**AI-ML Internship Project**

**Cine Buddy – Movie Recommender System**

****

**Submitted At Elevate Labs**

Developed by: Sujal Thakor

GitHub Repo : <https://github.com/Sujal17-git/Cine_Buddy.git>

**CineBuddy – A Smart Movie Recommendation System**

**Introduction**

CineBuddy is an intelligent, content-based movie recommendation system designed to help users discover films similar to their favorites. Built using Python and Streamlit, this project bridges machine learning with web development to offer an intuitive and appealing movie exploration experience. In an age of overwhelming media choices, CineBuddy serves as a smart assistant that simplifies the decision-making process for users by suggesting relevant and engaging movies based on their preferences.

**Objective**

The primary objective of CineBuddy is to provide a seamless movie recommendation experience that is both accurate and interactive. By leveraging natural language processing techniques on movie metadata and combining them with a user-friendly interface, the application aims to enhance content discovery for users. The project is focused on deploying a lightweight solution that does not rely on user history or collaborative filters but rather uses content-based similarity for immediate and contextually relevant results. Additional goals include ensuring aesthetic design with options like dark mode, visual representation of IMDb ratings, and interactive filters such as genre selection.

**Datasets and Preprocessing**

CineBuddy uses two primary datasets sourced from the publicly available **TMDb 5000 Movie Dataset** on Kaggle. These datasets include tmdb\_5000\_movies.csv and tmdb\_5000\_credits.csv. The first file contains essential movie metadata such as titles, genres, keywords, overviews, and IDs. The second file provides detailed information about cast and crew members associated with each film. These datasets were merged using the movie ID as the common key to form a unified dataset. Various columns such as genres, keywords, overview, cast, and crew were extracted and combined into a single metadata field referred to as a “tag” for each movie. These tags serve as the foundational input for the content-based filtering model.

Before modeling, all tags were normalized by converting to lowercase, removing spaces, and eliminating punctuation where necessary. This cleaning process ensures uniformity in vectorization and enhances the accuracy of the similarity algorithm. Vectorization was performed using either **CountVectorizer** or **TF-IDF Vectorizer** to convert the text-based metadata into numerical vectors suitable for similarity comparison. Finally, **cosine similarity** was computed across the vectorized data to identify movies with the highest content resemblance.

**API Integration**

CineBuddy enhances its recommendation experience by integrating with **The Movie Database (TMDb) API** to dynamically fetch movie posters and related metadata. While the core recommendation engine operates on a static dataset, the TMDb API is used to retrieve high-quality poster images based on each movie's unique identifier. This API integration ensures that the recommendations are not only accurate but also visually engaging, offering users a more immersive experience. By calling TMDb’s endpoints using Python’s requests library, the application retrieves poster paths and composes full image URLs, which are then displayed alongside movie titles and IMDb ratings in the UI. This approach bridges the gap between static data processing and real-time visual presentation

.

**Recommendation Logic**

The core recommendation engine in CineBuddy is built on content-based filtering using a bag-of-words approach. For any given movie selected by the user, the system retrieves its associated metadata vector and compares it with all other movie vectors in the dataset. The cosine similarity scores between the vectors are calculated to measure how closely related two movies are in terms of their content. The top 10 movies with the highest similarity scores are then selected and displayed as recommendations. This approach allows the system to remain effective without requiring prior user interactions, ratings, or preferences.

**Application Interface and Features**

The front-end of CineBuddy is developed using **Streamlit**, which allows for rapid development of interactive Python-based web applications. The user interface is clean and minimal, with a focus on user experience. Users can enter a movie title in a search bar, upon which the system processes the input and displays a list of similar movies along with their posters and IMDb ratings. One of the notable UI features is the **genre filter**, which allows users to narrow down the recommendations to a particular genre or opt for "None" to see general suggestions. The application also includes a **dark mode toggle** for visual comfort, especially for users browsing at night.

Another unique aspect of the UI is the **IMDb rating display**, which converts numerical values into star icons for better visual representation. Poster images for the recommended movies are retrieved using the movie IDs and TMDb poster paths, adding a rich, immersive layer to the recommendation experience. The combination of clean visuals, responsive design, and real-time recommendations makes CineBuddy both efficient and enjoyable to use.

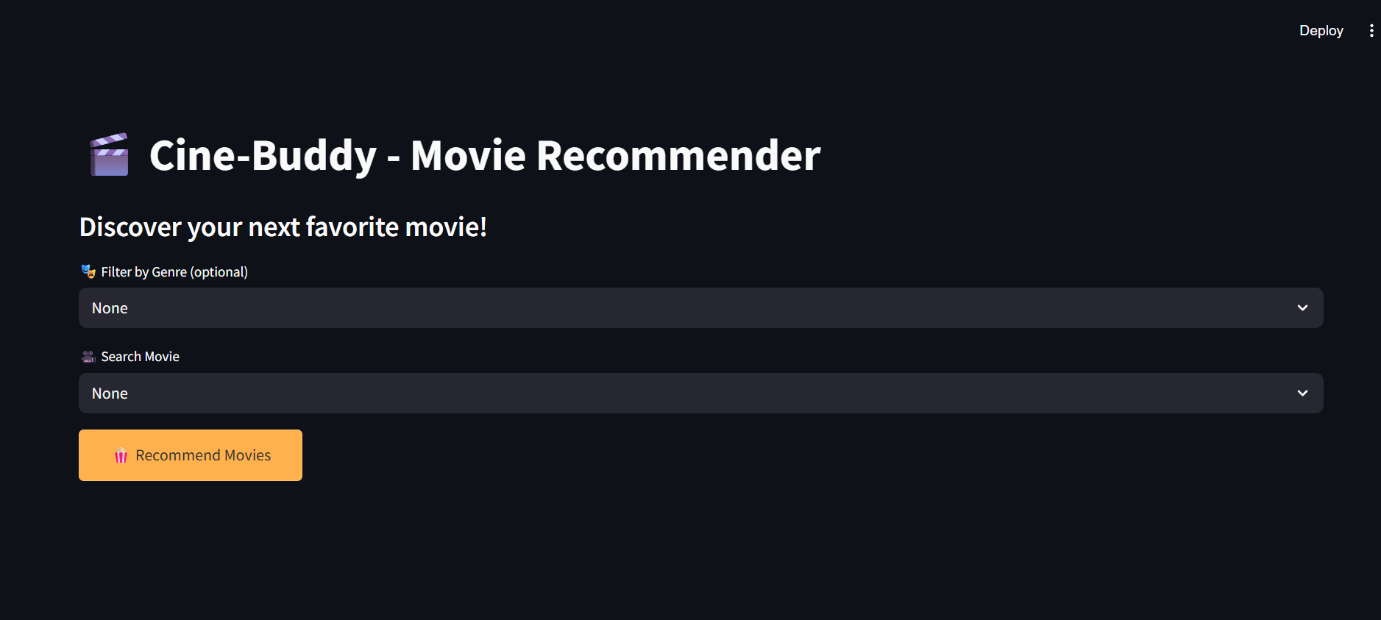
**Technologies Used**

CineBuddy integrates multiple technologies to deliver its full functionality. The primary programming language used is **Python**, owing to its robust data handling and machine learning libraries. For data manipulation, **Pandas** is used extensively. **Scikit-learn** is responsible for vectorization and similarity calculation. On the frontend, **Streamlit** handles the application layout and user interaction logic, offering widgets like dropdowns, toggles, and image rendering. Additionally, **Requests** and **BeautifulSoup** may be used to enhance features such as poster scraping or live data fetching in future extensions.

