K-means clustering

import library

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from matplotlib import cm
```

In [ ]:
```
# 색상 전체 명단 수집
colorlist = {}
colorlist.update(colors.CSS4_COLORS)

# 색상 이름과 코드로 분리해서 관리
colornames = []
colorcodes = []
for name, color in colorlist.items():
    colornames.append(name)
    colorcodes.append(color)
```

In [ ]:
```
colornames
```

Out[ ]:
```
['aliceblue',
 'antiquewhite',
 'aqua',
 'aquamarine',
 'azure',
 'beige',
 'bisque',
 'black',
 'blanchedalmond',
 'blue',
 'blueviolet',
 'brown',
 'burlywood',
 'cadetblue',
 'chartreuse',
 'chocolate',
 'coral',
 'cornflowerblue',
 'cornsilk',
 'crimson',
 'cyan',
 'darkblue',
 'darkcyan',
 'darkgoldenrod',
 'darkgray',
 'darkgreen',
 'darkgrey',
 'darkkhaki',
 'darkmagenta',
 'darkolivegreen',
 'darkorange',
 'darkorchid',
 'darkred',
 'darksalmon',
 'darkseagreen',
 'darkslateblue',
 'darkslategray',
 'darkslategrey',
 'darkturquoise',
 'darkviolet',
```

```
'deeppink',
'deepskyblue',
'dimgray',
'dimgrey',
'dodgerblue',
'firebrick',
'floralwhite',
'forestgreen',
'fuchsia',
'gainsboro',
'ghostwhite',
'gold',
'goldenrod',
'gray',
'green',
'greenyellow',
'grey',
'honeydew',
'hotpink',
'indianred',
'indigo',
'ivory',
'khaki',
'lavender',
'lavenderblush',
'lawngreen',
'lemonchiffon',
'lightblue',
'lightcoral',
'lightcyan',
'lightgoldenrodyellow',
'lightgray',
'lightgreen',
'lightgrey',
'lightpink',
'lightsalmon',
'lightseagreen',
'lightskyblue',
'lightslategray',
'lightslategrey',
'lightsteelblue',
'lightyellow',
'lime',
'limegreen',
'linen',
'magenta',
'maroon',
'mediumaquamarine',
'mediumblue',
'mediumorchid',
'mediumpurple',
'mediumseagreen',
'mediumslateblue',
'mediumspringgreen',
'mediumturquoise',
'mediumvioletred',
'midnightblue',
'mintcream',
'mistyrose',
'moccasin',
'navajowhite',
'navy',
'oldlace',
'olive',
'olivedrab',
'orange',
'orangered',
'orchid',
'palegoldenrod',
```

```
'palegreen',
'paleturquoise',
'palevioletred',
'papayawhip',
'peachpuff',
'peru',
'pink',
'plum',
'powderblue',
'purple',
'rebeccapurple',
'red',
'rosybrown',
'royalblue',
'saddlebrown',
'salmon',
'sandybrown',
'seagreen',
'seashell',
'sienna',
'silver',
'skyblue',
'slateblue',
'slategray',
'slategrey',
'snow',
'springgreen',
'steelblue',
'tan',
'teal',
'thistle',
'tomato',
'turquoise',
'violet',
'wheat',
'white',
'whitesmoke',
'yellow',
'yellowgreen']
```

load data

In [ ]:
```
fname_data = 'assignment_11_data.csv'

feature = np.genfromtxt(fname_data, delimiter=',')

x = feature[:,0]
y = feature[:,1]

number_data     = np.size(feature, 0)
number_feature  = np.size(feature, 1)

print('number of data : {}'.format(number_data))
print('number of feature : {}'.format(number_feature))
```

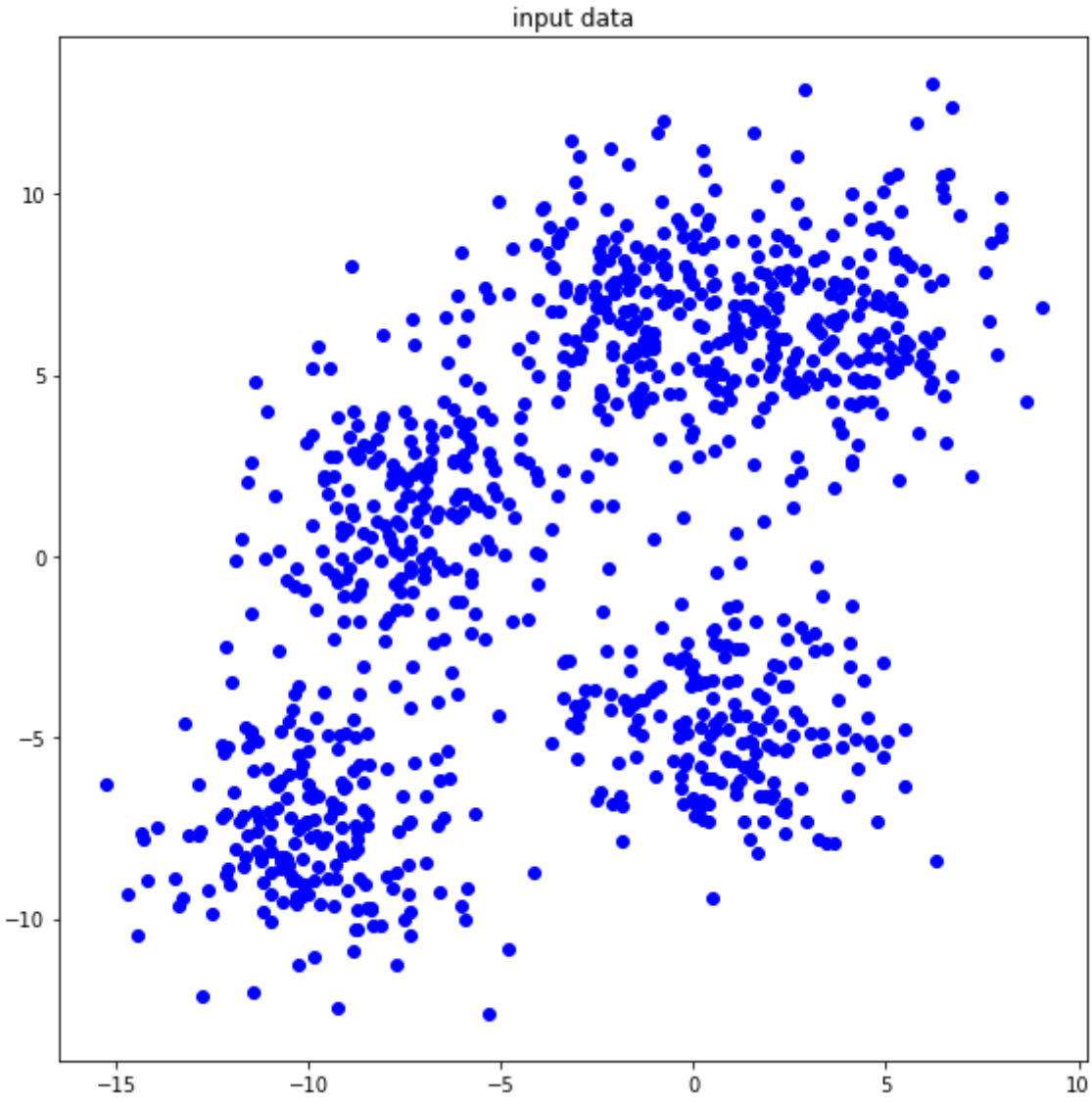
```
number of data : 1000
number of feature : 2
```

In [ ]:
```
feature
```

Out[ ]:
```
array([[-2.78158782,  6.13851704],
       [-5.79072687,  3.00770345],
       [-8.85288476,  3.99889271],
       ...,
       [ 8.01075285,  9.06955099],
       [ 1.12859575, -2.53389057],
       [ 0.03723381,  8.87194034]])
```

plot the input data

In [ ]:
```
plt.figure(figsize=(8,8))
plt.title('input data')

plt.scatter(x, y, color='blue')

plt.tight_layout()
plt.show()
```



compute distance

- feature : $n \times m$, center : $1 \times m$, distance : $n \times 1$
- n : number of data, m : number of features

In [ ]:
```
def compute_distance(feature, center):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
```

```
    distance = np.sqrt(np.sum(np.square(feature - center),axis=1))



    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return distance
```

compute centroid

- feature : $n \times m$, label_feature : $n \times 1$, value_label : 1×1, centroid : $1 \times m$
- n : number of data, m : number of features

In [ ]:
```
def compute_centroid(feature, label_feature, label):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #



    c_k = feature[label_feature==label].shape[0]


    centroid = np.divide(np.sum(feature[label_feature==label],axis=0),c_k)


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return centroid
```

compute label

- distance : $n \times k$, label_feature : $n \times 1$
- n : number of data, k : number of clusters

In [ ]:
```
def compute_label(distance):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #

    n = distance.shape[0]

    label_feature=np.zeros((n,))
    for i in range(n):
        label_feature[i]= distance[i,:].argmin()


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return label_feature
```

In [ ]:
```
distance=np.array([[1,2],[4,3],[6,5]])
compute_label(distance)
```

Out[ ]:
```
array([0., 1., 1.])
```

the number of clusters $K = 2$

In [ ]:
```
number_cluster          = 2
number_iteration        = 1000     # you can modify this value
loss_iteration_02       = np.zeros(number_iteration)
centroid_iteration_02   = np.zeros((number_iteration, number_cluster, number_feature)
label_feature_02        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#

def loss(distance, number_cluster,label_feature):
    n = distance.shape[0]
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
        loss = loss/n
    return loss


for i in range(number_iteration):
    distance_02 = np.zeros((number_data,number_cluster))
    for j in range(number_cluster):
        label = j
        if feature[label_feature_02==label].shape[0] == 0 :
            center = centroid_iteration_02[i-1,j,:]
        else:
            center = compute_centroid(feature, label_feature_02, label)
        distance_02[:,j] = compute_distance(feature, center)
        centroid_iteration_02[i,j,:] = center


    label_feature_02 = compute_label(distance_02)
    loss_iteration_02[i] = loss(distance_02,number_cluster,label_feature_02)









#
# ++++++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 4$

In [ ]:
```
label_feature_02
```

Out[ ]:
```
array([0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1.,
       1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0.,
       1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1.,
       1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0.,
       1., 0., 0., 1., 0., 0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 0., 1.,
       1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1.,
       0., 0., 0., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1.,
       1., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0.,
       1., 0., 0., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0.,
       0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 0., 0., 1., 0., 0., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0.,
```

```
       0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0., 1.,
       1., 0., 1., 0., 1., 0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 0., 1.,
       0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 1., 0.,
       1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 0., 1.,
       1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 0.,
       1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 0., 1.,
       1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 0.,
       0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1.,
       0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 1., 1., 0.,
       1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 0., 0., 1., 1.,
       0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 1.,
       0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0.,
       1., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 1.,
       1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0.,
       0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 1.,
       0., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
       1., 1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1.,
       1., 0., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1.,
       1., 1., 1., 0., 1., 0., 0., 0., 1., 0., 1., 1., 0., 0., 0., 1., 1.,
       1., 1., 0., 1., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0.,
       0., 1., 0., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 0.,
       0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0.,
       1., 0., 1., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0., 0.,
       1., 0., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
       0., 0., 1., 1., 1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 0.,
       1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0.,
       0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 1., 1., 1.,
       1., 1., 0., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1.,
       1., 0., 0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0., 0., 0.,
       0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0.,
       1., 0., 0., 0., 1., 0., 1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 1.,
       0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1.,
       0., 0., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 1.,
       0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 1., 0., 0.,
       1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 1., 0.,
       1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0.,
       1., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0.,
       0., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 1., 0.])
```

In [ ]:
```
number_cluster          = 4
number_iteration        = 1000     # you can modify this value
loss_iteration_04       = np.zeros(number_iteration)
centroid_iteration_04   = np.zeros((number_iteration, number_cluster, number_feature))
label_feature_04        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#


def loss(distance, number_cluster,label_feature):
    n = distance.shape[0]
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
        loss = loss/n
    return loss
```

```
for i in range(number_iteration):
    distance_04 = np.zeros((number_data,number_cluster))
    for j in range(number_cluster):
        label = j
        if feature[label_feature_04==label].shape[0] == 0 :
            center = centroid_iteration_04[i-1,j,:]
        else:
            center = compute_centroid(feature, label_feature_04, label)
        distance_04[:,j] = compute_distance(feature, center)
        centroid_iteration_04[i,j,:] = center

    label_feature_04 = compute_label(distance_04)
    loss_iteration_04[i] = loss(distance_04, number_cluster,label_feature_04)

#
# ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
label_feature_04
```

Out[ ]:
```
array([0., 3., 3., 0., 2., 1., 3., 0., 0., 2., 1., 2., 0., 2., 0., 3., 2.,
       2., 3., 2., 2., 0., 0., 1., 3., 3., 2., 0., 0., 3., 3., 0., 0., 0.,
       3., 2., 3., 3., 3., 0., 3., 0., 2., 1., 2., 0., 0., 1., 0., 1., 1.,
       1., 0., 2., 0., 2., 1., 3., 1., 3., 1., 1., 2., 0., 1., 3., 1., 0.,
       1., 0., 0., 2., 0., 0., 1., 2., 2., 0., 3., 2., 0., 0., 1., 3., 3.,
       1., 2., 1., 2., 0., 0., 0., 0., 1., 0., 3., 0., 0., 0., 1., 0., 2.,
       0., 0., 0., 0., 2., 2., 3., 0., 2., 2., 2., 3., 3., 0., 0., 0., 3.,
       1., 3., 3., 0., 3., 0., 2., 3., 0., 3., 3., 0., 0., 2., 2., 2., 0.,
       2., 0., 0., 3., 2., 0., 0., 3., 1., 0., 2., 1., 1., 1., 2., 0., 0.,
       0., 1., 2., 0., 0., 2., 0., 0., 2., 0., 3., 1., 1., 1., 2., 3., 0.,
       1., 3., 0., 3., 3., 0., 0., 1., 3., 0., 2., 3., 1., 0., 1., 0., 3.,
       0., 1., 0., 2., 0., 0., 3., 0., 0., 2., 3., 1., 1., 0., 0., 0., 1.,
       2., 0., 1., 0., 1., 0., 2., 3., 3., 3., 0., 2., 0., 2., 2., 0., 3.,
       0., 2., 2., 3., 0., 1., 0., 0., 1., 3., 2., 2., 0., 2., 3., 1., 0.,
       1., 1., 0., 3., 0., 3., 2., 0., 1., 3., 3., 0., 0., 1., 0., 0., 3.,
       2., 0., 3., 0., 1., 0., 1., 0., 1., 3., 3., 2., 1., 3., 3., 2., 1.,
       1., 2., 0., 0., 0., 0., 0., 3., 1., 3., 2., 3., 2., 0., 1., 3., 0.,
       3., 2., 1., 0., 1., 3., 0., 3., 1., 0., 3., 0., 3., 0., 2., 0., 3.,
       1., 1., 0., 1., 2., 1., 3., 0., 0., 2., 1., 1., 2., 2., 3., 0., 3.,
       1., 1., 3., 1., 3., 2., 3., 3., 3., 3., 0., 0., 2., 2., 2., 2., 0.,
       0., 0., 0., 0., 0., 3., 0., 2., 2., 0., 3., 0., 1., 1., 3., 0., 3.,
       2., 3., 2., 2., 3., 2., 3., 3., 0., 3., 1., 1., 1., 3., 3., 0., 1.,
       0., 0., 0., 3., 0., 3., 0., 0., 3., 3., 0., 1., 0., 0., 1., 2., 0.,
       2., 1., 1., 1., 0., 0., 2., 3., 1., 0., 3., 0., 2., 0., 0., 2., 1.,
       0., 1., 1., 1., 0., 3., 2., 1., 1., 1., 0., 0., 2., 2., 2., 1., 3.,
       1., 0., 3., 2., 0., 3., 0., 3., 0., 1., 0., 1., 2., 0., 3., 2., 3.,
       0., 2., 3., 1., 0., 1., 2., 2., 1., 2., 0., 1., 2., 1., 0., 1., 3.,
       3., 1., 2., 2., 3., 1., 0., 2., 3., 0., 3., 0., 0., 0., 0., 2., 3.,
       2., 0., 0., 0., 1., 0., 3., 0., 1., 0., 2., 2., 3., 1., 1., 1., 3.,
       3., 3., 2., 0., 0., 3., 2., 1., 0., 2., 0., 1., 3., 3., 0., 0., 0.,
       0., 2., 0., 3., 2., 0., 3., 0., 0., 3., 3., 3., 0., 0., 3., 2., 3.,
       2., 1., 3., 1., 2., 1., 2., 2., 2., 0., 2., 0., 1., 0., 2., 0., 2.,
       2., 0., 1., 2., 0., 1., 0., 3., 1., 0., 3., 1., 2., 2., 2., 0., 2.,
       2., 2., 0., 0., 0., 2., 3., 3., 3., 3., 3., 3., 1., 2., 0., 0., 1.,
       2., 0., 2., 0., 0., 0., 0., 2., 1., 1., 3., 1., 3., 0., 2., 0., 1.,
       3., 1., 3., 2., 1., 0., 3., 0., 3., 0., 2., 1., 0., 0., 3., 2., 1.,
       3., 2., 3., 1., 3., 0., 2., 0., 2., 0., 0., 0., 0., 0., 2., 1., 0.,
       0., 2., 3., 1., 0., 2., 0., 0., 0., 3., 1., 2., 0., 0., 0., 3., 0.,
       3., 3., 3., 1., 0., 3., 2., 3., 3., 0., 2., 0., 1., 1., 2., 3., 3.,
       2., 3., 2., 1., 1., 0., 1., 3., 3., 1., 1., 0., 0., 2., 0., 3., 0.,
       3., 2., 3., 0., 1., 0., 0., 0., 0., 2., 3., 0., 2., 1., 0., 0., 3.,
       2., 0., 1., 1., 0., 2., 3., 0., 3., 0., 1., 2., 1., 3., 3., 0., 1.,
       0., 0., 3., 2., 1., 2., 2., 0., 0., 2., 1., 0., 3., 2., 2., 2., 0.,
       2., 1., 0., 3., 3., 3., 1., 3., 1., 1., 0., 3., 3., 1., 0., 1., 0.,
       0., 0., 1., 1., 1., 0., 1., 0., 2., 3., 3., 0., 0., 0., 3., 1., 3.,
       3., 0., 2., 3., 1., 3., 0., 0., 3., 2., 1., 2., 1., 2., 0., 2., 2.,
       2., 0., 1., 0., 3., 3., 1., 3., 0., 0., 2., 0., 0., 0., 3., 1., 2.,
```

```
3., 1., 0., 2., 1., 0., 0., 1., 0., 1., 0., 2., 1., 0., 1., 0., 2.,
1., 0., 0., 1., 3., 0., 3., 1., 0., 3., 0., 0., 1., 0., 2., 0., 0.,
0., 1., 0., 2., 0., 0., 0., 0., 2., 3., 0., 1., 1., 0., 0., 0., 0.,
0., 0., 0., 2., 2., 3., 3., 2., 1., 3., 3., 1., 2., 2., 0., 2., 0.,
1., 0., 3., 3., 2., 0., 2., 1., 3., 1., 3., 0., 2., 0., 0., 0., 3.,
0., 0., 2., 0., 0., 0., 3., 0., 3., 0., 2., 1., 0., 0., 0., 0., 2.,
2., 0., 1., 2., 0., 0., 2., 3., 1., 0., 3., 1., 1., 0., 0., 1., 3.,
3., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 2., 1., 3., 0.,
1., 0., 2., 0., 0., 0., 0., 1., 0., 3., 1., 1., 0., 2., 0., 2., 0.,
3., 0., 1., 0., 3., 0., 0., 0., 0., 0., 0., 2., 3., 2., 1., 3., 0.,
3., 1., 1., 3., 0., 2., 1., 0., 0., 2., 0., 0., 0., 0., 1., 1., 0.,
0., 3., 1., 0., 1., 2., 1., 0., 0., 3., 3., 0., 2., 0.])
```

the number of clusters $K = 8$

In [ ]:
```
number_cluster          = 8
number_iteration        = 1000     # you can modify this value
loss_iteration_08       = np.zeros(number_iteration)
centroid_iteration_08   = np.zeros((number_iteration, number_cluster, number_feature))
label_feature_08        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#

def loss(distance, number_cluster,label_feature):
    n = distance.shape[0]
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
        loss = loss/n
    return loss


for i in range(number_iteration):
    distance_08 = np.zeros((number_data,number_cluster))
    for j in range(number_cluster):
        label = j
        if feature[label_feature_08==label].shape[0] == 0 :
            center = centroid_iteration_08[i-1,j,:]
        else:
            center = compute_centroid(feature, label_feature_08, label)
        distance_08[:,j] = compute_distance(feature, center)
        centroid_iteration_08[i,j,:] = center

    label_feature_08 = compute_label(distance_08)
    loss_iteration_08[i] = loss(distance_08, number_cluster,label_feature_08)


#
# ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
label_feature_08
```

Out[ ]:
```
array([5., 0., 0., 5., 3., 4., 6., 1., 1., 2., 4., 1., 5., 2., 5., 0., 3.,
       3., 6., 3., 2., 1., 1., 4., 0., 0., 3., 1., 1., 0., 5., 7., 1., 7.,
       0., 3., 0., 0., 6., 7., 0., 7., 2., 4., 2., 7., 1., 4., 5., 4., 4.,
       4., 5., 2., 7., 2., 4., 0., 4., 0., 4., 4., 2., 1., 4., 0., 4., 1.,
       4., 7., 5., 2., 5., 7., 4., 2., 3., 5., 5., 2., 7., 7., 4., 0., 6.,
       4., 3., 3., 3., 7., 5., 5., 1., 4., 1., 6., 7., 1., 1., 4., 5., 2.,
       7., 7., 5., 1., 2., 3., 0., 7., 2., 3., 3., 6., 0., 7., 7., 1., 6.,
       4., 6., 1., 1., 0., 7., 2., 6., 5., 6., 0., 1., 1., 3., 2., 2., 5.,
       2., 5., 1., 0., 2., 7., 5., 6., 4., 7., 2., 6., 4., 4., 2., 1., 5.,
       5., 4., 3., 5., 7., 2., 7., 5., 2., 1., 0., 4., 4., 4., 3., 0., 1.,
```

```
4., 0., 1., 0., 0., 5., 5., 4., 0., 5., 2., 6., 4., 5., 4., 1., 0.,
1., 4., 7., 2., 7., 7., 0., 5., 7., 3., 0., 4., 4., 7., 7., 5., 4.,
2., 7., 4., 5., 4., 5., 3., 6., 6., 0., 7., 2., 7., 2., 2., 1., 6.,
1., 3., 3., 0., 7., 4., 5., 1., 4., 5., 3., 2., 1., 3., 0., 4., 1.,
6., 4., 1., 0., 7., 6., 2., 7., 4., 0., 0., 1., 5., 4., 5., 1., 6.,
3., 7., 0., 5., 4., 7., 4., 1., 4., 6., 6., 2., 4., 6., 6., 3., 4.,
4., 1., 5., 5., 5., 5., 7., 0., 4., 0., 3., 0., 2., 5., 6., 0., 1.,
6., 2., 4., 7., 4., 6., 7., 3., 4., 5., 3., 1., 0., 5., 3., 1., 6.,
4., 4., 1., 4., 2., 4., 5., 5., 1., 3., 4., 4., 3., 3., 6., 7., 0.,
4., 4., 6., 4., 0., 2., 0., 0., 6., 0., 7., 1., 2., 2., 3., 3., 1.,
1., 5., 5., 5., 7., 0., 5., 3., 3., 5., 0., 5., 4., 4., 6., 5., 0.,
2., 6., 2., 2., 6., 2., 0., 6., 1., 6., 4., 4., 4., 0., 6., 5., 4.,
1., 1., 1., 0., 5., 0., 7., 1., 0., 6., 5., 4., 5., 5., 4., 3., 1.,
2., 4., 6., 4., 7., 5., 2., 6., 4., 5., 6., 1., 2., 7., 1., 2., 4.,
7., 6., 4., 4., 5., 6., 2., 6., 4., 4., 5., 7., 2., 2., 3., 4., 0.,
4., 1., 0., 2., 7., 0., 1., 6., 1., 4., 7., 4., 2., 5., 0., 2., 0.,
5., 2., 6., 4., 5., 4., 2., 2., 4., 3., 7., 4., 2., 4., 5., 4., 0.,
6., 4., 3., 2., 0., 4., 1., 3., 6., 5., 6., 7., 5., 5., 7., 2., 6.,
3., 5., 1., 5., 4., 1., 0., 5., 4., 1., 2., 2., 0., 4., 4., 4., 0.,
6., 6., 3., 7., 1., 0., 2., 6., 7., 3., 5., 4., 6., 6., 1., 7., 7.,
7., 2., 7., 6., 2., 5., 0., 7., 5., 0., 6., 0., 7., 5., 6., 2., 6.,
2., 6., 0., 4., 3., 4., 3., 2., 3., 1., 2., 7., 4., 5., 2., 7., 3.,
2., 5., 4., 3., 7., 4., 1., 0., 4., 5., 6., 4., 3., 3., 2., 1., 2.,
2., 2., 7., 1., 7., 2., 6., 6., 0., 0., 0., 6., 4., 3., 1., 1., 4.,
3., 7., 3., 7., 1., 7., 5., 3., 4., 4., 6., 4., 0., 5., 2., 5., 4.,
0., 4., 0., 3., 4., 5., 5., 5., 0., 7., 3., 4., 5., 5., 5., 2., 4.,
0., 2., 0., 4., 6., 1., 2., 5., 2., 5., 1., 1., 7., 5., 2., 4., 7.,
5., 3., 0., 4., 5., 3., 7., 7., 7., 0., 4., 2., 7., 1., 5., 0., 7.,
0., 0., 0., 4., 1., 6., 2., 6., 0., 1., 3., 1., 4., 4., 3., 0., 0.,
3., 6., 2., 4., 4., 7., 4., 0., 6., 4., 6., 5., 5., 2., 5., 0., 1.,
0., 2., 0., 1., 4., 5., 7., 5., 1., 2., 0., 1., 3., 4., 5., 5., 0.,
3., 1., 4., 6., 5., 3., 6., 7., 6., 7., 4., 3., 4., 0., 0., 7., 4.,
5., 1., 0., 2., 4., 2., 3., 5., 7., 2., 4., 5., 0., 2., 2., 3., 7.,
3., 4., 7., 0., 6., 0., 4., 0., 4., 4., 7., 0., 5., 6., 5., 4., 1.,
5., 1., 4., 4., 4., 7., 4., 1., 2., 6., 6., 7., 1., 7., 0., 4., 0.,
6., 5., 3., 6., 4., 0., 7., 5., 0., 2., 4., 3., 4., 3., 7., 3., 2.,
2., 1., 4., 7., 6., 6., 4., 0., 1., 7., 3., 1., 7., 7., 6., 6., 2.,
0., 3., 5., 2., 4., 1., 1., 4., 5., 4., 1., 3., 4., 1., 4., 7., 2.,
4., 1., 1., 4., 0., 7., 0., 4., 7., 6., 5., 1., 4., 5., 2., 1., 5.,
5., 4., 5., 2., 5., 7., 7., 5., 2., 6., 5., 4., 4., 1., 5., 1., 7.,
7., 5., 5., 2., 2., 0., 6., 2., 4., 6., 0., 4., 3., 3., 7., 2., 1.,
4., 7., 0., 0., 3., 1., 3., 4., 0., 4., 6., 7., 2., 7., 5., 7., 6.,
5., 7., 2., 1., 7., 5., 0., 7., 6., 7., 3., 4., 5., 7., 1., 7., 2.,
1., 7., 4., 2., 7., 7., 2., 6., 4., 7., 0., 4., 4., 7., 5., 4., 6.,
0., 5., 7., 7., 4., 1., 5., 4., 5., 4., 5., 5., 1., 3., 4., 5., 1.,
4., 7., 3., 5., 5., 5., 1., 4., 7., 5., 4., 4., 5., 2., 1., 3., 1.,
6., 5., 4., 7., 6., 7., 5., 7., 7., 5., 5., 2., 5., 3., 4., 5., 7.,
0., 4., 4., 0., 7., 2., 4., 1., 5., 3., 7., 5., 7., 1., 4., 4., 7.,
5., 0., 4., 7., 4., 2., 4., 7., 1., 0., 0., 7., 3., 5.])
```

the number of clusters $K = 16$

In [ ]:
```
number_cluster          = 16
number_iteration        = 1000     # you can modify this value
loss_iteration_16       = np.zeros(number_iteration)
centroid_iteration_16   = np.zeros((number_iteration, number_cluster, number_feature)
label_feature_16        = np.random.randint(0, number_cluster, size=(number_data))
```

In [ ]:
```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
def loss(distance, number_cluster,label_feature):
    n = distance.shape[0]
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
```

```
        loss = loss/n
    return loss


for i in range(number_iteration):
    distance_16 = np.zeros((number_data,number_cluster))
    for j in range(number_cluster):
        label = j
        if feature[label_feature_16==label].shape[0] == 0 :
            center = centroid_iteration_16[i-1,j,:]
        else:
            center = compute_centroid(feature, label_feature_16, label)
        distance_16[:,j] = compute_distance(feature, center)
        centroid_iteration_16[i,j,:] = center

    label_feature_16 = compute_label(distance_16)
    loss_iteration_16[i] = loss(distance_16, number_cluster,label_feature_16)

#
# ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
label_feature_16
```

Out[ ]:
```
array([15.,  5., 11., 15.,  9.,  8.,  1., 13.,  4., 12.,  8.,  6.,  7.,
       12.,  7.,  3.,  6.,  9., 11.,  6., 12., 13.,  2.,  8.,  5., 11.,
        3.,  2., 13.,  5., 15.,  4., 13.,  0., 11.,  9., 11., 11., 10.,
        0.,  5.,  4.,  6.,  8., 12., 13., 13.,  8.,  7.,  8.,  8.,  8.,
       15., 14.,  4., 12.,  8., 11.,  8.,  5.,  8.,  8., 12., 13.,  8.,
       11.,  8., 13.,  8.,  4.,  2., 12.,  7.,  0.,  8., 14.,  6.,  7.,
       15., 14.,  0., 13.,  8.,  3.,  1.,  8.,  6.,  9.,  9., 13.,  7.,
       15., 13.,  8.,  4.,  1.,  0., 13., 13.,  8., 15., 12.,  4.,  4.,
        2.,  2.,  6.,  6.,  5.,  0.,  6.,  9.,  9., 11., 11.,  4.,  0.,
        4.,  1.,  8.,  1.,  3.,  2.,  5.,  0., 12.,  1., 15.,  1., 11.,
       13., 13.,  9.,  6., 14., 15., 12.,  7., 13.,  3.,  6.,  0.,  7.,
       10.,  8.,  0., 12.,  1.,  9.,  8., 12., 13.,  7.,  2.,  8., 12.,
       15.,  0., 14.,  4., 15.,  6.,  2.,  3., 10.,  8.,  8.,  9., 11.,
        2.,  8.,  5., 13.,  3.,  5.,  7., 15.,  8.,  5., 15., 12., 10.,
        8.,  7.,  8.,  4.,  5.,  2.,  8.,  7., 12.,  0.,  4., 15.,  2.,
       13.,  9., 15.,  8.,  8.,  4., 13.,  7.,  8., 12.,  0.,  8.,  7.,
        8.,  7.,  6., 10.,  1.,  5.,  4., 12.,  0.,  6., 12.,  4.,  5.,
        4.,  9.,  6.,  3.,  0.,  8.,  2.,  2.,  8., 15.,  9., 14.,  2.,
       12.,  3.,  8.,  4., 10.,  8., 13., 11.,  4.,  5., 12., 13.,  8.,
       11., 11.,  2.,  7.,  8.,  7.,  2.,  1.,  9., 13.,  5.,  7.,  8.,
        4.,  8., 13.,  8.,  5., 10., 14.,  8.,  1., 10., 12.,  8.,  8.,
        6.,  7.,  7., 15.,  7., 13., 11.,  8.,  3.,  9., 11., 12.,  7.,
       10.,  5.,  2.,  1., 12.,  8., 13.,  8., 10.,  0., 10.,  8.,  7.,
        3.,  2.,  3., 15.,  6.,  2., 10.,  8.,  8., 13.,  8., 14.,  8.,
       15., 15., 13.,  9.,  8.,  8.,  9.,  9., 10., 13.,  5.,  8.,  8.,
        1.,  8.,  5., 14.,  5.,  5., 10.,  5.,  4.,  2.,  6., 14., 12.,
        9.,  2.,  2., 15.,  7.,  7.,  4.,  5.,  2.,  9.,  6., 15.,  5.,
        7.,  8.,  8.,  1., 15.,  5.,  6.,  1., 14., 12., 10., 12., 11.,
       10., 13., 10.,  1.,  8.,  8., 11.,  5.,  7.,  8.,  2., 13.,  4.,
        5.,  7.,  5.,  0.,  2.,  5., 10., 15.,  8.,  2.,  7.,  8.,  6.,
       13., 12.,  8., 10.,  8.,  0.,  2., 14.,  1.,  8.,  2., 10.,  4.,
       14.,  7.,  2., 12.,  8., 13.,  1.,  8.,  8.,  7.,  1.,  6., 10.,
        8.,  8.,  7.,  4., 14., 12.,  9.,  8., 11.,  8.,  4., 11.,  6.,
        4.,  5.,  2.,  1.,  2.,  8.,  4.,  8., 14.,  7.,  5.,  6.,  5.,
        7., 12., 10.,  8., 15.,  8., 14., 12.,  8.,  6., 13.,  8., 14.,
        8.,  2.,  8.,  5., 10.,  8.,  9., 12.,  5.,  8., 13.,  3., 10.,
        7.,  1., 13., 15., 15.,  4., 12., 10., 12.,  7., 13., 15.,  8.,
        2.,  5.,  7., 10., 13.,  6., 14., 11.,  8.,  8.,  8., 11., 10.,
       10.,  9.,  4., 13.,  5., 14.,  1.,  0.,  6.,  7.,  8.,  1.,  1.,
       13.,  4., 13.,  4., 12.,  4., 10.,  6.,  7.,  5.,  0.,  2.,  5.,
       10.,  5.,  0.,  2.,  1., 14., 10., 12., 10.,  3.,  8.,  6.,  8.,
        6., 14.,  6.,  2.,  6.,  0.,  8.,  2., 12.,  4.,  6.,  6.,  7.,
        8.,  6.,  0.,  8., 13.,  3.,  8.,  7., 11.,  8.,  9.,  6.,  6.,
        2., 12., 12., 12., 13., 13., 13.,  6.,  1.,  1.,  5., 11.,  5.,
```

```
 11.,  8., 12.,  2.,  4.,  8.,  9.,  4.,  9.,  0.,  2.,  4.,  7.,
  9.,  8.,  8., 10.,  8., 11., 15., 12.,  2.,  8.,  5.,  8.,  5.,
  6.,  8., 15., 15.,  2.,  5.,  0., 12.,  8.,  7.,  2., 15., 12.,
  8.,  5.,  6., 11., 10., 10.,  4., 12., 15., 12.,  7., 13., 13.,
 13.,  2., 12.,  8.,  4., 15.,  9.,  3.,  8.,  7.,  3.,  0.,  4.,
  0.,  5.,  8., 12.,  0.,  2., 15.,  5.,  4.,  3.,  5.,  5.,  8.,
 13.,  1., 12., 10.,  5.,  6., 12.,  2.,  8.,  8., 12.,  5.,  3.,
  6.,  1., 14.,  8.,  8., 13.,  8.,  5., 10.,  8.,  1., 15., 15.,
 12., 15.,  5., 13.,  5.,  6.,  3., 13.,  8., 15., 13., 15., 13.,
 12.,  5.,  2.,  9.,  8.,  7., 15.,  5.,  6., 13.,  8.,  1.,  7.,
  9.,  1.,  4., 10.,  4.,  8.,  9.,  8.,  5.,  5., 13.,  8.,  7.,
  4.,  5., 12.,  8., 12.,  6., 15.,  0.,  6.,  8.,  7.,  5., 14.,
 12.,  6.,  0., 12.,  8., 13.,  3.,  1., 11.,  8., 11.,  8.,  8.,
 13.,  5., 15., 10.,  2.,  8.,  2.,  2.,  4.,  8.,  8.,  8.,  4.,
  8., 13., 12.,  1., 10.,  0.,  4.,  4.,  3.,  8.,  5.,  1.,  2.,
 12., 11.,  8.,  5., 13., 15.,  5., 14.,  8.,  6.,  8.,  9., 13.,
  6., 12., 14.,  2.,  8.,  0.,  1.,  1.,  8., 11., 13., 13.,  9.,
 13.,  4.,  4.,  1.,  1., 14.,  3.,  9.,  7., 12.,  8.,  2., 13.,
  8.,  7.,  8.,  2., 12.,  8.,  2.,  8.,  7., 12.,  8.,  2., 13.,
  8.,  5., 13., 11.,  8.,  4., 11., 15., 13.,  8.,  7., 14., 13.,
 15.,  7.,  8.,  7.,  6.,  7.,  0., 13., 15., 14., 11., 15.,  8.,
  8.,  2., 15., 13.,  4.,  0., 15., 15., 14., 12., 11., 10., 12.,
  8., 10.,  5.,  8.,  9.,  9.,  0., 14., 13.,  8.,  4.,  3.,  2.,
  6., 13.,  9.,  8.,  5.,  8.,  1.,  0., 14.,  4., 15.,  0.,  5.,
  7.,  4., 12.,  4.,  0.,  2., 11., 13., 10., 13.,  6.,  8.,  7.,
  0., 13.,  0., 12.,  3.,  4.,  8., 12.,  4., 13., 14.,  1.,  8.,
  4.,  5.,  8.,  8.,  4.,  2.,  8.,  1.,  3., 15.,  0., 13.,  8.,
 13.,  7.,  8.,  2.,  8., 15., 15., 13.,  9.,  8., 15., 13.,  8.,
  4.,  9.,  7.,  2.,  2., 13.,  8.,  4., 15.,  8.,  8.,  7., 14.,
 13.,  9., 13., 10., 15.,  8.,  0.,  1., 13.,  2.,  4.,  4., 15.,
  2.,  6., 15.,  6.,  8., 15.,  4.,  3.,  8.,  8., 11.,  4., 12.,
  8.,  2., 15.,  6.,  4., 15.,  4.,  2.,  8.,  8., 13., 15., 15.,
  8., 13.,  8., 12.,  8., 13.,  2.,  5., 11.,  0.,  6.,  7.])
```

---

functions for presenting the results

In [ ]:
```
def function_result_01():

    print("final loss (K=2) = {:13.10f}".format(loss_iteration_02[-1]))
```

In [ ]:
```
def function_result_02():

    print("final loss (K=4) = {:13.10f}".format(loss_iteration_04[-1]))
```

In [ ]:
```
def function_result_03():

    print("final loss (K=8) = {:13.10f}".format(loss_iteration_08[-1]))
```

In [ ]:
```
def function_result_04():

    print("final loss (K=16) = {:13.10f}".format(loss_iteration_16[-1]))
```

In [ ]:
```
def function_result_05():
```

```
    plt.figure(figsize=(8,6))
    plt.title('loss (K=2)')

    plt.plot(loss_iteration_02, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_06():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=4)')

    plt.plot(loss_iteration_04, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

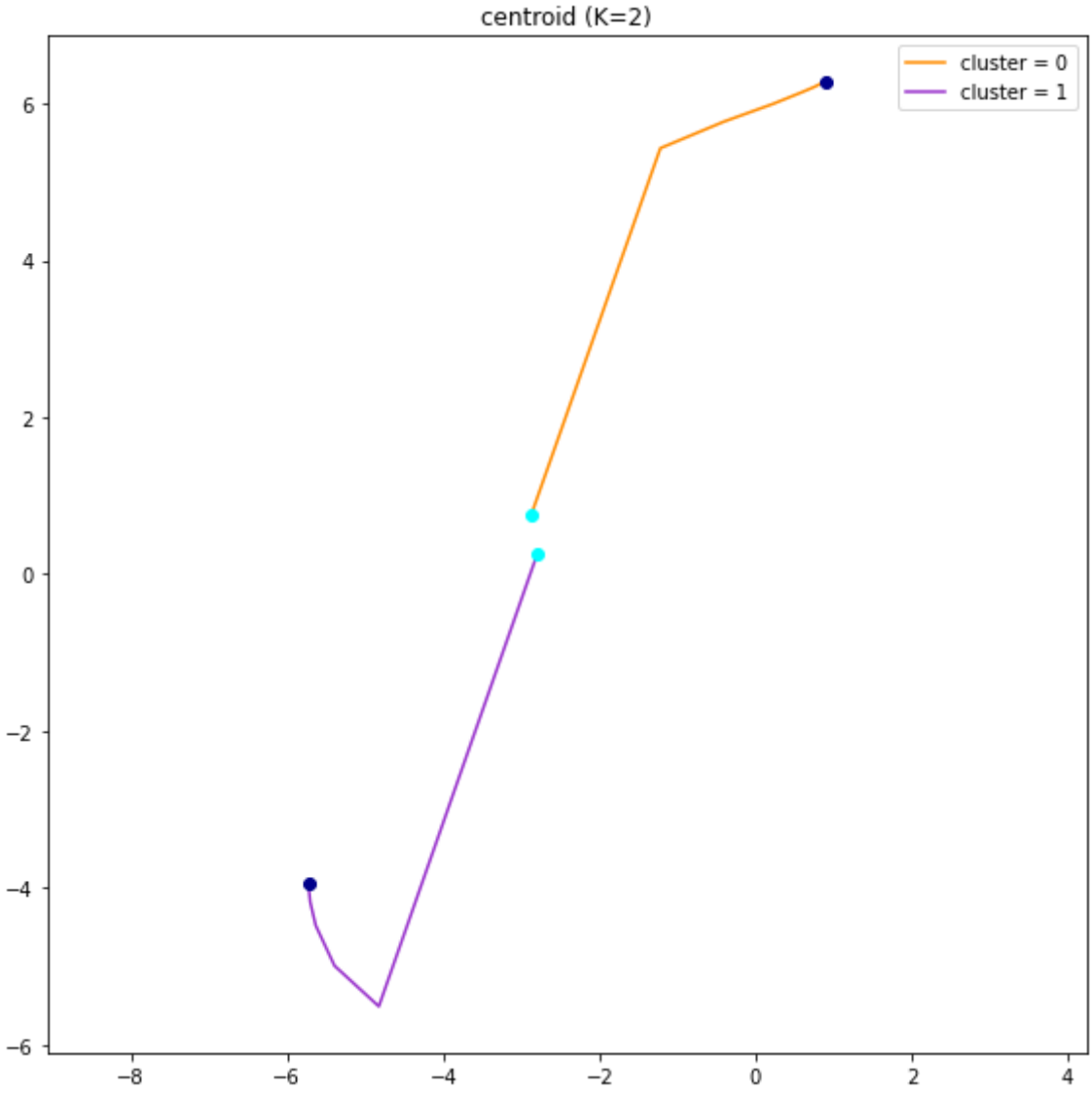
In [ ]:
```
def function_result_07():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=8)')

    plt.plot(loss_iteration_08, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_08():

    plt.figure(figsize=(8,6))
    plt.title('loss (K=16)')

    plt.plot(loss_iteration_16, '-', color='red')
    plt.xlabel('iteration')
    plt.ylabel('loss')

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def function_result_09():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=2)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    # cluster_0_center_x = centroid_iteration_02[:,0,0]
    # cluster_0_center_y = centroid_iteration_02[:,0,1]

    # cluster_1_center_x = centroid_iteration_02[:,1,0]
    # cluster_1_center_y = centroid_iteration_02[:,1,1]

    # cluster_0_center_x_initial = centroid_iteration_02[0,0,0]
    # cluster_0_center_y_initial = centroid_iteration_02[0,0,1]
```

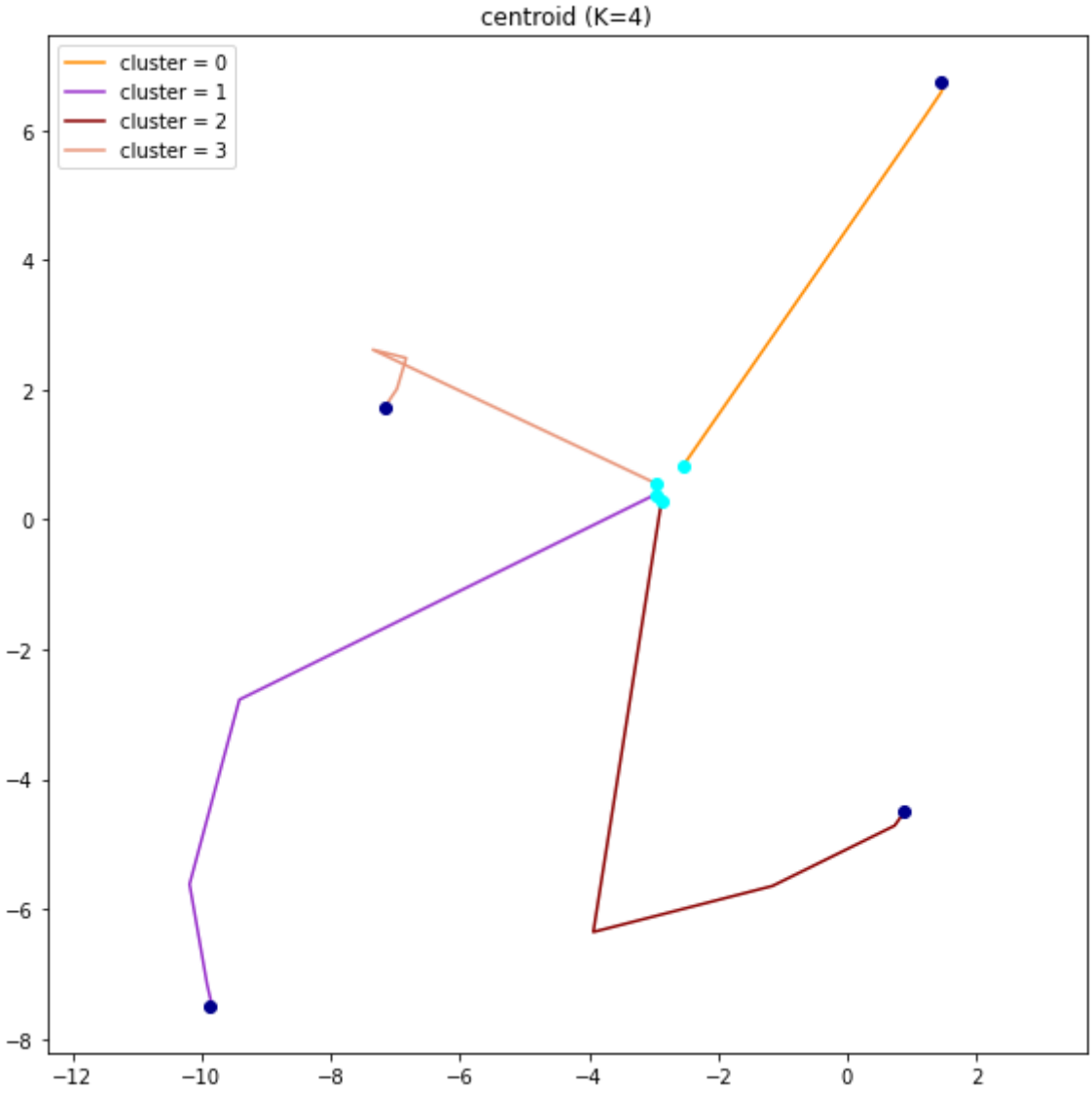
```
    # cluster_0_center_x_final = centroid_iteration_02[-1,0,0]
    # cluster_0_center_y_final = centroid_iteration_02[-1,0,1]

    # cluster_1_center_x_initial = centroid_iteration_02[0,1,0]
    # cluster_1_center_y_initial = centroid_iteration_02[0,1,1]

    # cluster_1_center_x_final = centroid_iteration_02[-1,1,0]
    # cluster_1_center_y_final = centroid_iteration_02[-1,1,1]



    # plt.plot(cluster_0_center_x, cluster_0_center_y, '-', color='blue', label='clust
    # plt.plot(cluster_1_center_x, cluster_1_center_y, '-', color='yellow', label='clu
    # plt.plot(cluster_0_center_x_initial,cluster_0_center_y_initial,'o', color='mager
    # plt.plot(cluster_0_center_x_final,cluster_0_center_y_final,'o', color='red', lab
    # plt.plot(cluster_1_center_x_initial,cluster_1_center_y_initial,'o', color='mager
    # plt.plot(cluster_1_center_x_final,cluster_1_center_y_final,'o', color='red')

    # plt.plot(cluster_0_center_x, cluster_0_center_y, '-', color='blue', label='clust
    # plt.plot(cluster_1_center_x, cluster_1_center_y, '-', color='yellow', label='clu
    # plt.plot(cluster_0_center_x_initial,cluster_0_center_y_initial,'o', color='mager
    # plt.plot(cluster_0_center_x_final,cluster_0_center_y_final,'o', color='red', lab
    # plt.plot(cluster_1_center_x_initial,cluster_1_center_y_initial,'o', color='mager
    # plt.plot(cluster_1_center_x_final,cluster_1_center_y_final,'o', color='red')



    for i in range(2):
        plt.plot(centroid_iteration_02[:,i,0], centroid_iteration_02[:,i,1], '-', colo
        plt.plot(centroid_iteration_02[0,i,0],centroid_iteration_02[0,i,1],'o', color=
        plt.plot(centroid_iteration_02[-1,i,0],centroid_iteration_02[-1,i,1],'o', colo
    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
function_result_09()
```



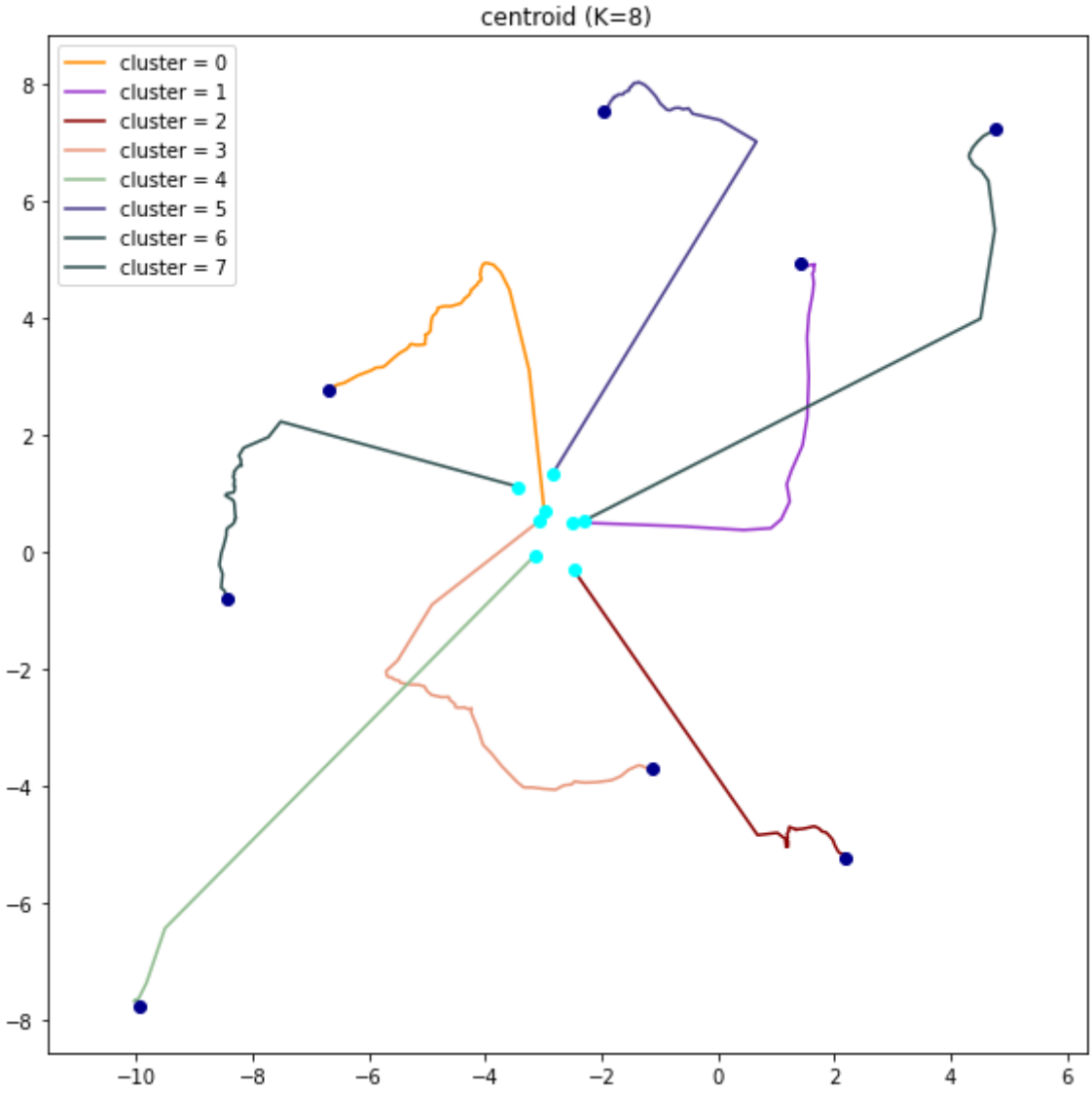
In [ ]:
```
def function_result_10():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=4)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(4):
        plt.plot(centroid_iteration_04[:,i,0], centroid_iteration_04[:,i,1], '-', colo
        plt.plot(centroid_iteration_04[0,i,0],centroid_iteration_04[0,i,1],'o', color=
        plt.plot(centroid_iteration_04[-1,i,0],centroid_iteration_04[-1,i,1],'o', colo
    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
function_result_10()
```



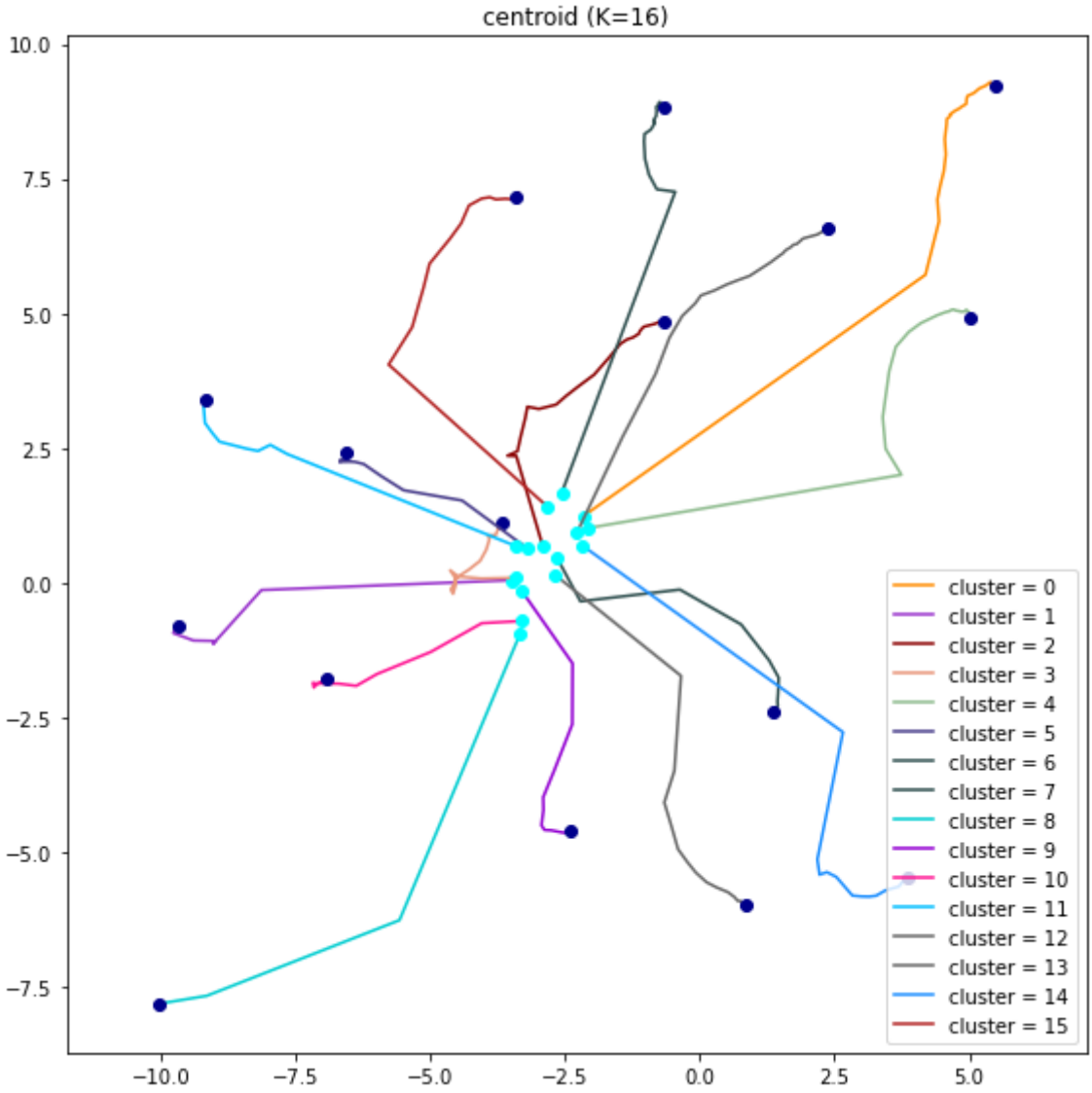
In [ ]:
```
def function_result_11():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=8)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(8):
        plt.plot(centroid_iteration_08[:,i,0], centroid_iteration_08[:,i,1], '-', colo
        plt.plot(centroid_iteration_08[0,i,0],centroid_iteration_08[0,i,1],'o', color=
        plt.plot(centroid_iteration_08[-1,i,0],centroid_iteration_08[-1,i,1],'o', colo

