```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
```

```
print("This project is done by Sripathi Uday Shankar ") # Changed 'Print' to 'print'
```

```
This project is done by Sripathi Uday Shankar
```

```
movies=pd.read_csv("/content/movies_1.csv")
ratings=pd.read_csv("/content/rating_1.csv")
print(movies.head())
ratings.head()
```

```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                         Comedy|Drama|Romance
4                                       Comedy
```

	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

```
final_dataset=ratings.pivot(index='movieId',columns='userId',values='rating')
final_dataset.head()
```

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	NaN	NaN	NaN	4.0	NaN	4.5	NaN	NaN	NaN	...	4.0	NaN	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	NaN	NaN	NaN	NaN	NaN	4.0	NaN	4.0	NaN	NaN	...	NaN	4.0	NaN	5.0	3.5	NaN	NaN	2.0	NaN	NaN
3	4.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN

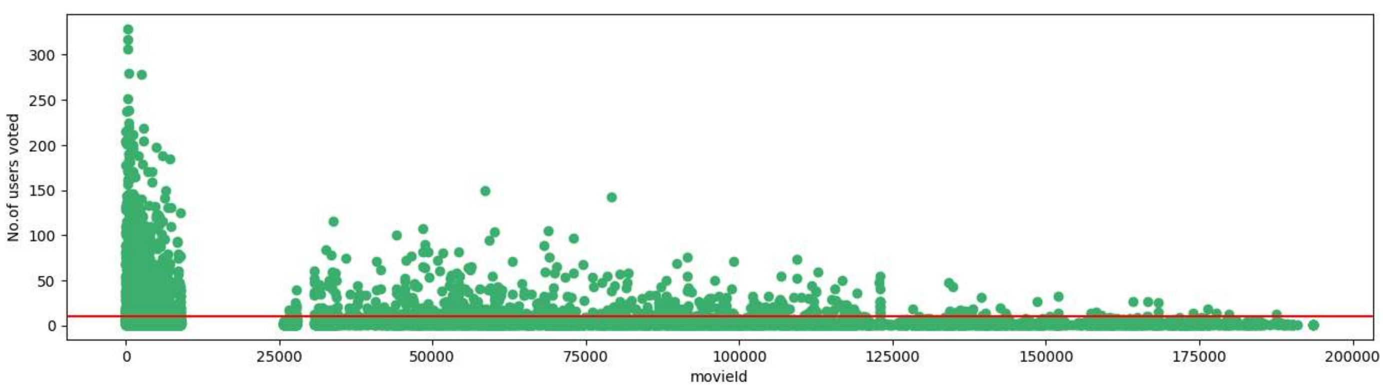
```
final_dataset.fillna(0,inplace=True)
final_dataset.head()
```

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
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	2.5	3.0	5.0
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	2.0	0.0	0.0
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	2.0	0.0	0.0
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	0.0	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	0.0	0.0	0.0

```
no_user_voted=ratings.groupby('movieId')['rating'].agg('count')
no_movie_voted=ratings.groupby('userId')['rating'].agg('count')
print(no_user_voted,no_movie_voted)
```

```
movieId
1         215
2         110
3          52
4           7
5          49
         ...
193581      1
193583      1
193585      1
193587      1
193609      1
Name: rating, Length: 9724, dtype: int64 userId
1       232
2        29
3        39
4       216
5        44
       ...
606    1115
607     187
608     831
609      37
610    1302
Name: rating, Length: 610, dtype: int64
```

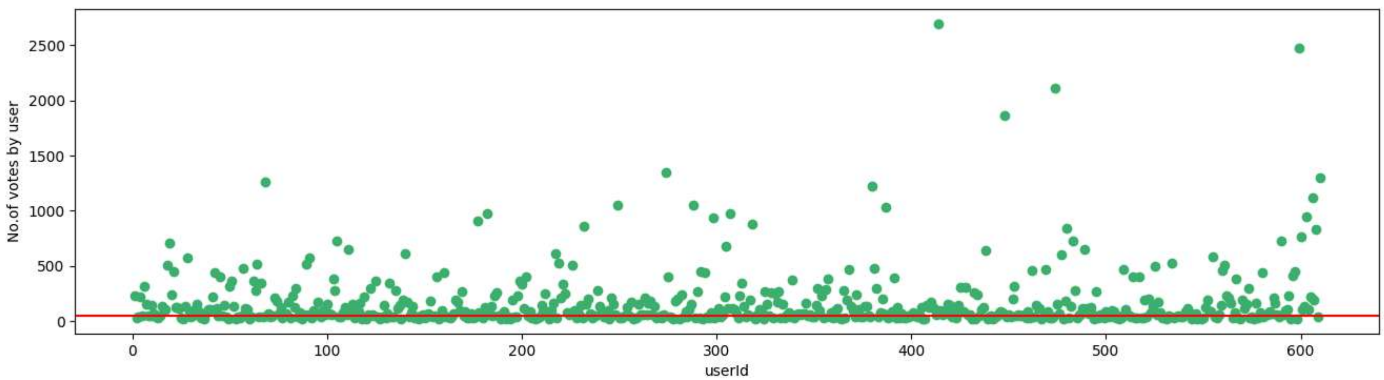
```
f,ax=plt.subplots(1,1, figsize=(16,4))
#ratings['rating'].plot(kind='hist)
plt.scatter(no_user_voted.index,no_user_voted,color='mediumseagreen')
plt.axhline(y=10,color='r')
plt.xlabel('movieId')
plt.ylabel('No.of users voted')
plt.show()
```




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

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
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	2.5	3.0	5.0
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	2.0	0.0	0.0
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	2.0	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	0.0	0.0	0.0
6	4.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...	0.0	3.0	4.0	3.0	0.0	0.0	0.0	0.0	0.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
174055	0.0	0.0	0.0	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	0.0	0.0
176371	0.0	0.0	0.0	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	0.0	0.0
177765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179819	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	0.0	0.0	0.0
187593	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	0.0	0.0	0.0

```
f,ax=plt.subplots(1,1, figsize=(16,4))
#ratings['rating'].plot(kind='hist')
plt.scatter(no_movie_voted.index,no_movie_voted,color='mediumseagreen')
plt.axhline(y=50,color='r')
plt.xlabel('userId')
plt.ylabel('No.of votes by user')
plt.show()
```




```
final_dataset=final_dataset.loc[:,no_movie_voted[no_movie_voted > 50].index]
final_dataset
```

userId	1	4	6	7	10	11	15	16	17	18	...	600	601	602	603	604	605	606	607	608	610
movieId																					
1	4.0	0.0	0.0	4.5	0.0	0.0	2.5	0.0	4.5	3.5	...	2.5	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	5.0
2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	...	4.0	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0
3	4.0	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	0.0	0.0	0.0	0.0	2.0	0.0
5	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.5	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
6	4.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0	0.0	4.0	...	0.0	0.0	3.0	4.0	3.0	0.0	0.0	0.0	0.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
174055	0.0	0.0	0.0	0.0	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	0.0
176371	0.0	0.0	0.0	0.0	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	0.0
177765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179819	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	0.0	0.0	0.0
187593	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	0.0	0.0	0.0

```
csr_data=csr_matrix(final_dataset.values)
final_dataset.reset_index(inplace=True)
```

```
knn=NearestNeighbors(metric='cosine',algorithm='brute',n_neighbors=20,n_jobs=-1)
knn.fit(csr_data)
```

NearestNeighbors

```
# Function to get recommendations
def get_recommendation(movie_name):
    movies_to_recommend = 10
    movie_list = movies[movies['title'].str.contains(movie_name, case=False, regex=False)]

    if not movie_list.empty:
        movie_idx = movie_list.iloc[0]['movieId']

        # Check if the movieId exists in final_dataset
        if movie_idx not in final_dataset['movieId'].values:
            return "Movie not found in the rating dataset."

        movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]

        distances, indices = knn.kneighbors(csr_data[movie_idx], n_neighbors=movies_to_recommend + 1)
        rec_movie_indices = sorted(
            list(zip(indices.squeeze().tolist(), distances.squeeze().tolist())),
            key=lambda x: x[1]
        )[1:]  # Exclude the first item (itself)

        recommend_frame = []
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index

            if not idx.empty:
                recommend_frame.append({'Title': movies.iloc[idx[0]]['title'], 'Distance': val[1]})

        df = pd.DataFrame(recommend_frame, index=range(1, len(recommend_frame) + 1))
        return df
    else:
        return "No movies found. Please check your input."


# Example test case
print(get_recommendation("Lion King, The"))
```

```
                                Title  Distance
1                      Aladdin (1992)  0.251999
2         Beauty and the Beast (1991)  0.253046
```

```
3                  Mrs. Doubtfire (1993)  0.324685
4                       Mask, The (1994)  0.342565
5                    Forrest Gump (1994)  0.349464
6                   Jurassic Park (1993)  0.350912
7                         Jumanji (1995)  0.377013
8   Snow White and the Seven Dwarfs (1937)  0.390670
9                       Toy Story (1995)  0.398578
10                     Home Alone (1990)  0.403325
```

```
# Example test case
print(get_recommendation("Batman"))
```

```
                               Title  Distance
1                      Batman (1989)  0.305549
2                   True Lies (1994)  0.359396
3   Ace Ventura: Pet Detective (1994)  0.384173
4               Jurassic Park (1993)  0.404032
5                   GoldenEye (1995)  0.405572
6                 Cliffhanger (1993)  0.408718
7                   Mask, The (1994)  0.409414
8                     Aladdin (1992)  0.426649
9              Lion King, The (1994)  0.427317
10  Die Hard: With a Vengeance (1995)  0.427554
```

```
# Example test case
print(get_recommendation("Iron Man"))
```

```
                            Title  Distance
1             Avengers, The (2012)  0.285319
2          Dark Knight, The (2008)  0.285835
3                    WALL·E (2008)  0.298138
4                Iron Man 2 (2010)  0.307492
5                    Avatar (2009)  0.310893
6             Batman Begins (2005)  0.362759
7                 Star Trek (2009)  0.366029
8                  Watchmen (2009)  0.368558
9   Guardians of the Galaxy (2014)  0.368758
10                       Up (2009)  0.368857
```

```
# Example test case
print(get_recommendation("Joe Black"))
```

```
                                   Title  Distance
1                   City of Angels (1998)  0.545736
2                 30 Days of Night (2007)  0.561262
3             Seven Years in Tibet (1997)  0.578296
4                 Cruel Intentions (1999)  0.590328
5                  What Women Want (2000)  0.592660
6            Six Days Seven Nights (1998)  0.595723
7                    Bachelor, The (1999)  0.596646
8                     Moulin Rouge (2001)  0.597911
9   Ever After: A Cinderella Story (1998)  0.598821
10                     Serendipity (2001)  0.600062
```

```
# Example test case
print(get_recommendation("Shawshank Redemption"))
```

```
                              Title  Distance
1                Forrest Gump (1994)  0.240724
2                Pulp Fiction (1994)  0.249804
3   Silence of the Lambs, The (1991)  0.300896
4            Schindler's List (1993)  0.329852
5                  Fight Club (1999)  0.336515
```