Clustering jerárquico y dendrogramas

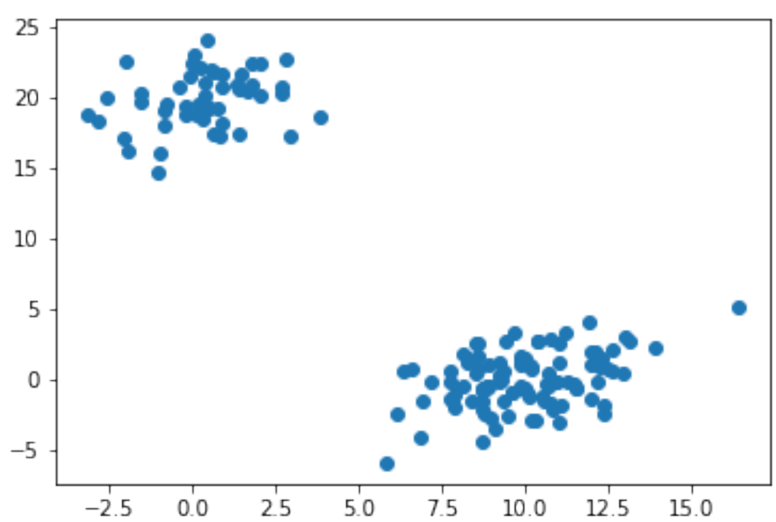
Antes de empezar, pongamos un poco de notación para hablar todos el mismo idioma

- X dataset (array de n x m) de puntos a clusterizar
- n número de datos
- m número de rasgos
- Z array de enlace del cluster con la información de las uniones
- k número de clusters

In [1]:
```
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram,linkage
import numpy as np
```

In [2]:
```
np.random.seed(4711)
a = np.random.multivariate_normal([10,0],[[3,1],[1,4]], size = [100,])
b = np.random.multivariate_normal([0,20], [[3,1],[1,4]], size = [50,])
X = np.concatenate((a,b))
print(X.shape)
plt.scatter(X[:,0], X[:,1])
plt.show()
```

```
(150, 2)
```



In [3]:
```
Z = linkage(X, "ward")
```

In [4]:
```
from scipy.cluster.hierarchy import cophenet
from scipy.spatial.distance import pdist
```

In [5]:
```
c, coph_dist = cophenet(Z, pdist(X))
c
```

Out[5]:
```
0.9800148387574268
```

In [6]:
```
Z[0]
```

Out[6]:
```
array([5.20000000e+01, 5.30000000e+01, 4.15105485e-02, 2.00000000e+00
])
```

In [7]:
```
Z[1]
```

Out[7]:
```
array([1.40000000e+01, 7.90000000e+01, 5.91375926e-02, 2.00000000e+00
])
```

In [8]:
```
Z[:20]
```

Out[8]:
```
array([[5.20000000e+01, 5.30000000e+01, 4.15105485e-02, 2.00000000e+0
0],
       [1.40000000e+01, 7.90000000e+01, 5.91375926e-02, 2.00000000e+0
0],
       [3.30000000e+01, 6.80000000e+01, 7.10677929e-02, 2.00000000e+0
0],
       [1.70000000e+01, 7.30000000e+01, 7.13712071e-02, 2.00000000e+0
0],
       [1.00000000e+00, 8.00000000e+00, 7.54313099e-02, 2.00000000e+0
0],
       [8.50000000e+01, 9.50000000e+01, 1.09277896e-01, 2.00000000e+0
0],
       [1.08000000e+02, 1.31000000e+02, 1.10071548e-01, 2.00000000e+0
0],
       [9.00000000e+00, 6.60000000e+01, 1.13022407e-01, 2.00000000e+0
0],
       [1.50000000e+01, 6.90000000e+01, 1.14289714e-01, 2.00000000e+0
0],
       [6.30000000e+01, 9.80000000e+01, 1.21200766e-01, 2.00000000e+0
0],
       [1.07000000e+02, 1.15000000e+02, 1.21671017e-01, 2.00000000e+0
0],
       [6.50000000e+01, 7.40000000e+01, 1.24900190e-01, 2.00000000e+0
0],
       [5.80000000e+01, 6.10000000e+01, 1.40277358e-01, 2.00000000e+0
0],
       [6.20000000e+01, 1.52000000e+02, 1.72599535e-01, 3.00000000e+0
0],
       [4.10000000e+01, 1.58000000e+02, 1.77901377e-01, 3.00000000e+0
0],
       [1.00000000e+01, 8.30000000e+01, 1.86354938e-01, 2.00000000e+0
0],
       [1.14000000e+02, 1.39000000e+02, 2.04186147e-01, 2.00000000e+0
0],
       [3.90000000e+01, 8.80000000e+01, 2.06282849e-01, 2.00000000e+0
0],
       [7.00000000e+01, 9.60000000e+01, 2.19312547e-01, 2.00000000e+0
0],
       [4.60000000e+01, 5.00000000e+01, 2.20492804e-01, 2.00000000e+0
0]])
```

