🎵 Hybrid Music Recommendation System (KMeans + KNN on Song Embeddings)

This repository contains a Python script that implements a **hybrid content-based recommendation system** for music. It combines:

- **KMeans clustering** to group songs with similar overall characteristics.
- **K-Nearest Neighbors (KNN)** in embedding space (EffNet or Maest) to find the most similar songs within the same cluster.

It also includes an **interactive visualization** using t-SNE and Plotly to explore recommendations visually.

📂 Folder Structure

css
CopiarEditar

```
MTG-102/
├── code/
│   └── data_visualization/
│       └── similarities/
│           └── Recommendator.py  # ← Main script
├── song_embeddings/
│   ├── embeddings_effnet/
│   │   ├── before_2012_effnet_embeddings.json
│   │   └── after_2018_effnet_embeddings.json
│   └── Archivo_maest_pkl/
│       ├── before_2012_maest_embeddings.pkl
│       └── after_2018_maest_embeddings.pkl
```

🚀 Features

- Uses either **EffNet** or **Maest** embeddings.
- Clusters all songs using **KMeans** (with customizable `k`).
- For a given song (by artist & title), finds the **N most similar songs** in the same cluster using **cosine** or **Euclidean** distance.

- Visualizes recommendations in 2D using **t-SNE** and **Plotly**.

✅ Requirements

Install dependencies (if not already installed):

bash
CopiarEditar

```
pip install numpy pandas scikit-learn plotly
```

⚙️ How to Use

1. Open and edit the top of the script to configure:

python
CopiarEditar

```
model = "maest"                # Choose between "effnet" or "maest"
song_name = "Jenifer"          # Song title
Artist_name = "Els_Catarres"   # Artist name
k = 4                          # Number of KMeans clusters
metric = "euclidean"           # "cosine" or "euclidean"
```

2. Then run the script:

bash
CopiarEditar

```
python Recommendator.py
```

3. You'll get:
    - A printed DataFrame of the top recommended songs
    - An **interactive Plotly visualization** highlighting:
        - The query song (black)
        - Recommendations (red)

- Cluster assignment (colored background)

📌 Example Output

text
CopiarEditar

```
         Artist          Song    Similarity
0  31_fam         al_cantu        0.7423
1  Doctor_Prats   caminem_lluny   0.8031
2  Zoo            estiu           0.8298
...
```

And a 2D t-SNE plot will be displayed interactively in your browser or Python environment.

🧠 How It Works

1. **Embeddings**:
    - **EffNet**: time-averaged across frames (from JSON)
    - **Maest**: flattened and truncated to 1000 dims (from PKL)
2. **Clustering**:
    - All songs clustered using `KMeans(n_clusters=k)`
3. **Similarity**:
    - Recommendations filtered to same cluster as query song
    - Similarity computed using `cosine` or `euclidean` distance
4. **Visualization**:
    - `t-SNE` reduces high-dimensional embeddings to 2D
    - Plotly shows cluster structure and recommendations

🧪 Customization Tips

Variable	Description
`model`	`"effnet"` or `"maest"`
`k`	Number of KMeans clusters
`metric`	`"cosine"` or `"euclidean"`
`n`	Number of recommended songs returned
`Artist_name`, `song_name`	Used to build the `query_id`

👥 Credits

Developed by Group [102] – Music Technology Lab – UPF
For the MTG-102 project
Contact: https://github.com/martiarmengol/MTG-102