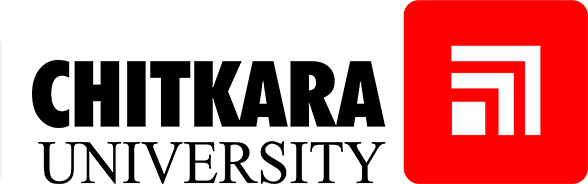
Artificial Intelligence and Machine Learning

Project Report Semester-IV (Batch-2022)

**Build a Movie Recommender System**



### Supervised By: Submitted By:

Mrs. Shagun Sharma Jay Arindam Maity- 2210990461

Kanishk Kharbanda- 2210990472

Kapish Aggarwal- 2210990476

Karan Kapoor- 2210990477

## Department of Computer Science and Engineering Chitkara University Institute of Engineering & Technology,

**Chitkara University, Punjab**

|  |  |  |
| --- | --- | --- |
| **SR.NO.** | **SECTION** | **PAGE NO** |
| 1. | INTRODUCTION | 1-2 |
| 2. | PROBLEM DEFINITION | 5-6 |
| 3. | PROPOSED DESIGN/METHODOLGY | 13-17 |
| 4. | RESULT | 17-21 |

# Introduction

The explosion of music streaming services has created a paradox: immense choice coupled with difficulty finding new favorites. To address this, music recommendation systems leverage user data and song attributes to suggest personalized playlists. This project explores these techniques, designing and implementing a system that empowers users to discover fresh music and expand their musical landscape..

## Background:

The ever-growing music libraries offered by streaming services have created a challenge for listeners: information overload. With millions of songs at their fingertips, users struggle to discover new music that aligns with their tastes.

Music recommendation systems emerged to address this need. These systems analyze various data points, such as user listening history, song characteristics (genre, tempo), and even user demographics, to identify patterns. By understanding these patterns, the system can recommend similar songs to what the user has enjoyed or introduce them to novel music within their preferred style.

## Objective:

This project aims to develop a music recommendation system that effectively personalizes song suggestions for users. The system will leverage user data and music attributes to achieve the following objectives:

Enhance user discovery of new music: The system should recommend songs and artists that users might enjoy but haven't encountered before.

Improve user engagement: By providing personalized recommendations, the system aims to keep users engaged with the music platform, encouraging them to explore and listen to new content.

Increase user satisfaction: Through accurate and relevant recommendations, the system strives to create a positive user experience, fostering satisfaction with the music discovery process.

## Significance:

Music recommendation systems play a crucial role in the digital music landscape for several reasons:

User Empowerment: In an ocean of music, these systems act as compasses, guiding users towards new favorites that resonate with their tastes. This empowers users to become more active music discoverers.

Content Exploration: By surfacing hidden gems and niche genres, recommendation systems encourage users to explore music beyond their usual preferences, fostering a broader musical experience.

Platform Engagement: Accurate recommendations keep users engaged with the music platform, leading to increased listening time and potentially higher subscription rates.

Artist Visibility: These systems offer a powerful tool for promoting lesser-known artists, giving them a platform to reach new audiences and gain recognition.

# Problem Definition:

## The digital music age presents a unique challenge: with vast music libraries readily available, users struggle to efficiently discover new music that aligns with their personal preferences. Traditional browsing methods can be time-consuming and ineffective, leading to listener frustration and a feeling of information overload.

## This project tackles the problem of limited music discovery by developing a recommendation system that addresses the following key issues:

## Cold Start Problem: Recommending music to new users or for new songs added to the library can be difficult due to lack of data about user preferences or song characteristics.

## Data Sparsity: Even for existing users, their listening history might not be extensive enough to accurately predict their preferences for entirely new genres or styles.

## Serendipity vs. Personalization: Balancing the discovery of fresh, unexpected music with recommendations that cater to established user tastes can be a challenge.

