In [1]:
```
# DSC 630
# Assignment Week 10
# Author :  Akila Selvaraj
# Created Date : 11/01/2022

# Change log
# Author: Akila Selvaraj
# Description : Initial version
```

In [2]:
```
# Loading the data
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# rating.csv that contains ratings of movies by users
ratings = pd.read_csv('ratings.csv')
ratings.head(10)
```

Out[3]:

	userId	movieId	rating	timestamp
0	1	16	4.0	1217897793
1	1	24	1.5	1217895807
2	1	32	4.0	1217896246
3	1	47	4.0	1217896556
4	1	50	4.0	1217896523
5	1	110	4.0	1217896150
6	1	150	3.0	1217895940
7	1	161	4.0	1217897864
8	1	165	3.0	1217897135
9	1	204	0.5	1217895786

In [4]:
```
# movie.csv that contains movie information
movies = pd.read_csv("movies.csv")
movies.head(10)
```

Out[4]:

	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
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller

In [5]:
```
# Merging both the datasets on movieId
data = ratings.merge(movies,on='movieId', how='left')
data.head(10)
```

Out[5]:

	userId	movieId	rating	timestamp	title	genres
0	1	16	4.0	1217897793	Casino (1995)	Crime|Drama
1	1	24	1.5	1217895807	Powder (1995)	Drama|Sci-Fi
2	1	32	4.0	1217896246	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
3	1	47	4.0	1217896556	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	4.0	1217896523	Usual Suspects, The (1995)	Crime|Mystery|Thriller
5	1	110	4.0	1217896150	Braveheart (1995)	Action|Drama|War
6	1	150	3.0	1217895940	Apollo 13 (1995)	Adventure|Drama|IMAX
7	1	161	4.0	1217897864	Crimson Tide (1995)	Drama|Thriller|War
8	1	165	3.0	1217897135	Die Hard: With a Vengeance (1995)	Action|Crime|Thriller
9	1	204	0.5	1217895786	Under Siege 2: Dark Territory (1995)	Action

In [6]:
```
# average rating for each and every movie in the dataset
Average_ratings = pd.DataFrame(data.groupby('title')['rating'].mean())
Average_ratings.head(10)
```

Out[6]:

title	rating
'71 (2014)	3.500000
'Hellboy': The Seeds of Creation (2004)	3.000000
'Round Midnight (1986)	2.500000
'Til There Was You (1997)	4.000000
'burbs, The (1989)	3.125000
'night Mother (1986)	3.000000
(500) Days of Summer (2009)	3.932432
***batteries not included (1987)**	3.318182
...And Justice for All (1979)	3.650000
10 (1979)	2.000000

In [7]:
```
# total ratings cast for each movie
Average_ratings['Total Ratings'] = pd.DataFrame(data.groupby('title')['rating'].c
Average_ratings.head(10)
```

Out[7]:

title	rating	Total Ratings
'71 (2014)	3.500000	1
'Hellboy': The Seeds of Creation (2004)	3.000000	1
'Round Midnight (1986)	2.500000	1
'Til There Was You (1997)	4.000000	3
'burbs, The (1989)	3.125000	20
'night Mother (1986)	3.000000	1
(500) Days of Summer (2009)	3.932432	37
***batteries not included (1987)**	3.318182	11
...And Justice for All (1979)	3.650000	10
10 (1979)	2.000000	3

In [8]:
```
# Calculating The Correlation
# creating a table where the rows are userIds and the columns represent the movie
# The values of the matrix represent the rating for each movie by each user.
movie_user = data.pivot_table(index='userId',columns='title',values='rating')
movie_user.head(10)
```

Out[8]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Til There Was You (1997)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...And Justice for All (1979)	10 (1979)	...
userId											
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

10 rows × 10323 columns

In [9]:
```
# we need to select a movie to test our recommender system.
# Choose any movie title from the data. Here, I chose Toy Story (1995)
correlations = movie_user.corrwith(movie_user['Toy Story (1995)'])
correlations.head()
```

Out[9]:
```
title
'71 (2014)                                    NaN
'Hellboy': The Seeds of Creation (2004)       NaN
'Round Midnight (1986)                        NaN
'Til There Was You (1997)                     NaN
'burbs, The (1989)                       0.470402
dtype: float64
```

In [10]:
```
# Now we will remove all the empty values and merge the total ratings to the corr
recommendation = pd.DataFrame(correlations,columns=['Correlation'])
recommendation.dropna(inplace=True)
recommendation = recommendation.join(Average_ratings['Total Ratings'])
recommendation.head()
```

Out[10]:

title	Correlation	Total Ratings
'burbs, The (1989)	0.470402	20
(500) Days of Summer (2009)	0.301871	37
***batteries not included (1987)**	-0.058926	11
...And Justice for All (1979)	-0.046829	10
10 (1979)	0.000000	3

In [11]:
```
# Testing The Recommendation System
# Let's filter all the movies with a correlation value to Toy Story (1995) and wi
recc = recommendation[recommendation['Total Ratings']>100].sort_values('Correlati
```

In [12]:
```
# Merging the movies dataset for verifying the recommendations.
recc = recc.merge(movies,on='title', how='left')
recc.head(10)
```

Out[12]:

	title	Correlation	Total Ratings	movieId	genres
0	Toy Story (1995)	1.000000	232	1	Adventure|Animation|Children|Comedy|Fantasy
1	Toy Story 2 (1999)	0.709677	104	3114	Adventure|Animation|Children|Comedy|Fantasy
2	Austin Powers: The Spy Who Shagged Me (1999)	0.580651	117	2683	Action|Adventure|Comedy
3	Crimson Tide (1995)	0.578642	107	161	Drama|Thriller|War
4	Austin Powers: International Man of Mystery (1...	0.533061	101	1517	Action|Adventure|Comedy
5	Bug's Life, A (1998)	0.506905	102	2355	Adventure|Animation|Children|Comedy
6	Babe (1995)	0.504718	129	34	Children|Drama
7	Shakespeare in Love (1998)	0.498968	106	2396	Comedy|Drama|Romance
8	Who Framed Roger Rabbit? (1988)	0.498878	115	2987	Adventure|Animation|Children|Comedy|Crime|Fant...
9	Mrs. Doubtfire (1993)	0.477007	166	500	Comedy|Drama

We can see that the top recommendations are pretty good. The movie that has the highest/full correlation to Toy Story is Toy Story itself. The movies such as The Incredibles, Finding Nemo and Alladin show high correlation with Toy Story.

Model 2

In [13]:
```
final_dataset=ratings.pivot(index="movieId",columns="userId",values="rating")
final_dataset.head()
```

Out[13]:

userId	1	2	3	4	5	6	7	8	9	10	...	659	660	661	662	66
movieId																
1	NaN	5.0	NaN	NaN	4.0	NaN	NaN	5.0	NaN	NaN	...	NaN	NaN	4.0	5.0	3
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	5.0	Na
3	NaN	2.0	NaN	NaN	NaN	NaN	NaN	4.0	3.0	NaN	...	NaN	NaN	3.0	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	Na
5	NaN	3.0	3.0	NaN	NaN	NaN	NaN	3.0	NaN	NaN	...	NaN	NaN	3.0	NaN	Na

5 rows × 668 columns

In [14]:
```
final_dataset.fillna(0,inplace=True)
final_dataset.head()
```

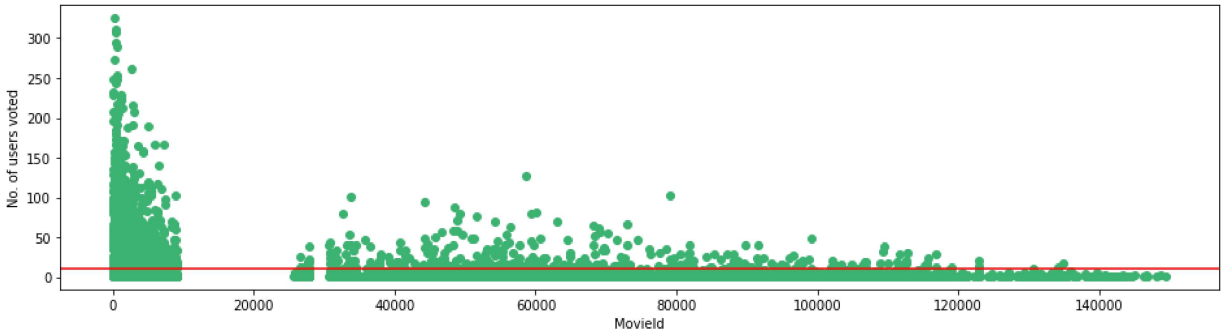
Out[14]:

userId	1	2	3	4	5	6	7	8	9	10	...	659	660	661	662	663	664	665	66
movieId																			
1	0.0	5.0	0.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0	...	0.0	0.0	4.0	5.0	3.0	0.0	0.0	0
2	0.0	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
3	0.0	2.0	0.0	0.0	0.0	0.0	0.0	4.0	3.0	0.0	...	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
5	0.0	3.0	3.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	...	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0

5 rows × 668 columns

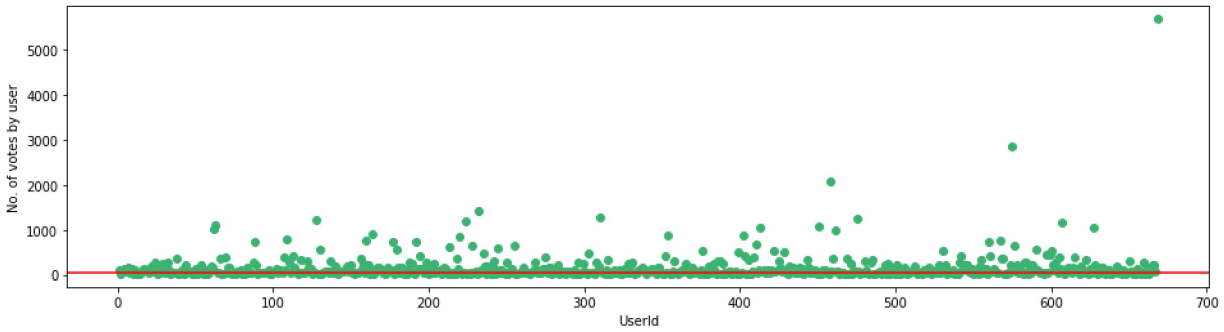
In [15]:
```
# Visualization of the data
no_user_voted=ratings.groupby('movieId')['rating'].agg('count')
no_movies_voted=ratings.groupby('userId')['rating'].agg('count')
```

In [16]:
```
f,ax=plt.subplots(1,1,figsize=(16,4))
# ratings['rating'].plot(kind='hist')
plt.scatter(no_user_voted.index,no_user_voted,color='mediumseagreen')
plt.axhline(y=10,color='r')
plt.xlabel('MovieId')
plt.ylabel('No. of users voted')
plt.show()
```



In [17]:
```
final_dataset=final_dataset.loc[no_user_voted[no_user_voted > 10].index,:]
```

In [18]:
```
f,ax = plt.subplots(1,1,figsize=(16,4))
plt.scatter(no_movies_voted.index,no_movies_voted,color='mediumseagreen')
plt.axhline(y=50,color='r')
plt.xlabel('UserId')
plt.ylabel('No. of votes by user')
plt.show()
```



In [19]:
```
final_dataset=final_dataset.loc[:,no_movies_voted[no_movies_voted >50].index]
final_dataset
```

Out[19]:

userId	1	3	4	5	6	7	8	9	11	15	...	656	657	659	661	662	664	665	66
movieId																			
1	0.0	0.0	0.0	4.0	0.0	0.0	5.0	0.0	4.0	0.0	...	0.0	0.0	0.0	4.0	5.0	0.0	0.0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3.0	0.0	0.0	0.0	5.0	0.0	0.0	0
3	0.0	0.0	0.0	0.0	0.0	0.0	4.0	3.0	0.0	0.0	...	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0
4	0.0	0.0	0.0	0.0	0.0	0.0	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	3.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	...	3.0	0.0	0.0	3.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
119145	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
122882	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
122892	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
134130	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
134853	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

2160 rows × 420 columns

In [20]:
```
# # Removing sparsity
# sample = np.array([[0,0,3,0,0],[4,0,0,0,2],[0,0,0,0,1]])
# sparsity = 1.0 - ( np.count_nonzero(sample) / float(sample.size) )
# print(sparsity)
```

In [21]:
```
# csr_sample=csr_matrix(sample)
# print(csr_sample)
```

In [22]:
```
# Removing sparsity
csr_data=csr_matrix(final_dataset.values)
final_dataset.reset_index(inplace=True)
```

In [28]:
```
# Making the movie recommendation system model
knn=NearestNeighbors(metric='cosine',algorithm='brute',n_neighbors=20,n_jobs=-1)
knn.fit(csr_data)
```

Out[28]:
```
NearestNeighbors(algorithm='brute', metric='cosine', n_jobs=-1, n_neighbors=20)
```

In [24]:
```
# Making the recommendation function
def get_movie_recommendation(movie_name):
    n_movies_to_reccomend = 10
    movie_list = movies[movies['title'].str.contains(movie_name)]
    if len(movie_list):
        movie_idx= movie_list.iloc[0]['movieId']
        movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
        distances , indices = knn.kneighbors(csr_data[movie_idx],n_neighbors=n_mo
        rec_movie_indices = sorted(list(zip(indices.squeeze().tolist(),distances.
        recommend_frame = []
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index
            recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'
        df = pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
        return df
```

In [25]:
```
# Let's Recommend some movies - Iron Man
get_movie_recommendation('Iron Man')
```

Out[25]:

	Title	Distance
1	Batman Begins (2005)	0.374727
2	WALL·E (2008)	0.370822
3	Watchmen (2009)	0.362019
4	300 (2007)	0.355342
5	Avengers, The (2012)	0.346966
6	Inception (2010)	0.340948
7	Bourne Ultimatum, The (2007)	0.335740
8	Casino Royale (2006)	0.307617
9	Star Trek (2009)	0.293528
10	Dark Knight, The (2008)	0.246059

In [26]:
```
# Let's Recommend some movies - Toy Story
get_movie_recommendation('Toy Story')
```

Out[26]:

	Title	Distance
1	Indiana Jones and the Last Crusade (1989)	0.388969
2	E.T. the Extra-Terrestrial (1982)	0.385306
3	Toy Story 2 (1999)	0.375542
4	Princess Bride, The (1987)	0.366165
5	Jurassic Park (1993)	0.363544
6	Back to the Future (1985)	0.360794
7	Star Wars: Episode V - The Empire Strikes Back...	0.355045
8	Raiders of the Lost Ark (Indiana Jones and the...	0.336430
9	Star Wars: Episode IV - A New Hope (1977)	0.332960
10	Star Wars: Episode VI - Return of the Jedi (1983)	0.312124

In [27]:
```
# Let's Recommend some movies - Jurassic Park
get_movie_recommendation('Jurassic Park')
```

Out[27]:

	Title	Distance
1	Star Wars: Episode IV - A New Hope (1977)	0.346537
2	Pretty Woman (1990)	0.346411
3	Silence of the Lambs, The (1991)	0.346194
4	Braveheart (1995)	0.327778
5	Batman (1989)	0.315648
6	True Lies (1994)	0.278189
7	Speed (1994)	0.267405
8	Fugitive, The (1993)	0.262388
9	Forrest Gump (1994)	0.261188
10	Terminator 2: Judgment Day (1991)	0.243140

Reference : https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset?select=movie.csv (https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset?select=movie.csv)