**MOVIE RECOMMENDATION SYSTEM**

***A Project Report***

*Submitted in fulfillment of the requirements for the course work of*

**DATA MINING TECHNOLOGIES**

**In**

**MINOR PROJECT**

**By**

**MONIKA . J(181FA04198)**

**R . PRATYUSHA(181FA4222)**

**SHIVANI . R (181FA04229)**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**VIGNAN’S Foundation for Science, Technology and Research**

**(Deemed to be University)**

**VADLAMUDI, GUNTUR - 522 213, INDIA**

**DECLARATION**

I hereby declare that work described in this minor project **MOVIE RECOMMENDATION SYSTEM** which is being submitted by me for the partial fulfillment of DATA MINING TECHNIQUES LAB minor project in Information Technology to Vignan’s Foundation For Science Technology and Research, deemed to be university Vadlamudi, Guntur District, Andhra Pradesh, and the result of investigations are carried out by me under the guidance of Mr.O.Gandhi

Place: Vadlamudi Signature of Students

Date:

**CERTIFICATE**

This is to certify that the minor project entitled **MOVIE RECOMMENDATION SYSTEM** being submitted by Monika.J (181FA04198), R.Pratyusha (181FA04222), Shivani.R (181FA04229), in partial fulfillment DATA MINING TECHNIQUES LAB minor project in INFORMATION TECHNOLOGY to Vignan’s Foundation for Science Technology and Research, Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by him under our guidance and supervision.

Internal Guide: Head of the department:

Mr.O.Gandhi Dr.D.Venkateswarulu

External Examiner

**ACKNOWLEDGEMENTS**

We express our sincere thanks to Mr.O.Gandhi, of Vignan’s Foundation for Science Technology and Research, deemed to be university Vadlamudi, Guntur District, Andhra Pradesh for their valuable guidance and suggestions. I sincerely thank to Dr.D.Venkateswarulu, Head of the department for his valuable encouragement. I thank the authorities of Vignan’s Foundation for Science Technology and Research, deemed to be university.

Monika . J

R .Pratyusha

Shivani. R

**TABLE OF CONTENTS**

**Contents Page. No**

**Declaration ii**

**Certificate iii**

**Acknowledgements iv**

**Table of Contents v**

**Abstract vi**

**1. Introduction vii**

**2. Objective viii**

**3. Problem Statement viii**

**4. Software Requirements ix**

**5. Hardware Requirements ix**

**6. Implementation -Algorithm x**

**7. Dataset Used xi-xii**

**8. Sample Code xiii-xv**

**9. Results xvi**

**10. Conclusion xvii**

**11. References xviii**

**ABSTRACT**

A movie recommendation system that has the ability to recommend movies to a new user as well as the others. It mines movie databases to collect all the important information, such as popularity and rating, required for recommendation. It generates movie swarms not only convenient for movie producer to plan a new movie but also useful for movie recommendation.

**1. INTRODUCTION**

A recommendation system has become an indispensable component in various e-commerce applications. Recommender systems collect information about the user’s preferences of different items (e.g., movies, shopping, tourism, TV, taxi) by two ways, either implicitly or explicitly. An implicit acquisition of user information typically involves observing the user’s behavior such as watched movies, purchased products, downloaded applications. On the other hand, a direct procurement of information typically involves collecting the user’s previous ratings or history. Collaborative filtering (CF) is the way of filtering or calculating items through the sentiments of other people.

**2. OBJECTIVE**

The movie recommendation system primary  objective is to suggest a recommender system through data clustering and computational intelligence. To assist users in classifying users with similar interests.

**3. PROBLEM STATEMENT**

The  movie recommendation system primary  objective is to suggest a recommender system through data clustering and computational intelligence. To assist users in classifying users with similar interests. Providing related content out of relevant and irrelevant collection of items to users of online service providers.

**4. Software Requirements**

∙ Windows

∙ Colaboratory

∙ Intel i3 processor

**5. Hardware Requirements**

Hard disk

∙ Processor: Minimum 1 GHz; Recommended 2GHz or more.

∙ Hard Drive: Minimum 32 GB; Recommended 64 GB or more.

∙ Memory (RAM): Minimum 1 GB; Recommended 4 GB or above.

**6. Implementation -Algorithm**

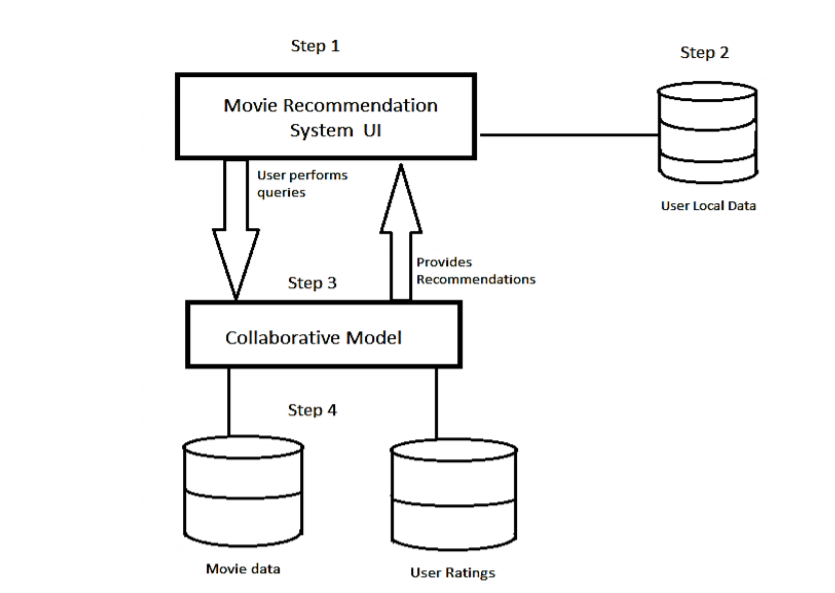
We will be using the KNN algorithm to compute similarity with cosine distance metric which is very fast. KNN algorithm is called K nearest neighbor classification algorithm. The core idea of the KNN algorithm is, if the majority of the k most similar neighbors of sample in the feature space belongs to a certain category, then the sample is considered to belong to this category.

STEPS:

Step 1:- First, a new user is provided with a screen that contains a search bar that allows him to search for a particular movie. If the user is an existing one, he/she will be provided a different screen.

Step 2:- In this step, the user’s local data, which is the movies he/she has previously watched and the ratings provided by him will be stored in a separate database.

Step 3:- In this step, all the information about movies such as genre, abstract, the title will be stored in a “Movie data” database and all the other users global ratings will be stored in a database called “User ratings”.

