

# 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',  
'dimgrey',  
'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]])

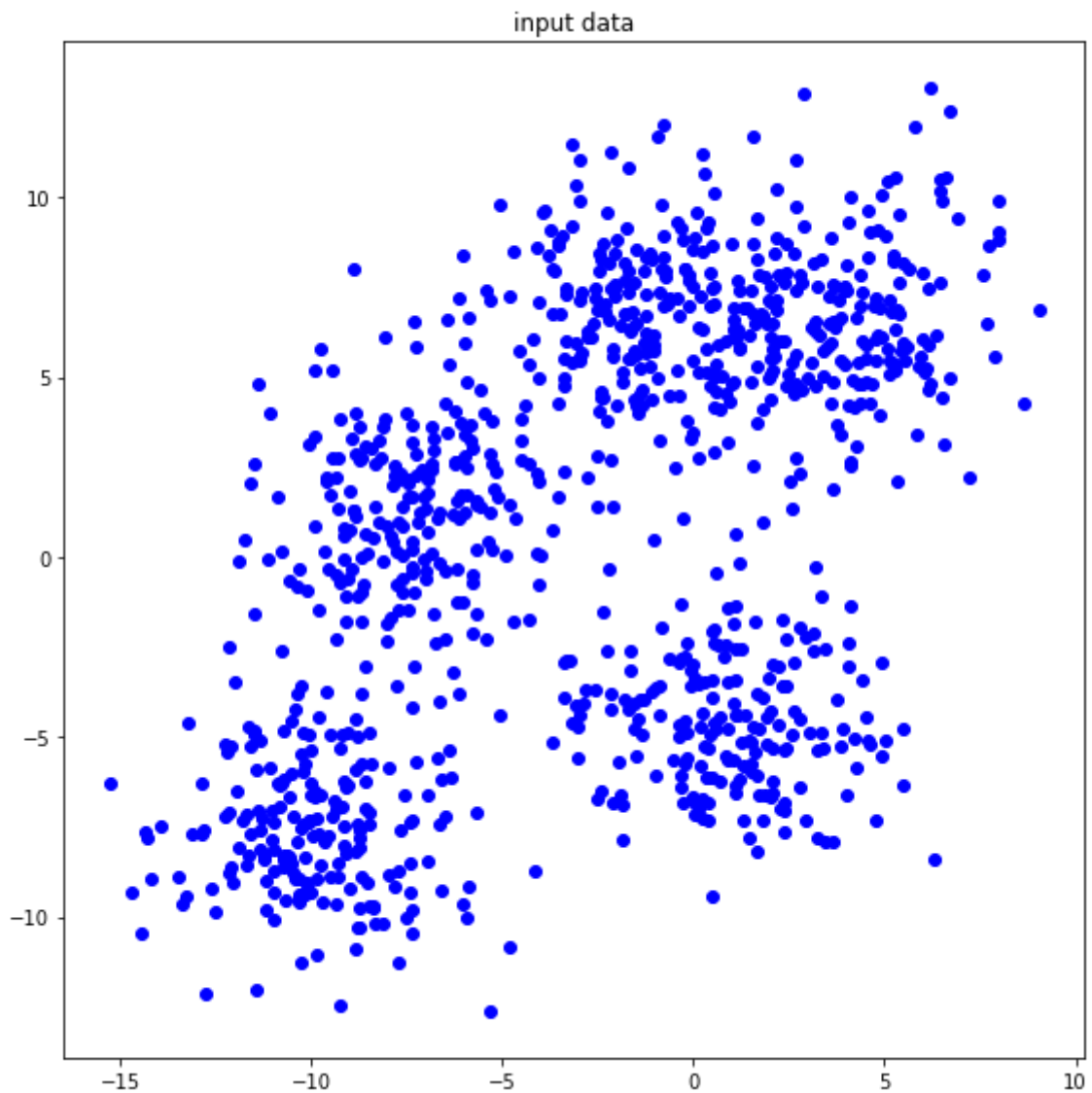
```

## 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 :  $1 \times 1$ , 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):
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
    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., 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., 0., 0., 1., 0., 0., 1.,
1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 0., 0., 0., 0., 0., 0., 0., 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., 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., 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., 0., 1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 1.,
1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 0., 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., 0., 0., 0., 1., 0., 1., 0., 1., 1., 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., 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):
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
    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([1., 3., 3., 1., 2., 0., 3., 1., 1., 2., 0., 2., 1., 2., 1., 3., 2.,
2., 3., 2., 2., 1., 1., 0., 3., 3., 2., 1., 1., 3., 3., 1., 1., 1.,
3., 2., 3., 3., 3., 1., 3., 1., 2., 0., 2., 1., 1., 0., 1., 0., 0.,
0., 1., 2., 1., 2., 0., 3., 0., 3., 0., 0., 2., 1., 0., 3., 0., 1.,
0., 1., 1., 2., 1., 1., 0., 2., 2., 1., 3., 2., 1., 1., 0., 3., 3.,
0., 2., 0., 2., 1., 1., 1., 1., 0., 1., 3., 1., 1., 1., 0., 1., 2.,
1., 1., 1., 1., 2., 2., 3., 1., 2., 2., 2., 3., 3., 1., 1., 1., 3.,
0., 3., 3., 1., 3., 1., 2., 3., 1., 3., 3., 1., 1., 2., 2., 2., 1.,
2., 1., 1., 3., 2., 1., 1., 3., 0., 1., 2., 0., 0., 0., 2., 1., 1.,
1., 0., 2., 1., 1., 2., 1., 1., 2., 1., 3., 0., 0., 2., 3., 1.,
0., 3., 1., 3., 3., 1., 0., 3., 1., 0., 3., 1., 2., 3., 0., 1., 3.,
1., 0., 1., 2., 1., 1., 3., 1., 1., 2., 3., 0., 0., 1., 1., 2., 3.,
0., 0., 1., 2., 1., 1., 3., 1., 1., 0., 3., 2., 2., 1., 2., 3., 0.,
1., 2., 2., 3., 1., 0., 1., 1., 0., 3., 2., 2., 1., 2., 3., 0.,
1., 2., 2., 3., 1., 0., 1., 1., 0., 3., 3., 1., 1., 0., 1., 1.,
3., 1., 3., 1., 3., 2., 1., 0., 3., 3., 1., 1., 0., 1., 1., 3.,
2., 1., 3., 1., 0., 1., 0., 1., 0., 3., 3., 2., 0., 3., 3., 2., 0.,
0., 2., 1., 1., 1., 1., 1., 3., 0., 3., 2., 3., 2., 1., 0., 3., 1.,
3., 2., 0., 1., 0., 3., 1., 3., 0., 1., 3., 1., 3., 1., 2., 1., 3.,
0., 0., 1., 0., 2., 0., 3., 1., 1., 2., 0., 0., 2., 2., 3., 1., 3.,
0., 0., 3., 0., 3., 2., 3., 3., 3., 3., 1., 1., 2., 2., 2., 2., 1.,
1., 1., 1., 1., 1., 3., 1., 2., 2., 1., 3., 1., 0., 0., 3., 1., 3.,
2., 3., 2., 2., 3., 2., 3., 3., 1., 3., 0., 0., 0., 3., 3., 1., 0.,
1., 1., 1., 3., 1., 3., 1., 1., 3., 3., 1., 0., 1., 1., 0., 2., 1.,
2., 0., 0., 0., 1., 1., 2., 3., 0., 1., 3., 1., 2., 1., 1., 2., 0.,
1., 0., 0., 0., 1., 3., 2., 0., 0., 0., 1., 1., 2., 2., 2., 0., 3.,
0., 1., 3., 2., 1., 3., 1., 3., 1., 0., 1., 0., 2., 1., 3., 2., 3.,
1., 2., 3., 0., 1., 0., 2., 2., 0., 2., 1., 0., 2., 0., 1., 0., 3.,
3., 0., 2., 2., 3., 0., 1., 2., 3., 1., 3., 1., 1., 1., 1., 2., 3.,
2., 1., 1., 1., 0., 1., 3., 1., 0., 1., 2., 2., 3., 0., 0., 0., 3.,
3., 3., 2., 1., 1., 3., 2., 0., 1., 2., 1., 0., 3., 3., 1., 1., 1.,
1., 2., 1., 3., 2., 1., 3., 1., 1., 3., 3., 3., 1., 1., 3., 2., 3.,
2., 0., 3., 0., 2., 0., 2., 2., 2., 1., 2., 1., 0., 1., 2., 1., 2.,
2., 1., 0., 2., 1., 0., 1., 3., 0., 1., 3., 0., 2., 2., 2., 1., 2.,
2., 2., 1., 1., 1., 2., 3., 3., 3., 3., 3., 3., 0., 2., 1., 1., 0.,
2., 1., 2., 1., 1., 1., 1., 2., 0., 0., 3., 0., 3., 1., 2., 1., 0.,
3., 0., 3., 2., 0., 1., 3., 1., 3., 1., 2., 0., 1., 1., 3., 2., 0.,
3., 2., 3., 0., 3., 1., 2., 1., 2., 1., 1., 1., 1., 1., 2., 0., 1.,
1., 2., 3., 0., 1., 2., 1., 1., 1., 3., 0., 2., 1., 1., 1., 3., 1.,
3., 3., 3., 0., 1., 3., 2., 3., 3., 1., 2., 1., 0., 0., 2., 3., 3.,
2., 3., 2., 0., 0., 1., 0., 3., 3., 0., 0., 1., 1., 2., 1., 3., 1.,
3., 2., 3., 1., 0., 1., 1., 1., 1., 2., 3., 1., 2., 0., 1., 1., 3.,
2., 1., 0., 0., 1., 2., 3., 1., 3., 1., 0., 2., 0., 3., 3., 1., 0.,
1., 1., 3., 2., 0., 2., 2., 1., 1., 2., 0., 1., 3., 2., 2., 1.,
2., 0., 1., 3., 3., 3., 0., 3., 0., 0., 1., 3., 3., 0., 1., 0., 1.,
1., 1., 0., 0., 0., 1., 0., 1., 2., 3., 3., 1., 1., 1., 3., 0., 3.,
3., 1., 2., 3., 0., 3., 1., 1., 3., 2., 0., 2., 0., 2., 1., 2., 2.,
2., 1., 0., 1., 3., 3., 0., 3., 1., 1., 2., 1., 1., 1., 3., 0., 2.,
3., 0., 1., 2., 0., 1., 1., 0., 1., 0., 1., 2., 0., 1., 0., 1., 2.,
0., 1., 1., 0., 3., 1., 3., 0., 1., 3., 1., 1., 0., 1., 2., 1., 1.,
1., 0., 1., 2., 1., 1., 1., 1., 2., 3., 1., 0., 0., 1., 1., 1., 1.,
1., 1., 1., 2., 2., 3., 3., 2., 0., 3., 3., 0., 2., 2., 1., 2., 1.,

```



```

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

```

## 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):
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
    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([7., 6., 6., 7., 2., 0., 1., 5., 5., 2., 0., 5., 7., 2., 7., 6., 2.,
 2., 1., 2., 4., 5., 5., 0., 6., 6., 2., 5., 5., 6., 7., 3., 5., 3.,
 6., 2., 6., 6., 1., 3., 6., 3., 4., 0., 4., 3., 5., 0., 7., 0., 0.,
 0., 7., 4., 3., 4., 0., 6., 0., 6., 0., 0., 2., 5., 0., 6., 0., 5.,
 0., 3., 7., 4., 7., 3., 0., 4., 2., 7., 7., 4., 3., 3., 0., 6., 1.,
 0., 2., 2., 2., 3., 7., 7., 5., 0., 5., 1., 3., 5., 5., 0., 7., 4.,
 3., 3., 7., 5., 4., 2., 6., 3., 4., 2., 2., 1., 6., 3., 3., 5., 1.,
 0., 1., 5., 5., 6., 3., 4., 1., 7., 1., 6., 5., 5., 2., 2., 4., 7.,
 4., 7., 5., 6., 4., 3., 7., 1., 0., 3., 4., 1., 0., 0., 2., 5., 7.,
 7., 0., 2., 7., 3., 4., 3., 7., 4., 5., 6., 0., 0., 0., 2., 6., 5.,
 0., 6., 5., 6., 6., 7., 7., 0., 6., 7., 4., 1., 0., 7., 0., 5., 6.,
 5., 0., 3., 4., 3., 3., 6., 5., 3., 2., 6., 0., 0., 3., 3., 7., 0.,
 4., 3., 0., 7., 0., 7., 2., 1., 1., 6., 3., 4., 3., 4., 4., 5., 1.,
 5., 2., 2., 6., 3., 0., 7., 5., 0., 7., 2., 4., 5., 2., 5., 0., 5.,
 1., 0., 5., 6., 3., 1., 4., 3., 0., 6., 6., 5., 7., 0., 7., 5., 1.,
 2., 3., 6., 7., 0., 3., 0., 5., 0., 1., 1., 4., 0., 1., 1., 2., 0.,

```

```

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

```

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):
    loss=0
    for i in range(number_cluster):
        loss = loss + np.sum(distance[label_feature==i][:,i])
    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([[13.,  4.,  4.,  1.,  7.,  2., 11., 15.,  6.,  5., 14.,  6.,  1.,
  5.,  1.,  0., 12.,  7., 11., 12.,  5., 15.,  6., 14.,  4.,  4.,
  7., 13., 15.,  4.,  1., 10., 15.,  9.,  4.,  7.,  4.,  4.,  0.,
  9.,  4., 10., 12., 14.,  5., 10., 15., 14.,  3., 14., 14., 14.,
  1.,  8., 10.,  5.,  2.,  4.,  2.,  4.,  2., 14.,  5., 15.,  2.,
  4.,  2., 15., 14., 10., 13.,  5.,  1.,  9.,  2.,  8., 12.,  1.,
  1.,  8.,  9., 10., 14., 13., 11.,  2., 12.,  7.,  7., 15.,  1.,
  1.,  3., 14., 10., 11.,  9., 15., 15., 14., 13.,  5., 10., 10.,
 13., 15., 12., 12.,  0.,  9., 12.,  7.,  7., 11.,  4., 10.,  9.,
 10., 11., 14., 11., 13., 13., 13.,  9.,  5., 11.,  1., 11.,  4.,
 15., 15.,  7., 12.,  8.,  1.,  5.,  1., 15.,  0., 12.,  9.,  3.,
  0.,  2.,  3.,  5.,  2.,  2., 14., 12., 15.,  3., 13.,  2.,  7.,
  1.,  9.,  8., 10.,  1., 12., 15.,  0.,  2.,  2.,  2.,  7.,  4.,
 15.,  2.,  4., 15., 13.,  4.,  3., 13., 14.,  4.,  1.,  5.,  0.,
 14.,  1.,  2., 10.,  4.,  6.,  2.,  3.,  5.,  9., 10.,  4., 13.,
  9.,  7.,  4., 14., 14., 10.,  3.,  3., 14.,  5.,  9., 14.,  1.,
 14.,  1., 12.,  0., 11., 13., 10.,  5.,  9.,  8.,  5., 10.,  0.,
 10.,  7., 12., 13.,  9.,  2., 13., 15., 14., 13.,  7.,  5., 15.,
  7., 13., 14.,  6.,  0., 14., 15.,  4., 10., 11.,  5.,  3., 14.,
  4.,  4.,  6.,  1., 14.,  3., 15., 11.,  7., 10.,  4.,  3., 14.,
 10.,  2., 15., 14., 11.,  0.,  8., 14., 11.,  0.,  5., 14.,  2.,
  6.,  1.,  3.,  1.,  1.,  3., 11., 14.,  0.,  7., 11., 12.,  3.,
  2.,  4., 15., 11.,  5.,  2.,  3.,  2.,  0.,  9.,  0., 14.,  3.,
  0., 15.,  0.,  1., 12., 13.,  0.,  2., 14.,  3., 14.,  8., 14.,
 13.,  1., 15.,  7.,  2.,  2.,  7.,  7.,  0., 15.,  0.,  2., 14.,
 11., 14.,  0.,  8.,  4.,  4.,  0.,  0., 10., 15., 12.,  8., 12.,
  7.,  6., 13.,  1.,  3.,  1., 10.,  0., 13.,  7., 12.,  1.,  4.,
  3., 14., 14., 11.,  1.,  4., 12., 11.,  8.,  5.,  0., 12.,  4.,
  0., 15.,  0.,  2., 14.,  2.,  4.,  0.,  1., 14., 13., 15., 10.,
  4.,  3.,  4.,  9., 13.,  4.,  0.,  1.,  2., 13.,  1.,  2., 12.,
 15.,  5., 14.,  2., 14.,  9., 13.,  8., 11., 14., 13.,  0., 10.,
  8.,  3.,  6.,  5., 14., 10.,  2., 14., 14.,  1., 11., 12.,  2.,
  2.,  2.,  1., 10.,  5.,  5.,  7., 14.,  4., 14., 10.,  4., 12.,
 10.,  4.,  6., 11.,  6., 14., 10.,  2.,  8.,  3.,  4., 12.,  0.,
  1.,  5.,  0., 14., 13., 14.,  8.,  5.,  2., 12.,  9., 14.,  8.,
 14., 13., 14.,  4.,  0., 14.,  7.,  5.,  0., 14., 15.,  6.,  0.,
  3., 11.,  3.,  1.,  1., 10.,  5.,  0.,  5.,  3.,  3.,  1.,  2.,
 15.,  4.,  1.,  2., 15., 12.,  8.,  4.,  2., 14., 14.,  4.,  0.,
  0.,  7., 10., 15.,  4.,  8., 11.,  9., 12.,  1., 14.,  0., 11.,
 15., 10.,  3., 10., 12., 10.,  0., 12.,  1.,  4.,  9., 13.,  4.,
  0.,  0.,  9., 13., 11.,  5.,  0.,  5.,  2.,  0., 14.,  6., 14.,
 12.,  5., 12., 15., 12.,  9.,  2., 13.,  5., 10., 12., 12.,  1.,
 14., 12.,  9., 14., 15., 13.,  2.,  3., 11., 14.,  7., 12., 12.,
 15.,  5.,  5.,  5.,  9., 15., 10., 12., 11., 11.,  0.,  4.,  0.,
 11.,  2.,  7., 15., 10.,  2.,  7., 10.,  7.,  9., 15., 10.,  3.,
  7., 14., 14.,  0.,  2.,  4.,  1.,  5., 13.,  2.,  4.,  2.,  4.,
  6.,  2.,  1.,  1., 13.,  0.,  9.,  7.,  2.,  3., 13.,  1.,  5.,
 14.,  4., 12.,  4.,  2.,  0., 10.,  5.,  1., 12., 15., 15., 15.,
  3., 13.,  5., 14., 10.,  1.,  7.,  6.,  2.,  1.,  6.,  3., 10.,
  9.,  4.,  2.,  5.,  9., 15.,  1., 11., 10., 13.,  4.,  4.,  2.,
 15., 11.,  5.,  0.,  0.,  6.,  5., 15., 14., 14.,  5.,  4.,  0.,
 12., 11.,  8.,  2., 14., 10.,  2.,  0.,  0., 14.,  2.,  1.,  1.,
  5.,  1.,  0.,  3.,  0., 12.,  0., 15., 14.,  1., 15.,  1., 15.,

```

```

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

```

---

## 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]
```

```

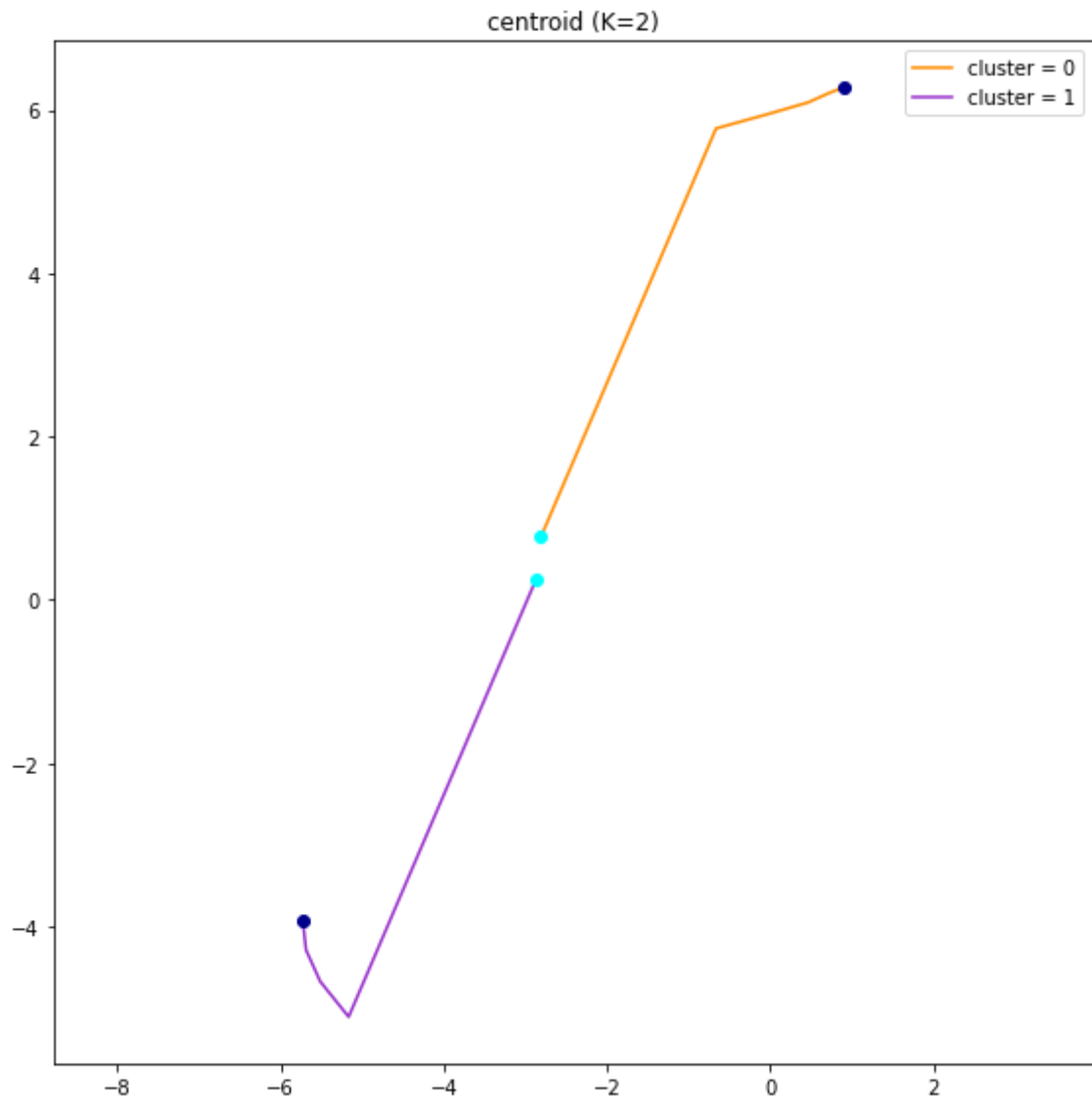
# plt.plot(cluster_0_center_x, cluster_0_center_y, '-', color='blue', label='cluster_0_center')
# plt.plot(cluster_1_center_x, cluster_1_center_y, '-', color='yellow', label='cluster_1_center')
# plt.plot(cluster_0_center_x_initial, cluster_0_center_y_initial, 'o', color='magenta', label='cluster_0_center_initial')
# plt.plot(cluster_0_center_x_final, cluster_0_center_y_final, 'o', color='red', label='cluster_0_center_final')
# plt.plot(cluster_1_center_x_initial, cluster_1_center_y_initial, 'o', color='magenta', label='cluster_1_center_initial')
# plt.plot(cluster_1_center_x_final, cluster_1_center_y_final, 'o', color='red', label='cluster_1_center_final')

# plt.plot(cluster_0_center_x, cluster_0_center_y, '-', color='blue', label='cluster_0_center')
# plt.plot(cluster_1_center_x, cluster_1_center_y, '-', color='yellow', label='cluster_1_center')
# plt.plot(cluster_0_center_x_initial, cluster_0_center_y_initial, 'o', color='magenta', label='cluster_0_center_initial')
# plt.plot(cluster_0_center_x_final, cluster_0_center_y_final, 'o', color='red', label='cluster_0_center_final')
# plt.plot(cluster_1_center_x_initial, cluster_1_center_y_initial, 'o', color='magenta', label='cluster_1_center_initial')
# plt.plot(cluster_1_center_x_final, cluster_1_center_y_final, 'o', color='red', label='cluster_1_center_final')

for i in range(2):
    plt.plot(centroid_iteration_02[:,i,0], centroid_iteration_02[:,i,1], '-', color='blue', label=f'centroid_{i}_iteration_02')
    plt.plot(centroid_iteration_02[0,i,0], centroid_iteration_02[0,i,1], 'o', color='magenta', label=f'centroid_{i}_iteration_02_initial')
    plt.plot(centroid_iteration_02[-1,i,0], centroid_iteration_02[-1,i,1], 'o', color='red', label=f'centroid_{i}_iteration_02_final')
plt.axis('equal')
plt.legend()
plt.tight_layout()
plt.show()
#
# ++++++

```

In [ ]: function\_result\_09()



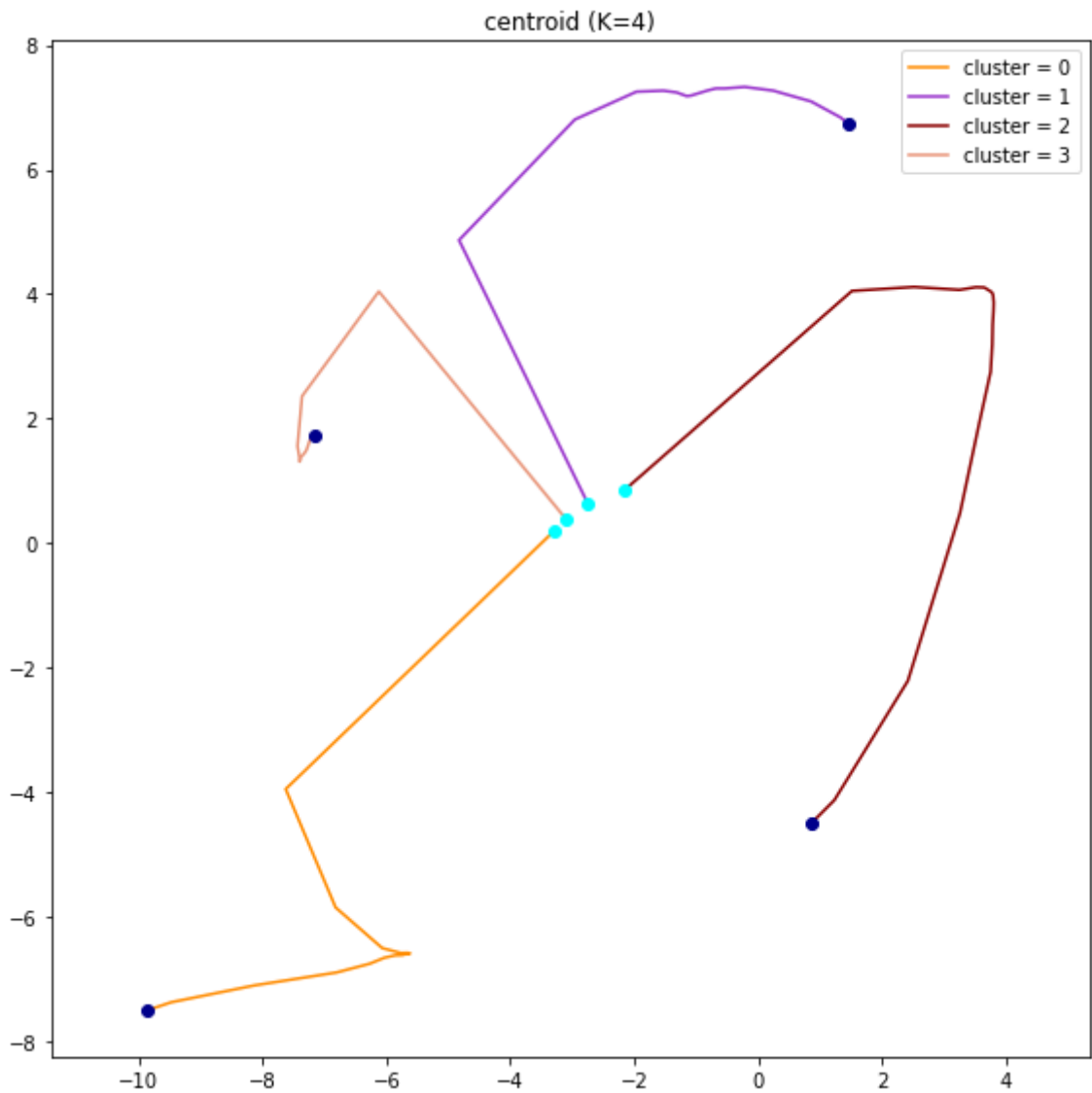
```
In [ ]: def function_result_10():

    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], '-.', color=
        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', color=
    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], '-', color=
        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', color=

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

    #
    # ++++++

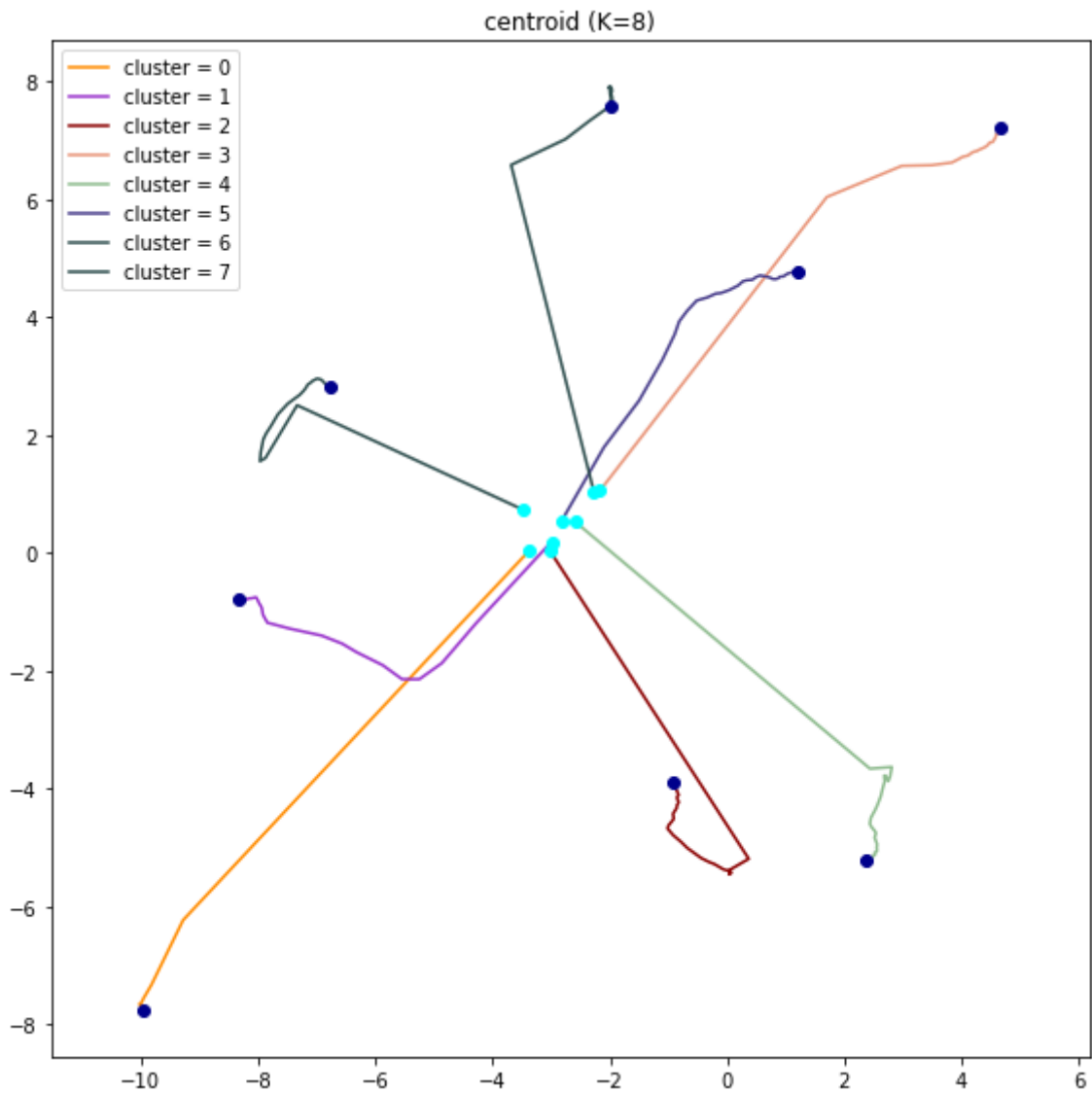
```

```

In [ ]: function_result_11()

```





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

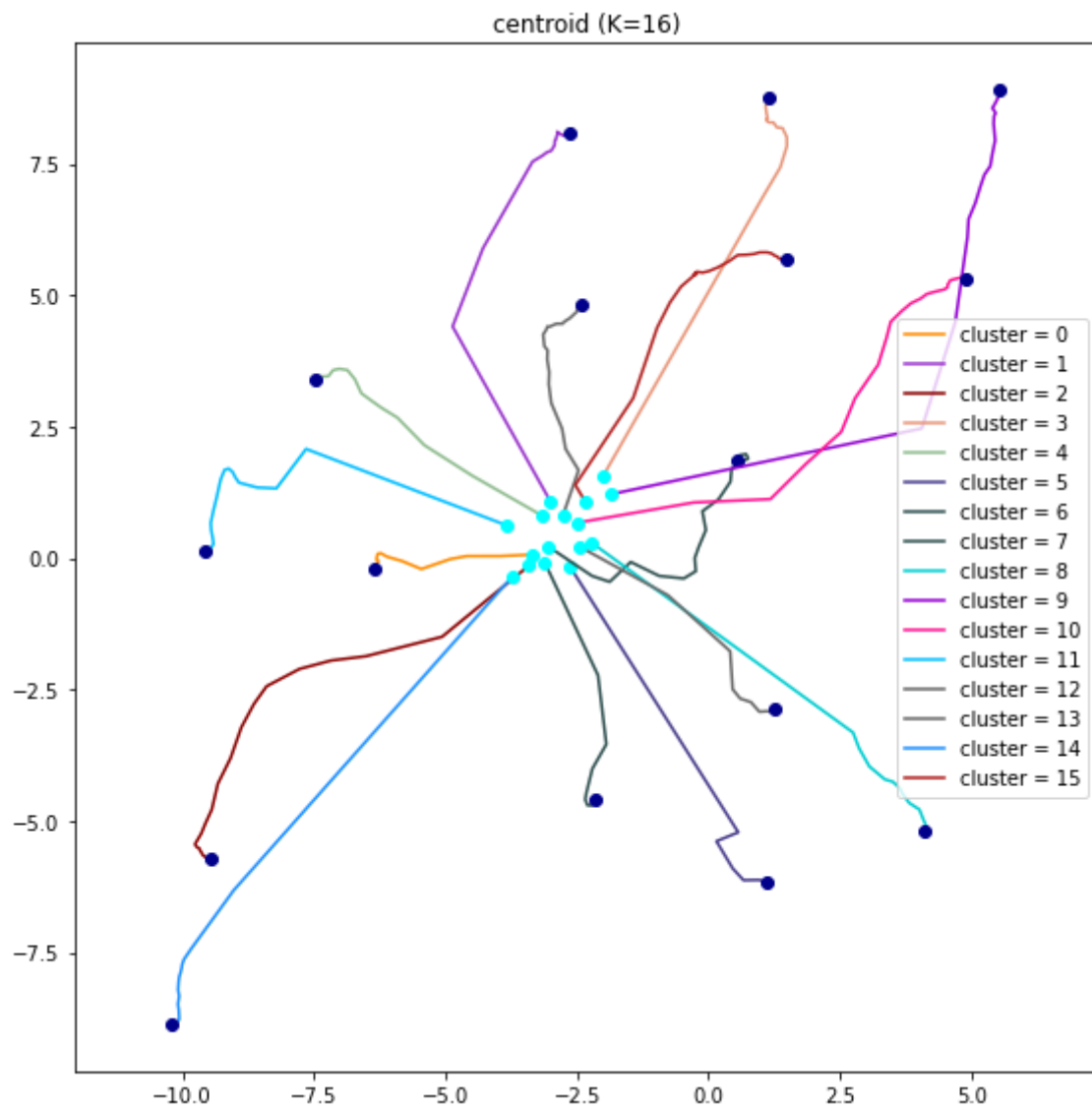
    plt.figure(figsize=(8,8))
    plt.title('centroid (K=16)')

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

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

    #
    # ++++++
```

