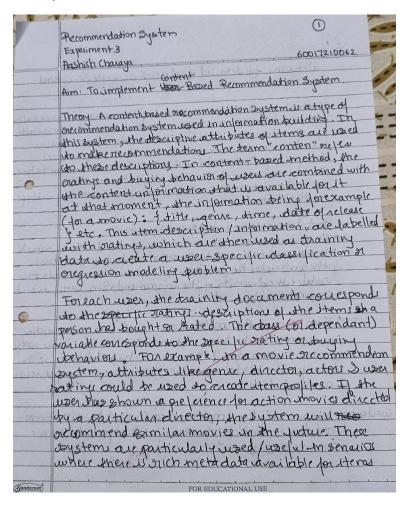
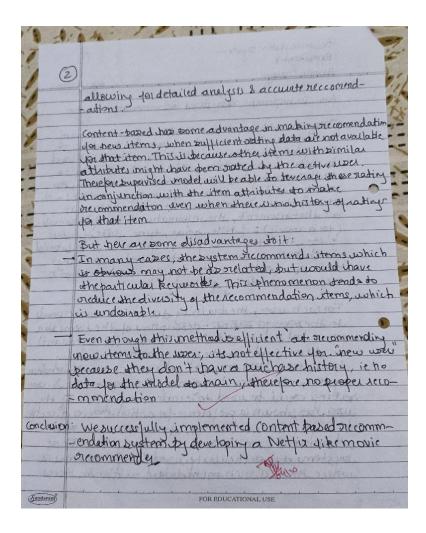
Recommendation System Experiment 3

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
                                    title
                                                   director
              type
0
       s1
             Movie
                     Dick Johnson Is Dead Kirsten Johnson
1
       s2 TV Show
                            Blood & Water
2
       s3
          TV Show
                                Ganglands
                                           Julien Leclercq
3
       s4
          TV Show Jailbirds New Orleans
                                                        NaN
4
           TV Show
                             Kota Factory
                                                        NaN
                                                 cast
                                                             country \
                                                       United States
0
                                                  NaN
  Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...
                                                        South Africa
   Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...
                                                                 NaN
```

```
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
     International TV Shows, TV Dramas, TV Mysteries
1
2 Crime TV Shows, International TV Shows, TV Act...
                             Docuseries, Reality TV
3
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
type
                  0
title
                  0
               2634
director
                825
cast
                831
country
date added
                 10
release year
                  0
                  4
rating
                  3
duration
listed in
                  0
description
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',
 '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