GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

# Movie Recommendation System using Machine Learning

**A Project Report of Capstone Project-2**

***Submitted by***

**Robin Sharma**

**(1613106009)**

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING WITH SPECIALIZATION IN OPEN SOURCE SOFTWARE AND OPEN STANDARDS**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the Supervision of**

**Dr. SANSAR SINGH CHAUHAN**

**Professor**

**APRIL / MAY- 2020**

GALGOTIAS UNIVERSITY

# SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

# BONAFIDE CERTIFICATE

Certified that this project report **"MOVIE RECOMMENDATION SYSTEM USING MACHINE LEARNING"** is the bonafide work of **"ROBIN SHARMA (1613106009)"** who carried out the project work under my supervision.

**SIGNATURE OF HEAD**

Dr. MUNISH SHABARWAL

PhD (Management), PhD (CS)

**Professor & Dean**

**School of Computing Science & Engineering**

**SIGNATURE OF SUPERVISOR**

Dr. SANSAR SINGH CHAUHAN

M. Tech , PhD

**Professor**

**School of Computing Science & Engineering**

# TABLE OF CONTENTS

# ABSTRACT

This analysis mainly focuses on the domain of Machine Learning example of Movie Recommendation System with a approach of finding the similarity scores between the two contents in the content based filtering. It helps us in finding the distance between the two vectors and their angle by the help of the cosine similarity formula and their magnitude of relative scores. In this model, we have considered two texts and plotted them in the form of a graph from where we have plotted the points taking two-dimension plane of x-y plane only. We have taken in consideration two fields (i.e.) in x-plane we have considered the word London of the text whereas in y-plane we have considered Paris word of the texts. This model analysis the points to find the two texts similarity scores by using Python shell and distance between two vector model approach more effectively and precisely.

In today computer world we have lots of stuff on our Internet sources to watch and see but every single stuff available not matches with our liking. We sometimes get the feed of the videos, vies, news, clothing etc. which is not according to our liking and interest. It makes the customer interest in the application lowest and he/she further doesn't want to get through the same application again. The need for the hour is to develop some code which can tell at a beginner level the matching pattern of the customer trend and recommend him with the best item of his interest level. This will help us in making the customer experience satisfactory and able to achieve good ratings and popularity as well.

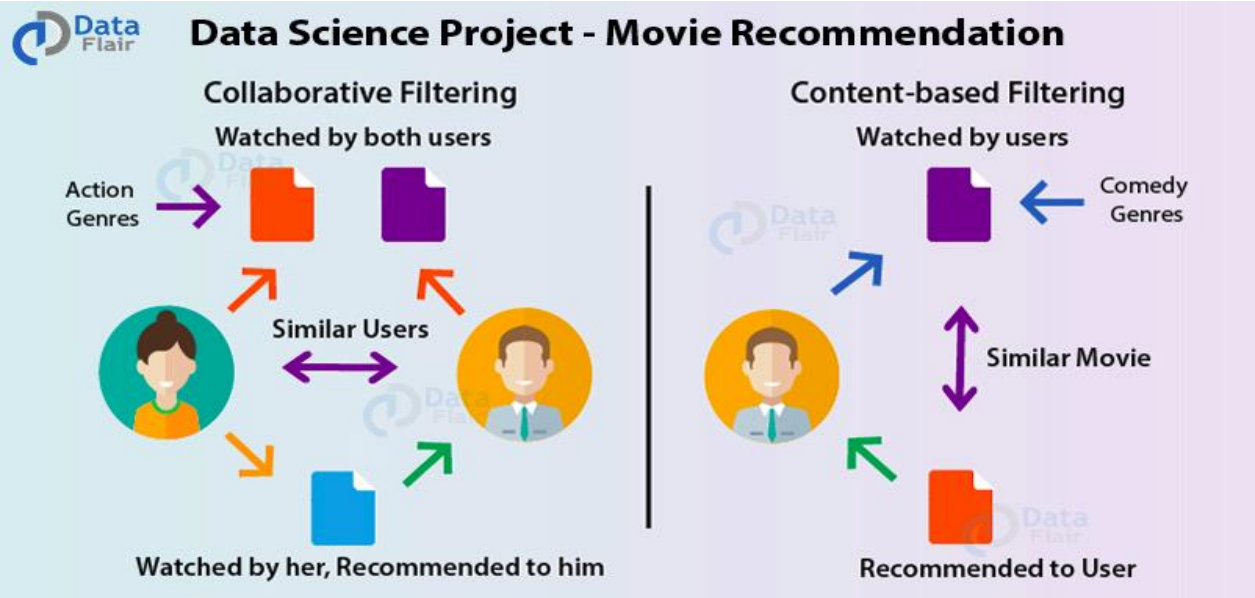
Many of the recommendation systems we are seeing today in our environment such as in the YouTube for example if I see lots of news regarding the GK and Current Affairs then it offers me the related videos according to it with different subscribers. It gains popularity by application rating and at the same time enhances the customer experience. This policy of recommendation system is really helpful in giving optimum results to an application profitability and to make the organisation more connected.

