**📄 User Guide: Content-Feature Based Song Recommender**

**🎯 What This File Does**

This Python script implements a **content-feature-based music recommender system** that:

* Loads audio and structural features from a music dataset (msd\_processed.csv).
* Accepts user **playlist feedback** (liked/disliked songs or segments).
* Computes similarity using **cosine similarity** on normalized audio features.
* Returns a song recommendation from the dataset based on that feedback.

**📂 File Requirements**

* msd\_processed.csv: A processed dataset containing audio and segment-level features.
* Libraries: pandas, numpy, scikit-learn, ast, random, json.

**📥 Input Format**You need to call the recommend(...) function with:

**1. current\_track\_id (str)**

The track ID of the **currently playing song**, present in the dataset.  
Example:

"TRAXLZU12903D05F94"

**2. playlist\_feedback (List[Dict])**

A list of **feedback objects**, each with the following keys:

| **Key** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| track\_id | str | ✅ | ID of the song in the dataset |
| rating | int | ✅ | 1 for like, -1 for dislike |
| segment\_index | int/None | Optional | Index of a section in the song |
| segment\_rating | int/None | Optional | Segment feedback (1 or -1) |

**✅ Example:**

[

{

"track\_id": "TRAAAAV128F421A322",

"rating": 1

},

{

"track\_id": "TRAAABD128F429CF47",

"rating": -1,

"segment\_index": 2,

"segment\_rating": -1

}

]

**📤 Output Format**

The output of recommend(...) is a dictionary like this:

**✅ Example:**

{

"sections\_start": {

"sections\_start": [0.0, 14.5, 29.8, 45.3,……..],

"duration": 198.7

},

"recommended\_song": "TRAAACN128F9355673"

}

**Breakdown:**

* **sections\_start**: Info about the current song’s structure:
  + sections\_start: List of timestamps (in seconds) for each section start.
  + duration: Total length of the song in seconds.
* **recommended\_song**: Track ID of the song recommended based on the feedback.

**🔁 How to Use It (Steps)**

1. Ensure msd\_processed.csv is placed in the same folder as recommender.py.
2. In your Python script or terminal:

from recommender import recommend

track\_id = "TRAXLZU12903D05F94"

feedback = [

{"track\_id": "TRAAAAV128F421A322", "rating": 1, segment\_index:1, segment\_rating:0},

{"track\_id": "TRAAABD128F429CF47", "rating": -1, segment\_index:1, segment\_rating:0}

]

output = recommend(track\_id, feedback)

print(output)

**🧠 Behind the Scenes**

* Features like danceability, energy, tempo, etc. are normalized.
* If segment-level feedback is present, the system computes a **hybrid feature vector** by combining both the full-song features and the features of the specific segment the user reacted to. This ensures that recommendations reflect both the overall identity of the song and the specific part the user found relevant. Songs already in the feedback are excluded from recommendations.
* If no usable vector is computed, a fallback **random unrated** song is chosen.
* Recommend is a wrapper function to return section data of current song and next recommended song at once.

**⚠️ Notes & Limitations**

* All track IDs in input must be present in the dataset, or they’ll be ignored with a warning.
* segment\_index must refer to a valid index in the sections\_start list of the corresponding song. Since,  
  + sections\_start is a **list of timestamps** (e.g., [0.0, 10.2, 20.4, 33.0,…….])
  + segment\_index = 2 means you're referring to the section that starts at sections\_start[2] and ends at sections\_start[3]
  + So even if segment\_index isn't stored in MSD, **you must use a valid index for sections\_start to avoid errors or empty segment vectors**.

**⚠️ Debug Info**

* If the **current song** passed as input is **not found in the MSD dataset**, the script will print:

[Warning] Input song <track\_id> not found in the dataset.

* If the **playlist cache** (feedback) contains songs that are **not found in the dataset**, the script will print:

[Warning] Track <track\_id> not found in the dataset. Skipping.

* If **none** of the feedback songs are found or no recommendation can be made, the script may print:

[Info] No valid songs found in feedback. Cannot compute recommendation.

[Info] No candidate songs left to recommend.