[**Cây quyết định (CART)**](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/)

* **11/05/2020**
* [**TOP 10 THUẬT TOÁN MACHINE LEARNING**](https://dataisg.org/category/tutorial/machine-learning/)

Menu

* [1. Giới thiệu](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#1_Gioi_thieu)
  + [1.1 Cây quyết định là gì?](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#1_1_Cay_quyet_dinh_la_gi)
  + [1.2 Tại sao sử dụng cây quyết định?](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#1_2_Tai_sao_su_dung_cay_quyet_dinh)
  + [1.3 Các loại cây quyết định](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#1_3_Cac_loai_cay_quyet_dinh)
  + [1.4 Các thuật toán cây quyết định](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#1_4_Cac_thuat_toan_cay_quyet_dinh)
* [2. Thuật toán CART](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#2_Thuat_toan_CART)
  + [2.1 CART là gì?](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#2_1_CART_la_gi)
  + [2.2 Cách hoạt động của CART](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#2_2_Cach_hoat_dong_cua_CART)
  + [2.3 Overfitting và Underfitting trong CART](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#2_3_Overfitting_va_Underfitting_trong_CART)
  + [2.4 Tiêu chuẩn dừng](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#2_4_Tieu_chuan_dung)
* [3. Demo đơn giản](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#3_Demo_don_gian)
* [4. Kết luận](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#4_Ket_luan)
  + [4.1 Ưu điểm của CART](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#4_1_Uu_diem_cua_CART)
  + [4.2 Nhược điểm của CART](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#4_2_Nhuoc_diem_cua_CART)
  + [4.3 Chuẩn bị dữ liệu cho CART](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#4_3_Chuan_bi_du_lieu_cho_CART)
* [5. Tài liệu tham khảo](https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/#5_Tai_lieu_tham_khao)

**1. Giới thiệu**

Phân loại (classification) là một quá trình gồm hai bước, bước học tập và bước dự đoán, trong học máy. Trong bước học tập, mô hình được xây dựng dựa trên dữ liệu đào tạo (training data). Trong bước dự đoán, mô hình được sử dụng để dự đoán đáp ứng với dữ liệu đã cho. **Cây quyết định (decision tree)** là một trong những thuật toán phân loại phổ biến, dễ hiểu và dễ giải thích.

Ảnh có chứa tối, xanh lục, cây, đóng

Mô tả được tạo tự động

**1.1 Cây quyết định là gì?**

* Cây quyết định là một trong những thuật toán học máy phổ biến và mạnh mẽ mà tôi đã học. Đây là một phương pháp học có giám sát, có thể được sử dụng cho cả nhiệm vụ phân loại (classification) và hồi quy (regression).
* Mục tiêu là tạo ra một mô hình dự đoán giá trị của biến mục tiêu (target variable) bằng cách học các quy tắc quyết định đơn giản được suy ra từ các đặc trưng dữ liệu (data features).
* Không giống như các thuật toán hộp đen (black box) như Mạng thần kinh (Neural Network), Cây quyết định tương đối dễ hiểu hơn vì nó chia sẻ logic ra quyết định một cách rõ ràng.

**1.2 Tại sao sử dụng cây quyết định?**

Mặc dù thực tế là nhiều nhà khoa học dữ liệu (data scientists) tin rằng đó là một phương pháp cũ và họ có thể nghi ngờ về tính chính xác của nó do vấn đề quá mức (overfitting). Tuy nhiên, các mô hình mạnh mẽ như Random forest (bagging method), gradient boosting (boosting method) và XGBoost (boosting method) đang được sử dụng phổ biến trong tất cả các loại vấn đề khoa học dữ liệu, chúng đều được xây dựng dựa trên thuật toán cây quyết định. Do đó, các khái niệm và thuật toán đằng sau Cây quyết định rất đáng để hiểu!

**1.3 Các loại cây quyết định**

* Cây quyết định biến phân loại (Categorical Variable Decision Tree): Cây quyết định có biến mục tiêu phân loại thì nó được gọi là cây quyết định biến phân loại.
  + Ví dụ: dữ liệu ở đây của ta có dạng YES/NO
* Cây quyết định biến liên tục (Continuous Variable Decision Tree): Cây quyết định có biến mục tiêu liên tục thì nó được gọi là cây quyết định biến liên tục
  + Ví dụ: dữ liệu ở đây của ta có dạng các giá trị liên tục như giá nhà

**1.4 Các thuật toán cây quyết định**

Có 4 loại thuật toán cây quyết định phổ biến:

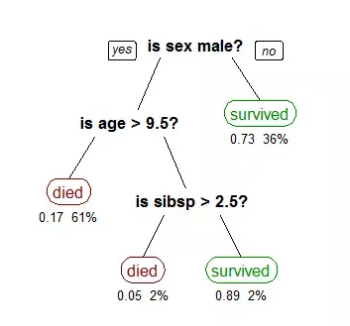
* ID3
* CART
* Chi-Square
* Reduction in Variance

⟹Trong bài này, chúng ta sẽ chỉ tập trung vào cây phân loại và các giải thích về **CART**

**2. Thuật toán CART**

**2.1 CART là gì?**

* CART hay *Classification and regression tree* (cây phân loại và hồi quy) là một thuật toán cây quyết định, được giới thiệu bởi Leo Breiman. Nó có thể giải quyết cả hai vấn đề phân loại và hồi quy.
* Để trực quan hơn, ở đây ta lấy ví dụ về cây phân loại để dự đoán sô người có thể sống sót sau vụ chìm tàu Titanic



* Nhìn một cách đơn giản, nó chính là một tập các IF-ELSE, là kết quả của quá trình đào tạo dữ liệu bằng cây quyết định.

**2.2 Cách hoạt động của CART**

CART thường sử dụng phương pháp *Gini* để tạo các điểm phân chia.

**Gini impurity**

* Là phương pháp hướng đến đo lường tần suất một đối tượng dữ liệu ngẫu nhiên trong tập dữ liệu ban đầu được phân loại không chính xác, trên cơ sở đối tượng dữ liệu đã nằm trong một tập con được phân ra từ tập dữ liệu ban đầu, có dán nhãn thể hiện thuộc tính chung bất kỳ của các đối tượng còn lại trong tập con này, giá trị phân loại chính là nhãn của tập con.
* Gini impurity chính là chỉ số đo lường mức độ đồng nhất hay nhiễu loạn của thông tin, hay sự khác biệt về các giá trị mà mỗi điểm dữ liệu trong một tập con, hoặc một nhánh của cây quyết định. Công thức Gini có thể dùng cho cả dữ liệu rời rạc và liên tục. Nếu điểm dữ liệu thuộc về một node và có chung thuộc tính bất kỳ thì node này thể hiện sự đồng nhất lúc này gini=0gini=0 và ngược lại gini sẽ lớn.

Công thức tổng quát của *Gini*:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Công thức trên để tính độ vẩn đục của một node, khi có nhiều cách phân nhánh mỗi cách có thể phân ra một số node nhất định. Cho nên, lúc này có thêm công thức thứ 2 để tìm ra các phân chia tối ưu nhất:



Trong đó:

* là số điểm dữ liệu có trong node của nhánh được phân
* *N* là số điểm dữ liệu có trong node được dùng để phân nhánh

Hệ số Gsplit càng nhỏ thì cách phân nhánh đó càng tối ưu.

**Bài toán**

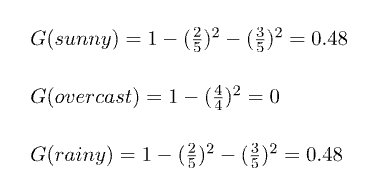
Để hiểu rõ hơn về cách thực hiện của *Gini*, ta sẽ thực hiện một bài toán: Hãy tưởng tượng bạn chơi tennis vào mỗi chủ nhật và bạn mời người bạn thân nhất của mình, Clare. Clare đôi khi đến tham gia nhưng đôi khi không. Đối với cô, nó phụ thuộc vào một số yếu tố, ví dụ như thời tiết, nhiệt độ, độ ẩm và gió. Tôi muốn sử dụng bộ dữ liệu dưới đây để dự đoán liệu Clare có tham gia cùng bạn chơi tennis hay không. Một cách trực quan để làm điều này là thông qua Cây quyết định.

Ảnh có chứa bàn

Mô tả được tạo tự động

Dựa vào bộ dữ liệu trên, ta sẽ sử dụng *Gini* để xây dựng cây quyết định.

*Gini* của thuộc tính *outlook = sunny*:



Từ đó có được *Gini* của thuộc tính *outlook* sẽ bằng:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Lần lượt, sẽ được giá trị *Gini* của các thuộc tính còn lại:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Như vậy, thấy rằng thuộc tính *outlook* sẽ có giá trị *Gini* nhỏ nhất cho nên chọn làm root node. Sau khi chọn được root node, dữ liệu sẽ được rút gọn lại như sau:

Ảnh có chứa bàn

Mô tả được tạo tự động

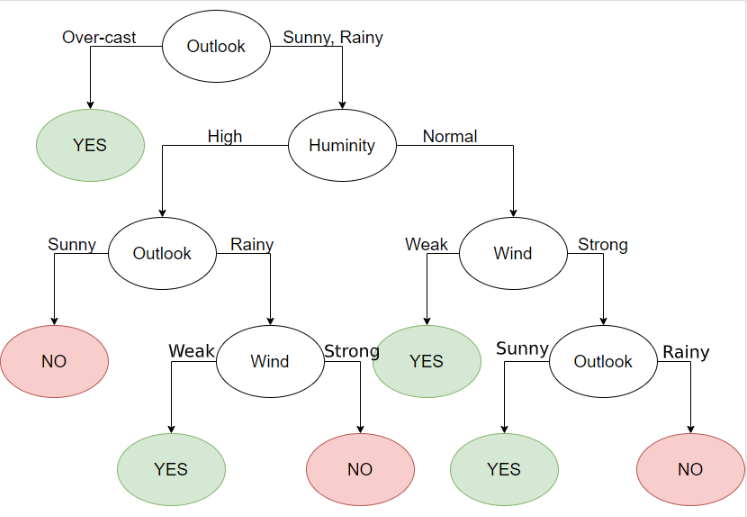
Tiếp tục phân chia theo các thuộc tính còn lại:

Ảnh có chứa văn bản

Mô tả được tạo tự động

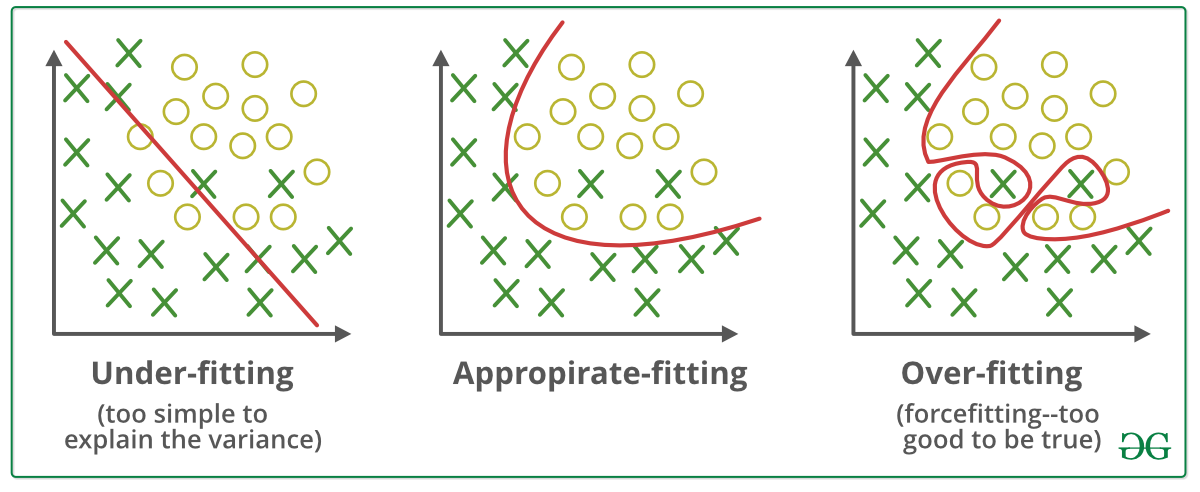
Như vậy thuộc tính tiếp theo được chọn là child node là *humidity*, tương tự các bước sau vẫn được làm như vậy.

Kết quả của *Gini* cho ví dụ trên sẽ cho ra một cây như sau:



**2.3 Overfitting và Underfitting trong CART**

* Nhắc lại một chút, overfitting có nghĩa là chúng ta cố gắn fit các dữ liệu một cách cực kì chính xác trong tập train, tuy nhiên nó sẽ dự đoán sai khi đưa dữ liệu vào test. Underfitting thì ngược lại, ta train một đường phân loại khá đơn giản, nó sẽ không chính xác trong cả tập train và tập test.



* Overfitting xảy ra khi độ sâu của cây là quá lớn
* Underfitting xảy ra khi cây khá ngắn

⟹Để hạn chế vấn đề underfitting và overfitting, ta có một kỹ thuật gọi là Tiêu chuẩn dừng (Stop criterion)

**2.4 Tiêu chuẩn dừng**

* Nếu chúng ta tiếp tục phát triển cây cho đến khi mỗi nút lá tương ứng với tạp chất thấp nhất, thì dữ liệu thường được cung cấp quá mức (overfitting).
* Nếu việc chia tách bị dừng quá sớm, lỗi về dữ liệu huấn luyện không đủ cao và hiệu suất sẽ bị ảnh hưởng do bias (underfitting).
* Do đó, việc ngăn chặn overfitting và underfitting là mấu chốt trong khi mô hình hóa Cây quyết định và nó có thể được thực hiện theo 2 cách:
  1. Đặt các ràng buộc về kích thước cây:
     + Cung cấp số lượng mẫu tối thiểu để phân chia nút.
     + Triển khai số lượng mẫu tối thiểu cho một nút lá (leaf node).
     + Cho phép độ sâu tối đa của cây (độ sâu dọc).
     + Số lượng terminal node tối đa.
     + Các tính năng tối đa để xem xét cho sự phân chia.
  2. Tỉa cây (Tree pruning):
     + Tỉa cây là một kỹ thuật trong học máy giúp giảm kích thước của Cây quyết định bằng cách loại bỏ các phần của cây. Nó cũng làm giảm độ phức tạp của phân loại cuối cùng và do đó cải thiện độ chính xác dự đoán bằng cách giảm overfitting.
     + Tỉa cây có thể được thực hiện theo hai cách:
       - Pre-prunning:
         * Dừng phân tách nút hiện tại nếu nó không cải thiện entropy bằng ít nhất một số giá trị (ngưỡng) được đặt trước.
         * Dừng phân vùng nếu số lượng điểm dữ liệu ít hơn một số giá trị đặt trước (ngưỡng).
         * Giới hạn độ sâu của cây đối với một số giá trị được đặt trước (ngưỡng).
       - Post-prunning:
         * Điều này có thể được thực hiện bằng cách trước tiên cho phép cây phát triển hết tiềm năng và sau đó tỉa cây ở mỗi cấp sau khi tính toán độ chính xác chéo ở mỗi cấp.

**3. Demo đơn giản**

Tiếp theo, ta sữ dụng thư viện sklearn để dào tạo mô hình cây quyết định dựa trên dữ liệu hoa Iris.

<Jupyter Notebook>

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn** **import** datasets

**import** **numpy** **as** **np**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn.preprocessing** **import** StandardScaler

iris = datasets.load\_iris()

*#2: petal length*

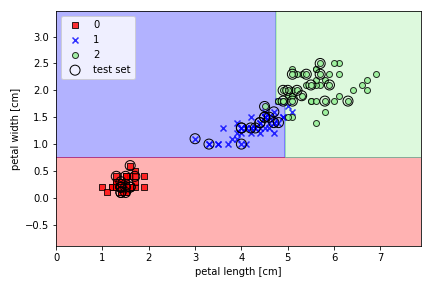
*#3: petal width*

X = iris.data[:, [2, 3]]

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=1, stratify=y)



Như đã đề cập, CART cũng có thể xử lý vấn đề hồi quy bằng cách sử dụng một tiêu chí phân tách khác: Lỗi bình phương trung bình (MSE) để xác định các điểm phân tách. Biến đầu ra của Cây hồi quy là số và các biến đầu vào cho phép kết hợp các biến liên tục và biến phân loại.

**4. Kết luận**

**4.1 Ưu điểm của CART**

* Cây quyết định có thể thực hiện phân loại đa lớp.
* Cung cấp hầu hết khả năng diễn giải mô hình bởi vì chúng đơn giản như là một loạt các điều kiện if-else.
* Có thể xử lý cả dữ liệu số và dữ liệu phân loại.
* Mối quan hệ phi tuyến (Nonlinear relationships) giữa các tính năng không ảnh hưởng đến hiệu suất của Cây quyết định.

**4.2 Nhược điểm của CART**

* Nhược điểm lớn nhất của Cây quyết định là vấn đề Overfitting.
* Một thay đổi nhỏ trong bộ dữ liệu có thể làm cho cấu trúc cây không ổn định có thể gây ra phương sai.
* Cây quyết định có thể bị underfit nếu dữ liệu mất cân bằng. Do đó, nên cân bằng tập dữ liệu trước khi phù hợp với Cây quyết định.

**4.3 Chuẩn bị dữ liệu cho CART**

* Việc phân chia các đặc trưng dạng số (splitting of numerical features) có thể được thực hiện bằng cách sắp xếp các đặc trưng theo thứ tự tăng dần và thử từng giá trị làm điểm ngưỡng, sau đó tính toán mức tăng thông tin cho từng giá trị làm ngưỡng. Cuối cùng, nếu giá trị đó thu được bằng với ngưỡng mang lại giá trị Gini Impurity tối thiểu thì hoan hô!!
* Đặc trưng chia tỷ lệ (Feature scaling) (tiêu chuẩn hóa cột) không cần thiết để thực hiện trong các cây quyết định. Tuy nhiên, nó giúp với việc hiển thị / thao tác dữ liệu và có thể hữu ích nếu bạn có ý định so sánh hiệu suất với dữ liệu khác hoặc các phương pháp khác như SVM.
* Để xử lý các đặc trưng dạng phân loại (categorical features) trong Cây quyết định, chúng ta không bao giờ phải thực hiện *one hot encoding* trên một biến phân loại vì hầu hết các thư viện có thể tự động xử lý các biến phân loại. Chúng ta vẫn có thể gán một số cho mỗi biến nếu muốn.
* Lớp không cân bằng (Imbalanced class) có tác động bất lợi đến cấu trúc cây, do đó có thể tránh được bằng cách sử dụng upsampling hoặc bằng cách sử dụng downsampling tùy thuộc vào tập dữ liệu.
* Hạn chế việc có quá nhiều đặc trưng (high dimensionality) vì nó có thể có tác động xấu đến cấu trúc của cây.
* Các ngoại lệ (Outliers) cũng tác động đến cấu trúc của cây vì độ sâu của cây có thể làm tăng các ngoại lệ.

**5. Tài liệu tham khảo**

<https://towardsdatascience.com/decision-trees-d07e0f420175>

[https://towardsdatascience.com/decision-tree](https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4)…

[https://www.vebuso.com/2020/01/decision-tree…](https://www.vebuso.com/2020/01/decision-tree-intuition-from-concept-to-application/)

[https://www.digitalvidya.com/blog/classification…](https://www.digitalvidya.com/blog/classification-and-regression-trees/)

[https://www.youtube.com/watch?v…](https://www.youtube.com/watch?v=DCZ3tsQIoGU&t=314s)

<https://dataisg.org/tutorial/machine-learning/cay-quyet-dinh-cart/>