Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin
Khoa khoa học máy tính

Ngành: Khoa học Máy tính

Môn học: Toán cho Khoa học Máy tính
(CS115.N11.KHTN)

Đề tài: Decision Tree

Giảng viên: Lương Ngọc Hoàng

Sinh viên thực hiện: Trương Thanh Minh - 21520064

Mục lục

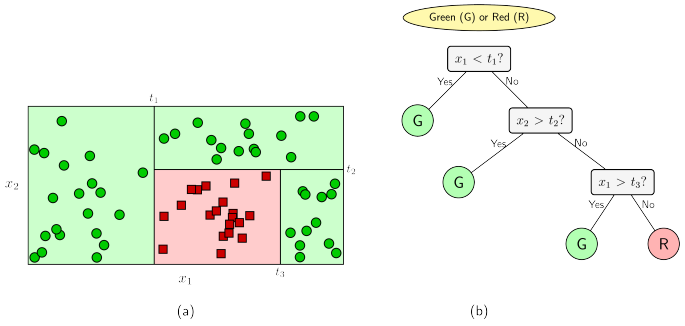
1 Giới thiệu về Decision Tree

Đã bao giờ bạn tự hỏi rằng, con người thường đưa ra quyết định của một vấn đề bằng cách nào chưa (ở đây, ta bỏ qua những quyết định mang tính ngẫu nhiên, hên xui, may rủi). Theo một cách logic, những quyết định mà chúng ta đưa ra thường được bắt đầu bằng những câu hỏi. Và trong Machine Learning cũng có một model như thế. Đó chính là Decision Tree.

Decision Tree (DT) hay còn gọi là **cây quyết định** là phương pháp học tập có giám sát (**supervised learning**) được sử dụng trong các bài toán phân loại hoặc hồi quy. Mục tiêu của DT là tạo ra một mô hình dự đoán giá trị bằng cách tạo ra các quy tắc quyết định được suy ra từ các đặc trưng dữ liệu (**data features**).

Về bản chất, nhiệm vụ của Decision Tree là đi tìm ranh giới để phân loại các lớp với nhau (trong bài toán Phân loại) hoặc là một đường để dự đoán giá trị của một biến liên tục (trong bài toán Hồi quy).

Có một điều mà chúng ta cần lưu ý đó là Decision Tree rất hữu ích cho phân tích dữ liệu và học máy (**Machine Learning**) vì chúng chia nhỏ dữ liệu phức tạp thành các thành phần dễ quản lý hơn. Ngoài ra, decision tree, có thể làm việc với nhiều loại dữ liệu khác nhau: dạng rời rạc, không có thứ tự (ví dụ: thích, ghét, nóng, lạnh,...) hay là dạng dữ liệu liên tục. Và đặc biệt, Decision Tree ít yêu cầu việc chuẩn hóa dữ liệu so với các model khác. Vì vậy, có thể sử dụng Decision Tree một cách dễ hiểu và đơn giản.



Ảnh 1: Decision Tree trong bài toán phân loại (nguồn: Tại đây
)

Hình (a) cho ta thấy là cách mà Decision Tree phân chia dữ liệu nhằm mục đích đưa dự đoán.

Hình (b) là một ví dụ về model Decision Tree.

Cấu trúc DT gồm 3 phần chính: node gốc (Root nodes), node trong (internal nodes) và node lá (leaf nodes).

Nút đầu tiên hay còn gọi là Root node. Cụ thể ở trong hình (b) thì nút gốc là nút hình chữ nhật màu xám trên cùng. Nút trong là nút ra quyết định vì đây là điểm mà tại đó nút phân chia sâu hơn dựa trên tính năng tốt nhất của nhóm con. Nút lá là nút cho ta biết label của điểm dữ liệu đó, nút lá là nút không thể chia thêm được nữa.

Trong bài báo cáo này, tôi xin phép trình này 2 nội dung chính sau:

- Công dụng của DT.
- Cách sử dụng của DT trong Sklearn.
- Các tiêu chí split.
- Một số thuật toán trong sklearn.
- Demo code.

2 Công dụng của Decision Tree

DT là công cụ thường được dùng để phân tích và khai thác dữ liệu sử dụng. Ngoài ra, chúng cũng là một công cụ phổ biến trong máy học và trí tuệ nhân tạo, nơi chúng được sử dụng làm thuật toán đào tạo (train) cho việc học có giám sát.

Decision Tree là công cụ mạnh mẽ và khá phổ biến trong lĩnh vực Machine Learning không chỉ bởi sự đơn giản, dễ hiểu, linh hoạt của chúng mà còn bởi công dụng của nó.

Như đã nói ở trên, Decision Tree được dùng trong các bài toán phân loại và hồi quy. Vậy thì trong các bài toán phân loại, hồi quy, Decision Tree có công dụng gì?

Công dụng của DT chính là dựa vào những thuộc tính của bộ dữ liệu để xây dựng model có khả năng dự đoán class của các điểm dữ liệu (bài toán phân loại) hay là dự đoán một giá trị số thực nào đó cho các điểm dữ liệu (bài toán hồi quy) mà các điểm dữ liệu đó chưa được nhìn thấy trong tập train (*unseen*).

3 Cách sử dụng Decision Tree

Ở đây, tôi xin phép chỉ trình bày về Decision Tree được dùng trong bài toán Classification, còn về cách sử dụng trong bài toán Regression, bạn có thể xem thêm tại đây.

Để hỗ trợ cho việc sử dụng Decision Tree cho mục đích phân loại một cách dễ dàng, trong sklearn người ta đã tích hợp sẵn lớp **DecisionTreeClassifier** để thực hiện phân loại nhiều lớp trên một tập dữ liệu.

Trong bài này, tôi xin được phép trình bày về cách sử dụng Decision Tree trong sklearn để phân loại các loài hoa trong bộ dữ liệu Iris.

Bộ dữ liệu này chứa 150 bản ghi, tương ứng với 50 mẫu từ mỗi loài hoa Iris (Iris Setosa, Iris Virginica, Iris Versicolor) cùng với đó là 3 thuộc tính: Chiều dài, chiều rộng của đài hoa và cánh hoa (được tính bằng cm).

Để phân loại các loài hoa trong bộ dữ liệu Iris, ở đây, tôi xin phép trình bày về cách sử dụng Decision Tree trong Sklearn để giải quyết bài toán này.

3.1 Import các thư viện cần thiết

Trong bất kì model nào, việc đầu tiên và cũng khá là quan trọng, đó chính là ta phải import những thư viện cần thiết vào.

```
import numpy as np
# For split dataset
import sklearn.model_selection import train_test_split
# For Evaluate model
from sklearn import metrics
# For load dataset
from sklearn.datasets import load_iris
# For draw tree
import graphviz
# For use some model in tree
from sklearn import tree
```

Source code 1: Import thư viện

3.2 Load dữ liệu

Vì bộ dữ liệu Iris đã có sẵn ở trong thư viện sklearn rồi, do đó, khi muốn sử dụng, ta chỉ cần sử dụng câu lệnh load như ở dưới.

```
data = load_iris()
print(data)
```

Source code 2: Load dữ liệu

Để tìm hiểu rõ hơn về bộ dữ liệu, mình sẽ show ra một số thông tin của bộ dữ liệu này:

```
'target_names': array(['setosa', 'versicolor', 'virginica'],
      dtype='<U10')
```

target_names cho ta biết các lớp mà ta sẽ phân loại.

```
'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
```

feature_names cho ta biết các đặc trưng mà ta sử dụng để phân loại trong bộ dữ liệu này.

Sau khi load dữ liệu xong, ta tiến hành lưu các giá trị các đặc trưng của nó vào X và label của nó vào y.

```
X, y = data.data, data.target()
```

3.3 Split dữ liệu

Sau khi đã load dữ liệu vào X, y, ta thực hiện bước split dữ liệu vào 2 tập *train*, *test* để huấn luyện và đánh giá model.

Vì sklearn đã tích hợp sẵn hàm để split dữ liệu, vậy nên ở đây để nhanh chóng, ta sử dụng luôn hàm ở trong sklearn.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size=0.2, random_state=112)
```

Hàm này nó có thể nhận khá nhiều tham số, tuy nhiên ở đây tôi chỉ trình bày một số tham số cần thiết.

- X là giá trị của các đặc trưng trong bộ dữ liệu.
- y là nhãn tương ứng của từng điểm dữ liệu.
- *test_size* là một giá trị nằm trong khoảng từ 0.0 đến 1.0, đại diện cho tỷ lệ của tập test trong bộ dữ liệu.
- *random_state* có vai trò kiểm soát việc xáo trộn được áp dụng cho dữ liệu trước khi split.

3.4 Sử dụng model phân loại của Decision Tree trong Sklearn

Để sử dụng model, mình gọi đến model *DecisionTreeClassifier* đã được tích hợp sẵn ở trong sklearn.

```
clf = tree.DecisionTreeClassifier()
```

	Chức năng	Giá trị có thể nhận	Giá trị mặc định
criterion	Đo lường chất lượng của một lần split node.	*gini*, *entropy*, *log_loss*	*gini*
splitter	Chọn chiến lược tách tại mỗi node.	*best*, *random*	*best*
max_depth	Chiều sâu tối đa của cây.	*int*	None
min_samples_split	Số lượng mẫu tối thiểu cần thiết để tách một nút nội bộ.	*int*, *float*	2
min_samples_leaf	Số lượng mẫu tối thiểu để có một nút lá. Một điểm tách ở bất kỳ độ sâu nào sẽ chỉ được xem xét nếu nó để lại ít nhất các mẫu huấn luyện *min_samples_leaf* trong mỗi nhánh bên trái và bên phải. Điều này có thể có tác dụng làm mịn mô hình, đặc biệt là trong hồi quy.	*int*, *float*	1

Bảng 1: Bảng tham số

Trong hàm này, ta có thể truyền vào rất nhiều tham số để có thể xây

dựng 1 cây theo ý của chúng ta. Tuy nhiên, trong bài báo cáo này, tôi chỉ trình này về một số tham số quen thuộc, thường sử dụng. Để hiểu hơn về các tham số được sử dụng trong hàm này, ta có thể xem bảng 1 được trình bày ở trên.

Như mọi người thấy, mỗi tham số đều sẽ có giá trị mặc định. Do đó, nếu ta không muốn chỉnh giá trị của tham số nào cả, thì ta có thể không cần điền bất kì tham số nào. Và khi đó, nó sẽ được hiểu rằng các giá trị tham số đều được đặt làm giá trị mặc định.

3.5 Train

Sau khi hoàn thành xong việc gọi hàm, thiết lập các tham số để phục vụ cho việc tạo cây, giờ là lúc ta sẽ cho model học để sau này có thể dự đoán những dữ liệu mà nó chưa từng nhìn thấy.

```
clf = clf.fit(X_train, y_train)
```

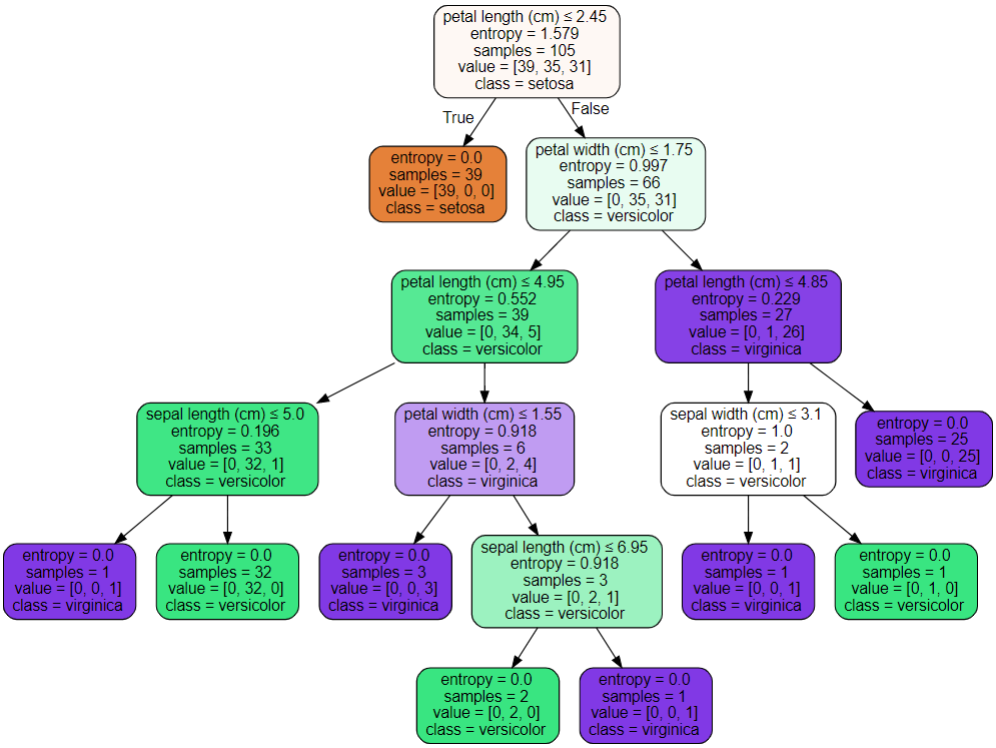
- X_train: dữ liệu mà ta dùng để train.
- y_train: label tương ứng của dữ liệu.

Tới đây, có lẽ nhiều bạn sẽ thắc mắc rằng hình dạng của cây mà ta vừa cho nó train sẽ như thế nào đúng không ?

Để show cây, ta sử dụng câu lệnh sau:

```
dot_data = tree.export_graphviz(clf, out_file=None,
    feature_names=data.feature_names,
class_names=data.target_names,
filled=True, rounded=True,
special_characters=True)
graph = graphviz.Source(dot_data)
```

Sau khi sử dụng dòng lệnh ở trên, ta thu được ảnh sau:



Ảnh 2: Decision Tree

3.6 Predict

Sau khi train xong cây quyết định, ta bắt đầu tiến hành *predict* dữ liệu ở tập *test* mà ban đầu ta split.

```
y_pred = clf.predict(X_test)
```

Trong model phân loại, hàm này sẽ trả về lớp dự đoán cho mỗi mẫu ở trong tập X_test.

Sau khi chạy dòng code ở trên, ta thu được giá trị của y_pred như bên dưới.

```
array([0, 0, 1, 0, 2, 1, 2, 0, 1, 2, 1, 1, 2, 1, 0, 2, 2, 1,
       1, 2, 1, 0, 2, 2, 2, 2, 2, 2, 1, 0])
```

Các giá trị $0, 1, 2$ tương ứng với các loại hoa $setosa, versicolor$ và $virginica$.

3.7 Đánh giá

Giả sử, bạn có một model, làm sao để bạn có thể biết được model của bạn hoạt động có tốt hay không và tốt như thế nào?

Nếu bạn đang thắc mắc điều đó, thì bước này sẽ cho bạn câu trả lời.

```
metrics.accuracy_score(y_test, y_pred)
```

Có khá nhiều phương pháp, nhưng ở đây, tôi xin phép được trình bày về phương pháp đánh giá đơn giản nhất. Sơ qua về phương pháp này thì đây chính là thang đo đánh giá xem model của chúng ta đạt được độ chính xác là bao nhiêu phần trăm.

Hàm này sẽ nhận vào 2 tham số: tham số thứ nhất đại diện cho label của tập test (y_test), tham số thứ 2 đại diện cho giá trị mà ta vừa thực hiện dự đoán ở phần Predict trên (y_pred).

4 Thuật toán xây dựng Decision Tree

4.1 Cách chọn thuộc tính tốt nhất ở mỗi node

Về cơ bản, tất cả các thuật toán trong Decision Tree đều yêu cầu các tiêu chí phân tách (split) để tách một nút thành một cây. Thuật toán đó thường dựa trên *thuộc tính tốt nhất (best attribute)* để phân chia. Có nhiều tiêu chí phân tách khác nhau dựa trên tạp chất *(impurity)* của một nút. Có nhiều tiêu chí phân tách khác nhau dựa trên tạp chất của một nút. Mục đích chính của tiêu chí phân tách là để giảm tạp chất của nút đó.

Dưới đây là một số phương pháp.

4.1.1 Entropy và Information Gain

Entropy là thước đo tạp chất của một nút. Nó được định nghĩa là:

$$Entropy(S) = -\sum_{i=1}^{n} p_i log_2(p_i)$$

- S là bộ dữ liệu mà chúng ta đang dùng.
- p_i là tỉ lệ các điểm dữ liệu thuộc class i trên tổng số điểm dữ liệu trong bộ data S.

Giá trị của Entropy có thể nằm trong khoảng từ 0 đến 1. Nếu tất cả các samples trong tập dữ liệu S thuộc một lớp thì Entropy sẽ bằng 0. Nếu một nửa số samples được phân loại là một lớp và nữa còn lại ở một lớp khác, entropy sẽ ở mức cao nhất là 1. Để chọn đặc trưng tốt nhất (*best feature*) để tách và tìm cây quyết định tối ưu thì khi đó thuộc tính có giá trị entropy **nhỏ nhất** nên được sử dụng.

Information Gain thể hiện sự khác biệt của Entropy trước và sau khi phân tách trên một thuộc tính nhất định. Thuộc tính có Information Gain **cao nhất** sẽ tạo ra sự phân tách tốt nhất vì nó đang thực hiện công việc tốt nhất trong việc phân loại dữ liệu train theo phân loại mục tiêu của nó. Information Gain cho ta biết mức độ không chắc chắn trong tập dữ liệu được giảm đi bao nhiêu sau khi tách tập dữ liệu đó trên một node cụ thể. Công thức của nó là:

$$IG(S, A) = Entropy(S) - \sum(\frac{|S_v|}{|S|} * Entropy(S_v)).$$

Trong đó:

- A đại diện cho một thuộc tính cụ thể hoặc nhãn lớp.
- $\frac{|S_v|}{|S|}$ đại diện cho tỷ lệ của các giá trị trong S_v với số lượng giá trị trong tập dữ liệu S.

Tóm lại, thuật toán tối ưu, khi ta có IG max và Entropy min.

4.1.2 Gini index

Gini index là cost function được sử dụng rộng rãi nhất trong Decision Tree. Chỉ số này tính toán xác suất mà một đặc tính cụ thể sẽ bị phân loại không chính xác khi nó được chọn ngẫu nhiên. Nó là một hàm xác định xem DT của ta được chia tốt như thế nào. Về cơ bản, nó giúp chúng ta xác định bộ tách nào là tốt nhất để ta có thể xây dựng một DT thuần túy. Gini có giá trị từ 0 (best) đến 0.5 (worst). Nó là một trong những phương pháp chọn bộ chia tốt nhất.

Gini index được tính như sau:

$$Gini = 1 - \sum_{i=1}^{n}(p_i)^2$$

Có một điều có thể hữu ích mà ta cần nhớ là: **Nên sử dụng Gini khi dữ liệu của ta là dữ liệu liên tục**.

4.2 ID3 (Iterative Dichotomiser 3)

4.2.1 Giới thiệu, ý tưởng thuật toán

ID3 được coi là một thuật toán xây dựng Decision Tree rất đơn giản (Quinlan, 1983). Nó xác định việc phân loại các đối tượng bằng cách kiểm tra các giá trị của các thuộc tính. ID3 xây dựng cây quyết định theo kiểu từ trên xuống (*top-down*), bắt đầu từ một tập hơn các đối tượng và một đặc điểm đặc biệt. Đây là một thuật toán phân loại theo cách tiếp cận tham lam bằng cách tại mỗi nút chọn một thuộc tính tốt nhất (*best attribute*) mang lại **Information Gain (IG)** lớn nhất hoặc **Entropy** nhỏ nhất. Sau đó, thuộc tính có giá trị IG lớn nhất và Entropy nhỏ nhất sẽ được chọn để tách tập đối tượng.

Quá trình này được thực hiện một cách đệ quy cho đến khi tập hợp trong một cây con nhất định là đồng nhất (tức là nó chứa các đối tượng thuộc cùng một loại). Sau đó, nút đó sẽ trở thành nút lá của cây.

Ngoài ra, vì ID3 sử dụng cách tìm kiếm tham lam. Do đó, nó làm cho việc tạo cây trở nên nhanh chóng.

Để xây dựng thuật toán ID3, ở mỗi bước, người ta chọn feature tốt nhất để xây dựng Decision Tree. Vậy định nghĩa thế nào là *feature tốt nhất* ? ID3 dùng **Information Gain** để tìm ra *feature tốt nhất*. Cụ thể hơn thì theo ID3, thuộc tính nào có IG lớn nhất chính là thuộc tính tốt nhất, nghĩa là nó có khả năng phân loại tốt nhất.

4.2.2 Cách bước xây dựng thuật toán

Để xây dựng Decision Tree bằng thuật toán ID3, ta thực hiện lần lượt các bước sau:

1. Nếu cây quyết định của chúng ta vẫn còn khả năng chia tiếp được hay nói cách khác vẫn tồn lại ít nhất 1 nút không phải là lá, thì ta tính toán **Information Gain** cho mỗi feature.

2. Tạo nút con S trong cây Decision Tree bằng cách sử dụng feature có **Information Gain** lớn nhất.

3. Chia các tập samples trong nút trên thành các tập con mà trong đó, mỗi tập con sẽ có giá trị features được chọn ở trên bằng nhau.

4. Nếu tất cả các samples trong một node thuộc cùng 1 class thì node hiện tại sẽ là node lá và mình sẽ có được label của nó.

5. Quay lại thực hiện (1) cho các features chưa được chọn cho đến khi Decision Tree có tất cả các nút là nút lá.

4.2.3 Ưu, nhược điểm của thuật toán

1. **Ưu điểm**

   - Các quy tắc dự đoán dễ hiểu được tạo ra từ dữ liệu train.
   - Tạo cây nhanh hơn một số thuật toán khác.
   - Chỉ cần kiểm tra đủ các thuộc tính cho đến khi tất cả dữ liệu được phân loại.

2. **Nhược điểm**

   - Có thể bị overfitted nếu dữ liệu quá ít.
   - Tại một thời điểm chỉ có một thuộc tính được được chọn để tạo ra một quyết định.
   - Không xử lý các thuộc tính có giá trị liên tục.

4.3 CART (Classification and Regression Trees)

4.3.1 Giới thiệu, ý tưởng thuật toán

CART (Breiman, 1984) là một thuật toán được cải tiến từ ID3, nhưng nó khác ở chỗ nó hỗ trợ cho các bài toán hồi quy (nghĩa là lá của nó dự đoán một số thực chứ không phải một lớp). Ở đây, tôi sẽ tập trung vào phần phân loại. Khi dùng CART để xây dựng thì cây quyết định của ta có dạng là một **cây nhị phân**, cụ thể là mỗi nút trong có chính xác 2 cạnh đi ra.

Một điều khá đặc biệt khi nhắc tới CART đó là scikit-learn sử dụng phiên bản tối ưu hóa của thuật toán CART.

CART xây dựng cây nhị phân bằng cách sử dụng 1 cặp (thuộc tính, threshold (tương ứng với thuộc tính đó)) có giá trị IG lớn nhất tại mỗi nút.

CART thường sử dụng Gini để xác định thuộc tính lý tưởng để phân tách. Gini đo tần suất một thuộc tính được chọn ngẫu nhiên bị phân loại sai. Khi đánh giá bằng Gini, giá trị càng thấp càng tốt.

4.3.2 Các bước thực hiện thuật toán

1. **Để thực hiện thuật toán CART, ta thực hiện theo các bước sau:**

   (a) Tính tất cả các giá trị Information Gain, sau đó chọn node có giá trị IG cao nhất là nút để split.

   (b) Sau đó, chọn một giá trị *threshold* tương ứng với $feature$ đó. Chia các samples trong nút đó thành 2 tập, trong đó, tập con thứ nhất lưu các samples có giá trị features $\leq t$, ngược lại, tập con còn lại lưu các samples có giá trị của features đó $> t$.

   (c) Tạo 2 node con của S, mỗi nút tương ứng với một tập con ở bước trên.

   (d) Lặp đi lặp lại từ bước (a) cho đến khi các node trong cây đều là lá hoặc thỏa 1 trong các điều kiện trong phần (2).

2. **Việc tạo cây sẽ dừng một trong các điều kiện sau được thỏa khi**:

   - Không còn mẫu nào chưa được phân tách.
   - Tất cả các mẫu cho một nút thuộc cùng một lớp.
   - Không còn thuộc tính nào để phân tách thêm.

4.3.3 Ưu, nhược điểm của thuật toán

1. **Ưu điểm**

   - Dễ dàng xử lý các thuộc tính có giá trị liên tục và các bài toán phân loại.
   - Nó sẽ tự xác định hầu hết các biến có ý nghĩa và loại bỏ các biến không quan trọng.

2. **Nhược điểm**

   - Chỉ split cây theo một biến.

4.4 Demo code

- Demo Decision Tree using Sklearn

5 Tổng về Decision Tree

Qua các phần ở trên, có thể thấy rằng, Decision Tree là một công cụ khá phổ biến bởi tính tiện dụng, dễ sử dụng và đơn giản của nó. Ngoài ra, nó còn có thể sử dụng cho cả bài toán Phân loại và Hồi quy. Không cần phải tiền xử lý, chuẩn hóa dữ liệu...

Tuy nhiên, nó cũng còn một số hạn chế như:

1. Dễ bị overfitting: Khi dữ liệu quá ít hoặc khi train quá khớp với dữ liệu, không tổng quát hóa trong dữ liệu mới. Do đó, dự đoán trên tập test độ chính xác sẽ không cao.

2. Một sự thay đổi nhỏ trong dữ liệu có thể gây ra sự thay đổi lớn trong cấu trúc cây, gây ra sự mất ổn định.

3. Đôi khi, độ phức tạp có thể lớn hơn so với một số thuật toán khác...

6 Tài liệu tham khảo

- Decision Tree Algorithm Explained with Examples
- ID3 Algorithm in Decision Trees
- Using ID3 Algorithm to build a Decision Tree to predict the weather
- Iterative Dichotomiser 3 (ID3) Algorithm From Scratch
- Decision Tree Algorithm in Python From Scratch
- Understanding the maths behind the Gini impurity method for decision tree split
- Decision Tree Classification Algorithm
- ML | Gini Impurity and Entropy in Decision Tree
- How to program a decision tree in Python from 0
- ID3 Decision Tree Classifier from scratch in Python
- Decision Tree in Sklearn

- COMPARATIVE STUDY ID3, CART AND C4.5 DECISION TREE ALGORITHM: A SURVEY