

Project report on

# MOVIE RECOMMENDATION SYSTEM USING MACHINE LEARNING WITH PYTHON

by  
**(2020-350-020)-Hamza**

under supervision of  
**Dr.Ihtiram Raza Khan**  
**(Associate Professor)**



**computer science engineering with artificial intelligence**  
**school of engineering and science Jamia Hamdard**

**May.2023**

## **ABSTRACT**

- ➔ Aim- of this project is to make is system which can take user input and suggest movie based on input given by the user in the form of movie name.
- ➔ The main factor by which we are recommending movie is **Cosine similarity algorithm in which we are taking input from the user and comparing that movie title with 4800 movies which are present in our data set.**
- ➔ But with textual data finding similarity is not easy that's why we convert into numerical.
- ➔ There were 4000 plus row but I done feature selection to make it handleable.
- ➔ To find closest match we had used library called difflib.
- ➔ Then will get 3 closest match and index value by which will find title of a movie name to recommend to the user.
- ➔ The high similarity score shows that movie we should recommend to user
- ➔ enumerate () to run loop in a particular list and it will give you the count.
- ➔ For loop used to get index value to find the title of movie.
- ➔ This system will enhance the user experience and improve engagement on movie streaming platforms.

## LIST OF CONTENT

S_no	Content	page no	Signature
1	ABSTRACT	2	
2	INTRODUCTION	4	
3	METHODOLOGY	5	
4	converting the text data to feature vectors	7	
5	Getting similarity score	8	
6	list of movie	10	
7	Movie Recommendation System	11	
8	Result	12	
9	output with another input	13	
10	link for entire program	15	
11	conclusion	16	
12	Reference	18	
13	Thank you	19	

# INTRODUCTION

A movie recommendation system, is an ML(machine learning)-based approach to predicting the users' film preferences based on their past choices and behavior.

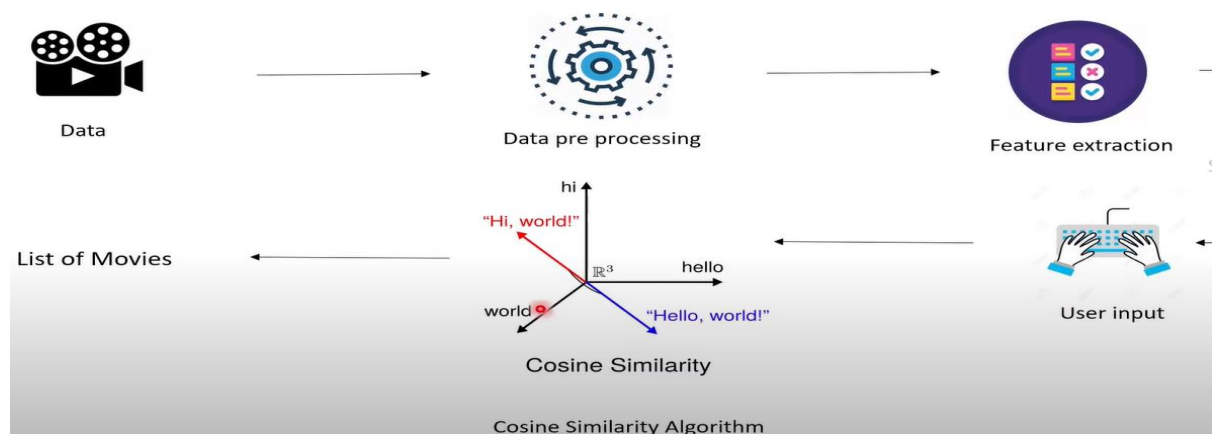
There are 3 main types of recommendation system

content

popularity

collaborative based recommendation system. So I will be making content and popularity based recommendation system.

