Atividade Prática: Sistema de Recomendação Musical com FastAPI

Objetivo

Criar um sistema de recomendação musical com 5 APIs diferentes utilizando FastAPI, implementando abordagens baseadas em conteúdo e filtro colaborativo.

Base de Dados

Você trabalhará com um dataset de músicas contendo os seguintes atributos: - Título, Artista, Gênero, Ano - Características musicais: BPM, Energia, Danceabilidade, Volume, etc. - Popularidade

Tarefa

Implemente 5 endpoints de API que oferecem diferentes abordagens de recomendação:

1. API de Recomendação Baseada em Conteúdo (Similaridade por Características)

- **Endpoint**: `/recommendations/content-based/{song_title}`
- **Método**: GET
- **Descrição**: Retorna músicas similares com base nas características musicais (BPM, energia, danceabilidade, etc.)
- **Parâmetros opcionais**:
  - `limit`: número de recomendações a retornar (padrão: 5)
  - `weights`: dicionário com pesos para cada característica

2. API de Recomendação Baseada em Gênero/Artista

- **Endpoint**: `/recommendations/genre-artist`
- **Método**: POST
- **Descrição**: Retorna músicas do mesmo gênero ou artista, ordenadas por popularidade
- **Corpo da requisição**:

```
{
  "genre": "nome do gênero",
  "artist": "nome do artista",
  "limit": 5
}
```

3. API de Filtro Colaborativo

- **Endpoint**: `/recommendations/collaborative/{user_id}`
- **Método**: GET

- **Descrição**: Sistema de recomendação baseado no comportamento de usuários similares
- **Funcionamento**:
  - Use dados fictícios de "usuários que gostaram disso também gostaram daquilo"
  - Implemente uma lógica simples de contagem de co-ocorrências

4. API de Recomendação Híbrida

- **Endpoint**: `/recommendations/hybrid`
- **Método**: POST
- **Descrição**: Combina abordagens baseadas em conteúdo e filtro colaborativo
- **Corpo da requisição**:

```
{
  "song_title": "nome da música",
  "user_id": "id do usuário",
  "content_weight": 0.7,
  "collab_weight": 0.3,
  "limit": 5
}
```

5. API de Recomendação por Popularidade/Ano

- **Endpoint**: `/recommendations/popular`
- **Método**: GET
- **Descrição**: Retorna músicas populares filtradas por ano ou gênero
- **Parâmetros opcionais**:
  - `year`: filtrar por ano
  - `genre`: filtrar por gênero
  - `limit`: número de resultados

Estrutura do Código Base

```python
from fastapi import FastAPI
from pydantic import BaseModel
from typing import Optional, Dict, List
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import MinMaxScaler
import numpy as np

app = FastAPI()

# Carregar dados
data = [
    # Seus dados aqui (pode carregar de um CSV também)
]

df = pd.DataFrame(data)

# Pré-processamento
features = ['Beats.Per.Minute', 'Energy', 'Danceability',
'Loudness/dB',
            'Liveness', 'Valence', 'Length', 'Acousticness',
'Speechiness', 'Popularity']
scaler = MinMaxScaler()
df[features] = scaler.fit_transform(df[features])

# Modelo de similaridade
similarity_matrix = cosine_similarity(df[features])

class GenreArtistRequest(BaseModel):
    genre: Optional[str] = None
    artist: Optional[str] = None
    limit: int = 5

class HybridRequest(BaseModel):
    song_title: str
    user_id: str
    content_weight: float = 0.7
    collab_weight: float = 0.3
    limit: int = 5

# Implementar os endpoints aqui...

@app.get("/recommendations/content-based/{song_title}")
async def content_based_recommendations(song_title: str, limit: int =
5, weights: Optional[Dict[str, float]] = None):
```

```
    """
    Implementar recomendação baseada em conteúdo
    """
    pass

@app.post("/recommendations/genre-artist")
async def genre_artist_recommendations(request: GenreArtistRequest):
    """
    Implementar recomendação por gênero/artista
    """
    pass

@app.get("/recommendations/collaborative/{user_id}")
async def collaborative_recommendations(user_id: str):
    """
    Implementar filtro colaborativo simulado
    """
    pass

@app.post("/recommendations/hybrid")
async def hybrid_recommendations(request: HybridRequest):
    """
    Implementar recomendação híbrida
    """
    pass

@app.get("/recommendations/popular")
async def popular_recommendations(year: Optional[int] = None, genre:
Optional[str] = None, limit: int = 5):
    """
    Implementar recomendação por popularidade/ano
    """
    pass
```

Requisitos de Implementação

1. Complete cada endpoint com a lógica de recomendação apropriada
2. Para o filtro colaborativo, simule dados de interação usuário-música
3. Documente cada endpoint com exemplos de requisição/resposta
4. Teste todas as APIs localmente

Dicas

- Use cosine similarity para similaridade baseada em conteúdo
- Para o filtro colaborativo, crie um dicionário simulando “usuários que gostaram de X”
- Combine os scores das diferentes abordagens na recomendação híbrida
- Ordene por popularidade quando relevante