**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**ĐỒ ÁN CUỐI KÌ**Môn học : Hệ hỗ trợ quyết định

Học kỳ II (2021-2022)

**ĐỀ TÀI: DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM Ở NGƯỜI**

Sinh viên thực hiện:

1. Lê Tuấn Khanh MSSV: 19521681

2. Trần Nhật Tân MSSV: 19522177

GVHD: Thầy Nguyễn Thanh Bình

Lớp: IS254.M21.TMCL

**Thành phố Hồ Chí Minh, tháng 6 năm 2022**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**ĐỒ ÁN CUỐI KÌ**Môn học : Hệ hỗ trợ quyết định

Học kỳ II (2021-2022)

**ĐỀ TÀI: DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM Ở NGƯỜI**

Sinh viên thực hiện:

1. Lê Tuấn Khanh MSSV: 19521681

2. Trần Nhật Tân MSSV: 19522177

GVHD: Thầy Nguyễn Thanh Bình

Lớp: IS254.M21.TMCL

**Thành phố Hồ Chí Minh, tháng 6 năm 2022**

**MỤC LỤC**

[**LỜI CẢM ƠN** 4](#_Toc104638777)

[**NHẬN XÉT CỦA GIẢNG VIÊN** 5](#_Toc104638778)

[**CHƯƠNG I – TỔNG QUAN ĐỀ TÀI** 6](#_Toc104638779)

[**1.1** **Lý do chọn đề tài** 6](#_Toc104638780)

[**1.2** **Nội dung đề tài** 7](#_Toc104638781)

[**1.3** **Mô tả dữ liệu gốc** 8](#_Toc104638782)

[**CHƯƠNG II – TRIỂN KHAI THỰC NGHIỆM** 15](#_Toc104638783)

[**2.1** **Công cụ sử dụng** 16](#_Toc104638784)

[**2.2** **Thuật toán Logistic Regression** 16](#_Toc104638785)

[**Định nghĩa** 16](#_Toc104638786)

[**2.3** **Thuật toán cây quyết định** 17](#_Toc104638787)

[**Định nghĩa** 17](#_Toc104638788)

[**2.4** **Thuật toán K-Nearest Neighbors** 21](#_Toc104638789)

[**Định nghĩa** 21](#_Toc104638790)

[**2.5** **Thuật toán Random Forest** 22](#_Toc104638791)

[**Định nghĩa** 22](#_Toc104638792)

[**2.6** **Các bước thực hiện** 23](#_Toc104638793)

[**2.6.1** **Tiền xử lý dữ liệu** 23](#_Toc104638794)

[**2.6.3** **Data Scaling** 40](#_Toc104638795)

[**CHƯƠNG III - KẾT QUẢ ĐẠT ĐƯỢC** 60](#_Toc104638796)

[**CHƯƠNG IV - PHẦN MỀM DỰ ĐOÁN** 64](#_Toc104638797)

[**4.1 Tổng quan phần mềm** 64](#_Toc104638798)

[**4.1.1 Thuật toán được sử dụng** 64](#_Toc104638799)

[**4.1.2 Các thuộc tính được sử dụng để đưa ra dự đoán** 64](#_Toc104638800)

[**4.1.3 Giao diện và kiểm thử** 65](#_Toc104638801)

[**4.1.3.1 Giao diện** 65](#_Toc104638802)

[**4.1.3.2 Kiểm thử** 65](#_Toc104638803)

[**4.2 Code phần mềm** 67](#_Toc104638804)

[**4.2.1 Code phần giao diện:** 67](#_Toc104638805)

[**4.2.2 Code phần xử lý** 72](#_Toc104638806)

[**CHƯƠNG V - KẾT LUẬN** 74](#_Toc104638807)

[**5.1 Ưu điểm, hạn chế của từng thuật toán** 74](#_Toc104638808)

[***5.1.1 Cây quyết định*** 74](#_Toc104638809)

[***5.1.2. Random Forest*** 76](#_Toc104638810)

[***5.1.3 Logistic Regression*** 76](#_Toc104638811)

[***5.1.4 K-nearest Neighbors*** 76](#_Toc104638812)

[**5.2 Hướng phát triển** 77](#_Toc104638813)

[**Tài liệu tham khảo** 78](#_Toc104638814)

[**Hết.** 78](#_Toc104638815)

# **LỜI CẢM ƠN**

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Với lòng biết ơn sâu sắc nhất, đầu tiên nhóm chúng em xin gởi lời cảm ơn chân thành đến tập thể quý Thầy Cô Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý Thầy Cô khoa Hệ thống thông tin đã giúp cho nhóm có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt nhóm chúng em xin gửi lời cảm ơn chân thành tới thầy Nguyễn Thanh Bình – giảng viên lí thuyết và thực hành môn Hệ hỗ trợ ra quyết định đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn nhóm trong suốt quá trình làm đồ án môn học. Nhờ đó, chúng em đã tiếp thu được nhiều kiến thức bổ ích trong việc vận dụng cũng như kỹ năng làm đồ án. Nếu không có những lời hướng dẫn, dạy bảo của Thầy thì nhóm chúng em nghĩ đồ án này của nhóm rất khó có thể hoàn thiện được. Một lần nữa, em xin chân thành cảm ơn Thầy. Ngoài ra, để đồ án được hoàn thành thì không thể nào cám ơn những người đã làm ra đó, cám ơn các bạn các thành viên trong nhóm đã chăm chỉ và chịu khó hoàn thành nhiệm vụ đúng tiến độ.

Cuối cùng, cảm ơn tất cả các thành viên trong nhóm đã làm việc hết công suất để hoàn thành tốt đề tài của mình. Xin chân thành cảm ơn!

# **NHẬN XÉT CỦA GIẢNG VIÊN**

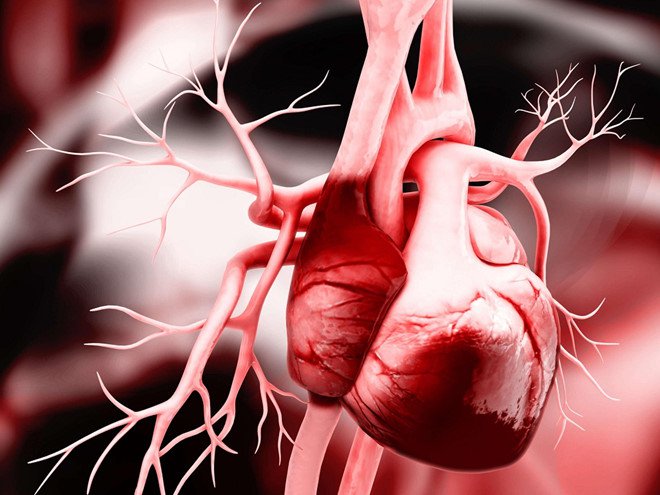
……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

# **CHƯƠNG I – TỔNG QUAN ĐỀ TÀI**

* 1. **Lý do chọn đề tài**

Trong thời đại ngày nay, khi mà công nghệ 4.0 ngày càng lên ngôi, việc khai thác dữ liệu phục vụ các yêu cầu trợ giúp quyết định ngày càng có ý nghĩa quan trọng và là nhu cầu to lớn trong mọi lĩnh vực hoạt động, như kinh doanh, quản lý, y tế, nghiên cứu khoa học,… Dữ liệu được lưu trữ và thu thập ngày cành nhiều từ đó có thể hỗ trợ con người trong việc dự báo hay phán đoán ra quyết định.

Hiện nay, bệnh lý tim mạch đang dần trở thành gánh nặng cho xã hội, kéo theo nhiều hệ lụy trong đời sống. Theo WHO - Tổ chức Y tế Thế giới, mỗi năm có khoảng 17,5 triệu người tử vong do mắc các bệnh lý về tim và mạch máu. Cứ mỗi 2 giây sẽ có một người chết vì bệnh tim mạch, cứ 5 giây sẽ có một người bị nhồi máu cơ tim.



Báo cáo của WHO cũng cho biết tỷ lệ bệnh tim mạch đang ngày càng tăng cao ở các nước đang phát triển trong đó có Việt Nam. Bên cạnh đó, chi phí cho khám và chữa bệnh tim mạch cũng là gánh nặng kinh tế với chi phí hàng trăm tỷ mỗi năm.

Chính vì thế, việc phân tích dữ liệu để đưa ra những quyết định đúng đắn đối với sức khỏe của con người là vô cùng cần thiết để nhanh chóng tìm ra giải pháp góp phần cải thiện sức khỏe của toàn xã hội.

* 1. **Nội dung đề tài**

Đây là Bộ dữ liệu phân loại rủi ro Heart Disease được chia sẻ trên <https://www.kaggle.com/> . Tập dữ liệu này có từ năm 1988 và bao gồm bốn cơ sở dữ liệu: Cleveland, Hungary, Thụy Sĩ và Long Beach V. Nó chứa 76 thuộc tính, bao gồm cả thuộc tính dự đoán, nhưng tất cả các thử nghiệm đã xuất bản đều sử dụng một tập con gồm 12 thuộc tính trong số đó. Trường ***"Target"*** đề cập đến sự hiện diện của bệnh tim ở bệnh nhân. Nó là số nguyên có giá trị 0 = không có bệnh và 1 = bệnh.

Người ta cho rằng những thông tin như vậy đóng một vai trò quan trọng trong việc dự đoán và đưa ra quyết định giảm thiểu bệnh tim.

Nguồn dữ liệu: ‎[Heart Disease Dataset | Kaggle](https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset)

Chủ đề: Y tế.

Dataset gồm : 14 cột thuộc tính và 3235 dòng dữ liệu

Graphical user interface

Description automatically generated with medium confidence

* 1. **Mô tả dữ liệu gốc**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | age | number | độ tuổi |
| 2 | sex | binary | giới tính |
| 3 | cp | number | loại đau ngực |
| 4 | trestbps | number | huyết áp lúc nghỉ (tính bằng mm Hg khi nhập viện) |
| 5 | chol | number | cholestoral trong huyết thanh tính bằng mg / dl |
| 6 | fbs | number | (đường huyết lúc đói & gt; 120 mg / dl) (1 = true; 0 = false) |
| 7 | restecg | number | kết quả điện tâm đồ lúc nghỉ ngơi |
| 8 | thalach | number | nhịp tim tối đa đạt được |
| 9 | exang | binary | cơn đau thắt ngực do tập thể dục gây ra (1 = có; 0 = không) |
| 10 | oldpeak | number | ST trầm cảm gây ra do tập thể dục liên quan đến nghỉ ngơi |
| 11 | slope | number | độ dốc của đoạn ST tập thể dục cao điểm |
| 12 | target | binary | có sự hiện diện của bệnh tim hay không (1 = có; 0 = không) |

**Chart, histogram

Description automatically generated**

**Chart, histogram

Description automatically generated**

**Chart, waterfall chart

Description automatically generated**

**Chart, histogram

Description automatically generated**

**Chart, waterfall chart

Description automatically generated**

**Chart, histogram

Description automatically generated**

# **CHƯƠNG II – TRIỂN KHAI THỰC NGHIỆM**

* 1. **Công cụ sử dụng**

Trong quá trình thực hiện, nhóm đã sử dụng một số phần mềm phục vụ cho việc tìm  hiểu và xây dựng đề tài:

* Phần thu thập và phân tích thông tin sử dụng thư viện  và ngôn ngữ lập trình python: python3.7, Jupyter notebook,…
* Dữ liệu: [Heart Disease Predication | Kaggle](https://www.kaggle.com/datasets/durgesh2050/heart-disease-predication?fbclid=IwAR2ojjXmzqj4rnrIdXQagmXAZ1WZ48_kaatO1SPkVWEqS1zJwntDEsdyjLA)

Tất cả các phần mềm trên được nhóm cài đặt và sử dụng trên Hệ điều hành Microsoft Windows 10. Việc tương thích các phần mềm trên với các hệ điều hành khác không nằm trong phạm vi nghiên cứu của đề tài này.

* 1. **Thuật toán Logistic Regression**

**Định nghĩa**

Phương pháp hồi quy logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu  ra rời rạc (discrete target variable) y ứng với một véc-tơ đầu vào **x**. Việc này tương đương với chuyện phân loại các đầu vào **x** vào các nhóm y tương ứng.

Phân tích hồi qui logistic là một kỹ thuật thống kê để xem xét mối liên hệ giữa biến độc lập (biến số hoặc biến phân loại) với biến phụ thuộc là biến nhị phân. Trong hồi qui tuyến tính đơn, biến độc lập x và phụ thuộc y là biến số liên tục liên hệ qua phương trình:

y = a + bx + e

Trong đó: a là alpha, b là beta, e là epxilon

Trong hồi qui logistic, biến phụ thuộc y chỉ có 2 trạng thái 1 và 0. Muốn đổi ra biến số liên tục người ta tính xác suất của 2 trạng thái này. Nếu gọi p là xác suất để một biến cố xảy ra, thì 1-p là xác suất để biến cố không xảy ra. Phương trình hồi qui logistic phát biểu:A picture containing text, watch, clock

Description automatically generated

Ta tính được xác xuất:Chart

Description automatically generated

* 1. **Thuật toán cây quyết định**

**Định nghĩa**

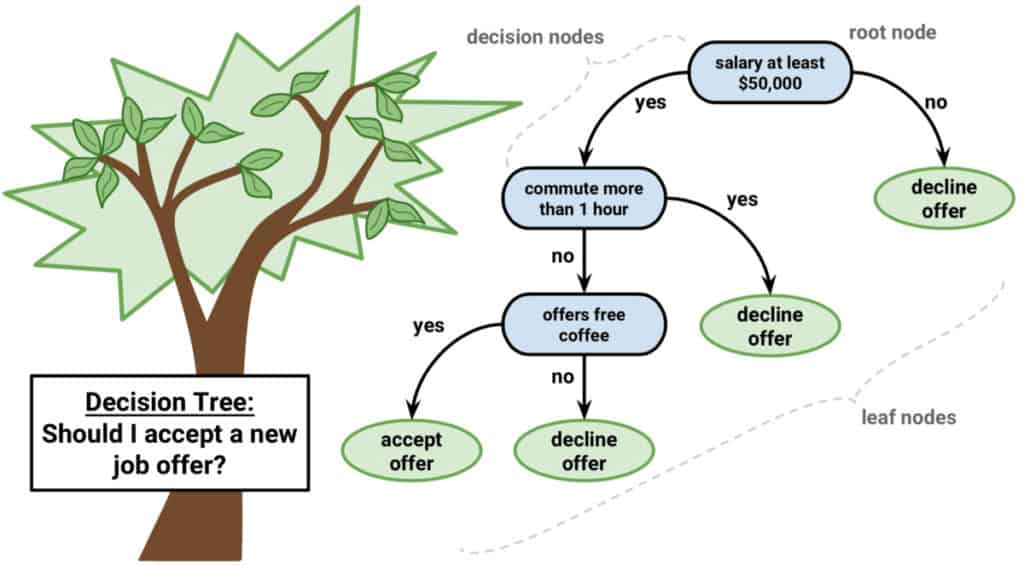
Cây quyết định (Decision Tree) là cấu trúc cây sao cho:

+ Mỗi nút trong ứng với một phép kiểm tra trên một thuộc tính

+ Mỗi nhánh biểu diễn kết quả phép kiểm tra

+ Các nút lá biểu diễn các lớp hay các phân bố lớp

+ Nút cao nhất trong cây là nút gốc. Hình dạng cây quyết định:



Chiến lược cơ bản:

+  Bắt đầu từ nút đơn biểu diễn tất cả các mẫu

+  Nếu các mẫu thuộc về cùng một lớp, nút trở thành nút lá và được gán nhãn bằng lớp đó

+  Ngược lại, dùng độ đo thuộc tính để chọn thuộc tính sẽ phân tách tốt nhất các mẫu vào các lớp

+  Một nhánh được tạo cho từng giá trị của thuộc tính được chọn và các mẫu được phân hoạch theo.

+  Dùng đệ quy cùng một quá trình để tạo cây quyết định

+  Tiến trình kết thúc chỉ khi bất kỳ điều kiện nào sau đây là đúng

* Tất cả các mẫu cho một nút cho trước đều thuộc về cùng một lớp.
* Không còn thuộc tính nào mà mẫu có thể dựa vào để phân hoạch xa hơn.
* Không còn mẫu nào tại nút

**ID3** là một thuật toán được sử dụng trong cây quyết định. Thuật toán này sử dụng độ lợi thông tin (Information Gain) để tiến hành xây dựng cây quyết định**. Thuộc tính Information Gain lớn nhất sẽ được chọn làm nút gốc.**

+ Độ lợi thông tin (Information Gain):

A picture containing text, watch, clock

Description automatically generated

+ Lượng thông tin cần để phân lớp một phần tử trong S dựa trên thuộc tính A: InfoA(S)Icon

Description automatically generated with medium confidence

* Information gain chính là độ sai biệt giữa giá trị thông tin Info(S) ban đầu (trước phân hoạch) và giá trị thông tin mới InfoA(S)(sau phân hoạch với A).



+ Độ bất định (Entropy):

* Entropy đo lường lượng thông tin trong một thuộc tính của một bộ giá trị thu thập được của tập mẫu.
* Entropy được dùng để xác định nút nào được tách kế tiếp trong thuật toán.
* Entropy càng cao thì khả năng cải thiện phân lớp càng cao.
* Công thức:

Text

Description automatically generated

𝑛 là số giá trị khác nhau của thuộc tính A đang xét

𝐴i là số mẫu tương ứng với mỗi giá trị 𝑖 của thuộc tính A

𝑓S(𝐴i) là tỷ lệ của số mẫu có thuộc tính 𝐴𝑖 với S

**CART:** Khác với ID3 dùng công thức Information Cain thì thuật toán Cart sử dụng công thức Gini. **Thuộc tính có giá trị Gini nhỏ nhất sẽ được làm nút gốc**.

+ Chỉ số Gini của tập S

A picture containing text, clock, clipart

Description automatically generated

P(j|S) là tần xuất của j trong S

+ Gini của thuộc tính

A picture containing text

Description automatically generated

Trong đó: ni là số mẫu trong nốt I, n là số mẫu trong nốt A

* 1. **Thuật toán K-Nearest Neighbors**

**Định nghĩa**

* K-nearest Neighbors là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai thác dữ liệu. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn của dữ liệu mới.
* Lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán từ các lớp (nhãn) của k hàng xóm gần nó nhất.
* Cho trước tập dữ liệu huấn luyện D với các lớp, phân lớp X vào các lớp dựa vào k phần tử tương tự với X nhất (quy tắc bỏ phiếu: majority vote).
* Các bước thực hiện:

+  Ta có D là tập các điểm dữ liệu đã được gắn nhãn và A là dữ liệu chưa được phân loại.

+  Đo khoảng cách (Euclidian, Manhattan, Minkowski, Minkowski hoặc Trọng số) từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D.

+  Chọn K (K là tham số mà bạn định nghĩa) khoảng cách nhỏ nhất.

+  Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và đếm số lượng của mỗi lớp xuất hiện.

+  Lấy đúng lớp (lớp xuất hiện nhiều lần nhất).

+  Lớp của dữ liệu mới là lớp đã nhận được ở bước 5.

* Phụ thuộc:

+  Độ đo khoảng cách để xác định sự tương tự.

+  Trị k, số phần tử láng giềng => k<=|D|1/2

* Độ đo Euclidean

A picture containing text, watch, gauge

Description automatically generated

* 1. **Thuật toán Random Forest**

**Định nghĩa**

Như chúng ta biết, Random Forest là một tập hợp mô hình (ensemble). Mô hình Random Forest rất hiệu quả cho các bài toán phân loại vì nó huy động cùng lúc hàng trăm mô hình nhỏ hơn bên trong với quy luật khác nhau để đưa ra quyết định cuối cùng. Mỗi mô hình con có thể mạnh yếu khác nhau, nhưng theo nguyên tắc **« wisdom of the crowd »**, ta sẽ có cơ hội phân loại chính xác hơn so với khi sử dụng bất kì một mô hình đơn lẻ nào.

Như tên gọi của nó, Random Forest (RF) dựa trên cơ sở :

1. Random = Tính ngẫu nhiên ;
2. Forest = nhiều cây quyết định (decision tree).

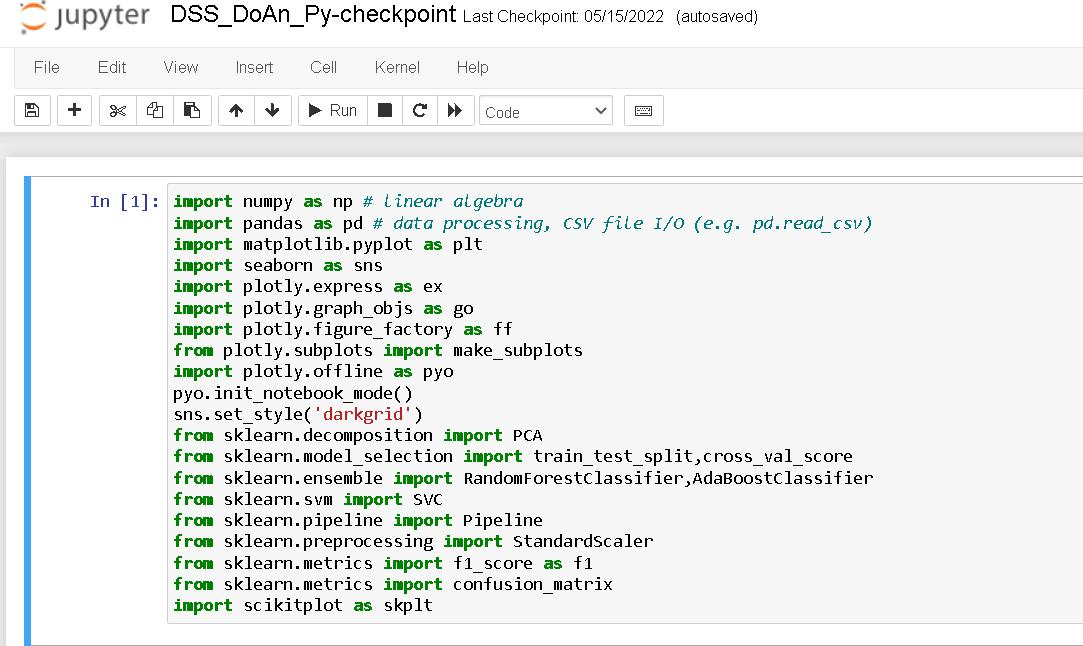
Đơn vị của RF là thuật toán cây quyết định, với số lượng hàng trăm. Mỗi cây quyết định được tạo ra một cách ngẫu nhiên từ việc : Tái chọn mẫu (bootstrap, random sampling) và chỉ dùng một phần nhỏ tập biến ngẫu nhiên (random features) từ toàn bộ các biến trong dữ liệu. Ở trạng thái sau cùng, mô hình RF thường hoạt động rất chính xác, nhưng đổi lại, ta không thể nào hiểu được cơ chế hoạt động bên trong mô hình vì cấu trúc quá phức tạp. RF do đó là một trong số những mô hình hộp đen (black box).

* 1. **Các bước thực hiện**

Code được upload tại : <https://github.com/TranNhatTan0312/DoAn_DSS.git>

* + 1. **Tiền xử lý dữ liệu**

Bước 1: Import các thư viện cần thiết.



Hình 1: import thư viện

Dataset là file /heart\_Disease\_prediction\_new\_final.csv.

Bước 2: Load dữ liệu từ file heart\_Disease\_prediction\_new\_final.csv vào c\_data.

Table

Description automatically generated with medium confidence

Hình 2: load dữ liệu

Xem cấu trúc data frame

Graphical user interface, application, table

Description automatically generated

Hình 3: Xem cấu trúc dataframe

Bước 3: Lọc dữ liệu.

Đầu tiên, ta sẽ chuyển những dòng có giá trị null thành giá trị rỗng.

Sau đó ta tiến hành drop những cột có giá trị rỗng và tiến hành reset bộ dữ liệu.

Vì những cột thuộc tính trong bộ dữ liệu đều rất cần thiết, không có khả năng gây nhiễu nên ta không cần phải xóa cột dữ liệu nào hết.

Table

Description automatically generated

Hình 4: Lọc null

Dữ liệu từ 3235 dòng, sau khi lọc đã còn 3187 dòng.

Bước 4: phân loại thuộc tính.

Ta tiến hành chọn thuộc tính quyết định của bộ dữ liệu, đó là cột “target”. Cột này có ý nghĩa là có sự hiện diện của bệnh tim hay không (1 = có; 0 = không)

Graphical user interface, text, application

Description automatically generated

Check phân bố của từng cột dữ liệu.

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated Chart, histogram

Description automatically generated

Graphical user interface, application

Description automatically generated

Lấy ra những cột dạng numeric

Graphical user interface, text, application

Description automatically generated

Đồ thị boxplot của dữ liệu

Graphical user interface, diagram, application, PowerPoint

Description automatically generated

Kết luận sau khi tiền xử lý:

Table

Description automatically generated

Table

Description automatically generatedApplication, table

Description automatically generated

Thống kê sau khi đã thay đổi dữ liệu:

|  |  |
| --- | --- |
| Thuộc tính | Giá trị thuộc tính |
| Age | interger |
| Sex | 0: male  1: female |
| cp | 0: type chest paint 1  1: type chest paint 2  2: type chest paint 3  3: type chest paint 4 |
| trestbps | interger |
| chol | interger |
| fbs | 1: fasting blood sugar & gt; 120mg/dl  0: not fasting… |
| restecg | 0:  1: |
| thalach | interger |
| exang | float |
| oldpeak | float |
| slope | 0: type 1  1: type 2  2: type 3 |
| target | 0: negative with heart disease  1: possitive with heart disease |

Sau khi tiến hành tiền xử lý dữ liệu, ta thu được tập dữ liệu có đầy đủ các giá trị của các thuộc tính. Các thuộc tính có độ phân tán dữ liệu và tính liên tục cạo được rời rạc hóa thành các khoảng dữ liệu. Các thuộc tính không có độ tương quan cao đồng nghĩa không mang lại độ phức tạp của dữ liệu. Lúc này, tập dữ liệu đã tinh gọn và chính xác hơn, sẵn sàng cho việc khai thác dữ liệu và đưa ra dự đoán.

* + 1. **Phân loại data**Graphical user interface, text, application, email

       Description automatically generatedGraphical user interface, text, application

       Description automatically generated

Chọn ra những giá trị lớn hơn alpha

Text, table

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

Sau Chi-square test và Anova test. Ta đã tìm được cái feature ảnh hưởng đến thuộc tính quyết định, đó là: chol, thalach, oldpeak, sex, cp, fbs, restecg, exang, slope, age

chọn chol, thalach, oldpeak, sex, cp, fbs, restecg, exang, slope, age làm input

target làm output

* + 1. **Data Scaling**

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated with low confidence

Chart, bar chart

Description automatically generated

Table

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart, box and whisker chart

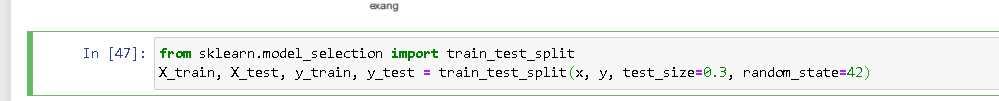
Description automatically generated

Chart, bar chart

Description automatically generated Chart, bar chart, box and whisker chart

Description automatically generated

Chia dữ liệu làm 2 phần train và test



Triển khai thuật toán Decision Tree

Graphical user interface, text

Description automatically generated

Table

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Mô hình cây quyết định:

Chart, scatter chart

Description automatically generated

Cây quyết định ID3:

Graphical user interface, text, application, email

Description automatically generated

Độ tin cậy của thuật toán

Table

Description automatically generated

Graphical user interface, application

Description automatically generated

Mô hình cây quyết định ID3:

Chart

Description automatically generated

Thuật toán Knearest:

Text

Description automatically generated

Chart, treemap chart

Description automatically generated

Thuật toán random forest:

Table

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Thuật toán hồi quy logistic:

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

Chart, treemap chart

Description automatically generated

# **CHƯƠNG III - KẾT QUẢ ĐẠT ĐƯỢC**

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Chart, bar chart

Description automatically generated

Graphical user interface, bar chart

Description automatically generated

Nhận xét:

Nhìn vào biểu đồ cũng như báo cáo về độ chính xác của lần lượt các thuật toán là

**Random Forest > ID3 > K-Nearest-Neighbors, Cart > Logistic Regression**

* Nếu quan tâm đến độ chính xác toàn mô hình tối ưu, ta có thể chọn thuật toán Random Forest để dự đoán tốt nhất cho bộ dữ liệu này với độ chính xác khoảng 96.23%
* Nếu quan tâm về độ nhanh của mô hình, ta có thể sử dụng ID3 hoặc Cart, tuy nhiên so về tổng thể thì thuật toán Random Forest vẫn là thuật toán tối ưu nhất.

.

# **CHƯƠNG IV - PHẦN MỀM DỰ ĐOÁN**

## **4.1 Tổng quan phần mềm**

### **4.1.1 Thuật toán được sử dụng**

Dựa vào kết quả thu được ở phần trước, nhóm quyết định sử dụng thuật toán Random Forest cho phần mềm này. Theo kết quả so sánh, thuật toán này dù có tốc độ không tốt nhưng nó lại cho độ chính xác cao nhất.

### **4.1.2 Các thuộc tính được sử dụng để đưa ra dự đoán**

Graphical user interface, text, application, email

Description automatically generated

=> Sau khi đưa vào tính toán độ tin cậy của các thuộc tính ta thu được 7 thuộc tính là **age, cp, trestbps, chol, thalach, oldpeak,slope** để đưa vào phần mềm. Đây là 7 thuộc tính có độ tin cậy cao nhất.

### **4.1.3 Giao diện và kiểm thử**

#### **4.1.3.1 Giao diện**

Graphical user interface, application

Description automatically generated

#### **4.1.3.2 Kiểm thử**

Bộ dữ liệu kiểm thử:

Table

Description automatically generated

Kết quả:

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Kết quả phần mềm đưa ra khớp với dữ liệu ban đầu.

## **4.2 Code phần mềm**

### **4.2.1 Code phần giao diện:**

from PyQt5 import QtCore, QtGui, QtWidgets

# from f import \*

class Ui\_MainWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(800, 524)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.label = QtWidgets.QLabel(self.centralwidget)

        self.label.setGeometry(QtCore.QRect(30, 10, 741, 51))

        font = QtGui.QFont()

        font.setPointSize(20)

        font.setBold(True)

        font.setWeight(60)

        self.label.setFont(font)

        self.label.setAlignment(QtCore.Qt.AlignCenter)

        self.label.setObjectName("label")

        self.label\_2 = QtWidgets.QLabel(self.centralwidget)

        self.label\_2.setGeometry(QtCore.QRect(30, 80, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label\_2.setFont(font)

        self.label\_2.setObjectName("label\_2")

        self.label\_3 = QtWidgets.QLabel(self.centralwidget)

        self.label\_3.setGeometry(QtCore.QRect(30, 110, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label\_3.setFont(font)

        self.label\_3.setObjectName("label\_3")

        self.label\_4 = QtWidgets.QLabel(self.centralwidget)

        self.label\_4.setGeometry(QtCore.QRect(30, 140, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label\_4.setFont(font)

        self.label\_4.setObjectName("label\_4")

        self.label\_5 = QtWidgets.QLabel(self.centralwidget)

        self.label\_5.setGeometry(QtCore.QRect(30, 170, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label\_5.setFont(font)

        self.label\_5.setObjectName("label\_5")

        self.label\_1 = QtWidgets.QLabel(self.centralwidget)

        self.label\_1.setGeometry(QtCore.QRect(30, 50, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label.setFont(font)

        self.label.setAlignment(QtCore.Qt.AlignCenter)

        self.label\_6 = QtWidgets.QLabel(self.centralwidget)

        self.label\_6.setGeometry(QtCore.QRect(30, 200, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        self.label\_6.setFont(font)

        self.label\_6.setObjectName("label\_6")

        font.setKerning(False)

        # self.label.setFont(font)

        # self.label.setAlignment(QtCore.Qt.AlignCenter)

        # self.label.setObjectName("label\_8")

        # self.label\_8 = QtWidgets.QLabel(self.centralwidget)

        # self.label\_8.setGeometry(QtCore.QRect(30, 260, 111, 31))

        # font = QtGui.QFont()

        # font.setPointSize(14)

        # font.setKerning(False)

        # self.label.setFont(font)

        # self.label.setAlignment(QtCore.Qt.AlignCenter)

        # self.label.setObjectName("label\_9")

        # self.label\_9 = QtWidgets.QLabel(self.centralwidget)

        # self.label\_9.setGeometry(QtCore.QRect(30, 290, 111, 31))

        # font = QtGui.QFont()

        # font.setPointSize(14)

        # font.setKerning(False)

        # self.label.setFont(font)

        # self.label.setAlignment(QtCore.Qt.AlignCenter)

        # self.label.setObjectName("label\_10")

        # self.label\_10 = QtWidgets.QLabel(self.centralwidget)

        # self.label\_10.setGeometry(QtCore.QRect(30, 310, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label\_1.setFont(font)

        self.label\_1.setObjectName("label\_1")

        self.lineEdit\_2 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_2.setGeometry(QtCore.QRect(150, 80, 611, 31))

        self.lineEdit\_2.setObjectName("lineEdit\_2")

        self.lineEdit\_3 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_3.setGeometry(QtCore.QRect(150, 110, 611, 31))

        self.lineEdit\_3.setObjectName("lineEdit\_3")

        self.lineEdit\_4 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_4.setGeometry(QtCore.QRect(150, 140, 611, 31))

        self.lineEdit\_4.setObjectName("lineEdit\_4")

        self.lineEdit\_1 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_1.setGeometry(QtCore.QRect(150, 50, 611, 31))

        self.lineEdit\_1.setObjectName("lineEdit\_1")

        self.lineEdit\_5 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_5.setGeometry(QtCore.QRect(150, 170, 611, 31))

        self.lineEdit\_5.setObjectName("lineEdit\_5")

        self.pushButton = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton.setGeometry(QtCore.QRect(50, 360, 251, 71))

        self.lineEdit\_7 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_7.setGeometry(QtCore.QRect(150, 230, 611, 31))

        self.lineEdit\_7.setObjectName("lineEdit\_7")

        # self.lineEdit\_8 = QtWidgets.QLineEdit(self.centralwidget)

        # self.lineEdit\_8.setGeometry(QtCore.QRect(150, 260, 611, 31))

        # self.lineEdit\_8.setObjectName("lineEdit\_8")

        # self.lineEdit\_9 = QtWidgets.QLineEdit(self.centralwidget)

        # self.lineEdit\_9.setGeometry(QtCore.QRect(150, 290, 611, 31))

        # self.lineEdit\_9.setObjectName("lineEdit\_9")

        # self.lineEdit\_10 = QtWidgets.QLineEdit(self.centralwidget)

        # self.lineEdit\_10.setGeometry(QtCore.QRect(150, 320, 611, 31))

        # self.lineEdit\_10.setObjectName("lineEdit\_10")

        # self.lineEdit\_10 = QtWidgets.QLineEdit(self.centralwidget)

        # self.lineEdit\_10.setGeometry(QtCore.QRect(150, 350, 611, 31))

        # self.lineEdit\_10.setObjectName("lineEdit\_10")

        font = QtGui.QFont()

        font.setPointSize(32)

        self.pushButton.setFont(font)

        self.pushButton.setObjectName("pushButton")

        self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)

        self.pushButton\_2.setGeometry(QtCore.QRect(490,360, 251, 71))

        font = QtGui.QFont()

        font.setPointSize(32)

        self.pushButton\_2.setFont(font)

        self.pushButton\_2.setObjectName("pushButton\_2")

        self.lineEdit\_6 = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit\_6.setGeometry(QtCore.QRect(150, 200, 611, 31))

        self.lineEdit\_6.setObjectName("lineEdit\_6")

        self.label\_7 = QtWidgets.QLabel(self.centralwidget)

        self.label\_7.setGeometry(QtCore.QRect(30, 230, 111, 31))

        font = QtGui.QFont()

        font.setPointSize(14)

        font.setKerning(False)

        self.label\_7.setFont(font)

        self.label\_7.setObjectName("label\_7")

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QtWidgets.QMenuBar(MainWindow)

        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 21))

        self.menubar.setObjectName("menubar")

        MainWindow.setMenuBar(self.menubar)

        self.statusbar = QtWidgets.QStatusBar(MainWindow)

        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.pushButton.clicked.connect(self.Crun)

        self.pushButton\_2.clicked.connect(self.Clr)

        # train()

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "Phần mềm dự đoán"))

        self.label.setText(\_translate("MainWindow", "DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH TIM"))

        self.label\_1.setText(\_translate("MainWindow", "age"))

        self.label\_2.setText(\_translate("MainWindow", "cp"))

        self.label\_3.setText(\_translate("MainWindow", "trestbps"))

        self.label\_4.setText(\_translate("MainWindow", "chol"))

        self.label\_5.setText(\_translate("MainWindow", "thalach"))

        self.label\_6.setText(\_translate("MainWindow", "oldpeak"))

        self.label\_7.setText(\_translate("MainWindow", "slope"))

        # self.label\_9.setText(\_translate("MainWindow", "sex"))

        # self.label\_10.setText(\_translate("MainWindow", "slope"))

        self.pushButton.setText(\_translate("MainWindow", "CHẠY"))

        self.pushButton\_2.setText(\_translate("MainWindow", "XÓA"))

        # self.label\_6.setText(\_translate("MainWindow", "thalach"))

    def Clr(self) -> None:

        self.lineEdit\_1.clear()

        self.lineEdit\_2.clear()

        self.lineEdit\_3.clear()

        self.lineEdit\_4.clear()

        self.lineEdit\_5.clear()

        self.lineEdit\_6.clear()

        self.lineEdit\_7.clear()

        # self.lineEdit\_8.clear()

        # self.lineEdit\_9.clear()

        # self.lineEdit\_10.clear()

    def Crun(self) -> None:

        my\_dict =   {"age":float(self.lineEdit\_1.text()), "cp":float(self.lineEdit\_2.text()), "trestbps":float(self.lineEdit\_3.text())

        , "chol":float(self.lineEdit\_4.text()), "thalach":float(self.lineEdit\_5.text()), "oldpeak":float(self.lineEdit\_6.text()),

        "slope":float(self.lineEdit\_7.text())}

        t=str('Bệnh nhân')

        print(my\_dict)

        output = check\_input(my\_dict)

        print(output)

        msg = QtWidgets.QMessageBox()

        msg.setIcon(QtWidgets.QMessageBox.Information)

        a = ""

        if output == 0:

            a="KHÔNG CÓ KHẢ NĂNG"

            msg.setInformativeText(" {} {}  bị mắc bệnh tim".format(t,str(a)))

        elif output ==1:

            a="CÓ KHẢ NĂNG"

            msg.setInformativeText(" {} {}  bị mắc bệnh tim".format(t,str(a)))

        msg.setWindowTitle("Kết quả")

        msg.exec\_()

    # from sklearn.metrics import accuracy\_score

if \_\_name\_\_ == '\_\_main\_\_':

        train()

        app = QtWidgets.QApplication(sys.argv)

        MainWindow = QtWidgets.QMainWindow()

        ui = Ui\_MainWindow()

        ui.setupUi(MainWindow)

        MainWindow.show()

        sys.exit(app.exec\_())

### **4.2.2 Code phần xử lý**

from cgi import test

from re import X

from tkinter import Y

from PyQt5 import QtCore, QtGui, QtWidgets

# app = FastAPI(debug=True)

# @app.get('/')

# def home():

#     return {'text':'Khanh cute

# if \_\_name\_\_ == '\_\_main\_\_':

#     uvicorn.run(app)

#  phần xử lí

import pandas as pd

from sklearn.metrics import accuracy\_score

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

import os

import sys

import pickle

import numpy as np

#For training

def train() -> None:

    with open('heart\_Disease\_prediction\_new\_final.csv') as f:

        df = pd.read\_csv(f)

    # df = df.drop(['contact', 'poutcome'], axis=1)

    df\_filtered = df.replace('unknown',np.nan)

    df\_filtered.dropna(inplace=True)

    df\_filtered.reset\_index(drop=True, inplace=True)

    dataset = df\_filtered.copy()

    accuracies = {}

    times = {}

    from sklearn.preprocessing import LabelEncoder

    le = LabelEncoder()

    for col in dataset.columns[ [i == object for i in dataset.dtypes] ]:

        dataset.loc[:,col] = le.fit\_transform(dataset[col])

    dataset = dataset[['age', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'slope','target']]

    x = dataset.iloc[:, :-1].values

    y = dataset.iloc[:, -1].values

    from sklearn.compose import ColumnTransformer

    from sklearn.preprocessing import OneHotEncoder

    ct = ColumnTransformer(transformers=[], remainder='passthrough' )

    x = np.array(ct.fit\_transform(x))

    from sklearn.preprocessing import LabelEncoder

    le = LabelEncoder()

    y = le.fit\_transform(y)

    from sklearn.compose import ColumnTransformer

    from sklearn.preprocessing import OneHotEncoder

    ct = ColumnTransformer(transformers=[], remainder='passthrough' )

    x = np.array(ct.fit\_transform(x))

    from sklearn.preprocessing import LabelEncoder

    le = LabelEncoder()

    y = le.fit\_transform(y)

#train test split

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

    x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=42)

    from sklearn.ensemble import RandomForestClassifier

    classifier = RandomForestClassifier(n\_estimators =10, criterion='entropy', random\_state=0)

    classifier.fit(x\_train, y\_train)

    R= classifier.fit(x\_train,y\_train)

#Save Model As Pickle File

    with open('R.pkl','wb') as m:

        pickle.dump(R,m)

    test(x\_test,y\_test)

#Test accuracy of the model

def test(X\_test,Y\_test):

    with open('R.pkl','rb') as mod:

        p=pickle.load(mod)

    pre=p.predict(X\_test)

    print (accuracy\_score(Y\_test,pre)) #Prints the accuracy of the model

def find\_data\_file(filename):

    if getattr(sys, "frozen", False): # The application is frozen.

        datadir = os.path.dirname(sys.executable)

    else:

# The application is not frozen.

        datadir = os.path.dirname( \_\_file\_\_)

    return os.path.join(datadir, filename)

def check\_input(data) ->int :

    df=pd.DataFrame(data=data,index=[0])

    with open(find\_data\_file('R.pkl'),'rb') as model:

        p=pickle.load(model)

    op=p.predict(df)

    return op

# **CHƯƠNG V - KẾT LUẬN**

**5.1 Ưu điểm, hạn chế của từng thuật toán**

***5.1.1 Cây quyết định***

***-* Ưu điểm:**

* Thuật toán đơn giản, trực quan, không quá phức tạp để hiểu ngay lần đầu tiên.
* Bộ dữ liệu training không nhất thiết phải quá lớn để tiến hành xây dựng mô hình phân tích.
* Một số thuật toán cây quyết định có khả năng xử lý dữ liệu bị missing và dữ liệu bị lỗi mà không cần áp dụng phương pháp như “imputing missing values” hay loại bỏ. Ít bị ảnh hưởng bởi các dữ liệu ngoại lệ.
* Không cần phải có các giả định ban đầu về các quy luật phân phối như trong thống kê, và nhờ đó kết quả phân tích có được là khách quan, “tự nhiên” nhất.
* Có thể giúp chúng ta phân loại đối tượng dữ liệu theo biến mục tiêu có nhiều lớp, nhiều nhóm khác nhau (multi-class classification) đặc biệt nếu biến mục tiêu là dạng biến định lượng phức tạp.
* Có thể áp dụng linh hoạt cho các biến target, biến mục tiêu.
* Mang lại kết quả dự báo có độ chính xác cao, dễ dàng thực hiện, nhanh chóng trong việc huấn luyện, không cần phải chuyển đổi các biến.
* Dễ diễn giải hay giải thích đến người nghe, người xem – những người muốn hiểu rõ về kết quả phân tích nhưng không có kiến thức gì về khoa học dữ liệu.
* Nói lên được mối liên hệ giữa các biến, các thuộc tính dữ liệu một cách trực quan nhất.
* Ngoài kinh tế, tài chính, thuật toán cây quyết định có thể được ứng dụng trong lĩnh vực y tế, nông nghiệp, sinh học.

**Khuyết điểm:**

* Thuật toán cây quyết định hoạt động hiệu quả trên bộ dữ liệu đơn giản có ít biến dữ liệu liên hệ với nhau, và ngược lại nếu áp dụng cho bộ dữ liệu phức tạp.
* Khi được áp dụng với bộ dữ liệu phức tạp, nhiều biến và thuộc tính khác nhau có thể dẫn đến mô hình bị overfitting, quá khớp với dữ liệu training dẫn đến vấn đề không đưa ra kết quả phân loại chính xác khi áp dụng cho dữ liệu test, và dữ liệu mới.
* Giá trị phương sai cao, khi có sự thay đổi nhỏ trong bộ dữ liệu có thể gây ảnh hưởng đến cấu trúc của mô hình.
* Thuật toán cây quyết định chỉ áp dụng cho biến định tính (classification tree) nếu phân loại sai có thể dẫn đến sai lầm nghiêm trọng.
* Thuật toán cây quyết định có khả năng “bias” hay thiên vị nếu bộ dữ liệu không được cân bằng.
* Yêu cầu bộ dữ liệu training và test phải được chuẩn bị hoàn hảo, chất lượng tốt phải được cân đối theo các lớp, các nhóm trong biến mục tiêu.
* Không  “hỗ trợ” kỹ thuật hay khả năng “truy vấn ngược”.

***5.1.2. Random Forest***

- Ưu điểm:

* Cải tiến hơn với thuật toán cây quyết định, giải quyết được sự nhiễu khi bộ dữ liệu thay đổi.

- Nhược điểm

* Nhược điểm chính của Random Forest đó chính là khối lượng tính toán lớn, tuy nhiên với năng lực tính toán ngày càng tăng của máy tính (theo cấp lũy thừa) thì hạn chế của Random Forest không phải là vấn đề lớn.

***5.1.3 Logistic Regression***

- Ưu điểm

* Phân lớp tốt khi dữ liệu có khả năng phân tách tuyến tích (linearly separable).
* Dễ triển khai và training.

- Nhược điểm

* Dễ bị ảnh hưởng bởi nhiễu.
* Không thể giải quyết các bài toán phi tuyến tính (non-linear).
* Nhạy cảm với overfitting.

***5.1.4 K-nearest Neighbors***

**- Ưu điểm:**

* Thuật toán đơn giản, dễ dàng triển khai.
* Độ phức tạp tính toán nhỏ.
* Xử lý tốt với tập dữ liệu nhiễu

**- Nhược điểm**

* Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác
* Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
* Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

**5.2 Hướng phát triển**

* Nghiên cứu và tìm hiểu thêm về dataset, điều chỉnh lại các thuộc tính và các tiền xử lý để đạt độ chính xác cao hơn.
* Tiếp tục áp dụng các thuật toán phân lớp khác như SVM, Artificial Neuron Network, … để tìm thuật toán tối ưu.

# **Tài liệu tham khảo**

* 1. Slide môn hệ hỗ trợ ra quyết định trường đại học CNTT – DHQG TPHCM.
  2. Ma trận nhầm lẫn: [What is Precision, Recall and F1-score? - The Conscious's notes (wordpress.com)](https://caihuuthuc.wordpress.com/2020/02/23/precision-recall-va-f1-score-la-gi/)
  3. Các thuật toán: [CART: Classification and Regression Trees for Clean but Powerful Models | by Saul Dobilas | Towards Data Science](https://towardsdatascience.com/cart-classification-and-regression-trees-for-clean-but-powerful-models-cc89e60b7a85)
  4. Các loại data: [What is Ordinal Data? [Definition, Analysis & Examples] (careerfoundry.com)](https://careerfoundry.com/en/blog/data-analytics/what-is-ordinal-data/)
  5. Các loại machine learning: [4 Machine Learning Approaches that Every Data Scientist Should Know | by Orhan G. Yalçın | Towards Data Science](https://towardsdatascience.com/4-machine-learning-approaches-that-every-data-scientist-should-know-e3a9350ec0b9#:~:text=Unsupervised%20learning%20is%20a%20type%20of%20machine%20learning,datasets%2C%20which%20do%20not%20contain%20labels.%20Figure%204.)
  6. Video youtube: [Project 9. Heart Disease Prediction using Machine Learning with Python | Machine Learning Projects - YouTube](https://www.youtube.com/watch?v=qmqCYC-MBQo)
  7. Dataset: [Heart Disease Dataset | Kaggle](https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset)

# **Hết.**