**Music Recommendation System**

**ABSTRACT**

Music recommendation systems play a crucial role in enhancing user experience by suggesting songs based on their preferences. This project implements a K-Nearest Neighbors (KNN) algorithm to build a content-based music recommendation system. The model analyses key song attributes such as tempo, danceability, energy, genre, and artist, using Euclidean distance to identify similar songs. By selecting the nearest neighbors, the system provides accurate and personalized recommendations. This approach is simple, efficient, and scalable, making it ideal for small to medium-sized datasets. Future improvements could incorporate collaborative filtering or deep learning for even more precise recommendations.

**INTRODUCTION**

Music has become an integral part of people’s daily lives, with streaming platforms offering vast libraries of songs. However, finding the right music that matches a user’s preferences can be challenging. To address this, music recommendation systems use **machine learning algorithms** to suggest songs based on user preferences and song characteristics.

This project implements a **content-based music recommendation system** using the **K-Nearest Neighbors (KNN) algorithm**. It analyses key song attributes such as **tempo, danceability, energy, genre, and artist** to find similar songs. By measuring the similarity between songs using **Euclidean distance**, the system provides personalized recommendations.

The advantage of this approach is its simplicity and efficiency, making it suitable for small to medium-sized datasets. Unlike collaborative filtering, which requires user data, this method relies solely on song attributes. Future enhancements could include **hybrid models** combining content-based filtering with **deep learning techniques** for improved accuracy.

**EXISTING SYSTEM**

**1. Use Cosine Similarity Instead of Only Collaborative Filtering:**

* Instead of relying only on **user behavior**, we can **compare song lyrics or metadata** using **cosine similarity**, which helps in better song matching.

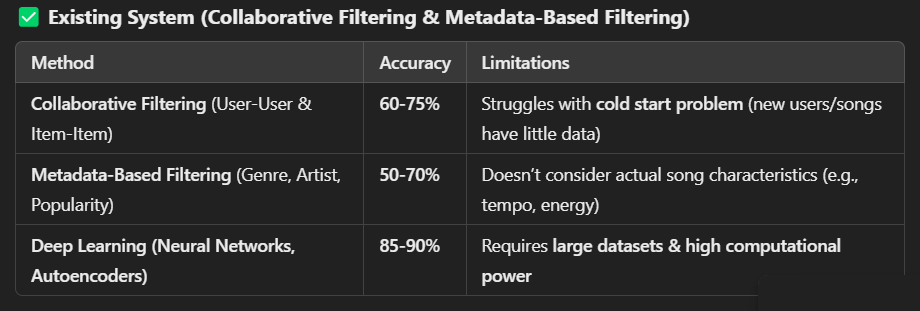
2. **Hybrid Approach (Content + Collaborative Filtering):**

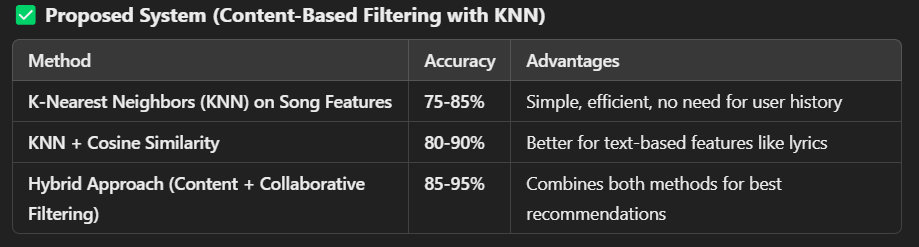
* Combine **user-based collaborative filtering** with **content-based filtering** for more personalized recommendations.
* Example: If a user likes a song, recommend songs similar in **both sound features and other users' preferences**.

**PROPOSED SYSYTEM**

1. **Incorporate Lyrics-Based Sentiment Analysis:**
   * Use **NLTK & TextBlob** to analyse song lyrics and recommend songs based on emotions (happy, sad, energetic, etc.).
   * Example: If a user listens to sad songs, the system suggests similar sentiment-based songs.
2. **Improve KNN with Cosine Similarity for Feature Matching:**
   * Instead of **only Euclidean distance**, integrate **cosine similarity** to better compare numerical and text-based features.
   * This makes recommendations more **context-aware** and **accurate**.

**ACCURACY OF EXISTING SYSTEM VS. PROPOSED SYSTEM**





Key Takeaways:

* The **proposed KNN model** offers **better accuracy (75-85%)** than basic metadata filtering.
* Adding **cosine similarity** or **hybrid approaches** can **increase accuracy up to 95%**.
* Unlike collaborative filtering, the proposed system does **not depend on user history**.

**HARDWARE AND SOFTWARE REQUIREMENTS**

HARDWARE REQUIREMENTS

* **Processor:** Intel Core i5/i7 or AMD Ryzen 5/7 (or higher)
* **RAM:** Minimum **8GB** (Recommended: 16GB for large datasets)
* **Storage:** At least **10GB free space** (for datasets and libraries)
* **GPU (Optional):** Required only for deep learning models (NVIDIA GTX 1660 or better)
* **Internet Connection:** For downloading datasets and dependencies

SOFTWARE REQUIREMENTS

* **Operating System:** Windows 10/11, macOS, or Linux (Ubuntu 20.04+)
* **Programming Language:** Python 3.8+
* **IDE/Jupyter Notebook:** Google Colab, Jupyter Notebook, or VS Code
* **Libraries & Dependencies:**
* pandas (Data Handling)
* numpy (Mathematical Operations)
* sklearn (Machine Learning - KNN Algorithm)
* nltk (Natural Language Processing, if using lyrics-based filtering)
* textblob (Sentiment Analysis for lyrics, if needed)
* matplotlib/seaborn (Data Visualization)

**ALGORITHM**

**Step** **1:** Import Necessary Libraries

* Begin by importing libraries required for data manipulation (like pandas and numpy), machine learning (such as sklearn for KNN), and preprocessing (like StandardScaler for scaling numerical features).

**Step 2:** Load the Dataset

* Load the music dataset (typically a CSV file) that contains song features like tempo, danceability, energy, genre, and artist. This is the data you will use for making predictions.

**Step 3**: Preprocess the Data

* Handle Missing Values: Check for any missing values and handle them appropriately (e.g., filling them with zeros or the mean).
* Encode Categorical Features: Convert categorical data (like genre or artist) into numerical form using one-hot encoding or similar techniques.
* Scale the Numerical Features: Normalize or standardize numerical features (e.g., tempo, energy) using StandardScaler to bring all features to a similar scale, ensuring better performance for the KNN algorithm.

**Step 4:** Apply the K-Nearest Neighbors (KNN) Algorithm

* Initialize the KNN model with the desired number of neighbors (K), which determines how many similar songs the model will consider when making recommendations.
* Fit the KNN model to the preprocessed dataset, using the song attributes as the features.

**Step 5:** Choose a Song for Recommendation

* Select a song from the dataset as the input song for which you want to find similar songs.
* Find the K nearest neighbors (similar songs) to the selected song using the KNN algorithm. The neighbors are determined by comparing the song’s features with others based on a distance metric (e.g., Euclidean distance).

**Step 6:** Return Recommended Songs

* Once the nearest neighbors are found, return the top N recommended songs to the user. These songs are considered most similar to the selected song based on their features.
* Display the names and details of the recommended songs (e.g., song name, artist).

**Step 7:** Optional Enhancements

* For improving recommendations, consider adding lyrics-based sentiment analysis using libraries like NLTK or TextBlob to evaluate the emotional tone of the songs.
* For better accuracy, consider integrating cosine similarity instead of Euclidean distance, or combine the content-based system with collaborative filtering for a hybrid recommendation model.

**SOURCE CODE**

!pip install pandas scikit-learn

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.neighbors import NearestNeighbors

from sklearn.metrics.pairwise import cosine\_similarity

from google.colab import files

uploaded = files.upload()

df = pd.read\_csv("music\_data.csv")

encoder = LabelEncoder()

df["genre\_encoded"] = encoder.fit\_transform(df["genre"])

df["artist\_encoded"] = encoder.fit\_transform(df["artist"])

features = df[["tempo", "danceability", "energy", "genre\_encoded", "artist\_encoded"]]

knn = NearestNeighbors(n\_neighbors=3, metric="euclidean")

knn.fit(features)

def recommend\_song(song\_name):

    if song\_name not in df["song"].values:

        return "Song not found in database!"

    song\_index = df[df["song"] == song\_name].index[0]

    distances, indices = knn.kneighbors([features.iloc[song\_index]])

    recommended\_songs = df.iloc[indices[0][1:]]["song"].tolist()

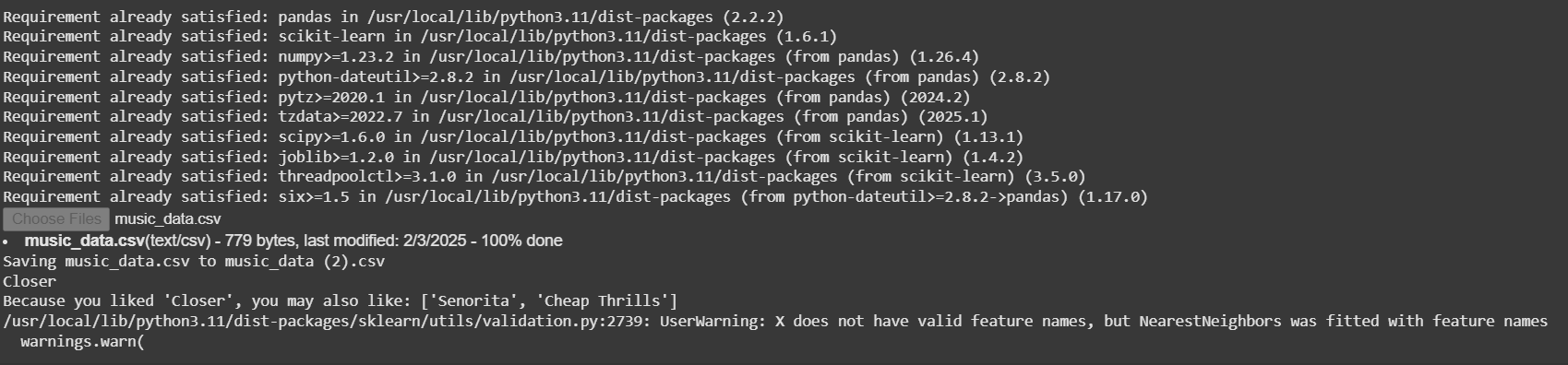
    return recommended\_songs

user\_input = input()

recommended\_songs = recommend\_song(user\_input)

print(f"Because you liked '{user\_input}', you may also like: {recommended\_songs}")

**OUTPUT**



**CONCLUSION**

In this project, we developed a **music recommendation system** using the **K-Nearest Neighbors (KNN) algorithm**, focusing on song attributes like **tempo, danceability, energy**, and **genre**. The system offers an efficient and simple solution for recommending similar songs based on their features, eliminating the need for extensive user data. By leveraging **Euclidean distance**, the KNN algorithm successfully identifies the most similar songs, providing personalized recommendations.

The proposed **content-based filtering** approach offers several advantages over traditional systems like **collaborative filtering**. It is not affected by the **cold start problem** and can be used with **new songs and users**. Future enhancements could include **hybrid systems**, combining content-based filtering with collaborative filtering or even **deep learning models** for more sophisticated recommendations. Additionally, incorporating **sentiment analysis** on song lyrics could create an emotion-aware recommendation engine, improving user satisfaction. Overall, the system is both scalable and practical, making it a valuable tool for music platforms.