ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: LẬP TRÌNH PYTHON CHO MÁY HỌC

ĐỀ TÀI

DECISION TREE REGRESSION

Giảng viên hướng dẫn: Nguyễn Vinh Tiệp

Sinh viên thực hiện: Tô Thanh Hiền - 19521490
Trần Vĩ Hào - 19521482
Lê Đặng Đăng Huy - 19521612

Lớp: CS116.M12.KHCL

Thành phố Hồ Chí Minh, ngày 14 tháng 12 năm 2021

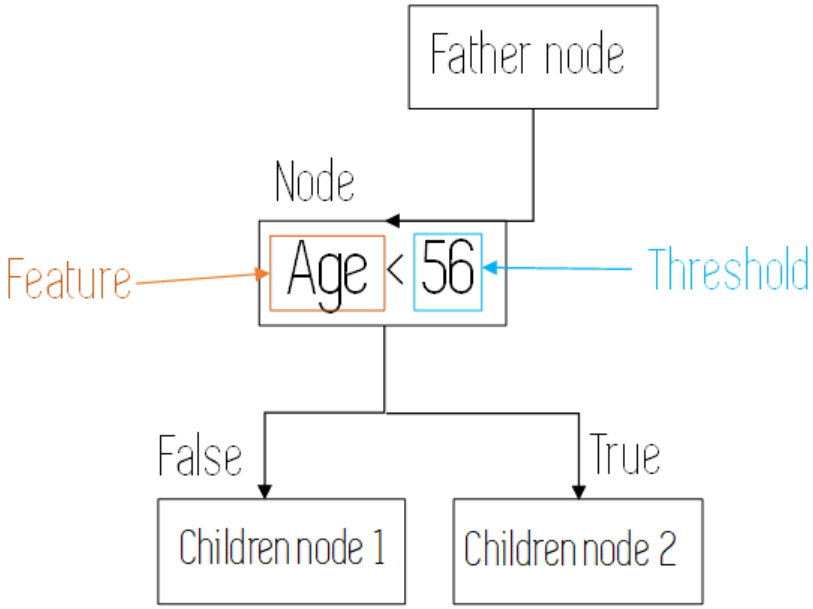
MỤC LỤC

I. GIỚI THIỆU CHUNG

1. Decision tree

Trong lý thuyết quyết định thống kê (chẳng hạn quản lý rủi ro), một ***cây quyết định*** (decision tree) là một đồ thị của các quyết định và các hậu quả có thể của nó (bao gồm rủi ro và hao phí tài nguyên). Cây quyết định được sử dụng để xây dựng một kế hoạch nhằm đạt được mục tiêu mong muốn. Các cây quyết định được dùng để hỗ trợ quá trình ra quyết định. Cây quyết định là một dạng đặc biệt của cấu trúc cây.

Trong lĩnh vực máy học, cây quyết định là một kiểu mô hình dự báo (predictive model), nghĩa là một ánh xạ từ các quan sát về một sự vật/hiện tượng tới các kết luận về giá trị mục tiêu của sự vật, hiện tượng. Hay nói một cách chính xác hơn thì cây quyết định là một mô hình học có giám sát (supervised learning) Mỗi một nút trong (internal node) tương ứng với một biến; đường nối giữa nó với nút con của nó thể hiện một giá trị cụ thể cho biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu, cho trước các giá trị của các biến được biểu diễn bởi đường đi từ nút gốc tới nút lá đó. Kỹ thuật học máy dùng trong cây quyết định được gọi là học bằng cây quyết định, hay chỉ gọi với cái tên ngắn gọn là cây quyết định.



Cấu trúc tại mỗi nút

Học bằng cây quyết định cũng là một phương pháp thông dụng trong khai phá dữ liệu. Khi đó, cây quyết định mô tả một cấu trúc cây, trong đó, các lá đại diện cho các phân loại còn cành đại diện cho các kết hợp của các thuộc tính dẫn tới phân loại đó. Một cây quyết định có thể được học bằng cách chia tập hợp nguồn thành các tập con dựa theo một kiểm tra giá trị thuộc tính. Quá trình này được lặp lại một cách đệ quy cho mỗi tập con dẫn xuất. Quá trình đệ quy hoàn thành khi không thể tiếp tục thực hiện việc chia tách được nữa, hay khi một phân loại đơn có thể áp dụng cho từng phần tử của tập con dẫn xuất. Một bộ phân loại rừng ngẫu nhiên (random forest) sử dụng một số cây quyết định để có thể cải thiện tỉ lệ phân loại.

Cây quyết định cũng là một phương tiện có tính mô tả dành cho việc tính toán các xác suất có điều kiện.

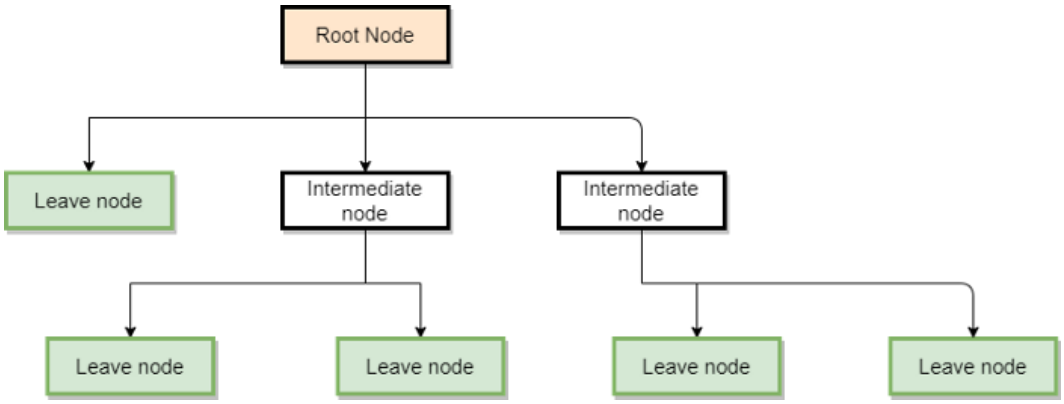
Cây quyết định có thể được mô tả như là sự kết hợp của các kỹ thuật toán học và tính toán nhằm hỗ trợ việc mô tả, phân loại và tổng quát hóa một tập dữ liệu cho trước.

Dữ liệu được cho dưới dạng các bản ghi có dạng:

$$(x, y) = (x_1, x_2, x_3..., x_k, y)$$

Trong đó:

- Biến phụ thuộc (dependant variable) y là biến mà chúng ta cần tìm hiểu, phân loại hay tổng quát hóa.
- x_1, x_2, x_3... là các biến sẽ giúp ta thực hiện công việc đó.



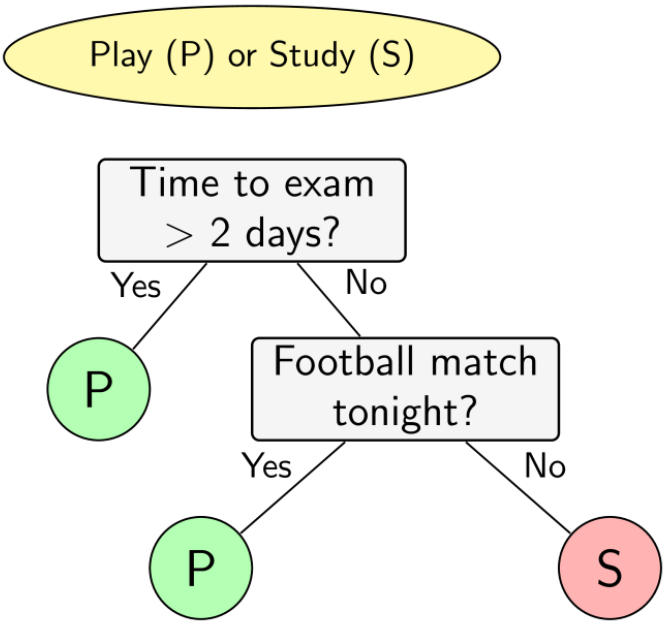
Mô hình cây quyết định

2. Ví dụ thực tiễn

Sắp đến kỳ thi, một cậu sinh viên tự đặt ra quy tắc ***học*** hay ***chơi*** của mình như sau:

- Nếu còn nhiều hơn hai ngày tới ngày thi, cậu ta sẽ đi chơi.
- Nếu còn không quá hai ngày và đêm hôm đó có một trận bóng đá, cậu sẽ sang nhà bạn chơi và cùng xem bóng đêm đó.
- Cậu sẽ chỉ học trong các trường hợp còn lại.

Việc ra quyết định của cậu sinh viên này có thể được mô tả trên sơ đồ trong sơ đồ bên dưới. Hình ellipse nền vàng thể hiện quyết định cần được đưa ra. Quyết định này phụ thuộc vào các câu trả lời của các câu hỏi trong các ô hình chữ nhật màu xám. Dựa trên các câu trả lời, quyết định cuối cùng được cho trong các hình tròn màu lục (***chơi***) và đỏ (***học***). Sơ đồ như bên dưới còn được gọi là một ***cây quyết định***.



3. Phân loại

Cây quyết định có hai loại:

- **Cây hồi quy (Decision Tree Regression)** ước lượng các hàm giá có giá trị là số thực thay vì được sử dụng cho các nhiệm vụ phân loại. (ví dụ: ước tính giá một ngôi nhà hoặc khoảng thời gian một bệnh nhân nằm viện)
- **Cây phân loại (Decision Tree Classification)**, nếu y là một biến phân loại như: giới tính (nam hay nữ), kết quả của một trận đấu (thắng hay thua).

→ Và ở bài báo cáo lần này, chúng ta sẽ đi sâu hơn về **Cây hồi quy**.

4. Một số thuật toán thường gặp

Thuật toán **ID3** (viết tắt của *Iterative Dichotomiser 3*) là một giải thuật khá lâu đời được tạo ra bởi ***Ross Quinlan*** (1986) nhằm xây dựng cây quyết định phù hợp từ một bộ dữ liệu. Đây là giải thuật tiền đề mà dựa trên cơ sở đó, rất nhiều những giải thuật khác liên quan tới cây quyết định được kế thừa và phá triển:

- **C4.5** [Quinlan (1993)]: Kế thừa của thuật toán ID3. Giải thuật này được sử dụng phổ biến trong machine learning và xử lý ngôn ngữ tự nhiên.
- **CART** [Breiman et al. (1984)]: Viết tắt của cụm từ *Classification And Regression Tree*. Ưu điểm của nó là có thể sử dụng cho cả bài toán phân loại và hồi qui.

- **CHAID**: Sử dụng phân phối χ^2 để tự động tương tác phát hiện phân chia khi tính toán cây phân loại.
- **MARS**: Áp dụng hồi qui đa biến theo splines. Đây là một phương pháp hồi qui chia để trị, có thể loại bỏ ảnh hưởng của outliers.

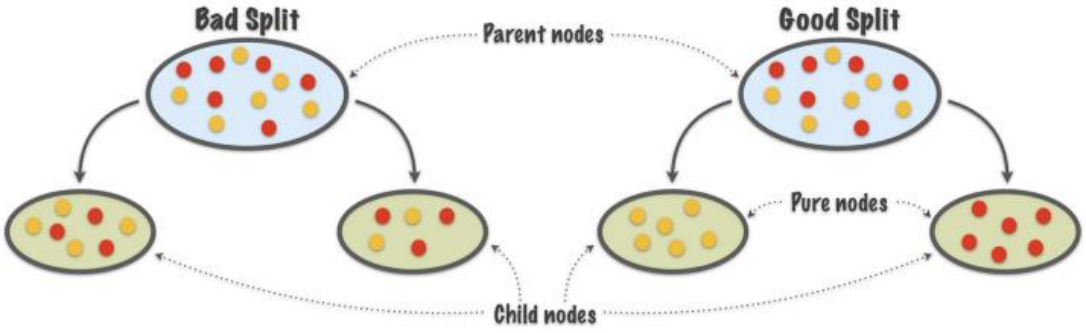
5. Xây dựng cây quyết định

Giải thuật học cây quyết định gồm 2 bước lớn: xây dựng cây (Top-down) và cắt nhánh (Bottom-up) để tránh học vẹt.

Quá trình xây dựng cây được làm như sau:

- Việc xây dựng cây quyết định được tiến hành một cách đệ qui, lần lượt từ nút gốc xuống tới tận các nút lá. Tại mỗi nút hiện hành đang xét, nếu kiểm tra thấy thoả điều kiện dừng: thuật toán sẽ tạo nút lá. Nút này được gán một giá trị của nhãn lớp tùy điều kiện dừng được thoả. Ngược lại, thuật toán tiến hành chọn điểm chia tốt nhất theo một tiêu chí cho trước, phân chia dữ liệu hiện hành theo điều kiện chia này.

- Lưu ý dữ liệu hiện hành không phải hoàn toàn là tập dữ liệu ngay khi bắt đầu thuật toán, có thể là tập dữ liệu đã được phân chia theo điều kiện chia của nút liền trước đó (nút cha).



- Sau bước phân chia trên, thuật toán sẽ lặp qua tất cả các tập con (đã được chia) và tiến hành gọi đệ qui như bước đầu tiên với dữ liệu chính là các tập con này.

Quá trình xây dựng cây chủ yếu phụ thuộc vào việc chọn thuộc tính tốt nhất để phân hoạch dữ liệu.

6. Điều kiện dừng

Quá trình đệ quy xây dựng cây quyết định được tiếp tục thực hiện cho đến khi gặp một trong các điều kiện dừng:

- Mọi các mẫu trong tập huấn luyện đều được phân lớp.
- Không còn thuộc tính nào có thể dùng để phân chia mẫu nữa.
- Không còn lại mẫu nào tại nút.

II. DECISION TREE REGRESSION

1. Định nghĩa hồi quy

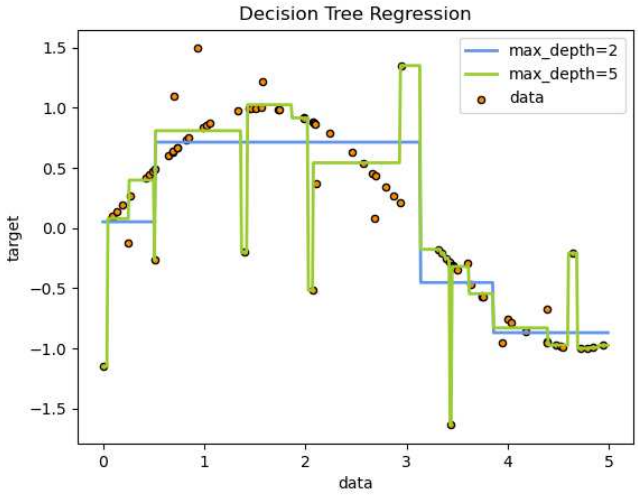
Hồi quy (Regression) là quá trình tìm một mô hình hoặc hàm để phân biệt dữ liệu thành các giá trị thực liên tục thay vì sử dụng các lớp. Về mặt toán học, với một vấn đề hồi quy, người ta đang cố gắng tìm gần đúng hàm với độ lệch lỗi tối thiểu. Trong hồi quy, sự phụ thuộc số dữ liệu được dự đoán để phân biệt nó.

Phân tích hồi quy là mô hình thống kê được sử dụng để dự đoán dữ liệu số thay vì nhãn. Nó cũng có thể xác định phong trào phân phối tùy thuộc vào dữ liệu có sẵn hoặc dữ liệu lịch sử.

Sự khác biệt chính giữa phân loại và hồi quy:

- Quá trình phân loại mô hình hóa một chức năng thông qua đó dữ liệu được dự đoán trong các nhãn lớp rời rạc. Mặt khác, hồi quy là quá trình tạo ra một mô hình dự đoán số lượng liên tục.
- Các thuật toán phân loại liên quan đến cây quyết định, hồi quy logistic, v.v ... Ngược lại, cây hồi quy (ví dụ: Random Forest,…) và hồi quy tuyến tính là những ví dụ về thuật toán hồi quy.
- Phân loại dự đoán dữ liệu không theo thứ tự trong khi hồi quy dự đoán dữ liệu theo thứ tự.
- Hồi quy có thể được đánh giá bằng cách sử dụng lỗi bình phương trung bình gốc. Ngược lại, phân loại được đánh giá bằng cách đo độ chính xác.

2. Cây quyết định hồi quy (Decision tree regression)



Hồi quy cây quyết định quan sát các đặc trưng của một đối tượng và đào tạo một mô hình trong cấu trúc của cây để dự đoán dữ liệu trong tương lai nhằm tạo ra đầu ra liên tục (continuous output) có ý nghĩa. Đầu ra liên tục nghĩa là đầu ra kết quả không rời rạc, tức là nó không được biểu diễn chỉ bằng một tập hợp số hoặc giá trị rời rạc, đã biết trước.

VD:

- Đầu ra rời rạc (Discrete output): Mô hình dự đoán thời tiết dự đoán có hay không có mưa vào một ngày cụ thể.
- Đầu ra liên tục(Continuous output): Một mô hình dự đoán lợi nhuận cho biết lợi nhuận có thể được tạo ra từ việc bán một sản phẩm.

Ở đây, các giá trị liên tục được dự đoán với sự trợ giúp của mô hình **hồi quy cây quyết định**.

Cây hồi quy không quá khác so với cây phân loại và điều chỉnh duy nhất mà chúng ta cần thực hiện là cách chúng ta đo độ tạp chất. Thay vì đo tạp chất của các nút, bây giờ chúng ta muốn chọn phép phân tách để giảm thiểu tổng sai số bình phương trong các nút con. Sau đó, khi đưa ra dự đoán, chúng tôi sử dụng giá trị trung bình (từ dữ liệu huấn luyện) của bất kỳ nút lá nào mà một quan sát rơi vào.

3. Các tham số của mô hình Decision Tree Regression

class sklearn.tree.DecisionTreeRegressor(*, *criterion='squared_error'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *ccp_alpha=0.0*)

- **criterion : *{"squared_error", "friedman_mse", "absolute_error", "poisson"}, default="squared_error"***

  Hàm số đo lường chất lượng phân chia ở mỗi nút. Các tiêu chí được hỗ trợ là:

  - *"squared_error"* cho sai số bình phương trung bình (MSE), bằng với giảm phương sai làm tiêu chí lựa chọn hàm số và giảm thiểu tổn thất L2 bằng cách sử dụng giá trị trung bình của mỗi nút đầu cuối
  - *"friedman_mse"*, sử dụng sai số bình phương trung bình với điểm cải thiện của Friedman cho khả năng chia tách
  - *"absolute_error"* cho lỗi tuyệt đối trung bình (MAE), giảm thiểu tổn thất L1 bằng cách sử dụng giá trị trung bình của mỗi nút đầu cuối
  - *"poisson"* sử dụng giảm độ lệch trong Poisson để tìm các phần tách.

- **splitter : *{"best", "random"}, default="best"***
  - Chiến lược được sử dụng để chọn phần tách tại mỗi nút.
  - Các chiến lược được hỗ trợ là "best" để chọn phần tách tốt nhất và "random" để chọn phần tách ngẫu nhiên tốt nhất.

- **max_depth : *int, default=None***
  - Chiều sâu tối đa của cây. Nếu None, thì các nút được mở rộng cho đến khi tất cả các lá đều thuần khiết, không pha trộn hoặc cho đến khi tất cả các lá chứa ít hơn mẫu min_samples_split.
  - Điều này cho biết cây có thể sâu đến mức nào. Cây càng sâu, càng có nhiều sự phân chia và nắm bắt được nhiều thông tin hơn về dữ liệu.
  - Đối với mô hình bị quá khớp thì chúng ta cần gia tăng độ sâu và vị khớp thì giảm độ sâu.

- **min_samples_split : *int or float, default=2***
  - Số lượng mẫu tối thiểu cần thiết để tách một nút trong (internal node):
  - Nếu int, thì hãy xem min_samples_split như là số nhỏ nhất.
  - Nếu float, thì min_samples_split là một phân số và ceil(min_samples_split * n_samples) là số lượng mẫu tối thiểu cho mỗi lần tách.

- Kích thước mẫu tối thiểu được yêu cầu để tiếp tục phân chia đối với nút quyết định. Được sử dụng để tránh kích thước của nút lá quá nhỏ nhằm giảm thiểu hiện tượng quá khớp.
- Điều này có thể khác nhau giữa việc xem xét ít nhất một mẫu tại mỗi nút đến việc xem xét tất cả các mẫu tại mỗi nút. Khi chúng ta tăng tham số này, cây trở nên hạn chế hơn vì nó phải xem xét nhiều mẫu hơn ở mỗi nút.

➢ **min_samples_leaf :** ***int or float, default=1***

- Số lượng mẫu tối thiểu cần thiết để có tại một nút lá. Một điểm phân 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.
  + Nếu int, thì hãy xem min_samples_leaf là số nhỏ nhất.
  + Nếu float, thì min_samples_leaf là một phân số và ceil(min_samples_leaf * n_samples) là số lượng mẫu tối thiểu cho mỗi nút.
- Tham số này tương tự như min_samples_splits, tuy nhiên tham số này mô tả số lượng mẫu tối thiểu của mẫu ở các lá, gốc của cây.

➢ **min_weight_fraction_leaf :** ***float, default=0.0***

- Phần có trọng số tối thiểu của tổng trọng số (của tất cả các mẫu đầu vào) cần có ở một nút lá.
- Các mẫu có trọng lượng bằng nhau khi sample_weight không được cung cấp.

➢ **max_features :** ***int, float or {"auto", "sqrt", "log2"}, default=None***

- Số lượng các biến được lựa chọn để tìm kiếm ra biến phân chia tốt nhất ở mỗi lượt phân chia:
  + Nếu int, thì hãy xem xét các đặc trưng max_features tại mỗi lần tách.
  + Nếu float, thì max_features là một phân số và các đặc trưng int(max_features * n_features) được xem xét ở mỗi lần tách.
  + Nếu "auto", thì max_features=n_features.
  + Nếu "sqrt", thì max_features=sqrt(n_features).
  + Nếu "log2", thì max_features=log2(n_features).
  + Nếu None, thì max_features=n_features.
- **Lưu ý:** việc tìm kiếm phân tách không dừng lại cho đến khi tìm thấy ít nhất một phân vùng hợp lệ của các mẫu nút, ngay cả khi nó yêu cầu kiểm tra hiệu quả nhiều hơn các đặc trưng max_features.

➢ **random_state :** ***int, RandomState instance or None, default=None***

- Kiểm soát tính ngẫu nhiên của công cụ ước tính. Các đặc trưng luôn được hoán vị ngẫu nhiên tại mỗi lần tách, ngay cả khi splitter được đặt thành "best". Khi max_features < n_features, thuật toán sẽ chọn max_features ngẫu nhiên tại mỗi lần tách trước khi tìm ra phần tách tốt nhất trong số chúng.
- Nhưng sự phân chia tốt nhất được tìm thấy có thể khác nhau trong các lần chạy khác nhau, ngay cả khi max_features=n_features. Đó là trường hợp, nếu sự cải tiến của **criterion** giống hệt nhau đối với một số phần tách và một phần tách phải

được chọn ngẫu nhiên. Để có được một hành vi xác định trong quá trình điều chỉnh, random_state phải được cố định thành một số nguyên.

- **max_leaf_nodes :** ***int, default=None***
  - Số lượng các nút lá tối đa của cây quyết định. Thường được thiết lập khi muốn kiểm soát hiện tượng quá khớp.
  - Xây dựng một cây với max_leaf_nodes theo cách tốt nhất.
  - Các nút tốt nhất được định nghĩa là giảm tạp chất (impurity) tương đối. Nếu None thì không giới hạn số nút lá.
- **min_impurity_decrease :** ***float, default=0.0***
  - Chúng ta sẽ tiếp tục phân chia một nút nếu như sự suy giảm của độ tinh khiết nếu phân chia lớn hơn hoặc bằng ngưỡng này.
  - Phương trình giảm trọng số tạp chất như sau:

    N_t / N * (impurity - N_t_R / N_t * right_impurity
    - N_t_L / N_t * left_impurity)

    Trong đó:

    + N là tổng số mẫu.
    + N_t là số lượng mẫu ở nút hiện tại.
    + N_t_L là số lượng mẫu ở nút con bên trái.
    + N_t_R là số lượng mẫu ở nút con bên phải.
  - N, N_t, N_t_R và N_t_L đều tham chiếu đến tổng trọng số, nếu sample_weight được thông qua.
- **ccp_alpha :** ***non-negative float, default=0.0***
  - Tham số độ phức tạp được sử dụng để **Minimal Cost - Complexity Pruning** (một phương pháp cắt tỉa cây tối đa).
  - Cây con có độ phức tạp chi phí lớn nhất nhỏ hơn ccp_alpha sẽ được chọn.
  - Theo mặc định, không có thao tác cắt tỉa nào được thực hiện.

4. Ưu điểm và khuyết điểm của mô hình

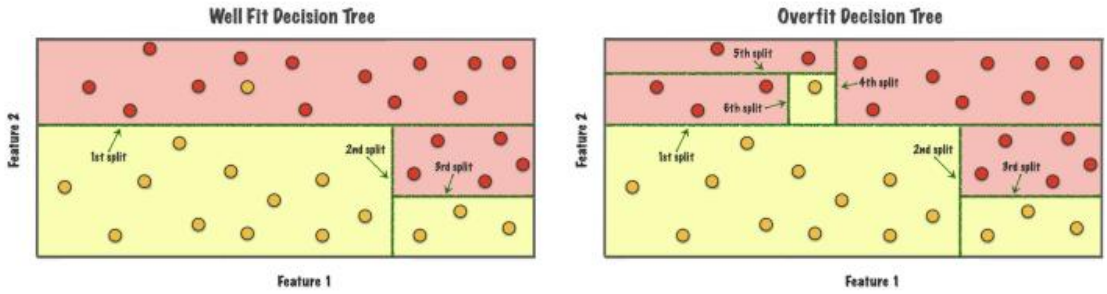
a. Ưu điểm

- Thuật toán Decision trees đơn giản, trực quan, không quá phức tạp để hiểu ngay lần đầu tiên, khác với các thuật toán ví dụ như Artificial Neural network không thể hiện rõ quy luật phân loại. Đồng thời bộ dữ liệu training không nhất thiết phải quá lớn để tiến hành xây dựng mô hình phân tích.
- Một số thuật toán cây quyết định có khả năng xử lý dữ liệu bị missing và dữ liệu bị lỗi mà không cần áp dụng phương pháp như "imputing missing values" hay loại bỏ. Bên cạnh đó Decision trees ít bị ảnh hưởng bởi các dữ liệu ngoại lệ (outliers)
- Thuật toán cây quyết định là phương pháp không sử dụng tham số, "nonparametric", nên không cần phải có các giả định ban đầu về các quy luật phân phối như trong thống kê, và nhờ đó kết quả phân tích có được là khách quan, "tự nhiên" nhất.

- Thuật toán cây quyết định mang lại kết quả dự báo có độ chính xác cao, dễ dàng thực hiện, nhanh chóng trong việc huấn luyện, không cần phải chuyển đổi các biến vì kết quả sẽ như nhau với bất kể loại biến dữ liệu biến đổi ra sao.
- Dựa trên quy luật ra quyết định (Decision rule) để xây dựng nên thuật toán cây quyết định rất dễ diễn giải hay giải thích đến người nghe, người xem – những người muốn hiểu rõ về kết .quả phân tích nhưng không có kiến thức gì về khoa học dữ liệu.
- Thuật toán cây quyết định vẫn nói lên được mối liên hệ giữa các biến, các thuộc tính dữ liệu một cách trực quan nhất mặc dù không thể hiện được rõ mối quan hệ tuyến tính, hay mức độ liên hệ giữa chúng như phương pháp phân tích hồi quy (regression analysis) có được.
- Ngoài kinh tế, tài chính, thuật toán cây quyết định có thể được ứng dụng trong lĩnh vực y tế, nông nghiệp, sinh học.

b. Khuyết điểm

- Thuật toán cây quyết định hoạt động hiệu quả trên bộ dữ liệu đơn giản có ít biến dữ liệu liên hệ với nhau, và ngược lại nếu áp dụng cho bộ dữ liệu phức tạp.
- Cụ thể, thuật toán cây quyết định khi được áp dụng với bộ dữ liệu phức tạp, nhiều biến và thuộc tính khác nhau có thể dẫn đến mô hình bị **overfitting**, quá khớp với dữ liệu training dẫn đến vấn đề không đưa ra kết quả phân loại chính xác khi áp dụng cho dữ liệu test, và dữ liệu mới.



Thêm ba phần tách mới để nắm bắt một điểm dữ liệu duy nhất có thể là quá mức

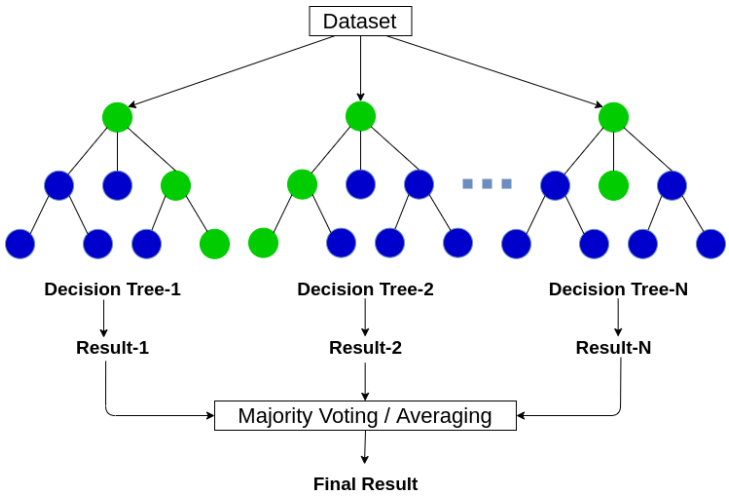
- Đối với thuật toán cây quyết định khi có sự thay đổi nhỏ trong bộ dữ liệu có thể gây ảnh hưởng đến cấu trúc của mô hình. Nghĩa là khi chúng ta điều chỉnh dữ liệu, cách thức phân nhánh, ngắt cây sẽ bị thay đổi, có thể dẫn đến kết quả sẽ khác so với ban đầu, phức tạp hơn. Các chuyên gia gọi đây là vấn đề "**high variance" -** giá trị phương sai cao.
- Đối với thuật toán cây quyết định áp dụng cho biến định lượng (regression tree), thì chỉ phân loại đối tượng, hay dự báo theo phạm vi giá trị (range) được tạo ra trước đó, vì vậy đây cũng là một hạn chế khi khả năng có nhiều phạm vi giá trị khác mà thuật toán chưa xét đến.
- Thuật toán cây quyết định có khả năng **"bias"** hay thiên vị nếu bộ dữ liệu không được cân bằng. Nói đơn giản, khi bộ dữ liệu được phân ra thành các nhóm theo các đặc trưng khác nhau nào đó, mà số lượng quan sát trong mỗi nhóm là quá chênh lệch hay khác biệt rõ rệt về đặc trưng, lúc này có thể dẫn đến mô hình bị **"bias"**, phân nhánh đơn giản, chỉ xét đến các giá trị tiêu biểu, và nguy cơ "**Underfitting"** (không rà soát hết các khả năng phân loại dữ liệu). Tuy nhiên khi mô hình quá phức tạp, mọi biến dữ liệu đều có khả năng phân nhánh và làm cơ sở

phân loại các đối tượng dữ liệu, thì lúc này **"bias"** ở mức thấp nhưng nguy cơ không thể áp dụng được dữ liệu mới.

- Thuật toán cây quyết định yêu cầu bộ dữ liệu training và test phải được chuẩn bị hoàn hảo, chất lượng tốt phải được cân đối theo các lớp, các nhóm trong biến mục tiêu. Ví dụ có sự chênh lệch không quá lớn giữa số lượng đối tượng dữ liệu thuộc lớp A của biến mục tiêu và số lượng đối tượng dữ liệu thuộc lớp B của biến mục tiêu. Ngoài ra biến mục tiêu phải có các giá trị "rời rạc" dễ nhận biết, không được quá đa dạng, và phải cụ thể để quá trình phân loại diễn ra dễ dàng hơn cho thuật toán.
- Thuật toán cây quyết định được hình thành trên các cách thức phân nhánh tại mỗi một thời điểm bất kỳ, ở một node hay biến dữ liệu bất kỳ và chỉ quan tâm duy nhất vào việc phân nhánh sao cho tối ưu tại thời điểm ấy, chứ không xét đến toàn bộ mô hình phải được thiết lập hiệu quả ra sao. Do đó sẽ có trường hợp chúng ta cảm thấy việc phân nhánh dễ dàng, cứ thế tiếp tục cho đến khi không còn đối tượng dữ liệu để phân loại nhưng khi kết thúc nhìn lại sao mô hình lại quá cồng kềnh, phức tạp. Lúc này thì không thể tìm ra nguyên nhân.
- Thuật toán cây quyết định **không** "hỗ trợ" kỹ thuật hay khả năng "truy vấn ngược" mà chỉ phân nhánh liên tục dựa trên các công thức phân nhánh cho đến khi thấy được kết quả sau cùng nên chúng ta khó phát hiện được các lỗi ở đâu nếu có sai sót.

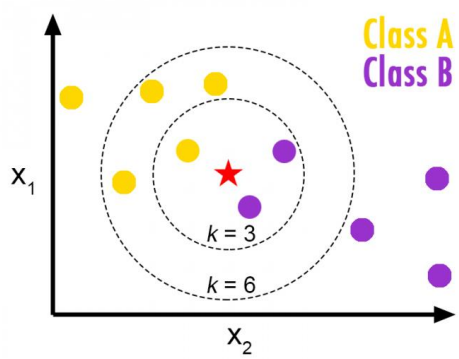
III. SO SÁNH VỚI CÁC MÔ HÌNH KHÁC

1. Decision tree vs Random Forest (Rừng ngẫu nhiên)



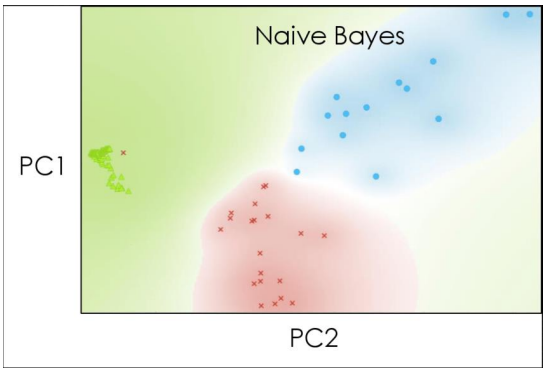
- Rừng ngẫu nhiên là một tập hợp các Cây quyết định và phiếu bầu trung bình hay đa số của khu rừng được chọn làm kết quả dự đoán.
- Mô hình Rừng ngẫu nhiên sẽ ít bị quá tải so với Cây quyết định và đưa ra giải pháp tổng quát hơn.
- Rừng Ngẫu nhiên mạnh mẽ và chính xác hơn Cây quyết định.

2. Decision tree vs KNN



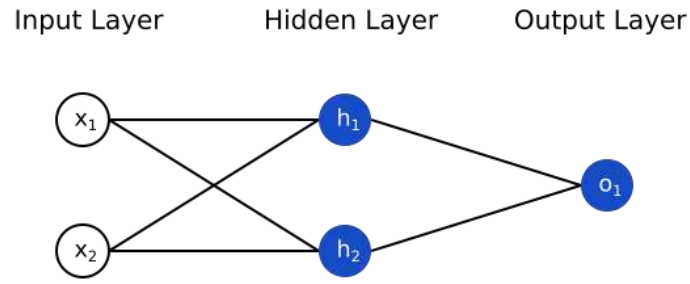
- Cả hai đều là phương pháp phi tham số (non-parametric).
- Cây quyết định hỗ trợ tương tác tính năng tự động, trong khi KNN không thể.
- Cây quyết định nhanh hơn do thực thi thời gian thực đắt đỏ của KNN.

3. Decision tree vs Naive Bayes



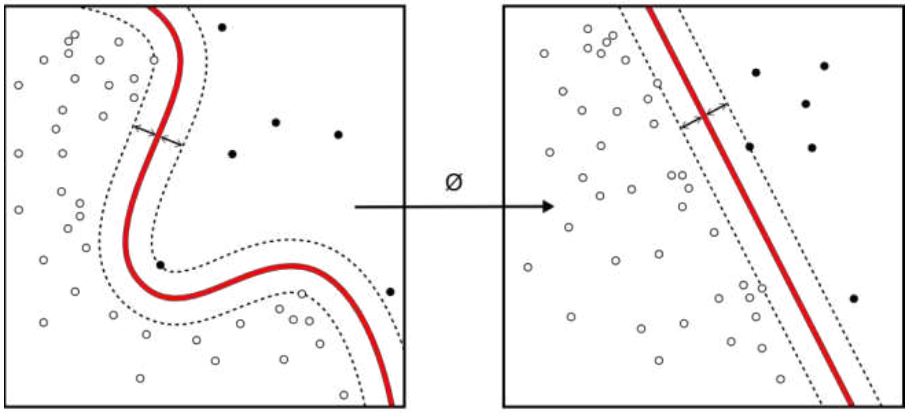
- Cây quyết định là một mô hình phân biệt, trong khi Naive Bayes là một mô hình tổng hợp.
- Cây quyết định linh hoạt và dễ dàng hơn.
- Việc cắt tỉa cây quyết định có thể bỏ qua một số giá trị quan trọng trong dữ liệu huấn luyện, điều này có thể dẫn đến độ chính xác của một lần tung (toss).

4. Decision tree vs Neural Network



- Cả hai đều tìm ra các nghiệm phi tuyến tính và có sự tương tác giữa các biến độc lập.
- Cây quyết định tốt hơn khi có một tập hợp lớn các giá trị phân loại trong dữ liệu huấn luyện.
- Cây quyết định tốt hơn Neural Network, khi kịch bản yêu cầu giải thích về quyết định.
- Neural Network hoạt động tốt hơn cây quyết định khi có đầy đủ dữ liệu huấn luyện.

5. Decision tree vs SVM



- SVM sử dụng thủ thuật hạt nhân (kernel trick) để giải quyết các vấn đề phi tuyến tính (non-linear) trong khi Cây quyết định lấy ra các siêu hình chữ nhật (hyper-rectangle) trong không gian đầu vào để giải quyết vấn đề.
- Cây quyết định tốt hơn cho dữ liệu phân loại và nó xử lý thuộc tính tốt hơn SVM.

IV. CÁCH TUNING THAM SỐ

Trong thuật toán cây quyết định chúng ta có thể quan tâm tới một số tham số có thể được sử dụng để tuning. Đó là những tham số chính bên dưới:

```
parameters = {'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
              'max_depth': [None,1,2,3,4,5,6,7,8,9,10],
              'min_samples_split': [2,4,6,8,10,12,14,16,18,20,22,24,26,28,30],
              'min_samples_leaf': [1,2,3,4,5,6,7,8,9,10],
              'max_leaf_nodes': [None,10,20,30,40,50,60,70,80],
              'min_impurity_decrease': [0.0,0.05,0.1,0.15,0.2]          }
model_cv = GridSearchCV(estimator = clf,
                        param_grid = hyper_params,
                        scoring= 'neg_mean_squared_error',
                        cv = 4,
                        verbose = 3,
                        return_train_score=True)
```

Ở trên, chúng ta đã phức tạp hóa các siêu tham số ngẫu nhiên bằng cách sử dụng **Gridsearch** để tìm các tham số tốt nhất cho mô hình cây quyết định.

Sau đó, chúng ta có thể kiểm tra các thông số tốt nhất do **GridSearchCV** tìm thấy trong thuộc tính best_params_ và công cụ ước tính tốt nhất trong thuộc tính best_estimator_.

V. MÔ TẢ BỘ DATASET

1. Mô tả bộ Dataset

Bộ dataset mà chúng em sử dụng là dataset rượu quang đỏ "Vinho Verde" lấy từ bên UCL machine learning. Link dataset tại đây.

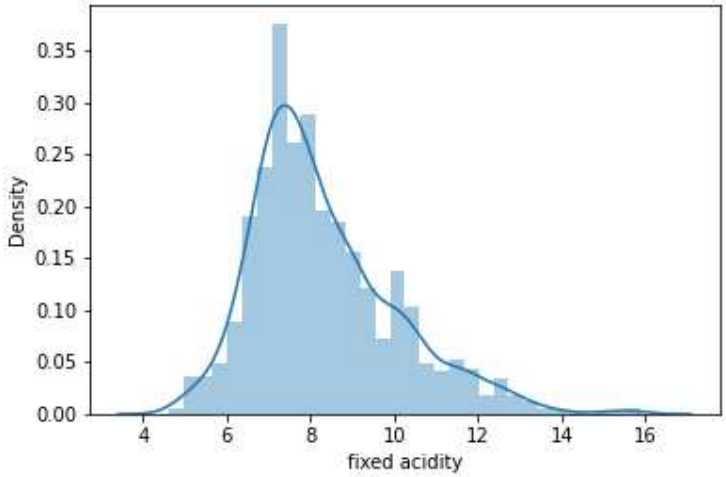
Input:

1. **Độ axit cố định (fixed acidity):** là thước đo axit không bay hơi hoặc cố định. Các axit này hầu hết bắt nguồn từ nho. Các axit cố định chủ yếu tìm thấy trong rượu vang là tartaric, malic, citric và succinic. Đơn vị tính là (g/dm^3).
2. **Độ bay hơi axit (volatile acidity):** là thước đo axit dễ bay hơi (hoặc ở dạng khí) của rượu vang. Axit dễ bay hơi chính trong rượu vang là axit axetic, đây cũng là axit chính liên quan đến mùi và vị của giấm. Mức độ quá nhiều có thể dẫn tới vị khó chịu và mùi giấm cho người uống. Đơn vị tính là (g/dm^3).
3. **Axit citric (citric acid):** là axit hữu cơ yếu, được cho vào rượu vang như chất bảo quản tự nhiên và tạo thêm độ chua và độ tươi cho rượu. Đơn vị tính là (g/dm^3).
4. **Đường dư (residual sugar):** là lượng đường tự nhiên từ trái nho còn lại sau khi quá trình lên men rượu vang đỏ kết thúc. Hiếm khi tìm thấy loại rượu nào dưới 1 gam/lít và những loại rượu lớn hơn 45 gam/lít được coi là ngọt .Đơn vị tính là (g/dm^3).
5. **Clorua(chlorides):** số lượng muối có trong rượu. Đơn vị tính (g/dm^3).
6. **Lưu huỳnh đioxit tự do (Free sulfur dioxide):** là thước đo lượng SO2 không liên kết với các phân tử khác và sử dụng để tính SO2 phân tử. Được dùng trong quá trình sản xuất rượu vang để ngăn chặn quá trình oxi hóa và sự phát triển của vi sinh vật. Đơn vị tính là (mg/dm^3).
7. **Tổng lượng lưu huỳnh đioxit (total sulfur dioxide):** là thước đo của dạng liên kết và tự do của SO2. SO2 liên kết dùng để chỉ các phân tử SO2 được liên kết với các hợp chất khác, chủ yếu là aldehyde, pyruvate và anthocyanins. Ở nồng độ thấp, hầu như không thể phát hiện được SO2 trong rượu vang, nhưng ở nồng độ SO2 tự do trên 50 ppm, SO2 trở nên rõ ràng trong mùi và vị của rượu. Đơn vị tính (mg/dm^3).
8. **Mật độ (density):** tỷ trọng của rượu tùy thuộc vào phần trăm độ cồn và hàm lượng đường Đơn vị tính (g/cm^3).
9. **Độ pH (pH):** mô tả độ axit hoặc bazơ của rượu trên thang từ 0 (rất axit) đến 14 (rất bazơ), hầu hết các loại rượu đều nằm trong khoảng 3-4 trên thang độ pH.
10. **Muối sulfat (sulphates):** Có công thức hóa học là (S04) (2-). là một chất phụ gia rượu vang có thể góp phần tạo ra khí ga sulfur dioxide (SO2), nó hoạt động như một chất chống vi khuẩn và chống oxi hóa. Đơn vị tính (g/dm^3).
11. **Cồn (alcohol):** nồng độ phần trăm cồn của rượu van. Đơn vị tính (%).

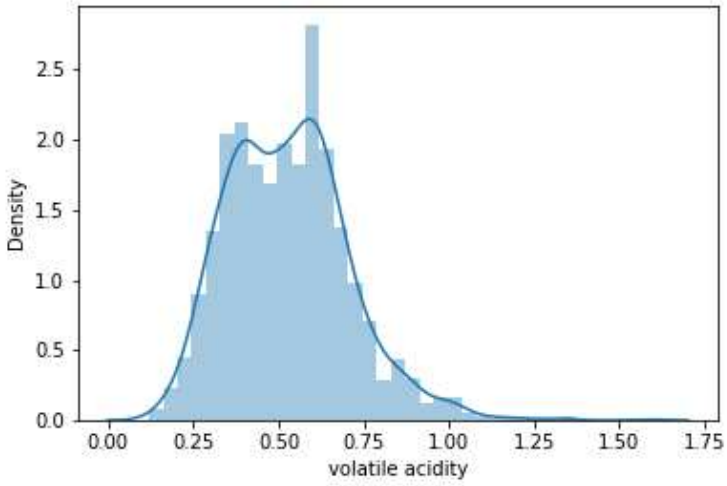
Output: **Chất lượng rượu (quality)** được tính trên thang điểm từ 0 đến 10.

2. Biểu đồ dữ liệu

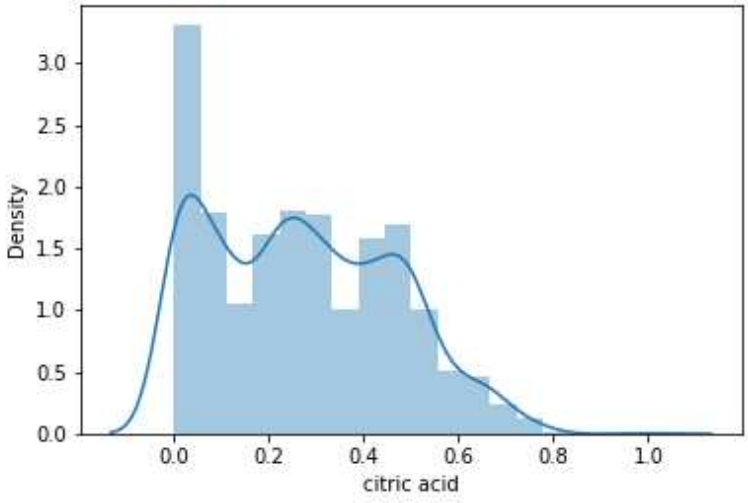
- Độ axit cố định (fixed acidity):



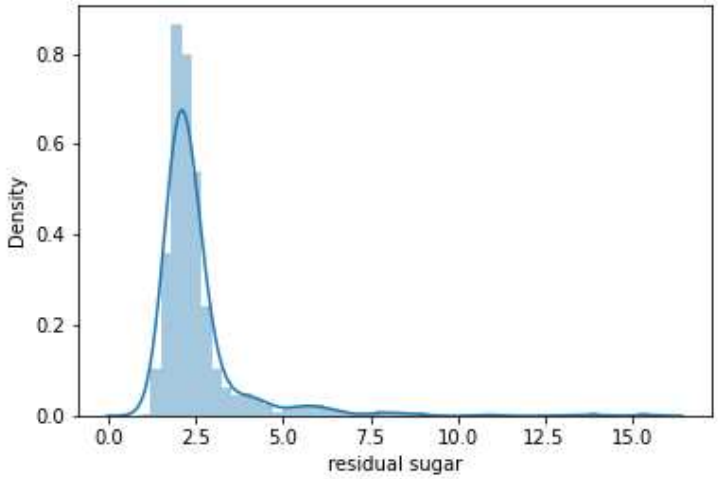
- Độ bay hơi axit (volatile acidity):



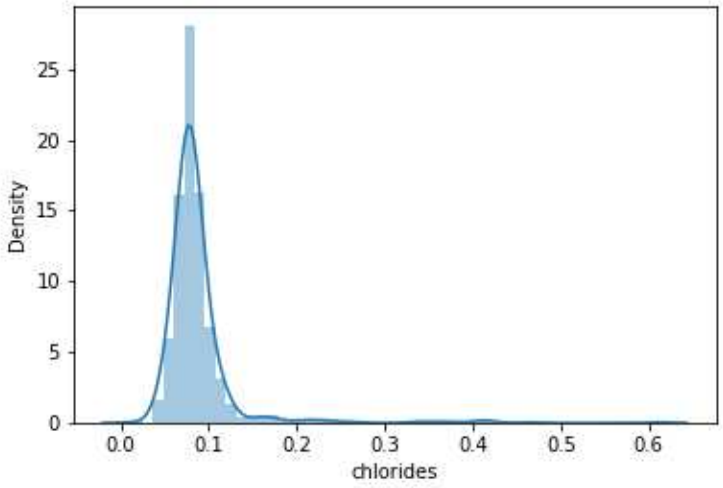
- Axit citric (citric acid):



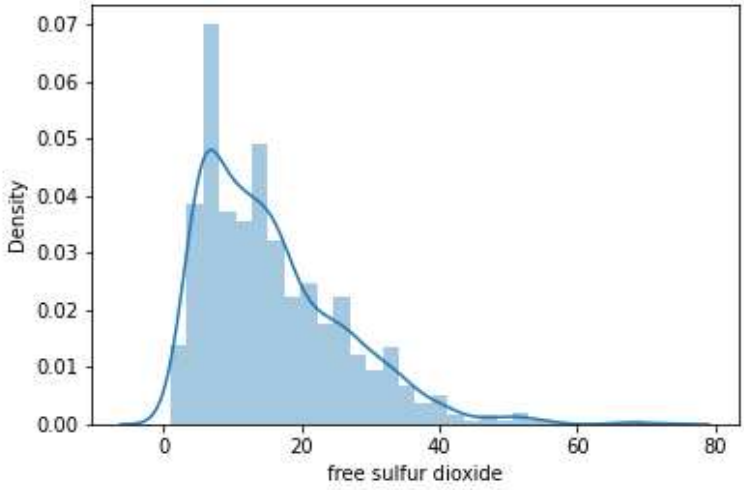
- Đường dư (residual sugar)



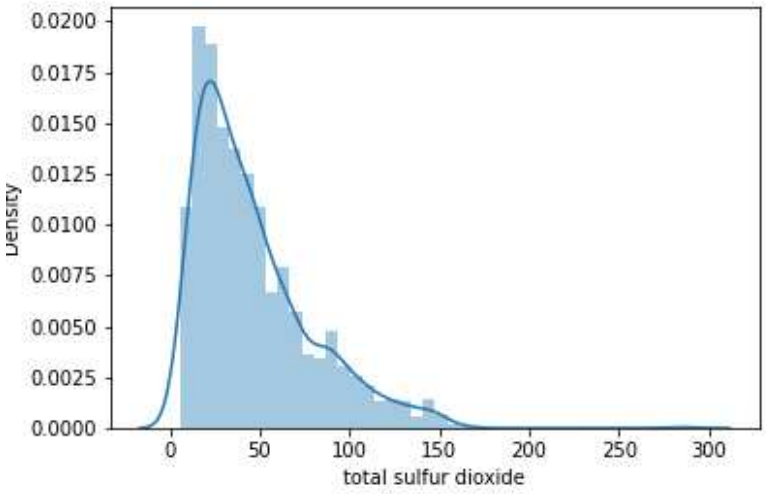
- Clorua(chlorides)



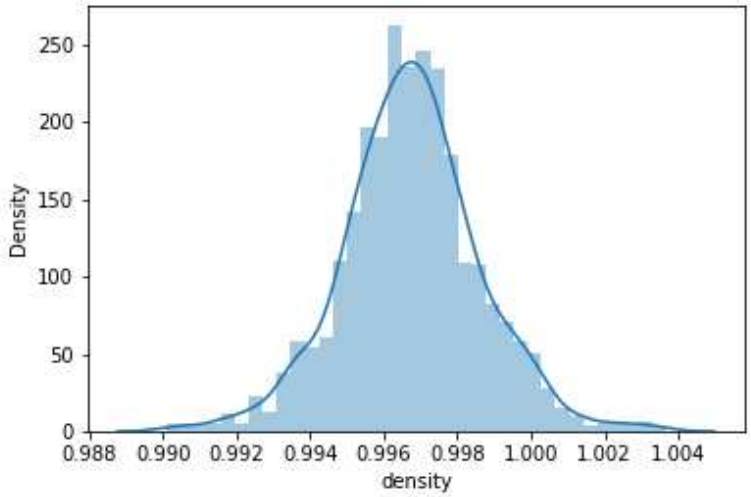
- Lưu huỳnh đioxit tự do (Free sulfur dioxide)



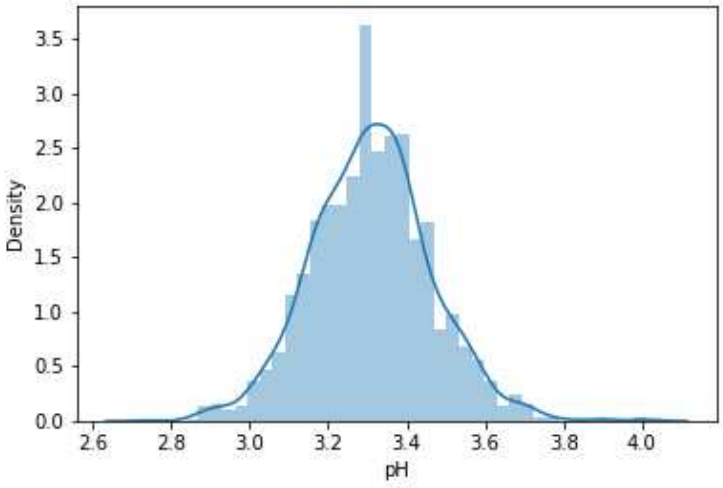
- Tổng lượng lưu huỳnh đioxit (total sulfur dioxide):



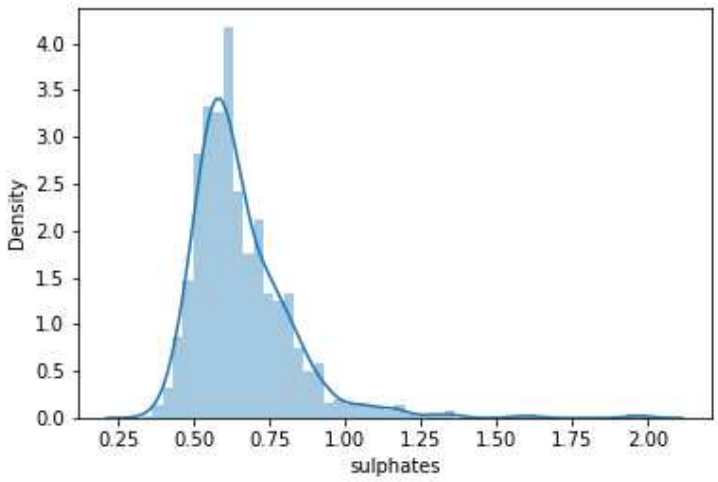
- Mật độ (density):



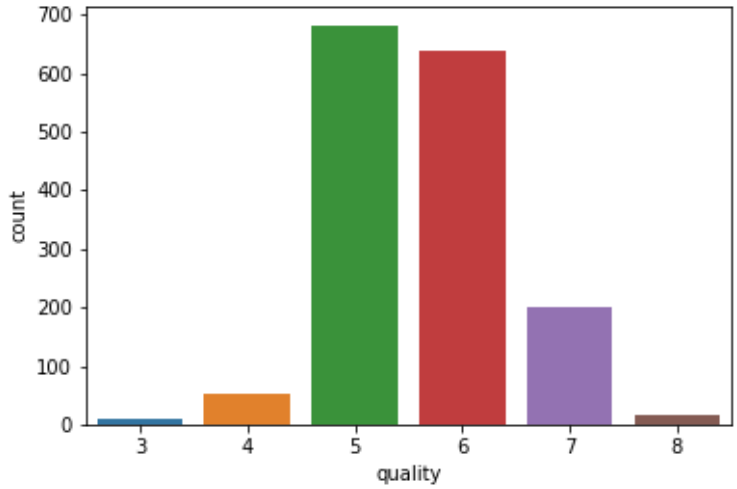
- Độ pH (pH):



- Muối sulfat (sulphates):



- Quality

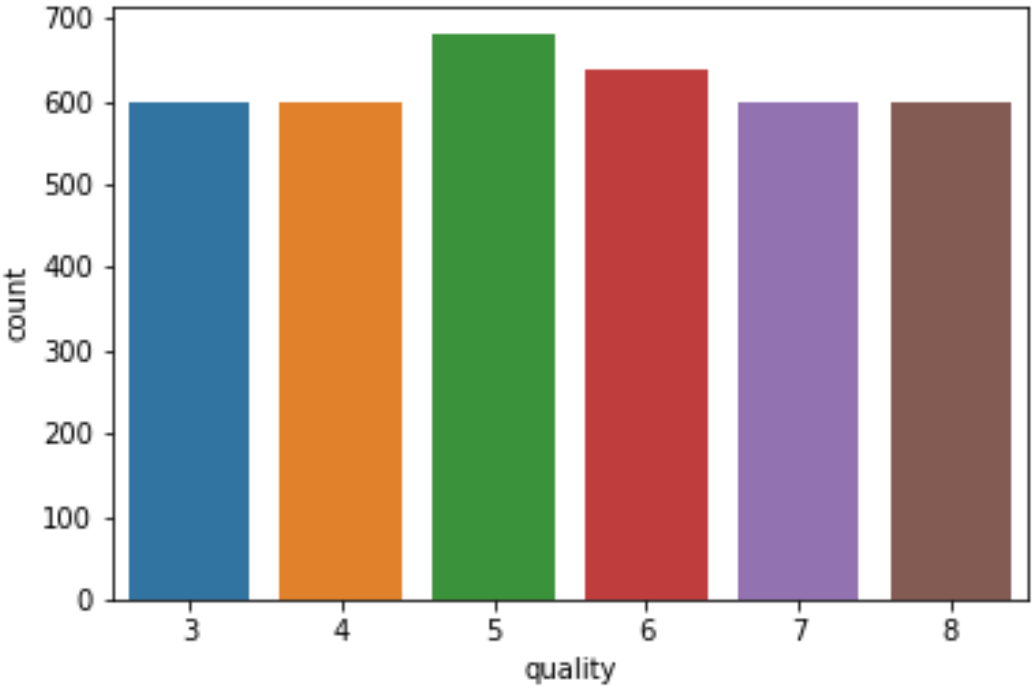


Nhận xét:

Theo biểu đồ dữ liệu ta thấy output đầu ra quality là không cân bằng chúng ta sẽ upsample các dữ liệu đầu ra có số ít. Đồng thời xử lý ouliers của các feature Clorua(chlorides), Đường dư (residual sugar), Lưu huỳnh đioxit tự do (Free sulfur dioxide), Tổng lượng lưu huỳnh đioxit (total sulfur dioxide) khi các ouliers rõ ràng và đáng kể nhất.

VI. THỰC NGHIỆM

Ở đây chúng ta sẽ dùng thư viện sklearn để upsample các output là số ít.



Đầu ra quality sau khi được resample

Tiếp đó xử lý ouliers. Ta xử lý bằng cái phương pháp tham khảo được trên Kaggle [13]. Và đánh giá bằng k-fold cross-validation với k-fold =5 trên decision tree regressor với cái siêu tham số là mặc định.

Phương pháp	MSE
Không xử lý outliers	0.185
Log transformation	0.188
Scalling	0.202
Cube Root Normalization	0.185
Box-Cox transformation	0.191
Median imputation	0.192
mean imputation	0.189
Zero value imputation	0.176

Nhận xét 1:

Các phương pháp xử lý outliers tỏ ra không hiệu quả . Decision tree regressor không bị ảnh hưởng nhiều bởi các outliers, đây là một trong những ưu điểm mà nhóm em đã nói đến trong phần trước. Nhóm em quyết định sẽ không xử lý outliers.

❖ **Tunning:**

Không sử dụng gridsearch CV, chỉ sử dụng những siêu tham số là mặc định:

0.185

Có gridsearch CV:

0.174

Best parameter:

```
{       'criterion': 'friedman_mse',
        'max_depth': None,
        'max_leaf_nodes': None,
        'min_impurity_decrease': 0.05,
        'min_samples_leaf': 1,
        'min_samples_split': 4          }
```

Xem toàn bộ source code trên Colab tại đây.

Nhận xét 2:

Hyperparameter tunning trên decision tree regression của tập dữ liệu này là không đáng. Mean square error chỉ sụt khoảng 0.01 với thời gian tunning là hơn 4 tiếng đồng hồ. Và nhóm em nhận ra là với mỗi lần fit, decision tree regresion lại trả về kết quả khác nhau dẫn tới việc sẽ khó tunning trên decison tree regression.

❖ **So sánh kết quả với các mô hình khác:**

Ở đây chúng ta sẽ sử dụng cross-validation với số fold là 5. Kết quả là mean square error trung bình của 5 fold trên val set với các siêu tham số của model là mặc định.

Model	MSE
SVM (kerenl=’rbf’)	2.633
SVM (kerenl=’poly’)	6.139
SVM (kerenl=’linear’)	0.966
Random forest regression	0.106
Naives Bayes Regression	1.031
KNN Regression	0.298
ANN (1 hidden layer với 32 hidden nodes)	0.832
Decision tree regression	0.179

Nhận xét 3:

Random forest , Decision tree và KNN cho thấy độ thích hợp với dữ liệu này. Tuy nhiên Random forest vẫn cho thấy sự vượt trội hơn hẳn so với Decision tree.

VII. TÀI LIỆU THAM KHẢO

❖ **Tài liệu tiếng Việt:**

1. Wikipedia, Cây quyết định, 09/12/2021, từ https://vi.wikipedia.org/wiki/Cây_quyết_định
2. Bigdata Solution, THUẬT TOÁN CÂY QUYẾT ĐỊNH (P.4): ƯU & KHUYẾT ĐIỂM, STOPPING & PRUNING METHOD, 09/12/2021, từ http://giaiphapbigdata.dvms.com.vn/ung-dung-bigdata/57-thuat-toan-cay-quyet-dinh-p-4-uu-khuyet-diem-stopping-pruning-method.html
3. ICHI.PRO, Cây quyết định: Giới thiệu đầy đủ, 09/12/2021, từ https://ichi.pro/vi/cay-quyet-dinh-gioi-thieu-day-du-106036283762742
4. Deep AI KhanhBlog, Mô hình cây quyết định (decision tree), 09/12/2021, từ https://phamdinhkhanh.github.io/deepai-book/ch_ml/DecisionTree.html
5. Nguyễn Thị Kim Anh, Nguyễn Phương Thanh Diệu (06/2014), MÔ HÌNH CÂY QUYẾT ĐỊNH DECISION TREE, 09/12/2021, từ https://123docz.net/document/2759951-tieu-luan-mon-he-ho-tro-quyet-dinh-mo-hinh-cay-quyet-dinh-decision-tree.htm
6. Nguyen Thi Hop, Decision Tree, 10/12/2021, từ https://viblo.asia/p/decision-tree-Do754bbBZM6#_4-ket-luan-7

❖ **Tài liệu tiếng Anh:**

7. z_ai, Decision Trees Explained, 10/12/2021, từ https://towardsdatascience.com/decision-trees-explained-3ec41632ceb6
8. Scikit-learn developers, sklearn.tree.DecisionTreeRegressor, 10/12/2021, từ https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
9. AnkanDas22, Python | Decision Tree Regression using sklearn, 11/12/2021, từ https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/
10. GEORGIOS DRAKOS, Decision Tree Regressor explained in depth, 11/12/2021, từ https://gdcoder.com/decision-tree-regressor-explained-in-depth/
11. Mohtadi Ben Fraj, InDepth: Parameter tuning for Decision Tree, 11/12/2021, từ https://medium.com/@mohtedibf/indepth-parameter-tuning-for-decision-tree-6753118a03c3
12. Danny Varghese, Comparative Study on Classic Machine learning Algorithms, 11/12/2021, từ https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222
13. Naresh Bhat, Outlier!!! The Silent Killer (2021), 12/12/2021, từ https://www.kaggle.com/nareshbhat/outlier-the-silent-killer