作業三 實作 knn 做資料分類

實作 knn 做資料分類：
利用 sklearn.neighbors 套件
參考 https://scikit-learn.org/stable/modules/neighbors.html 網站
亂數產生 20 個二維平面上的點，
顯示利用 knn 分類後的 indices 與 sparse graph。
擷取執行畫面與程式碼並送出。

參考範例

1.6.1. Unsupervised Nearest Neighbors

`NearestNeighbors` implements unsupervised nearest neighbors learning. It acts as a uniform interface to three different nearest neighbors algorithms: `BallTree`, `KDTree`, and a brute-force algorithm based on routines in `sklearn.metrics.pairwise`. The choice of neighbors search algorithm is controlled through the keyword `'algorithm'`, which must be one of `['auto', 'ball_tree', 'kd_tree', 'brute']`. When the default value `'auto'` is passed, the algorithm attempts to determine the best approach from the training data. For a discussion of the strengths and weaknesses of each option, see Nearest Neighbor Algorithms.

Warning

Regarding the Nearest Neighbors algorithms, if two neighbors k+1 and k have identical distances but different labels, the result will depend on the ordering of the training data.

1.6.1.1.Finding the Nearest Neighbors

For the simple task of finding the nearest neighbors between two sets of data, the unsupervised algorithms within`sklearn.neighbors`can be used:

```
>>>
>>> from sklearn.neighbors import NearestNeighbors
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3,
2]])
```

```
>>> nbrs = NearestNeighbors(n_neighbors=2,
algorithm='ball_tree').fit(X)
>>> distances, indices = nbrs.kneighbors(X)
>>> indices
array([[0, 1],
       [1, 0],
       [2, 1],
       [3, 4],
       [4, 3],
       [5, 4]]...)
>>> distances
array([[0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356]])
```

Because the query set matches the training set, the nearest neighbor of each point is the point itself, at a distance of zero.

It is also possible to efficiently produce a sparse graph showing the connections between neighboring points:

```
>>>
>>> nbrs.kneighbors_graph(X).toarray()
array([[1., 1., 0., 0., 0., 0.],
       [1., 1., 0., 0., 0., 0.],
       [0., 1., 1., 0., 0., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 0., 1., 1.]])
```

The dataset is structured such that points nearby in index order are nearby in parameter space, leading to an approximately block-diagonal matrix of K-nearest neighbors. Such a sparse graph is useful in a variety of circumstances which make use of spatial relationships between points for unsupervised learning: in particular, see**`Isomap`**,**`LocallyLinearEmbedding`**, and**`SpectralClustering`**.