

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CHUYÊN NGÀNH

TÌM HIỂU CÁC PHƯƠNG PHÁP HỌC MÁY DỰ
ĐOÁN CHẤT LƯỢNG RƯỢU VANG

GVHD: NGUYỄN THỊ DIỆU HIỀN

DƯƠNG TRỌNG BÌNH – 2001200657 – 11DHTH9

ĐOÀN QUANG MINH – 2001202153 – 11DHTH11

TRẦN MINH QUÂN – 2001207294 – 11DHTH11

TP. HỒ CHÍ MINH, tháng 12 năm 2023

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CHUYÊN NGÀNH

TÌM HIỂU CÁC PHƯƠNG PHÁP HỌC MÁY DỰ
ĐOÁN CHẤT LƯỢNG RƯỢU VANG

GVHD: NGUYỄN THỊ DIỆU HIỀN

DƯƠNG TRỌNG BÌNH – 2001200657 – 11DHTH9

ĐOÀN QUANG MINH – 2001202153 – 11DHTH11

TRẦN MINH QUÂN – 2001207294 – 11DHTH11

TP. HỒ CHÍ MINH, tháng 12 năm 2023

PHÂN CÔNG

Thành viên	Công việc	Mức độ hoàn thành
2001200657 Dương Trọng Bình	Phân tích, xử lý dữ liệu, MLP	100%
2001202153 Đoàn Quang Minh	Xây dựng ứng dụng, ANN	100%
2001207294 Trần Minh Quân	Cở sở lý thuyết, RNN	100%

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong Đồ án là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Đồ án này đã được cảm ơn và các thông tin trích dẫn trong Đồ án đã được chỉ rõ nguồn gốc.

Sinh viên thực hiện Đồ án

Dương Trọng Bình – 2001200657 – 11DHTH9

Đoàn Quang Minh – 2001202153 – 11DHTH11

Trần Minh Quân – 2001207294 – 11DHTH11

LỜI CẢM ƠN

Tôi xin gửi lời cảm ơn sâu sắc đến cô Nguyễn Thị Diệu Hiền giảng viên hướng dẫn nhóm chúng tôi thực hiện đồ án này và toàn thể giảng viên khoa Công nghệ thông tin trường Đại học Công Thương TP.HCM. Tôi muốn bày tỏ lòng biết ơn và trân trọng sự hướng dẫn và định hướng mà cô đã cung cấp trong suốt quá trình thực hiện đồ án.

Đồ án này không chỉ giúp nhóm nắm vững kiến thức về chuyên ngành khoa học phân tích dữ liệu và dữ báo mà còn trang bị cho chúng tôi những kỹ năng và công cụ cần thiết để làm việc trong lĩnh vực này. Qua việc áp dụng những phương pháp và kỹ thuật học được trong các môn học, chúng tôi đã có cơ hội áp dụng kiến thức thực tế vào việc nghiên cứu và xây dựng mô hình dự báo.

Tôi cũng muốn gửi lời cảm ơn đến các thành viên trong nhóm của tôi. Sự cống hiến, tương trợ và cộng tác của mọi người đã giúp tôi hoàn thành đồ án một cách thành công. Qua quá trình làm việc nhóm, tôi đã học được nhiều điều về việc làm việc trong môi trường đa dạng và phát triển kỹ năng giao tiếp và quản lý thời gian.

Xin chân thành cảm ơn!

Dương Trọng Bình

Đoàn Quang Minh

Trần Minh Quân

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục đích	1
1.3. Đối tượng và phạm vi nghiên cứu.....	1
1.4. Ý nghĩa khoa học và thực tiễn	1
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	3
2.1. Recurrent Neural Network (RNN).....	3
2.1.1. RNN là gì?	3
2.1.2. Các loại RNN	4
2.2. Artificial neuron network (ANN)	5
2.2.1. ANN là gì?	5
2.2.2. Cấu trúc của nơron nhân tạo	6
2.2.3. Mạng ANN học như thế nào?	7
2.3. Multi-layer Perceptron (MLP)	9
2.3.1. Kiến trúc mạng MLP	9
2.3.2. Huấn luyện mạng MLP	10
2.4. Hồi quy logistic	11
2.4.1. Hồi quy logistic là gì?	11
2.4.2. Huấn luyện mô hình.....	12
2.5. Cây quyết định (Decision Tree)	12
2.5.1. Cây quyết định là gì	12
2.5.2. Thuật toán cây quyết định hoạt động như thế nào?	12
2.5.3. Xây dựng cây quyết định	13
2.5.4. Các bước xây dựng cây quyết định.....	13
CHƯƠNG 3: GIỚI THIỆU TẬP DỮ LIỆU	14
3.1. Thông tin tập dữ liệu.....	14
3.2. Tóm tắt thống kê dữ liệu.....	15

3.3. Thông tin thuộc tính.....	15
3.4. Mô tả thuộc tính	16
3.5. Thống kê thuộc tính đầu ra	16
3.6. Biểu đồ histogram và boxplot	17
3.7. Biểu đồ heatmap.....	20
CHƯƠNG 4: TIỀN XỬ LÝ DỮ LIỆU.....	22
4.1. Loại bỏ cột dữ liệu không cần thiết	22
4.2. Kiểm tra dữ liệu thiếu và dữ liệu trùng.....	23
4.3. Gom nhóm dữ liệu	25
4.4. Cân bằng dữ liệu	25
CHƯƠNG 5: LỰA CHỌN ĐẶC TRƯNG.....	27
5.1. Chia dữ liệu train và test	27
5.2. Mô hình cây quyết định	27
5.3. Xây dựng mô hình logistic.....	31
CHƯƠNG 6: MÔ HÌNH MẠNG HỌC SÂU	35
6.1. Mô hình MLP.....	35
6.2. Mô hình ANN	35
6.3. Mô hình RNN.....	36
6.4. So sánh mô hình.....	37
6.4.1. So sánh độ chính xác.....	37
6.4.2. So sánh ma trận nhầm lẫn (confusion matrix)	38
6.4.3. Báo cáo phân loại.....	40
CHƯƠNG 7: GIAO DIỆN CHỨC NĂNG.....	42
7.1. Giao diện đăng nhập	42
7.2. Giao diện chức năng chính.....	44
7.3. Dự đoán trên tập dữ liệu.....	49
CHƯƠNG 8: TỔNG KẾT	55
8.1. Các yếu tố quan trọng	55
8.2. Hướng dẫn sử dụng.....	55
8.3. Đối tượng người dùng.....	55

KẾT LUẬN.....	56
TÀI LIỆU THAM KHẢO.....	57

DANH MỤC HÌNH ẢNH

Hình 2.1: Hoạt động của mạng RNN.....	3
Hình 2.2: Các loại RNN.....	4
Hình 2.3: Cấu trúc của nơron.....	6
Hình 2.4: Các lớp của nơron.....	7
Hình 2.5: Huấn luyện mạng ANN.....	8
Hình 2.6: Kiến trúc mạng MLP.....	10
Hình 2.7: Công thức hồi quy logistic.....	12
Hình 3.1 5: dòng dữ liệu.....	14
Hình 3.2: Thông tin dữ liệu.....	14
Hình 3.3: Bảng tóm tắt dữ liệu.....	15
Hình 3.4: Biểu đồ thống kê tần số cột quality.....	17
Hình 3.5: Biểu đồ histogram và boxplot 1.....	18
Hình 3.6: Biểu đồ histogram và boxplot 2.....	19
Hình 3.7: Biểu đồ histogram và boxplot 3.....	20
Hình 3.8: Biểu đồ heatmap.....	21
Hình 4.1: Loại bỏ cột Id.....	22
Hình 4.2: Tạo cột SO2.....	23
Hình 4.3: Kiểm tra dữ liệu thiếu.....	24
Hình 4.4: Kiểm tra dữ liệu trùng.....	24
Hình 4.5: Xóa dữ liệu trùng.....	24
Hình 4.6: Gom nhóm dữ liệu.....	25
Hình 4.7: Cân bằng dữ liệu.....	26
Hình 5.1: Chia dữ liệu train và test.....	27
Hình 5.2: Tìm độ sâu tối ưu.....	28
Hình 5.3: Xây dựng mô hình cây quyết định.....	28
Hình 5.4: Độ quan trọng của các cột.....	29
Hình 5.5: Xóa thuộc tính không quan trọng.....	30

Hình 5.6: Chia dữ liệu train và test lần 2	30
Hình 5.7: Tạo mô hình ngẫu nhiên	31
Hình 5.8: Xây dựng mô hình logistic.....	32
Hình 5.9: Chọn mô hình logistic tốt nhất.....	33
Hình 5.10: Tập dữ liệu với 4 biến đặc trưng.....	33
Hình 5.11: Chia dữ liệu train và test lần cuối	34
Hình 6.1: Mô hình MLP.....	35
Hình 6.2: Mô hình ANN	35
Hình 6.3: Biên dịch mô hình ANN	36
Hình 6.4: Chuyển đổi đầu ra mô hình ANN	36
Hình 6.5: Mô hình RNN	37
Hình 6.6: Chuyển đổi đầu ra mô hình RNN	37
Hình 6.7: So sánh Accuracy giữa các mô hình	38
Hình 6.8: Confusion Matrix – MLP.....	39
Hình 6.9: Confusion Matrix – ANN	39
Hình 6.10: Confusion Matrix – RNN	40
Hình 6.11: Báo cáo phân loại các mô hình	41
Hình 7.1: Kết nối MongoDB	42
Hình 7.2: Mở ứng dụng MongoDB Compass.....	42
Hình 7.3: Tạo mới database	43
Hình 7.4: Thêm dữ liệu cho collection	44
Hình 7.5: Hoàn thành tạo và thêm dữ liệu cho database	44
Hình 7.6: Giao diện chức năng chính	46
Hình 7.7: Dự đoán trên tập test.....	47
Hình 7.8: Dự đoán và lưu vào cơ sở dữ liệu	48
Hình 7.9: Dữ liệu lưu thành công	49
Hình 7.10: Giao diện dự đoán trên tập dữ liệu.....	50
Hình 7.11: Chọn tập dữ liệu cần dự đoán	51
Hình 7.12: Dự đoán trên tập dữ liệu	52
Hình 7.13: Lưu tập dữ liệu.....	53

Hình 7.14: Cập nhật giao diện chính	54
---	----

LỜI MỞ ĐẦU

Trong thời gian gần đây, ngành công nghiệp rượu vang đỏ đã trở thành một lĩnh vực thu hút sự quan tâm của nhiều người trên toàn thế giới. Việc hiểu rõ các yếu tố ảnh hưởng đến chất lượng của rượu vang đỏ không chỉ giúp người làm trong ngành này nắm bắt được những thông tin quan trọng, mà còn giúp các nhà sản xuất và người tiêu dùng có thể đưa ra quyết định thông minh và đáng tin cậy

Với tập dữ liệu về rượu vang đỏ, chúng tôi đã tiến hành phân tích chi tiết về các thuộc tính và tìm hiểu mối quan hệ giữa chúng. Chúng tôi đã xây dựng mô hình hồi quy để mô phỏng các mối quan hệ này và đưa ra dự đoán về chất lượng của rượu vang đỏ. Ngoài ra, chúng tôi đã xây dựng các mô hình mạng học sâu để dự đoán chất lượng rượu vang.

Đồ án này không chỉ góp phần mở rộng kiến thức và kỹ năng của chúng tôi về phân tích dữ liệu, mà còn đáp ứng nhu cầu thực tiễn trong lĩnh vực rượu vang đỏ. Chúng tôi tin rằng kết quả của đồ án sẽ mang lại những thông tin hữu ích và cung cấp các gợi ý quan trọng cho các nhà sản xuất và người tiêu dùng.

Chúng tôi hy vọng rằng đồ án này đáp ứng được sự mong đợi và sẽ được đánh giá cao về tính kỹ thuật, sự sáng tạo và khả năng ứng dụng. Chân thành cảm ơn giảng viên hướng dẫn Nguyễn Thị Diệu Hiền và giảng viên phản biện đã dành thời gian để xem xét đồ án của chúng tôi.

CHƯƠNG 1: TỔNG QUAN

1.1. Lý do chọn đề tài

Chất lượng rượu vang là một yếu tố quan trọng đối với người tiêu dùng và ngành công nghiệp rượu vang. Truyền thống, việc đánh giá chất lượng rượu vang thường dựa trên các phương pháp phân tích hóa học và cảm quan của chuyên gia. Tuy nhiên, nhờ sự phát triển mạnh mẽ của trí tuệ nhân tạo và mạng học sâu, chúng ta có thể áp dụng các phương pháp này để dự đoán chất lượng rượu vang một cách tự động và hiệu quả hơn. Vì vậy, nghiên cứu về việc áp dụng mạng học sâu để dự đoán chất lượng rượu vang là hết sức cần thiết để nắm bắt những tiềm năng và lợi ích của công nghệ này trong lĩnh vực rượu vang.

1.2. Mục đích

Mục đích của đề tài là tìm hiểu và nghiên cứu các phương pháp mạng học sâu (deep learning) được áp dụng để dự đoán chất lượng rượu vang. Điều này bao gồm việc tìm hiểu về kiến trúc mạng học sâu phù hợp, xây dựng và huấn luyện mô hình dự đoán chất lượng rượu vang từ các dữ liệu đầu vào như thông tin về thành phần hóa học và các thông số khác liên quan đến chất lượng.

1.3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài là các phương pháp mạng học sâu được áp dụng để dự đoán chất lượng rượu vang. Phạm vi của nghiên cứu sẽ tập trung vào việc khám phá, so sánh và đánh giá hiệu suất của các phương pháp này. Các dữ liệu đầu vào có thể bao gồm thông tin về thành phần hóa học và các thông số khác liên quan đến chất lượng rượu vang.

1.4. Ý nghĩa khoa học và thực tiễn

- **Ý nghĩa khoa học:** Nghiên cứu này có thể cung cấp kiến thức về ứng dụng của mạng học sâu trong lĩnh vực dự đoán chất lượng rượu vang. Việc tìm hiểu các phương pháp mạng học sâu có thể giúp nâng cao hiểu biết về khả năng của trí tuệ nhân tạo trong việc phân tích và dự đoán chất lượng sản phẩm.

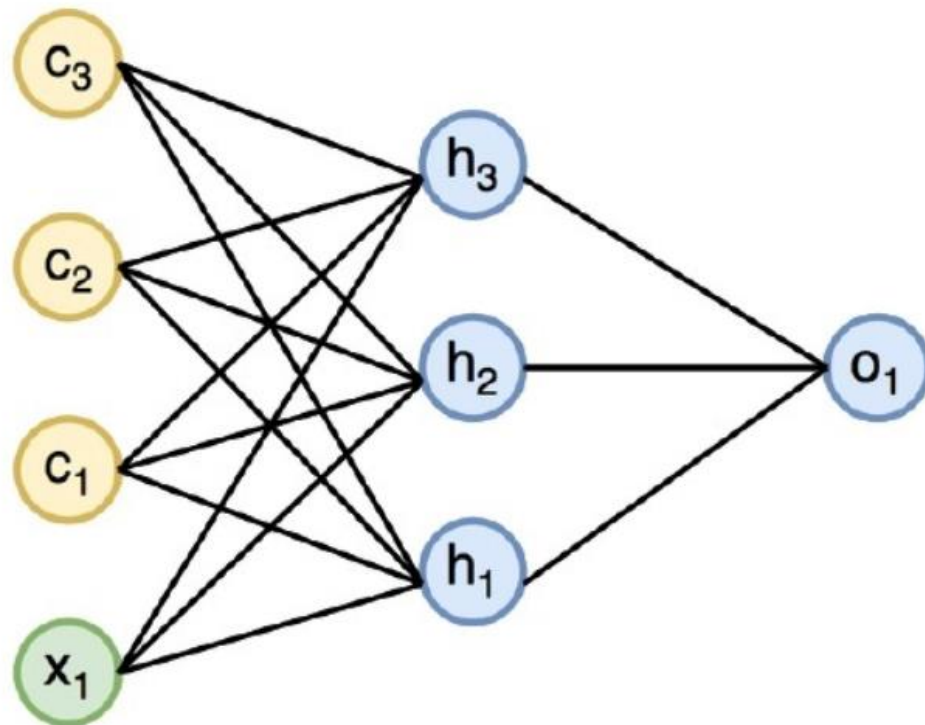
- **Ý nghĩa thực tiễn:** Kết quả của nghiên cứu có thể góp phần vào việc cải thiện quy trình đánh giá chất lượng rượu vang trong ngành công nghiệp. Việc áp dụng mạng học sâu để dự đoán chất lượng rượu vang có thể giúp tăng cường độ chính xác và hiệu quả trong quá trình đánh giá, từ đó giúp các nhà sản xuất rượu vang đưa ra quyết định sản xuất và đảm bảo chất lượng sản phẩm. Điều này có thể giúp tiết kiệm thời gian, công sức và tài nguyên, đồng thời tăng cường sự cạnh tranh và đáp ứng nhu cầu của thị trường.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Recurrent Neural Network (RNN)

2.1.1. RNN là gì?

- **Recurrent neural networks (RNN)** là một loại thuật toán định hướng học sâu theo cách tiếp cận tuần tự. Trong mạng nơ-ron, ta luôn giả định rằng mỗi đầu vào và đầu ra là độc lập với tất cả các lớp khác. Loại mạng nơ-ron này được gọi là mạng lặp lại vì chúng thực hiện các phép tính toán học một cách tuần tự hoàn thành nhiệm vụ này đến tác vụ khác.



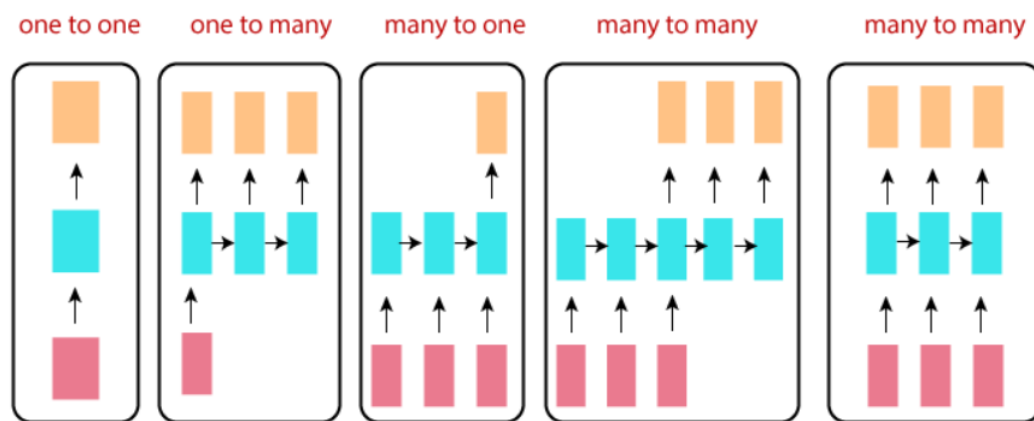
Hình 2.1: Hoạt động của mạng RNN

- Trong hình trên, c_1 , c_2 , c_3 và x_1 được coi là đầu vào bao gồm một số giá trị đầu vào ẩn cụ thể là h_1 , h_2 và h_3 cung cấp đầu ra tương ứng là o_1 . Bây giờ ta sẽ tập trung vào việc triển khai PyTorch để tạo ra một sóng sin với sự trợ giúp của các mạng RNN.

- Trong quá trình đào tạo, tôi sẽ thực hiện theo cách tiếp cận đào tạo mô hình với một điểm dữ liệu tại một thời điểm. Chuỗi đầu vào x bao gồm 20 điểm dữ liệu và chuỗi đích được coi là giống với chuỗi đầu vào.

