VIT BHOPAL
www.vitbhopal.ac.in

Date:

PROJECT REPORT
MUSIC RECOMMENDER SYSTEM
INTEGRATED MTECH(DATA SCIENCE)

Submitted to:Dr Bandla Pavan Babu

Submitted by:Garima Shekhawat

ACKNOWLEDGEMENT

This project has been shaped by the guidance, support and encouragement of many individuals. I express my sincere gratitude to my faculty mentor Dr Bandla Pavan babu for offering valuable suggestions, technical insights and continuous support throughout the development of this project.

I also thank my friends and family for their patience, motivation and encouragement during this work.

INTRODUCTION

The Music Recommender System is a machine learning-based application designed to suggest songs to users based on their preferences, listening history and similarity patterns. With the explosion of digital music platforms, users are often overwhelmed by the sheer volume of available content. A recommender system helps narrow down choices, making music discovery faster, personal and more enjoyable.

This system analyzes user behavior, musical attributes and patterns to predict what the listener is most likely to enjoy next. It leverages algorithms such as content-based filtering, collaborative filtering, and hybrid recommendation models to generate personalized playlists.

The primary objective of this project is to develop an efficient and user-friendly music suggestion engine that enhances the listening experience and reduces decision fatigue.

ABSTRACT

A Music Recommender System uses machine learning techniques to recommend songs that match a user's taste. In this project, we implemented a model capable of learning from user ratings, song features and similarity matrices. The system aims to:

- Understand user preferences
- Recommend songs based on historical data
- Improve user engagement on music platforms
- Reduce manual search time

I used Python, Pandas, NumPy, Scikit-Learn, and similarity algorithms to analyze song datasets and generate meaningful recommendations. The outcome is an interactive and efficient system that provides personalized suggestions based on either the input song or the user's listening profile.

AIM OF THIS PROJECT

The main aim of this project is to build a recommendation engine that predicts songs a user is likely to enjoy, based on patterns observed in the dataset. It focuses on:

- Personalized music suggestions
- Improving discovery of new songs
- Understanding listener behavior
- Implementing machine learning models for real-world usage

- -

Date:

MAIN PURPOSE

1. Reduce the time users spend searching for songs
2. Automatically generate playlists tailored to the listener
3. Provide recommendations using data-driven algorithms
4. Enhance user experience on music platforms
5. Demonstrate the practical application of machine learning

MAIN GOALS

1. User Satisfaction: Deliver music that fits mood, genre preference and listening habit
2. Time Saving: No need for endless scrolling
3. Accuracy: Use ML algorithms to improve predictions
4. Learning Patterns: Understand trends and preferences using data
5. Scalability: System should easily handle large music libraries

METHODS USED

- Content-Based Filtering
  Recommends songs based on musical features: genre, tempo, mood, artist similarity.

- Collaborative Filtering
  Suggests songs based on what similar users like.

- Cosine Similarity / KNN
  Used for measuring similarity between songs or user profiles.

- Python Libraries: Pandas, NumPy, Scikit-Learn

- Dataset: Music metadata or public song datasets such as Spotify or MillionSong Dataset

MODULES

1. User Module
    - Input their favorite song/artist/mood
    - View recommended songs
    - Create auto-generated playlists

2. Recommendation Module
    - Analyze data
    - Compute similarities
    - Return the top recommended tracks

3. Dataset Module
    - Read and process music metadata
    - Handle missing values
    - Store structured song information

BENEFITS OF A MUSIC RECOMMENDER SYSTEM

1. Personalized music discovery
2. Enhanced user engagement
3. Saves time and effort
4. Helps emerging artists get noticed through matching algorithms
5. Makes large music libraries easy to navigate

FUTURE SCOPE

- Integration with a mobile app
- Voice-based music discovery
- Mood-based recommendations using sentiment analysis
- Deep-learning models for higher accuracy
- Real-time dynamic playlist generation

CONCLUSION

The Music Recommender System greatly improves the music discovery experience by using machine learning to suggest personalized songs. It analyzes listening habits, predicts preferences and makes intelligent recommendations. With further development, this system can be integrated into modern music platforms to create smart, adaptive and highly engaging user experiences.

PROGRAM

```
!pip install streamlit
!pip install spotipy
import pickle
import streamlit as st
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

CLIENT_ID = "70a9fb89662f4dac8d07321b259eaad7"
CLIENT_SECRET = "4d6710460d764fbbb8d8753dc094d131"

# Initialize the Spotify client
client_credentials_manager = SpotifyClientCredentials(client_id=CLIENT_ID, client_secret=CLIENT_SEC
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

def get_song_album_cover_url(song_name, artist_name):
    search_query = f"track:{song_name} artist:{artist_name}"
    results = sp.search(q=search_query, type="track")

    if results and results["tracks"]["items"]:
        track = results["tracks"]["items"][0]
        album_cover_url = track["album"]["images"][0]["url"]
        print(album_cover_url)
        return album_cover_url
    else:
        return "https://i.postimg.cc/0QNxYz4V/social.png"

def recommend(song):
    index = music[music['song'] == song].index[0]
    distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])
    recommended_music_names = []
    recommended_music_posters = []
    for i in distances[1:6]:
        # fetch the movie poster
        artist = music.iloc[i[0]].artist
        print(artist)
        print(music.iloc[i[0]].song)
        recommended_music_posters.append(get_song_album_cover_url(music.iloc[i[0]].song, artist))
        recommended_music_names.append(music.iloc[i[0]].song)
```

```
    return recommended_music_names,recommended_music_posters

st.header('Music Recommender System')

try:
    music = pickle.load(open('df.pkl','rb'))
    similarity = pickle.load(open('similarity.pkl','rb'))

    music_list = music['song'].values
    selected_movie = st.selectbox(
        "Type or select a song from the dropdown",
        music_list
    )

    if st.button('Show Recommendation'):
        recommended_music_names,recommended_music_posters = recommend(selected_movie)
        col1, col2, col3, col4, col5= st.columns(5)
        with col1:
            st.text(recommended_music_names[0])
            st.image(recommended_music_posters[0])
        with col2:
            st.text(recommended_music_names[1])
            st.image(recommended_music_posters[1])

        with col3:
            st.text(recommended_music_names[2])
            st.image(recommended_music_posters[2])
        with col4:
            st.text(recommended_music_names[3])
            st.image(recommended_music_posters[3])
        with col5:
            st.text(recommended_music_names[4])
            st.image(recommended_music_posters[4])
except FileNotFoundError:
    st.error("Error: 'df.pkl' or 'similarity.pkl' not found. Please upload these files to your Colab environment
```

OUTPUT

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.12/dist-packages (1.51.0)
Requirement already satisfied: altair!=5.4.0,!=5.4.1,<6,>=4.0 in /usr/local/lib/python3.12/dist-packages (from stre
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (1.9.0
Requirement already satisfied: cachetools<7,>=4.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (5.5.
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (8.3.1)
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.12/dist-packages (from streamlit) (2.0.2)
Requirement already satisfied: packaging<26,>=20 in /usr/local/lib/python3.12/dist-packages (from streamlit) (25.0)
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<13,>=7.1.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (11.3.
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.12/dist-packages (from streamlit) (5.29.
Requirement already satisfied: pyarrow<22,>=7.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (18.1.0
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.12/dist-packages (from streamlit) (2.32.
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.12/dist-packages (from streamlit) (8.5
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.12/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/python3.12/dist-packages (from streaml
Requirement already satisfied: watchdog<7,>=2.1.5 in /usr/local/lib/python3.12/dist-packages (from streamlit) (6.0.
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.12/dist-packages (from strea
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3.12/dist-packages (from streamlit) (0.9.
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in /usr/local/lib/python3.12/dist-packages (from streamlit
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4.1,<6,>=4
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.4
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.12/dist-packages (from altair!=5.4.0,!=5.
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.12/dist-packages (from gitpython!=3.1.19,<
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->stre
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>=1.4.0->st
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.27->str
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.2
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.12/dist-packages (from gitdb<5,>=4.0.1->gi
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->altair!=5.4
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->alta
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=3.0->alt
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pa
```

THANK YOU