XGBoost 隔日股價漲跌預測-Python 實作

國立高雄科技大學金融資訊系教授兼AI金融科技中心主任
林萍珍

2023.12.1

Shubham Malik, Rohan Harode, Akash Singh
XGBoost: A Deep Dive into Boosting
Updated Frebruary 2020
https://www.researchgate.net/publication/339499154_XGBoost_A_Deep_Dive_into_Boosting_Introduction_Documentation

致謝

- **部分圖片和內容來自以下人士和機構:**

Shubham Malik, Rohan Harode, Akash Singh
XGBoost: A Deep Dive into Boosting
Updated Frebruary 2020
Technical report

https://www.researchgate.net/publication/339499154_XGBoost_A_Deep_Dive_into_Boosting_Introduction_Documentation

https://zhuanlan.zhihu.com/p/584124751

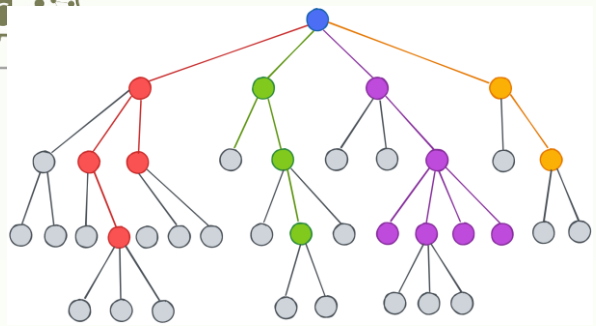
國立高雄科技大學 AI 金融科技中心版權所有

XGBOOST 隔日股價漲跌預測

- **複合式學習(Assemble Learning)**
- 提升(Boosting)演算法運作原理
- **梯度提升Gradient Boosting圖解**
- XGBoost為何是機器學習首選算法?
- **決策樹的修剪 (Tree pruning)**
- SSR計算葉與樹的誤差平方合
- **程式碼講解**
  - **訓練、驗證、測試資料切割**
  - 參數設定:樹深度、學習率...
  - **創建XGBoost分類器**
  - 訓練XGBoost模型
  - **預測測試集**
  - 測試結果做混淆矩陣計算
- **AI 實作之參數與資料集的校調**

國立高雄科技大學 AI 金融科技中心版權所有

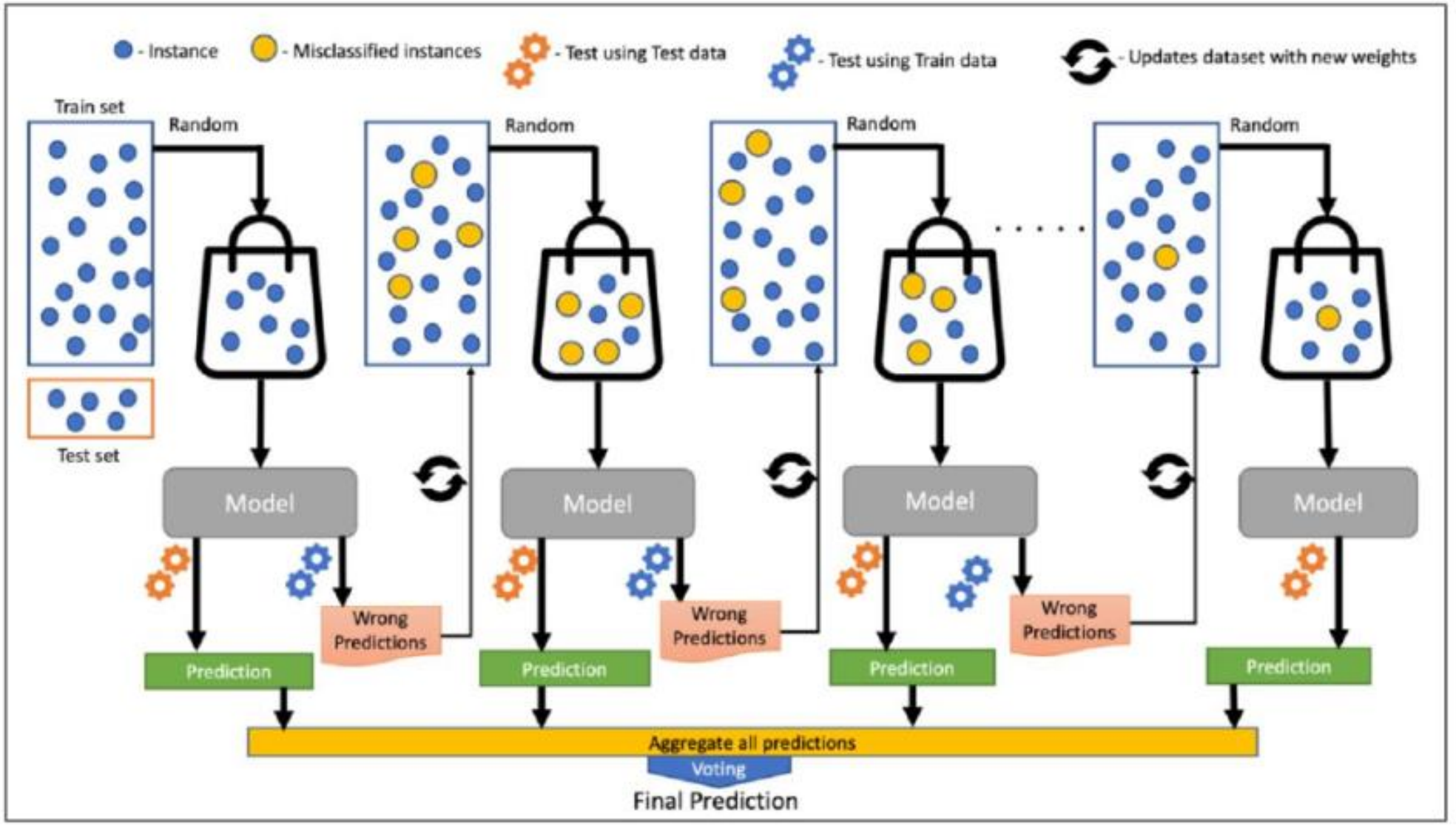
分類和回歸樹(CART)



- **Classification And Regression Trees (CART)**
  - 監督式機器學習算法，用於預測建模，多個獨立變數預測一個依賴變數（目標）
  - 類似樹的結構，頂部是根。
  - 分類(Classification): 當目標變量是固定類別(不連續)，這個算法被用來識別目標最有可能落入的類別，例如: 預測股市上漲或下跌兩類。
  - 回歸樹(Regression trees)：當目標變量是連續的，這棵樹/算法被用來預測連續的值，例如預測大盤指數, ex: 預測明天大盤是18000點。
- **XGBoost內部結構模型設計源自於CART，差別在於XGBoost是模型樹不是分類樹或回歸樹**
  - 模型樹的葉節點輸出值不是分到該葉節點的所有樣本點的均值（回歸樹），而是由一個函數產生的值。
- **XGBoost是機器學習一種非結構化梯度提升的演算法，改善CART準確度不足、損失高、結果變異大等問題。**
- **XGBoost可以處理回歸和分類問題，即可以預測實數(連續)也可以預測類別(不連續)。**

國立高雄科技大學 AI 金融科技中心版權所有

提升(Boosting)演算法運作原理



國立高雄科技大學 AI 金融科技中心版權所有

梯度提升Gradient Boosting

- **梯度提升是提升演算法(Boosting algorithm)的一種特例。**
- **依梯度下降(Gradient Descent)演算法最小化錯誤產生決策樹。**
- **梯度提升與梯度下降會根據錯誤更新模型（弱學習者）。**
  - **梯度提升藉由梯度下降的演算法來調整學習的權重。**
  - **此算法利用損失函數中的梯度-變化量的方向，迭代優化模型的誤差，藉此更新權重。**
  - **誤差指預測值和實際值之間的差異。**

$$w = w - \eta \nabla w$$
$$\nabla w = \frac{\partial L}{\partial w} \quad where\ L\ is\ loss$$

Gradient 做法是一階導數

w 代表向量的權重；η 是學習率

國立高雄科技大學 AI 金融科技中心版權所有

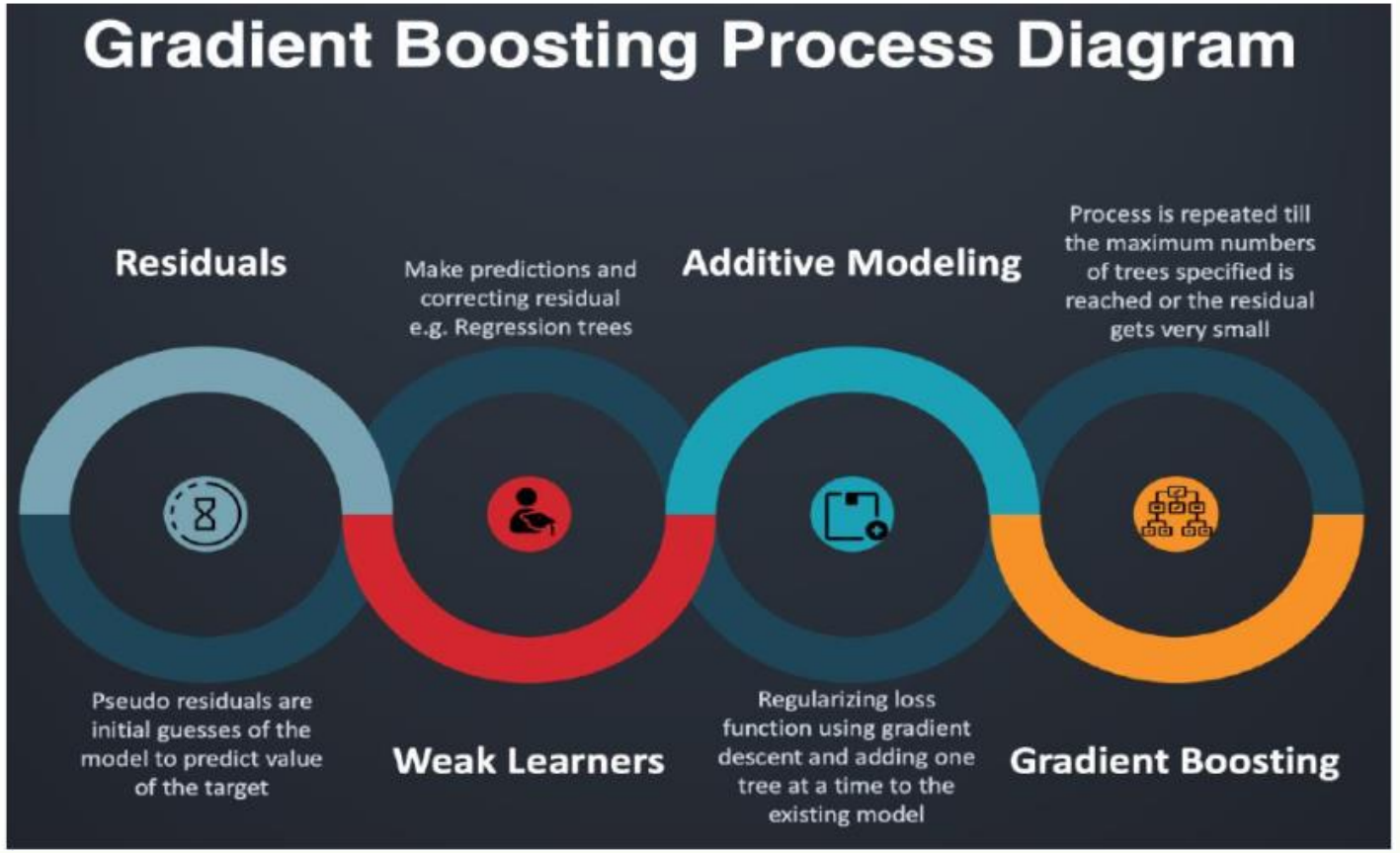
梯度提升過程圖解

殘差（Residuals）：對目標值進行初步猜測，結果為擬殘差。接著使用回歸樹進行預測並修正殘差。

弱學習者（Weak Learners）：透過加入弱學習者(一次加入一棵樹)，來對現有模型進行修正，並使用梯度下降來調整損失函數。

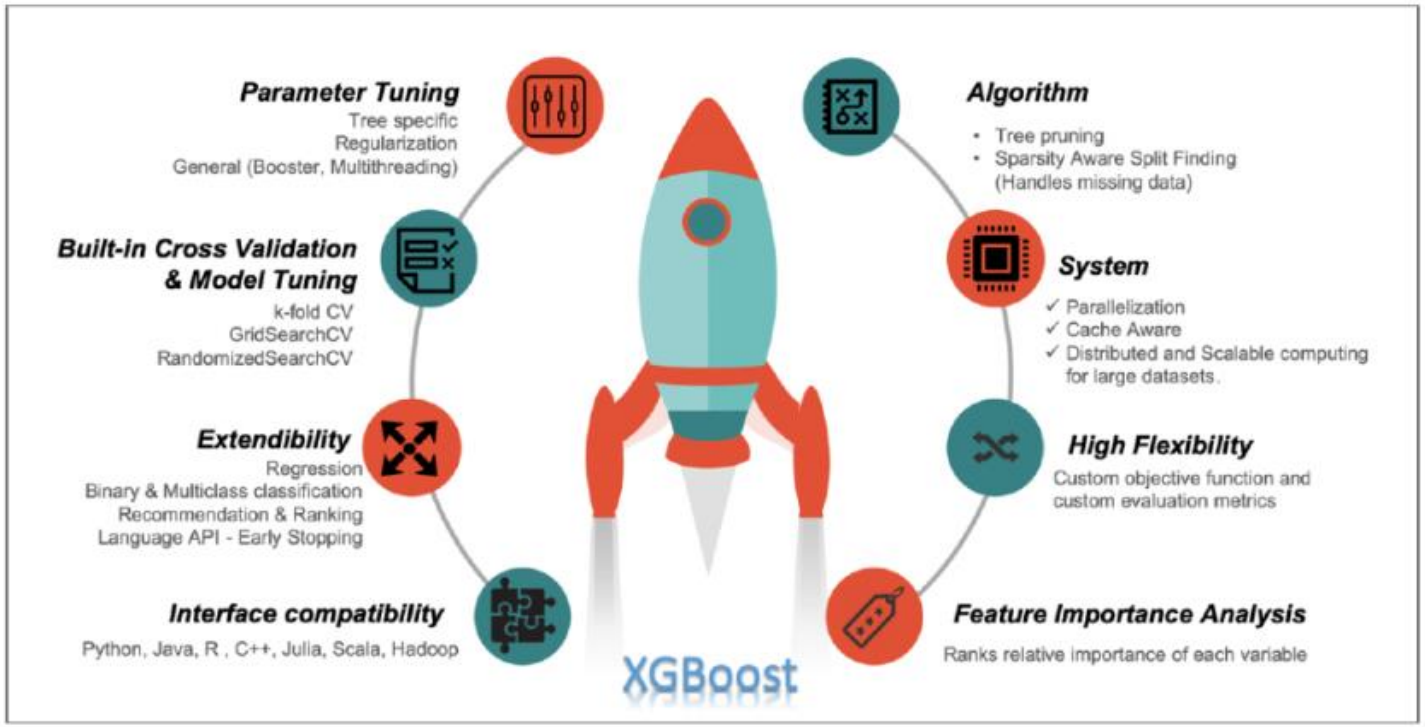
加法建模（Additive Modeling）：通過迭代添加弱學習者來不斷優化模型，每次添加為改善模型對數據的適應。

梯度提升（Gradient Boosting）：這個過程會持續重複，直到達到指定的樹的最大深度或殘差變得非常小為止。



XGBoost為何是機器學習首選算法?

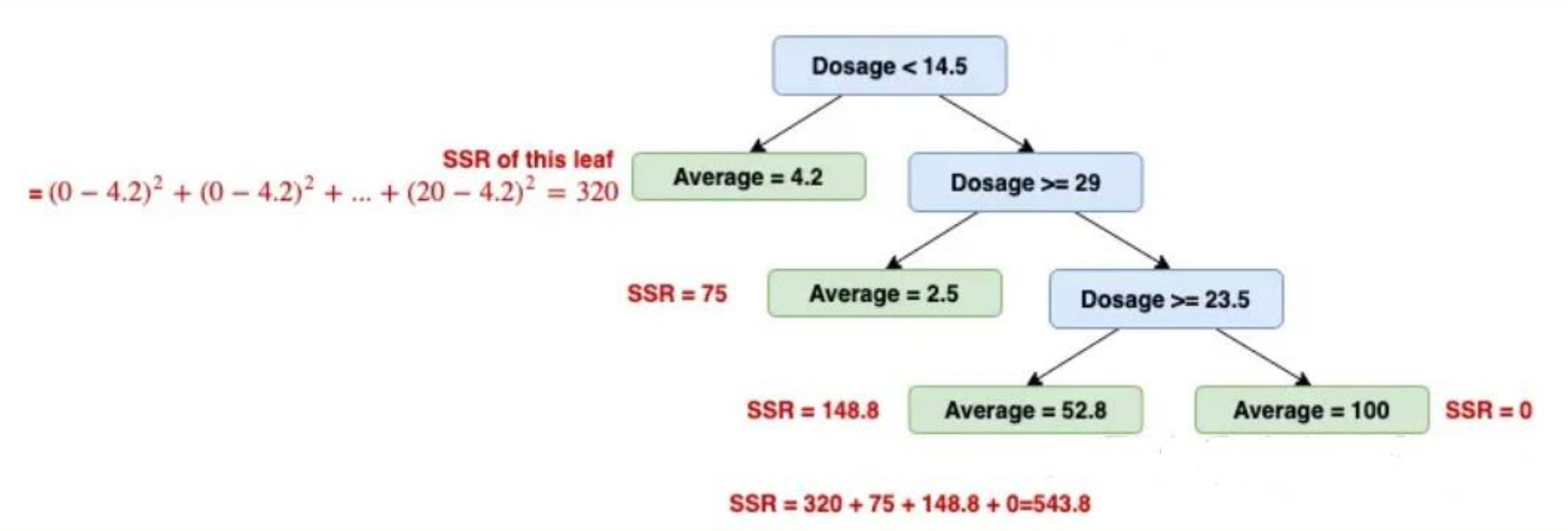
- **參數調整（Parameter Tuning）**:包含特定於樹的調整和一般性調整，比如提升方法和多執行緒。
- **內建交叉驗證與模型調整（Built-in Cross Validation & Model Tuning）**:支援k折交叉驗證、網格搜索和隨機搜索等方法來優化模型。
- **可擴展性（Extendibility）**:能夠處理二元和多類分類、回歸，並提供早停機制來防止過度配適。
- **介面兼容性（Interface compatibility）**:支持Python、Java、R、C++、Julia、Scala、Hadoop等多種程式語言。
- **系統（System）**:支持平行處理、高效利用緩存，適用於分佈式和大數據集的計算。
- **高度靈活性（High Flexibility）**:提供自定義目標函數和評估指標。
- **特徵重要性分析（Feature Importance Analysis）**:對每個變量的相對重要性進行排名。
- **樹修剪（Tree pruning）**: 實現了樹修剪分割點處理。



國立高雄科技大學 AI 金融科技中心版權所有

SSR計算葉與樹的誤差平方合

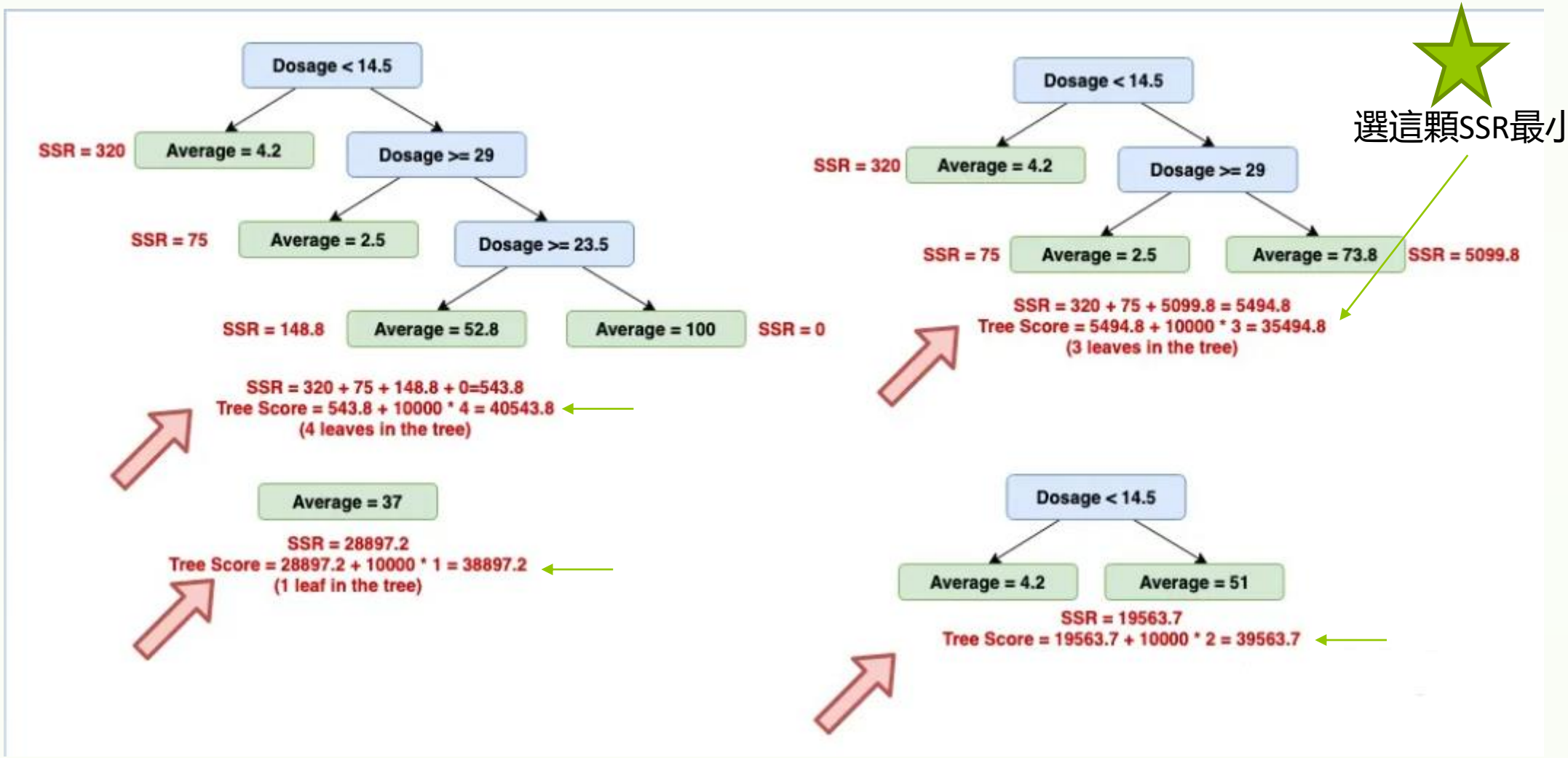
- **計算sum of the squared residuals (SSR) 誤差平方合, 每片葉子, 再加總整個樹。**



https://zhuanlan.zhihu.com/p/584124751

國立高雄科技大學 AI 金融科技中心版權所有

同一顆對不同修剪結構選最SSR最小



國立高雄科技大學 AI 金融科技中心版權所有

AI.FINTECH
AI 金融科技中心

XGBoost模型
進行股價預測

製作人：林萍珍、黃宥輔

包含程式碼：

1.get_metrics.py
2.split_date_set.py
3.XGBOOST模型_2330.py

國立高雄科技大學AI 金融科技中心 版權所有

效力檢定(混淆矩陣)

- 回召率(recall)：模型識別正類樣本的能力 $\frac{TP}{TP+FN}$
- 精確率(precision)：模型預測為正類的樣本中有多少是真正的正類樣 $\frac{TP}{TP+FP}$
- 特異性(specificity)：模型識別負類樣本的能力 $\frac{TN}{TN+FP}$
- 陰性預測值(NPV)：模型預測為負類的樣本中有多少是真正的負類樣本= $\frac{TN}{TN+FN}$
- F1-score = $2 \times \frac{(\text{回召率} \times \text{精確度})}{(\text{回召率} + \text{精確度})}$
- 準確度(accuracy)：(TP+TN)/(TP+TN+ FP+FN)

預測 \ 真實	上漲	下跌
上漲	TP	FP
下跌	FN	TN

涵義

預測 \ 真實	上漲	下跌
上漲	真實陽性 true positive(TP)	虛偽陽性 false positive(FP) (型 I 錯誤)
下跌	虛偽陰性 false negative(FN) (型 II 錯誤)	真實陰性 true negative(TN)

get_metric.py

程式位置
檔案：get_metric.py

國立高雄科技大學 AI 金融科技中心版權所有

引用套件、計算混淆矩陣


```python
# -*- coding: utf-8 -*-
from sklearn.metrics import confusion_matrix

def metrics_summary(actual, predicted):
    cm = confusion_matrix(actual, predicted)
    TN, FP, FN, TP = cm.ravel()

    # 計算準確度 (Accuracy = (TP + TN) / (TP + TN + FP + FN))
    accuracy = (TP + TN) / (TP + TN + FP + FN)

    # 計算精確度 (Precision = TP / (TP + FP))
    precision = TP / (TP + FP)

    # 計算召回率 (Recall = TP / (TP + FN))
    recall = TP / (TP + FN)

    # 計算F1分數 (F1 Score = 2 * (Precision * Recall) / (Precision + Recall))
    f1_score = 2 * (precision * recall) / (precision + recall)

    # 計算特異性 (Specificity = TN / (TN + FP))
    specificity = TN / (TN + FP)

    # 整理成字典並回傳
    metrics_dict = {
        "真陽性 (True Positive)": TP,
        "假陽性 (False Positive)": FP,
        "真陰性 (True Negative)": TN,
        "假陰性 (False Negative)": FN,
        "準確度 (Accuracy)": accuracy,
        "精確度 (Precision)": precision,
        "召回率 (Recall)": recall,
        "F1分數 (F1 Score)": f1_score,
        "特異性 (Specificity)": specificity
    }

    return metrics_dict
```

1.引用計算混淆矩陣的套件。

2.自訂函數中：呼叫混淆矩陣函數，傳入實際值與預測值，在將混淆矩陣的值分解為單個變量TN,FP,FN,TP。

程式位置

檔案： get_metrics.py

國立

計算指標並回傳


```python
# -*- coding: utf-8 -*-
from sklearn.metrics import confusion_matrix

def metrics_summary(actual, predicted):
    cm = confusion_matrix(actual, predicted)
    TN, FP, FN, TP = cm.ravel()

    # 計算準確度 (Accuracy = (TP + TN) / (TP + TN + FP + FN))
    accuracy = (TP + TN) / (TP + TN + FP + FN)

    # 計算精確度 (Precision = TP / (TP + FP))
    precision = TP / (TP + FP)

    # 計算召回率 (Recall = TP / (TP + FN))
    recall = TP / (TP + FN)

    # 計算F1分數 (F1 Score = 2 * (Precision * Recall) / (Precision + Recall))
    f1_score = 2 * (precision * recall) / (precision + recall)

    # 計算特異性 (Specificity = TN / (TN + FP))
    specificity = TN / (TN + FP)

    # 整理成字典並回傳
    metrics_dict = {
        "真陽性 (True Positive)": TP,
        "假陽性 (False Positive)": FP,
        "真陰性 (True Negative)": TN,
        "假陰性 (False Negative)": FN,
        "準確度 (Accuracy)": accuracy,
        "精確度 (Precision)": precision,
        "召回率 (Recall)": recall,
        "F1分數 (F1 Score)": f1_score,
        "特異性 (Specificity)": specificity
    }

    return metrics_dict
```

計算各指標公式。

整理要回傳的指標字典

程式位置

檔案： get_metrics.py

國立

split_date_set.py

程式位置
檔案：split_date_set.py

國立高雄科技大學 AI 金融科技中心版權所有

split_date.py 程式碼1

引入套件

```python
# -*- coding: utf-8 -*-
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

def main(stock_id):
    data_path = f"新增變數資料/{stock_id}.xlsx" # 檔案位置

    # 讀取資料
    df = pd.read_excel(data_path)
    df = df.dropna()

    # 訓練、驗證、測試集的比例
    train_rate = 0.7
    validate_rate = 0.2

    # 資料數輛
    data_num = df.shape[0]

    # 切割資料點
    validate_split = data_num * train_rate # 訓練、驗證集的切割點
    test_split = data_num * (train_rate + validate_rate) # 驗證、測試集的切割點
    validate_split, test_split = int(validate_split), int(test_split) # 讓切割點變成整數

    # 切割資料
    train_df = df.iloc[ : validate_split]
    validate_df = df.iloc[ validate_split : test_split ]
    test_df = df.iloc[ test_split : ]
```

設定檔案路徑，讀取檔案並刪除空值。

設定各個子集比例，並取得資料列數。

總列數乘上比例得到切割的資料點並轉成整數。

將資料點帶回資料集，得到切割後的子集。

程式位置

檔案： split_date_set.py

國立高雄科技大學 AI 金融科技中心版權所有

split_date.py 程式碼2

```
28
29      # 將資料的標籤(y)取出來
30      train_y = train_df["sign"]
31      validate_y = validate_df["sign"]
32      test_y = test_df["sign"]
33
34      # 將資料的特徵(x)取出來
35      train_x = train_df.drop(["日期", "sign"], axis = 1)
36      validate_x = validate_df.drop(["日期", "sign"], axis = 1)
37      test_x = test_df.drop(["日期", "sign"], axis = 1)
38
39      # 正規化到0與1之間
40      scaler = MinMaxScaler(feature_range=(0, 1))
41      scaler.fit(train_x.values)
42
43      # 將數值正規化
44      scaler_train_x = scaler.transform(train_x.values)
45      scaler_validate_x = scaler.transform(validate_x.values)
46      scaler_test_x = scaler.transform(test_x.values)
47
48      # 將正規化後的數值轉回DataFrame
49      train_x = pd.DataFrame(scaler_train_x, columns = train_x.columns)
50      validate_x = pd.DataFrame(scaler_validate_x, columns = validate_x.columns)
51      test_x = pd.DataFrame(scaler_test_x, columns = test_x.columns)
52
53      return train_x, validate_x, test_x, train_y, validate_y, test_y, train_df, validate_df, test_df
54
```

取得資料的特徵x與標籤y。

使用訓練集的特徵x擬合之比例，來正規化所有子集，並轉回df以便後續使用。

總共回傳6個結果，分別為訓練、驗證與測試集的x與y。

程式位置

檔案： split_date_set.py

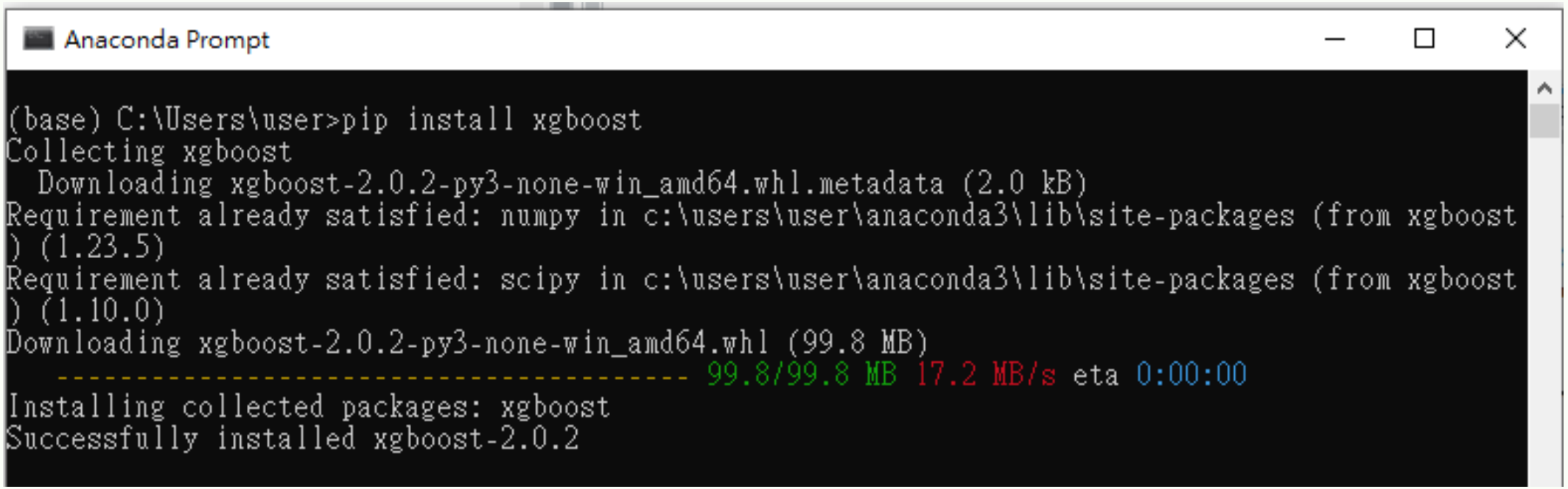
國立高雄科技大學 AI 金融科技中心版權所有

XGBOOST模型_2330.py

國立高雄科技大學 AI 金融科技中心版權所有

安裝套件TensorFlow

- 開啟Anaconda Promt>輸入pip install xgboost



國立高雄科技大學 AI 金融科技中心版權所有

檔案：XGBOOST模型_2330.py

AI.FINTECH
AI 金融科技中心

```python
# -*- coding: utf-8 -*-
import xgboost as xgb
from datetime import datetime as dt
from get_metrics import metrics_summary
from split_data_set import main as sds

save_path = "預測結果/" # 預測結果儲存位置

# 閥值
threshold = 0.5

def main(stock_id):

    train_x, validate_x, test_x, train_y, validate_y, test_y, train_df, validate_df, test_df = sds(stock_id)

    # 設定XGBoost的參數
    params = {
        'objective': 'binary:logistic',
        'max_depth':4,
        'alpha': 0.01,
        'learning_rate': 0.1,
        'n_estimators': 100
    }

    # 開始時間
    start_time = dt.now()
    print(start_time)
```

引用xgboost套件

呼叫sds函數進行分割資料集

設定參數

程式位置
檔案： **XGBOOST模型_2330.py**

國立高雄科技大學 AI 金融科技中心版權所有

檔案：XGBOOST模型_2330.py

```
29      # 創建XGBoost分類器
30      model = xgb.XGBClassifier(**params)
31
32      # 訓練模型
33      model.fit(train_x, train_y, eval_set=[(validate_x, validate_y)], verbose = False)
34
35      validate_pred_y = model.predict(validate_x) # 預測驗證集
36      test_pred_y = model.predict(test_x) # 預測測試集
37
38
39      validate_summary = metrics_summary(validate_pred_y, validate_y)
40      test_summary = metrics_summary(test_pred_y, test_y)
41
42      print("驗證集概要", validate_summary)
43      print("--------------------")
44      print("測試集概要", test_summary)
45      print("--------------------")
46      print("驗證集準確度", validate_summary["準確度 (Accuracy)"])
47      print("測試集準確度", test_summary["準確度 (Accuracy)"])
48      # 結束時間
49      end_time = dt.now()
50      print(end_time)
51      print("耗費時間", end_time - start_time)
52
53
54      validate_df.to_excel(f"{save_path}{stock_id}_驗證.xlsx")
55      test_df.to_excel(f"{save_path}{stock_id}_測試.xlsx")
56
57  main(2330)
```

呼叫xgb的類別XGBClassifier, 產生model物件實體

呼叫fit方法代入訓與驗證資料進行訓練

呼叫predict方法代入測試資料

呼叫metrics_summary代入測試資料計算混淆矩陣

呼叫metrics_summary代入測試資料計算混淆矩陣

顯示測試資料之準確度

程式位置

檔案： XGBOOST模型_2330.py

國立高雄科技大學 AI 金融科技中心版權所有

LSTM_2330.py 執行結果

3. LSTM模型.py × XGBOOST模型_2330.py ×

```
102
103     test_pred_y = model.predict(rolling_test_x) # 預測測試集
104     test_pred_y = test_pred_y > threshold
105     # 找出每一個window的最後一筆資料
106     test_pred_y = test_pred_y[:,-1].flatten().astype(int)
107
108     # 將預測結果放入測試集
109     test_df.loc[:, "LSTM預測"] = pd.NA # 新增一個空欄位
110     test_df.loc[test_df.index[window - 1:], "LSTM預測"] = test_pred_y # 放入預測結果
111
112     # 結束時間
113     end_time = dt.now()
114
116     test_summary = metrics_summary(test_pred_y, rolling_test_y[:,-1].flatten())
117
118     print("-------------------------------------")
119     print("LSTM 測試集概要")
120     print(test_summary)
121     print("-------------------------------------")
122     print("LSTM 測試集準確度 ")
123     print(test_summary["準確度 (Accuracy)"])
124     print("耗費時間", end_time - start_time)
125     validate_df.to_excel(f"{save_path}{stock_id}_驗證.xlsx")
127     test_df.to_excel(f"{save_path}{stock_id}_測試.xlsx")
128
129
130 stock_id = 2330
131 epochs = 256
132 batch_size = 1000
133
134 main(stock_id, epochs, batch_size)
```

Usage

Help Variable Explorer Plots Files

Console 1/A ×

```
0.638095238095238, '召回率 (Recall)': 0.8170731707317073, 'F1分
數 (F1 Score)': 0.7165775401069518, '特異性 (Specificity)':
0.7639751552795031}
-------------------------------------
LSTM 測試集準確度
0.7818930041152263

In [15]: runfile('F:/LISA/金融科技中心/工作坊/AI時代的投資策略：
LSTM股價漲跌預測/NKUST_AIFintech_LSTM/3. LSTM模型.py', wdir='F:/
LISA/金融科技中心/工作坊/AI時代的投資策略：LSTM股價漲跌預測/
NKUST_AIFintech_LSTM')
Reloaded modules: get_metrics, split_data_set
256 1000 1
2023-11-28 17:10:16.207480
-------------------------------------
LSTM 測試集概要
{'真陽性 (True Positive)': 67, '假陽性 (False Positive)': 38, '真
陰性 (True Negative)': 123, '假陰性 (False Negative)': 15, '準確
度 (Accuracy)': 0.7818930041152263, '精確度 (Precision)':
0.638095238095238, '召回率 (Recall)': 0.8170731707317073, 'F1分
數 (F1 Score)': 0.7165775401069518, '特異性 (Specificity)':
0.7639751552795031}
-------------------------------------
LSTM 測試集準確度
0.7818930041152263
耗費時間 0:00:31.840208
```

測試資料之準確度78.19%

執行時間35秒

程式位置

檔案： **LSTM_2330.py**

國立高雄科技大學 AI 金融科技中心版權所有

XGBOOST模型_2330.py

3.LSTM模型.py × XGBOOST模型_2330.py ×

```
19          'max_depth':4,
20          'alpha': 0.01,
21          'learning_rate': 0.1,
22          'n_estimators': 100
23      }
24
25      # 開始時間
26      start_time = dt.now()
27
28      # 創建XGBoost分類器
29      model = xgb.XGBClassifier(**params)
30
31      # 訓練模型
32      model.fit(train_x, train_y, eval_set=[(validate_x, validate_y)], verbose = Fals
33
34      # 預測測試集
35      test_pred_y = model.predict(test_x)
36
37      test_summary = metrics_summary(test_pred_y, test_y)
38      print("---------------------------------------")
39      print("XGBoost 測試集概要")
40      print(test_summary)
41      print("---------------------------------------")
42      print("XGBoost 測試集準確度 ")
44      print(test_summary["準確度 (Accuracy)"])
45      # 結束時間
46      end_time = dt.now()
47      print("耗費時間", end_time - start_time)
48
49      test_df.to_excel(f"{save_path}{stock_id}_測試.xlsx")
50
51
52  main(2330)
```

Usage

Help | Variable Explorer | Plots | Files

Console 1/A ×

```
數 (F1 Score)': 0.761467889908257, '特異性 (Specificity)':
0.8346456692913385}
---------------------------------------
XGBoost 測試集準確度
0.7842323651452282
2023-11-28 16:59:23.568024
耗費時間 0:00:00.699914

In [13]: runfile('F:/LISA/金融科技中心/工作坊/第9回_AI數據分析/選股
+LSTM+NSGA_20231123_差異百分比_T+1_T-2/XGBOOST模型_2330.py',
wdir='F:/LISA/金融科技中心/工作坊/第9回_AI數據分析/選股
+LSTM+NSGA_20231123_差異百分比_T+1_T-2')
Reloaded modules: get_metrics, split_data_set
---------------------------------------
XGBoost 測試集概要
{'真陽性 (True Positive)': 83, '假陽性 (False Positive)': 21, '真
陰性 (True Negative)': 106, '假陰性 (False Negative)': 31, '準確
度 (Accuracy)': 0.7842323651452282, '精確度 (Precision)':
0.7980769230769231, '召回率 (Recall)': 0.7280701754385965, 'F1分
數 (F1 Score)': 0.761467889908257, '特異性 (Specificity)':
0.8346456692913385}
---------------------------------------
XGBoost 測試集準確度
0.7842323651452282
耗費時間 0:00:00.720777
```

測試資料之準確度78.42%

執行時間不到1秒

程式位置
檔案： **XGBOOST模型_2330.py**

國立高雄科技大學 AI 金融科技中心版權所有

AI 實作之參數與資料集的校調

- **機器學習方法(AI的子集), 屬資料導向(Data driven)**
- **以非線性模形大量資料的樣版(Pattern)預測股價走勢**
- **參數與資料集改變會改變預測結果;**
- **資料集校調**
  - **LSTM: 採用前2日收盤價大於隔日收盤價為上漲; 反之為下跌，準確率達78%。**

    XGBoost: 採用前2日收盤價與當日收盤價的百分比，準確率達78%。

國立高雄科技大學 AI 金融科技中心版權所有

參考文獻

1. Han, Y. C., Kim, J., & Enke, D. (2023). A machine learning trading system for the stock market based on N-period Min-Max labeling using XGBoost [Article]. *Expert Systems with Applications, 211, 10, Article 118581.* https://doi.org/10.1016/j.eswa.2022.118581

2. Yun, K. K., Yoon, S. W., & Won, D. (2021). Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process [Article]. *Expert Systems with Applications, 186, 21, Article 115716.* https://doi.org/10.1016/j.eswa.2021.115716

3. Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & Shahab, S. (2020). Deep Learning for Stock Market Prediction [Article]. *Entropy, 22(8), 23, Article 840.* https://doi.org/10.3390/e22080840

4. Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers [Article]. *North American Journal of Economics and Finance, 47, 552-567.* https://doi.org/10.1016/j.najef.2018.06.013

5. Ampomah, E. K., Qin, Z. G., & Nyame, G. (2020). Evaluation of Tree-Based Ensemble Machine Learning Models in Predicting Stock Price Direction of Movement [Article]. *Information, 11(6), 21, Article 332.* *https://doi.org/10.3390/info11060332*

國立高雄科技大學 AI 金融科技中心版權所有

簡報排版非正式參考文獻格式，敬請見諒。

章節到此結束，有任何問題歡迎提出來討論！

國立高雄科技大學AI金融科技中心 版權所有