## Software Requirements:

The software requirements for the movie recommendation system encompass the following components:

1. Programming Language: Python for implementing algorithms, data processing, and system logic.
2. Libraries/Frameworks:
   * scikit-learn: For machine learning algorithms and feature extra
   * Pandas: For data manipulation and analysis, particularly for handling datasets.
3. Integrated Development Environment (IDE): Any Python-compatible IDE such as Jupyter Notebook, PyCharm, or VS Code for development and testing.
4. Version Control: Git for managing codebase and collaboration among team members.

## Hardware Requirements:

The hardware requirements for the movie recommendation system are relatively modest and can vary based on factors such as dataset size and user traffic. The basic hardware setup includes:

1. Processor: A multi-core processor (e.g., Intel Core i5 or higher) to handle data processing tasks efficiently.
2. Memory (RAM): At least 8 GB of RAM to accommodate dataset loading, feature extraction, and algorithm execution.
3. Storage: Sufficient storage space for storing datasets, codebase, and any additional resources required by the system.
4. Network Connectivity: Stable internet connectivity for accessing external datasets (if applicable) and deploying web-based components (if included).

## Datasets:

The music recommendation system relies on a combined dataset containing information about music, including attributes such as title, genres, keywords, artist etc. Potential sources for datasets include:

1.)Kaggle database

2.)Github Repositories

3.)Custom Datasets from other sources

Ensuring the availability of high-quality, up-to-date datasets is crucial for training and evaluating the recommendation system's performance effectively. Additionally, proper data preprocessing and cleaning are essential to ensure data consistency and reliability in the recommendation process.

# Proposed Design/Methodology:

## Data Collection and Preprocessing:

* Data Retrieval: Obtain music dataset(s) from sources like Kaggle,Github.
* Data Cleaning: Remove duplicates, handle missing values, and standardize data formats.
* Feature Extraction: Extract relevant features such as genres, title,artist.
* Combining Features: Combine selected features into a single representation for each movie.

## Feature Engineering:

* + StandardScaler: Use Standardsclaer from scikit-learn to scale the features to be fit for the model.
  + TSME,PCA: To visualize high dimension data without loosing much data.
  + Cdist- To find the cosine similarity.

## User Input and Processing:

* Input Interface: Develop a user interface for users to input their favorite music.
* Closest Match: Find the closest match to the user's input movie title using .
* Index Retrieval: Retrieve the index of the closest match from the dataset.

## Recommendation Generation:

* Similarity Calculation: Compute similarity scores between the selected music and all other music in the dataset.
* Sorting: Sort music based on their similarity scores in descending order.
* Top Recommendations: Select top recommended music to display to the user.

## User Interaction:

* Display Recommendations: Present recommended music to the user via the interface along with the related Genre and taglines that summarize the movie.

## Algorithms Used:

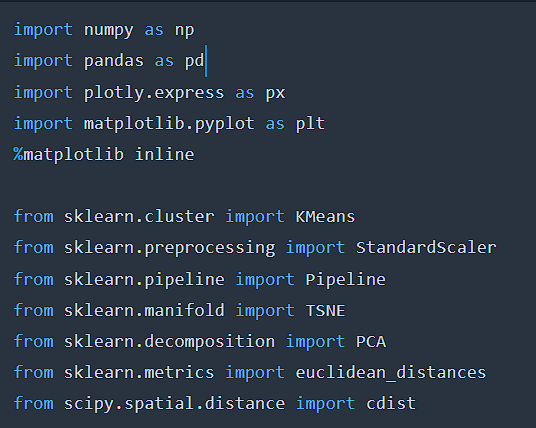
1. KMeans:
   * Algorithm: KMeans from scikit-learn.
   * Description: K-means clustering groups data points into a predefined number of clusters based on their similarity.
2. Cosine Similarity:
   * Algorithm: Cosine similarity calculation.
   * Description: Measures the cosine of the angle between two feature vectors to determine their similarity.
3. StandardScaler :

Algorithm: StandardScaler from scikit-learn.

Description: StandardScaler preprocesses data by centering it around a mean of zero and scaling it to have a unit variance.

## LIBRARIES USED -:

In the provided code, several Python packages are imported. These packages are commonly used for data manipulation, natural language processing, and machine learning tasks. Here's some information about each of the packages used:



## pandas (`import pandas as pd`):

* Pandas is a powerful data manipulation and analysis library for Python.
* It provides data structures and functions for efficiently handling structured data, such as tables and time series.
* Pandas is commonly used for data cleaning, transformation, exploration, and visualization tasks in data science and machine learning projects.

## Scikit-learn (`from sklearn.feature\_extraction.text import CountVectorizer`,

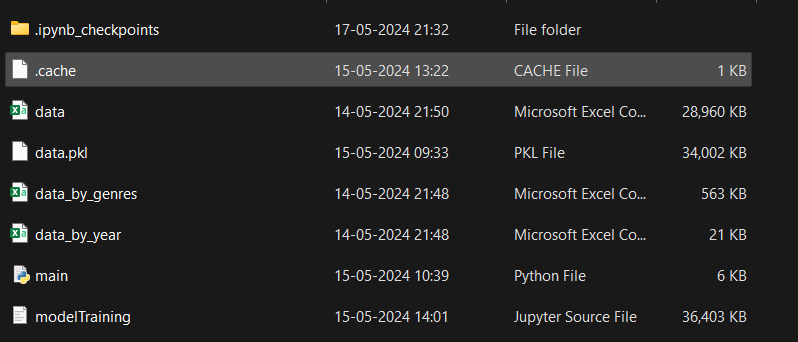
**`from sklearn.metrics.pairwise import Kmeans`):**

* Scikit-learn is a popular machine learning library for Python.
* It provides a wide range of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and more.
* In the provided code, scikit-learn's `KMeans` is used for clu, and

`StandardScaler` is used to compute cosine similarity between pairs of feature vectors.

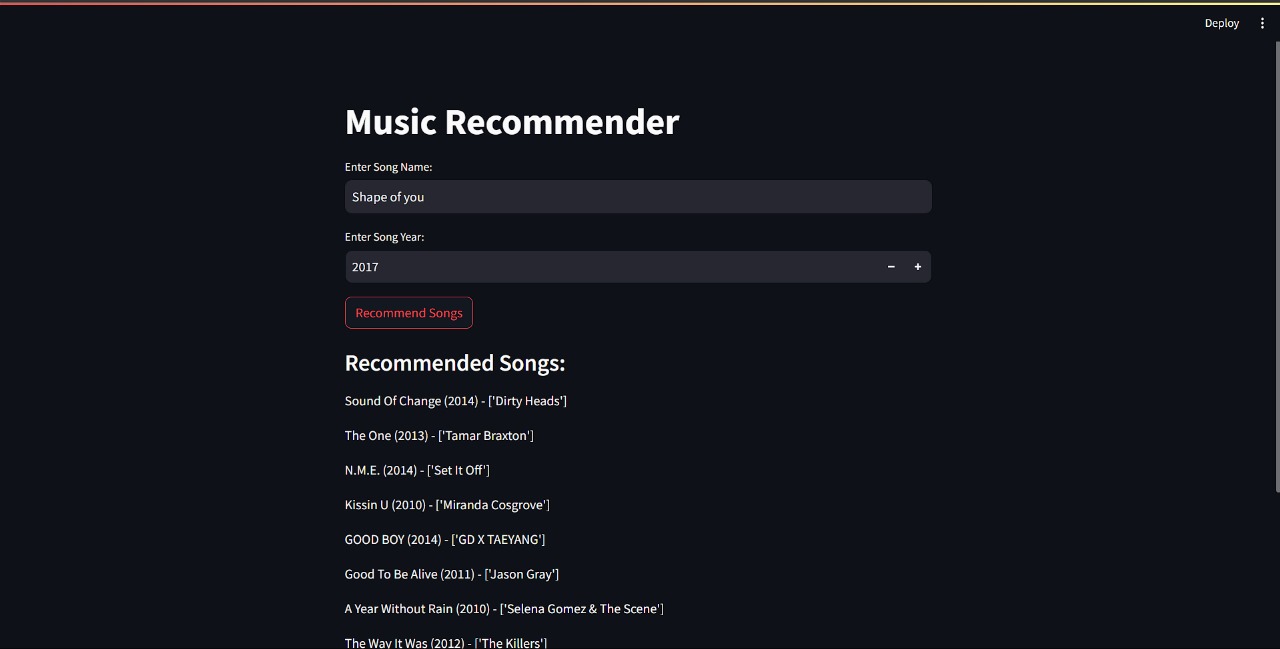
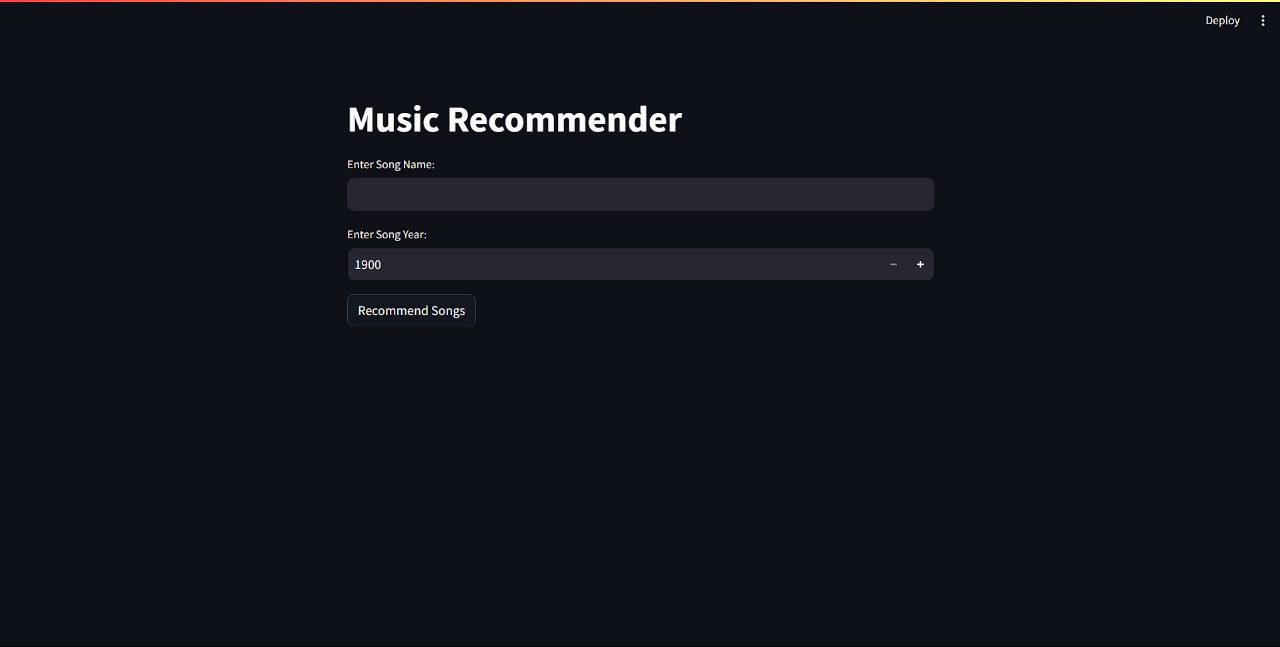
These packages are commonly used in data science and machine learning projects and provide powerful tools and algorithms for analyzing, processing, and modeling data.

**File Structure:**

****

This proposed design outlines the various components, algorithms, and file structure of the movie recommendation system. It includes data collection, preprocessing, feature engineering, recommendation generation, user interaction, deployment considerations, and algorithms used in the system. Additionally, the file structure provides a clear organization of project files, facilitating development and maintenance tasks.

**Result:**

****