The workflow that I will be follow:-



Getting data which is having all day details of movie

Step 2 data preprocessing cleaning missing values if exist

Step 3 feature extraction we have textual data so we can't use it we convert into numerical data then will give similarities score

stop 4 asking user input then will give suggestion **using Cosine similarity algorithm** which use to find similarity between vectors

Step 5 list of movies will be shown on terminal

## METHODOLOGY

### Using data set movies.CSV having several attributes

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
index	budget	genres	homepageid	keywords	original_la	original_ti	overview	popularity	production production release_d	revenue	runtime	spoken_la	status	tagline	title	vo		

### importing libraries/Importing the dependencies

**numpy** for creating numpy arrays.

**Pandas** for creating data frames(structured table)

**difflib** can handle wrong spelling entered by user and for matching which movie & to get close match.

from **sklearn.feature\_extraction.text** import **TfidfVectorizer** it used to convert textual data to numerical(feature vector) once we have them so it is easier to find **Cosine** similarity value

from **sklearn.metrics.pairwise** import **cosine\_similarity** it gives similarity scores by which we can recommend movies

IDEA is **Cosine similarity algorithm** which is used to find similarity value if the similarity value of 2 movies is 1 then both movies is similar to each other and there is 1 probability that user will like it.

```
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

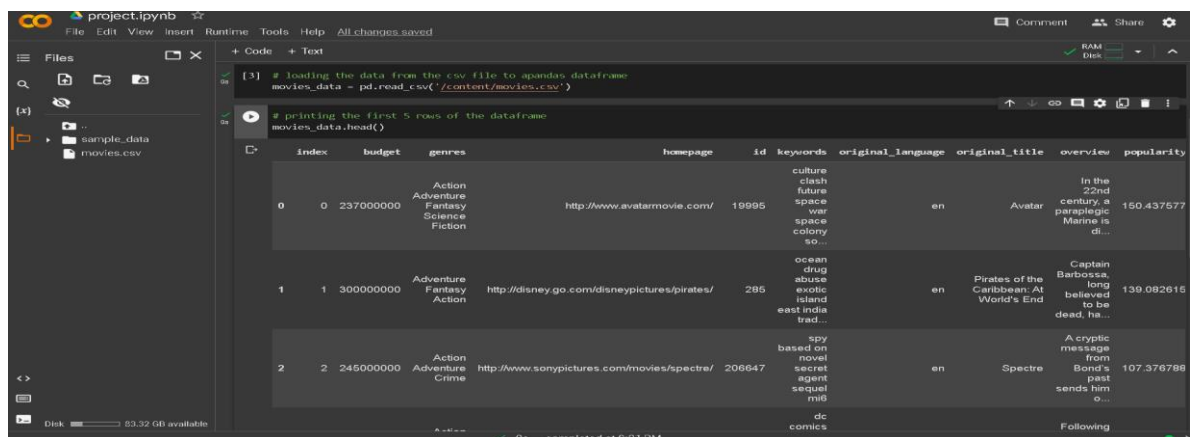
### Data Collection and Pre-Processing

we have data in CSV file and loading it into data frame.

Printing 5 rows of data frame.

## Movie Dataset

1. Title: Movie Title.
2. Overview: Abstract of the Movie.
3. Popularity: Movie popularity rating as per TMDB.
4. Vote\_count: Number of votes from the users.
5. Release\_date: Date of release of the movie.
6. Keywords: Keywords for the movie by TMDB in the list.
7. Genres: Movie Genres in the list.
8. Cast: Cast of the movie on the list.
9. Crew: Crew of the movie in the list etc.



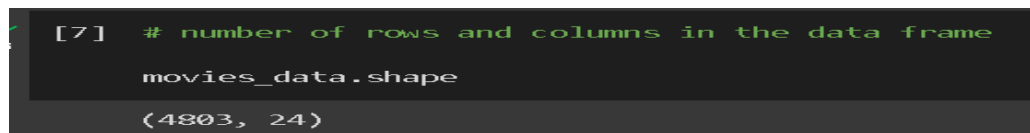
The screenshot shows a Jupyter Notebook interface with the following code and output:

```
[3]: # loading the data from the csv file to a pandas dataframe
movies_data = pd.read_csv('/content/movies.csv')

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

	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india red...	en	Pirates of the Caribbean: At World's End	Captain Barbossa long believed to be dead, he...	139.082615
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel m06	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788

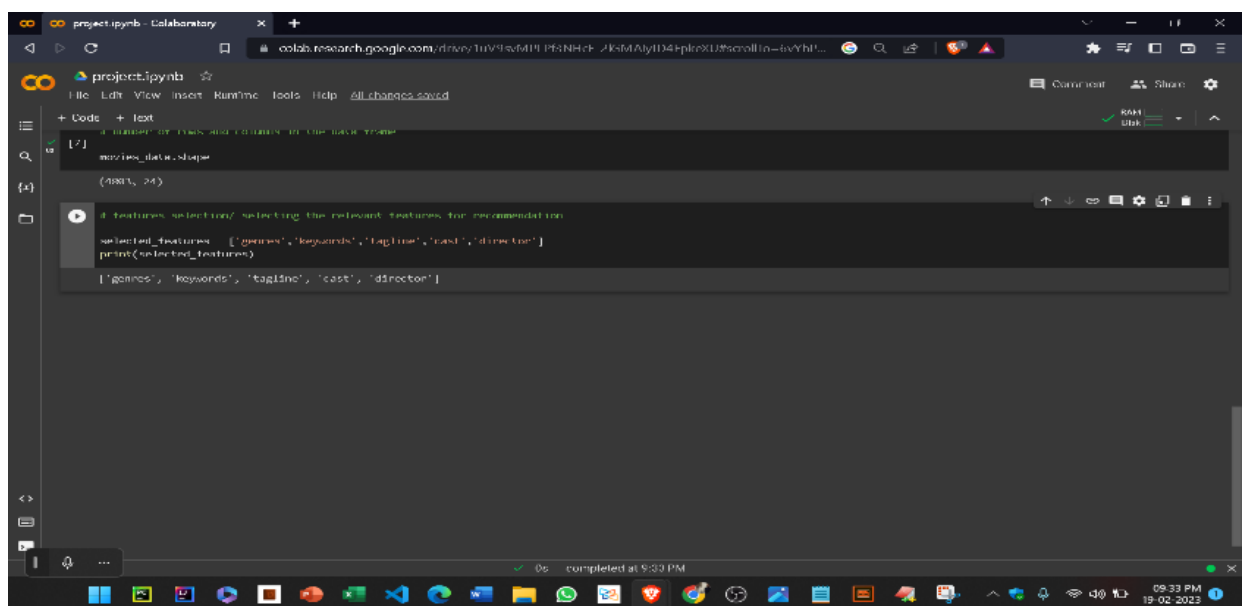
Feature selection taking 5 attributes out of 24 because all of them we don't want



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

movies_data.shape

(4803, 24)
```



```
[7]: # number of rows and columns in the data frame
movies_data.shape

(4803, 24)

# features selection/ selecting the relevant features for recommendation
selected_features = ['genres', 'keywords', 'tagline', 'cast', 'director']
print(selected_features)

['genres', 'keywords', 'tagline', 'cast', 'director']
```

replacing the null values with null string

```
✓ 0s # replacing the null values with null string

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

combining all the 5 selected features

```
✓ 0s [13] # combining all the 5 selected features

combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movies_data['tagline']+' '+movies_data['cast']+' '+movies_data['director']

✓ 0s print(combined_features)

0    Action Adventure Fantasy Science Fiction cultu...
1    Adventure Fantasy Action ocean drug abuse exot...
2    Action Adventure Crime spy based on novel secr...
3    Action Crime Drama Thriller dc comics crime fi...
4    Action Adventure Science Fiction based on nove...
...
4798  Action Crime Thriller united states\u2013mexic...
4799  Comedy Romance  A newlywed couple's honeymoon ...
4800  Comedy Drama Romance TV Movie date love at fir...
4801  A New Yorker in Shanghai Daniel Henney Eliza...
4802  Documentary obsession camcorder crush dream gi...
Length: 4803, dtype: object
```

converting the text data to feature vectors reason is with textual data we not easy to find **Cosine similarity** but with numerical easily we can find. Fitting and transforming to get a feature vector.

```
vectorizer = TfidfVectorizer()

[19] feature_vectors = vectorizer.fit_transform(combined_features)

print(feature_vectors)
```

(0, 2432)	0.17272411194153
(0, 7755)	0.1128035714854756
(0, 13024)	0.1942362060108871
(0, 10229)	0.16058685400095302
(0, 8756)	0.22709015857011816
(0, 14608)	0.15150672398763912
(0, 16668)	0.19843263965100372
(0, 14064)	0.20596090415084142
(0, 13319)	0.2177470539412484
(0, 17290)	0.20197912553916567
(0, 17007)	0.23643326319898797
(0, 13349)	0.15021264094167086
(0, 11503)	0.27211310056983656
(0, 11192)	0.09049319826481456
(0, 16998)	0.1282126322850579
(0, 15261)	0.07095833561276566
(0, 4945)	0.24025852494110758
(0, 14271)	0.21392179219912877
(0, 3225)	0.24960162956997736
(0, 16587)	0.12549432354918996
(0, 14378)	0.33962752210959823
(0, 5836)	0.1646750903586285
(0, 3065)	0.22208377802661425

**Getting the similarity scores using cosine similarity and printing it.**

```
# getting the similarity scores using cosine similarity
similarity = cosine_similarity(feature_vectors)

[25] print(similarity)
```

[[1.	0.07219487	0.037733	...	0.	0.	0.	]
[0.07219487	1.	0.03281499	...	0.03575545	0.	0.	]
[0.037733	0.03281499	1.	...	0.	0.05389661	0.	]
...							
[0.	0.03575545	0.	...	1.	0.	0.02651502	]
[0.	0.	0.05389661	...	0.	1.	0.	]
[0.	0.	0.	...	0.02651502	0.	1.	]]

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

(4803, 4803)

**it will be taking each movie name and comparing with remaining one**



Getting the movie name from the user

& creating a list with all the movie names given in the dataset

```
✓ 5s # getting the movie name from the user

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

Enter your favourite movie name : iron man

✓ 0s [30] # 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)

['Avatar', 'Pirates of the Caribbean: At World's End', 'Spectre', 'The Dark Knight Rises', 'John Carter', 'Spider-Man 3', 'Tangled', 'Avengers: Age of Ultron', 'Harry Potter ar
```

after input will check it is present in data set then will work with similarities score  
creating list for comparing with user input

will try to find close match using difflib

tolist() you used to create list

finding the close match for the movie name given by the user

```
✓ 1s [31] # 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)

['Iron Man', 'Gridiron Gang', 'Iron Man 3']

✓ 0s [32] close_match = find_close_match[0]
print(close_match)

Iron Man
```

so these 3 movies are closest to the user input

finding the index of the movie with title

finding index of iron man movie using title.

```
# finding the index of the movie with title

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

68

68 is it index of Iron Man movie

by which we can find similar movies

now getting similarities score and comparing Iron Man movie with other 4800 movies

the high similarity score shows that movie we should recommend to user

getting a list of similar movies

```
[39] similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)

[(0, 0.033570748780675445), (1, 0.0546448279236134), (2, 0.013735500604224323), (3, 0.006468756104392058), (4, 0.03268943310073386), (5, 0.013907256685755473), (6, 0.076928375...)]

[40] len(similarity_score)

4803
```

enumerate () to run loop in a particular list and it will give you the count.

showing index of a movie and similarities score.

**sorting the movies based on their similarity score**

**order is according to data set we have to sort it**

**so now we are making it in descending order.**

Lambda shows score and reverse signifies descending order.

```
[41] # 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)

[(68, 1.0000000000000002), (79, 0.40890433998005965), (31, 0.31467052449477506), (7, 0.23944423963486405), (16, 0.22704403782296803), (26, 0.21566241096831154), (85, 0.20615862...)]
```

sort() for higher similarity value to lower


x is similarity score

A lambda function can take any number of arguments, but can only have one expression.

**print the name of similar movies based on the index**


[x]

for loop is used to get index value to find the title of movie.

```
3s  print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
    index = movie[0]
    title_from_index = movies_data[movies_data.index==index]['title'].values[0]
    if (i<30):
        print(i, '.',title_from_index)
        i+=1
```

 Movies suggested for you :

```
1 . Iron Man
2 . Iron Man 2
3 . Iron Man 3
4 . Avengers: Age of Ultron
5 . The Avengers
6 . Captain America: Civil War
7 . Captain America: The Winter Soldier
8 . Ant-Man
9 . X-Men
10 . Made
11 . X-Men: Apocalypse
12 . X2
13 . The Incredible Hulk
14 . The Helix... Loaded
15 . X-Men: First Class
16 . X-Men: Days of Future Past
17 . Captain America: The First Avenger
18 . Kick-Ass 2
19 . Guardians of the Galaxy
20 . Deadpool
21 . Thor: The Dark World
22 . G-Force
23 . X-Men: The Last Stand
24 . Duets
25 . Mortdecai
26 . The Last Airbender
27 . Southland Tales
28 . Zathura: A Space Adventure
29 . Sky Captain and the World of Tomorrow
```

finding title of movie by index

showing superheroes movies

now combining above cell without print command to make a movie recommendation system

## ❖ Movie Recommendation System

```
✓ 2s ▶ movie_name = input(' Enter your favourite movie name : ')

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

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)

close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]

similarity_score = list(enumerate(similarity[index_of_the_movie]))

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

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

i = 1

for movie in sorted_similar_movies:
    index = movie[0]
    title_from_index = movies_data[movies_data.index==index]['title'].values[0]
    if (i<30):
        print(i, '.',title_from_index)
        i+=1
```

## Result: DC movies

```
👤 Enter your favourite movie name : bat man
Movies suggested for you :

1 . Batman
2 . Batman Returns
3 . Batman & Robin
4 . The Dark Knight Rises
5 . Batman Begins
6 . The Dark Knight
7 . A History of Violence
8 . Superman
9 . Beetlejuice
10 . Bedazzled
11 . Mars Attacks!
12 . The Sentinel
13 . Planet of the Apes
14 . Man of Steel
15 . Suicide Squad
16 . The Mask
17 . Salton Sea
18 . Spider-Man 3
19 . The Postman Always Rings Twice
20 . Hang 'em High
21 . Spider-Man 2
22 . Dungeons & Dragons: Wrath of the Dragon God
23 . Superman Returns
24 . Jonah Hex
25 . Exorcist II: The Heretic
26 . Superman II
27 . Green Lantern
28 . Superman III
29 . Something's Gotta Give
```

➔ if we enter  
movie name  
Iron Man  
which is  
superhero  
movie

```
👤 Enter your favourite movie name : iron man
Movies suggested for you :

1 . Iron Man
2 . Iron Man 2
3 . Iron Man 3
4 . Avengers: Age of Ultron
5 . The Avengers
6 . Captain America: Civil War
7 . Captain America: The Winter Soldier
8 . Ant-Man
9 . X-Men
10 . Made
11 . X-Men: Apocalypse
12 . X2
13 . The Incredible Hulk
14 . The Helix... Loaded
15 . X-Men: First Class
16 . X-Men: Days of Future Past
17 . Captain America: The First Avenger
18 . Kick-Ass 2
19 . Guardians of the Galaxy
20 . Deadpool
21 . Thor: The Dark World
22 . G-Force
23 . X-Men: The Last Stand
24 . Duets
25 . Mortdecai
26 . The Last Airbender
27 . Southland Tales
28 . Zathura: A Space Adventure
29 . Captain and the World of Tomorrow
```

→ let give input **Avatar** movie which is space and fantasy related.

```
➡ Enter your favourite movie name : avatar
Movies suggested for you :

1 . Avatar
2 . Alien
3 . Aliens
4 . Guardians of the Galaxy
5 . Star Trek Beyond
6 . Star Trek Into Darkness
7 . Galaxy Quest
8 . Alien³
9 . Cargo
10 . Trekkies
11 . Gravity
12 . Moonraker
13 . Jason X
14 . Pocahontas
15 . Space Cowboys
16 . The Helix... Loaded
17 . Lockout
18 . Event Horizon
19 . Space Dogs
20 . Machete Kills
21 . Gettysburg
22 . Clash of the Titans
23 . Star Wars: Clone Wars: Volume 1
24 . The Right Stuff
25 . Terminator Salvation
26 . The Astronaut's Wife
27 . Planet of the Apes
28 . Star Trek
29 . Wing Commander
```

Link to reach Colab notebook for source code.

<https://colab.research.google.com/drive/1fTBPqfMZL-5JKZaLlsy6lpepnl-PoPn6#scrollTo=fC24koMrRQwX>

## **CONCLUSION**

The Cosine Similarity algorithm has been used to recommend the best movies that are related to the movie entered by the user based on different factors such as the genre of the movie, overview, the cast as well as the ratings given to the movie.

It is effective way to provide personalized movie recommendations to users.

it could be used to provide personalized recommendations to users based on their movie preferences and viewing history.

Hence this system will enhance the user experience and improve engagement on movie streaming platforms.



## **REFERENCES**

- periodically advise by Dr. **Ihtiram Raza Khan** sir
- some of my classmate for debugging in errors.
- Machine Learning Paperback by [Anuradha Srinivasaraghavan](#)
- idea of Hollywood movie by **Shahi Raza Khan**
- one of my senior Arpit Mittal
- Greek for Greek website
- YouTube

