D28：實作樹型(Tree Base)模型

課程閱讀 本日作業 問題討論 學習心得(完成)

重要知識點 >
Scikit-learn >
安裝套件 >

重要知識點

- 學習透過第三方套件使用樹型模型

(本日課程為實作導向課程，同學可參考附件"實作 TreeBase 模型_2.ipynb ")

Scikit-learn

在先前的課程中，我們學習到如何使用 Python 從頭開始實作決策樹與隨機森林。但我們可以利用第三方的套件來使用已經實作好的模型，並非每次都需要重造輪子。在這次的課程中，我們會使用經典的機器學習套件 sklearn 來使用樹型模型。

Scikit-learn 是機器學習中的經典套件，內含多個以實現的函式與模型。

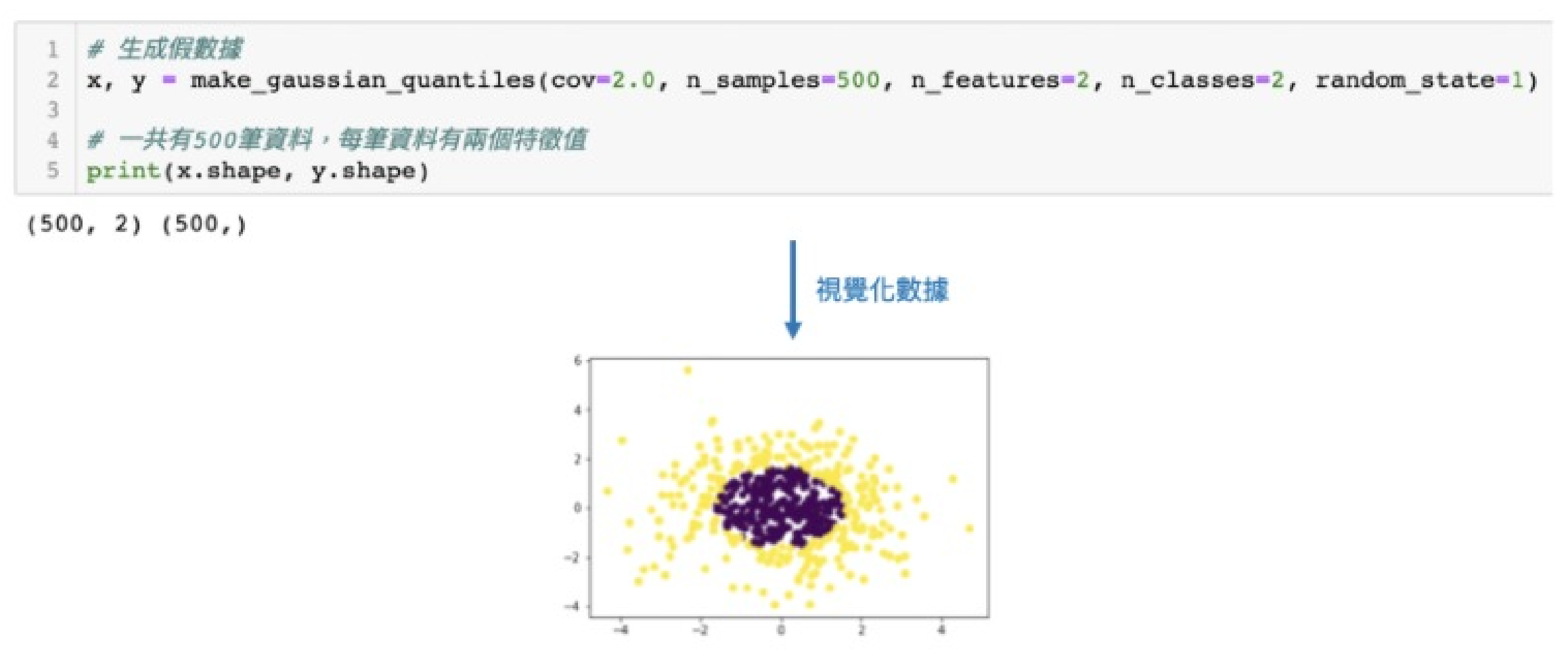
安裝套件

在開始使用套件之前，我們需要先安裝，可以透過 python 的套件管理器 pip 來進行安裝。

pip install scikit-learn

生成假數據

這邊我們利用 sklearn 的函式自行製作假數據。



```
# 生成假數據
x, y = make_gaussian_quantiles(cov=2.0, n_samples=500, n_features=2, n_classes=2, random_state=1)

# 一共有500筆資料，每筆資料有兩個特徵值
print(x.shape, y.shape)
```

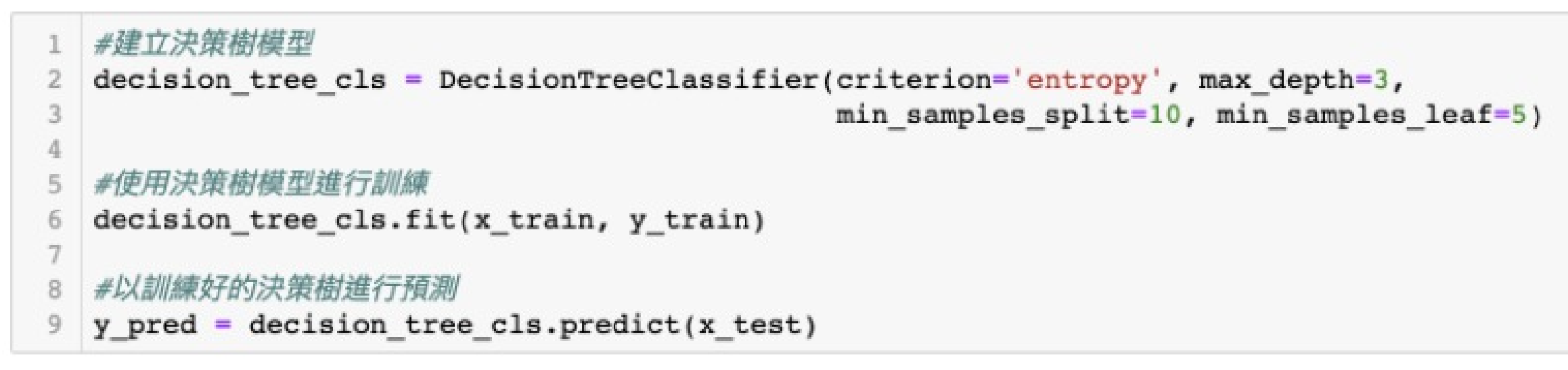
```
(500, 2) (500,)
```

```
# 生成假數據
x, y = make_gaussian_quantiles(cov=2.0, n_samples=500, n_features=2, n_classes=2, random_state=1)

# 一共有500筆資料，每筆資料有兩個特徵值
print(x.shape, y.shape)
```

決策樹模型

使用 sklearn 進行決策樹模型的搭建、訓練與預測。 決策樹中常見的參數如下：

- criterion：給定使用計算訊息增益的方法，有 "gini" 與 "entropy"，預設值為"entropy"
- max_depth：樹最深可生長深度
- min_samples_split：可分割節點最少需含樣本數
- min_samples_leaf：最為終端節點(leaf node)最少需含樣本數，預設為 1
- max_leaf_nodes：最多終端節點數



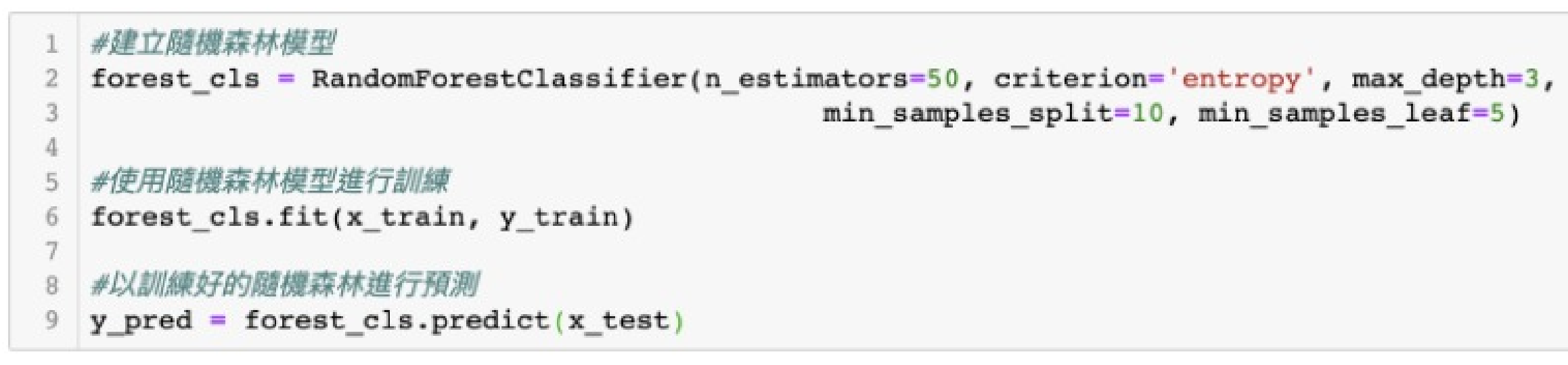
```
#建立決策樹模型
decision_tree_cls = DecisionTreeClassifier(criterion='entropy', max_depth=3,
                                           min_samples_split=10, min_samples_leaf=5)

#使用決策樹模型進行訓練
decision_tree_cls.fit(x_train, y_train)

#以訓練好的決策樹進行預測
y_pred = decision_tree_cls.predict(x_test)
```

```
#建立決策樹模型
decision_tree_cls = DecisionTreeClassifier(criterion='entropy', max_depth=3,
                                           min_samples_split=10, min_samples_leaf=5)

#使用決策樹模型進行訓練
decision_tree_cls.fit(x_train, y_train)

#以訓練好的決策樹進行預測
y_pred = decision_tree_cls.predict(x_test)
```

隨機森林模型

使用 sklearn 進行隨機森林模型的搭建、訓練與預測。 隨機森林中常見的參數如下：

- n_estimators：在隨機森林中決策樹個數，預設值為100 (100 顆決策樹)
- criterion：給定使用計算訊息增益的方法，有 "gini" 與 "entropy"，預設值為"entropy"
- max_depth：樹最深可生長深度
- min_samples_split：可分割節點最少需含樣本數
- min_samples_leaf：最為終端節點(leaf node)最少需含樣本數，預設為 1
- max_leaf_nodes：最多終端節點數



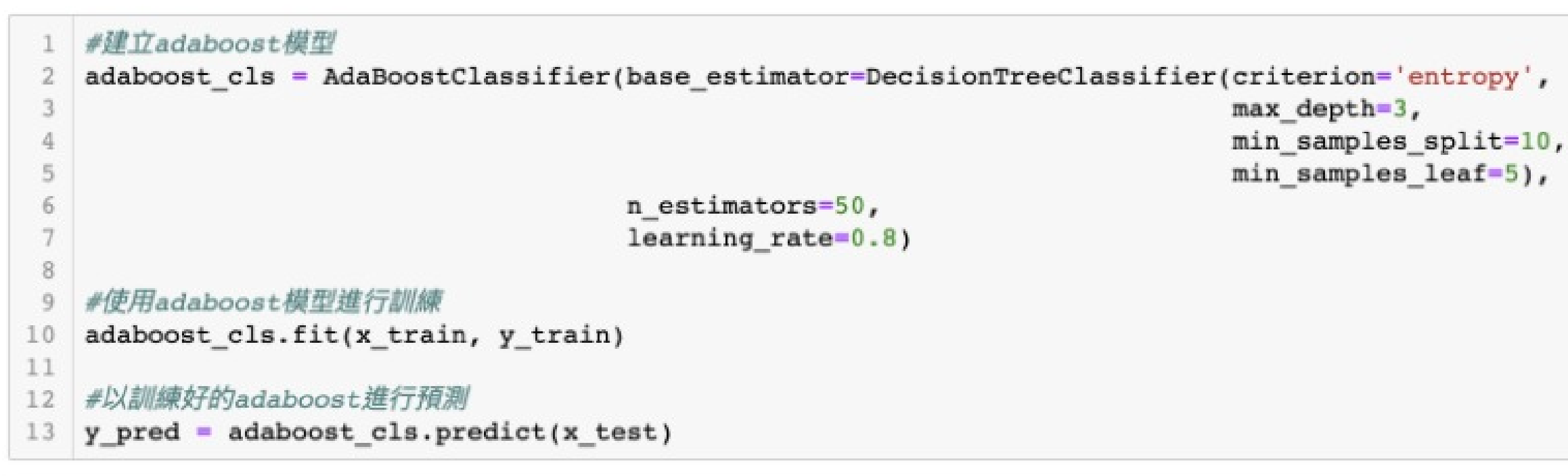
```
#建立隨機森林模型
forest_cls = RandomForestClassifier(n_estimators=50, criterion='entropy', max_depth=3,
                                    min_samples_split=10, min_samples_leaf=5)

#使用隨機森林模型進行訓練
forest_cls.fit(x_train, y_train)

#以訓練好的隨機森林進行預測
y_pred = forest_cls.predict(x_test)
```

```
#建立隨機森林模型
forest_cls = RandomForestClassifier(n_estimators=50, criterion='entropy', max_depth=3,
                                    min_samples_split=10, min_samples_leaf=5)

#使用隨機森林模型進行訓練
forest_cls.fit(x_train, y_train)

#以訓練好的隨機森林進行預測
y_pred = forest_cls.predict(x_test)
```

隨機 Adaboost 模型

使用 sklearn 進行 Adaboost 模型的搭建、訓練與預測。 隨機森林中常見的參數如下：

- base_estimator：指定基礎分類器，可以是各種不同分類器(ex：DecisionTreeClassifier, LogisticRegression)
- n_estimators：最多的分類器個數
- learning_rate：每個分類器的權重衰減速率



```
#建立adaboost模型
adaboost_cls = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(criterion='entropy',
                                                                         max_depth=3,
                                                                         min_samples_split=10,
                                                                         min_samples_leaf=5),
                                  n_estimators=50,
                                  learning_rate=0.8)

#使用adaboost模型進行訓練
adaboost_cls.fit(x_train, y_train)

#以訓練好的adaboost進行預測
y_pred = adaboost_cls.predict(x_test)
```

```
#建立adaboost模型
adaboost_cls = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(criterion='entropy',
                                                                         max_depth=3,
                                                                         min_samples_split=10,
                                                                         min_samples_leaf=5),
                                                                         n_estimators=50,
                                                                         learning_rate=0.8)

#使用adaboost模型進行訓練
adaboost_cls.fit(x_train, y_train)

#以訓練好的adaboost進行預測
```

詳細使用操作

請參照使用實做 TreeBase 模型_2.ipynb 檔進行更詳細的使用操作

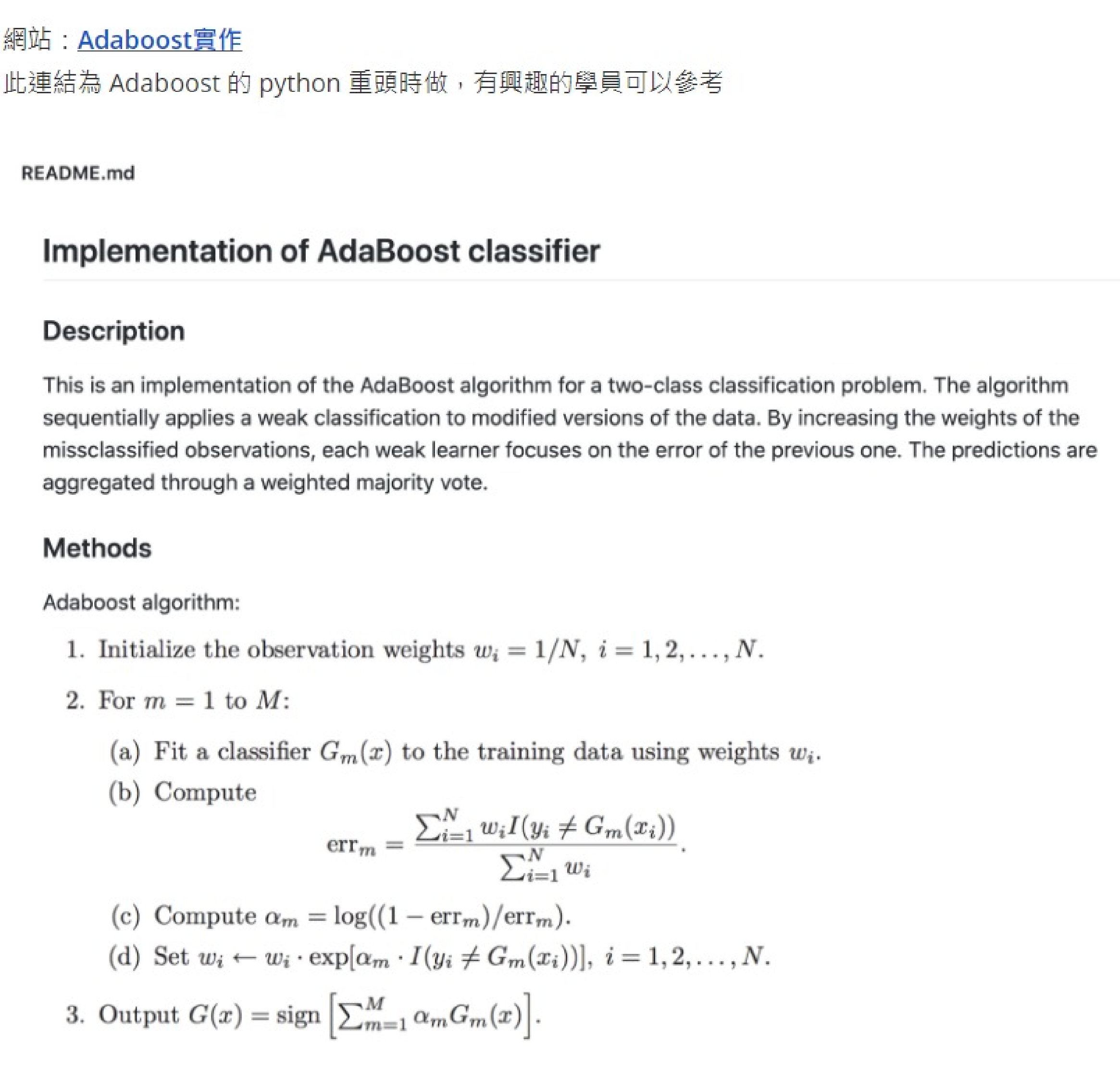
知識點回顧

在這章節我們學習到了

- 透過第三方套件(Scikit-learn)使用樹型模型

延伸閱讀

網站：Adaboost實作
此連結為 Adaboost 的 python 重頭時做，有興趣的學員可以參考



README.md

Implementation of AdaBoost classifier

Description

This is an implementation of the AdaBoost algorithm for a two-class classification problem. The algorithm sequentially applies a weak classification to modified versions of the data. By increasing the weights of the missclassified observations, each weak learner focuses on the error of the previous one. The predictions are aggregated through a weighted majority vote.

Methods

Adaboost algorithm:

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.

下一步：完成作業