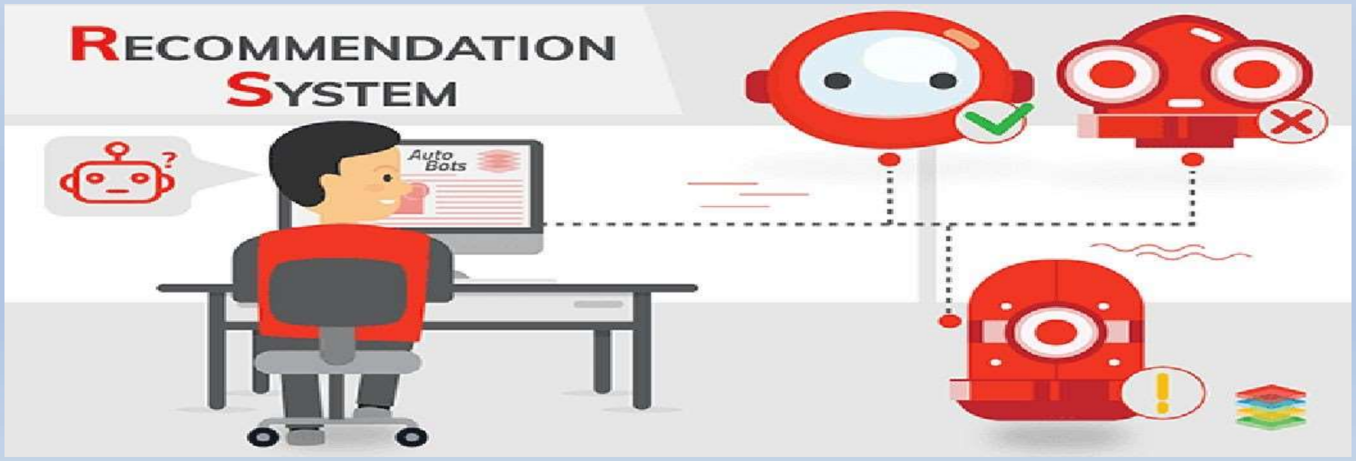
Movie Recommendation System

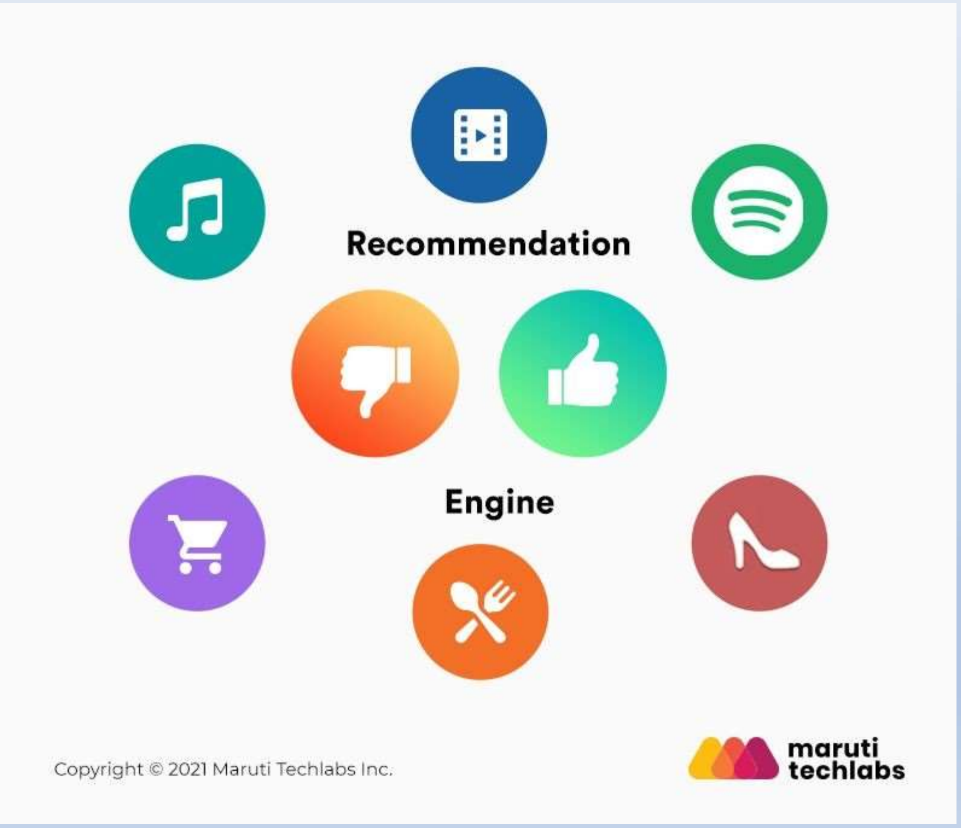
Introduction

Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general.

Movies are one of the sources of entertainment, but the problem is in finding the desired content from the ever-increasing millions of content every year.However, recommendation systems come much handier in these situations.The aim of this project is to build a movie recommendation system using - content based and Popularity approach.



Examples



- If a user listens to rock music every day, his youtube recommendation feed will get full of rock music and music of related genres.
- All the websites such as Amazon,Flipkart
- Uber also uses it(Suppose frequently you are using mini cab then it will recommend you to use that by default)
- Google also feeds new as the notification(Based on what you have searched or read in past).
- Netflix,Amazon Prime.

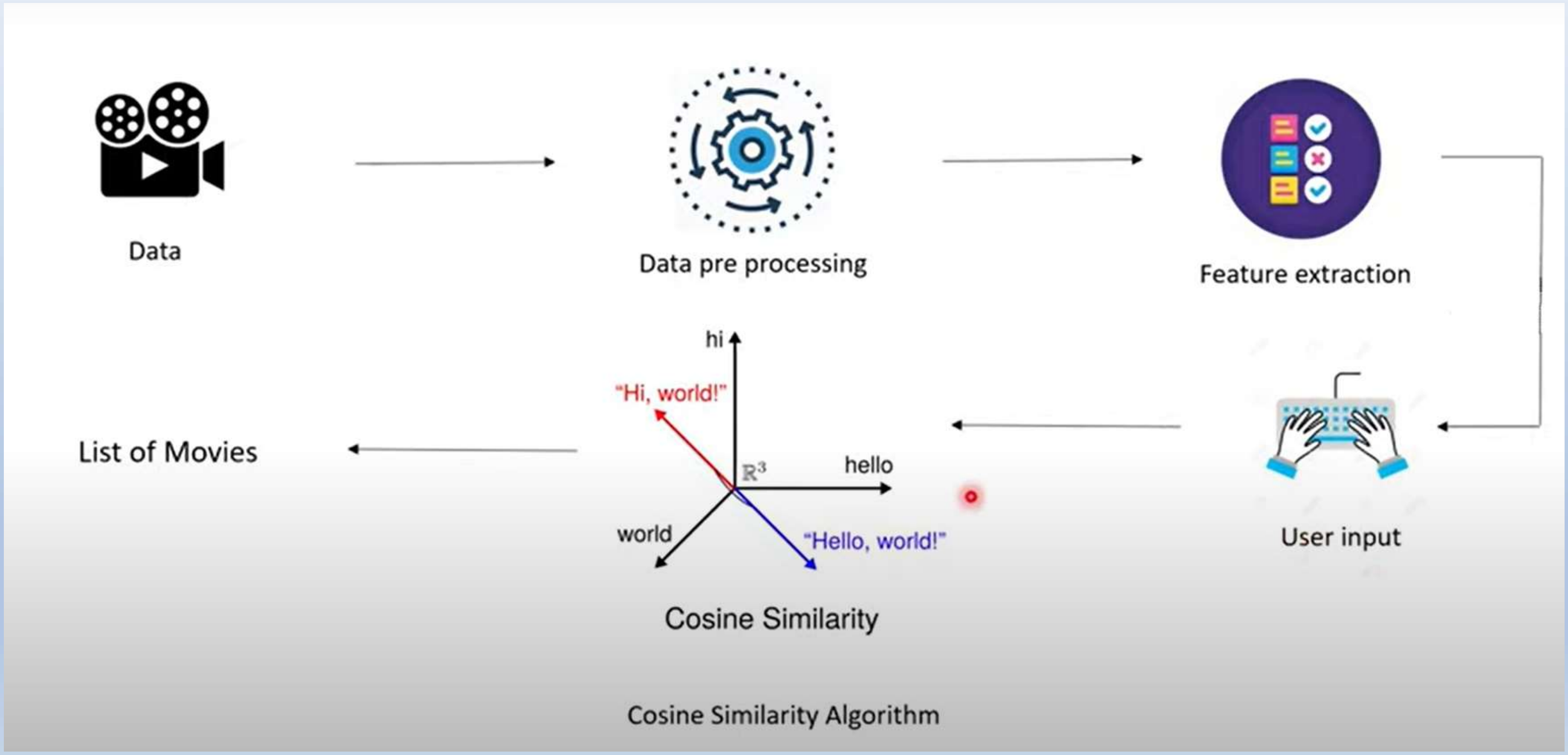
<u>APPROCHES..</u>

We can approach in 3 ways:

- ❑ **Content based recommendation system.**
- ❑ **Popularity based recommendation system.**
- ❑ **Collaborative recommendation system.**

ML model was created based with the help of the **content and popularity** based

Work Flow:


Data
Data pre processing
Feature extraction
User input
hi
"Hi, world!"
$\mathbb{R}^3$
hello
world
"Hello, world!"
Cosine Similarity
List of Movies
Cosine Similarity Algorithm

Working Principle

- Initially we collected a data containing 26 features like keyword, popularity, Directors,genre etc.,. Dataset plays a key role for any ML model since having more data helps to increase the accuracy of the model.

- Before extracting the features we have to replace the null spaces in the dataset with the null strings as a part of data processing.

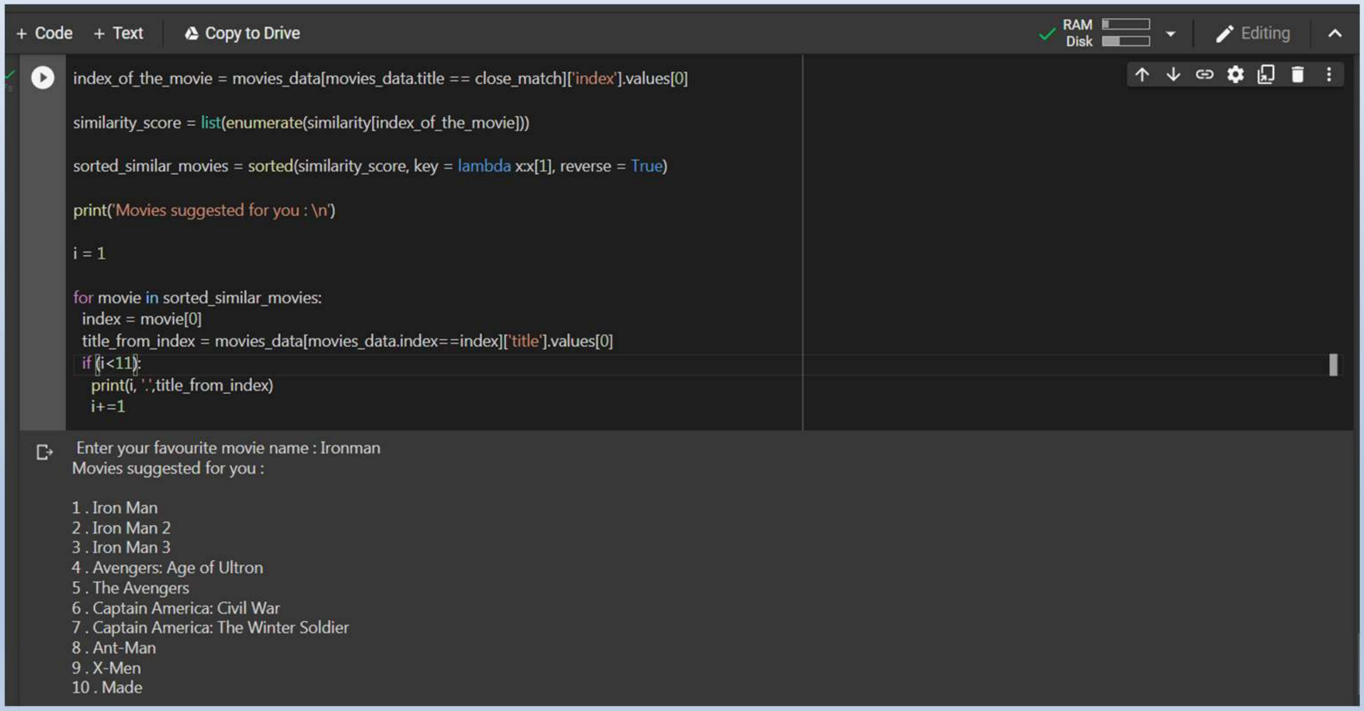
- User have to input the required features so that recommended movies will be based on those specefic features like based on director (or) based on the genres and also many multiple features as required .

- Based on the selected input feature, extracting the feature with the help of the **tfidfvectorizer.** The term tf–idf stands for term frequency–inverse document frequency. With the help of vectorizer we can convert the text data into the vectors.

CONT…

- Now we have to ask the input movie name by the user.
- And then creating a list consisting of all movie names after that with the help of the difflib we can find the close match for the movie name given by the user.
- With the help of the **cosine_similarity** we can estimate the similarity score of the individual movie and then sorting in a decreasing order of the similarity score.
- Finally printing the top 10 movie names which is having high similarity score.

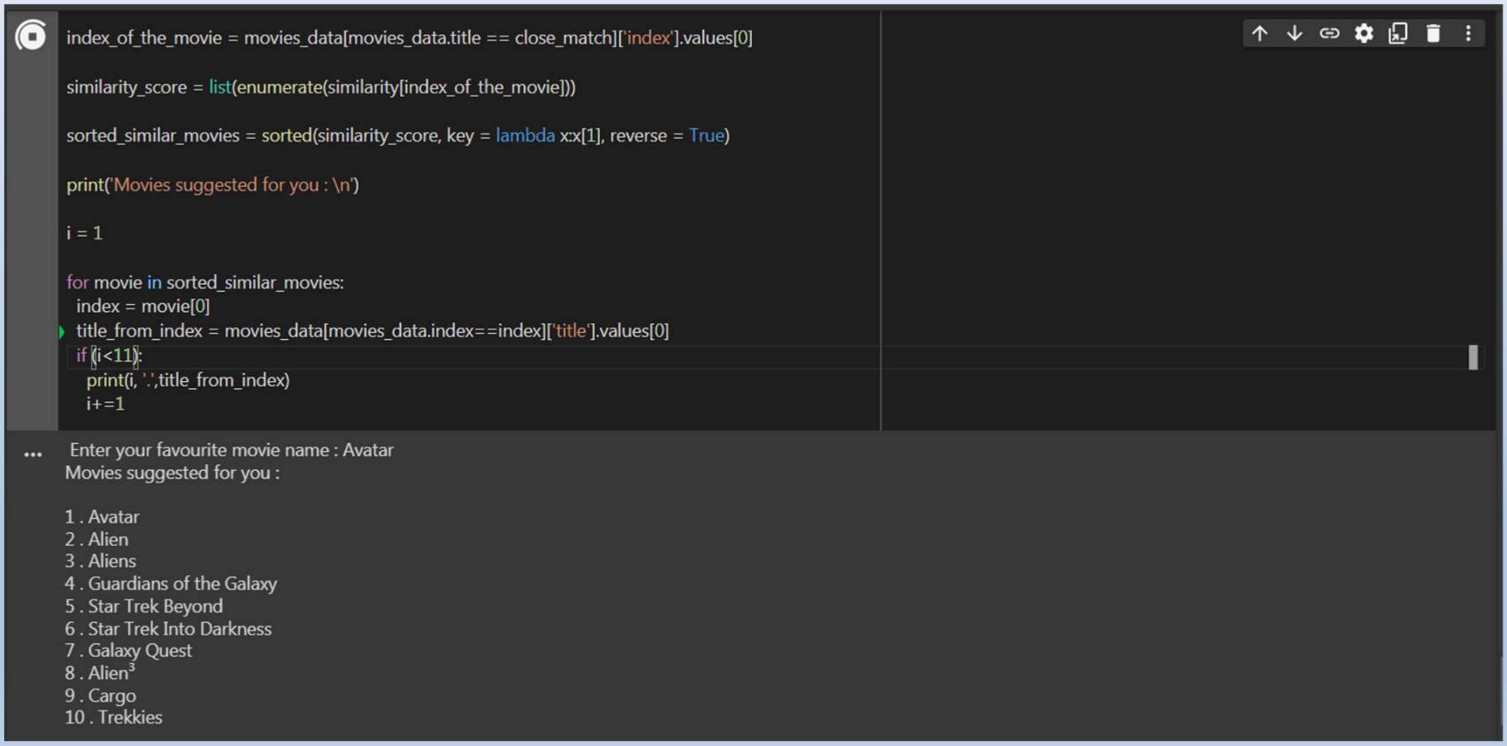
RESULTS:



```
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<11):
    print(i, '.',title_from_index)
    i+=1
```

```
Enter your favourite movie name : Ironman
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
```

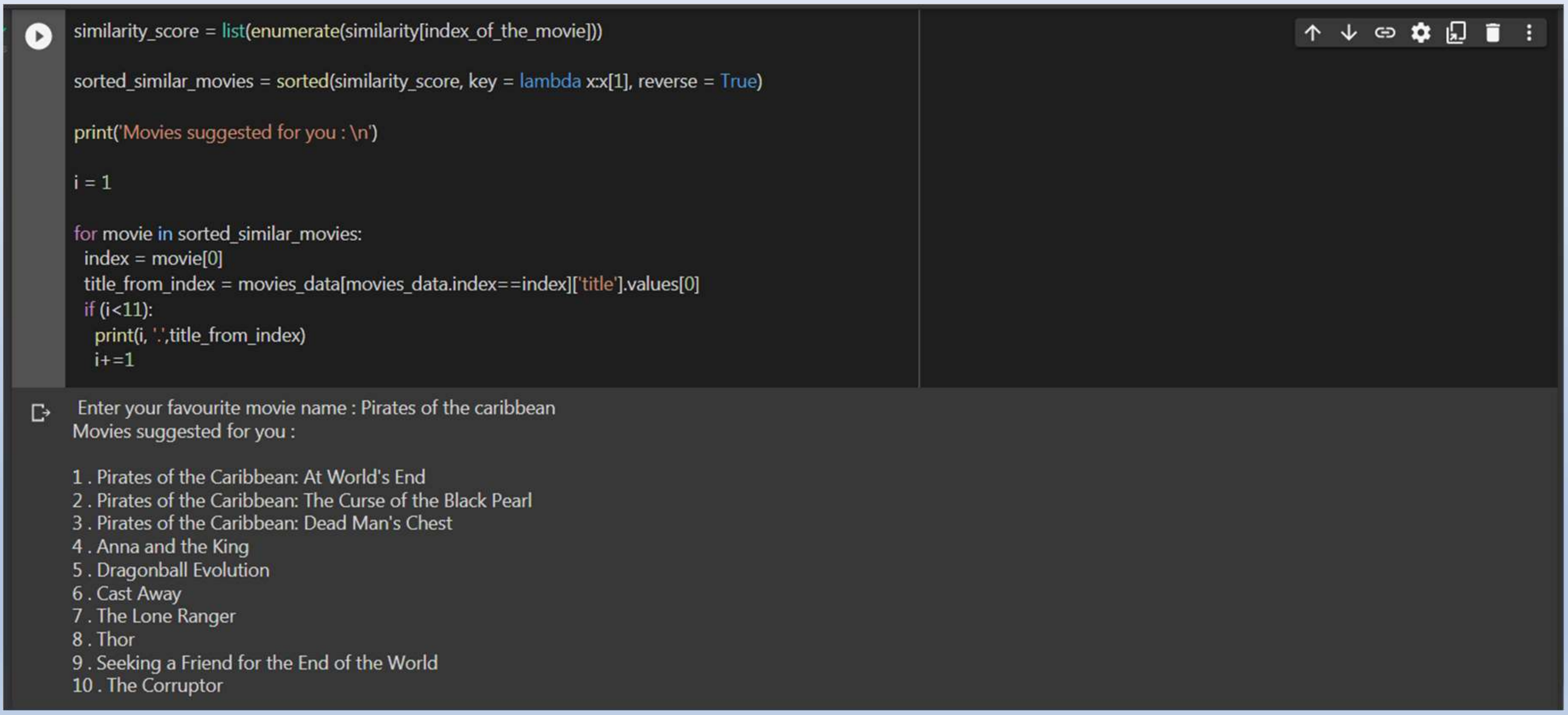
Cont..



```python
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<11):
  print(i, '.',title_from_index)
  i+=1
```

```
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
```

Cont..



```
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<11):
    print(i, '.',title_from_index)
    i+=1
```

```
 Enter your favourite movie name : Pirates of the caribbean
Movies suggested for you :

1 . Pirates of the Caribbean: At World's End
2 . Pirates of the Caribbean: The Curse of the Black Pearl
3 . Pirates of the Caribbean: Dead Man's Chest
4 . Anna and the King
5 . Dragonball Evolution
6 . Cast Away
7 . The Lone Ranger
8 . Thor
9 . Seeking a Friend for the End of the World
10 . The Corruptor
```

Conclusion:

Content Based Recommender: the one that took movie overview,taglines, cast, crew, genre and keywords as input and come up with predictions. We also devised a simple filter to give greater preference to movies with more votes and higher ratings.

We have developed the Machine Learning model based on **content and the popularity**.