請測試以 cascade 架構下的 Adaboost、KNN 二種分類器進行訓練，以各種不同的參數，以及不同的訓練集、測試集比例，觀察調整以上變數的情況下，對於訓練出來的準確率有什麼變化。

Adaboost

主要是透過 python sklearn.ensemble 函式使用，裡面其中最重要的三個參數如下：

- **n_estimators**
  為公式 $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k}, & \text{if } h_k(x^i) = y_i \\ e^{\alpha_k}, & \text{if } h_k(x^i) \neq y_i \end{cases}$ 的可最大疊代次數，其中 W_{k+1} 就是每次的疊代次數記錄，通常標準是 50，如果太大可能會導致 overfitting，太小則又不夠精確。
- **learning_rate**
  為每個分類器的權重，如果每個分類器的權重越大，那建議 n_estimators 可以小點，如果每個分類器的權重越小，則建議 n_estimators 要更大些。

透過鳶尾花資料集進行練習，我發現最主要的問題是 train_test_split 中的 test_size 比例越小時則其準確性越高，反之則越大。

程式碼

```
iris = datasets.load_iris()

x = iris.data
y = iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

#test_size 越小，準確率越高，越大則準確率越低
#在learning_rate 越大，感覺越好，n_estimators注意不要 overfitting
clf = AdaBoostClassifier(n_estimators=50, learning_rate=1, random_state=0)
clf.fit(x_train, y_train)
print("AdaBoost accuracy")
print(clf.score(x, y))
print()
```

輸出結果

```
AdaBoost accuracy
0.96
```

其中最酷的是 n_estimators 在 40 的情況時，準確率會降低，但在

n_estimators 是 30 的情況下時，準確率卻又會上升至 n_estimators 等於 50。
學習連結如下
sklearn.ensemble.AdaBoostClassifier()函数解析（最清晰的解释）by 我是管小亮
sklearn.ensemble.AdaBoostClassifier by scikit learn
[Day26]機器學習：KNN 分類演算法！by iT 邦幫忙

KNN

主要是透過 python sklearn.neighbors 函式使用，裡面其中最重要的三個參數如下：

- n_neighbors
  預設是 5，是 KKN 演算法中已離你最近的 K 的點為主，將自己也視為此類別。
- weights
  以要被判斷的點為中心，有兩種方法 uniform 為只要在此範圍內的點權重都相同，而 distance 每個點權重都不同，取決於每個點到中心點的距離。
- algorithm
  使用的演算法，預設是 auto，也就是函式自動使用適合的演算法，brute 是暴力搜尋，當訓練集大時非常耗時，而 kd_tree and ball_tree 則是用來改善 brute，而產生的演算法。

程式碼

```
iris = datasets.load_iris()

x = iris.data
y = iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)


#感覺不到差異
neigh = KNeighborsClassifier(n_neighbors=5, algorithm='kd_tree', weights='distance')
neigh.fit(x_train, y_train)
print("KNN accuracy")
print(neigh.predict(x_test))

xx = neigh.predict(x_test)
cnt = 0
for i in range(0, len(x_test)):
  if(xx[i] == y_test[i]):
    cnt+=1
print("Accuracy", cnt/len(x_test))
```

結果

```
KNN accuracy
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1
0]
Accuracy 1.0
```

其中由於 KNN 會出來的是對於每個點的預測，於是我有在第 30 行到第 35 行簡單的對每筆資料進行判斷。
其中得出我們的 weights 如果使用 distance 的準確率會比 uniform 來的更好。
而 n_neighbors 在這邊我們則不會感受到太大的差異，主要是因為資料量太小且 5 是標準函式預設的 n_neighbors，如果再小可能就會導致 overfitting。

注意：n_neighbors 不可以設定為 1，此函式出來的結果畢竟會是 100%準確，沒辦法發揮此函式的功用。

學習連結
sklearn.neighbors.KNeighborsClassifier()函数解析（最清晰的解释）by 我是管小亮
sklearn.neighbors.KNeighborsClassifier by scikit learn
[Day26]機器學習：KNN 分類演算法！by iT 邦幫忙