from google.colab import drive drive.mount('/content/drive', force_remount=True)

Movie Recommendation System

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np

# from sklearn.metrics.pairwise import cosine_similarity
```

```
movies=pd.read_csv('/content/drive/MyDrive/ML dataset/movies.csv')
ratings=pd.read_csv('/content/drive/MyDrive/ML dataset/ratings.csv')
```

```
dataset = ratings.pivot(index="movieId",columns="userId",values="rating")
```

```
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

5 rows × 610 columns

```
dataset.fillna(0,inplace=True)
```

```
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

5 rows × 610 columns

```
num_user_voted = ratings.groupby('movieId')['rating'].agg('count')
num_movies_voted = ratings.groupby('userId')['rating'].agg('count')
```

```
print(num_user_voted)
```

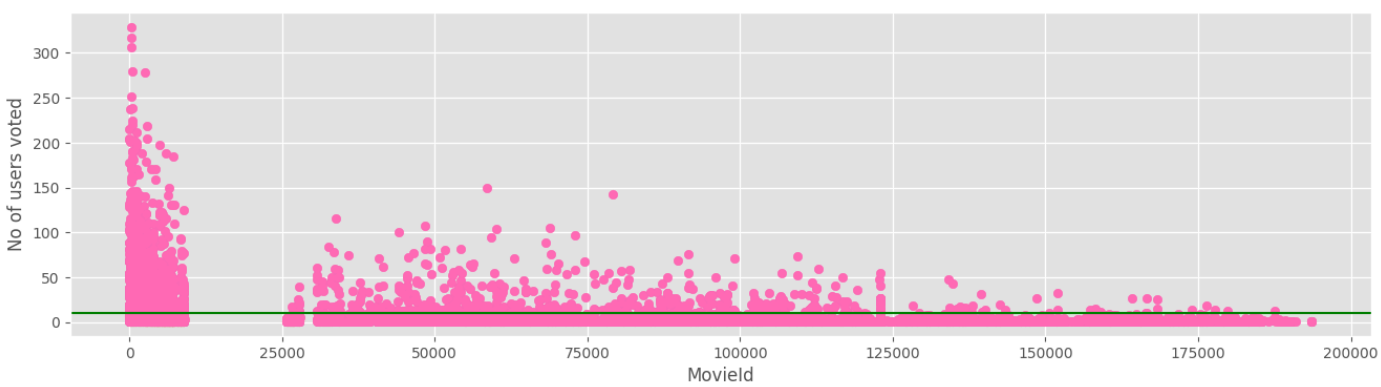
```
movieId
1         215
2         110
3          52
4           7
5          49
         ...
193581       1
```

```
193583      1
193585      1
193587      1
193609      1
Name: rating, Length: 9724, dtype: int64
```

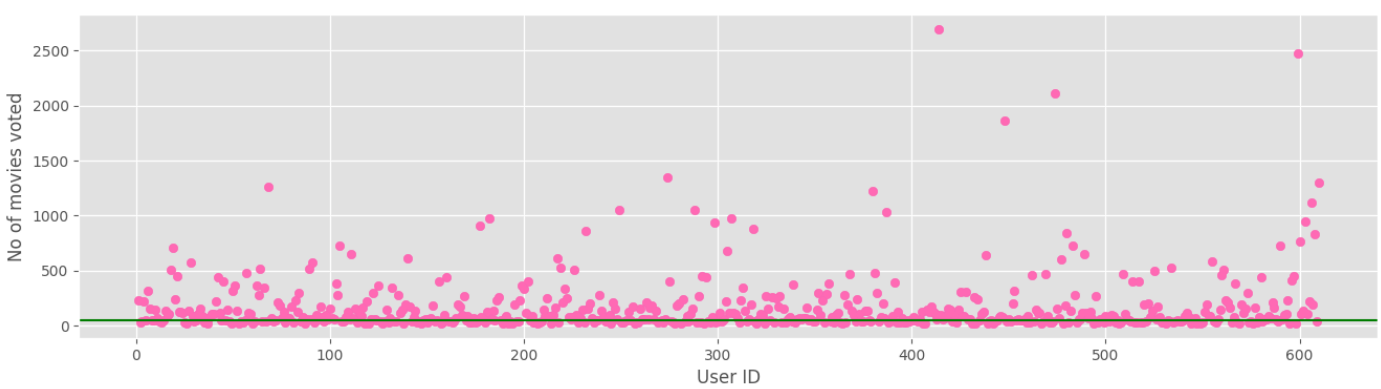
```
print(num_movies_voted)
```

```
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
```

```
import matplotlib.pyplot as plt
plt.style.use("ggplot")
fig,axes = plt.subplots(1,1, figsize=(16,4))
plt.scatter(num_user_voted.index, num_user_voted, color="hotpink")
plt.axhline(y=10, color='green')
plt.xlabel("MovieId")
plt.ylabel("No of users voted")
plt.show()
```




```
plt.style.use("ggplot")
fig,axes = plt.subplots(1,1, figsize=(16,4))
plt.scatter(num_movies_voted.index, num_movies_voted, color="hotpink")
plt.axhline(y=50, color='green')
plt.xlabel("User ID")
plt.ylabel("No of movies voted")
plt.show()
```



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

```
final_dataset = final_dataset.loc[: , num_movies_voted[num_movies_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

2121 rows x 378 columns

```
from scipy.sparse import csr_matrix
```

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

```
print(csr_data)
```

```
  (0, 0)        4.0
  (0, 3)        4.5
  (0, 6)        2.5
  (0, 8)        4.5
  (0, 9)        3.5
  (0, 10)       4.0
  (0, 12)       3.5
  (0, 16)       3.0
  (0, 19)       3.0
  (0, 20)       3.0
  (0, 25)       5.0
  (0, 28)       5.0
  (0, 29)       4.0
  (0, 31)       3.0
  (0, 34)       5.0
  (0, 38)       5.0
  (0, 39)       4.0
  (0, 40)       4.0
  (0, 41)       2.5
  (0, 43)       4.5
  (0, 46)       0.5
  (0, 47)       4.0
  (0, 50)       2.5
  (0, 53)       4.0
  (0, 55)       3.0
  :     :
  (2118, 205)   4.0
  (2118, 345)   1.5
  (2118, 357)   4.0
  (2118, 369)   4.5
  (2119, 37)    3.5
  (2119, 62)    3.0
  (2119, 98)    0.5
  (2119, 127)   4.5
  (2119, 156)   4.5
  (2119, 236)   0.5
  (2119, 256)   4.5
  (2119, 317)   2.0
  (2119, 345)   2.0
  (2119, 357)   5.0
```

```
(2119, 365)   3.5
(2120, 37)    4.0
(2120, 62)    5.0
(2120, 146)   2.5
(2120, 155)   4.5
(2120, 156)   5.0
(2120, 186)   5.0
(2120, 205)   4.0
(2120, 236)   3.0
(2120, 317)   3.5
(2120, 357)   4.0
```

```
from sklearn.neighbors import NearestNeighbors
```

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

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

```
def recommendation(movie_name):
    movie_list = movies[movies['title'].str.contains(movie_name)]
    print(movie_list)
    if len(movie_list):
        movie_index = movie_list.iloc[0]['movieId']
        movie_index = final_dataset[final_dataset['movieId'] == movie_index].index[0]

        distances, indices = knn.kneighbors(csr_data[movie_index], n_neighbors=11)

        indices_list = indices.squeeze().tolist()
        distances_list = distances.squeeze().tolist()
        index_distance_pairs = list(zip(indices_list, distances_list))
        rec_movies_indices = sorted(index_distance_pairs[1:], key=lambda x: x[1], reverse=True)

        recommended_movies = []
        for val in rec_movies_indices:
            movie_index = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_index].index
            recommended_movies.append({'Title': movies.iloc[idx]['title'].values[0], 'Distance': val[1]})
        df = pd.DataFrame(recommended_movies, index=range(1, 11))
        return df
    else:
        return "Movie not found..."
```

```
recommendation('Toy Story')
```

```
       movieId               title  \
0            1    Toy Story (1995)
2355      3114  Toy Story 2 (1999)
7355     78499  Toy Story 3 (2010)

                                            genres
0          Adventure|Animation|Children|Comedy|Fantasy
2355       Adventure|Animation|Children|Comedy|Fantasy
7355  Adventure|Animation|Children|Comedy|Fantasy|IMAX
```

	Title	Distance
1	Groundhog Day (1993)	0.399495
2	Lion King, The (1994)	0.398578
3	Pulp Fiction (1994)	0.398293
4	Star Wars: Episode VI - Return of the Jedi (1983)	0.390663
5	Apollo 13 (1995)	0.384633
6	Shrek (2001)	0.381472
7	Star Wars: Episode IV - A New Hope (1977)	0.380789
8	Toy Story 2 (1999)	0.371637
9	Forrest Gump (1994)	0.356542
10	Jurassic Park (1993)	0.334884

```
recommendation('Jumanji')
```

```
       movieId                                title  \
1            2                         Jumanji (1995)
9636    179401  Jumanji: Welcome to the Jungle (2017)

                          genres
1     Adventure|Children|Fantasy
9636   Action|Adventure|Children
```

	Title	Distance
1	Casper (1995)	0.474253
2	Stargate (1994)	0.469654
3	Nightmare Before Christmas, The (1993)	0.462612
4	Home Alone (1990)	0.443432
5	Beauty and the Beast (1991)	0.435007
6	Aladdin (1992)	0.425428
7	Jurassic Park (1993)	0.420563
8	Mrs. Doubtfire (1993)	0.416164
9	Mask, The (1994)	0.413743
10	Lion King, The (1994)	0.377013

```
! pip install streamlit -q
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 1.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.6/8.6 MB 50.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 68.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 4.8 MB/s eta 0:00:00
```

```
!wget -q -O - ipv4.icanhazip.com
```

```
34.55.139.0
```

```
%%writefile movieapp.py

import streamlit as st
import pandas as pd
import numpy as np

from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors

movies=pd.read_csv('/content/drive/MyDrive/ML dataset/movies.csv')
ratings=pd.read_csv('/content/drive/MyDrive/ML dataset/ratings.csv')

a = movies['title']
movie_titles_list = a.to_list()

dataset = ratings.pivot(index="movieId",columns="userId",values="rating")
dataset.fillna(0,inplace=True)

num_user_voted = ratings.groupby('movieId')['rating'].agg('count')
num_movies_voted = ratings.groupby('userId')['rating'].agg('count')

final_dataset = dataset.loc[num_user_voted[num_user_voted > 10].index, :]
final_dataset = final_dataset.loc[: , num_movies_voted[num_movies_voted>50].index]

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

knn = NearestNeighbors(metric='cosine', algorithm = 'brute', n_neighbors = 30, n_jobs = -1)
knn.fit(csr_data)

def recommendation(movie_name,no_of_movies):
    movie_list = movies[movies['title'].str.contains(movie_name)]
    print(type(movie_list))
    if len(movie_list):
        movie_index = movie_list.iloc[0]['movieId']
        movie_index = final_dataset[final_dataset['movieId'] == movie_index].index[0]
```

```
        distances, indices = knn.kneighbors(csr_data[movie_index], n_neighbors=no_of_movies+1)

        indices_list = indices.squeeze().tolist()
        distances_list = distances.squeeze().tolist()
        index_distance_pairs = list(zip(indices_list, distances_list))
        rec_movies_indices = sorted(index_distance_pairs[1:], key=lambda x: x[1], reverse=True)

        recommended_movies = []
        for val in rec_movies_indices:
            movie_index = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_index].index
            recommended_movies.append({'Title': movies.iloc[idx]['title'].values[0], 'Distance': val[1]})
        df = pd.DataFrame(recommended_movies, index=range(1, no_of_movies+1))
        return df
    else:
        return "Movie not found..."


st.title("Movie Recommendation System")

#selected_movie=st.selectbox("Select a movie", movie_titles_list)
selected_movie=st.text_input("Enter a Movie Name: ")
no_of_movies=st.text_input("No of Movies to Recommend: ")
if(st.button("Recommend: ")):
  st.text("Recommended Movies are: ")

  recommended_movies=recommendation(selected_movie,int(no_of_movies))
  st.table(recommended_movies)
```

```
Writing movieapp.py
```

```
!streamlit run movieapp.py & npx localtunnel --port 9999
```

```
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://34.55.139.0:8501

your url is: https://evil-birds-sip.loca.lt
  Stopping...
^C
```