Movie Recommender on the basis of Release Date of Movie(s)

In [1]:
```
import numpy as np
```

1. create 2 dataframes netflixMovie_df and imdbMovie_df

preprocessing netflix dataframe

In [2]:
```
netflixMovie_df = pd.read_csv('Netflix_Dataset_Movie.csv')
```

In [3]:
```
# we don't need rating dataframe as there is no year of launch of movies
```

Out[3]:

	User_ID	Rating	Movie_ID
0	712664	5	3

In [4]:

Out[4]:

	Movie_ID	Year	Name
0	1	2003	Dinosaur Planet
1	2	2004	Isle of Man TT 2004 Review
2	3	1997	Character
3	4	1994	Paula Abdul's Get Up & Dance
4	5	2004	The Rise and Fall of ECW
...	...	...	...
17765	17766	2002	Where the Wild Things Are and Other Maurice Se...
17766	17767	2004	Fidel Castro: American Experience
17767	17768	2000	Epoch

	Movie_ID	Year	Name
17768	17769	2003	The Company
17769	17770	2003	Alien Hunter

In [5]:
```
# we need movie dataFrame of only netflixMovie_df = ['movie_names', 'release date']

netflixMovie_df = netflixMovie_df[['Name', 'Year']]
```

Out[5]:

	Name	Year
0	Dinosaur Planet	2003
1	Isle of Man TT 2004 Review	2004
2	Character	1997
3	Paula Abdul's Get Up & Dance	1994
4	The Rise and Fall of ECW	2004
...	...	...
17765	Where the Wild Things Are and Other Maurice Se...	2002
17766	Fidel Castro: American Experience	2004
17767	Epoch	2000
17768	The Company	2003
17769	Alien Hunter	2003

17770 rows × 2 columns

In [6]:
```
# convert 'Name' -> 'Movie_Title'
# convert 'Year' -> 'Released_Year'

netflixMovie_df = netflixMovie_df.rename(columns={'Name': 'Movie_Title'})
```

In [7]:

Out[7]:

	Movie_Title	Released_Year

preprocessing imdb dataframe

In [8]:

In [9]:

Out[9]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins

In [10]:

```
# we need movie dataFrame of only imdbMovie_df = ['movie_names', 'release date']
```

In [11]:

Out[11]:

	Series_Title	Released_Year
0	The Shawshank Redemption	1994

In [12]:

```
# convert 'Series_Title' -> 'Movie_Title'
```

In [13]:

Out[13]:

	Movie_Title	Released_Year
0	The Shawshank Redemption	1994

In [14]:

```
# check if there is some incorrect string value in imdbMovie_df & store it in movie_idx

def checkIncorrectValues():
    movie_idx = -1
    for i in range(len(imdbMovie_df['Released_Year'])):
        if(imdbMovie_df['Released_Year'].iloc[i] != 'PG'):
            imdbMovie_df['Released_Year'].iloc[i] = int(imdbMovie_df['Released_Year'].iloc[i])
        else:
            movie_idx = i
```

In [15]:

In [16]:
```
# check the incorrect 'Movie_Title' value in imdbMovie_df
```

Out[16]:
```
'Apollo 13'
```

In [17]:
```
# fill the movie's 'Released_Year' w/ movie's release date
```

2. Review of netflixMovie_df and imdbMovie_df

In [18]:

Out[18]:

	Movie_Title	Released_Year
0	Dinosaur Planet	2003
1	Isle of Man TT 2004 Review	2004
2	Character	1997
3	Paula Abdul's Get Up & Dance	1994
4	The Rise and Fall of ECW	2004
...	...	...
17765	Where the Wild Things Are and Other Maurice Se...	2002
17766	Fidel Castro: American Experience	2004
17767	Epoch	2000
17768	The Company	2003
17769	Alien Hunter	2003

17770 rows × 2 columns

In [19]:
```
[illegible]
```

Out[19]:

	Movie_Title	Released_Year
0	The Shawshank Redemption	1994
1	The Godfather	1972
2	The Dark Knight	2008
3	The Godfather: Part II	1974
4	12 Angry Men	1957
...	...	...
995	Breakfast at Tiffany's	1961
996	Giant	1956
997	From Here to Eternity	1953
998	Lifeboat	1944
999	The 39 Steps	1935

1000 rows × 2 columns

In [20]:
```
print('shape of netflixMovie_df  : ', netflixMovie_df.shape)
[illegible]
```

```
shape of netflixMovie_df   :  (17770, 2)
shape of imdbMovie_df      :  (1000, 2)
```

3. merge netflixMovie_df & imdbMovie_df to form yearMovie_df

In [21]:
```
# now merge the two dataframes into yearMovie_df

yearMovie_df = pd.concat([netflixMovie_df, imdbMovie_df], axis=0)
```

In [22]:

Out[22]:

	Movie_Title	Released_Year
0	Dinosaur Planet	2003
1	Isle of Man TT 2004 Review	2004
2	Character	1997
3	Paula Abdul's Get Up & Dance	1994
4	The Rise and Fall of ECW	2004
...	...	...
995	Breakfast at Tiffany's	1961
996	Giant	1956
997	From Here to Eternity	1953
998	Lifeboat	1944
999	The 39 Steps	1935

18770 rows × 2 columns

In [23]:

```
# sort yearMovie_df on the basis of 'Released_Year'
```

In [24]:

Out[24]:

	Movie_Title	Released_Year
17666	Eros Dance Dhamaka	1915
7653	Lumiere Brothers' First Films	1915
13146	Chaplin's Essanay Comedies: Vol. 1	1915
8820	The Birth of a Nation	1915
14686	Chaplin's Essanay Comedies: Vol. 2	1915
...	...	...

	Movie_Title	Released_Year
612	The Trial of the Chicago 7	2020
205	Soul	2020
20	Soorarai Pottru	2020
18	Hamilton	2020
613	Druk	2020

In [25]:
```
# add 'User_id' to yearMovie_df

[illegible]
```

In [26]:
```
[illegible]
```

Out[26]:

	Movie_Title	Released_Year	User_Id
17666	Eros Dance Dhamaka	1915	0
7653	Lumiere Brothers' First Films	1915	1
13146	Chaplin's Essanay Comedies: Vol. 1	1915	2
8820	The Birth of a Nation	1915	3
14686	Chaplin's Essanay Comedies: Vol. 2	1915	4
...	...	...	...
612	The Trial of the Chicago 7	2020	18765
205	Soul	2020	18766
20	Soorarai Pottru	2020	18767
18	Hamilton	2020	18768
613	Druk	2020	18769

18770 rows × 3 columns

In [27]:
```
[illegible]
```

In [28]:

Out[28]:
```
array([['Eros Dance Dhamaka', 1915, 0],
       ["Lumiere Brothers' First Films", 1915, 1],
       ["Chaplin's Essanay Comedies: Vol. 1", 1915, 2],
       ...,
       ['Soorarai Pottru', 2020, 18767],
       ['Hamilton', 2020, 18768],
       ['Druk', 2020, 18769]], dtype=object)
```

4. Create a pivot table movieUser_df

In [29]:
```
[illegible]
```

In [30]:
```
[illegible]
```

Out[30]:

	Movie_Title	Released_Year	User_Id
17666	Eros Dance Dhamaka	1915	0
7653	Lumiere Brothers' First Films	1915	1
13146	Chaplin's Essanay Comedies: Vol. 1	1915	2
8820	The Birth of a Nation	1915	3
14686	Chaplin's Essanay Comedies: Vol. 2	1915	4
...	...	...	...
612	The Trial of the Chicago 7	2020	18765
205	Soul	2020	18766
20	Soorarai Pottru	2020	18767
18	Hamilton	2020	18768
613	Druk	2020	18769

17867 rows × 3 columns

In [31]:
```
# drop first 12867 rows
N = 12867
```

In [32]:

Out[32]:

	Movie_Title	Released_Year	User_Id
3591	The North Face Expeditions: Everest and Bonus ...	2001	13543
2292	Gaudi Afternoon	2001	13544
15523	A Woman's a Helluva Thing	2001	13545
12669	Absolutely Fabulous: Series 4	2001	13546
5176	Abandoned	2001	13547
...	...	...	...
612	The Trial of the Chicago 7	2020	18765
205	Soul	2020	18766
20	Soorarai Pottru	2020	18767
18	Hamilton	2020	18768
613	Druk	2020	18769

5000 rows × 3 columns

In [33]:

Out[33]:
```
'Dinosaur Planet'
```

In [34]:
```
# reserialize 'User_Id'
```

```
C:\Users\adiso\AppData\Local\Temp\ipykernel_13696\2219890618.py:3: SettingWithCopyWarning:
```

In [35]:
```
movieUser_df = pd.pivot_table(yearMovie_df, index='Movie_Title', columns='User_Id', values='Released_Year')
```

In [36]:

Out[36]:

User_Id	0	1	2	3	4	5	6	7	8	9	...	4990	4991	4992	4993	4994	4995	4996	4997	4998	49
Movie_Title																					
'N Sync: PopOdyssey Live	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
(500) Days of Summer	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
.Com for Murder	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
.hack//Legend of the Twilight	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
.hack//SIGN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Zus & Zo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
Zwartboek	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
e-Dreams	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
s-Cry-ed	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
Ôkami kodomo no Ame to Yuki	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

5000 rows × 5000 columns

In [37]:
```
# drop last
N = 3000
```

In [38]:
```
[illegible]
```

Out[38]:

User_Id	0	1	2	3	4	5	6	7	8	9	...	1990	1991	1992	1993	1994	1995	1996	1997	1998	19
Movie_Title																					
'N Sync: PopOdyssey Live	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
(500) Days of Summer	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
.Com for Murder	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
.hack//Legend of the Twilight	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
.hack//SIGN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Zus & Zo	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
Zwartboek	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
e-Dreams	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
s-Cry-ed	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
Ôkami kodomo no Ame to Yuki	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

5000 rows × 2000 columns

5. make movieUser_df sparse

In [39]:
```
[illegible]
```

In [40]:

Out[40]:
```
'(500) Days of Summer'
```

In [41]:
```python
# create a list containing all movie names

movieList=[]
for i in range(len(movieUser_df.index)):
```

In [42]:
```python
# store the values in userMovie_df
```

In [43]:
```python
# make it sparse
```

In [44]:
```python
# fill all 'nan' values with 0
```

Out[44]:

	0	1	2	3	4	5	6	7	8	9	...	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
'N Sync: PopOdyssey Live	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
(500) Days of Summer	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
.Com for Murder	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
.hack//Legend of the Twilight	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
.hack//SIGN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Zus & Zo	0	0	0	0	3	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Zwartboek	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
e-Dreams	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
s-Cry-ed	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Ôkami kodomo no Ame to Yuki	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5000 rows × 2000 columns

6. Core Logic of Recommender System using Binary Search

In [45]:
```
# this function returns the last index of highest valued rating, and the correponding movie name

def lastIndexOfTopRatedMoviesByUserX(user_series_of_movies, rating, l, h):
    ans_idx = -1
    while l <= h:
        mid = l + (h-l)//2
        if user_series_of_movies[mid] >= rating:
            ans_idx = mid
            last_coordinated_movie = user_series_of_movies.index[mid]
            l = mid + 1
        else:
            h = mid - 1
```

In [103]:
```
# n = number of movies per top ratings of user
# u = 'User_id'
# rating = lowest best rating -> [1, 5]
# last_coordinated_movie -> name of last highly rated movie by the user-X
n = 5
u = 0
rating = 4
# last_coordinated_movie
```

In [104]:

Out[104]:

	0	1	2	3	4	5	6	7	8	9	...	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
Bubble Boy	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Boom	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Led Zeppelin	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
The Phantom of the Opera: Special Edition	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Two Weeks Notice	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

	0	1	2	3	4	5	6	7	8	9	...	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
Goldfish Memory	0	0	0	0	0	0	0	4	0	0	...	0	0	0	0	0	0	0	0	0	0
Gokusen	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	5	0	0	0	0
Good Times	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Gojoe: Spirit War Chronicle	0	0	5	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Ôkami kodomo no Ame to Yuki	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

In [105]:
```
if 'Avatar' in movieUser_df.index:
```
```
True
```

In [106]:

In [107]:
```
# call the lastIndexOfTopRatedMoviesByUserX
l = 0
h = len(movieUser_df.columns)-1
```

In [108]:
```
print("the index of last highly rated movie by the user-X: ", idx)
```
```
the index of last highly rated movie by the user-X:  23
the name of last highly rated movie by the user-X:  Renegade
```

In [109]:
```
# verify the name of the movie received
```
Out[109]:
```
'Renegade'
```

In [110]:

Out[110]:
```
4
```

In [112]:

Out[112]:
```
3
```

In [113]:
```
# Hence the calculation is correct
```

7. Recommendation code returning a list of movies

In [114]:
```
# Helper function to return whether the movie is rated or not to avoid recommending already rated movie

def isRated(movie_name):
    if movieUser_df[u][movie_name] > 0:
        return True
```

In [115]:
```
# This function returns:-
# 1. the recommended movie list
# 2. the year of the movie in year sorted yearMovie_df OR, the year of last_coordinated_movie for verificatio

def recommendMovies(u, n, idx, last_coordinated_movie, yearMovie_df_array):
    movie_list = []
    pivot_movie_idx = -1
    pivot_movie_year = -1
    for i in range(len(yearMovie_df_array)):
        j = len(yearMovie_df_array) - i - 1
        if(yearMovie_df_array[i][0] == last_coordinated_movie):
            pivot_movie_idx = i
            pivot_movie_year = yearMovie_df_array[i][1]
            print(yearMovie_df_array[i][0])
            break
        if(yearMovie_df_array[j][0] == last_coordinated_movie):
            pivot_movie_idx = j
            pivot_movie_year = yearMovie_df_array[j][1]
            print(yearMovie_df_array[j][0])
            break


    # store closest movies greater than or equal to current year
    right_movie_cnt = 0
    right_starter_idx = pivot_movie_idx + 1
    while right_movie_cnt < n:
        if(right_starter_idx > len(yearMovie_df_array)-1):
            break

        if(isRated(yearMovie_df_array[right_starter_idx][0]) == False):
            movie_list.append( (yearMovie_df_array[right_starter_idx][0], yearMovie_df_array[right_starter_id
            right_movie_cnt += 1

        right_starter_idx += 1



    # store closest movies less than or equal to current year
    left_movie_cnt = 0
    left_starter_idx = pivot_movie_idx - 1
    while left_movie_cnt < n:
        if(left_starter_idx == 0):
            break

        if(isRated(yearMovie_df_array[left_starter_idx][0]) == False):

            movie_list.append( (yearMovie_df_array[left_starter_idx][0], yearMovie_df_array[left_starter_idx]
            left_movie_cnt += 1

        left_starter_idx -= 1

[illegible]
```

In [116]:
```
# convert yearMovie_df -> numpy array
[illegible]
```

In [117]:
```
[illegible]
```

Out[117]:
```
5000
```

In [118]:
```
[illegible]
```

Out[118]:
```
array([['The North Face Expeditions: Everest and Bonus Footage', 2001, 0],
       ['Gaudi Afternoon', 2001, 1],
       ["A Woman's a Helluva Thing", 2001, 2],
       ...,
       ['Soorarai Pottru', 2020, 4997],
       ['Hamilton', 2020, 4998],
       ['Druk', 2020, 4999]], dtype=object)
```

In [119]:
```
# movieUser_df[0]['Hamilton']
```

In [120]:
```
# idx variable contains the last index of highest valued rating
# last_coordinated_movie contains the name of the last highly rated movie by user-X
# pivot_movie_year = the year of movie obtained as last_coordinated_movie

[illegible]
```

```
Renegade
```

In [125]:
```
[illegible]
```

Out[125]:
```
[('Denise Austin: Personal Training System', 2004),
 ('The O.C.: Season 2', 2004),
 ('Two Brothers and a Bride', 2004)]
```

In [122]:
```
# validate the year of obtained movie with obve recommended movies
pivot_movie_year

# the movie year which was predicted for the selected user it is recommending the movies closest to this year
[illegible]
```

Out[122]:
```
2004
```

In [123]:
```
# Here is the important part
# we can verify that all the recommendations above are "NON-RATED"
# if the rating value in movieUser_df = 0 of any user 'u' (for any movie above)
# then that must mean the movie is of course NON-RATED

[illegible]
```

Out[123]:
```
0
```

In [124]:
```
# verify the recommendations are unrater

[illegible]
```

Out[124]:
```
0
```

In [ ]:
```
```