Cây quyết định Decision Tree

Cây quyết định là một cấu trúc giống như lưu đồ, trong đó mỗi nút bên trong đại diện cho một "phép thử" trên một thuộc tính (ví dụ: liệu một lần lật đồng xu xuất hiện ngửa hay sấp), mỗi nhánh đại diện cho kết quả của thử nghiệm và mỗi nút lá đại diện cho một nhãn lớp (quyết định được thực hiện sau khi tính toán tất cả các thuộc tính). Các đường dẫn từ gốc đến lá đại diện cho các quy tắc phân loại.

Trong phân tích quyết định, cây quyết định và sơ đồ ảnh hưởng có liên quan chặt chẽ được sử dụng như một công cụ hỗ trợ ra quyết định trực quan và phân tích, nơi các giá trị mong muốn của các lựa chọn thay thế cạnh tranh được tính toán.

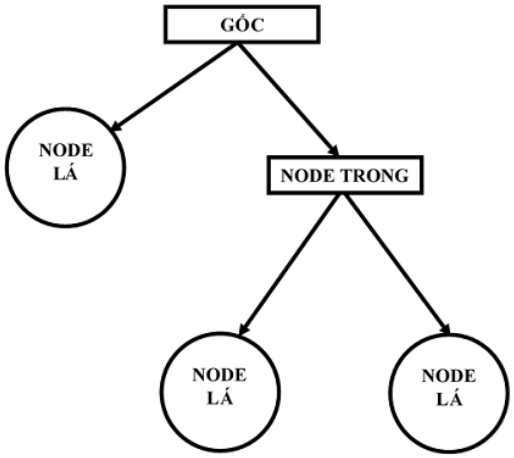
Đặc điểm của cây quyết định: là một cây có cấu trúc:

+ Root (Gốc): Nút trên cùng của cây.

+ Node trong: nút trung gian trên một thuộc tính đơn.

+ Nhánh: Biểu diễn các kết quả của kiểm tra trên nút.

+ Node lá: Biểu diễn lớp hay sự phân phối lớp (hình vuông hoặc chữ nhật).



Mục đích: sinh viên hiểu được cấu trúc cây quyết định và thực hành code các ví dụ sử dụng cây quyết định để giải quyết bài toán phân loại đơn giản.

Ví dụ 1: Phân biệt quả cam và quả táo dựa vào trọng lượng và bề mặt vỏ

Nguồn tham khảo: video Google AI
https://www.youtube.com/watch?v=cKxRvEZd3Mw&list=PLOU2XLYxmsIIuiBfYad6rFYQU_jL2ryal

1. Dữ liệu huấn luyện: bảng chứa đặc trưng của 2 loại quả về trọng lượng và bề mặt vỏ (nhẵn, sần)
   Nhận xét: trọng lượng lớn và bề mặt sần là quả cảm, trọng lượng nhỏ hơn và bề mặt nhẵn là quả táo

Weight (g)	Texture	Label
150	Bumpy	Orange
170	Bumpy	Orange
140	Smooth	Apple
130	Smooth	Apple
...	...	...

2. Import thư viện

```
from sklearn import tree
print("imported sklearn successfully")
```

```
imported sklearn successfully
```

3. Chuyển các đặc trưng và nhãn của dữ liệu huấn luyện từ bảng trên thành code:

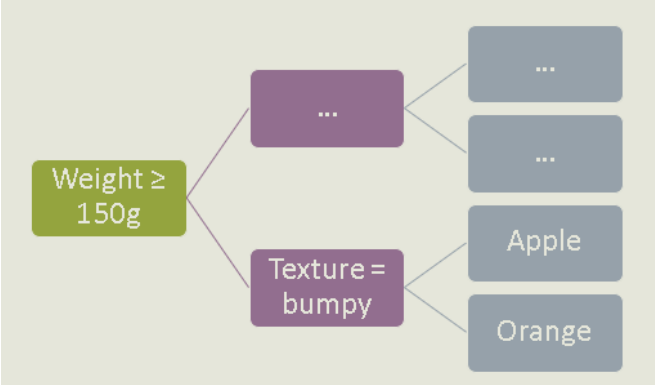
```
features = [[150, "bumpy"],
            [170, "bumpy"],
            [140, "smooth"],
            [130, "smooth"]]
labels = ["apple",
          "apple",
          "orange",
          "orange"]
```

Tuy nhiên, nếu đặc trưng và nhãn là text (như bumpy, smooth, apple, orange) thì không thể tính toán được nên phải chuyển đặc trưng bề mặt vỏ thành 0 (sần); 1 (nhẵn) và nhãn trong code ở trên thành 0 (quả táo) và 1 (quả cam) như sau:

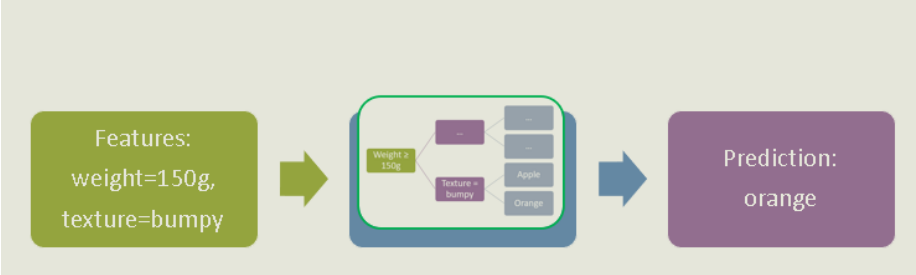
```
fruits = ["orange", "apple"]
```

```
features = [[150, 0],
            [170, 0],
            [140, 1],
            [130, 1]]
labels = [0,
          0,
          1,
          1]
```

4. Từ các đặc trưng, ta xây dựng mô hình cây quyết định để phân biệt quả cam và quả táo cho bài toán này sẽ là:



Ví dụ đầu ra dự đoán từ các đặc trưng:



5. Mô hình phân loại sử dụng cây quyết định và huấn luyện mô hình với dữ liệu huấn luyện (đặc trưng và nhãn) ở trên:

```
classifier = tree.DecisionTreeClassifier()
classifier = classifier.fit(features, labels)
```

6. Sử dụng mô hình phân loại được huấn luyện để đưa ra dự đoán cho dữ liệu đầu vào mới:

```
prediction = classifier.predict([[150, 0]])
print("Prediction of [150, 0]: ")
print("Prediction label: ", fruits[prediction[0]])

prediction = classifier.predict([[160, 0]])
print("Prediction of [160, 0]: ")
print("Prediction label: ", fruits[prediction[0]])

prediction = classifier.predict([[140, 0]])
print("Prediction of [140, 0]: ")
print("Prediction label: ", fruits[prediction[0]])

prediction = classifier.predict([[150, 1]])
print("Prediction of [150, 1]: ")
print("Prediction label: ", fruits[prediction[0]])

prediction = classifier.predict([[140, 1]])
print("Prediction of [140, 1]: ")
print("Prediction label: ", fruits[prediction[0]])

prediction = classifier.predict([[130, 1]])
print("Prediction of [130, 1]: ")
print("Prediction label: ", fruits[prediction[0]])
```

Kết quả:

```
Prediction of [150, 0]:
Prediction label:  orange
Prediction of [160, 0]:
Prediction label:  orange
Prediction of [140, 0]:
Prediction label:  orange
Prediction of [150, 1]:
Prediction label:  apple
Prediction of [140, 1]:
Prediction label:  apple
Prediction of [130, 1]:
Prediction label:  apple
```

Kiểm tra mô hình: sinh viên tự code thêm để kiểm tra mô hình với [135, 1], [155, 0], quan sát kết quả thu được

Ví dụ 2: Phân biệt 3 loại hoa lan sử dụng cây quyết định

Link video tham khảo của GG AI:
https://www.youtube.com/watch?v=tNa99PG8hR8&list=PLOU2XLYxmsIIuiBfYad6rFYQU_jL2ryal&index=3

Tập Dữ liệu:

Dữ liệu có tổng cộng 150 mẫu, mỗi loại hoa 50 mẫu

Mỗi loại hoa có 4 đặc trưng: chiều dài, chiều rộng của cánh hoa (petal) và đài hoa (sepal)

Ví dụ mỗi loại hoa 10 dòng dữ liệu như 3 bảng dưới đây:

Irish Setosa Examples

Dataset Order	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	I. setosa
2	4.9	3	1.4	0.2	I. setosa
3	4.7	3.2	1.3	0.2	I. setosa
4	4.6	3.1	1.5	0.2	I. setosa
5	5	3.6	1.4	0.3	I. setosa
6	5.4	3.9	1.7	0.4	I. setosa
7	4.6	3.4	1.4	0.3	I. setosa
8	5	3.4	1.5	0.2	I. setosa
9	4.4	2.9	1.4	0.2	I. setosa
10	4.9	3.1	1.5	0.1	I. setosa

Irish Versicolor Examples

Dataset Order	Sepal length	Sepal width	Petal length	Petal width	Species
51	7	3.2	4.7	1.4	I. versicolor
52	6.4	3.2	4.5	1.5	I. versicolor
53	6.9	3.1	4.9	1.5	I. versicolor

54	5.5	2.3	4	1.3	I. versicolor
55	6.5	2.8	4.6	1.5	I. versicolor
56	5.7	2.8	4.5	1.3	I. versicolor
57	6.3	3.3	4.7	1.6	I. versicolor
58	4.9	2.4	3.3	1	I. versicolor
59	6.6	2.9	4.6	1.3	I. versicolor
60	5.2	2.7	3.9	1.4	I. versicolor

Irish Virginica Examples

Dataset Order	**Sepal length**	**Sepal width**	**Petal length**	**Petal width**	**Species**
101	6.3	3.3	6	2.5	I. virginica
102	5.8	2.7	5.1	1.9	I. virginica
103	7.1	3	5.9	2.1	I. virginica
104	6.3	2.9	5.6	1.8	I. virginica
105	6.5	3	5.8	2.2	I. virginica
106	7.6	3	6.6	2.1	I. virginica
107	4.9	2.5	4.5	1.7	I. virginica
108	7.3	2.9	6.3	1.8	I. virginica
109	6.7	2.5	5.8	1.8	I. virginica
110	7.2	3.6	6.1	2.5	I. virginica

Nhận xét về dữ liệu:

- Không có giá trị nào bị thiếu.
- Đài hoa dài hơn cánh hoa. Độ dài đài hoa nằm trong khoảng từ 4,3 đến 7,9 với chiều dài trung bình là 5,8, trong khi chiều dài cánh hoa dao động trong khoảng 1 đến 6,9 với chiều dài trung bình là 3,7.
- Đài hoa cũng rộng hơn cánh hoa. Chiều rộng đài hoa nằm trong khoảng từ 2 đến 4,4 với chiều rộng trung bình là 3,05, trong khi chiều rộng của cánh hoa nằm trong khoảng 0,1 đến 2,5 với chiều rộng trung bình là 1,19.
- Chiều dài cánh hoa trung bình của setosa nhỏ hơn nhiều so với versicolor và virginica; tuy nhiên chiều rộng đài hoa trung bình của setosa cao hơn versicolor và virginica.
- Chiều dài và chiều rộng của cánh hoa có mối tương quan chặt chẽ, nghĩa là 96% chiều rộng tăng khi tăng chiều dài.
- Chiều dài cánh hoa có mối tương quan nghịch với chiều rộng đài hoa, nghĩa là 42% chiều rộng đài hoa tăng lên sẽ làm giảm chiều dài cánh hoa.

Kết luận ban đầu về dữ liệu: Dựa trên chiều dài và chiều rộng của đài hoa / cánh hoa, có thể kết luận rằng versicolor / virginica có thể giống về kích thước; tuy nhiên các đặc điểm của setosa dường như là khác biệt đáng kể so với hai cái còn lại.

Bắt đầu xây dựng mô hình:

1. Import thư viện

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn import tree
```

2. Sử dụng scikit-learn để load dataset Iris như sau:

```
irisData = load_iris()
print("### Iris data successfully loaded ###")
```

```
### Iris data successfully loaded ###
```

3. Chọn dữ liệu để kiểm tra (là những dữ liệu khác dữ liệu huấn luyện) từ 150 dòng dữ liệu được đánh idx từ 0 đến 149:

```
test_idx = [0, 9, 19, 29, 39, 49, 59, 69, 79,
            89, 99, 109, 119, 129, 139, 149]
```

4. Tập huấn luyện (là toàn bộ dữ liệu trừ đi tập kiểm tra):

```
# get training data
train_target = np.delete(irisData.target, test_idx)
train_data = np.delete(irisData.data, test_idx, axis=0)
print("### Training dataset successfully generated ###")
```

```
### Training dataset successfully generated ###
```

5. Tập kiểm tra

```
# get testing data
test_target = irisData.target[test_idx]
test_data = irisData.data[test_idx]
print("### Testing dataset successfully generated ###")
```

```
### Testing dataset successfully generated ###
```

6. Xây dựng mô hình phân loại dựa vào thuật toán cây quyết định

```
irisClassifier = tree.DecisionTreeClassifier()
irisClassifier = irisClassifier.fit(train_data, train_target)
print("### Decision tree successfully created and trained ###")
```

```
### Decision tree successfully created and trained ###
```

7. Kết quả dự đoán

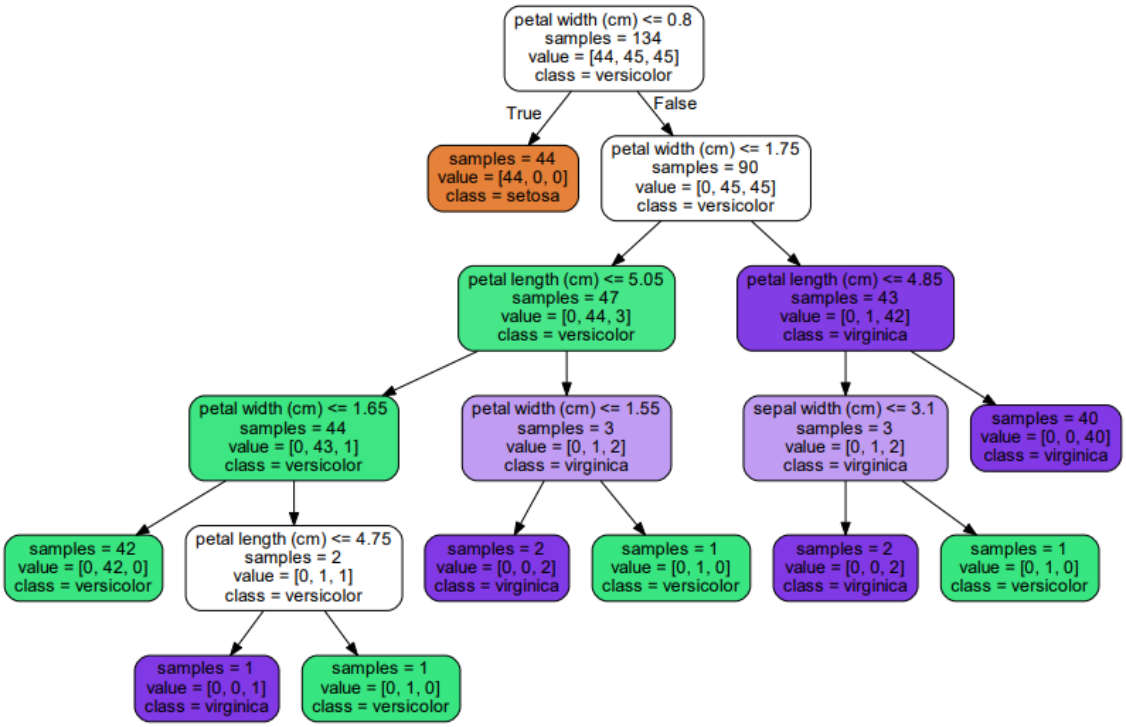
```
prediction = irisClassifier.predict(test_data)

print("### 1st testing data ###")
print("Features: ", test_data[0])
print("Lables:", test_target[0])
print("Iris type: ", irisData.target_names[test_target[0]])

print("### 1st testing data's prediction result ###")
print("Prediction lables:", prediction[0])
print("Prediction Iris type: ", irisData.target_names[prediction[0]])

print("### Testing target labels ###")
print(test_target)

print("### Prediction labels ###")
print(prediction)
```

```
### 1st testing data ###
Features:  [5.1 3.5 1.4 0.2]
Lables: 0
Iris type:  setosa
### 1st testing data's prediction result ###
Prediction lables: 0
Prediction Iris type:  setosa
### Testing target labels ###
[0 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2]
### Prediction labels ###
[0 0 0 0 0 0 1 1 1 1 1 2 1 2 2 2]
```

Sinh viên quan sát: Testing target labels (nhãn đích) và Prediction labels (nhãn dự đoán) trong hình trên và cho nhận xét về mô hình. Độ chính xác của mô hình có phải 100% ko?

8. Sơ đồ cây quyết định phân loại hoa lan dùng pydot (tuy nhiên pydot ko còn dùng được ở phiên bản mới)

```
#show decision tree
from sklearn.externals.six import StringIO
import pydot
dot_data = StringIO()
tree.export_graphviz(irisClassifier,
                     out_file = dot_data,
                     feature_names = irisData.feature_names,
                     class_names = irisData.target_names,
                     filled = True,
                     rounded = True,
                     impurity = False)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph[0].write_pdf("irisDecisionTree.pdf")
```



Giải thích cho sơ đồ

Ở các node của cây quyết định chúng ta có một nhóm các đặc trưng, từ giá trị của các đặc trưng này chúng ta xác định điều kiện đúng - 1 (Yes) hoặc sai - 0 (No). Ví dụ khi ta xác định được node đầu tiên có cánh hoa (petal) rộng từ 0.8cm trở xuống, nếu trả về đúng (True) thì chuyển sang node dưới bên trái và kết luận loài hoa này là hoa 'setosa', ngược lại điều kiện trả về sai (False) thì chuyển sang node dưới bên phải.

Khi chuyển sang node dưới bên phải này, cây quyết định sẽ xác định xem cánh hoa có độ rộng nhỏ hơn 1.75 hay không, nếu trả về đúng (True) thì chuyển sang node dưới tiếp theo của nó bên trái, ngược lại sẽ chuyển sang node dưới tiếp theo của nó bên phải.

Tại các node tiếp theo, cây quyết định sẽ dựa vào một số đặc trưng và giá trị của chúng để đưa ra quyết định rẽ nhánh, và cuối cùng kết quả đạt được là đưa ra quyết định với giá trị đưa vào nó sẽ thuộc một loài hoa nào đó.

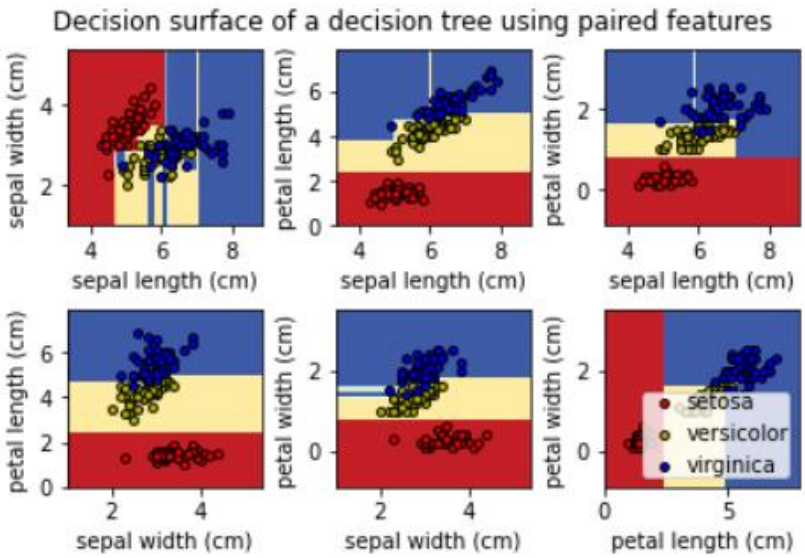
Code tham khảo để vẽ cây quyết định

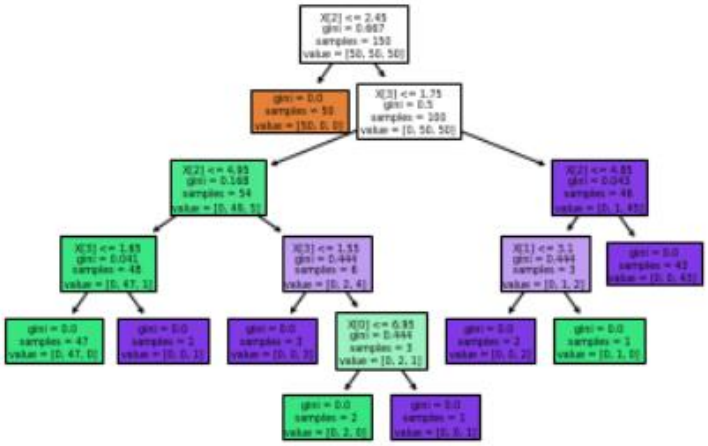
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02

# Load data
iris = load_iris()
```

```
for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
    # We only take the two corresponding features
    X = iris.data[:, pair]
    y = iris.target

    # Train
    clf = DecisionTreeClassifier().fit(X, y)

    # Plot the decision boundary
    plt.subplot(2, 3, pairidx + 1)

    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                         np.arange(y_min, y_max, plot_step))
    plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

    plt.xlabel(iris.feature_names[pair[0]])
    plt.ylabel(iris.feature_names[pair[1]])

    # Plot the training points
    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
                    cmap=plt.cm.RdYlBu, edgecolor='black', s=15)
```

```
plt.suptitle("Decision surface of a decision tree using paired features")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis("tight")

plt.figure()
clf = DecisionTreeClassifier().fit(iris.data, iris.target)
plot_tree(clf, filled=True)
plt.show()
```

Kết quả





==========

Bài tập 1: Dựa vào hướng dẫn từ ví dụ 1 hãy code với dữ liệu như sau:

Horsepower	Seats	Label
300	2	sport car
450	2	sport car
200	8	minivan
150	9	minivan
…	…	…

Gợi ý: sinh viên có thể sử dụng đặc trưng về số ghế hoặc mã lực để phân loại 2 loại xe này.

Bài tập 2: Dựa vào hướng dẫn từ ví dụ 1 hãy code với dữ liệu như sau:

Màu tóc	Chiều cao	Cân nặng	Bị cận	Đi khám bệnh
Đen	Thấp	Nhẹ	Không	Có
Trắng	Trung bình	Trung bình	Có	Không
Trắng	Cao	Nặng	Không	Không
Đen	Trung bình	Nhẹ	Có	Không
Vàng	Thấp	Trung bình	Không	Không
Đen	Trung bình	Trung bình	Không	Có
Vàng	Trung bình	Nặng	Không	Có
Đen	Cao	Trung bình	Có	Không
Trắng	Cao	Nặng	Có	Có
Trắng	Thấp	Nặng	Không	Không

*Cho biết thuộc tính phân lớp là **"Đi khám bệnh"**.*

a. Sử dụng thuật toán ID3 để xây dựng cây quyết định.

b. Sử dụng cây quyết định đã xây dựng để xác định lớp cho các bộ dữ liệu mới sau:

X = (Đen, Thấp, Nhẹ, Có) và Y = (Vàng, Cao, Nặng, Không)