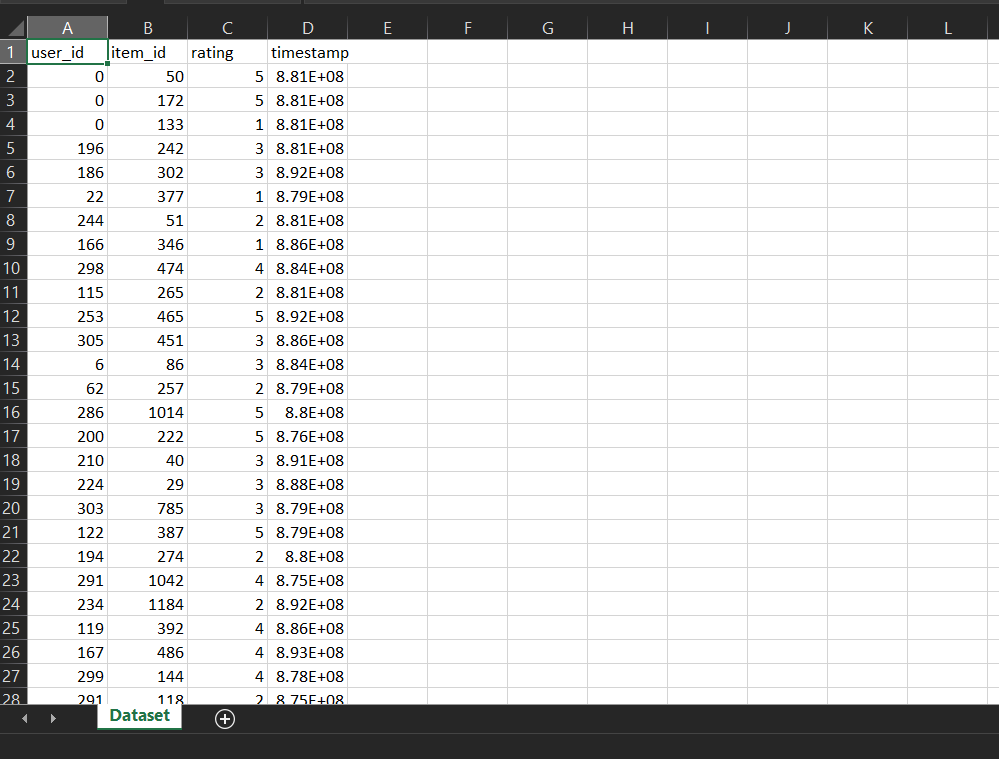
****

**7. Dataset Used**

For our own system, we’ll use the open-source MovieLens dataset from GroupLens. This dataset contains around 100K data points of various movies and users.

We will use three columns from the data:

* user­\_Id
* item\_Id
* rating
* “user\_id” is the id of each user which differs for each user.
* “item\_id” is the id of the movie which is also unique for each and every movie/item.
* “rating” is given to the movie by the user. The rating can only be given till a specified number.



**8. Sample Code**

import pandas as pd

import numpy as np

from scipy.sparse import csr\_matrix

from sklearn.neighbors import NearestNeighbors

import matplotlib.pyplot as plt

import seaborn as sns

movies = pd.read\_csv("movies.csv")

ratings = pd.read\_csv(".csv")

movies.head()

ratings.head()

final\_dataset = ratings.pivot(index='movieId',columns='userId',values='rating')

final\_dataset.head()

final\_dataset.fillna(0,inplace=True)

final\_dataset.head()

no\_user\_voted = ratings.groupby('movieId')['rating'].agg('count')

no\_movies\_voted = ratings.groupby('userId')['rating'].agg('count')

f,ax = plt.subplots(1,1,figsize=(16,4))

# ratings['rating'].plot(kind='hist')

plt.scatter(no\_user\_voted.index,no\_user\_voted,color='mediumseagreen')

plt.axhline(y=10,color='r')

plt.xlabel('MovieId')

plt.ylabel('No. of users voted')

plt.show()

final\_dataset = final\_dataset.loc[no\_user\_voted[no\_user\_voted > 10].index,:]

f,ax = plt.subplots(1,1,figsize=(16,4))

plt.scatter(no\_movies\_voted.index,no\_movies\_voted,color='mediumseagreen')

plt.axhline(y=50,color='r')

plt.xlabel('UserId')

plt.ylabel('No. of votes by user')

plt.show()

final\_dataset=final\_dataset.loc[:,no\_movies\_voted[no\_movies\_voted > 50].index]

final\_dataset

sample = np.array([[0,0,3,0,0],[4,0,0,0,2],[0,0,0,0,1]])

sparsity = 1.0 - ( np.count\_nonzero(sample) / float(sample.size) )

print(sparsity)

csr\_sample = csr\_matrix(sample)

print(csr\_sample)

csr\_data = csr\_matrix(final\_dataset.values)

final\_dataset.reset\_index(inplace=True)

knn = NearestNeighbors(metric='cosine', algorithm='brute', n\_neighbors=20, n\_jobs=-1)

knn.fit(csr\_data)

def get\_movie\_recommendation(movie\_name):

n\_movies\_to\_reccomend = 10

movie\_list = movies[movies['title'].str.contains(movie\_name)]

if len(movie\_list):

    movie\_idx= movie\_list.iloc[0]['movieId']

    movie\_idx = final\_dataset[final\_dataset['movieId'] == movie\_idx].index[0]

    distances , indices = knn.kneighbors(csr\_data[movie\_idx],n\_neighbors=n\_movies\_to\_reccomend+1)

    rec\_movie\_indices = sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),key=lambda x: x[1])[:0:-1]

    recommend\_frame = []

    for val in rec\_movie\_indices:

        movie\_idx = final\_dataset.iloc[val[0]]['movieId']

        idx = movies[movies['movieId'] == movie\_idx].index

        recommend\_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':val[1]})

    df = pd.DataFrame(recommend\_frame,index=range(1,n\_movies\_to\_reccomend+1))

    return df

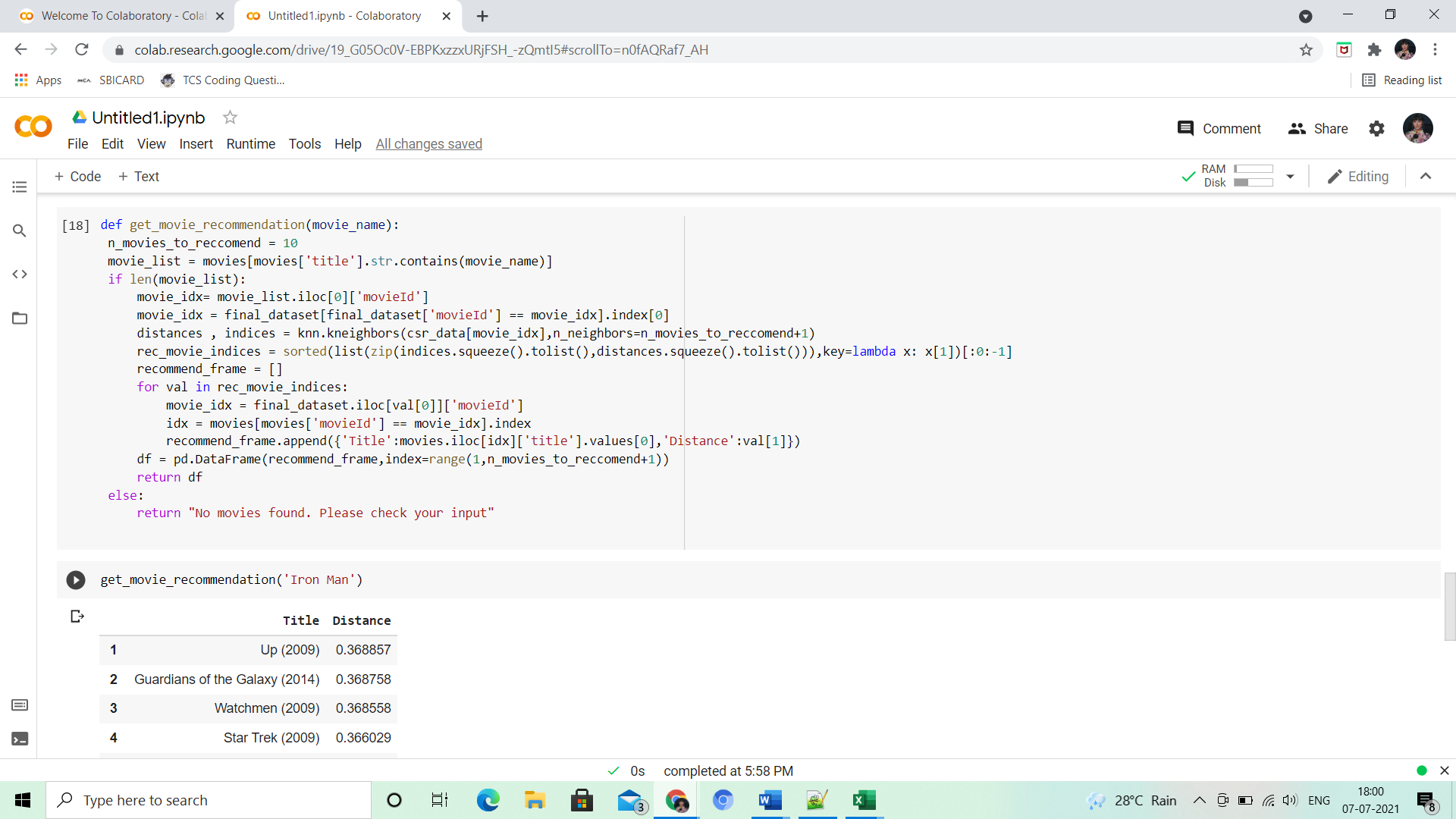
else:

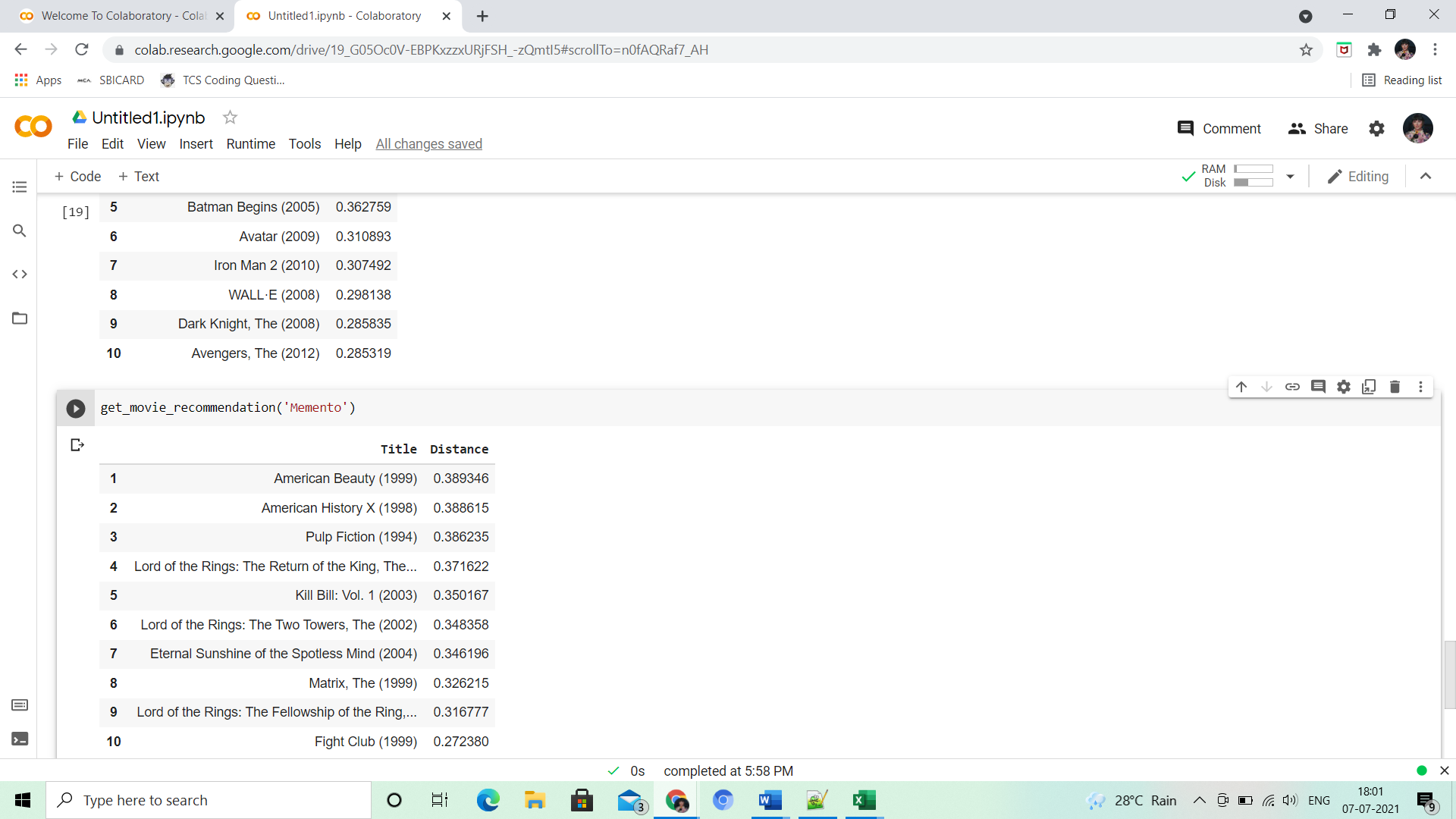
    return "No movies found. Please check your input"

get\_movie\_recommendation('Iron Man')

get\_movie\_recommendation('Memento')

**10. Results (Output & Screen Shots)**





**11. Conclusion**

This recommendation system recommends different movies to users. Since this system is based on a collaborative approach, it will give progressively explicit outcomes contrasted with different systems that are based on the content-based approach. Content-based recommendation systems are constrained to people, these systems don't prescribe things out of the box. These systems work on individual users’ ratings, hence limiting your choice to explore more. While our system which is based on a collaborative approach computes the connection between different clients and relying upon their ratings, prescribes movies to others who have similar tastes, subsequently allowing users to explore more. It is a web application that allows users to rate movies as well as recommends them appropriate movies based on other's ratings.

.

**12. References**

[**https://www.sciencedirect.com/science/article/pii/S1110866516300470**](https://www.sciencedirect.com/science/article/pii/S1110866516300470)

[**https://hcis-journal.springeropen.com/articles/10.1186/s13673-018-0161-6**](https://hcis-journal.springeropen.com/articles/10.1186/s13673-018-0161-6)