> length(unique(user\_artists$userID))

[1] 1892

> length(unique(user\_artists$artistID))

[1] 17632

> length(unique(user\_taggedartists$userID))

[1] 1892

> length(unique(user\_taggedartists$artistID))

[1] 12523

> View(artists)

> user\_artists <- read.delim("G:/BITS/4-2/Information retrieval/my\_assignment/last\_fm\_dataset/user\_artists.dat")

> View(user\_artists)

> memory.size()

[1] 49.23

<https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/> - knn from scratch

<https://pandas.pydata.org/pandas-docs/stable/comparison_with_r.html> -> R and python code comparison

<https://github.com/letiantian/kmedoids/blob/master/kmedoids.py> ->kmedoids code

# 3 points in dataset

data = np.array([[1,1],

[2,2],

[10,10]])

# distance matrix

D = pairwise\_distances(data, metric='euclidean')

# split into 2 clusters

M , C= kMedoids(D, 2)

print('medoids:')

for point\_idx in M:

print( data[point\_idx] )

print('')

print('clustering result:')

for label in C:

for point\_idx in C[label]:

print('label {0}:　{1}'.format(label, data[point\_idx]))

<https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/>

Setup complete in 658.4465420246124 time

154 ,score: 0.0568964085528

163 ,score: 0.0568964085528

185 ,score: 0.0568964085528

209 ,score: 0.0568964085528

357 ,score: 0.0568964085528

562 ,score: 0.0568964085528

563 ,score: 0.0568964085528

704 ,score: 0.0568964085528

868 ,score: 0.0568964085528

874 ,score: 0.0568964085528

1104 ,score: 0.0568964085528

1122 ,score: 0.0568964085528

1412 ,score: 0.0568964085528

1414 ,score: 0.0568964085528

1415 ,score: 0.0568964085528

1793 ,score: 0.0568964085528

2444 ,score: 0.0568964085528

2526 ,score: 0.0568964085528

4123 ,score: 0.0568964085528

7792 ,score: 0.0568964085528

9646 ,score: 0.0568964085528

9647 ,score: 0.0568964085528

86 ,score: 0.056236896068

917 ,score: 0.0555773835831

954 ,score: 0.0555773835831

961 ,score: 0.0555773835831

1130 ,score: 0.0555773835831

1274 ,score: 0.0555773835831

1406 ,score: 0.0555773835831

1979 ,score: 0.0555773835831

3227 ,score: 0.0555773835831

3500 ,score: 0.0555773835831

3940 ,score: 0.0555773835831

4917 ,score: 0.0555773835831

7232 ,score: 0.0555773835831

7241 ,score: 0.0555773835831

8281 ,score: 0.0555773835831

8282 ,score: 0.0555773835831

8283 ,score: 0.0555773835831

8284 ,score: 0.0555773835831

8285 ,score: 0.0555773835831

8286 ,score: 0.0555773835831

8288 ,score: 0.0555773835831

8289 ,score: 0.0555773835831

8290 ,score: 0.0555773835831

8291 ,score: 0.0555773835831

2179 ,score: 0.0536534669525

314 ,score: 0.0525669885227

330 ,score: 0.0517295503219

468 ,score: 0.0517295503219

Recommendation complete in 35.18560004234314 time