Item Based Collaboritve Filtering (For Movie Reccomendations)

```
import pandas as pd
import numpy as np
```

```
df = pd.DataFrame({'user_0':[0,3,0,5,0,0,4,5,0,2], 'user_1':[0,0,3,2,5,0,4,0,3,0], 'user_2':[3,1,0,3,5,0,0,4,0,0], 'use
                   'user_4':[2,0,0,0,0,4,4,3,5,0], 'user_5':[1,0,2,4,0,0,4,0,5,0], 'user_6':[2,0,0,3,0,4,3,3,0,0], 'use
                   'user_8':[5,0,0,0,5,3,0,3,0,4], 'user_9':[1,0,2,0,4,0,4,3,0,0]}, index=['movie_0','movie_1','movie_2
df
```

	user_0	user_1	user_2	user_3	user_4	user_5	user_6	user_7	user_8	user_9
movie_0	0	0	3	4	2	1	2	0	5	1
movie_1	3	0	1	3	0	0	0	0	0	0
movie_2	0	3	0	4	0	2	0	0	0	2
movie_3	5	2	3	2	0	4	3	3	0	0
movie_4	0	5	5	0	0	0	0	0	5	4
movie_5	0	0	0	0	4	0	4	2	3	0
movie_6	4	4	0	0	4	4	3	4	0	4
movie_7	5	0	4	2	3	0	3	3	3	3
movie_8	0	3	0	0	5	5	0	4	0	0
movie_9	2	0	0	0	0	0	0	0	4	0

```
df.values
```

```
array([[0, 0, 3, 4, 2, 1, 2, 0, 5, 1],
       [3, 0, 1, 3, 0, 0, 0, 0, 0, 0],
       [0, 3, 0, 4, 0, 2, 0, 0, 0, 2],
       [5, 2, 3, 2, 0, 4, 3, 3, 0, 0],
       [0, 5, 5, 0, 0, 0, 0, 0, 5, 4],
       [0, 0, 0, 0, 4, 0, 4, 2, 3, 0],
       [4, 4, 0, 0, 4, 4, 3, 4, 0, 4],
       [5, 0, 4, 2, 3, 0, 3, 3, 3, 3],
       [0, 3, 0, 0, 5, 5, 0, 4, 0, 0],
       [2, 0, 0, 0, 0, 0, 0, 0, 4, 0]])
```

```
from sklearn.neighbors import NearestNeighbors

knn = NearestNeighbors(metric='cosine', algorithm='brute')
knn.fit(df.values)
distances, indices = knn.kneighbors(df.values, n_neighbors=3)
```

```
indices
```

```
array([[0, 7, 5],
       [1, 3, 7],
       [2, 1, 6],
       [3, 6, 7],
       [4, 0, 7],
       [5, 7, 0],
       [6, 8, 3],
       [7, 3, 0],
       [8, 6, 3],
       [9, 0, 7]])
```

```
distances
```

```
array([[0.00000000e+00, 3.19586183e-01, 4.03404722e-01],
       [4.44089210e-16, 3.68421053e-01, 3.95436458e-01],
       [0.00000000e+00, 5.20766162e-01, 5.24329288e-01],
```

```
       [2.22044605e-16, 2.72367798e-01, 2.86615021e-01],
       [0.00000000e+00, 4.04534842e-01, 4.80655057e-01],
       [0.00000000e+00, 3.87174123e-01, 4.03404722e-01],
       [0.00000000e+00, 2.33726809e-01, 2.72367798e-01],
       [1.11022302e-16, 2.86615021e-01, 3.19586183e-01],
       [2.22044605e-16, 2.33726809e-01, 4.96677704e-01],
       [1.11022302e-16, 4.22649731e-01, 4.81455027e-01]])
```

```
for title in df.index:

  index_user_likes = df.index.tolist().index(title) # get an index for a movie
  sim_movies = indices[index_user_likes].tolist() # make list for similar movies
  movie_distances = distances[index_user_likes].tolist() # the list for distances of similar movies
  id_movie = sim_movies.index(index_user_likes) # get the position of the movie itself in indices and distances

  print('Similar Movies to '+str(df.index[index_user_likes])+':\n')


  sim_movies.remove(index_user_likes) # remove the movie itself in indices
  movie_distances.pop(id_movie) # remove the movie itself in distances

  j = 1

  for i in sim_movies:
    print(str(j)+': '+str(df.index[i])+', the distance with '+str(title)+': '+str(movie_distances[j-1]))
    j = j + 1

  print('\n')
```

```
Similar Movies to movie_0:

1: movie_7, the distance with movie_0: 0.3195861825602283
2: movie_5, the distance with movie_0: 0.40340472183738674


Similar Movies to movie_1:

1: movie_3, the distance with movie_1: 0.3684210526315791
2: movie_7, the distance with movie_1: 0.39543645824165696


Similar Movies to movie_2:

1: movie_1, the distance with movie_2: 0.5207661617014769
2: movie_6, the distance with movie_2: 0.5243292879915494


Similar Movies to movie_3:

1: movie_6, the distance with movie_3: 0.27236779788557686
2: movie_7, the distance with movie_3: 0.2866150207251553


Similar Movies to movie_4:

1: movie_0, the distance with movie_4: 0.40453484184315647
2: movie_7, the distance with movie_4: 0.4806550570967598


Similar Movies to movie_5:

1: movie_7, the distance with movie_5: 0.38717412297165876
2: movie_0, the distance with movie_5: 0.40340472183738674


Similar Movies to movie_6:

1: movie_8, the distance with movie_6: 0.23372680904614496
2: movie_3, the distance with movie_6: 0.27236779788557686


Similar Movies to movie_7:

1: movie_3, the distance with movie_7: 0.2866150207251553
2: movie_0, the distance with movie_7: 0.3195861825602283
```

```
Similar Movies to movie_8:

1: movie_6, the distance with movie_8: 0.23372680904614496
2: movie_3, the distance with movie_8: 0.49667770431528346


Similar Movies to movie_9:

1: movie_0, the distance with movie_9: 0.42264973081037427
2: movie_7, the distance with movie_9: 0.4814550271298651
```

Create a Movie Recommender using Movie Cosine Similarity/KNN

```
def recommend_movie(title):

  index_user_likes = df.index.tolist().index(title) # get an index for a movie
  sim_movies = indices[index_user_likes].tolist() # make list for similar movies
  movie_distances = distances[index_user_likes].tolist() # the list for distances of similar movies
  id_movie = sim_movies.index(index_user_likes) # get the position of the movie itself in indices and distances

  print('Similar Movies to '+str(df.index[index_user_likes])+': \n')

  sim_movies.remove(index_user_likes) # remove the movie itself in indices
  movie_distances.pop(id_movie) # remove the movie itself in distances

  j = 1

  for i in sim_movies:
    print(str(j)+': '+str(df.index[i])+', the distance with '+str(title)+': '+str(movie_distances[j-1]))
    j = j + 1
```

```
recommend_movie('movie_3')
```

```
Similar Movies to movie_3:

1: movie_6, the distance with movie_3: 0.27236779788557686
2: movie_7, the distance with movie_3: 0.2866150207251553
```

Recommend Similar Movies to a User

1. Calculate similar movies using KNN
2. Predict the user's rating for unwatched movies.
3. Recommend movies with highest predicted user rating

1. Calculate Similar Movies

```
knn = NearestNeighbors(metric='cosine', algorithm='brute')
knn.fit(df.values)
distances, indices = knn.kneighbors(df.values, n_neighbors=3)
```

```
index_for_movie = df.index.tolist().index('movie_0') # it returns 0
sim_movies = indices[index_for_movie].tolist() # make list for similar movies
movie_distances = distances[index_for_movie].tolist() # the list for distances of similar movies
id_movie = sim_movies.index(index_for_movie) # get the position of the movie itself in indices and distances
sim_movies.remove(index_for_movie) # remove the movie itself in indices
movie_distances.pop(id_movie) # remove the movie itself in distances

print('The Nearest Movies to movie_0:', sim_movies)
print('The Distance from movie_0:', movie_distances)
```

```
The Nearest Movies to movie_0: [7, 5]
The Distance from movie_0: [0.3195861825602283, 0.40340472183738674]
```

2. Predict the user's rating for unwatched movies.

```
movie_similarity = [-x+1 for x in movie_distances] # inverse distance

predicted_rating = (movie_similarity[0]*df.iloc[sim_movies[0],7] + movie_similarity[1]*df.iloc[sim_movies[1],7])/sum(mo
print(predicted_rating)
```

```
2.5328183015946415
```

3. Recommend movies with highest predicted user rating

```
# find the nearest neighbors using NearestNeighbors(n_neighbors=3)
number_neighbors = 3
knn = NearestNeighbors(metric='cosine', algorithm='brute')
knn.fit(df.values)
distances, indices = knn.kneighbors(df.values, n_neighbors=number_neighbors)

# copy df
df1 = df.copy()

# convert user_name to user_index
user_index = df.columns.tolist().index('user_4')

# t: movie_title, m: the row number of t in df
for m,t in list(enumerate(df.index)):

  # find movies without ratings by user_4
  if df.iloc[m, user_index] == 0:
    sim_movies = indices[m].tolist()
    movie_distances = distances[m].tolist()

    # Generally, this is the case: indices[3] = [3 6 7]. The movie itself is in the first place.
    # In this case, we take off 3 from the list. Then, indices[3] == [6 7] to have the nearest NEIGHBORS in the list.
    if m in sim_movies:
      id_movie = sim_movies.index(m)
      sim_movies.remove(m)
      movie_distances.pop(id_movie)

    # However, if the percentage of ratings in the dataset is very low, there are too many 0s in the dataset.
    # Some movies have all 0 ratings and the movies with all 0s are considered the same movies by NearestNeighbors().
    # Then,even the movie itself cannot be included in the indices.
    # For example, indices[3] = [2 4 7] is possible if movie_2, movie_3, movie_4, and movie_7 have all 0s for their rat
    # In that case, we take off the farthest movie in the list. Therefore, 7 is taken off from the list, then indices[3
    else:
      sim_movies = sim_movies[:number_neighbors-1]
      movie_distances = movie_distances[:number_neighbors-1]

    # movie_similarty = 1 - movie_distance
    movie_similarity = [1-x for x in movie_distances]
    movie_similarity_copy = movie_similarity.copy()
    nominator = 0

    # for each similar movie
    for s in range(0, len(movie_similarity)):

      # check if the rating of a similar movie is zero
      if df.iloc[sim_movies[s], user_index] == 0:

        # if the rating is zero, ignore the rating and the similarity in calculating the predicted rating
        if len(movie_similarity_copy) == (number_neighbors - 1):
          movie_similarity_copy.pop(s)

        else:
          movie_similarity_copy.pop(s-(len(movie_similarity)-len(movie_similarity_copy)))
```

```
        # if the rating is not zero, use the rating and similarity in the calculation
        else:
          nominator = nominator + movie_similarity[s]*df.iloc[sim_movies[s],user_index]

      # check if the number of the ratings with non-zero is positive
      if len(movie_similarity_copy) > 0:

        # check if the sum of the ratings of the similar movies is positive.
        if sum(movie_similarity_copy) > 0:
          predicted_r = nominator/sum(movie_similarity_copy)

        # Even if there are some movies for which the ratings are positive, some movies have zero similarity even though
        # in this case, the predicted rating becomes zero as well
        else:
          predicted_r = 0

      # if all the ratings of the similar movies are zero, then predicted rating should be zero
      else:
        predicted_r = 0

    # place the predicted rating into the copy of the original dataset
      df1.iloc[m,user_index] = predicted_r


def recommend_movies(user, num_recommended_movies):

  print('The list of the Movies {} Has Watched \n'.format(user))

  for m in df[df[user] > 0][user].index.tolist():
    print(m)

  print('\n')

  recommended_movies = []

  for m in df[df[user] == 0].index.tolist():

    index_df = df.index.tolist().index(m)
    predicted_rating = df1.iloc[index_df, df1.columns.tolist().index(user)]
    recommended_movies.append((m, predicted_rating))

  sorted_rm = sorted(recommended_movies, key=lambda x:x[1], reverse=True)

  print('The list of the Recommended Movies \n')
  rank = 1
  for recommended_movie in sorted_rm[:num_recommended_movies]:

    print('{}: {} - predicted rating:{}'.format(rank, recommended_movie[0], recommended_movie[1]))
    rank = rank + 1



recommend_movies('user_4',5)
```

```
The list of the Movies user_4 Has Watched

movie_0
movie_5
movie_6
movie_7
movie_8


The list of the Recommended Movies

1: movie_2 - predicted rating:4.0
2: movie_3 - predicted rating:3.504943460433221
3: movie_1 - predicted rating:3.0
4: movie_9 - predicted rating:2.473170201830165
5: movie_4 - predicted rating:2.4658595597666277
```

```
df1 = df.copy()

def movie_recommender(user, num_neighbors, num_recommendation):

  number_neighbors = num_neighbors

  knn = NearestNeighbors(metric='cosine', algorithm='brute')
  knn.fit(df.values)
  distances, indices = knn.kneighbors(df.values, n_neighbors=number_neighbors)

  user_index = df.columns.tolist().index(user)

  for m,t in list(enumerate(df.index)):
    if df.iloc[m, user_index] == 0:
      sim_movies = indices[m].tolist()
      movie_distances = distances[m].tolist()

      if m in sim_movies:
        id_movie = sim_movies.index(m)
        sim_movies.remove(m)
        movie_distances.pop(id_movie)

      else:
        sim_movies = sim_movies[:num_neighbors-1]
        movie_distances = movie_distances[:num_neighbors-1]

      movie_similarity = [1-x for x in movie_distances]
      movie_similarity_copy = movie_similarity.copy()
      nominator = 0

      for s in range(0, len(movie_similarity)):
        if df.iloc[sim_movies[s], user_index] == 0:
          if len(movie_similarity_copy) == (number_neighbors - 1):
            movie_similarity_copy.pop(s)

          else:
            movie_similarity_copy.pop(s-(len(movie_similarity)-len(movie_similarity_copy)))

        else:
          nominator = nominator + movie_similarity[s]*df.iloc[sim_movies[s],user_index]

      if len(movie_similarity_copy) > 0:
        if sum(movie_similarity_copy) > 0:
          predicted_r = nominator/sum(movie_similarity_copy)

        else:
          predicted_r = 0

      else:
        predicted_r = 0

      df1.iloc[m,user_index] = predicted_r
  recommend_movies(user,num_recommendation)
```

```
ratings = pd.read_csv('ratings.csv', usecols=['userId','movieId','rating'])
movies = pd.read_csv('movies.csv', usecols=['movieId','title'])
ratings2 = pd.merge(ratings, movies, how='inner', on='movieId')
```

```
df = ratings2.pivot_table(index='title',columns='userId',values='rating').fillna(0)
df1 = df.copy()
```

```
def recommend_movies(user, num_recommended_movies):

  print('The list of the Movies {} Has Watched \n'.format(user))
```

```
    for m in df[df[user] > 0][user].index.tolist():
      print(m)

    print('\n')

    recommended_movies = []

    for m in df[df[user] == 0].index.tolist():

      index_df = df.index.tolist().index(m)
      predicted_rating = df1.iloc[index_df, df1.columns.tolist().index(user)]
      recommended_movies.append((m, predicted_rating))

    sorted_rm = sorted(recommended_movies, key=lambda x:x[1], reverse=True)

    print('The list of the Recommended Movies \n')
    rank = 1
    for recommended_movie in sorted_rm[:num_recommended_movies]:

      print('{}: {} - predicted rating:{}'.format(rank, recommended_movie[0], recommended_movie[1]))
      rank = rank + 1



  def movie_recommender(user, num_neighbors, num_recommendation):

    number_neighbors = num_neighbors

    knn = NearestNeighbors(metric='cosine', algorithm='brute')
    knn.fit(df.values)
    distances, indices = knn.kneighbors(df.values, n_neighbors=number_neighbors)

    user_index = df.columns.tolist().index(user)

    for m,t in list(enumerate(df.index)):
      if df.iloc[m, user_index] == 0:
        sim_movies = indices[m].tolist()
        movie_distances = distances[m].tolist()

        if m in sim_movies:
          id_movie = sim_movies.index(m)
          sim_movies.remove(m)
          movie_distances.pop(id_movie)

        else:
          sim_movies = sim_movies[:num_neighbors-1]
          movie_distances = movie_distances[:num_neighbors-1]

        movie_similarity = [1-x for x in movie_distances]
        movie_similarity_copy = movie_similarity.copy()
        nominator = 0

        for s in range(0, len(movie_similarity)):
          if df.iloc[sim_movies[s], user_index] == 0:
            if len(movie_similarity_copy) == (number_neighbors - 1):
              movie_similarity_copy.pop(s)

            else:
              movie_similarity_copy.pop(s-(len(movie_similarity)-len(movie_similarity_copy)))

          else:
            nominator = nominator + movie_similarity[s]*df.iloc[sim_movies[s],user_index]

        if len(movie_similarity_copy) > 0:
          if sum(movie_similarity_copy) > 0:
            predicted_r = nominator/sum(movie_similarity_copy)

          else:
            predicted_r = 0
```

```
      else:
        predicted_r = 0

      df1.iloc[m,user_index] = predicted_r
  recommend_movies(user,num_recommendation)
```

```
movie_recommender(15, 10, 10)
```

```
Passengers (2016)
Patriot, The (2000)
Pinocchio (1940)
Prestige, The (2006)
Prometheus (2012)
Pulp Fiction (1994)
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
Ratatouille (2007)
Requiem for a Dream (2000)
Road Trip (2000)
Rogue One: A Star Wars Story (2016)
Ronin (1998)
Sausage Party (2016)
Saving Private Ryan (1998)
Schindler's List (1993)
Seven (a.k.a. Se7en) (1995)
Shawshank Redemption, The (1994)
Shrek (2001)
Shrek 2 (2004)
Sixth Sense, The (1999)
Source Code (2011)
Spirited Away (Sen to Chihiro no kamikakushi) (2001)
Star Wars: Episode III - Revenge of the Sith (2005)
Star Wars: Episode IV - A New Hope (1977)
Star Wars: Episode V - The Empire Strikes Back (1980)
Star Wars: Episode VI - Return of the Jedi (1983)
Star Wars: Episode VII - The Force Awakens (2015)
Sully (2016)
Terminator 2: Judgment Day (1991)
Terminator, The (1984)
The Butterfly Effect (2004)
The Hunger Games (2012)
The Martian (2015)
Total Recall (1990)
Toy Story (1995)
U-571 (2000)
Unbreakable (2000)
Up (2009)
WALL·E (2008)
What Women Want (2000)
World War Z (2013)
X-Files: Fight the Future, The (1998)
X-Men: Apocalypse (2016)
Zootopia (2016)


The list of the Recommended Movies

1: Army of Darkness (1993) - predicted rating:5.000000000000001
2: Finding Forrester (2000) - predicted rating:5.000000000000001
3: Home Alone 2: Lost in New York (1992) - predicted rating:5.000000000000001
4: Jaws (1975) - predicted rating:5.000000000000001
5: Master and Commander: The Far Side of the World (2003) - predicted rating:5.000000000000001
6: Speed (1994) - predicted rating:5.000000000000001
7: Thank You for Smoking (2006) - predicted rating:5.000000000000001
8: 2001: A Space Odyssey (1968) - predicted rating:5.0
9: Alien: Resurrection (1997) - predicted rating:5.0
10: Beverly Hills Cop (1984) - predicted rating:5.0
```