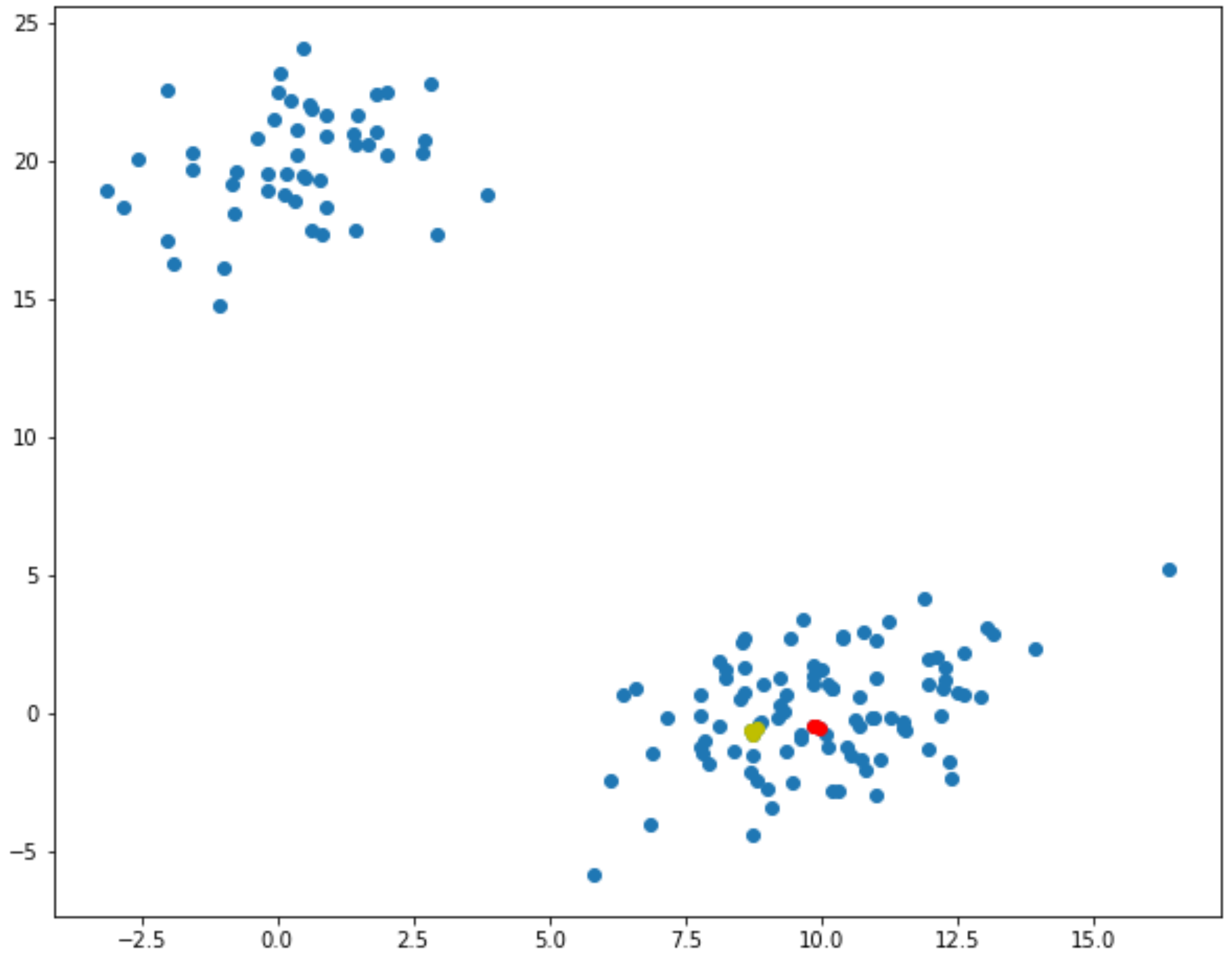
In [9]:
```
print(Z[152-len(X)])# cluster 152
print(Z[158-len(X)])#cluster 158
```

```
[33.         68.          0.07106779  2.        ]
[15.         69.          0.11428971  2.        ]
```

In [10]:
```
X[[33,62,68]]
```

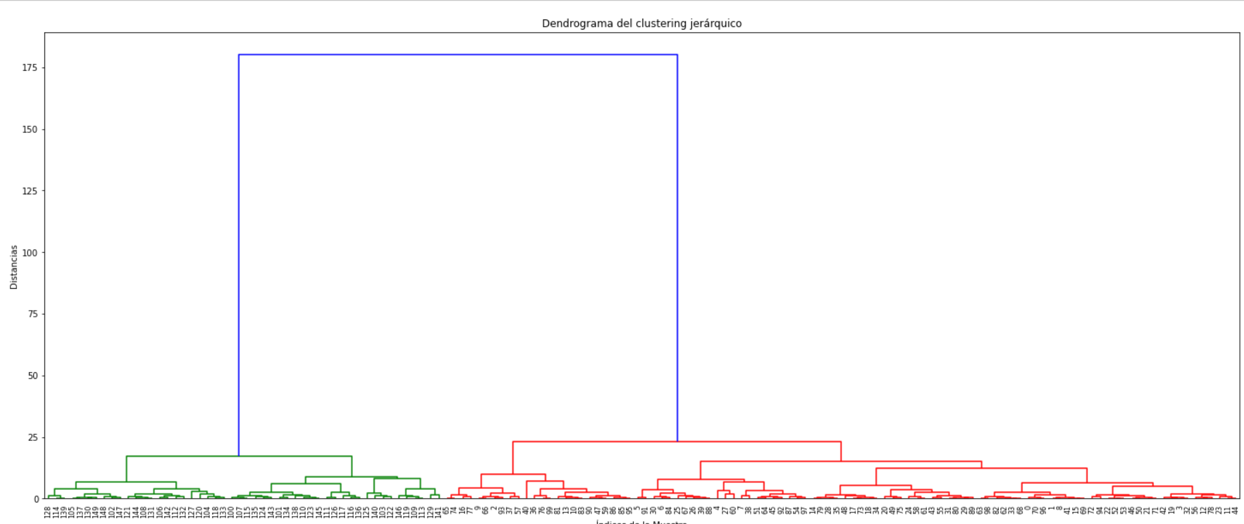
Out[10]:
```
array([[ 9.83913054, -0.48729797],
       [ 9.97792822, -0.56383202],
       [ 9.8934927 , -0.44152257]])
```

In [11]:
```
idx = [33,62,68]
idx2 = [15,69,41]
plt.figure(figsize=(10,8))
plt.scatter(X[:,0], X[:,1])##pintar todos los puntos
plt.scatter(X[idx,0], X[idx,1], c='r')##destacamos en rojo los puntos
plt.scatter(X[idx2,0], X[idx2,1], c='y')##destacamos en amarillo el se
plt.show()
```



Representación gráfica de un dendrograma

In [12]:
```
plt.figure(figsize=(25,10))
plt.title("Dendrograma del clustering jerárquico")
plt.xlabel("Índices de la Muestra")
plt.ylabel("Distancias")
dendrogram(Z, leaf_rotation=90., leaf_font_size=8.0,
           color_threshold=0.7*180)
plt.show()
```

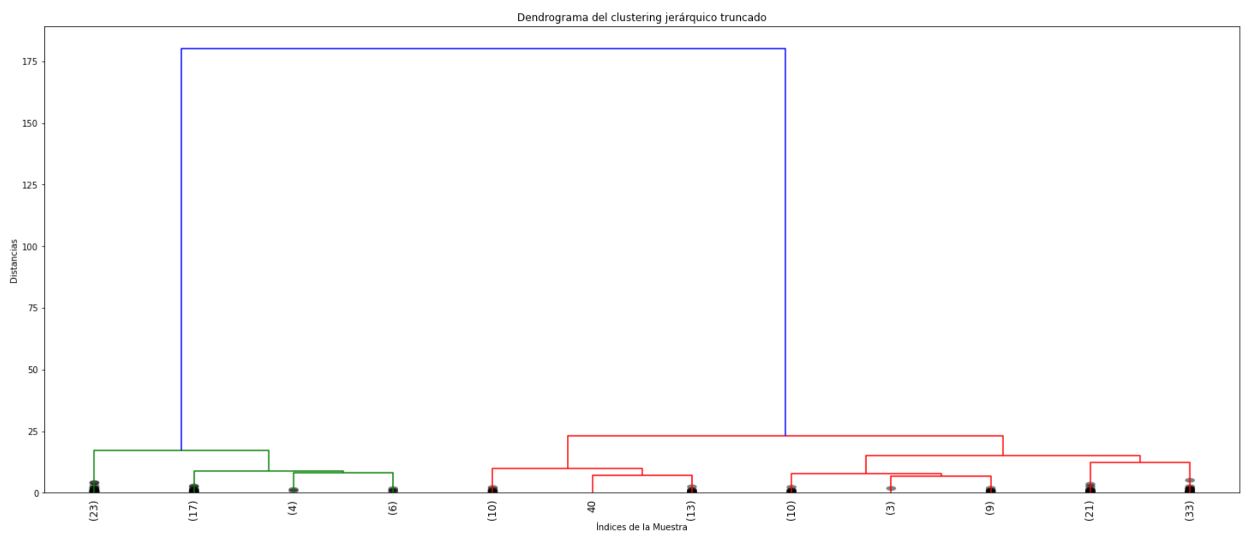


In [13]:
```
Z[-4:,]
```

Out[13]:
```
array([[290.        , 294.        ,  15.11533118,  76.        ],
       [287.        , 292.        ,  17.11527362,  50.        ],
       [293.        , 295.        ,  23.12198936, 100.        ],
       [296.        , 297.        , 180.27043021, 150.        ]])
```

Truncar el dendrograma

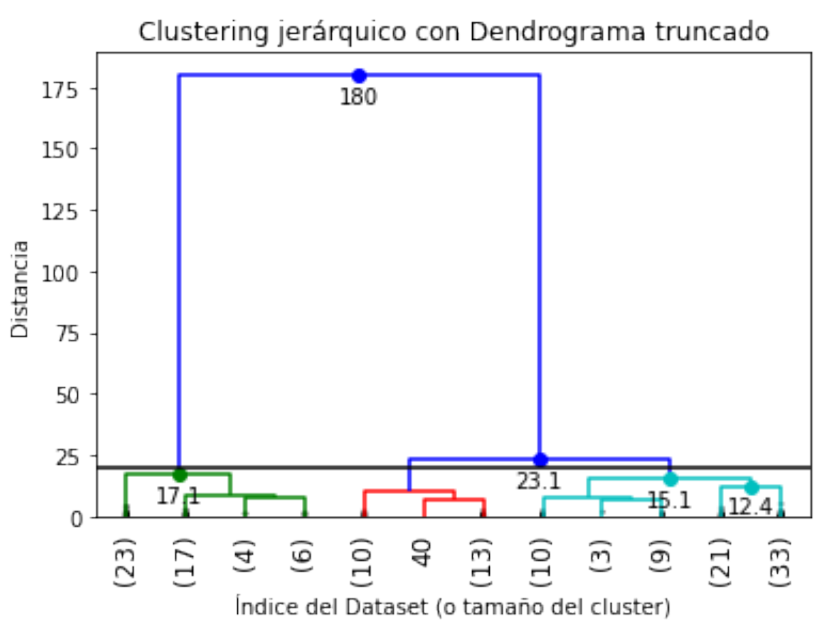
In [14]:
```
plt.figure(figsize=(25,10))
plt.title("Dendrograma del clustering jerárquico truncado")
plt.xlabel("Índices de la Muestra")
plt.ylabel("Distancias")
dendrogram(Z, leaf_rotation=90., leaf_font_size=12.0,
           color_threshold=0.7*180,
           truncate_mode="lastp", p=12, show_leaf_counts=True,
           show_contracted=True,)
plt.show()
```



Dendrograma tuneado

In [15]:
```
def dendrogram_tune(*args, **kwargs):

    max_d=kwargs.pop("max_d", None)
    if max_d and 'color_threshold' not in kwargs:
        kwargs['color_threshold'] = max_d
    annotate_above = kwargs.pop('annotate_above', 0)

    ddata = dendrogram(*args,**kwargs)

    if not kwargs.get('no_plot', False):
        plt.title("Clustering jerárquico con Dendrograma truncado")
        plt.xlabel("Índice del Dataset (o tamaño del cluster)")
        plt.ylabel("Distancia")
        for i, d, c in zip(ddata['icoord'], ddata['dcoord'],
                               ddata['color_list']):
            x = 0.5 * sum(i[1:3])
            y = d[1]
            if y>annotate_above:
                plt.plot(x,y,'o',c=c)
                plt.annotate('%.3g'%y, (x,y), xytext=(0,-5),
                            textcoords="offset points", va="top",
                             ha="center")

    if max_d:
        plt.axhline(y=max_d, c='k')

    return ddata
```

In [16]:
```
dendrogram_tune(Z,truncate_mode='lastp',p=12, leaf_rotation=90.,
                leaf_font_size=12.,
                show_contracted=True,annotate_above=10, max_d=20)
plt.show()
```



Corte automático del dendrograma

inconsistency_i = (h_i-avg(h_j))/std(h_j)

In [17]:
```
from scipy.cluster.hierarchy import inconsistent
```

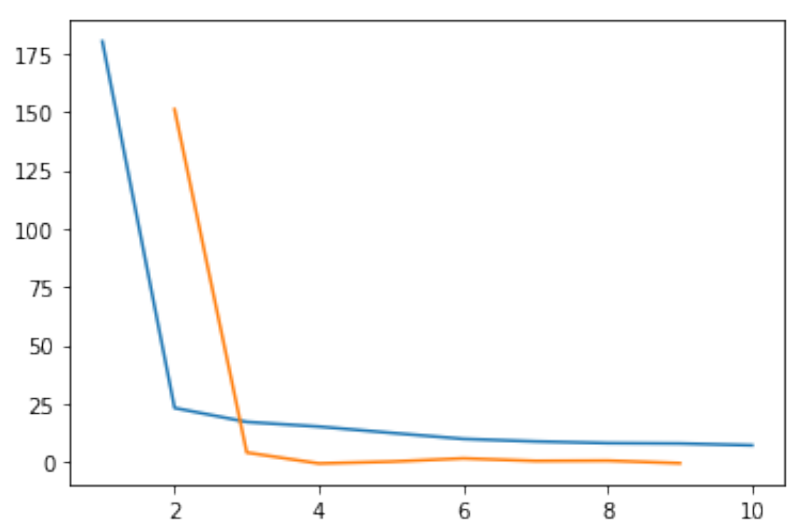
In [18]:
```
depth = 3
incons = inconsistent(Z, depth)
incons[-10:]
```

Out[18]:
```
array([[ 3.63777835,  2.5556114 ,  4.        ,  1.35908084],
       [ 3.89767268,  2.57216151,  7.        ,  1.54388156],
       [ 3.05885714,  2.66707272,  6.        ,  1.87115096],
       [ 4.92746418,  2.73259589,  7.        ,  1.39821573],
       [ 4.76943311,  3.16276553,  6.        ,  1.60455941],
       [ 5.27287862,  3.56604844,  7.        ,  2.00627335],
       [ 8.22057081,  4.07583053,  7.        ,  1.69162096],
       [ 7.83287032,  4.46681266,  7.        ,  2.07808207],
       [11.38091435,  6.29430022,  7.        ,  1.86535033],
       [37.25844589, 63.31539362,  7.        ,  2.25872377]])
```

Método del codo

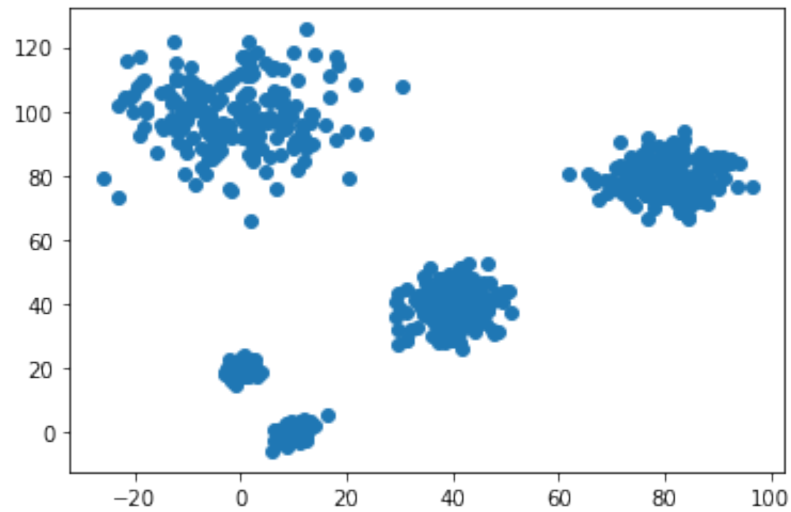
In [19]:
```
last = Z[-10:,2]
last_rev = last[::-1]
print(last_rev)
idx = np.arange(1, len(last)+1)
plt.plot(idx, last_rev)

acc = np.diff(last,2)
acc_rev = acc[::-1]
plt.plot(idx[:-2]+1, acc_rev)
plt.show()
k = acc_rev.argmax() +2
print("El número óptimo de cluster es %s"%str(k))
```

```
[180.27043021  23.12198936  17.11527362  15.11533118  12.42734657
   9.84427829   8.74822275   8.04935282   7.86878542   7.11106083]
```



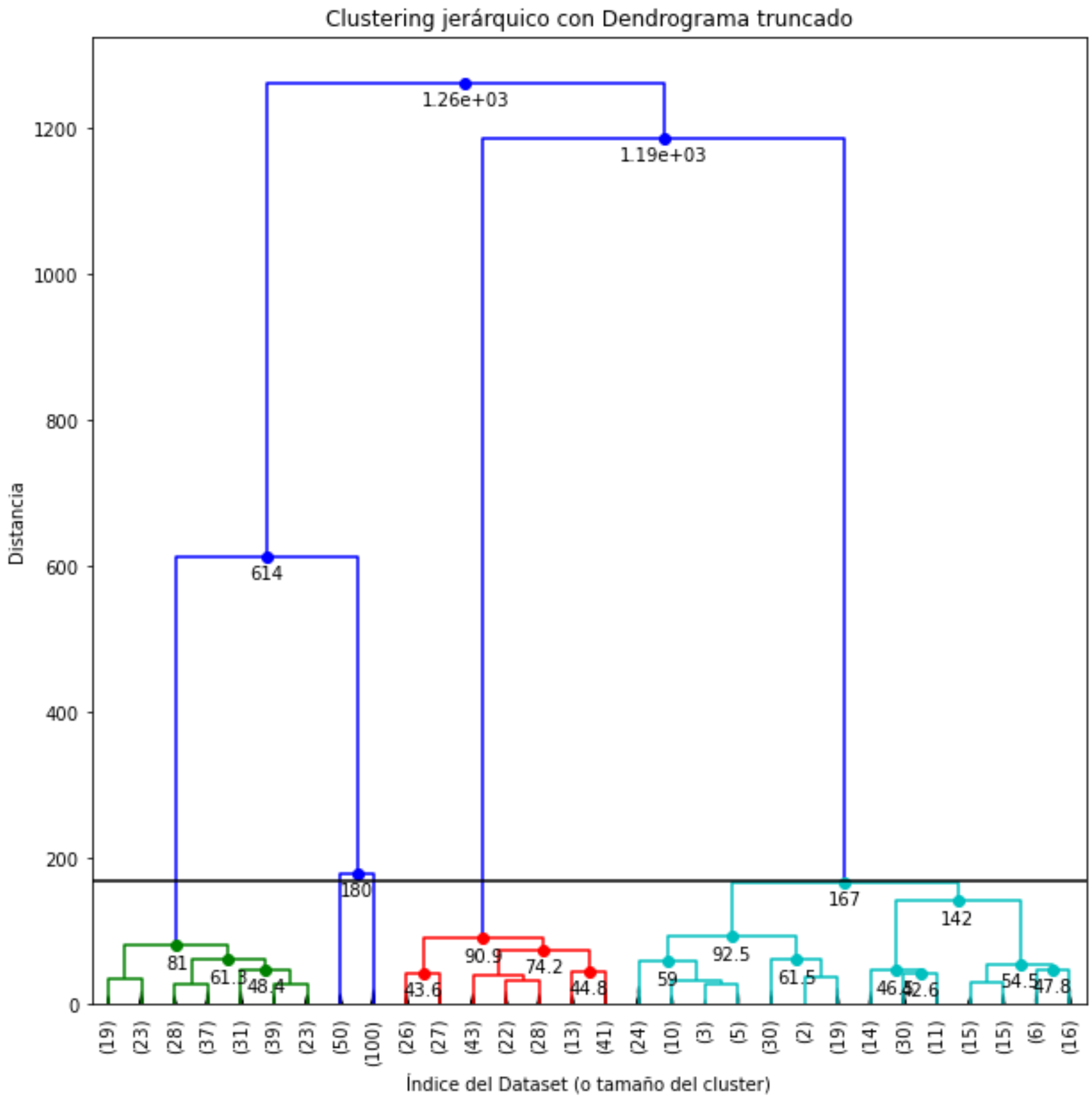

```
El número óptimo de cluster es 2
```

In [20]:
```
c = np.random.multivariate_normal([40,40],[[20,1],[1,30]], size=[200,]
d = np.random.multivariate_normal([80,80],[[30,1],[1,30]], size=[200,]
e = np.random.multivariate_normal([0,100],[[100,1],[1,100]],
                                  size=[200,])
X2 = np.concatenate((X,c,d,e),)
plt.scatter(X2[:,0], X2[:,1])
plt.show()
```



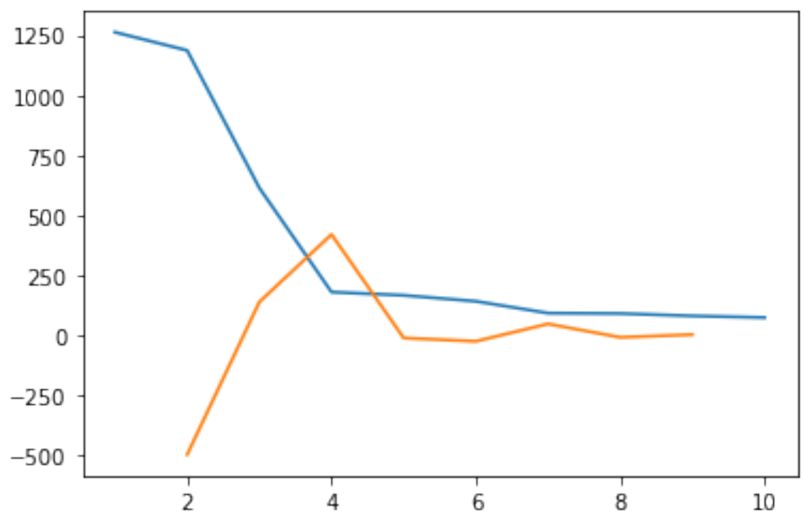
In [21]:
```
Z2 = linkage(X2,"ward")
```

In [22]:
```python
plt.figure(figsize=(10,10))
dendrogram_tune(
    Z2,
    truncate_mode="lastp",
    p=30,
    leaf_rotation=90.,
    leaf_font_size=10.,
    show_contracted=True,
    annotate_above = 40,
    max_d = 170
)

plt.show()
```



In [23]:
```
last = Z2[-10:,2]
last_rev = last[::-1]
print(last_rev)
idx = np.arange(1, len(last)+1)
plt.plot(idx, last_rev)

acc = np.diff(last,2)
acc_rev = acc[::-1]
plt.plot(idx[:-2]+1, acc_rev)
plt.show()
k = acc_rev.argmax() +2
print("El número óptimo de cluster es %s"%str(k))
```

```
[1262.52130994 1186.7588235   614.06504667  180.27043021  166.6643465
8
  141.92437181   92.54599212   90.91214341   80.96733501   74.1701531
2]
```




```
El número óptimo de cluster es 4
```

In [24]:
```
print(inconsistent(Z2, 5)[-10:])
```

```
[[ 13.99221995  15.56655759  30.           3.8658472 ]
 [ 16.73940735  18.56390061  30.           3.45982932]
 [ 19.05945013  20.53210626  31.           3.49952861]
 [ 19.25573887  20.8265769   29.           3.51907342]
 [ 21.36116189  26.77659523  30.           4.50255938]
 [ 36.58100874  37.08602393  31.           3.50761079]
 [ 12.12200256  32.15467931  30.           5.22936105]
 [ 42.61369802 111.38576865  31.           5.13038026]
 [ 81.75198678 208.31582073  31.           5.30447871]
 [147.25602023 307.95700562  31.           3.62149673]]
```

Recuperar los clusters y sus elementos

In [25]:
```
from scipy.cluster.hierarchy import fcluster
```

In [26]:
```python
max_d=25
clusters = fcluster(Z, max_d, criterion="distance")
clusters
```

Out[26]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=i
nt32)
```

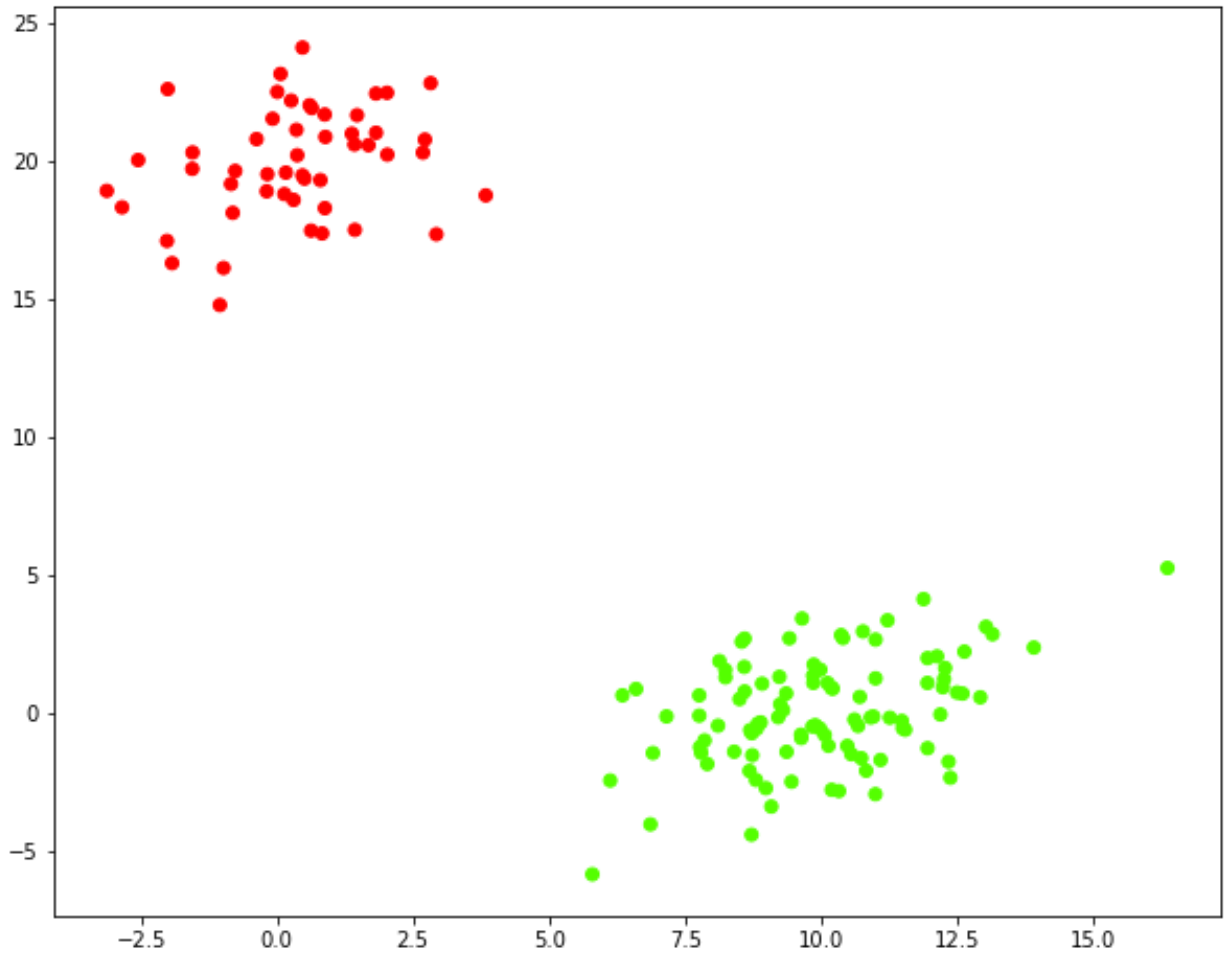
In [27]:
```python
k=2
clusters = fcluster(Z, k, criterion="maxclust")
clusters
```

Out[27]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=i
nt32)
```

In [28]:
```python
fcluster(Z, 8, depth=10)
```

Out[28]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=i
nt32)
```

In [29]:
```
plt.figure(figsize=(10,8))
plt.scatter(X[:,0], X[:,1], c = clusters, cmap="prism")
plt.show()
```



In [30]:
```
max_d=170
clusters = fcluster(Z2, max_d, criterion="distance")
clusters

plt.figure(figsize=(10,8))
plt.scatter(X2[:,0], X2[:,1], c = clusters, cmap="prism")
plt.show()
```