

TRƯỜNG ĐẠI HỌC THỦY LỢI



BÁO CÁO BÀI TẬP LỚN HỌC PHẦN : HỌC MÁY

ĐỀ TÀI: PHẦN MỀM CHUẨN ĐOÁN BỆNH TIM

Người hướng dẫn: Dr. Tạ Quang Chiêu

Lớp: 64HTT4

Nhóm sinh viên thực hiện:

Nhóm 6

Thành viên nhóm:

- | | |
|----------------------|------------|
| 1. Nguyễn Thanh Bình | 2251161954 |
| 2. Hoàng Đức Long | 2251162062 |
| 3. Nguyễn Ngọc Long | 2251162063 |
| 4. Đoàn Hải Yến | 2251162214 |

Hà Nội, 07 Tháng 01 Năm 2025

MỤC LỤC

CHƯƠNG I: KHỞI ĐỘNG DỰ ÁN.....	4
1. Giới thiệu bài toán, mô tả bài toán	4
1.1. Giới thiệu bài toán	4
1.2. Mô tả bài toán	5
2. Tổ chức dự án	7
2.1. Cấu trúc thư viện	7
2.2. Giải thích cấu trúc file chính và file dữ liệu :	7
3. Các mô hình thuật toán được sử dụng trong dự án.....	9
3.1. Cách thức huấn luyện chung:	9
3.2. Các mô hình thuật toán.....	10
3.2.1. Mô hình Logistic Regression (LR):.....	10
3.2.2. Mô hình Perceptron Learning Algorithm (PLA):	10
3.2.3. Mô hình Decision Tree (ID3):.....	11
CHƯƠNG II: THỰC NGHIỆM	13
1. Mô tả dữ liệu dự án	13
1.1 Mục đích của bài toán:	13
1.2 Input và Output:.....	13
1.3 Tóm tắt công việc thực hiện của dự án:.....	13
1.4 Công việc chính:	14
1.5 Mô tả tập dữ liệu của bài toán:	14
1.6 Thuật toán sử dụng trong bài toán:.....	15
1.7 Kết quả và ứng dụng thực tiễn:	15
2. Cách thức triển khai.....	15
2.1 Import thư viện	15
2.2. Tải và hiển thị dữ liệu từ các tệp CSV	16
2.3. Huấn luyện Logistic Regression (LR).....	23
2.3.1 Không dùng thư viện:.....	23
2.3.2 Dùng thư viện:.....	27

2.4. Độ chính xác của mô hình thuật toán Logistic Regression (LR):	27
2.5. Độ chính xác của mô hình thuật toán Perceptron Learning Algorithm (PLA): .	30
2.6. Độ chính xác của mô hình thuật toán Decision Tree (ID3):	32
3. So sánh độ chính xác của ba mô hình:	34
4. Tạo giao diện phần mềm:	36
CHƯƠNG III: LỜI KẾT.....	38
Tài liệu tham khảo:	39

CHƯƠNG I: KHỞI ĐỘNG DỰ ÁN

1. Giới thiệu bài toán, mô tả bài toán

1.1. Giới thiệu bài toán

- Các bệnh tim mạch là nguyên nhân hàng đầu gây tử vong trên toàn cầu, chiếm khoảng 17,9 triệu ca tử vong mỗi năm, tương đương 32% tổng số ca tử vong toàn cầu (theo Tổ chức Y tế Thế giới). Việc phát hiện sớm các dấu hiệu nguy cơ và can thiệp kịp thời có thể cứu sống hàng triệu người, đồng thời giảm tải áp lực lên hệ thống y tế. Tuy nhiên, chẩn đoán bệnh tim không phải lúc nào cũng dễ dàng, đặc biệt khi các triệu chứng có thể không rõ ràng hoặc tương đồng với các bệnh khác.
- Trong bối cảnh đó, việc áp dụng các kỹ thuật học máy (machine learning) để dự đoán nguy cơ mắc bệnh tim dựa trên dữ liệu y tế là một giải pháp đầy tiềm năng. Học máy cho phép khai thác và phân tích các bộ dữ liệu y tế phức tạp, từ đó nhận diện những yếu tố có ảnh hưởng lớn đến sức khỏe tim mạch. Với sự hỗ trợ của các thuật toán học máy, việc dự đoán bệnh tim trở nên chính xác và hiệu quả hơn, hỗ trợ bác sĩ và bệnh nhân đưa ra quyết định điều trị kịp thời.
- Bài toán đặt ra là làm thế nào để xây dựng một hệ thống dự đoán nguy cơ mắc bệnh tim với độ chính xác cao, dựa trên các đặc điểm sức khỏe của bệnh nhân. Những yếu tố như tuổi tác, giới tính, huyết áp, nhịp tim, hay lượng đường trong máu đều có thể là những chỉ báo quan trọng, và chúng cần được mô hình hóa một cách hiệu quả thông qua các thuật toán học máy.
- Việc ứng dụng học máy trong dự đoán bệnh tim sẽ góp phần mang lại những lợi ích vượt trội cho ngành y tế, ví dụ như:
 - Hỗ trợ chẩn đoán sớm nguy cơ bệnh tim cho bệnh nhân.
 - Tăng hiệu quả của các quyết định y khoa.
 - Giảm thiểu chi phí xét nghiệm không cần thiết.
 - Nâng cao nhận thức của cộng đồng về tầm quan trọng của việc theo dõi sức khỏe tim mạch.

1.2. Mô tả bài toán

Tổng quát chung:

- Dự án lần này mà nhóm chúng em thực hiện và phát triển sẽ tập trung vào việc xây dựng một hệ thống phần mềm sử dụng các mô hình học máy để dự đoán nguy cơ mắc bệnh tim dựa trên dữ liệu y tế. Cụ thể, các mô hình học máy sẽ được thử nghiệm trên một tập dữ liệu đầu vào cụ thể, hệ thống sẽ được phát triển bằng cách áp dụng các thuật toán học máy phổ biến như: **Logistic Regression, Perceptron Learning Algorithm (PLA)** và **Decision Tree (ID3)**. Và sau đó, tiến hành so sánh các kết quả dự đoán thu được để xác định mô hình nào hoạt động hiệu quả nhất dựa trên các tiêu chí đánh giá cụ thể.
- Dữ liệu được sử dụng trong bài toán gồm các thông tin về sức khỏe cá nhân như:
 - Tuổi (Age)
 - Giới tính (Sex)
 - Loại đau ngực (Chest Pain Type)
 - Huyết áp nghỉ ngơi (Resting Blood Pressure)
 - Mức đường huyết lúc đói (Fasting Blood Sugar)
 - Nhịp tim tối đa (Maximum Heart Rate)
 - Điện tâm đồ khi nghỉ (Resting ECG)
 - Tình trạng đau thắt ngực do vận động (Exercise Induced Angina)

Các trường thông tin được sử dụng:

1. Tuổi (Age):

- Kiểu dữ liệu: số nguyên.
- Phản ánh mức độ nguy cơ cao hơn khi tuổi tăng.

2. Giới tính (Sex):

- Kiểu dữ liệu: nhị phân (0 = nữ, 1 = nam).
- Mức độ nguy cơ bệnh tim có thể khác nhau giữa nam và nữ.

3. Loại đau ngực (Chest Pain Type):

- Giá trị:
 - 1 = Đau thắt ngực điển hình.
 - 2 = Đau thắt ngực không điển hình.
 - 3 = Đau không liên quan đến tim.
 - 4 = Không có đau ngực.

4. **Huyết áp nghỉ ngơi (Resting Blood Pressure):**

- Kiểu dữ liệu: số thực (mmHg).
- Chỉ số huyết áp cao có thể liên quan đến nguy cơ mắc bệnh tim.

5. **Đường huyết lúc đói (Fasting Blood Sugar):**

- Giá trị nhị phân: 0 (< 120 mg/dL), 1 (\geq 120 mg/dL).
- Mức đường huyết cao là một yếu tố nguy cơ quan trọng.

6. **Nhịp tim tối đa đạt được (Maximum Heart Rate):**

- Kiểu dữ liệu: số nguyên.
- Thể hiện khả năng đáp ứng của tim khi hoạt động thể chất.

7. **Điện tâm đồ nghỉ (Resting ECG):**

- Giá trị:
 - 0 = Bình thường.
 - 1 = Có bất thường ST-T.
 - 2 = Có khả năng phì đại thất trái.

8. **Kết quả (Target):**

- Nhãn kết quả: 0 (không mắc bệnh tim), 1 (mắc bệnh tim).

Chức năng và mục tiêu:

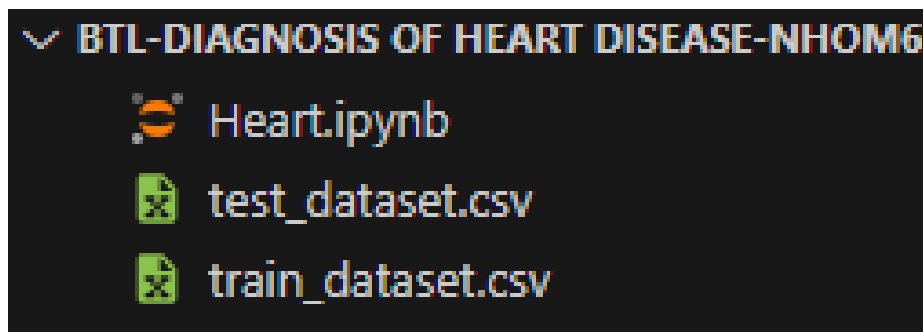
- **Chức năng:**

- Người dùng nhập các thông số sức khỏe vào hệ thống thông qua giao diện đơn giản được phát triển bằng **CustomTkinter**.

- Hệ thống sẽ dự đoán khả năng mắc bệnh tim và hiển thị kết quả ngay lập tức.
 - Ngoài ra, hệ thống cung cấp báo cáo chi tiết về hiệu suất của các mô hình học máy, hiển thị biểu đồ như **ma trận nhầm lẫn** và **đường cong ROC**.
- **Mục tiêu:**
- Cung cấp một công cụ hỗ trợ y tế hiệu quả cho các bác sĩ và chuyên gia y tế.
 - Nâng cao khả năng chẩn đoán sớm, từ đó giảm nguy cơ tử vong do bệnh tim.

2. Tổ chức dự án

2.1. Cấu trúc thư viện



2.2 Giải thích cấu trúc file chính và file dữ liệu :

- **Heart.ipynb:** File sẽ thực hiện dự đoán bệnh tim bằng cách xử lý dữ liệu từ các tệp CSV, kiểm tra và trực quan hóa thông tin. Sau khi chuẩn hóa dữ liệu, các mô hình Logistic Regression, PLA và ID3 được huấn luyện và đánh giá dựa trên độ chính xác và ma trận nhầm lẫn. Kết quả được trực quan hóa bằng biểu đồ, và cuối cùng, một giao diện dự đoán bệnh tim được xây dựng bằng tkinter.

```

Heartipynb > ...
+ Code + Markdown ...
Select Kernel

import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import customtkinter as cus

[1] Python

KHÁM PHÁ DỮ LIỆU

# Đọc dữ liệu từ các tệp
train_file_path = 'train_dataset.csv'
test_file_path = 'test_dataset.csv'

train_data = pd.read_csv(train_file_path)
test_data = pd.read_csv(test_file_path)

train_data.head()

[2] Python
...

```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	54.0	1.0	0	124.0	266.0	0.0	2.0	109.0	1.0	2.2	2.0	1.0	2	1
1	46.0	1.0	2	101.0	197.0	1.0	0.0	156.0	0.0	0.0	1.0	0.0	2	0
2	43.0	1.0	1	130.0	315.0	0.0	0.0	162.0	0.0	1.9	1.0	1.0	1	0
3	47.0	1.0	1	138.0	257.0	0.0	2.0	156.0	0.0	0.0	1.0	0.0	1	0
4	58.0	1.0	0	150.0	270.0	0.0	2.0	111.0	1.0	0.8	1.0	0.0	2	1

```

# Kiểm tra xem có giá trị NaN (giá trị thiếu)
train_data.isna().sum()

[3] Python
...
Age      0
Sex      0
ChestPain 0

```

- **train_dataset.csv**: là tập dữ liệu dùng để huấn luyện mô hình học máy. File chứa các đặc điểm sức khỏe như tuổi, giới tính, huyết áp, cholesterol cùng với nhãn mục tiêu (tình trạng bệnh tim: có hoặc không). Mô hình học cách liên kết các đặc điểm này với tình trạng bệnh, nhằm đưa ra dự đoán chính xác trên các dữ liệu tương tự.

```

test_dataset.csv > data
1 Age,Sex,ChestPain,RestBP,Chol,Fbs,RestECG,MaxHR,ExAng,Oldpeak,Slope,Ca,Thal,AHD
2 41.0,1.0,2,120.0,157.0,0.0,0.0,182.0,0.0,0.0,1.0,0.0,1,0
3 49.0,0.0,0,130.0,269.0,0.0,0.0,163.0,0.0,0.0,1.0,0.0,1,0
4 66.0,1.0,0,120.0,302.0,0.0,2.0,151.0,0.0,0.4,2.0,0.0,1,0
5 57.0,0.0,0,128.0,303.0,0.0,2.0,159.0,0.0,0.0,1.0,1.0,1,0
6 54.0,1.0,1,150.0,232.0,0.0,2.0,165.0,0.0,1.6,1.0,0.0,2,0
7 42.0,0.0,1,120.0,209.0,0.0,0.0,173.0,0.0,0.0,2.0,0.0,1,0
8 58.0,1.0,1,132.0,224.0,0.0,2.0,173.0,0.0,3.2,1.0,2.0,2,1
9 49.0,0.0,2,134.0,271.0,0.0,0.0,162.0,0.0,0.0,2.0,0.0,1,0
10 60.0,1.0,1,140.0,185.0,0.0,2.0,155.0,0.0,3.0,2.0,0.0,1,1
11 50.0,0.0,0,110.0,254.0,0.0,2.0,159.0,0.0,0.0,1.0,0.0,1,0
12 67.0,1.0,0,100.0,299.0,0.0,2.0,125.0,1.0,0.9,2.0,2.0,1,1
13 45.0,0.0,2,112.0,160.0,0.0,0.0,138.0,0.0,0.0,2.0,0.0,1,0
14 59.0,1.0,0,135.0,234.0,0.0,0.0,161.0,0.0,0.5,2.0,0.0,2,0
15 49.0,1.0,1,120.0,188.0,0.0,0.0,139.0,0.0,2.0,2.0,3.0,2,1
16 54.0,1.0,0,120.0,188.0,0.0,0.0,113.0,0.0,1.4,2.0,1.0,2,1
17 45.0,1.0,3,110.0,264.0,0.0,0.0,132.0,0.0,1.2,2.0,0.0,2,1
18 62.0,0.0,0,140.0,268.0,0.0,2.0,160.0,0.0,3.6,3.0,2.0,1,1
19 58.0,0.0,3,150.0,283.0,1.0,2.0,162.0,0.0,1.0,1.0,0.0,1,0
20 58.0,1.0,0,128.0,216.0,0.0,2.0,131.0,1.0,2.2,2.0,3.0,2,1
21 57.0,1.0,0,130.0,131.0,0.0,0.0,115.0,1.0,1.2,2.0,1.0,2,1
22 29.0,1.0,2,130.0,204.0,0.0,2.0,202.0,0.0,0.0,1.0,0.0,1,0
23 42.0,1.0,2,120.0,295.0,0.0,0.0,162.0,0.0,0.0,1.0,0.0,1,0
24 46.0,1.0,0,140.0,311.0,0.0,0.0,120.0,1.0,1.8,2.0,2.0,2,1
25 46.0,0.0,2,105.0,204.0,0.0,0.0,172.0,0.0,0.0,1.0,0.0,1,0
26 57.0,1.0,0,110.0,335.0,0.0,0.0,143.0,1.0,3.0,2.0,1.0,2,1
27 64.0,1.0,3,170.0,227.0,0.0,2.0,155.0,0.0,6.6,2.0,0.0,2,0
28 68.0,0.0,1,120.0,211.0,0.0,2.0,115.0,0.0,1.5,2.0,0.0,1,0
29 67.0,1.0,0,120.0,237.0,0.0,0.0,71.0,0.0,1.0,2.0,0.0,1,1
30 37.0,1.0,1,130.0,210.0,0.0,0.0,187.0,0.0,1.5,3.0,0.0,1,0

```

- **test_dataset.csv**: là tập dữ liệu kiểm tra, sử dụng để đánh giá mô hình sau khi huấn luyện. File chứa các đặc điểm sức khỏe tương tự tập huấn luyện nhưng không có nhãn mục tiêu, giúp kiểm tra khả năng dự đoán của mô hình trên dữ liệu mới chưa từng thấy trước đó.


```
test_dataset.csv > data
1 Age,Sex,ChestPain,RestBP,Chol,Fbs,RestECG,MaxHR,ExAng,Oldpeak,Slope,Ca,Thal,AHD
2 41.0,1.0,2,120.0,157.0,0.0,0,182.0,0.0,0.0,1.0,0.0,1,0
3 49.0,0.0,0,120.0,289.0,0.0,0,163.0,0.0,0.0,1.0,0.0,1,0
4 66.0,1.0,0,120.0,302.0,0.0,2,151.0,0.0,0.0,4.2,0.0,1,0
5 57.0,0.0,0,120.0,303.0,0.0,2,159.0,0.0,0.0,1.0,1.0,1,0
6 54.0,1.0,1,150.0,232.0,0.0,2,165.0,0.0,1.0,1.0,0.0,2,0
7 42.0,0.0,1,120.0,209.0,0.0,0,173.0,0.0,0.0,2.0,0.0,1,0
8 58.0,1.0,1,132.0,224.0,0.0,2,173.0,0.0,3.2,1.0,2.0,2,1
9 49.0,0.0,2,134.0,271.0,0.0,0,162.0,0.0,0.0,2.0,0.0,1,0
10 60.0,1.0,1,140.0,185.0,0.0,2,155.0,0.0,3.0,2.0,0.0,1,1
11 50.0,0.0,0,110.0,254.0,0.0,2,159.0,0.0,0.0,1.0,0.0,1,0
12 67.0,1.0,0,100.0,289.0,0.0,2,125.0,1.0,0.0,2.0,2.0,1,1
13 45.0,0.0,2,112.0,160.0,0.0,0,138.0,0.0,0.0,2.0,0.0,1,0
14 59.0,1.0,0,135.0,234.0,0.0,0,161.0,0.0,0.5,2.0,0.0,2,0
15 49.0,1.0,1,120.0,188.0,0.0,0,139.0,0.0,2.0,2.0,3.0,2,1
16 54.0,1.0,0,120.0,188.0,0.0,0,113.0,0.0,1.4,2.0,1.0,2,1
17 45.0,1.0,3,110.0,264.0,0.0,0,132.0,0.0,1.2,2.0,0.0,2,1
18 62.0,0.0,0,140.0,268.0,0.0,2,160.0,0.0,3.0,3.0,2.0,1,1
19 58.0,0.0,3,150.0,283.0,1.0,2,162.0,0.0,1.0,1.0,0.0,1,0
20 58.0,1.0,0,120.0,216.0,0.0,2,131.0,1.0,2.2,2.0,3.0,2,1
21 57.0,1.0,0,130.0,121.0,0.0,0,115.0,1.0,1.2,2.0,1.0,2,1
22 29.0,1.0,2,130.0,204.0,0.0,2,202.0,0.0,0.0,1.0,0.0,1,0
23 42.0,1.0,2,120.0,295.0,0.0,0,162.0,0.0,0.0,1.0,0.0,1,0
24 46.0,1.0,0,140.0,311.0,0.0,0,120.0,1.0,1.0,2.0,2.0,2,1
25 46.0,0.0,2,105.0,204.0,0.0,0,172.0,0.0,0.0,1.0,0.0,1,0
26 57.0,1.0,0,110.0,335.0,0.0,0,143.0,1.0,3.0,2.0,1.0,2,1
27 64.0,1.0,3,170.0,227.0,0.0,2,155.0,0.0,0.0,2.0,0.0,2,0
28 68.0,0.0,1,120.0,211.0,0.0,2,115.0,0.0,1.5,2.0,0.0,1,0
29 67.0,1.0,0,120.0,237.0,0.0,0,71.0,0.0,1.0,2.0,0.0,1,1
30 37.0,1.0,1,130.0,150.0,0.0,0,127.0,0.0,3.0,0.0,0.0,1,0
```

🚦 Trình tự hoạt động:

1. Chuẩn bị dữ liệu: Tải và kiểm tra dữ liệu từ train_dataset.csv (dùng huấn luyện mô hình) và test_dataset.csv (dùng kiểm tra mô hình).
2. Tiền xử lý dữ liệu: Xử lý và chuẩn hóa dữ liệu để đưa vào mô hình.
3. Huấn luyện mô hình: Sử dụng các thuật toán Logistic Regression, PLA, và ID3 để huấn luyện trên dữ liệu huấn luyện.
4. Đánh giá mô hình: Kiểm tra mô hình trên test_dataset.csv và đánh giá độ chính xác, hiển thị ma trận nhầm lẫn.
5. Dự đoán và giao diện: Xây dựng giao diện tkinter để người dùng nhập thông tin và nhận dự đoán bệnh tim.

3. Các mô hình thuật toán được sử dụng trong dự án

3.1. Cách thức huấn luyện chung:

Quá trình huấn luyện mô hình học máy bao gồm việc chuẩn bị dữ liệu (chia thành tập huấn luyện và kiểm tra, tiền xử lý dữ liệu), chọn mô hình phù hợp (như trong dự án này là Logistic Regression, PLA, ID3), sau đó huấn luyện mô hình bằng tập huấn luyện để tối ưu tham số. Sau khi huấn luyện, mô hình sẽ được đánh giá bằng dữ liệu kiểm tra để

đo lường độ chính xác và các chỉ số hiệu suất khác. Cuối cùng, mô hình sẽ được sử dụng để dự đoán và có thể được điều chỉnh thêm để cải thiện kết quả.

3.2. Các mô hình thuật toán

3.2.1. Mô hình Logistic Regression (LR):

Input: Tập dữ liệu huấn luyện đã được gán nhãn ($X_{\text{train}}, y_{\text{train}}$).

Output: Hàm phân loại $h_{\theta}(x) = \alpha(z)$, với $z = X \cdot \theta + b$ là hàm sigmoid, giúp phân loại dữ liệu thành hai lớp: dương tính (1) hoặc âm tính (0).

Hàm mất mát:

$$L(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Cách thực hiện:

- Bước 1: Khởi tạo trọng số θ và bias b ngẫu nhiên.
- Bước 2: Tính giá trị $z^{(i)} = X^{(i)} \cdot \theta + b$ cho mỗi mẫu, sau đó áp dụng hàm sigmoid để tính xác suất: $h_{\theta}(x^{(i)}) = \frac{1}{1 + e^{-z^{(i)}}}$
- Bước 3: Tính hàm mất mát $L(\theta)$ bằng cách so sánh giữa nhãn thực tế $y^{(i)}$ và dự đoán $h_{\theta}(x^{(i)})$
- Bước 4: Cập nhật trọng số θ và bias b theo gradient descent:

$$\theta := \theta - \alpha \cdot \frac{\partial L(\theta)}{\partial \theta}, \quad b := b - \alpha \cdot \frac{\partial L(\theta)}{\partial b}$$

- Bước 5: Lặp lại Bước 2 và Bước 3 cho đến khi hàm mất mát hội tụ.

3.2.2. Mô hình Perceptron Learning Algorithm (PLA):

Input: Tập dữ liệu huấn luyện đã được gán nhãn ($X_{\text{train}}, y_{\text{train}}$).

Output: Siêu phẳng phân chia dữ liệu sao cho các mẫu cùng nhãn thuộc cùng một phía của siêu phẳng, được mô tả bằng hàm phân loại $f(x, w) = w^T x$

Hàm mất mát: $J(w) = \sum_{x_i \in M} (-y_i \cdot w^T x_i)$

Cách thực hiện:

- Bước 1: Khởi tạo trọng số w_0 ngẫu nhiên.
- Bước 2: Kiểm tra tất cả các điểm dữ liệu. Nếu không có điểm nào bị phân loại sai, dừng thuật toán.
- Bước 3: Nếu có điểm dữ liệu x_i bị phân lớp sai (tức là $y_i \neq f(x_i)$), cập nhật trọng số: $w_{i+1} = w_i + y_i x_i$
- Bước 4: Tiếp tục cập nhật trọng số cho đến khi không có điểm dữ liệu nào bị phân loại sai.

3.2.3. Mô hình Decision Tree (ID3):

Input: Tập dữ liệu huấn luyện đã được gán nhãn ($X_{\text{train}}, y_{\text{train}}$).

Output: Cây quyết định phân lớp, trong đó mỗi nút là một thuộc tính của dữ liệu và mỗi nhánh là một giá trị của thuộc tính đó. Các lá của cây quyết định sẽ là các nhãn lớp.

Hàm mất mát:

- ID3 không sử dụng hàm mất mát cụ thể như các thuật toán hồi quy hay phân lớp thông thường. Thay vào đó, ID3 sử dụng **entropy** và **information gain** để xây dựng cây quyết định.

Cách thực hiện:

- **Bước 1:** Tính **Entropy** của toàn bộ tập dữ liệu.
 - **Entropy** đo lường mức độ không chắc chắn hoặc không đồng nhất của tập dữ liệu.

$$Entropy(S) = - \sum_{i=1}^n p_i \log_2(p_i)$$

Trong đó, p_i là xác suất của lớp i trong tập S

- **Bước 2:** Đối với mỗi thuộc tính, tính **Information Gain (IG)** để xác định thuộc tính nào sẽ được chọn để phân chia dữ liệu tại mỗi bước.

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$$

Trong đó:

- A là thuộc tính được xét.
- $Values(A)$ là các giá trị của thuộc tính A .
- S_v là tập con của S chứa các mẫu có giá trị v cho thuộc tính A .
- **Bước 3:** Chọn thuộc tính có **Information Gain** lớn nhất để phân chia dữ liệu thành các nhánh.
- **Bước 4:** Lặp lại quá trình này cho các nhánh con cho đến khi:
 - Tất cả các điểm dữ liệu trong nhánh thuộc về một lớp duy nhất (tạo lá).
 - Không còn thuộc tính nào để phân chia hoặc dữ liệu không thể phân chia thêm được.

- **Bước 5:** Cây quyết định hoàn thành.

Các độ đo để đánh giá chất lượng mô hình dự đoán:

- **Accuracy** là chỉ số đo lường tỷ lệ phần trăm các mẫu dữ liệu được phân loại chính xác. Nó được tính bằng công thức sau:

$$Accuracy = \frac{\text{số lượng dự đoán chính xác}}{\text{tổng số mẫu dữ liệu}}$$

- **Precision** là chỉ số đo lường tỷ lệ phần trăm các mẫu dữ liệu được phân loại là dương tính thực sự là dương tính. Nó được tính bằng công thức sau:

$$Precision = \frac{\text{số lượng dự đoán dương tính thực sự}}{\text{tổng số dự đoán dương tính}}$$

- **Recall** là chỉ số đo lường tỷ lệ phần trăm các mẫu dữ liệu dương tính thực sự được phân loại là dương tính. Nó được tính bằng công thức sau:

$$Recall = \frac{\text{số lượng dự đoán dương tính thực sự}}{\text{tổng số mẫu dữ liệu dương tính thực sự}}$$

- **F1-score** là chỉ số kết hợp độ chính xác và độ thu hồi. Nó được tính bằng công thức sau:

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

CHƯƠNG II: THỰC NGHIỆM

1. Mô tả dữ liệu dự án

1.1 Mục đích của bài toán:

Xây dựng một mô hình học máy nhằm dự đoán khả năng mắc bệnh tim dựa trên các thông số y khoa của bệnh nhân. Mô hình này giúp hỗ trợ bác sĩ trong việc chẩn đoán nhanh hơn, chính xác hơn, từ đó đưa ra các quyết định điều trị phù hợp.

1.2 Input và Output:

- **Input:** Hai file dữ liệu train_dataset.csv và test_dataset.csv chứa thông tin y khoa của bệnh nhân.
- **Output:** Các độ đo đánh giá hiệu suất của từng thuật toán (Logistic Regression, PLA, Decision Tree) và biểu đồ so sánh độ chính xác của các mô hình.

1.3 Tóm tắt công việc thực hiện của dự án:

- Sử dụng dữ liệu y khoa của bệnh nhân với các thuộc tính như sau:

- **Age:** Tuổi của bệnh nhân.
- **Sex:** Giới tính của bệnh nhân (0 = Nữ, 1 = Nam).
- **Fbs:** Lượng đường trong máu lúc đói (> 120 mg/dL: 1 = Đúng, 0 = Sai).
- **Ca:** Loại đau ngực (0 = Đau thắt ngực điển hình, 1 = Không điển hình, 2 = Không đau thắt ngực, 3 = Không triệu chứng).

1.4 Công việc chính:

- Tiền xử lý dữ liệu, chuẩn hóa bằng StandardScaler.
- Áp dụng các thuật toán: Logistic Regression (không dùng thư viện và dùng thư viện), PLA (Perceptron), và ID3 (Decision Tree).
- Đánh giá hiệu suất của từng mô hình thông qua các độ đo: Accuracy, Confusion Matrix, Classification Report.
- So sánh hiệu suất các mô hình bằng biểu đồ trực quan.
- Xây dựng giao diện người dùng (GUI) để nhập thông tin và dự đoán bệnh tim.

1.5 Mô tả tập dữ liệu của bài toán:

 **Tập dữ liệu gồm:**

- Các thuộc tính:

- **Age:** Tuổi.
- **Sex:** Giới tính.
- **Fbs:** Lượng đường trong máu lúc đói.
- **Ca:** Loại đau thắt ngực.
- **AHD:** Nhãn lớp (1 = Có bệnh tim, 0 = Không có bệnh tim).

- Kích thước tập dữ liệu:

- Số lượng mẫu trong tập huấn luyện và kiểm tra chưa được cung cấp cụ thể.

- Chia dữ liệu:

- **70%** dữ liệu được sử dụng để huấn luyện mô hình.
- **30%** dữ liệu được sử dụng để kiểm tra hiệu suất mô hình.

1.6 Thuật toán sử dụng trong bài toán:

- **Logistic Regression (Hồi quy Logistic).**
- **PLA (Perceptron).**
- **Decision Tree (ID3).**

Mỗi thuật toán được đánh giá bằng độ chính xác (Accuracy) và ma trận nhầm lẫn (Confusion Matrix). Các thuật toán được so sánh để tìm ra mô hình hiệu quả nhất.

1.7 Kết quả và ứng dụng thực tiễn:

- Mô hình có khả năng dự đoán chính xác khả năng mắc bệnh tim của bệnh nhân.
- Hỗ trợ các bác sĩ trong việc đưa ra quyết định điều trị nhanh chóng và hiệu quả.

2. Cách thức triển khai

2.1 Import thư viện

- Các thư viện cần thiết phục vụ cho việc xử lý dữ liệu, đào tạo mô hình và trực quan hóa kết quả.
- Import các thư viện cần thiết để xử lý dữ liệu (Pandas, NumPy, StandardScaler), đánh giá mô hình học máy (accuracy, confusion matrix), trực quan hóa kết quả (Matplotlib, Seaborn), và tạo giao diện người dùng (CustomTkinter).

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import customtkinter as cus
```

2.2. Tải và hiển thị dữ liệu từ các tệp CSV

+ Đọc dữ liệu từ các tệp CSV:

- **train_file_path** và **test_file_path** lưu đường dẫn tới các tệp CSV chứa dữ liệu huấn luyện và kiểm tra.
- **pd.read_csv()** là hàm trong Pandas để đọc các tệp CSV và chuyển dữ liệu thành DataFrame.

+ Hiển thị 5 dòng đầu tiên của dữ liệu huấn luyện:

- **train_data.head()** sẽ hiển thị 5 dòng đầu tiên của DataFrame **train_data**, giúp người dùng xem trước dữ liệu.

```
# Đọc dữ liệu từ các tệp CSV
train_file_path = 'train_dataset.csv'
test_file_path = 'test_dataset.csv'

train_data = pd.read_csv(train_file_path)
test_data = pd.read_csv(test_file_path)

train_data.head()
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	54.0	1.0	0	124.0	266.0	0.0	2.0	109.0	1.0	2.2	2.0	1.0	2	1
1	46.0	1.0	2	101.0	197.0	1.0	0.0	156.0	0.0	0.0	1.0	0.0	2	0
2	43.0	1.0	1	130.0	315.0	0.0	0.0	162.0	0.0	1.9	1.0	1.0	1	0
3	47.0	1.0	1	138.0	257.0	0.0	2.0	156.0	0.0	0.0	1.0	0.0	1	0
4	58.0	1.0	0	150.0	270.0	0.0	2.0	111.0	1.0	0.8	1.0	0.0	2	1

+ Kiểm tra và đếm số giá trị thiếu (NaN) trong từng cột của dữ liệu huấn luyện:

```
# Kiểm tra xem có giá trị NaN (giá trị thiếu)
train_data.isna().sum()
```



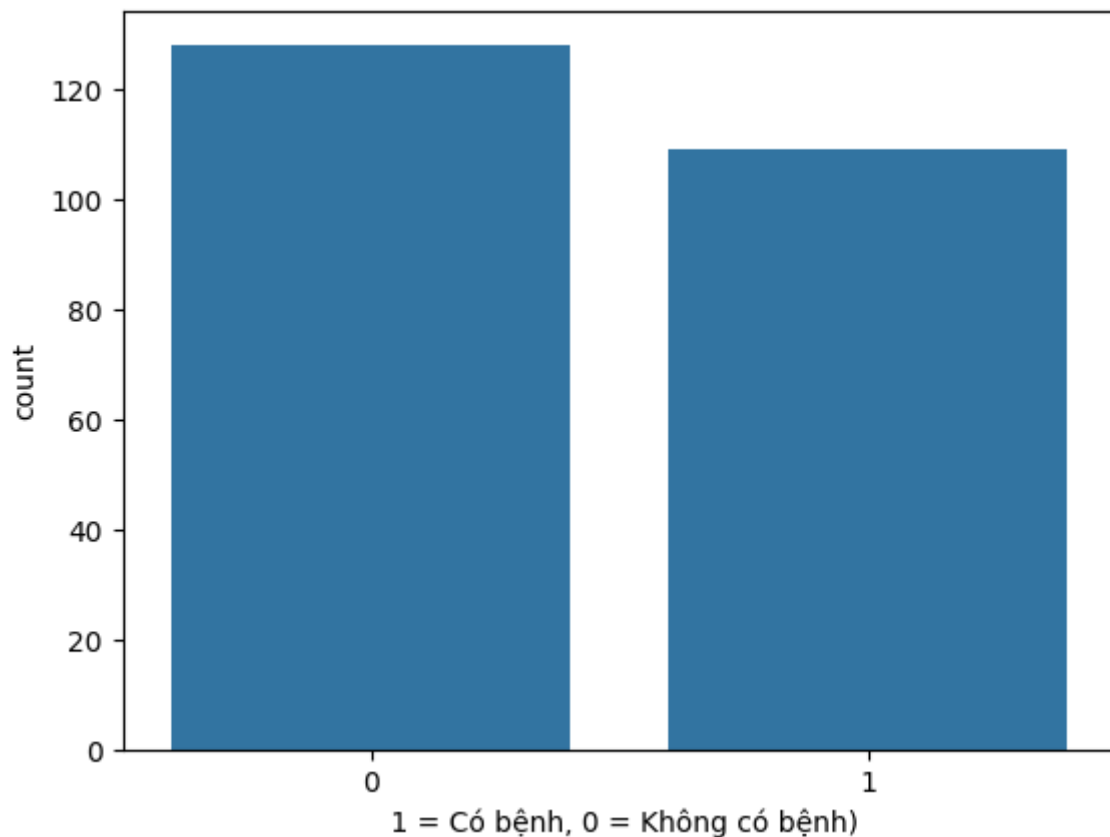
```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           0
Thal         0
AHD          0
dtype: int64
```

✚ Vẽ biểu đồ cột (countplot) để đếm số lượng các giá trị trong cột "AHD" của dữ liệu huấn luyện:

- `sns.countplot(x="AHD", data=train_data)`: Vẽ biểu đồ đếm số lượng các giá trị trong cột "AHD" (1 = Có bệnh, 0 = Không có bệnh).
- `plt.xlabel()`: Thêm nhãn cho trục x, mô tả ý nghĩa của giá trị 1 và 0.
- `plt.show()`: Hiện thị biểu đồ.

⇒ Giúp trực quan hóa phân bố các giá trị trong cột "AHD".

```
sns.countplot(x="AHD", data=train_data)
plt.xlabel(" 1 = Có bệnh, 0 = Không có bệnh")
plt.show()
```



✚ **Tính toán và in ra phần trăm bệnh nhân có và không có bệnh tim trong dữ liệu huấn luyện:**

- **Đếm số bệnh nhân:**
 - Không có bệnh tim (AHD = 0)
 - Có bệnh tim (AHD = 1)
- **Tính phần trăm:**
 - Tính tỷ lệ phần trăm cho mỗi nhóm bệnh nhân.

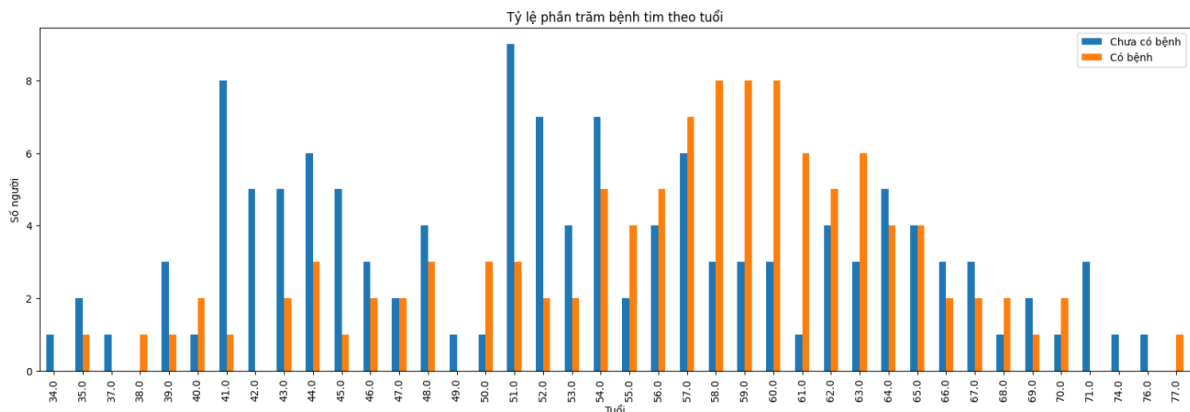
```
countNoDisease = len(train_data[train_data.AHD == 0])
countHaveDisease = len(train_data[train_data.AHD == 1])
print("Phần trăm bệnh nhân không có bệnh tim: {:.2f}%".format((countNoDisease / (len(train_data.AHD))*100))
print("Phần trăm bệnh nhân có bệnh tim: {:.2f}%".format((countHaveDisease / (len(train_data.AHD))*100))
```

Phần trăm bệnh nhân không có bệnh tim: 54.01%
Phần trăm bệnh nhân có bệnh tim: 45.99%

🔗 Vẽ biểu đồ cột để phân tích tỷ lệ bệnh tim theo độ tuổi:

- Tạo bảng tần suất giữa cột "Age" (tuổi) và "AHD" (bệnh tim) bằng `pd.crosstab()`.
 - Vẽ biểu đồ cột với `plot(kind="bar")` để trực quan hóa số người ở từng độ tuổi có hoặc không có bệnh tim.
 - Đặt tiêu đề và nhãn cho trục x (tuổi) và trục y (số người).
 - Hiển thị biểu đồ với `plt.show()`.
- ⇒ Giúp phân tích sự phân bố bệnh tim theo từng nhóm tuổi.

```
pd.crosstab(train_data.Age, train_data.AHD).plot(kind="bar", figsize=(20,6))
plt.title('Tỷ lệ phần trăm bệnh tim theo tuổi')
plt.xlabel('Tuổi')
plt.legend(["Chưa có bệnh", "Có bệnh"])
plt.ylabel('Số người')
plt.show()
```

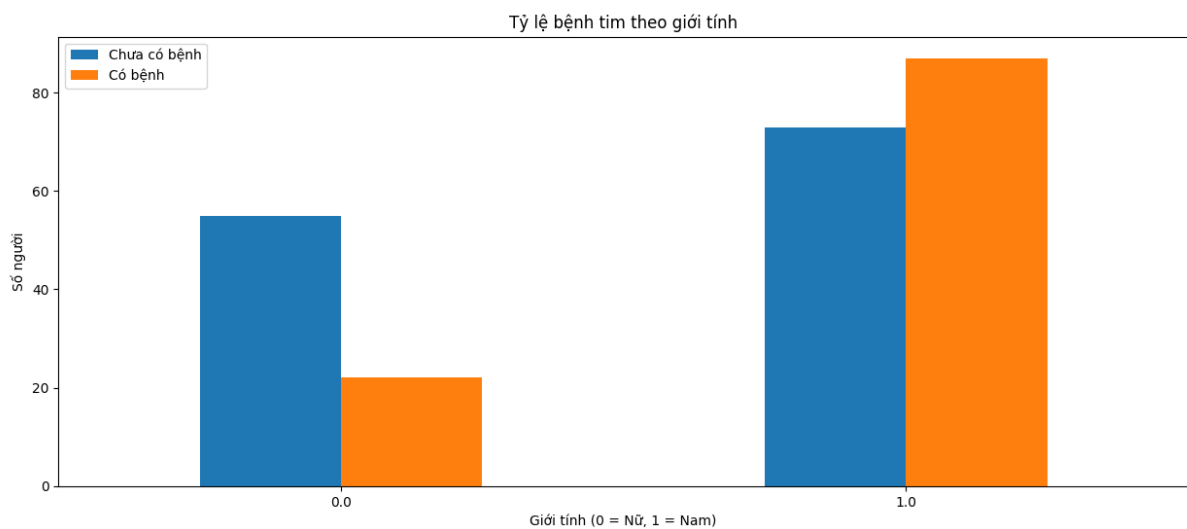


🔗 Phân tích tỷ lệ bệnh tim theo giới tính và vẽ biểu đồ cột:

- Tạo bảng tần suất giữa "Sex" (giới tính) và "AHD" (bệnh tim) bằng `pd.crosstab()` để hiển thị số lượng bệnh nhân có và không có bệnh tim theo giới tính.
 - Vẽ biểu đồ cột với `plot(kind="bar")` để thể hiện số người theo từng giới tính (Nữ = 0, Nam = 1), với kích thước biểu đồ là (15,6).
 - Đặt tiêu đề: "Tỷ lệ bệnh tim theo giới tính".
 - Đặt nhãn trục x cho giới tính (0 = Nữ, 1 = Nam) và nhãn trục y cho số người.
 - Điều chỉnh góc các nhãn trục x bằng `plt.xticks(rotation=0)` để nhãn hiển thị dễ đọc.
 - Thêm chú thích với `plt.legend()`, phân biệt giữa bệnh nhân có và không có bệnh tim.
 - Hiển thị biểu đồ với `plt.show()`.
- ⇒ Giúp đánh giá tỷ lệ bệnh tim theo giới tính trong dữ liệu.

```
print(pd.crosstab(train_data.Sex,train_data.AHD))
pd.crosstab(train_data.Sex,train_data.AHD).plot(kind="bar",figsize=(15,6))
plt.title('Tỷ lệ bệnh tim theo giới tính')
plt.xlabel('Giới tính (0 = Nữ, 1 = Nam)')
plt.xticks(rotation=0)
plt.legend(["Chưa có bệnh", "Có bệnh"])
plt.ylabel('Số người')
plt.show()
```

```
AHD    0    1
Sex
0.0    55   22
1.0    73   87
```



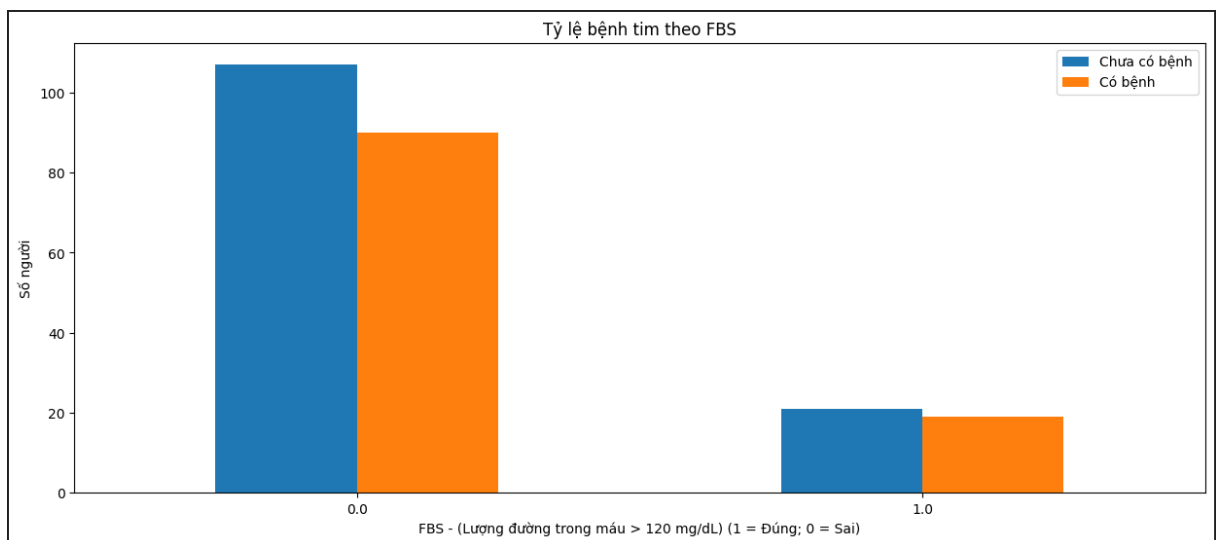
Phân tích tỷ lệ bệnh tim dựa trên mức độ đường huyết (FBS) và vẽ biểu đồ cột:

- **Tạo bảng tần suất** giữa cột "Fbs" (mức đường huyết > 120 mg/dL) và "AHD" (bệnh tim) bằng `pd.crosstab()` để hiển thị số lượng bệnh nhân có và không có bệnh tim tương ứng với từng giá trị FBS (1 = Đúng, 0 = Sai).
- **Vẽ biểu đồ cột** với `plot(kind="bar")` để thể hiện số bệnh nhân có và không có bệnh tim theo mức FBS, kích thước biểu đồ là (15,6).
- **Đặt tiêu đề:** "Tỷ lệ bệnh tim theo FBS".
- **Đặt nhãn trục x** cho FBS (1 = Đúng, 0 = Sai) và **nhãn trục y** cho số người.
- **Điều chỉnh góc các nhãn trục x** bằng `plt.xticks(rotation = 0)` để nhãn hiển thị dễ đọc.
- **Thêm chú thích** với `plt.legend()` để phân biệt giữa bệnh nhân có và không có bệnh tim.
- **Hiển thị biểu đồ** với `plt.show()`.

⇒ Giúp phân tích sự liên quan giữa mức đường huyết và bệnh tim.

```
print(pd.crosstab(train_data.Fbs,train_data.AHD))
pd.crosstab(train_data.Fbs,train_data.AHD).plot(kind="bar",figsize=(15,6))
plt.title('Tỷ lệ bệnh tim theo FBS')
plt.xlabel('FBS - (Lượng đường trong máu > 120 mg/dL) (1 = Đúng; 0 = Sai)')
plt.xticks(rotation = 0)
plt.legend(["Chưa có bệnh", "Có bệnh"])
plt.ylabel('Số người')
plt.show()
```

AHD	0	1
0.0	107	90
1.0	21	19



🔗 Phân tích tỷ lệ bệnh tim theo loại đau ngực (Ca) và vẽ biểu đồ cột:

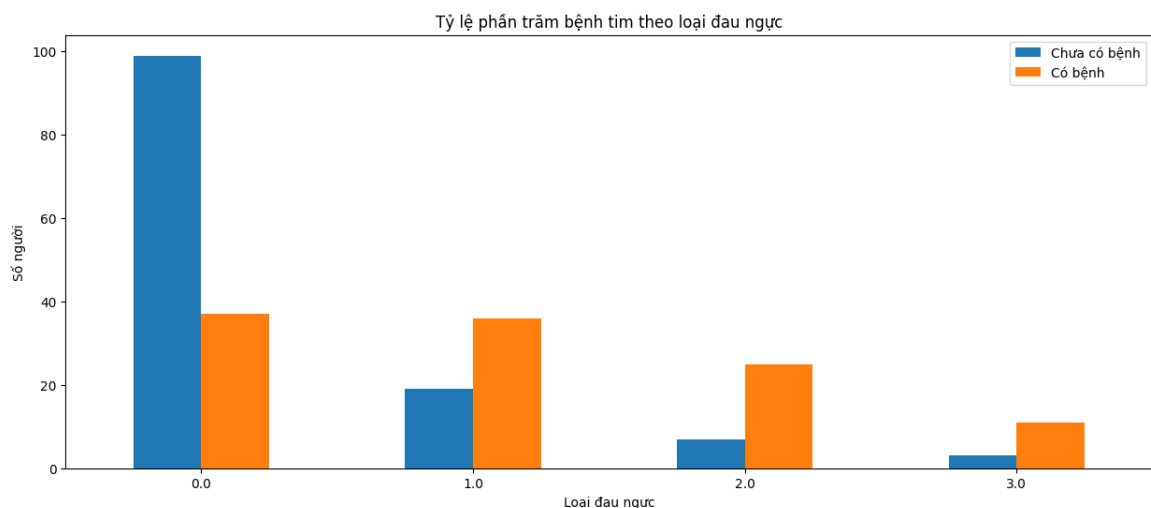
- Tạo bảng tần suất giữa cột "Ca" (loại đau ngực) và "AHD" (bệnh tim) bằng `pd.crosstab()` để hiển thị số lượng bệnh nhân có và không có bệnh tim theo từng loại đau ngực.
- Vẽ biểu đồ cột với `plot(kind="bar")` để thể hiện số bệnh nhân có và không có bệnh tim theo các loại đau ngực, với kích thước biểu đồ là (15,6).
- Đặt tiêu đề: "Tỷ lệ phần trăm bệnh tim theo loại đau ngực".
- Đặt nhãn trục x cho các loại đau ngực:
 - 0 = Đau thắt ngực điển hình.
 - 1 = Đau thắt ngực không điển hình.

- 2 = Không đau thắt ngực.
- 3 = Không có triệu chứng.
- Điều chỉnh góc các nhãn trục x bằng plt.xticks(rotation=0) để nhãn hiển thị dễ đọc.
- Đặt nhãn trục y cho số người.
- Hiển thị biểu đồ với plt.show().
- ⇒ Giúp phân tích mối liên hệ giữa các loại đau ngực và sự hiện diện của bệnh tim trong dữ liệu.

```
print(pd.crosstab(train_data.Ca,train_data.AHD))
pd.crosstab(train_data.Ca,train_data.AHD).plot(kind="bar",figsize=(15,6))
plt.title('Tỷ lệ phần trăm bệnh tim theo loại đau ngực')
plt.xlabel('Loại đau ngực')
print(" 0 = Đau thắt ngực điển hình\n", "1 = Đau thắt ngực không điển hình\n", "2 = Không đau thắt ngực\n", "3 = Không có triệu chứng\n")
plt.xticks(rotation = 0)
plt.legend(["Chưa có bệnh", "Có bệnh"])
plt.ylabel('Số người')
plt.show()
```

AHD	0	1
Ca		
0.0	99	37
1.0	19	36
2.0	7	25
3.0	3	11

0 = Đau thắt ngực điển hình
 1 = Đau thắt ngực không điển hình
 2 = Không đau thắt ngực
 3 = Không có triệu chứng



🔧 Chuẩn bị dữ liệu cho việc huấn luyện và kiểm tra mô hình:

- **X_train**: Chứa các đặc trưng (features) từ tập huấn luyện, loại bỏ cột "AHD" (để không sử dụng nó làm đặc trưng trong mô hình).

- **y_train**: Chứa nhãn (labels) từ cột "AHD" trong tập huấn luyện, là mục tiêu mà mô hình cần dự đoán.
 - **X_test**: Tương tự như X_train, nhưng đây là các đặc trưng từ tập kiểm tra.
 - **y_test**: Nhãn (labels) từ cột "AHD" trong tập kiểm tra, dùng để đánh giá hiệu suất mô hình.
- ⇒ Kết quả là dữ liệu được tách thành các đặc trưng và nhãn cho cả tập huấn luyện và tập kiểm tra, sẵn sàng để đưa vào mô hình học máy.

```
X_train = train_data.drop('AHD', axis=1)
y_train = train_data['AHD']
X_test = test_data.drop('AHD', axis=1)
y_test = test_data['AHD']
```

Chuẩn hóa (scaling) dữ liệu để đưa tất cả các đặc trưng vào cùng một phạm vi:

- **Tạo đối tượng chuẩn hóa:**

`scaler = StandardScaler()` tạo đối tượng chuẩn hóa từ `sklearn` để chuẩn hóa dữ liệu.

- **Chuẩn hóa tập huấn luyện:**

`X_train_scaled = scaler.fit_transform(X_train)` tính toán các thông số chuẩn hóa từ dữ liệu huấn luyện và áp dụng chuẩn hóa cho tập huấn luyện.

- **Chuẩn hóa tập kiểm tra:**

`X_test_scaled = scaler.transform(X_test)` chuẩn hóa tập kiểm tra sử dụng các thông số đã tính toán từ tập huấn luyện.


⇒ Giúp mô hình học máy hoạt động hiệu quả hơn.

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2.3. Huấn luyện Logistic Regression (LR)

2.3.1 Không dùng thư viện:

-  Xây dựng mô hình Logistic Regression từ đầu và huấn luyện nó bằng Gradient Descent, sau đó đánh giá mô hình:

1. **Hàm Sigmoid:**
sigmoid(z) tính toán hàm kích hoạt Sigmoid, dùng để chuyển giá trị đầu vào thành xác suất (trong khoảng 0-1).
 2. **Hàm huấn luyện Logistic Regression:**
logistic_regression(X, y, ...) huấn luyện mô hình Logistic Regression bằng Gradient Descent:
 - Tính toán dự đoán (\hat{y}) từ đặc trưng (X).
 - Tính toán hàm mất mát (log loss).
 - Cập nhật trọng số (θ) và bias (b) theo Gradient Descent.
 - Điều kiện dừng sớm khi gradient gần 0 hoặc hàm mất mát không thay đổi đáng kể.
 3. **Hàm dự đoán:**
predict(X, best_theta, best_bias) tính toán dự đoán bằng mô hình đã huấn luyện và trả về kết quả (1 hoặc 0) dựa trên xác suất dự đoán.
 4. **Đánh giá mô hình:**
Các hàm **accuracy**, **confusion_matrix**, và **classification_report** được sử dụng để đánh giá độ chính xác và các chỉ số đánh giá khác như **precision**, **recall** và **F1-score**.
 - **accuracy**: Tính toán độ chính xác của mô hình.
 - **confusion_matrix**: Xây dựng ma trận nhầm lẫn.
 - **classification_report**: Tính toán **precision**, **recall**, **F1-score** dựa trên ma trận nhầm lẫn.
- ⇒ Mô hình được huấn luyện và sau đó dự đoán trên tập kiểm tra để đánh giá hiệu quả.


```

# Hàm kích hoạt Sigmoid
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Hàm huấn luyện Logistic Regression bằng Gradient Descent
def logistic_regression(X, y, learning_rate=0.001, epochs=2000, tolerance=1e-6):
    m, n = X.shape
    theta = np.zeros(n)
    bias = 0

    loss_list = []
    min_loss = float('inf')
    best_theta = theta
    best_bias = bias

    for epoch in range(epochs):
        z = np.dot(X, theta) + bias
        predictions = sigmoid(z)

        loss = -np.mean(y * np.log(predictions) + (1 - y) * np.log(1 - predictions))
        loss_list.append(loss)

        if loss < min_loss:
            min_loss = loss
            best_theta = theta
            best_bias = bias

        dw = (1/m) * np.dot(X.T, (predictions - y))
        db = (1/m) * np.sum(predictions - y)

        # Early stopping condition: gradient gần 0 hoặc loss không thay đổi
        if np.linalg.norm(dw) < tolerance and abs(db) < tolerance:
            print(f"Hội tụ tại epoch {epoch}")
            break

        theta -= learning_rate * dw
        bias -= learning_rate * db

    return best_theta, best_bias, loss_list

# Hàm dự đoán
def predict(X, best_theta, best_bias):
    z = np.dot(X, best_theta) + best_bias
    predictions = sigmoid(z)
    return [1 if i >= 0.5 else 0 for i in predictions]

# Huấn luyện mô hình Logistic Regression
best_theta, best_bias, loss_list = logistic_regression(X_train_scaled, y_train)

# Dự đoán trên tập kiểm tra
y_test_pred = predict(X_test_scaled, best_theta, best_bias)

```

```

# Đánh giá mô hình
def accuracy(y_true, y_pred):
    return np.mean(y_true == y_pred)

def confusion_matrix(y_true, y_pred):
    cm = np.zeros((2, 2), dtype=int)
    for i in range(len(y_true)):
        cm[int(y_true[i])][int(y_pred[i])] += 1
    return cm

def classification_report(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    tp = cm[1][1]
    tn = cm[0][0]
    fp = cm[0][1]
    fn = cm[1][0]

    accuracy = (tp + tn) / (tp + tn + fp + fn)
    precision = tp / (tp + fp) if (tp + fp) != 0 else 0
    recall = tp / (tp + fn) if (tp + fn) != 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) != 0 else 0

    report = {
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1': f1
    }

    return report

```

🔗 Đánh giá mô hình Logistic Regression đã huấn luyện:

- **Đo độ chính xác:**

`print("Độ chính xác của thuật toán hồi quy logistic:", accuracy(y_test, y_test_pred))` tính toán và in ra độ chính xác của mô hình trên tập kiểm tra. Độ chính xác là tỷ lệ dự đoán đúng so với tổng số dự đoán.

- **In ma trận nhầm lẫn:**

`print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_test_pred))` in ra ma trận nhầm lẫn, giúp xem xét các giá trị thực tế (True/False Positive, True/False Negative) của mô hình.

- **In báo cáo:**

`print("\nClassification Report:\n", classification_report(y_test, y_test_pred))` in ra bảng tóm tắt các chỉ số đánh giá hiệu suất của mô hình trên tập kiểm tra.

⇒ Giúp đánh giá và kiểm tra hiệu quả của mô hình trên tập kiểm tra.

```
print(f"Độ chính xác của thuật toán hồi quy logistic: {accuracy(y_test, y_test_pred) * 100:.2f}%")
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
print("\nClassification Report:\n", classification_report(y_test, y_test_pred))
```

Độ chính xác của thuật toán hồi quy logistic: 91.67%

Confusion Matrix:

```
[[30  2]
 [ 3 25]]
```

Classification Report:

```
{'accuracy': np.float64(0.9166666666666666), 'precision': np.float64(0.9259259259259259), 'recall': np.float64(0.8928571428571429), 'f1': np.float64(0.9090909090909091)}
```

2.3.2 Dùng thư viện:

✚ Sử dụng thư viện sklearn để huấn luyện mô hình Logistic Regression:

1. Tạo mô hình Logistic Regression:

`log_model = LogisticRegression(max_iter=6000, solver='lbfgs', C=1.0)`

tạo một đối tượng mô hình Logistic Regression:

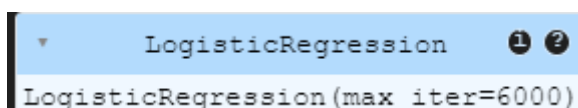
- **max_iter=6000**: Đặt số vòng lặp tối đa cho thuật toán tối ưu.
- **solver='lbfgs'**: Sử dụng thuật toán L-BFGS để tối ưu hóa, phù hợp với dữ liệu lớn.
- **C=1.0**: Đặt tham số điều chỉnh độ mạnh của regularization.

2. Huấn luyện mô hình:

`log_model.fit(X_train_scaled, y_train)` huấn luyện mô hình với dữ liệu huấn luyện đã chuẩn hóa (`X_train_scaled`) và nhãn (`y_train`).

```
from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression(max_iter=6000, solver='lbfgs', C=1.0)
log_model.fit(X_train_scaled, y_train)
```



The screenshot shows a Jupyter Notebook cell with the following code:

```
LogisticRegression
LogisticRegression(max_iter=6000)
```

2.4. Độ chính xác của mô hình thuật toán Logistic Regression (LR):

✚ Thực hiện dự đoán và đánh giá mô hình Logistic Regression đã huấn luyện:

- **Dự đoán trên tập kiểm tra:**
log_predictions = log_model.predict(X_test_scaled) sử dụng mô hình đã huấn luyện để dự đoán nhãn trên tập kiểm tra (X_test_scaled).
 - **Đánh giá độ chính xác:**
log_accuracy = accuracy_score(y_test, log_predictions) tính toán độ chính xác của mô hình bằng cách so sánh dự đoán (log_predictions) với nhãn thực tế (y_test).
 - **Ma trận nhầm lẫn:**
log_cm = confusion_matrix(y_test, log_predictions) tạo ma trận nhầm lẫn để phân tích kết quả dự đoán (True/False Positives/Negatives).
 - **Báo cáo phân loại:**
log_report = classification_report(y_test, log_predictions) tạo báo cáo phân loại chứa các chỉ số như precision, recall và F1-score.
 - **Đo độ chính xác với hàm tự tạo:**
log_acc = accuracy(y_test, y_test_pred) tính toán độ chính xác bằng cách sử dụng hàm accuracy tự định nghĩa, nhưng nó không cần thiết vì đã có log_accuracy sử dụng accuracy_score.
 - **In kết quả:**
 - print("Độ chính xác của thuật toán hồi quy logistic:", log_accuracy) in ra độ chính xác của mô hình.
 - print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_test_pred)) in ma trận nhầm lẫn.
 - print("\nClassification Report:\n", log_report) in ra bảng tóm tắt các chỉ số đánh giá hiệu suất của mô hình trên tập kiểm tra
- ⇒ Giúp đánh giá mô hình Logistic Regression về độ chính xác, ma trận nhầm lẫn và các chỉ số phân loại.

```
# Dự đoán trên tập kiểm tra
log_predictions = log_model.predict(X_test_scaled)
# Đánh giá mô hình
log_accuracy = accuracy_score(y_test, log_predictions)
log_cm = confusion_matrix(y_test, log_predictions)
log_report = classification_report(y_test, log_predictions)
log_acc = accuracy(y_test, y_test_pred)

print(f"Độ chính xác kiểm thử của thuật toán hồi quy logistic: {log_accuracy * 100:.2f}%")
print("Confusion Matrix:\n", log_cm)
print("\nClassification Report:\n", log_report)
```

Độ chính xác kiểm thử của thuật toán hồi quy logistic: 85.00%

Confusion Matrix:

```
[[27  5]
 [ 4 24]]
```

Classification Report:

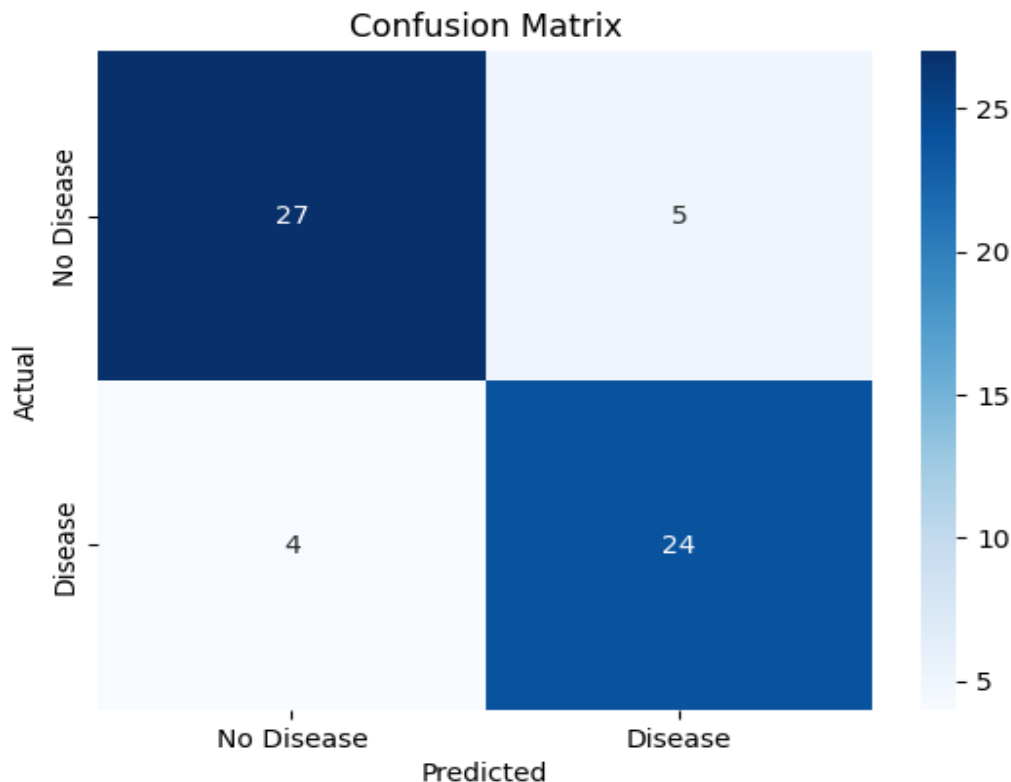
```
{'accuracy': np.float64(0.85), 'precision': np.float64(0.8275862068965517), 'recall': np.float64(0.8571428571428571), 'f1': np.float64(0.8421052631578947)}
```

Vẽ ma trận nhầm lẫn dưới dạng bản đồ nhiệt (heatmap) để trực quan hóa kết quả dự đoán của mô hình:

- **Vẽ heatmap:** `sns.heatmap()` vẽ ma trận nhầm lẫn với màu sắc thể hiện số lượng dự đoán đúng và sai.
- **Hiển thị giá trị:** `annot=True` cho phép hiển thị giá trị trong mỗi ô.
- **Những nhãn:** Nhãn `No Disease` và `Disease` được gán cho cả trục X (dự đoán) và trục Y (thực tế).
- **Hiển thị biểu đồ:** `plt.show()` hiển thị heatmap.

⇒ Giúp dễ dàng nhìn thấy hiệu quả của mô hình qua các dự đoán đúng/sai.

```
sns.heatmap(log_cm, annot=True, fmt='d', cmap='Blues',
             xticklabels=['No Disease', 'Disease'],
             yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



2.5. Độ chính xác của mô hình thuật toán Perceptron Learning Algorithm (PLA):

🔧 Huấn luyện và đánh giá mô hình Perceptron Learning Algorithm (PLA):

- **Khởi tạo và huấn luyện mô hình PLA:**
 - `pla = Perceptron(max_iter=1000, eta0=1.0, random_state=42)` tạo một mô hình Perceptron:
 - `max_iter=1000`: Số lần lặp tối đa.
 - `eta0=1.0`: Tốc độ học (learning rate).
 - `random_state=42`: Đảm bảo kết quả tái tạo được.
 - `pla.fit(X_train_scaled, y_train)` huấn luyện mô hình với dữ liệu huấn luyện.
- **Dự đoán trên tập kiểm tra:**

`pla_predictions = pla.predict(X_test_scaled)` sử dụng mô hình đã huấn luyện để dự đoán trên tập kiểm tra.
- **Đánh giá mô hình:**

`pla_accuracy = accuracy_score(y_test, pla_predictions)` tính độ chính xác của mô hình.

pla_cm = confusion_matrix(y_test, pla_predictions) tạo ma trận nhầm lẫn.

- **In kết quả:**

- print("Độ chính xác kiểm thử của PLA:", pla_accuracy) in độ chính xác của mô hình.
- print("\nConfusion Matrix:\n", pla_cm) in ma trận nhầm lẫn.
- print("\nClassification Report:\n", pla_report) in ra bảng tóm tắt các chỉ số đánh giá hiệu suất của mô hình trên tập kiểm tra

```
from sklearn.linear_model import Perceptron
pla = Perceptron(max_iter=1000, eta0=1.0, random_state=42)
pla.fit(X_train_scaled, y_train)

pla_predictions = pla.predict(X_test_scaled)

pla_accuracy = accuracy_score(y_test, pla_predictions)
pla_cm = confusion_matrix(y_test, pla_predictions)
pla_report = classification_report(y_test, pla_predictions)

print(f"Độ chính xác kiểm thử của thuật toán PLA: {pla_accuracy * 100:.2f}%")
print("Confusion Matrix:\n", pla_cm)
print("\nClassification Report:\n", pla_report)
```

Độ chính xác kiểm thử của thuật toán PLA: 83.33%

Confusion Matrix:

```
[[26  0]
 [ 4 24]]
```

Classification Report:

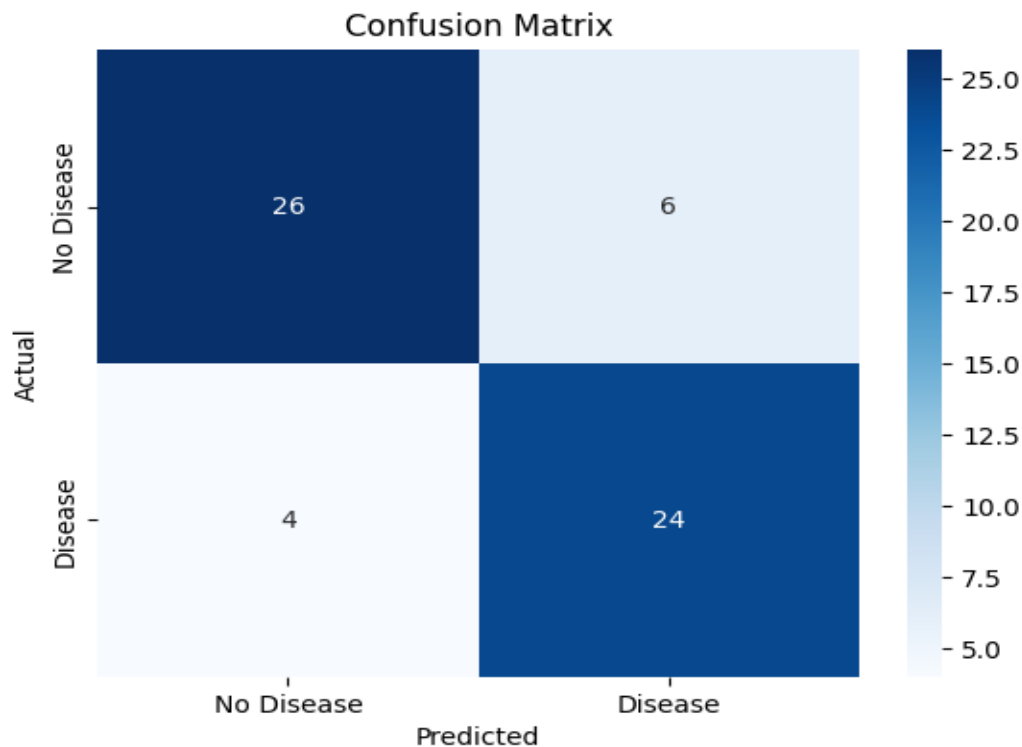
```
{'accuracy': np.float64(0.8333333333333334), 'precision': np.float64(0.8), 'recall': np.float64(0.8571428571428571), 'f1': np.float64(0.8275862068965518)}
```

✚ Vẽ ma trận nhầm lẫn của mô hình PLA dưới dạng bản đồ nhiệt (heatmap):

- **Hiển thị giá trị trong ô:** annot=True hiển thị số liệu trong các ô của ma trận.
- **Màu sắc:** cmap='Blues' dùng màu xanh để phân biệt các giá trị.
- **Nhãn cho các trục:** xticklabels và yticklabels xác định nhãn cho các trục X (dự đoán) và Y (thực tế).
- **Hiển thị biểu đồ:** plt.show() sẽ hiển thị bản đồ nhiệt.

⇒ Giúp trực quan hóa độ chính xác của mô hình.

```
sns.heatmap(pla_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Disease', 'Disease'],
            yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



2.6. Độ chính xác của mô hình thuật toán Decision Tree (ID3):

✚ Huấn luyện và đánh giá mô hình Cây quyết định (Decision Tree) với tiêu chí "entropy":

- Khởi tạo và huấn luyện mô hình PLA:

- `id3 = DecisionTreeClassifier(criterion="entropy", random_state=42)` tạo mô hình cây quyết định, sử dụng tiêu chí "entropy" để chia dữ liệu.
- `id3.fit(X_train_scaled, y_train)` huấn luyện mô hình trên dữ liệu huấn luyện đã chuẩn hóa (`X_train_scaled` và `y_train`).

- Dự đoán trên tập kiểm tra:

`id3_predictions = id3.predict(X_test_scaled)` sử dụng mô hình để dự đoán nhãn trên tập kiểm tra.

- Đánh giá mô hình:

- `id3_accuracy = accuracy_score(y_test, id3_predictions)` tính độ chính xác của mô hình.
- `id3_cm = confusion_matrix(y_test, id3_predictions)` tạo ma trận nhầm lẫn.

- `id3_report = classification_report(y_test, id3_predictions)` tạo báo cáo phân loại (chưa in ra trong mã này).
- **Dự đoán và in kết quả:**
 - `y_pred = id3.predict(X_test)` dự đoán lại trên tập kiểm tra (bị trùng với phân trước).
 - `print("Độ chính xác kiểm thử của thuật toán ID3:", id3_accuracy)` in ra độ chính xác của mô hình.

```
from sklearn.tree import DecisionTreeClassifier
id3 = DecisionTreeClassifier(criterion="entropy", random_state=42)
id3.fit(X_train_scaled, y_train)

# Dự đoán trên tập kiểm tra
id3_predictions = id3.predict(X_test_scaled)
# Đánh giá mô hình
id3_accuracy = accuracy_score(y_test, id3_predictions)
id3_cm = confusion_matrix(y_test, id3_predictions)
id3_report = classification_report(y_test, id3_predictions)

y_pred = id3.predict(X_test)
print(f"Độ chính xác kiểm thử của thuật toán ID3: {id3_accuracy * 100:.2f}%")
print("\nClassification Report:\n", id3_report)
```

Độ chính xác kiểm thử của thuật toán ID3: 78.33%

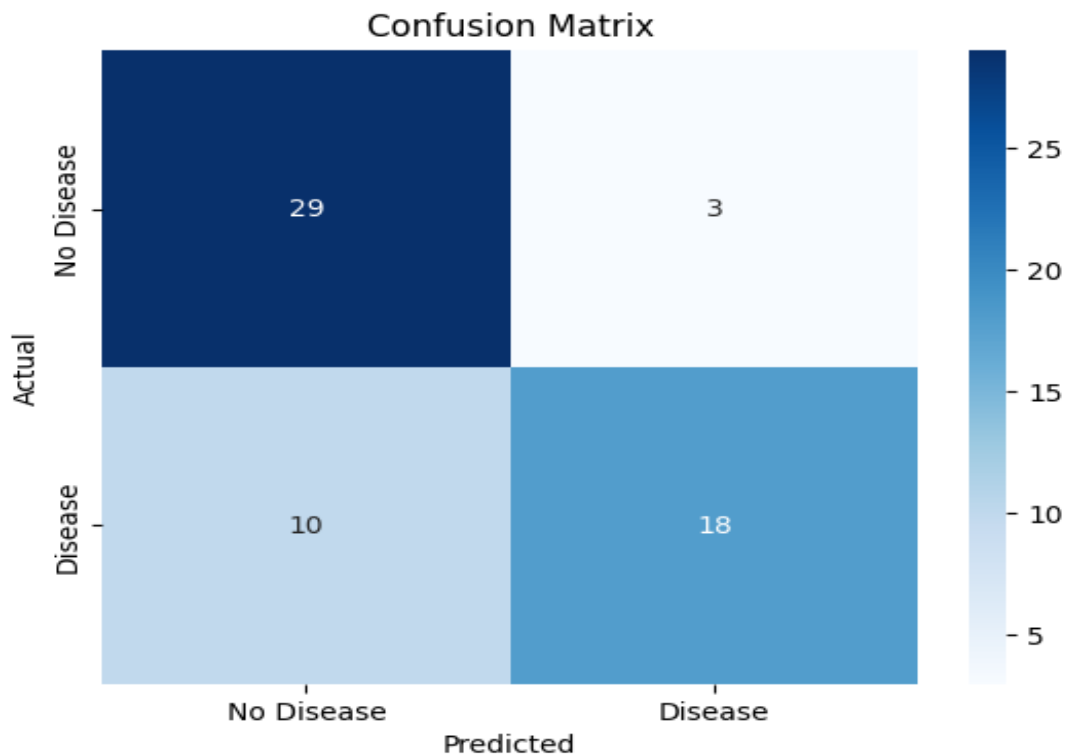
Classification Report:
{ 'accuracy': np.float64(0.7833333333333333), 'precision': np.float64(0.8571428571428571), 'recall': np.float64(0.6428571428571429), 'f1': np.float64(0.7346938775510204)}

Vẽ ma trận nhầm lẫn của mô hình ID3 dưới dạng bản đồ nhiệt (heatmap):

- **Hiển thị giá trị:** `annot=True` hiển thị số liệu trong các ô.
- **Màu sắc:** `cmap='Blues'` dùng màu xanh để phân biệt các giá trị.
- **Nhãn trục:** `xticklabels` và `yticklabels` đặt nhãn cho các trục dự đoán và thực tế.
- **Hiển thị biểu đồ:** `plt.show()` sẽ hiển thị ma trận nhầm lẫn.

⇒ Giúp đánh giá hiệu quả của mô hình.

```
sns.heatmap(id3_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Disease', 'Disease'],
            yticklabels=['No Disease', 'Disease'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



3. So sánh độ chính xác của ba mô hình:

🔧 Vẽ biểu đồ cột để so sánh độ chính xác của ba mô hình:

- **Dữ liệu đầu vào:**
 - **accuracies** = [log_acc, pla_accuracy, id3_accuracy]: Danh sách độ chính xác của các mô hình.
 - **models** = ['Logistic Regression', 'PLA', 'Decision Tree']: Danh sách tên các mô hình.
- **Vẽ biểu đồ cột:**
 - `plt.bar(models, accuracies, color=['blue', 'green', 'red'])`: Vẽ biểu đồ cột với mỗi mô hình có màu khác nhau.
- **Thêm tiêu đề và nhãn trục:**
 - `plt.title('So sánh độ chính xác của các thuật toán')`: Tiêu đề cho biểu đồ.
 - `plt.xlabel('Thuật toán')`: Nhãn cho trục X.
 - `plt.ylabel('Độ chính xác')`: Nhãn cho trục Y.
- **Hiển thị giá trị trên cột:**

- `plt.text(i, accuracies[i] + 0.01, f'{accuracies[i]:.2f}', ...)`: Hiển thị giá trị độ chính xác trên đỉnh mỗi cột.
- **Giới hạn trục Y:**
 - `plt.ylim(0, 1)`: Giới hạn trục Y từ 0 đến 1.
- **Hiển thị biểu đồ:**
 - `plt.show()` hiển thị biểu đồ.

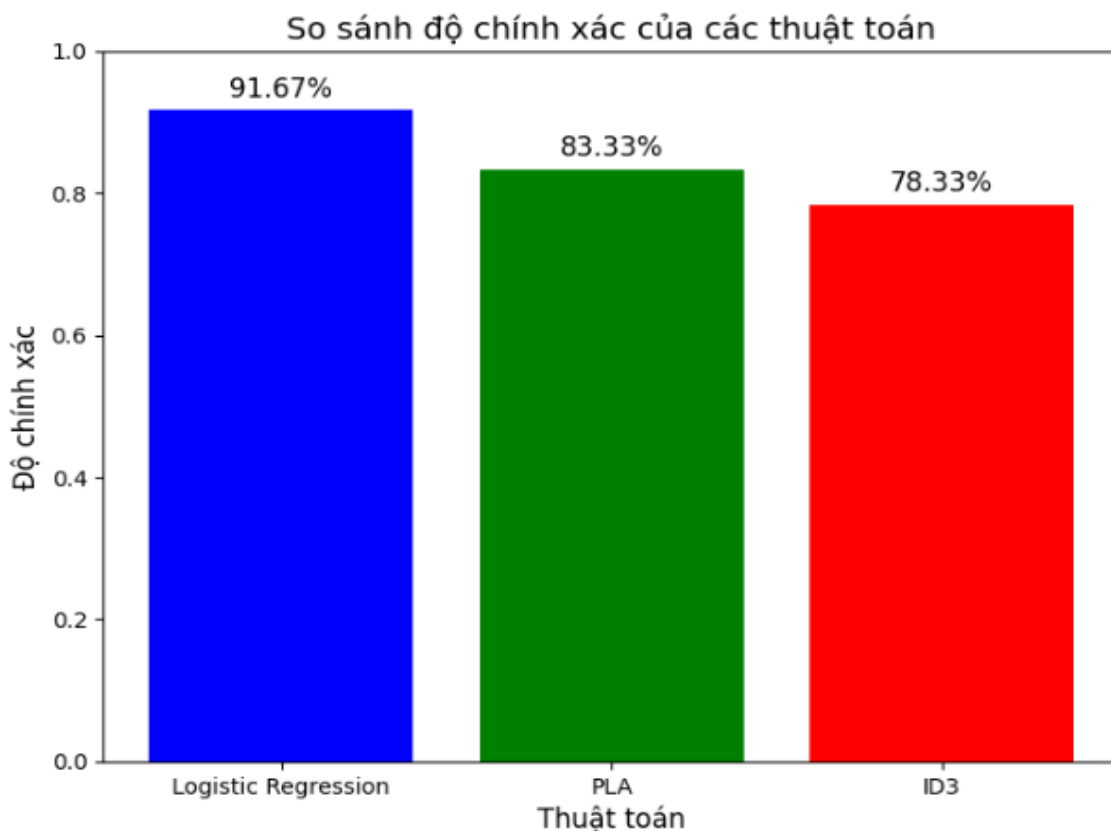
⇒ Giúp so sánh hiệu quả giữa các thuật toán dựa trên độ chính xác của chúng.

```
accuracies = [log_acc, pla_accuracy, id3_accuracy]
models = ['Logistic Regression', 'PLA', 'ID3']

plt.figure(figsize=(8, 6))
plt.bar(models, accuracies, color=['blue', 'green', 'red'])
plt.title('So sánh độ chính xác của các thuật toán', fontsize=14)
plt.xlabel('Thuật toán', fontsize=12)
plt.ylabel('Độ chính xác', fontsize=12)

for i in range(len(accuracies)):
    plt.text(i, accuracies[i] + 0.01, f'{accuracies[i]*100:.2f}%', ha='center', va='bottom', fontsize=12)

plt.ylim(0, 1)
plt.show()
```



⇒ Nhận xét: Logistic Regression tự triển khai cho thấy tính hiệu quả và khả năng áp dụng vào bài toán, vượt trội hơn PLA và ID3. PLA và ID3.

4. Tạo giao diện phần mềm:

✚ Tạo giao diện phần mềm dự đoán bệnh tim sử dụng Tkinter và thư viện customtkinter:

- **Tạo cửa sổ chính:**
 - **root = cus.CTk():** Khởi tạo cửa sổ chính với thư viện customtkinter.
 - **root.geometry(f'{{width}}x{{height}}+{{x}}+{{y}}'):** Đặt kích thước và vị trí cửa sổ ở giữa màn hình.
- **Tiêu đề ứng dụng:**
 - **app_title = cus.CTkLabel(...):** Tạo một nhãn hiển thị tiêu đề ứng dụng ở đầu cửa sổ.
- **Khởi tạo khung để chứa các ô nhập liệu:**
 - **frame = cus.CTkFrame(...):** Tạo một khung (frame) để chứa các ô nhập liệu cho người dùng.
 - **values = {}:** Dùng dictionary values để lưu trữ các ô nhập liệu tương ứng với các cột dữ liệu.
- **Hàm dự đoán:**
 - **predict():** Hàm này lấy giá trị người dùng nhập vào các ô, chuyển thành mảng và thực hiện dự đoán với mô hình logistic regression.
 - Kết quả dự đoán được hiển thị trong result_label.
- **Nút dự đoán:**
 - **predictBtn = cus.CTkButton(...):** Nút để người dùng bấm và nhận dự đoán từ mô hình.
- **Kết quả dự đoán:**
 - **result_label:** Hiển thị kết quả dự đoán (có hoặc không có bệnh tim).
- **Chạy ứng dụng:**
 - **root.mainloop():** Chạy vòng lặp chính để hiển thị và tương tác với giao diện.

Ứng dụng cho phép người dùng nhập các giá trị đặc trưng và nhận dự đoán về khả năng mắc bệnh tim dựa trên mô hình hồi quy logistic.

```

import tkinter as tk
from tkinter import filedialog, messagebox
# Giao diện người dùng
root = cus.CTk()
root.title("Phần mềm dự đoán bệnh tim")

width, height = 800, 600
x = (root.winfo_screenwidth() - width) // 2
y = (root.winfo_screenheight() - height) // 2
root.geometry(f"{width}x{height}+{x}+{y}")

# Tiêu đề ứng dụng
app_title = cus.CTkLabel(root, text="Phần mềm dự đoán bệnh tim", font=("Arial", 18, "bold"), text_color="blue")
app_title.pack(pady=10)

# Tính toán độ chính xác mô hình
accuracy_score_value = accuracy_score(y_test, y_test_pred)

# Đảm bảo accuracy_score_value là số thực
accuracy_score_value = float(accuracy_score_value)

# Hiển thị độ chính xác mô hình
info_label = cus.CTkLabel(
    root,
    text=f"Độ chính xác mô hình hồi quy logistic: {accuracy_score_value*100:.2f}%",
    font=("Arial", 14),
    justify="left",
)
info_label.pack(pady=20)

# Frame để chứa các ô nhập liệu
frame = cus.CTkFrame(root, border_width=2)
frame.pack(pady=20, padx=20)

# Lưu trữ các ô nhập liệu cho đặc trưng
values = {}
for i, col in enumerate(X_train.columns):
    label = cus.CTkLabel(frame, text=col, font=("Arial", 12))
    label.grid(row=i, column=0, padx=10, pady=5, sticky="w")
    entry = cus.CTkEntry(frame, width=200)
    entry.grid(row=i, column=1, padx=10, pady=5)
    values[col] = entry

# Hàm dự đoán
def predict():
    input_data = np.array([[float(values[col].get()) for col in X_train.columns]])
    prediction = log_model.predict(input_data)[0]
    result_text = "Có nguy cơ mắc bệnh" if prediction == 1 else "Không có nguy cơ mắc bệnh"
    result_label.configure(text=f"Kết quả dự đoán: {result_text}")

# Nút dự đoán và hiển thị kết quả
predictBtn = cus.CTkButton(frame, text="Dự đoán", command=predict)
predictBtn.grid(column=4, row=9, padx=10)

# Hiển thị kết quả dự đoán
result_label = cus.CTkLabel(frame, text="", font=("Time New Roman", 14, "bold"))
result_label.grid(column=4, row=10, pady=5, padx=5)

# Chạy ứng dụng
root.mainloop()

```

CHƯƠNG III: LỜI KẾT

Trải qua quá trình cùng nhau nghiên cứu và thực hiện dự án này của môn học, nhóm chúng em không chỉ học hỏi được những kiến thức về việc áp dụng các thuật toán học máy vào thực tế mà còn hiểu rõ hơn về cách xây dựng các mô hình dự đoán bệnh tim. Các thuật toán như Logistic Regression, Perceptron, và

Decision Tree đã giúp chúng em thực hiện dự đoán với độ chính xác cao và đánh giá hiệu suất của từng mô hình. Dự án không chỉ là một bài tập nghiên cứu mà còn là cơ hội để chúng em cải thiện kỹ năng phân tích dữ liệu, huấn luyện mô hình và làm việc nhóm. Đây là một trải nghiệm thực tế quý báu, tạo nền tảng vững chắc cho chúng em trong các dự án tương lai.

Chúng em xin chân thành cảm ơn thầy Tạ Quang Chiêu đã tận tình hướng dẫn và hỗ trợ chúng em trong suốt quá trình học trên giảng đường và những kiến thức thầy đã truyền đạt trong buổi vấn đáp trước đó đã giúp chúng em có thể hiểu sâu hơn, tự tin cùng nhau hoàn thành dự án lần này. Chúng em rất biết ơn sự giúp đỡ và sự quan tâm của thầy, đồng thời hy vọng sẽ tiếp tục nhận được sự hỗ trợ từ thầy trong các dự án tương lai.

Nhóm 6 xin chân thành cảm ơn thầy ạ  

Tài liệu tham khảo:

- ✚ Slide của giáo viên hướng dẫn Tạ Quang Chiêu.
- ✚ Sách Machine Learning cơ bản của Vũ Hữu Tiệp.
- ✚ Thuật toán Logistic Regression (LR): https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- ✚ Thuật toán Perceptron Learning Algorithm (PLA): https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html
- ✚ Thuật toán Decision Tree (ID3): <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>