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

In [3]:
```
movies = pd.read_csv('movies.csv')
print('Shape of this dataset :',movies.shape)
movies.head()
```

```
Shape of this dataset : (9742, 3)
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:
```
ratings = pd.read_csv('ratings.csv')
print('Shape of this dataset :',ratings.shape)
ratings.head()
```

```
Shape of this dataset : (100836, 4)
```

Out[4]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [5]:
```
users = pd.read_csv('users.csv')
print('Shape of this dataset :',users.shape)
users.head()
```

```
Shape of this dataset : (6040, 1)
```

Out[5]:

	userId;gender;age;occupation;zip-code
0	1;F;1;10;48067
1	2;M;56;16;70072
2	3;M;25;15;55117
3	4;M;45;7;2460
4	5;M;25;20;55455

In [6]:
```
rating_pivot = ratings.pivot_table(values='rating',columns='userId',index='movie
print('Shape of this pivot table :',rating_pivot.shape)
rating_pivot.head()
```

```
Shape of this pivot table : (9724, 610)
```

Out[6]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	60
movieId																			
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2
2	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2
3	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
4	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	0.0	0.0	0.0	0.0	0
5	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0

5 rows × 610 columns

In [7]:
```
from sklearn.neighbors import NearestNeighbors
nn_algo = NearestNeighbors(metric='cosine')
nn_algo.fit(rating_pivot)
```

Out[7]:
```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

In [8]:
```
class Recommender:
    def __init__(self):
        # This list will stored movies that called atleast ones using recommend_
        self.hist = []
        self.ishist = False # Check if history is empty

    # This method will recommend movies based on a movie that passed as the para
    def recommend_on_movie(self,movie,n_reccomend = 5):
        self.ishist = True
        movieid = int(movies[movies['title']==movie]['movieId'])
        self.hist.append(movieid)
        distance,neighbors = nn_algo.kneighbors([rating_pivot.loc[movieid]],n_ne
        movieids = [rating_pivot.iloc[i].name for i in neighbors[0]]
        recommeds = [str(movies[movies['movieId']==mid]['title']).split('\n')[0]
        return recommeds[:n_reccomend]

    # This method will recommend movies based on history stored in self.hist lis
    def recommend_on_history(self,n_reccomend = 5):
        if self.ishist == False:
            return print('No history found')
        history = np.array([list(rating_pivot.loc[mid]) for mid in self.hist])
        distance,neighbors = nn_algo.kneighbors([np.average(history,axis=0)],n_n
        movieids = [rating_pivot.iloc[i].name for i in neighbors[0]]
        recommeds = [str(movies[movies['movieId']==mid]['title']).split('\n')[0]
        return recommeds[:n_reccomend]
```

In [9]:
```
# Linitializing the Recommender Object
recommender = Recommender()
```

In [10]:
```
# Recommendation based on past watched movies, but the object just initialized. 
recommender.recommend_on_history()
```

```
No history found
```

In [11]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('Father of the Bride Part II (1995)')
```

Out[11]:
```
['Sabrina (1995)',
 'Juror, The (1996)',
 'Striptease (1996)',
 "Mr. Holland's Opus (1995)",
 'Grumpier Old Men (1995)']
```

In [12]:
```
# Recommendation based on past watched movies, and this time a movie is there in
recommender.recommend_on_history()
```

Out[12]:
```
['Sabrina (1995)',
 'Juror, The (1996)',
 'Striptease (1996)',
 "Mr. Holland's Opus (1995)",
 'Grumpier Old Men (1995)']
```

In [13]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('Tigerland (2000)')
```

Out[13]:
```
['Tsotsi (2005)',
 'Shape of Things, The (2003)',
 'Malèna (2000)',
 'Max (2002)',
 'Dancer Upstairs, The (2002)']
```

In [14]:
```
# Recommendation based on past watched movies, and this time two movies is there
recommender.recommend_on_history()
```

Out[14]:
```
['Sabrina (1995)',
 'Juror, The (1996)',
 'Striptease (1996)',
 'Grumpier Old Men (1995)',
 'Willy Wonka & the Chocolate Factory (1971)']
```

In [18]:
```
# Recommendation based on past watched movies, and this time three movies is ther
recommender.recommend_on_history()
```

Out[18]:
```
['Sabrina (1995)',
 'Juror, The (1996)',
 'Striptease (1996)',
 'Grumpier Old Men (1995)',
 'Willy Wonka & the Chocolate Factory (1971)']
```

In [19]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('Money Train (1995)')
```

Out[19]:
```
['Ali G Indahouse (2002)',
 'Cube Zero (2004)',
 'Major League II (1994)',
 'Savages (2012)',
 'Knights of Badassdom (2013)']
```

In [20]:
```
# Recommendation based on past watched movies, and this time four movies is there
recommender.recommend_on_history()
```

Out[20]:
```
['Sabrina (1995)',
 'Woman in Red, The (1984)',
 'Down and Out in Beverly Hills (1986)',
 'Dream Team, The (1989)',
 'Twister (1996)']
```

In [21]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('GoldenEye (1995)')
```

Out[21]:
```
['Die Hard: With a Vengeance (1995)',
 'True Lies (1994)',
 'Clear and Present Danger (1994)',
 'Speed (1994)',
 'Batman (1989)']
```

In [22]:
```
# Recommendation based on past watched movies, and this time five movies is there
recommender.recommend_on_history()
```

Out[22]:
```
['Die Hard: With a Vengeance (1995)',
 'Mission: Impossible (1996)',
 'Speed (1994)',
 'True Lies (1994)',
 'Clear and Present Danger (1994)']
```

In [23]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(stop_words='english')
genres = vectorizer.fit_transform(movies.genres).toarray()
contents = pd.DataFrame(genres,columns=vectorizer.get_feature_names())
print('Shape of the content table :',contents.shape)
contents.head()
```

```
Shape of the content table : (9742, 23)
```

Out[23]:

	action	adventure	animation	children	comedy	crime	documentary	drama	fantasy	fi	...	in
0	0	1	1	1	1	0	0	0	1	0	...	
1	0	1	0	1	0	0	0	0	1	0	...	
2	0	0	0	0	1	0	0	0	0	0	...	
3	0	0	0	0	1	0	0	1	0	0	...	
4	0	0	0	0	1	0	0	0	0	0	...	

5 rows × 23 columns

In [24]:
```
from sklearn.neighbors import NearestNeighbors
nn_algo = NearestNeighbors(metric='cosine')
nn_algo.fit(contents)
```

Out[24]:
```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                 radius=1.0)
```

In [25]:
```
class Recommender:
    def __init__(self):
        # This list will stored movies that called atleast ones using recommend_
        self.hist = []
        self.ishist = False # Check if history is empty

    # This method will recommend movies based on a movie that passed as the para
    def recommend_on_movie(self,movie,n_reccomend = 5):
        self.ishist = True
        iloc = movies[movies['title']==movie].index[0]
        self.hist.append(iloc)
        distance,neighbors = nn_algo.kneighbors([contents.iloc[iloc]],n_neighbor
        recommeds = [movies.iloc[i]['title'] for i in neighbors[0] if i not in [
        return recommeds[:n_reccomend]

    # This method will recommend movies based on history stored in self.hist lis
    def recommend_on_history(self,n_reccomend = 5):
        if self.ishist == False:
            return print('No history found')
        history = np.array([list(contents.iloc[iloc]) for iloc in self.hist])
        distance,neighbors = nn_algo.kneighbors([np.average(history,axis=0)],n_n
        recommeds = [movies.iloc[i]['title'] for i in neighbors[0] if i not in s
        return recommeds[:n_reccomend]
```

In [27]:
```
# Linitializing the Recommender Object
recommender = Recommender()
```

In [28]:
```
# Recommendation based on past watched movies, but the object just initialized. 
recommender.recommend_on_history()
```

```
No history found
```

In [29]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('Father of the Bride Part II (1995)')
```

Out[29]:
```
['Castle, The (1997)',
 'Take This Job and Shove It (1981)',
 'Wake Up, Ron Burgundy (2004)',
 "Trippin' (1999)",
 'Caddyshack II (1988)']
```

In [30]:
```
# Recommendation based on past watched movies, and this time a movie is there in
recommender.recommend_on_history()
```

Out[30]:
```
['Castle, The (1997)',
 'Take This Job and Shove It (1981)',
 'Wake Up, Ron Burgundy (2004)',
 "Trippin' (1999)",
 'Caddyshack II (1988)']
```

In [31]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('Tigerland (2000)')
```

Out[31]:
```
['Town is Quiet, The (Ville est tranquille, La) (2000)',
 'Life as a House (2001)',
 'Eros (2004)',
 '3 Women (Three Women) (1977)',
 'Pelé: Birth of a Legend (2016)']
```

In [32]:
```
# Recommendation based on past watched movies, and this time two movies is there
recommender.recommend_on_history()
```

Out[32]:
```
['High Heels (Tacones lejanos) (1991)',
 'Pygmalion (1938)',
 'Project X (1987)',
 'Rocket Singh: Salesman of the Year (2009)',
 'TV Set, The (2006)']
```

In [34]:
```
# Recommendation based on past watched movies, and this time three movies is ther
recommender.recommend_on_history()
```

Out[34]:
```
['High Heels (Tacones lejanos) (1991)',
 'Pygmalion (1938)',
 'Project X (1987)',
 'Rocket Singh: Salesman of the Year (2009)',
 'TV Set, The (2006)']
```

In [35]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('Money Train (1995)')
```

Out[35]:
```
['Last Boy Scout, The (1991)',
 'Metro (1997)',
 'Another 48 Hrs. (1990)',
 'Bad Boys (1995)',
 'Wasabi (2001)']
```

In [36]:
```
# Recommendation based on past watched movies, and this time four movies is there
recommender.recommend_on_history()
```

Out[36]:
```
['Another 48 Hrs. (1990)',
 'Bad Boys (1995)',
 'Wasabi (2001)',
 'Metro (1997)',
 'Last Boy Scout, The (1991)']
```

In [37]:
```
# Recommendation based on this movie
recommender.recommend_on_movie('GoldenEye (1995)')
```

Out[37]:
```
['Surviving the Game (1994)',
 'Broken Arrow (1996)',
 'Octopussy (1983)',
 'Mission: Impossible - Fallout (2018)',
 'Mission: Impossible II (2000)']
```

In [38]:
```
# Recommendation based on past watched movies, and this time five movies is there
recommender.recommend_on_history()
```

Out[38]:
```
['Last Boy Scout, The (1991)',
 'Another 48 Hrs. (1990)',
 'Wasabi (2001)',
 'Bad Boys (1995)',
 'Hunting Party, The (2007)']
```

In [ ]: