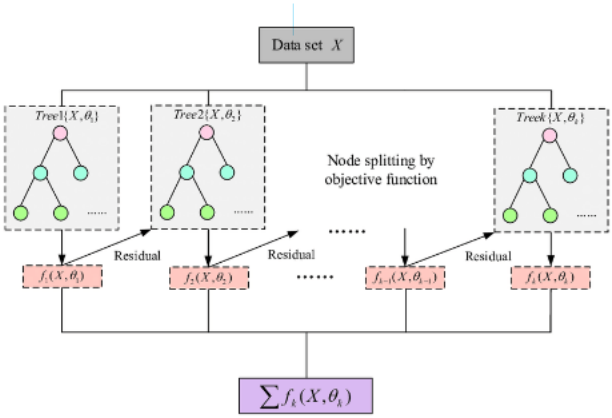
AI VIET NAM – COURSE 2024

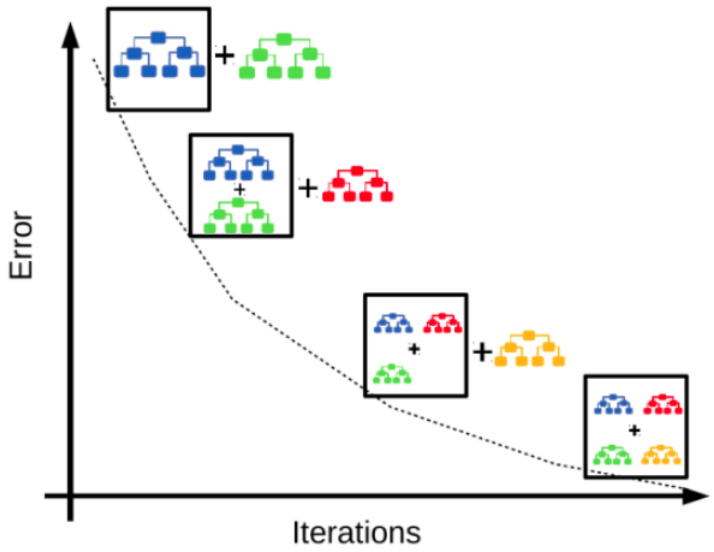
XGBoost

Tho-Anh-Khoa Nguyen, Dang-Nha Nguyen

Ngày 13 tháng 9 năm 2024

PHẦN I: Giới Thiệu





XGBoost là viết tắt của "Extreme Gradient Boosting", một thuật ngữ có nguồn gốc từ bài báo có tiêu đề "Greedy Function Approximation: A Gradient Boosting Machine" của Friedman. Thuật toán này được sử dụng trong các tình huống học có giám sát nơi chúng ta sử dụng dữ liệu với nhiều thuộc tính để dự báo một biến kết quả. Kể từ khi ra mắt vào năm 2014, XGBoost đã trở thành kỹ thuật học máy được ưa chuộng cho nhiều chuyên gia trong lĩnh vực này.

1. **XGBoost for Regression**

Các nhiệm vụ hồi quy (regression) nhằm dự đoán một biến đầu ra liên tục dựa trên một hoặc nhiều biến đầu vào. Trong bối cảnh của XGBoost, các vấn đề hồi quy được tiếp cận bằng cách sử dụng các trees nơi mà các leaf đại diện cho các giá trị đầu ra có tính chất liên tục. Một trong những điểm mạnh của XGBoost trong hồi quy là khả năng mô phỏng các mối quan hệ phi tuyến mà không cần phải biến đổi các đặc trưng. Thêm vào đó, việc kết hợp regularization vào hàm mất mát của nó giúp cho XGBoost có thể chống lại việc overfit tốt hơn so với các mô hình hồi quy truyền thống. Điều này làm cho nó được chú trọng hơn trong các tình huống mà dữ liệu có nhiều đặc trưng hoặc khi mô hình trở nên quá phức tạp.

(a) **Step1**: Khởi tạo giá trị f0 dự đoán của model bằng cách lấy trung bình của Y (giá trị của target)

(b) **Step2**: Tính toán Similarity Score của root

- Similarity Score = $\frac{(SumofResiduals)^2}{NumberofResiduals + \lambda}$
- $SumofResiduals$ là tổng của các giá trị trong Y - f0
- $NumberofResiduals$ là số lượng sample
- λ is regularization parameter

(c) **Step3**: Có nhiều cách chọn điều kiện root, cơ bản nhất là lấy trung bình của 2 sample liên tiếp nhau. Sau đó tính Similarity Score cho các node trong nhánh trái và nhánh phải

(d) **Step4**: Tính Gain cho từng điều kiện của root đã chọn ở trên và chọn ra Gain có giá trị lớn nhất

- Gain = Left Similarity Score + Right Similarity Score - Root Similarity Score

(e) **Step5** Tuỳ vào điều kiện độ sâu của tree mà ta sẽ thực hiện chia nhánh bằng cách lặp lại Step2 đến Step4. Sau đó ta đi tìm output cho root theo điều kiện có gain lớn nhất:

- Output = $\frac{SumofResiduals}{NumberofResiduals}$
- $SumofResiduals$ là tổng của các giá trị trong Y - f0
- $NumberofResiduals$ là số lượng sample

(f) **Step6**: Dùng công thức sau $f0 + lr * Output$ để dự đoán kết quả cho toàn bộ training sample (thay thế cho f0) và tiếp tục thực hiện step 2 đến step 5 cho đến khi thoả mãn điều kiện dừng

2. **XGBoost for Classification**

Các nhiệm vụ phân loại (classification) liên quan đến việc dự đoán các nhãn hoặc danh mục rời rạc (labels) dựa trên các đặc trưng đầu vào. Trong phân loại nhị phân, XGBoost xuất ra một điểm xác suất cho biết khả năng của một sample thuộc về một class cụ thể. Xác suất này sau đó dựa trên một ngưỡng nhất định, thường ở mức 0.5, để gán nhãn lớp.

(a) **Step1**: Khởi tạo giá trị f0 dự đoán của model thông thường là 0.5 (xác suất của dự đoán)

(b) **Step2**: Tính toán Similarity Score của root

- Similarity Score = $\frac{(SumofResiduals)^2}{\sum[PreviousProbability * (1 - PreviousProbability)] + \lambda}$
- $SumofResiduals$ là tổng của các giá trị trong Y (ở dạng xác suất) - f0
- $PreviousProbability$ là xác suất của model trước dự đoán ví dụ ở đây là 0.5
- λ is regularization parameter

(c) **Step3**: Có nhiều cách chọn điều kiện root, cơ bản nhất là lấy trung bình của 2 sample liền kề. Sau đó tính Similarity Score cho các node trong nhánh trái và nhánh phải

(d) **Step4**: Tính Gain cho từng điều kiện của root đã chọn ở trên và chọn ra Gain có giá trị lớn nhất

- Gain = Left Similarity Score + Right Similarity Score - Root Similarity Score

(e) **Step5** Tuỳ vào điều kiện độ sâu của tree mà ta sẽ thực hiện chia nhánh bằng cách lặp lại Step2 đến Step4. Sau đó ta đi tìm output cho root theo điều kiện có gain lớn nhất

- Output = $\frac{SumofResiduals}{\sum[PreviousProbability * (1 - PreviousProbability)]}$
- $SumofResiduals$ là tổng của các giá trị trong Y (ở dạng xác suất) - f0
- $PreviousProbability$ là xác suất của model trước dự đoán ví dụ ở đây là 0.5

(f) **Step6**: Dùng công thức bên dưới để dự đoán kết quả (Probability) cho toàn bộ training sample (thay thế cho f0) và tiếp tục thực hiện step 2 đến step 5 cho đến khi thoả mãn điều kiện dừng

- Cần chọn nhánh phù hợp theo giá trị của X.
- Khi chọn được nhánh phù hợp ta tính Log prediction theo công thức
  - $LogPrediction = log(\frac{PreviousProbability}{1 - PreviousProbability}) + lr * Output$
  - log là Natural logarithm
- Xác suất dự đoán của các sample được tính theo công thức
  - $Probability = \frac{e^{log(LogPrediction)}}{1 + e^{log(LogPrediction)}}$

PHẦN II: Bài Tập

1. **(XGBoost cho bài toán Regression) Dựa vào thuật toán miêu tả ở trên cho bài toán Regression, thực hiện tính toán từng bước theo các yêu cầu bên dưới với tập data như sau**

X	Y	f0 = 71.25
23	50	-21.25
24	70	-1.25
26	80	8.75
27	85	13.75

Hình 1: X là input, Y là target

(a) Build tree với điều kiện sau $\lambda = 0$, depth = 1, lr = 0.3

(b) **Step1**: Khởi tạo giá trị f0 dự đoán của model bằng cách lấy trung bình của Y

(c) **Step2**: Tính toán Similarity Score của root

- Similarity Score = $\frac{(Sum of Residuals)^2}{Number of Residuals + \lambda}$ 0
- $Sum of Residuals$ là tổng của các giá trị trong Y - f0
- $Number of Residuals$ là số lượng sample

(d) **Step3**: Chọn root theo từng yêu cầu sau và tính Similarity Score cho các node trong nhánh trái và nhánh phải

Gain

i. X < 23.5 451.5625 150.5208 602.0833 left = -21.25 right = 7.0833
ii. X < 25 253.125 253.125 506.25
iii. X < 26.5 63.0208 189.0625 252.0833

(e) **Step4**: Tính Gain cho từng case i, ii, iii ở trên và chọn ra Gain có giá trị lớn nhất

- Gain = Left Similarity Score + Right Similarity Score - Root Similarity Score

(f) **Step5** Sau khi chọn được model cho gain có giá trị lớn nhất ta tính giá Ouput cho từng node trong nhánh trái và phải theo công thức sau:

- Output = $\frac{Sum of Residuals}{Number of Residuals}$
- $Sum of Residuals$ là tổng của các giá trị trong Y - f0
- $Number of Residuals$ là số lượng sample

(g) **Step6**: Bây giờ ta sẽ thực hiện dự đoán kết quả khi x = 25 theo công thức bên dưới

- Cần chọn nhánh phù hợp theo giá trị của X.
- Khi chọn được nhánh phù hợp ta lấy giá trị Ouput của nhánh và thực hiện dự đoán theo công thức $f0 + lr * Output$ 73.37499

2. **(XGBoost cho bài toán Classification) Dựa vào thuật toán miêu tả ở trên cho bài toán Classification, thực hiện tính toán từng bước theo các yêu cầu bên dưới với tập data như sau**

X	Y	
23	False	- 0.5
24	False	-0.5
26	True	0.5
27	True	0.5

Hình 2: X là input, Y là target

(a) Build tree với điều kiện sau $\lambda = 0$, depth $= 1$, lr $= 0.3$. Ta sẽ biến đổi giá trị của các sample trong Y thành xác suất, tương ứng False là 0 và True là 1 f0 = 0.5

(b) **Step1**: Khởi tạo giá trị f0 dự đoán của model là một hằng số 0.5 (xác suất của dự đoán)

(c) **Step2**: Tính toán Similarity Score của root 0

- Similarity Score $= \frac{(SumofResiduals)^2}{\sum[PreviousProbability * (1 - PreviousProbability)] + \lambda}$
- $SumofResiduals$ là tổng của các giá trị trong Y (ở dạng xác suất) - f0
- $PreviousProbability$ là xác suất của model trước dự đoán ví dụ ở đây là 0.5

(d) **Step3**: Chọn root theo từng yêu cầu sau và tính Similarity Score cho các node trong nhánh trái và nhánh phải

i. X $<$ 23.5 1 0.333
ii. X $<$ 25 2 2 Gain = 4: left = -2 right = 2
iii. X $<$ 26.5 0.333 1

(e) **Step4**: Tính Gain cho từng case i, ii, iii ở trên và chọn ra Gain có giá trị lớn nhất

- Gain = Left Similarity Score + Right Similarity Score - Root Similarity Score

(f) **Step5** Sau khi chọn được model cho gain có giá trị lớn nhất ta tính giá Ouput cho từng node trong nhánh trái và phải theo công thức sau:

- Output $= \frac{SumofResiduals}{\sum[PreviousProbability * (1 - PreviousProbability)]}$
- $SumofResiduals$ là tổng của các giá trị trong Y (ở dạng xác suất) - f0
- $PreviousProbability$ là xác suất của model trước dự đoán ví dụ ở đây là 0.5

(g) **Step6**: Bây giờ ta sẽ thực hiện dự đoán kết quả khi x $= 25$ theo công thức bên dưới

- Cần chọn nhánh phù hợp theo giá trị của X.
- Khi chọn được nhánh phù hợp ta tính Log prediction theo công thức
  - $LogPrediction = log(\frac{PreviousProbability}{1 - PreviousProbability}) + lr * Output$
  - log là Natural logarithm
- Xác suất dự đoán của x $= 25$ được tính theo công thức
  - $Probability = \frac{e^{LogPrediction}}{1 + e^{LogPrediction}}$ 0.646

Hình 3: Forest fire prediction

3. **(XGBoost Regressor) Bài tập này chúng ta sẽ tập trung vào việc triển khai và đào tạo mô hình XGBoost. Mục tiêu chính của chúng ta là sử dụng mô hình này để giải quyết một bài toán regresion, cụ thể là dự đoán diện tích khu vực bị cháy trong rừng. Để thực hiện điều này, chúng ta sẽ tuân theo một số bước cụ thể mà sẽ được trình bày sau đây.**

   (a) Trước khi bắt đầu quá trình xử lý dữ liệu và huấn luyện mô hình, việc đầu tiên chúng ta cần làm là nhập các thư viện cần thiết. Trong đó 'xgboost' là thư viện mà chúng ta sẽ sử dụng để cài đặt và huấn luyện mô hình XGBoost.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
```

   (b) Load data để xử lý và phân tích. Chúng ta sẽ sử dụng phương thức read_csv() của pandas để đọc dữ liệu từ một tệp .csv. (Tải data tại đây)

```
dataset_path = '/content/Problem3.csv'
data_df = pd.read_csv(dataset_path)
data_df
```

   Khi đó, ta có thể thấy nội dung của bảng dữ liệu có dạng như sau:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	4.468204	26.2	94.3	1.808289	8.2	51	6.7	False	0.000000
1	7	4	oct	tue	4.517431	35.4	669.1	2.041220	18.0	33	0.9	False	0.000000
2	7	4	oct	sat	4.517431	43.7	686.9	2.041220	14.6	33	1.3	False	0.000000
3	8	6	mar	fri	4.529368	33.3	77.5	2.302585	8.3	97	4.0	True	0.000000
4	8	6	mar	sun	4.503137	51.3	102.2	2.360854	11.4	99	1.8	False	0.000000

Hình 4: Một vài mẫu dữ liệu trong data

Trong bài toán dự đoán này, chúng ta quan tâm đến việc ước lượng diện tích khu vực bị cháy trong rừng dựa trên một loạt các thuộc tính được cung cấp. Diện tích khu vực cháy, được lưu trữ trong cột "area", ta sẽ gán giá trị từ cột này vào biến y. Những thuộc tính còn lại sẽ được gán vào biến X.

(c) Trong data chúng ta nhận thấy rằng sẽ có cột month và day là dạng string và cột rain là dạng bool. Do đó, ta sẽ phải đổi toàn bộ các giá trị này về dạng số.

```
categorical_cols = data_df.select_dtypes(include=['object', 'bool']).columns.
    to_list()

for col_name in categorical_cols:
    n_categories = data_df[col_name].nunique()
    print(f'Number of categories in {col_name}: {n_categories}')

ordinal_encoder = OrdinalEncoder()
encoded_categorical_cols = ordinal_encoder.fit_transform(data_df[
    categorical_cols])

encoded_categorical_df = pd.DataFrame(
  encoded_categorical_cols ,
  columns = categorical_cols
)

numerical_df = data_df.drop(categorical_cols , axis =1)
encoded_df = pd.concat([numerical_df, encoded_categorical_df ] , axis =1)
```

Khi đã hoàn tất, ta được một DataFrame mới có dạng như sau:

	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	area	month	day	rain
0	7	5	4.468204	26.2	94.3	1.808289	8.2	51	6.7	0.000000	7.0	0.0	0.0
1	7	4	4.517431	35.4	669.1	2.041220	18.0	33	0.9	0.000000	10.0	5.0	0.0
2	7	4	4.517431	43.7	686.9	2.041220	14.6	33	1.3	0.000000	10.0	2.0	0.0
3	8	6	4.529368	33.3	77.5	2.302585	8.3	97	4.0	0.000000	7.0	0.0	1.0
4	8	6	4.503137	51.3	102.2	2.360854	11.4	99	1.8	0.000000	7.0	3.0	0.0

Hình 5: Bộ dữ liệu sau khi đã chuyển đổi toàn bộ các đặc trưng Categorical về dạng số

(d) Vì bài toán của chúng ta là supervised learning nên tập data train cần có input các đặc trưng (X) và target/label (y), do đó ta sẽ tách các cột dữ liệu thành hai biến X, y. Trong bài toán này target/label là 'area' diện tích cháy rừng

```
X = encoded_df.drop(columns=['area'])
y = encoded_df['area']
```

(e) Dựa vào bộ dữ liệu gốc, ta cần chia thành hai tập dữ liệu con, một dùng cho việc huấn luyện mô hình và một cho việc đánh giá mô hình. Ở đây, ta sẽ chia theo tỷ lệ 7:3 và tham số ngẫu nhiên random_state = 7:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=7
)
```

(f) Chúng ta xây dựng model XGBoost cho Regression task với các than số sau seed=7, learning_rate=0.01, n_estimators=102, max_depth=3. Sau đó ta sẽ huấn luyện mô hình với tập data train

```
xg_reg = xgb.XGBRegressor(seed=7,
                          learning_rate=0.01,
                          n_estimators=102,
                          max_depth=3
)

xg_reg.fit(X_train, y_train)
```

(g) Để đảm bảo mô hình của chúng ta không chỉ hoạt động tốt trên dữ liệu huấn luyện mà còn có khả năng tổng quát hóa tốt trên dữ liệu mới, việc sử dụng một tập test là rất quan trọng. Tập test này sẽ chứa một phần dữ liệu mà mô hình chưa từng thấy trong quá trình huấn luyện. Đầu tiên, ta cho mô hình đã huấn luyện thực hiện dự đoán trên toàn bộ tập test:

```
preds = xg_reg.predict(X_test)
```

Để thực hiện đánh giá mô hình trong bài toán regression thì Mean Absolute Error (MAE) và Mean Squared Error (MSE) là hai trong số những độ đo phổ biến nhất được sử dụng:

1. **Mean Absolute Error (MAE)**: Độ đo này cho chúng ta biết trung bình sự chênh lệch giữa giá trị dự đoán và giá trị thực tế. Nó được tính bằng cách lấy trung bình của trị tuyệt đối của sự khác biệt giữa giá trị dự đoán và giá trị thực sự.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Trong đó: - y_i là giá trị thực sự. - $\hat{y}_i$ là giá trị dự đoán. - n là số lượng mẫu trong tập dữ liệu.

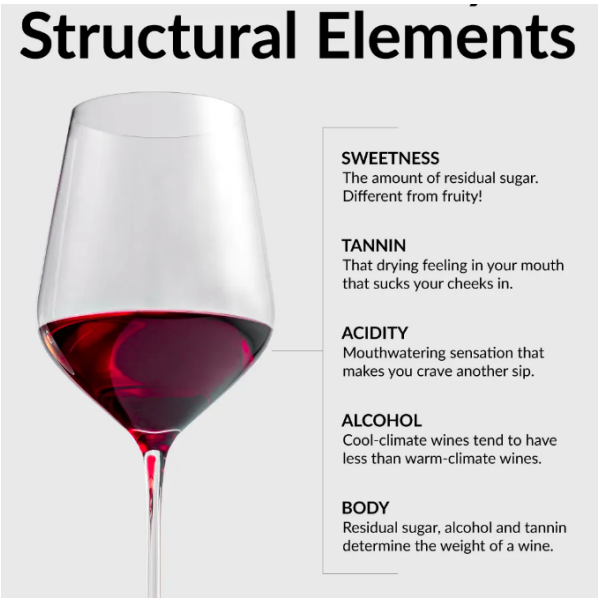
2. **Mean Squared Error (MSE)**: Tương tự như MAE, nhưng thay vì lấy trị tuyệt đối của sự khác biệt, MSE sẽ lấy bình phương của sự khác biệt. Điều này có nghĩa là các giá trị dự đoán xa giá trị thực tế sẽ bị trừng phạt nặng hơn.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Cả hai độ đo này cung cấp góc nhìn khác nhau về hiệu suất của mô hình. MAE cho chúng ta biết sự chênh lệch trung bình giữa dự đoán và thực tế, trong khi MSE tập trung nhiều hơn vào việc trừng phạt các dự đoán sai lệch lớn. Kết hợp cả hai giúp chúng ta có cái nhìn toàn diện hơn về chất lượng của mô hình.

```
mae = mean_absolute_error(y_test, preds)
mse = mean_squared_error(y_test, preds)

print('Evaluation results on test set:')
print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
```



4. **(XGBoost Classifier) Bài tập này chúng ta sẽ tập trung vào việc triển khai và đào tạo mô hình XGBoost. Mục tiêu chính của chúng ta là sử dụng mô hình này để giải quyết một bài toán classification, cụ thể là dự đoán chất lượng của wine (3 classes: 0, 1, 2). Để thực hiện điều này, chúng ta sẽ tuân theo một số bước cụ thể mà sẽ được trình bày sau đây.**

   (a) Trước khi bắt đầu quá trình xử lý dữ liệu và huấn luyện mô hình, việc đầu tiên chúng ta cần làm là nhập các thư viện cần thiết. Trong đó 'xgboost' là thư viện mà chúng ta sẽ sử dụng để cài đặt và huấn luyện mô hình XGBoost.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

   (b) Load data để xử lý và phân tích. Chúng ta sẽ sử dụng phương thức read_csv() của pandas để đọc dữ liệu từ một tệp .csv. (Tải data tại đây)

```
dataset_path = '/content/Problem4.csv'
data_df = pd.read_csv(dataset_path)
data_df
```

   Khi đó, ta có thể thấy nội dung của bảng dữ liệu có dạng như sau:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	Target
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0

Hình 6: Một vài mẫu dữ liệu trong data

Trong bài toán dự đoán này, chúng ta quan tâm đến việc dự đoán chất lượng của rượu vang (loại 0, 1, 2) dựa trên một loạt các thuộc tính được cung cấp. Loại chất lượng của rượu vang,

được lưu trữ trong cột "Target", ta sẽ gán giá trị từ cột này vào biến y (là cột cuối cùng). Những thuộc tính còn lại sẽ được gán vào biến X.

(c) Trong bộ dữ liệu chỉ chứa các đặc trưng dạng số và không có bất kỳ đặc trưng dạng danh mục (categorical) nào, thì không cần thiết phải thực hiện bước mã hóa (encoding).

(d) Vì bài toán của chúng ta là supervised learning nên tập data train cần có input các đặc trưng (X) và target/label (y), do đó ta sẽ tách các cột dữ liệu thành hai biến X, y. Trong bài toán này target/label là 'Target' là loại chất lượng và là cột cuối cùng

```
X, y = data_df.iloc[:,:-1],data_df.iloc[:,-1]
```

(e) Dựa vào bộ dữ liệu gốc, ta cần chia thành hai tập dữ liệu con, một dùng cho việc huấn luyện mô hình và một cho việc đánh giá mô hình. Ở đây, ta sẽ chia theo tỷ lệ 7:3 và tham số ngẫu nhiên random_state = 7:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=7
)
```

(f) Chúng ta xây dựng model XGBoost cho Classification task với các tham số sau seed=7. Sau đó ta sẽ huấn luyện mô hình với tập data train

```
xg_class = xgb.XGBClassifier(seed=7)

xg_class.fit(X_train, y_train)
```

(g) Để đảm bảo mô hình của chúng ta không chỉ hoạt động tốt trên dữ liệu huấn luyện mà còn có khả năng tổng quát hóa tốt trên dữ liệu mới, việc sử dụng một tập test là rất quan trọng. Tập test này sẽ chứa một phần dữ liệu mà mô hình chưa từng thấy trong quá trình huấn luyện. Đầu tiên, ta cho mô hình đã huấn luyện thực hiện dự đoán trên toàn bộ tập test:

```
preds = xg_class.predict(X_test)
```

Đối với bài toán phân loại (Classification), việc đánh giá hiệu suất mô hình thường sử dụng các độ đo khác nhau so với bài toán hồi quy (Regression). Trong bài toán phân loại, "accuracy" là một trong những độ đo cơ bản và phổ biến nhất.

Accuracy (Độ chính xác): Độ chính xác cho biết tỷ lệ mẫu được phân loại đúng trong tất cả các mẫu của tập dữ liệu. Nó được tính bằng cách lấy số mẫu được phân loại đúng chia cho tổng số mẫu.

$$\text{Accuracy} = \frac{\text{Số mẫu phân loại đúng}}{\text{Tổng số mẫu}}$$

Tuy nhiên, cần lưu ý rằng trong một số trường hợp, đặc biệt là khi dữ liệu không cân bằng (một lớp có số lượng mẫu nhiều hơn nhiều so với lớp khác), độ chính xác không phản ánh đúng hiệu suất của mô hình. Trong những trường hợp như vậy, chúng ta cần xem xét các độ đo khác như Precision, Recall, F1-score, ROC, AUC, vv.

Nhưng nếu bạn chỉ muốn một cái nhìn tổng quan và đơn giản về hiệu suất mô hình, thì accuracy là một lựa chọn tốt để bắt đầu.

```
train_acc = accuracy_score(y_train, xg_class.predict(X_train))
test_acc = accuracy_score(y_test, preds)

print(f'Train ACC: {train_acc}')
print(f'Test ACC: {test_acc}')
```

III. Câu hỏi trắc nghiệm

Câu hỏi 1: Trong số các phương án sau, cái nào là đúng về hyperparameter "max_depth" trong Gradient Boosting?

1. Giá trị thấp hơn là tốt hơn trong trường hợp độ chính xác validation giống nhau.
2. Giá trị cao hơn là tốt hơn trong trường hợp độ chính xác validation giống nhau.
3. Tăng giá trị của max_depth có thể làm mô hình overfit với dữ liệu.
4. Tăng giá trị của max_depth có thể làm mô hình underfit với dữ liệu.

(A). 1 và 3
(B). 2 và 4
(C). 1 và 4
(D). 2 và 3

Câu hỏi 2: Giả sử bạn đã được cung cấp các kịch bản sau cho Training Error và Test Error trong Gradient Boosting. Trong trường hợp như vậy, bạn sẽ chọn hyperparameter nào?

Case	Depth	Training Error	Test Error
1	2	100	120
2	4	80	115
3	6	50	105
4	8	40	105

(A). Case 1
(B). Case 2
(C). Case 3
(D). Case 4

Câu hỏi 3: Đối với câu 1 trong phần bài tập thì giá trị khởi tạo của f0 bằng bao nhiêu?

(A). 71.25
(B). 70.25
(C). 72.25
(D). 69.25

Câu hỏi 4: Đối với câu 1 trong phần bài tập, ở step 2 thì giá trị Similarity Score của root bằng bao nhiêu?

(A). 0.25
(B). 0.0
(C). 0.5
(D). 1.25

Câu hỏi 5: Đối với câu 1 trong phần bài tập, ở step 3 thì giá trị Similarity Score của nhánh bên trái trong trường hợp $X < 23.5$ bằng bao nhiêu?

(A). ≈ 450.5625
(B). ≈ 450.3625
(C). ≈ 451.5625
(D). ≈ 451.3625

Câu hỏi 6: Đối với câu 1 trong phần bài tập, ở step 4 thì giá trị Gain trong trường hợp $X < 25$ bằng bao nhiêu?

(A). ≈ 1009.5
(B). ≈ 1010.5
(C). ≈ 1011.5
(D). Tất cả đều sai

Câu hỏi 7: Đối với câu 2 trong phần bài tập, ở step 2 thì giá trị Residual của X=23 thì bằng bao nhiêu?

(A). -0.5
(B). 0.0
(C). 1.0
(D). 0.5

Câu hỏi 8: Đối với câu 2 trong phần bài tập, ở step 2 thì giá trị Similarity Score của root bằng bao nhiêu?
(A). 0.25
(B). 0.0
(C). 0.5
(D). 1.25

Câu hỏi 9: Đối với câu 2 trong phần bài tập, ở step 3 thì giá trị Similarity Score của nhánh bên trái trong trường hợp $X < 26.5$ bằng bao nhiêu?
(A). ≈ 0.0
(B). ≈ 0.1
(C). ≈ 0.33
(D). ≈ 0.5

Câu hỏi 10: Đối với câu 2 trong phần bài tập, ở step 4 thì giá trị Gain trong trường hợp $X < 25$ bằng bao nhiêu?
(A). ≈ 1.0
(B). ≈ 2.0
(C). ≈ 3.0
(D). ≈ 4.0

Câu hỏi 11: Dựa vào hướng dẫn lập trình của câu 3 trong phần bài tập, kết quả đánh giá MAE của mô hình XGBosst cho regression task đã huấn luyện được trên tập test là bao nhiêu (lấy giá trị gần đúng nhất)?
(A). ≈ 1.0923
(B). ≈ 2.0923
(C). ≈ 3.0923
(D). ≈ 4.0923

Câu hỏi 12: Dựa vào hướng dẫn lập trình của câu 4 trong phần bài tập, kết quả đánh giá accuracy score của mô hình XGBosst cho classification task đã huấn luyện được trên tập test là bao nhiêu (lấy giá trị gần đúng nhất)?
(A). ≈ 0.8815
(B). ≈ 0.9815
(C). ≈ 0.7815
(D). ≈ 0.6815

IV. (Đọc Thêm) Time Series Classification với XGBoost

Trong bài hướng dẫn này, chúng ta sẽ thực hiện phân loại chuỗi thời gian bằng thuật toán XGBoost trên tập dữ liệu FordA. Tập dữ liệu này chứa các sample có length là 500 để đo tiếng ồn cửa động cơ thông qua một motor sensor, và nhiệm vụ của chúng ta là phân loại động cơ có đang gặp vấn đề (+1) hay không (-1) dựa trên tiếng ồn. Mục tiêu là huấn luyện một bộ phân loại XGBoost và đánh giá hiệu suất của nó.

1 Phần lập trình

1.1 Import các thư viện cần thiết

Đầu tiên, chúng ta import các thư viện cần thiết.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

1.2 Mô tả data và tải data

File notebook cung cấp cho các bạn đã bao gồm lệnh tải data cần thiết (chúng ta sẽ tải tập train "FordA_TRAIN" và tập test "FordA_TEST").

Bộ dataset mà chúng ta sử dụng trong trường hợp này được gọi là FordA, lấy từ kho lưu trữ UCR. Dataset này bao gồm 3601 mẫu cho việc huấn luyện và 1320 mẫu cho việc kiểm thử. Mỗi timeseries trong dataset đại diện cho dữ liệu tiếng ồn động cơ được thu thập thông qua cảm biến động cơ. Mục tiêu của task này là tự động phát hiện các sự cố động cơ. Đây là một bài toán balanced binary classification. Các bạn có thể tìm hiểu chi tiết về dataset này (tại đây).

1.3 Đọc file data

Chúng ta sẽ dùng file FordA_TRAIN để huấn luyện và file FordA_TEST để kiểm thử. Sự đơn giản của dataset này giúp chúng ta dễ dàng minh họa hiệu quả của việc áp dụng XGBoost cho việc phân loại timeseries. Trong các tệp này, cột đầu tiên chứa nhãn của dữ liệu.

```
def readucr(filename):
    data = np.loadtxt(filename, delimiter="\t")
    y = data[:, 0]
    x = data[:, 1:]
    return x, y.astype(int)

x_train, y_train = readucr("FordA_TRAIN.tsv")
x_test, y_test = readucr("FordA_TEST.tsv")
```

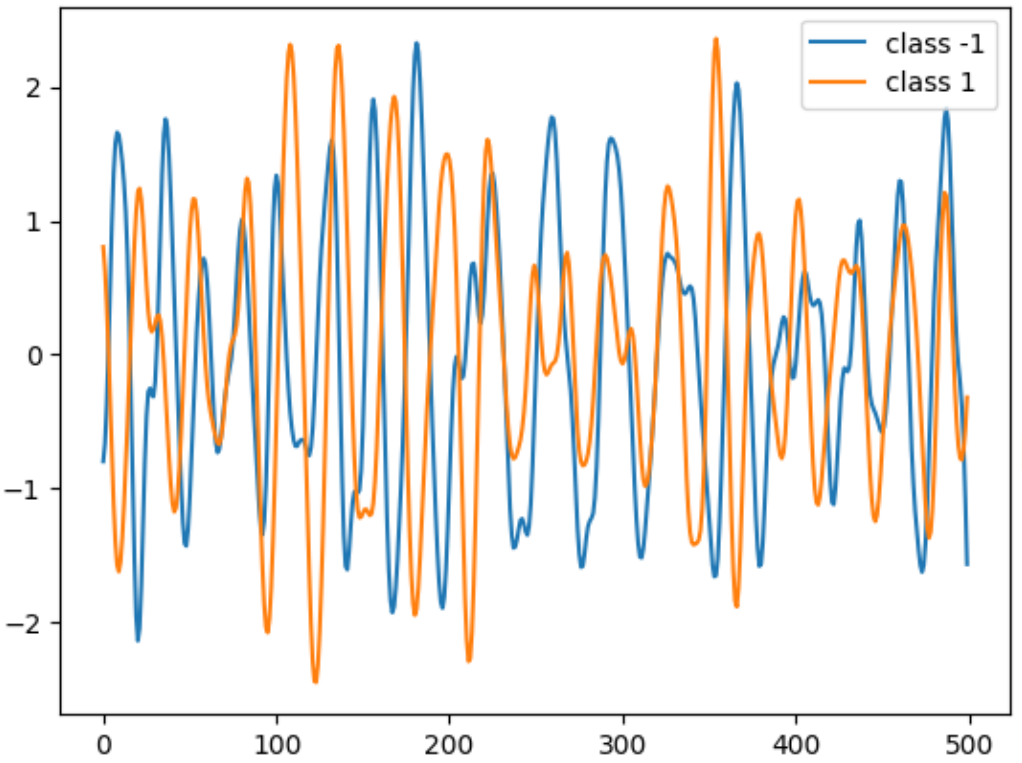
1.4 Visualize data

Chúng ta sẽ trực quan hóa một chuỗi sample cho mỗi class trong dataset.

```
classes = np.unique(np.concatenate((y_train, y_test), axis=0))

plt.figure()
for c in classes:
    c_x_train = x_train[y_train == c]
```

```
    plt.plot(c_x_train[0], label="class " + str(c))
plt.legend(loc="best")
plt.show()
plt.close()
```



Hình 7: Chuỗi sample tiếng ồn thể hiện động cơ có vấn đề (1) và không có vấn đề (-1)

1.5 Khai báo, huấn luyện mô hình XGBoost

Đối với dataset này, dữ liệu đã được chuẩn hóa: mỗi sample có trung bình bằng không và độ lệch chuẩn bằng một.

Lưu ý rằng dữ liệu timeseries được sử dụng ở đây là univariate, có nghĩa là chúng ta chỉ có một kênh cho mỗi sample.

Chúng ta sẽ biến đổi lable của class -1 sang 0 để sau này có thể thực hiện cho nhiều model khác.

```
y_train[y_train == -1] = 0
y_test[y_test == -1] = 0
```

Thực hiện gọi tới mô hình XGBClassifier từ thư viện xgboost, truyền vào các tham số cần thiết để khởi tạo mô hình. Gọi đến phương thức fit() của đối tượng model, truyền vào X_train, y_train để thực hiện huấn luyện.

```
model = XGBClassifier(n_estimators=200, random_state=42)
model.fit(x_train, y_train)
```

1.6 Đánh giá mô hình

Sử dụng confusion matrix và các chỉ số precision, recall, f1-score để đánh giá khả năng của mô hình

```python
y_pred = model.predict(x_test)

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("Classification Report:")
print(classification_report(y_test, y_pred))

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

Tham khảo thêm ở link.

—— Hết ——