MOVIE RECOMMENDATION SYSTEM

A Course Based Project Report in Machine Learning & Neural Networks

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN
CSE (Artificial Intelligence and Machine Learning)

Submitted by

B. Mokshagna	(21071A6606)
G. Swaroop	(21071A6622)
M. Sriram	(21071A6636)
M. Likitha	(21071A6639)

Under the guidance of

Dr. N. Sandhya
(Professor, Department of CSE-AIML & IoT, VNR VJIET)

Department of CSE

Artificial Intelligence and Machine Learning & Internet of Things

Vallurupalli Nageswara Rao Vignana Jyothi Institute Of

Engineering And Technology

(An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC) ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute, NAAC Accredited with ‘A++’ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as “College with Potential for Excellence” by UGC) ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE

Artificial Intelligence and Machine Learning & Internet of Things

TAMASŌMA JYOTIRGAMAYA

CERTIFICATE

This is to certify that the project report entitled “**Movie Recommendation System”** is a bonafide work done under our supervision and is being submitted by B. Mokshagna(21071A6606), G. Swaroop (21071A6622), M. Sriram (21071A6636), M.Likitha(21071A6639) in partial fulfillment for the award of the degree of Bachelor of Technology in CSE(Internet of Things), of the VNRVJIET, Hyderabad during the academic year 2022-2023. Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Naga Durga Saile. K	**Dr. N. Sandhya**
Assistant Professor	**Professor & HoD**
Dept. of CSE (AIML & IoT)	**Dept. of CSE (AIML & IoT)**
VNR VJIET	**VNR VJIET**

DECLARATION

We declare that the major project work entitled "**Movie Recommendation System**" submitted in the department of CSE(Internet of Things), Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degreeof **Bachelor of Technology** in **CSE(Internet of Things)** is a bonafide record of our own work carried out under the supervision of Dr.N.Sandhya **Professor, Department of CSE(AIML & IoT), VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

B. Mokshagna	(21071A6606)
G. Swaroop	(21071A6622)
M. Sriram	(21071A6636)
M. Likitha	(21071A6639)

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr. Challa Dhanunjaya Naidu,** and our Head of Department, **Dr. N. Sandhya**, Professor for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide **Dr. N .Sandhya**, professor for enthusiastic guidance throughout the course of our project.

We extend our heartfelt thanks to our incharge **Naga Durga Saile**. K, Assistant Professor for enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Computer Science & Engineering (AIML & IoT) department and to our fellow classmates who directly or indirectly helped us.

B. Mokshagna	(21071A6606)
G. Swaroop	(21071A6622)
M. Sriram	(21071A6636)
M. Likitha	(21071A6639)

ABSTRACT

The **"Movie Recommendation System Project"** aims to design, develop, and implement an intelligent recommendation system for movies, leveraging machine learning algorithms and collaborative filtering techniques. The project addresses the challenge of information overload in the entertainment industry, where users are presented with a vast array of movie options, making it challenging to discover content tailored to their preferences.

The recommendation system employs collaborative filtering to analyze user behavior, preferences, and historical data to generate personalized movie recommendations. It combines user-based and item-based collaborative filtering methodologies to enhance the accuracy and relevance of recommendations. Additionally, the system incorporates content-based filtering to consider the inherent characteristics of movies, such as genre, director, and actors, further refining.

The **"Movie Recommendation System Project"** is not only a valuable tool for movie enthusiasts seeking personalized content but also serves as a platform for exploring and implementing state-of-the-art machine learning techniques in the context of collaborative filtering and recommendation systems.

CONTENTS

1. INTRODUCTION

1.1 Movie Recommendation System in OTT Platforms

In the era of digital streaming, where over-the-top (OTT) platforms have revolutionized the way we consume entertainment, the abundance of digital content has brought forth both opportunities and challenges. With an unprecedented volume of movies available at our fingertips, navigating the vast landscape of cinematic offerings across various OTT platforms has become a formidable task. Catering to diverse genres, directors, and cultural influences, the process of discovering content that aligns with individual preferences can be overwhelming.

To address this challenge and enhance the user experience in the realm of entertainment, movie recommendation systems have emerged as a transformative solution. These systems leverage the power of data analysis, machine learning, and collaborative filtering to provide users with personalized movie suggestions tailored to their tastes and viewing history. By intelligently analyzing user behavior, preferences, and movie characteristics, these recommendation systems streamline the overwhelming array of choices, presenting viewers with content that resonates with their unique interests.

This project explores the development and implementation of a sophisticated Movie Recommendation System for OTT platforms, going beyond conventional approaches. By employing advanced machine learning algorithms, collaborative filtering techniques, and considering both user behavior and movie metadata, the system aims to deliver accurate and relevant recommendations. This not only enhances user engagement but also fosters a deeper connection between viewers and the vast cinematic offerings available on OTT platforms.

As we delve into the intricacies of this Movie Recommendation System for OTT platforms, we will explore the methodologies employed, the underlying algorithms utilized, and the impact of such systems on user satisfaction and content consumption patterns. Through this project, we endeavor to create an effective recommendation system that contributes to the broader discourse on the intersection of technology and entertainment. By bridging the gap between viewers and the ever-expanding world of cinematic storytelling, we aim to elevate the streaming experience, allowing audiences to effortlessly discover their next favorite film amidst the abundance of choices available on OTT platforms.

1.2 OBJECTIVE

The primary objective of this project is to develop an intelligent and user-friendly movie recommendation system that suggests relevant and personalized movie recommendations based on a given movie input. The system aims to enhance the movie discovery experience for users by leveraging advanced data analysis and machine learning techniques to provide tailored suggestions that align with their preferences and interests.

To achieve this objective, the project will incorporate the following key components:

1. Movie Recommendation Engine: At the core of the system lies a robust recommendation engine that employs sophisticated algorithms and machine learning models. This engine will analyze the characteristics of the input movie, such as genre, director, cast, plot summary, and user ratings, to identify similar movies that match the user's potential interests. By leveraging techniques like collaborative filtering, content-based filtering, or hybrid approaches, the engine will generate a list of relevant and engaging movie recommendations.

2. Visual User Interface: To facilitate seamless interaction with the recommendation system, a visually appealing and user-friendly interface will be developed. Streamlit, a powerful Python library for building data applications, will be utilized to create an intuitive web interface. This interface will allow users to input their preferred movie, and the system will dynamically display a curated list of recommended movies based on the recommendation engine's output. The interface will incorporate features such as movie posters, synopses, ratings, and additional information to assist users in making informed decisions about their next movie selection.

By combining the power of machine learning and data analysis with an engaging user interface, this movie recommendation system aims to revolutionize the way users discover and explore new cinematic experiences. The system will not only suggest movies based on the user's input but also provide a platform for exploring a vast collection of movies, enabling users to discover hidden gems and expand their cinematic horizons.

Furthermore, the project will involve the integration of the recommendation engine with a comprehensive movie database, ensuring access to a vast collection of movie metadata and user ratings. This integration will enable the system to continually learn and adapt, providing increasingly accurate and relevant recommendations as more users interact with the platform.

Overall, this movie recommendation system project seeks to streamline the movie discovery process, enhance user satisfaction, and foster a deeper appreciation for the rich tapestry of cinematic arts.

