AS 讀書會

Search

XGBoost

AS2 Jack

分類演算法常見評價指標

◘ Accuracy（準確率）

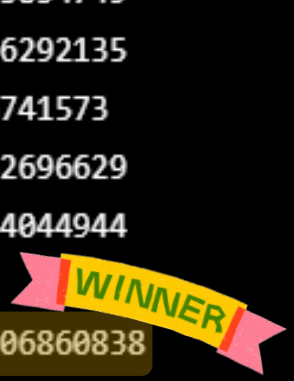
$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{all classifications}}$$

Baseline Score = 0.6

在 Titanic 的訓練資料集中，約有 6 成的船員未倖存，因此若預測船員倖存與否時全猜死亡，Accuracy 大約會是 0.6，而我們就可以將我們機器學習的 Baseline Score 設定為 0.6，來檢視我們的模型預測結果是否有比盲猜死亡來的好，藉以判斷模型好壞。

使用 XGBoost 模型預測的準確率結果

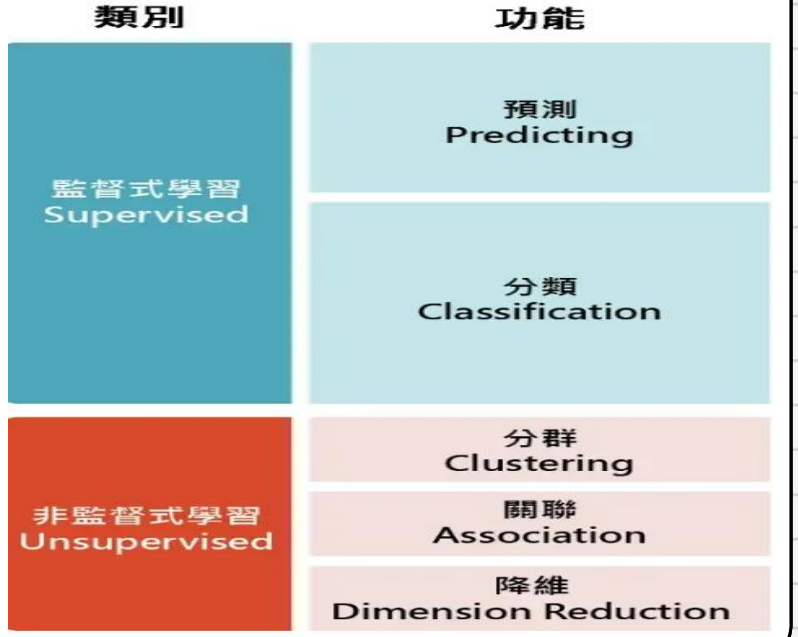
```
Fold 0 Acc: 0.7932960893854749
Fold 1 Acc: 0.8258426966292135
Fold 2 Acc: 0.848314606741573
Fold 3 Acc: 0.8539325842696629
Fold 4 Acc: 0.7808988764044944

Overall Acc: 0.8204569706860838
CPU times: user 1min 47s, sys: 643 ms, total: 1min 48s
Wall time: 1min 46s
```



先備知識

- 機器學習（ML）是人工智慧的（AI）的一部份
- 機器學習依據是否貼標，可以區分為監督式與非監督式學習
- 機器學習依據是否集成，可以區分為天才型與投票型模型
- 常見的迴歸預測模型效果評斷指標有 MAE、MSE、RMSE、R2
- 常見的分類預測模型效果評斷指標有 Accuracy、Precision 等

補充



先備知識

- 機器學習（ML）是人工智慧的（AI）的一部份
- 機器學習依據是否貼標，可以區分為監督式與非監督式學習
- 機器學習依據是否集成，可以區分為天才型與投票型模型
- 常見的迴歸預測模型效果評斷指標有 MAE、MSE、RMSE、R2
- 常見的分類預測模型效果評斷指標有 Accuracy、Precision 等

天才型 vs. 投票型

- 集成：當一個機器學習模型是由多個弱學習器結合而成，我們稱之為集成模型（aka 強學習器）
- 天才型與投票型模型的分類並不是一個正式的分類
- 投票型模型是一種集成模型，代表該機器學習的結果是由多個弱學習器「投票」決定的，較可廣泛應用於不同類型的資料上；反之則是天才型模型

先備知識

- 機器學習（ML）是人工智慧的（AI）的一部份
- 機器學習依據是否貼標，可以區分為監督式與非監督式學習
- 機器學習依據是否集成，可以區分為天才型與投票型模型
- 常見的迴歸預測模型效果評斷指標有 MAE、MSE、RMSE、R2
- 常見的分類預測模型效果評斷指標有 Accuracy、Precision 等

指標

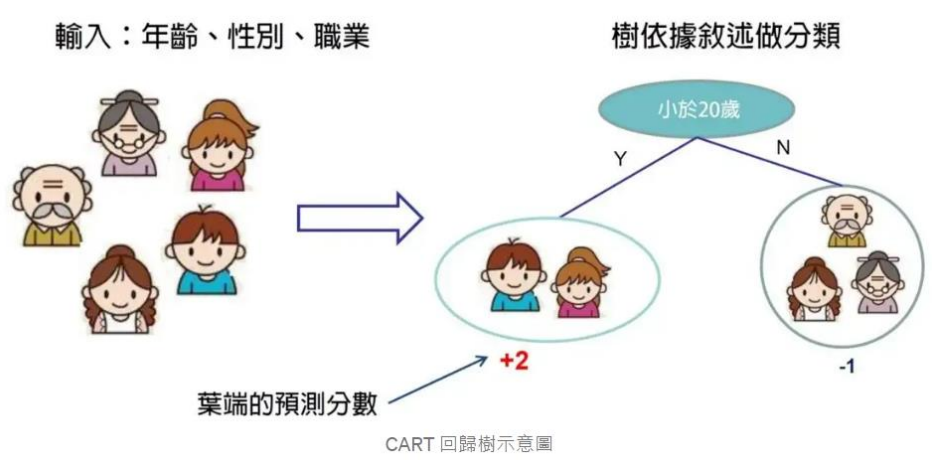
預測模型效果評斷指標

- 當使用機器學習模型為我們做預測時，我們會需要一個指標來協助我們判定模型是否有幫助到我們在預測上有更好的成效
- 迴歸預測上，通常會希望與實際值的誤差越小越好，因此指標多是在評斷誤差大小
- 分類預測上，通常會希望分類愈精準越好，因此指標多是在評斷準確率或精準度的高低

XGBoost

全名 eXtreme Gradient Boosting（極限梯度提升）

Decision Tree（決策樹）



CART 回歸樹示意圖

Decision Tree（決策樹）

優點：

- 運算時間較短
- 運算過程就像人類決策過程，較直覺

重點：

- 透過演算法協助決定以什麼資料特徵做為決策條件
- 為避免過擬合，透過預剪枝、後剪枝兩種方法將不好的分支剪去

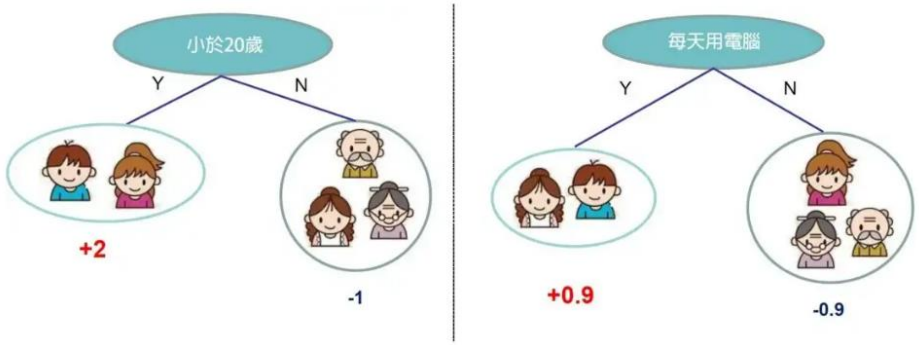
基底

XGBoost

全名 eXtreme Gradient Boosting（極限梯度提升）

一、XGBoost 是什麼

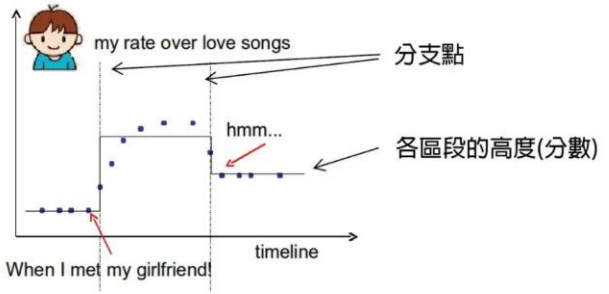
是監督式學習，也是投票型模型



二、XGBoost 學什麼

最小化目標函數，因為目標函數值越低，代表整體樹結構越好

- 讓誤差函數下降（代表猜得越準）
- 讓正則函數下降（代表較無過擬合）

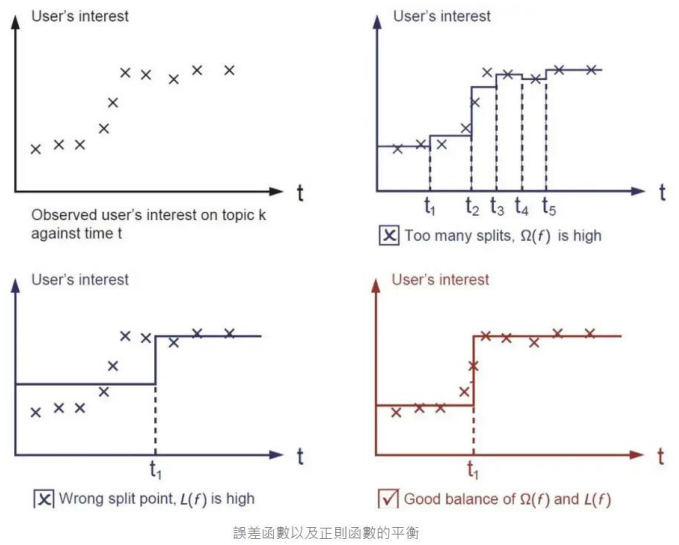


簡介

XGBoost

全名 eXtreme Gradient Boosting（極限梯度提升）

XGBoost 學什麼（續）



誤差函數以及正則函數的平衡

XGBoost 怎麼學

Gradient Descent ✗

Gradient Boosting ✓

- 每次預測時加入新的學習函數（參數，也就是樹模型中新樹、新分支）
- 怎麼加：貪婪演算法從深度 0 開始加，一個一個可能的分支嘗試
- 貪婪演算法缺點：可能找到的只是相對最好的結果，錯失更好的模型結果可能

簡介

PYTHON

XGBoost 在分類預測上的程式碼

<載入套件>

```
from sklearn.model_selection import KFold, cross_val_score,
        train_test_split
import optuna
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
```

PYTHON

XGBoost 在分類預測上的程式碼

<將資料切分為訓練集跟驗證集>

```
from sklearn.model_selection import train_test_split
# split data for train and valid
train_x, val_x, train_y, val_y = \
    train_test_split(X, y, random_state=17, stratify=y, test_size=0.2)
```

PYTHON

XGBoost 在分類預測上的程式碼

<讓程式幫我們選擇參數>（Optional）

```
import optuna
def objective(trial):
    xgb_params = {...}
    model = XGBClassifier(...)
    model.fit(...)
    y_preds = model.predict(val_x)
    return accuracy_score(y_true = val_y,
                          y_pred = y_preds)   # return accuracy
study = optuna.create_study(direction = "maximize")
study.optimize(objective, n_trials=50)
```

PYTHON

XGBoost 在分類預測上的程式碼

<訓練 XGBoost 預測分類結果>

```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
best_params = study.best_params  # get best parameters
# define xgbclassifier model
model = XGBClassifier(**best_params, ...)
model.fit(train_x, train_y, ...)
y_preds = model.predict(val_x)
acc = accuracy_score(y_true = val_y, y_pred = y_preds)
# print accuracy
print("\nxgbclassifier accuracy : {}".format(acc))
```

PYTHON

XGBoost 在迴歸預測上的程式碼

<載入套件>

```
from sklearn.model_selection import KFold, cross_val_score,
        train_test_split
import optuna
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error
```

PYTHON

XGBoost 在迴歸預測上的程式碼

<將資料切分為訓練集跟驗證集>

```
from sklearn.model_selection import train_test_split
# split data for train and valid
train_x, val_x, train_y, val_y = \
    train_test_split(X, y, random_state=42, test_size=0.1)
```

PYTHON

XGBoost 在迴歸預測上的程式碼

<讓程式幫我們選擇參數>（Optional）

```
import optuna
def objective(trial):
    xgb_params = {...}
    model = XGBRegressor(...)
    model.fit(...)
    y_preds = model.predict(val_x)
    return mean_squared_error(val_y, y_hat,
                                        squared=False)   # return RMSE
study = optuna.create_study(direction = "minimize")
study.optimize(objective, n_trials=50)
```

PYTHON

XGBoost 在迴歸預測上的程式碼

<訓練 XGBoost 預測分類結果>

```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error
best_params = study.best_params  # get best parameters
# define xgbregressor model
model = XGBRegressor(**best_params, ...)
model.fit(train_x, train_y, ...)
y_preds = model.predict(val_x)
acc = mean_squared_error(y_true = val_y, y_pred = y_preds)
# print RMSE
print("\nxgbregressor RMSE : {}".format(acc))
```

參考資料

- 第一則參考資料講述的是 XGBoost 的基礎 — 決策樹模型
- 第二則與第三則則是在介紹 XGBoost，第二則偏重程式撰寫，第三則偏重原理介紹與數學推導

① Decision tree 決策樹 — 單純、快速、解釋性高的決策術
② [資料分析&機器學習] 第5.2講: Kaggle機器學習競賽神器 XGBoost介紹
③ 機器學習 — Gradient Boosting (1)

THANK YOU