MOVIE RECOMMENDATION

**Import Libraries**

import numpy as np

import pandas as pd

import difflib

from sklearn.feature\_extraction.text import TfidfVectorizer

# TfidfVectorizer - This is used to convert text data into numerical values

from sklearn.metrics.pairwise import cosine\_similarity

import os

**Check the current working directory**

display (os.getcwd())

**Change the current working directory and read data**

os.chdir('C:\\Noble\\Training\\Acmegrade\\Data Science\\Projects\\PRJ Movie Recommendation\\')

movies\_data =pd.read\_csv('movies.csv')

movies\_data.head()

**Display the shape**

display (movies\_data.shape)

**Selecting the relevant features for recommendation**

selected\_features = ['genres','keywords','tagline','cast','director']

print(selected\_features)

**Display the info**

display (movies\_data.info())

**Check for Null Values**

display (movies\_data.isna().sum())

**Display the selected columns**

display (movies\_data[selected\_features].head())

**Check Null Values in selected columns**

display (movies\_data[selected\_features].isna().sum())

**Replacing the null values with null string**

for feature in selected\_features:

movies\_data[feature] = movies\_data[feature].fillna('')

display (movies\_data.head())

**Combining all the 5 selected features**

combined\_features = movies\_data['genres']+' '+movies\_data['keywords']+' '+movies\_data['tagline']+' '+movies\_data['cast']+' '+movies\_data['director']

display (combined\_features)

**Converting the text data to feature vectors**

This is to find cosine similarity

# Vector shape is (4803, 17318). This is based on the number of distinct words. All the words will be converted to their equivalent numbers.

vectorizer = TfidfVectorizer()

feature\_vectors = vectorizer.fit\_transform(combined\_features)

display (feature\_vectors.shape)

print (feature\_vectors)

**Getting the similarity scores using cosine similarity**

similarity = cosine\_similarity(feature\_vectors)

print (similarity )

**Print Shape Cosine Similarity**

display(similarity.shape)

**Enter the movie name to get Similarity**

movie\_name = input(' Enter your favourite movie name : ')

**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)

**Length of the List**

len(list\_of\_all\_titles)

**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)

**Display the close match**

close\_match = find\_close\_match[0]

print(close\_match)

**Finding the index of the movie with the title**

index\_of\_the\_movie = movies\_data[movies\_data.title == close\_match]['index'].values[0]

print(index\_of\_the\_movie)

**Get the similarity row for the selected index**

These are the similarity values for the movie entered by the user

similarity\_score = list(enumerate(similarity[index\_of\_the\_movie]))

print(similarity\_score)

**Length of Similarity Score**

len(similarity\_score)

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

Display the index and similarity rating as a tuple

This list is sorted based on the similarity score

sorted\_similar\_movies = sorted(similarity\_score, key = lambda x:x[1], reverse = True)

print(sorted\_similar\_movies)

**Print the name of similar movies based on the index – Top 30**

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

**Consolidated code**

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