A
Mini Project Report
On
"Movie Recommendation Model"

NGSPM'S

BRAHMA VALLEY COLLEGE OF ENGINEERING AND RESEARCH INSTITUTE NASHIK

NAAC ACCREDITED INSTITUTE WITH "A" GRADE

Department of Computer Engineering

Submitted By

Mr. Akhilesh Jadhav

Under the guidance of

Prof. M. S. Salve

Submitted in partial fulfillment of Bachelor of Engineering – Computer Engineering, bySavitribai Phule Pune University

Savitribai Phule Pune University

A.Y 2024-25

NGSPM'S
BRAHMA VALLEY COLLEGE OF ENGINEERING AND RESEARCH INSTITUTE NASHIK, Nashik- 422013

NAAC ACCREDITED INSTITUTE WITH "A" GRADE

C E R T I F I C A T E

This is to certify that ***Mr. Akhilesh Keru Jadhav*** have successfully completed the Mini Project of Web Technology entitled **"Movie Recommendation Model"** in the partial fulfillment of Bachelor of Engineering – Computer Engineering of Savitribai Phule Pune University, Pune.

Date:

Place: Nashik

Prof. M. S. Salve **Guide**	**Prof. P.P. Kakade** **Head Of Department**	**Dr. H. N. Kudal** **Principle**

ACKNOWLEDGEMENTS

I wish to express my sincere Gratitude to my Guide **Prof. M. S. Salve** of Computer Engineering Department. **Brahma Valley College Of Engineering, Nashik** for providing invaluable guidance and encouraging me to come up with such potential subject. I express my deep thanks to all the faculty members of Computer Engineering Department for providing the useful guidance and feel privileged to express my gratitude towards them.

I am also thankful to our respected **Prof. P. P. Kakade, Head of Department (Computer)** who constantly guided me throughout my thesis work.
I would like to thank our Respected **Prof. H. N. Kudal, Brahma Valley College Of Engineering, Nashik** for their encouragement at every step and those who helped me directly or indirectly this project.

Mr. Akhilesh Jadhav

INDEX

PROBLEM STATEMENT:

Develop a movie recommendation model using the sci-kit-learn library in Python.

OBJECTIVE:

The main objective of this recommendation system is to provide users with top-notch movie suggestions that match their preferences and to ensure the system is user-friendly and easy to operate. To achieve this goal, the system employs a blend of metadata analysis and historical user ratings to create personalized movie recommendations.

The system is specifically engineered to reduce the amount of information users need to provide by employing data analytics to analyze their past ratings and activities. This approach enables the system to deliver precise movie recommendations without demanding extensive user input.

TECHNOLOGY USED

Machine Learning Libraries:
- Pandas
- Numpy
- Difflib
- AST
- Scikit-learn

Frontend:
- HTML
- CSS
- JAVASCRIPT
- FLASK

Requirements:
- Python 3.6.3
- **IDE:** VS Code
- Anaconda

Introduction:

- ## What is Movie Recomendation System?

  A movie recommendation system is software that suggests movies to users based on their interests and preferences. These systems are built to analyze various user data, including past movie ratings and behavior, to offer personalized movie recommendations that the user is likely to enjoy.

  The movie recommendation system employs statistical techniques and machine learning algorithms to analyze user data and offer movie recommendations. These algorithms are often trained using large datasets of movie metadata, user ratings, and other relevant information, which enables them to identify patterns and similarities between users and movies.

  Some movie recommendation systems rely on collaborative filtering, which involves analyzing the behavior of other users who share similar movie preferences with the user. Other systems use content-based filtering, which examines movie features such as genre, actors, and director to provide recommendations based on similarities between movies.

- ## What is scikit-learn?

  Scikit-learn, also known as sklearn, is a widely-used open-source machine learning library designed for Python. It provides an array of tools for data mining, data analysis, and machine learning tasks, including regression, classification, clustering, and dimensionality reduction.

  Scikit-learn is built on top of other popular scientific computing libraries such as NumPy, SciPy, and Matplotlib, and offers a consistent API for using a range of machine learning models and algorithms. Additionally, it includes many useful features such as cross- validation, model selection, and pre-processing. Some of the most commonly used machine learning algorithms that can be implemented using scikit-learn include linear regression, logistic regression, support vector machines (SVM), k-nearest neighbors (KNN), decision trees, random forests, and neural networks.