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



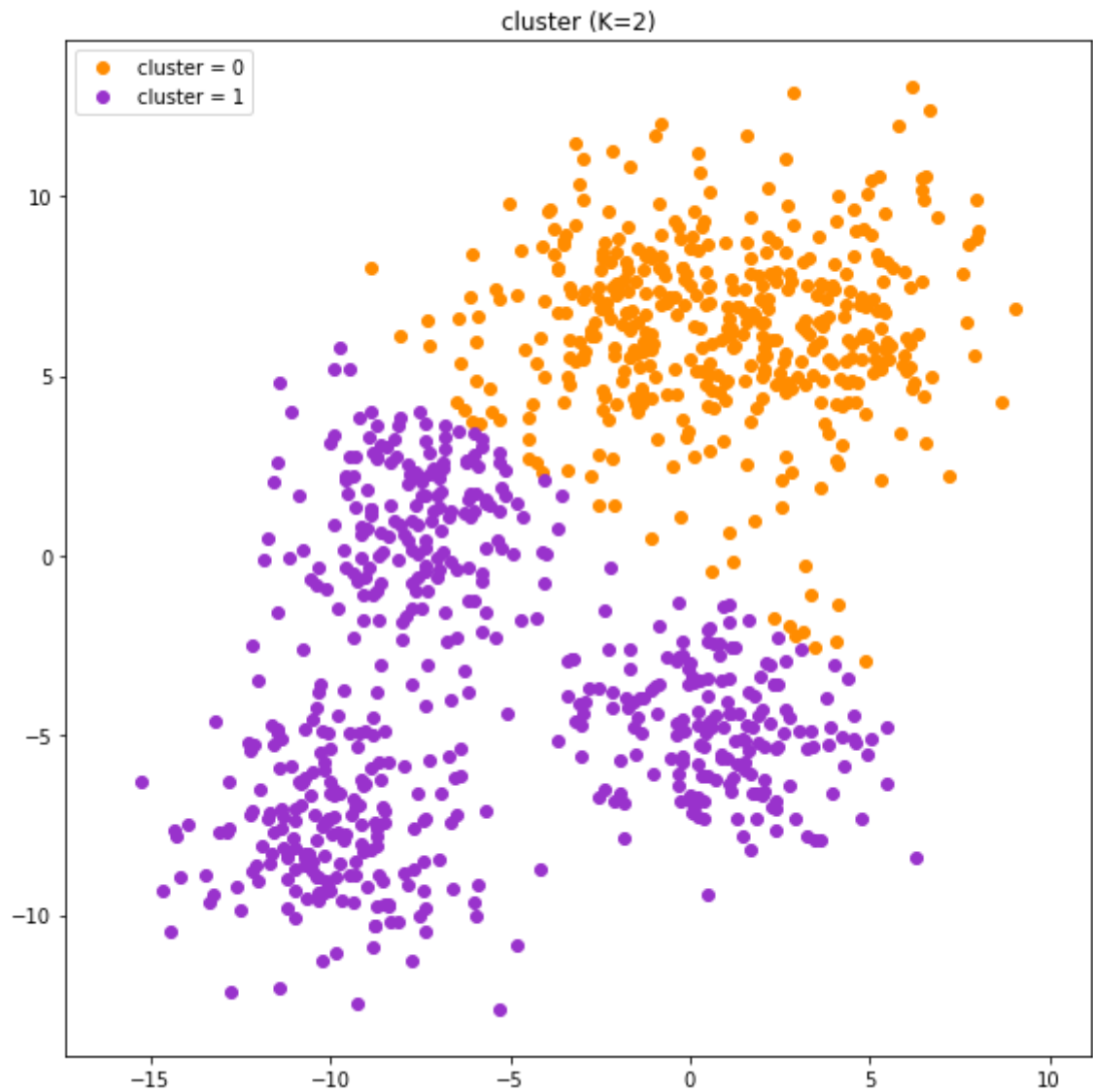
```
In [ ]: def function_result_13():

    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

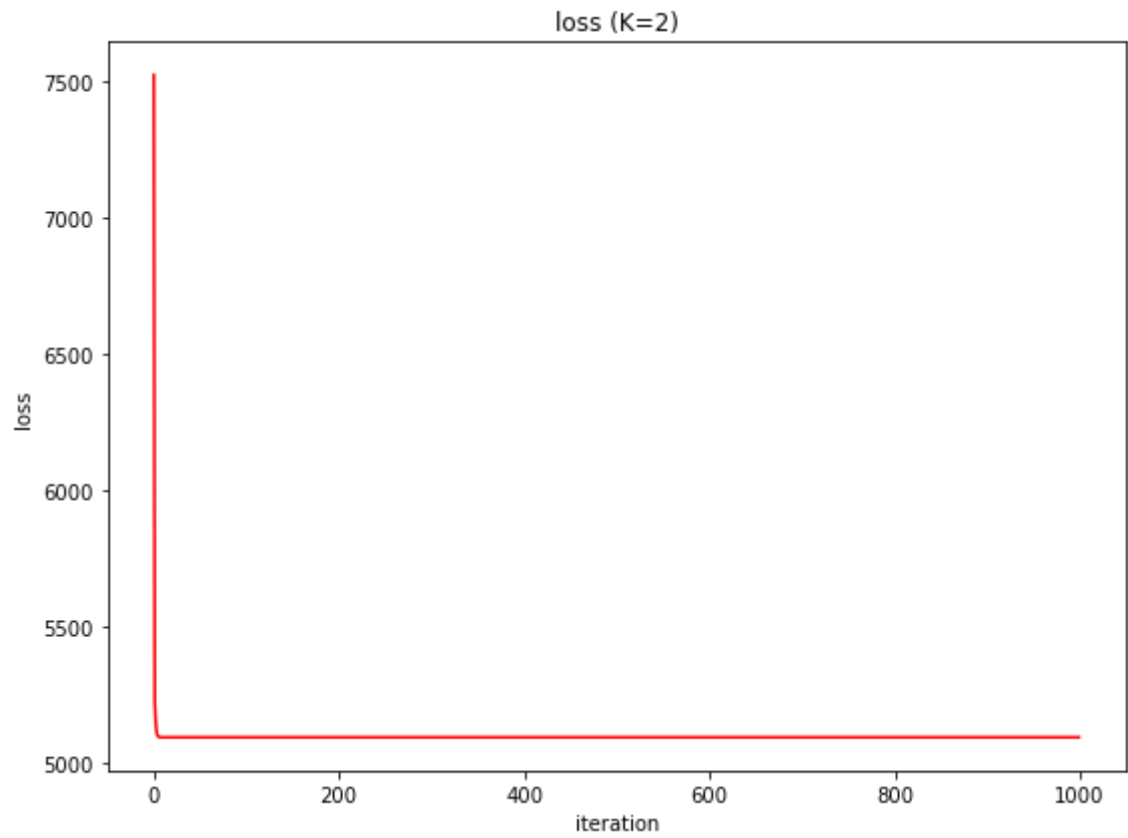
for i in range(number_result):
    title = '## [RESULT {:02d}]'.format(i+1)
    name_function = 'function_result_{:02d}()'.format(i+1)

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

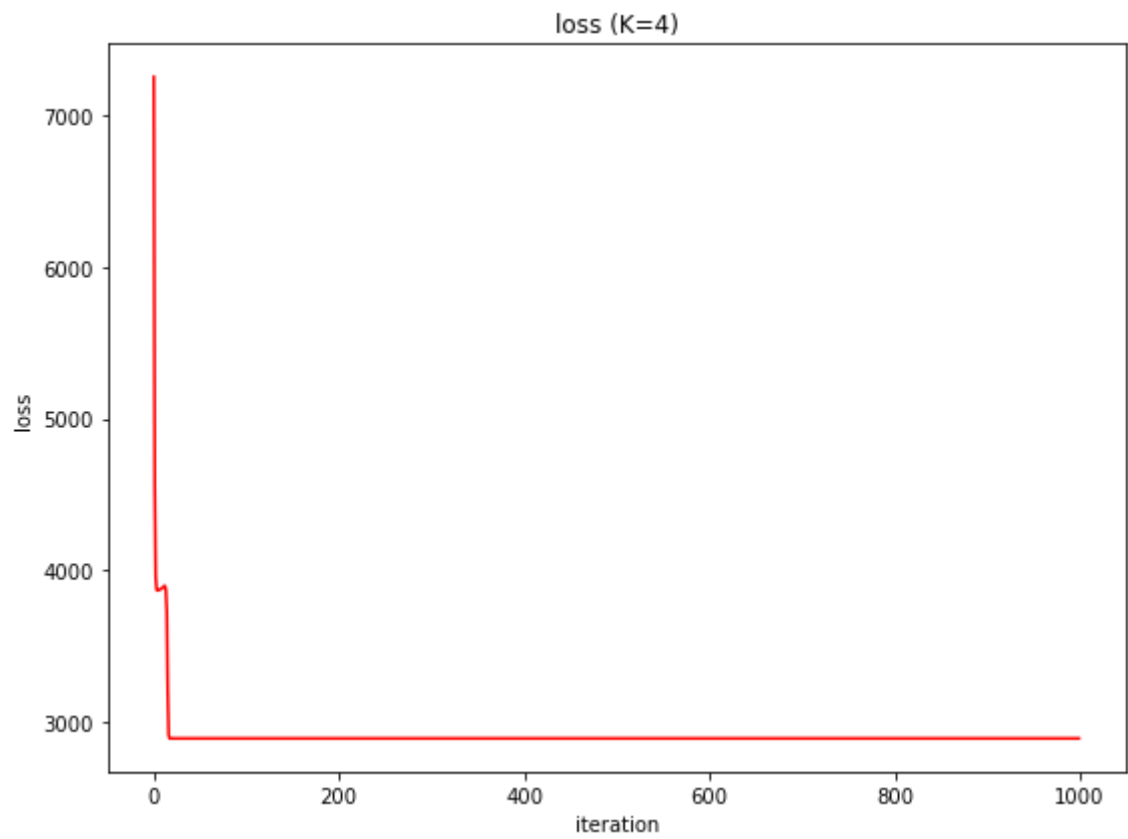
*****
## [RESULT 01]
*****
final loss (K=2) = 5093.3204564471
*****
## [RESULT 02]
*****
final loss (K=4) = 2886.7264895697
*****
## [RESULT 03]
*****
final loss (K=8) = 2109.2664792951
*****
## [RESULT 04]
*****
final loss (K=16) = 1602.8313078212

```

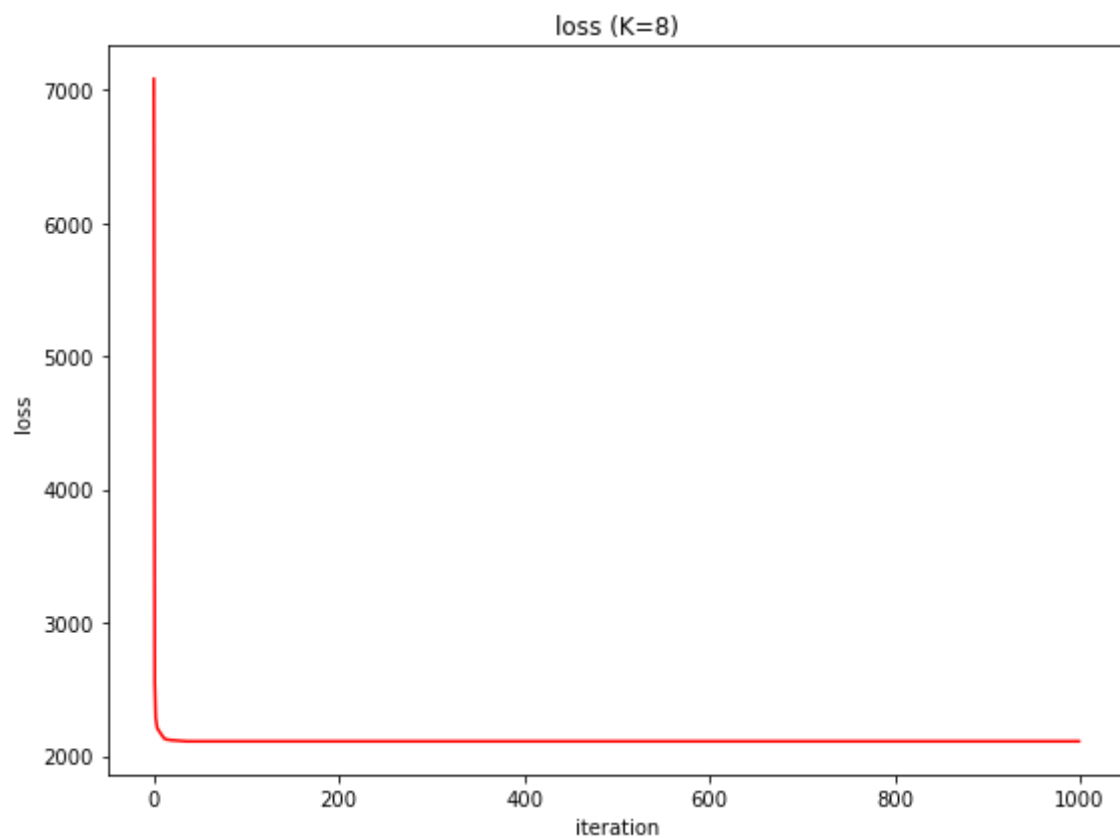
```
*****  
## [RESULT 05]  
*****
```



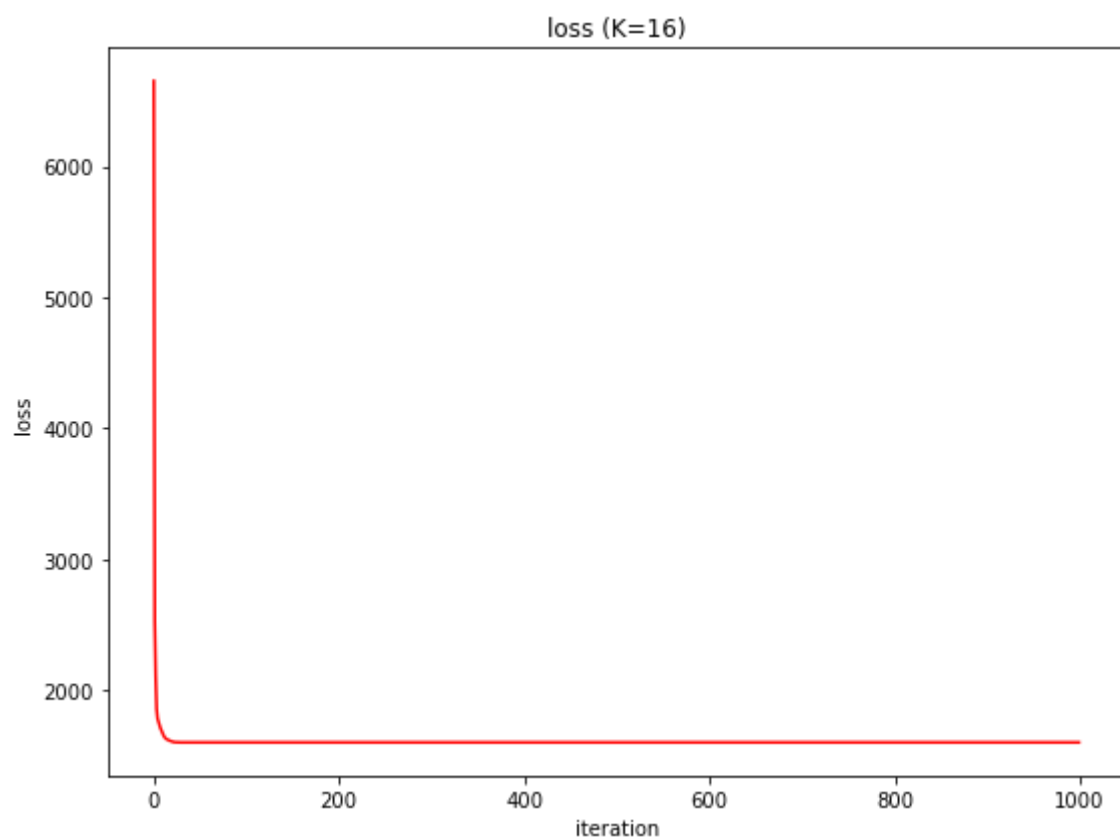
```
*****  
## [RESULT 06]  
*****
```



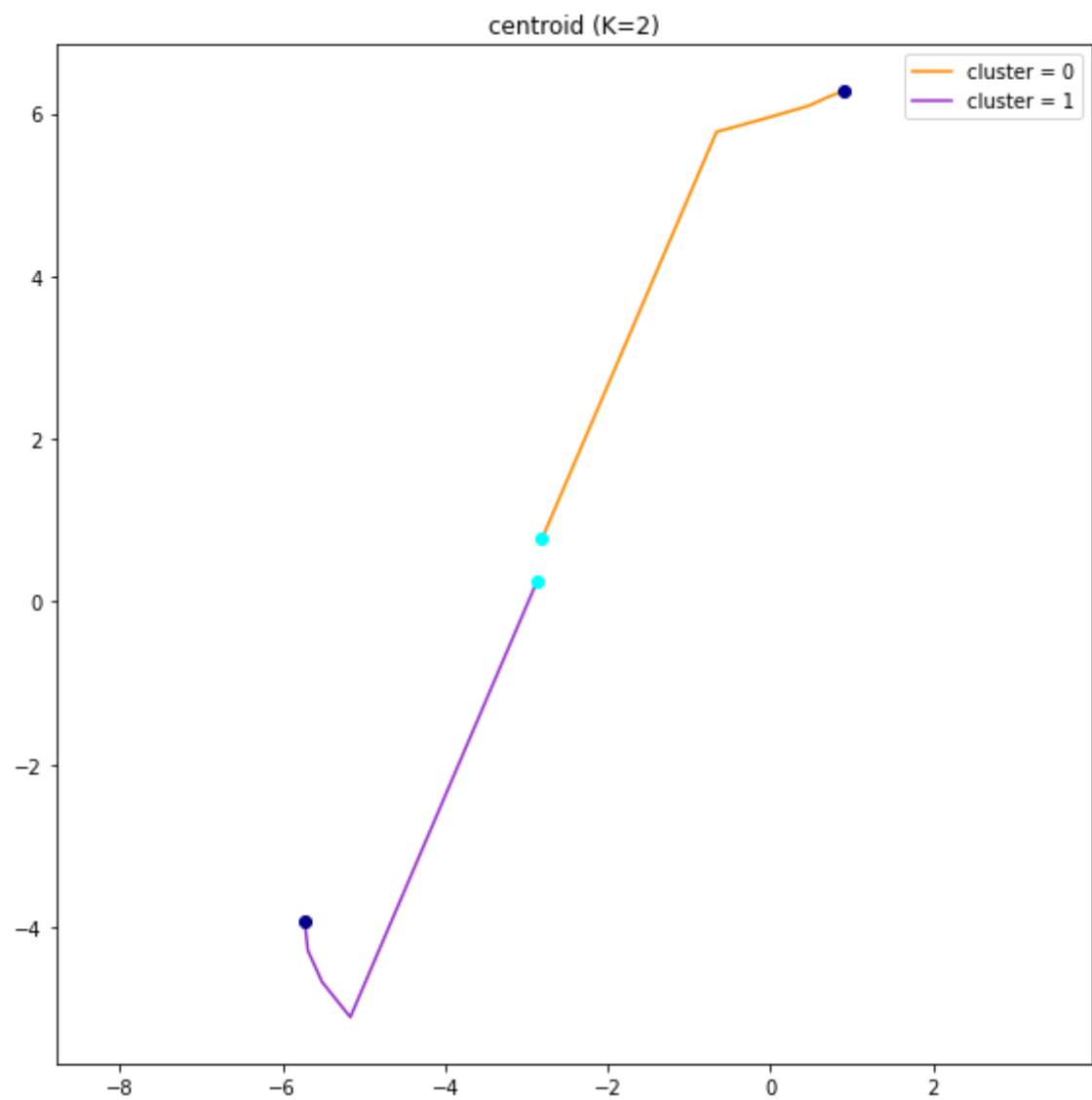
```
*****  
## [RESULT 07]  
*****
```



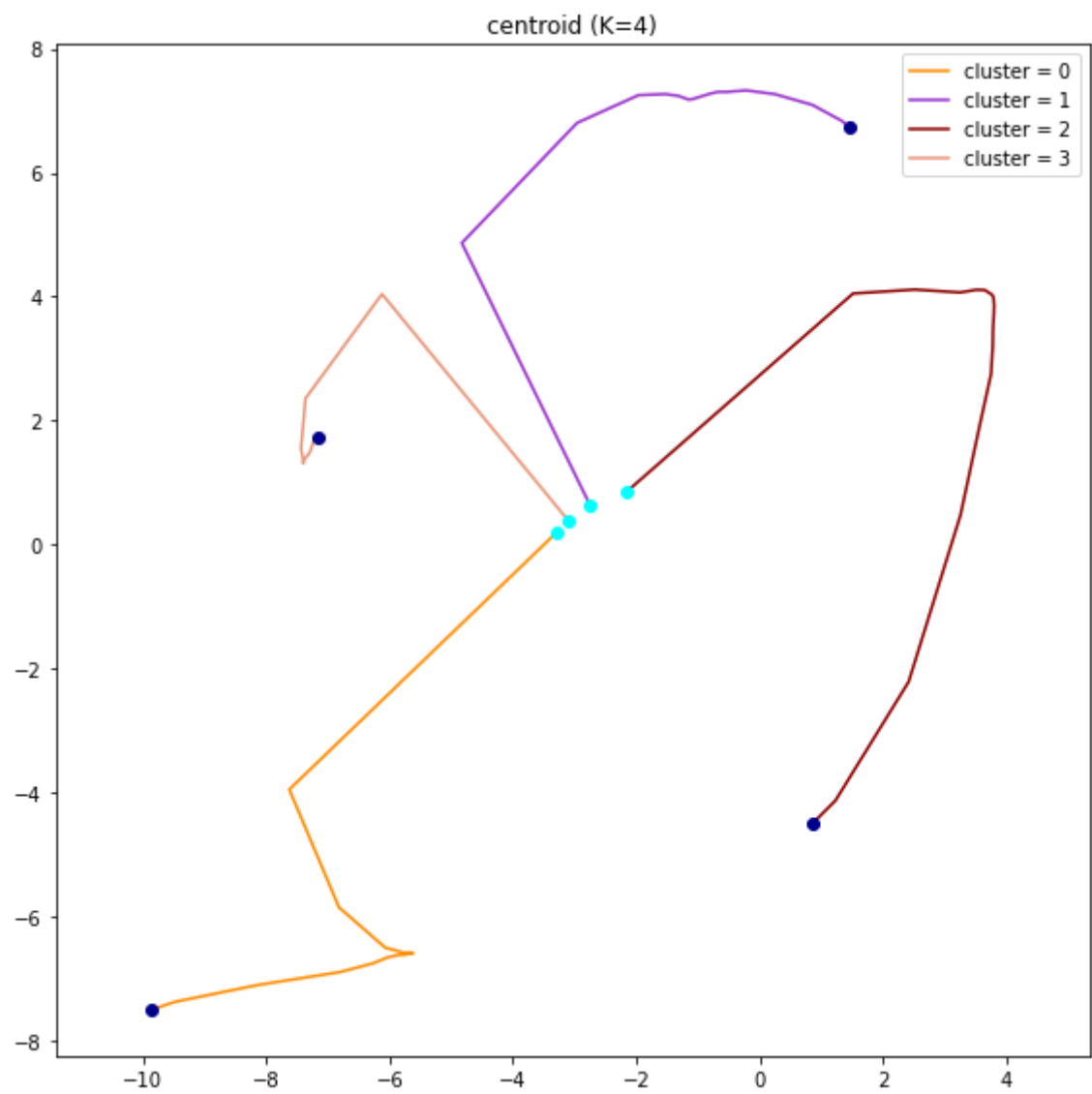
```
*****  
## [RESULT 08]  
*****
```



```
*****  
## [RESULT 09]  
*****
```

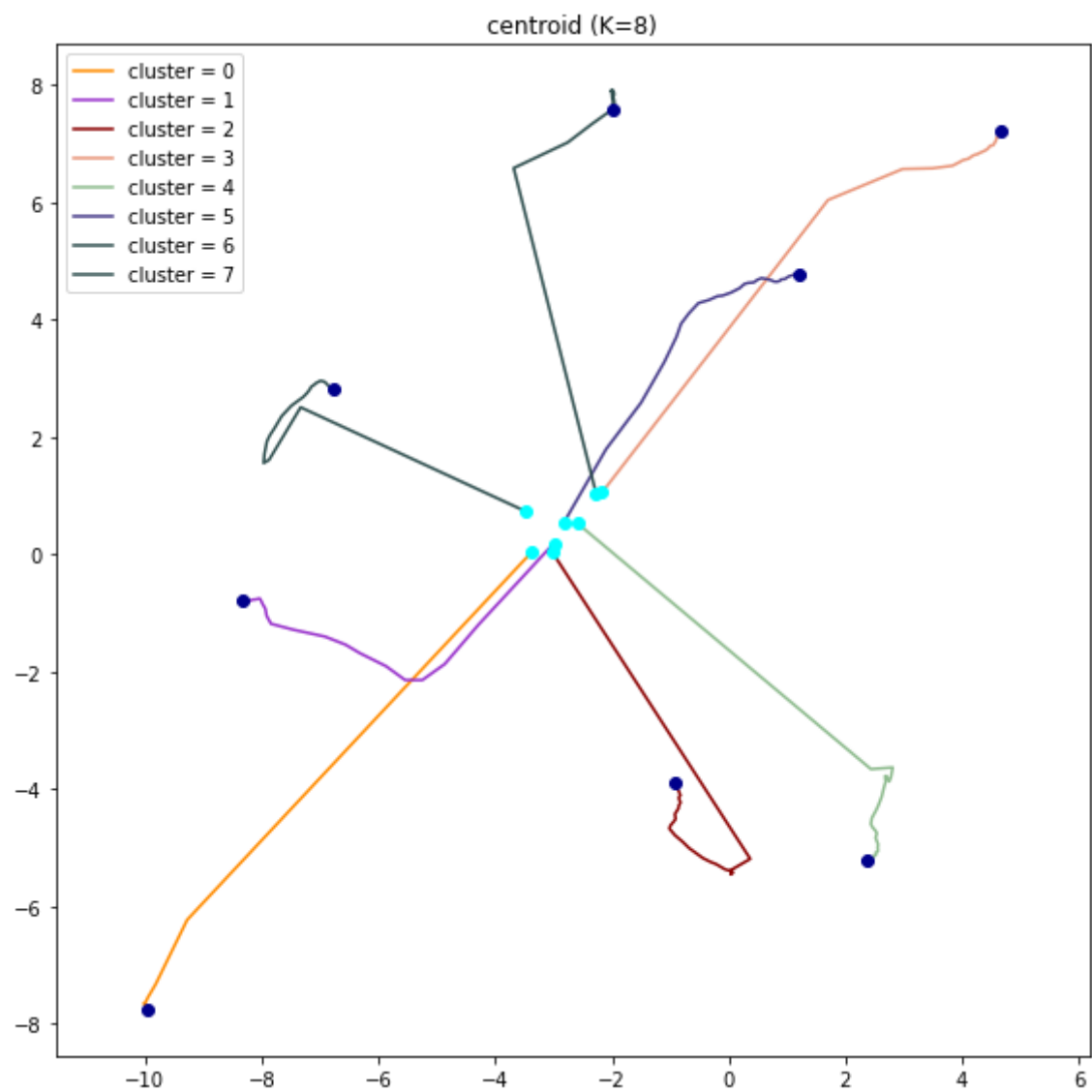


```
*****  
## [RESULT 10]  
*****
```

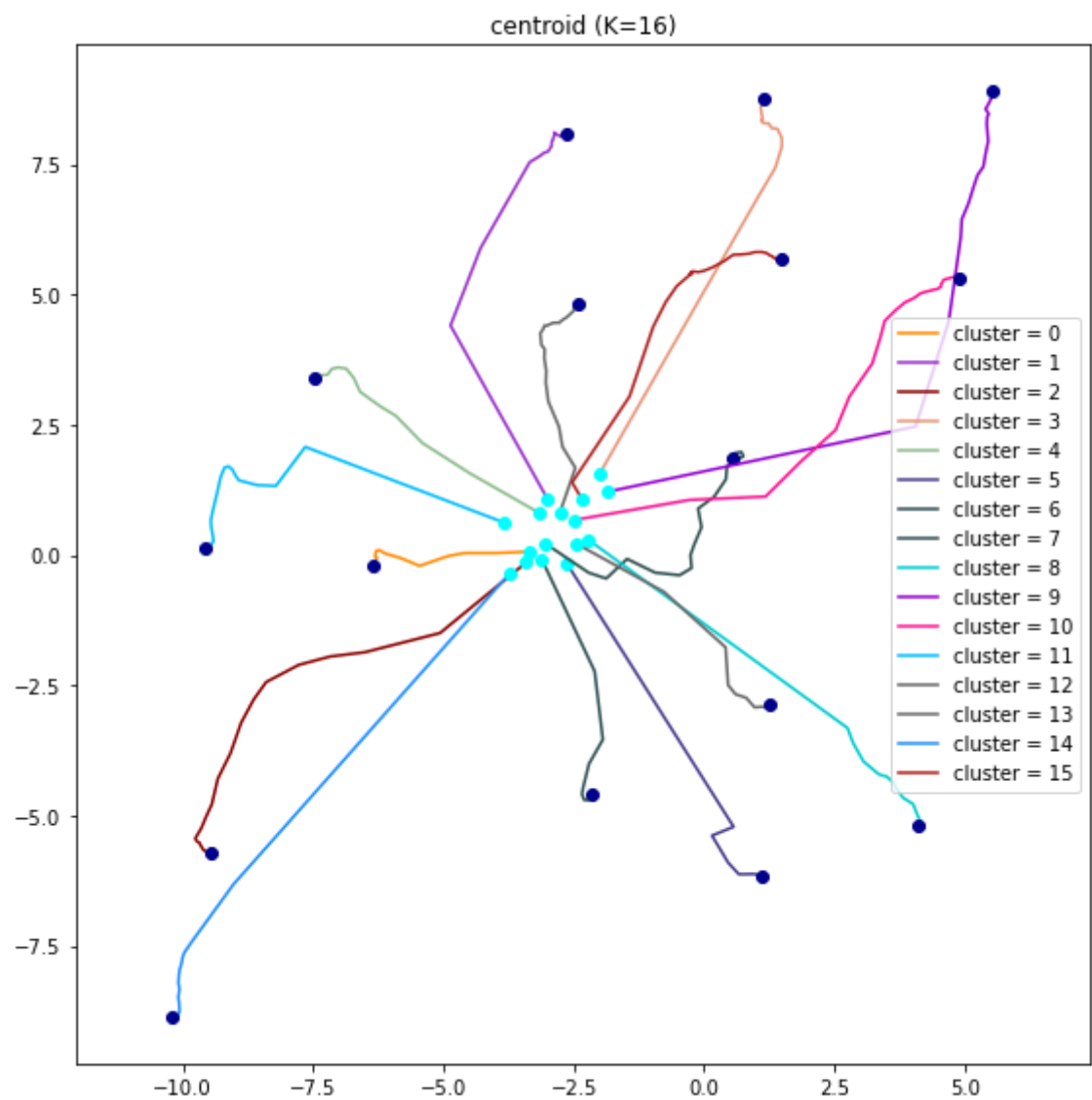


```
*****  
## [RESULT 11]  
*****
```

