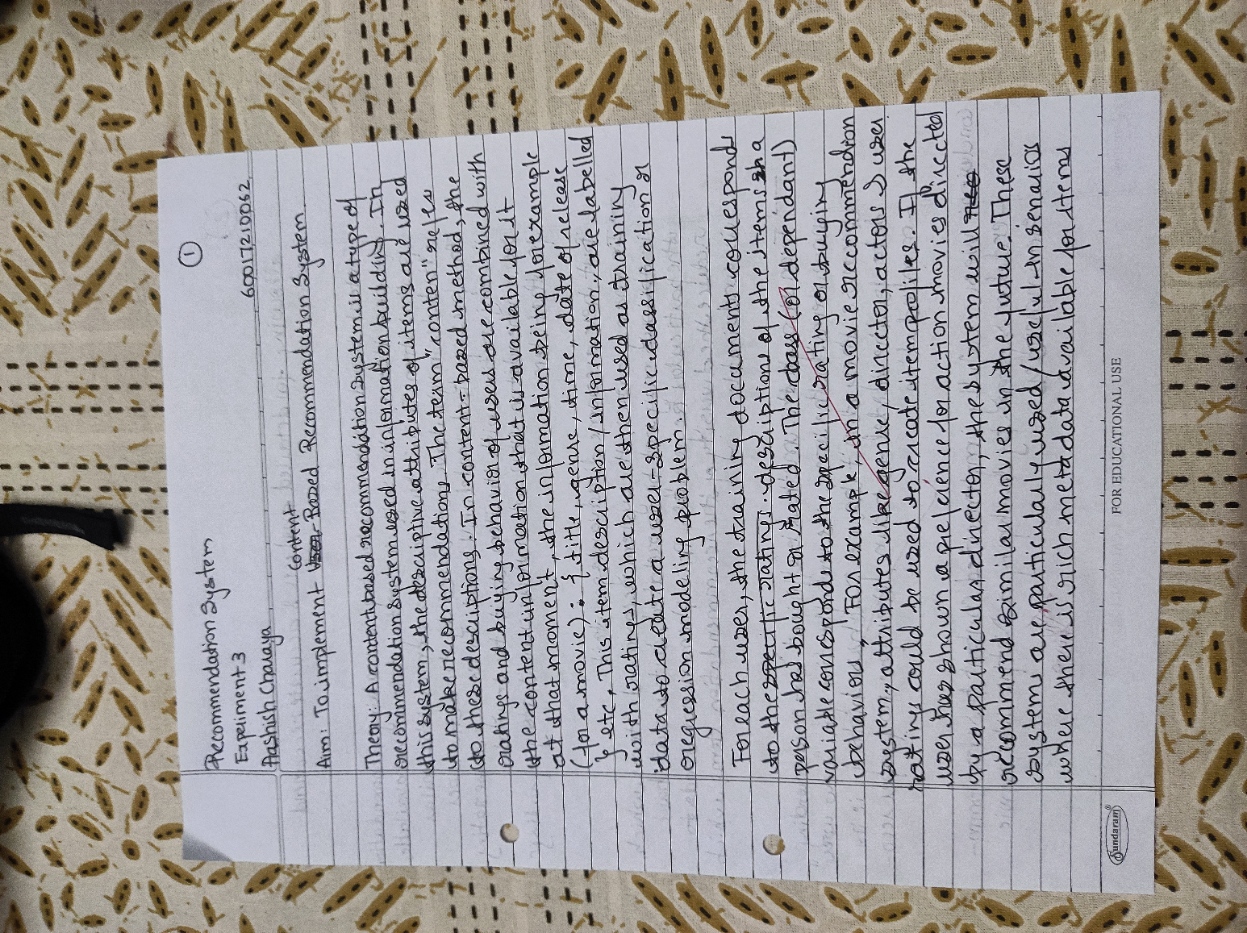
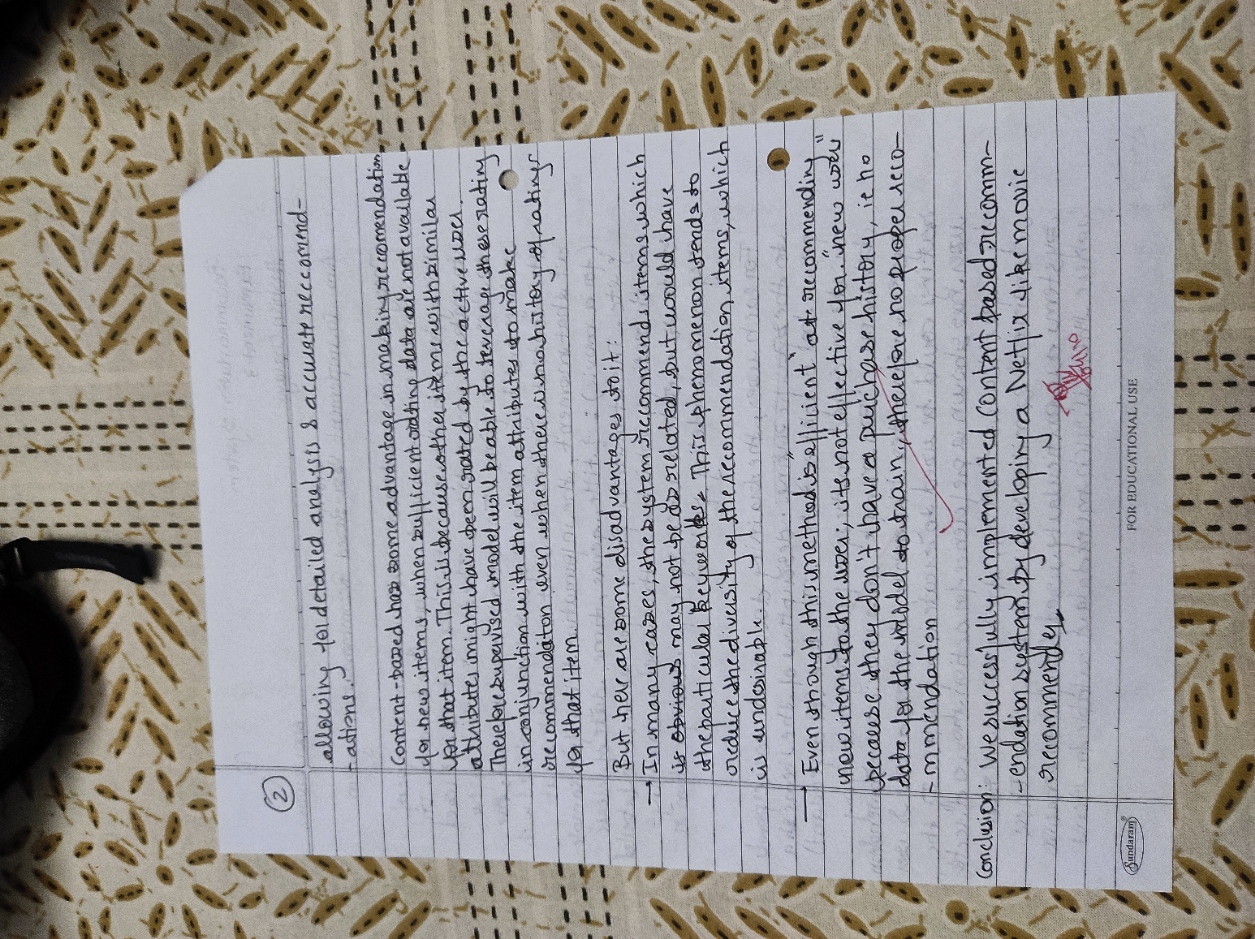
Recommendation System

Experiment 3

Aim: To Build a Content based Recommendation System.

Theory:





Code:

import pandas as pd  
import numpy as np

data = pd.read\_csv("netflix\_titles.csv")

data.shape

(8807, 12)

data.head()

show\_id type title director \  
0 s1 Movie Dick Johnson Is Dead Kirsten Johnson   
1 s2 TV Show Blood & Water NaN   
2 s3 TV Show Ganglands Julien Leclercq   
3 s4 TV Show Jailbirds New Orleans NaN   
4 s5 TV Show Kota Factory NaN   
  
 cast country \  
0 NaN United States   
1 Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban... South Africa   
2 Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi... NaN   
3 NaN NaN   
4 Mayur More, Jitendra Kumar, Ranjan Raj, Alam K... India   
  
 date\_added release\_year rating duration \  
0 September 25, 2021 2020 PG-13 90 min   
1 September 24, 2021 2021 TV-MA 2 Seasons   
2 September 24, 2021 2021 TV-MA 1 Season   
3 September 24, 2021 2021 TV-MA 1 Season   
4 September 24, 2021 2021 TV-MA 2 Seasons   
  
 listed\_in \  
0 Documentaries   
1 International TV Shows, TV Dramas, TV Mysteries   
2 Crime TV Shows, International TV Shows, TV Act...   
3 Docuseries, Reality TV   
4 International TV Shows, Romantic TV Shows, TV ...   
  
 description   
0 As her father nears the end of his life, filmm...   
1 After crossing paths at a party, a Cape Town t...   
2 To protect his family from a powerful drug lor...   
3 Feuds, flirtations and toilet talk go down amo...   
4 In a city of coaching centers known to train I...

data.isnull().sum()

show\_id 0  
type 0  
title 0  
director 2634  
cast 825  
country 831  
date\_added 10  
release\_year 0  
rating 4  
duration 3  
listed\_in 0  
description 0  
dtype: int64

import numpy as np  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.metrics.pairwise import cosine\_similarity

tfidf\_vectorizer = TfidfVectorizer(stop\_words='english')  
tfidf\_matrix = tfidf\_vectorizer.fit\_transform(data['description'])  
  
# Calculate cosine similarity between all pairs of movies  
cosine\_similarities = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

cosine\_similarities.shape

(8807, 8807)

title\_to\_find = 'Ganglands'  
movie\_indices = data.index[data['title'] == title\_to\_find].tolist()  
  
# Print the indices  
print(f"Indices of movies with title '{title\_to\_find}': {movie\_indices}")

Indices of movies with title 'Ganglands': [2]

def recommend\_movie(title\_to\_find,num\_rec):  
 movie\_indices = data.index[data['title'] == title\_to\_find].tolist()[0]  
 print(movie\_indices)  
 similarities = cosine\_similarities[movie\_indices]  
 sorted\_indices = np.argsort(similarities)[::-1]  
 top\_n\_movies\_indices = sorted\_indices[:num\_rec+1]  
 top\_n\_movie\_titles = data.loc[top\_n\_movies\_indices]['title'].tolist()  
 return top\_n\_movie\_titles[1:]

recommend\_movie('Thor: Ragnarok',10)

8580

['Pandigai',  
 'Lusers',  
 'The Outsider',  
 'Angel Beats!',  
 'Inhuman Kiss',  
 'Gour Hari Dastaan: The Freedom File',  
 'Octonauts & the Great Barrier Reef',  
 'IO',  
 'Santa Clarita Diet',  
 'Dukhtar']

data['description'] = data['description'].fillna('') + data['director'].fillna('') + data['listed\_in'].fillna('') + data['type'].fillna('')

import gensim  
from gensim.models import Word2Vec  
from nltk.tokenize import word\_tokenize  
import pandas as pd  
import nltk

tokenized\_descriptions = [word\_tokenize(desc.lower()) for desc in data['description']]  
  
# Train Word2Vec model  
embedding\_size = 100 # You can adjust this based on your needs  
model = Word2Vec(tokenized\_descriptions, vector\_size=embedding\_size, window=5, min\_count=1, sg=0)  
  
# You can save the model for later use  
model.save("movie\_descriptions\_word2vec.model")

embedding\_vector = model.wv['action']  
embedding\_vector

array([-0.5184265 , 1.4336095 , -1.1279651 , 0.88217217, -1.2734706 ,  
 -0.78887534, -0.6492172 , 1.9589189 , -0.22012177, -3.234885 ,  
 -0.6573134 , -0.31242514, -2.2324436 , 0.05188342, 1.0118003 ,  
 0.9074837 , -0.24146658, 0.51064444, -2.0512233 , -1.4412698 ,  
 0.47179124, 0.8397431 , 0.2969836 , 0.4541243 , 0.625643 ,  
 1.3979965 , -1.7820641 , -0.7373029 , -0.7427942 , 2.8741224 ,  
 0.2065601 , -1.0646011 , 2.603339 , -2.4967897 , 0.10420928,  
 2.4822884 , 0.62088567, 2.4308279 , 0.14125063, 1.7300371 ,  
 0.82063913, -3.754287 , -2.241115 , 1.4041905 , 1.1542271 ,  
 -2.9635465 , -0.24426544, 0.0709093 , 1.9430627 , 0.80214655,  
 1.1739455 , -2.1500995 , -0.11427487, -0.25127465, -2.6248567 ,  
 1.5922418 , 1.3441266 , 1.499648 , -0.6430934 , 1.3876617 ,  
 0.5905826 , 1.0687064 , 1.8153685 , -0.7929102 , -0.13924253,  
 1.6574264 , 0.9343099 , 1.2155977 , -0.4150191 , 1.1826063 ,  
 0.19636567, -1.0935277 , -0.04743399, 0.09356517, 1.6312791 ,  
 0.37776977, -1.9807703 , 0.94442785, 0.19939026, -0.6204835 ,  
 -0.02583213, -0.883639 , -0.8620559 , -1.1972033 , 0.06164009,  
 0.8097816 , 0.6966729 , 0.2618618 , 1.2695332 , 1.886175 ,  
 0.24356358, 0.31909695, 1.3420364 , -0.533573 , 0.6255839 ,  
 -0.74140763, 1.7684685 , -0.07185961, -0.20391285, -0.6468024 ],  
 dtype=float32)

def get\_description\_embedding(description):  
 words = word\_tokenize(description.lower())  
 embedding = [model.wv[word] for word in words if word in model.wv]  
 return sum(embedding) / len(embedding) if embedding else [0] \* embedding\_size

data['description\_embedding'] = [get\_description\_embedding(desc) for desc in data['description']]

embedding\_matrix = data['description\_embedding']

description\_embeddings = np.array(data['description\_embedding'].to\_list())  
  
# Calculate cosine similarity between movie descriptions  
cosine\_similarities = cosine\_similarity(description\_embeddings, description\_embeddings)  
  
def recommend\_movie(title\_to\_find, num\_rec):  
 movie\_indices = data.index[data['title'] == title\_to\_find].tolist()[0]  
 print(movie\_indices)  
 similarities = cosine\_similarities[movie\_indices]  
 sorted\_indices = np.argsort(similarities)[::-1]  
 top\_n\_movies\_indices = sorted\_indices[:num\_rec + 1]  
 top\_n\_movie\_titles = data.loc[top\_n\_movies\_indices]['title'].tolist()  
   
 # Get similarity scores for the recommended movies  
 similarity\_scores = [similarities[idx] for idx in top\_n\_movies\_indices]  
   
 # Create a list of tuples with movie titles and similarity scores  
 recommended\_movies\_with\_scores = list(zip(top\_n\_movie\_titles[1:], similarity\_scores[1:]))  
   
 return recommended\_movies\_with\_scores

recommended\_movies = recommend\_movie('Thor: Ragnarok',10)  
print("Recommended Movies with Similarity Scores:")  
for movie, score in recommended\_movies:  
 print(f"Movie: {movie}, Similarity Score: {score}")

8580  
Recommended Movies with Similarity Scores:  
Movie: The Matrix Reloaded, Similarity Score: 0.9980129599571228  
Movie: Money Talks, Similarity Score: 0.9978792071342468  
Movie: Spider-Man 3, Similarity Score: 0.9978455901145935  
Movie: The Lord of the Rings: The Two Towers, Similarity Score: 0.9977781772613525  
Movie: The Book of Eli, Similarity Score: 0.9975395798683167  
Movie: In the Shadow of the Moon, Similarity Score: 0.997494101524353  
Movie: Red Dawn, Similarity Score: 0.9973134398460388  
Movie: Black Panther, Similarity Score: 0.9971238374710083  
Movie: Seventh Son, Similarity Score: 0.9968461990356445  
Movie: Ant-Man and the Wasp, Similarity Score: 0.9968146681785583

data.shape

(8807, 13)

Conclusion: We successfully implemented Content Based Recommendation System by making a Movie recommender using a Neflix dataset

https://github.com/Aashish-Charaya/RS\_practicals/tree/main/exp3