper link
drive link

ML project Proposal

User uploads a face photo → our model reads the face emotion → the system suggests songs that match that emotion.
We used your numeric audio data for 278k songs. We scaled the features and ran K-Means to make 7 song groups. Then we named each group as one emotion → Sad, Happy, Angry, Neutral, Fear, Disgust, Surprised. When the face model detects an emotion, we pick songs from that matching group and show them

What data we have

- Around **28k songs**.
- Each song already has **numeric audio features** (not plain lyrics). Example features: **danceability**, **energy**,

loudness,**speechiness**,**acousticness**,**instrumentalness**, **liveness**, **valence**, **tempo**, **spectral rates** etc.

Note: We are clustering songs based on these audio numbers (not on lyrics). Audio features tell us if a song is upbeat, slow, happy-sounding, acoustic, etc.

1. Pick the song features

We used the numeric columns like danceability, energy, valence, tempo, etc. Each song is a point with many numbers.

2. Make numbers comparable

Some features have big ranges (tempo = 70–200). Some are 0–1 (danceability).
We used:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

This makes every feature have mean 0 and similar scale. It stops big numbers from dominating.

3. Use K-Means to group songs into 7 clusters

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=7, random_state=42)
```

clusters = kmeans.fit_predict(X_scaled)

df['cluster'] = clusters

What K-Means did:

- It picked 7 center points.
- Each song went to the nearest center.
- It moved centers and repeated until stable.
- Result: each song has a cluster ID 0..6.

4. Give human names to clusters

K-Means returns numbers (0..6). We mapped them to emotion words:

emotion_map = {

0: "Sad", 1: "Happy", 2: "Angry",

3: "Neutral", 4: "Fear", 5: "Disgust",

6: "Surprised"

}

df['emotion'] = df['cluster'].map(emotion_map)

This is a manual step. We looked at cluster features (like low valence + low energy → “Sad”) and chose names.

5. Save the result

We saved all songs with their assigned emotion:

df.to_csv("clustered_songs.csv", index=False)

Now we have a big file with song id + cluster + emotion.

How recommendation works

When the face model says **detected_emotion = "Happy"**, we pick songs from the "Happy" cluster:

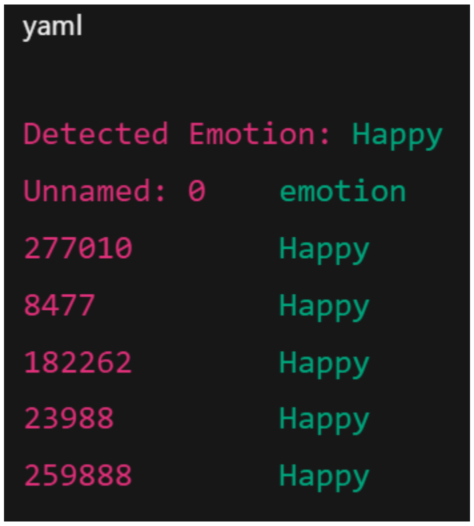
```
def recommend_songs(detected_emotion, df, n=5):

    songs = df[df['emotion'] == detected_emotion.capitalize()].sample(n)

    return songs[['Unnamed: 0', 'emotion']]
```

This picks 5 random songs labeled as Happy and shows their IDs and emotion.

Example output:



```
yaml

Detected Emotion: Happy
Unnamed: 0     emotion
277010         Happy
8477           Happy
182262         Happy
23988          Happy
259888         Happy
```