A

High Level Document on

Cinematic Movies Prediction System

Submitted in completion of internship at

INEURON INTELLIGENCE PRIVATE LIMITED

Submitted by:

DEEPAK GANGWANI

(deepakgangwani512@gmail.com)

Year 01/02/2024 – 01/04/2024

DECLARATION

I, *Deepak Gangwani* hereby declare that the Research Project Report titled *"Cinematic* Movies Prediction System Using Content Based Filtering Algorithm*" has been* prepared by me under the guidance of *iNeuron faculty members.* I declare that this Project work is towards the partial fulfillment of the Company Regulations for the award of Internship Certificate from iNeuron, Bengaluru. I have undergone a project for a period of Eight Weeks. I further declare that this Project is based on the original study undertaken by me and has not been submitted for the award of any degree/diploma from any other University/ Institution.

Deepak Gangwani

Abstract

This project report presents the comprehensive design, development, and implementation of a Content-Based Filtering Cinematic Movies Prediction System (CBF-CMRS) aimed at delivering personalized movie recommendations based on individual user preferences. The primary objectives of the project are as follows:

Feature Extraction and Representation:
Explore and define relevant features for movie representation.
Develop techniques for extracting meaningful content-related features, such as genres, keywords, and thematic elements.

Content-Based Recommendation Algorithm:
Build a robust content-based recommendation algorithm that leverages the extracted features to generate personalized movie suggestions.
Implement machine learning models for accurate prediction of user preferences using content-based information.

Enhanced Similarity Measures:
Explore and implement advanced similarity measures to assess the relevance between movies based on content features.
Investigate techniques such as cosine similarity, Jaccard index, or hybrid measures to improve recommendation accuracy.

Integration of External Data Sources:
Investigate the integration of external data sources, such as movie metadata and user reviews, to enrich the content-based recommendation system.
Evaluate the impact of additional data on the system's performance and accuracy.

User Interface and Experience:
Design an intuitive user interface that seamlessly integrates content-based recommendations into the movie discovery process.
Optimize user experience by presenting recommendations in a visually appealing and user-friendly manner.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who have contributed to the successful completion of this project. Our journey would not have been possible without the support, guidance, and encouragement we received from various individuals and entities.

We are also indebted to the officials of iNeuron for providing us with the opportunity toundertake this project and for creating an environment conducive to learning and innovation.
Their commitment to fostering excellence among students has been truly inspiring.

In conclusion, every individual and entity mentioned here has played an essential role in shaping our project and enriching our learning experience. We are truly grateful for their contributions and proud to have been a part of this collective effort.

Thank you.

Deepak Gangwani

Table of Contents

Title	Page Nos.
Executive Summary	6
List of Tables	7
List of Graphs	7
Chapter 1: Introduction, Scope and Background	8
Chapter 2: Review of Literature	12
Chapter 3: Project Planning and Methodology	17
Chapter 4: Data Requirements Analysis, Design and Implementation	22
Chapter 5: Results, Findings, Recommendations, Future Scope and Conclusion	34
Bibliography	65
Appendices	66
Annexure	68
References	69

Executive Summary

In the rapidly evolving landscape of digital entertainment, the importance of recommendation systems has grown significantly. This executive summary encapsulates the essence of our college project, which focused on designing and implementing a Movies Prediction System using a content-based algorithm. Through this project, we aimed to leverage the power of machine learning to enhance the user experience of movie enthusiasts by offering personalized movie suggestions based on their preferences.

Our project employed a content-based recommendation algorithm that utilizes various attributes of movies to identify similarities and generate relevant recommendations. By extracting features such as genres, cast, directors, and plot summaries from a comprehensive movie dataset, we aimed to create a system that understands the content of movies and provides accurate and targeted suggestions.

Through rigorous implementation and evaluation, we found that our content-based Movies Prediction system demonstrated promising results. The algorithm effectively calculated similarities between movies based on extracted features, allowing it to provide recommendationsaligned with user preferences. The system's accuracy in suggesting movies that users are likely toenjoy was a significant achievement of our project.

The success of this project has several implications for both the academic and practical realms. Content-based recommendation systems hold the potential to transform how users interact with digital content, offering a personalized and engaging experience. In the future, we envision enhancing our system by incorporating user feedback, exploring hybrid recommendation techniques, and adapting to evolving user preferences.

In conclusion, our Movies Prediction system project exemplifies the fusion of data science and entertainment, demonstrating the tangible benefits of content-based algorithms in tailoring recommendations to individual tastes. This endeavor not only allowed us to gain valuable insights into the intricacies of recommendation systems but also highlighted the role of AI in shaping modern digital experiences.

We are confident that our project's findings and implications will contribute to the ongoing discourse surrounding recommendation systems and inspire further exploration in this dynamic field.

List of Tables

Table No.	Table Title	Page No.
T-01	Conceptual goals of various Recommender Systems	16
T-02	Gantt Chart	19
T-03	Filtered Movies Dataset	53

List of Graphs

Graph No.	Graph Title	Page No.
F-01	Class Diagram	27
F-02	Activity Diagram	28
F-03	Entities And Attributes	28
F-04	System Flowchart	29
F-05	Sequence Diagram	29
F-06	DFD Diagram	30

CHAPTER 1

INTRODUCTION, SCOPE AND BACKGROUND

1. INTRODUCTION, SCOPE AND BACKGROUND

Overview of Project Case / Business case

In this project, we recommend movies to the users based on the history of their personalized searches and reviews. Both people and online streaming services need a user based personalized Movies Prediction System which helps in analyzing the interests of each individual and recommends the best movies suited to them.

The approach used for building the Movies Prediction System is content-based filtering. As we know that content-based filtering analyses the user's past behavior and recommends items similar to it based on the parameters considered.

In this Content-based Movies Prediction system gives their recommendations based onattributes like Ratings, Genre, and Casts

Background and Motivation

Recommender systems are based on a variety of approaches such as content based, collaborative approach hybrid Furthermost Movies Prediction Systems are centered on collaborative filtering and clustering. In Movies Prediction systems the user is asked to rate the movies which user has already seen then these ratings are applied to recommend other movies to the userthat user has not perceived by utilizing collaborative filtering that is based on similar ratings. Collaborative filtering is tremendously spreading in such a way that this approach influences most of the recommender systems. Collaborative filtering majorly classified into two principal classes such as memory-based collaborative filtering and model based collaborative filtering. Memory-based collaborative filtering explores for nearest neighbors in the user space for an active user and dynamically recommend the movies.

Problem definition

Movies Prediction System provides related content out of relevant and irrelevant collection of items to users of online service providers. Online Movies Prediction System aims to recommend movies to users based on user-movie item ratings, genre, castand previous searches or preferences. It is a web application that allows users to search movies as well as it recommends them appropriate movies based on their search patterns.

Project Scope

Outline

Chapter 1: Introduction gives a general overview of the project along with the background and motivation, problem statement, objective, scope, limitation and the methodology used. Overview provides a basic information about the inner workings of the program. Background and motivation describes how visualization aids better understanding and incentives that led to the development of the program. The problem statement addresses the question "What is the problem that the project will address?" Objectives claims the statements that outlines the problem addressed by the project. The scope defines the area and the information included in the project. The limitation defines the areas in which the project should work on. The methodology defines which software model is used to conduct the project.

Chapter 2: Background Study and Literature Review includes background study, literature review, the current system being used and the problems concerned with it. Background study describes the fundamental theories and concepts related to the project. Literature review highlights the importance of visualization in learning algorithms.

Chapter 3: System Analysis includes requirement analysis, functional requirements, non-functional requirements, feasibility analysis and the analysis. Requirement analysis provides the descriptions of what the system should do; the services it provides and the constraints on its operation. Functional requirements describe the functionality of the system whereas the non-functional requirements describes the system properties related to system performance and usability. Feasibility analysis is an analysis that considers all the project related factors. The analysis may involve the UML diagram based on structured or object-oriented approach

Chapter 4: System Design includes a design and algorithm. Design is the design of UML diagram which is involved in the project. Algorithm is the process which is used to develop the project

Chapter 5: Implementation and Testing presents the tools and technology used for the development of the project, implementation and testing. Implementation describes the system at a finer level of detail, down to the code level. Testing provides the techniques that is used to remove the bugs from the project.

Conclusion and Future Recommendation Conclusion gives a summary of the entire project and proposes question for further study and aims for broader implications. Future Recommendation gives the future work that can be added to the project to make it more feasible.

Objective

The Movies Prediction System provides a mechanism to help users categorize users with similar interests. Basically the purpose of a recommendation system is to search for material that will be interesting to a person. Moreover, it involves a number of factors to create personalized lists of useful and interesting content specific to each user/individual. Recommendation algorithm collects from the user interest. A set of keywords (or features) of an item is the Item profile. For example, consider a scenario in which a person goes to buy his favorite cake 'X' to a pastry. Unfortunately, cake 'X' has been sold out andas a result of this the shopkeeper recommends the person to buy cake 'Y' which is made up of ingredients similar to cake 'X'.

Scope and Limitation

- **Work on several numbers of data:** The number of choices for anything on internet is very high and it's tedious to refine most wanted data by self while searching. The scope of this proposal system includes working within numerous data, with ease.

- **Saving of time**: Many people have problem selecting the alternative item of movie due to lack of time and due to search issues. Also movie recommendations from friends can be time consuming. The system helps in saving lots of time.

- **Relief from processor problem**: Many mobile phone and limited processing power computers can't handle recommender system due to its extremely large dataset. The solution opted for this can be use of web services. The proposed system uses web services, thus makes process simpler.

CHAPTER 2
REVIEW OF LITERATURE

2. REVIEW OF LITERATURE

Literature Review

Background Study With the various types of datasets that have been utilized for detection jobs in today's world, the detection sector has been able to accomplish significant growth. However, because real- life objects are often variable in nature due to environmental differences such as size, background, size, and many other aspects, annotation jobs can be a significant difficulty in the field of categorization and detection.

Supervised Learning There is little doubt machine learning has become one of the most powerful technologies in the last decade. The emphasis on "learning" in machine learning allows computers to make better and better decisions, based on previous experiences. Classical machine learning is often categorized into supervised, unsupervised, semi-supervised or reinforced learning depending on how the algorithm learns to become more accurate in its predictions.

Among them, supervised learning is one of the heavily explored and important form of ML. In Supervised Learning, the learning process is done under the seen label of observation variables. Datasets are trained with the training sets to build a model which is used later on to label new observations or data points from the testing set. As for the training set, the input variables are the features which will influence the accuracy of predicted variable. It contains both quantitative and qualitative variables; the output variable is the label class that Supervised Learning will label the new observations.

Literature Review There are three techniques of recommendation system: Collaborative Filtering, Content Based Filtering and Hybrid Filtering. In Content Based recommender system, user provides data either explicitly (rating) or implicitly (by clicking on a link). The system captures this data and generates user profile for every user. By making use of user profile, recommendation is generated. In content based filtering, recommendation is given by only watching single user's profile. System tries to recommend item similar to that item based on users past activity. Unlike content based, collaborative filtering finds those users whose likings are similar to a given user. It then recommends item or any product, by considering that the given user will also like the item which other users like because their taste are similar. Both these technique have their own strength and weakness so to overcome this, hybrid technique came into picture, which is a combination of both these techniques. Hybrid filtering can be used in various types. We can use content based filtering first and then pass those results to collaborative recommender (and vice-versa) or by integrating both the filter into one model to generate the result. These kinds of modifications are also uses to cope up with cold start, data sparsity and scalability problem.

Content-Based Filtering Content-Based Filtering are also known as cognitive filtering. This filtering recommends item to the user based on his past experience. For example, if a user likes only action movies then the system predicts him only action movies similar to it which he has highly rated. The broader explanation could be suppose the user likes only politics

related content so the system suggests the websites, blogs or the news similar to that content.

Unlike collaborative filtering, content based filtering do not face new user problem. It does not have other user interaction in it. It only deals with particular user's interest. Content based filtering first checks the user preference and then suggest him with the movies or any other product to him. It only focus on single user's ideas, thoughts and give prediction based on his interest. So if we talk about movies, then the content based filtering technique checks the rating given by the user.

Collaborative Filtering The concept of collaborative filtering was first introduced in 1991 by Goldberg et al. [20]. The Tapestry system applies only to smaller user groups (e.g. a single unit), and has too many demands on the user. As a proto-type of collaborative filtering recommendation system, Tapestry presents a new recommendation, but there are many technical deficiencies. Since then, there has been a scoring based collaborative filtering recommendation system, such as Grouplens, which recommends news and films. At present many ecommerce sites have been using the recommendation system such as Amazon, CDNow, Drugstor and Moviefinder etc. There is massive amount of data available. As we all know that today in this busy life no one has time to search hundreds of thousands of item and select the one which is similar to their taste. So collaborative filtering is one of the ways to filter the data and provide the relevant information in which the user is interested in. Collaborative Filtering is one of the most well known techniques for recommending items. This technique suggests relevant item to the user based on neighbour's choice. It first finds out the similarity between the user and his neighbour and then predicts the items. There can be n number of users. This technique finds the similar user from the list of user's. But the similarity between users is found out based the ratings which the users have given to the particular item. This way the approach continues and the desired result is generated. This strategy takes ratings given by user for any item from the large catalog of item catalog of ratings given by the user. This large catalog is referred as user-item matrix.

Current System The reason behind this improvement is the popularity gained by organizations like Netflix whose primary objective is customer satisfaction. Before existence the recommendation system, individuals would physically choose movies to watch from movie libraries. They either had to read the user's reviews or based on the review they would select a movie or had to randomly select a movie. This procedure isn't feasible, as there is an enormous number of spectators with a unique preference for movies. Hence many recommendation systems have been developed over the past decade. These systems use different approaches like a collaborative approach, a content-based approach, a hybrid approach, etc. Taking a look at the behavior and history of different clients, based on their ratings, the system suggests to us what to watch without having to put effort into deciding what to watch. These recommendation systems are categorized into two types, i.e. collaborative filtering approach and content-based approach. The collaborative approach combines the ratings of different users that have similar tastes and then recommends the movies whereas the content-based approach is limited to a single user, where the user's past history and ratings are used for providing recommendations. There are a number of methodologies introduced to implement this recommendation system which includes various fields of Data Mining, Clustering and Bayesian Network methodology.

The Problem with Current System Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations. It's no coincidence that the companies most identified with having excellent recommendations are those with a lot of consumer user data: Google, Amazon and Netflix. A good recommender system firstly needs item data (from a catalog or other form), then it must capture and analyze user data (behavioral events), and then the magic algorithm does its work. The more item and user data a recommender system has to work with, the stronger the chances of getting good recommendations. But it can be a chicken and egg problem – to get good recommendations, you need a lot of users, so you can get a lot of data for the recommendations. There are many other issues that can happen with recommender systems – some offer up too many 'lowest common denominator' recommendations, some don't support The Long Tail enough and just recommend obvious items, outliers can be a problem, and so on. Some other problems are unpredictable items, changing user preferences, changing data etc.

Approach	Conceptual goal	Input
Collaborative	Give me recommendations based on a collaborative approach that leverages the ratings and actions of my peers/myself.	User ratings+ community ratings
Content-Based	Give me recommendations based on the content (attributes) based I have favoured in my past ratings and actions.	User ratings+ Item attributes
Knowledge-Based	Give me recommendations based on my explicit specification based of the kind of content (attributes) I want.	User specification+ Item attributes+ knowledge domain

T-01 Conceptual goals of various Recommender Systems

Feasibility Analysis

Technical Feasibility

In this project, the system was implemented as a web-based application. The primary programming language selected to build the system was Python (version 3.6.5), which is an open-source programming language. The programming language was selected due its ease of use and my experience in working with the programming language. Internal technical capabilities are required to support the project requirements that show the project technically feasible to work upon. Except for python Streamlit technologies will be used in this project, which are one of the most feasible technologies in web development. The tools, systems, modules and libraries needed to build the system are all open source, freely available and are easy to use. Hence, the project was determined technically feasible.

Operational Feasibility The system built in the project follows a client-server architecture. Training of the model, which is resource-intensive and time consuming, is a one-time process and need not be done repeatedly as long as the model holds true for future data acquisition. The well planned design of the system will ensure the optimal utilization of the computer resources (such as storage, memory processing, etc). The system will have simple user interface, so that any non-technical user can operate with the system. As the hardware required for operation of the system is readily available and fairly inexpensive, the product is deemed operationally feasible

Economic Feasibility Personal computer, personal mobile and personal internet subscription was used for the majority of the project. This system is very economically feasible it will not require the budget of huge, amount to create the entire system. As it will not contain the large database and similarly the data will not necessarily require being stored permanently forever so it will become somehow easy to maintain the entire system and the maintenance cost are not over headed. Once the system is ready and user gets guidelines to use it, it is the easiest and very low cost system to implement.

Schedule Feasibility

The schedule feasibility analysis is carried out using the Critical Path Method (CPM). With CPM, critical tasks were identified and interrelationship between tasks were identified which helped in planning that defines critical and non-critical tasks with the goal of preventing time-frame problems and process bottlenecks.

CHAPTER 3
PROJECT PLANNING AND METHODOLOGY

3. PROJECT PLANNING AND METHODOLOGY

Project Planning

Project Plan

- Project Goal: To build a Movies Prediction System that recommends movies to users based on theirpast movie ratings and preferences.
- Project Scope: The system will be implemented using a content-based algorithm. The system will be able to recommend movies to users based on the following features: movie title, genre, cast, director, and keywords.
- Project Deliverables: The system will be delivered as a web application. The application will allow users to view recommendations, and search for movies.
- Project Schedule: The project will be completed in 6 weeks.

T-02 Gantt Chart

The following is a Gantt chart for the project:

```
Week| Activity
------|-----------------------------
1 | Project meeting
1 | Data collection
2 | Data preprocessing
2 | Exploratory data analysis
3 | Feature engineering
3 | Model selection
4 | Model training
4 | Hyper parameter tuning
5 | Model evaluation
5 | System implementation
6 | System testing
6 | Project presentation
```

Communication Plan

The project team will communicate with each other using a project management tool such as Asana or Trello. The team will also hold weekly status meetings to discuss progress and roadblocks.

Acceptance Plan

The system will be accepted by the project sponsor once it meets the following criteria:

- The system is able to recommend movies to users based on their past movie ratings and preferences.
- The system is easy to use.
- The system is reliable.

Resource Plan

The following resources will be needed for the project:

- Hardware: A computer with sufficient processing power and memory.

- Software: A programming language such as Python, a machine learning library such as scikit-learn, and a web development framework such as Django.
- Data: A dataset of movie ratings and metadata.

The following risks have been identified for the project:

- The data may be incomplete or inaccurate.
- The machine learning algorithm may not be able to learn the patterns in the data.
- The system may not be user-friendly.

The project team will mitigate these risks by carefully selecting the data set, using a robust machine learning algorithm, and conducting usability testing.

Methodology

Incremental methodology was chosen for this project as the requirements for the project are clearly known, defined as well as understood. As the methodology supports the process of design, implementation and testing with each increment added over a course of time, increment methodology would be a great fit for the project as animal detection task needs a lot of hit and trial in order to fulfill the proposed requirements and functionalities of the aimed project as well as to achieve a good user experience. However, in the making process of the system some minor features can be added but as the main requirements are understood this methodology would be great fit.

The main requirement of the attendance system is to be able to input photo in the system and get classification / attendance as output. Based on this, the system development was broken down into many partial development projects. These partial system development projects were:

- The development of the system that takes the text as input and passes it to the model.
- The system that takes a text as the input and passes it to the model.
- The system that takes the output generated by the model and gives a output result.

CHAPTER 4
DATA ANALYSIS, DESIGN AND IMPLEMENTATION

4. DATA ANALYSIS, DESIGN AND IMPLEMENTATION

Requirement Analysis
Data Collectio006E

For the implementation of the Movies Prediction System using a Content-Based Algorithm, we utilized a dataset available on Kaggle from The Movie Database (TMDB). The TMDB dataset is a comprehensive collection of movie-related information, including details about movie titles, genres, cast, directors, synopses, ratings, and more. The dataset serves as a valuableresource for building a content-based recommendation system ant the link to download the dataset is **https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?select=tmdb_5000_movies.csv**
.

Steps for Data Collection:

- **Identifying the Dataset:**
  We began by searching for a suitable dataset that aligns with our project's objectives. The TMDB dataset, available on Kaggle, met our requirements as it contains a wide range of attributes related to movies, making it ideal for creating a content-based recommendation system.

- **Dataset Exploration:**
  We thoroughly explored the TMDB dataset to understand its structure, the types of information available, and the relevance of each attribute to our project. This step helped us identify which attributes to utilize for generating movie recommendations.

- **Dataset Download:**
  After confirming the suitability of the TMDB dataset, we downloaded the dataset from Kaggle. The dataset was available in a structured format, typically in CSV or JSON files, which made it easy to import and process the data.

- **Data Preprocessing:**
  Preprocessing the dataset was essential to ensure data quality, consistency, and relevance. We conducted several preprocessing steps, including handling missing values, cleaning text data, and converting categorical attributes into a suitable format for analysis.

- **Feature Selection:**
  We carefully selected relevant attributes for building the content-based recommendation system. Attributes such as movie genres, cast, directors, and synopses were of particular interest, as they form the basis for generating recommendations.

- **Feature Extraction:**
  For certain attributes like synopses, we employed natural language processing (NLP) techniques to extract relevant features that could be used for comparison and recommendation.

- **Creating the Feature Matrix:**
  By combining the selected attributes, we constructed a feature matrix that represents the characteristics of each movie in a structured format. This matrix served as the foundation

for computing similarities between movies and generating recommendations.

Benefits of Using Kaggle TMDB Dataset:

- Comprehensive: The dataset provides a wealth of information about movies, making it suitable for building a detailed content-based recommendation system.
- Preprocessed: The dataset is already cleansed and organized, reducing the need for extensive data cleaning efforts.
- Variety: The dataset contains a diverse range of movie attributes, allowing for the creation of nuanced and personalized recommendations.

Data Analysis and tools of data analysis

- **NumPy:** NumPy is a library for scientific computing in Python. It provides a high-performance multidimensional array object, along with a suite of functions for operating on these arrays. NumPy is used in a wide variety of scientific and engineering applications, including machine learning, data science, and financial modeling.

NumPy library in Python

- **Pandas:** Pandas is a library for data analysis in Python. It provides high-performance, easy-to-use data structures and data analysis tools. Pandas is used for a wide variety of data analysis tasks, including data cleaning, data wrangling, data visualization, and statistical analysis.

Pandas library in Python

- **ast:** ast is a library for abstract syntax trees in Python. It provides classes and functions for representing and manipulating abstract syntax trees. ast is used by the Python interpreter to parse Python code, and it is also used by other Python libraries, such as NumPy and Pandas, to represent data structures.

Python abstract syntax tree structure: x = y + 3

ast library in Python

- **Scikit-learn (CountVectorizer):** Scikit-learn is a machine learning library in Python. It provides a variety of machine learning algorithms, including classification, regression, clustering, and dimensionality reduction algorithms. The CountVectorizer class in Scikit-learn is used to convert text into a numerical representation that can be used by machine learning algorithms.

```
text = ['Hello my name is james, this is my python notebook']
```

The text is transformed to a sparse matrix as shown below.

	hello	is	james	my	name	notebook	python	this
0	1	2	1	2	1	1	1	1

Scikit-learn library in Python

- **NLTK (PorterStemmer):** NLTK is a natural language processing library in Python. It provides a variety of natural language processing tools, including stemmers, tokenizers, and parsers. The PorterStemmer class in NLTK is used to stem words, which is the process of removing common prefixes and suffixes from words.

NLTK library in Python

- **Scikit-learn (Cosine Similarity):** The Cosine Similarity class in Scikit-learn is used to measure the similarity between two vectors. The cosine similarity is a measure of the angle between two vectors. Two vectors with a cosine similarity of 1 are perfectly aligned, while two vectors with a cosine similarity of 0 are orthogonal.

scikit-learn

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

Figure 1: Cosine Similarity Formula

- **Pickle:** Pickle is a library for serializing and deserializing Python objects. It can be used to save Python objects to a file and then load them back into memory. Pickle is used to store machine learning models, data structures, and other Python objects.

Pickle library in Python

These libraries play crucial roles in various areas of programming, data analysis, natural language processing, and machine learning. By utilizing their functionalities, developers and data scientists can efficiently manipulate data, perform advanced computations, process text, build recommendation systems, and more.

Design

Class Diagram

Activity Diagram

Entities and attributes

System Flowchart

Sequence Diagram

DFD Diagram

Fig: DFD Level-0

Figure : DFD Level-1

Implementation (for application-based project)

Movies Prediction System is a web-based app, which provides all the details of the requested movie. Details include recommended movies, as well as top cast, ratings, reviews and so on.
The requirement of this project is given below:

Functional Requirement

- Autosuggest user for smooth experience.
- Simple loading screen to inform user that work is in progress.
- Notify user if result is not found.
- Simply UI to show more details about the casts.

Non-Functional Requirement

• The processing of each request should be done around 10 seconds.
• Display default data if some data are missing.
• Search result, although autosuggest unable to suggest.

Hardware Requirements

• Processor: Pentium IV or higher
• RAM: 256 MB
• Network: Bandwidth greater than 50 KBps (400 kbps)

Software Requirements

• OS: Windows, iOS, Linux
• Browser: Chrome, Brave, Mozilla Firefox, Microsoft Edge, Apple Safari, Opera.
- Application Software: IDE(Integrating Developing Environment) Python Libraries like numpy, pandas, streamlit for user interface, sklearn, textvectorizer, pickle, etc

Algorithm Details

Similarity Score

It is a numerical value ranges between zero to one which helps to determine how much two items are similar to each other on a scale of zero to one. This similarity score is obtained measuring the similarity between the text details of both of the items. So, similarity score is the measure of similarity between given text details of two items. This can be done by cosine similarity.

Cosine Similarity

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

Figure 1: Cosine Similarity Formula

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

Figure 2: Working with Cosine Similarity Formula

Example

Let's take two texts, Text1 and Text2. Both of them have similar word or similar content, but they are not exact sentences.

Text A: London Paris London

Text B: Paris Paris London

Figure 3: Text Example

Now, we try to quantify the similarity. To come up with mathematical number or a score, which gives us an indication of how similar these words are

Figure 4: Graph to Plot

We count the number of times they appeared then plot them under a graph x-axis being London and y-axis is Pairs.

Figure 5: Plotting two Vectors

Now we can use the cosine similarity to find the theta (θ)

Figure 6: Finding Degree (θ)

This will return the value in between 0 to 1, with 1 being the most similar.

CHAPTER 5
RESULTS, FINDINGS, RECOMMENDATIONS, FUTURE SCOPE and CONCLUSION

5. RESULTS, FINDINGS, RECOMMENDATIONS, FUTURE SCOPE AND CONCLUSION

Results of the project work

HOME
ABOUT
TEAM
CONTACT
himanshu@123
Wybierz język
Steps Taken To Create The Project
Step 1
Step 2
Step 3
Step 4
Step 5
#1
Movie Dataset
Downloading Movie Dataset
The TMDB (The Movie Database) movies dataset is a comprehensive collection of information about thousands of movies, including details such as titles, release dates, genres, ratings, cast and crew lists, and plot summaries. The dataset is frequently updated to reflect new releases and changes in information.
It is a popular resource for researchers, developers, and movie enthusiasts who use it for various purposes such as building movie recommendation systems, analyzing trends in the movie industry, and conducting content analysis of films. The dataset is available in various formats, including CSV, JSON, and SQL, and can be accessed through the TMDB API or downloaded from their website.

HOME
ABOUT
TEAM
CONTACT
GUEST USER
REGISTER
LOGIN
Wybierz język
Cinematic
Despicable Me 2
Movie
6.92/10
HD
Animation, Comedy, Family
2013
98 min
WATCH NOW

HOME
ABOUT
TEAM
CONTACT
GUEST USER
Wybierz język
ONLINE STREAMING
All Time Movies
Movies
TV Shows
Anime
OCEANS
Wallace & Gromit
THE CURSE OF THE WERE-RABBIT
SURF'S UP
ALEX RIDER
OPERATION STORMBREAKER
Oceans
2010
HD
84 min
7.4/10
Wallace & Gromit: The Curse of the Were-Rabbit
2005
HD
85 min
7.071/10
Surf's Up
2007
HD
85 min
6.406/10
Stormbreaker
2006
HD
93 min
5.2/10

HOME
ABOUT
TEAM
CONTACT
GUEST USER
Wybierz język
Only $3.99
HD 4K
Resolution!
DOWNLOAD
OUR SERVICES
Download Your Shows Watch Offline.
Websites use algorithms to recommend movies based on a user's viewing history, ratings, and preferences. This helps users discover new movies they might enjoy and saves them time from browsing through endless lists of movies.
Enjoy on Your TV.
Users can watch movies and TV shows online without having to download them. This is a convenient option for people who do not want to wait for a movie to download or do not have enough storage space on their devices.
Watch Everywhere.
Some websites allow users to rent movies for a specific period, usually 24-48 hours, at a lower cost than purchasing or streaming. This is a good option for people who want to watch a movie once or twice and do not want to commit to buying it.

HOME
ABOUT
TEAM
CONTACT
GUEST USER
Wybierz język
ONLINE STREAMING
Top Rated Movies
MOVIES
TV SHOWS
DOCUMENTARY
SPORTS
Muppets from Space 1999
HD 87 min 6.0/10
Bandslam 2009
HD 111 min 5.8/10
Against the Wild 2013
HD 90 min 5.164/10
Aquamarine 2006
HD 104 min 6.4/10
TRIAL START FIRST 30 DAYS.
Enter your email to create or restart your membership.
Enter your email
GET STARTED
FAQ
HELP CENTER
TERMS OF USE
PRIVACY
© 2023 . All Rights Reserved
Assistant
You 12:50 PM
Hi
12:50 PM
Hello and welcome to my portfolio website how can I assist you ?
Type something...
Built with IBM watsonx

HOME
ABOUT
TEAM
CONTACT
himanshu@123
LOGOUT
Wybierz język
Pirates of the Caribbean: Dead Man's Chest
7.355/10
HD
Adventure, Fantasy, Action
2006
151 min
'Captain Jack Sparrow works his way out of a blood debt with the ghostly Davy Jones to avoid eternal damnation.',
Your previous rating was: 10
Share
Prime Video
Streaming Channels
WATCH NOW
DOWNLOAD

HOME
ABOUT
TEAM
CONTACT
himanshu@123
Wybierz język
MOVIE CAST DETAILS
Director And Top 3 Actor Details
Gore Verbinski
Director
Johnny Depp
Jack Sparrow
Orlando Bloom
Will Turner
Keira Knightley
Elizabeth Swann

HOME
ABOUT
TEAM
CONTACT
himanshu@123
Wybierz język
BEST MOVIE SERIES
World Best Recommended Movie Series
Pirates of the Caribbean: At World's End
2007
HD
169 min
7.3/10
Pirates of the Caribbean: On Stranger Tides
2011
HD
137 min
6.547/10
Pirates of the Caribbean: The Curse of the Black Pearl
2003
HD
143 min
7.8/10
The Pirates! In an Adventure with Scientists!
2012
HD
88 min
6.457/10

HOME
ABOUT
TEAM
CONTACT
GUEST USER
Wybierz język
Login
Username:
This field is required.
Password:
SHOW
This field is required.
LOGIN
CREATE A NEW ACCOUNT

HOME
ABOUT
TEAM
CONTACT
GUEST USER
Wybierz język
Register
Username:
This field is required.
Password:
SHOW
This field is required.
Confirm Password:
SHOW
This field is required.
SUBMIT
ALREADY REGISTERED GO TO LOGIN

HOME
ABOUT
TEAM
CONTACT
GUEST USER
Wybierz język
Login
Username:
himanshu@123
Password:
SHOW
LOGIN
CREATE A NEW ACCOUNT

HOME
ABOUT
TEAM
CONTACT
himanshu@123
LOGOUT
Wybierz język
Movies Show and More
BEST MOVIE SERIES
World Best Recommended Movie Series
NO MOVIE FOUND PLEASE SELECT OR TYPE YOUR MOVIE PREFERENCE.

HOME
ABOUT
TEAM
CONTACT
himanshu@123
Wybierz język
Ava
Avatar
The Adventures of Sharkboy a...
Savages
Caravans
The Savages
Force 10 from Navarone
Savage Grace
BEST MOVIE SERIES
est Recommended Movie Series
MOVIE FOUND PLEASE SELECT OR TYPE YOUR MOVIE PREFERENCE.

Movies Show and More
Avatar
7.38/10
HD
Action, Adventure, Fantasy, Science Fiction
2009
162 min
('In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.',)
Your previous rating was: 9
AVATAR
Share
Prime Video
Streaming Channels
WATCH NOW
DOWNLOAD

HOME
ABOUT
TEAM
CONTACT
himanshu@123
Director And Top 3 Actor Details
James Cameron
Sam Worthington
Zoe Saldaña
Sigourney Weaver
World Best Recommended Movie Series
TITAN A.E.
SMALL SOLDIERS
INDEPENDENCE DAY
AVP REQUIEM
Titan A.E.
Small Soldiers
Independence Day
Aliens vs Predator: Requiem

CINEMATIC
HOME
ABOUT
TEAM
CONTACT
himanshu@123
Wybierz język
Team Member

CINEMATIC
HOME
ABOUT
TEAM
CONTACT
himanshu@123
Wybierz język
Team Member
Deepak
Gangwani
PYTHON DEVELOPER
Jodhpur
Rajasthan
Directions
View larger map
Umaid Bhawan Palace
Contact us
For any queries and feedback you
anytime we are available 24/7 for a
DEEPAK GANGWANI
deepakgangwani512@gmail.com
Hello my name is deepak
PHONE
+91 7414804515
EMAIL
deepakgangwani512@gmail.com
Assistant
You 1:15 PM
hi
1:15 PM
Hello and welcome to my portfolio website how can I assist you ?
You 1:15 PM
Want to know more about your project
1:15 PM
Want to have conversation with me then select yes otherwise no.
Yes
No
Type something...
Built with IBM watsonx

Findings based on analysis of data

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [22]:
```
import numpy as np
import pandas as pd
```

Creating A Content Based Recommender System Of Kaggle Dataset

In [23]:
```
# now importing the csv files of kaggle dataset into variable using pandas library
movies=pd.read_csv("tmdb_5000_movies.csv")#this file represents the data related movie theme and history
credits=pd.read_csv("tmdb_5000_credits.csv")# this file represents the cast or actor who acted in this movie
```

In [24]:
```
movies.head(1)
```

Out[24]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	production_
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166 "name": "Uni

In [25]:
```
credits.head(1)
```

Out[25]:

	movie_id	title	cast	crew

In [25]:
```
credits.head(1)
```

Out[25]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

In [26]:
```
credits.head(1)
```

In [30]:
```
movies.head(1)
```

Out[30]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	...	runtime
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	...	162.0

1 rows × 23 columns

In [31]:
```
# movies available in different type of languages in dataset
movies["original_language"].value_counts()
```

Out[31]:
```
en    4510
fr      70
zh      27
de      27
hi      19
ja      16
```

```
id      2
ar      2
vi      1
sl      1
ps      1
no      1
ky      1
hu      1
pl      1
af      1
nb      1
tr      1
is      1
xx      1
te      1
el      1
Name: original_language, dtype: int64
```

In [32]:
```
# To see number of columns and it's data type
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4809 entries, 0 to 4808
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4809 entries, 0 to 4808
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4809 non-null   int64
 1   genres                4809 non-null   object
 2   homepage              1713 non-null   object
 3   id                    4809 non-null   int64
 4   keywords              4809 non-null   object
 5   original_language     4809 non-null   object
 6   original_title        4809 non-null   object
 7   overview              4806 non-null   object
 8   popularity            4809 non-null   float64
 9   production_companies  4809 non-null   object
 10  production_countries  4809 non-null   object
 11  release_date          4808 non-null   object
 12  revenue               4809 non-null   int64
 13  runtime               4807 non-null   float64
 14  spoken_languages      4809 non-null   object
 15  status                4809 non-null   object
 16  tagline               3965 non-null   object
 17  title                 4809 non-null   object
 18  vote_average          4809 non-null   float64
 19  vote_count            4809 non-null   int64
 20  movie_id              4809 non-null   int64
 21  cast                  4809 non-null   object
 22  crew                  4809 non-null   object
dtypes: float64(3), int64(5), object(15)
memory usage: 901.7+ KB
```

In [33]:
```
# Now I will remove those column which is not recommended for data analysis for recommendation
# Those column which I put on database is given below
# genres
# id
# keywords
# title
# overview
# cast
# crew
# this variable is getting updated after doin filtering for recommendation
movies=movies[["movie_id","title","overview","genres","keywords","cast","crew"]]
```

In [34]:
```
movies.head(1)
```

Out[34]:

	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...

In [35]:
```
movies.isnull().sum() #findind those column rows whose data is empty
```

Out[35]:
```
movie_id    0
title       0
overview    3
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

In [36]:
```
movies.dropna(inplace=True)#removing the empty column row in the dataset that is overview is showing three
```

In [37]:
```
movies.isnull().sum() #findind those column rows whose data is empty
```

Out[37]:
```
movie_id    0
title       0
overview    0
genres      0
keywords    0
cast        0
crew        0
dtype: int64
```

In [38]:
```
# code to check duplicate data is their or not using below code
movies.duplicated().sum()
```

Out[38]:
```
0
```

In [39]:
```
movies.iloc[0].genres #mining first row with genres column data
```

Out[39]:
```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fi
ction"}]'
```

In [40]:
```
# I will convert these string in the above data into integer for fetch pupose using one library
import ast
```

In [41]:
```
# creating a function to convert the above data into single array which value of name key
def convert(obj):
    l=[]
    for i in ast.literal_eval(obj):
        l.append(i["name"])
    return l
```

In [42]:
```
# Implementing the function on dataset to seperate the generes data into list on all the rows
movies["genres"]=movies["genres"].apply(convert)
```

In [43]:
```
movies["genres"]
```

Out[43]:
```
0       [Action, Adventure, Fantasy, Science Fiction]
1                        [Adventure, Fantasy, Action]
2                         [Action, Adventure, Crime]
```

```
In [43]: movies["genres"]
```

```
Out[43]: 0       [Action, Adventure, Fantasy, Science Fiction]
         1                       [Adventure, Fantasy, Action]
         2                         [Action, Adventure, Crime]
         3                    [Action, Crime, Drama, Thriller]
         4                [Action, Adventure, Science Fiction]
                                        ...
         4804                      [Action, Crime, Thriller]
         4805                                [Comedy, Romance]
         4806               [Comedy, Drama, Romance, TV Movie]
         4807                                              []
         4808                                   [Documentary]
         Name: genres, Length: 4806, dtype: object
```

```
In [44]: movies.head(1)
```

Out[44]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

```
In [45]: # The same process will be done to keywords column using the same above steps
         # Implementing the function on dataset to seperate the keywords data into list on all the rows
         movies["keywords"]=movies["keywords"].apply(convert)
         movies["genres"]
```

```
Out[45]: 0       [Action, Adventure, Fantasy, Science Fiction]
         1                       [Adventure, Fantasy, Action]
         2                         [Action, Adventure, Crime]
         3                    [Action, Crime, Drama, Thriller]
         4                [Action, Adventure, Science Fiction]
                                        ...
         4804                      [Action, Crime, Thriller]
         4805                                [Comedy, Romance]
         4806               [Comedy, Drama, Romance, TV Movie]
         4807                                              []
         4808                                   [Documentary]
         Name: genres, Length: 4806, dtype: object
```

```
In [46]: movies.head(1)
```

Out[46]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

```
In [47]: # creating a function to convert the above data into single array which value of name key
         # In cast column I will fetch those actor which are in the first top 3 that I will show
         def convert3(obj):
             l=[]
             count=0
             for i in ast.literal_eval(obj):
                 if(count<3):
                     l.append(i["name"])
                     count=count+1
                 else:
                     break
             return l
```

```
In [48]: # Selecting the cast column from dataset for filtering the top 3 actors in every movie
         movies["cast"]=movies["cast"].apply(convert3)
         movies["cast"]
```

```
Out[48]: 0        [Sam Worthington, Zoe Saldana, Sigourney Weaver]
         1           [Johnny Depp, Orlando Bloom, Keira Knightley]
         2            [Daniel Craig, Christoph Waltz, Léa Seydoux]
         3            [Christian Bale, Michael Caine, Gary Oldman]
         4          [Taylor Kitsch, Lynn Collins, Samantha Morton]
                                          ...
         4804    [Carlos Gallardo, Jaime de Hoyos, Peter Marqua...
         4805         [Edward Burns, Kerry Bishé, Marsha Dietlein]
         4806           [Eric Mabius, Kristin Booth, Crystal Lowe]
         4807            [Daniel Henney, Eliza Coupe, Bill Paxton]
         4808    [Drew Barrymore, Brian Herzlinger, Corey Feldman]
         Name: cast, Length: 4806, dtype: object
```

In [49]:
```
movies.head(1)
```

Out[49]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[{"credit_id": "52fe48009251416c750aca23", "de...

In [50]:
```
# now I will do filtering in crew column
# let just see the column before filtering
movies["crew"][0]
```

Out[50]:
```
'[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor", "name": "Steph
en E. Rivkin"}, {"credit_id": "539c47ecc3a36810e3001f87", "department": "Art", "gender": 2, "id": 496, "job": "Production Des
ign", "name": "Rick Carter"}, {"credit_id": "54491c89c3a3680fb4001cf7", "department": "Sound", "gender": 0, "id": 900, "job":
"Sound Designer", "name": "Christopher Boyes"}, {"credit_id": "54491cb70e0a267480001bd0", "department": "Sound", "gender": 0,
"id": 900, "job": "Supervising Sound Editor", "name": "Christopher Boyes"}, {"credit_id": "539c4a4cc3a36810c9002101", "depart
ment": "Production", "gender": 1, "id": 1262, "job": "Casting", "name": "Mali Finn"}, {"credit_id": "5544ee3b925141499f0008f
c", "department": "Sound", "gender": 2, "id": 1729, "job": "Original Music Composer", "name": "James Horner"}, {"credit_id":
"52fe48009251416c750ac9c3", "department": "Directing", "gender": 2, "id": 2710, "job": "Director", "name": "James Cameron"},
```

In [51]:
```
# Now I will filter that data from crew column whose job role is director with the director name

# creating a function to convert the above data into single array which value of name key with director name
# In crew column I will fetch those director name who the directed the movie
def fetch_director(obj):
    l=[]
    for i in ast.literal_eval(obj):
        if (i["job"]=="Director"):
            l.append(i["name"])
            break
    return l
```

In [52]:
```
movies["crew"]=movies["crew"].apply(fetch_director)
movies["crew"]
```

Out[52]:
```
0           [James Cameron]
1          [Gore Verbinski]
2              [Sam Mendes]
3       [Christopher Nolan]
4          [Andrew Stanton]
                  ...
4804     [Robert Rodriguez]
4805         [Edward Burns]
4806          [Scott Smith]
4807          [Daniel Hsia]
4808     [Brian Herzlinger]
Name: crew, Length: 4806, dtype: object
```

In [57]:
```
movies.head(1)
```

Out[57]:

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]

In [59]:
```
# Now I have to convert the cast data by removing whitespace
# example
# befor:-Sam Worthington
# after:-SamWorthington
# I am doing this filtering for not recommending the same name of different movie
# If their is one sam in cast and another sam in crew the machine will get confuse and this will recommend wring movies
# so we are joining the first name and last name by removing same to find the person easily
movies["genres"]=movies["genres"].apply(lambda x:[i.replace(" ","") for i in x])#list comprehension for filtering
movies["keywords"]=movies["keywords"].apply(lambda x:[i.replace(" ","") for i in x])#list comprehension for filtering
movies["cast"]=movies["cast"].apply(lambda x:[i.replace(" ","") for i in x])#list comprehension for filtering
movies["crew"]=movies["crew"].apply(lambda x:[i.replace(" ","") for i in x])#list comprehension for filtering
```

In [60]:
```
movies.head(1)
```

Out[60]:

	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]

In [61]:
```
# Now I will create a tag column which will cancatenate all 4 list columns in one tags columns
movies["tags"]=movies["overview"]+movies["genres"]+movies["keywords"]+movies["cast"]+movies["crew"]
movies.head(1)
```

Out[61]:

	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...

In [67]:
```
# Now removing the list column because I have cancatenate them in one column that is tags
# creating a new data frame which will have 3 column that is title and tags for recommendation analysis
new_df=movies[["movie_id","title","tags"]]
new_df.head()
```

Out[67]:

	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...

In [69]:
```
#Now I will convert the the tags list into string by using list comprehension method
new_df["tags"]=new_df["tags"].apply(lambda x:" ".join(x))
new_df["tags"]
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_1056\972886567.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df["tags"]=new_df["tags"].apply(lambda x:" ".join(x))
```

Out[69]:
```
0       In the 22nd century, a paraplegic Marine is di...
1       Captain Barbossa, long believed to be dead, ha...
2       A cryptic message from Bond’s past sends him o...
3       Following the death of District Attorney Harve...
4       John Carter is a war-weary, former military ca...
                                ...
4804    El Mariachi just wants to play his guitar and ...
4805    A newlywed couple's honeymoon is upended by th...
4806    "Signed, Sealed, Delivered" introduces a dedic...
4807    When ambitious New York attorney Sam is sent t...
4808    Ever since the second grade when he first saw ...
Name: tags, Length: 4806, dtype: object
```

In [71]:
```
new_df["tags"][0]# particular first row tags column converted into string
```

Out[71]:
```
'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between follo
wing orders and protecting an alien civilization. Action Adventure Fantasy ScienceFiction cultureclash future spacewar spacecol
ony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrela
tions mindandsoul 3d SamWorthington ZoeSaldana SigourneyWeaver JamesCameron'
```

In [72]:
```
new_df.head()
```

Out[72]:

	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...

In [74]:
```
# converting the tags data into lowercase due to some case-sensitive reason
new_df["tags"]=new_df["tags"].apply(lambda x:x.lower())
new_df.head()
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_1056\4094177471.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df["tags"]=new_df["tags"].apply(lambda x:x.lower())
```

Out[74]:

	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...

```
new_df["tags"]=new_df["tags"].apply(lambda x:x.lower())
```

Out[74]:

	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...

Filtered Movie Dataset Table

Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

Text Vectorization Process Will Be Done After Filtering All the Dataset

```
# whichever the strings in tags who are having the closest vector rate they can be considered as similar type of movie which
can be recommended
# For converting text into vectors their are many methods like:-1)Bag Of Words,2)tfidf,3)word to vec
# But I will use bag of words which is more simpler compare to other vector methods
# In this method I will seperate those tags in which the maximum times particular word is reapeated
# Finding the 5000 words which is reapeted maximum times
```

In [77]:
```
# Now I have to identify the similarity between the first two columns of tags using some concept
new_df["tags"][0]
```

Out[77]:
```
'in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between follo
wing orders and protecting an alien civilization. action adventure fantasy sciencefiction cultureclash future spacewar spacecol
ony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrela
tions mindandsoul 3d samworthington zoesaldana sigourneyweaver jamescameron'
```

In [78]:
```
new_df["tags"][1]
```

Out[78]:
```
"captain barbossa, long believed to be dead, has come back to life and is headed to the edge of the earth with will turner and
elizabeth swann. but nothing is quite as it seems. adventure fantasy action ocean drugabuse exoticisland eastindiatradingcompan
y loveofone'slife traitor shipwreck strongwoman ship alliance calypso afterlife fighter pirate swashbuckler aftercreditsstinger
johnnydepp orlandobloom keiraknightley goreverbinski"
```

In [102]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=5000,stop_words="english")
```

In [103]:
```
vectors=cv.fit_transform(new_df["tags"]).toarray()
vectors
```

Out[103]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [104]:
```
vectors.shape
```

Out[104]:
```
(4806, 5000)
```

In [105]:
```
# showing the first movie data in vector form that is avatar
vectors[0]
```

Out[105]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [87]: # list of 5000 words which is reapeated maximum times in alphabetical order
         cv.get_feature_names()
```

```
Out[87]: ['000',
          '007',
          '10',
          '100',
          '11',
          '12',
```

```
In [88]: len(cv.get_feature_names())
```

```
Out[88]: 5000
```

```
In [90]: # Applying steming method which convert the singular and plural words into one word
         # example
         # Before:-["loved","loving","love"]
         # After:-["Love","Love","Love"]
         !pip install nltk
```

```
Collecting nltk
  Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
     ---------------------------------------- 1.5/1.5 MB 3.3 MB/s eta 0:00:00
Requirement already satisfied: click in c:\users\owner\appdata\local\programs\python\python310\lib\site-packages (from nltk)
(8.1.3)
Collecting regex>=2021.8.3
```

```
In [91]: import nltk # it is a natural laguage processing library in python
```

```
In [92]: from nltk.stem.porter import PorterStemmer
         ps=PorterStemmer()
```

```
In [94]: # creating function with help of library which will convert the plural word into singular
         # example
         ps.stem("loving")
```

```
Out[94]: 'love'
```

```
In [96]: ps.stem("loved")
```

```
Out[96]: 'love'
```

```
In [97]: def stem(text):
             y=[]
             for i in text.split():
                 y.append(ps.stem(i))

             return " ".join(y) #converting list into string again after doing changes
```

ile Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [100]:
```
# applying stem function in dataframe
new_df["tags"]=new_df["tags"].apply(stem)
new_df["tags"]
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_1056\782021188.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df["tags"]=new_df["tags"].apply(stem)
```

Out[100]:
```
0       in the 22nd century, a parapleg marin is dispa...
1       captain barbossa, long believ to be dead, ha c...
2       a cryptic messag from bond' past send him on a...
3       follow the death of district attorney harvey d...
4       john carter is a war-weary, former militari ca...
                                ...
4804    el mariachi just want to play hi guitar and ca...
4805    a newlyw couple' honeymoon is upend by the arr...
4806    "signed, sealed, delivered" introduc a dedic q...
4807    when ambiti new york attorney sam is sent to s...
4808    ever sinc the second grade when he first saw h...
Name: tags, Length: 4806, dtype: object
```

In [106]:
```
# After using steming let see the changes in 5000 words
cv.get_feature_names()
```

Out[106]:
```
['000',
 '007',
 '10',
```

Creating a method to calculate the cosine distance between every movie which get compared

In [107]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [108]:
```
similarity=cosine_similarity(vectors)#this will check the 4806 times similarity between every single movie
similarity
```

Out[108]:
```
array([[1.        , 0.08458258, 0.08718573, ..., 0.04559608, 0.        ,
        0.        ],
       [0.08458258, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
        0.02615329],
       [0.08718573, 0.06063391, 1.        , ..., 0.02451452, 0.        ,
        0.        ],
       ...,
       [0.04559608, 0.02378257, 0.02451452, ..., 1.        , 0.03962144,
        0.04229549],
       [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
        0.08714204],
       [0.        , 0.02615329, 0.        , ..., 0.04229549, 0.08714204,
        1.        ]])
```

In [109]:
```
similarity.shape
```

Out[109]:
```
(4806, 4806)
```

In [110]:
```
similarity[0] #this represents the 1st movie similarity relation with all other 4806 movies including itself which will count 1
# that is why the first one in the array is 1
```

Out[110]:
```
array([1.        , 0.08458258, 0.08718573, ..., 0.04559608, 0.        ,
       0.        ])
```

In [111]:
```
similarity[0].shape
```

Out[111]:
```
(4806,)
```

In [112]:
```
sorted(similarity[0],reverse=True)  # sorted in nearest similar movie descending form
```

```
 0.26401000024165,
 0.25903973506580724,
 0.2537477434955704,
 0.25070610528195014,
 0.2484013136974297,
```

```
In [114]: list(enumerate(similarity[0]))

Out[114]: [(0, 0.9999999999999998),
           (1, 0.08458258116519013),
           (2, 0.08718572905786445),
           (3, 0.074458079104994),
           (4, 0.19184045508446734),
           (5, 0.1098436937909367),
           (6, 0.04047223292581764),
           (7, 0.1487044791289829),
           (8, 0.06003002251876642),
           (9, 0.09802861627917438),
           (10, 0.10397504898200727),
           (11, 0.09592022754223367),
           (12, 0.09004503377814962),
           (13, 0.04559607525875532),
           (14, 0.12996881122750908),
           (15, 0.06367145399670132),
           (16, 0.08000711205939975),
           (17, 0.14165281524036175),
           (18, 0.09620723036050562),

In [121]: sorted(list(enumerate(similarity[0])),reverse=True,key=lambda x:x[1])[1:6]

Out[121]: [(1216, 0.29061909685954823),
           (3730, 0.26401000024165),
           (507, 0.25903973506580724),
           (539, 0.2537477434955704),
           (2409, 0.25070610528195014)]

In [124]: # Creating a function which which will return the top 5 movies which nearest similar to the given data
          def recommend(movie):
              movie_index = new_df[new_df["title"]==movie].index[0]
              distances=similarity[movie_index]
              movies_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]

              for i in movies_list:
          #         print(i[0])
                  print(new_df.iloc[i[0]].title)

In [125]: recommend("Avatar")

          Aliens vs Predator: Requiem
          Falcon Rising
          Independence Day
          Titan A.E.
          Aliens

In [126]: recommend("Batman Begins")

          The Dark Knight
          Batman
          Batman
          The Dark Knight Rises
          10th & Wolf
```

Next step is to convert this predicted data into website

```
Model building completed

In [128]: import pickle

In [129]: pickle.dump(new_df,open("movies.pkl","wb"))#wb stands for write binary mode

In [130]: pickle.dump(new_df.to_dict(),open("movie_dict.pkl","wb"))#wb stands for write binary mode

In [131]: pickle.dump(similarity,open("similarity.pkl","wb"))

In [ ]:
```

Recommendation based on findings

EXPLORER

index.html × | team.html | Release Notes: 1.86.0

cineapp > templates > index.html > html > body#top

```
CINEMATICDJANGO
  .ipynb_checkpoints
  cineapp
    __pycache__
    media
    migrations
    static
      assets
      css
      images
      js
      .gitattributes
      family_movie_id2.pkl
      movie_id.pkl
      movies.pkl
      similarity.pkl
    templates
      404.html
      about.html
      base.html
      contact.html
      index.html
      movie_details.html
      movie_id.csv
      search.html
      team.html
    __init__.py
    admin.py
    apps.py
```

```
1   {% load static %}
2   <!DOCTYPE html>
3   <html lang="en">
4   <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>Cinematic|Home</title>
9     <!-- ----fav icon -----  -->
10    <link rel="apple-touch-icon" sizes="180x180" href="{% static 'images/apple-touch-icon.png' %}">
11    <link rel="icon" type="image/png" sizes="32x32" href="{% static 'images/favicon-32x32.png' %}">
12    <link rel="icon" type="image/png" sizes="16x16" href="{% static 'images/favicon-16x16.png' %}">
13    <link rel="manifest" href="{% static 'images/site.webmanifest' %}">
14    <!-- ------ fav icone code is over  ---  -->
15    <link rel="shortcut icon" href="./favicon.svg" type="image/svg+xml">
16    <!-- custom css link -->
17    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
18    <!-- google font link -->
19    <link rel="preconnect" href="https://fonts.googleapis.com">
20    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
21    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap" rel="stylesheet">
22
23
24  </head>
25
26  <body id="top">
27
28    <!-- #HEADER -->
29  > <header class="header" data-header>···
128   </header>
129
130 > <main>···
```

EXPLORER

index.html × | team.html | Release Notes: 1.86.0

cineapp > templates > index.html > html > body#top

```
CINEMATICDJANGO
  .ipynb_checkpoints
  cineapp
    __pycache__
    media
    migrations
    static
      assets
      css
      images
      js
      .gitattributes
      family_movie_id2.pkl
      movie_id.pkl
      movies.pkl
      similarity.pkl
    templates
      404.html
      about.html
      base.html
      contact.html
      index.html
      movie_details.html
      movie_id.csv
      search.html
      team.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  cineproject
  deepakenv
  db.sqlite3
```

```
24  </head>
25
26  <body id="top">
27
28    <!-- #HEADER -->
29  > <header class="header" data-header>···
128   </header>
129
130 > <main>···
514   </main>
515   <!-- - #FOOTER -->
516 > <footer class="footer">···
616   </footer>
617   <!--- #GO TO TOP-->
618 > <a href="#top" class="go-top" data-go-top>···
620   </a>
621   <!-- - custom js link -->
622
623
624   <script src="{% static 'js/script.js' %}"></script>
625
626   <!-- - ionicon link-->
627   <script type="module" src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.esm.js"></script>
628   <script nomodule src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js"></script>
629
630
631   <!-- movie details page google translator code -->
632   <script type="text/javascript"
633     src="//translate.google.com/translate_a/element.js?cb=googleTranslateElementInit"></script>
634 > <script type="text/javascript">···
638   </script>
639
640
641   <!-- // Get the iframe element -->
642 > <script>···
652   </script>
653
654   <!-- // google spreadsheet code   -->
655 > <script>···
682   </script>
```

File Edit Selection View Go Run Terminal Help
EXPLORER
CINEMATICDJANGO
.ipynb_checkpoints
cineapp
cineproject
deepakenv
db.sqlite3
manage.py
movie-recommender-system.ipynb
requirements.txt
tmdb_5000_credits.csv
tmdb_5000_movies.csv
index.html
manage.py
Release Notes: 1.86.0
manage.py > ...
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
"""Run administrative tasks."""
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "cineproject.settings")
try:
from django.core.management import execute_from_command_line
except ImportError as exc:
raise ImportError(
"Couldn't import Django. Are you sure it's installed and "
"available on your PYTHONPATH environment variable? Did you "
"forget to activate a virtual environment?"
) from exc
execute_from_command_line(sys.argv)
if __name__ == "__main__":
main()

CINEMATICDJANGO
.ipynb_checkpoints
cineapp
cineproject
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
deepakenv
db.sqlite3
manage.py
movie-recommender-system.ipynb
requirements.txt
tmdb_5000_credits.csv
tmdb_5000_movies.csv
cineproject > settings.py > ...
"""
Django settings for cineproject project.
Generated by 'django-admin startproject' using Django 4.2.8.
For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/
For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""
from pathlib import Path
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = "django-insecure-sn&4#e0$2rx*fvrb1hu0sx0a3up2l#+*01h%rx-mlg$+)u_hqr"
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
"django.contrib.admin",
"django.contrib.auth",
"django.contrib.contenttypes",
"django.contrib.sessions",
"django.contrib.messages",
"django.contrib.staticfiles",
"cineapp",
]

EXPLORER

CINEMATICDJANGO

- .ipynb_checkpoints
- cineapp
- cineproject
  - __pycache__
  - __init__.py
  - asgi.py
  - settings.py
  - urls.py
  - wsgi.py
- deepakenv
- db.sqlite3
- manage.py
- movie-recommender-system.ipynb
- requirements.txt
- tmdb_5000_credits.csv
- tmdb_5000_movies.csv

index.html | urls.py | Release Notes: 1.86.0

cineproject > urls.py > ...

```python
"""
URL configuration for cineproject project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
# from django.conf.urls import handler404


urlpatterns = [
    path("admin/", admin.site.urls),
    path("", include('cineapp.urls')),
    # path("404/", views.custom_page_not_found, name="custom_page_not_found"),

]

```

Suggestions for areas of improvement

1. Contextual Recommendations:
   Incorporate contextual information, such as time of day, location, and user mood, into the recommendation process. This can enable the system to provide recommendations that are suitable for specific situations and user states.

2. Personalized Notifications:
   Implement a notification system that proactively informs users about newly released movies, upcoming releases, or movies that align with their preferences. This can encourage users to return to the platform regularly.

3. A/B Testing and Experimentation:
   Conduct A/B testing to evaluate the effectiveness of different recommendation algorithms and user interfaces. Experimentation can provide valuable insights into user behavior and preferences, leading to iterative improvements.

4. Integration with Streaming Platforms:
   Collaborate with streaming platforms to integrate the recommendation system directly into their user interfaces. This would allow users to receive personalized recommendations seamlessly while using their preferred streaming services.

5. Cross-Domain Recommendations:
   Explore the possibility of expanding the recommendation system to recommend content from other domains, such as TV shows, books, music, and more. This would provide users with a holistic entertainment experience.

6. Accessibility and Inclusivity:
   Ensure that the recommendation system is designed to accommodate diverse user preferences, including cultural and accessibility considerations. Implement features that cater to a wide range of users

Scope for future work

The Movies Prediction System project using the Content-Based Filtering Algorithm has laid a strong foundation for enhancing user experience and engagement in the realm of movie consumption. As the digital entertainment landscape continues to evolve, there are several avenues for further development and expansion of the project. The following outlines potential areas for future work:

1. Hybrid Recommendation Approaches:
Integrate collaborative filtering techniques with the existing content-based approach to create a hybrid recommendation system. By combining both content-based and collaborative filtering algorithms, the system can leverage the strengths of both methods, leading to even more accurate and diverse recommendations.

2. Real-time Interaction and Feedback:
Enhance the recommendation system's accuracy by incorporating real-time user interactions and feedback. Allow users to rate and review recommended movies, and use this information to continuously update user profiles and improve recommendation accuracy over time.

3. Integration of External Data Sources:
Expand the dataset by incorporating external data sources such as user reviews, social media activity, and movie ratings from other platforms. This additional data can provide a more comprehensive view of users' preferences and improve the system's ability to make accurate recommendations.

4. Natural Language Processing (NLP) Techniques:
Implement advanced NLP techniques to extract even more detailed features from movie descriptions, such as sentiment analysis, thematic analysis, and emotion detection. This could lead to more nuanced understanding of movie content and enhance recommendation precision.

5. Dynamic User Profiles:
Develop user profiles that adapt to changing preferences over time. Implement algorithms that capture shifts in user interests and update profiles accordingly, ensuring that recommendations remain relevant as users' tastes evolve.

6. Multi-modal Content:
Extend the recommendation system to support multi-modal content, such as images and trailers, in addition to text-based attributes. This can provide a richer understanding of movies and lead to more diverse and engaging recommendations

Conclusion

In conclusion, the development and implementation of the Movies Prediction System usingthe Content-Based Filtering Algorithm have yielded promising results and significant insights into personalized content recommendation. The primary goal of this project was to enhance user experience by providing tailored movie suggestions based on users' preferences and movie features. Through the project's execution, we have achieved several important milestones and observed the system's impact on user engagement and satisfaction.

The Content-Based Filtering Algorithm demonstrated its efficacy in generating relevant and accurate movie recommendations by leveraging movie attributes such as genre, cast, director, and synopsis. The project's methodology involved data collection, preprocessing, feature extraction, user profile creation, similarity computation, and recommendation generation. These steps collectively contribute to the system's ability to understand user preferences and match them with movies that align with their tastes.

Throughout the project, we thoroughly evaluated the system's performance using accuracy metrics such as precision, recall, and F1-score. The results of these evaluations highlighted the algorithm's success in recommending movies that resonate with users' interests. Users' feedback further confirmed the system's effectiveness in consistently suggesting movies that aligned with their preferences, resulting in increased user satisfaction and engagement.

One of the project's key strengths is its potential for scalability and future enhancements. As the system accumulates more user interactions and preferences, its recommendations will become even more accurate and personalized. Additionally, the project could be expanded to incorporate hybrid recommendation approaches that combine content-based and collaborative filtering methods, further enhancing recommendation accuracy.

In conclusion, the Movies Prediction System using the Content-Based Filtering Algorithmhas successfully demonstrated its ability to transform the way users discover and engage with movies. By focusing on users' individual preferences and movie attributes, the system offers a personalized and enriching movie-watching experience. This project not only contributes to the field of recommendation systems but also underscores the significance of data-driven approaches in enhancing user satisfaction and engagement in the ever-expanding digital entertainment landscape.

BIBLIOGRAPHY

- 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. https://ieeexplore.ieee.org/document/1430572

- 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. https://link.springer.com/chapter/10.1007/978-0-387-85820-3_3

- Melville, P., & Sindhwani, V. (2010). Recommender systems. In Encyclopedia of Machine Learning (pp. 829-838). Springer. https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_710

- Chen, L., Ou, J., Hu, X., & Chen, J. (2019). A survey of content-based recommendation algorithms on movies. Journal of Physics: Conference Series, 1349(1), 012020. https://iopscience.iop.org/article/10.1088/1742-6596/1349/1/012020/meta

- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web (WWW'01) (pp. 285-295). https://dl.acm.org/doi/10.1145/371920.372071

Appendices

Appendix A: Dataset Description

The dataset used for this project contains the following attributes for each movie:

Movie ID: A unique identifier for each movie.

Title: The title of the movie.

Genres: A list of genres associated with the movie (e.g., Action, Comedy, Drama).

Release Year: The year the movie was released.

Plot Summary: A brief summary of the movie's plot.

Cast: List of main actors/actresses in the movie.

Director: The director of the movie.

Keywords: Keywords associated with the movie's content.

Average Rating: The average user rating for the movie.

Popularity Score: A measure of the movie's popularity.

Runtime: The duration of the movie in minutes.

Appendix B: Algorithm Pseudocode

Algorithm: Content-Based Movies Prediction System

Input: User preferences and historical interactions
Output: Movie recommendations

1. Extract features from the movie dataset (title, genres, plot summary, etc.).
2. Create user profile based on user preferences and historical interactions.
3. Normalize user profile vector.
4. For each movie in the dataset:
   Calculate similarity score between user profile vector and movie's feature vector.
   Store movie's ID and similarity score.

5. Sort movies by similarity scores in descending order.
6. Filter out movies already interacted with by the user.
7. Generate a list of recommended movies.
8. Return recommended movies.

Appendix C: Sample Recommendations

Based on the user's preferences and interactions, here are some movie recommendations:

Movie Title 1

- Genres: Action, Adventure
- Release Year: 20XX
- Plot Summary: [Brief summary of the plot]
- Cast: [Main actors/actresses]
- Director: [Director's name]

Movie Title 2

- Genres: Comedy, Romance
- Release Year: 20XX
- Plot Summary: [Brief summary of the plot]
- Cast: [Main actors/actresses]
- Director: [Director's name]

Movie Title 3

- Genres: Drama
- Release Year: 20XX
- Plot Summary: [Brief summary of the plot]
- Cast: [Main actors/actresses]
- Director: [Director's name]

ANNEXURE (if any)

Annexure : Questionnaire

The following questionnaire was used to collect data from users about their movie preferences.

1. What are your favorite movies?
2. What genres of movies do you like?
3. What actors and directors do you like?
4. What are some of the things you look for in a movie?
5. How often do you watch movies?
6. How do you usually find movies to watch?
7. Would you be interested in using a Movies Prediction system?

8. What are your favorite movies?
9. What genres of movies do you like?
10. What actors and directors do you like?
11. What are some of the things you look for in a movie?
12. How often do you watch movies?
13. How do you usually find movies to watch?
14. Would you be interested in using a Movies Prediction system?

References

HTML Documentation: https://developer.mozilla.org/en-US/docs/Web/HTML

CSS Documentation: https://developer.mozilla.org/en-US/docs/Web/CSS

JavaScript Documentation: https://developer.mozilla.org/en-US/docs/Web/JavaScript

jQuery Documentation: https://api.jquery.com/

Bootstrap Documentation: https://getbootstrap.com/docs/5.1/getting-started/introduction/

Python Documentation: https://docs.python.org/3/

Django Documentation: https://docs.djangoproject.com/en/3.2/

SQLite Documentation: https://www.sqlite.org/docs.html

scikit-learn Documentation: https://scikit-learn.org/stable/user_guide.html

NumPy Documentation: https://numpy.org/doc/stable/

Pandas Documentation: https://pandas.pydata.org/docs/

Pickle Documentation: https://docs.python.org/3/library/pickle.html

Jupyter Notebook Documentation: https://jupyter-notebook.readthedocs.io/en/stable/