K-means clustering

import library

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from matplotlib import cm
```

load data

In [ ]:

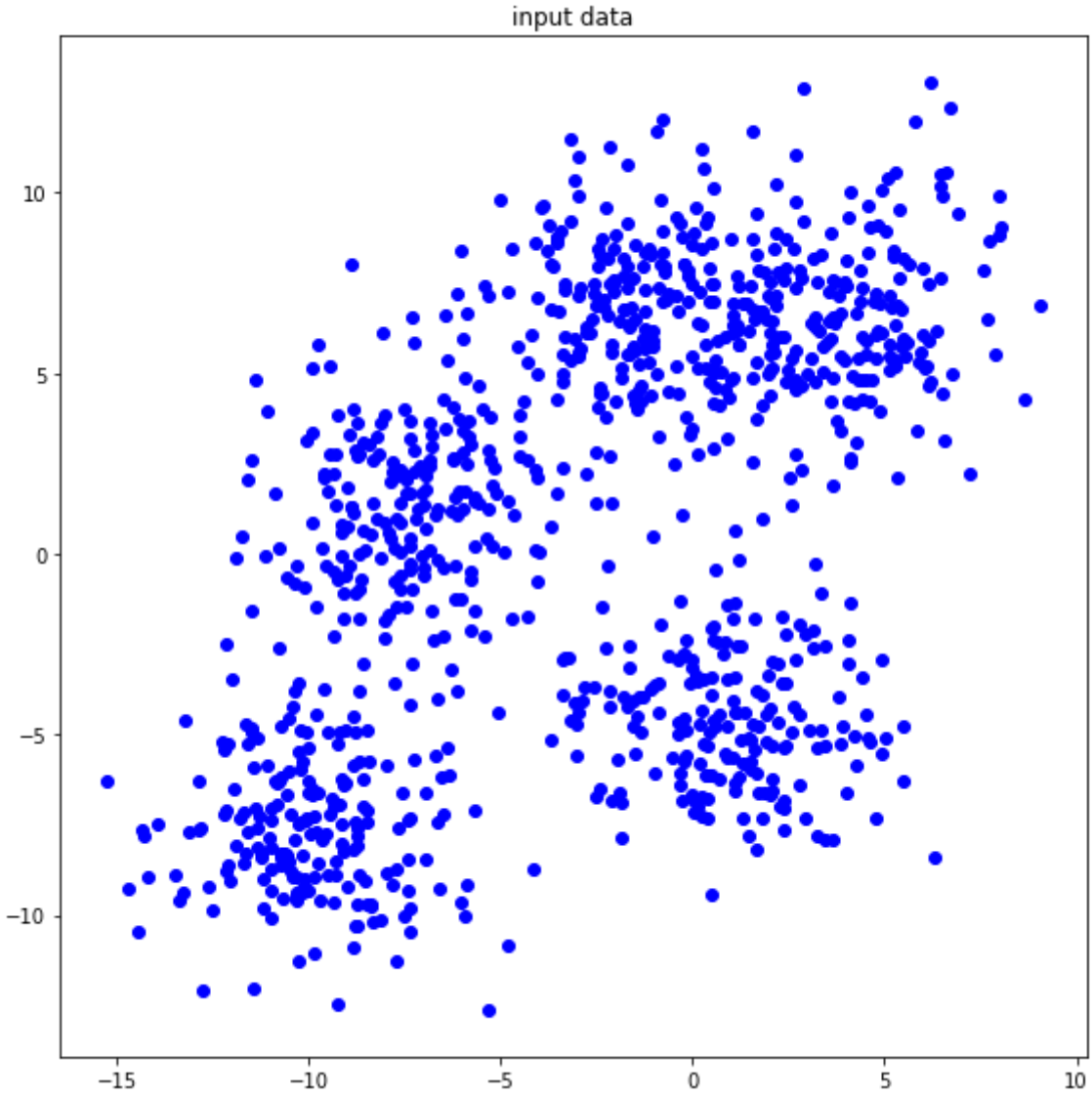
```
fname_data = 'assignment_11_data.csv'

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
```

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((feature-center)**2, 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
    #
    centroid = np.mean(feature[label_feature == label], axis = 0)

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

    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
    #

    label_feature = np.argmin(distance, axis = 1)
    #
    # ++++++++++++++++++++++++++++++++++++++++++++

    return label_feature
```

the number of clusters $K = 2$

In [ ]:

```
number_cluster          = 2
number_iteration        = 100     # 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
#

for i in range(number_iteration):
    loss = 0
    distance = np.zeros((number_data, number_cluster))

    for j in range(number_cluster):
        centroid = compute_centroid(feature, label_feature_02, j)
        if np.isnan(centroid).any():
            centroid = centroid_iteration_02[i-1, j, :]

        centroid_iteration_02[i, j, :] = centroid
        distance[:, j] = compute_distance(feature, centroid)

    label_feature_02 = compute_label(distance)

    loss = np.mean(np.min(distance**2, axis = 1))
    loss_iteration_02[i] = loss
#
# ++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 4$

In [ ]:

```
number_cluster          = 4
number_iteration        = 100     # 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
#

for i in range(number_iteration):
    loss = 0
    distance = np.zeros((number_data, number_cluster))

    for j in range(number_cluster):
        centroid = compute_centroid(feature, label_feature_04, j)
        if np.isnan(centroid).any():
            centroid = centroid_iteration_04[i-1, j, :]

        centroid_iteration_04[i, j, :] = centroid
        distance[:, j] = compute_distance(feature, centroid)

    label_feature_04 = compute_label(distance)

    loss = np.mean(np.min(distance**2, axis = 1))
    loss_iteration_04[i] = loss
#
# ++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 8$

In [ ]:

```
number_cluster          = 8
number_iteration        = 100     # 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
#

for i in range(number_iteration):
    loss = 0
    distance = np.zeros((number_data, number_cluster))

    for j in range(number_cluster):
        centroid = compute_centroid(feature, label_feature_08, j)
        if np.isnan(centroid).any():
            centroid = centroid_iteration_08[i-1, j, :]

        centroid_iteration_08[i, j, :] = centroid
        distance[:, j] = compute_distance(feature, centroid)

    label_feature_08 = compute_label(distance)

    loss = np.mean(np.min(distance**2, axis = 1))
    loss_iteration_08[i] = loss
#
# ++++++++++++++++++++++++++++++++++++++++++++++
```

the number of clusters $K = 16$

In [ ]:

```
number_cluster          = 16
number_iteration        = 100     # 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
#

for i in range(number_iteration):
    loss = 0
    distance = np.zeros((number_data, number_cluster))

    for j in range(number_cluster):
        centroid = compute_centroid(feature, label_feature_16, j)
        if np.isnan(centroid).any():
            centroid = centroid_iteration_16[i-1, j, :]

        centroid_iteration_16[i, j, :] = centroid
        distance[:, j] = compute_distance(feature, centroid)

    label_feature_16 = compute_label(distance)

    loss = np.mean(np.min(distance**2, axis = 1))
    loss_iteration_16[i] = loss
#
# ++++++++++++++++++++++++++++++++++++++++++++++++
```

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
    #



    initial = np.row_stack((centroid_iteration_02[0,0,:],centroid_iteration_02[0,1,:]))
    final = np.row_stack((centroid_iteration_02[-1,0,:],centroid_iteration_02[-1,1,:]))
    plt.plot(centroid_iteration_02[:,0,0],centroid_iteration_02[:,0,1],'-', label = 'cluster=
0')
    plt.plot(centroid_iteration_02[:,1,0],centroid_iteration_02[:,1,1],'-', label = 'cluster=
1')
    plt.plot(initial[:,0], initial[:,1], 'bo', label = 'initial')
    plt.plot(final[:,0], final[:,1], 'rs', label = 'final')

    plt.legend()

    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_10():

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

    # ++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    k = 4
    initial = np.zeros((k,2))
    final = np.zeros((k,2))
    for i in range(k):
        initial[i] = centroid_iteration_04[0, i, :]
        final[i] = centroid_iteration_04[-1, i, :]

    for i in range(k):
        plt.plot(centroid_iteration_04[:, i, 0], centroid_iteration_04[:, i, 1], '-', label =
'cluster={0}'.format(i))
    plt.plot(initial[:,0], initial[:,1], 'bo', label = 'initial')
    plt.plot(final[:,0], final[:,1], 'rs', label = 'final')

    plt.legend()

    plt.tight_layout()
    plt.show()

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

In [ ]:

```
def function_result_11():

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

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

    k = 8
    initial = np.zeros((k,2))
    final = np.zeros((k,2))
    for i in range(k):
        initial[i] = centroid_iteration_08[0, i, :]
        final[i] = centroid_iteration_08[-1, i, :]

    for i in range(k):
        plt.plot(centroid_iteration_08[:, i, 0], centroid_iteration_08[:, i, 1], '-', label =
'cluster={0}'.format(i))
    plt.plot(initial[:,0], initial[:,1], 'bo', label = 'initial')
    plt.plot(final[:,0], final[:,1], 'rs', label = 'final')

    plt.legend()

    plt.tight_layout()
    plt.show()
    # ++++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_12():

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

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

    k = 16
    initial = np.zeros((k,2))
    final = np.zeros((k,2))
    for i in range(k):
        initial[i] = centroid_iteration_16[0, i, :]
        final[i] = centroid_iteration_16[-1, i, :]

    for i in range(k):
        plt.plot(centroid_iteration_16[:, i, 0], centroid_iteration_16[:, i, 1], '-', label =
'cluster={0}'.format(i))
    plt.plot(initial[:,0], initial[:,1], 'bo', label = 'initial')
    plt.plot(final[:,0], final[:,1], 'rs', label = 'final')
    plt.xlim(-14, 6.5)

    plt.legend(loc ='upper left')

    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_13():

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

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

    k = 2
    plt.scatter(x, y, c=label_feature_02, cmap=plt.cm.get_cmap('jet',k))

    cbar = plt.colorbar(label = 'cluster', ticks = np.linspace(0.25, 0.75, k))
    label = list(map(str, range(k)))
    cbar.set_ticklabels(label)

    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_14():

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

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

    k = 4
    plt.scatter(x, y, c=label_feature_04, cmap=plt.cm.get_cmap('jet',k))

    cbar = plt.colorbar(label = 'cluster', ticks = np.linspace(0.4, 2.6, k))
    label = list(map(str, range(k)))
    cbar.set_ticklabels(label)

    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

```
def function_result_15():

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

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

    k = 8
    plt.scatter(x, y, c=label_feature_08, cmap=plt.cm.get_cmap('jet',k))

    cbar = plt.colorbar(label = 'cluster', ticks = np.linspace(0.4, 6.6, k))
    label = list(map(str, range(k)))
    cbar.set_ticklabels(label)

    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++
```

In [ ]:

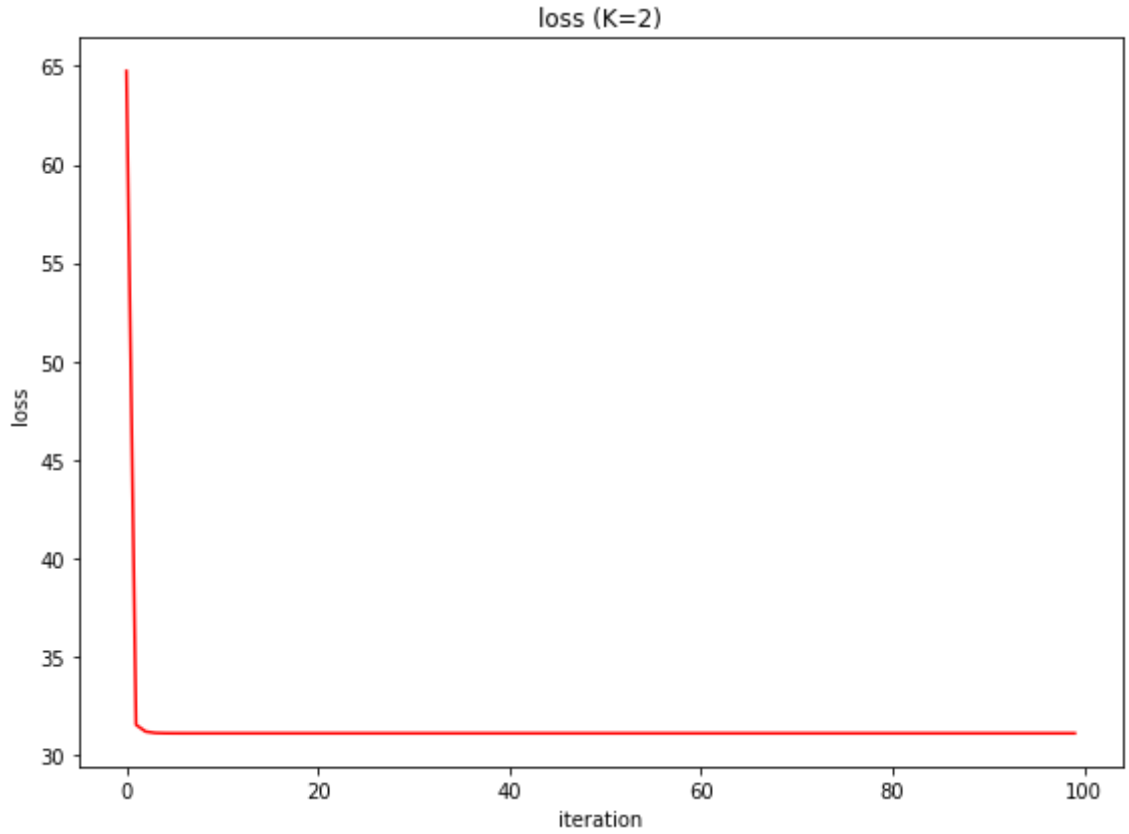
```
def function_result_16():

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

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

    k = 16
    plt.scatter(x, y, c=label_feature_16, cmap=plt.cm.get_cmap('jet',k))

    cbar = plt.colorbar(label = 'cluster', ticks = np.linspace(0.45, 14.55, k))
    label = list(map(str, range(k)))
    cbar.set_ticklabels(label)

    plt.tight_layout()
    plt.show()
    #
    # ++++++++++++++++++++++++++++++++++++++++++++
```

results

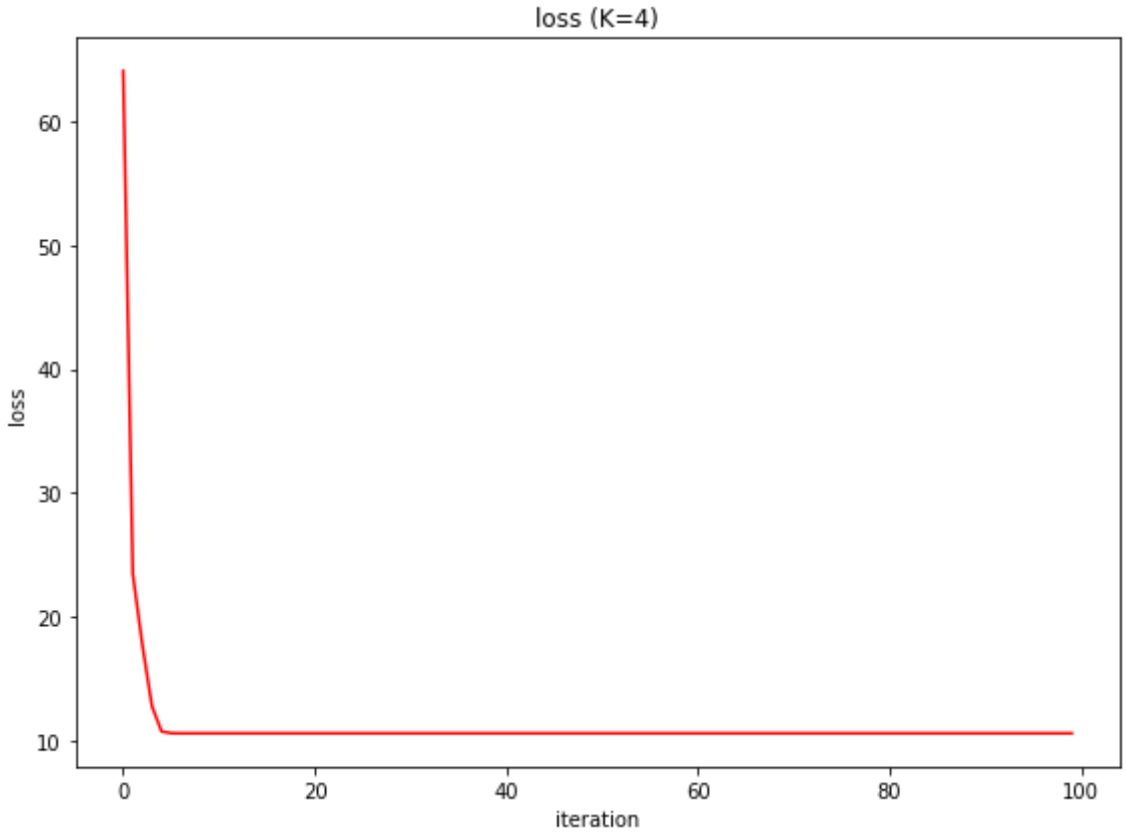
In [ ]:

```
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)
```

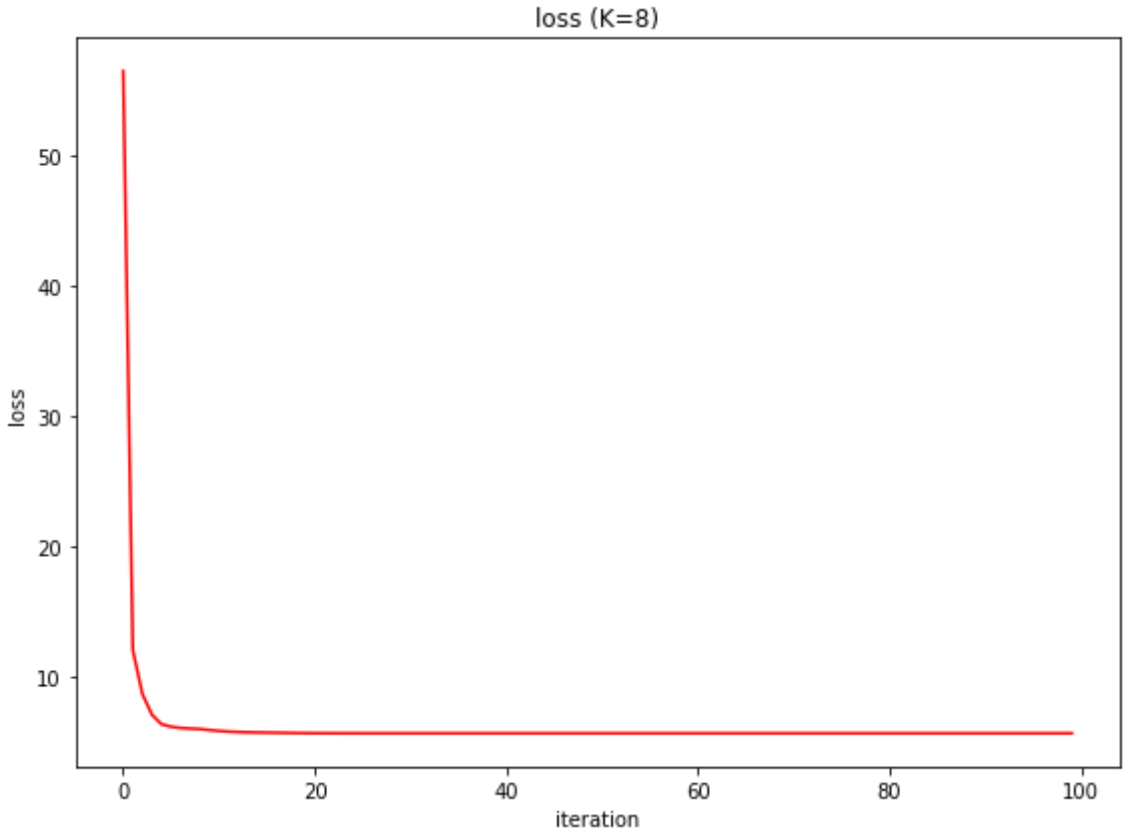
```
**************************************************
## [RESULT 01]
**************************************************
final loss (K=2) = 31.1123356206
**************************************************
## [RESULT 02]
**************************************************
final loss (K=4) = 10.5831291650
**************************************************
## [RESULT 03]
**************************************************
final loss (K=8) =  5.6790749344
**************************************************
## [RESULT 04]
**************************************************
final loss (K=16) =  3.0322223594
**************************************************
## [RESULT 05]
**************************************************
```



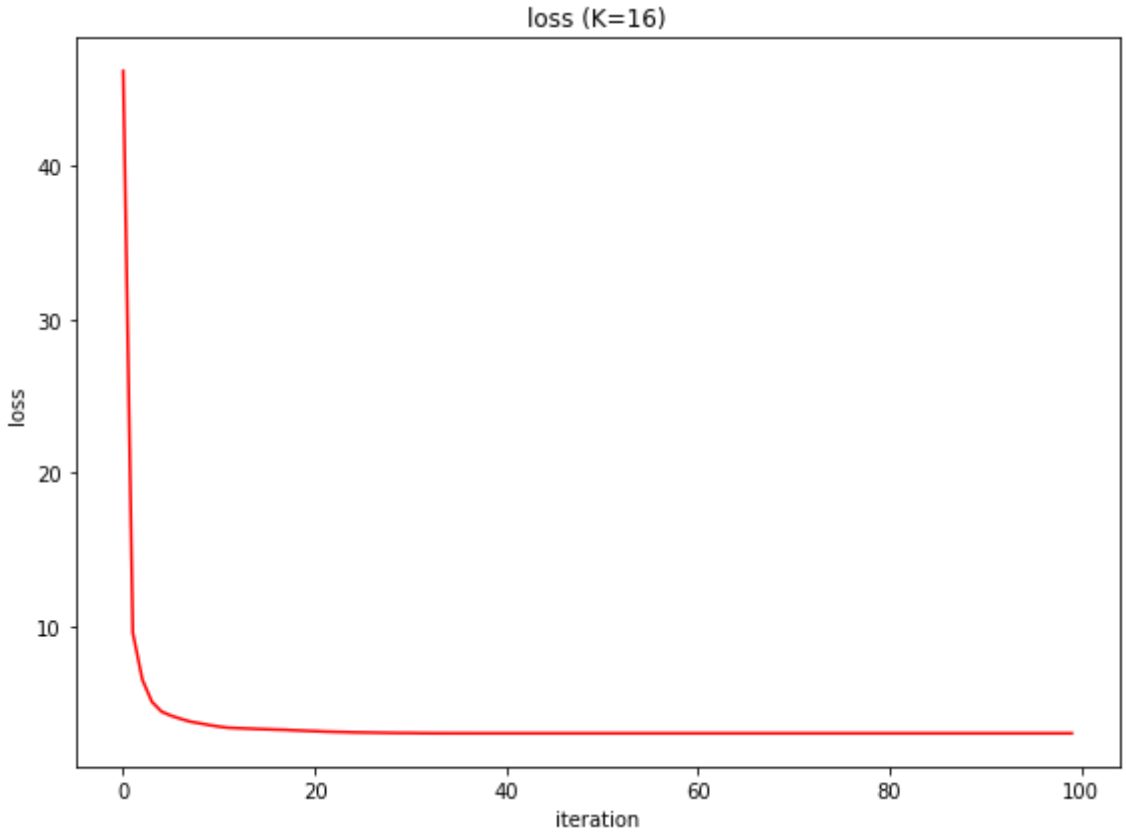

```
**************************************************
## [RESULT 06]
**************************************************
```



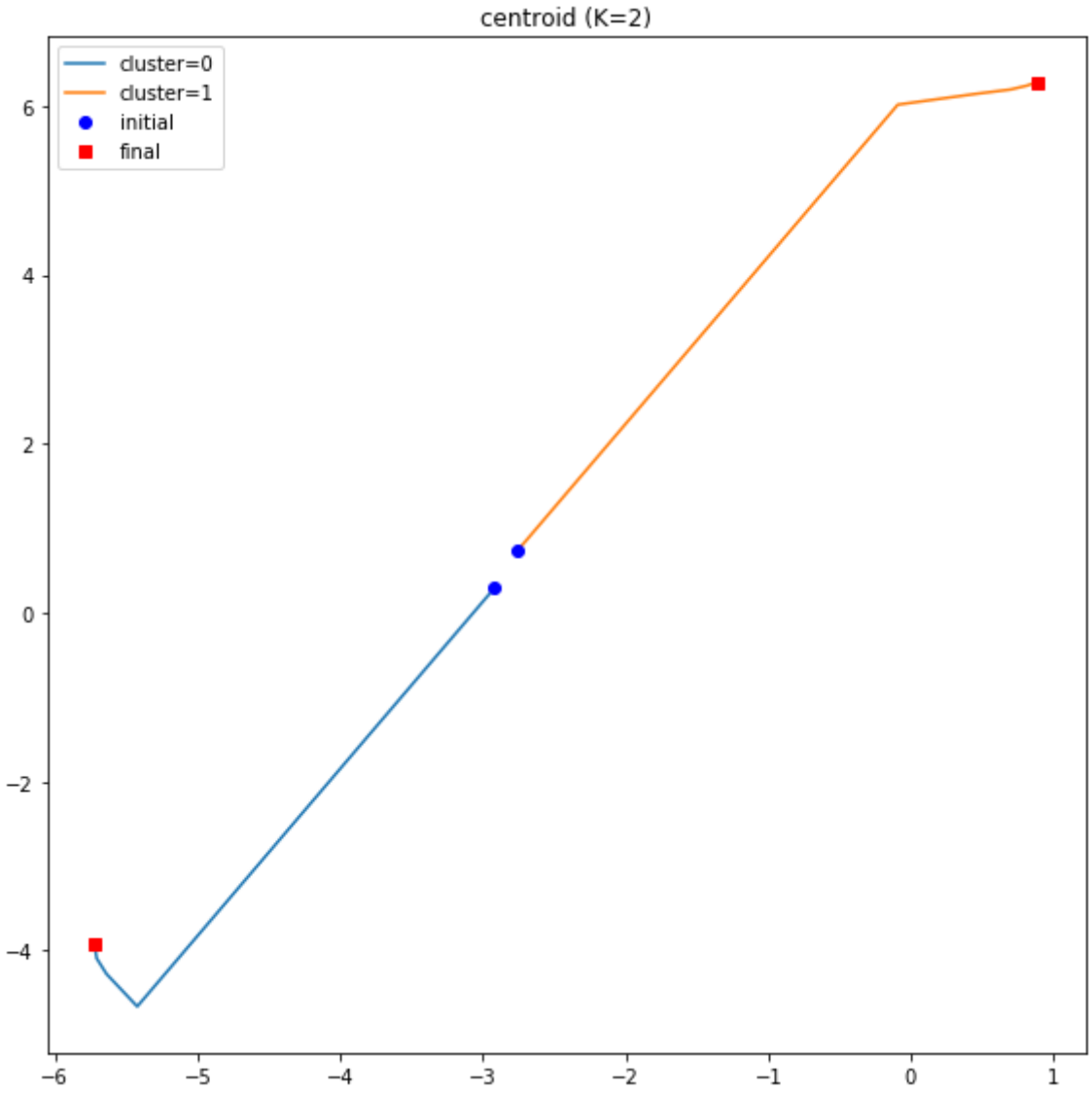

```
**************************************************
## [RESULT 07]
**************************************************
```



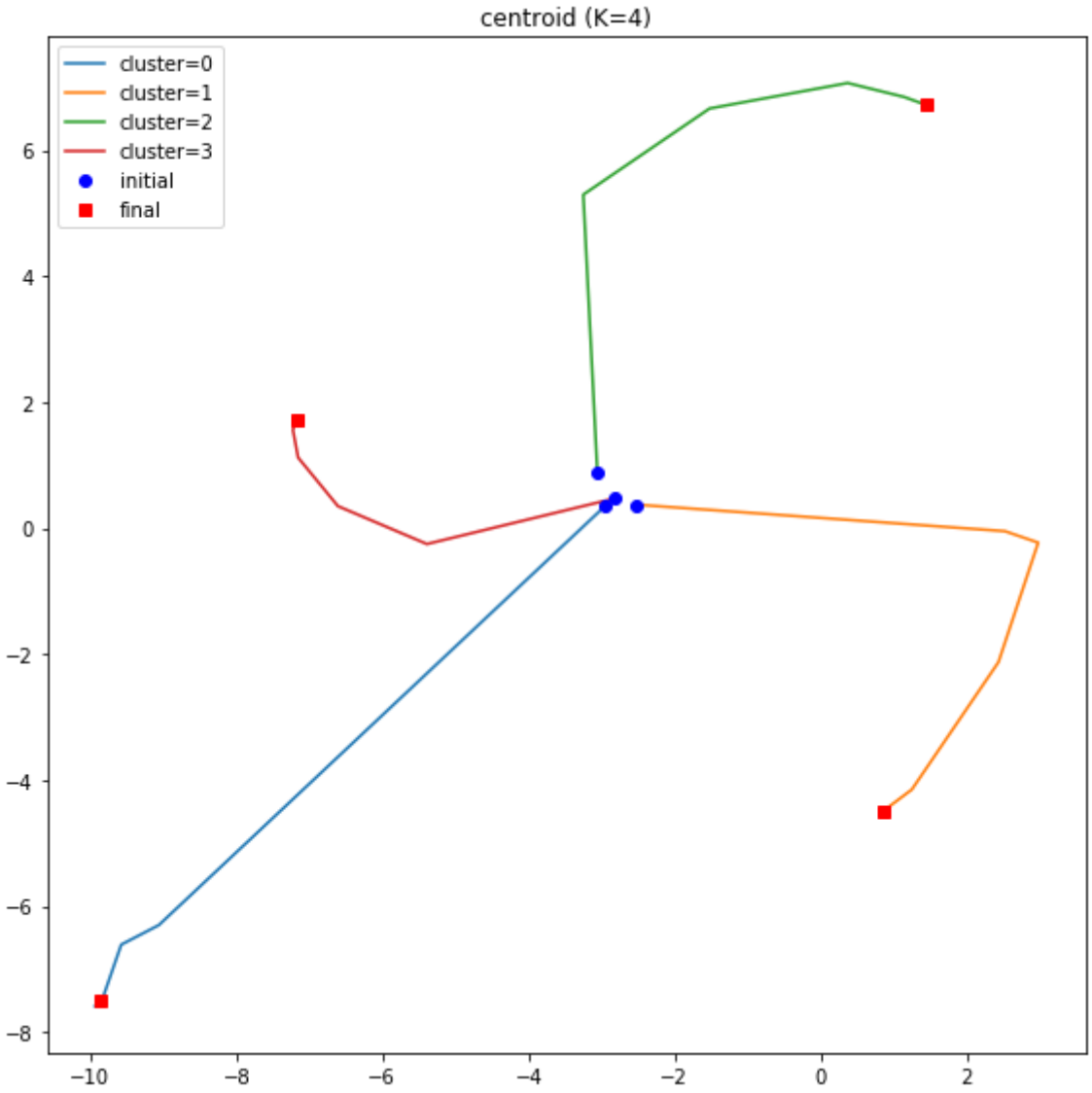

```
**************************************************
## [RESULT 08]
**************************************************
```



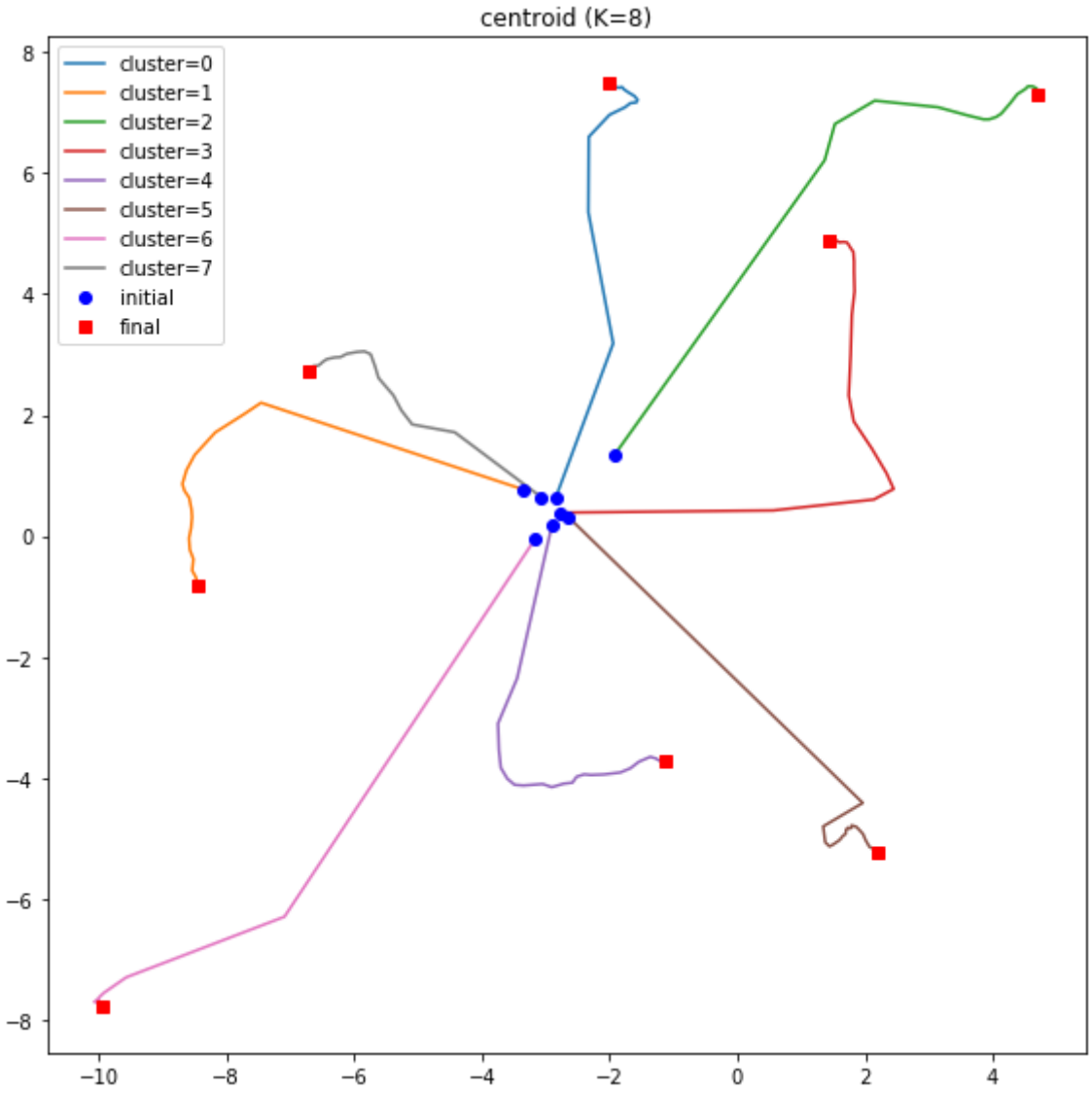

```
**************************************************
## [RESULT 09]
**************************************************
```


centroid (K=2)
cluster=0
cluster=1
initial
final
6
4
2
0
−2
−4
−6
−5
−4
−3
−2
−1
0
1

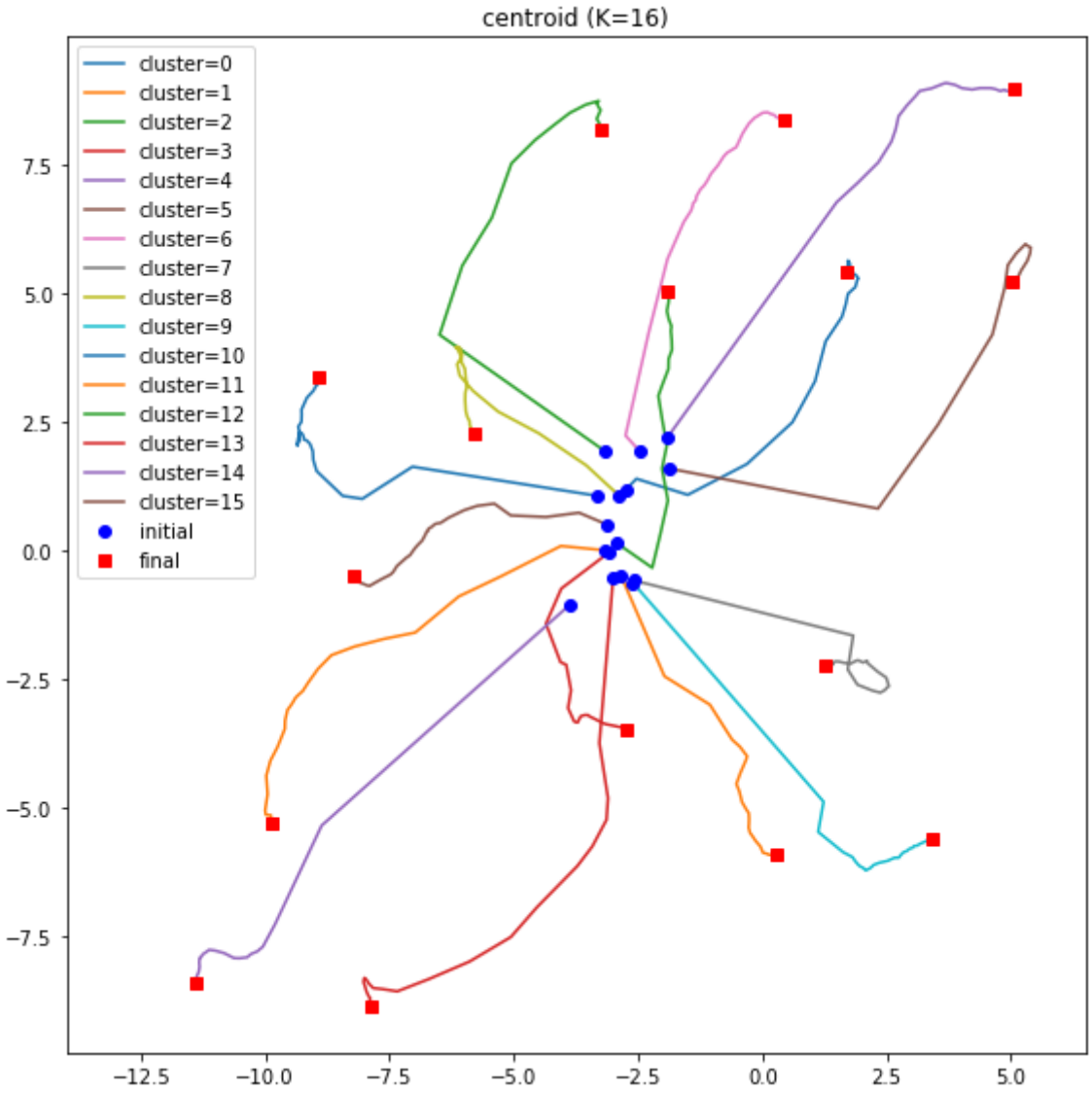
```
**************************************************
## [RESULT 10]
**************************************************
```


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

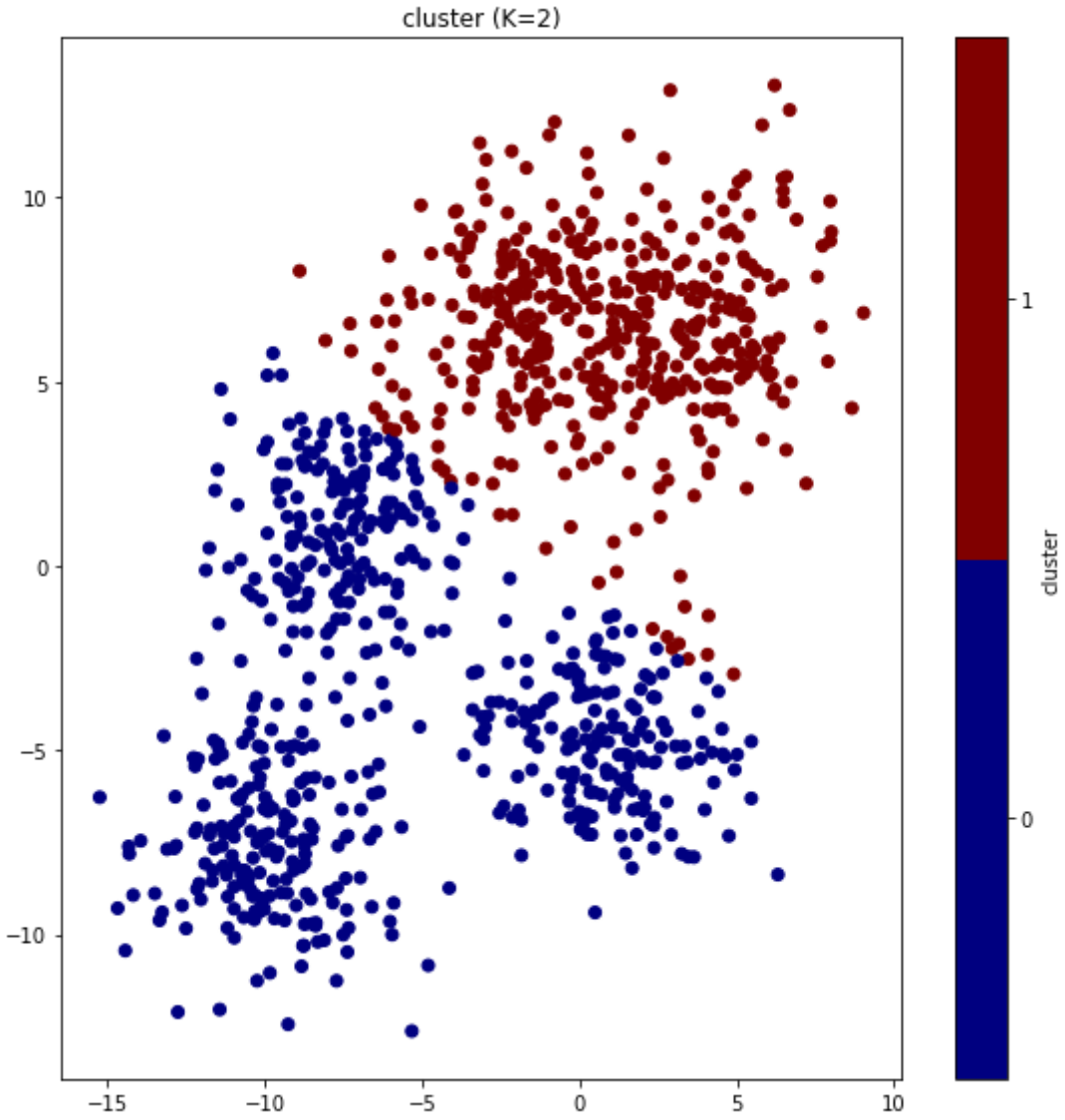
```
**************************************************
## [RESULT 11]
**************************************************
```


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

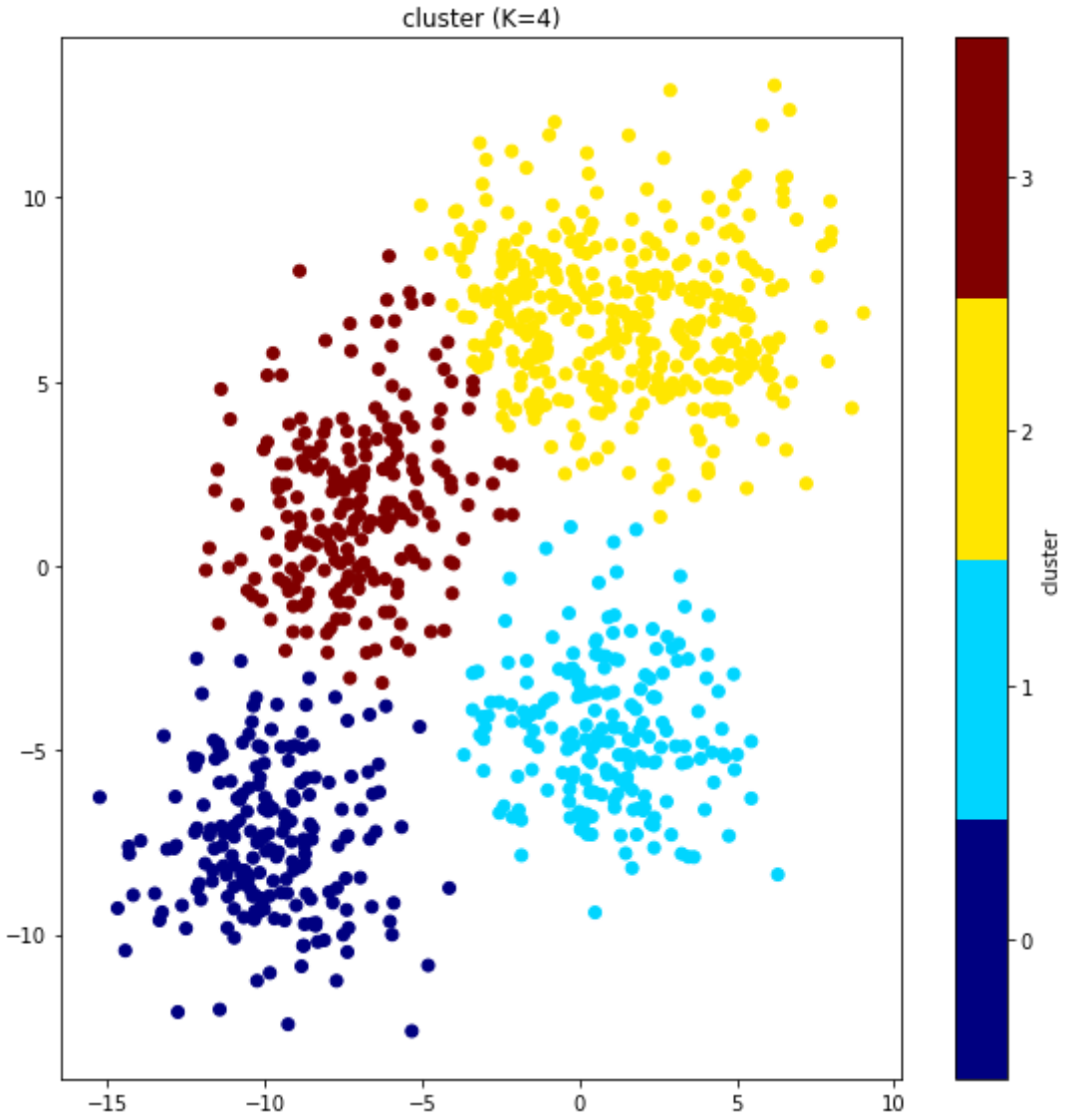
```
**************************************************
## [RESULT 12]
**************************************************
```


centroid (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
initial
final
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0

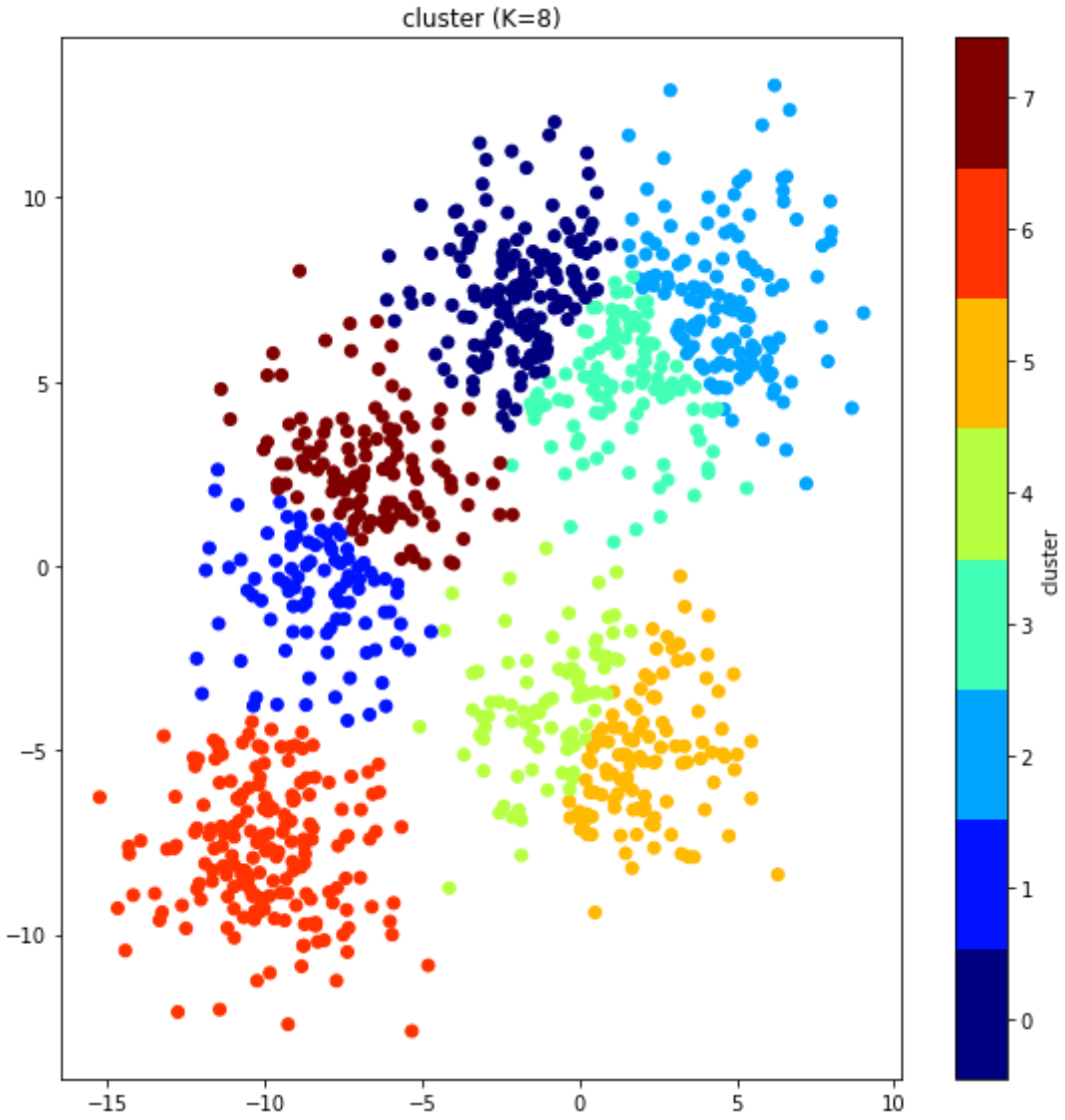
```
**************************************************
## [RESULT 13]
**************************************************
```



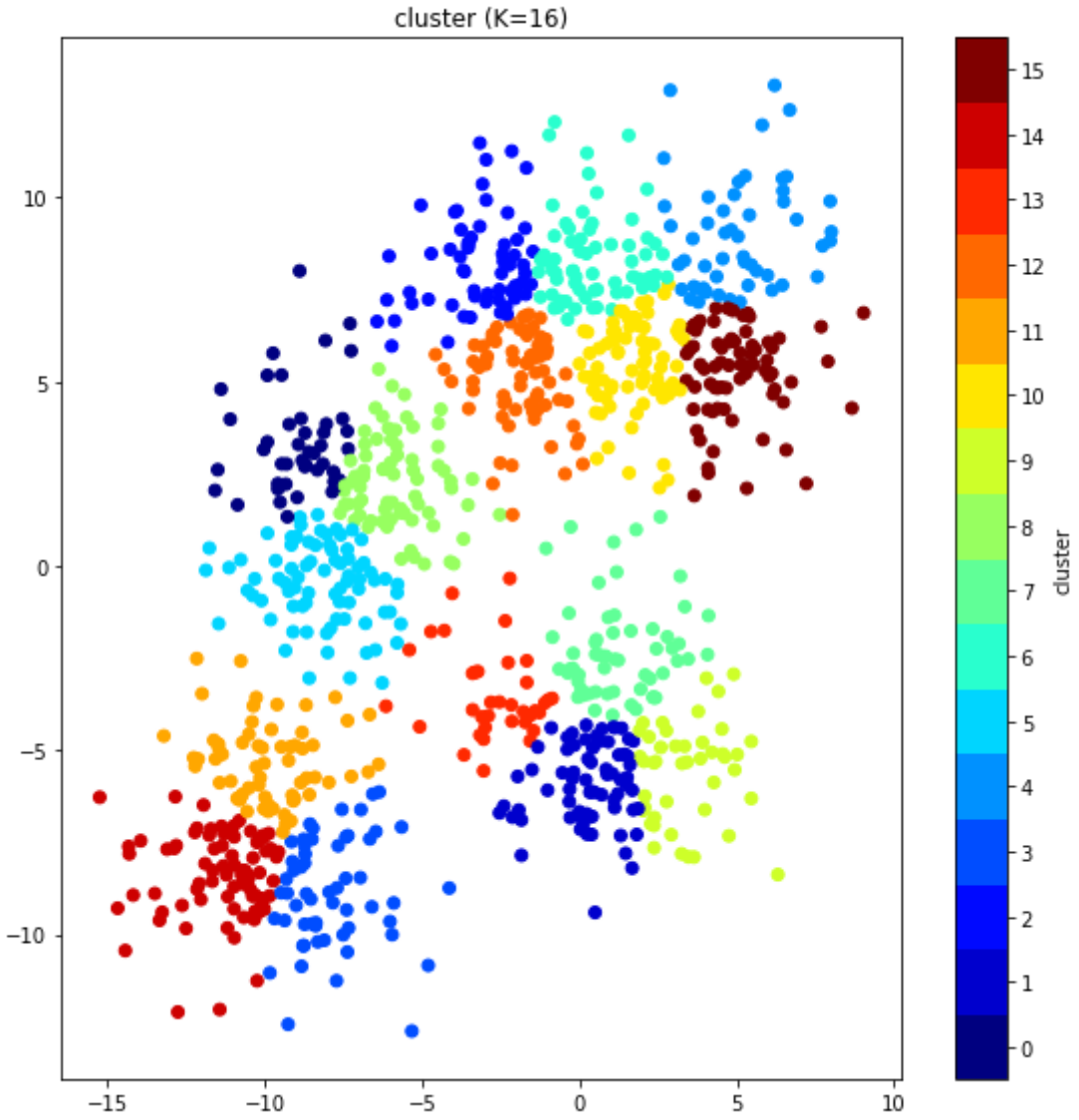

```
**************************************************
## [RESULT 14]
**************************************************
```




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




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


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

In [ ]: