Fr. Conceicao Rodrigues College of Engineering

Father Agnel Ashram, Bandstand, Bandra –west, Mumbai-50

Department of Computer Engineering

SOCIAL MEDIA ANALYTICS LAB

Experiment No: 8

Aim: Implement content based and collaborative based filtering.

Objective: To design a recommendation system.

Lab outcomes:

At the end of this lab session, students will be able to…

1. Recommend an item to the user using the concept of content and collaborative based filtering

Theory

Content-based filtering is a recommendation technique that suggests items to users based on their preferences and past interactions with similar items. Unlike collaborative filtering, which relies on user-item interactions, content-based filtering analyzes the attributes of the items themselves. It works by creating profiles of users and items based on features such as keywords, genres, or metadata. Then, it recommends items to users that match their preferences or exhibit similar characteristics to items they have liked in the past. This approach is particularly useful when dealing with sparse or cold-start user-item interactions and can provide personalized recommendations even for new or niche items.

Collaborative filtering is a recommendation technique that leverages the behavior and preferences of a group of users to generate recommendations for individuals. It works by analyzing the interactions (such as ratings or purchases) between users and items in a dataset. The underlying assumption is that users who have interacted similarly with items in the past will have similar tastes and preferences in the future.

There are two main types of collaborative filtering: user-based and item-based. User-based collaborative filtering recommends items to a user based on the preferences of users with similar tastes. In contrast, item-based collaborative filtering recommends items that are similar to those that a user has liked in the past.

Fr. Conceicao Rodrigues College of Engineering

Father Agnel Ashram, Bandstand, Bandra –west, Mumbai-50

Department of Computer Engineering

SOCIAL MEDIA ANALYTICS LAB

One of the key advantages of collaborative filtering is its ability to provide personalized recommendations without requiring explicit knowledge about the items being recommended. However, it can suffer from the cold-start problem when there is insufficient user interaction data, and it may struggle to recommend items that are outside the preferences of the user community.

Student's Task:

1. Load the movie dataset of your choice.
2. Apply content-based recommendation system for the selected dataset.
3. Apply collaborative-based recommendation system for the selected dataset.
4. Evaluate the performance in both the cases.

Content based filtering

+ Code + Markdown

[45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations
import seaborn as sns

ratings = pd.read_csv('/kaggle/input/dataset-content-based-and-colaborating-filtering/ratings.csv', sep='\t', encoding='latin-1', usecols=['user_id', 'movie_id', 'rating'])
users = pd.read_csv('/kaggle/input/dataset-content-based-and-colaborating-filtering/users.csv', sep='\t', encoding='latin-1', usecols=['user_id', 'gender', 'zipcode', 'age_desc'
movies = pd.read_csv('/kaggle/input/dataset-content-based-and-colaborating-filtering/movies.csv', sep='\t', encoding='latin-1', usecols=['movie_id', 'title', 'genres'])
```

[46]:
```
ratings.shape
```

[46]:
```
(1000209, 3)
```

[47]:
```
ratings.sample(5)
```

[47]:

	user_id	movie_id	rating
783333	4679	2070	4
264513	1609	3082	3
703325	4217	3671	4
788236	4715	2321	3
853766	5122	2139	5

[48]:
```
movies.sample(5)
```

[48]:

	movie_id	title	genres
3751	3821	Nutty Professor II: The Klumps (2000)	Comedy
2441	2510	Just the Ticket (1999)	Comedy|Romance
2503	2572	10 Things I Hate About You (1999)	Comedy|Romance
921	933	To Catch a Thief (1955)	Comedy|Romance|Thriller
2744	2813	Source, The (1999)	Documentary

[49]:
```
users.head()
```

[49]:

	user_id	gender	zipcode	age_desc	occ_desc
0	1	F	48067	Under 18	K-12 student
1	2	M	70072	56+	self-employed
2	3	M	55117	25-34	scientist
3	4	M	02460	45-49	executive/managerial
4	5	M	55455	25-34	writer

[50]:
```
def weighted_average_score(df, k=0.8):
    n_views = df.groupby('movie_id', sort=False).movie_id.count()
    ratings = df.groupby('movie_id', sort=False).rating.mean()
    scores = ((1-k)*(n_views/n_views.max()) +
              k*(ratings/ratings.max())).to_numpy().argsort()[::-1]
    df_deduped = df.groupby('movie_id', sort=False).agg({'title':'first',
                                                         'genres':'first',
                                                         'rating':'mean'})
    return df_deduped.assign(views=n_views).iloc[scores]
```

[51]:
```
df = movies.merge(ratings).merge(users)
weighted_average_score(df).head(10)
```

[51]:

movie_id	title	genres	rating	views
2858	American Beauty (1999)	Comedy|Drama	4.317386	3428
260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	4.453694	2991
1198	Raiders of the Lost Ark (1981)	Action|Adventure	4.477725	2514
1196	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Drama|Sci-Fi|War	4.292977	2990
318	Shawshank Redemption, The (1994)	Drama	4.554558	2227
527	Schindler's List (1993)	Drama|War	4.510417	2304
858	Godfather, The (1972)	Action|Crime|Drama	4.524966	2223
2028	Saving Private Ryan (1998)	Action|Drama|War	4.337354	2653
2762	Sixth Sense, The (1999)	Thriller	4.406263	2459
593	Silence of the Lambs, The (1991)	Drama|Thriller	4.351823	2578

[52]:
```
genre_popularity = (movies.genres.str.split('|')
                          .explode()
                          .value_counts()
                          .sort_values(ascending=False))
genre_popularity.head(10)
```

[52]:
```
genres
Drama        1603
Comedy       1200
Action        503
Thriller      492
Romance       471
Horror        343
Adventure     283
Sci-Fi        276
Children's    251
Crime         211
Name: count, dtype: int64
```

[53]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[54]:
```
movies.head(3)
```

[54]:

	movie_id	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance

[55]:
```
s = "Animation Children's Comedy"
tf_wrong = TfidfVectorizer(analyzer='word', ngram_range=(1,2))
tf_wrong.fit([s])
tf_wrong.get_feature_names_out()
# ['animation', 'animation children', 'children', 'children comedy', 'comedy']
```

[55]:
```
array(['animation', 'animation children', 'children', 'children comedy',
       'comedy'], dtype=object)
```

[56]:
```
[c for i in range(1,2) for c in combinations(s.split(), r=i)]
```

[56]:
```
[('Animation',), ("Children's",), ('Comedy',)]
```

[57]:
```
tf = TfidfVectorizer(analyzer=lambda s: (c for i in range(1,4)
                                         for c in combinations(s.split('|'), r=i)))
tfidf_matrix = tf.fit_transform(movies['genres'])
tfidf_matrix.shape
# (3883, 353)
```

[57]:
```
(3883, 353)
```

[58]:
```
pd.DataFrame(tfidf_matrix.todense(), columns=tf.get_feature_names_out(), index=movies.title).sample(5, axis=1).sample(10, axis=0)
```

[58]:

title	(Drama, Fantasy, Romance)	(Action, Animation, Horror)	(Drama, Film-Noir, Thriller)	(Children's, Comedy, Sci-Fi)	(Children's, Drama, Musical)
Gone Fishin' (1997)	0.0	0.0	0.0	0.0	0.0
Lolita (1997)	0.0	0.0	0.0	0.0	0.0
Wild Man Blues (1998)	0.0	0.0	0.0	0.0	0.0
Dead Zone, The (1983)	0.0	0.0	0.0	0.0	0.0
Make Mine Music (1946)	0.0	0.0	0.0	0.0	0.0
Psycho (1960)	0.0	0.0	0.0	0.0	0.0
Reservoir Dogs (1992)	0.0	0.0	0.0	0.0	0.0
Miracle on 34th Street (1947)	0.0	0.0	0.0	0.0	0.0
Last Temptation of Christ, The (1988)	0.0	0.0	0.0	0.0	0.0
Prince Valiant (1997)	0.0	0.0	0.0	0.0	0.0

[59]:
```
from sklearn.metrics.pairwise import cosine_similarity
cosine_sim = cosine_similarity(tfidf_matrix)
```

[60]:
```
cosine_sim_df = pd.DataFrame(cosine_sim, index=movies['title'], columns=movies['title'])
print('Shape:', cosine_sim_df.shape)
cosine_sim_df.sample(5, axis=1).round(2)
```

```
Shape: (3883, 3883)
```

[60]:

title	With Friends Like These... (1998)	True Crime (1999)	Rumble in the Bronx (1995)	Dark City (1998)	Touch of Evil (1958)
Toy Story (1995)	0.17	0.00	0.00	0.0	0.00
Jumanji (1995)	0.00	0.00	0.07	0.0	0.00
Grumpier Old Men (1995)	0.40	0.00	0.00	0.0	0.00
Waiting to Exhale (1995)	0.45	0.00	0.00	0.0	0.00
Father of the Bride Part II (1995)	1.00	0.00	0.00	0.0	0.00
...	...	...	...	...	...
Meet the Parents (2000)	1.00	0.00	0.00	0.0	0.00
Requiem for a Dream (2000)	0.00	0.00	0.00	0.0	0.00

Tigerland (2000)	0.00	0.00	0.00	0.0	0.00
Two Family House (2000)	0.00	0.00	0.00	0.0	0.00
Contender, The (2000)	0.00	0.23	0.00	0.1	0.11

3883 rows × 5 columns

[61]:
```
def genre_recommendations(i, M, items, k=10):
    """
    Recommends movies based on a similarity dataframe

    Parameters
    ----------
    i : str
        Movie (index of the similarity dataframe)
    M : pd.DataFrame
        Similarity dataframe, symmetric, with movies as indices and columns
    items : pd.DataFrame
        Contains both the title and some other features used to define similarity
    k : int
        Amount of recommendations to return

    """
    ix = M.loc[:,i].to_numpy().argpartition(range(-1,-k,-1))
    closest = M.columns[ix[-1:-(k+2):-1]]
    closest = closest.drop(i, errors='ignore')
    return pd.DataFrame(closest).merge(items).head(k)
```

[62]:
```
movies[movies.title.eq('2001: A Space Odyssey (1968)')]
```

[62]:

	movie_id	title	genres
912	924	2001: A Space Odyssey (1968)	Drama|Mystery|Sci-Fi|Thriller

[63]:
```
genre_recommendations('2001: A Space Odyssey (1968)', cosine_sim_df, movies[['title', 'genres']])
```

[63]:

	title	genres
0	X-Files: Fight the Future, The (1998)	Mystery|Sci-Fi|Thriller
1	Client, The (1994)	Drama|Mystery|Thriller
2	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller
3	Communion (1989)	Drama|Sci-Fi|Thriller
4	Gattaca (1997)	Drama|Sci-Fi|Thriller
5	Thirteenth Floor, The (1999)	Drama|Sci-Fi|Thriller
6	Event Horizon (1997)	Action|Mystery|Sci-Fi|Thriller
7	2010 (1984)	Mystery|Sci-Fi
8	Stalker (1979)	Mystery|Sci-Fi
9	Deep Impact (1998)	Action|Drama|Sci-Fi|Thriller

[64]:
```
print(movies[movies.title.eq('Contact (1997)')])
```

```
      movie_id          title        genres
1543      1584  Contact (1997)  Drama|Sci-Fi
```

[65]:
```
genre_recommendations('Contact (1997)', cosine_sim_df, movies[['title', 'genres']])
```

[65]:

	title	genres
0	Nineteen Eighty-Four (1984)	Drama|Sci-Fi
1	Twelve Monkeys (1995)	Drama|Sci-Fi
2	Day the Earth Stood Still, The (1951)	Drama|Sci-Fi
3	Solaris (Solyaris) (1972)	Drama|Sci-Fi
4	Powder (1995)	Drama|Sci-Fi
5	Goodbye, 20th Century (Zbogum na dvadesetiot v...	Drama|Sci-Fi
6	Until the End of the World (Bis ans Ende der W...	Drama|Sci-Fi
7	Conceiving Ada (1997)	Drama|Sci-Fi
8	Brother from Another Planet, The (1984)	Drama|Sci-Fi
9	Close Encounters of the Third Kind (1977)	Drama|Sci-Fi

[66]:
```
movies[movies.title.eq('Jungle Book, The (1967)')]
```

[66]:

	movie_id	title	genres
2009	2078	Jungle Book, The (1967)	Animation|Children's|Comedy|Musical

[67]:
```
genre_recommendations('Jungle Book, The (1967)', cosine_sim_df, movies[['title', 'genres']])
```

[67]:

	title	genres
0	Steamboat Willie (1940)	Animation|Children's|Comedy|Musical
1	Aladdin (1992)	Animation|Children's|Comedy|Musical
2	Hercules (1997)	Adventure|Animation|Children's|Comedy|Musical
3	Little Mermaid, The (1989)	Animation|Children's|Comedy|Musical|Romance
4	Lady and the Tramp (1955)	Animation|Children's|Comedy|Musical|Romance

5	Alice in Wonderland (1951)	Animation|Children's|Musical
6	Cinderella (1950)	Animation|Children's|Musical
7	Beauty and the Beast (1991)	Animation|Children's|Musical
8	Lion King, The (1994)	Animation|Children's|Musical
9	Cats Don't Dance (1997)	Animation|Children's|Musical

[ ]:

Collaborative recommendation System

[68]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from itertools import combinations
import seaborn as sns
```

[69]:
```
ratings = pd.read_csv('/kaggle/input/dataset-content-based-and-colaborating-filtering/ratings.csv', sep='\t', encoding='latin-1', usecols=['user_id', 'movie_id', 'rating'])
users = pd.read_csv('/kaggle/input/dataset-content-based-and-colaborating-filtering/users.csv', sep='\t', encoding='latin-1', usecols=['user_id', 'gender', 'zipcode', 'age_desc'
movies = pd.read_csv('/kaggle/input/dataset-content-based-and-colaborating-filtering/movies.csv', sep='\t', encoding='latin-1', usecols=['movie_id', 'title', 'genres'])
```

[70]:
```
# User-item matrix
user_item_m = ratings.pivot(values='rating', index='user_id',columns='movie_id').fillna(0)
print(f'Shape: {user_item_m.shape}')
```

```
Shape: (6040, 3706)
```

[71]:
```
user_item_m.iloc[:10,:15].astype('i1').T.join(movies.set_index('movie_id').title).set_index('title').T.rename_axis('user_id')
```

[71]:

user_id	Toy Story (1995)	Jumanji (1995)	Grumpier Old Men (1995)	Waiting to Exhale (1995)	Father of the Bride Part II (1995)	Heat (1995)	Sabrina (1995)	Tom and Huck (1995)	Sudden Death (1995)	GoldenEye (1995)	American President, The (1995)	Dracula: Dead and Loving It (1995)	Balto (1995)	Nixon (1995)	Cutthroat Island (1995)
1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
6	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
8	4	0	0	3	0	0	0	0	0	0	0	0	0	4	0
9	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	5	5	0	0	0	0	4	0	0	0	0	0	0	0	0

[72]:
```
# Similarity matrix
from sklearn.metrics.pairwise import cosine_similarity
X_user = cosine_similarity(user_item_m)
X_user.shape
```