2.1.2. Các loại RNN

- Lý do chính khiến các mạng lặp lại thú vị hơn là chúng cho phép chúng ta hoạt động trên các chuỗi vector: Trình tự trong đầu vào, đầu ra hoặc trong trường hợp chung nhất là cả hai. Một vài ví dụ có thể cụ thể hơn:



Hình 2.2: Các loại RNN

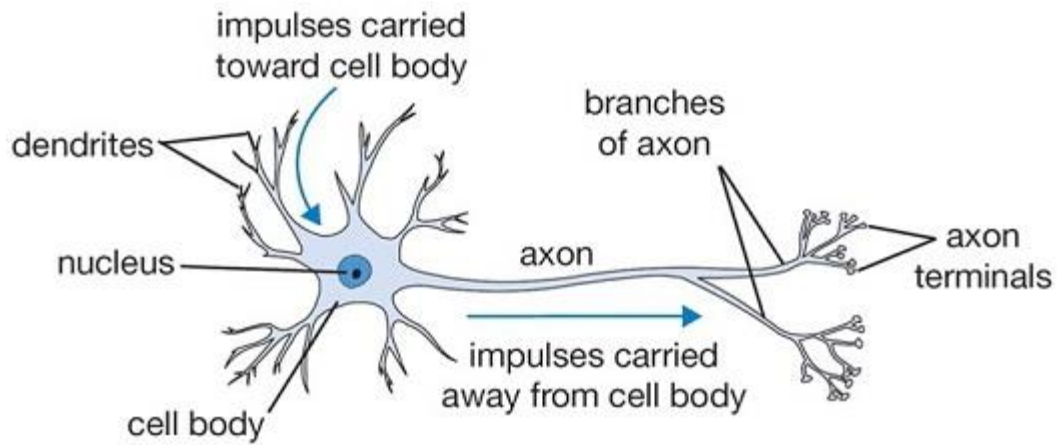
- Mỗi hình chữ nhật trong hình trên đại diện cho các vector và các mũi tên đại diện cho các hàm. Vector đầu vào có màu Đỏ, vector đầu ra có màu xanh lam và màu xanh lá cây giữ trạng thái của RNN.
- **One-to-One:**
 - Đây còn được gọi là mạng Neural thuần túy. Nó xử lý một kích thước cố định của đầu vào với kích thước cố định của đầu ra, nơi chúng độc lập với thông tin / đầu ra trước đó.
 - Ví dụ: Phân loại ảnh.
- **One-to-Many:**
 - Nó xử lý một kích thước cố định của thông tin làm đầu vào cung cấp một chuỗi dữ liệu làm đầu ra.
 - Ví dụ: Image Captioning lấy hình ảnh làm đầu vào và đầu ra một câu từ.
- **Many-to-One:**

- Nó lấy một chuỗi thông tin làm đầu vào và đầu ra với kích thước cố định của đầu ra.
- Ví dụ: phân tích tình cảm trong đó bất kỳ câu nào được phân loại là thể hiện tình cảm tích cực hoặc tiêu cực.
- **Many-to-Many:**
 - Nó lấy Chuỗi thông tin làm đầu vào và xử lý các đầu ra lặp lại dưới dạng Chuỗi dữ liệu.
 - Ví dụ: Dịch máy, trong đó RNN đọc bất kỳ câu nào bằng tiếng Anh và sau đó xuất ra câu đó bằng tiếng Pháp.
- **Bidirectional Many-to-Many:** Đầu vào và đầu ra trình tự được đồng bộ hóa. Lưu ý rằng trong mọi trường hợp không có ràng buộc nào được chỉ định trước đối với trình tự độ dài bởi vì phép biến đổi lặp lại (màu xanh lá cây) là cố định và có thể được áp dụng nhiều lần tùy thích.

2.2. Artificial neuron network (ANN)

2.2.1. ANN là gì?

- **Mạng lưới nơron trong trí tuệ nhân tạo (Artificial Neural Network - ANN)** là một mô hình máy học phổ biến dựa trên cấu trúc và chức năng của não người xử lý thông tin. ANN được hình thành từ các đơn vị xử lý thông tin được gọi là nơron, chúng được kết nối với nhau thông qua hàm số phi tuyến tính hay còn gọi là trọng số để truyền thông tin.
- 1 nơron bao gồm các phần:
 - **Dendrite (các sợi nhánh):** chúng phân thành các nhánh cây xung quanh tế bào thể hiện cho inputs (dữ liệu đầu vào).
 - **Cell body:** nơi lưu trữ dữ liệu đầu vào sau đó có trách nhiệm xử lý thông tin đến và gửi tín hiệu đến các nơron.
 - **Axon (sợi trục):** nó là 1 cấu trúc dài, hình ống là cầu nối cho nơron đang xử lý thông tin gửi những thông tin đó đến các nơron khác.
 - **Axon terminal:** đầu ra của axon.

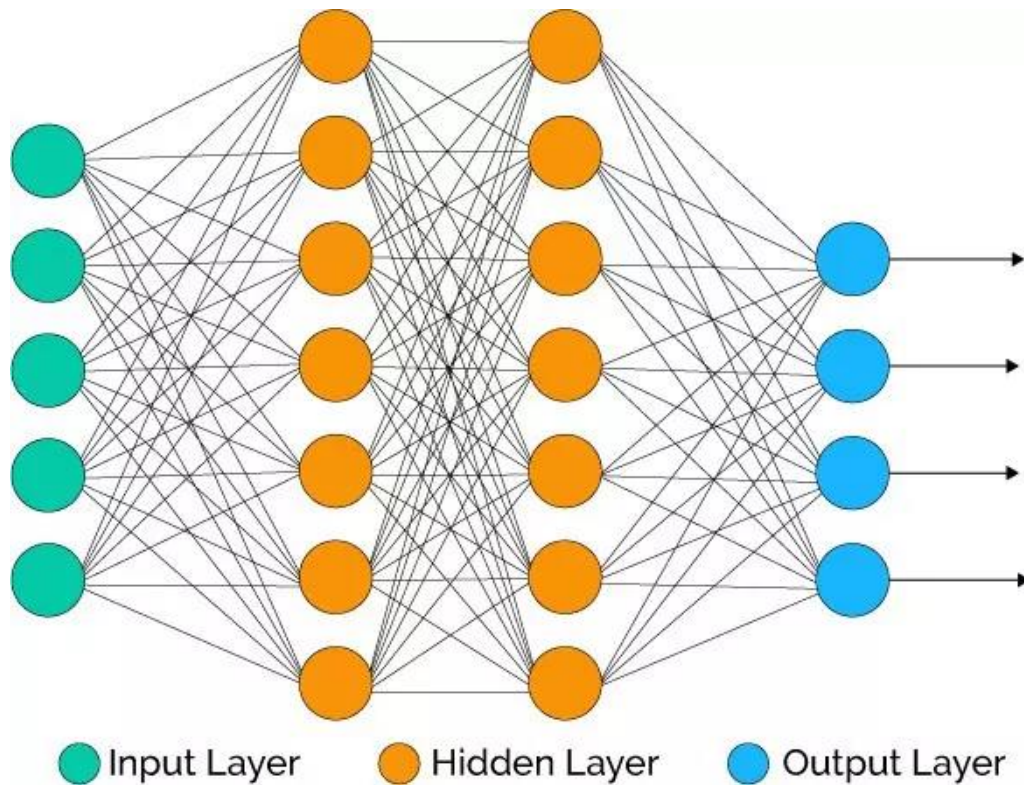


Hình 2.3: Cấu trúc của noron

2.2.2. Cấu trúc của noron nhân tạo

Noron bao gồm 4 phần chính:

- **Tầng đầu vào (input layer):** giống với Dendrite là nơi tiếp nhận dữ liệu đầu vào.
- **Tầng ẩn (hidden layer):** nơi tiếp nhận dữ liệu đầu vào và bắt đầu xử lý các đặc trưng tạo ra các đặc trưng mới dựa trên thông tin từ dữ liệu đầu vào.
- **Tầng đầu ra (output layer):** dự đoán đưa ra kết quả.
- **Thông tin chuyển theo dạng thẳng (feed forward):** dữ liệu được đi từ input layer và ra bằng output layer.

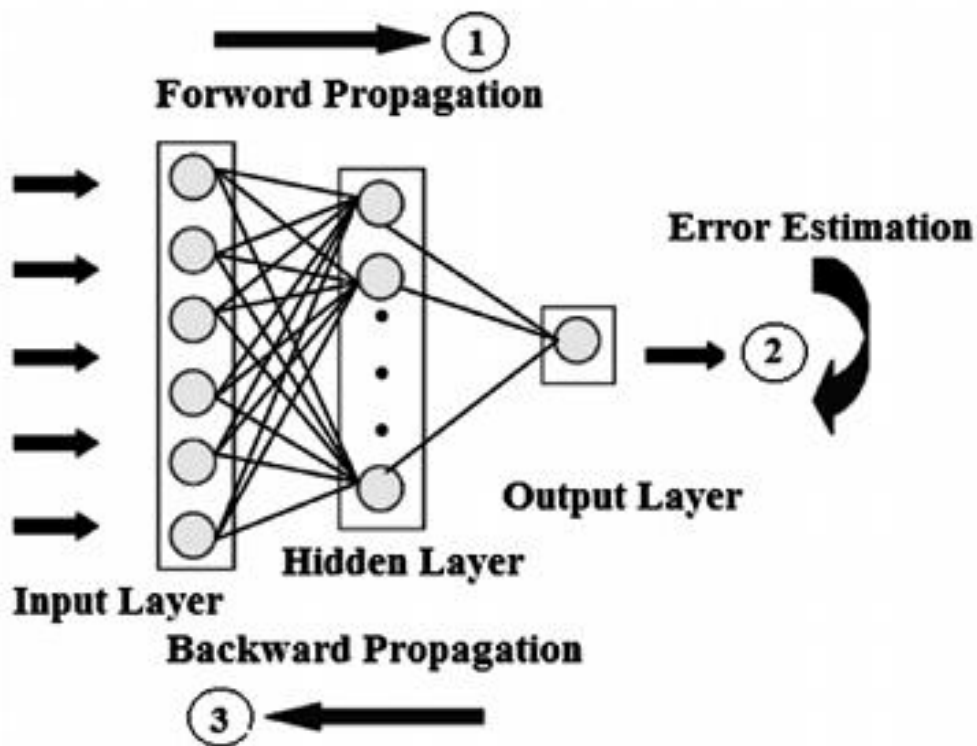


Hình 2.4: Các lớp của noron

2.2.3. Mạng ANN học như thế nào?

- Huấn luyện

- Cách huấn luyện mạng lưới ANN giống như cách con người chúng ta học vậy nếu chúng ta học và có lỗi sai thì có giáo viên giám sát để hỗ trợ cho việc học tập, việc của giáo viên giám sát ở đây đó là mô tả những gì được tạo ra từ dữ liệu đầu vào. Dựa trên giá trị đầu vào và giá trị dự đoán thì chúng ta có một giá trị lỗi đó là chênh lệch bình phương giữa giá trị đầu vào và giá trị dự đoán gọi tắt là hàm chi phí (cost function).
- Khi hàm chi phí đi qua các lớp thì trọng số sẽ được điều chỉnh trọng số sao hợp lý, sau mỗi lần chạy thì hàm chi phí sẽ giảm và giá trị dự đoán sẽ gần với giá trị đầu vào. Để làm cho hàm chi phí nhỏ hơn sau mỗi lần chạy thì cần phải dùng đến quy trình được gọi là **Back propagation** cho đến khi hàm chi phí đạt ở mức tối thiểu.



Hình 2.5: Huấn luyện mạng ANN

- **Hàm kích hoạt (activation function)**

- Nó rất là quan trọng nằm trong nơron, công việc của hàm này là sau khi tiếp nhận dữ liệu đầu ra từ nơron khác sẽ bắt đầu tính trọng số phù hợp và đưa ra giá trị của hàm kích hoạt nếu giá trị đó đạt ở một ngưỡng nhất định thì dữ liệu đầu ra sẽ được chuyển tiếp, ngược lại thì không.
- Và có 4 hàm kích hoạt:
 - **Hàm RELU:** giá trị hàm kích hoạt được tính theo công thức: $\text{ReLU}(x) = \max(0, x)$ nếu giá trị hàm kích hoạt ≥ 0 thì nơron sẽ truyền tiếp dữ liệu được gọi là tích cực, còn ngược lại < 0 sẽ không truyền thông tin gọi là tiêu cực.
 - **Hàm Sigmoid:** Sau khi tính giá trị hàm kích hoạt theo công thức: $\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$. Giá trị này sẽ nằm trong khoảng $(0,1)$, nếu gần 1 sẽ chuyển tiếp dữ liệu đầu ra.

- **Hàm Tanh:** Giá trị của hàm Tanh nằm trong khoảng $[-1,1]$ với công thức: $\text{Tanh}(x) = \frac{e^{2x}-1}{e^{2x}+1}$. Nếu càng gần 1 thì giá trị sẽ tích cực, ngược lại thì gần -1 sẽ càng tiêu cực.
- **Hàm Softmax:** hàm này thường được nằm ở lớp ẩn cuối cùng trước lớp đầu ra và không có trường hợp đầu ra không có bởi vì kết quả của hàm này sẽ là kết quả dự đoán cuối cùng. Công thức: $\text{Softmax}(x_j) = \frac{e^{x_j}}{\sum_j e^{x_j}}$

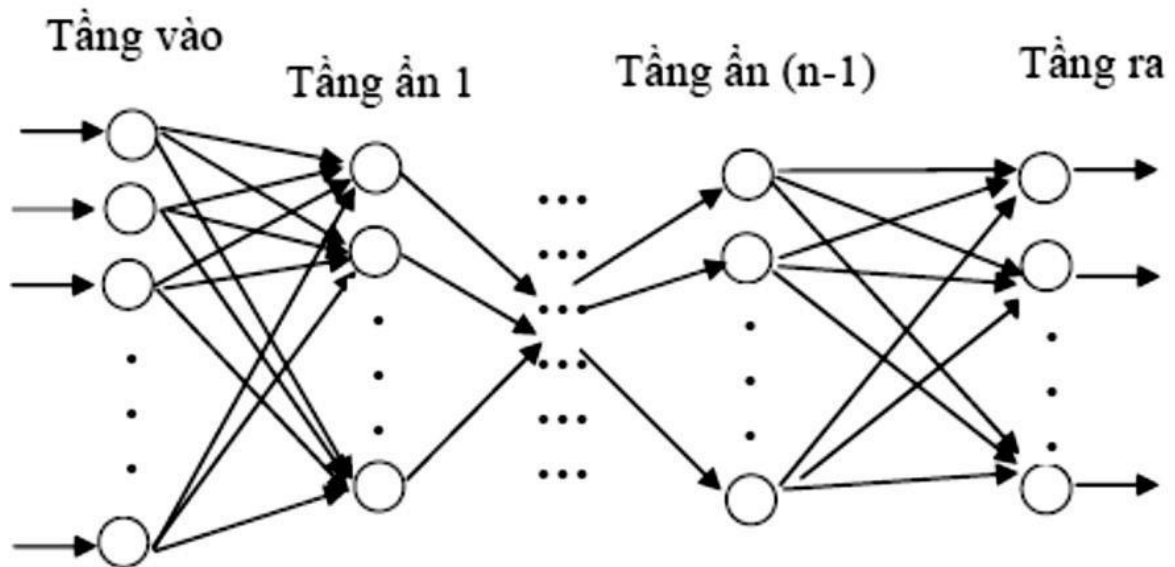
- **Áp dụng mạng lưới vào dữ liệu**

- Đầu tiên cần cấu hình cho mạng lưới ANN với 2 hidden layer (lớp ẩn) lớp đầu tiên với 192 đơn vị nơron và sử dụng hàm kích hoạt là RELU và các đặc trưng của dữ liệu.
- Lớp thứ 2 với 128 đơn vị nơron và sử dụng hàm kích hoạt là RELU.
- Lớp thứ 3 sẽ là lớp đầu ra với 3 phân loại và hàm kích hoạt là Softmax.

2.3. Multi-layer Perceptron (MLP)

2.3.1. Kiến trúc mạng MLP

- Mô hình mạng nơron được sử dụng rộng rãi nhất là mô hình mạng nhiều tầng truyền thẳng (MLP: Multi Layer Perceptron). Một mạng MLP tổng quát là mạng có n ($n \geq 2$) tầng (thông thường tầng đầu vào không được tính đến): trong đó gồm một tầng đầu ra (tầng thứ n) và $(n-1)$ tầng ẩn.



Hình 2.6: Kiến trúc mạng MLP

- **Kiến trúc của một mạng MLP tổng quát có thể mô tả như sau:**

- Đầu vào là các vector (x_1, x_2, \dots, x_p) trong không gian p chiều, đầu ra là các vector (y_1, y_2, \dots, y_q) trong không gian q chiều. Đối với các bài toán phân loại, p chính là kích thước của mẫu đầu vào, q chính là số lớp cần phân loại.
 - Mỗi neural thuộc tầng sau liên kết với tất cả các neuron thuộc tầng liền trước nó.
 - Đầu ra của neural tầng trước là đầu vào của neuron thuộc tầng liền sau nó.
- Hoạt động của mạng MLP như sau: tại tầng đầu vào các neural nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền); kết quả này sẽ được truyền tới các neural thuộc tầng ẩn thứ nhất; các neuron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến tầng ẩn thứ 2. Quá trình tiếp tục cho đến khi các neural thuộc tầng ra cho kết quả.

2.3.2. Huấn luyện mạng MLP

- **Học có giám sát:**

- Là quá trình học có sự tham gia giám sát của một “thầy giáo”. Cũng giống như việc ta dạy một em nhỏ các chữ cái. Ta đưa ra một chữ “a” và bảo với

em đó rằng đây là chữ “a”. Việc này được thực hiện trên tất cả các mẫu chữ cái. Sau đó khi kiểm tra ta sẽ đưa ra một chữ cái bất kì (có thể viết hơi khác đi) và hỏi em đó đây là chữ gì?

- Như vậy với học có giám sát, số lớp cần phân loại đã được biết trước. Nhiệm vụ của thuật toán là phải xác định được một cách thức phân lớp sao cho với mỗi vector đầu vào sẽ được phân loại chính xác vào lớp của nó.

- **Học không giám sát:**

- Là việc học không cần có bất kỳ một sự giám sát nào.
- Trong bài toán học không giám sát, tập dữ liệu huấn luyện được cho dưới dạng: $D = \{(x_1, x_2, \dots, x_N)\}$, với (x_1, x_2, \dots, x_N) là vector đặc trưng của mẫu huấn luyện. Nhiệm vụ của thuật toán là phải phân chia tập dữ liệu D thành các nhóm con, mỗi nhóm chứa các vector đầu vào có đặc trưng giống nhau.
- Như vậy với học không giám sát, số lớp phân loại chưa được biết trước, và tùy theo tiêu chuẩn đánh giá độ tương tự giữa các mẫu mà ta có thể có các lớp phân loại khác nhau.

2.4. Hồi quy logistic

2.4.1. Hồi quy logistic là gì?

- Hồi quy logistic là một phương pháp toán học trong thống kê để dự đoán xác suất dựa trên yếu tố là dữ liệu đầu vào và từ đó dự đoán dữ liệu đầu ra là 1 giá trị nằm trong khoảng $(0,1)$, được áp dụng trong machine learning để dự đoán xác suất của một sự kiện xảy ra dựa trên một hoặc nhiều biến độc lập. Nó là một dạng của mô hình hồi quy được sử dụng chủ yếu trong các bài toán phân loại, nơi mục tiêu là dự đoán xem một quan sát thuộc về một trong một số lớp cố định.
- Trong hồi quy logistic có 2 hàm được dùng để tính toán cho các bài toán phân loại là:
 - Hàm **logistic** và hàm **sigmoid** tuy nhiên công thức của hàm này đều giống nhau và chỉ khác nhau ở ký hiệu.
 - Công thức chung cho hồi quy logistic:

$$P(Y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \dots + b_p X_p)}}$$

Hình 2.7: Công thức hồi quy logistic

2.4.2. Huấn luyện mô hình

- Để huấn luyện được mô hình hồi quy logistic đầu tiên cần phải có dữ liệu đầu vào và cần phải được làm sạch dữ liệu nếu dùng mô hình để dự đoán phân loại.
- Tiếp theo là xây dựng mô hình hồi quy logistic mô hình được xây dựng theo công thức chung của hồi quy logistic.
- Cuối cùng là đánh giá mô hình bằng cách sử dụng phép đo lường recall, F1-score, ROC curve và AUC.

2.5. Cây quyết định (Decision Tree)

2.5.1. Cây quyết định là gì

- Cây quyết định là một cấu trúc cây giống như sơ đồ trong đó mỗi nút bên trong biểu thị tính năng, các nhánh biểu thị các quy tắc và các nút lá biểu thị kết quả của thuật toán.
- Nó là một thuật toán học máy có giám sát linh hoạt, được sử dụng cho cả các vấn đề phân loại và hồi quy. Đây là một trong những thuật toán rất mạnh mẽ. Và nó cũng được sử dụng trong Rừng ngẫu nhiên để huấn luyện trên các tập hợp con dữ liệu huấn luyện khác nhau, điều này khiến rừng ngẫu nhiên trở thành một trong những thuật toán mạnh mẽ nhất trong học máy.

2.5.2. Thuật toán cây quyết định hoạt động như thế nào?

- Cây quyết định hoạt động bằng cách phân tích tập dữ liệu để dự đoán phân loại của nó.
- Nó bắt đầu từ nút gốc của cây, nơi thuật toán xem giá trị của thuộc tính gốc so với thuộc tính của bản ghi trong tập dữ liệu thực tế. Dựa trên sự so sánh, nó tiến hành đi theo nhánh và di chuyển đến nút tiếp theo.

2.5.3. Xây dựng cây quyết định

- Một cây có thể được “học” bằng cách chia tập nguồn thành các tập con dựa trên các Biến pháp lựa chọn thuộc tính. Thước đo lựa chọn thuộc tính (ASM) là một tiêu chí được sử dụng trong thuật toán cây quyết định để đánh giá mức độ hữu ích của các thuộc tính khác nhau trong việc phân chia tập dữ liệu. Mục tiêu của ASM là xác định thuộc tính sẽ tạo ra các tập hợp con dữ liệu đồng nhất nhất sau khi phân chia, từ đó tối đa hóa lợi ích thu được thông tin. Quá trình này được lặp lại trên mỗi tập con dẫn xuất theo cách đệ quy được gọi là phân vùng đệ quy. Quá trình đệ quy được hoàn thành khi tập hợp con tại một nút đều có cùng giá trị của biến mục tiêu hoặc khi việc phân tách không còn thêm giá trị vào dự đoán. Việc xây dựng bộ phân loại cây quyết định không yêu cầu bất kỳ miền kiến thức hoặc thiết lập tham số nào và do đó phù hợp cho việc khám phá kiến thức thăm dò. Cây quyết định có thể xử lý dữ liệu nhiều chiều.
- Thuật toán lặp lại hành động này cho mọi nút tiếp theo bằng cách so sánh các giá trị thuộc tính của nó với giá trị của các nút phụ và tiếp tục quá trình này. Nó lặp đi lặp lại cho đến khi tới nút lá của cây. Cơ chế hoàn chỉnh có thể được giải thích rõ hơn thông qua thuật toán được đưa ra dưới đây.

2.5.4. Các bước xây dựng cây quyết định

- **Bước 1:** Bắt đầu cây với nút gốc, gọi là S, chứa tập dữ liệu hoàn chỉnh.
- **Bước 2:** Tìm thuộc tính tốt nhất trong tập dữ liệu bằng cách sử dụng Thước đo lựa chọn thuộc tính (ASM).
- **Bước 3:** Chia S thành các tập con chứa các giá trị có thể có của các thuộc tính tốt nhất.
- **Bước 4:** Tạo nút cây quyết định chứa thuộc tính tốt nhất.
- **Bước 5:** Tạo đệ quy các cây quyết định mới bằng cách sử dụng các tập hợp con của tập dữ liệu được tạo ở bước -3. Tiếp tục quá trình này cho đến khi đạt đến giai đoạn mà bạn không thể phân loại thêm các nút và gọi nút cuối cùng là thuật toán Phân loại nút lá và Cây hồi quy.

CHƯƠNG 3: GIỚI THIỆU TẬP DỮ LIỆU

3.1. Thông tin tập dữ liệu

- Tập dữ liệu này liên quan đến các biến thể màu đỏ của rượu "Vinho Verde" Bồ Đào Nha. Tập dữ liệu mô tả lượng các chất hóa học khác nhau có mặt trong rượu và tác động của chúng đến chất lượng của rượu. Các lớp được sắp xếp theo thứ tự và không cân bằng (ví dụ, có nhiều rượu bình thường hơn rượu xuất sắc hoặc rượu kém chất lượng).
- Tập dữ liệu gồm **1143** dòng và **13** cột, với **11** biến đầu vào kiểu dữ liệu định lượng liên tục và **1** thuộc tính đầu ra kiểu dữ liệu định lượng rời rạc.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

Hình 3.1 5: dòng dữ liệu

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   fixed acidity                         1143 non-null   float64
1   volatile acidity                      1143 non-null   float64
2   citric acid                          1143 non-null   float64
3   residual sugar                       1143 non-null   float64
4   chlorides                           1143 non-null   float64
5   free sulfur dioxide                  1143 non-null   float64
6   total sulfur dioxide                  1143 non-null   float64
7   density                             1143 non-null   float64
8   pH                                  1143 non-null   float64
9   sulphates                           1143 non-null   float64
10  alcohol                             1143 non-null   float64
11  quality                             1143 non-null   int64
12  Id                                  1143 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

Hình 3.2: Thông tin dữ liệu

3.2. Tóm tắt thống kê dữ liệu

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1143.0	8.311111	1.747595	4.60000	7.10000	7.90000	9.100000	15.90000
volatile acidity	1143.0	0.531339	0.179633	0.12000	0.39250	0.52000	0.640000	1.58000
citric acid	1143.0	0.268364	0.196686	0.00000	0.09000	0.25000	0.420000	1.00000
residual sugar	1143.0	2.532152	1.355917	0.90000	1.90000	2.20000	2.600000	15.50000
chlorides	1143.0	0.086933	0.047267	0.01200	0.07000	0.07900	0.090000	0.61100
free sulfur dioxide	1143.0	15.615486	10.250486	1.00000	7.00000	13.00000	21.000000	68.00000
total sulfur dioxide	1143.0	45.914698	32.782130	6.00000	21.00000	37.00000	61.000000	289.00000
density	1143.0	0.996730	0.001925	0.99007	0.99557	0.99668	0.997845	1.00369
pH	1143.0	3.311015	0.156664	2.74000	3.20500	3.31000	3.400000	4.01000
sulphates	1143.0	0.657708	0.170399	0.33000	0.55000	0.62000	0.730000	2.00000
alcohol	1143.0	10.442111	1.082196	8.40000	9.50000	10.20000	11.100000	14.90000
quality	1143.0	5.657043	0.805824	3.00000	5.00000	6.00000	6.000000	8.00000
Id	1143.0	804.969379	463.997116	0.00000	411.00000	794.00000	1209.500000	1597.00000

Hình 3.3: Bảng tóm tắt dữ liệu

3.3. Thông tin thuộc tính

- **11 thuộc tính đầu vào:**

1. Fixed acidity (g/dm^3)
2. Volatile acidity (g/dm^3)
3. Citric acid (g/dm^3)
4. Residual sugar (g/dm^3)
5. Chlorides (g/dm^3)
6. Free sulfur dioxide (mg/dm^3)
7. Total sulfur dioxide (mg/dm^3)
8. Density (g/cm^3)
9. pH
10. Sulphate (g/dm^3)
11. Alcohol (%)

- **1 thuộc tính đầu ra:**

12. Quality (điểm từ 0 đến 10).

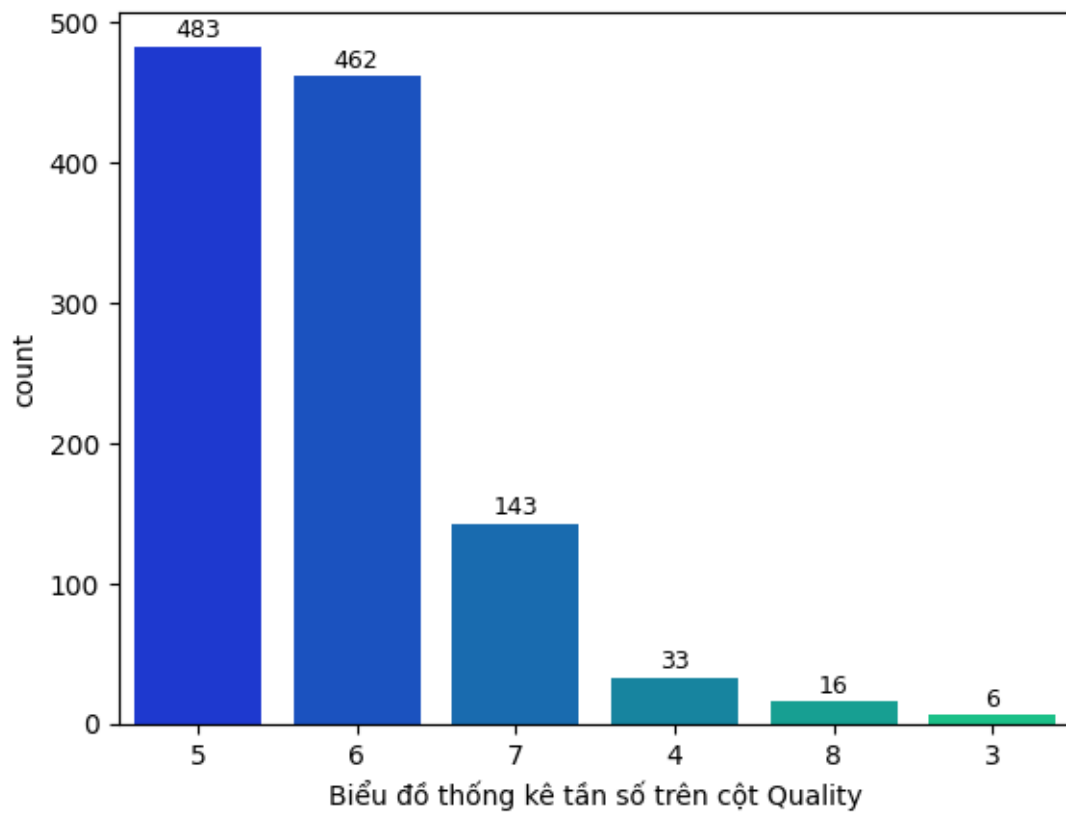
3.4. Mô tả thuộc tính

1. Độ axit cố định (Fixed Acidity): Hầu hết axit trong rượu vang là axit cố định hoặc không bay hơi (không bay hơi dễ dàng).
2. Độ axit bay hơi (Volatile Acidity): Là lượng axit axetic trong rượu vang, ở mức cao có thể gây ra hương vị khó chịu giống như giấm.
3. Axit citric (Citric Acid): Có trong lượng nhỏ, axit citric có thể thêm vào "sự tươi mới" và hương vị cho rượu vang.
4. Đường còn lại (Residual Sugar): Là lượng đường còn lại sau khi quá trình lên men kết thúc; hiếm khi có rượu vang dưới 1 gram/lít và rượu vang có hơn 45 gram/lít được coi là ngọt.
5. Clo (Chlorides): Lượng muối trong rượu vang.
6. Lưu huỳnh dioxide tự do (Free Sulfur Dioxide): Dạng tự do của SO_2 tồn tại trong sự cân bằng giữa SO_2 phân tử (dưới dạng khí tan) và bisulfite ion; ngăn chặn sự phát triển vi khuẩn và sự ô nhiễm của rượu vang.
7. Lưu huỳnh dioxide tổng cộng (Total Sulfur Dioxide): Lượng SO_2 dạng tự do và dạng kết hợp; ở nồng độ thấp, SO_2 hầu như không thể phát hiện được trong rượu vang, nhưng ở nồng độ SO_2 tự do trên 50 ppm, SO_2 trở nên rõ ràng trong mũi và vị của rượu vang.
8. Khối lượng riêng (Density): Khối lượng riêng của nước gần bằng với nước tùy thuộc vào hàm lượng cồn và đường.
9. pH: Mô tả mức độ axit hoặc kiềm của rượu vang trên một thang đo từ 0 (rất axit) đến 14 (rất kiềm); hầu hết rượu vang nằm trong khoảng 3-4 trên thang đo pH.
10. Sulfates: Là một chất phụ gia trong rượu vang có thể góp phần vào việc tạo ra khí sulfur dioxide (SO_2), hoạt động như một chất chống vi khuẩn và chống oxy hóa.
11. Cồn (Alcohol): Phần trăm nồng độ cồn của rượu vang.
12. Chất lượng (Quality): Điểm đánh giá chất lượng rượu vang.

3.5. Thống kê thuộc tính đầu ra

- Quan sát biểu đồ cột có thể thấy, chất lượng rượu đạt điểm 3, 4, 8 chỉ chiếm khoảng 6% trong tập dữ liệu => hiếm.

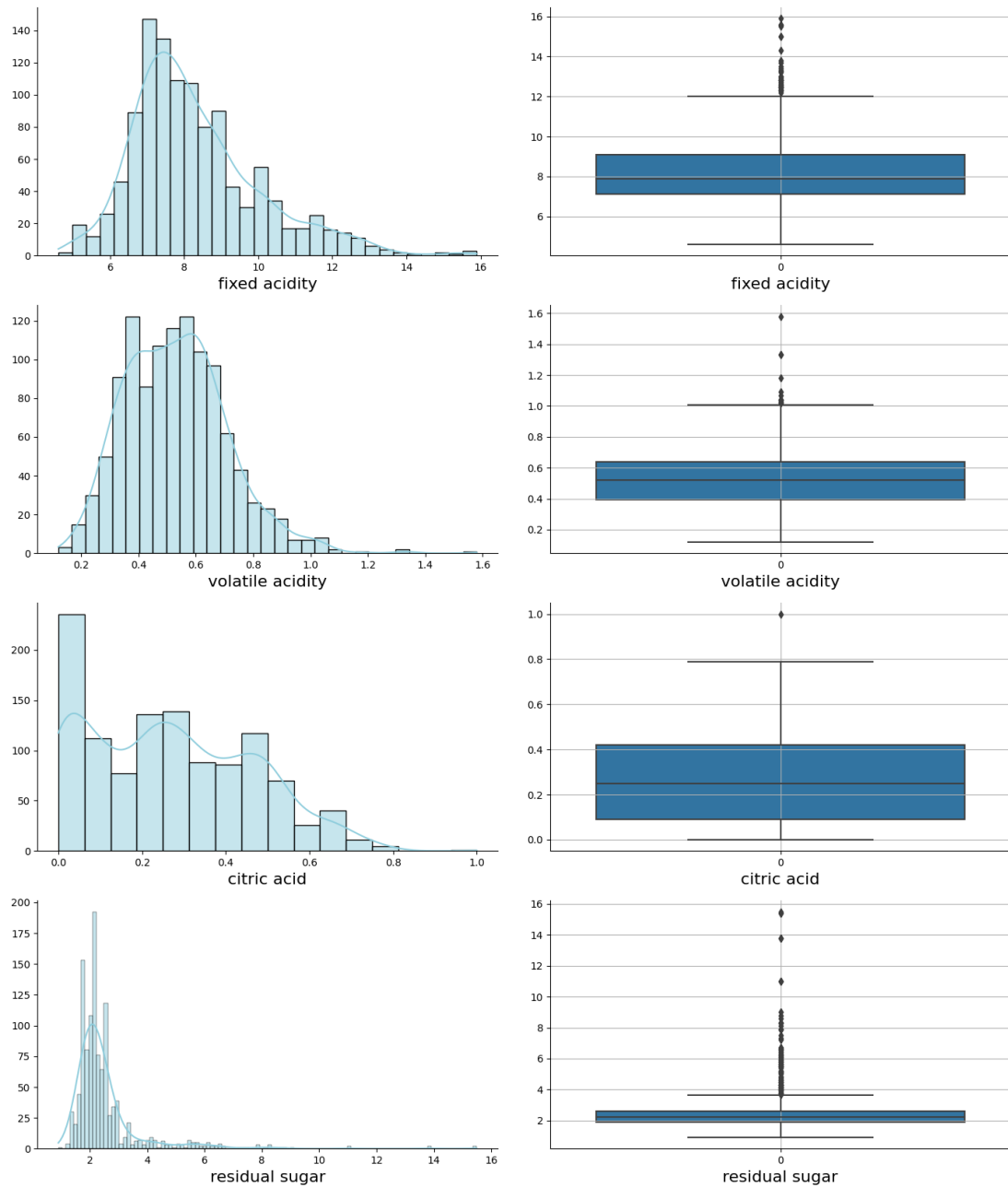
- Phần lớn được phân bố ở mức điểm 5 và 6 (khoảng 80%), dữ liệu đang mất cân bằng.



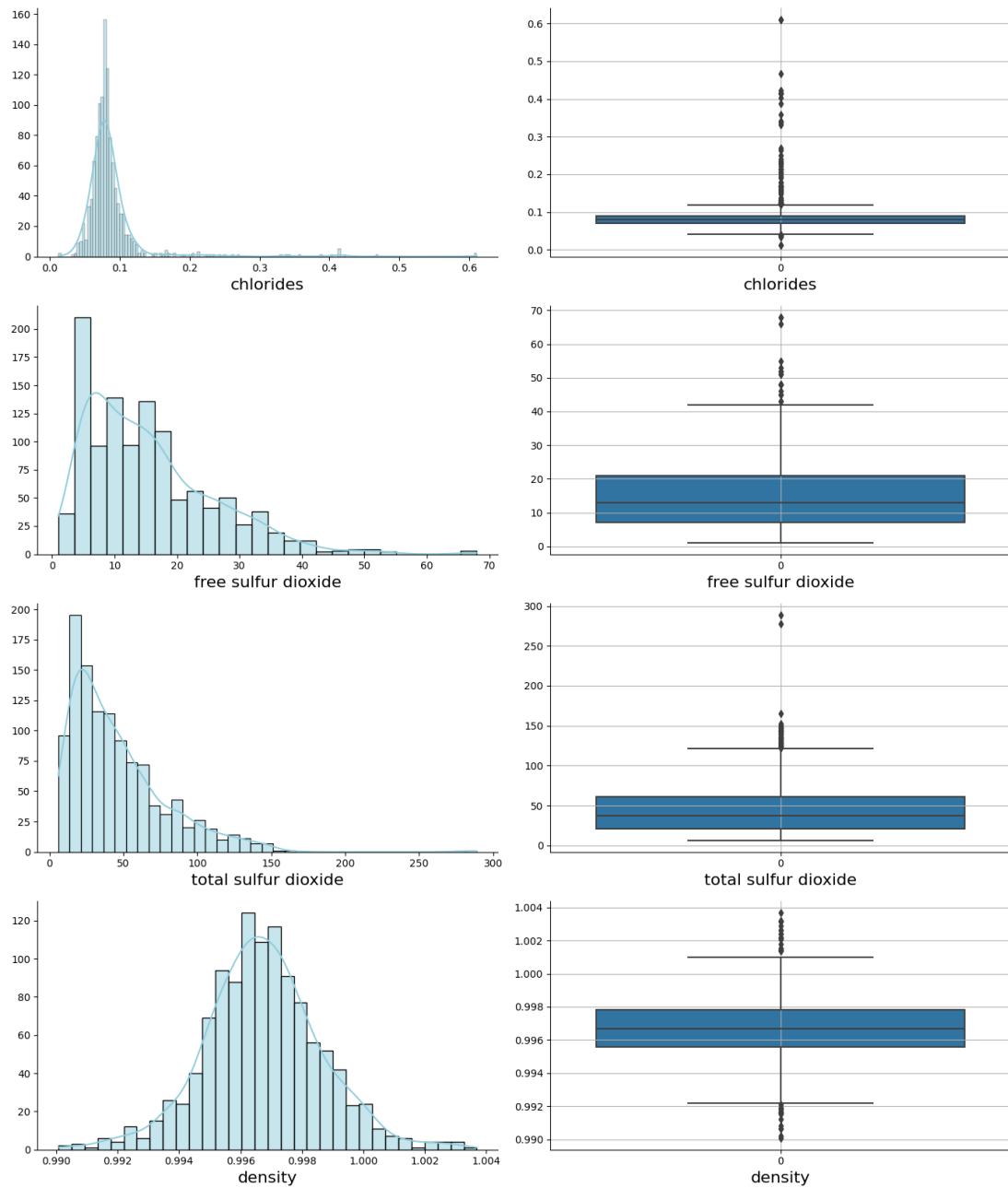
Hình 3.4: Biểu đồ thống kê tần số cột quality

3.6. Biểu đồ histogram và boxplot

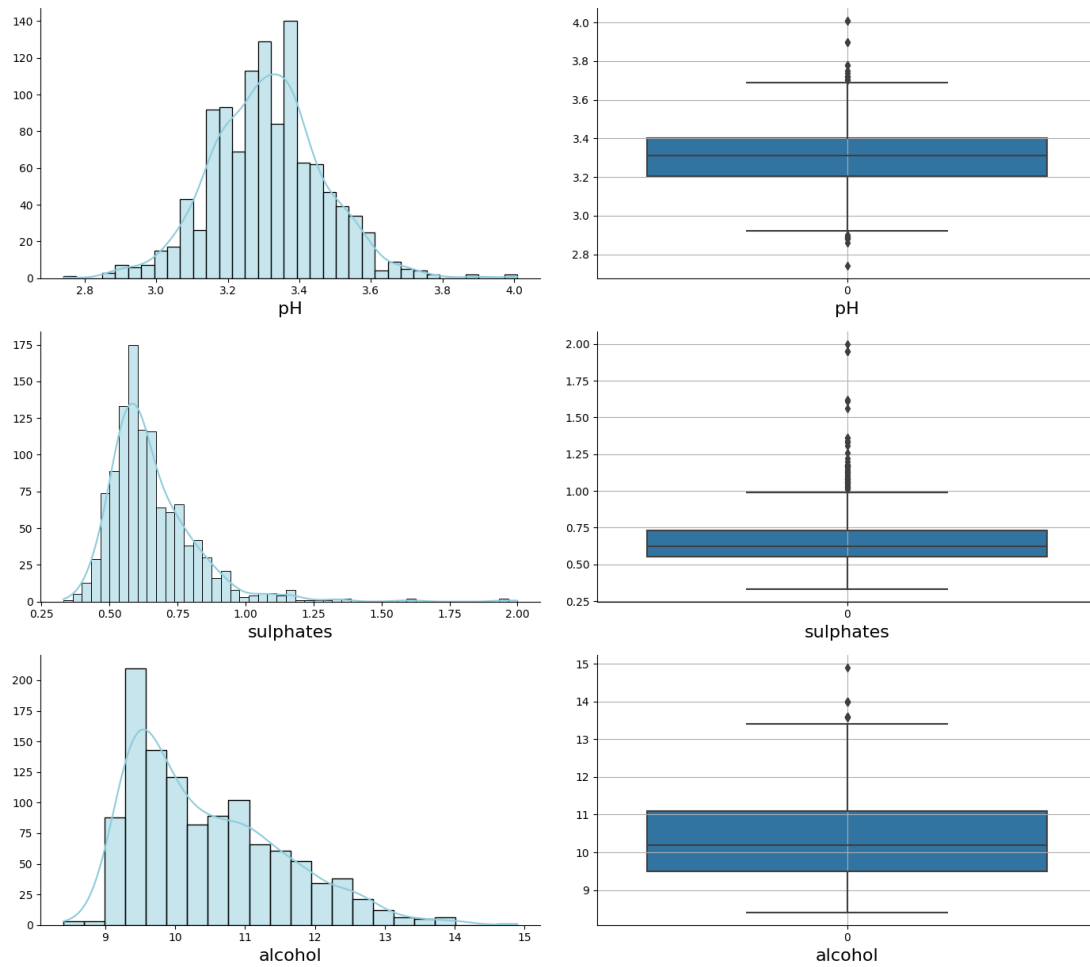
- Dữ liệu được phân phối phần lớn là lệch phải.
- Có nhiều giá trị ngoại lai trong nhiều cột dữ liệu.



Hình 3.5: Biểu đồ histogram và boxplot 1



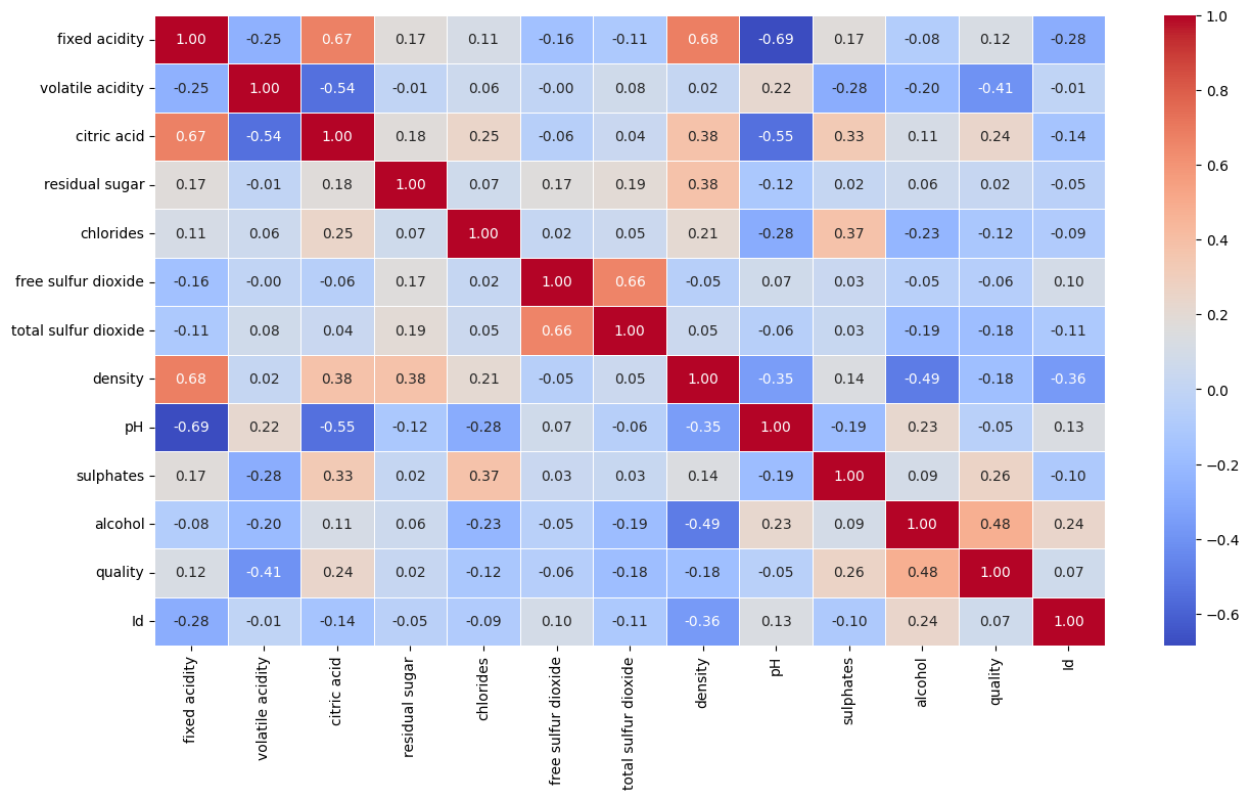
Hình 3.6: Biểu đồ histogram và boxplot 2



Hình 3.7: Biểu đồ histogram và boxplot 3

3.7. Biểu đồ heatmap

- Cột **total sulphur dioxide** và **free sulphur dioxide** có mối tương quan dương cao.
- Các cột **fixed acidity**, **pH** và **citric acid** có mối tương quan cao.
- Hai cột **fixed acidity** và **density** có mối tương quan dương cao.
- Hai cột **alcohol** và **density** cũng có mối tương quan âm cao.
- Chỉ có 2 biến volatile acidity tương quan âm và **alcohol** tương quan dương với **quality**.



Hình 3.8: Biểu đồ heatmap

CHƯƠNG 4: TIỀN XỬ LÝ DỮ LIỆU

4.1. Loại bỏ cột dữ liệu không cần thiết

- Loại bỏ cột Id vì nó không có nhiệm vụ trong việc phân tích và đánh giá mô hình.

```
# Xóa cột Id không cần thiết trong việc phân tích và dự đoán
data = data.drop(columns='Id', axis=1)

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   fixed acidity          1143 non-null   float64
1   volatile acidity       1143 non-null   float64
2   citric acid            1143 non-null   float64
3   residual sugar         1143 non-null   float64
4   chlorides              1143 non-null   float64
5   free sulfur dioxide    1143 non-null   float64
6   total sulfur dioxide   1143 non-null   float64
7   density                1143 non-null   float64
8   pH                    1143 non-null   float64
9   sulphates              1143 non-null   float64
10  alcohol                1143 non-null   float64
11  quality                1143 non-null   int64   
dtypes: float64(11), int64(1)
memory usage: 107.3 KB
```

Hình 4.1: Loại bỏ cột Id


- Sau khi quan sát biểu đồ tương quan ở bước trực quan hóa dữ liệu, ta có thấy hai cột **total sulfur dioxide** và **free sulfur dioxide** có mối tương quan cao và còn có liên quan nhau. Nên ta sẽ tiến hành loại bỏ 1 cột và sẽ có hai trường hợp xử lý như sau:
 - **Trừ:** Nếu quan tâm đến lượng **sulfur dioxide** đã kết hợp trong mẫu rượu và muốn đo lường mức độ tồn tại của **sulfur dioxide** đã kết hợp độc lập với lượng **sulfur dioxide tự do**. Cột mới sẽ biểu thị lượng sulfur dioxide đã kết hợp trong mẫu rượu.

- **Cộng:** Nếu quan tâm đến tổng lượng sulfur dioxide trong mẫu rượu, bao gồm cả lượng **free sulfur dioxide** và lượng **total sulfur dioxide**.

```
[18] # Định nghĩa biến SO2 từ 2 biến total sulfur dioxide và free sulfur dioxide
data['SO2'] = data['total sulfur dioxide'] - data['free sulfur dioxide']
```

```
data = data.drop('total sulfur dioxide', axis=1)
data = data.drop('free sulfur dioxide', axis=1)
```

```
[19] # Chuyển cột 'SO2' lên vị trí trước 'quality'
cols = list(data.columns)
cols.remove('SO2')
cols.insert(cols.index('quality'), 'SO2')
data = data[cols]
```

 data.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1018 entries, 0 to 1142
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   fixed acidity          1018 non-null   float64
1   volatile acidity       1018 non-null   float64
2   citric acid            1018 non-null   float64
3   residual sugar         1018 non-null   float64
4   chlorides              1018 non-null   float64
5   density                1018 non-null   float64
6   pH                    1018 non-null   float64
7   sulphates              1018 non-null   float64
8   alcohol                1018 non-null   float64
9   SO2                   1018 non-null   float64
10  quality                1018 non-null   int64
dtypes: float64(10), int64(1)
memory usage: 95.4 KB
```

Hình 4.2: Tạo cột SO2

4.2. Kiểm tra dữ liệu thiếu và dữ liệu trùng

- Các cột trong tập dữ liệu không có giá trị thiếu.

```
# Kiểm tra giá trị thiếu
data.isnull().sum()

fixed acidity      0
volatile acidity   0
citric acid        0
residual sugar     0
chlorides          0
free sulfur dioxide 0
total sulfur dioxide 0
density           0
pH               0
sulphates         0
alcohol           0
quality           0
dtype: int64
```

Hình 4.3: Kiểm tra dữ liệu thiếu

- Tập dữ liệu xuất hiện đến 240 dòng dữ liệu trùng lặp nhau.

Số lượng dữ liệu bị trùng lặp: 125

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
4	7.4	0.700	0.00	1.90	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
46	7.2	0.725	0.05	4.65	0.086	4.0	11.0	0.99620	3.41	0.39	10.9	5
64	8.6	0.490	0.28	1.90	0.110	20.0	136.0	0.99720	2.93	1.95	9.9	6
65	7.7	0.490	0.26	1.90	0.062	9.0	31.0	0.99660	3.39	0.64	9.6	5
71	8.1	0.545	0.18	1.90	0.080	13.0	35.0	0.99720	3.30	0.59	9.0	6
...
1076	7.5	0.380	0.57	2.30	0.106	5.0	12.0	0.99605	3.36	0.55	11.4	6
1113	7.8	0.600	0.26	2.00	0.080	31.0	131.0	0.99622	3.21	0.52	9.9	5
1114	7.8	0.600	0.26	2.00	0.080	31.0	131.0	0.99622	3.21	0.52	9.9	5
1116	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1119	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5

125 rows x 12 columns

Hình 4.4: Kiểm tra dữ liệu trùng

- Thực loại bỏ những dòng dữ liệu trùng lặp ra khỏi tập dữ liệu.

```
# Xoá các dòng trùng lặp
data = data.drop_duplicates()
print("Số lượng dữ liệu bị trùng lặp: ",data.duplicated().sum())

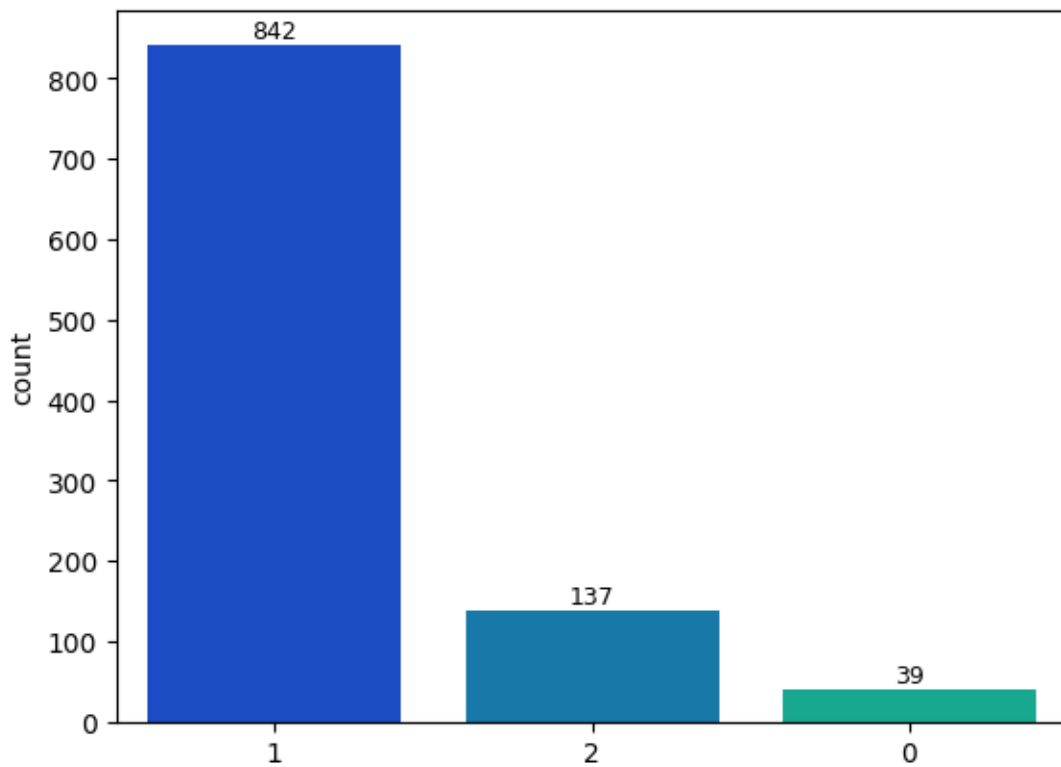
Số lượng dữ liệu bị trùng lặp: 0
```

Hình 4.5: Xóa dữ liệu trùng

4.3. Gom nhóm dữ liệu

Để giảm độ phức tạp cho việc xây dựng và dự đoán mô hình, tôi thực hiện chia nhãn đầu ra quality thành 3 nhóm chính:

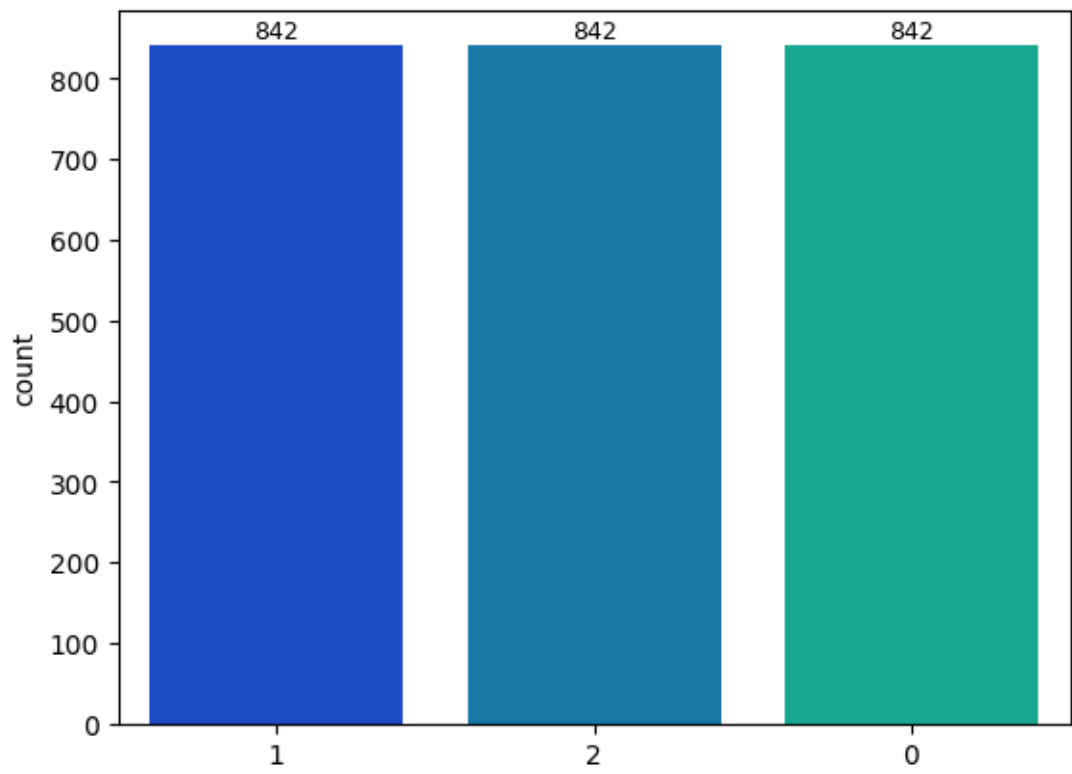
- **Nhóm 1:** Chất lượng rượu vang từ 3 đến 4.
- **Nhóm 2:** Chất lượng rượu vang từ 5 đến 6.
- **Nhóm 3:** Chất lượng rượu vang từ 7 đến 8.



Hình 4.6: Gom nhóm dữ liệu

4.4. Cân bằng dữ liệu

Sau khi gom dữ liệu thành 3 nhóm, nhận thấy rằng nhóm 2 và 0 rơi vào dữ liệu thiếu số so với nhóm 1. Ta sẽ tiến hành cân bằng chúng.



Hình 4.7: Cân bằng dữ liệu

CHƯƠNG 5: LỰA CHỌN ĐẶC TRƯNG

5.1. Chia dữ liệu train và test

- Thực hiện chia tập dữ liệu thành 2 tập riêng biệt với 80% là dùng cho tập huấn luyện và 20% dùng cho tập kiểm tra.
- Chuẩn hóa dữ liệu để loại bỏ giá trị ngoại lai và đưa dữ liệu về cùng phạm vi, sau khi chuẩn hóa thực hiện gán lại nhãn cho các thuộc tính.

```
# Lấy danh sách tên cột
def split_data(dt):
    labels = list(dt.iloc[:, :-1].columns)

    # Tách tập dữ liệu thành train và test
    X_train, X_test, y_train, y_test = train_test_split(dt[labels], dt['quality'], test_size=0.2, random_state=42)

    # Chuẩn hóa dữ liệu
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    # Gán lại nhãn cho dữ liệu train và test
    X_train = pd.DataFrame(X_train, columns=labels)
    X_test = pd.DataFrame(X_test, columns=labels)

    return X_train, X_test, y_train, y_test

X_train, X_test, y_train, y_test = split_data(data)
```

Hình 5.1: Chia dữ liệu train và test

5.2. Mô hình cây quyết định

- Đầu tiên sử dụng thư viện GridSearchCV cho cây quyết định để tìm ra max_depth tối ưu nhất (dựa trên accuracy), với cross_validation là 4 folds. Kết quả độ sâu (max_depth) tối ưu nhất sẽ là 16.

```
# Tìm độ sâu tối ưu cho cây quyết định
dt = DecisionTreeClassifier(random_state=42)
n_max = int(math.log2(len(X_train)))
params = {'max_depth': [1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, n_max]}
cv = GridSearchCV(dt, param_grid=params, scoring='accuracy', cv=4, return_train_score=True)
cv.fit(X_train, y_train)

> GridSearchCV
> estimator: DecisionTreeClassifier
  > DecisionTreeClassifier

depth = cv.best_params_['max_depth']
depth

16
```

Hình 5.2: Tìm độ sâu tối ưu

- Sau khi đã có độ sâu tối ưu ta tiến hành đi xây dựng cây quyết định để tìm ra độ quan trọng của mỗi biến.

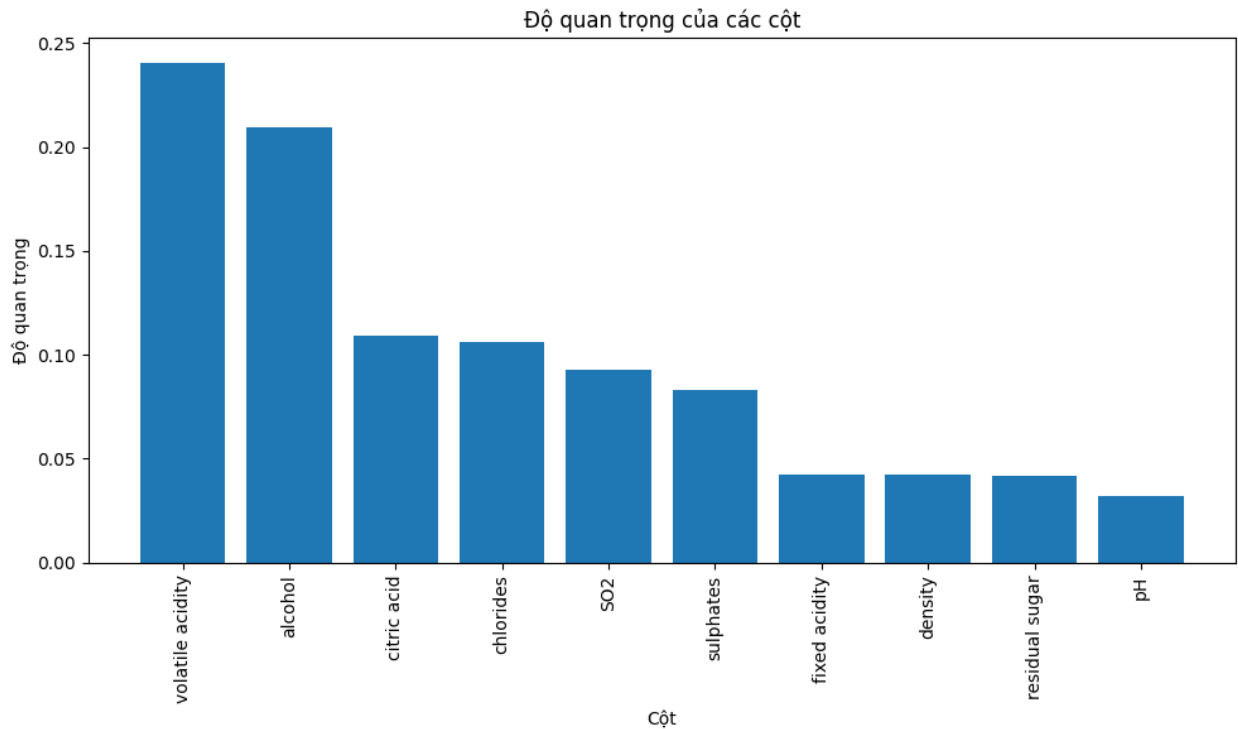
```
# Xử dụng mô hình để tìm biến quan trọng
dt = DecisionTreeClassifier(max_depth= depth, random_state=42)
dt.fit(X_train, y_train)
score = dt.feature_importances_
# Hiển thị cột của từng độ quan trọng
for feature, importance in zip(X.columns, score):
    print(f'Cột {feature}: {importance}')

# Trực quan hóa độ quan trọng của các cột
indices = np.argsort(score)[::-1]
sorted_features = [X_train.columns[i] for i in indices]
sorted_importances = score[indices]

plt.figure(figsize=(10, 6))
plt.bar(range(len(sorted_importances)), sorted_importances, align='center')
plt.xticks(range(len(sorted_importances)), sorted_features, rotation=90)
plt.xlabel('Cột')
plt.ylabel('Độ quan trọng')
plt.title('Độ quan trọng của các cột')
plt.tight_layout()
plt.show()
```

Hình 5.3: Xây dựng mô hình cây quyết định

- Quan sát biểu đồ ta có thể thấy 2 thuộc tính alcohol và volatile acidity có độ quan trọng cao nhất trong việc xây dựng và dự đoán mô hình. Còn 4 thuộc tính có độ quan trọng thấp nhất là residual sugar, fixed acidity, pH và density.




Hình 5.4: Độ quan trọng của các cột

- Ta sẽ tiến hành loại bỏ 4 cột có độ quan trọng thấp nhất, nhằm giảm độ phức tạp khi xây dựng mô hình.

```
[130] # Xóa 4 cột có độ quan trọng thấp nhất
sorted_indices = np.argsort(score)[:4]
lowest_columns = X.columns[sorted_indices]

for column in lowest_columns:
    del data[column]
```

 data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2526 entries, 0 to 2525
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   volatile acidity      2526 non-null   float64
1   citric acid           2526 non-null   float64
2   chlorides             2526 non-null   float64
3   sulphates             2526 non-null   float64
4   alcohol               2526 non-null   float64
5   SO2                  2526 non-null   float64
6   quality               2526 non-null   int64
dtypes: float64(6), int64(1)
memory usage: 138.3 KB
```

Hình 5.5: Xóa thuộc tính không quan trọng

- Dữ liệu lúc này chỉ còn lại 7 cột với 6 biến giải thích và 1 biến phản hồi.
- Thực hiện tách dữ liệu và chuẩn hóa dữ liệu lần 2.

```
data.head()
```

	volatile acidity	citric acid	chlorides	sulphates	alcohol	SO2	quality
0	0.70	0.00	0.076	0.56	9.4	23.0	1
1	0.88	0.00	0.098	0.68	9.8	42.0	1
2	0.76	0.04	0.092	0.65	9.8	39.0	1
3	0.28	0.56	0.075	0.58	9.8	43.0	1
4	0.66	0.00	0.075	0.56	9.4	27.0	1

```
X_train, X_test, y_train, y_test = split_data(data)
```

Hình 5.6: Chia dữ liệu train và test lần 2

5.3. Xây dựng mô hình logistic

- Có tất cả 6 biến giải thích, mã hóa 6 biến này thành 6 con số nhị phân. Số 1 tương ứng chọn thuộc tính, 0 tương ứng không chọn. Mỗi cặp 6 con số được gọi là 1 mã mô hình.
- Sử dụng Random để tạo ra 64 mã mô hình khác nhau, ở đây 64 mô hình vì có 6 thuộc tính và 2 giá trị 0 và 1.

```
# Xác định các thuộc tính đầu vào
features = list(data.iloc[:, :-1].columns)

# Tạo 64 mô hình nhị phân khác nhau
np.random.seed(10)
random_model = np.random.choice([0, 1], size=(64, 6))

# Loại bỏ các hàng có toàn giá trị 0
random_model = random_model[~np.all(random_model == 0, axis=1)]
```

```
random_model
```

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

Hình 5.7: Tạo mô hình ngẫu nhiên

- Ứng với mỗi mã mô hình, xây dựng mô hình hồi quy logistic tương ứng. Vì đầu ra có 3 lớp, nên tôi sẽ sử dụng logistic đa lớp.
- Thực hiện xây dựng 64 mô hình logistic và đánh giá độ chính xác, lưu vào danh sách.

```
# Xây dựng mô hình hồi quy logistic cho từng mã mô hình
accuracies = []
# Mã hóa lại các biến được chọn
for code in random_model:
    selected_features = [features[i] for i in range(6) if code[i] == 1]

    X_train_df = pd.DataFrame(X_train, columns=features)
    X_test_df = pd.DataFrame(X_test, columns=features)

    X_train_selected = X_train_df.loc[:, selected_features]
    X_test_selected = X_test_df.loc[:, selected_features]

    # Áp dụng mô hình Logistic
    model = LogisticRegression(multi_class='multinomial')
    model.fit(X_train_selected, y_train)

    # Dự đoán và đánh giá độ chính xác accuracy
    y_pred = model.predict(X_test_selected)
    accuracy = accuracy_score(y_test, y_pred)

    # Lưu model và accuracy của 64 mô hình vào danh sách
    accuracies.append(accuracy)
```

Hình 5.8: Xây dựng mô hình logistic

- Sau khi xây dựng 64 mô hình logistic và có được độ chính xác (accuracy), tôi sẽ lấy ra mô hình có độ chính xác tốt nhất.
- Mô hình có độ chính xác cao nhất với 4 biến: 'volatile acidity', 'chlorides', 'sulphates', 'alcohol'. Tôi sẽ sử dụng 4 biến này để bắt đầu xây dựng mô hình mạng học sâu.

```
# Chọn mô hình có độ chính xác cao nhất từ các mô hình cuối cùng
best_final_model_index = np.argmax(accuracies)

best_final_model_code = random_model[best_final_model_index]

variables = [features[i] for i in range(6) if best_final_model_code[i] == 1]
print("Mô hình tốt nhất\n",variables)
print("Accuracy: ",random_model[best_final_model_index])
```

Mô hình tốt nhất
['volatile acidity', 'chlorides', 'sulphates', 'alcohol']
Accuracy: [1 0 1 1 1 0]

Hình 5.9: Chọn mô hình logistic tốt nhất

- Cuối cùng ta đã chọn lọc ra được 4 thuộc tính có ảnh hưởng nhất đến chất lượng rượu vang.
- Thực hiện chia tập dữ liệu train và test, sau đó mã hóa lần cuối.
- Lưu lại trình chuẩn hóa để sử dụng khi dự đoán dữ liệu sau này.

```
# Thêm cột quality
variables.append('quality')
data = data.loc[:,variables]
```

```
data.head()
```

	volatile acidity	chlorides	sulphates	alcohol	quality
0	0.70	0.076	0.56	9.4	1
1	0.88	0.098	0.68	9.8	1
2	0.76	0.092	0.65	9.8	1
3	0.28	0.075	0.58	9.8	1
4	0.66	0.075	0.56	9.4	1

Hình 5.10: Tập dữ liệu với 4 biến đặc trưng

```
# Lấy danh sách tên cột của 4 biến giải thích
labels = list(data.iloc[:, :-1].columns)

# Tách tập dữ liệu thành train và test
X_train, X_test, y_train, y_test = train_test_split(data[labels], data['quality'], test_size=0.2, random_state=42)

# Chuẩn hóa dữ liệu
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Lưu trình biến đổi
joblib.dump(sc, "scaler.pkl")

['scaler.pkl']
```

Hình 5.11: Chia dữ liệu train và test lần cuối

CHƯƠNG 6: MÔ HÌNH MẠNG HỌC SÂU

6.1. Mô hình MLP

- `hidden_layer_sizes = (100, 100)`: Đây là kích thước của các tầng ẩn trong mạng nơ-ron. Trong trường hợp này, có hai tầng ẩn, mỗi tầng có 100 nơ-ron.
- `activation='relu'`: Hàm kích hoạt được sử dụng trong các tầng ẩn là Rectified Linear Unit (ReLU).
- `max_iter = 1000`: Số lượng vòng lặp tối đa cho quá trình huấn luyện (epochs).

```
# Xây dựng mô hình MLP
mlp = MLPClassifier(hidden_layer_sizes=(100, 100), activation='relu', max_iter = 1000, random_state=42)
# Huấn luyện mô hình
mlp.fit(X_train, y_train)
# Dự đoán nhãn cho tập kiểm tra
mlp_y_pred = mlp.predict(X_test)
```

Hình 6.1: Mô hình MLP

6.2. Mô hình ANN

- Đầu tiên cần cấu hình cho mạng lưới ANN với 2 hidden layer (lớp ẩn) lớp đầu tiên với 192 đơn vị nơron và sử dụng hàm kích hoạt là RELU và các đặc trưng của dữ liệu.
- Lớp thứ 2 với 128 đơn vị nơron và sử dụng hàm kích hoạt là RELU.
- Lớp thứ 3 sẽ là lớp đầu ra với 3 phân loại và hàm kích hoạt là Softmax.

```
#cấu hình mạng lưới nơ ron nhân tạo
ann = Sequential([Dense(192, activation="relu", input_shape=(X_train.shape[1],)),
                  Dense(128, activation="relu"),
                  Dense(3, activation="softmax")])
```

Hình 6.2: Mô hình ANN

- Tiếp đến là biên dịch mạng lưới ANN với optimizer là 'adam' để tối ưu hóa trong việc điều chỉnh trọng số của mạng lưới trong quá trình huấn luyện.
- Với loss là 'sparse_categorical_crossentropy' phù hợp cho việc bài phân loại đa lớp.
- Accuracy để hiển thị độ chính xác cho mỗi lần huấn luyện.

- Phần huấn luyện sẽ cho 50 huấn luyện mạng lưới và bắt đầu huấn luyện mô hình.

```
#biên dịch mạng lưới
ann.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
#huấn luyện mạng lưới
model_ann = ann.fit(X_train,y_train,epochs=50,validation_data=(X_test, y_test))
```

Hình 6.3: Biên dịch mô hình ANN

- Sử dụng hàm argmax từ thư viện NumPy để chuyển đổi đầu ra dự đoán từ dạng xác suất (hoặc giá trị liên tục) thành nhãn dự đoán.

```
# Dự đoán nhãn từ dữ liệu kiểm thử
y_pred = ann.predict(X_test)

# Chuyển đổi đầu ra dự đoán thành nhãn
ann_y_pred = np.argmax(y_pred, axis=1)
```

Hình 6.4: Chuyển đổi đầu ra mô hình ANN

6.3. Mô hình RNN

- Dòng đầu tiên tạo một đối tượng Sequential mô hình RNN. Sequential cho phép xây dựng một mạng nơ-ron tuần tự bằng cách thêm các layer lần lượt.
- Dòng thứ hai thêm một layer LSTM vào mạng RNN. LSTM là một biến thể của RNN được sử dụng để xử lý dữ liệu tuần tự và giải quyết vấn đề mất mát thông tin dài hạn. Đối số đầu tiên của LSTM là số lượng đơn vị (units) trong layer, ở đây là 32. Đối số input_shape xác định kích thước đầu vào của layer, trong trường hợp này là (X_train.shape[1], 1) để phù hợp với dữ liệu đầu vào.
- Dòng thứ ba thêm một layer Dropout vào mạng. Dropout là một kỹ thuật chống quá khớp (overfitting) bằng cách loại bỏ ngẫu nhiên một phần các đơn vị trong quá trình huấn luyện. Giá trị 0.2 cho đối số Dropout chỉ ra rằng 20% các đơn vị sẽ bị loại bỏ.
- Dòng thứ tư thêm một layer Dense vào mạng. Dense là một layer kết nối đầy đủ trong mạng nơ-ron với các đơn vị tính toán và kích hoạt tương ứng. Ở đây, Dense có 3 đơn vị và kích hoạt tuyến tính (linear).
- Dòng thứ năm biên dịch mô hình. Đối số loss='mse' chỉ ra rằng hàm mất mát (loss function) được sử dụng là mean squared error (MSE), tức là sai số bình phương

trung bình giữa đầu ra thực tế và dự đoán. Đối số optimizer='adam' xác định thuật toán tối ưu hóa được sử dụng là Adam optimizer. Đối số metrics=['accuracy'] chỉ ra rằng độ chính xác sẽ được tính để đánh giá mô hình.

- Dòng cuối cùng huấn luyện mô hình, fit () là phương thức huấn luyện trong Keras. Đối số X_train và y_train là dữ liệu huấn luyện. Đối số epochs=50 chỉ ra rằng quá trình huấn luyện sẽ chạy qua tập dữ liệu huấn luyện 50 lần. Đối số batch_size=32 chỉ ra rằng dữ liệu huấn luyện sẽ được chia thành các batch có kích thước 32 để cập nhật trọng số. Đối số validation_data = (X_test, y_test) chỉ ra rằng dữ liệu kiểm tra (X_test và y_test) sẽ được sử dụng để đánh giá mô hình trong quá trình huấn luyện.

```
#Xây dựng mô hình RNN

rnn = tf.keras.Sequential()
rnn.add(tf.keras.layers.LSTM(32, input_shape=(X_train.shape[1],1)))
rnn.add(tf.keras.layers.Dropout(0.2))
rnn.add(tf.keras.layers.Dense(3, activation='linear'))

# Biên dịch mô hình
rnn.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

# Huấn luyện mô hình
rnn.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
```

Hình 6.5: Mô hình RNN

- Sử dụng hàm argmax từ thư viện NumPy để chuyển đổi đầu ra dự đoán từ dạng xác suất (hoặc giá trị liên tục) thành nhãn dự đoán.

```
# Dự đoán nhãn từ dữ liệu kiểm thử
y_pred1 = rnn.predict(X_test)

# Chuyển đổi đầu ra dự đoán thành nhãn
rnn_y_pred = np.argmax(y_pred1, axis=1)
```

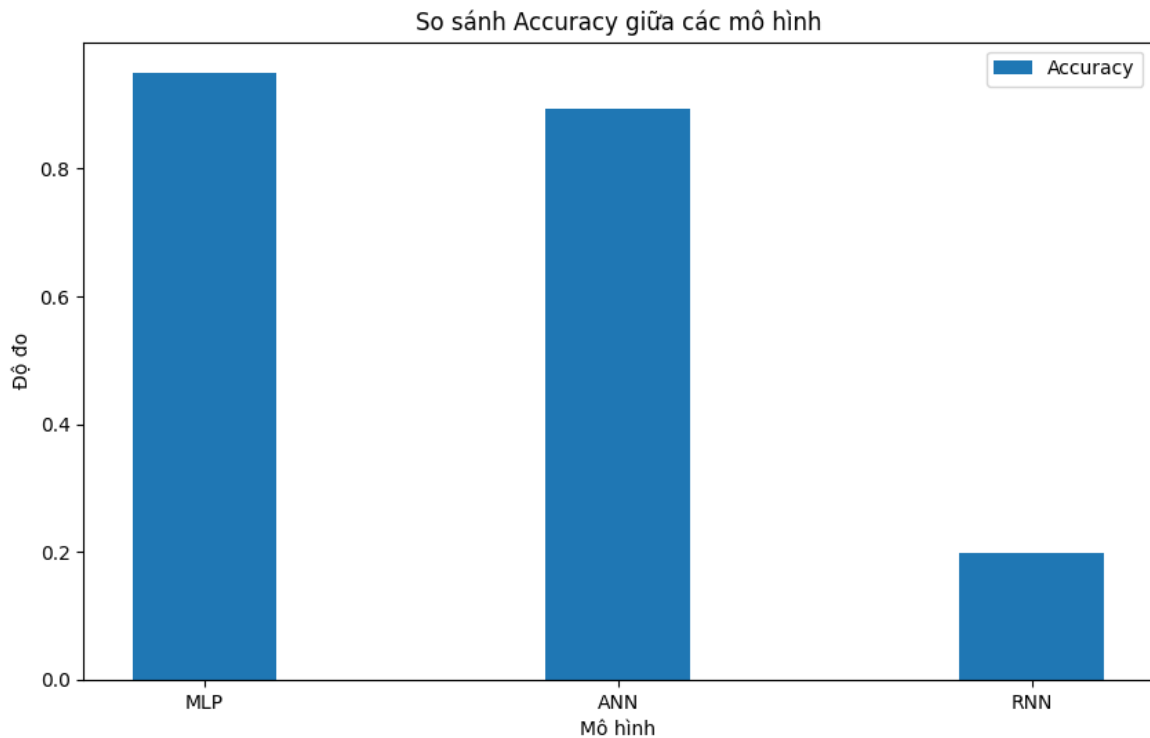
Hình 6.6: Chuyển đổi đầu ra mô hình RNN

6.4. So sánh mô hình

6.4.1. So sánh độ chính xác

- Đầu tiên ta sẽ đi so sánh độ chính xác (accuracy) của 3 mô hình.

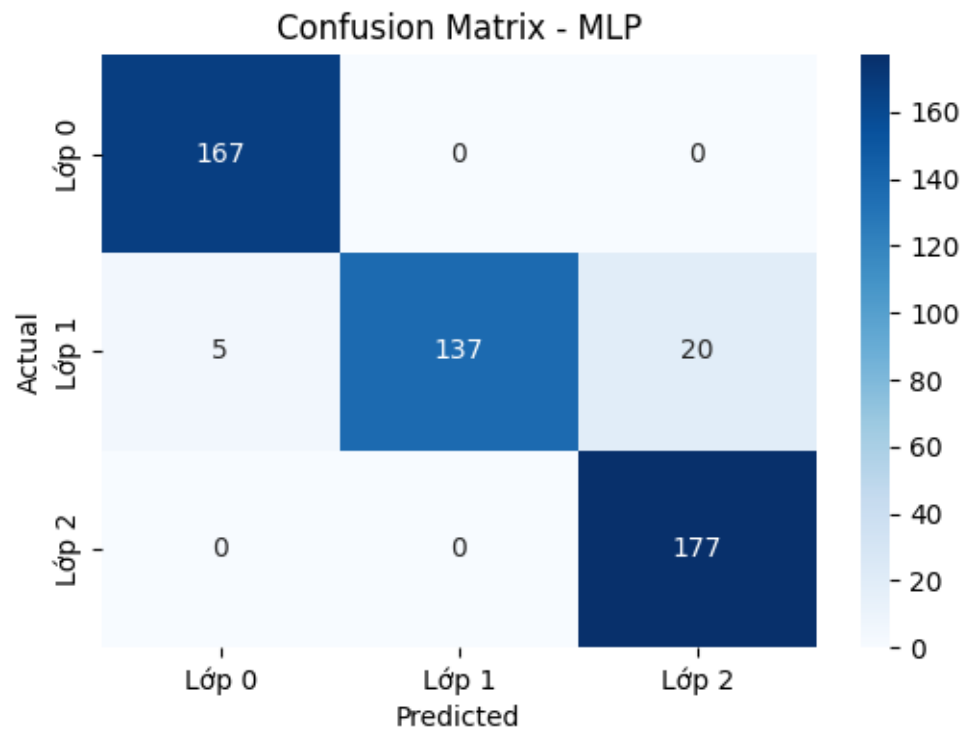
- Quan sát biểu đồ, ta dễ dàng thấy được mô hình MLP có độ chính xác cao nhất, mô hình ANN chỉ thấp hơn 1 tí và thấp nhất là mô hình RNN.



Hình 6.7: So sánh Accuracy giữa các mô hình

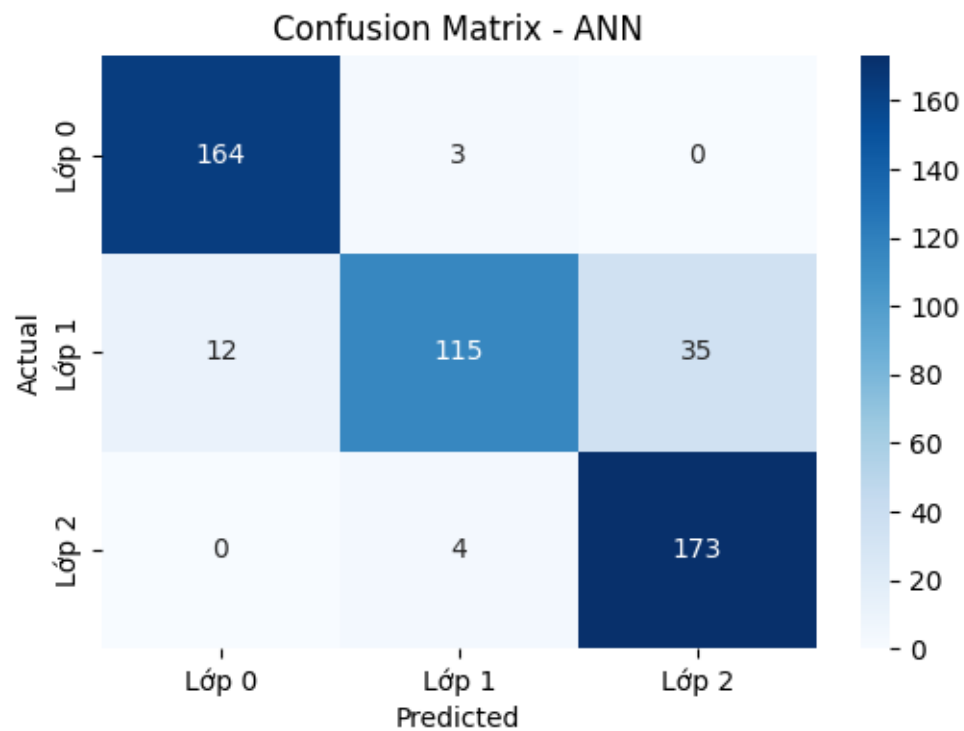
6.4.2. So sánh ma trận nhầm lẫn (confusion matrix)

- Quan sát ma trận nhầm lẫn (confusion matrix) của mô hình MLP, ở lớp 0 và lớp 2 mô hình dự đoán hoàn toàn chính xác các giá trị. Nhưng lớp 1 có 5 giá trị bị dự đoán sai vào lớp 0 và 20 giá trị sai vào lớp 2.



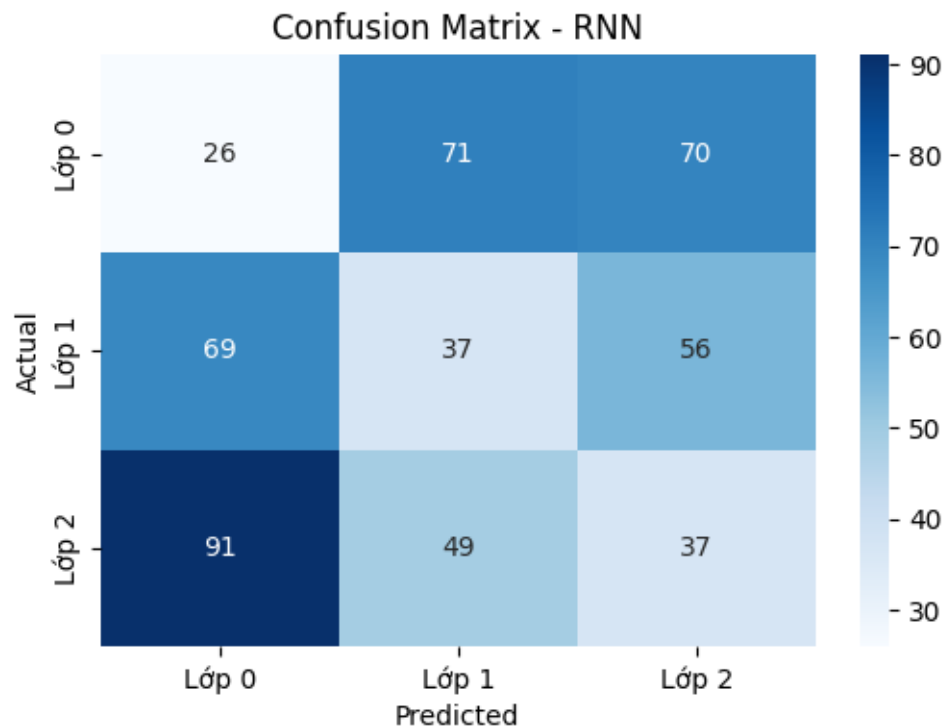
Hình 6.8: Confusion Matrix – MLP

- Với mô hình ANN, ở lớp 1 mô hình dự đoán sai nhiều hơn hẳn 2 lớp còn lại.
- Ở lớp 0 và lớp 1 dự đoán sai khá ít, có thể cải thiện bằng việc điều chỉnh thông số.



Hình 6.9: Confusion Matrix – ANN

- Cuối cùng là mô hình RNN, ở cả 3 lớp thì tỉ lệ dự đoán sai là khá nhiều, hiệu suất rất kém.



Hình 6.10: Confusion Matrix – RNN

6.4.3. Báo cáo phân loại

- **Mô hình MLP:**
 - Accuracy cao: 95%.
 - Precision, recall, và f1-score đều khá cao đối với các lớp 0 và 2, nhưng lớp 1 có recall thấp hơn (0.85).
 - Macro avg và weighted avg đều khá cao, cho thấy mô hình hoạt động tốt.
- **Mô hình ANN:**
 - Accuracy thấp hơn so với MLP: 89%.
 - Precision và recall đều thấp hơn đáng kể đối với lớp 1, cho thấy mô hình có khả năng nhận diện lớp này kém hơn.
 - Tuy nhiên, mô hình vẫn còn khá tốt đối với các lớp khác.
- **Mô hình RNN:**
 - Accuracy thấp nhất: 20%.

- Precision, recall, và f1-score đều thấp đối với tất cả các lớp.
- Mô hình RNN không hiệu quả trong việc phân loại, có thể cần xem xét lại cấu trúc mô hình hoặc các tham số.

Báo cáo phân loại MLP:					
	precision	recall	f1-score	support	
0	0.97	1.00	0.99	167	
1	1.00	0.85	0.92	162	
2	0.90	1.00	0.95	177	
accuracy			0.95	506	
macro avg	0.96	0.95	0.95	506	
weighted avg	0.95	0.95	0.95	506	
Báo cáo phân loại ANN:					
	precision	recall	f1-score	support	
0	0.93	0.98	0.96	167	
1	0.94	0.71	0.81	162	
2	0.83	0.98	0.90	177	
accuracy			0.89	506	
macro avg	0.90	0.89	0.89	506	
weighted avg	0.90	0.89	0.89	506	
Báo cáo phân loại RNN:					
	precision	recall	f1-score	support	
0	0.14	0.16	0.15	167	
1	0.24	0.23	0.23	162	
2	0.23	0.21	0.22	177	
accuracy			0.20	506	
macro avg	0.20	0.20	0.20	506	
weighted avg	0.20	0.20	0.20	506	

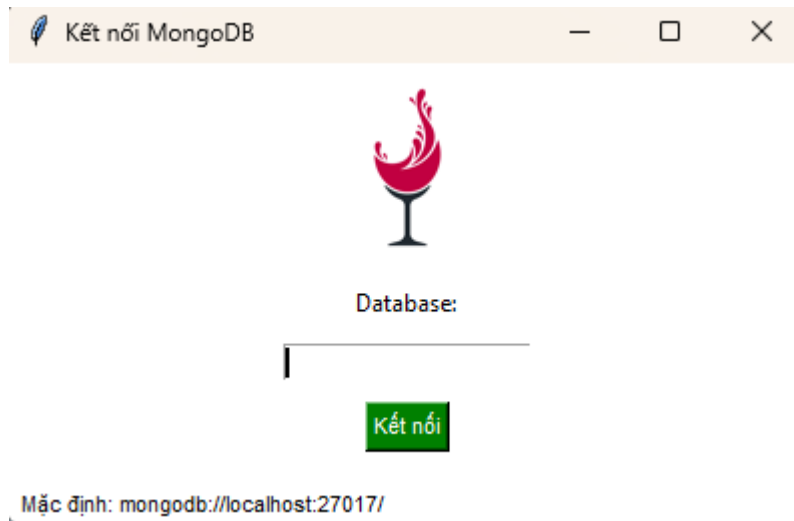
Hình 6.11: Báo cáo phân loại các mô hình

- **Nhận xét chung:** Mô hình MLP có vẻ phù hợp và hiệu quả nhất trong ba mô hình này, vì có accuracy cao và độ chính xác tốt trên nhiều lớp. Mô hình ANN cũng tốt, nhưng cần xem xét cải thiện độ nhận diện của lớp 1. Mô hình RNN hiện không phù hợp cho bài toán này, với accuracy thấp và độ chính xác kém.

CHƯƠNG 7: GIAO DIỆN CHỨC NĂNG

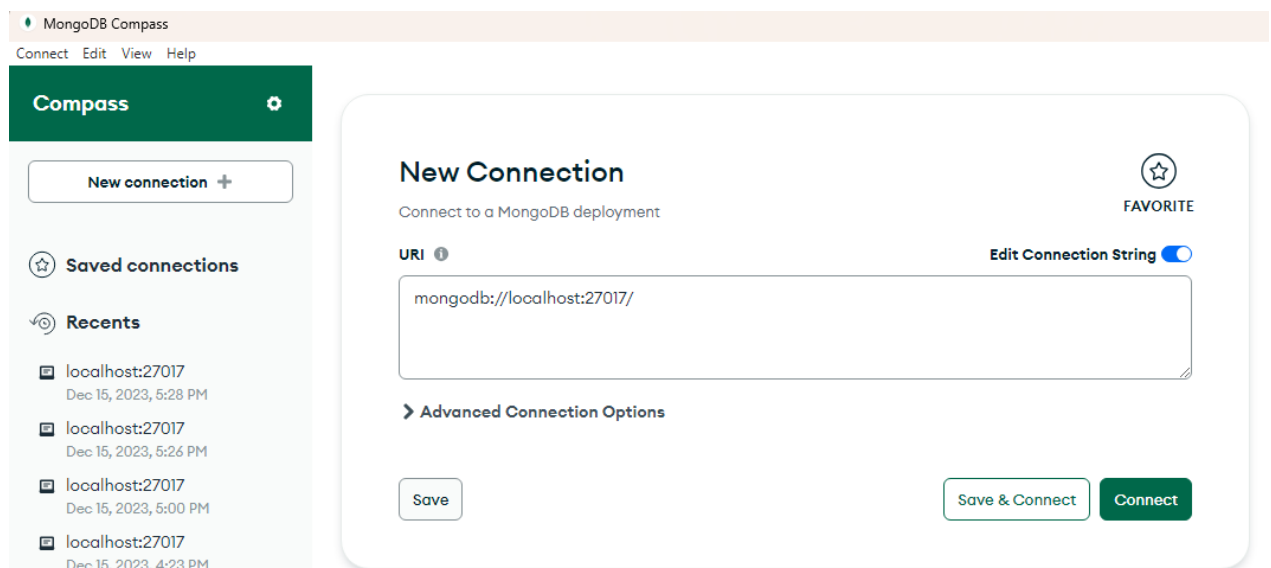
7.1. Giao diện đăng nhập

- Để đăng nhập vào ứng dụng, cần nhập vào localhost của MongoDB, nếu không nhập thì sẽ mặc định sử dụng localhost mongodb://localhost:27017/
- **Lưu ý:** Tại localhost, phải có trước database Wine và collection ThongSo.



Hình 7.1: Kết nối MongoDB

- Để tạo database trên mongoDB, có thể thực hiện các bước dưới đây:
 - o **Bước 1:** Mở ứng dụng MongoDB Compass và kết nối vào localhost.



Hình 7.2: Mở ứng dụng MongoDB Compass

- o **Bước 2:** Thực hiện tạo database mới với tên Wine và collection là ThongSo.

Create Database ×

Database Name

Wine

Collection Name

ThongSo

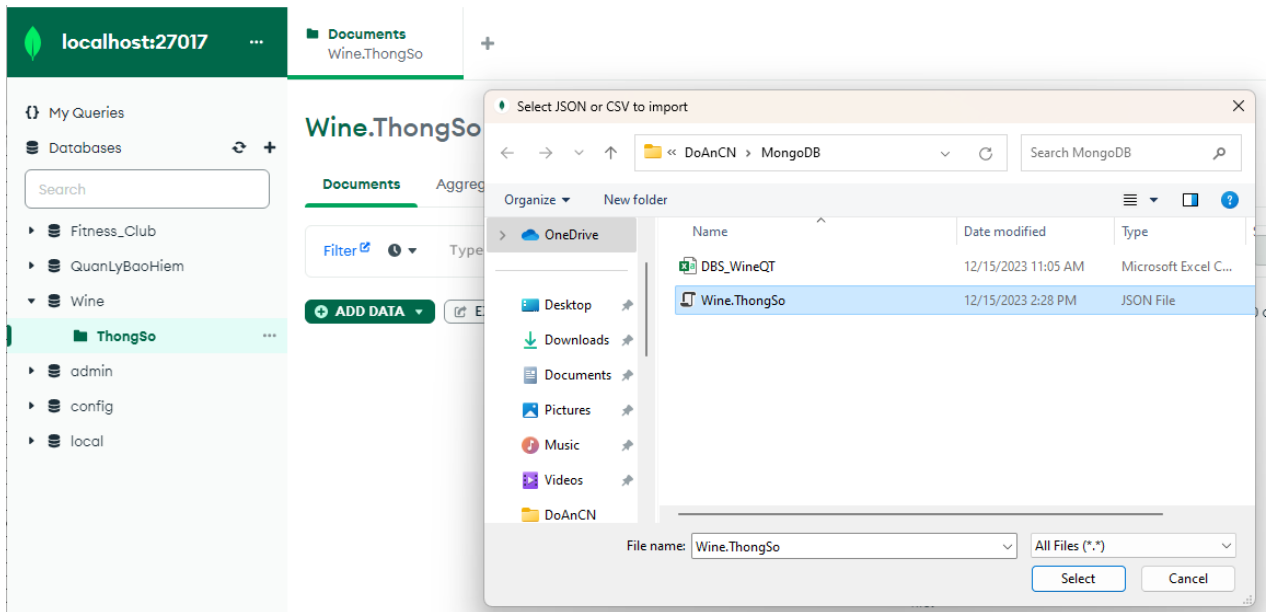
☐ **Time-Series**
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

> Additional preferences (e.g. Custom collation, Capped, Clustered collections)

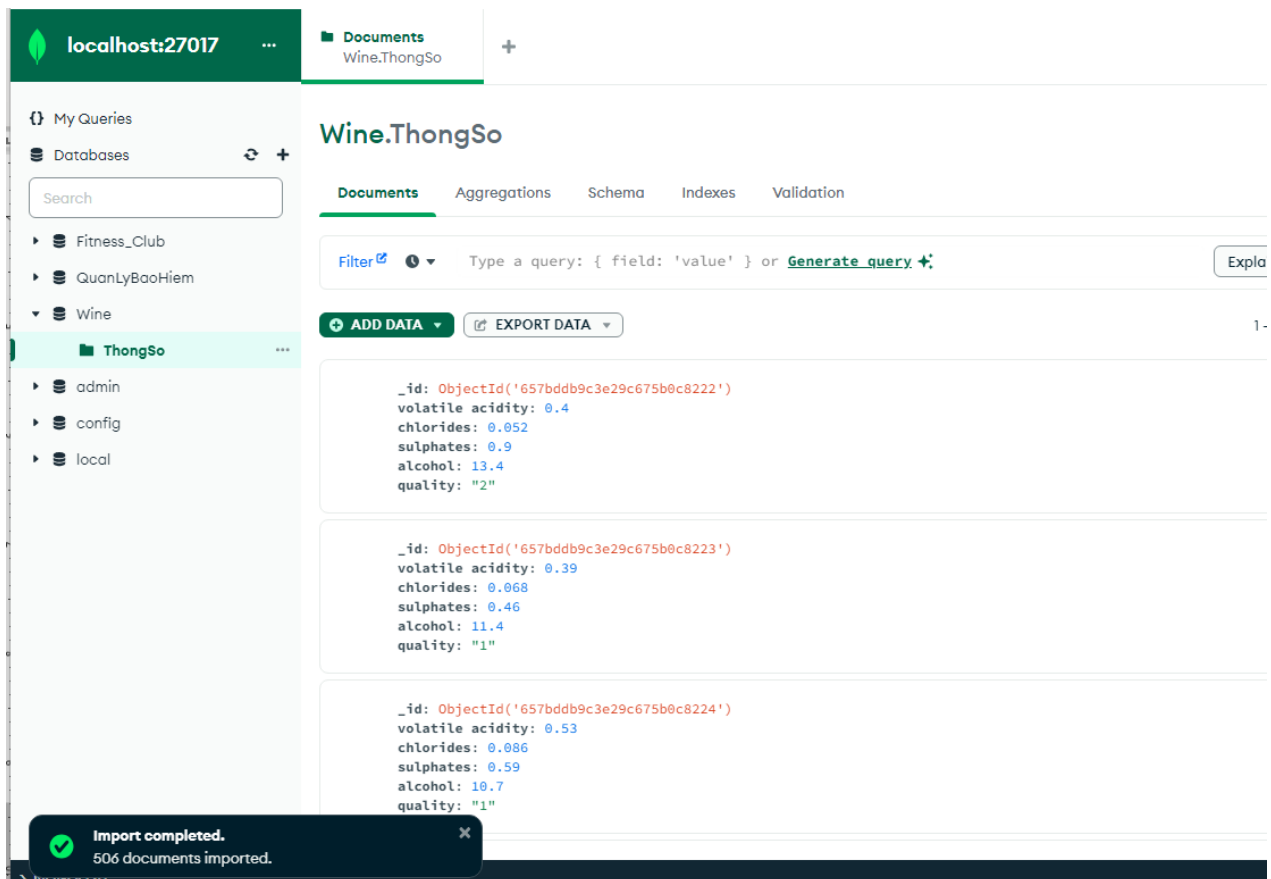
Cancel Create Database

Hình 7.3: Tạo mới database

- **Bước 3:** Thực hiện thêm dữ liệu cho collection ThongSo bằng cách thêm file json (trong thư mục MongoDB).



Hình 7.4: Thêm dữ liệu cho collection

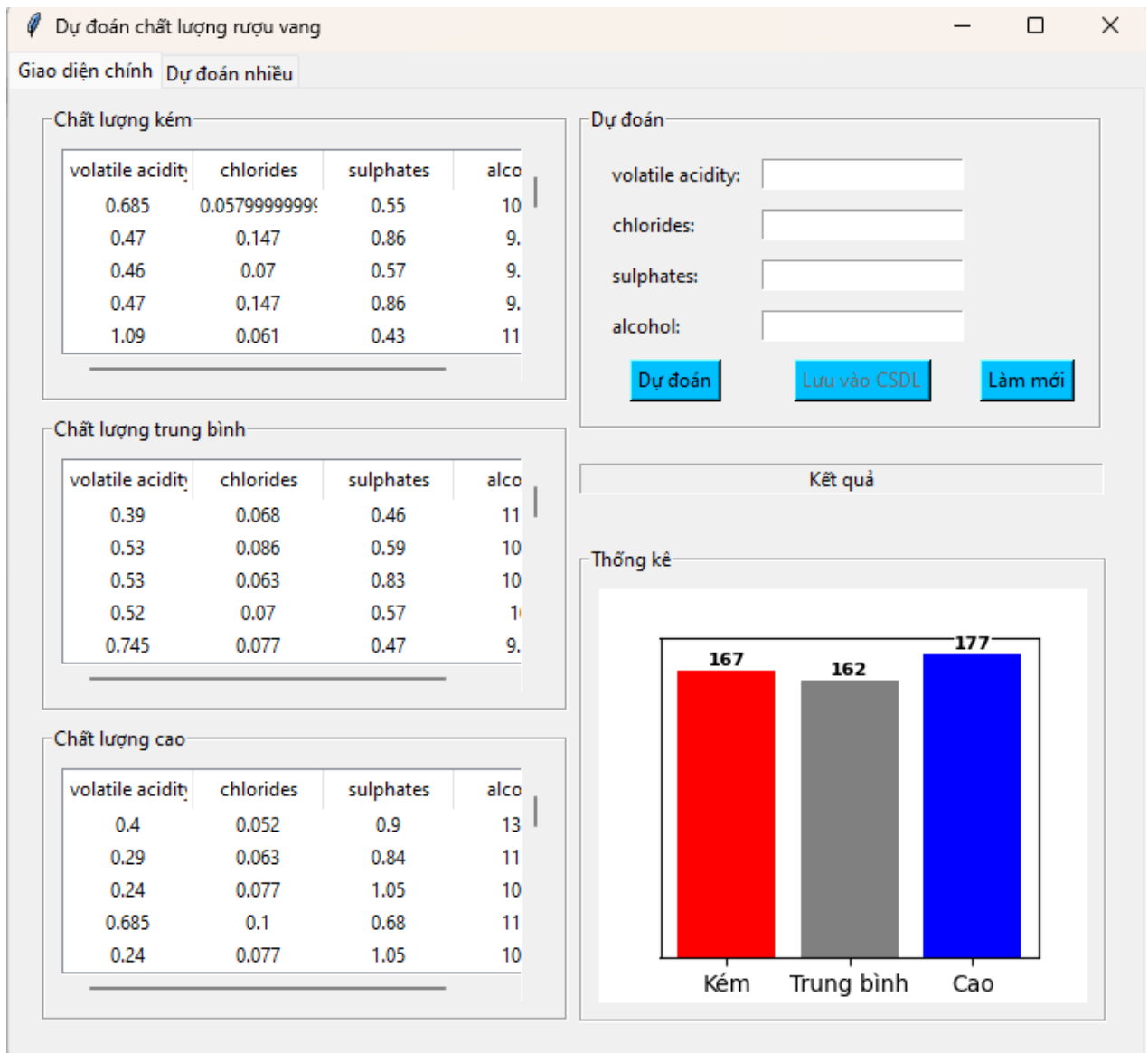


Hình 7.5: Hoàn thành tạo và thêm dữ liệu cho database

7.2. Giao diện chức năng chính

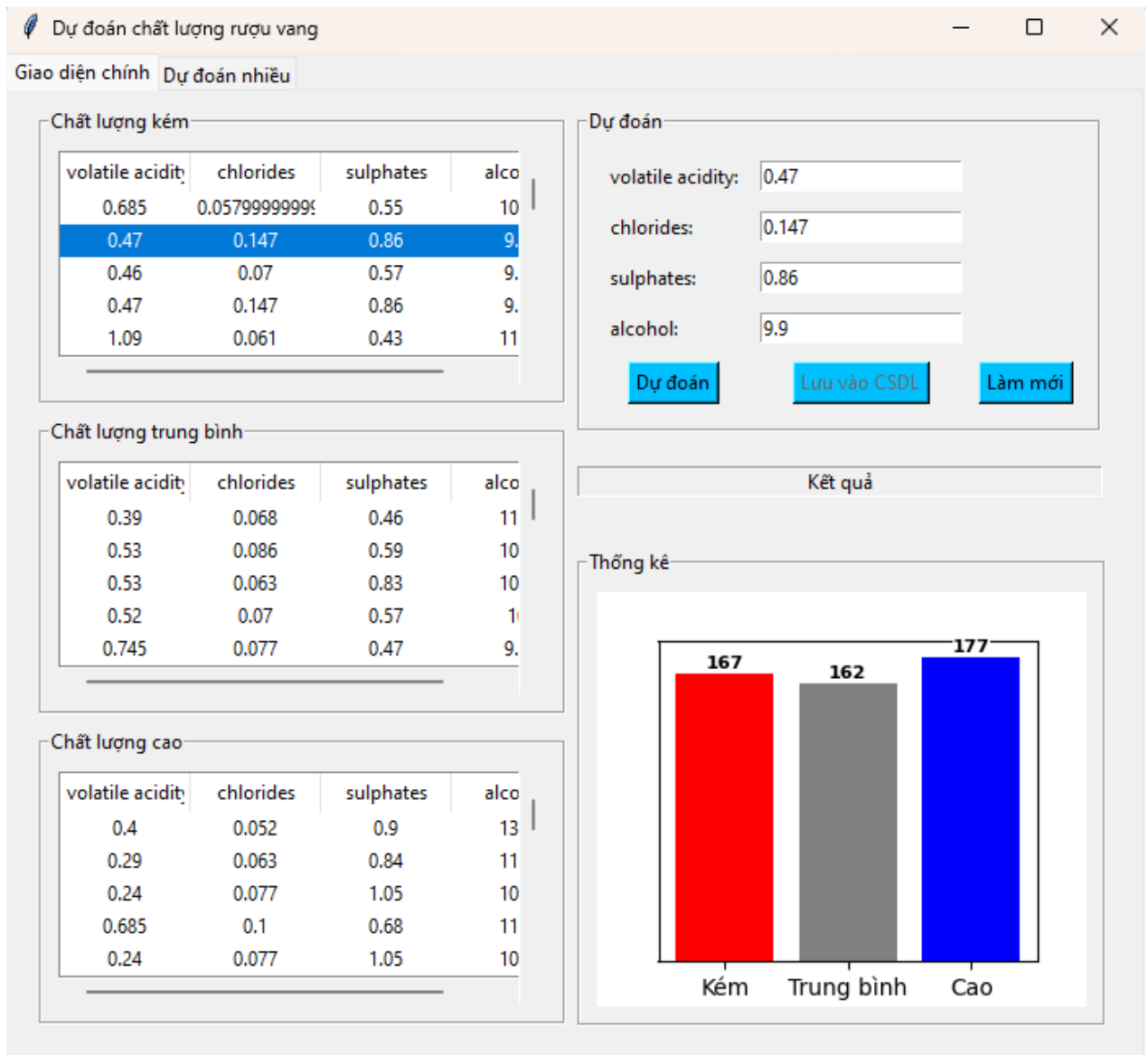
- Với giao diện chính sẽ có 5 groupbox lần lượt là:

- **Chất lượng kém:** Treeview hiển thị thông tin những loại rượu có chất lượng kém.
- **Chất lượng trung bình:** Treeview hiển thị thông tin những loại rượu có chất lượng trung bình.
- **Chất lượng cao:** Treeview hiển thị thông tin những loại rượu có chất lượng cao.
- **Thống kê:** Biểu đồ trực quan theo số lượng của mỗi nhóm chất lượng rượu.
- **Dự đoán:** Sẽ có 2 chức năng là dự đoán chất lượng rượu và lưu lại thông tin vào cơ sở dữ liệu. Bạn có thể nhập vào những thông số tương ứng với 4 thuộc tính của mô hình để dự đoán, sau đó lưu vào cơ sở dữ liệu để làm phong phú thêm tập dữ liệu.



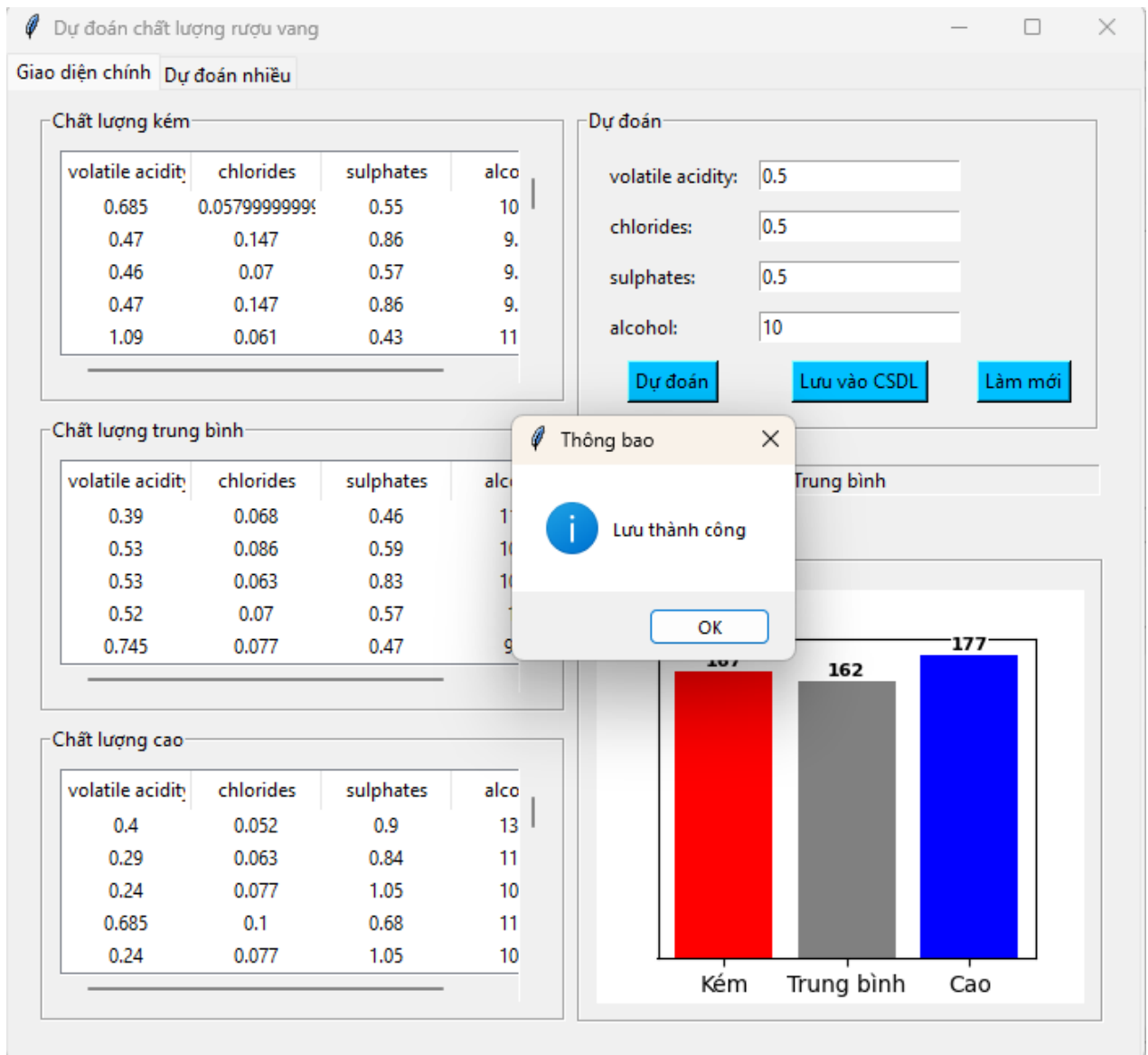
Hình 7.6: Giao diện chức năng chính

- Có thể dự đoán lại các giá trị đã có trên cơ sở dữ liệu bằng cách lựa chọn bất kì dòng nào trên cả 3 treeview. Những dữ liệu này được lấy từ tập X_test trong quá trình xây dựng và huấn luyện các mô hình.



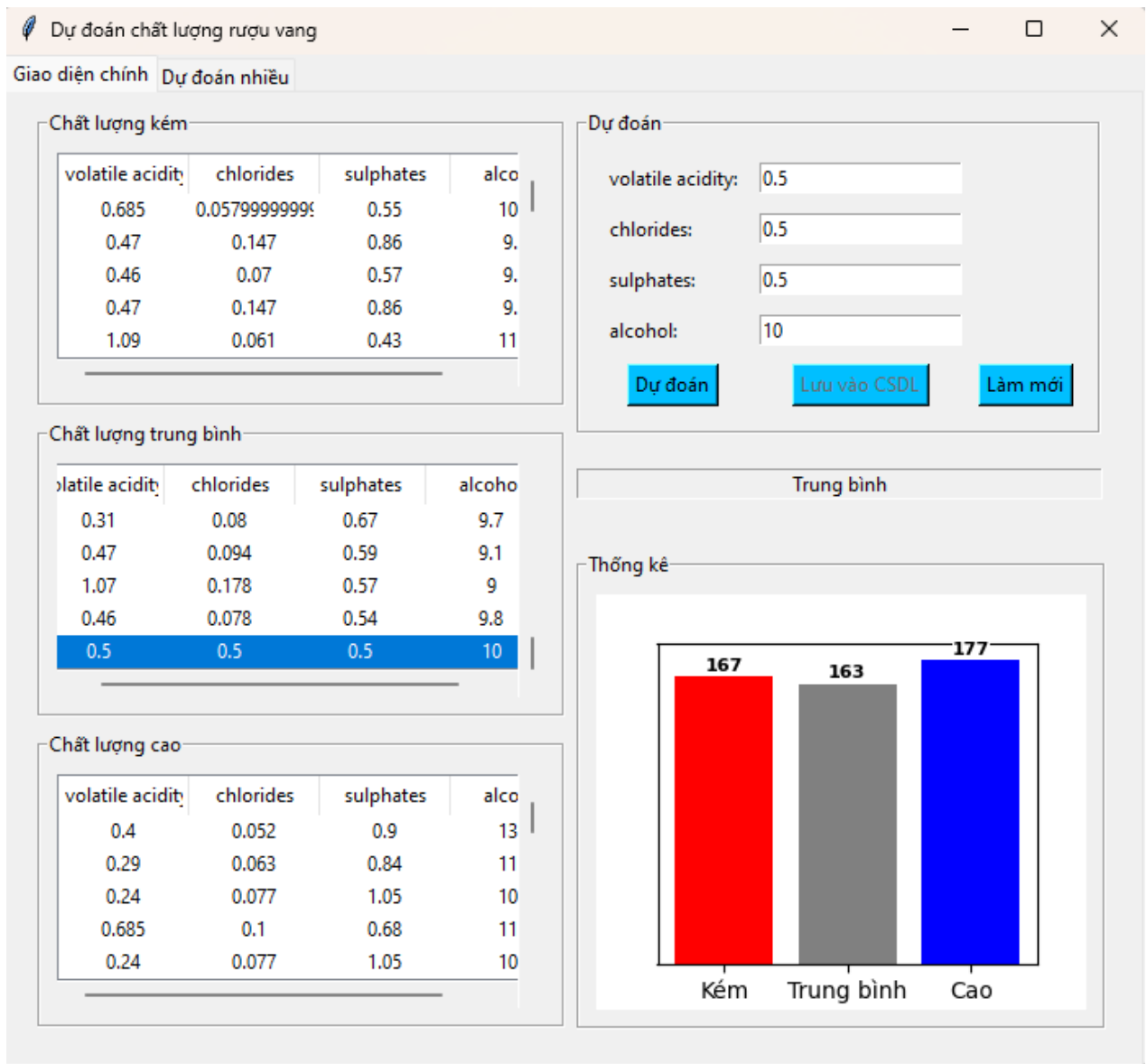
Hình 7.7: Dự đoán trên tập test

- Có thể nhập mới thông số để dự đoán. Sau đó chỉ cần nhấn vào nút “Lưu vào CSDL”, thì dữ liệu sẽ được thêm vào collection `ThongSo`.



Hình 7.8: Dự đoán và lưu vào cơ sở dữ liệu

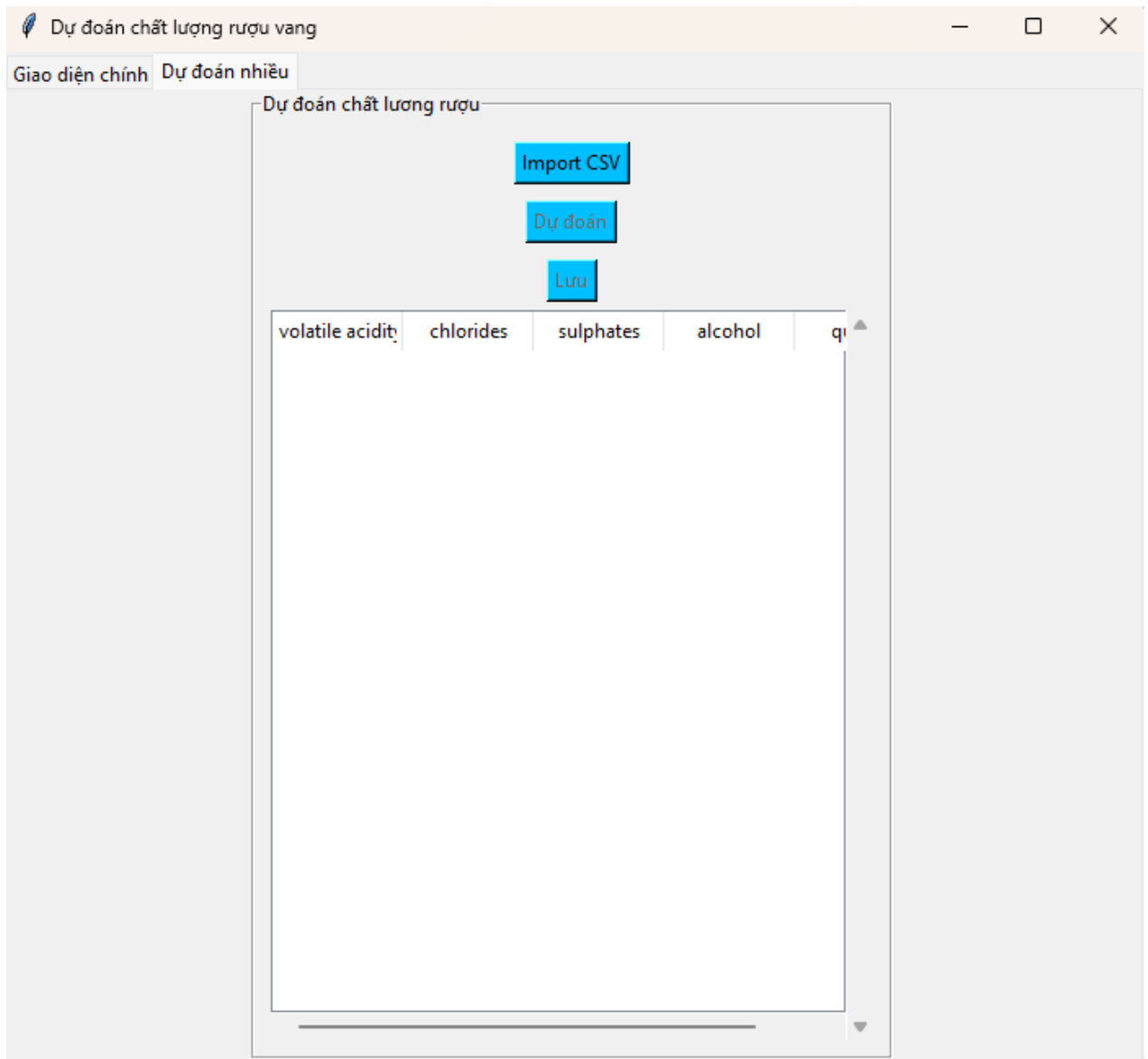
- Sau khi lưu, trên giao diện sẽ tự cập nhật lại thông tin trên treeview và biểu đồ.



Hình 7.9: Dữ liệu lưu thành công

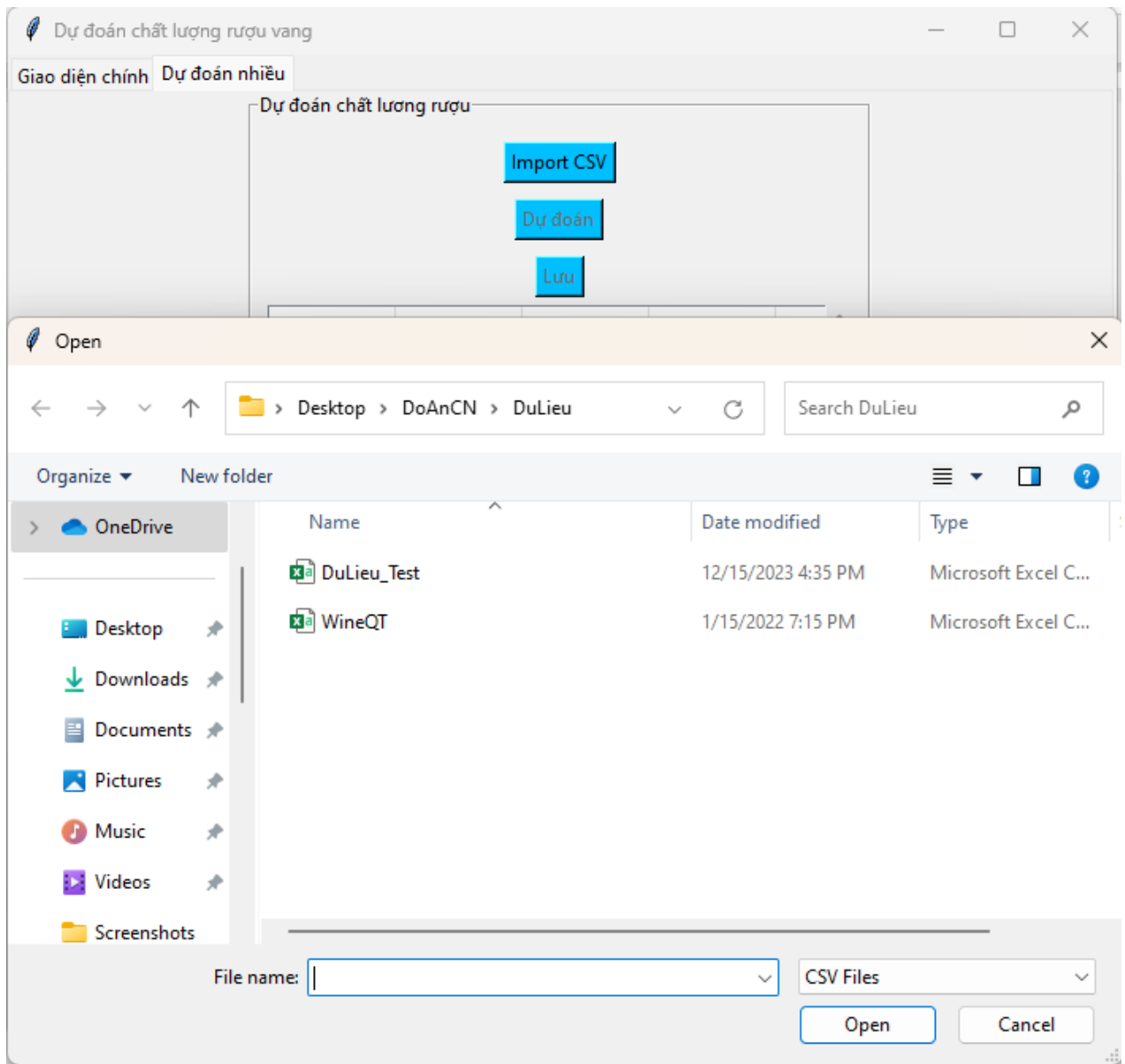
7.3. Dự đoán trên tập dữ liệu

- Ngoài ra, còn có chức năng dự đoán trên 1 tập dữ liệu (CSV) gồm nhiều dòng.



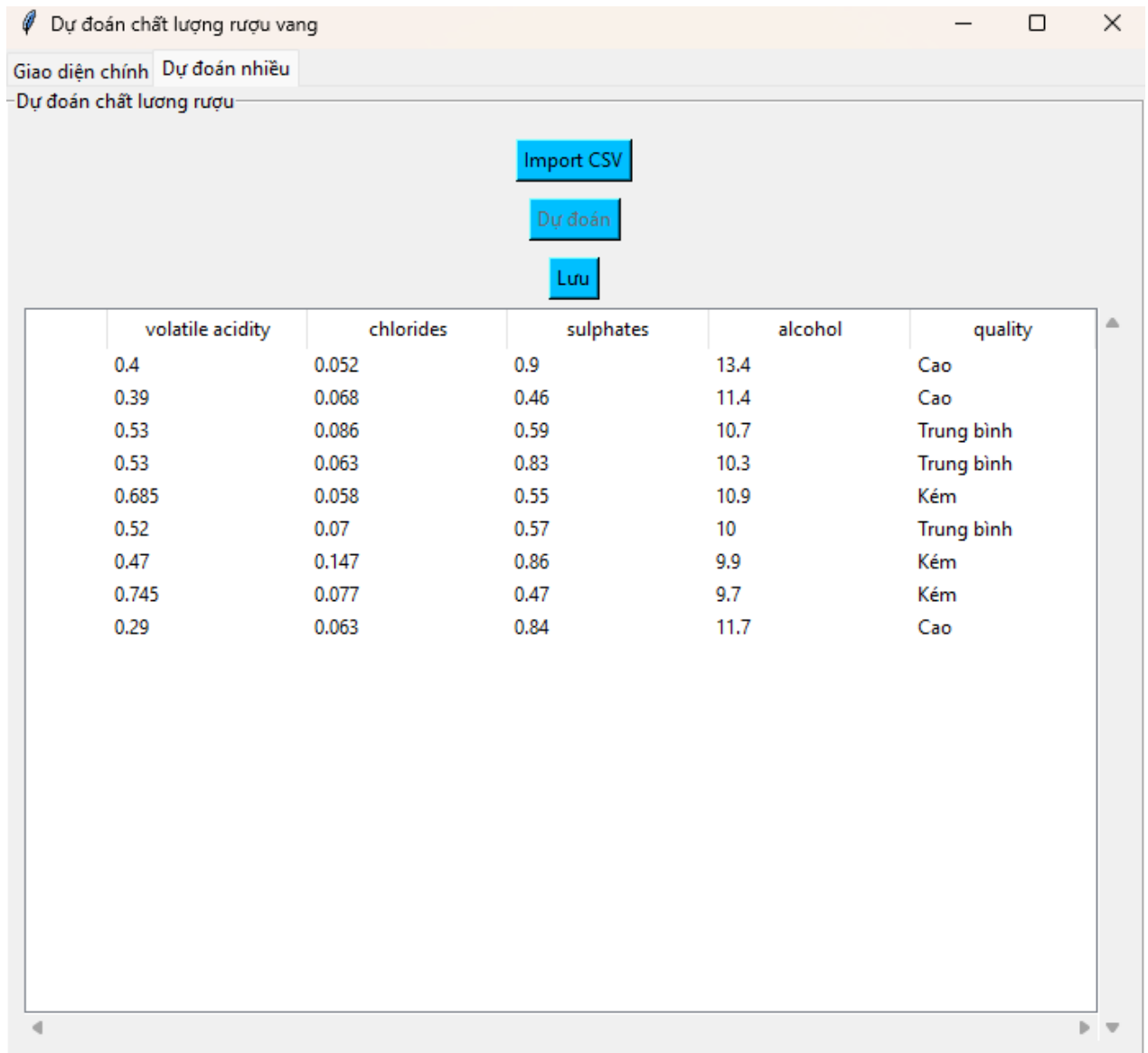
Hình 7.10: Giao diện dự đoán trên tập dữ liệu

- Nhấn vào nút “Import CSV” để chọn tập dữ liệu cần dự đoán.



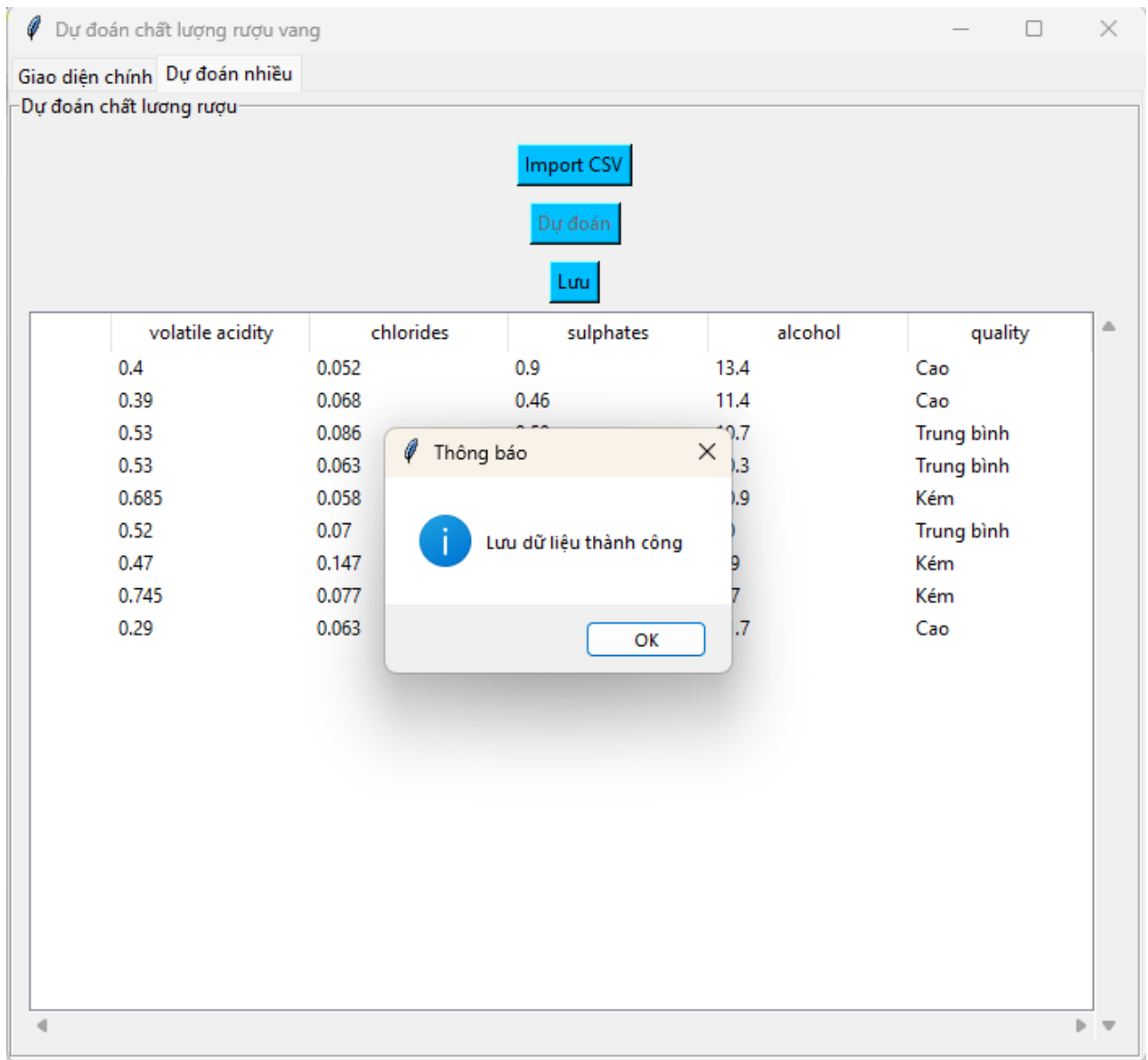
Hình 7.11: Chọn tập dữ liệu cần dự đoán

- Sau khi mở thành công tập dữ liệu, ta chỉ cần nhấn vào nút dự đoán, ứng dụng sẽ sử dụng mô hình và lặp lần lượt mỗi dòng trên treeview để dự đoán và trả về kết quả vào cột quality.



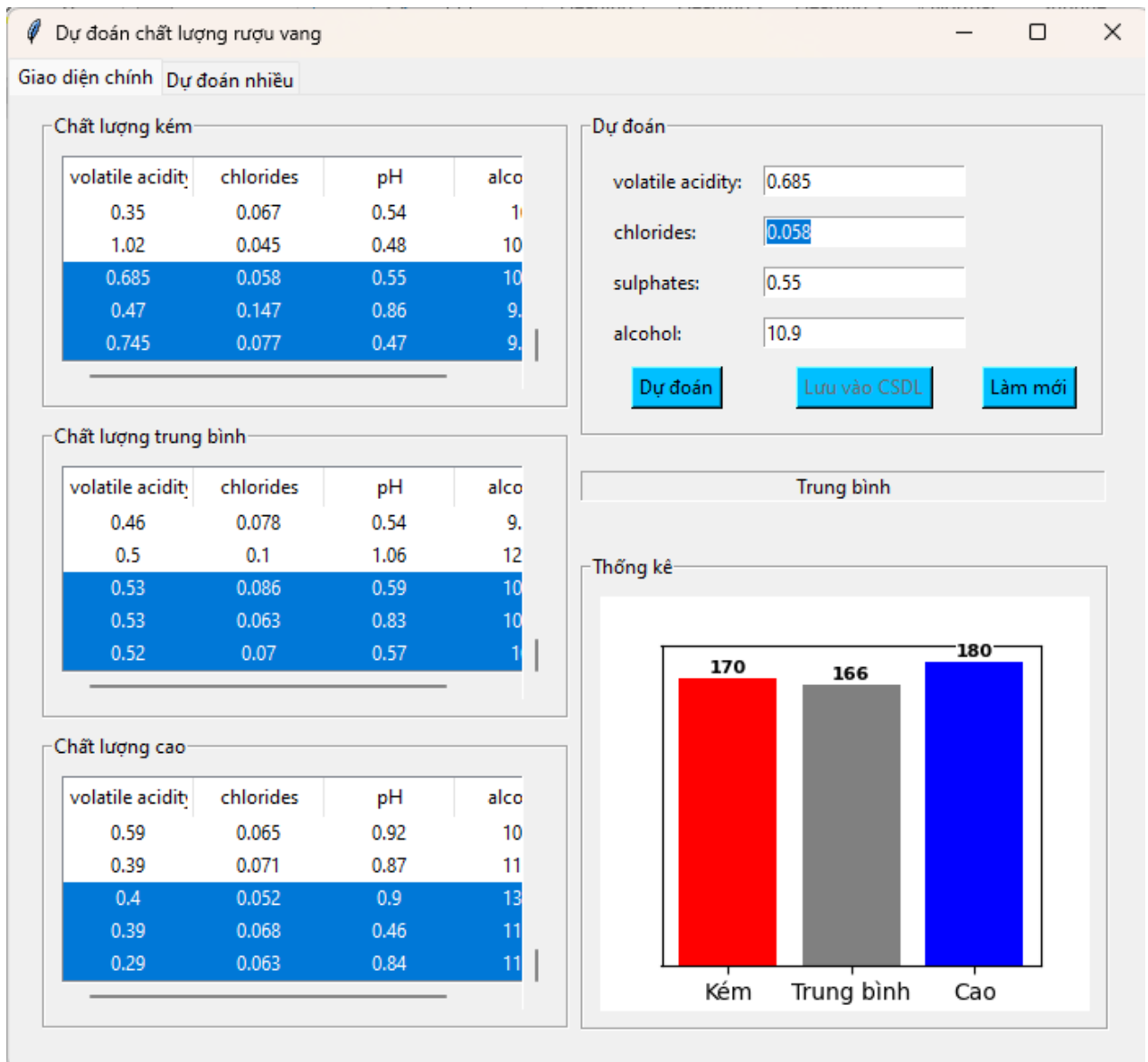
Hình 7.12: Dự đoán trên tập dữ liệu

- Để lưu lại tập dữ liệu này, chỉ cần nhấn nút “Lưu”.



Hình 7.13: Lưu tập dữ liệu

- Tại giao diện chính dữ liệu và biểu đồ sẽ được tự động cập nhật theo.



Hình 7.14: Cập nhật giao diện chính

CHƯƠNG 8: TỔNG KẾT

8.1. Các yếu tố quan trọng

Nhóm đã xác định được ảnh hưởng của các yếu tố đến chất lượng rượu thông qua việc xây dựng mô hình cây quyết định và mô hình hồi quy Logistic. Các được xác định là có tác động đặc biệt đáng chú ý: volatile acidity, chlorides, sulphates và alcohol.

8.2. Hướng dẫn sử dụng

- Để sử dụng mô hình với mục đích dự đoán chất lượng rượu, người dùng chỉ cần cung cấp giá trị của các yếu tố sau đây:
 - volatile acidity.
 - chlorides.
 - sulphates.
 - alcohol.
- Mô hình sẽ trả về một dự đoán về chất lượng rượu dựa trên các giá trị người dùng nhập vào.

8.3. Đối tượng người dùng

Mô hình có thể được tích hợp vào quy trình sản xuất rượu vang, để dự đoán chất lượng rượu và hỗ trợ quyết định sản xuất. Điều này có thể giúp tối ưu hóa quy trình và cải thiện chất lượng sản phẩm.

KẾT LUẬN

Trong cuộc hành trình khám phá chất lượng rượu vang, chúng tôi đã sâu sát vào các chi tiết hóa học và sinh học để tìm ra những yếu tố ẩn sau chất lượng. Phương pháp hóa dữ liệu, xây dựng mô hình đã đồng lòng góp phần tạo nên một cái nhìn đa chiều về cách những yếu tố như volatile acidity, chlorides, sulphates và alcohol ảnh hưởng đến chất lượng rượu.

Kết quả của chúng tôi không chỉ là số liệu và mô hình hóa, mà còn là sự hiểu biết sâu rộng về cách những thuộc tính này tương tác và ảnh hưởng đến trải nghiệm của người thưởng thức rượu. Những chiến lược kiểm soát và cải thiện chất lượng rượu mà chúng tôi đề xuất có thể đóng góp vào sự hoàn thiện của quy trình sản xuất và tạo ra những sản phẩm rượu vang tinh tế và độc đáo.

Cuối cùng, cuộc nghiên cứu này không chỉ mở ra những cửa sổ mới về khoa học rượu vang mà còn đặt ra những cơ hội cho những nhà sản xuất mong muốn nâng cao chất lượng sản phẩm của mình. Sự hiểu biết sâu sắc về những yếu tố chi phối chất lượng rượu không chỉ là tri thức mà còn là chìa khóa mở cánh cửa cho sự đổi mới và thành công trong ngành công nghiệp thưởng thức rượu vang.

TÀI LIỆU THAM KHẢO

- Daria Alekseeva (2009). Red and White Wine Quality by Daria Alekseeva [online], viewed 20/11/2023, from:< [rstudio-pubs-static.s3.amazonaws.com/57835_c4ace81da9dc45438ad0c286bcbb4224.html](https://static.s3.amazonaws.com/57835_c4ace81da9dc45438ad0c286bcbb4224.html)>.
- M YASSER H (2021). Wine Quality Dataset [online], viewed 07/11/2023, from:< [Wine Quality Dataset \(kaggle.com\)](https://www.kaggle.com/datasets/yasserh/wine-quality-dataset)>.