  To use scikit-learn, you must first install the SciPy (Scientific Python) stack, which includes NumPy for base n-dimensional array support, SciPy for fundamental scientific computing library support, Matplotlib for comprehensive 2D/3D plotting, Python for an enhanced interactive console, Sympy for symbolic mathematics, and Pandas for data structures and analysis.

What is a Recommendation System?

A recommendation system refers to software or an algorithm that offers customized suggestions to users based on their interests, preferences, and behavior. These systems are used in various domains, including e-commerce, social media, and entertainment, to assist users in discovering new products, services, or content that they may be interested in.

To provide personalized recommendations, recommendation systems usually rely on a combination of machine learning algorithms, statistical techniques, and data mining. They analyze user data, including browsing history, purchase history, ratings, user behavior, and other relevant information, to identify patterns and similarities between users and products or content. The algorithms then use this data to make recommendations that are tailored to each user's individual preferences and interests.

Recommendation System Mechanism:

The mechanism underlying a recommendation system generally involves a series of steps:

- ❖ Data Collection: The first step is to collect user data, including browsing history, purchase history, ratings, user behavior, and other pertinent information.

- ❖ Data Preprocessing: Once the data is gathered, it must be preprocessed to remove duplicates, manage missing values, and transform it into a format suitable for analysis.

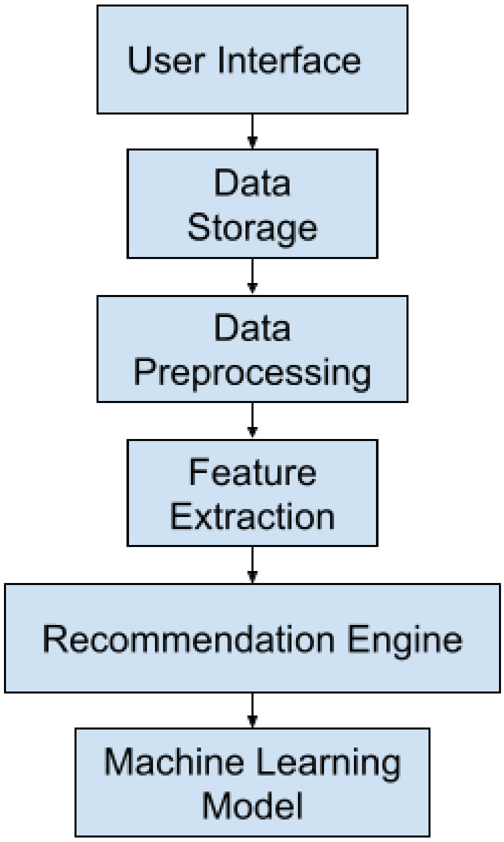
- ❖ Feature Extraction: In this step, the system extracts relevant features from the data, such as genre, actors, or keywords, which can be utilized to identify patterns and similarities between items.

- ❖ Similarity Calculation: The system computes the similarity between items based on their features, using various techniques such as cosine similarity or Euclidean distance.

- ❖ Recommendation Generation: Based on the similarity scores, the system generates a list of recommended items that are comparable to the ones the user has interacted with previously.

- ❖ Evaluation: Finally, the system evaluates the recommendations by measuring their accuracy and relevance to the user, using various metrics such as precision, recall, and F1-score.

Architecture:



Types of recommendation systems:

There are several types of recommendation systems, including Collaborative filtering, Content-based recommendation systems, and Hybrid recommendation systems. These systems are widely used because they are cost-effective and efficient.

Collaborative Filtering:

This type of recommendation system relies on user data such as previous ratings, reviews, and transactions to recommend products or services. Collaborative Filtering can be implemented in two ways:

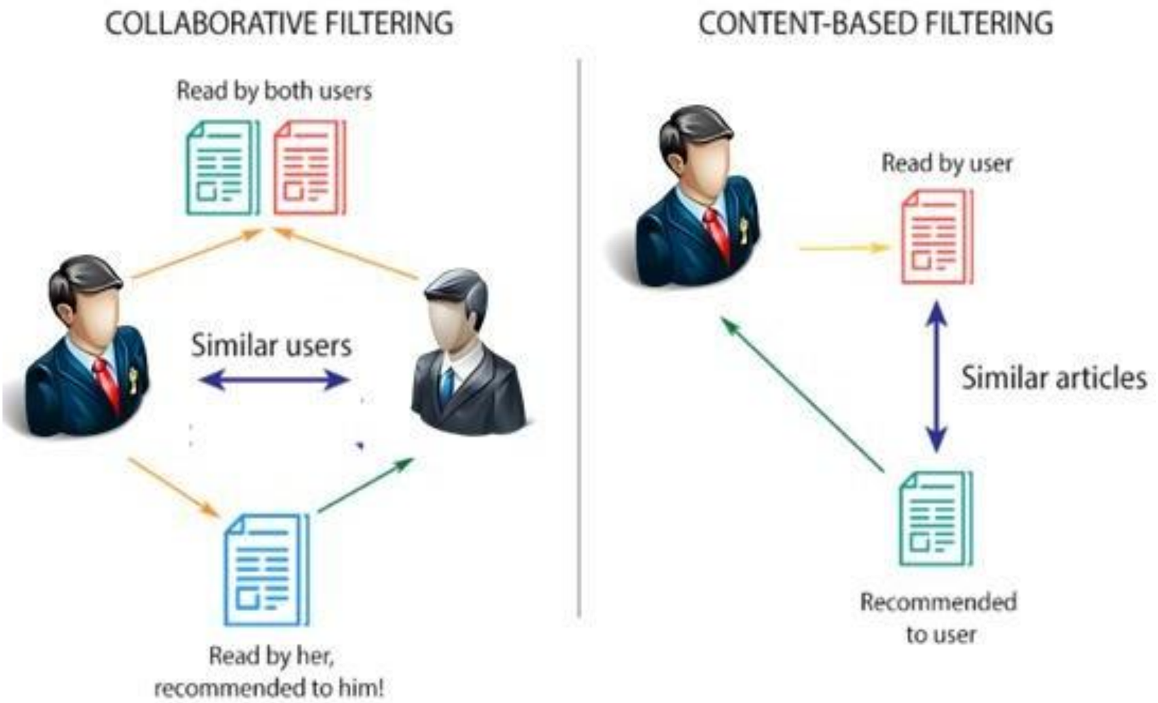
i. User-User Collaborative Filtering: In this approach, a particular user is compared with a pool of other users, and products or services that are liked by users with similar purchase patterns and product reviews are recommended to them.

ii. Item-Item Collaborative Filtering: This approach involves pairing items or services with similar ratings and review sentiments and recommending them to users who have shown interest in similar items or services.

Content-based recommendation systems :

Content-based recommendation systems don't rely on user data to provide recommendations, but instead use Artificial Intelligence to identify similarities between product or service descriptions to suggest compatible items. For example, if a user is watching Spiderman 2, which is an action movie, the system might recommend Spiderman 3, but not The Avengers, which is also an action movie but not as similar to Spiderman 2. These systems are often used to improve user experience and increase sales by suggesting complementary or upgraded products.

These types of recommendation systems are used for upselling and used to improve user experience and increasing sales of the companies.



The approach to building the movie recommendation engine consists of the following:

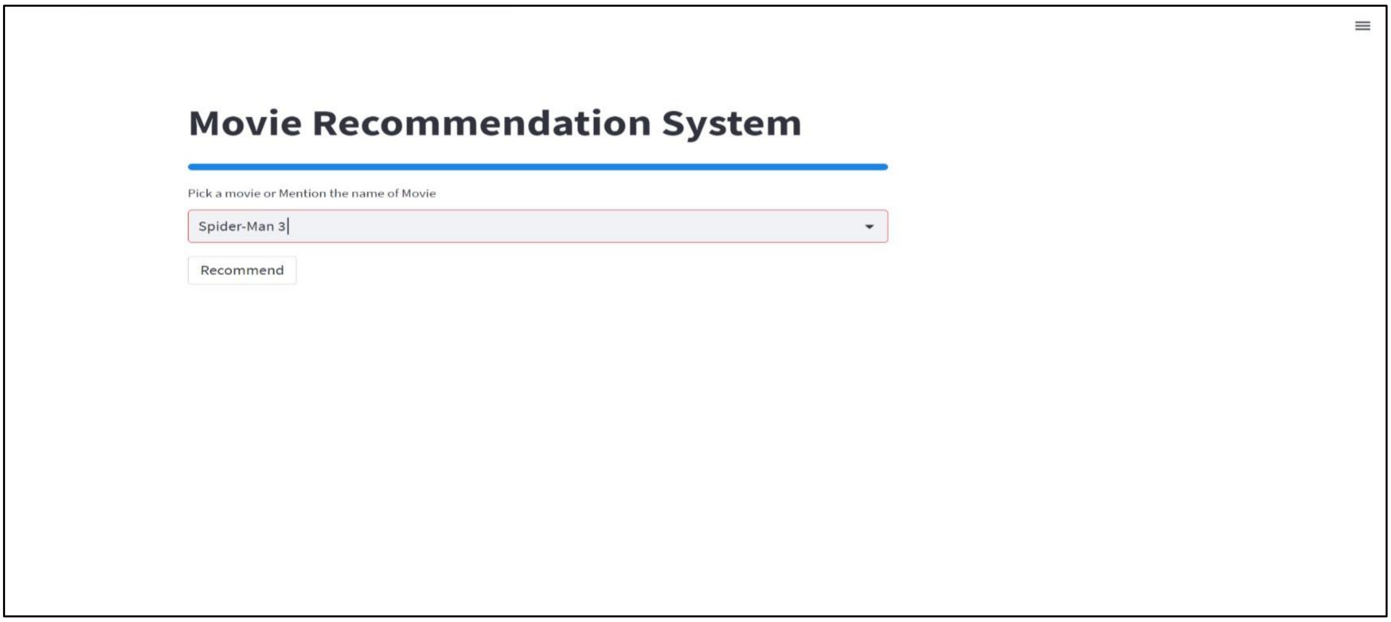
1. Perform Exploratory Data Analysis (EDA) on the data.
2. Build the recommendation system.
3. Get recommendations.

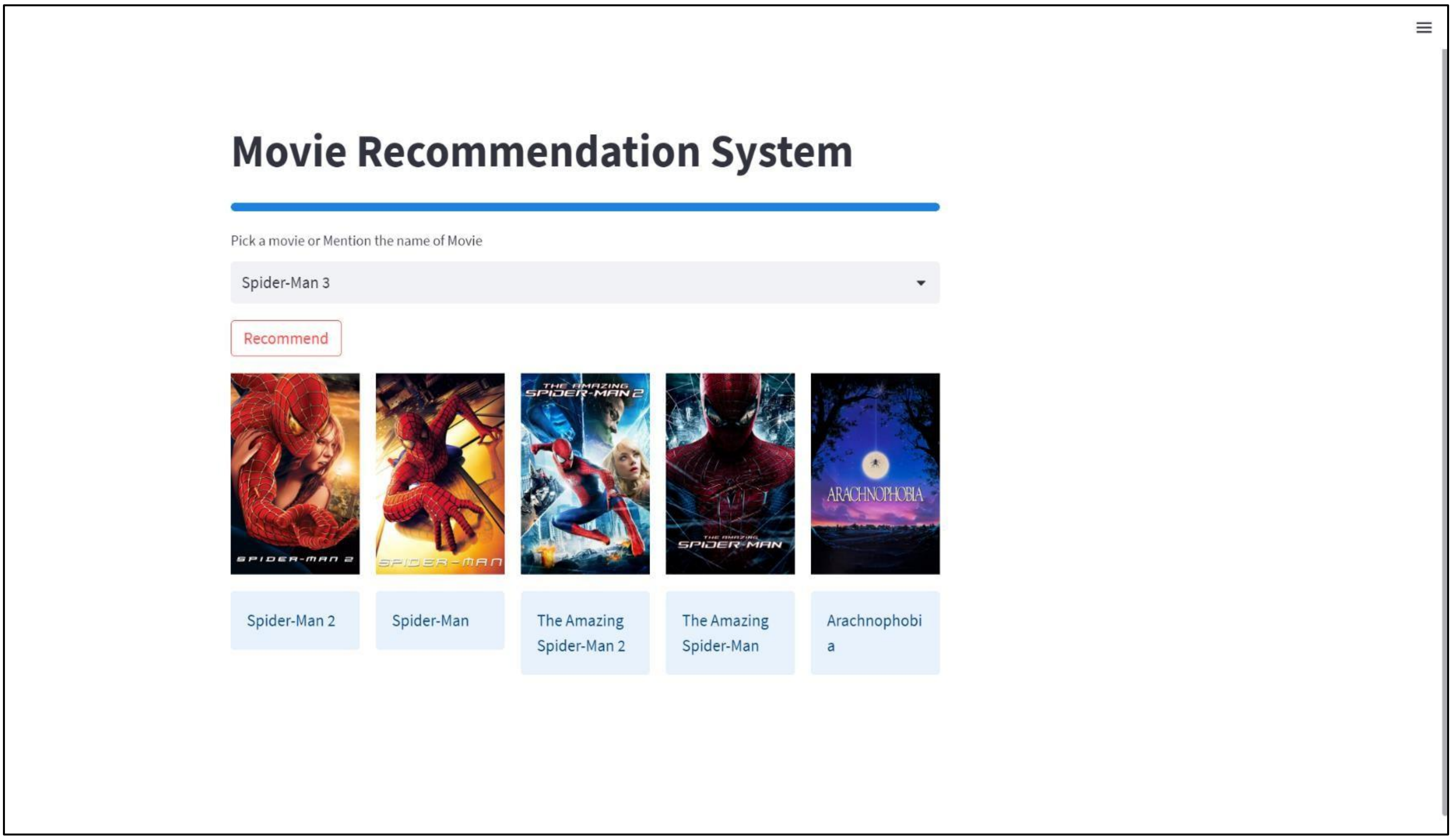
Code:

```
import   pandas   as
pd  import  streamlit
as st import pickle
import requests

def fetch_poster(movie_id):
   response =
                 requests.get('https://api.themoviedb.org/3/movie/{}?api_key=9647
                 47c0 9d17107c9c42c03777ecd5a7&language=en-
                 US'.format(movie_id))
   data = response.json()
   return "https://image.tmdb.org/t/p/w500/" + data['poster_path']

def recommend(movie):
   movie_index = movies[movies['title'] ==
   movie].index[0] distances =
   similarity[movie_index]
   movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x:
                   x[1])[1:6]

   recommended_movies = []
   recommended_movies_posters = []
   for i in movies_list:
      movie_id = movies.iloc[i[0]].movie_id
      recommended_movies.append(movies.iloc[i[0]].title)
      recommended_movies_posters.append(fetch_poster(movie_id))
   return recommended_movies, recommended_movies_posters

movies_dict = pickle.load(open('movies_dict.pkl',
'rb')) movies = pd.DataFrame(movies_dict)

similarity = pickle.load(open('similarity.pkl', 'rb'))
st.title("Movie Recommendation System")
st.progress(100)
selected_movie_name = st.selectbox('Pick a movie or Mention the name of Movie ',
movies['title'].values)

if st.button('Recommend'):
   names, posters =
   recommend(selected_movie_name) col1, col2,
   col3, col4, col5 = st.columns(5)
   with col1:
      st.image(posters[0])
      st.info(names[0])
   with col2:
```

```
        st.image(posters[1])
        st.info(names[1])
    with col3:
        st.image(posters[2])
        st.info(names[2])
    with col4:
        st.image(posters[3])
        st.info(names[3])
    with col5:
        st.image(posters[4])
        st.info(names[4])
```

OUTPUT:





Conclusion:

Recommendation systems have become an indispensable part of our daily lives, especially in the entertainment industry. With an ever-increasing number of movies and TV shows being released worldwide each year, it can be overwhelming for viewers to decide what to watch next. As a result, viewers often miss out on hidden gems and remarkable works of art due to a lack of accurate recommendations.

To tackle this challenge, machine learning-based recommendation systems are becoming increasingly essential to provide viewers with personalized and relevant recommendations. Content-based recommendation systems, which analyze the features and attributes of movies and TV shows, are effective in providing recommendations based on a viewer's preferences. However, these systems can have limitations when there is limited information available about a user's preferences.

Overall, recommendation systems are crucial in helping viewers navigate the vast array of available content and discover hidden gems. By utilizing various techniques such as collaborative filtering, content-based filtering, and hybrid approaches, recommendation systems can provide personalized and accurate recommendations to enhance the viewing experience.

Future Scope:

The future of movie recommendation systems looks promising as more and more people are using these systems to enhance their viewing experience. Here are some potential future scopes for movie recommendation systems:

1. Incorporating more advanced AI techniques: As AI and machine learning technologies continue to evolve, movie recommendation systems can become more sophisticated and accurate in their recommendations. Future systems could incorporate deep learning techniques to analyze even more data and provide even more personalized recommendations.

2. Utilizing social media data: Movie recommendation systems could integrate social media data to improve their recommendations. For example, by analyzing a user's social media activity, these systems could determine what types of movies or TV shows their friends are interested in and make recommendations based on that information.

3. Incorporating real-time data: Future movie recommendation systems could use real-time data to make recommendations based on the user's current mood or context. For example, a recommendation system could suggest a comedy movie on a rainy day or a thriller movie at night.

4. Personalized trailers: Movie recommendation systems could generate personalized trailers for users based on their viewing history and preferences. These trailers could include snippets of movies that the user might be interested in, making it easier for them to decide whether or not to watch the movie.

5. Expanding to other domains: While movie recommendation systems are currently used in the entertainment industry, there is potential to expand to other domains. For example, these systems could be used to recommend books, music, or even products based on the user's preferences and history.