    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
function_result_11()
```



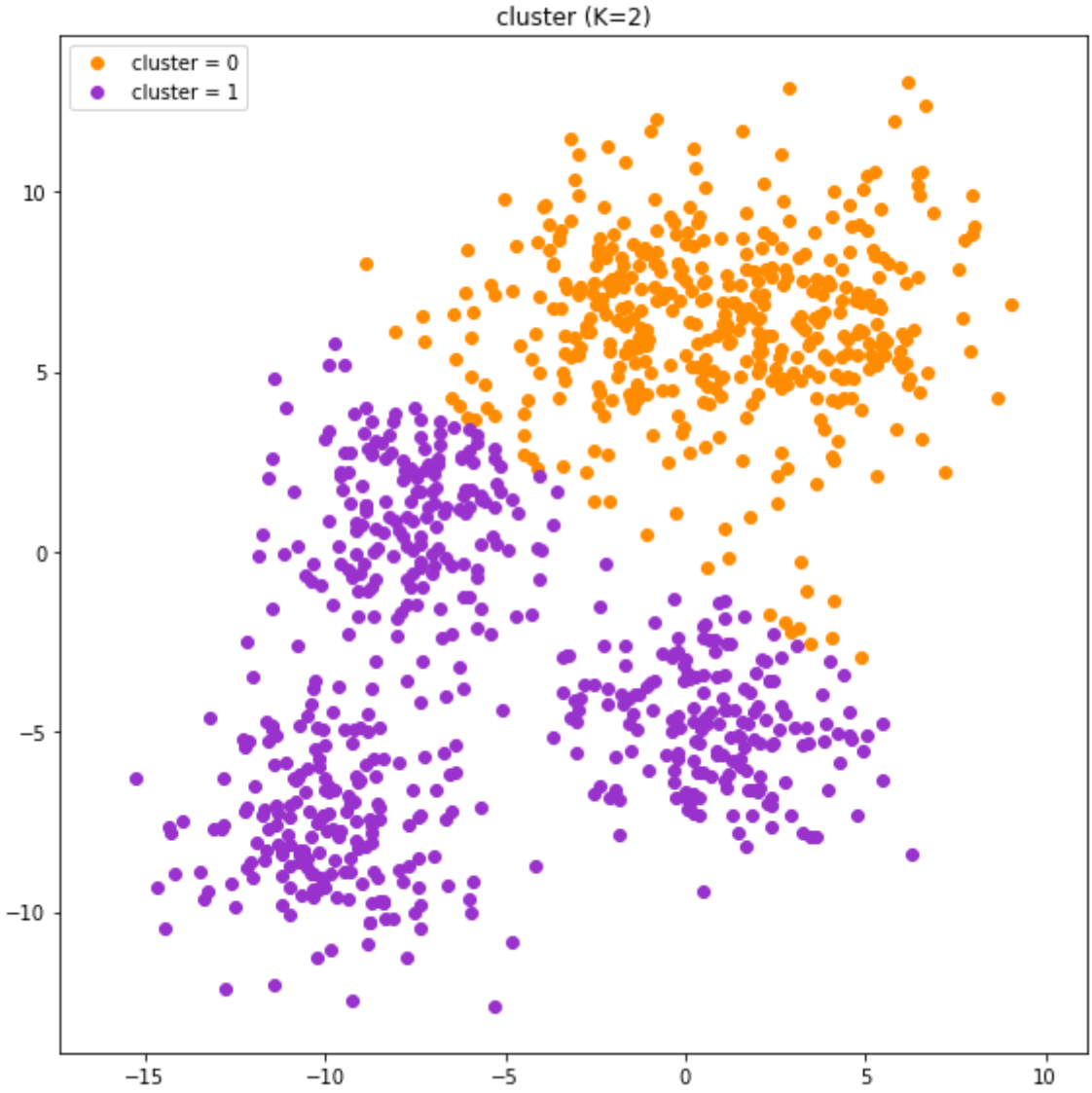
In [ ]:
```
def function_result_12():

    plt.figure(figsize=(8,8))
    plt.title('centroid (K=16)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(16):
        plt.plot(centroid_iteration_16[:,i,0], centroid_iteration_16[:,i,1], '-', colo
        plt.plot(centroid_iteration_16[0,i,0],centroid_iteration_16[0,i,1],'o', color=
        plt.plot(centroid_iteration_16[-1,i,0],centroid_iteration_16[-1,i,1],'o', colo

    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
function_result_12()
```



In [ ]:
```
def function_result_13():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=2)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(2):
        plt.plot(x[label_feature_02==i],y[label_feature_02==i],'o',color=colornames[i
    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
function_result_13()
```



In [ ]:
```
def function_result_14():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=4)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(4):
        plt.plot(x[label_feature_04==i],y[label_feature_04==i],'o',color=colornames[i
    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
def function_result_15():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=8)')

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(8):
```

```
        plt.plot(x[label_feature_08==i],y[label_feature_08==i],'o',color=colornames[i
    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()


    #
    # +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:
```
def function_result_16():

    plt.figure(figsize=(8,8))
    plt.title('cluster (K=16)')

    # +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    for i in range(16):
        plt.plot(x[label_feature_16==i],y[label_feature_16==i],'o',color=colornames[i
    plt.axis('equal')
    plt.legend()
    plt.tight_layout()
    plt.show()


    #
    # +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

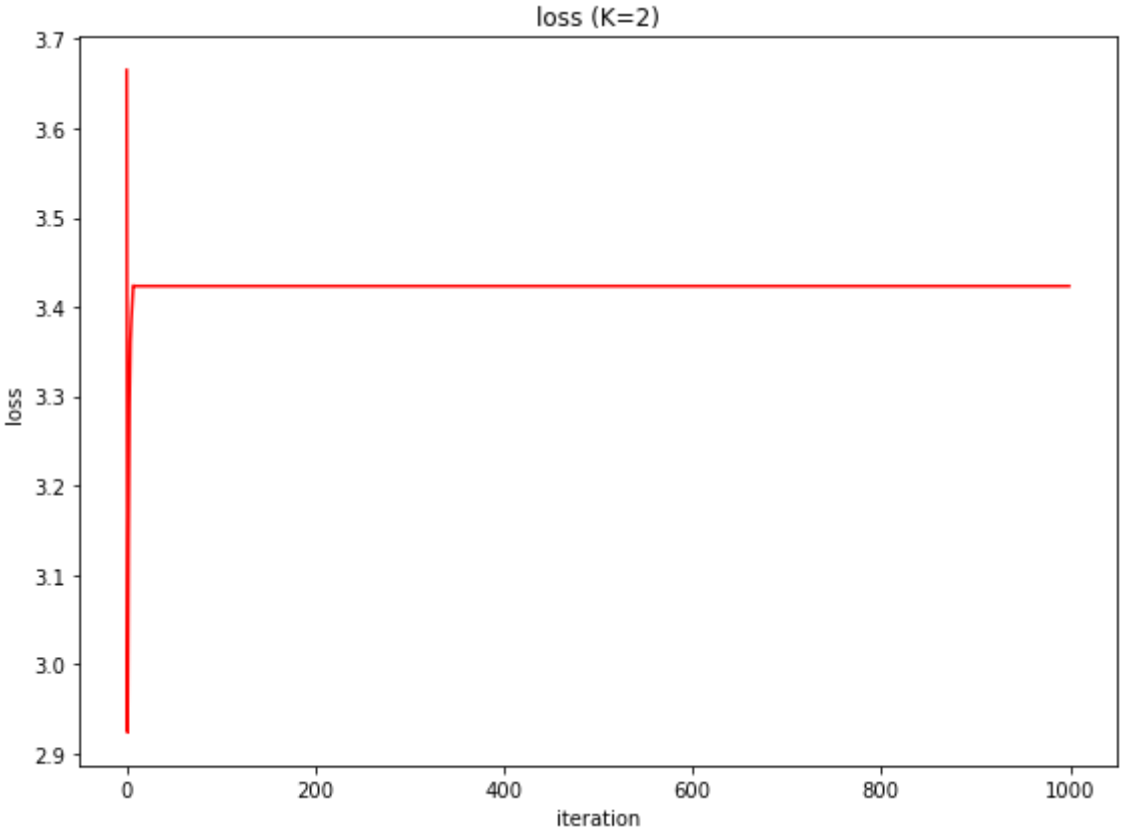
---

results

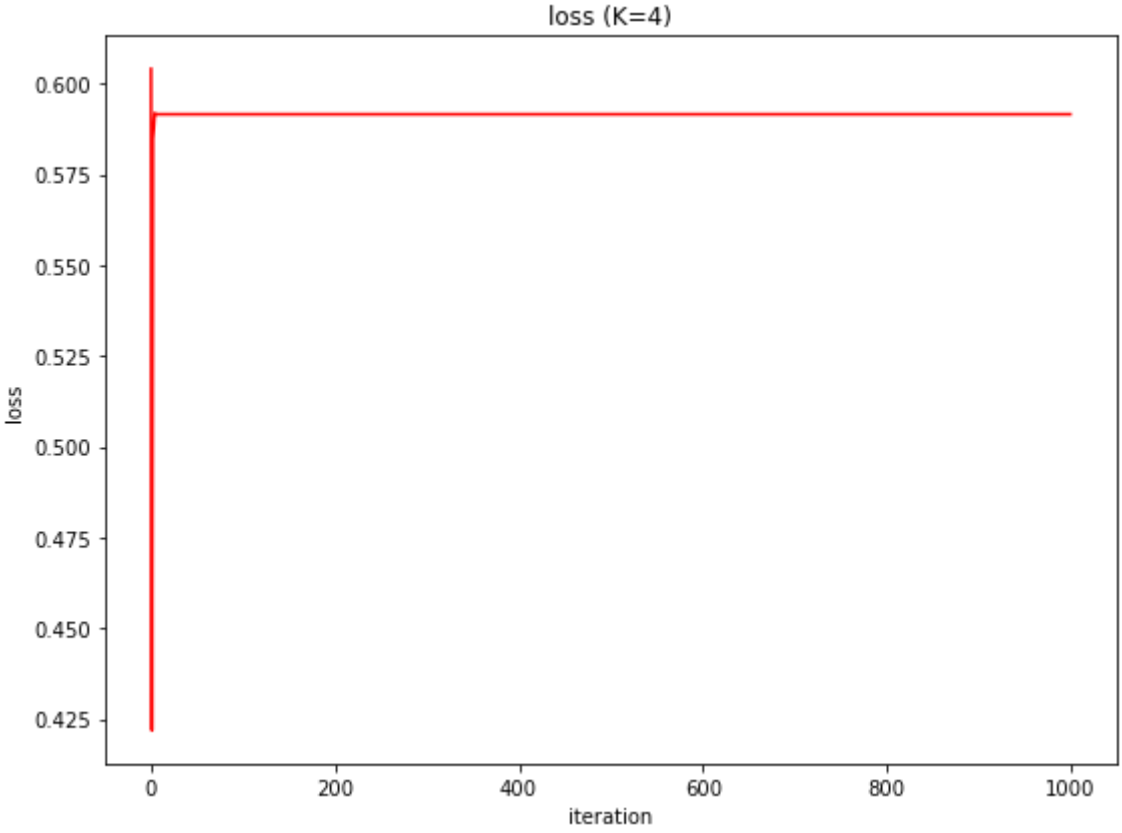
In [ ]:
```
number_result = 16

for i in range(number_result):
    title = '## [RESULT {:02d}]'.format(i+1)
    name_function = 'function_result_{:02d}()'.format(i+1)

    print('************************************************')
    print(title)
    print('************************************************')
    eval(name_function)
```

```
************************************************
## [RESULT 01]
************************************************
final loss (K=2) =  3.4233454695
************************************************
## [RESULT 02]
************************************************
final loss (K=4) =  0.5916203934
************************************************
## [RESULT 03]
************************************************
final loss (K=8) =  0.2972164116
************************************************
## [RESULT 04]
************************************************
final loss (K=16) =  0.1011249390
```

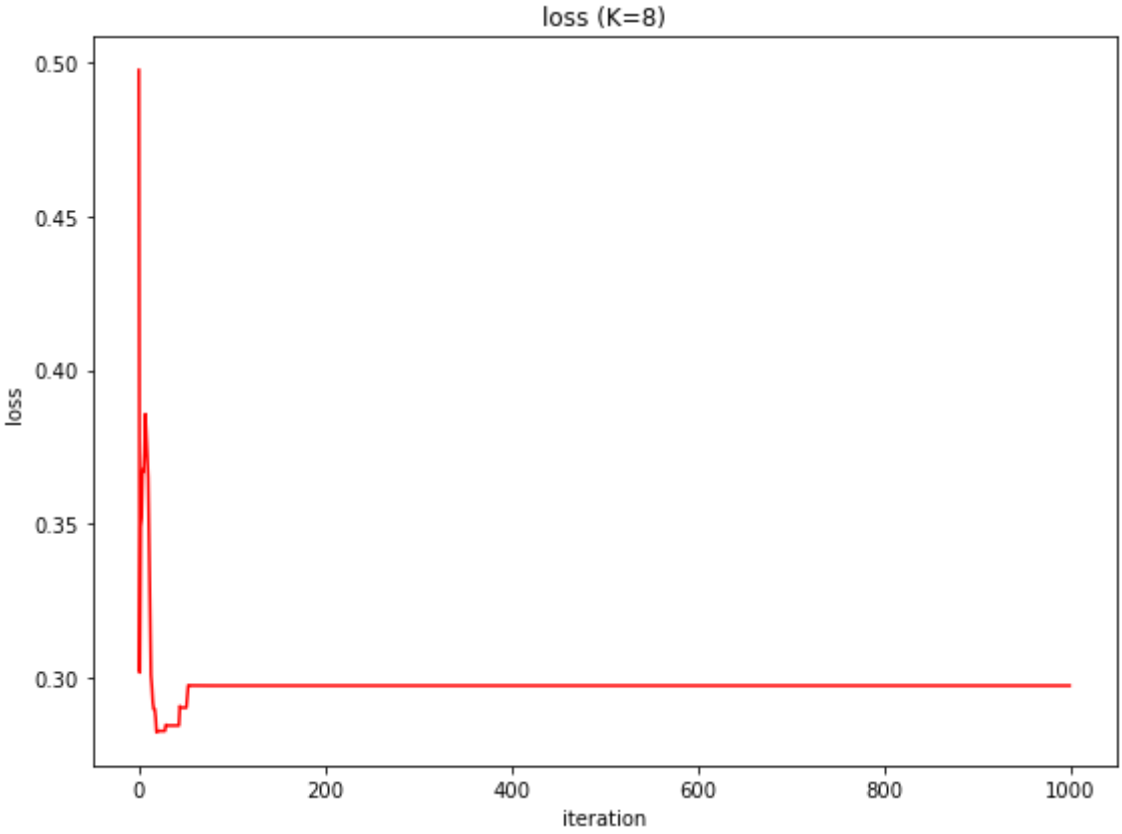
****
[RESULT 05]
****



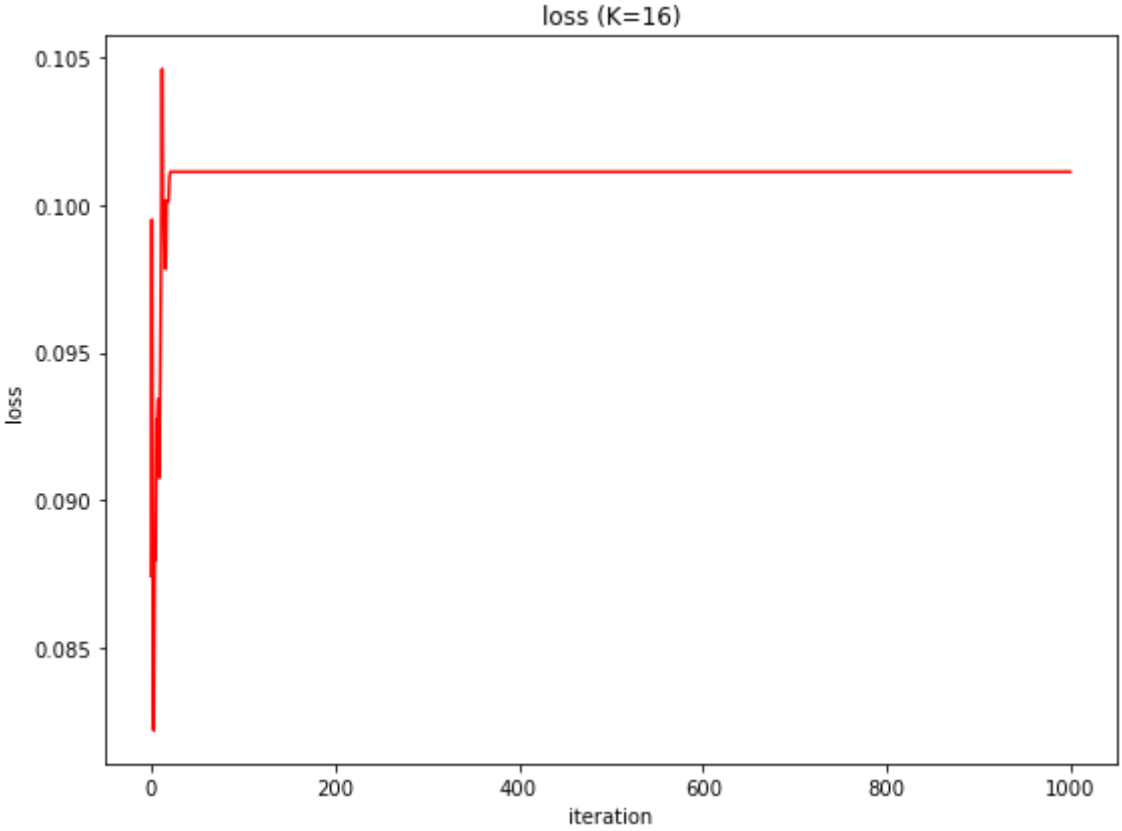
****
[RESULT 06]
****



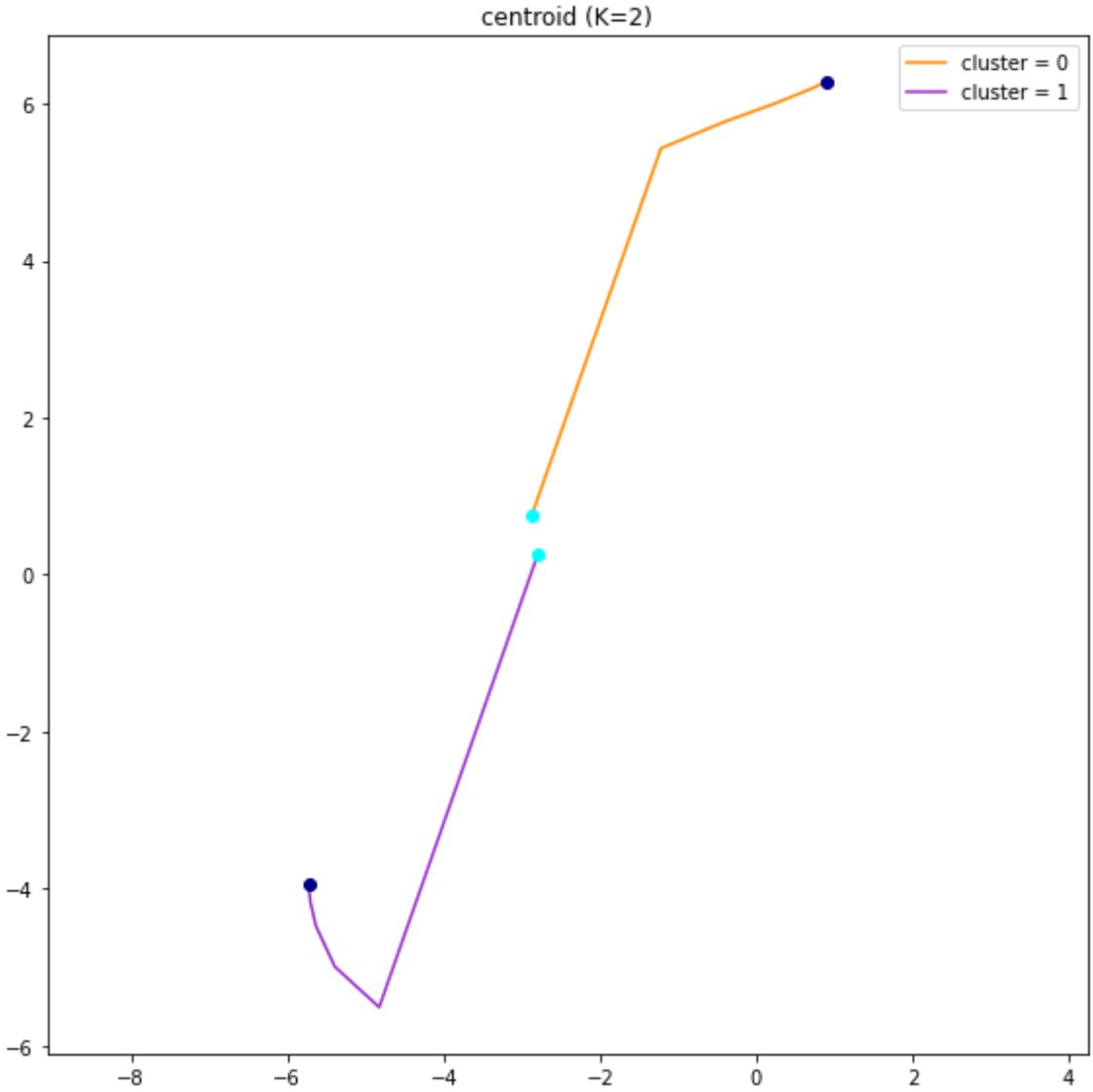
****
[RESULT 07]
****



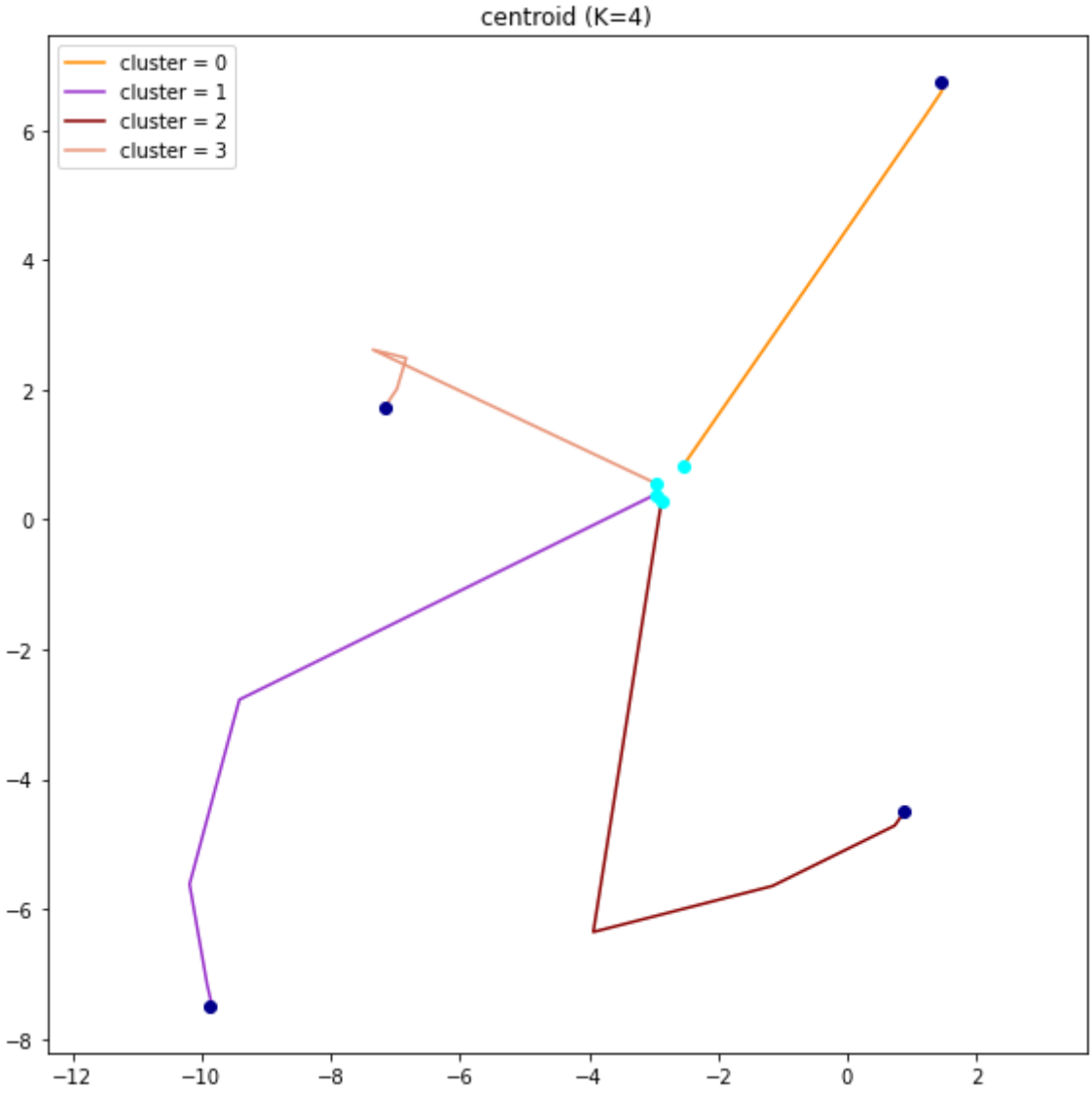
****
[RESULT 08]
****



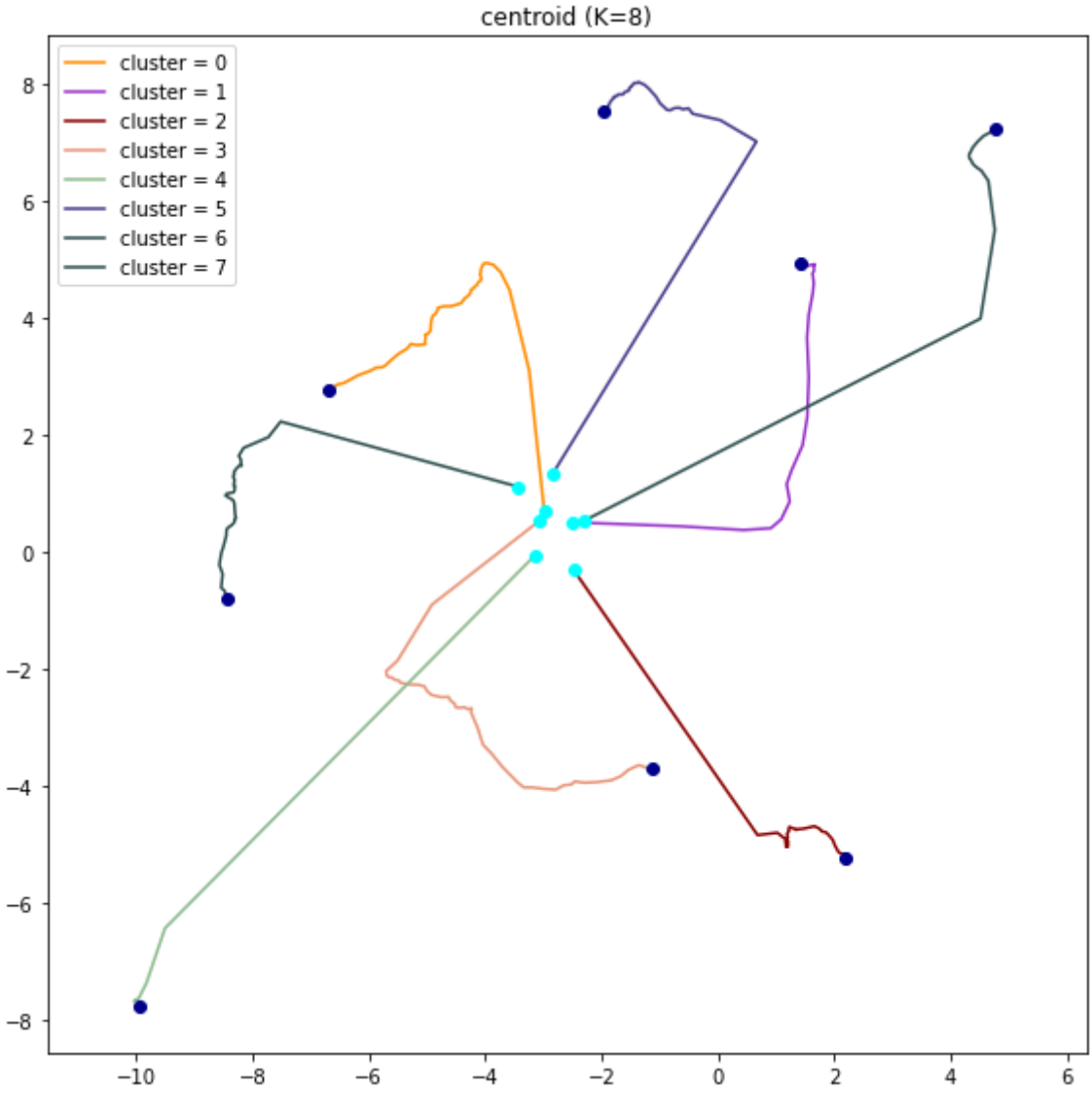
****
[RESULT 09]
****



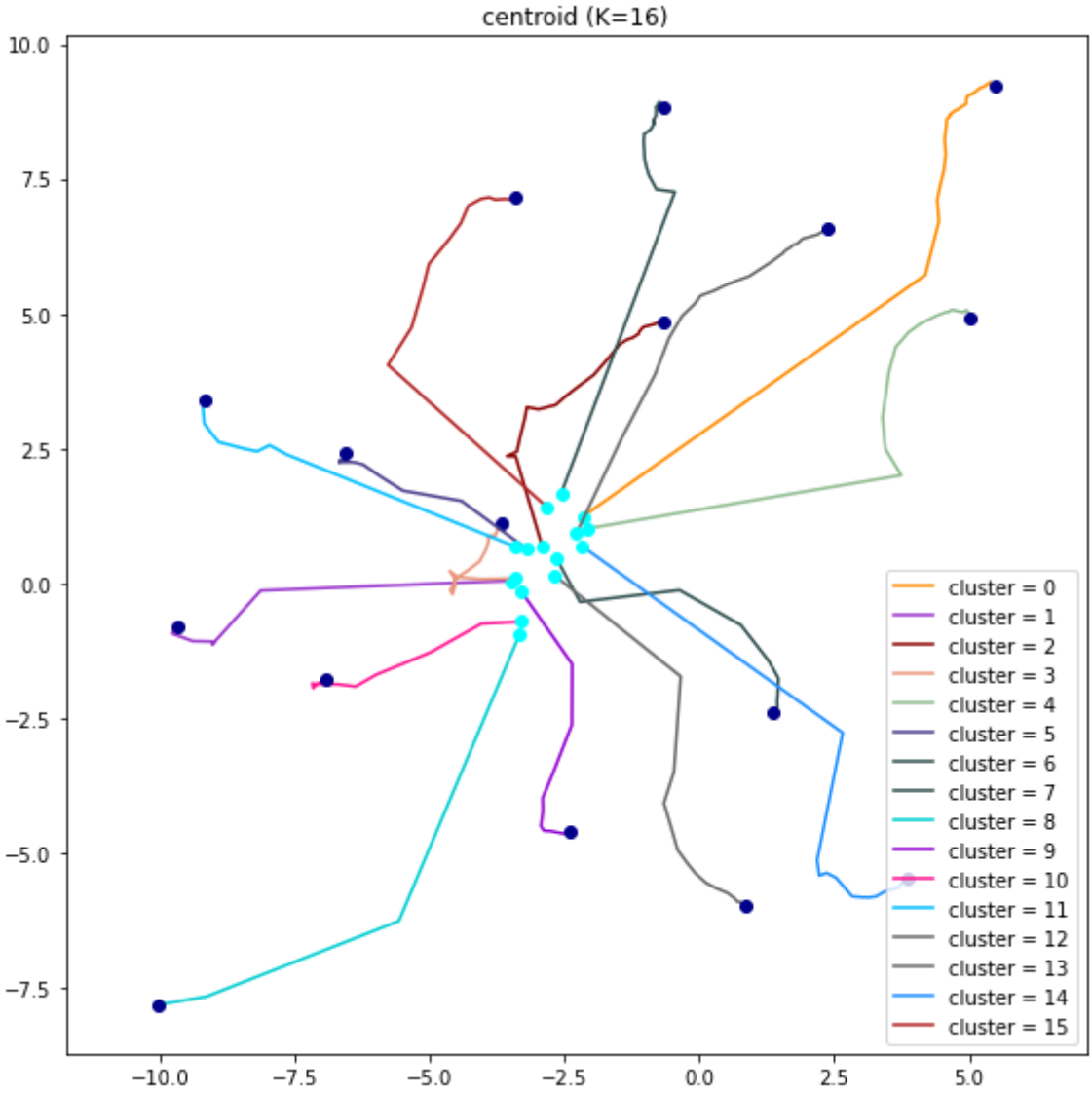

