Movie Recommendation System

By- Trisha Roy (12101718)

CONTENTS

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

It has been great honour and privilege to undergo training at **Fifth Force.**

I am highly indebted to **Prof. Arnab Chakraborty** for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. His constant guidance and willingness to share his vast knowledge made us understand this project and its manifestations in great depths helped us to complete the assigned tasks on time.

I would like to express my gratitude towards my parents and members of alpha net technologies pvt.ltd. for their kind cooperation and encouragement which help me in completion of this project.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities.

Project Objective

Problem Statement:

Developing a movie recommendation system that utilizes a filtering technique to provide personalized movie recommendations to users based on their preferences. The system should take into account movie attributes such as genre, actors, director and many more to generate accurate and relevant recommendations. The goal is to create a user-friendly and effective recommendation system that enhances the movie-watching experience for users by suggesting movies that align with their interests and preferences.

Project Objective:

The objective of this project is to develop a movie recommendation system that leverages content-based filtering techniques to provide personalized and accurate movie recommendations to users. The project aims to enhance the movie-watching experience by delivering relevant and engaging movie suggestions to users, thereby increasing user satisfaction and engagement with the movie platform.

Methodology for solving the problem statement:

- **Data Acquisition:** This process typically involves the following steps:
  - **Download the Dataset:** Start by downloading the TMDB dataset from the Kaggle platform.
  - **Understand the Dataset:** Familiarizing myself with the structure and content of the dataset. Explore the provided files and documentation to gain insights into the available information about movies, such as titles, overview, genres, ratings, and user reviews.
  - **Load the Dataset:** Load the dataset into my programming environment.

- **Preprocessing:** Performs necessary preprocessing steps to clean and prepare the dataset.
    - **Data Preprocessing**: This may include handling missing values, removing duplicates, normalizing data, and converting data types if needed.
    - **Text Preprocessing:** This may include tokenization, lowercasing, stop word removal and stemming.
- **Feature Engineering:** This is the process in which we are transforming raw data into meaningful features that can be used as input for machine learning algorithms. We are going to use one common feature engineering technique used in content-based recommendation systems is the Bag of Words (BoW) approach.
- **Model Building:** We have to create our own model based on content-based filtering system. And then apply it on the prepared dataset.
- **Forming a web dashboard to show the implementation of our model more specifically using posters of movie.**

Project Scope

This report focuses on the development and evaluation of a content-based movie recommendation system. The scope of the project includes the following key aspects:

- **Dataset:** The recommendation system utilizes the TMDB dataset from Kaggle, consisting of movie attributes such as titles, descriptions, genres, and ratings. The report assumes that the dataset is complete and properly preprocessed.

- **Feature Engineering:** The recommendation system employs feature engineering techniques such as Bag of Words to extract meaningful features from movie descriptions and genres. The report covers the implementation and utilization of these features for recommendation generation.

- **Algorithm Selection:** The project focuses on implementing content-based filtering algorithms to generate movie recommendations. Collaborative filtering or hybrid approaches are beyond the scope of this report.

- **Model Building and Evaluation:** The report covers the process of model building, including data preprocessing, feature engineering, and training the recommendation model. Evaluation metrics such as precision, recall, and mean average precision are utilized to assess the model's performance.

- **User Interface:** The project scope includes a basic command-line interface for users to input movie titles and receive recommendations. Extensive development of a full-fledged graphical user interface (GUI) or integration into a web application is beyond the scope of this report.

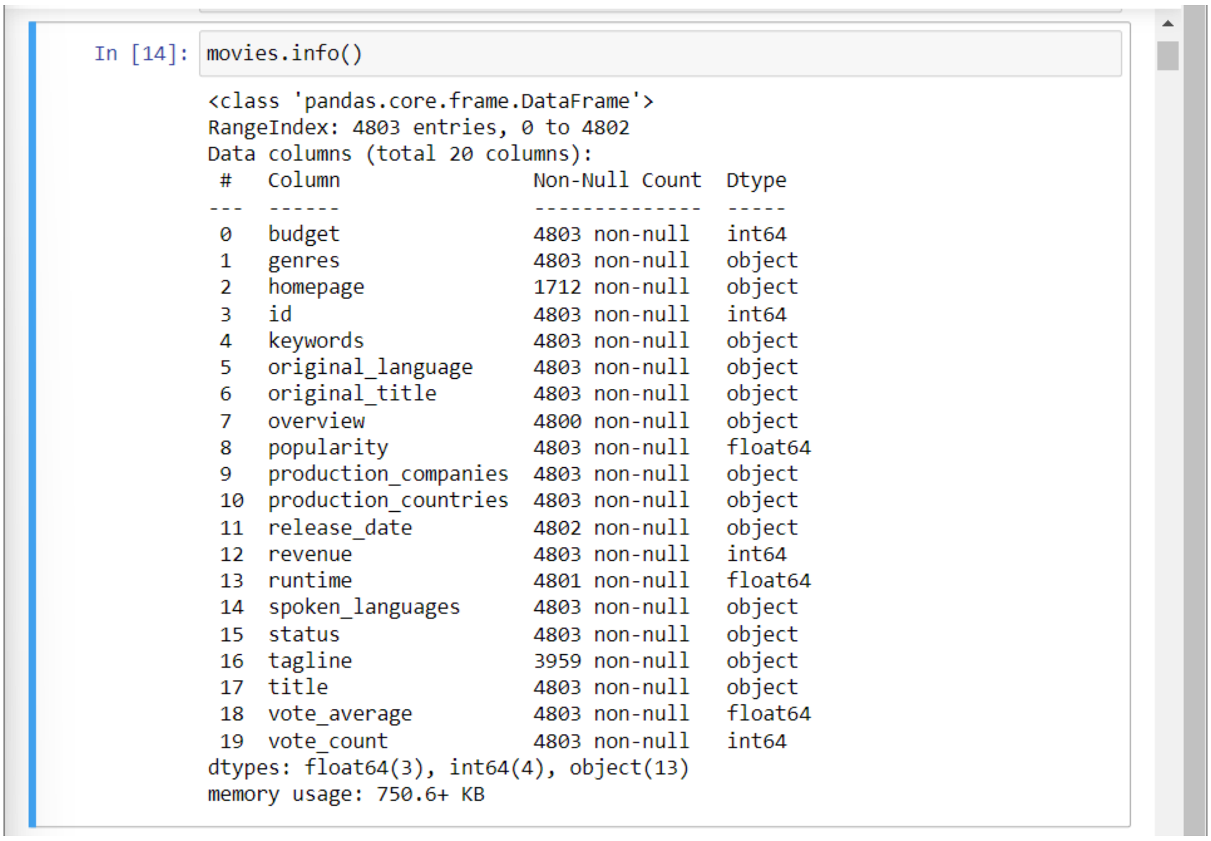
- **Limitations:** The report acknowledges certain limitations, such as the reliance on textual movie attributes, the absence of user feedback incorporation, and the exclusion of collaborative filtering techniques.

- **Performance Optimization:** While basic model performance optimization techniques like hyperparameter tuning are explored, in-depth optimization approaches, distributed computing, or scalability aspects are outside the scope of this report.

Data Description

Source of Data: We have downloaded the datasets from Kaggle platform. We have two datasets with us i.e movies.csv and credits.csv.

Movies.csv



```
In [14]: movies.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

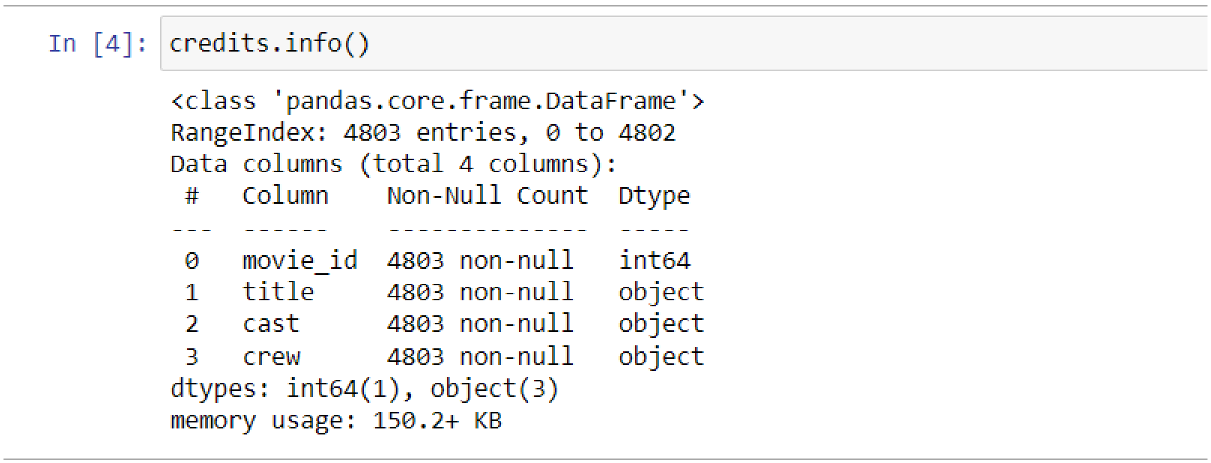
The provided dataset is a pandas DataFrame containing information about movies. Here is a description of the columns in the dataset:

- **budget:** The budget allocated for producing the movie (numeric).
- **genres:** The genre(s) to which the movie belongs (textual, object).
- **homepage:** The URL or website associated with the movie (textual, object).
- **id:** A unique identifier for each movie (numeric).

- **keywords:** Descriptive keywords or phrases associated with the movie (textual, object).
- **original_language:** The original language in which the movie was produced (textual, object).
- **original_title:** The original title of the movie (textual, object).
- **overview:** A brief description or summary of the movie's plot or storyline (textual, object).
- **popularity:** A measure of the movie's popularity (numeric, float).
- **production_companies:** The production companies responsible for producing the movie (textual, object).
- **production_countries:** The countries where the movie was produced (textual, object).
- **release_date:** The date when the movie was released (textual, object).
- **revenue:** The revenue generated by the movie (numeric).
- **runtime:** The duration or length of the movie in minutes (numeric, float).
- **spoken_languages:** The languages spoken in the movie (textual, object).
- **status:** The status of the movie (textual, object).
- **tagline:** A catchy phrase or slogan associated with the movie (textual, object).
- **title:** The title of the movie (textual, object).
- **vote_average:** The average rating or score given to the movie by users (numeric, float).
- **vote_count:** The total number of votes received by the movie (numeric).

The dataset contains 4,803 entries or rows. Some columns have missing values, such as "homepage," "overview," "release_date," and "tagline." The dataset provides a combination of numeric, textual, and categorical data types.

Credits.csv



```
In [4]: credits.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   movie_id  4803 non-null   int64
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

The provided dataset is a pandas DataFrame containing information about movies. Here is a description of the columns in the dataset:

- **movie_id:** A unique identifier for each movie (numeric, int64).
- **title:** The title of the movie (textual, object).
- **cast:** The main actors or actresses who appear in the movie (textual, object).
- **crew:** The main individuals involved in the production of the movie, such as directors, producers, and writers (textual, object).

The dataset consists of 4 columns and 4,803 entries or rows. The "movie_id" column contains unique identifiers for each movie. The "title" column represents the title of the movie. The "cast" column includes the names of the main actors or actresses associated with each movie. The "crew" column provides information about the main individuals involved in the movie's production.

Preprocessing

Data preprocessing is an essential step in preparing the dataset for analysis or building a movie recommendation system. Here are data preprocessing steps that we have applied :

- Combining Datasets: Our first step should be merging both the datasets on the basis of title column.
- Choosing necessary columns: Next step should be filtering the unnecessary columns which are not adding value to our recommending system. I have removed columns like budget, homepage, original_language , popularity and many more. As we are working on content based recommendation system we need to focus on the important details of movie which makes it interesting for us to watch.
    - The columns which we considered for further use are:
        1. Genre
        2. Movie_id
        3. Title
        4. Overview
        5. Keywords
        6. Cast
        7. Crew
- Handling Missing Values: Identify columns with missing values and decide on an appropriate strategy for handling them. Options include removing rows with missing values, filling in missing values with the mean or median, or using more advanced techniques such as imputation.
- Removing Duplicates: Check for and remove any duplicate entries in the dataset, ensuring that each movie record is unique.

Text preprocessing refers to the series of steps taken to clean and transform raw text data into format that is more suitable for analysis or natural language processing tasks. It involves various techniques to standardize and normalize text, remove unnecessary elements, and extract meaningful information. Text preprocessing plays a crucial role in improving the quality and accuracy of text-based models and algorithms. Here are some text preprocessing steps that we have applied :

- **Lowercasing:** Converting all text to lowercase. This helps ensure consistent comparisons and reduces the vocabulary size by treating words with different cases as the same.
- **Tokenization:** Splitting text into individual tokens or words. Tokenization breaks down sentences or paragraphs into smaller units, making it easier to process and analyze text.
- **Stopword Removal:** Removing commonly used words that do not carry significant meaning, such as "a," "the," "is." Stopword removal reduces noise in the text data and helps focus on more meaningful words.
- **Lemmatization and Stemming:** Reducing words to their base or root form. Lemmatization and stemming help normalize words, such as converting "running" and "runs" to their base form "run." This reduces vocabulary size and ensures similar words are treated as the same.

Feature Engineering

Feature engineering is the process of transforming raw data into meaningful features that can be used as input for machine learning algorithms. It involves selecting, creating, and transforming features to improve the performance of a machine learning model.

In the context of a content-based movie recommendation system, feature engineering aims to extract useful information from movie-related data (such as titles, descriptions, genres, etc.) and represent it in a numerical format that can be utilized by the recommendation algorithm.

We have used one of the common feature engineering technique i.e Bag of Words(BoW) approach. The Bag of Words technique represents text documents as numerical vectors, disregarding the order and structure of the words in the document. It focuses solely on the presence or absence of words and their frequencies.

The steps involved in Bag Of Words Technique is:

1. **Tokenization:** The first step is to break down the movie description or text into individual words or tokens. This process involves splitting the text based on spaces or punctuation marks.
2. **Vocabulary Creation:** A vocabulary is constructed by collecting all unique words/tokens from the movie descriptions. Each unique word becomes a feature in the vocabulary.
3. **Vector Representation:** For each movie description, a vector is created where each element represents the count or presence of a word from the vocabulary. The value in each element can be the raw count (the number of times a word appears in the description) or a binary indicator (1 if the word is present, 0 otherwise).
4. **Feature Scaling:** Optionally, feature scaling techniques (e.g., normalization or TF-IDF) can be applied to adjust the importance of each word in the vector representation. This helps to address

issues related to the varying lengths of movie descriptions and the relative importance of words.

5. **Similarity Calculation:** Once the movie descriptions are represented as numerical vectors, similarity measures (e.g., cosine similarity) can be used to compare the vectors and identify movies with similar descriptions. Movies with higher similarity scores can be recommended to users who have shown interest in similar movies.

The Bag of Words technique provides a way to represent textual information in a numerical format suitable for machine learning algorithms. It allows the content-based recommendation system to find similarities between movies based on the words present in their descriptions, enabling personalized recommendations for users with similar movie preferences.

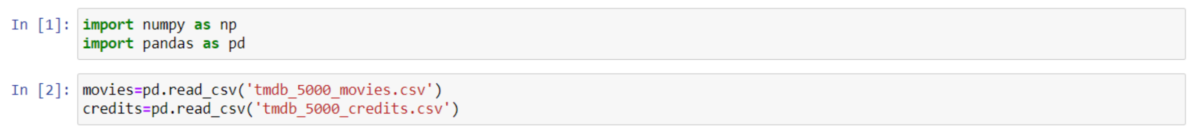
Model Used

Here's a description of the features of my model:

- **Movie Similarity:** My model calculates the similarity between movies based on their descriptions. It uses a similarity matrix called similarity, which likely represents the pairwise similarity scores between movies.
- **Input:** The recommend() function takes a movie title as input. This allows users to specify a movie for which they want to receive recommendations.
- **Movie Index:** My model uses the input movie title to find the corresponding movie index in the dataset (new_df) using the line movie_index = new_df[new_df['title']== movie].index[0].
- **Distance Calculation:** The model then retrieves the similarity scores of the input movie with all other movies from the similarity matrix using the line distances = similarity[movie_index].
- **Top Similar Movies:** The model identifies the top similar movies to the input movie based on the similarity scores. It sorts the movies in descending order of similarity, excluding the input movie itself, and selects the top 5 movies as recommendations using the line movies_list = sorted(list(enumerate(distances)),reverse=True, key = lambda x:x[1])[1:6].
- **Recommendation Display:** The model prints the titles of the recommended movies using a loop, iterating over the movies_list, and accessing the corresponding movie titles from the dataset (new_df) using the line print(new_df.iloc[i[0]].title).

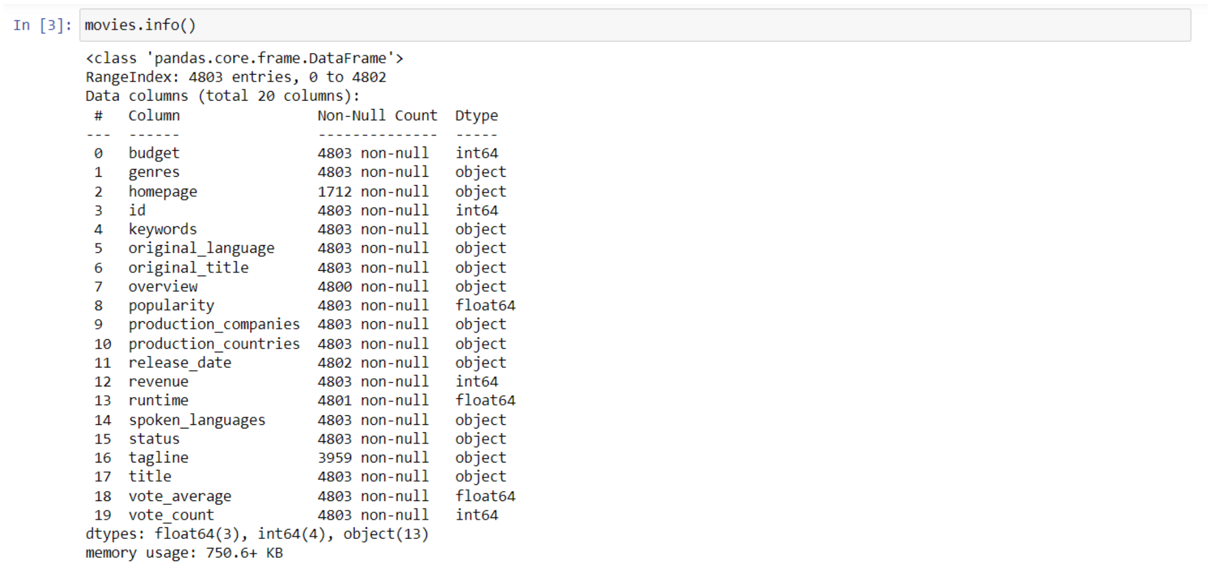
Overall, my model utilizes a content-based recommendation approach that relies on the similarity of movie descriptions to make recommendations. By comparing the input movie's description with the descriptions of other movies, it identifies the most similar movies and presents them as recommendations to users.

Code



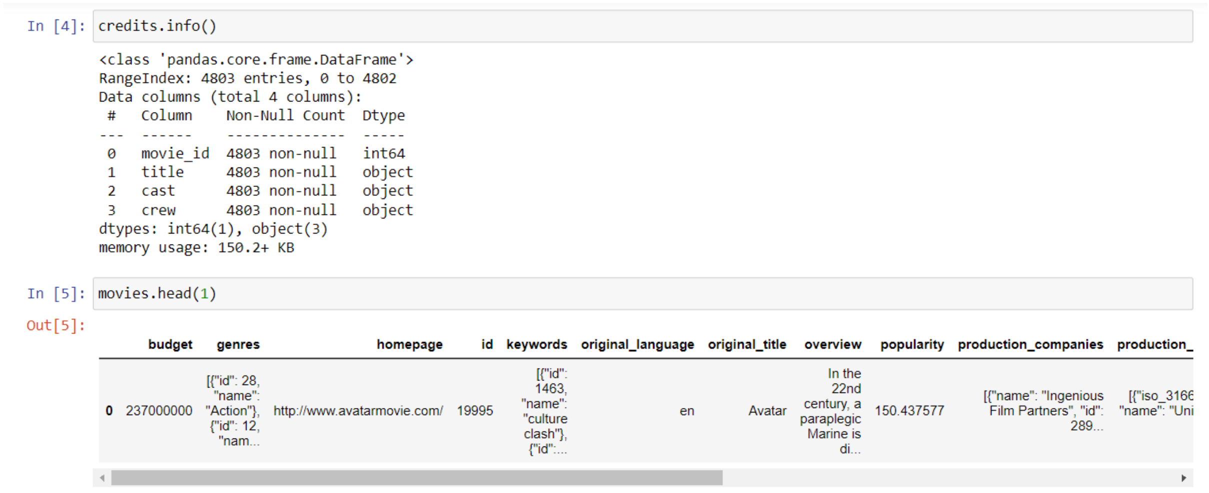
```
In [1]: import numpy as np
        import pandas as pd
```

```
In [2]: movies=pd.read_csv('tmdb_5000_movies.csv')
        credits=pd.read_csv('tmdb_5000_credits.csv')
```



```
In [3]: movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   budget                4803 non-null   int64  
 1   genres                4803 non-null   object 
 2   homepage              1712 non-null   object 
 3   id                    4803 non-null   int64  
 4   keywords              4803 non-null   object 
 5   original_language     4803 non-null   object 
 6   original_title        4803 non-null   object 
 7   overview              4800 non-null   object 
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object 
 10  production_countries  4803 non-null   object 
 11  release_date          4802 non-null   object 
 12  revenue               4803 non-null   int64  
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object 
 15  status                4803 non-null   object 
 16  tagline               3959 non-null   object 
 17  title                 4803 non-null   object 
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64  
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```



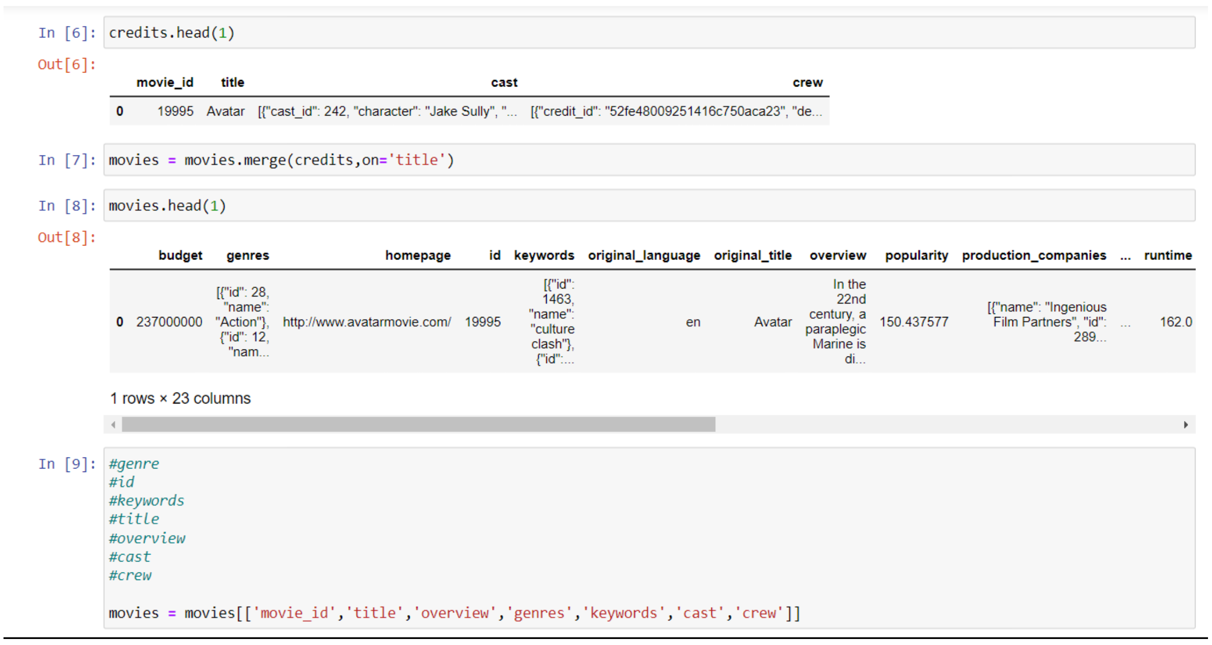
```
In [4]: credits.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   movie_id  4803 non-null   int64 
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

```
In [5]: movies.head(1)
```

Out[5]:

	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 [6]:
```
credits.head(1)
```
Out[6]:

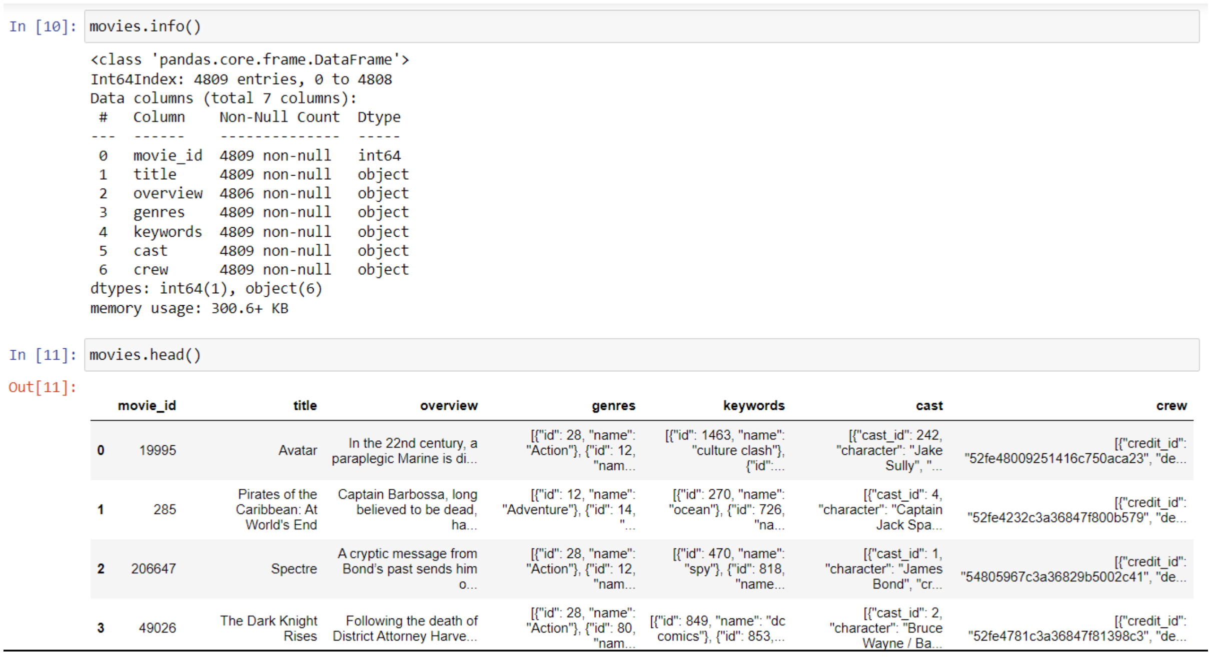
	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

In [7]:
```
movies = movies.merge(credits,on='title')
```
In [8]:
```
movies.head(1)
```
Out[8]:

	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

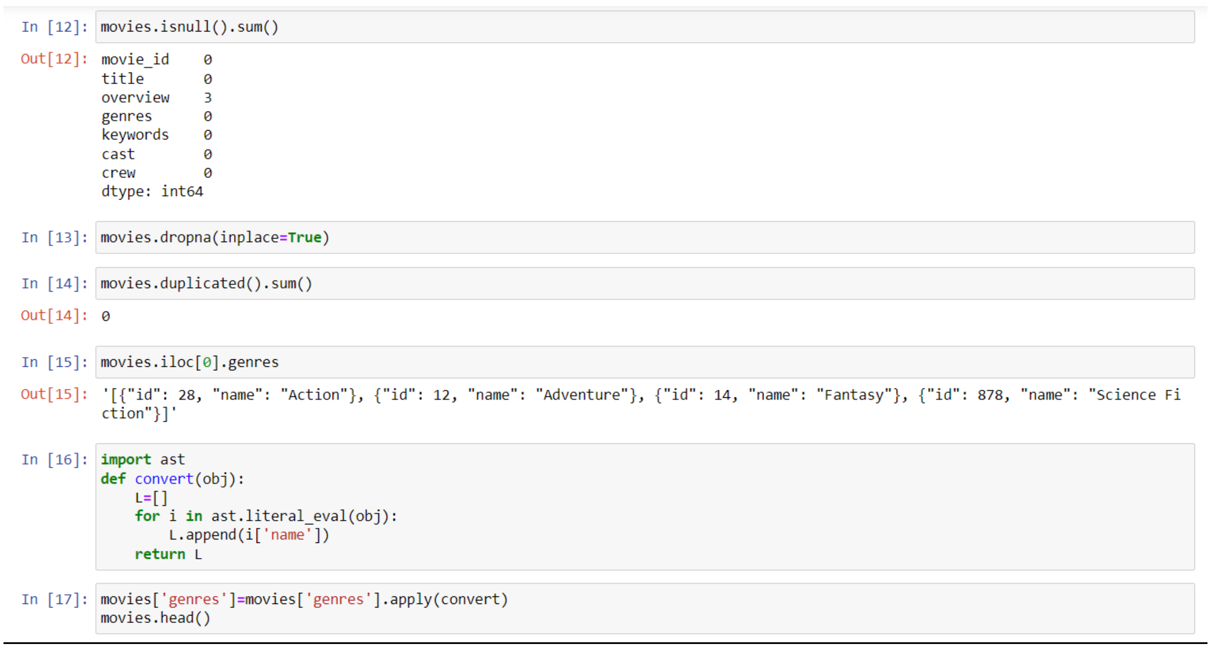
1 rows × 23 columns

In [9]:
```
#genre
#id
#keywords
#title
#overview
#cast
#crew

movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```

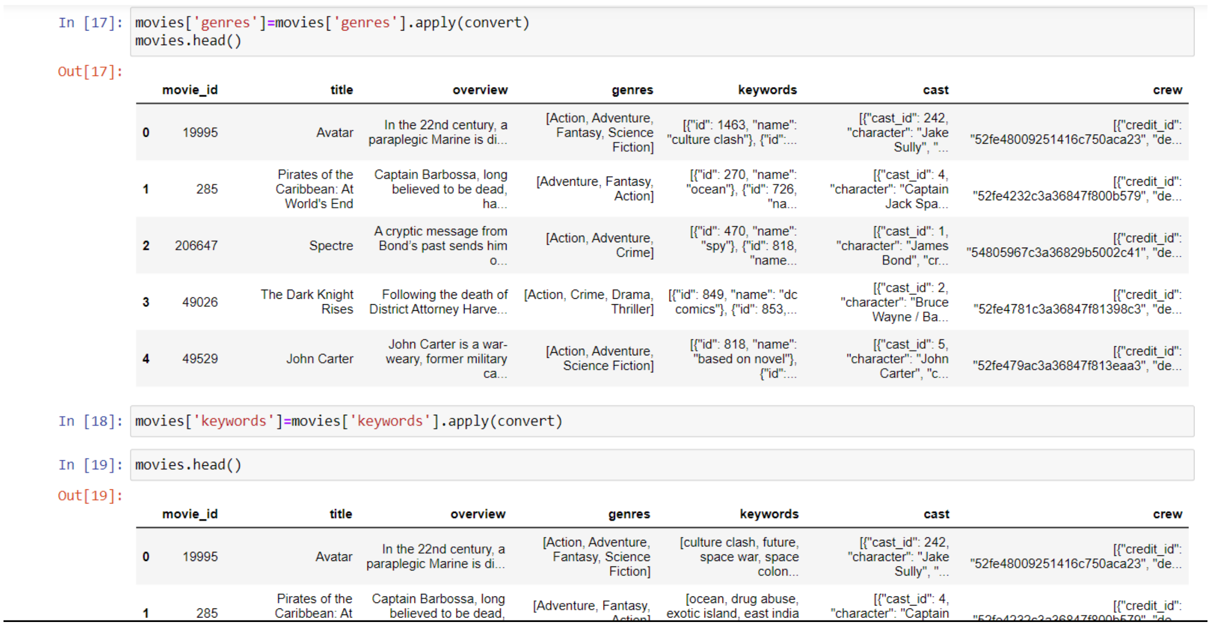


In [10]:
```
movies.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4809 entries, 0 to 4808
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   movie_id  4809 non-null   int64
 1   title     4809 non-null   object
 2   overview  4806 non-null   object
 3   genres    4809 non-null   object
 4   keywords  4809 non-null   object
 5   cast      4809 non-null   object
 6   crew      4809 non-null   object
dtypes: int64(1), object(6)
memory usage: 300.6+ KB
```
In [11]:
```
movies.head()
```
Out[11]:

	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...



```
In [12]: movies.isnull().sum()

Out[12]: movie_id    0
         title       0
         overview    3
         genres      0
         keywords    0
         cast        0
         crew        0
         dtype: int64

In [13]: movies.dropna(inplace=True)

In [14]: movies.duplicated().sum()

Out[14]: 0

In [15]: movies.iloc[0].genres

Out[15]: '[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fi
         ction"}]'

In [16]: import ast
         def convert(obj):
             L=[]
             for i in ast.literal_eval(obj):
                 L.append(i['name'])
             return L

In [17]: movies['genres']=movies['genres'].apply(convert)
         movies.head()
```



```
In [17]: movies['genres']=movies['genres'].apply(convert)
         movies.head()

Out[17]:
```

	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...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"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...	[Action, Adventure, Crime]	[{"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...	[Action, Crime, Drama, Thriller]	[{"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...	[Action, Adventure, Science Fiction]	[{"id": 818, "name": "based on novel"}, {"id":...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
In [18]: movies['keywords']=movies['keywords'].apply(convert)

In [19]: movies.head()

Out[19]:
```

	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...
1	285	Pirates of the Caribbean: At	Captain Barbossa, long believed to be dead,	[Adventure, Fantasy,	[ocean, drug abuse, exotic island, east india	[{"cast_id": 4, "character": "Captain	[{"credit_id":

```
In [20]: def convert3(obj):
             L=[]
             counter=0
             for i in ast.literal_eval(obj):
                 if counter!=3:
                     L.append(i['name'])
                     counter+=1
                 else:
                     break
             return L
```

```
In [21]: movies['cast']=movies['cast'].apply(convert3)
```

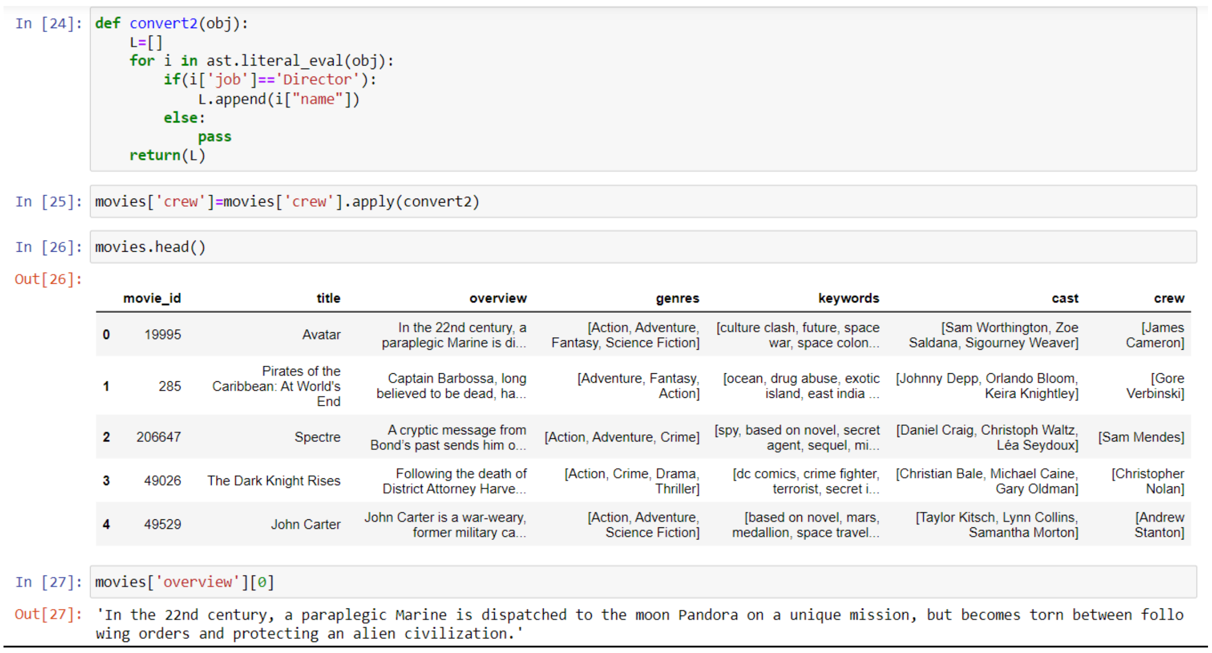
```
In [22]: movies.head()
```

Out[22]:

	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...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
In [23]: movies['crew'][0]
```

```
Out[23]: '[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor", "name": "Stephen
E. Rivkin"}, {"credit_id": "539c47ecc3a36810e3001f87", "department": "Art", "gender": 2, "id": 496, "job": "Production Design",
"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": 90
0, "job": "Supervising Sound Editor", "name": "Christopher Boyes"}, {"credit_id": "539c4a4cc3a36810c9002101", "department": "Pr
oduction", "gender": 1, "id": 1262, "job": "Casting", "name": "Mali Finn"}, {"credit_id": "5544ee3b925141499f0008fc", "departme
nt": "Sound", "gender": 2, "id": 1729, "job": "Original Music Composer", "name": "James Horner"}, {"credit_id": "52fe4800925141
6c750ac9c3", "department": "Directing", "gender": 2, "id": 2710, "job": "Director", "name": "James Cameron"}, {"credit_id": "52
fe48009251416c750ac9d9", "department": "Writing", "gender": 2, "id": 2710, "job": "Writer", "name": "James Cameron"}, {"credit_
id": "52fe48009251416c750aca17", "department": "Editing", "gender": 2, "id": 2710, "job": "Editor", "name": "James Cameron"},
{"credit_id": "52fe48009251416c750aca29", "department": "Production", "gender": 2, "id": 2710, "job": "Producer", "name": "Jame
s Cameron"}, {"credit_id": "52fe48009251416c750aca3f", "department": "Writing", "gender": 2, "id": 2710, "job": "Screenplay",
"name": "James Cameron"}, {"credit_id": "539c4987c3a36810ba0021a4", "department": "Art", "gender": 2, "id": 7236, "job": "Art D
irection", "name": "Andrew Menzies"}, {"credit_id": "549598c3c3a3686ae9004383", "department": "Visual Effects", "gender": 0, "i
d": 6690, "job": "Visual Effects Producer", "name": "Jill Brooks"}, {"credit_id": "52fe48009251416c750aca4b", "department": "Pr
oduction", "gender": 1, "id": 6347, "job": "Casting", "name": "Margery Simkin"}, {"credit_id": "570b6f419251417da70032fe", "dep
artment": "Art", "gender": 2, "id": 6878, "job": "Supervising Art Director", "name": "Kevin Ishioka"}, {"credit_id": "5495a0fac
3a3686ae9004468", "department": "Sound", "gender": 0, "id": 6883, "job": "Music Editor", "name": "Dick Bernstein"}, {"credit_i
d": "54959706c3a3686af3003e81", "department": "Sound", "gender": 0, "id": 8159, "job": "Sound Effects Editor", "name": "Shannon
Mills"}, {"credit_id": "54491d58c3a3680fb1001ccb", "department": "Sound", "gender": 0, "id": 8160, "job": "Foley", "name": "Den
nie Thorpe"}, {"credit_id": "54491d6cc3a3680fa5001b2c", "department": "Sound", "gender": 0, "id": 8163, "job": "Foley", "name":
"Jana Vance"}, {"credit_id": "52fe48009251416c750aca57", "department": "Costume & Make-Up", "gender": 1, "id": 8527, "job": "Co
stume Design", "name": "Deborah Lynn Scott"}, {"credit_id": "52fe48009251416c750aca2f", "department": "Production", "gender":
2, "id": 8529, "job": "Producer", "name": "Jon Landau"}, {"credit_id": "539c4937c3a36810ba002194", "department": "Art", "gende
r": 0, "id": 9618, "job": "Art Direction", "name": "Sean Haworth"}, {"credit_id": "539c49b6c3a36810c10020e6", "department": "Ar
t", "gender": 1, "id": 12653, "job": "Set Decoration", "name": "Kim Sinclair"}, {"credit_id": "570b6f2f9251413a0e00020d", "depa
rtment": "Art", "gender": 1, "id": 12653, "job": "Supervising Art Director", "name": "Kim Sinclair"}, {"credit_id": "54491a6c0e
0a26748c001b19", "department": "Art", "gender": 2, "id": 14350, "job": "Set Designer", "name": "Richard F. Mays"}, {"credit_i
d": "56928cf4c3a3684cff0025c4", "department": "Production", "gender": 1, "id": 20294, "job": "Executive Producer", "name": "Lae
ta Kalogridis"}, {"credit_id": "52fe48009251416c750aca51", "department": "Costume & Make-Up", "gender": 0, "id": 17675, "job":
"Costume Design", "name": "Mayes C. Rubeo"}, {"credit_id": "52fe48009251416c750aca11", "department": "Camera", "gender": 2, "i
```



```
In [24]: def convert2(obj):
             L=[]
             for i in ast.literal_eval(obj):
                 if(i['job']=='Director'):
                     L.append(i["name"])
                 else:
                     pass
             return(L)
```

```
In [25]: movies['crew']=movies['crew'].apply(convert2)
```

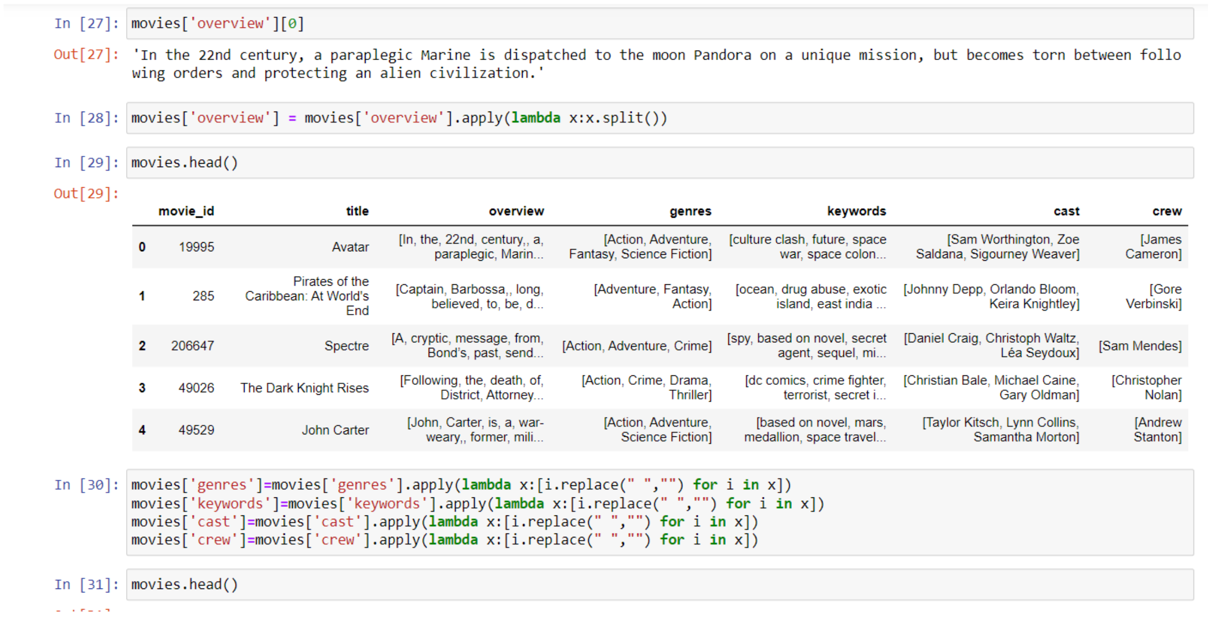
```
In [26]: movies.head()
```

Out[26]:

	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]	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

```
In [27]: movies['overview'][0]
```

```
Out[27]: '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.'
```



```
In [27]: movies['overview'][0]
```

```
Out[27]: '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.'
```

```
In [28]: movies['overview'] = movies['overview'].apply(lambda x:x.split())
```

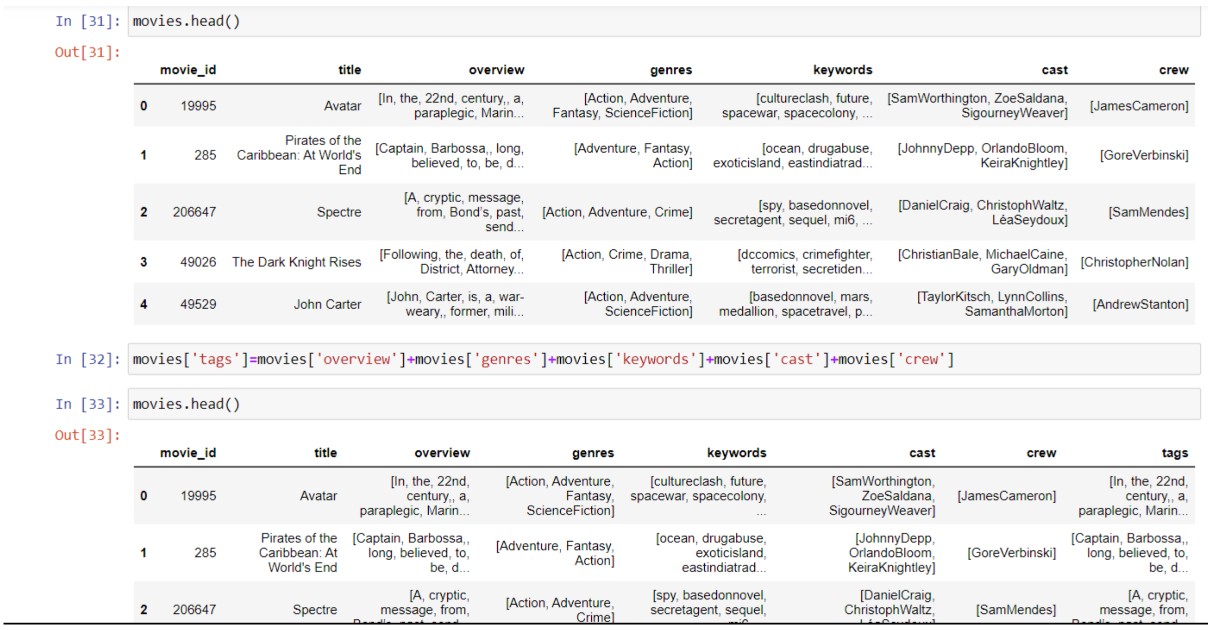
```
In [29]: movies.head()
```

Out[29]:

	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]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

```
In [30]: movies['genres']=movies['genres'].apply(lambda x:[i.replace(" ","") for i in x])
         movies['keywords']=movies['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
         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])
```

```
In [31]: movies.head()
```



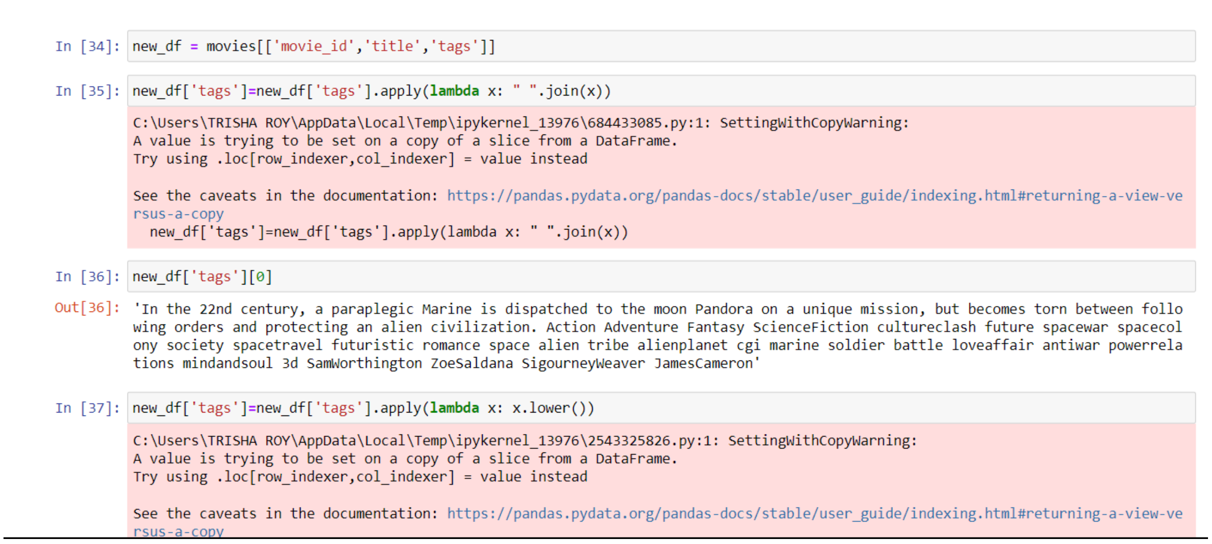
```
In [31]: movies.head()
Out[31]:
```

	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]

```
In [32]: movies['tags']=movies['overview']+movies['genres']+movies['keywords']+movies['cast']+movies['crew']
```

```
In [33]: movies.head()
Out[33]:
```

	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...
2	206647	Spectre	[A, cryptic, message, from,	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel,	[DanielCraig, ChristophWaltz,	[SamMendes]	[A, cryptic, message, from,

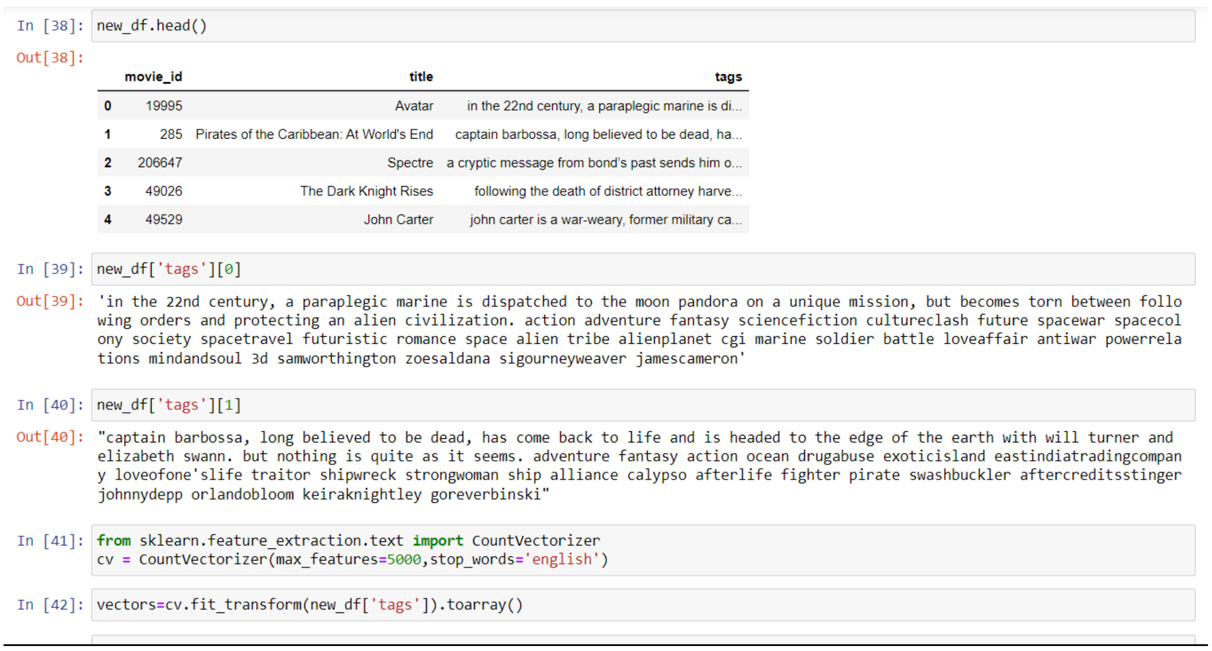


```
In [34]: new_df = movies[['movie_id','title','tags']]
```

```
In [35]: new_df['tags']=new_df['tags'].apply(lambda x: " ".join(x))

C:\Users\TRISHA ROY\AppData\Local\Temp\ipykernel_13976\684433085.py:1: 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-ve
rsus-a-copy
  new_df['tags']=new_df['tags'].apply(lambda x: " ".join(x))
```

```
In [36]: new_df['tags'][0]

Out[36]: '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 [37]: new_df['tags']=new_df['tags'].apply(lambda x: x.lower())

C:\Users\TRISHA ROY\AppData\Local\Temp\ipykernel_13976\2543325826.py:1: 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-ve
rsus-a-copy
```



```
In [38]: new_df.head()
Out[38]:
```

	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 [39]: new_df['tags'][0]

Out[39]: '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 [40]: new_df['tags'][1]

Out[40]: "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 [41]: from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer(max_features=5000,stop_words='english')
```

```
In [42]: vectors=cv.fit_transform(new_df['tags']).toarray()
In [43]: vectors
```

```
Out[43]: 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 [44]: for i in cv.get_feature_names_out():
             print(i)
```

```
bring
bringing
brings
brink
britain
british
britishsecretservice
brittanymurphy
broadway
broke
broken
broker
brooklyn
brooks
brothel
brother
brotherbrotherrelationship
brothers
```

```
In [45]: import nltk
```

```
In [46]: from nltk.stem.porter import PorterStemmer
         ps= PorterStemmer()
```

```
In [47]: def stem(text):
             y=[]
             for i in text.split():
                 y.append(ps.stem(i))
             return " ".join(y)
```

```
In [48]: new_df['tags'] = new_df['tags'].apply(stem)
```

```
C:\Users\TRISHA ROY\AppData\Local\Temp\ipykernel_13976\3213734980.py:1: 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-ve
rsus-a-copy
  new_df['tags'] = new_df['tags'].apply(stem)
```

```
In [49]: vectors=cv.fit_transform(new_df['tags']).toarray()
         vectors
```

```
Out[49]: 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)
```

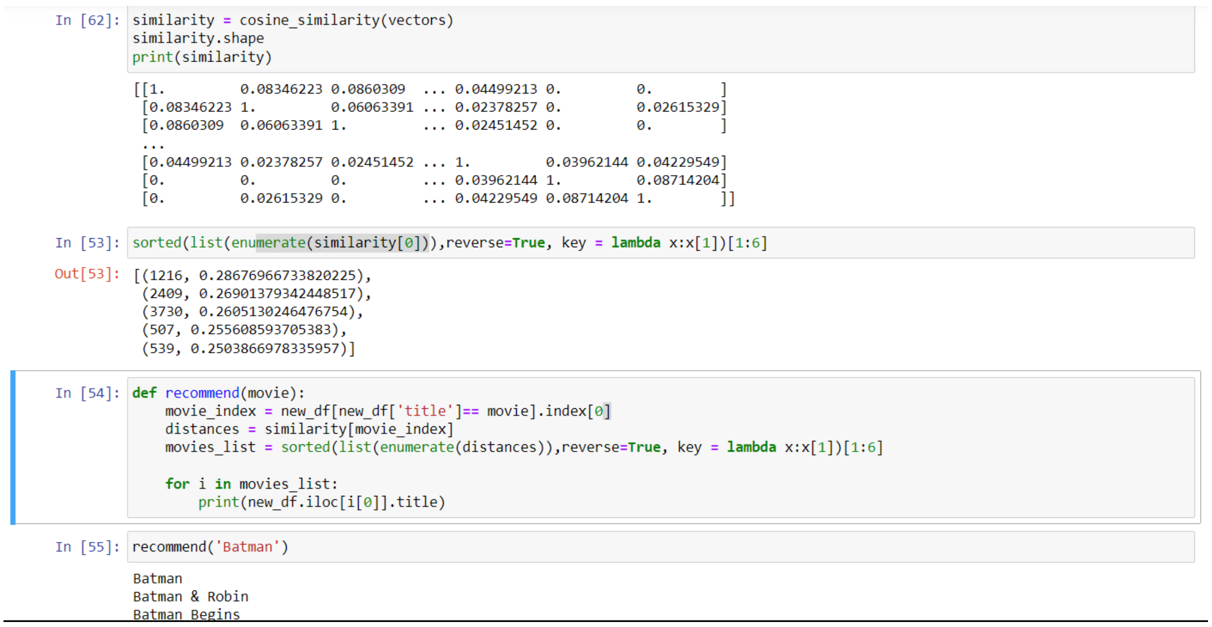
```
In [50]: for i in cv.get_feature_names_out():
             print(i)
```

```
17th
18
18th
18thcenturi
19
1910
1920
1930
1940
1944
1950
1950s
1960
1960s
1970
1970s
1971
1974
1976
1980
```

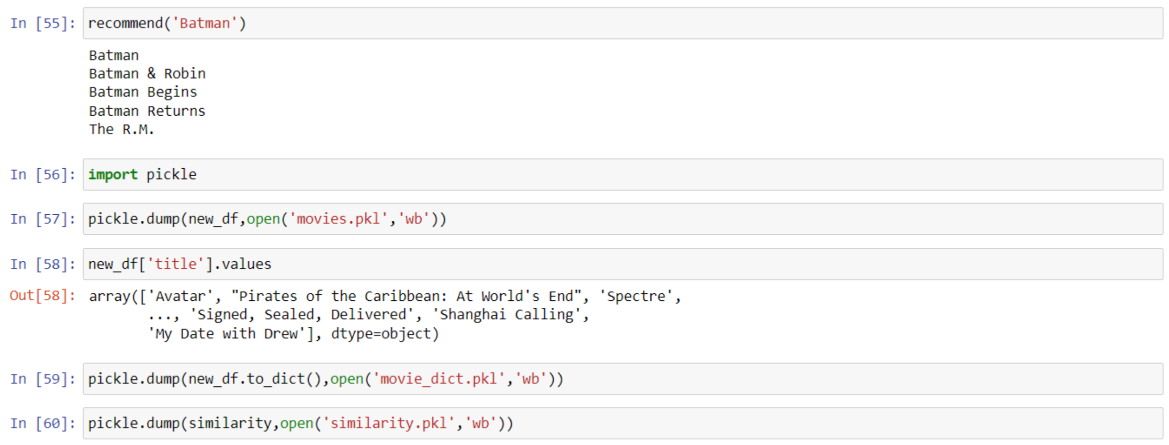
```
In [51]: from sklearn.metrics.pairwise import cosine_similarity
```

```
In [62]: similarity = cosine_similarity(vectors)
         similarity.shape
         print(similarity)
```

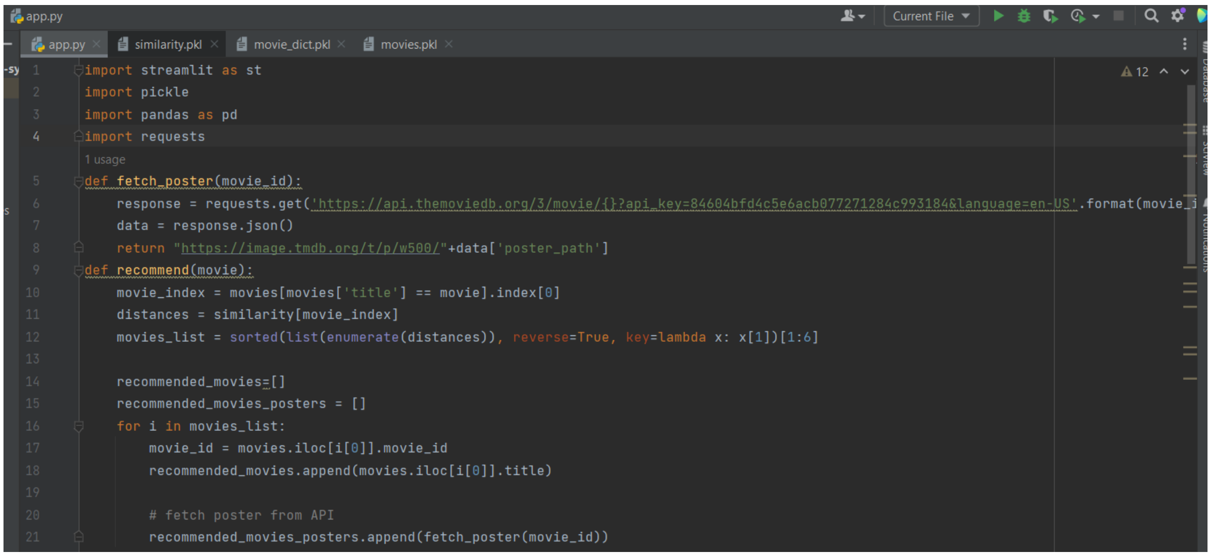
```
[[1.         0.08346223 0.0860309  ... 0.04499213 0.         0.        ]
 [0.08346223 1.         0.06063391 ... 0.02378257 0.         0.02615329]
 [0.0860309  0.06063391 1.         ... 0.02451452 0.         0.        ]
```



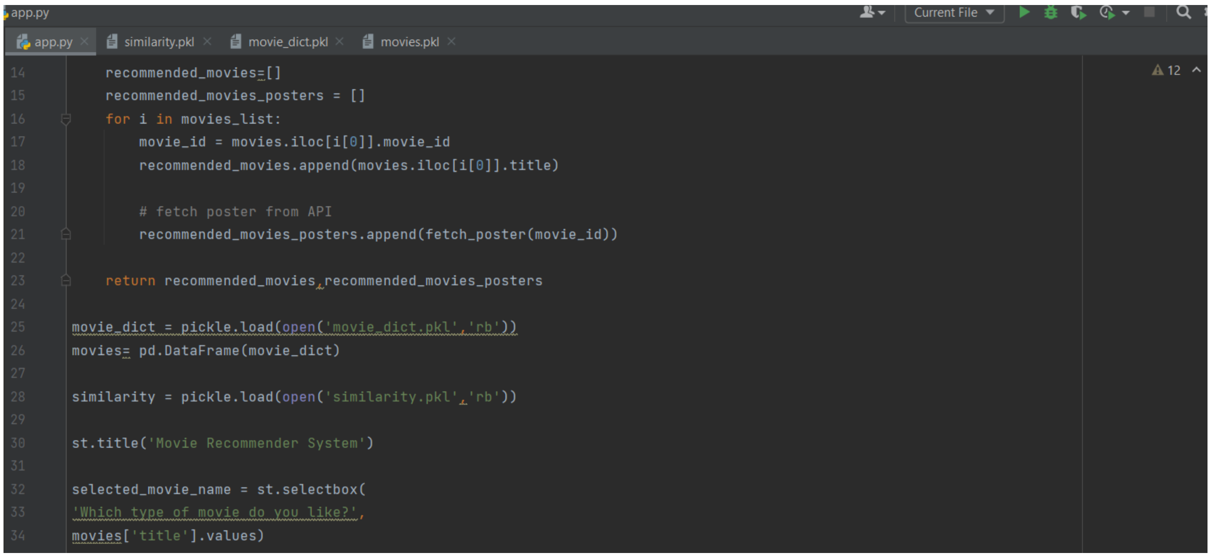
```
In [62]: similarity = cosine_similarity(vectors)
         similarity.shape
         print(similarity)

         [[1.         0.08346223 0.0860309  ... 0.04499213 0.         0.        ]
          [0.08346223 1.         0.06063391 ... 0.02378257 0.         0.02615329]
          [0.0860309  0.06063391 1.         ... 0.02451452 0.         0.        ]
          ...
          [0.04499213 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 [53]: sorted(list(enumerate(similarity[0])),reverse=True, key = lambda x:x[1])[1:6]

Out[53]: [(1216, 0.28676966733820225),
          (2409, 0.26901379342448517),
          (3730, 0.2605130246476754),
          (507, 0.255608593705383),
          (539, 0.2503866978335957)]

In [54]: 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(new_df.iloc[i[0]].title)

In [55]: recommend('Batman')

         Batman
         Batman & Robin
         Batman Begins
```



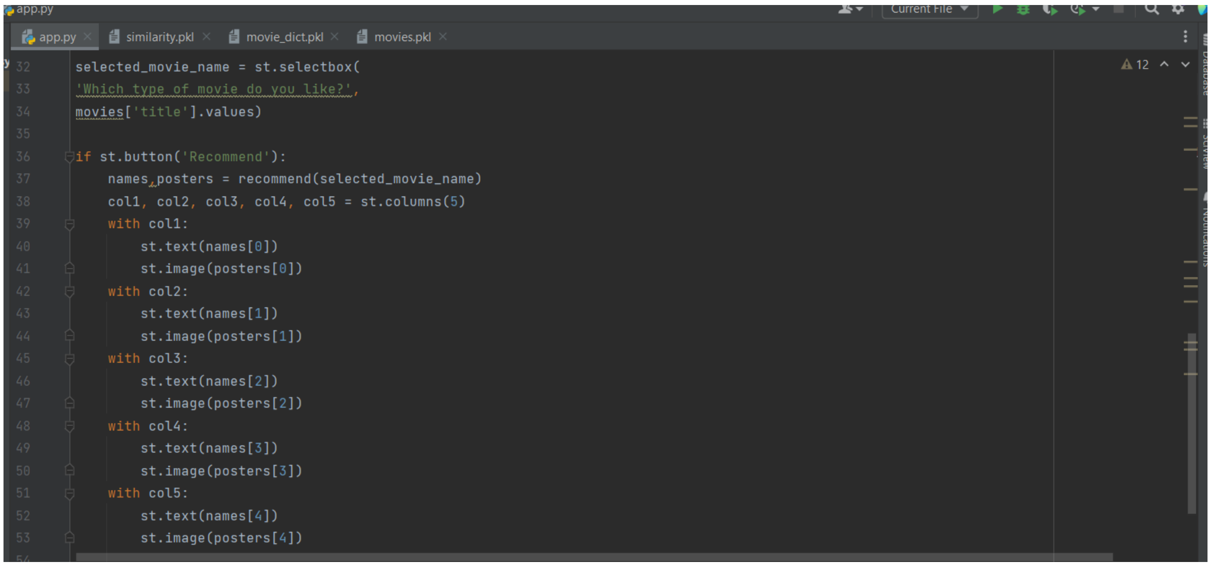
```
In [55]: recommend('Batman')

         Batman
         Batman & Robin
         Batman Begins
         Batman Returns
         The R.M.

In [56]: import pickle

In [57]: pickle.dump(new_df,open('movies.pkl','wb'))

In [58]: new_df['title'].values

Out[58]: array(['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre',
                ..., 'Signed, Sealed, Delivered', 'Shanghai Calling',
                'My Date with Drew'], dtype=object)

In [59]: pickle.dump(new_df.to_dict(),open('movie_dict.pkl','wb'))

In [60]: pickle.dump(similarity,open('similarity.pkl','wb'))
```



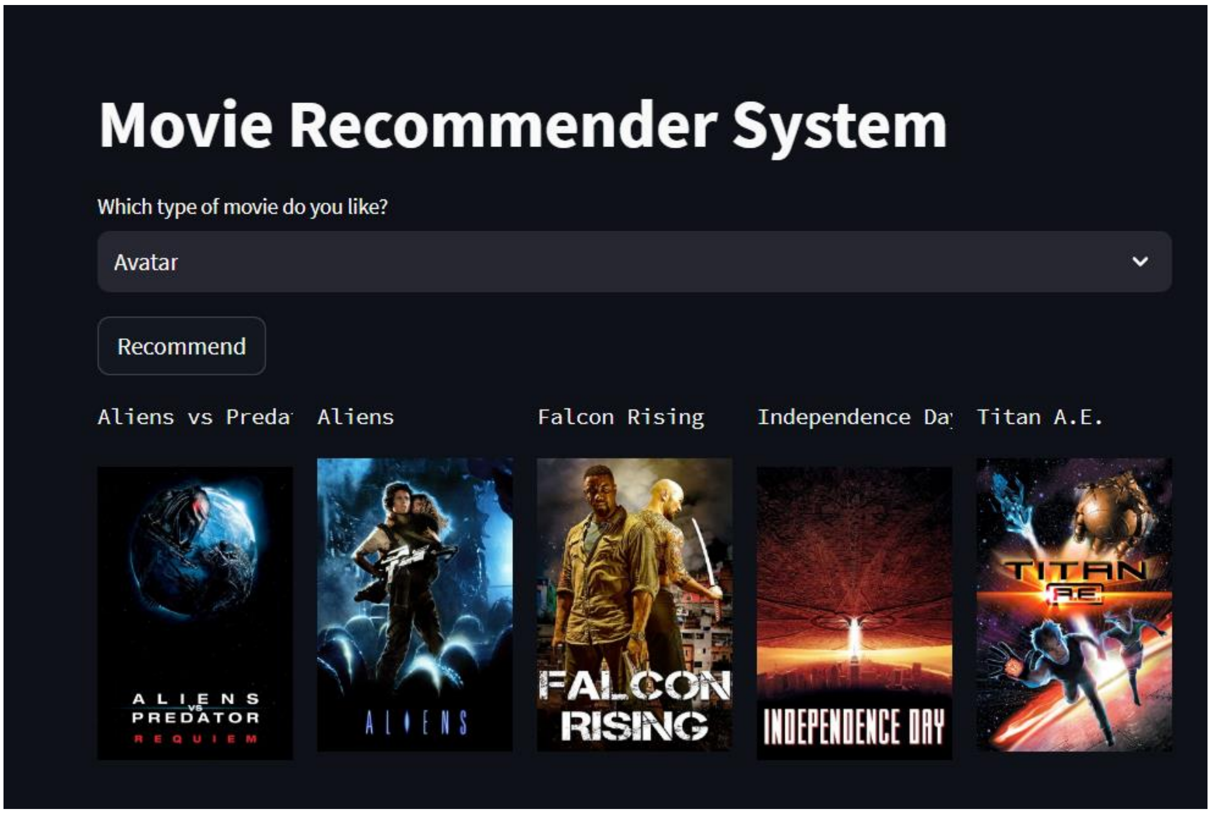
```
import streamlit as st
import pickle
import pandas as pd
import requests

def fetch_poster(movie_id):
    response = requests.get('https://api.themoviedb.org/3/movie/{}?api_key=84604bfd4c5e6acb077271284c993184&language=en-US'.format(movie_i
    data = response.json()
    return "https://image.tmdb.org/t/p/w500/"+data['poster_path']
def recommend(movie):
    movie_index = movies[movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]

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

        # fetch poster from API
        recommended_movies_posters.append(fetch_poster(movie_id))
```



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

        # fetch poster from API
        recommended_movies_posters.append(fetch_poster(movie_id))

    return recommended_movies,recommended_movies_posters

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

similarity = pickle.load(open('similarity.pkl','rb'))

st.title('Movie Recommender System')

selected_movie_name = st.selectbox(
'Which type of movie do you like?',
movies['title'].values)
```



```
selected_movie_name = st.selectbox(
'Which type of movie do you like?',
movies['title'].values)

if st.button('Recommend'):
    names,posters = recommend(selected_movie_name)
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        st.text(names[0])
        st.image(posters[0])
    with col2:
        st.text(names[1])
        st.image(posters[1])
    with col3:
        st.text(names[2])
        st.image(posters[2])
    with col4:
        st.text(names[3])
        st.image(posters[3])
    with col5:
        st.text(names[4])
        st.image(posters[4])
```



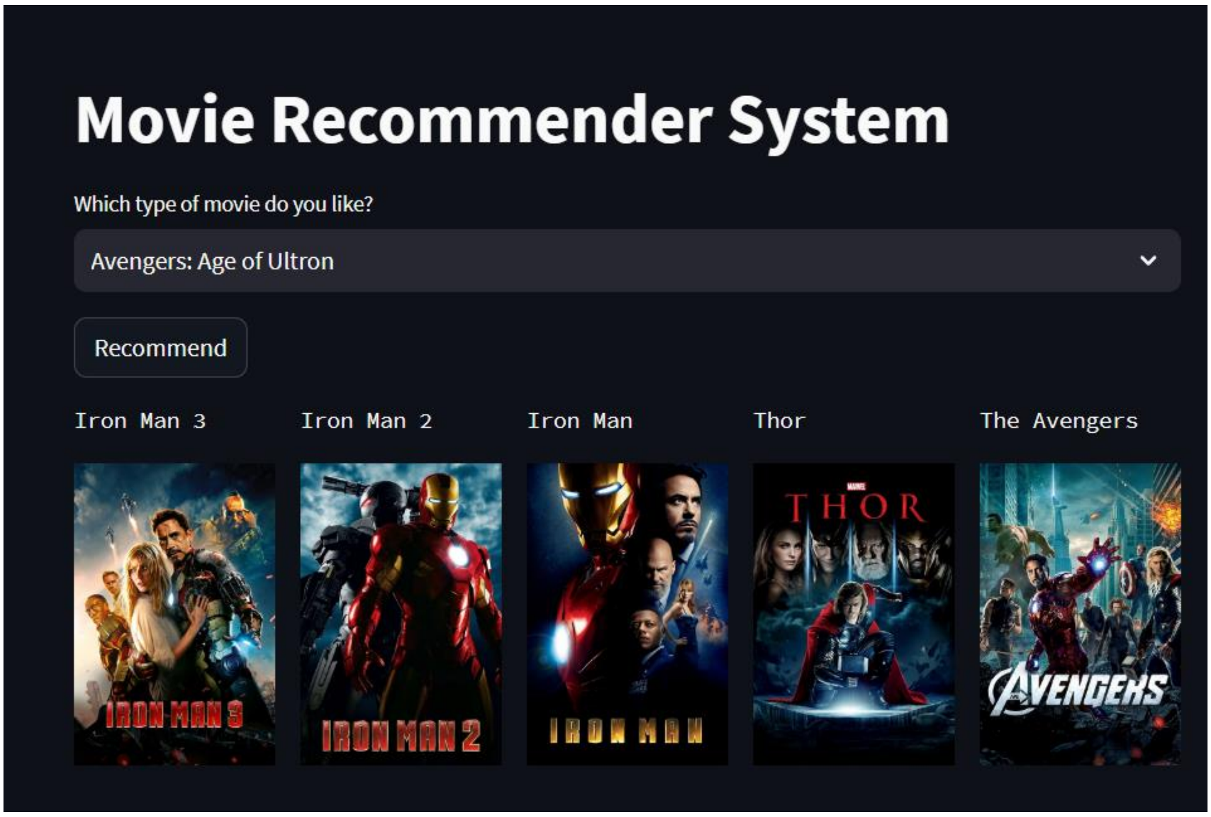
Movie Recommender System

Which type of movie do you like?

Avatar

Recommend

Aliens vs Preda Aliens Falcon Rising Independence Da Titan A.E.



Movie Recommender System

Which type of movie do you like?

Avengers: Age of Ultron

Recommend

Iron Man 3 Iron Man 2 Iron Man Thor The Avengers

Future Scope Of Improvements

1. **Advanced Similarity Measures:** Explore and experiment with different similarity measures for comparing movie descriptions. While the current model likely uses a predefined similarity matrix, you can explore other techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) or Word Embeddings (e.g., Word2Vec, GloVe) to capture more nuanced relationships between movies.
2. **Incorporate User Feedback:** Currently, the model does not take into account user feedback or preferences. To enhance the recommendation system, consider integrating mechanisms for collecting and incorporating user ratings, reviews, or implicit feedback (e.g., user interactions, watch history). This can help personalize the recommendations and improve their accuracy.
3. **Genre-specific Recommendations:** Expand the model to consider genre-specific recommendations. Instead of solely relying on overall movie descriptions, you can extract genre information and incorporate it into the recommendation process. This would allow users to receive recommendations tailored to their preferred genres or explore movies across different genres.
4. **Hybrid Recommender Systems:** Consider combining content-based filtering (using movie descriptions) with collaborative filtering techniques. Hybrid recommender systems leverage both content-based and collaborative filtering approaches to provide more accurate and diverse recommendations. You can explore techniques like matrix factorization, neighborhood-based collaborative filtering, or hybrid ensemble models.

5. **Incorporate External Data:** Augment the existing dataset with additional sources of movie-related information, such as movie reviews, social media sentiment analysis, or external databases. By integrating external data, you can enrich the recommendation system with more comprehensive and up-to-date information about movies, improving the quality and relevance of the recommendations.
6. **Fine-tune Hyperparameters:** Experiment with different hyperparameter settings for the recommendation model. This includes the number of top similar movies to consider, the threshold for similarity scores, or the number of features in the Bag of Words representation. Fine-tuning these hyperparameters can optimize the performance of the recommendation system.
7. **User Interface and Visualization:** Enhance the user interface and visualization aspects of the application. Consider incorporating interactive visualizations, movie posters, trailers, or additional metadata to provide a more engaging and informative user experience. Ensure that the recommendations are presented in a user-friendly and intuitive manner.

Certificate

This is to certify that Ms. Trisha Roy of Lovely Professional University, registration number: 12101718, has successfully completed a project on Movie Recommendation System using Machine Learning with Python under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty

Fifth Force