**Screens**

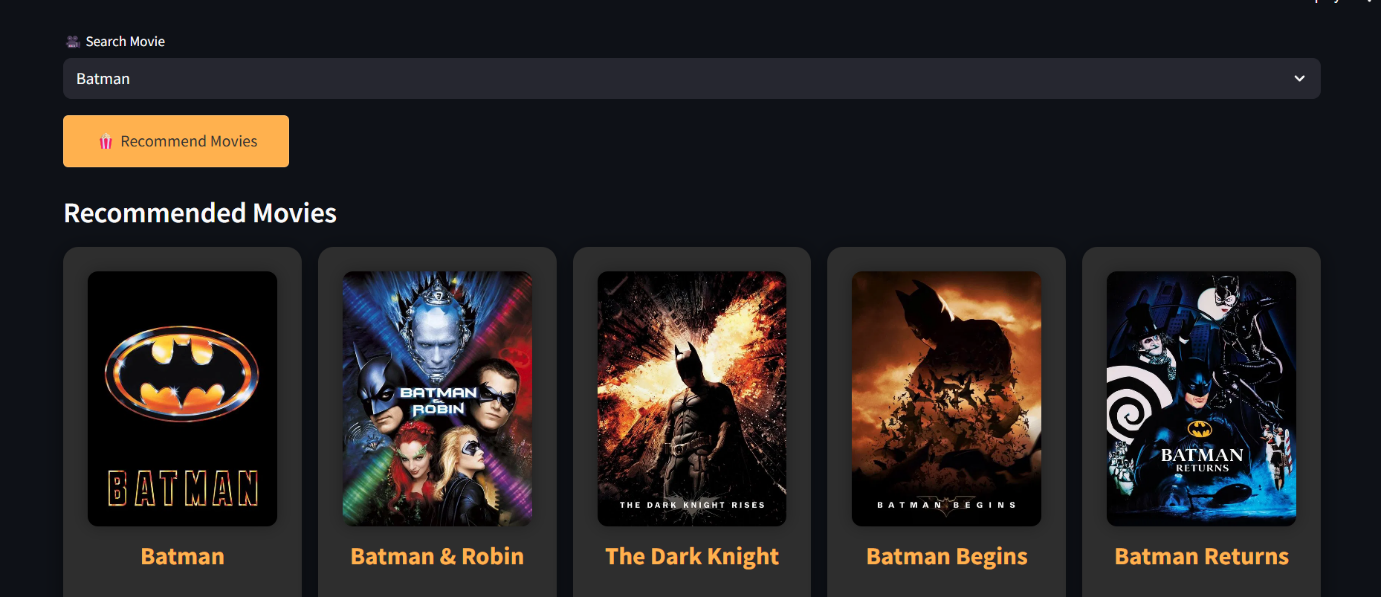
This is main and only page of recommendation system made by using Streamlit Framework,

Page contain two Input-fields and one button First input use for gerne classification (e.g. Action, Drama, Horror), Second Input Field include name of all movies

****

After click **on Recommend Movies Button** user get 5 most related movies of entered movie

(e.g. id we search Batman movie then the result shows as below)



**Future Scope**

While CineBuddy already delivers a compelling recommendation experience, several features can enhance its functionality. One major improvement would be integrating a **hybrid recommendation system** that combines collaborative filtering with content-based methods to offer more personalized results. Another enhancement would be user authentication, allowing individuals to create profiles, save favorites, and receive tailored suggestions based on their history. Further, the app could include **trailer previews** using YouTube or TMDb APIs, as well as **live data fetching** instead of relying solely on static datasets. Making the UI **responsive for mobile devices** and deploying it to platforms like **Streamlit Cloud or Heroku** would also significantly expand its accessibility and usage.

**Conclusion**

CineBuddy demonstrates how a relatively simple machine learning model can be transformed into a practical, real-world application through thoughtful integration with modern web technologies. The project not only highlights the effectiveness of content-based filtering in recommendation systems but also showcases how minimal yet meaningful UI choices can significantly elevate the user experience. By leveraging openly available datasets, a solid Python-based backend, and a responsive frontend built in Streamlit, CineBuddy delivers a polished movie discovery tool suitable for casual users and tech enthusiasts alike.

**Developer Information**

This project was developed as a part of a self-driven initiative to explore the intersection of machine learning and web development. It serves as a strong portfolio project demonstrating backend development, frontend design, and ML integration. The developer is passionate about creating intelligent applications that solve real-world problems through accessible design and clean code architecture.