BUỔI 3. GIẢI THUẬT CÂY QUYẾT ĐỊNH

Mục đích:

- Củng cố lý thuyết và có thể cài đặt giải thuật cây quyết định.
- Kiểm thử và đánh giá kết quả sau khi huấn luyện mô hình.

3.1. Cài đặt giải thuật cây quyết định phân lớp

Để huấn luyện mô hình với giải thuật cây quyết định, chúng ta sử dụng tập dữ liệu Outlook

https://www.kaggle.com/datasets/sachinrawat782/outlook-dataset

Dữ liệu có 2 thuộc tính là **Weather (**Thời tiết), **Temp (**Nhiệt độ) và nhãn là **Play (** chơi hoặc không chơi)

- Đọc dữ liệu và hiển thị lên các thông tin về tập dữ liệu này để có cái nhìn trực quan.
- Sử dụng nghi thức "hold out" theo tỉ lệ 8-2 (80% cho tập huấn luyện, 20% cho tập kiểm tra).

```
# Đọc dữ liệu từ file Excel
data = pd.read_excel('Play.xlsx', engine='openpyxl')

# Định nghĩa biến đặc trưng và biến mục tiêu
X = data[['Weather', 'Temp']]
y = data['Play']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Vì giá trị các biến hiện tại kiểu chữ, hàm DecisionTreeClassifier thực hiện trên kiểu số, nên cần mã hóa thành dạng số trước khi đưa vào huấn luyện mô hình.

```python
# Vì các biến đặc trưng là biến phân loại, ta cần mã hóa chúng thành dạng số trước khi huấn luyện mô hình
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for col in X_train.columns: # Với mỗi cột trong tập đặc trưng
    X_train[col] = le.fit_transform(X_train[col]) # Áp dụng Label Encoding cho tập huấn luyện
    X_test[col] = le.transform(X_test[col]) # Áp dụng Label Encoding cho tập kiểm tra dựa trên mã hóa của tập huấn luyện
```

Chúng ta có thể hiển thị kết quả sau khi mã hóa để kiểm tra

```python
print(f'X_train:\n{X_train}\n')
```

```
X_train:
    Weather  Temp
12        0     1
5         1     0
8         2     0
2         0     1
1         2     1
13        1     2
4         1     0
7         2     2
10        2     2
3         1     2
6         0     0
```

Sau khi phân chia xong tập dữ liệu, chúng ta tiến hành xây dựng mô hình với giải thuật cây quyết định bằng đoạn code sau:

```python
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier() # Tạo mô hình cây quyết định phân loại

model.fit(X_train, y_train) # Huấn luyện mô hình với tập huấn luyện
y_pred = model.predict(X_test) # Dự đoán trên tập kiểm tra
```

Một số tham số cơ bản của hàm DecisionTreeClassifier:

+ **criterion** – chỉ số để phân hoạch dữ liệu trên cây (mặc định là ‘gini’). Nếu chúng ta muốn sử dụng độ lợi thông tin thì sử dụng ‘entropy’.

+ **max_depth** – Độ sâu tối đa của cây, nếu không thiết lập thì mặc định sẽ mở rộng cây đến các nút lá kết quả (cây có thể sẽ khá lớn nếu tập dữ liệu lớn).

+ **max_features** – Số lượng thuộc tính tối đa sử dụng để xây dựng cây, mặc định lấy tất cả.

+ **min_samples_split** – Số lượng mẫu tối thiểu cần có trong một nút để nút đó được phép tách.

+ **min_samples_leaf** – Số lượng mẫu tối thiểu trong mỗi nút lá sau khi tách.

Các tham số khác, chúng ta có thể tìm hiểu thêm tại đây:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Sau khi cài đặt xong giải thuật cây quyết định, chúng ta tiến hành đánh giá kết quả như sau:

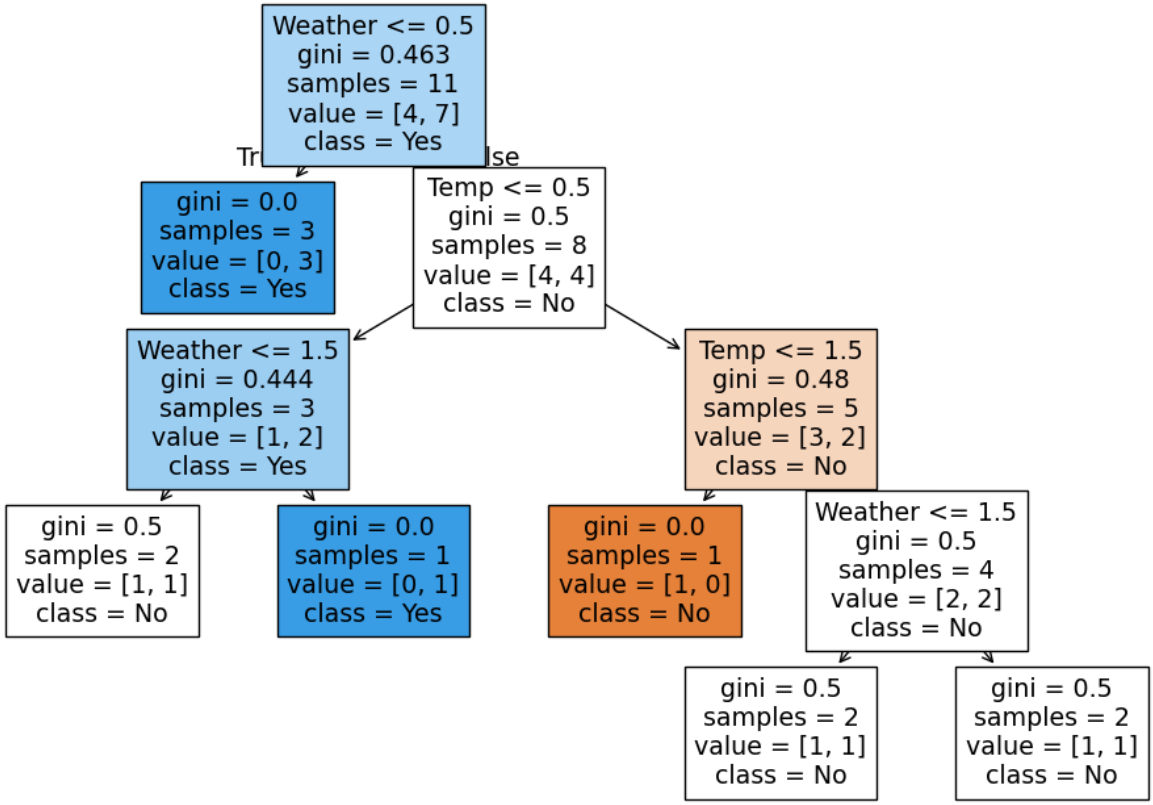
```
# Đánh giá mô hình
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print('Classification Report:')
print(classification_report(y_test, y_pred))
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

✓ 0.0s

```
Accuracy: 0.67
Classification Report:
              precision    recall  f1-score   support

          No       0.50      1.00      0.67         1
         Yes       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3

Confusion Matrix:
[[1 0]
 [1 1]]
```

Ta có thể trực quan hóa cây quyết định vừa huấn luyện được bằng hàm plot_tree do thư viện sklearn cung cấp:

```
# Vẽ cây quyết định
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
plot_tree(model, feature_names=X.columns, class_names=model.classes_, filled=True)
plt.show()
```



3.2. Cài đặt giải thuật cây quyết định hồi quy

Để huấn luyện mô hình với giải thuật cây quyết định, chúng ta sử dụng tập dữ liệu giá nhà ở ở Boston tại đường dẫn:

https://www.kaggle.com/datasets/vikrishnan/boston-house-prices

Trước tiên, chúng ta sẽ tiến hành phân tích tập dữ liệu như sau:

Bộ dữ liệu này sử dụng 13 thuộc tính đầu vào như sau:

- **crim:** Tỷ lệ phạm tội phạm bình quân đầu người theo thị trấn.
- **zn:** Tỷ lệ đất ở được quy hoạch cho các lô trên 25.000 foot square.
- **indus:** Tỷ lệ diện tích thuộc lĩnh vực kinh doanh phi bán lẻ trên mỗi thị trấn.
- **chas:** Biến giả, = 1 nếu được bao bởi sông Charles River, = 0 nếu ngược lại.
- **nox:** Nồng độ khí Nitơ oxit.
- **rm:** Trung bình số phòng trên một căn hộ.

- **age:** Tỷ lệ căn hộ được xây dựng trước năm 1940.
- **dis:** Khoảng cách trung bình có trọng số tới 5 trung tâm việc làm lớn nhất ở Boston.
- **rad:** Chỉ số về khả năng tiếp cận đường cao tốc.
- **tax:** Giá trị thuế suất tính trên đơn vị 10.000$.
- **ptratio:** Tỷ lệ học sinh-giáo viên trên mỗi thị trấn.
- **black:** Tỷ lệ số người da đen trong thị trấn.
- **lstat:** Tỷ lệ phần trăm dân số thu nhập thấp.

Giá trị nhãn: Giá nhà trung bình.

Ở phần minh họa này, chúng ta sẽ chia tập dữ liệu ra thành hai phần theo tỷ lệ 8:2 (80% cho tập huấn luyện, 20% cho tập kiểm tra).

Đầu tiên, chúng ta cần load dữ liệu vào và hiển thị lên các thông tin về tập dữ liệu này bằng đoạn code sau:

```
import pandas as pd
data = pd.read_csv('housing.csv', header=None, delim_whitespace=True)
print(data.head())
```

✓ 0.0s

```
        0     1     2  3      4      5     6       7  8      9     10  \
0 0.00632  18.0  2.31  0  0.538  6.575  65.2  4.0900  1  296.0  15.3
1 0.02731   0.0  7.07  0  0.469  6.421  78.9  4.9671  2  242.0  17.8
2 0.02729   0.0  7.07  0  0.469  7.185  61.1  4.9671  2  242.0  17.8
3 0.03237   0.0  2.18  0  0.458  6.998  45.8  6.0622  3  222.0  18.7
4 0.06905   0.0  2.18  0  0.458  7.147  54.2  6.0622  3  222.0  18.7

       11    12    13
0  396.90  4.98  24.0
1  396.90  9.14  21.6
2  392.83  4.03  34.7
3  394.63  2.94  33.4
4  396.90  5.33  36.2
```

Tiếp đó chúng ta cần xác định các biến đặc trưng và biến mục tiêu của tập dữ liệu trên:

```
# Xác định các đặc trưng và biến mục tiêu
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

Vì dữ liệu ở các đặc trưng có giá trị lệch nhau (ví dụ: cột 11 có miền giá trị từ 0 – 400 còn cột 0 có giá trị trong khoảng từ 0 – 1) điều này gây mất cân bằng về mức độ quan trọng giữa các thuộc tính. Do đó ta cần chuẩn hóa dữ liệu trước khi phân chia thành tập train và test như sau:

```
# chuẩn hóa dữ liệu nếu bằng min-max scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled[:2])

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

✓ 0.0s

```
[0.00000000e+00 1.80000000e-01 6.78152493e-02 0.00000000e+00
 3.14814815e-01 5.77505269e-01 6.41606591e-01 2.69203139e-01
 0.00000000e+00 2.08015267e-01 2.87234043e-01 1.00000000e+00
 8.96799117e-02]
[2.35922539e-04 0.00000000e+00 2.42302053e-01 0.00000000e+00
 1.72839506e-01 5.47997701e-01 7.82698249e-01 3.48961980e-01
 4.34782609e-02 1.04961832e-01 5.53191489e-01 1.00000000e+00
 2.04470199e-01]]
```

Sau khi đã chuẩn hóa dữ liệu chúng ta sẽ tiến hành xây dựng mô hình cây quyết định hồi quy bằng thư viện sklearn:

```
# Xây dựng mô hình cây quyết định hồi quy
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_scaled, y)
y_pred_reg = regressor.predict(X_scaled)
```

Các tham số khi xây dựng mô hình huấn luyện cây quyết định hồi quy có thể kể lần lượt như sau:

+ **criterion** – chỉ số để phân hoạch dữ liệu trên cây (mặc định là ‘gini’). Nếu chúng ta muốn sử dụng độ lợi thông tin thì sử dụng ‘entropy’.

+ **max_depth** – Độ sâu tối đa của cây, nếu không thiết lập thì mặc định sẽ mở rộng cây đến các nút lá kết quả (cây có thể sẽ khá lớn nếu tập dữ liệu lớn).

+ **max_features** – Số lượng thuộc tính tối đa sử dụng để xây dựng cây, mặc định lấy tất cả.

+ **min_samples_split** – Số lượng mẫu tối thiểu cần có trong một nút để nút đó được phép tách.

+ **min_samples_leaf** – Số lượng mẫu tối thiểu trong mỗi nút lá sau khi tách.

Các tham số khác, chúng ta có thể tìm hiểu thêm tại đây:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

Sau khi cài đặt xong ta đánh giá mô hình cây quyết định hồi quy vừa xây dựng được trên 2 tham số là Mean squared error và R-squared:

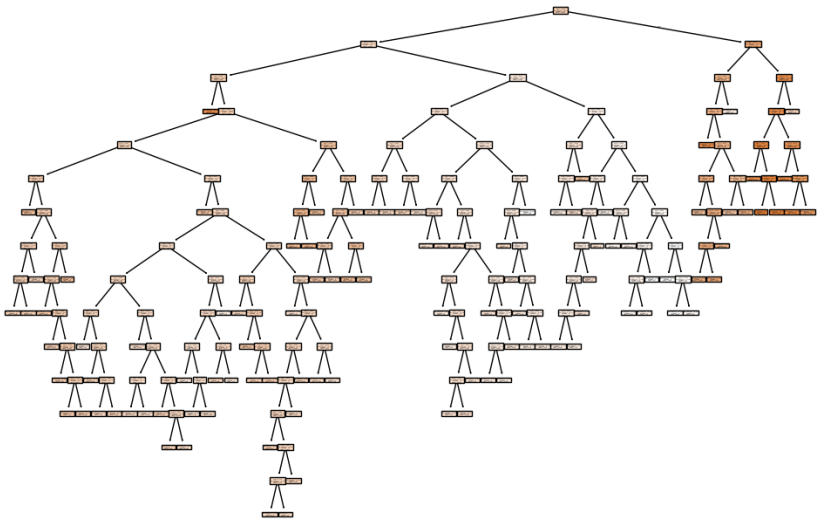
```
# Đánh giá mô hình cây quyết định hồi quy
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y, y_pred_reg)
r2 = r2_score(y, y_pred_reg)
print(f'Mean Squared Error: {mse:.2f}')
print(f'R^2 Score: {r2:.2f}')
```

✓ 0.0s

```
Mean Squared Error: 2.90
R^2 Score: 0.97
```

Đồng thời vẽ cây quyết định của mô hình vừa tạo để trực quan hóa kết quả huấn luyện:

```
# Vẽ cây quyết định hồi quy
plt.figure(figsize=(12,8))
plot_tree(regressor, filled=True)
plt.show()
```



Sinh viên thực hành chạy ra kết quả và tự rút ra nhận xét về các mô hình huấn luyện bên trên. Hãy tạo thư mục tên "TH_Buoi03_<HoTen>_<MSSV>", sau đó để tất cả các tập tin thực hành ở các mục 2.2, 2.3, 2.4 vào thư mục này, sau đó nén cả thư mục lại để nộp bài.

3.4. Bài tập làm thêm

1/. Tải Tập dữ liệu và thực hiện các yêu cầu sau:

Drugs A, B, C, X, Y for Decision Trees

a/. Mô tả tập dữ liệu (số thuộc tính, số phần tử, nhãn là gì).

b/. Chia tập dữ liệu với hai trường hợp 60% train, 40% test và 80% train, 20% test.

c/. Với mỗi trường hợp phân chia sử dụng cây quyết định dựa vào chỉ số độ lợi thông tin với số lượng mẫu tối thiểu tại mỗi nút lá là 5, 10, 15, 20 với criterion='gini'.

d./ Đánh giá mô hình với mỗi trường hợp đã sử dụng và tìm ra cách phân chia dữ liệu và bộ tham số sẽ ra được độ chính xác cao nhất.

2/. Xây dựng cây quyết định dựa vào chỉ số độ lợi thông tin và dự đoán nhãn

a/. Đọc dữ liệu từ tập dữ liệu đánh giá chất lượng rượu vang trắng

https://archive.ics.uci.edu/ml/datasets/wine+quality

b/. Dữ liệu có bao nhiêu thuộc tính? Cột nào là cột nhãn? Giá trị của các nhãn?

c/. Với tập dữ liệu White Wine sử dụng nghi thức K-Fold để phân chia tập dữ liệu huấn luyện với *K=50*, sử dụng tham số "Shuffle" để xáo trộn tập dữ liệu trước khi phân chia. Xác định số lượng phần tử có trong tập test và tập huấn luyện nếu sử dụng nghi thức đánh giá này.

d/. Xây dựng mô hình cây quyết định dựa trên tập dữ liệu học tạo ra ở bước c.

e/. Đánh giá độ chính xác cho từng phân lớp dựa vào giá trị dự đoán của câu c cho mỗi lần lặp. Ghi nhận lại kết quả độ chính xác cho từng phân lớp của lần lặp cuối.

f/. Tính độ chính xác tổng thể cho mỗi lần lặp và độ chính xác tổng thể của trung bình 50 lần lặp. Sử dụng giải thuật KNN, Bayes thơ ngây ở buổi thực hành số 2 để so sánh hiệu quả phân lớp của giải thuật cây quyết định với nghi thức đánh giá k-fold với K=50

3/. Xây dựng cây quyết định dựa vào chỉ số Gini và dự đoán nhãn

a/. Đọc dữ liệu từ tập **Boston House Prices:**

https://www.kaggle.com/datasets/vikrishnan/boston-house-prices

b/. Phân tích cấu trúc dữ liệu để xác định số lượng thuộc tính, cột nhãn và các giá trị của nhãn.

c/. Sử dụng nghi thức **Hold-out** với tỉ lệ 70% train và 30% test, thực hiện 10 lần với các tham số random_state [1,3,5,7,9,..,19].

d/. Xây dựng mô hình **Decision Tree Regressor** với criterion='entropy' trên tập huấn luyện ở mỗi lần lặp.

e/. Tính và ghi lại độ chính xác cho từng phân lớp sau mỗi lần lặp, lưu kết quả lần lặp cuối.

f/. Tính độ chính xác trung bình sau 10 lần lặp và so sánh độ chính xác trung bình của 10 lần lập so với khi sử dụng mô hình Naive Bayes hồi quy.