**COLLEGE OF ENGINEERING, GUINDY**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**PROJECT**

**MOVIE RECOMMENDATION**

**USING AIRTIFICIAL INTELLIGENCE**

**Done by**

1. **Vijai Kumar S 2020178060**

# Abstract:

In this project, we proposed a solution to recommend a movie to the users based on their interest. We have used content based filtering to recommend movies to the users based on their watch history, favourite actor, actress and directors and finally we sort it and give it to the users based on the cosine similarity index. This proposed solution will help users to find their favourite movies quickly without spending time to search and find. It also helps OTT platform to better engage with their customers

# Introduction

OTT platforms are gaining popularity among people especially during the period of covid. Delivering what user wants is a key for a business to success. ML provides various types of recommendation system. Choosing the right one will help organization to grow quickly. There are majorly six types of recommender systems which work primarily in the Media and Entertainment industry: **Collaborative Recommender system, Content-based recommender system, Demographic based recommender system, Utility based recommender system, and knowledge based recommender system and Hybrid recommender system.** In this project we have used content based recommendation system to recommend top 7 movies that the users will mostly like to watch depending upon the previous watch history

# Dataset:

We have collected dataset from kaggle and we have utilised dataset of 1000 Hollywood movies. We performed exploratory data analysis and we found that there are no missing or null values and we are using count vectorizer to convert the text into numerical values (matrix) and cosine similarity to find the similarity score of the different movies

Following data from the dataset have been collected

1. Actors
2. Directors
3. Genre
4. Title
5. Description
6. Director
7. Year
8. Ratings

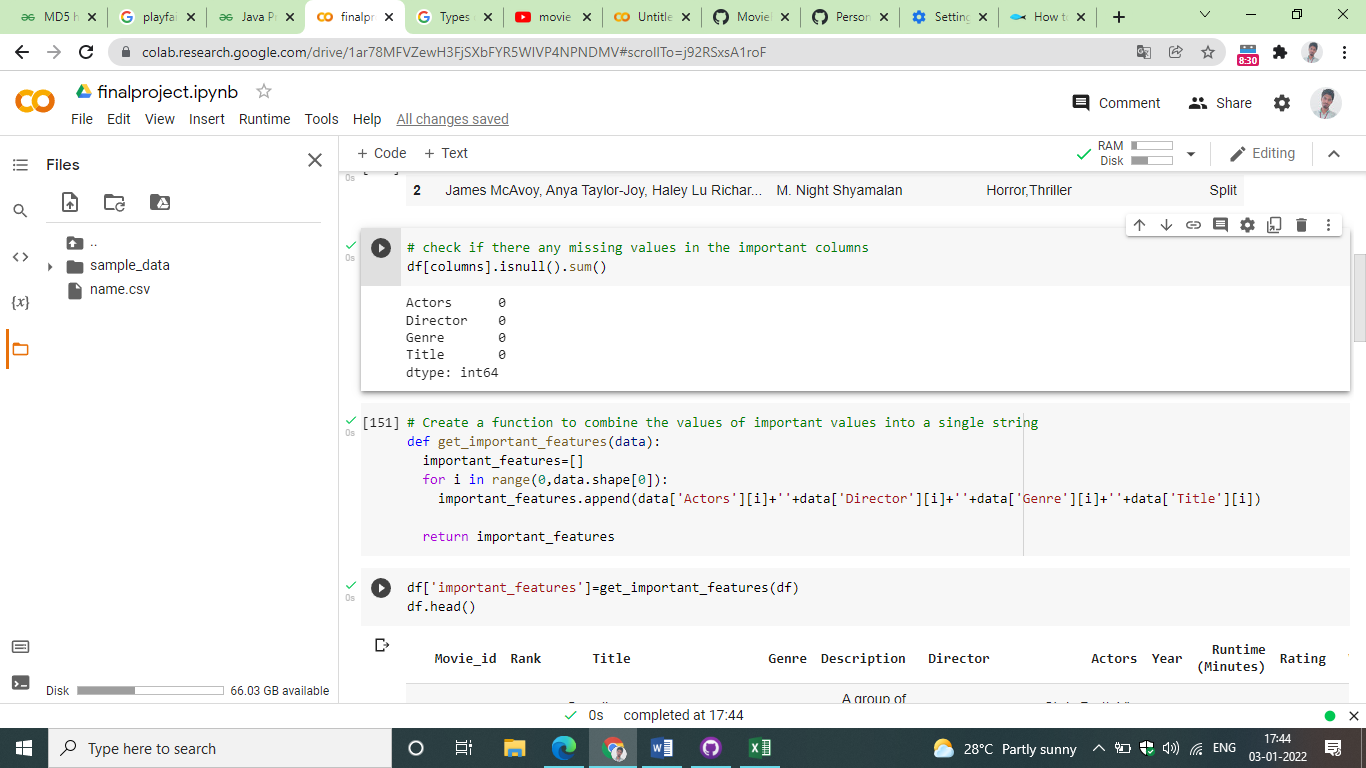


Figure : Checking for null

We have append all important columns and used countvectorizer to transform into matrix

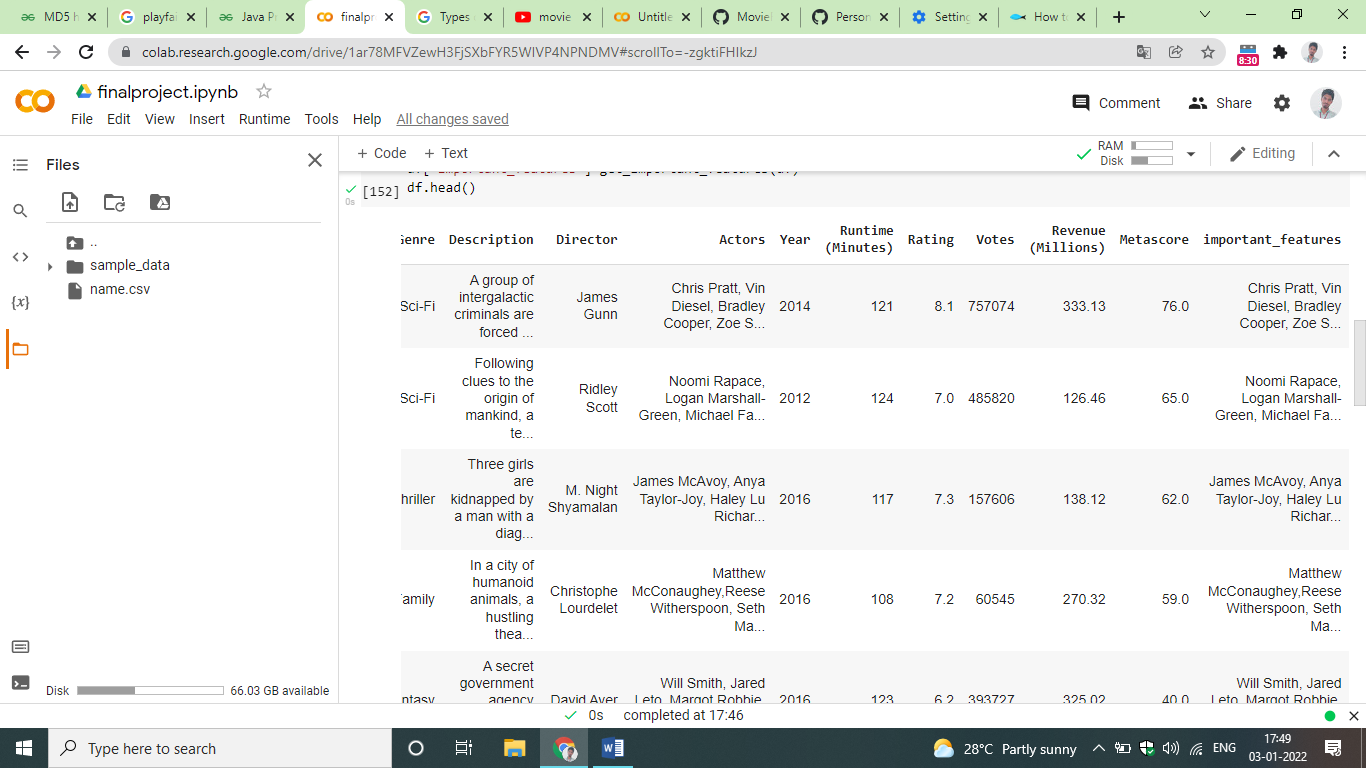


Figure Parsing Important features

**Cosine Similarity**

We have taken cosine similarity for the input movie and parse top 7 movies that have high similarity score and sorted it in a descending order

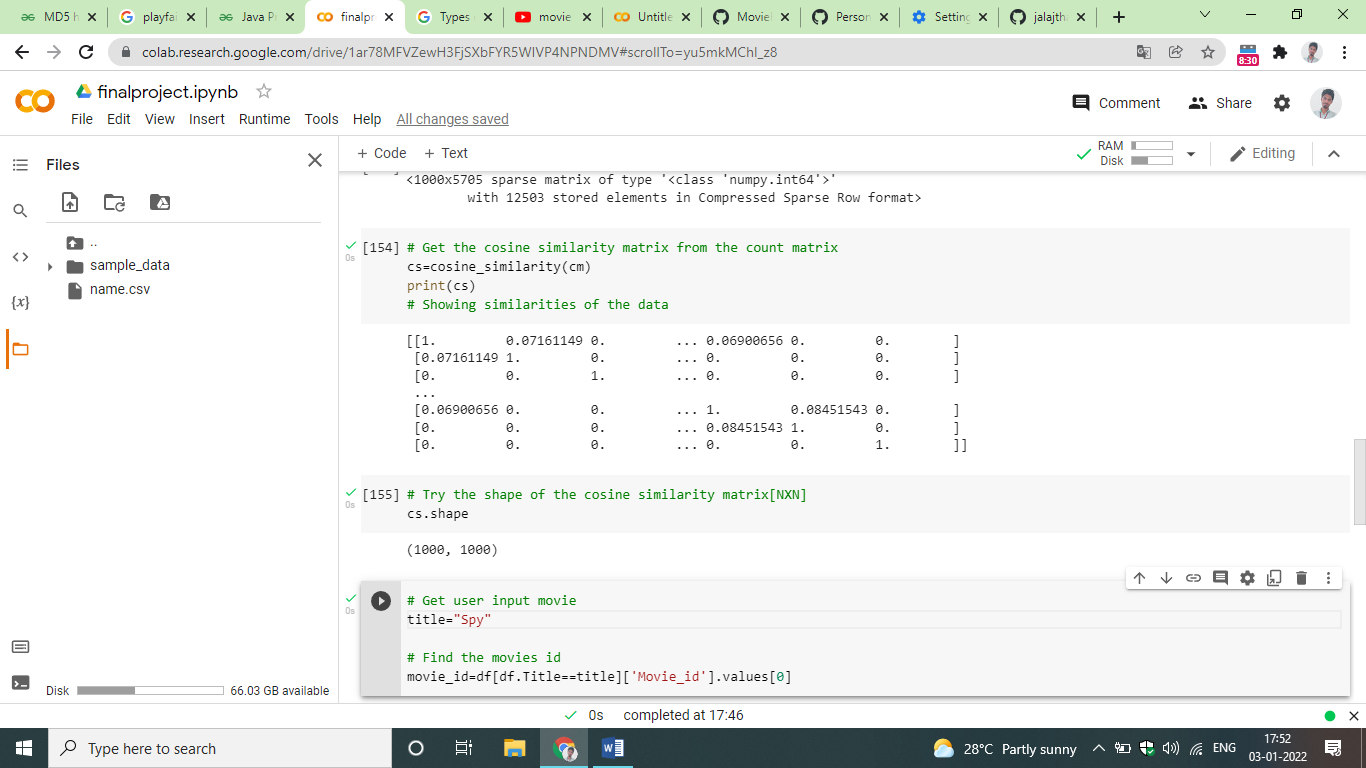


Figure 3: cosine similarity

# Formula:

**Cosine similarity** is a metric, helpful in determining, how similar the data objects are irrespective of their size. We can measure the [similarity between two sentences in Python](https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/) using Cosine Similarity. In cosine similarity, data objects in a dataset are treated as a vector. The formula to find the cosine similarity between two vectors is –

**Cos(x, y) = x . y / ||x|| \* ||y||**

where,

* **x . y** = product (dot) of the vectors ‘x’ and ‘y’.
* **||x||**and**||y||** = length of the two vectors ‘x’ and ‘y’.
* **||x|| \* ||y||** = cross product of the two vectors ‘x’ and ‘y’.

**Count Vectorizer:**

**CountVectorizer**is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).

# Analysis and Conclusion

Once we performed all the analysis we found that the top 7 movies similar to user interest have been shown. However this system can be improved by incorporating additional type of recommendation like collaborative filtering and including additional fields into the training dataset

# References:

[1]. Ashirta Kashyap , A Movie Recommender system – MOVREC using machine learning techniques

[2]. Bei-Bei Cui, Design and Implementation of Movie Recommendation System based on collaborative Filtering, 2017

[3].Dataset - <https://www.kaggle.com/gorochu/complete-imdb-movies-dataset>

**Coding**

import pandas as pd

import numpy as np

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import CountVectorizer

df=pd.read\_csv('name.csv')

df.shape

# create a list of important columns for the recommendation engine

columns=['Actors','Director','Genre','Title']

# show the necessary data only

df[columns].head(3)

# check if there any missing values in the important columns

df[columns].isnull().sum()

# Create a function to combine the values of important values into a single string

def get\_important\_features(data):

  important\_features=[]

  for i in range(0,data.shape[0]):

    important\_features.append(data['Actors'][i]+''+data['Director'][i]+''+data['Genre'][i]+''+data['Title'][i])

  return important\_features

df['important\_features']=get\_important\_features(df)

df.head()

cm=CountVectorizer().fit\_transform(df['important\_features'])

cs=cosine\_similarity(cm)

print(cs)

# Get user input movie

title="Spy"

# Find the movies id

movie\_id=df[df.Title==title]['Movie\_id'].values[0]

# Create a  list of enumerations for the similarity scores(Rank[0], Similarity Score[1])

scores=list(enumerate(cs[movie\_id]))

# Sort the list

# If we put X[0] then sorting will be based on rank rather than similarity score

sorted\_scores=sorted(scores,key=lambda x: x[1],reverse=True)

sorted\_scores=sorted\_scores[1:]

j=0

print("The 7 Most Recommended movies to",title," likes...")

for item in sorted\_scores:

  movie\_title=df[df.Movie\_id==item[0]]['Title'].values[0]

  print(j+1,movie\_title)

  j=j+1

  if j>6:

    break;

**Output:**