```
**************************************************
## [RESULT 10]
**************************************************
```


centroid (K=4)
cluster = 0
cluster = 1
cluster = 2
cluster = 3
6
4
2
0
−2
−4
−6
−8
−12
−10
−8
−6
−4
−2
0
2

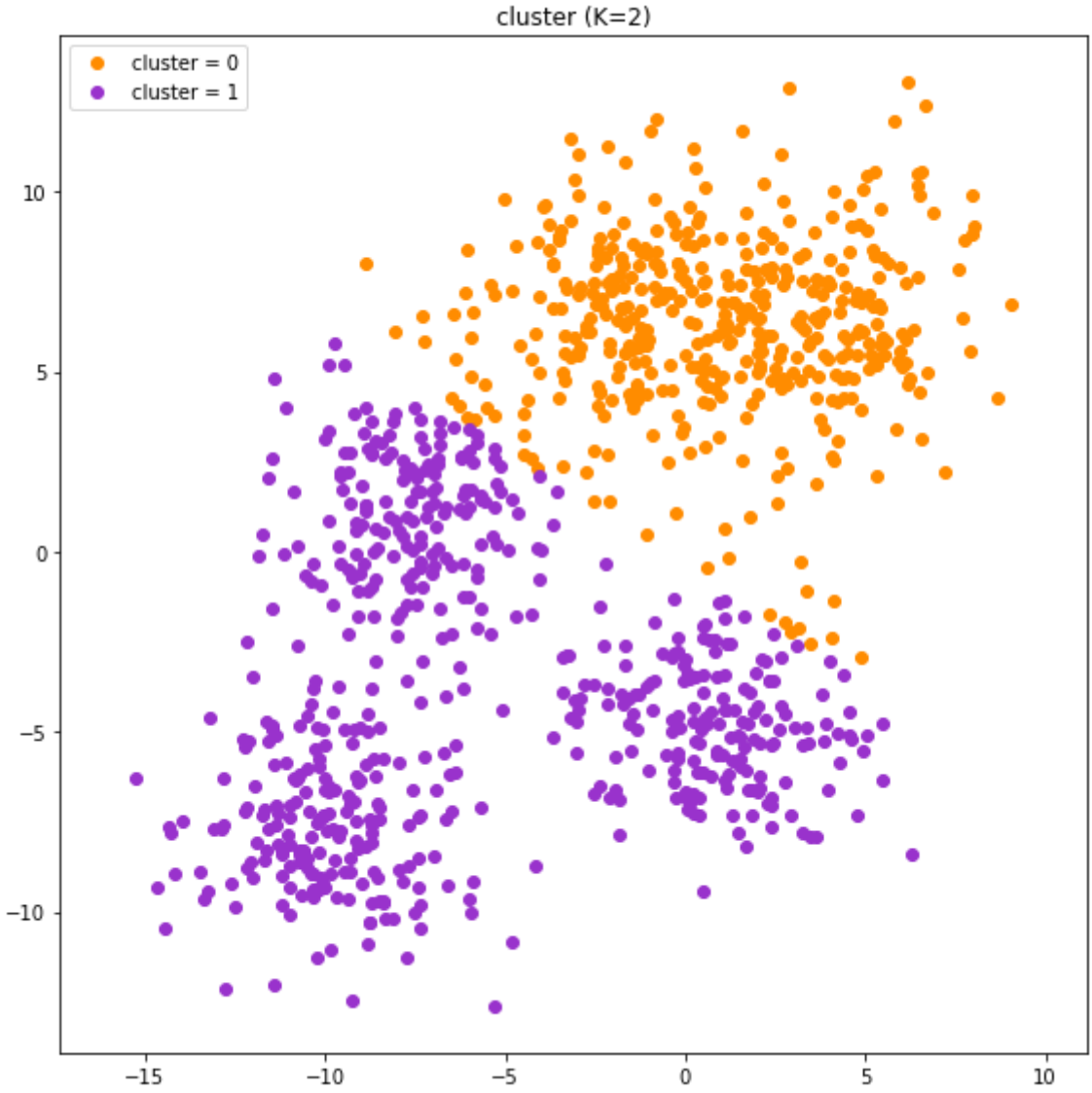
```
**************************************************
## [RESULT 11]
**************************************************
```


centroid (K=8)
cluster = 0
cluster = 1
cluster = 2
cluster = 3
cluster = 4
cluster = 5
cluster = 6
cluster = 7
8
6
4
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
4
6

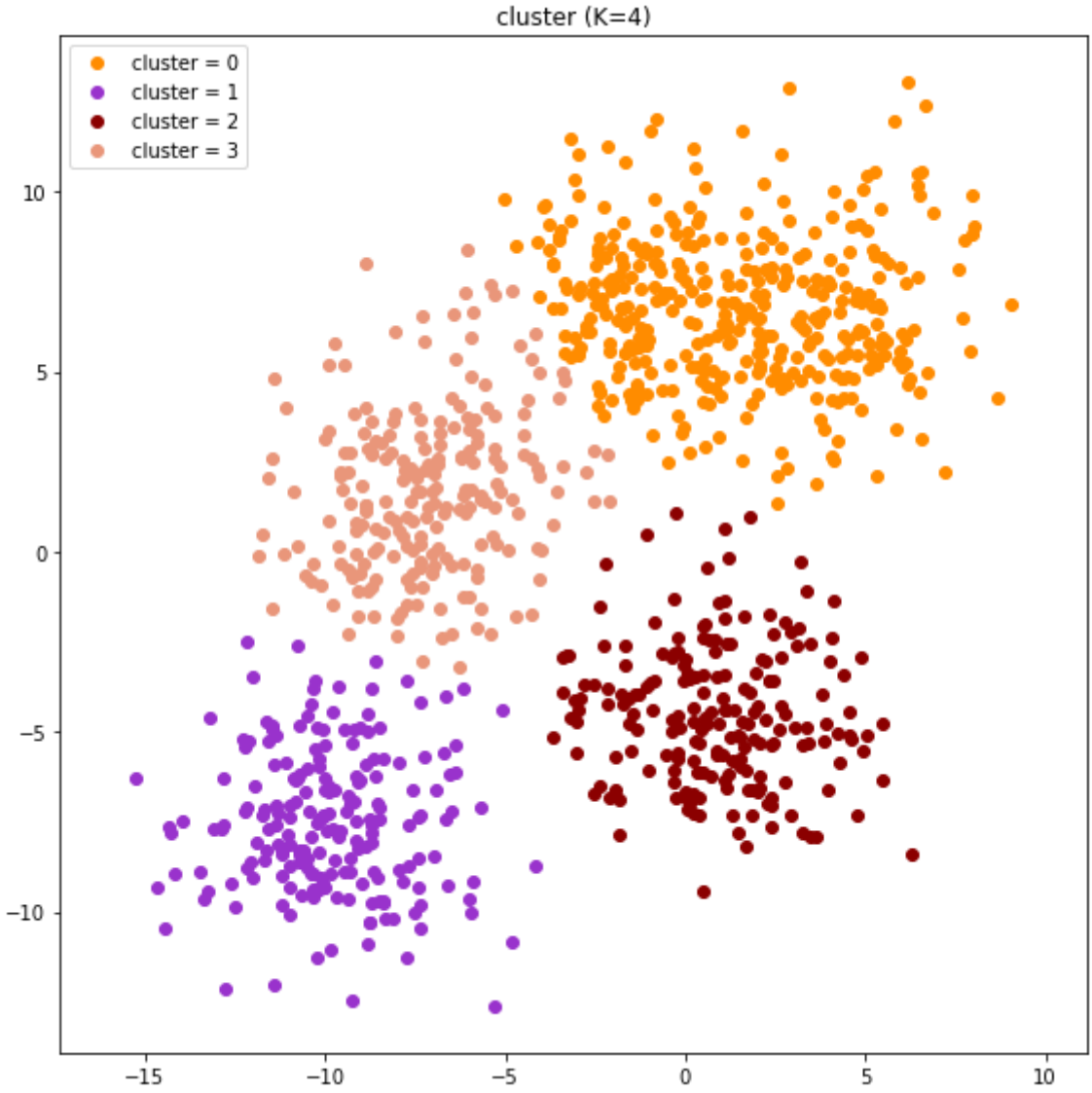
****
[RESULT 12]
****


centroid (K=16)
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
cluster = 0
cluster = 1
cluster = 2
cluster = 3
cluster = 4
cluster = 5
cluster = 6
cluster = 7
cluster = 8
cluster = 9
cluster = 10
cluster = 11
cluster = 12
cluster = 13
cluster = 14
cluster = 15

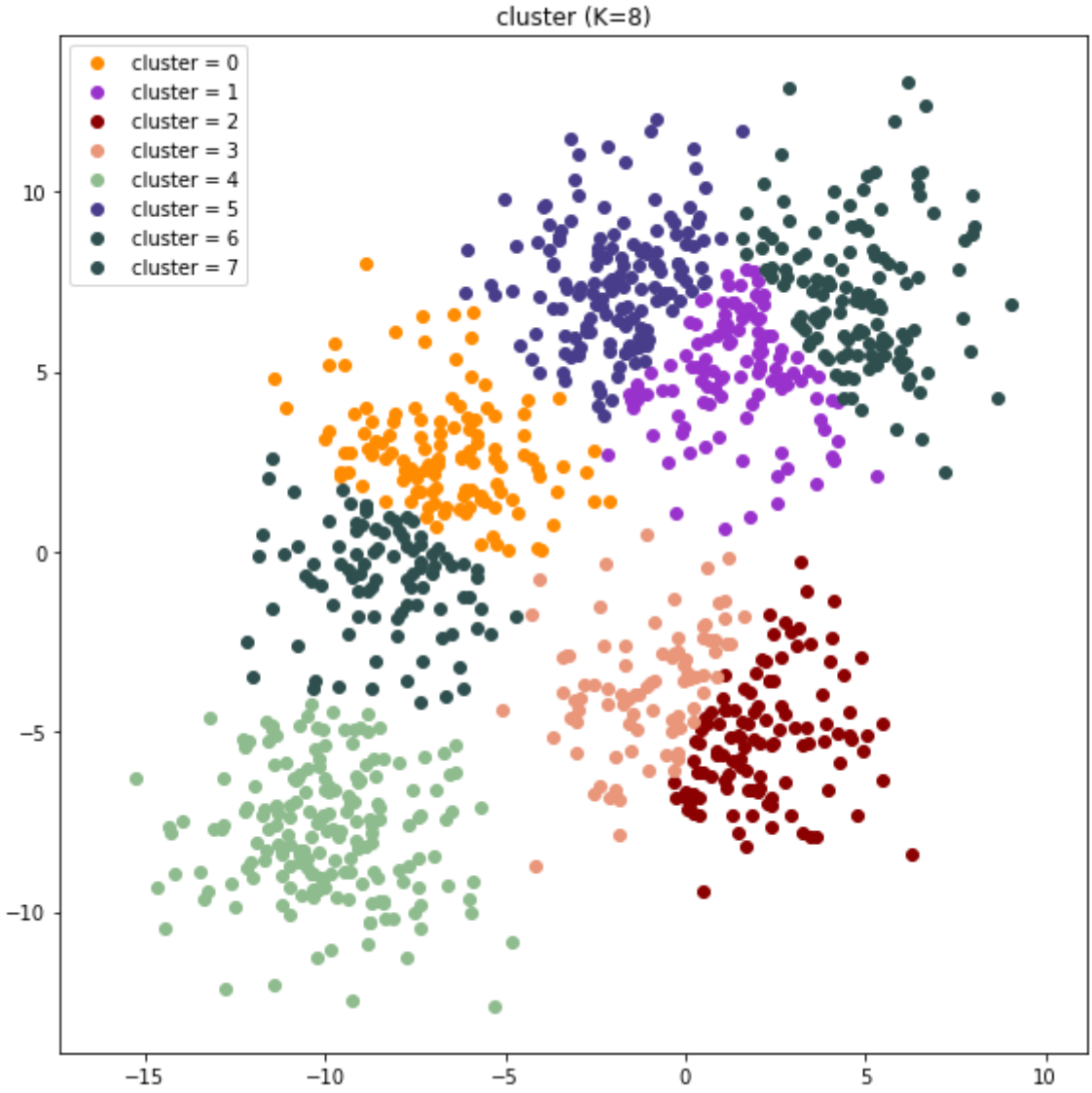
****
[RESULT 13]
****



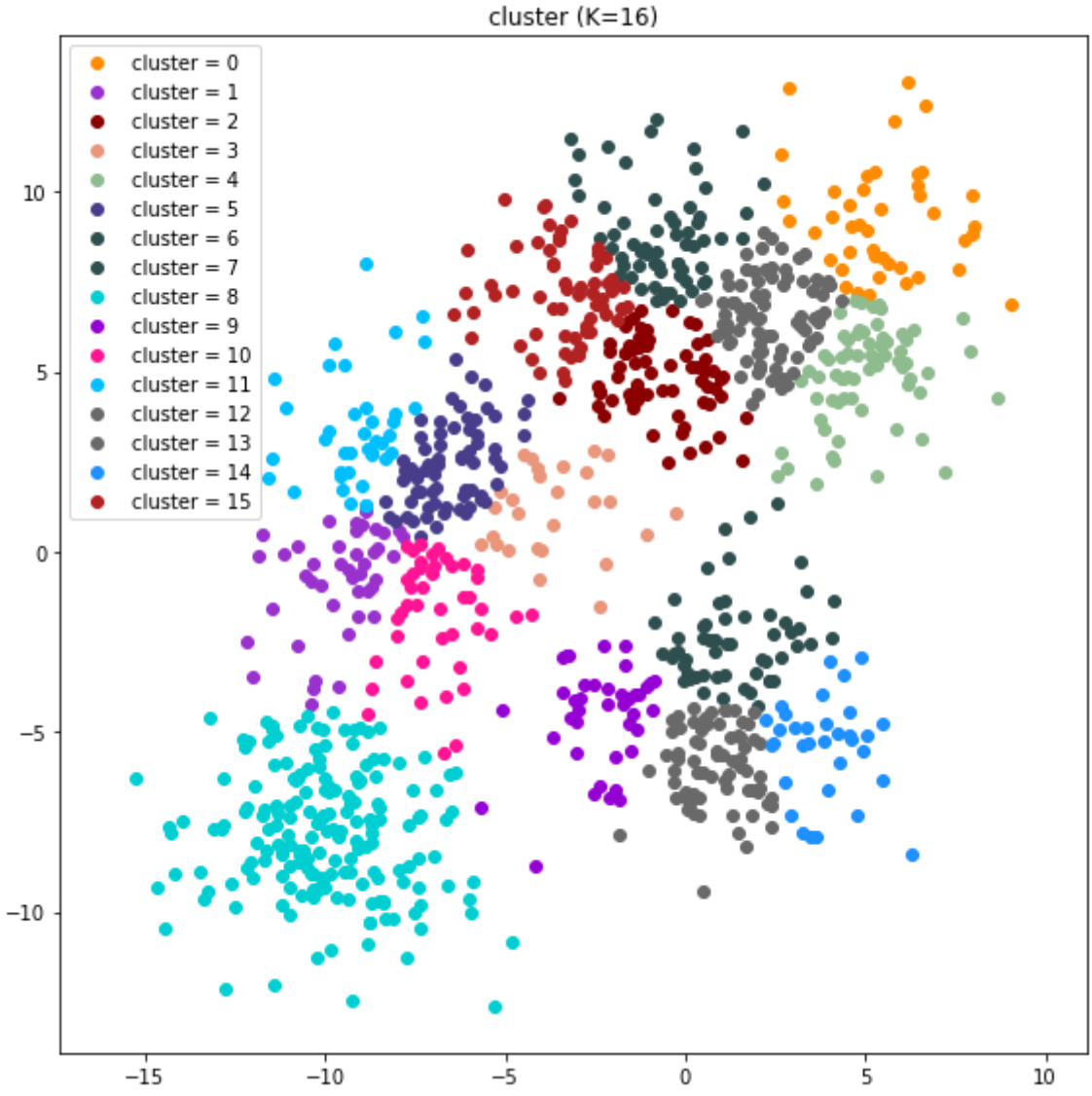

```
**************************************************
## [RESULT 14]
**************************************************
```


cluster (K=4)
cluster = 0
cluster = 1
cluster = 2
cluster = 3
10
5
0
−5
−10
−15
−10
−5
0
5
10

```
**************************************************
## [RESULT 15]
**************************************************
```


cluster (K=8)
cluster = 0
cluster = 1
cluster = 2
cluster = 3
cluster = 4
cluster = 5
cluster = 6
cluster = 7
10
5
0
−5
−10
−15
−10
−5
0
5
10

```
**************************************************
## [RESULT 16]
**************************************************
```


cluster (K=16)
cluster = 0
cluster = 1
cluster = 2
cluster = 3
cluster = 4
cluster = 5
cluster = 6
cluster = 7
cluster = 8
cluster = 9
cluster = 10
cluster = 11
cluster = 12
cluster = 13
cluster = 14
cluster = 15
10
5
0
−5
−10
−15
−10
−5
0
5
10

In [ ]: