Recommenders

In [1]:
```
import difflib
import random
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from surprise import Reader, Dataset
from surprise.prediction_algorithms import SVD, KNNWithZScore
from surprise.model_selection import train_test_split
from collections import defaultdict
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Importing and Splitting data

In [2]:
```
full = pd.read_csv('../Data/filtered-cleaned')
full = full.drop(columns = 'Unnamed: 0')
full['title'] = full.title.str.extract('([a-zA-Z\s]+)')
min_cols = full[[ 'userId', 'movieId', 'rating']]
min_cols = min_cols.sample(1000000)
smaller = min_cols.sample(50000)
```

In [3]:
```
reader = Reader()
data = Dataset.load_from_df(min_cols, reader)
datasmall = Dataset.load_from_df(smaller, reader)
kdata = datasmall.build_full_trainset()
trainset, testset = train_test_split(data, test_size = 0.10)
```

In [4]:
```
trainset = data.build_full_trainset()
print('Unique users: ', trainset.n_users, '\n')
print('Unique Movies: ', trainset.n_items)
```

```
Unique users:  99078

Unique Movies:  6044
```

In [5]:
```
# Recommendations Based on Movie Title input
```

In [6]:
```
knn_zscore = KNNWithZScore(sim_options={'name':'pearson', 'user_based':Fals
knn_zscore.fit(trainset)
```

```
Computing the pearson similarity matrix...
Done computing similarity matrix.
```

Out[6]:
```
<surprise.prediction_algorithms.knns.KNNWithZScore at 0x7fe27d3221f0>
```

In [7]:
```
def out_neighbors(m_id):
    tsr_inner_id = knn_zscore.trainset.to_inner_iid(m_id)
    tsr_neighbors = knn_zscore.get_neighbors(tsr_inner_id, k=5)
    neighbors = full[full.movieId.isin([knn_zscore.trainset.to_raw_iid(inne
                        for inner_id in tsr_neighbors])]
    print(list(neighbors.title.unique()))

def neighborer(df):
    movie_title = input("Enter Movie Title... Spelled correctly" )
    searchable = df.copy()
    searchable['search'] = searchable['title']
    searchable['search'] = searchable['search'].astype('string')
    searchable['search'] = searchable['search'].str.lower()
    searchable['search'] = searchable['search'].str.replace(' ', '')
    movie_title = movie_title.lower()
    movie_title = movie_title.replace(' ', '')
    for i, title in searchable.search.items():
        if movie_title == title:
            m_id = searchable.iloc[i]['movieId']
            name = full.loc[full['movieId'] == m_id, 'title'].iloc[0]
            out_neighbors(m_id)
            return m_id, name
```

In [8]:
```
print('Sample Movies: ' , list(full.title.sample(5)))
```

```
Sample Movies:  ['Finding Neverland ', 'Brother Bear ', 'Motorcycle Diari
es', 'Love', 'Dumb and Dumber To ']
```

Enter Title Here ----------

In [9]:
```
neighborer(full)
```

```
Enter Movie Title... Spelled correctlyBrokeback Mountain
['Won', 'Joy Ride ', 'Icarus ', 'Who Am I ', 'Red State ']
```

Out[9]:
```
(39183, 'Brokeback Mountain ')
```

Recommendations From Rating Input

In [10]:
```
def movie_rater(movie_df,num, genre=None):
    userID = 1000
    rating_list = []
    while num > 0:
        if genre:
            movie = movie_df[movie_df['genres'].str.contains(genre)].sample
        else:
            movie = movie_df.sample(1)
        print(movie.title)
        rating = input('How do you rate this movie on a scale of 1-5, press
        if rating == 'n':
            continue
        else:
            rating_one_movie = {'userId':userID,'movieId':movie['movieId'].
            rating_list.append(rating_one_movie)
            num -= 1
    return rating_list
```

Rate Movies Here ----------

Run Next 2 Cells

In [ ]:
```
user_rating = movie_rater(full, 5)
```

```
1104586     A Space Odyssey
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
n
1857414    Snatch
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
n
243096    Thing
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
4
823367    Day After Tomorrow
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
4
13674    Burn After Reading
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
n
1899284    Whip It
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
n
1861565    Birthday Girl
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
n
204015    Avengers
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
5
364397    Kingsman
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
4
1909652    Oz the Great and Powerful
Name: title, dtype: object
How do you rate this movie on a scale of 1-5, press n if you have not see
n :
n
957452    X
Name: title, dtype: object
```

In [13]:
```
print("Please Wait... looking for awesome movies \n \n")

new_ratings_df = min_cols.append(user_rating,ignore_index=True)
new_data = Dataset.load_from_df(new_ratings_df,reader)

svd_ = SVD(n_factors= 50, reg_all=0.05)
svd_.fit(new_data.build_full_trainset())


list_of_movies = []
for m_id in min_cols['movieId'].unique():
    list_of_movies.append( (m_id,svd_.predict(1000,m_id)[3]))



ranked_movies = sorted(list_of_movies, key=lambda x:x[1], reverse=True)



def recommended_movies(user_ratings,movie_title_df,n):
        for idx, rec in enumerate(user_ratings):
            title = movie_title_df.loc[movie_title_df['movieId'] == int(rec
            print('Recommendation # ', idx+1, ': ', title, '\n')
            n-= 1
            if n == 0:
                break

print('Movie Recs: \n')
recommended_movies(ranked_movies,full,5)
```

```
Please Wait... looking for awesome movies


Movie Recs:

Recommendation #  1 :  HyperNormalisation

Recommendation #  2 :  Planet Earth

Recommendation #  3 :  Planet Earth II

Recommendation #  4 :  O

Recommendation #  5 :  Blue Planet II
```

In [ ]: