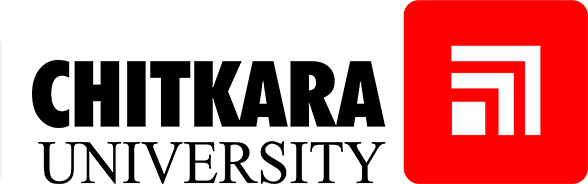
Artificial Intelligence and Machine Learning

Project Report Semester-IV (Batch-2022)

**Build a Movie Recommender System**



### Supervised By: Submitted By:

Mrs. Shagun Sharma Jaivardhan Khanna- 429

Jashan Jot Kaur- 436

Jasmeet Singh- 449

Jasmine Rathor- 450

## 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 | 3-4 |
| 3. | PROPOSED DESIGN/METHODOLGY | 5-9 |
| 4. | RESULT | 10 |

# Introduction

In the era of overwhelming choices, recommending personalized content has become crucial in enhancing user experience. The film industry is no exception, with countless movies released each year, users often find it challenging to discover films aligned with their preferences. To address this issue, we present a Movie Recommender System leveraging Python and scikit-learn.

## Background:

In today's digital age, the sheer volume of available content, particularly in the realm of movies and entertainment, can be overwhelming for consumers. With countless options available across various platforms, users often seek personalized recommendations to discover content that aligns with their preferences. Traditional methods of content discovery, such as manual browsing or generic recommendations, may not suffice in providing tailored suggestions. Hence, the development of advanced recommendation systems leveraging artificial intelligence and machine learning techniques has become increasingly important.

## Objective:

The primary objective of this project is to design and implement a movie recommendation system based on artificial intelligence and machine learning. The system aims to provide users with personalized movie suggestions by analyzing the characteristics and features of films. By leveraging techniques such as content-based filtering, the system can understand user preferences and recommend movies that are similar to those the user has enjoyed in the past. Through the utilization of advanced algorithms and data processing methods, the system strives to enhance the user experience by facilitating seamless content discovery.

## Significance:

The significance of this project lies in its potential to revolutionize the way users interact with movie platforms and streaming services. By offering personalized recommendations, the system can assist users in discovering new and relevant content that aligns with their interests and preferences. This not only enhances user satisfaction but also improves engagement and retention rates for movie platforms. Furthermore, the project showcases the practical application of artificial

intelligence and machine learning in addressing real-world challenges, highlighting the transformative power of these technologies in the entertainment industry. Overall, the development of an intelligent movie recommendation system represents a significant step towards creating more personalized and enriching content experiences for users worldwide.

# Problem Definition:

The problem statement revolves around developing an intelligent movie recommendation system utilizing artificial intelligence and machine learning techniques. The objective is to create a system that can analyze user preferences and recommend movies based on their characteristics and features. The system should be capable of providing personalized suggestions to enhance the user experience and facilitate seamless content discovery. Key aspects of the problem definition include:

* Designing algorithms for content-based filtering to recommend movies similar to those the user has enjoyed.
* Ensuring scalability and efficiency to handle large datasets and accommodate growing user bases.
* Displaying some recommendable movies to the user listed along with their Genres and Taglines.

## 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 extraction (CountVectorizer, cosine\_similarity).
   * Pandas: For data manipulation and analysis, particularly for handling datasets.
   * PorterStemmer: For finding close matches between movie titles(by removing suffixes from english words and obtain the stem)
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 movie recommendation system relies on a combined dataset containing information about movies, including attributes such as title, genres, keywords, cast, director, etc. Potential sources for datasets include:

1.)ImDB Database

2.)Kaggle database

3.)Github Repositories

4.)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 movie dataset(s) from sources like MovieLens, IMDb, or Kaggle.
* Data Cleaning: Remove duplicates, handle missing values, and standardize data formats.
* Feature Extraction: Extract relevant features such as genres, keywords, cast, and director.
* Combining Features: Combine selected features into a single representation for each movie.

## Feature Engineering:

* + CountVectorizer: Use CountVectorizer from scikit-learn to convert text features into numerical vectors.
  + Cosine Similarity: Compute cosine similarity between feature vectors to measure similarity between movies.

## User Input and Processing:

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

## Recommendation Generation:

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

## User Interaction:

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

## Algorithms Used:

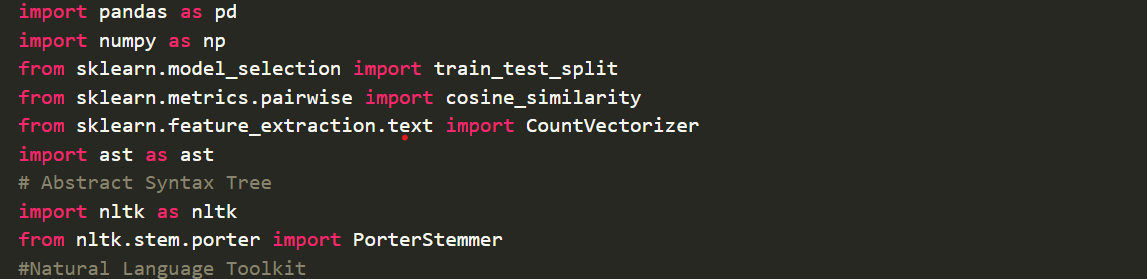
1. CountVectorization:
   * Algorithm: CountVectorizer from scikit-learn.
   * Description: Converts text data into numerical feature vectors using token scores.
2. Cosine Similarity:
   * Algorithm: Cosine similarity calculation.
   * Description: Measures the cosine of the angle between two feature vectors to determine their similarity.
3. PorterStemmer :

Algorithm: PorterStemmer from nltk(natural language toolkit)

Description: Remove suffixes from english words and obtain the stem(a part of a word common to all its variants)

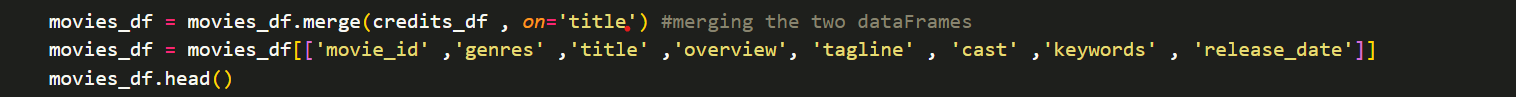
## LIBRARIES USED -:

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`):

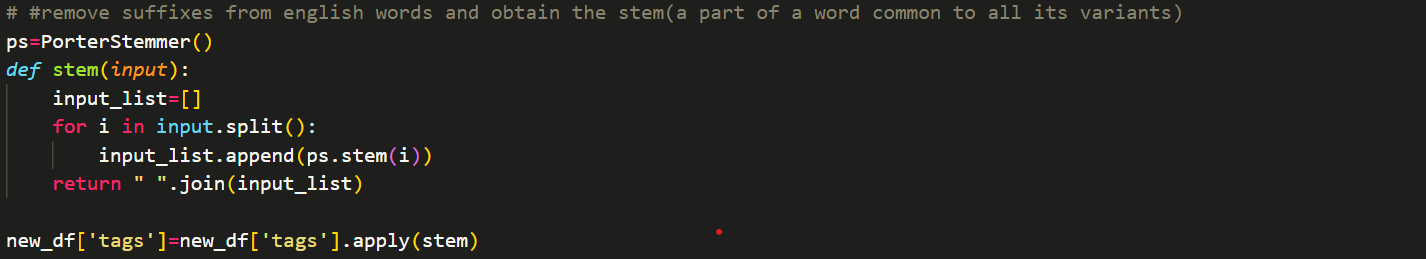
* Pandas is a powerful data manipulation and analysis library for Python.
* It provides data structures and functions for efficiently handling structured data, such as tables and time series.
* Pandas is commonly used for data cleaning, transformation, exploration, and visualization tasks in data science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





## PorterStemmer (`from nltk import PorterStemmer`):

* The `PorterStemmer` module provides tools for comparing sequences and calculating the differences between them.
* It includes functions for performing basic string matching, such as finding the longest common subsequence or computing the similarity ratio between strings.
* `PorterStemmer` is often used for tasks like approximate string matching, text diffing, and finding close matches in a list of strings.



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

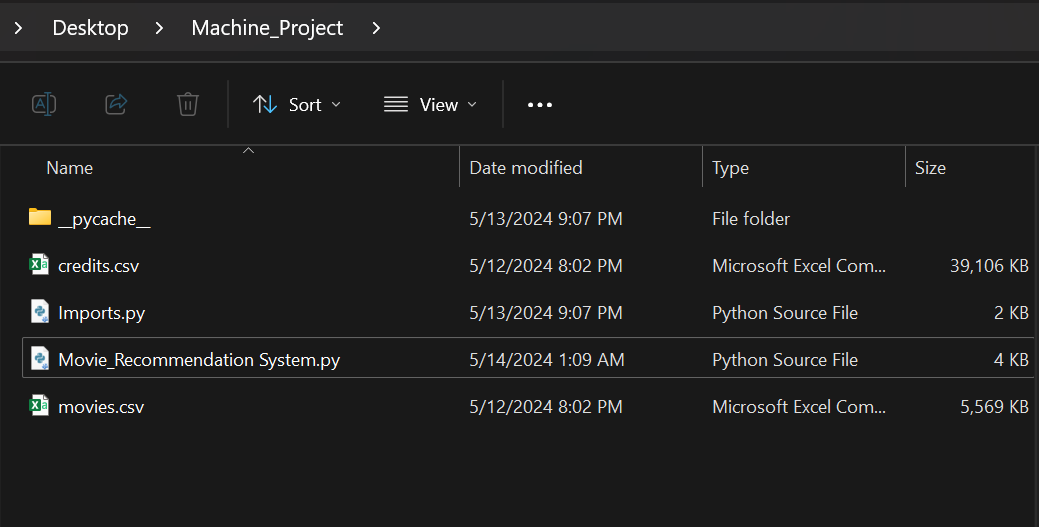
**`from sklearn.metrics.pairwise import cosine\_similarity`):**

* 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 `CountVectorizer` is used for text vectorization, and

`cosine\_similarity` 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

# The *new\_df* dataset is updated with the new values and can be seen as below.



# The user is prompted with a line asking for a movie’s name to ask for recommendations.

