B093040024
郭晏涵

深度視覺

HW：使用 K-Nearest Neighbors 實現鳶尾花分類

程式執行：

Input

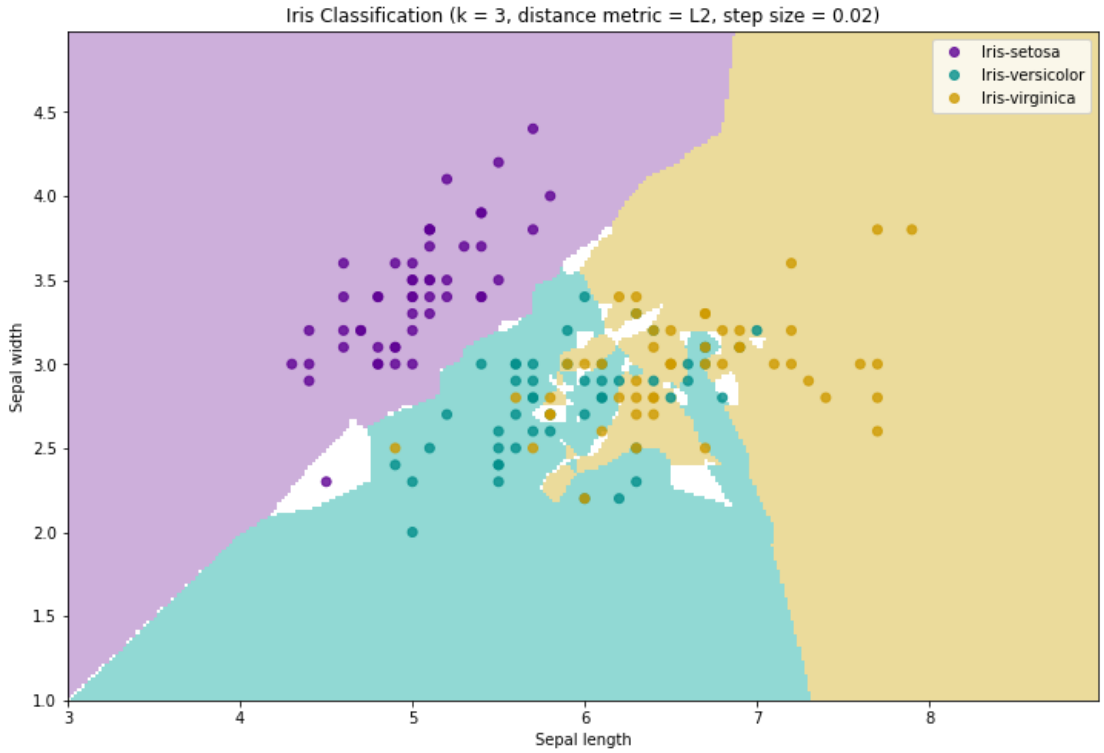
包含三筆輸入，第一與第二筆為選擇輸入的兩個鳶尾花參數（string），第三筆輸入為 K 值（int）。

Output

以圖表顯示預測結果。橫坐標為第一個參數，縱坐標為第二個參數。

```
p1 = input("attribute1:")
p2 = input("attribute2:")
k = int(input("nearest neighbors:"))
knn(p1, p2, k)
```

```
attribute1:Sepal length
attribute2:Petal width
nearest neighbors:3
```



演算法說明:

依照輸入的字串判斷選取的鳶尾花參數，選出所對應的資料點，接著依序計算資料點到地圖上點座標的距離，至少選取最近的 K 點，判斷該點為何種分類，最後將分類結果輸出。

程式碼說明:

```
cmap_bold = ListedColormap(['#5E0094', '#018F8B', '#CF9B00'])
cmap_light = ListedColormap(['#CDAFDB', '#91D9D4', '#EBDB9B', '#FFFFFF'])
h = 0.02
```

第一部分是設定各分類輸出的顏色，cmap_bold 為資料集（實心點）的顏色，cmap_light 為預測結果（底圖）的顏色，h 為預測座標點的間距，h 越小則預測結果越細緻，但計算所需的時間也越長。

```
def classify(t, X, Y, K):

    distance = list(map(lambda x, y: ((x - t[0]) ** 2 + (y - t[1]) ** 2) ** 0.5, iris_data[X], iris_data[Y]))

    nearest = heapq.nsmallest(K, enumerate(distance), key=lambda x: x[1])
    _, nearestDistance = zip(*nearest)
    nearestClass = [iris_data.at[i, 'class'] for i in range(0, len(distance)) if distance[i] <= nearestDistance[-1]]

    countDic = Counter(nearestClass)
    count_max = countDic.most_common(1)[0][1]
    majarity = [key for key, count in countDic.most_common() if count == count_max]

    return majarity[0] if len(majarity)==1 else 3
```

第二部分是函式 classify，對單一點進行分類。參數 t 為欲分類的點座標，X 、 Y 和 K 為使用者輸入參數。

首先使用 L2 作為距離函數，依照字串參數計算資料集中每一個點到 t 點的距離，存入 distance，接著從 distance 中選取最短的 K 個距離，由小到大存入 nearestDistance。

需要注意若只選取 K 個點，可能會有相同距離的點但只考慮到 index 較小的點的情況，因此將做法改成依照 K 值選取最遠的可能距離，也就是 nearestDistance[-1]，將小於等於最遠可能距離的點篩選出來，並將這些點的分類編號存入 nearestClass。

接著統計 nearestClass 中各分類出現的次數，判斷何種分類的數量最多，回傳該種分類的編號，若有兩個以上的分類出現次數相同，則回傳 3。

```
def knn(x, y, k):

    x_min, x_max = round(min(iris_data[x])) - 1, round(max(iris_data[x])) + 1
    y_min, y_max = round(min(iris_data[y])) - 1, round(max(iris_data[y])) + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    z = np.array(list(map(lambda t: classify(t, x, y, k), np.c_[xx.ravel(), yy.ravel()])))
    z = z.reshape(xx.shape)

    plt.figure(figsize=(2 * (x_max-x_min), 2 * (y_max-y_min)))
    plt.pcolormesh(xx, yy, z, cmap=cmap_light)
    scatter = plt.scatter(iris_data[x], iris_data[y], c=iris.target, cmap=cmap_bold, alpha=0.8)
    plt.legend(handles=scatter.legend_elements()[0], labels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
    plt.title("Iris Classification (k = %i, distance metric = L2, step size = %g)" % (k, h))
    plt.xlabel(x)
    plt.ylabel(y)
```

第三部分則是函式 knn，對每一個點進行預測並產生圖表。參數 X 、 Y 和 K 為使用者輸入參數。

首先利用資料集中數值最大與最小的點，設定圖表橫坐標與縱座標的顯示範圍，並利用 meshgrid()產生此範圍的座標矩陣 xx 與 yy。將座標矩陣攤平後，依序將欲預測的點座標放入函式 classify 中，再將預測結果存入 z，使用 reshape()將預測結果恢復成二維矩陣。最後依照預測結果輸出圖形即可。