sklearn.datasets.IRIS

# 引入的模組

## 基本模組

import numpy as np

import pandas as pd

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

## 學習器模組

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import LinearSVC

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

整體學習

from sklearn.ensemble import VotingClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

## 模型驗證模組

from sklearn import metrics

from sklearn.model\_selection import KFold,StratifiedKFold

from sklearn.learning\_curve import learning\_curve

## 繪圖模組

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

# 實作步驟

## 引入基本模組

## 引入資料集(load)並進行探索性資料分析(Exploratory data analysis)

* 資料集內容檢視
* 敘述性統計
* 相關性分析、熱度圖
* 散佈圖

## 資料預處理與特徵工程 (Data Clean/Preprocessing & Feature Engineering)

* X、y資料建立
* 資料切割
* 資料標準化

## 訓練模型與校調 (Model Training)

1. PCA (n = 4)
2. PCA (n = 2)：以此使用
   1. 資料切割
   2. 資料標準化
3. Linear SVC
   1. LinearSVC() + PCA\_std
   2. LinearSVC() + PCA
   3. LinearSVC()
   4. LinearSVC() + std
4. KNeighborsClassifier
   1. KNeighborsClassifier(n\_neighbors=3, weights='uniform')
   2. KNeighborsClassifier (n\_neighbors=3, weights='uniform') + std
   3. KNeighborsClassifier(n\_neighbors=3, weights='uniform') + PCA
   4. KNeighborsClassifier(n\_neighbors=3, weights='uniform') + PCA\_std

## 後續優化

1. SVM
   1. SVC(C = 1, kernel = 'rbf') + std
   2. SVC(C = 1, kernel = 'rbf')
   3. SVC(C = 1, kernel = 'rbf') + PCA
   4. SVC(C = 1, kernel = 'rbf') + PCA\_std
   5. SVC(C = 1, kernel = 'poly') + std
   6. SVC(C = 1, kernel = 'linear') + std
2. Ensemble Learning
   1. VotingClassifier(estimators = [('KNN', clf1), ('SVC\_r', clf2), ('SVC\_p', clf3), ('SVC\_l', clf4)], voting = 'soft', weights = [1, 5, 1, 1]) + std
   2. BaggingClassifier(n\_estimators= 100, oob\_score= True)
   3. BaggingClassifier(n\_estimators= 100)
   4. BaggingClassifier(n\_estimators= 100) + std
   5. BaggingClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True), n\_estimators= 100, oob\_score= True)
   6. BaggingClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True), n\_estimators= 100)
   7. BaggingClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True), n\_estimators= 100) + std
   8. BaggingClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True), n\_estimators= 100) + PCA
   9. BaggingClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True), n\_estimators= 100) + PCA + std
   10. RandomForestClassifier(n\_estimators= 90, criterion='gini', max\_features= 'auto', oob\_score=True)
   11. RandomForestClassifier(n\_estimators= 100, criterion='gini', max\_features= 'auto')
   12. AdaBoostClassifier(n\_estimators= 100)
   13. AdaBoostClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True),n\_estimators= 100) + std
   14. AdaBoostClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True),n\_estimators= 100)
   15. AdaBoostClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True),n\_estimators= 100) + PCA
   16. AdaBoostClassifier(base\_estimator = SVC(C = 1, kernel = 'rbf', probability = True),n\_estimators= 100) + PCA + std

# 資料集內容

資料集索引dict\_keys：'data'、'target'、'target\_names'、'DESCR'、'feature\_names'

資料筆數：150 (每類50筆資料)

類別數：3

類別：

- Iris-Setosa

- Iris-Versicolou

- Iris-Virginica

特徵數量：4

特徵(屬性)類型：數值型

特徵：

- sepal length in cm

- sepal width in cm

- petal length in cm

- petal width in cm

敘述性統計 (Summary Statistics)：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Min | Max | Mean | SD | Class Correlation |
| sepal length | 4.3 | 7.9 | 5.84 | 0.83 | 0.7826 |
| sepal width | 2.0 | 4.4 | 3.05 | 0.43 | -0.4194 |
| petal length | 1.0 | 6.9 | 3.76 | 1.76 | 0.9490 (high!) |
| petal width | 0.1 | 2.5 | 1.20 | 0.76 | 0.9565 (high!) |

遺失值：沒有

類別分布：33.3% / 類

資料建立時間：1988年7月