2. LITERATURE SURVEY

[1] Yehuda Koren and Robert Bell. "Advances in Collaborative Filtering." In Recommender Systems Handbook, pp. 145-186. Springer, Boston, MA, 2015. In this work, the authors provide a comprehensive overview of collaborative filtering techniques, which are widely used in recommendation systems, including movie recommendations. They discuss various algorithms, such as matrix factorization, neighborhood-based methods, and hybrid approaches, and their respective strengths and weaknesses. The authors also analyze the challenges of sparse data and scalability in collaborative filtering.

[2] Xiaoyuan Su and Taghi M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques." Advances in Artificial Intelligence 2009 (2009): 1-19. This survey paper presents a detailed analysis of collaborative filtering algorithms used in recommendation systems. It covers user-based, item-based, and model-based collaborative filtering techniques, as well as hybrid approaches that combine collaborative filtering with content-based filtering. The authors highlight the advantages and limitations of each technique and discuss their potential applications in various domains, including movie recommendations.

[3] Muhammet Baydogan and Gökhan Erçil. "A Content-Based Movie Recommendation System Using Machine Learning Techniques." In Proceedings of the 2021 International Conference on Artificial Intelligence and Machine Learning (ICAIML), pp. 113-118. IEEE, 2021. In this study, the researchers developed a content-based movie recommendation system that utilizes machine learning techniques. They employed natural language processing and topic modeling to extract relevant features from movie synopses and combined them with other metadata, such as genre and director. The system was trained on a large dataset and evaluated using various metrics, demonstrating promising results in providing accurate and relevant movie recommendations.

[4] Yongrui Qin et al. "A Hybrid Approach for Personalized Movie Recommendation." In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), pp. 2619-2628. IEEE, 2019. This paper presents a hybrid movie recommendation system that combines collaborative filtering and content-based filtering techniques. The authors utilized matrix factorization for collaborative filtering and integrated it with content-based features extracted from movie metadata. They evaluated their approach on a large-scale dataset and compared it with several baseline methods, showing improved performance in terms of recommendation accuracy and diversity.

[5] Shuai Zhang et al. "Deep Learning for Multimedia Recommendation: A Survey." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 18, no. 1s (2022): 1-33. This survey paper provides a comprehensive overview of deep learning techniques used in multimedia recommendation systems, including movie recommendations. The authors discuss various deep learning architectures, such as convolutional neural networks, recurrent neural networks, and attention mechanisms, and their applications in extracting relevant features from multimedia data. They also highlight the challenges and future research directions in this field.

2.1 SYSTEM STUDY

There are many algorithms that can be used to detect whether a transaction is Movie Recommendation System. We have considered two algorithms – Porter Stemmer & Vectorization and Cosine Similarity.

2.1.1 Porter Stemmer & Vectorization:

The Porter Stemmer is an algorithm used in natural language processing (NLP) to reduce words to their base or root form, also known as the stem. This process is called stemming and is particularly useful in information retrieval systems, such as search engines and recommendation systems, where it helps in matching related words and improving search results.

The Porter Stemmer algorithm follows a series of rules and steps to remove common prefixes and suffixes from words. For example, the words "playing," "played," and "plays" would all be stemmed to the root word "play."

Vectorization, on the other hand, is a process of converting text data into numerical vectors, which can be used as input for machine learning algorithms. One common technique for vectorization is the Bag-of-Words (BoW) model, where each unique word in the corpus is assigned a unique index, and each document is represented as a vector of word counts or frequencies.

Another popular vectorization technique is TF-IDF (Term Frequency-Inverse Document Frequency), which weighs the importance of a word not only by its frequency in a document but also by its rarity across the entire corpus. This helps in identifying words that are more relevant and discriminative for a particular document.

2.1.2 Cosine Similarity

Cosine similarity is a widely used metric in recommendation systems and other applications involving textual data. It measures the similarity between two non-zero vectors by calculating the cosine of the angle between them. The cosine similarity between two vectors A and B is calculated as:

cosine_similarity = (A · B) / (||A|| * ||B||)

where:

- A · B is the dot product of the vectors A and B
- ||A|| and ||B|| are the Euclidean norms (lengths) of vectors A and B, respectively

The cosine similarity ranges from 0 (vectors are orthogonal or dissimilar) to 1 (vectors are identical). In the context of text data, the cosine similarity between two document vectors can be used to measure their similarity based on the overlap of words or terms.

For example, in a movie recommendation system, the cosine similarity between the vector representations of a user's movie preferences and the vector representations of movie descriptions can be calculated to identify movies that are most similar to the user's interests.

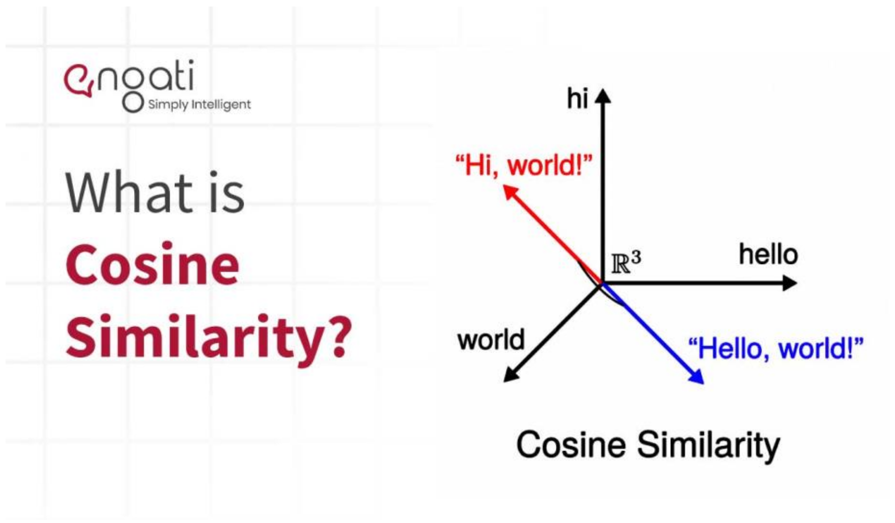


Fig 1: Cosine Similarity

2. SYSTEM REQUIREMENTS

2.1 REQUIREMENT SPECIFICATION

2.1.1 Functional Requirements

Every software application must fulfill a large number of functions in order for it to operate correctly. These functions are nothing more than different tasks that are carried out during each stage of the application development process. This step is included in the best practices for creating applications. Together, functional and non-functional requirements establish a set of rules for the efficient operation of an application. They also assist the user and developer in identifying the software and hardware requirements necessary to operate the application. Functional Requirements that are required to run this application are:

Python:

Guido Van Rossum created the Python programming language in 1991. The language's syntaxes make it incredibly comfortable and simple for programmers to use. This programming language is suitable for usage on both small and big scales precisely because of this reason. They are dynamic and garbage collected.

2.1.2 Non Functional Requirements

Non-functional criteria are used to set conditions for tracking an application's performance characteristic. It explains the working of a specific application function. They also determine the overall quality of the project and hence it is a very important aspect in any software development process.

The Non-Functional Requirements include:

Usability: It relates to the application's user-friendliness and establishes how simple it is for the user to utilize. When using an application requires little or no specialized expertise and is highly functional, usability is considered to be high. It is also a key factor that can determine if a user is satisfied.

Accuracy: The degree to which the value generated by the system is near to the ideal value is determined by accuracy. The accuracy increases as the difference between the system value and the ideal value decreases. It is also a means of figuring out how the application performs better than other applications of a similar nature.

Responsiveness: Responsiveness is determined by completing the software operations with minimal errors or no errors. It is directly correlated with the application's performance and stability. This criterion can also be used to determine robustness and recoverability.

Scalability: Scalability is used to determine the growth of the project. It establishes how much room the application can have for future feature expansion. It determines the project's viability. This is one of the criteria that is used to create long-term models for corporate success.

2.1.3 Software Requirements

Software:

OS: Windows 11

Programming Language: Python

Libraries: Matplotlib, sklearn, numpy, pandas, nltk

Algorithms: Cosine Similarity

Editor: Pycharm, Jupyter Notebook

sklearn: Simple and efficient tools for predictive data analysis. Accessible to everybody, and reusable in various contexts. Built on NumPy, SciPy, and matplotlib. Open source, commercially usable - BSD license

Numpy: NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices.

Pandas: pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Nltk: The Natural Language Toolkit (NLTK) is a popular open-source Python library used for natural language processing tasks. It provides easy-to-use interfaces for working with corpora, tokenization, stemming, tagging, parsing, and semantic reasoning, making it a valuable toolkit for researchers and developers in the field of computational linguistics and natural language applications.

2.1.4 Hardware Requirements

Hardware:

Processor: Intel I3 processor

Storage Space: 500 GB

Devices Required: Mouse and a Keyboard

Minimum Ram: 4GB and a good Internet connection

2.2 METHODOLOGY

Here is the methodology in one paragraph, up to the web application development using Streamlit:

The methodology involves data collection and preprocessing of a comprehensive movie dataset containing metadata such as title, genre, director, cast, plot summary, and user ratings. After handling missing values and inconsistencies, the dataset is split into training and testing sets. Relevant features are engineered, including text preprocessing techniques like tokenization, stemming, and stop-word removal on textual features, and vectorization using methods like Bag-of-Words or TF-IDF. Categorical features are encoded appropriately. Suitable machine learning algorithms, such as collaborative filtering, content-based filtering, or hybrid approaches, are selected for the movie recommendation task. The chosen models are trained using the preprocessed data, with hyperparameter tuning and cross-validation for optimizing performance. Model evaluation is conducted using metrics like precision, recall, F1-score, and mean average precision (MAP), leading to the selection of the best-performing model. For the web application development, the Streamlit library is utilized to create a user-friendly interface where users can input a movie title, and the integrated recommendation model generates personalized movie recommendations.

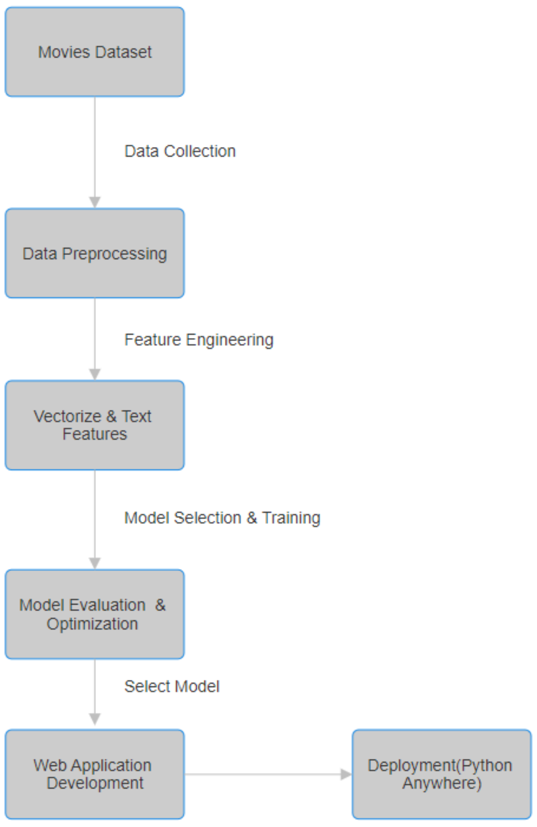

Movies Dataset
Data Collection
Data Preprocessing
Feature Engineering
Vectorize & Text Features
Model Selection & Training
Model Evaluation & Optimization
Select Model
Web Application Development
Deployment(Python Anywhere)

Fig 2: Work flowchart

3. DATASET

3.1 Source

Dataset source: Git Hub.

3.2 Description

The movies & credits dataset is freely available in Kaggle. It contains information about title, genres, cast, crew of the movie.

4. UML DIAGRAMS

UML diagrams are used to understand the system in a better and simple way. The elements are like components which can be associated in different ways to make a complete UML picture, which is known as a diagram. There are different types of UML Diagrams.

4.1 Use case Diagram:

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, a use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. They are usually used to illustrate the various actions taken by the application. They also show the several users who can carry out these functions. Use-case diagrams fall under behaviour diagrams due to their emphasis on the tasks carried out and the users (actors) who carry out these task

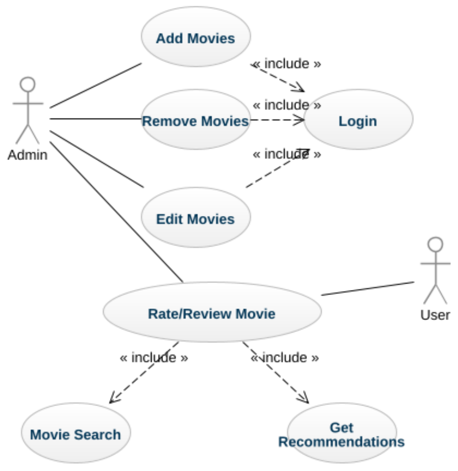
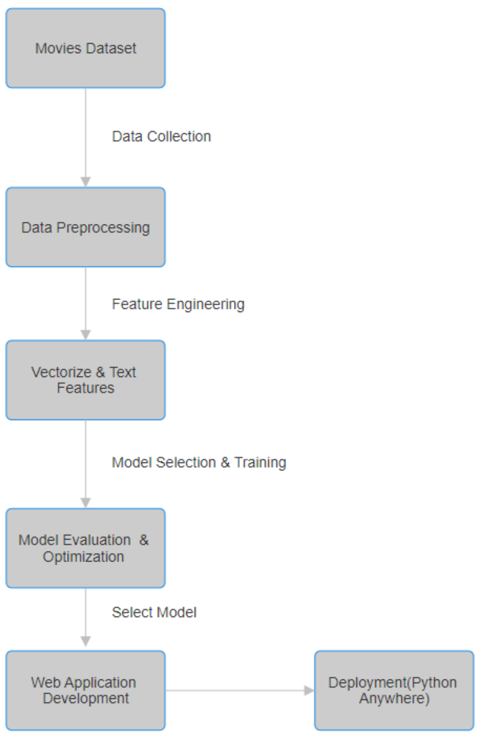


Fig 3: Usecase Diagram

4.2 Class Diagram:

A class diagram is a static diagram. It represents the application's static view. Class diagrams are used to create executable code for software applications as well as for visualizing, explaining, and documenting various elements of systems. The attributes and functions of a class are described in a class diagram, along with the constraints placed on the system. Because they are the only UML diagrams that can be directly mapped with object-oriented languages, class diagrams are frequently employed in the modelling of object-oriented systems.


Movies Dataset
Data Collection
Data Preprocessing
Feature Engineering
Vectorize & Text Features
Model Selection & Training
Model Evaluation & Optimization
Select Model
Web Application Development
Deployment(Python Anywhere)

` **Fig 5: Class Diagram**

4.3 Activity Diagram:

An application's control flow or sequence is shown in an activity diagram. In fact, we may utilise the activity diagram to validate each action depicted in the use-case diagram. It also shows the process steps. They basically show how the software application's workflows are done. It also places emphasis on the order in which tasks should be completed and the conditions that must be met for each activity to be successful. In this manner, it informs us about the factors that influence a specific task. As a result, we can now see the application at a high level. The primary goals are:

- Depict the activity flow
- Describe the sequence (branched or sequential)

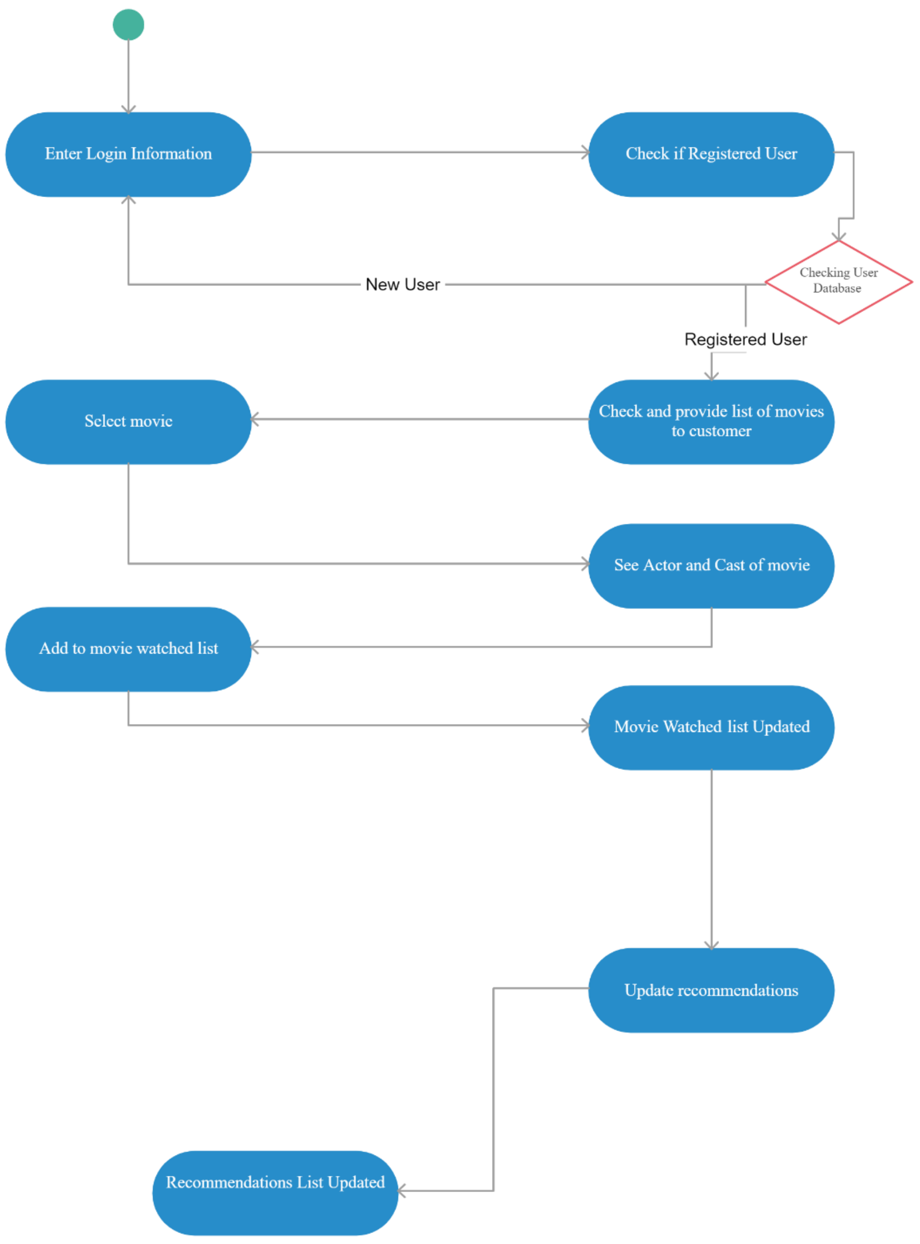

Enter Login Information
Check if Registered User
Checking User Database
New User
Registered User
Select movie
Check and provide list of movies to customer
See Actor and Cast of movie
Add to movie watched list
Movie Watched list Updated
Update recommendations
Recommendations List Updated

Fig 6: Activity Diagram

5. IMPLEMENTATION

5.1 MODULES

The proposed system has 3 modules:

1. Data Pre-Processing
2. Training Model
3. Testing Module

5.1.1 Data Pre-Processing

Preprocessing is a crucial step in building a movie recommendation system. It involves cleaning and transforming the raw dataset to make it suitable for training machine learning models. Here are some common preprocessing activities for a movie recommendation system dataset:

1. Handling Missing Data:

- ➢ Identify and handle missing values in the dataset, whether they are in user demographics, movie details, or user ratings.
- ➢ Techniques include imputation (filling missing values with a reasonable estimate) or removing rows or columns with missing values.

2. Removing Duplicates:

- ➢ Check for and remove duplicate entries to ensure that each user-movie interaction is unique.

3. Encoding Categorical Variables:

- ➢ Convert categorical variables such as genres, directors, and actors into numerical representations. This can be done using techniques like one-hot encoding or label encoding.

4. Timestamp Processing:

- ➢ If the dataset includes timestamps, convert them to a standardized format and consider extracting additional features, such as day of the week or time of day.

5. Dealing with Outliers:

- ➢ Analyze and handle outliers in the ratings or other numerical features to prevent them from unduly influencing the recommendation model.

6. Handling User and Item Cold Start:

- ➢ Address the "cold start" problem where there is insufficient data for new users or items. This may involve using demographic information or default recommendations until sufficient data is collected.

7. Normalization/Scaling:

- ➢ Normalize or scale numerical features, especially if different features have significantly different scales. This ensures that the recommendation model is not biased toward features with larger magnitudes.

8. Feature Engineering:

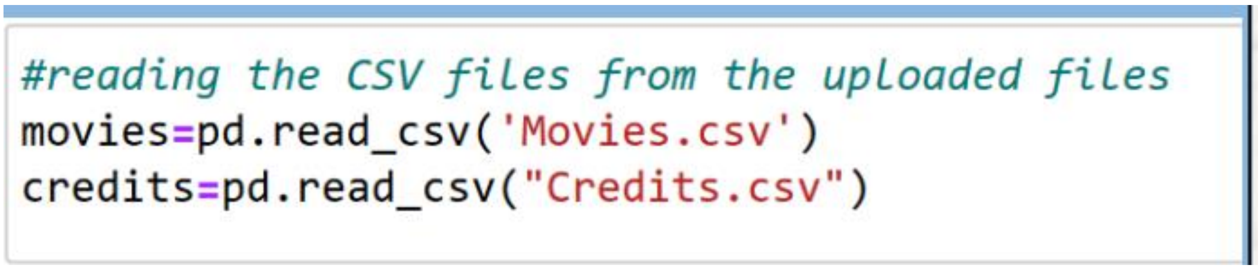
- ➢ Create new features that could enhance the recommendation system. For example, you might derive features from user demographics or movie details to capture additional information.

9. Splitting the Dataset:

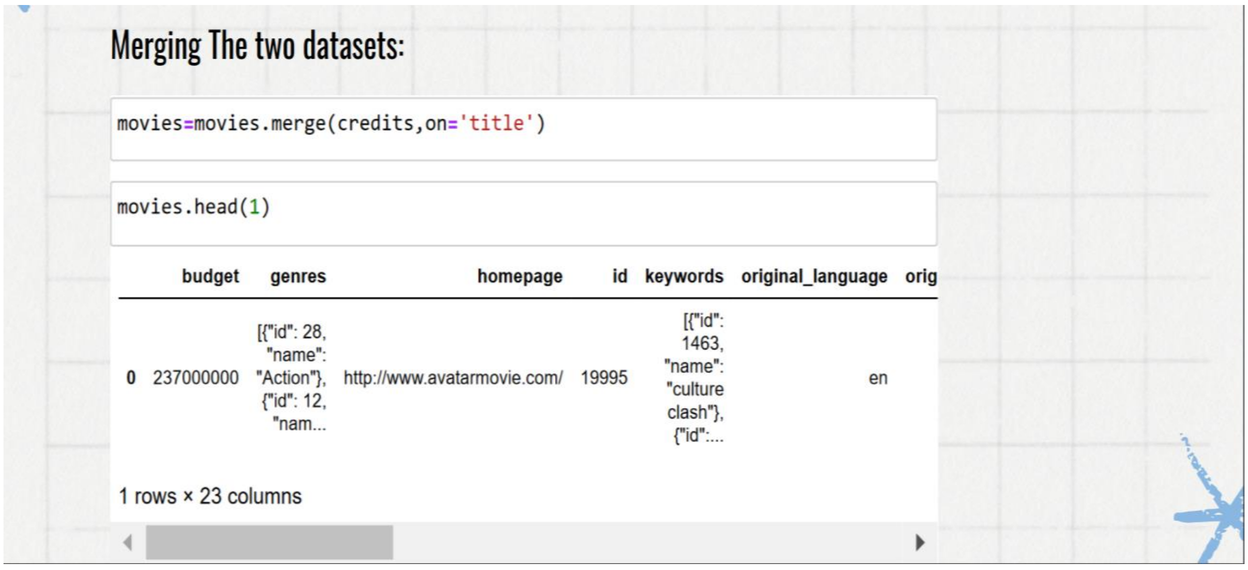
- ➢ Divide the dataset into training and testing sets. The training set is used to train the recommendation model, while the testing set is used to evaluate its performance.

10. Handling Long-Tail Items:

- ➢ Address the issue of long-tail items by considering strategies to improve recommendations for less popular movies. This may involve adjusting the recommendation algorithm or introducing personalized recommendations based on user preferences



```python
#reading the CSV files from the uploaded files
movies=pd.read_csv('Movies.csv')
credits=pd.read_csv("Credits.csv")
```



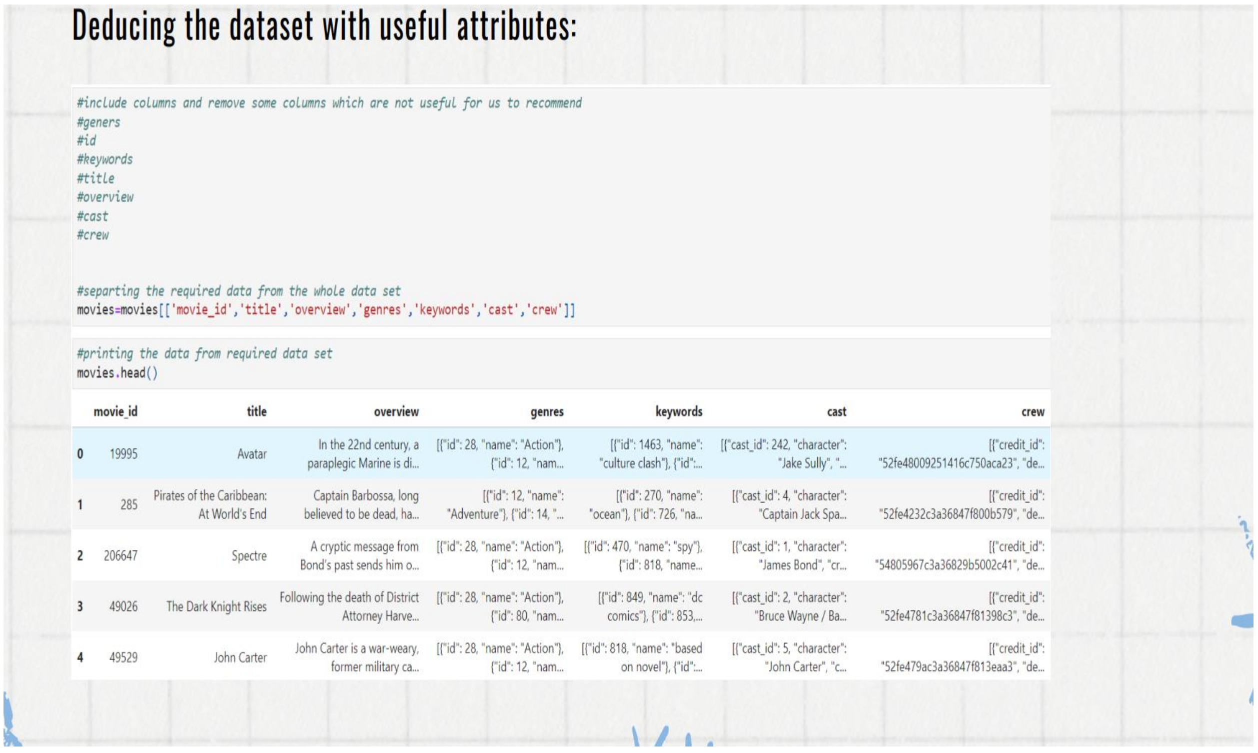
Merging The two datasets:

```python
movies=movies.merge(credits,on='title')
```

```python
movies.head(1)
```

	budget	genres	homepage	id	keywords	original_language	orig
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	

1 rows × 23 columns



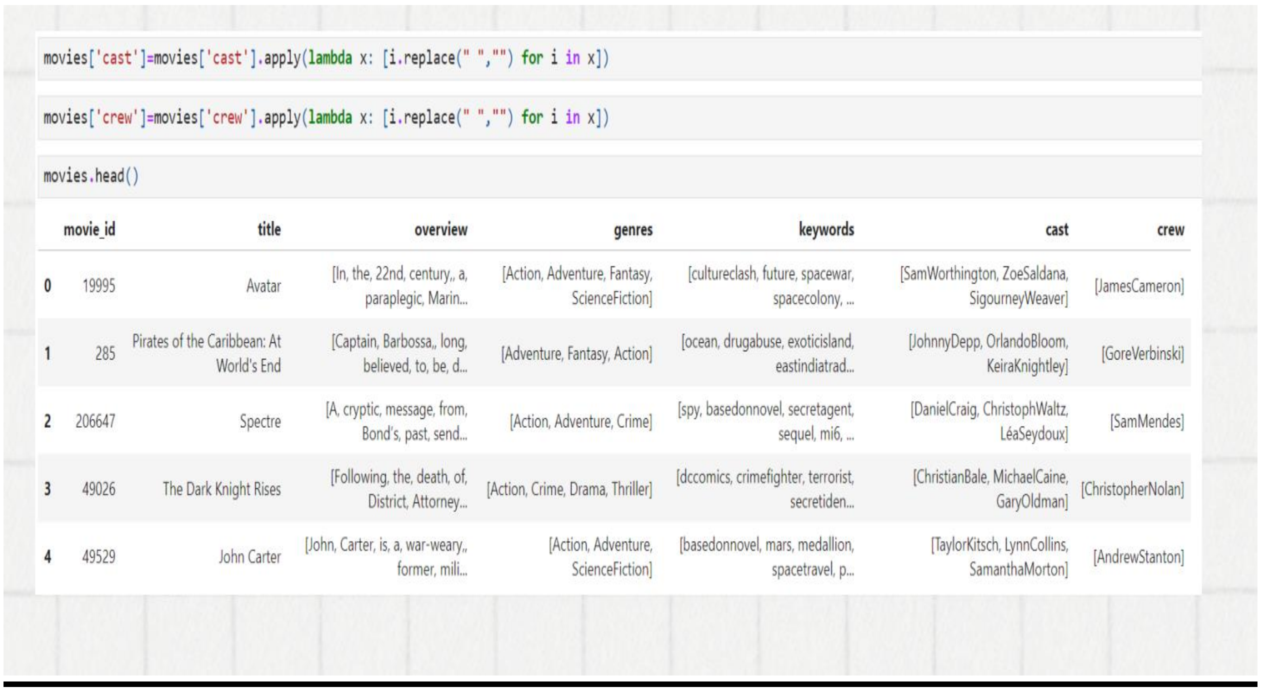
Deducing the dataset with useful attributes:

```python
#include columns and remove some columns which are not useful for us to recommend
#geners
#id
#keywords
#title
#overview
#cast
#crew


#separting the required data from the whole data set
movies=movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```

```python
#printing the data from required data set
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 470, "name": "spy"}, {"id": 818, "name...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	[{"id": 849, "name": "dc comics"}, {"id": 853,...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	[{"id": 818, "name": "based on novel"}, {"id":...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

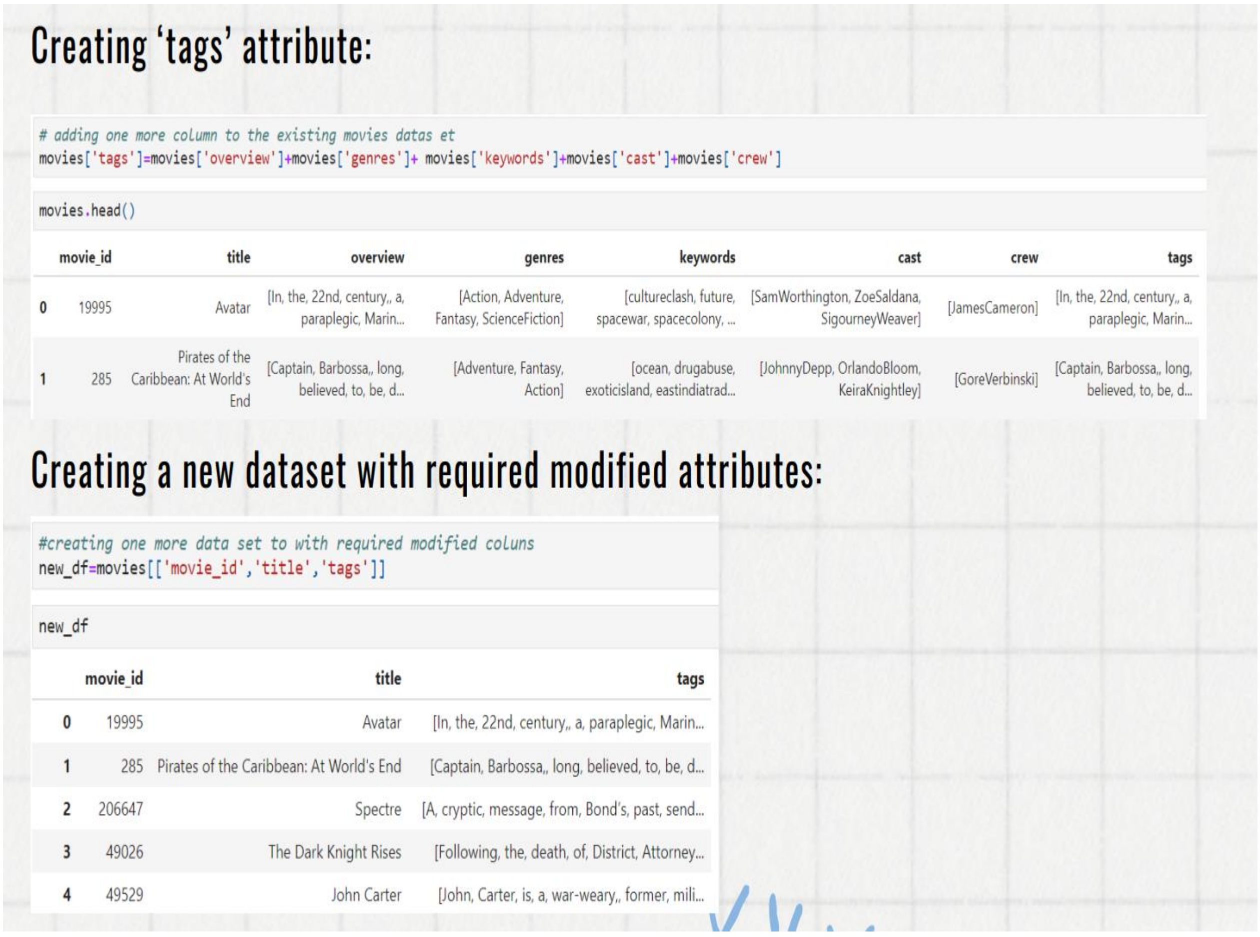


```
movies['cast']=movies['cast'].apply(lambda x: [i.replace(" ","") for i in x])
```

```
movies['crew']=movies['crew'].apply(lambda x: [i.replace(" ","") for i in x])
```

```
movies.head()
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton]



Creating 'tags' attribute:

```
# adding one more column to the existing movies datas et
movies['tags']=movies['overview']+movies['genres']+ movies['keywords']+movies['cast']+movies['crew']
```

```
movies.head()
```

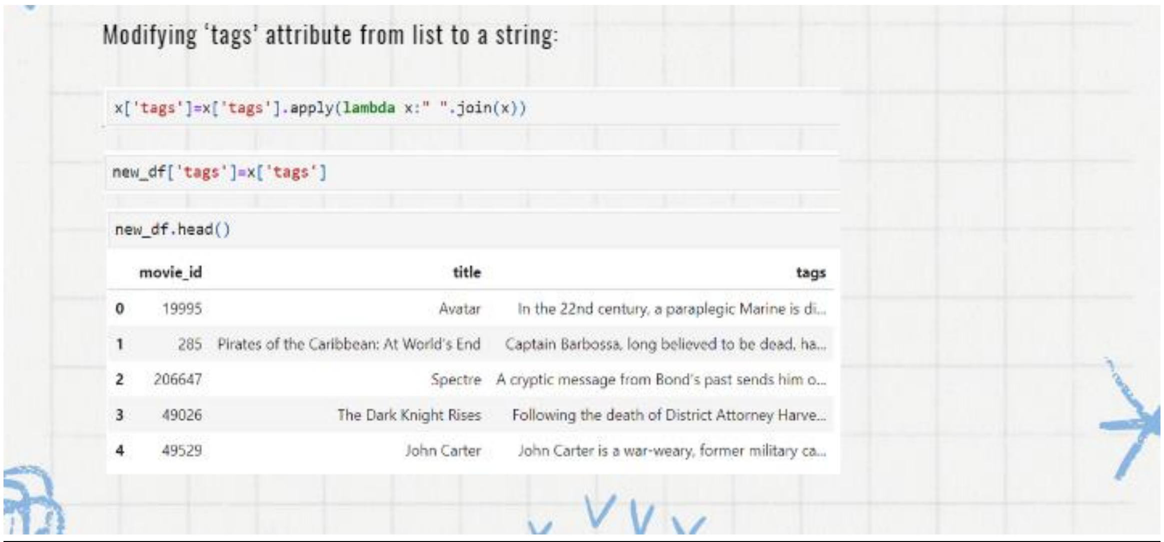
	movie_id	title	overview	genres	keywords	cast	crew	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]	[Captain, Barbossa,, long, believed, to, be, d...

Creating a new dataset with required modified attributes:

```
#creating one more data set to with required modified coluns
new_df=movies[['movie_id','title','tags']]
```

```
new_df
```

	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...



Modifying 'tags' attribute from list to a string:

```
x['tags']=x['tags'].apply(lambda x:" ".join(x))
```

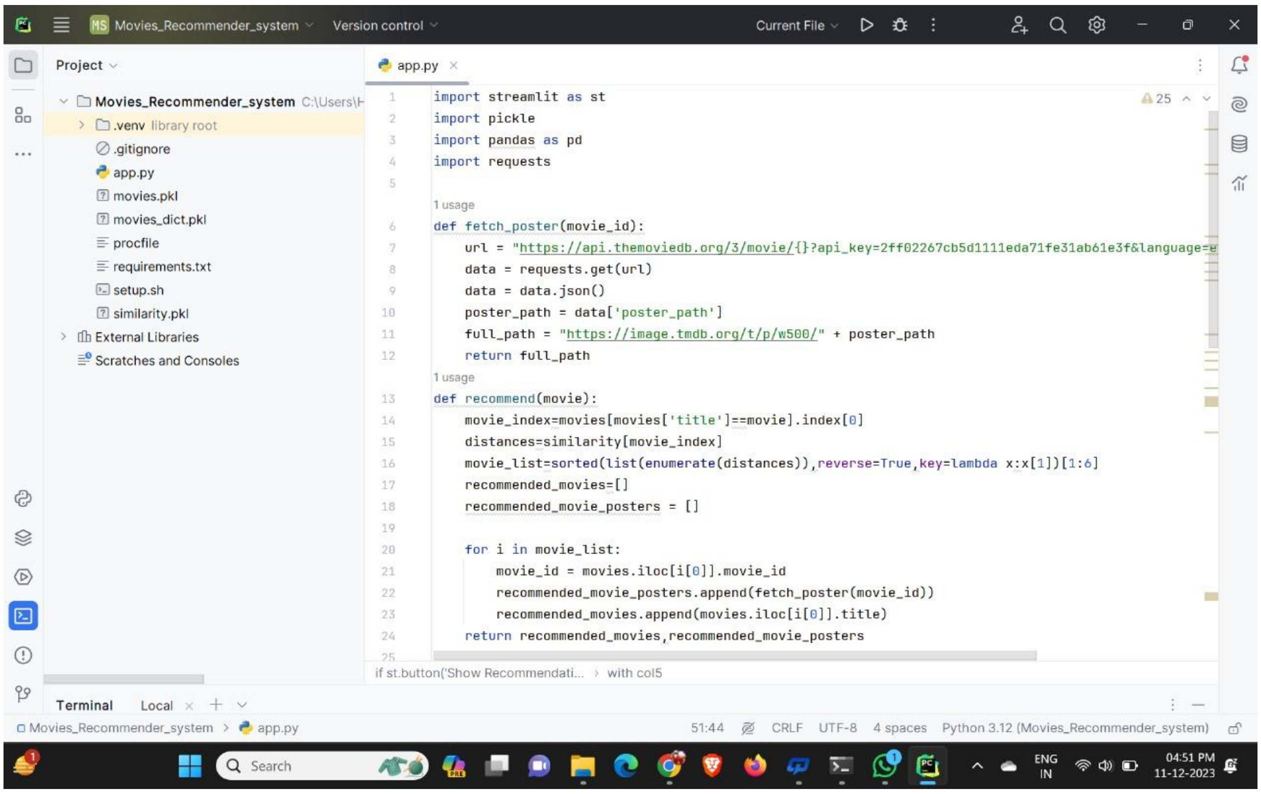
```
new_df['tags']=x['tags']
```

```
new_df.head()
```

	movie_id	title	tags
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	206647	Spectre	A cryptic message from Bond's past sends him o...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...
4	49529	John Carter	John Carter is a war-weary, former military ca...

5.1.2 Training Module

To recommend the movies the dataset is to be trained on cosine similarity.



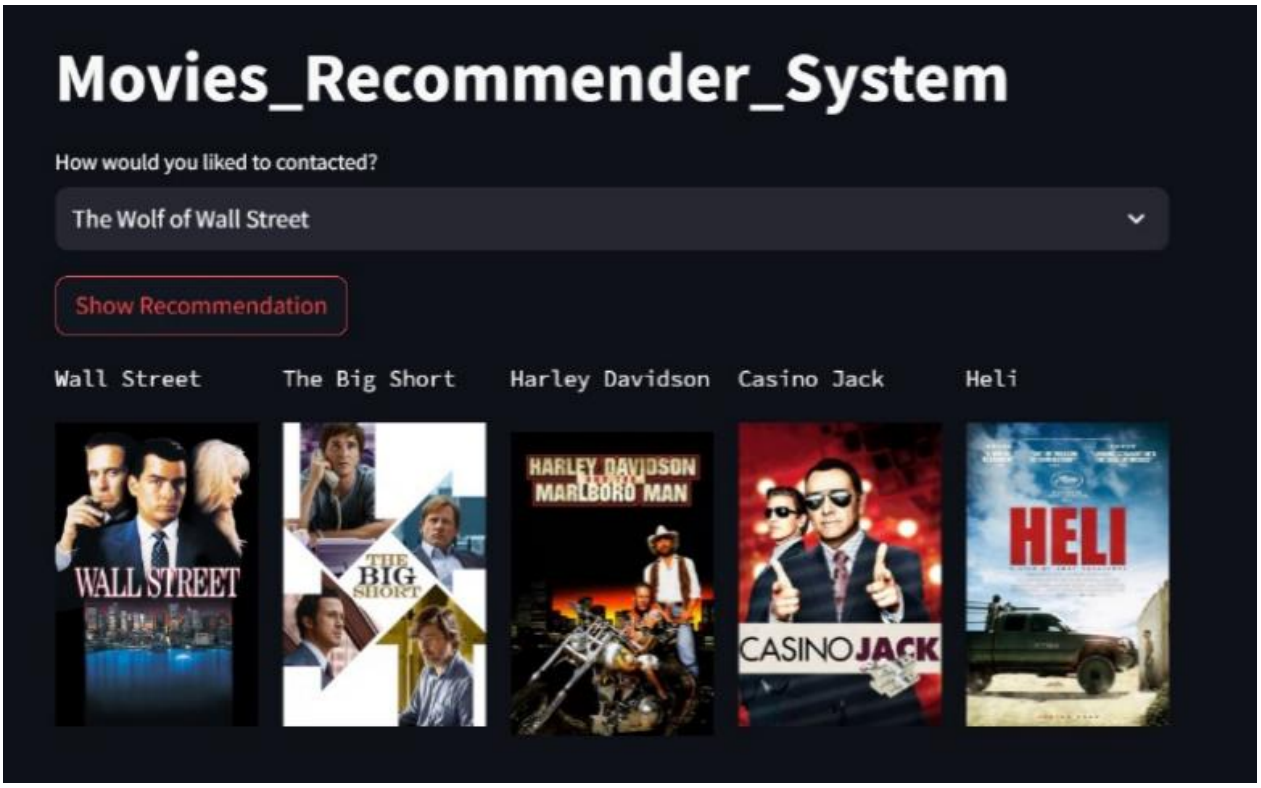
```
import streamlit as st
import pickle
import pandas as pd
import requests

def fetch_poster(movie_id):
    url = "https://api.themoviedb.org/3/movie/{}?api_key=2ff02267cb5d1111eda71fe31ab61e3f&language=e
    data = requests.get(url)
    data = data.json()
    poster_path = data['poster_path']
    full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
    return full_path
def recommend(movie):
    movie_index=movies[movies['title']==movie].index[0]
    distances=similarity[movie_index]
    movie_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]
    recommended_movies=[]
    recommended_movie_posters = []

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

6. TESTING

- Using Cosine Similarity



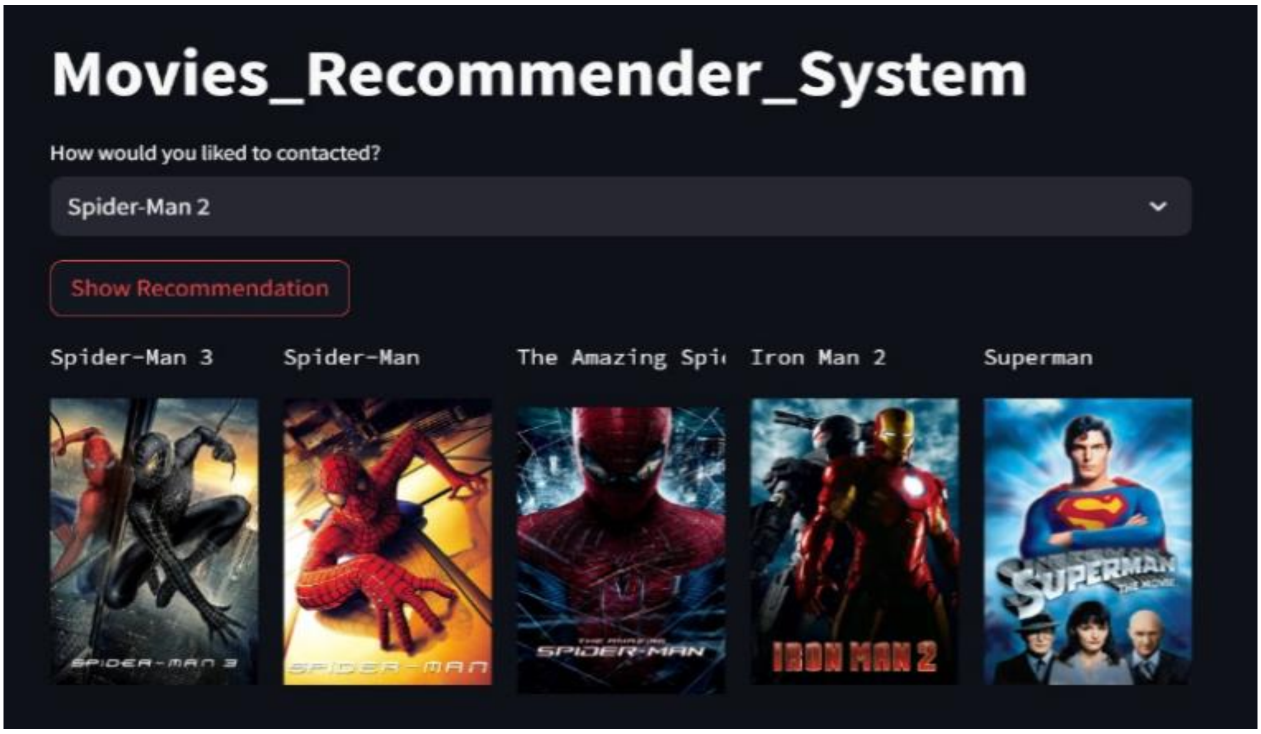
Movies_Recommender_System

How would you liked to contacted?

The Wolf of Wall Street

Show Recommendation

Wall Street · The Big Short · Harley Davidson · Casino Jack · Heli



Movies_Recommender_System

How would you liked to contacted?

Spider-Man 2

Show Recommendation

Spider-Man 3 · Spider-Man · The Amazing Spi · Iron Man 2 · Superman

Fig 9: Logistic Regression Implementation

7. RESULTS

The implemented Python code integrates the selected recommendation model (or a combination of models) and generates personalized movie recommendations for users based on their input. These recommendations are then presented to the users through a user-friendly interface or application.

8. CONCLUSION

The implementation of a sophisticated movie recommendation system presents a transformative solution in the ever-evolving digital entertainment landscape. By harnessing the power of advanced algorithms, machine learning techniques, and personalization, this project offers users a tailored and immersive viewing experience. Through the analysis of individual preferences, viewing histories, and relevant metadata, the system generates curated movie recommendations that resonate with each user's unique tastes.

This personalized approach not only streamlines the content discovery process but also fosters enhanced user engagement. As viewers explore a diverse array of movies aligned with their interests, a deeper connection with the vast cinematic tapestry is cultivated, cultivating a sense of excitement and anticipation for their next cinematic adventure.

Moreover, the movie recommendation system is designed to be adaptive and responsive, continuously evolving through feedback loops and user interactions. This iterative process ensures that the system remains relevant and attuned to the ever-changing preferences of its users, embodying responsible practices in the realm of data-driven recommendations.

By seamlessly integrating cutting-edge technologies with a user-centric approach, this project represents a significant stride towards revolutionizing the way we explore and consume digital entertainment. As the landscape of movie streaming and on-demand services continues to expand, the implementation of such a recommendation system holds the potential to redefine the viewer experience, fostering a deeper appreciation for the art of storytelling and the rich diversity of cinematic offerings.

9. Bibliography

[1] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

[2] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009, 4.

[3] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web (pp. 285-295).

[4] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 76-80.

[5] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

[6] Mobasher, B., Burke, R., Bhaumik, R., & Williams, C. (2007). Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Transactions on Internet Technology, 7(4), 23.

[7] Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (Eds.). (2011). Recommender systems handbook. Springer Science & Business Media.

[8] Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191-198).

[9] Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys, 52(1), 1-38.

[10] Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, R. (2016). Wide & deep learning for recommender systems. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (pp. 7-10).

[11] Deldjoo, Y., Elahi, M., Quadrana, M., & Cremonesi, P. (2021). Explaining movie recommendations: Prose and codecs. Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization (pp. 32-41).

[12] Aggarwal, C. C. (2016). Recommender systems: The textbook. Springer.

[13] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender systems: An introduction. Cambridge University Press.

[14] Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender systems handbook (pp. 73-105). Springer, Boston, MA.