EXPERIMENT - 8

AIM: Recommendation system using Machine Learning

1) Use any dataset
2) Use any type of Recommendation technique

THEORY:

Recommender systems are the systems that are designed to recommend things to the user based on many different factors. These systems predict the most likely product that the users are most likely to purchase and are of interest to. Companies like Netflix, Amazon, etc. use recommender systems to help their users to identify the correct product or movies for them.

The recommender system deals with a large volume of information present by filtering the most important information based on the data provided by a user and other factors that take care of the user's preference and interest. It finds out the match between user and item and imputes the similarities between users and items for recommendation.

Recommender System is different types

- **Collaborative Filtering:** Collaborative Filtering recommends items based on similarity measures between users and/or items. The basic assumption behind the algorithm is that users with similar interests have common preferences.
- **Content-Based Recommendation:** It is supervised machine learning used to induce a classifier to discriminate between interesting and uninteresting items for the user.

Content-Based Recommendation System: Content-Based systems recommends items to the customer similar to previously high-rated items by the customer. It uses the features and properties of the item. From these properties, it can calculate the similarity between the items.

Advantages and Disadvantages:

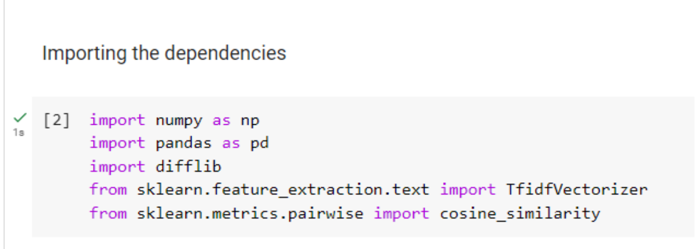
- Advantages:
    - No need for data on other users when applying to similar users.
    - Able to recommend to users with unique tastes.
    - Able to recommend new & popular items
    - Explanations for recommended items.
- Disadvantages:
    - Finding the appropriate feature is hard.
    - Doesn't recommend items outside the user profile.

Collaborative Filtering: Collaborative filtering is based on the idea that similar people (based on the data) generally tend to like similar things. It predicts which item a user will like based on the item preferences of other similar users.

Collaborative filtering uses a user-item matrix to generate recommendations. This matrix contains the values that indicate a user's preference towards a given item. These values can represent either explicit feedback (direct user ratings) or implicit feedback (indirect user behavior such as listening, purchasing, watching).

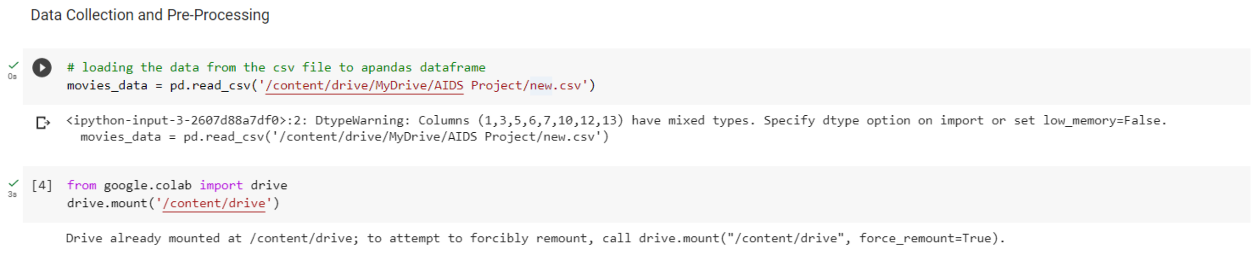
- **Explicit Feedback:** The amount of data that is collected from the users when they choose to do so. Many of the times, users choose not to provide data for the user. So, this data is scarce and sometimes costs money. For example, ratings from the user.
- **Implicit Feedback:** In implicit feedback, we track user behavior to predict their preference.

Implementation:



Importing the dependencies

```
[2] import numpy as np
    import pandas as pd
    import difflib
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.metrics.pairwise import cosine_similarity
```



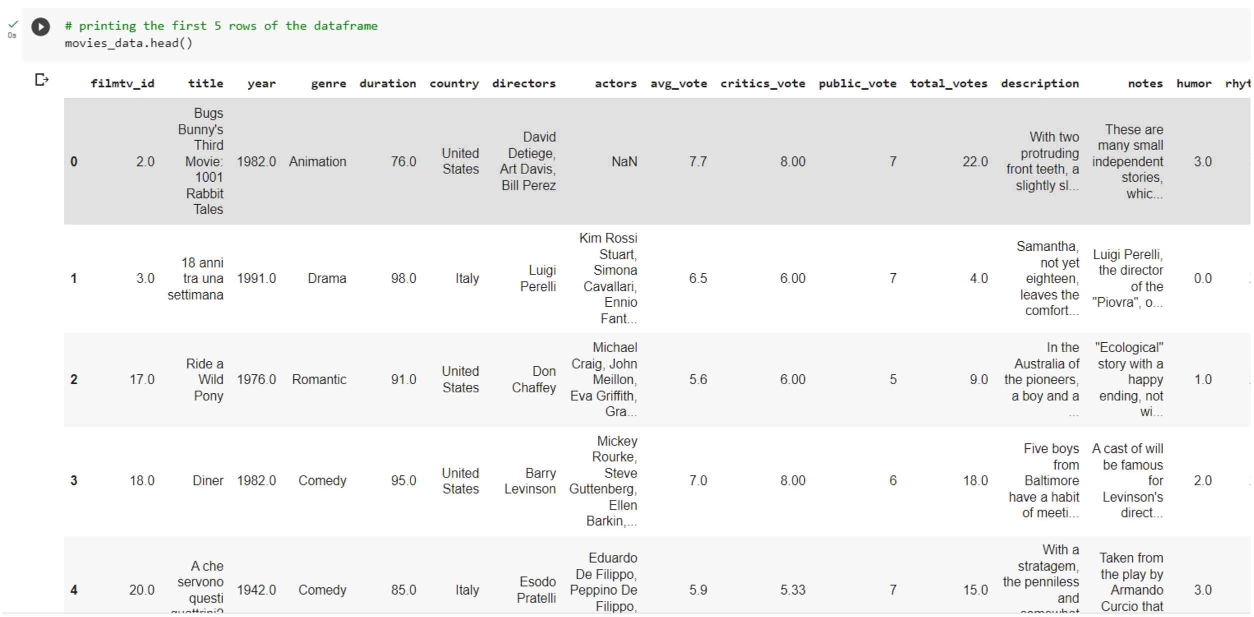
Data Collection and Pre-Processing

```
# loading the data from the csv file to apandas dataframe
movies_data = pd.read_csv('/content/drive/MyDrive/AIDS Project/new.csv')
```

```
<ipython-input-3-2607d88a7df0>:2: DtypeWarning: Columns (1,3,5,6,7,10,12,13) have mixed types. Specify dtype option on import or set low_memory=False.
  movies_data = pd.read_csv('/content/drive/MyDrive/AIDS Project/new.csv')
```

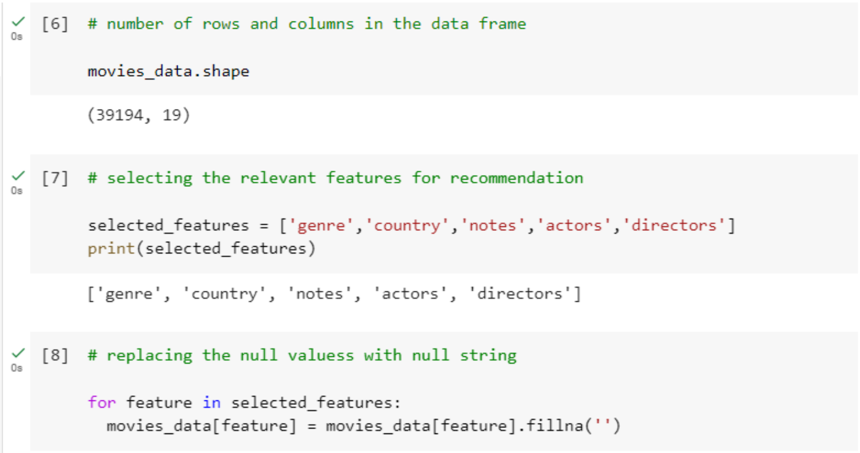
```
[4] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```



```
# printing the first 5 rows of the dataframe
movies_data.head()
```

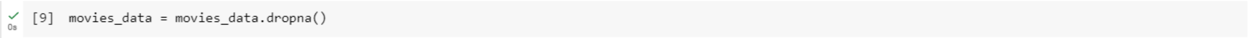
	filmtv_id	title	year	genre	duration	country	directors	actors	avg_vote	critics_vote	public_vote	total_votes	description	notes	humor	rhyt
0	2.0	Bugs Bunny's Third Movie: 1001 Rabbit Tales	1982.0	Animation	76.0	United States	David Detiege, Art Davis, Bill Perez	NaN	7.7	8.00	7	22.0	With two protruding front teeth, a slightly sl...	These are many small independent stories, whic...	3.0	
1	3.0	18 anni tra una settimana	1991.0	Drama	98.0	Italy	Luigi Perelli	Kim Rossi Stuart, Simona Cavallari, Ennio Fant...	6.5	6.00	7	4.0	Samantha, not yet eighteen, leaves the comfort...	Luigi Perelli, the director of the "Piovra", o...	0.0	
2	17.0	Ride a Wild Pony	1976.0	Romantic	91.0	United States	Don Chaffey	Michael Craig, John Meillon, Eva Griffith, Gra...	5.6	6.00	5	9.0	In the Australia of the pioneers, a boy and a ...	"Ecological" story with a happy ending, not wi...	1.0	
3	18.0	Diner	1982.0	Comedy	95.0	United States	Barry Levinson	Mickey Rourke, Steve Guttenberg, Ellen Barkin,...	7.0	8.00	6	18.0	Five boys from Baltimore have a habit of meeti...	A cast of will be famous for Levinson's direct...	2.0	
4	20.0	A che servono questi quattrini?	1942.0	Comedy	85.0	Italy	Esodo Pratelli	Eduardo De Filippo, Peppino De Filippo,	5.9	5.33	7	15.0	With a stratagem, the penniless and somewhat	Taken from the play by Armando Curcio that	3.0	



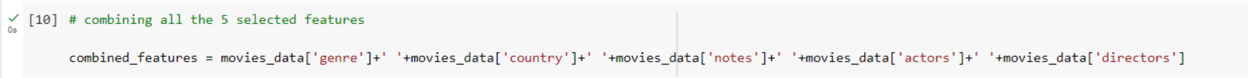
```
[6] # number of rows and columns in the data frame

    movies_data.shape

    (39194, 19)
```

```
[7] # selecting the relevant features for recommendation

    selected_features = ['genre','country','notes','actors','directors']
    print(selected_features)

    ['genre', 'country', 'notes', 'actors', 'directors']
```

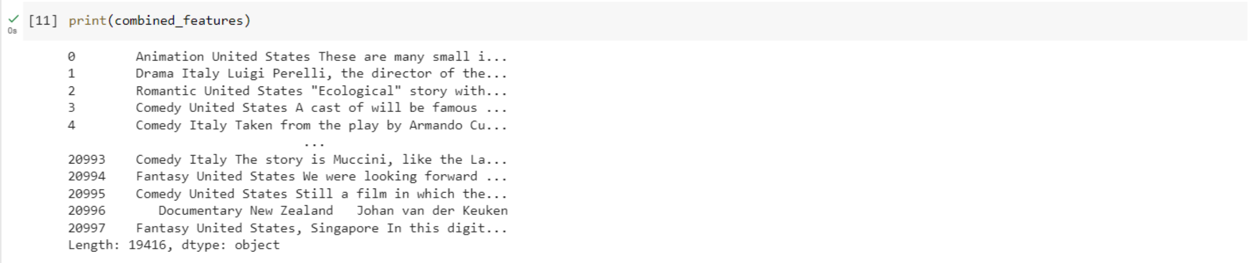
```
[8] # replacing the null valuess with null string

    for feature in selected_features:
      movies_data[feature] = movies_data[feature].fillna('')
```



```
[9] movies_data = movies_data.dropna()
```

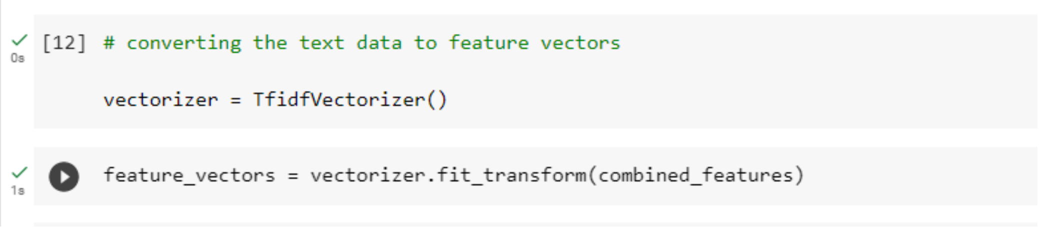


```
[10] # combining all the 5 selected features

     combined_features = movies_data['genre']+' '+movies_data['country']+' '+movies_data['notes']+' '+movies_data['actors']+' '+movies_data['directors']
```



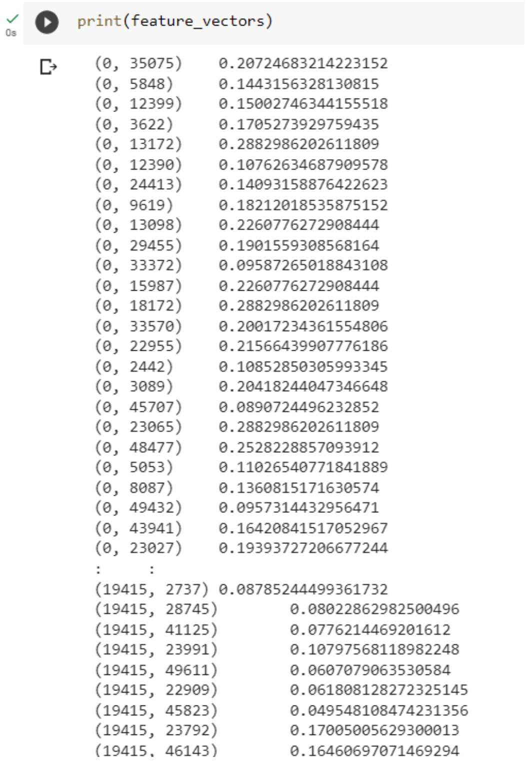
```
[11] print(combined_features)

     0        Animation United States These are many small i...
     1        Drama Italy Luigi Perelli, the director of the...
     2        Romantic United States "Ecological" story with...
     3        Comedy United States A cast of will be famous ...
     4        Comedy Italy Taken from the play by Armando Cu...
                                    ...
     20993    Comedy Italy The story is Muccini, like the La...
     20994    Fantasy United States We were looking forward ...
     20995    Comedy United States Still a film in which the...
     20996       Documentary New Zealand   Johan van der Keuken
     20997    Fantasy United States, Singapore In this digit...
     Length: 19416, dtype: object
```

- Vectorization is done to enhance the performance of our system



```
[12] # converting the text data to feature vectors

     vectorizer = TfidfVectorizer()
```

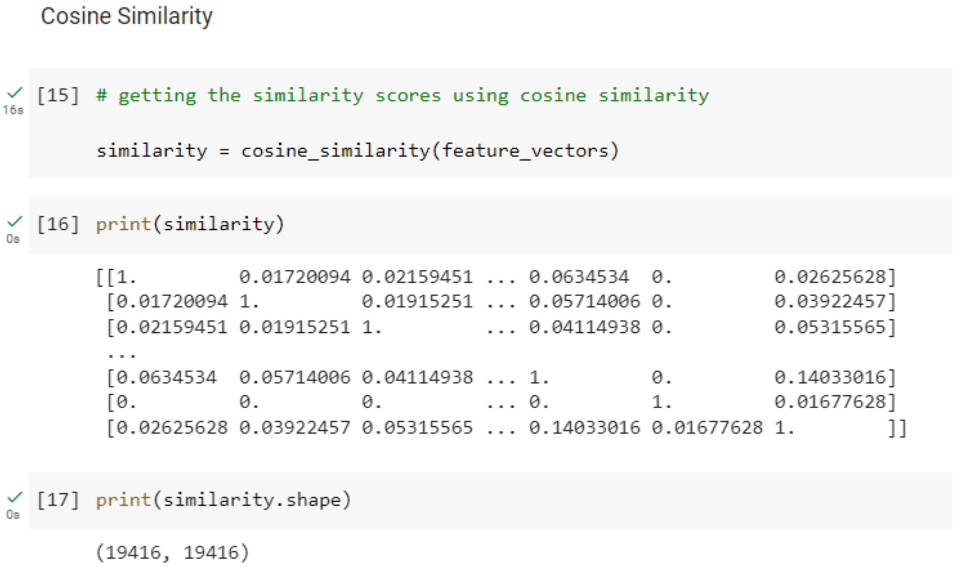
```
feature_vectors = vectorizer.fit_transform(combined_features)
```



```
print(feature_vectors)
```

```
(0, 35075)    0.20724683214223152
(0, 5848)     0.1443156328130815
(0, 12399)    0.15002746344155518
(0, 3622)     0.1705273929759435
(0, 13172)    0.2882986202611809
(0, 12390)    0.10762634687909578
(0, 24413)    0.14093158876422623
(0, 9619)     0.18212018535875152
(0, 13098)    0.2260776272908444
(0, 29455)    0.1901559308568164
(0, 33372)    0.09587265018843108
(0, 15987)    0.2260776272908444
(0, 18172)    0.2882986202611809
(0, 33570)    0.20017234361554806
(0, 22955)    0.21566439907776186
(0, 2442)     0.10852850305993345
(0, 3089)     0.20418244047346648
(0, 45707)    0.0890724496232852
(0, 23065)    0.2882986202611809
(0, 48477)    0.2528228857093912
(0, 5053)     0.11026540771841889
(0, 8087)     0.1360815171630574
(0, 49432)    0.0957314432956471
(0, 43941)    0.16420841517052967
(0, 23027)    0.19393727206677244
:     :
(19415, 2737) 0.08785244499361732
(19415, 28745)        0.08022862982500496
(19415, 41125)        0.0776214469201612
(19415, 23991)        0.10797568118982248
(19415, 49611)        0.0607079063530584
(19415, 22909)        0.061808128272325145
(19415, 45823)        0.049548108474231356
(19415, 23792)        0.17005005629300013
(19415, 46143)        0.16460697071469294
```

- Cosine Similarity helps us to find similarity between two parameters present. In our case, it finds similarity based on title name and genre.



Cosine Similarity

```
[15] # getting the similarity scores using cosine similarity

     similarity = cosine_similarity(feature_vectors)
```

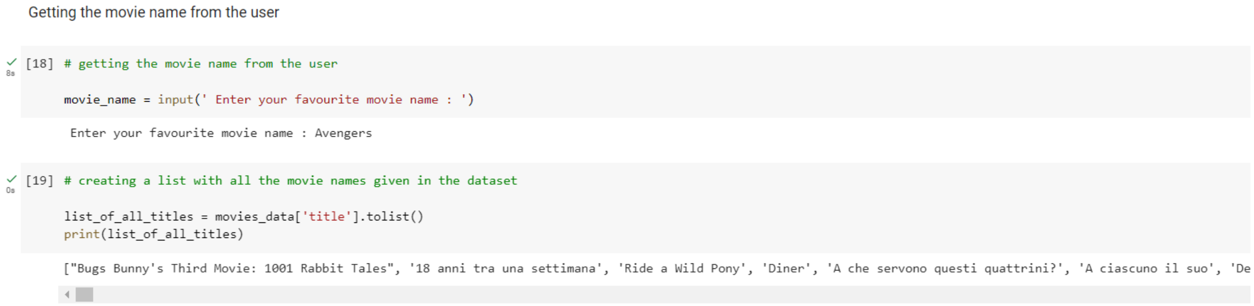
```
[16] print(similarity)
```

```
[[1.         0.01720094 0.02159451 ... 0.0634534  0.         0.02625628]
 [0.01720094 1.         0.01915251 ... 0.05714006 0.         0.03922457]
 [0.02159451 0.01915251 1.         ... 0.04114938 0.         0.05315565]
 ...
 [0.0634534  0.05714006 0.04114938 ... 1.         0.         0.14033016]
 [0.         0.         0.         ... 0.         1.         0.01677628]
 [0.02625628 0.03922457 0.05315565 ... 0.14033016 0.01677628 1.        ]]
```

```
[17] print(similarity.shape)
```

```
(19416, 19416)
```

- Getting the input from the user

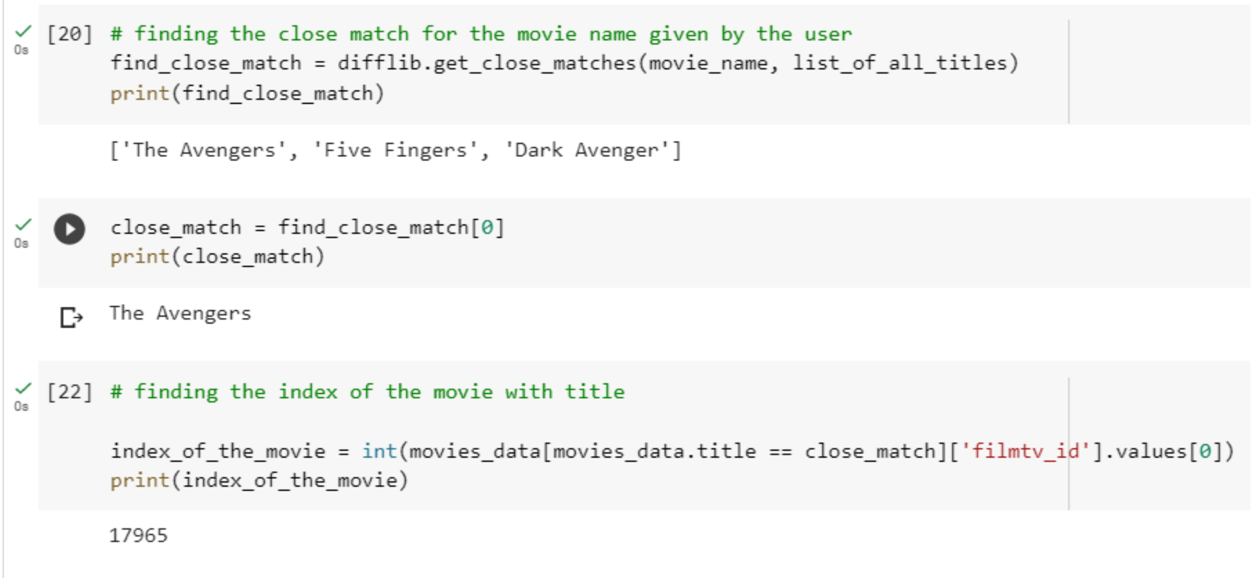


Getting the movie name from the user

```
[18] # getting the movie name from the user

     movie_name = input(' Enter your favourite movie name : ')

      Enter your favourite movie name : Avengers
```

```
[19] # creating a list with all the movie names given in the dataset

     list_of_all_titles = movies_data['title'].tolist()
     print(list_of_all_titles)

     ["Bugs Bunny's Third Movie: 1001 Rabbit Tales", '18 anni tra una settimana', 'Ride a Wild Pony', 'Diner', 'A che servono questi quattrini?', 'A ciascuno il suo', 'De
```

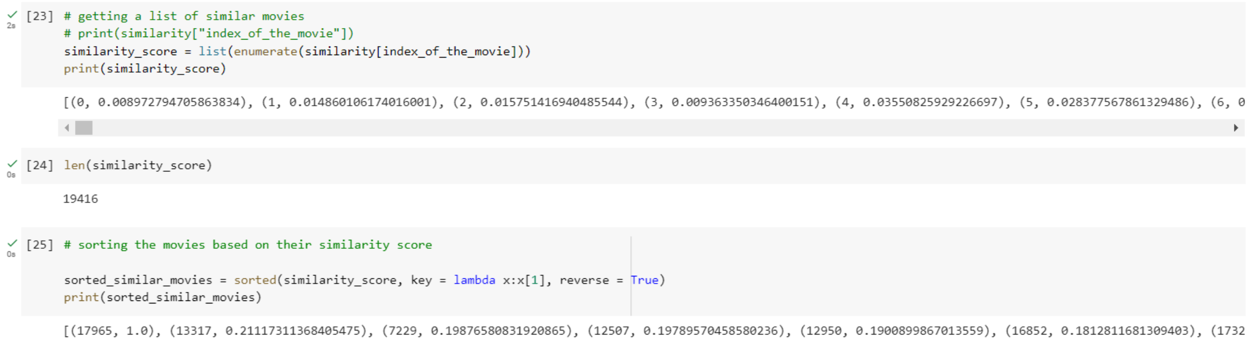
- We find the closest match and calculate the similarity score. We sort the data based on similarity scores.



```
[20] # finding the close match for the movie name given by the user
     find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
     print(find_close_match)

     ['The Avengers', 'Five Fingers', 'Dark Avenger']
```

```
     close_match = find_close_match[0]
     print(close_match)

     The Avengers
```

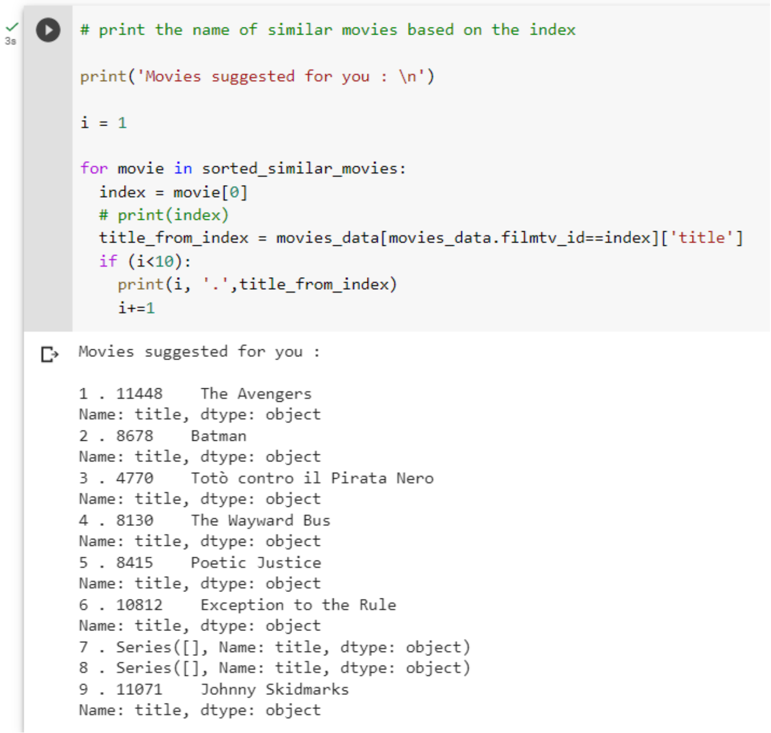
```
[22] # finding the index of the movie with title

     index_of_the_movie = int(movies_data[movies_data.title == close_match]['filmtv_id'].values[0])
     print(index_of_the_movie)

     17965
```



```
[23] # getting a list of similar movies
     # print(similarity["index_of_the_movie"])
     similarity_score = list(enumerate(similarity[index_of_the_movie]))
     print(similarity_score)

     [(0, 0.008972794705863834), (1, 0.014860106174016001), (2, 0.015751416940485544), (3, 0.009363350346400151), (4, 0.03550825929226697), (5, 0.028377567861329486), (6, 0
```

```
[24] len(similarity_score)

     19416
```

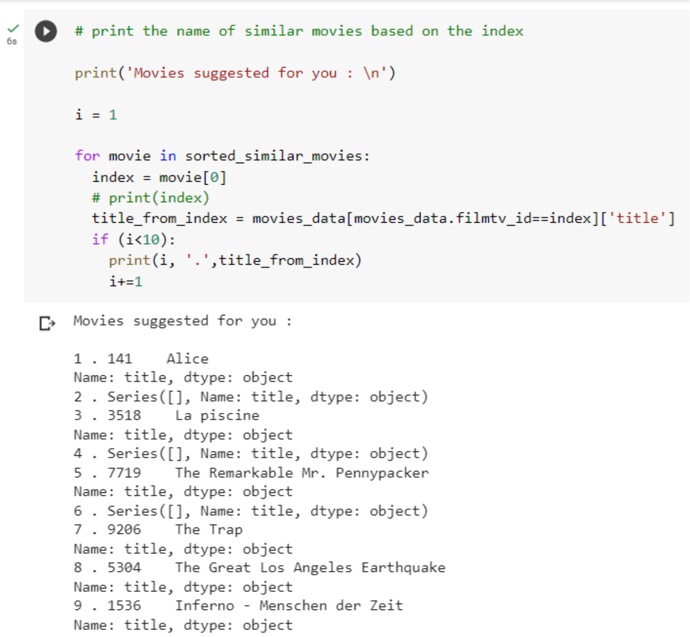
```
[25] # sorting the movies based on their similarity score

     sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
     print(sorted_similar_movies)

     [(17965, 1.0), (13317, 0.21117311368405475), (7229, 0.19876580831920865), (12507, 0.19789570458580236), (12950, 0.1900899867013559), (16852, 0.1812811681309403), (1732
```



```python
# print the name of similar movies based on the index

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  # print(index)
  title_from_index = movies_data[movies_data.filmtv_id==index]['title']
  if (i<10):
    print(i, '.',title_from_index)
    i+=1
```

```
Movies suggested for you :

1 . 11448    The Avengers
Name: title, dtype: object
2 . 8678    Batman
Name: title, dtype: object
3 . 4770    Totò contro il Pirata Nero
Name: title, dtype: object
4 . 8130    The Wayward Bus
Name: title, dtype: object
5 . 8415    Poetic Justice
Name: title, dtype: object
6 . 10812    Exception to the Rule
Name: title, dtype: object
7 . Series([], Name: title, dtype: object)
8 . Series([], Name: title, dtype: object)
9 . 11071    Johnny Skidmarks
Name: title, dtype: object
```

- Input: Alice
- We get the similar movies to Alice’s genre



```python
# print the name of similar movies based on the index

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  # print(index)
  title_from_index = movies_data[movies_data.filmtv_id==index]['title']
  if (i<10):
    print(i, '.',title_from_index)
    i+=1
```

```
Movies suggested for you :

1 . 141    Alice
Name: title, dtype: object
2 . Series([], Name: title, dtype: object)
3 . 3518    La piscine
Name: title, dtype: object
4 . Series([], Name: title, dtype: object)
5 . 7719    The Remarkable Mr. Pennypacker
Name: title, dtype: object
6 . Series([], Name: title, dtype: object)
7 . 9206    The Trap
Name: title, dtype: object
8 . 5304    The Great Los Angeles Earthquake
Name: title, dtype: object
9 . 1536    Inferno - Menschen der Zeit
Name: title, dtype: object
```

CONCLUSION: In this recommendation system we used Explicit Feedback as we take input from the user and then we provide the output based on that. This system deals with providing similar movies by calculating the cosine similarities between two parameters.Thus we successfully created a recommendation system.