[72]:
```
(6040, 6040)
```

[73]:
```
X_user[:10,:8].round(3)
```

[73]:
```
array([[1.   , 0.096, 0.121, 0.132, 0.09 , 0.179, 0.06 , 0.138],
       [0.096, 1.   , 0.151, 0.171, 0.114, 0.101, 0.306, 0.203],
       [0.121, 0.151, 1.   , 0.151, 0.063, 0.075, 0.138, 0.078],
       [0.132, 0.171, 0.151, 1.   , 0.045, 0.014, 0.13 , 0.101],
       [0.09 , 0.114, 0.063, 0.045, 1.   , 0.047, 0.126, 0.221],
       [0.179, 0.101, 0.075, 0.014, 0.047, 1.   , 0.05 , 0.075],
       [0.06 , 0.306, 0.138, 0.13 , 0.126, 0.05 , 1.   , 0.238],
       [0.138, 0.203, 0.078, 0.101, 0.221, 0.075, 0.238, 1.   ],
       [0.226, 0.19 , 0.126, 0.094, 0.261, 0.111, 0.162, 0.291],
       [0.255, 0.227, 0.214, 0.121, 0.117, 0.205, 0.093, 0.154]])
```

[74]:
```
X_item = cosine_similarity(user_item_m.T)
```

[75]:
```
X_item.shape
```

[75]:
```
(3706, 3706)
```

[76]:
```
X_item[:10,:11].round(2)
```

[76]:
```
array([[1.  , 0.39, 0.27, 0.18, 0.26, 0.35, 0.3 , 0.13, 0.11, 0.38, 0.42],
       [0.39, 1.  , 0.24, 0.16, 0.25, 0.24, 0.26, 0.2 , 0.16, 0.39, 0.28],
       [0.27, 0.24, 1.  , 0.19, 0.31, 0.19, 0.29, 0.09, 0.13, 0.25, 0.29],
       [0.18, 0.16, 0.19, 1.  , 0.27, 0.13, 0.22, 0.05, 0.06, 0.13, 0.24],
       [0.26, 0.25, 0.31, 0.27, 1.  , 0.15, 0.31, 0.1 , 0.14, 0.24, 0.31],
       [0.35, 0.24, 0.19, 0.13, 0.15, 1.  , 0.18, 0.06, 0.17, 0.42, 0.28],
       [0.3 , 0.26, 0.29, 0.22, 0.31, 0.18, 1.  , 0.05, 0.08, 0.25, 0.44]
```

```
[0.13, 0.2 , 0.09, 0.05, 0.1 , 0.06, 0.05, 1.  , 0.05, 0.11, 0.07],
[0.11, 0.16, 0.13, 0.06, 0.14, 0.17, 0.08, 0.05, 1.  , 0.22, 0.13],
[0.38, 0.39, 0.25, 0.13, 0.24, 0.42, 0.25, 0.11, 0.22, 1.  , 0.31]])
```

[77]:
```python
def movie_recommender(user_item_m, X_user, user, k=20, top_n=10):
    # Get location of the actual movie in the User-Items matrix
    user_ix = user_item_m.index.get_loc(user)
    # Use it to index the User similarity matrix
    user_similarities = X_user[user_ix]
    # obtain the indices of the top k most similar users
    most_similar_users = user_item_m.index[user_similarities.argpartition(-k)[-k:]]
    # Obtain the mean ratings of those users for all movies
    rec_movies = user_item_m.loc[most_similar_users].mean(0).sort_values(ascending=False)
    # Discard already seen movies
    m_seen_movies = user_item_m.loc[user].gt(0)
    seen_movies = m_seen_movies.index[m_seen_movies].tolist()
    rec_movies = rec_movies.drop(seen_movies).head(top_n)
    # return recommendations - top similar users rated movies
    return rec_movies.index.to_frame().reset_index(drop=True).merge(movies)
```

[78]:
```python
class CfRec():
    def __init__(self, M, X, items, k=20, top_n=10):
        self.X = X
        self.M = M
        self.k = k
        self.top_n = top_n
        self.items = items

    def recommend_user_based(self, user):
        ix = self.M.index.get_loc(user)
        # Use it to index the User similarity matrix
        u_sim = self.X[ix]
        # obtain the indices of the top k most similar users
        most_similar = self.M.index[u_sim.argpartition(-(self.k+1))[-(self.k+1):]]
        # Obtain the mean ratings of those users for all movies
        rec_items = self.M.loc[most_similar].mean(0).sort_values(ascending=False)
        # Discard already seen movies
        # already seen movies
        seen_mask = self.M.loc[user].gt(0)
        seen = seen_mask.index[seen_mask].tolist()
        rec_items = rec_items.drop(seen).head(self.top_n)
        # return recommendations - top similar users rated movies
        return (rec_items.index.to_frame()
                               .reset_index(drop=True)
                               .merge(self.items))

    def recommend_item_based(self, item):
        liked = self.items.loc[self.items.movie_id.eq(item), 'title'].item()
        print(f"Because you liked {liked}, we'd recommend you to watch:")
        # get index of movie
        ix = self.M.columns.get_loc(item)
        # Use it to index the User similarity matrix
        i_sim = self.X[ix]
        # obtain the indices of the top k most similar users
        most_similar = self.M.columns[i_sim.argpartition(-(self.k+1))[-(self.k+1):]]
        return (most_similar.difference([item])
                                .to_frame()
                                .reset_index(drop=True)
                                .merge(self.items)
                                .head(self.top_n))
```

[79]:
```python
def because_user_liked(user_item_m, movies, ratings, user):
    ix_user_seen = user_item_m.loc[user]>0.
    seen_by_user = user_item_m.columns[ix_user_seen]
    return (seen_by_user.to_frame()
                .reset_index(drop=True)
                .merge(movies)
                .assign(user_id=user)
                .merge(ratings[ratings.user_id.eq(user)])
                .sort_values('rating', ascending=False).head(10))
```

[80]:
```python
# testing
# user based recommendations
# Drama movies
rec = CfRec(user_item_m, X_user, movies)
because_user_liked(user_item_m, movies, ratings, 69)
```

[80]:

	movie_id	title	genres	user_id	rating
0	14	Nixon (1995)	Drama	69	5
27	1041	Secrets & Lies (1996)	Drama	69	5
20	515	Remains of the Day, The (1993)	Drama	69	5
48	1810	Primary Colors (1998)	Drama	69	5
45	1747	Wag the Dog (1997)	Comedy|Drama	69	5
24	593	Silence of the Lambs, The (1991)	Drama|Thriller	69	5
25	617	Flower of My Secret, The (La Flor de Mi Secret...	Drama	69	5
26	866	Bound (1996)	Crime|Drama|Romance|Thriller	69	5
28	1095	Glengarry Glen Ross (1992)	Drama	69	5
53	2336	Elizabeth (1998)	Drama	69	5

[81]:
```python
rec.recommend_user_based(69)
```

[81]:

	movie_id	title	genres
0	527	Schindler's List (1993)	DramalWar

1	318	Shawshank Redemption, The (1994)	Drama
2	608	Fargo (1996)	Crime|Drama|Thriller
3	1213	GoodFellas (1990)	Crime|Drama
4	150	Apollo 13 (1995)	Drama
5	1094	Crying Game, The (1992)	Drama|Romance|War
6	1179	Grifters, The (1990)	Crime|Drama|Film-Noir
7	36	Dead Man Walking (1995)	Drama
8	2333	Gods and Monsters (1998)	Drama
9	3100	River Runs Through It, A (1992)	Drama

[82]:
```
# fan of Horror movies
because_user_liked(user_item_m, movies, ratings, 2155)
```

[82]:

	movie_id	title	genres	user_id	rating
0	12	Dracula: Dead and Loving It (1995)	Comedy|Horror	2155	5
51	2548	Rage: Carrie 2, The (1999)	Horror	2155	5
33	1999	Exorcist III, The (1990)	Horror	2155	5
36	2026	Disturbing Behavior (1998)	Horror|Thriller	2155	5
37	2120	Needful Things (1993)	Drama|Horror	2155	5
38	2148	House (1986)	Comedy|Horror	2155	5
39	2279	Urban Legend (1998)	Horror|Thriller	2155	5
41	2328	Vampires (1998)	Horror	2155	5
42	2389	Psycho (1998)	Crime|Horror|Thriller	2155	5
45	2459	Texas Chainsaw Massacre, The (1974)	Horror	2155	5

[83]:
```
rec.recommend_user_based(2155)
```

[83]:

	movie_id	title	genres
0	1339	Bram Stoker's Dracula (1992)	Horror|Romance
1	1214	Alien (1979)	Action|Horror|Sci-Fi|Thriller
2	3499	Misery (1990)	Horror
3	1258	Shining, The (1980)	Horror
4	3081	Sleepy Hollow (1999)	Horror|Romance
5	1982	Halloween (1978)	Horror
6	1350	Omen, The (1976)	Horror
7	2160	Rosemary's Baby (1968)	Horror|Thriller
8	1387	Jaws (1975)	Action|Horror
9	1215	Army of Darkness (1993)	Action|Adventure|Comedy|Horror|Sci-Fi

[84]:
```
#  item based recommendations
rec = CfRec(user_item_m, X_item, movies)
rec.recommend_item_based(2021)
```

```
Because you liked Dune (1984), we'd recommend you to watch:
```

[84]:

	movie_id	title	genres
0	541	Blade Runner (1982)	Film-Noir|Sci-Fi
1	1200	Aliens (1986)	Action|Sci-Fi|Thriller|War
2	1240	Terminator, The (1984)	Action|Sci-Fi|Thriller
3	1371	Star Trek: The Motion Picture (1979)	Action|Adventure|Sci-Fi
4	1374	Star Trek: The Wrath of Khan (1982)	Action|Adventure|Sci-Fi
5	1375	Star Trek III: The Search for Spock (1984)	Action|Adventure|Sci-Fi
6	1376	Star Trek IV: The Voyage Home (1986)	Action|Adventure|Sci-Fi
7	1527	Fifth Element, The (1997)	Action|Sci-Fi
8	2011	Back to the Future Part II (1989)	Comedy|Sci-Fi
9	2105	Tron (1982)	Action|Adventure|Fantasy|Sci-Fi

[85]:
```
rec.recommend_item_based(47)
```

```
Because you liked Seven (Se7en) (1995), we'd recommend you to watch:
```

[85]:

	movie_id	title	genres
0	6	Heat (1995)	Action|Crime|Thriller
1	16	Casino (1995)	Drama|Thriller
2	32	Twelve Monkeys (1995)	Drama|Sci-Fi
3	50	Usual Suspects, The (1995)	Crime|Thriller
4	293	Professional, The (a.k.a. Leon: The Profession...	Crime|Drama|Romance|Thriller
5	296	Pulp Fiction (1994)	Crime|Drama
6	318	Shawshank Redemption, The (1994)	Drama
7	457	Fugitive, The (1993)	Action|Thriller
8	593	Silence of the Lambs, The (1991)	Drama|Thriller
9	608	Fargo (1996)	Crime|Drama|Thriller

[86]:
```
movies[movies.title.str.contains('Se7en')]
```

[86]:

	movie_id	title	genres
46	47	Seven (Se7en) (1995)	Crime|Thriller

[87]:
```
# animation movie
rec.recommend_item_based(2018)
```

```
Because you liked Bambi (1942), we'd recommend you to watch:
```

[87]:

	movie_id	title	genres
0	364	Lion King, The (1994)	Animation|Children's|Musical
1	588	Aladdin (1992)	Animation|Children's|Comedy|Musical
2	594	Snow White and the Seven Dwarfs (1937)	Animation|Children's|Musical
3	595	Beauty and the Beast (1991)	Animation|Children's|Musical
4	596	Pinocchio (1940)	Animation|Children's
5	1022	Cinderella (1950)	Animation|Children's|Musical
6	1025	Sword in the Stone, The (1963)	Animation|Children's
7	1029	Dumbo (1941)	Animation|Children's|Musical
8	1032	Alice in Wonderland (1951)	Animation|Children's|Musical
9	1033	Fox and the Hound, The (1981)	Animation|Children's

[ ]: