

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN**



TIỂU LUẬN

Học phần: Học máy (Machine Learning)

**Đề tài: Nghiên cứu và cài đặt chương trình phần mềm sử dụng kỹ thuật
DecisionTreeCART để dự đoán chất lượng của các loại rượu. Đánh giá hiệu quả
của mô hình phân lớp**

Nhóm sinh viên	:	Nhóm 4
Lớp học phần	:	01
Giảng viên hướng dẫn	:	TS. Lưu Minh Tuấn

Hà Nội, Năm 2024

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI	4
1.1. Tính cấp thiết của đề tài	4
1.2. Mục tiêu của đề tài	4
1.3. Đối tượng và phạm vi nghiên cứu	4
1.4. Phương pháp nghiên cứu	5
CHƯƠNG 2: CÁC CƠ SỞ LÝ THUYẾT	6
2.1. Giới thiệu về cây quyết định	6
2.1.2. Thuật toán cây quyết định phân lớp (Decision Tree Classification Algorithm)	7
2.1.3. Ưu và nhược điểm của cây quyết định	8
2.1.4. Thuật toán Cây quyết định hoạt động như thế nào?	10
2.2. Thuật toán CART (Classification and Regression Trees)	10
2.2.1. Ý tưởng	11
2.2.2. Độ thuần nhất Gini (Gini impurity)	11
2.2.3. Điều kiện dừng	12
2.2.4. Pruning	13
CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM	15
3.1. Bài toán	15
a. Dataset	15
b. Ngôn ngữ và những thư viện cần thiết	16
3.2. Phân tích dữ liệu khám phá ban đầu (EDA) và tiền xử lý dữ liệu	17
a. Feature Engineering	20
b. Chọn đặc trưng	21
3.3. Chia dữ liệu	21
3.4. Xây dựng mô hình cơ bản	23
3.5. Kỹ thuật DecisionTree Pruning	24
3.6. Điều chỉnh siêu tham số: FINE TUNNING VỚI RandomSearchCV	24
3.7. Model tối ưu	25
3.8. Nhận xét và Đánh giá	25
3.9. Những features đặc trưng nhất	26
3.10. Màn hình chức năng sử dụng thư viện Streamlit & Joblib	27

DANH SÁCH THÀNH VIÊN NHÓM 4

STT	Họ và tên	Mã sinh viên	Vai trò
1	Nguyễn Ngọc Long	11223918	Nhóm trưởng
2	Nguyễn Trần Trung	11226625	Thành viên
3	Nguyễn Khánh Linh	11223556	Thành viên
4	Vũ Thị Xuân Hương	11222736	Thành viên
5	Nguyễn Thị Khánh Huyền	11222930	Thành viên
6	Trần Thị Huyền Anh	11220632	Thành viên

LỜI MỞ ĐẦU

Học máy là một lĩnh vực thuộc trí tuệ nhân tạo (AI), sử dụng các thuật toán được huấn luyện để tìm kiếm các mẫu và mối tương quan trong các tập dữ liệu lớn, từ đó đưa ra những quyết định và dự đoán tối ưu dựa trên quá trình phân tích. Trong thực tế, Học máy hiện đang được ứng dụng rộng rãi trong nhiều lĩnh vực như khai thác dữ liệu, chẩn đoán y tế, phát hiện gian lận thẻ tín dụng, phân tích thị trường chứng khoán, phân loại chuỗi DNA, nhận dạng giọng nói và chữ viết, dịch tự động, v.v. Tuy nhiên, để tránh sai lệch và đảm bảo tính chính xác, quá trình phân tích dữ liệu trong Học máy vẫn cần sự can thiệp và lựa chọn kỹ thuật từ con người.

Nhận thấy tiềm năng rộng lớn của Học máy, chúng em quyết định nghiên cứu về việc dự đoán chất lượng các loại rượu thông qua việc áp dụng các phương pháp học máy. Trong phạm vi bài tiểu luận này, chúng em tập trung nghiên cứu và triển khai chương trình phần mềm sử dụng kỹ thuật cây quyết định CART để dự đoán chất lượng rượu dựa trên các biến số liên quan.

Chúng em xin chân thành cảm ơn!

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1.1. Tính cấp thiết của đề tài

Chất lượng của rượu vang đóng vai trò quan trọng đối với cả người tiêu dùng lẫn ngành công nghiệp rượu. Phương pháp truyền thống để đánh giá chất lượng rượu thông qua các chuyên gia thường tốn nhiều thời gian và công sức. Hiện nay, các mô hình học máy đã trở thành công cụ quan trọng, thay thế một phần công việc của con người. Tuy nhiên, không phải tất cả các đặc điểm của rượu đều liên quan đến việc dự đoán chất lượng tốt hơn, mà chỉ có một số tính năng chính là cần thiết.

Khi đánh giá chất lượng của một chai rượu, mỗi người thường có quan điểm khác nhau. Đối với nhiều nhà phê bình rượu, chất lượng thường liên quan đến việc phân biệt rượu "ngon" hay "dở", dựa trên sở thích cá nhân và sự hài lòng hoặc không hài lòng về rượu. Đánh giá này thường được đặt trong bối cảnh các tiêu chuẩn đã được thiết lập cho từng loại rượu, do vậy chất lượng không chỉ mang tính chủ quan mà còn phụ thuộc vào cả yếu tố cảm quan bên trong (mùi vị) và yếu tố bên ngoài (ngữ cảnh).

Tuy nhiên, con người chúng ta chủ yếu bị ảnh hưởng bởi thị giác, do đó, hình dáng chai rượu thường ảnh hưởng đến cách chúng ta cảm nhận hương thơm và vị giác của rượu. Vì thế, để tránh thiên vị, những người thực sự muốn đánh giá rượu một cách khách quan thường ném thử rượu mà không biết về nguồn gốc của nó, sử dụng các ly ném rượu tiêu chuẩn hóa màu đen (ISO), chỉ đánh giá chất lượng dựa trên những gì rượu truyền tải đến khứu giác và vị giác.

Vậy làm thế nào để đánh giá được một lượng lớn rượu về chất lượng tốt hay xấu? Bằng cách áp dụng học máy, chúng ta có thể sử dụng phương pháp sàng lọc, hay còn gọi là kỹ thuật phân lớp, để so sánh các mẫu rượu chất lượng cao với những mẫu khác, từ đó tìm ra các mẫu có chất lượng tương đồng nhất.

1.2. Mục tiêu của đề tài

Sử dụng kỹ thuật cây quyết định CART để phân tích và dự đoán chất lượng của các loại rượu dựa trên bộ dữ liệu **winequality-red.csv**, sử dụng các biến số có sẵn trong tập dữ liệu để dự đoán nhãn chất lượng.

Thực hiện phân lớp dựa trên mô hình CART và sử dụng tập test để kiểm tra độ chính xác, từ đó đánh giá hiệu quả của mô hình dựa trên các chỉ số như độ chính xác, độ nhạy và các tiêu chí khác trong quá trình dự đoán.

1.3. Đối tượng và phạm vi nghiên cứu

- Đối tượng: kỹ thuật cây quyết định CART, bài toán tối ưu hóa tổ hợp.
- Phạm vi nghiên cứu: dự trên bộ dữ liệu **winequality-red.csv**

1.4. Phương pháp nghiên cứu

Trong tiểu luận này, nhóm sử dụng phương pháp nghiên cứu thực nghiệm để thu thập bằng chứng có thể kiểm chứng để đi đến kết quả nghiên cứu.

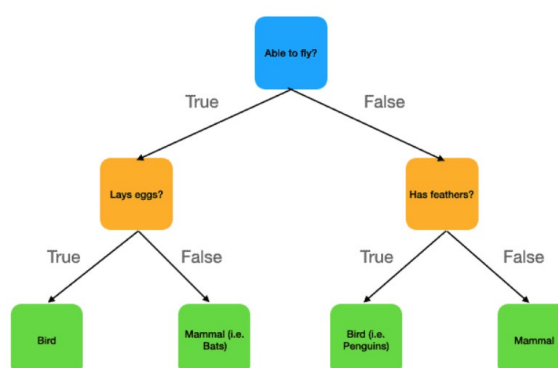
CHƯƠNG 2: CÁC CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về cây quyết định:

Trong lĩnh vực học máy, cây quyết định là một mô hình dự báo, nghĩa là nó ánh xạ từ các quan sát về một sự vật hoặc hiện tượng đến các kết luận về giá trị mục tiêu của sự vật, hiện tượng đó. Mỗi nút bên trong của cây tương ứng với một biến, và các nhánh nối giữa các nút biểu thị giá trị cụ thể của biến đó. Mỗi nút lá đại diện cho giá trị dự đoán của biến mục tiêu, dựa trên các giá trị của các biến được xác định từ đường đi từ nút gốc đến nút lá. Phương pháp học máy được sử dụng trong cây quyết định gọi là học bằng cây quyết định.

Cây quyết định là một công cụ mạnh mẽ trong việc phân loại và dự đoán. Sự hấp dẫn của phương pháp này nằm ở khả năng xử lý nhiều loại sự kiện, và khác với mạng nơ-ron, cây quyết định thể hiện các quy tắc, những quy tắc này dễ hiểu đối với con người. Cây quyết định có nhiều ứng dụng, ví dụ như trong hệ thống thư tín của công ty, một mô hình có thể dự đoán chính xác thành viên nào trong nhóm sẽ phản hồi một yêu cầu mà không cần biết chi tiết cách thức hoạt động của mô hình. Trong một số trường hợp khác, cây quyết định có thể giải thích lý do của một quyết định, điều này rất quan trọng trong các ứng dụng yêu cầu sự phân loại hay dự đoán chính xác.

2.1.1. Ví dụ về cây quyết định:



Thuật toán cây quyết định trong hình ảnh cung cấp một ví dụ rõ ràng về cách phân loại một loài động vật thành hai nhóm: "Chim" hoặc "Thú", dựa trên một loạt câu hỏi về đặc điểm sinh học.

Câu hỏi đầu tiên tại nút gốc là "Động vật có thể bay không?" Đây là điểm phân nhánh đầu tiên. Nếu câu trả lời là "có", thuật toán tiếp tục hỏi "Động vật này có đẻ trứng không?" Nếu đúng, động vật đó được phân loại là Chim. Nếu không, nó sẽ được phân loại là Thú (ví dụ như dơi). Ngược lại, nếu động vật không thể bay, thuật toán sẽ chuyển sang câu hỏi tiếp theo: "Động vật này có lông vũ không?" Nếu đúng, động vật đó vẫn là Chim (ví dụ như chim cánh cụt, không bay nhưng có lông vũ). Nếu không, động vật đó sẽ được phân loại là Thú.

Thuật toán cây quyết định này sử dụng một chuỗi các câu hỏi phân cấp để phân loại đối tượng dựa trên các đặc điểm rõ ràng và cụ thể. Điều này cho thấy cây quyết định là một công cụ trực quan và hiệu quả trong việc phân loại các đối tượng trong nhiều bài toán học máy và dữ liệu thực tế, nhờ vào việc chia nhỏ dữ liệu thành các quyết định dựa trên các thuộc tính riêng lẻ.

2.1.2. Thuật toán cây quyết định phân lớp (Decision Tree Classification Algorithm):

Thuật toán cây quyết định là một kỹ thuật học có giám sát, được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Nó thuộc nhóm các mô hình dự đoán có khả năng học hỏi từ dữ liệu đã có nhãn để đưa ra dự đoán cho các dữ liệu mới. Thuật toán cây quyết định đặc biệt hữu ích khi cần giải thích kết quả vì nó chia nhỏ các vấn đề phức tạp thành các quyết định nhỏ hơn và rõ ràng. Điều này giúp cho cây quyết định trở thành một trong những thuật toán dễ hiểu và dễ triển khai trong thực tế.

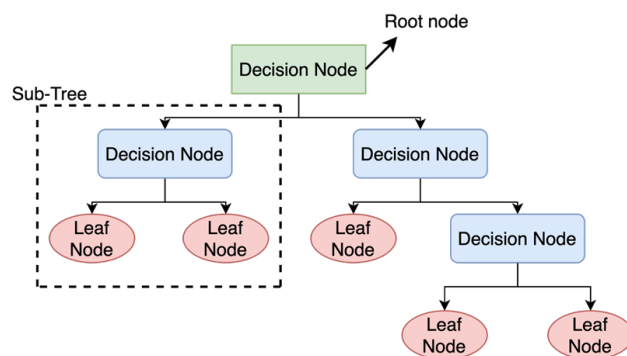
Cấu trúc của cây quyết định bao gồm ba thành phần chính: nút gốc, các nút bên trong (nút quyết định), và các nút lá. Nút gốc là nơi bắt đầu quá trình phân loại hoặc dự đoán, thường là dựa trên thuộc tính quan trọng nhất trong tập dữ liệu. Các nút bên trong đại diện cho các câu hỏi hoặc các phép kiểm tra liên quan đến các thuộc tính của dữ liệu, và mỗi nhánh từ một nút bên trong tương ứng với một giá trị hoặc một kết quả của phép kiểm tra đó. Cuối cùng, các nút lá đại diện cho các kết quả cuối cùng, tức là phân loại hoặc dự đoán của thuật toán.

Cách hoạt động của thuật toán cây quyết định bắt đầu từ nút gốc, tại đây thuật toán chọn thuộc tính tốt nhất để phân chia dữ liệu dựa trên một số tiêu chí như Gini impurity, entropy (trong thuật toán ID3), hoặc information gain (lợi ích thông tin). Dữ

liệu sẽ được phân chia thành các tập con dựa trên giá trị của thuộc tính đó, và quá trình này tiếp tục đệ quy tại mỗi nút con cho đến khi tất cả các tập con đều thuần nhất hoặc đạt đến điều kiện dừng.

Thuật toán này được gọi là cây quyết định bởi vì quá trình phân chia dữ liệu diễn ra tương tự như cách các nhánh cây phân nhánh từ một thân cây chính. Các quyết định được đưa ra liên tiếp tại các nút nội bộ, và kết quả cuối cùng xuất hiện ở các nút lá, tương tự như cách các nhánh cây kết thúc bằng các lá. Sự đơn giản trong cấu trúc và khả năng chia nhỏ các vấn đề phức tạp thành các quyết định tuần tự là yếu tố làm nên sức mạnh của cây quyết định.

Sơ đồ dưới đây giải thích cấu trúc chung của cây quyết định:



2.1.3. Ưu và nhược điểm của cây quyết định

- *Ưu điểm của cây quyết định:*
 - Clear Visualization: Thuật toán đơn giản để hiểu, diễn giải và trực quan vì ý tưởng chủ yếu được sử dụng trong cuộc sống hàng ngày của chúng ta. Đầu ra của cây quyết định có thể được con người giải thích một cách dễ dàng.
 - Đơn giản và dễ hiểu: Cây quyết định trông giống như các câu lệnh if-else đơn giản và rất dễ hiểu.
 - Cây quyết định có thể được sử dụng cho cả bài toán phân loại và bài toán hồi quy.
 - Cây quyết định có thể xử lý cả biến liên tục và biến phân loại.
 - Không yêu cầu feature scaling: Không yêu cầu feature scaling (standardization và normalization) trong trường hợp cây quyết định vì nó sử dụng phương pháp tiếp cận dựa trên quy tắc thay vì tính toán.

- Xử lý các tham số phi tuyến tính một cách hiệu quả: Các tham số phi tuyến tính không ảnh hưởng đến hiệu suất của cây quyết định không giống như các thuật toán dựa trên đường cong. Vì vậy, nếu có sự không tuyến tính cao giữa các biến độc lập, cây quyết định có thể hoạt động tốt hơn so với các thuật toán dựa trên đường cong khác.
- Cây quyết định có thể tự động xử lý các giá trị bị thiếu.
- Cây quyết định thường mạnh mẽ đối với các trường hợp ngoại lệ và có thể xử lý chúng một cách tự động.
- Thời gian đào tạo ít hơn: Thời gian đào tạo ít hơn so với rừng ngẫu nhiên (Random Forest) vì nó chỉ tạo ra một cây không giống như rừng của các cây trong Random Forest.
- *Nhược điểm của Cây quyết định:*
 - Overfitting: Đây là vấn đề chính của cây quyết định. Nó thường dẫn đến việc quá khớp dữ liệu mà cuối cùng dẫn đến dự đoán sai. Để fit với dữ liệu (ngay cả dữ liệu nhiễu), nó tiếp tục tạo ra các nút mới và cuối cùng cây trở nên quá phức tạp để diễn giải, dẫn đến mất khả năng tổng quát hóa. Nó hoạt động rất tốt trên dữ liệu được đào tạo nhưng bắt đầu mắc nhiều lỗi trên dữ liệu không nhìn thấy được.
 - Phương sai cao: Như đã đề cập ở điểm 1, cây quyết định thường dẫn đến việc quá khớp dữ liệu. Chính vì thế, có rất nhiều khả năng sai lệch cao trong đầu ra, dẫn đến nhiều sai sót trong ước tính cuối cùng và cho thấy kết quả không chính xác cao. Quá khớp dẫn đến phương sai cao.
 - Không ổn định: Việc thêm một điểm dữ liệu mới có thể dẫn đến việc tạo lại cây tổng thể và tất cả các nút cần được tính toán lại và tạo lại.
 - Bị ảnh hưởng bởi nhiễu: Một chút dữ liệu nhiễu có thể làm cho nó không ổn định, dẫn đến dự đoán sai.
 - Không phù hợp với tập dữ liệu lớn: Nếu kích thước dữ liệu lớn, thì một cây đơn lẻ có thể phát triển phức tạp và dẫn đến quá khớp. Vì vậy, trong trường hợp này, chúng ta nên sử dụng Random Forest thay vì một cây quyết định. Để khắc phục những hạn chế của cây quyết định, chúng ta nên sử dụng Random Forest để không dựa vào một cây nào. Nó tạo ra một rừng cây và đưa ra quyết định dựa trên số phiếu bầu.

2.1.4. Thuật toán Cây quyết định hoạt động như thế nào?

Trong cây quyết định, để dự đoán lớp của tập dữ liệu đã cho, thuật toán bắt đầu từ nút gốc của cây. Thuật toán này so sánh các giá trị của thuộc tính gốc với thuộc tính bản ghi (tập dữ liệu thực) và dựa trên sự so sánh, đi theo nhánh và nhảy đến nút tiếp theo.

Đối với nút tiếp theo, thuật toán lại so sánh giá trị thuộc tính với các nút con khác và di chuyển xa hơn. Nó tiếp tục quá trình cho đến khi nó đạt đến nút lá của cây. Quy trình hoàn chỉnh có thể được hiểu rõ hơn bằng cách sử dụng thuật toán dưới đây:

Bước 1: Bắt đầu xây dựng cây quyết định với nút gốc (S), nơi chứa toàn bộ tập dữ liệu ban đầu. Nút gốc này là điểm khởi đầu để phân chia dữ liệu dựa trên các thuộc tính.

Bước 2: Tìm thuộc tính tốt nhất trong tập dữ liệu bằng cách sử dụng các phép đo lựa chọn thuộc tính (Attribute Selection Measure - ASM), chẳng hạn như entropy, Gini impurity, hoặc information gain. Đây là bước quan trọng để xác định thuộc tính có khả năng phân loại dữ liệu tốt nhất.

Bước 3: Dựa trên thuộc tính tốt nhất được chọn, chia tập dữ liệu S thành các tập con nhỏ hơn. Mỗi tập con sẽ chứa các phần tử tương ứng với các giá trị khác nhau của thuộc tính đó.

Bước 4: Tạo một nút quyết định mới đại diện cho thuộc tính tốt nhất vừa chọn. Nút này sẽ có các nhánh dẫn tới các tập con đã được chia nhỏ ở bước trước.

Bước 5: Đệ quy lặp lại quá trình này đối với các tập con vừa tạo ra. Thuật toán tiếp tục chọn thuộc tính tốt nhất cho mỗi tập con và phân chia tiếp tục cho đến khi không thể chia nhỏ thêm nữa, tại đó các nút cuối cùng sẽ trở thành nút lá, đại diện cho phân loại cuối cùng hoặc kết quả dự đoán.

2.2. Thuật toán CART (Classification and Regression Trees).

Thuật toán CART (cây phân loại và hồi quy - Classification and Regression Tree), là một thuật toán được Leo Breiman giới thiệu năm 1984, dùng để xây dựng cây quyết định đa năng, với mục đích phân loại và hồi quy dữ liệu.

2.2.1. Ý tưởng

Ý tưởng chính của thuật toán CART là xây dựng một cây quyết định bằng cách phân chia dữ liệu thành các nhóm con dựa trên thuộc tính tốt nhất tại mỗi nút. Thuật toán này sử dụng các tiêu chí đánh giá như Gini impurity (cho phân loại) và mean squared error (MSE) (cho hồi quy) để đảm bảo rằng các nhóm con được phân chia có tính thuần nhất cao, từ đó đưa ra các quyết định hoặc dự đoán chính xác hơn.

So với các thuật toán cây quyết định khác như ID3 và C4.5, CART có một số khác biệt rõ rệt. Thứ nhất, CART sử dụng Gini impurity và MSE làm chỉ số đo lường để chọn thuộc tính tốt nhất, trong khi ID3 và C4.5 sử dụng entropy và information gain để đánh giá thuộc tính. Ngoài ra, CART luôn thực hiện phân chia nhị phân, ngay cả khi thuộc tính có nhiều giá trị khác nhau, trong khi ID3 và C4.5 cho phép phân chia thành nhiều nhánh dựa trên tất cả các giá trị của thuộc tính đó.

Cuối cùng, CART có cơ chế cắt tỉa cây (pruning) sau khi cây đã được xây dựng để tránh overfitting, bằng cách loại bỏ các nhánh không cần thiết. C4.5 cũng có tính năng cắt tỉa cây, nhưng ID3 không thực hiện việc này một cách rõ ràng, điều này có thể dẫn đến cây phân loại phức tạp hơn và dễ bị overfitting.

2.2.2. Độ thuần nhất Gini (Gini impurity)

Độ thuần nhất Gini (Gini Impurity) là một chỉ số dùng để đo lường mức độ "thuần nhất" của một tập dữ liệu, hay nói cách khác, nó đánh giá mức độ hỗn loạn hoặc không đồng nhất của các nhãn trong tập dữ liệu tại một nút của cây quyết định. Giá trị của độ thuần nhất Gini dao động từ 0 đến 1:

- Gini = 0: Tập dữ liệu hoàn toàn thuần nhất, tức là tất cả các phần tử trong tập đều thuộc cùng một lớp.
- Gini = 1: Tập dữ liệu hoàn toàn hỗn loạn, tức là các phần tử được phân phối đồng đều giữa các lớp khác nhau.

Công thức tính Gini Impurity cho một tập dữ liệu chứa nhiều lớp như sau: $Gini = 1 - \sum_{i=1}^n p_i^2$. Trong đó:

- p_i là tỷ lệ của các phần tử thuộc lớp i trong tập dữ liệu.

- n là số lớp trong tập dữ liệu.

Cách cây quyết định sử dụng Gini Impurity để phân chia nhánh:

1. Tính Gini Impurity tại nút hiện tại: Ban đầu, thuật toán sẽ tính độ thuần nhất Gini cho tập dữ liệu tại nút gốc, xác định mức độ hỗn loạn của dữ liệu trước khi phân chia.
2. Phân chia dữ liệu dựa trên thuộc tính: Cây quyết định thử phân chia dữ liệu dựa trên các thuộc tính khác nhau. Với mỗi thuộc tính, thuật toán chia tập dữ liệu thành hai tập con dựa trên giá trị của thuộc tính đó (cây CART luôn chia thành hai nhánh nhị phân). Sau đó, độ thuần nhất Gini của các tập con này sẽ được tính toán.
3. Tính Gini Impurity trung bình có trọng số: Độ thuần nhất Gini của mỗi tập con sau khi phân chia sẽ được tính toán và trung bình có trọng số của Gini Impurity của các tập con cũng được xác định. Trọng số được tính dựa trên kích thước của mỗi tập con so với tổng dữ liệu ban đầu.
4. Chọn thuộc tính có Gini Impurity thấp nhất: Thuộc tính nào làm giảm Gini Impurity nhiều nhất (tức là có sự cải thiện tốt nhất về mức độ thuần nhất sau khi phân chia) sẽ được chọn để chia nhánh tại nút đó. Mục tiêu là chọn thuộc tính làm cho các tập con sau khi chia có mức độ thuần nhất cao nhất (tức là Gini Impurity thấp nhất).
5. Lặp lại quá trình: Sau khi đã chọn thuộc tính và phân chia dữ liệu, cây quyết định tiếp tục áp dụng quá trình này một cách đệ quy cho các tập con cho đến khi đạt điều kiện dừng (tập dữ liệu hoàn toàn thuần nhất hoặc không còn thuộc tính nào để chia).

2.2.3. Điều kiện dừng

Trong các thuật toán cây quyết định nói chung và CART nói riêng, nếu ta tiếp tục phân chia các nút chưa thuần nhất, kết quả là một cây quyết định mà mọi điểm trong tập huấn luyện đều được dự đoán chính xác (giả sử không có trường hợp hai input giống nhau nhưng cho output khác nhau). Tuy nhiên, cây sẽ trở nên rất phức tạp với nhiều nút lá và chỉ chứa một vài điểm dữ liệu tại mỗi nút. Điều này làm tăng nguy cơ xảy ra overfitting.

Để tránh tình trạng overfitting, một số phương pháp có thể được áp dụng.

- Ngưỡng tối thiểu số lượng mẫu: Dừng phân chia khi số lượng mẫu trong một tập con nhỏ hơn một giá trị ngưỡng nhất định, giúp tránh việc tạo ra các nhánh quá nhỏ và không có ý nghĩa.
- Độ thuần nhất cao: Nếu tất cả các mẫu trong một tập con đều thuộc cùng một lớp (phân loại) hoặc có giá trị rất tương đồng (hồi quy), thuật toán sẽ dừng phân chia.
- Không còn thuộc tính để phân chia: Khi tất cả các thuộc tính đã được sử dụng hết, thuật toán không thể tiếp tục phân chia thêm và dừng tại đó.
- Cải thiện lỗi không đáng kể: Dừng phân chia nếu việc tiếp tục không làm giảm đáng kể các chỉ số như Gini impurity (phân loại) hoặc mean squared error (hồi quy).
- Giới hạn độ sâu của cây: Cây sẽ dừng phát triển khi đạt đến độ sâu tối đa do người dùng đặt trước, giúp kiểm soát độ phức tạp của cây.

2.2.4. Pruning

Cắt tỉa (pruning) trong thuật toán CART (Classification and Regression Trees) là một kỹ thuật quan trọng giúp cải thiện hiệu quả của cây quyết định và tránh hiện tượng quá khớp (overfitting). Khi xây dựng cây quyết định, nếu cây quá phức tạp và có quá nhiều nút hoặc nhánh, mô hình có thể khớp quá sát với dữ liệu huấn luyện, làm giảm khả năng tổng quát hóa của nó đối với dữ liệu mới. Cắt tỉa giúp giảm kích thước của cây, giữ lại những phần quan trọng nhất và loại bỏ những nhánh ít hữu ích.

- Hai loại cắt tỉa trong thuật toán CART:
 - Cắt tỉa trước (Pre-pruning): Quá trình cắt tỉa này diễn ra trong khi cây đang được xây dựng. Thuật toán sẽ ngừng phân chia thêm tại một nút khi đạt đến một số điều kiện nhất định, chẳng hạn như khi số lượng mẫu trong một nhánh nhỏ hơn một ngưỡng cụ thể, hoặc khi mức độ cải thiện của việc phân chia là không đáng kể. Điều này giúp hạn chế sự phát triển của cây từ sớm, ngăn chặn việc tạo ra quá nhiều nhánh không cần thiết.
 - Cắt tỉa sau (Post-pruning): Đây là quá trình cắt tỉa sau khi cây đã được xây dựng hoàn chỉnh. Thuật toán sẽ loại bỏ các nhánh không mang lại nhiều thông tin, tức là những nhánh mà việc phân chia không cải thiện đáng kể độ

chính xác của mô hình. Mục tiêu là làm đơn giản hóa cấu trúc cây mà không làm giảm quá nhiều hiệu suất.

- Tiêu chí cắt tỉa

- Giá trị lỗi của cây (tree error): Thuật toán sẽ tính toán lỗi của cây với và không có nhánh đó. Nếu việc loại bỏ một nhánh giúp giảm lỗi tổng thể hoặc không làm tăng lỗi đáng kể, nhánh đó sẽ bị loại bỏ.
- Lỗi cắt tỉa với cây quyết định CART: Trong cắt tỉa sau, thuật toán có thể sử dụng tiêu chí lỗi cắt tỉa (pruning error), chẳng hạn như dựa trên dữ liệu xác thực (validation set) hoặc phương pháp Cross-validation, để đánh giá hiệu quả của mỗi nhánh sau khi cắt tỉa.

- Quá trình cắt tỉa trong CART

Trong thuật toán CART, cắt tỉa thường được thực hiện bằng cách sử dụng phương pháp chi phí phức tạp cắt tỉa (Cost Complexity Pruning), một kỹ thuật cắt tỉa sau (post-pruning). Quá trình này kết hợp giữa chi phí lỗi và độ phức tạp của cây để loại bỏ các nhánh không cần thiết. Công thức để tính toán chi phí phức tạp là:

$$R\alpha(T) = R(T) + \alpha * |T|$$

Trong đó:

- $R(T)$ là lỗi của cây T.
- α là tham số điều chỉnh giữa độ phức tạp và lỗi.
- $|T|$ là số lượng nút trong cây.

Giá trị α được sử dụng để điều chỉnh mức độ mà độ phức tạp của cây được ưu tiên so với độ chính xác. Khi α tăng lên, cây sẽ ưu tiên đơn giản hóa hơn và nhiều nhánh có thể bị cắt bỏ.

CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM

3.1. Bài toán:

Nghiên cứu và cài đặt chương trình phần mềm sử dụng kỹ thuật cây quyết định CART để dự đoán chất lượng (quality) của các loại rượu theo các biến còn lại. Từ kết quả phân lớp và dữ liệu trong tập test, đánh giá hiệu quả của mô hình dự đoán.

- Đầu vào (Input): Tập dữ liệu của các loại chất hoá học có trong một loại rượu.
- Đầu ra (Output): Dự đoán chất lượng của loại rượu nêu trên.

a. Dataset:

- Các thuộc tính có trong dataset:
 1. fixed acidity: Hầu hết các axit liên quan đến rượu vang là hoặc cố định hoặc không bay hơi (không dễ bay hơi).
 2. volatile acidity: Lượng axit axetic trong rượu vang ở mức quá cao có thể khiến rượu có vị chua của giấm, rất khó chịu.
 3. citric acid: Với số lượng không đáng kể, axit citric có thể thêm 'độ tươi' và hương vị cho rượu vang.
 4. residual sugar: Lượng đường còn lại sau khi ngừng lên men, hiếm có loại rượu nào có chỉ số dưới 1 gam / lít.
 5. chlorides: Lượng muối trong rượu.
 6. free sulfur dioxide: Dạng tự do của SO₂ tồn tại ở trạng thái cân bằng giữa phân tử SO₂ (ở dạng khí hòa tan) và ion bisulfit
 7. total sulfur dioxide: Lượng các dạng liên kết và tự do của S₀₂, ở nồng độ thấp, hầu như không thể phát hiện được SO₂ trong rượu vang, nhưng ta có thể phát hiện SO₂ ở thể tự do.
 8. density: Tỷ trọng của nước gần với tỷ trọng của hỗn hợp nước phụ thuộc vào phần trăm độ cồn và lượng đường.

9. pH: Mô tả mức độ axit hoặc bazơ của một loại rượu trên thang điểm từ 0 (rất axit) đến 14 (rất kiềm), hầu hết các loại rượu đều nằm trong khoảng từ 3-4.
10. sulfates: Một chất phụ gia rượu vang có thể góp phần làm tăng nồng độ khí sulfur dioxide (SO₂), hoạt động như một chất kháng khuẩn.
11. alcohol: Nồng độ cồn của rượu.
12. quality: Kết quả đầu ra là chất lượng của rượu (dựa trên dữ liệu cảm quan, cho điểm từ 0 đến 10).

b. Ngôn ngữ và những thư viện cần thiết:

- Ngôn ngữ:

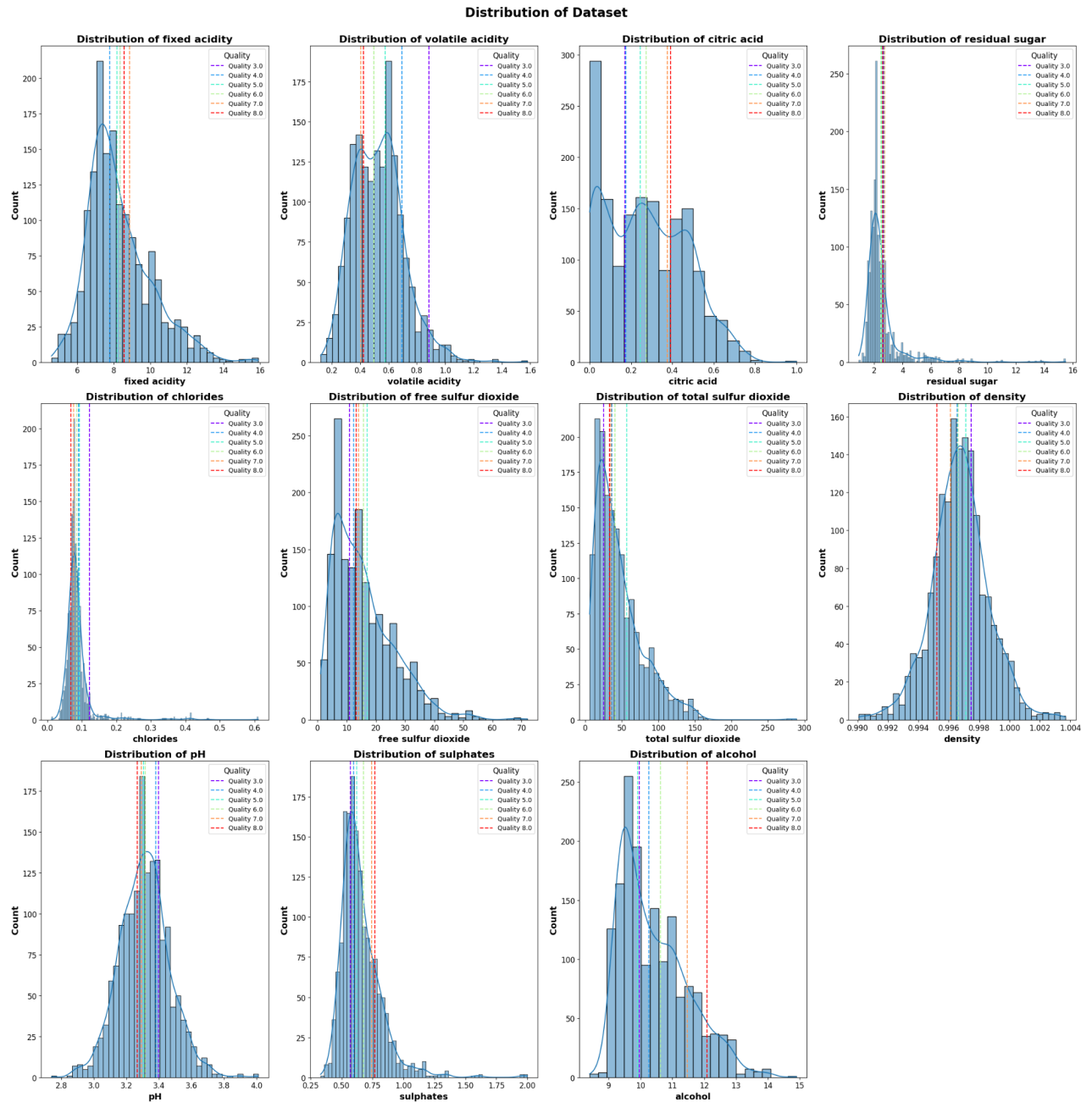
Python cho phép các lập trình viên thực hiện viết code ngắn, dễ đọc hơn trong khi các thuật toán phức tạp và quy trình làm việc linh hoạt của Machine Learning, AI

- Thư viện:

Những thư viện được sử dụng để giải quyết bài toán này bao gồm:

- numpy: thư viện phục vụ việc tính toán
- pandas: thư viện cung cấp các cấu trúc dữ liệu
- seaborn: thư viện mô hình hoá dữ liệu
- matplotlib: thư viện mô hình hoá dữ liệu
- scikit-learn (sklearn): thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction...
- imbalanced-learn (imblearn): thư viện chuyên xử lý các bộ dữ liệu mất cân bằng, nơi mà số lượng mẫu của các lớp không đồng đều.

3.2 . Phân tích dữ liệu khám phá ban đầu (EDA) và tiền xử lý dữ liệu:



Từ biểu đồ phân bố giá trị của các thuộc tính ta có thể thấy :

Các thuộc tính 'fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide' và 'density' có phân bố dữ liệu không đều, có nhiều giá trị nằm ở phía bên trái của biểu đồ.

→ Điều này có thể làm cho mô hình dễ bị nhiễu bởi các giá trị ngoại lai.

- **Fixed acidity:**

Phân phối lệch phải, với đỉnh chính ở khoảng 6-8. Rượu chất lượng cao (màu đỏ) có xu hướng có độ axit cố định cao hơn.

- **Volatile acidity:**

Phân phối khá cân đối, với đỉnh ở khoảng 0.3-0.4. Rượu chất lượng thấp có xu hướng có độ axit bay hơi cao hơn.

- **Citric acid:**

Phân phối đa phương thức với nhiều đỉnh. Rượu chất lượng cao có xu hướng có hàm lượng axit citric cao hơn.

- **Residual sugar:**

Phân phối lệch phải mạnh, với đa số mẫu có lượng đường dư thấp (dưới 5). Không có sự khác biệt rõ ràng giữa các mức chất lượng.

- **Chlorides:**

Phân phối lệch phải, tập trung chủ yếu ở khoảng 0.05-0.1. Rượu chất lượng cao có xu hướng có hàm lượng clorua thấp hơn.

- **Free sulfur dioxide và Total sulfur dioxide:**

Cả hai đều có phân phối lệch phải.

Rượu chất lượng cao có xu hướng có hàm lượng sulfur dioxide tự do và tổng số thấp hơn.

- **Density:**

Phân phối gần như chuẩn, tập trung quanh 0.995-1.000. Không có sự khác biệt rõ ràng giữa các mức chất lượng.

- **pH:**

Phân phối gần như chuẩn, tập trung quanh 3.0-3.4. Không có sự khác biệt rõ ràng giữa các mức chất lượng.

- **Sulphates:**

Phân phối lệch phải, với đỉnh ở khoảng 0.5. Rượu chất lượng cao có xu hướng có hàm lượng sunphat cao hơn.

- **Alcohol:**

Phân phối lệch phải nhẹ. Rượu chất lượng cao có xu hướng có nồng độ cồn cao hơn.

- **Quality:**

Phân phối gần như chuẩn, với đa số mẫu tập trung ở mức chất lượng trung bình (5-6). Ít mẫu ở mức chất lượng rất thấp hoặc rất cao.

- Tính không cân đối:

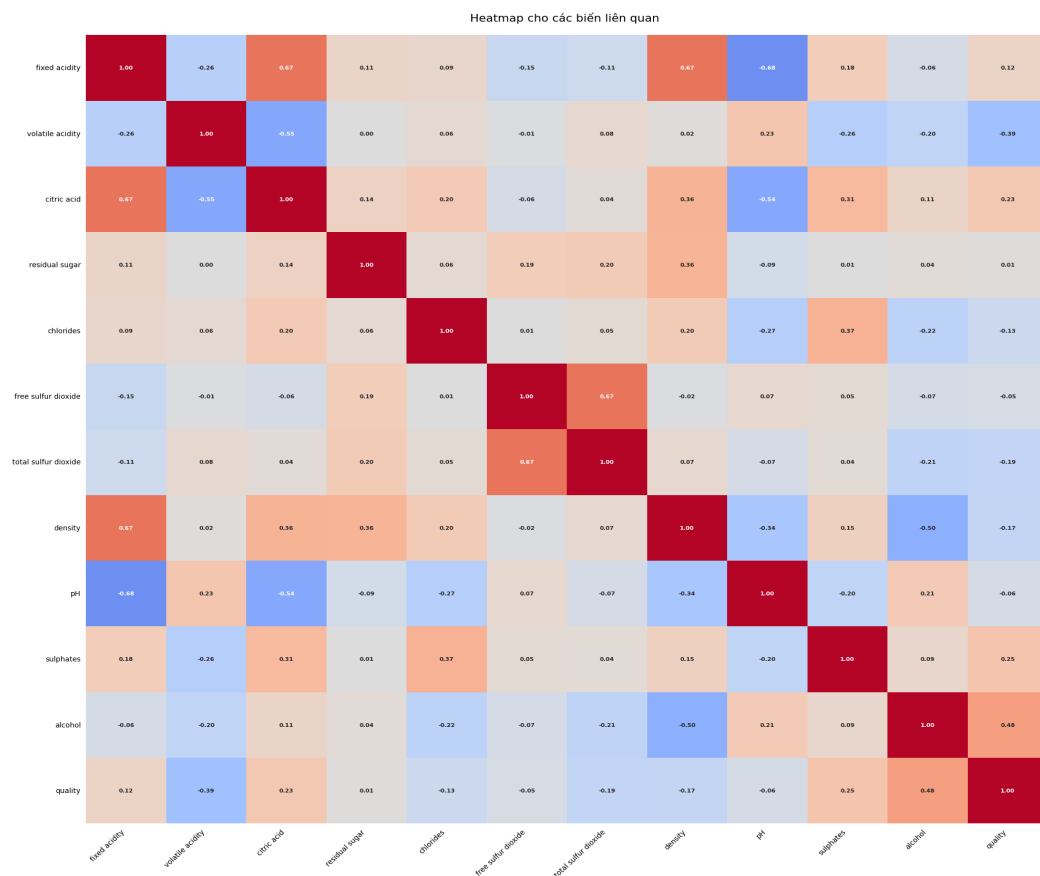
Phân phối hơi lệch phải, với đuôi dài hơn về phía chất lượng cao.

Kết luận:

- Nhiều đặc trưng có mối tương quan với chất lượng rượu, đặc biệt là volatile acidity, citric acid, chlorides, sulfur dioxide, sulphates và alcohol.
- Phần lớn các đặc trưng có phân phối lệch, cho thấy sự không đồng đều trong dữ liệu. → Ta tiến hành chia bins, áp dụng SMOTE và tạo ra các features mới để máy học tốt hơn

→ Bộ dữ liệu khá đồng nhất với 1599 mẫu cho mỗi đặc trưng, không có giá trị null.

→ Sự mất cân đối trong phân phối có thể ảnh hưởng đến việc xây dựng mô hình dự đoán.



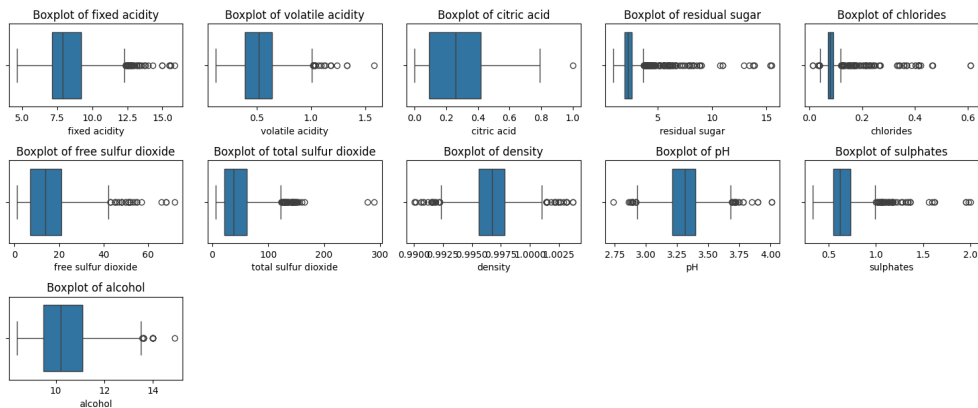
- Tương quan của các biến ảnh hưởng tới chất lượng rượu:

```
Correlation of Wine Quality":
quality                1.000000
alcohol                0.476166
volatile acidity       0.390558
sulphates              0.251397
citric acid            0.226373
total sulfur dioxide   0.185100
density                0.174919
chlorides              0.128907
fixed acidity          0.124052
pH                     0.057731
free sulfur dioxide    0.050656
residual sugar         0.013732
Name: quality, dtype: float64
```

Nhiều đặc trưng có mối tương quan với chất lượng rượu, đặc biệt là volatile acidity, citric acid, chlorides, sulfur dioxide, sulphates và alcohol.

→ Các biến liên quan có tương quan khá vừa ($\text{correlation} < 0.8$) vậy ta nên chọn hết để build model.

- Sử dụng IQR method kiểm tra Outliers



→ Các cột có outliers: ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']

Kết luận: Ta sẽ giữ outliers do dataset này khá bé đồng nhất và không có giá trị null.

a. Feature Engineering:

Cần biến đổi những feature có tương quan với nhau thấp hoặc cao mà có thể gây ra đa cộng tuyến, có liên hệ với nhau theo biến, có phân phối giá trị không đồng đều, đó được coi là “những features nhiễu” nên để giúp máy học tốt hơn mà không bị ảnh hưởng bởi những features nhiễu này, ta tiến hành việc nhóm, chia bins.

```
# Các cặp biến có tương quan rất cao so với các biến khác có thể chỉ ra thông tin dư thừa hoặc nhiễu:
# 'free sulfur dioxide' và 'total sulfur dioxide' có tương quan 0.67, điều này có thể gây ra đa cộng tuyến.
df['sulfur_ratio'] = df['free sulfur dioxide'] / df['total sulfur dioxide'] # Tỷ lệ giữa sulfur dioxide tự do và tổng sulfur dioxide.
df['bound_sulfur'] = df['total sulfur dioxide'] - df['free sulfur dioxide'] # Lượng sulfur dioxide liên kết.

df['fixed_volatile_acidity_interaction'] = df['fixed acidity'] * df['volatile acidity']
# Tương tác giữa độ axit cố định và độ axit bay hơi.

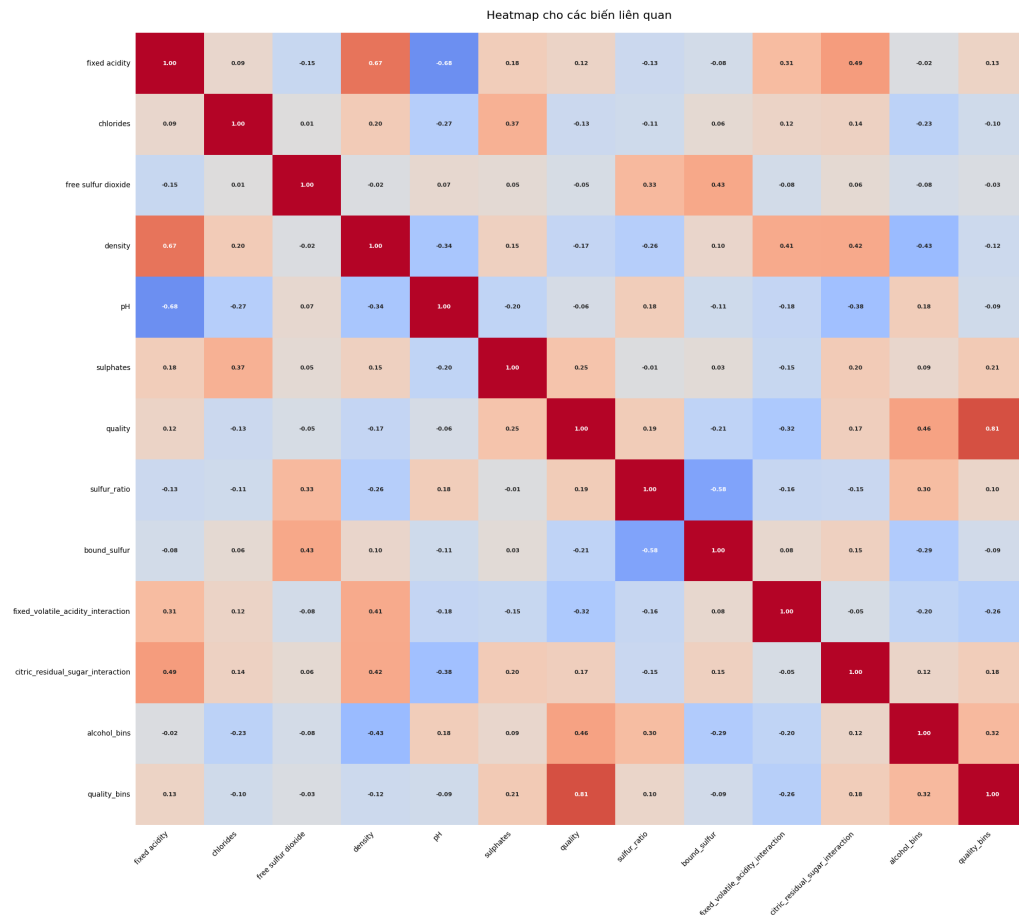
df['citric_residual_sugar_interaction'] = df['citric acid'] * df['residual sugar']
# Tương tác giữa axit citric và đường dư. (do cả hai đều có tương quan với density và cả hai đều có tương quan khá yếu với nhau)

q = df['alcohol'].quantile([0.2, 0.4, 0.6, 0.8])
# Phân loại nồng độ cồn thành các nhóm (rất thấp, thấp, trung, cao, rất cao) vì có sự phân loại khá rõ.
# gán nhãn
df['alcohol_bins'] = pd.cut(df['alcohol'], bins=[-np.inf, q[0.2], q[0.4], q[0.6], q[0.8], np.inf], labels=[0, 1, 2, 3, 4])

df['quality_bins'] = pd.cut(df['quality'], bins=[-np.inf, 4, 6, np.inf], labels=[0, 1, 2])
# Phân loại chất lượng thành 3 loại (tệ trung bình, tốt) nhằm giảm thiểu vấn đề mất cân bằng và rút gọn lại giúp máy học tốt hơn
```

b. Chọn đặc trưng:

- Sau khi thêm biến mới, ta kiểm tra lại correlation và sử dụng heatmap hiển thị mức độ tương quan giữa các biến, giúp dễ dàng nhận biết các biến có tương quan mạnh với nhau.
- Sử dụng hàm correlation (đưa vào 1 function) giúp tự động xác định các cặp biến có hệ số tương quan tuyệt đối lớn hơn 0.7. Điều này rất hữu ích cho việc lựa chọn đặc trưng và tránh đa cộng tuyến trong mô hình học máy.
- Sử dụng heatmap hiển thị trực quan ma trận tương quan giữa các biến



3.3. Chia dữ liệu:

Để xử lý bài toán này, ta tiến hành tách dữ liệu thành 2 phần: 1 dataframe (X) mới gồm tất cả các cột ngoại trừ quality, là tập hợp các đặc trưng và 1 biến mục tiêu y được gán từ cột quality. Ta tiến hành tính toán tần suất của từng giá trị trong cột quality.

proportion	
quality	
5.0	0.425891
6.0	0.398999
7.0	0.124453
4.0	0.033146
8.0	0.011257
3.0	0.006254

dtype: float64

→Ta thấy tập y có sự mất cân bằng khá lớn giữa các giá trị.

Để có thể cân bằng lại sự lệch của tập mẫu, giảm tình trạng bias, ta sử dụng kỹ thuật **SMOTE**. Cụ thể hơn, ở đây chúng em sử dụng **SMOTETomek** (kết hợp kỹ thuật SMOTE và kỹ thuật Tomek Links) thay vì SMOTE thông thường để loại bỏ các mẫu dữ liệu của lớp đa số nằm gần biên giới giữa các lớp, giúp cải thiện hiệu quả phân loại.

proportion	
quality	
3.0	0.173224
7.0	0.171691
8.0	0.171691
4.0	0.170669
5.0	0.156362
6.0	0.156362

dtype: float64

→Sau khi sử dụng kết hợp 2 kỹ thuật trên, ta thấy tập y đã có sự cân bằng giữa các giá trị.

Bên cạnh đó, một trong những vấn đề phổ biến của bài toán phân lớp là sự mất cân bằng trong dữ liệu tập mẫu dẫn đến vấn đề bias và overfitting. Vì vậy, để đảm bảo tính nguyên vẹn của dataset cùng với đó là giảm thiểu sự ảnh hưởng của các giá trị ngoại lai mà không loại bỏ chúng, ta sử dụng **RobustScaler** - bộ công cụ chuẩn hóa dữ liệu.

Ngoài ra, chúng em còn tiến hành chia dataset thành 2 tập dữ liệu là tập train và tập test với tỉ lệ là 80-20. Mục đích của việc này là để chuẩn bị dữ liệu cho học máy 2 tập riêng biệt và áp dụng cơ chế đặc trưng để làm cho các đặc trưng phù hợp với mô hình.

3.4. Xây dựng mô hình cơ bản:

▼

DecisionTreeClassifier

i


?

DecisionTreeClassifier(random_state=42)

Định nghĩa, khởi tạo và huấn luyện model dt - Decision Tree Classifier dựa trên bộ dữ liệu training, với tham số random state là 42.

Sau khi sử dụng model huấn luyện để dự đoán trên tập test, nhóm đánh giá hiệu năng mô hình dựa trên độ chính xác và ma trận nhầm lẫn.

Kết quả:



Accuracy

0.9029374201787995

Classification Report

	precision	recall	f1-score	support
3.0	0.95	0.96	0.95	130
4.0	0.96	0.94	0.95	125
5.0	0.78	0.76	0.77	118
6.0	0.81	0.83	0.82	143
7.0	0.98	0.94	0.96	133
8.0	0.94	0.98	0.96	134
accuracy			0.90	783
macro avg	0.90	0.90	0.90	783
weighted avg	0.90	0.90	0.90	783

[[125 5 0 0 0 0]

[7 118 0 0 0 0]

[0 0 90 28 0 0]

[0 0 25 118 0 0]

[0 0 0 0 125 8]

[0 0 0 0 3 131]]

Accuracy: 0.9029 (~90.29%)

Hiệu suất theo từng lớp:

- Lớp 3.0: Hiệu suất rất tốt với precision 0.95, recall 0.96 và F1-score 0.95. Mô hình có khả năng nhận diện và phân loại lớp này rất tốt.
- Lớp 4.0: Hiệu suất khá tốt với precision 0.96, recall 0.94 và F1-score 0.95.
- Lớp 5.0: Hiệu suất trung bình với precision 0.78, recall 0.76 và F1-score 0.77. Có thể cần cải thiện, đặc biệt là recall.
- Lớp 6.0: Hiệu suất khá cân bằng trong các lớp, với precision 0.81, recall 0.83 và F1-score 0.82.
- Lớp 7.0: Hiệu suất khá tốt với precision 0.98, recall 0.94 và F1-score 0.96.
- Lớp 8.0: Hiệu suất rất tốt, gần như ngang bằng với lớp 3.0, có precision 0.94, recall 0.98 và F1-score 0.96.

=> TÓM LẠI:

- Mô hình hoạt động khá tốt, với độ chính xác tổng thể là 90%.
- Mô hình hoạt động tốt nhất cho nhãn 3-4-7-8, với Precision và Recall cao (~95% và 95%).
- Mức độ dự đoán đúng của mô hình cho nhãn 6 và nhãn 5 cũng khá cao, nhưng có một số trường hợp dự đoán nhầm lẫn giữa nhãn 6 và nhãn 5 (dự đoán nhãn 6 thành nhãn 5 và ngược lại).

=> Ta cần ưu tiên hơn về sự cân bằng giữa precision và recall để cho lấy được kết quả sát nhất so với tập mẫu.


3.5. Kỹ thuật DecisionTree Pruning

Tạo một model mới dt_test sử dụng DecisionTreeClassifier. Sau đó cho chạy vòng lặp for để tìm ra ccp_alpha tối ưu nhất cho mô hình cây quyết định.

Kết quả:

alpha tốt nhất: 0.0002965734361454579

Chạy lại đánh giá hiệu năng bằng độ chính xác và ma trận nhầm lẫn, đạt được kết quả mới như sau:



Accuracy

0.9042145593869731

Classification Report

	precision	recall	f1-score	support
3.0	0.95	0.96	0.95	130
4.0	0.96	0.94	0.95	125
5.0	0.78	0.78	0.78	118
6.0	0.82	0.82	0.82	143
7.0	0.98	0.94	0.96	133
8.0	0.94	0.98	0.96	134
accuracy			0.90	783
macro avg			0.90	783
weighted avg			0.90	783

```

[[125  5  0  0  0  0]
 [  7 118  0  0  0  0]
 [  0  0 92 26  0  0]
 [  0  0 26 117  0  0]
 [  0  0  0  0 125  8]
 [  0  0  0  0  3 131]]

```

Sau khi áp dụng pruning, có một số cải thiện nhỏ trong hiệu suất của mô hình:

- Độ chính xác tổng thể tăng nhẹ (lên 0.2%).
- Lớp 5.0 và 6.0 có sự cải thiện rõ rệt nhất.
- Lớp 5.0: Trước: 0.78 / 0.76 / 0.77 - Sau: 0.78 / 0.78 / 0.78

=> Cải thiện nhẹ ở cả ba chỉ số.

- Lớp 6.0: Trước: 0.81 / 0.83 / 0.82 - Sau: 0.82 / 0.82 / 0.82

=> Cải thiện precision và F1-score, recall bị giảm nhẹ nhưng mô hình đã cho thấy sự cân bằng rất tốt ở lớp này .

Còn lại các lớp khác không đổi

=> Kết luận:

- Pruning đã giúp mô hình cải thiện một chút về hiệu suất tổng thể, đặc biệt là Precision và F1-Score cho nhãn 0 và nhãn 1.
- Độ chính xác tổng thể chỉ tăng nhẹ, nhưng pruning giúp mô hình trở nên tổng quát hơn và tránh overfitting.

3.6. Điều chỉnh siêu tham số: FINE TUNNING VỚI RandomSearchCV

Sử dụng Fine tuning với các tham số như sau:

- Tham số `max_depth` giá trị từ 2 đến 20
- Tham số `min_sample_leafs` giá trị từ 1 đến 15
- Tham số `min_sample_split` giá trị từ 1 đến 15
- Tham số `criterion` là “gini” hoặc “entropy”
- Tham số `splitter` là “best” hoặc “random”

Sử dụng hàm `RandomSearchCV` với ưu tiên scoring là độ chính xác (accuracy) cho mô hình đã pruning, để tìm ra mô hình tối ưu.

Kết quả:

```
Best Parameters (Random Search): {'splitter': 'best', 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': 15, 'criterion': 'entropy'}
Best Model (Random Search): DecisionTreeClassifier(ccp_alpha=0.0002965734361454579, criterion='entropy',
max_depth=15, random_state=42)
```

⇒ Vậy model tối ưu của ta sẽ có những tham số, `criterion='entropy'`, `max_depth=15`, `min_samples_leaf=1`, `random_state=42`. Tuy nhiên, để có được sự cân bằng giữa precision và recall, ta nên dùng `criterion 'gini'` (phù hợp hơn cho CART và dẫn đến sự phân bổ lớp học cân bằng hơn).

3.7. Model tối ưu

Sử dụng mô hình tối ưu với các tham số vừa tìm được và chạy lại độ chính xác và ma trận nhầm lẫn, ta được kết quả mới:

```
Accuracy for the best tree model 0.9054916985951469
Classification Report
```

	precision	recall	f1-score	support
3.0	0.95	0.96	0.95	130
4.0	0.96	0.94	0.95	125
5.0	0.79	0.79	0.79	118
6.0	0.83	0.83	0.83	143
7.0	0.98	0.93	0.95	133
8.0	0.94	0.98	0.96	134
accuracy			0.91	783
macro avg	0.91	0.90	0.90	783
weighted avg	0.91	0.91	0.91	783

```
[[125  5  0  0  0  0]
 [  7 118  0  0  0  0]
 [  0  0  93 25  0  0]
 [  0  0  25 118  0  0]
 [  0  0  0  0 124  9]
 [  0  0  0  0  3 131]]
```

3.8. Nhận xét và Đánh giá

Accuracy:

Sau khi pruning: 0.9042

Sau khi `RandomizedSearchCV`: 0.9055

Precision, Recall, F1-score:

- Class 3.0, 4.0, 7.0, 8.0: Không có thay đổi đáng kể giữa các mô hình. Các chỉ số precision, recall và f1-score gần như giữ nguyên ở cả hai mô hình.
- Class 5.0 và 6.0:
 - Class 5.0: F1-score sau khi `RandomizedSearchCV` tăng từ 0.78 lên 0.79. Precision và recall cũng cải thiện nhẹ.

- Class 6.0: F1-score tăng từ 0.82 lên 0.83. Precision và recall cũng tăng nhẹ.

Confusion Matrix:

Cả hai confusion matrix rất giống nhau, ngoại trừ một vài thay đổi nhỏ:

- Class 5.0: Sau khi RandomizedSearchCV, số mẫu dự đoán đúng (93) tăng lên một đơn vị so với mô hình sau khi pruning (92).
- Class 6.0: Số mẫu dự đoán đúng (118) cũng tăng lên 1 so với sau pruning (117).
- Macro và Weighted Averages:

Cả hai mô hình đều có macro và weighted averages gần như giống nhau, với một sự cải thiện nhỏ trong precision và f1-score sau khi áp dụng RandomizedSearchCV.

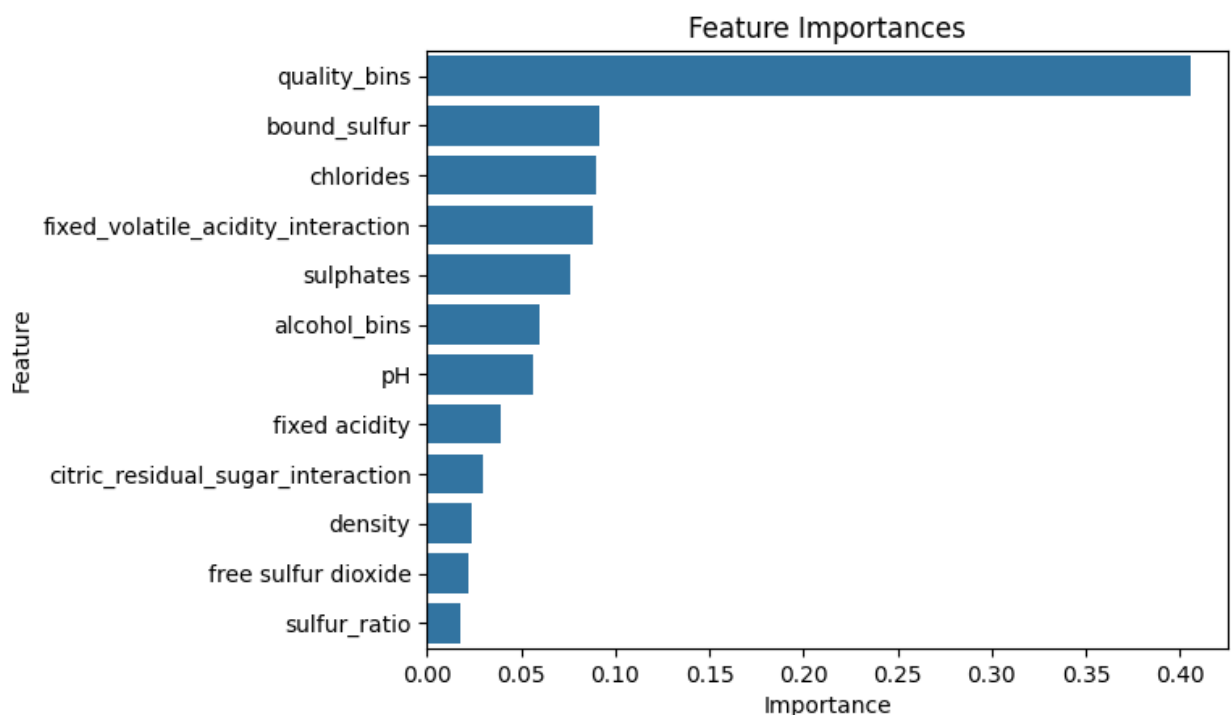
⇒ Kết Luận :

- Model tối ưu sau RandomSearchCV và pruning có độ chính xác (90.55 %) và hiệu suất cao hơn. Đặc biệt, Precision, Recall, và F1-Score của tất cả các nhãn đều được cải thiện hoặc duy trì tốt hơn.
- Số mẫu dự đoán đúng cho các nhãn cũng tăng lên, đặc biệt là nhãn 0 và nhãn 2, giúp mô hình dự đoán chính xác hơn cho các nhãn này.

⇒ TA XÂY DỰNG ĐƯỢC 1 MÔ HÌNH TỐI ƯU NHẤT SỬ DỤNG
DecisionTreeClassifier

3.9. Những features đặc trưng nhất

Dùng hàm feature_importances để lấy ra các features đặc trưng nhất và trực quan hóa, ta được kết quả:



3.10. Màn hình chức năng sử dụng thư viện Streamlit & Joblib:

MÔ HÌNH DỰ ĐOÁN CHẤT LƯỢNG RƯỢU

Nhập nồng độ cho fixed acidity

0.00

-

+

Nhập nồng độ cho chlorides

0.00

-

+

Nhập nồng độ cho free sulfur dioxide

0.00

-

+

Nhập nồng độ cho density

0.00

-

+

Nhập nồng độ cho pH

0.00

-

+

2.00

-

+

Nhập nồng độ cho citric residual sugar interaction

1.00

-

+

Chọn nồng độ cồn alcohol bins

trung bình : 2

▼

Chọn chất lượng quality bins

tốt : 2

▼

DỰ ĐOÁN

Kết quả dự đoán chất lượng rượu: 7.0

TÀI LIỆU THAM KHẢO

1. Tek4. (n.d.). *Thuật toán CART*. Tek4. Truy cập ngày [1/10/2024], từ <https://tek4.vn/khoa-hoc/machine-learning-co-ban/thuat-toan-cart>.
2. Selawsky, J. (2021, December 6). *Decision trees with CART algorithm*. Medium. Truy cập ngày [2/10/2024], từ <https://medium.com/geekculture/decision-trees-with-cart-algorithm-7e179acee8ff>.
3. GeeksforGeeks. (n.d.). *CART (Classification and Regression Tree) in Machine Learning*. Truy cập ngày [2/10/2024], từ <https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/>.