

MUSIC RECOMMENDATION SYSTEM

AI-Powered Personalized Song Discovery
Using Collaborative Filtering & Content-
Based Hybrid

BY: NO NA

INDU NISA INDIRA ALZA MILI



SpotYourVibes

Music Recommendation System

AI-Powered Personalized Song Discovery
From Listening History to Perfect Recommendations



8.5M
User Interactions



226K
Song Catalog



15,603
Total Users



Why a Recommendation System?

User Pain Points

Information Overload

Millions of songs are available, but not all are relevant. Users struggle to find the right song without guidance.

Cold Start Problem

New users have no history. The platform doesn't know their taste — 51.9% of cold users have only 1 song.

Evolving Taste

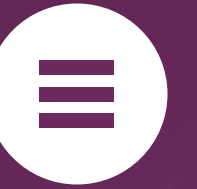
Music preferences evolve over time and context. A static model cannot keep up.

Real-World Dataset

The dataset used is real interaction data from a music platform — not a cleaned academic dataset. Challenges encountered (MemoryError, 99.9% sparsity, cold start) are real industry-level problems.

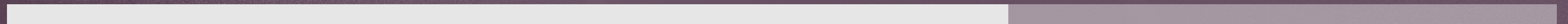
Production-Scale Data

8,549,544	user interaction rows	226,497	songs in master catalog
15,603	unique users	1,944,513	real interactions in matrix



PROBLEM STATEMENT

The vast number of available songs makes it difficult for users to find music that matches their preferences, highlighting the need for a music recommendation system that provides personalized recommendations based on users' playlist history and similarities in song characteristics.



What We Built ?

SongMatch is an AI-powered music recommendation system that automatically selects the best model for each user based on their listening history.

INPUT

Simple Input

Enter a User ID — the system instantly returns Top-10 personalized song recommendations in real-time.

MODEL

Automatic Model Selection

The system automatically picks the best model (ALS, Hybrid CBF, or Popularity) based on the user's interaction count.

DEPLOY

Production-Ready

Deployed on Streamlit Cloud. Model size only ~35 MB. Real-time inference without re-training.

Three Types of System Output

User Type	Available Data	Model	Recommendation Logic
Warm User	≥ 5 songs · enough to learn patterns	ALS (Collaborative Filtering)	"Users like you also enjoy this"
Cold User	< 5 songs · too little to learn	Content-Based Filtering	"This song is similar to what you've heard"
New User	No history yet	Popularity	"Here's what everyone is listening to"

Data Understanding & Preparation

df_user**8,549,544 baris**`user_id · artistname · trackname · playlistname`

Listening history per user from a real music platform

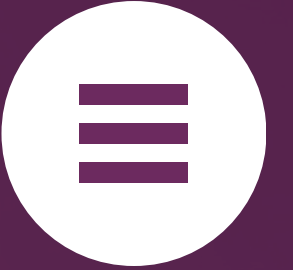
df_master**226,497 songs**`id · artist_main · name · 10 audio features · genres · popularity`

Complete song catalog with audio characteristics

Data Preparation Flow

**Normalize
Artist Name**lowercase + strip
punctuation**Chunk Merge
(500K/batch)**Hindari MemoryError
pada 8.5M baris**Data Cleaning**Drop bug columns
Deduplicate + join audio**Warm / Cold
Split** ≥ 5 interactions = warm
 < 5 interactions = cold**Warm Users (CF) 14,880 users —
95.4%****Cold Users (Hybrid CBF)
723 users — 4.6%**

COLLABORATIVE FILTERING



- Utilize other users with similar preference to recommend next item to a user.
- Measure User's Preference : Implicit and Explicit.
- How to measure "Similar Preference" : Cosine Similarity, Pearson Correlation, Dot Product

Method:

Memory Based :

- User Based
- Item Based

Model Based : dimensional reduction

- SVD (Singular Value Decomposition)
- ALS (Alternating Least Square)
- WLS (Weight Alternating Least Square)



Collaborative Filtering (ALS)

How CF Works

it looks at PATTERNS.
"Users who liked the same songs as you also like this."

Input	Sparse matrix 14,880 users x 15,502 items, 1,944,513 real interactions
Feedback	play_count sebagai confidence signal: $e = 1 + \alpha \times \text{play_count}$
Split	Per-user 80/20 split, not random rows — prevents data leakage
Output	Top-10 unheard songs, ranked by predicted score

Why ALS?

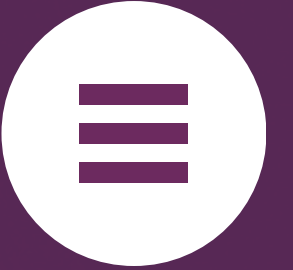
	ALS	SVD
Implicit feedback	YES	NO
Sparsity 99.9%	YES	Partial
Scalable 8.5M	YES	Partial

Hyperparameter Tuning Results

Parameter	Baseline	Best
factors	64	256
alpha	40	10
iterations	20	50
NDCG@10	0.1241	0.2268

+83% NDCG improvement from baseline — alpha=10 works better because play_count is not a pure preference signal

CONTENT-BASED FILTERING



Content-Based Filtering is a recommendation system method that gives recommendations based on the similarity of item characteristics to items that users have previously liked, viewed, or purchased.

Method

- Extract item features
- Build user profile
- Calculate similarity
- Recommend the most similar items

Basic Model

- Based on user preferences and item characteristics
- Uses similarity between liked items and new items
- Higher similarity score means better recommendation result



Content-Based Filtering (Hybrid)

How CBF Works

CBF tidak melihat user lain — it looks at SONG CHARACTERISTICS.
"This song has similar energy, tempo, and genre to what you listen to."