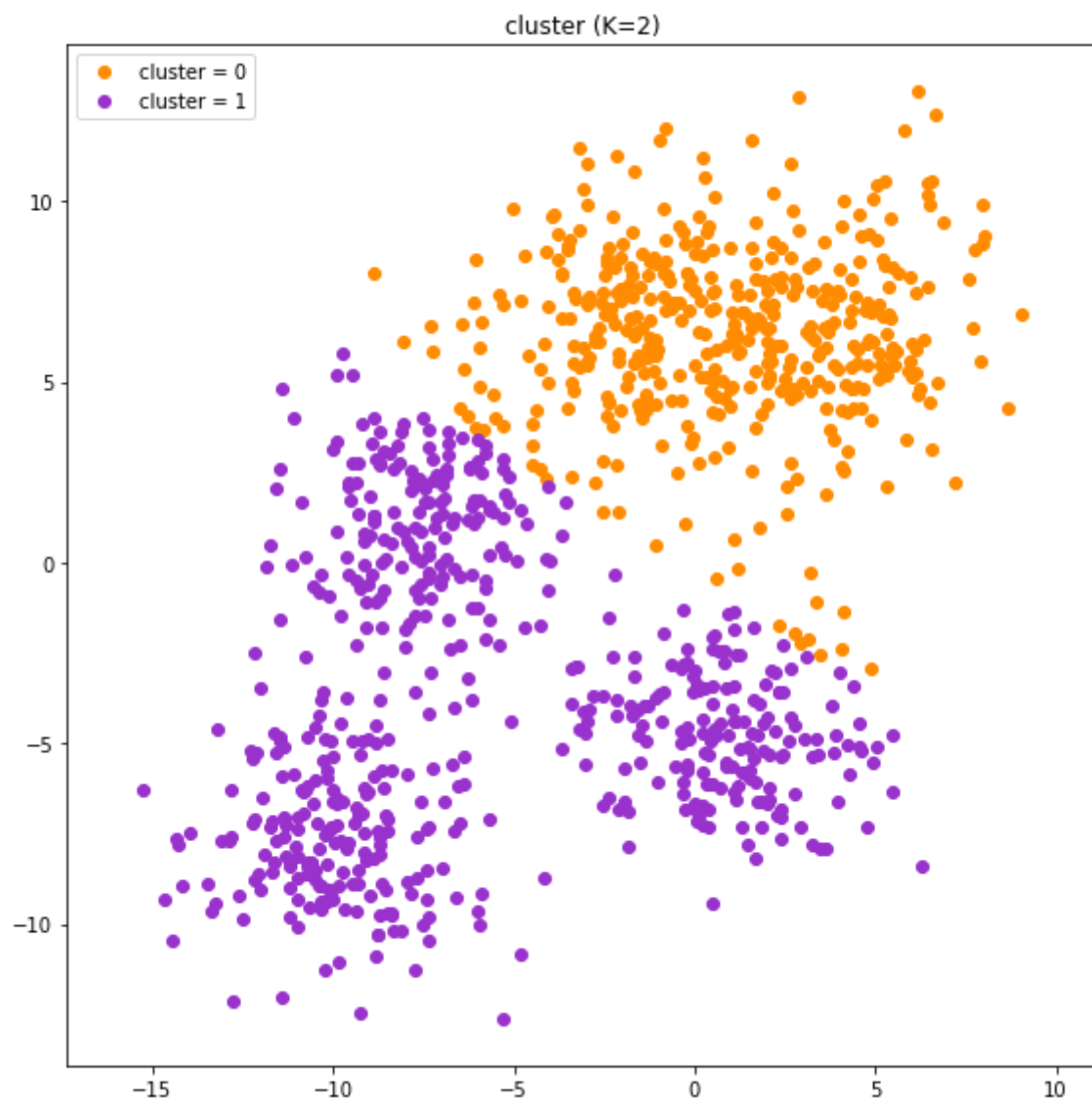




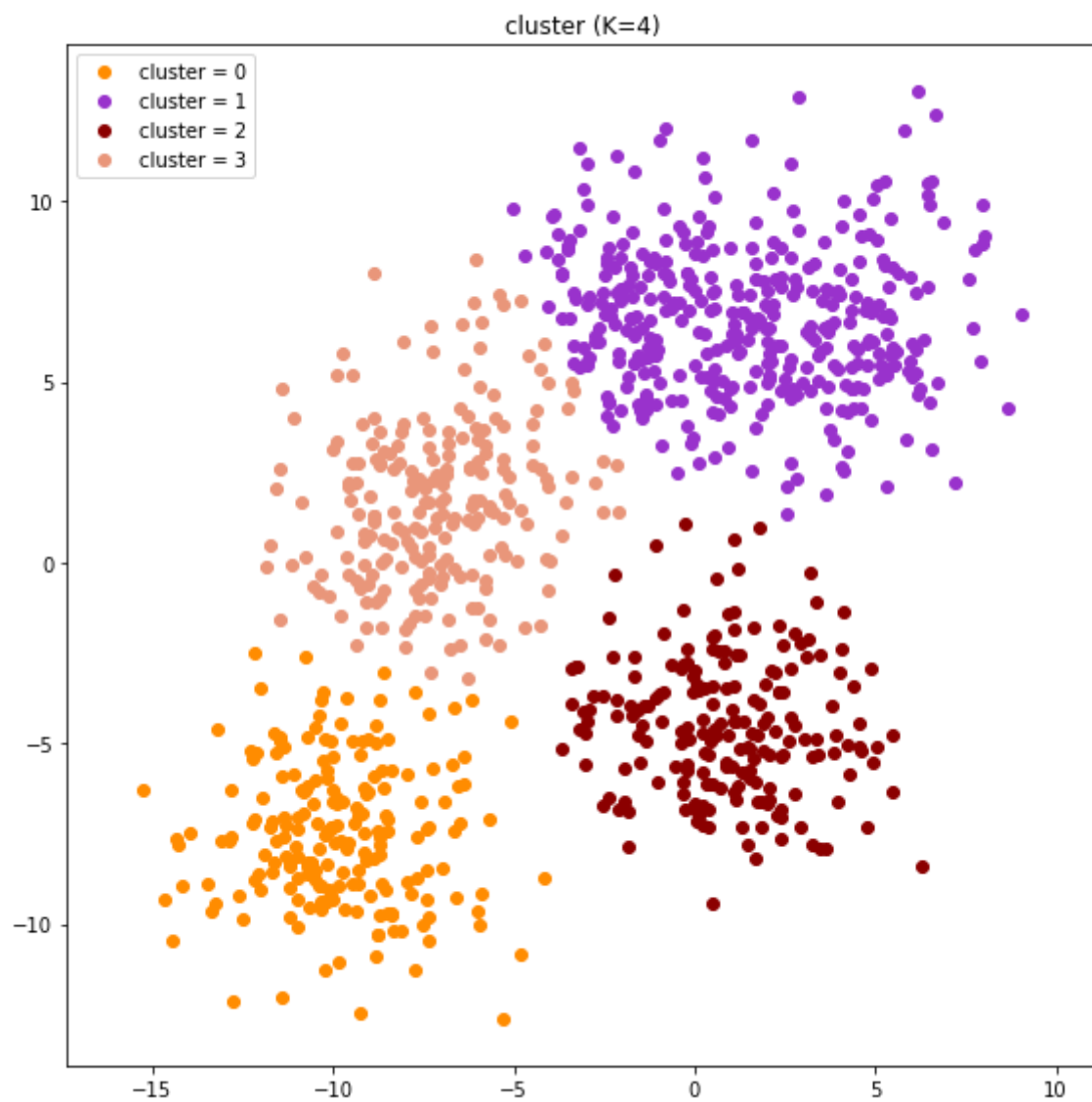
\*\*\*\*\*  
## [RESULT 12]  
\*\*\*\*\*



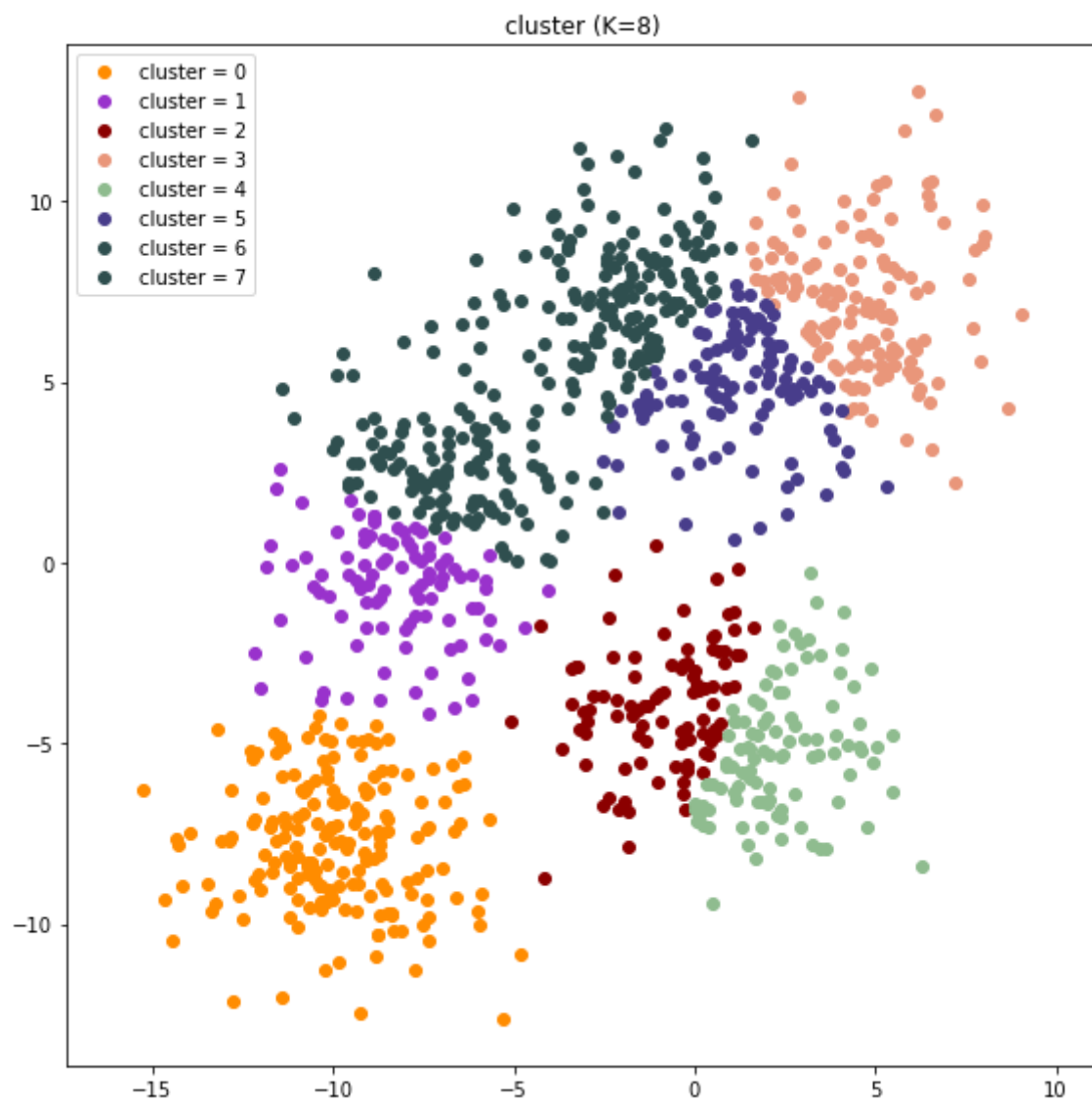
```
*****  
## [RESULT 13]  
*****
```



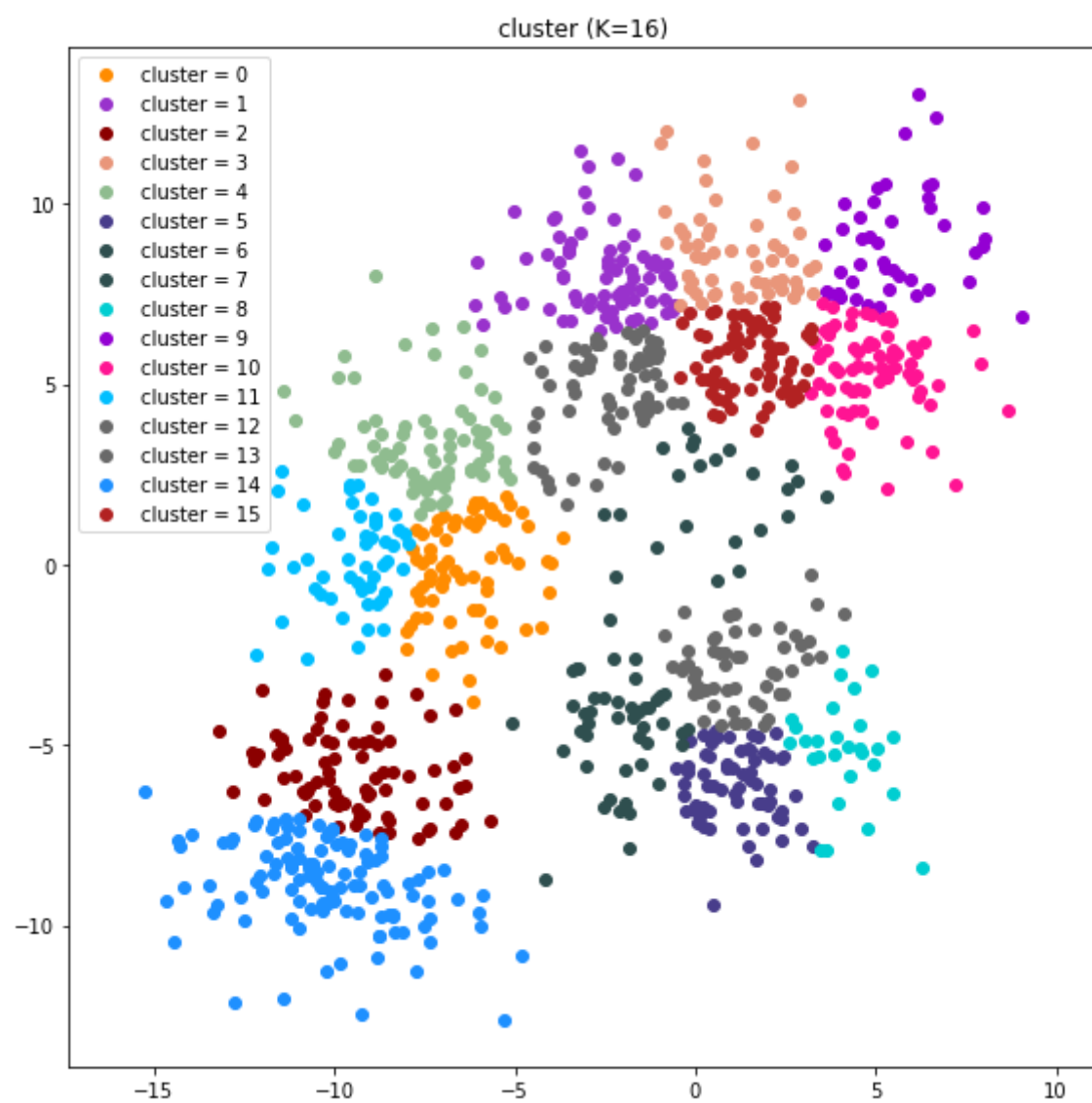
```
*****  
## [RESULT 14]  
*****
```



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



\*\*\*\*\*  
## [RESULT 16]  
\*\*\*\*\*



In [ ]: