



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No. 5
Implement Item-based Nearest neighbor recommendation
Date of Performance:
Date of Submission:
Marks:
Sign:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

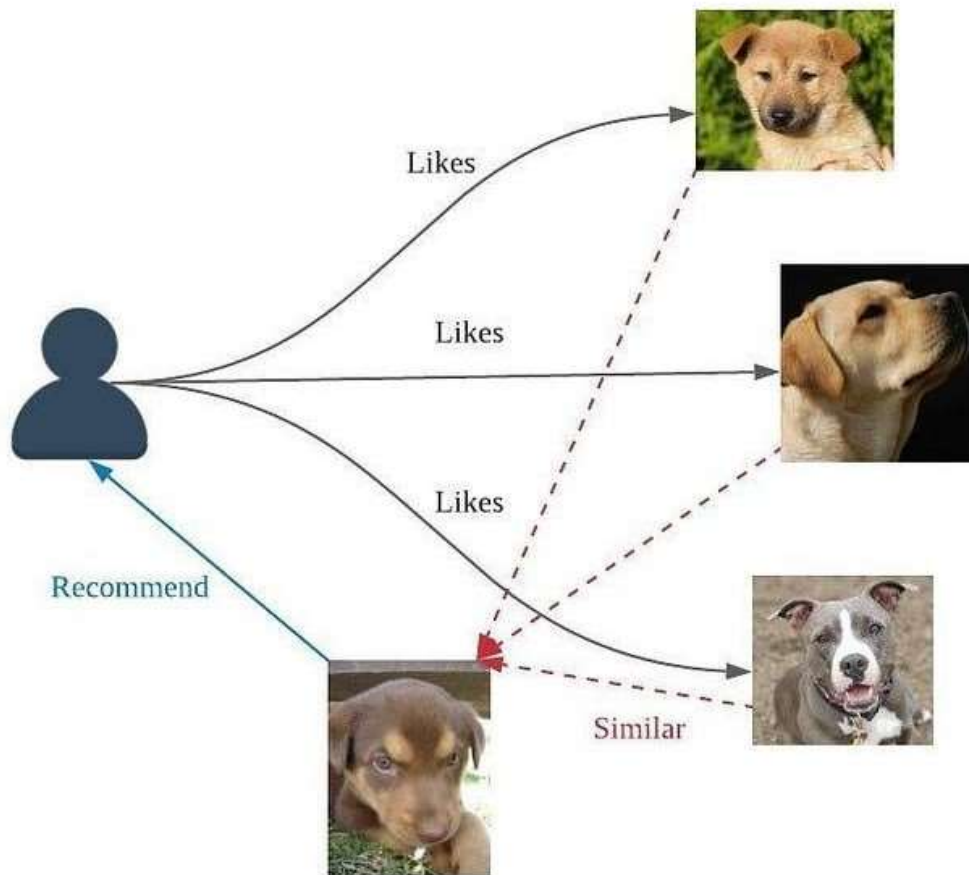
Aim: Implement Item-based Nearest neighbor recommendation.

Objective: Able to interpret and implement Item based Nearest neighbor recommendation.

Theory:

Item-based Collaborative Filtering

Item-based collaborative filtering uses the rating of co-rated item to predict the rating on specific item.



For example, we want to predict rating of user 2 on item 2.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

	user 1	user 2	user 3	user 4
item 1	3	4	nan	nan
item 2	2	?	1	nan
item 3	nan	nan	2	4
item 4	2	1	4	nan

To predict the rating,

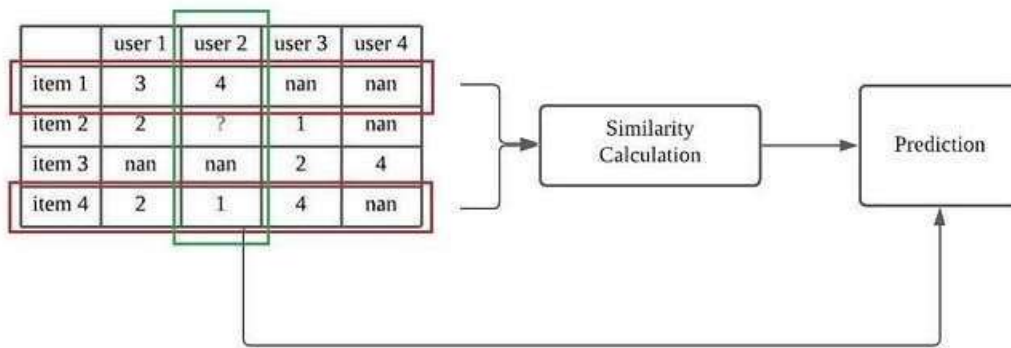
1. Find the co-rated items of user 2, which is item 1 and item 4

2. Calculate the similarity between item 2 and item 1, 4

3. Calculate the prediction based on the similarity and co-rated rating, such that

$$prediction_{u,i} = \frac{\sum_n \omega_{i,n} * r_{u,n}}{\sum_n |\omega_{i,n}|}$$

where w is the similarity, and r is the rating value.





Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Implementation:

```
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],
'user_3':[4,3,4,2,0,0,0,2,0,0],
            '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],
'user_7':[0,0,0,3,0,2,4,3,4,0],
            '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','movie_3',
'movie_4','movie_5','movie_6','movie_7','movie_8','movie_9'])
df
df.values
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
distances
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')

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')
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
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
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)
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(movie_similarity)
print(predicted_rating)
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)
CSDOL8022: Recommendation Systems Lab
```

```
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 ratings.
# In that case, we take off the farthest movie
in the list. Therefore, 7 is taken off from the
list, then indices[3] == [2 4].
else:
sim_movies =
sim_movies[:number_neighbors-1]
movie_distances =
movie_distances[:number_neighbors-1]

# movie_similarity = 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
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
else:
    nominator = nominator +
    movie_similarity[s]*df.iloc[sim_movies[s],use
    r_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 they are selected
as similar movies.
```

```
# 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)
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]
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
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],use
r_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_recommendatio
n)

movie_recommender('user_4', 4, 5)
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='use
rId',values='rating').fillna(0)
df1 = df.copy()
```

CSDOL8022: Recommendation Systems Lab

```
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:
```




Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
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],use
r_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_recommendatio
n)
movie_recommender(15, 10, 10)
```

Output:

The list of the Movies 15 Has Watched

(500) Days of Summer (2009)
10 Cloverfield Lane (2016)
101 Dalmatians (One Hundred and One Dalmatians) (1961)
28 Days Later (2002)
9 (2009)
A.I. Artificial Intelligence (2001)
Adjustment Bureau, The (2011)
Aladdin (1992)
Alien (1979)
Aliens (1986)
American Beauty (1999)
American History X (1998)
American Psycho (2000)
Apocalypto (2006)
Avatar (2009)
Avengers, The (2012)
Back to the Future (1985)
Back to the Future Part II (1989)

Back to the Future Part III (1990)
Beautiful Mind, A (2001)
Bicentennial Man (1999)
Bolt (2008)
Bridge of Spies (2015)
Captain America: The Winter Soldier (2014)
Captain Phillips (2013)
Casper (1995)
Cast Away (2000)
Catch Me If You Can (2002)
Chappie (2015)
Children of Men (2006)
Cloudy with a Chance of Meatballs (2009)
Dark Knight Rises, The (2012)
Dark Knight, The (2008)
Deadpool (2016)
District 9 (2009)
Django Unchained (2012)
Doctor Strange (2016)
Edge of Tomorrow (2014)
Escape from L.A. (1996)



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Ex Machina (2015)	My Neighbor Totoro (Tonari no Totoro) (1988)
Fifth Element, The (1997)	Nightmare on Elm Street, A (1984)
Fight Club (1999)	Oblivion (2013)
Finding Nemo (2003)	Others, The (2001)
Flintstones, The (1994)	Passengers (2016)
Forrest Gump (1994)	Patriot, The (2000)
Frequency (2000)	Pinocchio (1940)
Gattaca (1997)	Prestige, The (2006)
Gladiator (2000)	Prometheus (2012)
Godfather, The (1972)	Pulp Fiction (1994)
Gods Must Be Crazy, The (1980)	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
Gone Girl (2014)	Ratatouille (2007)
Gran Torino (2008)	Requiem for a Dream (2000)
Grand Budapest Hotel, The (2014)	Road Trip (2000)
Gravity (2013)	Rogue One: A Star Wars Story (2016)
Green Mile, The (1999)	Ronin (1998)
Groundhog Day (1993)	Sausage Party (2016)
Guardians of the Galaxy (2014)	Saving Private Ryan (1998)
I Am Legend (2007)	Schindler's List (1993)
I, Robot (2004)	Seven (a.k.a. Se7en) (1995)
Inception (2010)	Shawshank Redemption, The (1994)
Incredibles, The (2004)	Shrek (2001)
Independence Day (a.k.a. ID4) (1996)	Shrek 2 (2004)
Inside Out (2015)	Sixth Sense, The (1999)
Interstellar (2014)	Source Code (2011)
Iron Man (2008)	Spirited Away (Sen to Chihiro no kamikakushi) (2001)
John Wick (2014)	Star Wars: Episode III - Revenge of the Sith (2005)
Johnny Mnemonic (1995)	Star Wars: Episode IV - A New Hope (1977)
Junior (1994)	Star Wars: Episode V - The Empire Strikes Back (1980)
Kill Bill: Vol. 1 (2003)	Star Wars: Episode VI - Return of the Jedi (1983)
Kill Bill: Vol. 2 (2004)	Star Wars: Episode VII - The Force Awakens (2015)
Lethal Weapon 2 (1989)	Sully (2016)
Life of Pi (2012)	Terminator 2: Judgment Day (1991)
Limitless (2011)	Terminator, The (1984)
Lion King, The (1994)	The Butterfly Effect (2004)
Little Mermaid, The (1989)	The Hunger Games (2012)
Looper (2012)	The Martian (2015)
Lord of the Rings: The Fellowship of the Ring, The (2001)	Total Recall (1990)
Lord of the Rings: The Two Towers, The (2002)	Toy Story (1995)
Léon: The Professional (a.k.a. The Professional) (Léon) (1994)	U-571 (2000)
Mad Max: Fury Road (2015)	Unbreakable (2000)
Matrix, The (1999)	Up (2009)
Memento (2000)	WALL·E (2008)
Minority Report (2002)	
Misery (1990)	
Monsters, Inc. (2001)	
Moon (2009)	
Mortal Kombat (1995)	



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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: Exorcist, The (1973) - 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: Master and Commander: The Far Side of the World (2003) - predicted rating:5.000000000000001
5: Speed (1994) - predicted rating:5.000000000000001
6: Thank You for Smoking (2006) - predicted rating:5.000000000000001
7: 2001: A Space Odyssey (1968) - predicted rating:5.0
8: Army of Darkness (1993) - predicted rating:5.0
9: Beverly Hills Cop (1984) - predicted rating:5.0
10: Blood Diamond (2006) - predicted rating:5.0

Conclusion:

Item-based nearest neighbor recommendation systems analyze user-item interactions to suggest items similar to those a user has interacted with. By measuring item-item similarities, these systems identify patterns and recommend items that share characteristics with ones the user has liked. This approach is efficient for sparse datasets and provides personalized recommendations based on item similarities, enhancing user experience and engagement.