Step 1 **Build Item Profile**

846 cold user songs → vektor 10 audio features.
MinMaxScaler normalization (0-1) so all features are on equal scale.

Step 2 **Build User Profile**

Weighted average of audio features from listened songs. Weight = play_count — frequently played songs have more influence.

Step 3 **Cosine Similarity**

Measures the angle between the user profile and all candidates. Score 1.0 = identical, 0.0 = completely dissimilar.

10 Audio Features

valence	tempo	loudness
danceability	energy	popularity
instrumentalness	acousticness	
liveness	speechiness	

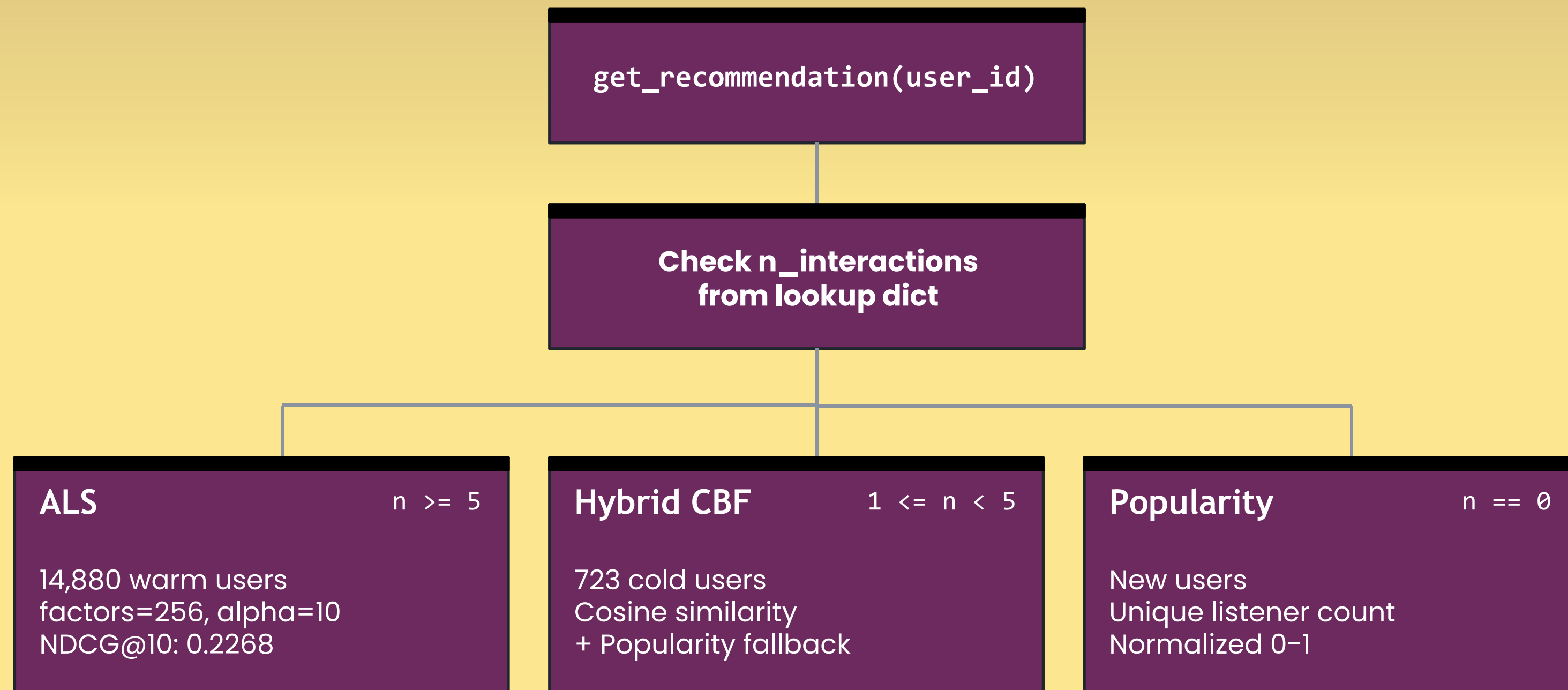
Hybrid Logic + Real Example

- 0 songs → Pure Popularity
- 1 song → 50% CBF + 50% Popularity
- 2-4 songs → Pure CBF
- 5+ songs → ALS (warm user)

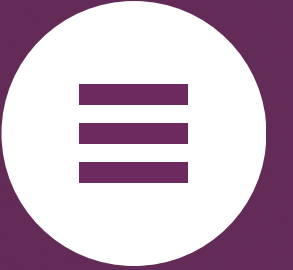
Example: 1 song (Gang Starr — hip-hop)
→ Missy Elliott, Warren G, Macklemore (all relevant hip-hop)

Unified Hybrid System

- One function `get_recommendation(user_id)` automatically selects the best model.
- Users never need to know which model is used.



LIMITATION & CONCLUSION



Limitations :

- Model does not update in real-time — periodic re-training required
- No feedback loop — skip/dislike signals are not yet utilized
- Popularity fallback is not personalized for new users
- Artist matching coverage is limited outside the full dataset

Conclusions :

- 100% of user types served — warm, cold, and new — in one unified pipeline
- ALS outperforms SVD for implicit feedback data
- Hybrid CBF proven to work even from just 1 song



THANK YOU!

