In [1]:
```
import pandas as pd
import numpy as np
from textblob import TextBlob
from sklearn.metrics.pairwise import cosine_similarity
```

Load .csv file

In [2]:
```
df = pd.read_csv("cloths-rating.csv")
df.head()
```

Out[2]:

	ProductID	UserID	Rating	Text
0	777	AV1YnR7wglJLPUi8IJmi	4	Great taffy at a great price.
1	767	AVpfpK8KLJeJML43BCuD	4	Absolutely wonderful - silky and sexy and comf...
2	1080	AVqkIdntQMlgsOJE6fuB	5	Love this dress! it's sooo pretty.
3	1077	AVpfpK8KLJeJML43BCuD	3	I had such high hopes for this dress and reall...
4	1049	AVpfpK8KLJeJML43BCuD	5	I love, love, love this jumpsuit. it's fun, fl...

Find Sentiment on Text(Reviews)

In [3]:
```
def sentiment_calc(text):
    try:
        return TextBlob(str(text)).sentiment.polarity
    except:
        return None
df['sentiment'] = df['Text'].apply(sentiment_calc)
df
```

Out[3]:

	ProductID	UserID	Rating	Text	sentiment
0	777	AV1YnR7wglJLPUi8IJmi	4	Great taffy at a great price.	0.800000
1	767	AVpfpK8KLJeJML43BCuD	4	Absolutely wonderful - silky and sexy and comf...	0.633333
2	1080	AVqkIdntQMlgsOJE6fuB	5	Love this dress! it's sooo pretty.	0.437500
3	1077	AVpfpK8KLJeJML43BCuD	3	I had such high hopes for this dress and reall...	0.120000
4	1049	AVpfpK8KLJeJML43BCuD	5	I love, love, love this jumpsuit. it's fun, fl...	0.550000
...	...	...	...	...	...
629	823	B08GWV3SM6	1	I placed order 4+1 soaps.But I have received w...	0.000000
630	823	B08GWV3SM6	3	The soap is ok for bathing, no scent at all, m...	0.325000

	ProductID	UserID	Rating	Text	sentiment
631	847	B08GWV3SM6	5	For a long time I was searching for Indian soa...	-0.025000
632	910	AVph0EeEilAPnD_x9myq	3	Good but not great	0.150000
633	333	AVqkIdntQMlgsOJE6fuB	5	Quick,easy to make & tasty too.	0.000000

634 rows × 5 columns

Apply Multiplication B/W Rating&Sentiment

In [4]:
```
df['Updated_score'] = df['Rating']*df['sentiment']
df
```

Out[4]:

	ProductID	UserID	Rating	Text	sentiment	Updated_score
0	777	AV1YnR7wglJLPUi8IJmi	4	Great taffy at a great price.	0.800000	3.200000
1	767	AVpfpK8KLJeJML43BCuD	4	Absolutely wonderful - silky and sexy and comf...	0.633333	2.533333
2	1080	AVqkIdntQMlgsOJE6fuB	5	Love this dress! it's sooo pretty.	0.437500	2.187500
3	1077	AVpfpK8KLJeJML43BCuD	3	I had such high hopes for this dress and reall...	0.120000	0.360000
4	1049	AVpfpK8KLJeJML43BCuD	5	I love, love, love this jumpsuit. it's fun, fl...	0.550000	2.750000
...	...	...	...	...	...	...
629	823	B08GWV3SM6	1	I placed order 4+1 soaps.But I have received w...	0.000000	0.000000
630	823	B08GWV3SM6	3	The soap is ok for bathing, no scent at all, m...	0.325000	0.975000
631	847	B08GWV3SM6	5	For a long time I was searching for Indian soa...	-0.025000	-0.125000
632	910	AVph0EeEilAPnD_x9myq	3	Good but not great	0.150000	0.450000
633	333	AVqkIdntQMlgsOJE6fuB	5	Quick,easy to make & tasty too.	0.000000	0.000000

