# Task-Five:

# Movie Recommendation System

## Project Overview

This project builds a **Content-Based Movie Recommendation System** using the **TMDB 5000 Movies dataset**.  
It uses **TF-IDF Vectorization** on movie overviews and **Cosine Similarity** to recommend similar movies.

## Dataset

* Source: TMDB 5000 Movies dataset (tmdb\_5000\_movies.csv)
* Size: ~4,803 movies
* Key Columns Used:

title – Movie title

overview – Short description of the movie

## Approach

### 1. Data Preprocessing

* Missing values in overview are filled with an empty string.
* The overview column is converted into a format suitable for text processing.

### 2. Feature Extraction (TF-IDF)

* **TF-IDF (Term Frequency–Inverse Document Frequency)** is applied to the movie overviews
* This represents each movie’s overview as a **vector of word importance scores**.

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf.fit\_transform(df['overview'].values.astype('U'))

* Shape of TF-IDF matrix: (4803, 20978)

### 3. Similarity Computation (Cosine Similarity)

* **Cosine Similarity** is calculated between all movie vectors.
* This gives a similarity score between every pair of movies.

from sklearn.metrics.pairwise import cosine\_similarity

cosine\_sim = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

* Shape of similarity matrix: (4803, 4803)

### 4. Recommendation Function

def recommend(movie\_title, cosine\_sim=cosine\_sim, df=df):

idx = df[df['title'] == movie\_title].index[0]

sim\_scores = list(enumerate(cosine\_sim[idx]))

sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)

sim\_scores = sim\_scores[1:6] # Top 5 similar movies

movie\_indices = [i[0] for i in sim\_scores]

return df['title'].iloc[movie\_indices]

## Sample Output

### Recommendations for Avatar

* Apollo 18
* The American
* The Matrix
* The Inhabited Island
* Tears of the Sun

### Recommendations for The Dark Knight Rises

* The Dark Knight
* Batman Forever
* Batman Returns
* Batman
* Batman: The Dark Knight Returns

## Key Features

* Content-based filtering (does not require user ratings).
* Simple implementation with TF-IDF and cosine similarity.
* Scalable to thousands of movies.

## Future Improvements

* Use **genres, keywords, cast, and crew metadata** for better recommendations.
* Implement **hybrid systems** by combining content-based and collaborative filtering.
* Build a **web app** using Streamlit or Flask for user interaction.

窗体底端