We can also see the recommendation work in online food applications such as Zomato, Food Panda and Swiggy which offers their customers the food restaurants which supplies their taste food. They learn upon the behaviour of the customer from the previous orders and tries to impress them with the latest add-ons of their favourite cuisines and stuffs.

It is been investigated from decade towards our needs that most of the ML implementation algorithms and other aspects are been governed by the recommendation engines to compute the similarity scores. This research paper takes upon the consideration a graph based model, which is quite affordable and easy to understand by software developer ,where we have taken two Texts A and B and try to find out the distance between these two words of texts "London" and "Paris" on X-axis and Y-axis respectively.

# INTRODUCTION

Supervised Machine Learning is when the model is getting trained on a labelled dataset. Labelled dataset is one which have both input and output parameters. In this type of learning both training and validation datasets are labelled Here, we have three components such as Training Data, Test Data and features. Training data is that where data is usually split in the ratio of 80:20 i.e. 80% as training data and rest as testing data. Testing data is that when data is good to be tested. At the time of testing, input is fed from remaining 20% data which the model has never seen before, the model will predict some value and we will compare it with actual output and calculate the accuracy.




Three different types of Machine Learning methods are as follows-

1.) Data Extraction and Cleaning

This is been used to extract and clean the data by using scripting languages such as Python, Shell Scripting etc. Here we extract and filter the useful data of movies datasets according to our need.

2.) Build ML Model

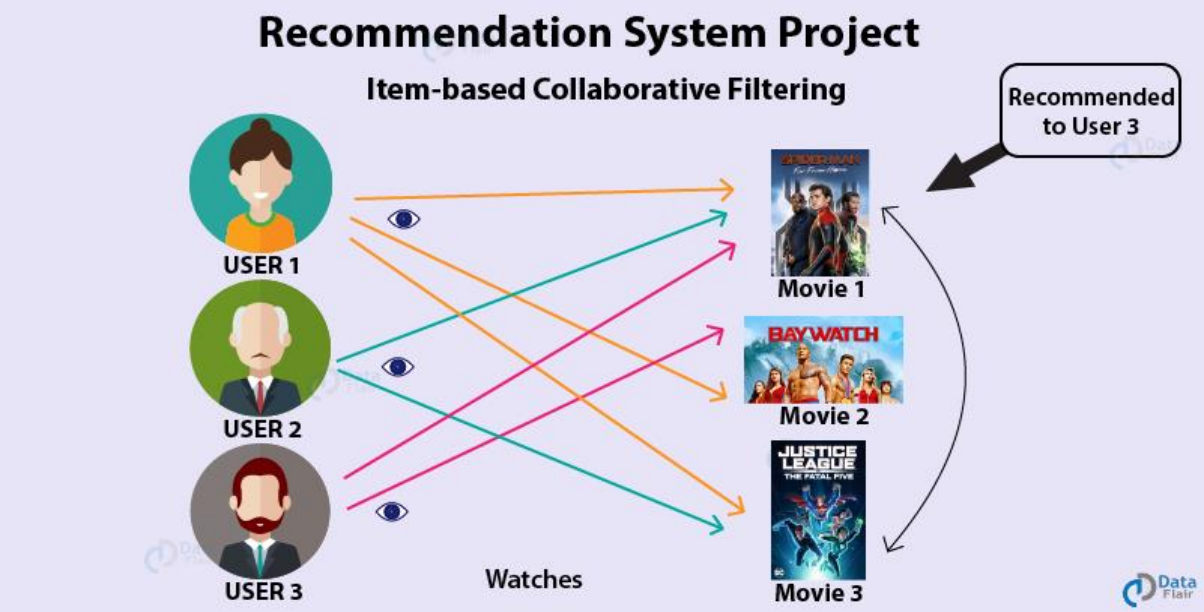
Once we will extract and clean data will start building up the model with tools such as Tensor Flow, Azure ML etc. We build our movie recommendation engine here which would be in form of a Python script.

3.) Build Software Infrastructure

Here we have to build ML components such as a product for the users by using the ML algorithms in the form of a software by using JavaScript. A little knowledge of cloud infrastructure such as AWS (Amazon Web Services) and to collaborate with people a little knowledge of GitHub is also known

Users- They are the one who uses these services or acts as the consumer.

Items-Here these are the different sets of movies which are been recommended in a sort of zig-zag manner according to our previous searches




Three ways of performing the filtering are as follows-

- Trending based filtering- Here the movies are been classified by the ratings and the stuff that is been liked by a majority of the population.
- Content based filtering-Here the similar articles are been recommended to the user according to his previous content search.
- Collaborative based filtering-Here the two similar user likings act as a recommender to each other. Like, we two users watch comedy movies so if a new comedy stuff appears and is watched by A user it will also be recommended to user B.

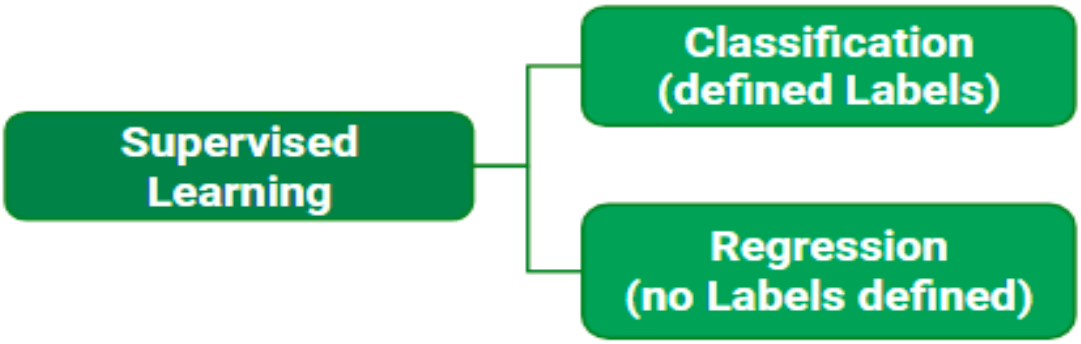
A recommender system, or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item They are primarily used in commercial applications. Recommender systems are utilized in a variety of areas and are most commonly recognized as playlist generators for video and music services like Netflix, YouTube and Spotify, product recommenders for services such as Amazon, or content recommenders for social media platforms such as Facebook and Twitter. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators, and financial services

These both are the recommendation engines that recommend us the movies and other related stuff based on our previous searches and watched experience.


.

.

- Classification: It is a Supervised Learning task where output is having defined labels (discrete value).
- Regression: It is a Supervised Learning task where output is having continuous value.



# EXISTING SYSTEM

- We use Digital libraries for a wide variety of digital objects (research papers, publications, journals, research projects, newspapers, magazines, and past questions).

- But some digital libraries even offer millions of digital objects. Therefore, getting or finding favourite digital objects from a large collection of available digital objects in the digital library is one of the major problems.

- The users need help in finding items that are in accordance with their interests.

- Recommender systems offer a solution to this problem as library users will get recommendations using a form of smart search.

- The problem considered here is to develop recommendation to find a particular variety of the objects to user (like publications, research paper etc.) only in a large collection of items.

Recommender systems are software applications that suggest or recommend items or products (in the case of ecommerce) to users. These systems use users' preferences or interests (supplied as inputs) and an appropriate algorithm in finding the relevant or desired items or products. Recommender systems deal with information overload problems by filtering items that potentially may match the users' preferences or interests. These systems aid users to efficiently overcome the problem by filtering irrelevant information when users search for desired information.

Content-based recommenders provide recommendations by comparing representation of contents describing an item or a product to the representation of the content describing the interest of the user (User's profile of interest). They are sometimes referred to as content-based filtering. The content-based technique is adopted or considered here for the design of the recommender system for digital libraries. Content-based technique is suitable in situations or domains where items are more than users.

The use of collaborative-filtering technique in recommending research papers has been criticized by some authors. Authors like [15] suggest that collaborative-filtering technique is ineffective in domains where items (e.g. research papers) are more than users. [16] Said; "Users are not willing to spend time to rate items explicitly". Hence, content-based approach is adopted for the design and implementation of research paper recommender system. This approach does not depend on the ratings of other users but uses the contents describing the items and the users' taste or needs. The researchers used the following data collection procedure and methods in representing the research papers, users' profile of interest, and also in providing recommendations to the users.

The Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The researchers used this method to determine how similar a research paper is to a user's query or paper that a user has liked in the past. The research papers are represented as vectors of weights, where each weight indicates the degree of association between the research papers and the term.

It is helpful in implementing the model with cosine similarity formula and recommend the string similar to the user choice by analysing his patterns of writing with that of the stated other given string.

When two or more movies of some different genres come it becomes hard for Sublime Text to understand.

- It gives the results of similar movies related to my applied set of movies.
- It is confined of providing the results with the help of cosine similarity formula and count vectorizer.
- It provides the output of movies with the help of datasets governed through csv files.
- Not confined to provide advance hybrid string matching elements solutions and responses.

# PROPOSED SYSTEM

- To find out the related content of the user in a given set of collection and to his interest feed.

- Providing the accurate and most confined results by using the distance between two vectors model and graph plotting examples.

- Using the cosine similarity formula for finding the similarity scores/matching of the two texts more accurately.

- By providing accurate results by the usage of movies datasets by IMDB in the existing project work.

- Classifying the users interest movies and recommend them to their searches fast by help of libraries and software as pip, panda etc.

Keyword-Based Vector-Space Model: The researchers used this model with basic TF-IDF weighing technique to represent a research paper as a vector of weights, where each weight indicates the degree of association between a research paper and a term or keyword. • Item Representation: The items (research papers) are represented by a set of features (also called attributes or properties). These attributes are: title of the paper, abstract, keywords, research area, ID of the paper, and the authors. The abstract represents the research paper when the frequency of a term in the research paper is being determined. Recommender systems are software applications that provide or suggest items to intended users. These systems use filtering techniques to provide recommendations. The major ones of these techniques are collaborative-based filtering technique, content-based technique, and hybrid algorithm. The motivation came as a result of the need to integrate recommendation feature in digital libraries in order to reduce information overload. Content-based technique is adopted because of its suitability in domains or situations where items are more than the users. TF-IDF (Term Frequency Inverse Document Frequency) and cosine similarity were used to determine how relevant or similar a research paper is to a user's query or profile of interest. Research papers and user's query were represented as vectors of weights using Keyword-based Vector Space model. The weights indicate the degree of association between a research paper and a user's query. This paper also presents an algorithm to provide or suggest recommendations based on users' query. The algorithm employs both TF-IDF weighing scheme and cosine similarity measure. Based on the result or output of the system, integrating recommendation feature in digital libraries will help library users to find most relevant research papers to their needs

Step 1: Install Python

If you do not have Python installed on your computer.

Install the latest version of Python from Web.

Step 2: Download the pip package manager for Python

Once you have installed Python from the instructions above.

We need to install some libraries which we are going to use in our workshop. We will install Numpy, Matplotlib and Pandas to work with our datasets.

pip is a package management system used to install and manage software packages written in Python.

Right Click the following link and select Save Link As

(Save Target As): https://bootstrap.pypa.io/get-pip.py

Go to the folder where you saved this file. In windows explorer use Shift + Right Click and then select Open command window here to open command prompt in this directory. Then run the following command:

```
python get-pip.py
```

Step 3: Install Libraries

Open Command Prompt.

Run the following command to install necessary libraries.

```
pip install numpy matplotlib pandas scikit-learn gym opencv-python
```

If the installation completes without any errors, you are all set!

# IMPLEMENTATION OF CODING DIAGRAMS

- Step1: Read CSV file of dataset.
- Step2: Select features of datasets
- Step3: Create a column in DF which contains all selected features
- Step 4: Create count matrix from this new combined column
- Step5: Compute the cosine similarity based on count matrix
- Step6: Get index of this movie from title
- Step7: We will get the list of similar movies in descending order of similarity score
- Step8: Lastly, Print title of first 15 movies

- Step1: Read CSV file of dataset

```
12
13  ##Step 1: Read CSV File
14  df = pd.read_csv("movie_dataset.csv")
15  print df.head()
16
17  ##Step 2: Select Features
18
19  ##Step 3: Create a column in DF which combines all selected features
20
21  ##Step 4: Create count matrix from this new combined column
22
23  ##Step 5: Compute the Cosine Similarity based on the count_matrix
24
25  movie_user_likes = "Avatar"
```

```
   index  ...           director
0      0  ...      James Cameron
1      1  ...     Gore Verbinski
2      2  ...         Sam Mendes
3      3  ...  Christopher Nolan
4      4  ...     Andrew Stanton

[5 rows x 24 columns]
[Finished in 3.5s]
```

- Step2: Select features of datasets

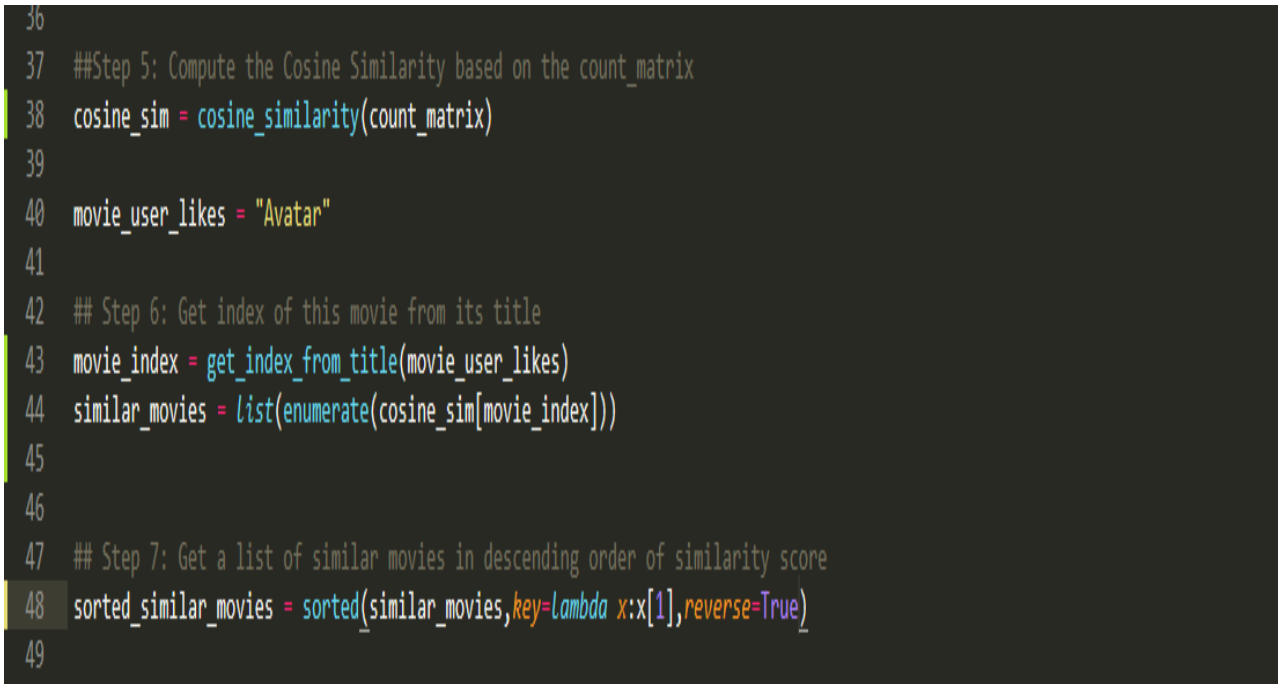
- Step3: Create a column in DF which contains all selected features


- Step 4: Create count matrix from this new combined column

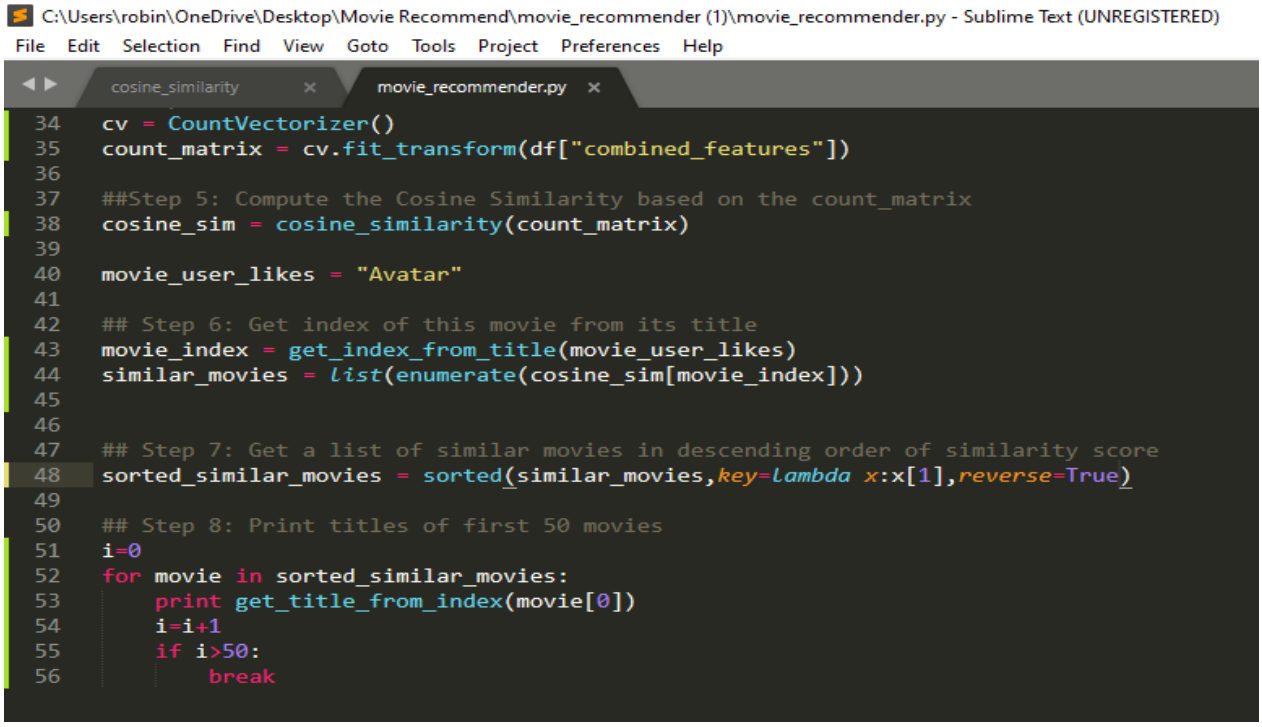

- Step5: Compute the cosine similarity based on count matrix

- Step6: Get index of this movie from title


- Step7: We will get the list of similar movies in descending order of similarity score



- Step8: Lastly, Print title of first 15 movies




I have taken Imdb datasets of movies here for building this project https://datasets.imdbws.com/.

- Each dataset is contained in a gipped, tab-separated-values (TSV) formatted file in the UTF-8character set. The first line in each file contains headers that describe what is in each column. A '\N' is used to denote that a particular field is missing or null for that title/name. The available datasets are as follows:
- name.basics.tsv.gz
- title.akas.tsv.gz
- title.basics.tsv.gz
- title.crew.tsv.gz
- title.episode.tsv.gz
- title.principals.tsv.gz
- title.ratings.tsv.gz

# OUTPUT/RESULT/SCREENSHOT

C:\Users\robin\OneDrive\Desktop\Project Files and Datasets\movie_recommender2(sublime text).py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

movie_recommender2(sublime text).py

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
###### helper functions. Use them when needed #######
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
##################################################

##Step 1: Read CSV File
df = pd.read_csv("movie_dataset.csv")
print df.head()

##Step 2: Select Features
features = ['keywords','cast','genres','director']

##Step 3: Create a column in DF which combines all selected features
for feature in features:
    df[feature] = df[feature].fillna('')
    def combine_features(row):
        try:
            return row["keywords"]+" "+row["cast"]+" "+row["genres"]+" "+row["director"]
        except:
            print "Error:" , row


    df["combined_features"] = df.apply(combine_features,axis=1)
    print "combined features:", df["combined_features"].head()

##Step 4: Create count matrix from this new combined column
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
```

C:\Users\robin\OneDrive\Desktop\Project Files and Datasets\movie_recommender2(sublime text).py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

movie_recommender2(sublime text).py

```
            print "Error:" , row


    df["combined_features"] = df.apply(combine_features,axis=1)
    print "combined features:", df["combined_features"].head()

##Step 4: Create count matrix from this new combined column
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])

##Step 5: Compute the Cosine Similarity based on the count_matrix
cosine_sim = cosine_similarity(count_matrix)

movie_user_likes = "Avatar"

## Step 6: Get index of this movie from its title
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index]))


## Step 7: Get a list of similar movies in descending order of similarity score
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)

## Step 8: Print titles of first 50 movies
i=0
for movie in sorted_similar_movies:
    print get_title_from_index(movie[0])
    i=i+1
    if i>50:
        break
```

```
Avatar
Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
Alien
Lockout
Jason X
The Helix... Loaded
Moonraker
Planet of the Apes
Galaxy Quest
Gravity
Alien³
```

C:\Users\robin\OneDrive\Desktop\Movie Recommend\movie_recommender (1)\movie_recommender.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

cosine_similarity × movie_recommender.py ×

```
45
46
47  ## Step 7: Get a list of similar movies in descending order of similarity score
48  sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)
49
50  ## Step 8: Print titles of first 50 movies
51  i=0
52  for movie in sorted_similar_movies:
53      print get_title_from_index(movie[0])
54      i=i+1
55      if i>50:
56          break
```

```
combined features: 0    culture clash future space war space colony so...
1    ocean drug abuse exotic island east india trad...
2    spy based on novel secret agent sequel mi6 Dan...
3    dc comics crime fighter terrorist secret ident...
4    based on novel mars medallion space travel pri...
Name: combined_features, dtype: object
Avatar
Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
Alien
Lockout
Jason X
The Helix... Loaded
Moonraker
Planet of the Apes
```

Line 48, Column 77

Type here to search

# CONCLUSION/FUTURE ENHANCEMENT

- In this project we have implemented and learn the following things such as-
- Building a Movie Recommendation System
- To find the Similarity Scores and Indexes.
- Compute Distance Between Two Vectors
- Cosine Similarity
- To find Euclidian Distance and many more ML related concepts and techniques.

Research paper recommender systems help library users in finding or getting most relevant research papers over a large volume of research papers in a digital library. This paper adopted content-based filtering technique to provide recommendations to the intended users. Based on the results of the system, integrating recommendation features in digital libraries would be useful to library users.

The solution to this problem came as a result of the availability of the contents describing the items and users' profile of interest. Content-based techniques are independent of the users ratings but depend on these contents. This paper also presents an algorithm to provide or suggest recommendations based on the users' query. The algorithm employs both TF-IDF weighing and cosine similarity measure

The next step of our future work is to adopt hybrid algorithm to see how the combination of collaborative and content-based filtering techniques can gives us a better recommendation compared to the adopted technique in this paper.

The content-based technique is adopted or considered here for the design of the recommender system for digital libraries. Content-based technique is suitable in situations or domains where items are more than users.

Library users do experience difficulties in getting or finding favourite digital objects (e.g. research papers) from a large collection of digital objects in digital libraries.

# REFERENCES


- https://drive.google.com/file/d/1sJ9N.
- https://pdfs.semanticscholar.org/c9f9/6d22422953625f1f8d9dbee221cba38e6c08.pdf
- https://www.google.com/search?sxsrf=ALeKk00P037U5bp3y51QqWE-wCkvkCDUKw:1588619431266&source=univ&tbm=isch&q=content+based+filtering+diagrams&sa=X&ved=2ahUKEwi33YjH9JrpAhWFXSsKHR56DOAQsAR6BAgKEAE&biw=1366&bih=625
- https://www.imdb.com/list/ls063596142/
- https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52?gi=61b501c11dd4
- https://www.google.com/search?q=numpy+in+python&oq=numpy+in+python&aqs=chrome..69i57j0l6j69i60.4625j0j7&sourceid=chrome&ie=UTF-8
- https://www.google.com/search?q=pip+python&oq=pip&aqs=chrome.2.69i57j0l6j69i60.3726j0j7&sourceid=chrome&ie=UTF-8
- https://www.google.com/search?q=sublime+text+python&oq=sublime+&aqs=chrome.2.69i57j35i39l2j0j69i60l4.4504j0j7&sourceid=chrome&ie=UTF-8
- https://www.google.com/search?q=cosine+similarity+python&oq=cosine&aqs=chrome.5.69i57j0l7.6310j0j7&sourceid=chrome&ie=UTF-8
- https://docs.python.org/3/tutorial
- https://en.wikipedia.org/wiki/Recommender_system
- https://www.learnpython.org/en/Pandas_Basics
- https://www.google.com/search?q=sublime+text+python&oq=sublime+&aqs=chrome.2.69i57j35i39l2j0j69i60l4.4504j0j7&sourceid=chrome&ie=UTF-8


Yours Sincerely,

Robin Sharma

(1613106009)