634 rows × 6 columns

Make UserId into Normal Form

In [5]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['UserID'] = le.fit_transform(df['UserID'])
df
```

Out[5]:

	ProductID	UserID	Rating	Text	sentiment	Updated_score
0	777	0	4	Great taffy at a great price.	0.800000	3.200000
1	767	3	4	Absolutely wonderful - silky and sexy and comf...	0.633333	2.533333
2	1080	13	5	Love this dress! it's sooo pretty.	0.437500	2.187500
3	1077	3	3	I had such high hopes for this dress and reall...	0.120000	0.360000
4	1049	3	5	I love, love, love this jumpsuit. it's fun, fl...	0.550000	2.750000
...	...	...	...	...	...	...
629	823	41	1	I placed order 4+1 soaps.But I have received w...	0.000000	0.000000
630	823	41	3	The soap is ok for bathing, no scent at all, m...	0.325000	0.975000
631	847	41	5	For a long time I was searching for Indian soa...	-0.025000	-0.125000
632	910	7	3	Good but not great	0.150000	0.450000
633	333	13	5	Quick,easy to make & tasty too.	0.000000	0.000000

634 rows × 6 columns

Make function for classify updated_score

In [6]:
```
dict1 = {'1':[-5,-4,-3.75,-3,-2.5,-2.25,-2,-1.5,-1.25,-1,-0.75,-0.5,-0.25],'2':[

def fun(val):

    #Your Logic

    for i in dict1:
        if val >= dict1[i][0] and val <= dict1[i][-1]:
            return int(i)
```

Apply function on updated_score and put into New_score column

In [7]:
```
df['New_score'] = df['Updated_score'].apply(fun)
df['New_score'] = pd.to_numeric(df['New_score'])
df
```

Out[7]:

	ProductID	UserID	Rating	Text	sentiment	Updated_score	New_score
0	777	0	4	Great taffy at a great price.	0.800000	3.200000	5.0
1	767	3	4	Absolutely wonderful - silky and sexy and comf...	0.633333	2.533333	4.0
2	1080	13	5	Love this dress! it's sooo pretty.	0.437500	2.187500	4.0
3	1077	3	3	I had such high hopes for this dress and reall...	0.120000	0.360000	2.0
4	1049	3	5	I love, love, love this jumpsuit. it's fun, fl...	0.550000	2.750000	4.0
...	...	...	...	...	...	...	...
629	823	41	1	I placed order 4+1 soaps.But I have received w...	0.000000	0.000000	2.0
630	823	41	3	The soap is ok for bathing, no scent at all, m...	0.325000	0.975000	2.0
631	847	41	5	For a long time I was searching for Indian soa...	-0.025000	-0.125000	2.0
632	910	7	3	Good but not great	0.150000	0.450000	2.0
633	333	13	5	Quick,easy to make & tasty too.	0.000000	0.000000	2.0

634 rows × 7 columns

Pivot table of ProductID, UserID and New_score

In [8]:
```
df_pivot = df.pivot_table(index='ProductID',columns='UserID',values='New_score')
df_pivot
```

Out[8]:

UserID	0	1	2	3	4	5	6	7	8	9	...	32	3
ProductID													
89	0.0	0.000000	0.000000	3.0	0.0	0.0	5.0	2.000000	2.000000	0.0	...	0.000000	3.(
333	0.0	3.000000	0.000000	0.0	0.0	4.0	0.0	0.000000	4.333333	1.5	...	0.000000	1.(
369	5.0	2.000000	0.000000	3.0	0.0	4.0	5.0	3.000000	0.000000	0.0	...	0.000000	0.(
444	2.0	2.000000	0.000000	0.0	2.0	0.0	4.0	5.000000	2.000000	0.0	...	3.000000	0.(

UserID	0	1	2	3	4	5	6	7	8	9	...	32	3
ProductID													
684	0.0	0.000000	4.000000	4.0	2.0	0.0	2.0	3.000000	2.000000	0.0	...	3.500000	0.(
697	0.0	2.000000	2.000000	0.0	0.0	0.0	5.0	0.000000	2.000000	2.0	...	4.000000	3.(
767	2.0	0.000000	0.000000	4.0	0.0	0.0	2.0	0.000000	0.000000	0.0	...	0.000000	2.(
777	5.0	0.000000	0.000000	0.0	0.0	0.0	2.0	0.000000	0.000000	2.0	...	0.000000	1.(
823	0.0	0.000000	0.000000	5.0	0.0	0.0	5.0	1.000000	0.000000	0.0	...	0.000000	5.(
847	0.0	0.000000	0.000000	3.0	0.0	0.0	3.0	3.000000	5.000000	0.0	...	0.000000	4.(
853	0.0	0.000000	0.000000	0.0	0.0	0.0	2.0	0.000000	0.000000	0.0	...	0.000000	0.(
858	0.0	0.000000	0.000000	0.0	3.0	0.0	3.5	1.500000	3.000000	5.0	...	0.000000	0.(
862	3.0	3.000000	0.000000	4.0	3.5	0.0	3.0	1.500000	3.000000	0.0	...	2.000000	3.5
910	0.0	3.666667	0.000000	0.0	0.0	0.0	4.0	3.666667	4.000000	0.0	...	0.000000	4.(
949	3.0	3.000000	2.500000	0.0	0.0	0.0	2.0	0.000000	4.000000	0.0	...	2.666667	0.(
1002	5.0	5.000000	2.000000	0.0	0.0	0.0	2.0	0.000000	0.000000	0.0	...	2.000000	0.(
1003	0.0	0.000000	0.000000	0.0	0.0	0.0	2.0	0.000000	3.000000	0.0	...	3.000000	0.(
1049	3.0	4.000000	0.000000	4.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.000000	0.(
1060	2.5	3.000000	2.333333	0.0	2.0	0.0	0.0	3.000000	3.000000	0.0	...	2.666667	0.(
1065	0.0	0.000000	0.000000	1.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.000000	0.(
1077	0.0	3.000000	0.000000	2.0	2.0	0.0	2.0	1.750000	2.000000	2.0	...	0.000000	0.(
1080	0.0	3.000000	0.000000	0.0	2.0	0.0	3.0	2.000000	0.000000	2.0	...	0.000000	5.(
1095	0.0	0.000000	0.000000	2.0	4.0	0.0	0.0	2.333333	0.000000	2.0	...	0.000000	0.(
1120	0.0	0.000000	2.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	...	0.000000	0.(
6969	0.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	4.0	...	0.000000	4.(
8001	0.0	2.000000	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	4.5	...	0.000000	0.(
9696	2.0	3.000000	0.000000	3.0	0.0	0.0	2.0	3.500000	2.000000	2.0	...	0.000000	3.2

27 rows × 42 columns

Sparse Matrix (Compressed Sparse Row)

In [9]:

```
from scipy.sparse import csr_matrix

df_pivot_matrix = csr_matrix(df_pivot.values)
print(df_pivot_matrix)
```

```
  (0, 3)        3.0
  (0, 6)        5.0
  (0, 7)        2.0
  (0, 8)        2.0
  (0, 11)       4.0
  (0, 16)       3.5
```

```
(0, 18)       3.4
(0, 20)       2.0
(0, 22)       5.0
(0, 28)       4.0
(0, 33)       3.0
(0, 37)       4.0
(0, 41)       3.75
(1, 1)        3.0
(1, 5)        4.0
(1, 8)        4.333333333333333
(1, 9)        1.5
(1, 10)       2.0
(1, 13)       2.6666666666666665
(1, 17)       2.0
(1, 22)       5.0
(1, 25)       2.0
(1, 29)       3.0
(1, 33)       1.0
(1, 34)       3.0
:     :
(25, 17)      4.5
(25, 21)      2.0
(25, 22)      2.0
(25, 25)      2.6666666666666665
(25, 30)      4.0
(25, 38)      3.0
(25, 40)      1.6666666666666667
(26, 0)       2.0
(26, 1)       3.0
(26, 3)       3.0
(26, 6)       2.0
(26, 7)       3.5
(26, 8)       2.0
(26, 9)       2.0
(26, 11)      3.0
(26, 16)      3.0
(26, 17)      2.2
(26, 18)      3.0
(26, 19)      1.0
(26, 22)      2.0
(26, 23)      2.0
(26, 25)      4.0
(26, 28)      3.0
(26, 33)      3.25
(26, 37)      2.25
```

Fitting data into NearestNeighborsModel

In [40]:
```
from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric = 'cosine', n_neighbors=20, radius=1)
model_knn.fit(df_pivot_matrix)
```

Out[40]:
```
NearestNeighbors(radius=2)
```

Matrix of cosine similarity

In [41]:
```
similarity_matrix = cosine_similarity(df_pivot)
similarity_matrix
```

Out[41]:
```
array([[1.        , 0.33180003, 0.34731885, 0.39380839, 0.54676023,
        0.41816981, 0.29605353, 0.31227489, 0.82835423, 0.35071361,
        0.17221508, 0.3041334 , 0.54150012, 0.44539966, 0.26513706,
        0.08735731, 0.27681686, 0.10227159, 0.19656347, 0.09882579,
        0.3852686 , 0.36294349, 0.18041067, 0.        , 0.52433909,
        0.17769639, 0.71075896],
       [0.33180003, 1.        , 0.4317589 , 0.29088282, 0.19310629,
        0.44027757, 0.17970091, 0.03840277, 0.19777939, 0.26150968,
        0.12907336, 0.21441901, 0.39948681, 0.46466525, 0.39699107,
        0.25267484, 0.16365961, 0.12325556, 0.37963804, 0.        ,
        0.25313087, 0.20203635, 0.18434039, 0.        , 0.39145634,
        0.37185258, 0.42543225],
       [0.34731885, 0.4317589 , 1.        , 0.41152506, 0.20979493,
        0.35668523, 0.53740406, 0.41814325, 0.30614128, 0.37308693,
        0.16864772, 0.21273128, 0.33222111, 0.2708804 , 0.26285977,
        0.37745334, 0.10029936, 0.37370127, 0.31576952, 0.02108673,
        0.34450389, 0.22848954, 0.45471893, 0.        , 0.08465232,
        0.3660576 , 0.39251867],
       [0.39380839, 0.29088282, 0.41152506, 1.        , 0.64136155,
        0.56572701, 0.34303773, 0.36231101, 0.33034422, 0.35094711,
        0.42278148, 0.38094506, 0.62260256, 0.57529477, 0.60711865,
        0.43588408, 0.28597736, 0.12042511, 0.48405061, 0.        ,
        0.54299787, 0.47611277, 0.32947492, 0.        , 0.1488889 ,
        0.26717479, 0.48570938],
       [0.54676023, 0.19310629, 0.20979493, 0.64136155, 1.        ,
        0.59222933, 0.26553897, 0.22547243, 0.40028813, 0.3204898 ,
        0.34822874, 0.34412874, 0.57314248, 0.44616195, 0.46055901,
        0.35278792, 0.37317762, 0.14120289, 0.47297268, 0.075803  ,
        0.36184095, 0.27489761, 0.14011135, 0.09808857, 0.11531741,
        0.08029287, 0.45176795],
       [0.41816981, 0.44027757, 0.35668523, 0.56572701, 0.59222933,
        1.        , 0.38144145, 0.22509121, 0.47354328, 0.52911244,
        0.54252219, 0.35312124, 0.5172958 , 0.52822412, 0.61525194,
        0.44608741, 0.50674675, 0.06963289, 0.42337461, 0.18690773,
        0.41850638, 0.45240741, 0.18024682, 0.22976447, 0.24642674,
        0.21337675, 0.34207172],
       [0.29605353, 0.17970091, 0.53740406, 0.34303773, 0.26553897,
        0.38144145, 1.        , 0.49399344, 0.39103869, 0.67378007,
        0.46758061, 0.37902734, 0.36112378, 0.25924273, 0.2672615 ,
        0.16908968, 0.08986322, 0.57737721, 0.18291836, 0.35423738,
        0.66902792, 0.50313734, 0.50609375, 0.34378559, 0.09979508,
        0.27794012, 0.29922109],
       [0.31227489, 0.03840277, 0.41814325, 0.36231101, 0.22547243,
        0.22509121, 0.49399344, 1.        , 0.28086947, 0.25338261,
        0.16982115, 0.31936977, 0.26771558, 0.08133361, 0.23233787,
        0.33475159, 0.05521182, 0.30406254, 0.15234377, 0.03869207,
        0.37828933, 0.38576552, 0.30705639, 0.        , 0.14715344,
        0.30737873, 0.39264797],
       [0.82835423, 0.19777939, 0.30614128, 0.33034422, 0.40028813,
        0.47354328, 0.39103869, 0.28086947, 1.        , 0.44479648,
        0.28320435, 0.21582523, 0.59123596, 0.40779753, 0.25849976,
        0.09058867, 0.31955329, 0.17675771, 0.08623762, 0.19737153,
        0.41669189, 0.42747165, 0.19725169, 0.1105085 , 0.51967539,
        0.09380988, 0.59847463],
       [0.35071361, 0.26150968, 0.37308693, 0.35094711, 0.3204898 ,
        0.52911244, 0.67378007, 0.25338261, 0.44479648, 1.        ,
        0.64275651, 0.41023336, 0.38214967, 0.39070323, 0.41824048,
        0.07312089, 0.32132625, 0.39592137, 0.31218506, 0.58363823,
        0.7390936 , 0.63209256, 0.38655163, 0.52528699, 0.12428708,
        0.23798018, 0.31342278],
```

```
[0.17221508, 0.12907336, 0.16864772, 0.42278148, 0.34822874,
 0.54252219, 0.46758061, 0.16982115, 0.28320435, 0.64275651,
 1.        , 0.43294433, 0.33428324, 0.39911406, 0.41516079,
 0.13035811, 0.40639681, 0.1816831 , 0.12893185, 0.61251534,
 0.62345772, 0.61239816, 0.21006381, 0.60580183, 0.03956713,
 0.17218555, 0.14207064],
[0.3041334 , 0.21441901, 0.21273128, 0.38094506, 0.34412874,
 0.35312124, 0.37902734, 0.31936977, 0.21582523, 0.41023336,
 0.43294433, 1.        , 0.37552769, 0.32273952, 0.20448515,
 0.14592538, 0.27918894, 0.25625823, 0.23152743, 0.28940444,
 0.59246488, 0.5622339 , 0.45532423, 0.21757191, 0.22736665,
 0.19428979, 0.40895908],
[0.54150012, 0.39948681, 0.33222111, 0.62260256, 0.57314248,
 0.5172958 , 0.36112378, 0.26771558, 0.59123596, 0.38214967,
 0.33428324, 0.37552769, 1.        , 0.79034897, 0.6076059 ,
 0.49063538, 0.49050858, 0.31387991, 0.57310597, 0.20051796,
 0.51094197, 0.49709911, 0.23291308, 0.11447157, 0.40300164,
 0.23054059, 0.63509675],
[0.44539966, 0.46466525, 0.2708804 , 0.57529477, 0.44616195,
 0.52822412, 0.25924273, 0.08133361, 0.40779753, 0.39070323,
 0.39911406, 0.32273952, 0.79034897, 1.        , 0.63911511,
 0.38153688, 0.47993035, 0.10635134, 0.57247597, 0.15570923,
 0.41829992, 0.51135421, 0.11790353, 0.06044601, 0.44217034,
 0.41141375, 0.57774087],
[0.26513706, 0.39699107, 0.26285977, 0.60711865, 0.46055901,
 0.61525194, 0.2672615 , 0.23233787, 0.25849976, 0.41824048,
 0.41516079, 0.20448515, 0.6076059 , 0.63911511, 1.        ,
 0.55471462, 0.52640647, 0.19822399, 0.61078005, 0.1702625 ,
 0.43830941, 0.47768287, 0.10750849, 0.18359877, 0.15417651,
 0.44550074, 0.3340562 ],
[0.08735731, 0.25267484, 0.37745334, 0.43588408, 0.35278792,
 0.44608741, 0.16908968, 0.33475159, 0.09058867, 0.07312089,
 0.13035811, 0.14592538, 0.49063538, 0.38153688, 0.55471462,
 1.        , 0.21190849, 0.43223054, 0.50939662, 0.        ,
 0.25377963, 0.21433544, 0.05114705, 0.06862972, 0.        ,
 0.16853576, 0.30862222],
[0.27681686, 0.16365961, 0.10029936, 0.28597736, 0.37317762,
 0.50674675, 0.08986322, 0.05521182, 0.31955329, 0.32132625,
 0.40639681, 0.27918894, 0.49050858, 0.47993035, 0.52640647,
 0.21190849, 1.        , 0.        , 0.37988182, 0.39954726,
 0.3063798 , 0.34127802, 0.        , 0.27697043, 0.1447196 ,
 0.16794158, 0.2499955 ],
[0.10227159, 0.12325556, 0.37370127, 0.12042511, 0.14120289,
 0.06963289, 0.57737721, 0.30406254, 0.17675771, 0.39592137,
 0.1816831 , 0.25625823, 0.31387991, 0.10635134, 0.19822399,
 0.43223054, 0.        , 1.        , 0.26540716, 0.33633189,
 0.45232597, 0.31696213, 0.29107965, 0.33477785, 0.        ,
 0.14615513, 0.34965722],
[0.19656347, 0.37963804, 0.31576952, 0.48405061, 0.47297268,
 0.42337461, 0.18291836, 0.15234377, 0.08623762, 0.31218506,
 0.12893185, 0.23152743, 0.57310597, 0.57247597, 0.61078005,
 0.50939662, 0.37988182, 0.26540716, 1.        , 0.11015942,
 0.3689791 , 0.37147129, 0.19918805, 0.06929298, 0.13965255,
 0.34032908, 0.36617158],
[0.09882579, 0.        , 0.02108673, 0.        , 0.075803  ,
 0.18690773, 0.35423738, 0.03869207, 0.19737153, 0.58363823,
 0.61251534, 0.28940444, 0.20051796, 0.15570923, 0.1702625 ,
 0.        , 0.39954726, 0.33633189, 0.11015942, 1.        ,
 0.56642248, 0.57002643, 0.19644423, 0.84828448, 0.10141851,
 0.11769232, 0.13348202],
[0.3852686 , 0.25313087, 0.34450389, 0.54299787, 0.36184095,
 0.41850638, 0.66902792, 0.37828933, 0.41669189, 0.7390936 ,
 0.62345772, 0.59246488, 0.51094197, 0.41829992, 0.43830941,
 0.25377963, 0.3063798 , 0.45232597, 0.3689791 , 0.56642248,
 1.        , 0.77975977, 0.54110587, 0.45790805, 0.22466739,
```

```
       0.42958295, 0.43741342],
      [0.36294349, 0.20203635, 0.22848954, 0.47611277, 0.27489761,
       0.45240741, 0.50313734, 0.38576552, 0.42747165, 0.63209256,
       0.61239816, 0.5622339 , 0.49709911, 0.51135421, 0.47768287,
       0.21433544, 0.34127802, 0.31696213, 0.37147129, 0.57002643,
       0.77975977, 1.        , 0.42922523, 0.43577727, 0.33795123,
       0.43974172, 0.51414596],
      [0.18041067, 0.18434039, 0.45471893, 0.32947492, 0.14011135,
       0.18024682, 0.50609375, 0.30705639, 0.19725169, 0.38655163,
       0.21006381, 0.45532423, 0.23291308, 0.11790353, 0.10750849,
       0.05114705, 0.        , 0.29107965, 0.19918805, 0.19644423,
       0.54110587, 0.42922523, 1.        , 0.16176206, 0.10867135,
       0.42666866, 0.35645222],
      [0.        , 0.        , 0.        , 0.        , 0.09808857,
       0.22976447, 0.34378559, 0.        , 0.1105085 , 0.52528699,
       0.60580183, 0.21757191, 0.11447157, 0.06044601, 0.18359877,
       0.06862972, 0.27697043, 0.33477785, 0.06929298, 0.84828448,
       0.45790805, 0.43577727, 0.16176206, 1.        , 0.        ,
       0.        , 0.        ],
      [0.52433909, 0.39145634, 0.08465232, 0.1488889 , 0.11531741,
       0.24642674, 0.09979508, 0.14715344, 0.51967539, 0.12428708,
       0.03956713, 0.22736665, 0.40300164, 0.44217034, 0.15417651,
       0.        , 0.1447196 , 0.        , 0.13965255, 0.10141851,
       0.22466739, 0.33795123, 0.10867135, 0.        , 1.        ,
       0.48407839, 0.6070744 ],
      [0.17769639, 0.37185258, 0.3660576 , 0.26717479, 0.08029287,
       0.21337675, 0.27794012, 0.30737873, 0.09380988, 0.23798018,
       0.17218555, 0.19428979, 0.23054059, 0.41141375, 0.44550074,
       0.16853576, 0.16794158, 0.14615513, 0.34032908, 0.11769232,
       0.42958295, 0.43974172, 0.42666866, 0.        , 0.48407839,
       1.        , 0.40182743],
      [0.71075896, 0.42543225, 0.39251867, 0.48570938, 0.45176795,
       0.34207172, 0.29922109, 0.39264797, 0.59847463, 0.31342278,
       0.14207064, 0.40895908, 0.63509675, 0.57774087, 0.3340562 ,
       0.30862222, 0.2499955 , 0.34965722, 0.36617158, 0.13348202,
       0.43741342, 0.51414596, 0.35645222, 0.        , 0.6070744 ,
       0.40182743, 1.        ]])
```

In [45]:

```
product_ID = int(input('Enter Product ID according to data set : '))
data = list(df_pivot.index) #shows list of ProductID in data-set
print(data)
```

```
Enter Product ID according to data set : 369
[89, 333, 369, 444, 684, 697, 767, 777, 823, 847, 853, 858, 862, 910, 949, 1002,
1003, 1049, 1060, 1065, 1077, 1080, 1095, 1120, 6969, 8001, 9696]
```

In [46]:

```
query_index = data.index(product_ID) #shows index of productID by USER
print(query_index)
```

```
2
```

In [47]:

```
similarity, indices = model_knn.kneighbors(df_pivot.iloc[query_index,:].values.r
print(similarity) #shows similarity distance through productID by USER
print(indices) #shows indexs of productID by USER
```

```
[[ 0.         12.27576655 12.86791876 12.99679448 13.05144734 13.09898215
  13.7343244  13.89644239]]
[[ 2  6  7 15  1 17 26 22]]
```

In [48]:

```
i=df_pivot.index[indices.flatten()]
i
```

Out[48]:
```
Int64Index([369, 767, 777, 1002, 333, 1049, 9696, 1095], dtype='int64', name='Pr
oductID')
```

In [49]:
```
d=similarity.flatten()
d
```

Out[49]:
```
array([ 0.        , 12.27576655, 12.86791876, 12.99679448, 13.05144734,
       13.09898215, 13.7343244 , 13.89644239])
```

In [50]:
```
new=list(zip(i,d))
new
```

Out[50]:
```
[(369, 0.0),
 (767, 12.27576655221353),
 (777, 12.86791876463841),
 (1002, 12.996794476587935),
 (333, 13.051447344175196),
 (1049, 13.098982148752372),
 (9696, 13.734324397896922),
 (1095, 13.896442390450554)]
```

In [51]:
```
pd.DataFrame(new)
```

Out[51]:

	0	1
0	369	0.000000
1	767	12.275767
2	777	12.867919
3	1002	12.996794
4	333	13.051447
5	1049	13.098982
6	9696	13.734324
7	1095	13.896442

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Create dictionary and store recommendations of productID which is given by USER

In [ ]:
```
data_dict={}
for i in range(0, len(similarity.flatten())):

    if i == 0:
        print('Recommendations for {0}:\n'.format(df_pivot.index[query_index]))
    else:
        data_dict[str(df_pivot.index[indices.flatten()[i]])] = float(similarity.
        print(f'{df_pivot.index[indices.flatten()[i]]}, is similarity distance =

print(data_dict)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: