BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG

PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI

BỘ MÔN CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI

Dự đoán nguy cơ mắc bệnh suy tim

Giảng viên:	Ths. Vũ Thị Hạnh
Sinh viên thực hiện:	2351267265 Nguyễn Đức Huy 2351267261Nguyễn Lê Minh Hậu
Lớp:	S26-65TTNT

TP. Hồ Chí Minh, ngày … tháng … năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TP. Hồ Chí Minh, ngày … tháng … năm 2025

Chữ ký của giảng viên

LỜI CẢM ƠN

Để hoàn thành báo cáo kết thúc môn học này, bên cạnh sự nỗ lực của các thành viên trong nhóm, là sự dẫn dắt tận tình và tâm huyết của Giảng viên hướng dẫn.

Lời đầu tiên và trân trọng nhất, chúng em xin được gửi lời tri ân sâu sắc đến Cô Vũ Thị Hạnh người đã trực tiếp giảng dạy và đồng hành cùng chúng em trong suốt thời gian qua.

Môn học này không chỉ mang lại cho chúng em những kiến thức nền tảng quan trọng về Công nghệ thông tin và Học máy, mà còn rèn luyện cho chúng em tư duy giải quyết vấn đề thực tế. Chúng em đặc biệt biết ơn Cô vì những bài giảng lôi cuốn, sự kiên nhẫn khi giải đáp từng thắc mắc và những định hướng chuyên môn quý báu giúp nhóm vượt qua những bế tắc trong quá trình xử lý dữ liệu và xây dựng mô hình dự đoán bệnh tim.

Sự khắt khe về mặt học thuật nhưng cũng đầy bao dung của cô chính là động lực lớn nhất để chúng em không ngừng hoàn thiện đồ án này một cách chỉn chu nhất có thể.

Nhóm cũng xin cảm ơn cộng đồng Kaggle đã cung cấp bộ dữ liệu, giúp chúng em có cơ sở thực tiễn để áp dụng những kiến thức cô đã dạy vào bài toán cụ thể.

Do thời gian thực hiện và kiến thức còn hạn chế, bài báo cáo khó tránh khỏi những thiếu sót. Chúng em rất mong nhận được những lời nhận xét và góp ý thẳng thắn từ cô. Đó sẽ là những bài học quý giá để chúng em rút kinh nghiệm và trưởng thành hơn trên con đường học tập phía trước.

Chúng em xin chân thành cảm ơn Cô!

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Ký hiệu	Ý nghĩa
ML	Machine Learning (Học máy)
EDA	Exploratory Data Analysis (Phân tích khám phá dữ liệu)
IQR	Interquartile Range (Khoảng tứ phân vị)
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
CV	Cross-Validation (Kiểm chứng chéo)
SVM	Support Vector Machine
RF	Random Forest
LR	Logistic Regression

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

MỞ ĐẦU

A. GIỚI THIỆU ĐỀ TÀI

1. Bối cảnh và tính cấp thiết của đề tài

Theo báo cáo của Tổ chức Y tế Thế giới (WHO), Bệnh tim mạch (Cardiovascular Disease - CVDs) hiện là nguyên nhân gây tử vong hàng đầu trên toàn cầu, cướp đi sinh mạng của khoảng 17,9 triệu người mỗi năm, chiếm 31% tổng số ca tử vong. Tại Việt Nam, dưới tác động của lối sống hiện đại và già hóa dân số, gánh nặng bệnh tật từ các bệnh lý tim mạch đang gia tăng với tốc độ đáng báo động, đặt ra thách thức lớn cho hệ thống y tế quốc gia.

Trong kỷ nguyên Công nghiệp 4.0, sự bùng nổ của dữ liệu y tế cùng với sự phát triển vượt bậc của Trí tuệ nhân tạo (AI) đã mở ra những hướng đi mới trong việc chẩn đoán và tiên lượng bệnh. Việc ứng dụng Học máy (Machine Learning) vào y học không còn là lý thuyết xa vời mà đã trở thành công cụ đắc lực hỗ trợ các bác sĩ đưa ra quyết định chính xác hơn.

Đặt trong bối cảnh môn học Machine Learning, đề tài "Dự đoán người mắc bệnh suy tim sử dụng bộ dữ liệu Heart Failure Prediction Dataset" được lựa chọn không chỉ để đáp ứng yêu cầu học thuật mà còn nhằm mục đích tiếp cận xu hướng "Y tế thông minh" (Smart Healthcare). Đây là cơ hội để nhóm nghiên cứu hiện thực hóa các lý thuyết toán học vào một bài toán phân loại nhị phân (Binary Classification) có ý nghĩa nhân sinh sâu sắc.

2. Ý nghĩa khoa học và mục tiêu nghiên cứu

Việc thực hiện đề tài này hướng tới việc xây dựng một quy trình khai phá dữ liệu hoàn chỉnh, mang lại những giá trị cốt lõi trên ba phương diện:

- Về mặt Kiến thức (Theoretical Contributions):
  - Củng cố và làm sâu sắc thêm hiểu biết về quy trình khoa học dữ liệu (Data Science Pipeline) trong y tế: từ khâu thu thập, làm sạch đến trực quan hóa dữ liệu.
  - Phân tích so sánh hiệu năng giữa các thuật toán học máy kinh điển (Logistic Regression, Random Forest, SVM) để hiểu rõ ưu/nhược điểm của từng mô hình khi xử lý dữ liệu y sinh – vốn thường có đặc thù là nhiễu và mất cân bằng.
- Về mặt Kỹ năng (Technical Skills):
  - Rèn luyện tư duy phản biện trong việc lựa chọn đặc trưng (Feature Selection) và tinh chỉnh tham số (Hyperparameter Tuning) để tối ưu hóa mô hình.
  - Làm chủ các kỹ thuật xử lý dữ liệu nâng cao và các chỉ số đánh giá mô hình chuyên biệt cho y tế (như Recall/Sensitivity – độ nhạy, nhằm giảm thiểu việc bỏ sót bệnh nhân).
- Về mặt Thực tiễn (Practical Implications):

- Xây dựng tiền đề cho một hệ thống hỗ trợ ra quyết định lâm sàng (Clinical Decision Support System - CDSS), giúp sàng lọc sớm các ca bệnh có nguy cơ cao.
- Minh chứng cho tiềm năng của việc kết hợp giữa tri thức y khoa và công nghệ dữ liệu, góp phần giảm tải áp lực cho đội ngũ y bác sĩ và chi phí điều trị cho cộng đồng.
-

B. YÊU CẦU BÀI TOÁN

Dựa trên yêu cầu học thuật của giảng viên hướng dẫn và đặc thù của dữ liệu y sinh, bài toán được xác định và cụ thể hóa thông qua các phạm vi và tiêu chuẩn kỹ thuật sau:

1. Đối tượng và phạm vi nghiên cứu

Đối tượng dữ liệu (Data Subject): Nghiên cứu sử dụng bộ dữ liệu chuẩn hóa Heart Failure Prediction Dataset, bao gồm hồ sơ y tế của 918 bệnh nhân. Dữ liệu được cấu thành từ 11 biến đặc trưng lâm sàng (Clinical Features) đa dạng, bao gồm:

Thông tin nhân khẩu học: Tuổi (Age), Giới tính (Sex).

Chỉ số sinh lý: Huyết áp nghỉ (RestingBP), Cholesterol, Đường huyết lúc đói (FastingBS), Nhịp tim tối đa (MaxHR).

Triệu chứng & Kết quả cận lâm sàng: Loại đau ngực (ChestPainType), Điện tâm đồ (RestingECG), Đau thắt ngực khi vận động (ExerciseAngina), Oldpeak và Độ dốc đoạn ST (ST_Slope).

Biến mục tiêu (Target Variable): HeartDisease (0: Bình thường, 1: Có bệnh tim).

Phạm vi bài toán (Problem Scope): Đề tài giới hạn trong phạm vi bài toán Phân loại nhị phân có giám sát (Supervised Binary Classification). Mục tiêu cốt lõi là xây dựng một hàm ánh xạ f(x) có khả năng dự báo chính xác trạng thái sức khỏe tim mạch của bệnh nhân dựa trên véc-tơ đặc trưng đầu vào, hỗ trợ công tác sàng lọc y tế ban đầu.

2. Các yêu cầu kỹ thuật cần đạt được

Để đảm bảo tính khoa học và độ tin cậy của mô hình dự đoán, nhóm nghiên cứu đặt ra các tiêu chuẩn kỹ thuật khắt khe cho từng giai đoạn:

- Giai đoạn 1: Tiền xử lý và Làm sạch dữ liệu (Data Preprocessing & Cleansing)
  - Thực hiện làm sạch dữ liệu (xử lý giá trị bất thường), mã hóa biến phân loại (Label Encoding, One-Hot Encoding) và chuẩn hóa dữ liệu (StandardScaler).

- Giai đoạn 2: Xây dựng và thiết kế mô hình (Model Development)
  - Xây dựng và so sánh các thuật toán phân loại phổ biến: Logistic Regression, Random Forest, Support Vector Machine.
- Giai đoạn 3: Tối ưu hóa mô hình (Model Optimization)
  - Sử dụng GridSearchCV để tìm siêu tham số tối ưu cho mô hình.
- Giai đoạn 4: Tiêu chuẩn đánh giá (Evaluation Metrics)
  - Mô hình phải đạt được các chỉ số đánh giá (Accuracy, Precision, Recall, F1-Score, ROC-AUC) phù hợp với bài toán y tế, trong đó ưu tiên chỉ số Recall để giảm thiểu việc bỏ sót bệnh nhân.

C. GIỚI THIỆU TẬP DỮ LIỆU

Bộ dữ liệu được sử dụng trong đề tài là "Heart Failure Prediction Dataset" từ Kaggle, được tổng hợp từ nhiều nguồn dữ liệu y khoa uy tín bao gồm Cleveland, Hungary, Switzerland, Long Beach VA và Stalog Heart.

1. Quy mô và cấu trúc dữ liệu

- Tổng số lượng mẫu: 918 bệnh nhân
- Số lượng biến: 12 biến (11 biến đầu vào + 1 biến mục tiêu)
- Tỷ lệ phân lớp: 55% mắc bệnh tim (508 mẫu), 45% không mắc bệnh tim (410 mẫu)

2. Mô tả chi tiết các biến

STT	Tên biến	Loại biến	Mô tả ý nghĩa	Giá trị / Đơn vị
1	Age	Định lượng	Tuổi của bệnh nhân	28 – 77 (năm)
2	Sex	Định tính	Giới tính	M: Nam, F: Nữ
3	ChestPainType	Định tính	Loại đau ngực	TA: Đau thắt ngực điển hình ATA: Đau thắt ngực không điển hình NAP: Đau không do tim

				ASY: Không triệu chứng
4	RestingBP	Định lượng	Huyết áp khi nghỉ	mmHg
5	Cholesterol	Định lượng	Nồng độ cholesterol	mg/dL
6	FastingBS	Định tính	Đường huyết khi đói	0: ≤ 120 mg/dL 1: > 120 mg/dL
7	RestingECG	Định tính	Kết quả điện tâm đồ	Normal: Bình thường ST: Có sóng ST-T bất thường LVH: Phì đại thất trái
8	MaxHR	Định lượng	Nhịp tim tối đa	60 – 202 (bpm)
9	ExerciseAngina	Định tính	Đau ngực khi gắng sức	Y: Có N: Không
10	Oldpeak	Định lượng	Độ chênh đoạn ST	mm (Giá trị số thực)
11	ST_Slope	Định tính	Độ dốc đoạn ST	Up: Dốc lên Flat: Bằng phẳng Down: Dốc xuống
12	HeartDisease	Mục tiêu	Kết quả chẩn đoán	0: Không bệnh 1: Có bệnh

Bảng 1 Mô tả chi tiết các biến

CHƯƠNG I: TIỀN XỬ LÝ DỮ LIỆU

1.Tải dữ liệu và xem dữ liệu

1.1 Import thư viện và đọc dữ liệu

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math

df = pd.read_csv("heart.csv")
df.head()
```

Giải thích chi tiết:

pandas (pd): Thư viện xử lý dữ liệu dạng bảng, cung cấp cấu trúc DataFrame giúp thao tác và phân tích dữ liệu hiệu quả.

numpy (np): Thư viện tính toán số học, hỗ trợ các phép toán trên mảng đa chiều.

matplotlib.pyplot (plt): Thư viện vẽ biểu đồ cơ bản trong Python.

seaborn (sns): Thư viện trực quan hóa dữ liệu nâng cao, tích hợp tốt với pandas.

math: Thư viện toán học chuẩn của Python.

pd.read_csv("heart.csv"): Đọc dữ liệu từ file CSV và chuyển thành DataFrame. File heart.csv gồm 918 quan sát về bệnh tim.

df.head(): Hiển thị 5 dòng dữ liệu đầu tiên nhằm kiểm tra nhanh cấu trúc dữ liệu.

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Hình 1 5 dòng đầu của tập dữ liệu

Nhận xét: 5 dòng đầu tiên cho thấy mỗi quan sát tương ứng với một bệnh nhân, các cột thể hiện các thông số lâm sàng và biến mục tiêu HeartDisease.

1.1. Kiểm tra thông tin tổng quát

```
# Xem thông tin tổng quan của DataFrame
df.info()
```

Giải thích

- df.info() được sử dụng để hiển thị thông tin tổng quan của DataFrame, bao gồm:
  - Số lượng dòng (observations) và số lượng cột (features)
  - Tên các cột và kiểu dữ liệu tương ứng (int64, float64, object)
  - Số lượng giá trị không bị thiếu (non-null) của từng cột
  - Dung lượng bộ nhớ mà DataFrame sử dụng

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

Hình 2 Thông tin tổng quan của dữ liệu.

Nhận xét

- Tập dữ liệu gồm 918 dòng dữ liệu và 12 cột.
- Dữ liệu bao gồm:
  - Các biến số (int64, float64)
  - Một số biến phân loại (object)
- Không tồn tại giá trị bị thiếu trong tập dữ liệu, tất cả các cột đều có 918 non-null values.

1.2. Kiểm tra dữ liệu thiếu và trùng lặp

```
# Kiểm tra giá trị thiếu (NaN) và trùng lặp
print(f"- Tổng số dòng trùng lặp: {df.duplicated().sum()}")
print("\n- Giá trị thiếu (NaN) từng cột:")
print(df.isnull().sum())
```

Hình 3 Kiểm tra dữ liệu trùng và thiếu

Giải thích chi tiết:

- df.duplicated(): Trả về Series boolean, True nếu dòng đó trùng lặp với dòng trước đó.
- .sum(): Đếm tổng số dòng trùng lặp.
- df.isnull(): Trả về DataFrame boolean, True nếu giá trị là NaN (null/missing).
- df.isnull().sum(): Đếm số giá trị thiếu của từng cột.

```
- Tổng số dòng trùng lặp: 0

- Giá trị thiếu (NaN) từng cột:
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

Hình 4 Kiểm tra giá trị trùng lặp

Nhận xét: Kết quả cho thấy dataset không có giá trị bị trùng lặp và không có giá trị NaN.

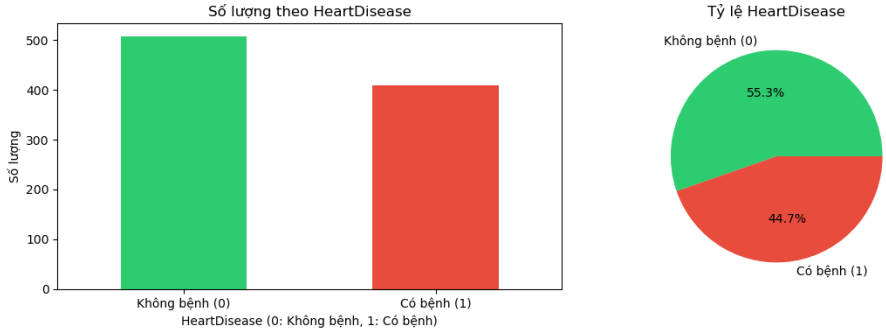
1.3. Kiểm tra mất cân bằng dữ liệu (Imbalanced Data)

```python
# Kiểm tra phân phối biến mục tiêu
print("Phân phối biến mục tiêu HeartDisease:")
print(df['HeartDisease'].value_counts())
print("\nTỷ lệ phần trăm:")
print(df['HeartDisease'].value_counts(normalize=True).mul(100).round(2))

# Trực quan hóa
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

# Biểu đồ cột
df['HeartDisease'].value_counts().plot(kind='bar', ax=axes[0],
                                       color=['#2ecc71', '#e74c3c'])
axes[0].set_title('Số lượng theo HeartDisease')

# Biểu đồ tròn
df['HeartDisease'].value_counts().plot(kind='pie', ax=axes[1],
                                       autopct='%1.1f%%',
                                       colors=['#2ecc71', '#e74c3c'])
axes[1].set_title('Tỷ lệ HeartDisease')

plt.tight_layout()
plt.show()
```

Giải thích chi tiết:

- value_counts(): Đếm số lượng mỗi giá trị trong cột.
- normalize=True: Chuyển đổi thành tỷ lệ phần trăm (0-1).
- mul(100).round(2): Nhân 100 và làm tròn 2 chữ số để hiển thị %.
- plt.subplots(1, 2): Tạo figure với 1 hàng, 2 cột để vẽ 2 biểu đồ cạnh nhau.
- kind='bar': Vẽ biểu đồ cột.
- kind='pie': Vẽ biểu đồ tròn.
- autopct='%1.1f%%': Hiển thị % trên biểu đồ tròn với 1 chữ số thập phân.

```
Phân phối biến mục tiêu HeartDisease:
HeartDisease
1    508
0    410
Name: count, dtype: int64

Tỷ lệ phần trăm:
HeartDisease
1    55.34
0    44.66
Name: proportion, dtype: float64
```



Hình 5 biểu đồ IMBALANCED

Nhận xét về mất cân bằng dữ liệu:

- Lớp HeartDisease = 1 (mắc bệnh) chiếm khoảng 55% (508 mẫu)
- Lớp HeartDisease = 0 (không mắc bệnh) chiếm khoảng 45% (410 mẫu)

Kết luận: Dữ liệu có sự mất cân bằng nhẹ (tỷ lệ 55:45), không nghiêm trọng như các bài toán với tỷ lệ 95:5. Tuy nhiên, nhóm vẫn sử dụng các kỹ thuật xử lý như stratify khi chia dữ liệu và class_weight='balanced' trong mô hình để đảm bảo công bằng.

1.4. Thống kê mô tả các biến

1.4.1. Thống kê mô tả biến số

```
df.describe()
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Hình 6 Thống kê các biến.

Nhận xét quan trọng:

- RestingBP = 0 và Cholesterol = 0: Đây là các giá trị không hợp lý về mặt y khoa (huyết áp và cholesterol của người sống không thể bằng 0). Nhiều khả năng đây là dữ liệu thiếu được mã hóa bằng 0 → Cần xử lý ở bước làm sạch.
- Oldpeak có giá trị âm (-2.6): Giá trị âm của ST depression có ý nghĩa lâm sàng (ST elevation), cần được xem xét kỹ.
- Cholesterol std = 109.4: Độ biến động rất lớn, có khả năng tồn tại outliers.

1.4.2. Thống kê mô tả biến phân loại.

```
df.describe(include='object')
```

	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
count	918	918	918	918	918
unique	2	4	3	2	3
top	M	ASY	Normal	N	Flat
freq	725	496	552	547	460

Hình 7 Thống kê mô tả biến phân loại.

Nhận xét:

- Nam giới chiếm đa số (725/918 ≈ 79%), phản ánh thực tế nam giới có nguy cơ bệnh tim cao hơn
- ASY (không triệu chứng) là loại đau ngực phổ biến nhất và cũng nguy hiểm nhất do dễ bị bỏ sót
- ST_Slope = Flat xuất hiện nhiều nhất, đây là dấu hiệu bất thường liên quan đến thiếu máu cơ tim

1.5. Đánh giá mức độ đa dạng của từng biến

```
summary_count = pd.DataFrame({
    "Cột": df.columns,
    "Số lượng": df.count().values,
    "Số lượng giá trị khác nhau": [df[col].nunique() for col in df.columns],
    "Tỷ lệ khác nhau (%)": [df[col].nunique() / df[col].count() * 100 for col in df.columns]
})
summary_count
```

	Cột	Số lượng	Số lượng giá trị khác nhau	Tỷ lệ khác nhau (%)
0	Age	918	50	5.45
1	Sex	918	2	0.22
2	ChestPainType	918	4	0.44
3	RestingBP	918	67	7.30
4	Cholesterol	918	222	24.18
5	FastingBS	918	2	0.22
6	RestingECG	918	3	0.33
7	MaxHR	918	119	12.96
8	ExerciseAngina	918	2	0.22
9	Oldpeak	918	53	5.77
10	ST_Slope	918	3	0.33
11	HeartDisease	918	2	0.22

Hình 8 Mức độ phổ biến của các biến.

Nhận xét:

- Các biến nhị phân (Sex, FastingBS, ExerciseAngina, HeartDisease) có tỷ lệ khác nhau rất thấp (~0.22%)
- Các biến định lượng như Cholesterol (24.18%), MaxHR (12.96%) có độ đa dạng cao → Mang nhiều thông tin phân biệt
- Kết quả này là cơ sở để lựa chọn phương pháp mã hóa phù hợp ở bước tiền xử lý

CHƯƠNG II. KHAI PHÁ VÀ LÀM SẠCH DỮ LIỆU

2.1. Phát hiện Outlier bằng phương pháp IQR

Phương pháp IQR (Interquartile Range) là kỹ thuật phổ biến để phát hiện các giá trị ngoại lệ dựa trên phân phối dữ liệu.
Công thức toán học:

$$IQR = Q_3 - Q_1$$
$$\text{Lower Bound} = Q_1 - 1.5 \times IQR$$
$$\text{Upper Bound} = Q_3 + 1.5 \times IQR$$

Trong đó:
Q_1: Tứ phân vị thứ nhất (25th percentile)
Q_3: Tứ phân vị thứ ba (75th percentile)
Outlier: Giá trị nằm ngoài khoảng. [lower bound, upper bound]
Code thực hiện:

```
iqr_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']

Q1 = df[iqr_cols].quantile(0.25)
Q3 = df[iqr_cols].quantile(0.75)
IQR = Q3 - Q1

outliers = ((df[iqr_cols] < (Q1 - 1.5 * IQR)) |
            (df[iqr_cols] > (Q3 + 1.5 * IQR))).sum()

outliers
```

Giải thích code:

quantile(0.25): Tính giá trị tại phân vị 25% (Q1)

quantile(0.75): Tính giá trị tại phân vị 75% (Q3)

Điều kiện < (Q1 - 1.5*IQR) hoặc > (Q3 + 1.5*IQR): Xác định outlier

.sum(): Đếm số lượng outlier trong mỗi cột

Tại sao dùng hệ số 1.5?
Hệ số 1.5 được John Tukey đề xuất năm 1977, đảm bảo với phân phối chuẩn thì khoảng 99.3% dữ liệu nằm trong ngưỡng này. Đây là ngưỡng cân bằng giữa việc phát hiện outlier thực sự và tránh loại bỏ dữ liệu hợp lệ.

```
Age             0
RestingBP      28
Cholesterol   183
MaxHR           2
Oldpeak        16
dtype: int64
```

Hình 9: kết quả Outlier.

Nhận xét:

- Biến Age không xuất hiện outlier → Phân bố tuổi hợp lý.
- Biến RestingBP và Cholesterol có số lượng outlier đáng kể.
- Biến Oldpeak xuất hiện một số giá trị ngoại lai nhưng mang ý nghĩa lâm sàng.

2.2 Kiểm tra giá trị bất thường

```python
print("Kiểm tra số giá trị = 0 và giá trị âm trong các cột số:\n")

for col in df.select_dtypes(include='number').columns:
    print(col)
    print(f"\t Số giá trị = 0: {(df[col] == 0).sum()}")
    print(f"\t Số giá trị âm: {(df[col] < 0).sum()}")
```

```
Kiểm tra số giá trị = 0 và giá trị âm trong các cột số:

Age
         Số giá trị = 0: 0
         Số giá trị âm: 0
RestingBP
         Số giá trị = 0: 1
         Số giá trị âm: 0
Cholesterol
         Số giá trị = 0: 172
         Số giá trị âm: 0
FastingBS
         Số giá trị = 0: 704
         Số giá trị âm: 0
MaxHR
         Số giá trị = 0: 0
         Số giá trị âm: 0
Oldpeak
         Số giá trị = 0: 368
         Số giá trị âm: 13
HeartDisease
         Số giá trị = 0: 410
         Số giá trị âm: 0
```

Hình 10 Kiểm tra giá trị bất thường.

Phân tích y học:

1. RestingBP = 0 (1 mẫu): Huyết áp nghỉ ngơi của người sống không thể bằng 0. Đây là dữ liệu thiếu được mã hóa bằng 0.
2. Cholesterol = 0 (172 mẫu): Nồng độ cholesterol trong máu không thể bằng 0. Đây là dữ liệu thiếu được mã hóa bằng 0 và chiếm tỷ lệ đáng kể (172/918 ≈ 18.7%).
3. Oldpeak âm (-2.6): Giá trị âm của ST depression thực chất là ST elevation, có ý nghĩa lâm sàng trong chẩn đoán bệnh tim → Giữ nguyên.
4. Oldpeak = 0 (384 mẫu): Giá trị 0 có ý nghĩa lâm sàng (không có ST depression) → Giữ nguyên.

2.3 Xử lý và làm sạch dữ liệu

Chiến lược xử lý:

.

Biến	Vấn đề	Cách xử lý	Lý do
RestingBP	1 giá trị = 0	Thay bằng Median	Median ít bị ảnh hưởng bởi outlier
Cholesterol	172 giá trị = 0	Thay bằng Median	Median phản ánh tốt giá trị trung tâm
Oldpeak	Giá trị âm	Giữ nguyên	Có ý nghĩa lâm sàng (ST elevation)

Bảng 2 Chiến lược xử lý làm sạch dữ liệu

Code xử lý

```python
df_clean = df.copy()

# 1. RestingBP: giá trị = 0 là không hợp lý → thay bằng median (loại 0)
rbp_median = df_clean.loc[df_clean['RestingBP'] > 0, 'RestingBP'].median()
df_clean.loc[df_clean['RestingBP'] == 0, 'RestingBP'] = rbp_median

# 2. Cholesterol: giá trị = 0 là không hợp lý → thay bằng median (loại 0)
chol_median = df_clean.loc[df_clean['Cholesterol'] > 0, 'Cholesterol'].median()
df_clean.loc[df_clean['Cholesterol'] == 0, 'Cholesterol'] = chol_median

# 3. Oldpeak: GIỮ NGUYÊN (kể cả giá trị âm)

# Kiểm tra nhanh sau xử lý
print("Kiểm tra sau xử lý:\n")
for col in ['RestingBP', 'Cholesterol', 'Oldpeak']:
    print(col)
    print(f"  Số giá trị = 0: {(df_clean[col] == 0).sum()}")
    print(f"  Số giá trị âm: {(df_clean[col] < 0).sum()}")
```

Giải thích chi tiết từng dòng code:

Dòng code	Giải thích
df_clean = df.copy()	Tạo bản sao để không ảnh hưởng dữ liệu gốc
df_clean.loc[df_clean['RestingBP'] > 0, 'RestingBP'].median()	Tính median chỉ từ các giá trị > 0 (loại bỏ các giá trị 0 khỏi phép tính)
df_clean.loc[df_clean['RestingBP'] == 0, 'RestingBP'] = rbp_median	Thay thế các giá trị = 0 bằng median vừa tính

Bảng 3 Chi tiết code xử lý làm sạch dữ liệu.

Tại sao chọn Median thay vì Mean?

Phương pháp	Ưu điểm	Nhược điểm
Mean (Trung bình)	Tận dụng tất cả dữ liệu	Bị ảnh hưởng mạnh bởi outlier
Median (Trung vị)	Ít bị ảnh hưởng bởi outlier	Không tận dụng hết thông tin

Bảng 4 Giải thích các xử lý.

→ Trong trường hợp này, Cholesterol có nhiều outlier (giá trị rất cao đến 603), nên Median là lựa chọn phù hợp hơn.

Kết quả sau xử lý:

```
RestingBP
  Số giá trị = 0: 0
  Số giá trị âm: 0
Cholesterol
  Số giá trị = 0: 0
  Số giá trị âm: 0
Oldpeak
  Số giá trị = 0: 368
  Số giá trị âm: 13
```

Hình 11 Kết quả sau khi xử lý.

2.4. Phân tích phân phối theo biến mục tiêu

Sau khi làm sạch dữ liệu, tiến hành phân tích sự khác biệt giữa nhóm mắc bệnh và không mắc bệnh.

2.4.1. So sánh đặc điểm trung bình:

```python
numeric_cols = df_clean.select_dtypes(include='number').columns.drop('HeartDisease')

summary_target_clean = (
    df_clean
    .groupby('HeartDisease')[numeric_cols]
    .agg(['mean', 'median'])
    .round(2)
)

summary_target_clean
```

Kết quả:

Biến	HeartDisease=0 (Mean)	HeartDisease=1 (Mean)	Nhận xét
Age	50.6	55.9	Người mắc bệnh có tuổi cao hơn
RestingBP	130.2	134.4	Huyết áp cao hơn ở nhóm mắc bệnh
Cholesterol	238.7	246.9	Cholesterol cao hơn nhẹ
MaxHR	148.2	127.7	Nhịp tim tối đa thấp hơn ở nhóm mắc bệnh
Oldpeak	0.41	1.27	ST depression cao hơn đáng kể

Bảng 5 Mô tả kết quả.

Phát hiện quan trọng:

- Oldpeak có sự khác biệt rõ rệt nhất (0.41 vs 1.27) → Biến này có khả năng phân biệt tốt
- MaxHR thấp hơn ở nhóm mắc bệnh → Khả năng gắng sức tim mạch giảm
- Age cao hơn ở nhóm mắc bệnh → Tuổi là yếu tố nguy cơ

	Age		RestingBP		Cholesterol		FastingBS		MaxHR		Oldpeak	
	mean	median	mean	median	mean	median	mean	median	mean	median	mean	median
HeartDisease												
0	50.55	51.0	130.18	130.0	238.68	235.0	0.11	0.0	148.15	150.0	0.41	0.0
1	55.90	57.0	134.44	132.0	246.85	237.0	0.33	0.0	127.66	126.0	1.27	1.2

Hình 12 So sánh đặc điểm trung bình của người mắc bệnh tim và không mắc bệnh tim

2.4.2. Phân tích biến phân loại

```
categorical_cols = df_clean.select_dtypes(include='object').columns

for col in categorical_cols:
    ct = (
        pd.crosstab(df_clean[col], df_clean['HeartDisease'], normalize='index')
        .mul(100)
        .round(2)
    )
    print(f"\nTỉ lệ HeartDisease theo {col}:")
    print(ct)
```

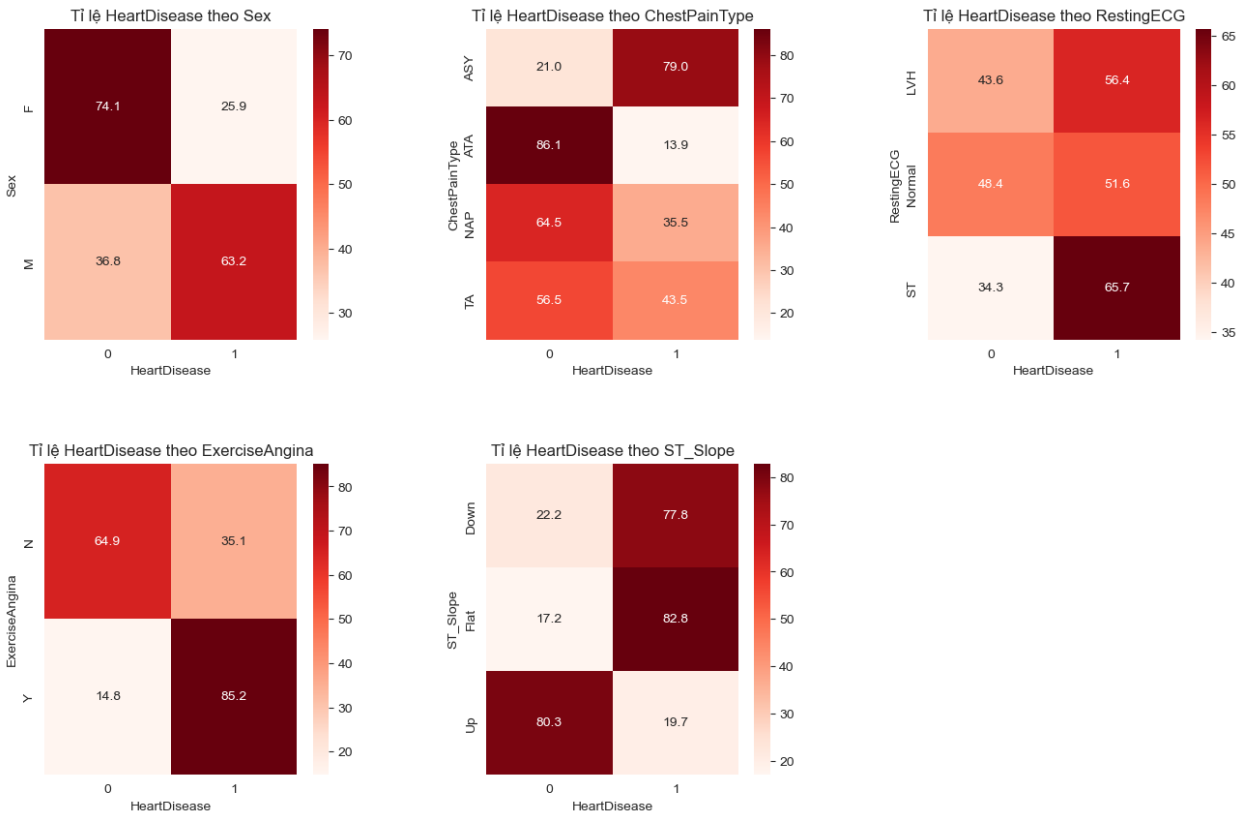
Kết quả:

Biến	Giá trị	Tỷ lệ mắc bệnh (%)
Sex	M (Nam)	63.17%
	F (Nữ)	25.91%
ChestPainType	ASY	79.03%
	ATA	13.87%
	NAP	35.00%
	TA	42.11%
ExerciseAngina	Y (Có)	85.18%
	N (Không)	35.10%
ST_Slope	Flat	82.83%
	Down	77.78%
	Up	19.75%

Bảng 6 Phân tích biến phân loại

Nhận xét quan trọng:

- Nam giới có tỷ lệ mắc bệnh tim cao gấp 2.4 lần nữ giới
- ASY (không triệu chứng) có tỷ lệ mắc bệnh cao nhất (79%) → Đau ngực không điển hình nguy hiểm nhất
- ExerciseAngina = Y có tỷ lệ mắc bệnh 85% → Đau thắt ngực khi vận động là dấu hiệu nghiêm trọng
- ST_Slope = Flat/Down có tỷ lệ mắc bệnh rất cao (>77%) → Độ dốc ST bất thường là chỉ số quan trọng



Hình 13 : Heatmap tỉ lệ mắc bệnh theo biến phân loại.

2.5 Ma trận tương quan

```python
# Chọn các biến số để tính correlation
numeric_df = df_clean.select_dtypes(include=['int64', 'float64'])

# Tính ma trận tương quan
correlation_matrix = numeric_df.corr()

# Vẽ heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(
    correlation_matrix,
    annot=True,
    fmt='.2f',
    cmap='RdBu_r',
    center=0,
    square=True,
    linewidths=0.5,
    vmin=-1, vmax=1
)
plt.title('Ma trận tương quan giữa các biến số', fontsize=14)
plt.tight_layout()
plt.show()

# Hiển thị tương quan với biến mục tiêu
print("\n📊 Tương quan với HeartDisease (sắp xếp theo độ mạnh):")
target_corr = correlation_matrix['HeartDisease'].drop('HeartDisease').sort_values(key=abs, ascending=False)
print(target_corr.to_string())
```

Giải thích các tham số:

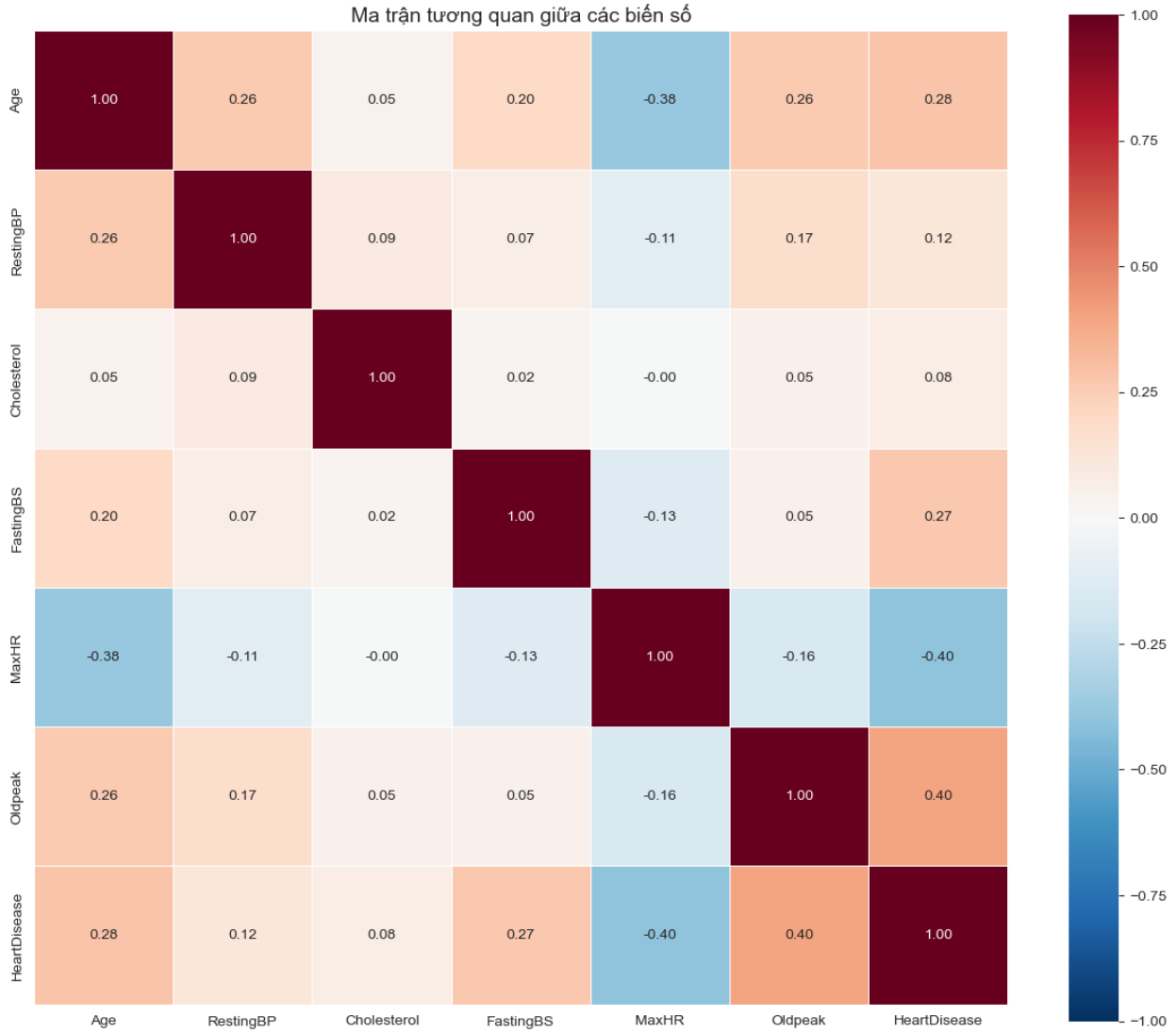
annot=True: Hiển thị giá trị correlation trên heatmap

cmap='RdBu_r': Bảng màu đỏ-xanh (đỏ = dương, xanh = âm)

center=0: Đặt trung tâm màu tại 0

vmin=-1, vmax=1: Giới hạn giá trị từ -1 đến 1

Kết quả tương quan với HeartDisease:



Hình 14 Ma trận tương quan giữa các biến số

Biến	Correlation	Ý nghĩa
Oldpeak	+0.40	Tương quan dương mạnh nhất
MaxHR	-0.40	Tương quan âm mạnh
FastingBS	+0.27	Tương quan dương vừa
Age	+0.28	Tương quan dương vừa
RestingBP	+0.11	Tương quan yếu
Cholesterol	+0.06	Tương quan rất yếu

Bảng 7 Phân tích ý nghĩa các biến.

Nhận xét:

Tương quan với biến mục tiêu:

Oldpeak và MaxHR có tương quan mạnh nhất → Biến quan trọng nhất

Cholesterol có tương quan yếu → Có thể không phải yếu tố quyết định

Kiểm tra đa cộng tuyến:

Không có cặp biến nào có |correlation| > 0.8

Age và MaxHR có tương quan âm (~-0.40) → Hợp lý về sinh lý

→ Không cần loại bỏ biến do đa cộng tuyến

CHƯƠNG III. TIỀN XỬ LÝ DỮ LIỆU

Sau khi làm sạch dữ liệu, bước tiếp theo là tiền xử lý để chuyển dữ liệu về dạng phù hợp cho các thuật toán Machine Learning.

3.1. Mã hóa phân loại biến (Encoding)

Các thuật toán Machine Learning yêu cầu dữ liệu đầu vào ở dạng số. Do đó, cần mã hóa (encode) các biến categorical.

Chiến lược mã hóa:

Loại biến	Phương pháp	Biến áp dụng
Biến nhị phân (2 giá trị)	Label Encoding	Sex, ExerciseAngina
Biến đa lớp (>2 giá trị)	One-Hot Encoding	ChestPainType, RestingECG, ST_Slope

Bảng 8 Mã hóa chiến lược.

3.1.1. Label Encoding (cho biến nhị phân)

```
from sklearn.preprocessing import LabelEncoder

df_encoded = df_clean.copy()

# Label Encoding cho các biến nhị phân
label_cols = ['Sex', 'ExerciseAngina']

le = LabelEncoder()
for col in label_cols:
    df_encoded[col] = le.fit_transform(df_encoded[col])
```

Giải thích:

LabelEncoder(): Chuyển đổi mỗi giá trị unique thành số nguyên (0, 1, 2,...)

Phù hợp cho biến nhị phân vì không tạo ra thứ tự giả

Kết quả:

Biến	Giá trị gốc	Sau mã hóa
Sex	F	0
Sex	M	1
ExerciseAngina	N	0
ExerciseAngina	Y	1

Bảng 9 Kết quả mã hóa

3.1.2. One-Hot Encoding (cho biến đa lớp)

```
# One-Hot Encoding cho các biến còn lại
onehot_cols = ['ChestPainType', 'RestingECG', 'ST_Slope']
df_encoded = pd.get_dummies(df_encoded, columns=onehot_cols, drop_first=True)

df_encoded.head()
```

Giải thích các tham số:

Tham số	Giá trị	Ý nghĩa
columns	['ChestPainType', 'RestingECG', 'ST_Slope']	Danh sách các cột cần mã hóa
drop_first	True	Bỏ cột đầu tiên để tránh Dummy Variable Trap

Bảng 10 Giải thích tham số mã hóa.

Tại sao cần drop_first=True?

Nếu một biến có **k** giá trị, One-Hot Encoding tạo ra **k** cột. Tuy nhiên, cột cuối cùng có thể suy ra từ **k-1** cột còn lại (nếu tất cả = 0 thì cột cuối = 1). Điều này gây ra đa cộng tuyến hoàn hảo trong các mô hình tuyến tính.

Ví dụ với ChestPainType (4 giá trị):

Giá trị gốc	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA
ASY (bị drop)	0	0	0
ATA	1	0	0
NAP	0	1	0
TA	0	0	1

Bảng 11 Giải thích tại sao cần drop_first=True

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	HeartDisease	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST	ST_Slope_Flat	ST_Slope_Up
0	40	1	140	289	0	172	0	0.0	0	True	False	False	True	False	False	True
1	49	0	160	180	0	156	0	1.0	1	False	True	False	True	False	True	False
2	37	1	130	283	0	98	0	0.0	0	True	False	False	False	True	False	True
3	48	0	138	214	0	108	1	1.5	1	False	False	False	True	False	True	False
4	54	1	150	195	0	122	0	0.0	0	False	True	False	True	False	False	True

Hình 15 Kết quả sau khi xử lý.

Kết quả sau mã hóa:

Số cột ban đầu: 12

Số cột sau mã hóa: 16 (6 cột số + 2 Label + 8 One-Hot)

Không còn cột dữ liệu dạng object

3.2. Chia tập Train/Test

Tại sao phải chia Train/Test TRƯỚC khi chuẩn hóa?

Nếu chuẩn hóa TRƯỚC rồi mới chia → Thông tin thống kê (mean, std) của tập test sẽ "rò rỉ" vào quá trình huấn luyện → Kết quả đánh giá không phản ánh đúng khả năng tổng quát hóa của mô hình.

Quy trình đúng: Chia train/test TRƯỚC → Chuẩn hóa SAU (fit trên train, transform cả train và test)

Code thực hiện:

```python
from sklearn.model_selection import train_test_split

# Tách X (features) và y (target)
X = df_encoded.drop('HeartDisease', axis=1)
y = df_encoded['HeartDisease']

# Chia tập huấn luyện và kiểm tra
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3,
    random_state=42,
    stratify=y
)

print("Kích thước tập huấn luyện:", X_train.shape)
print("Kích thước tập kiểm tra:", X_test.shape)
```

Giải thích chi tiết các tham số:

Tham số	Giá trị	Ý nghĩa
test_size	0.3	30% dữ liệu dành cho test, 70% cho training
random_state	42	Seed cố định để đảm bảo reproducibility (kết quả giống nhau khi chạy lại)
stratify	y	Stratified sampling: Đảm bảo tỷ lệ lớp (0/1) trong train và test giống với tỷ lệ trong dataset gốc

Bảng 12 Chi tiết các tham số.

Tại sao cần stratify=y?

Với dữ liệu mất cân bằng (55% positive, 45% negative), nếu chia ngẫu nhiên, có thể tập test chỉ có toàn lớp 0 hoặc toàn lớp 1 → Đánh giá mô hình sai lệch.

Kết quả:

```
Kích thước tập huấn luyện: (642, 15)
Kích thước tập kiểm tra: (276, 15)
```

Hình 16 Kết quả sau khi chia tập Train/Test.

Tập	Số mẫu	Tỷ lệ
Train	642	70%
Test	276	30%

Bảng 13 Kết quả chia tập Train/Test.

3.3. Chuẩn hóa dữ liệu (Standardization)

Tại sao cần chuẩn hóa?

Các biến số có thang đo khác nhau:

- Age: 28 - 77 (vài chục)
- Cholesterol: 0 - 603 (hàng trăm)
- Oldpeak: -2.6 - 6.2 (số thực nhỏ)

Nếu không chuẩn hóa, các biến có giá trị lớn sẽ chi phối quá trình học, đặc biệt với:

- Logistic Regression (gradient descent)
- SVM (dựa trên khoảng cách)
- K-NN (dựa trên khoảng cách)

Phương pháp StandardScaler (Z-score Normalization)

Công thức toán học:

$$z = \frac{x - \mu}{\sigma}$$

Trong đó:

- x: Giá trị gốc
- μ: Giá trị trung bình (mean) của feature
- σ: Độ lệch chuẩn (standard deviation) của feature
- z: Giá trị sau chuẩn hóa

Kết quả: Dữ liệu sau chuẩn hóa có mean = 0 và std = 1

Code thực hiện:

```python
from sklearn.preprocessing import StandardScaler

# Xác định các cột số cần chuẩn hóa
numeric_cols = X_train.select_dtypes(include=['int64', 'float64']).columns

# Khởi tạo scaler
scaler = StandardScaler()

# Fit scaler trên tập TRAIN
X_train_scaled = X_train.copy()
X_train_scaled[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])

# Transform tập TEST bằng scaler đã fit
X_test_scaled = X_test.copy()
X_test_scaled[numeric_cols] = scaler.transform(X_test[numeric_cols])

X_train_scaled.head()
```

Giải thích từng bước:

Bước	Code	Ý nghĩa
1	scaler = StandardScaler()	Khởi tạo object StandardScaler
2	scaler.fit_transform(X_train[numeric_cols])	Fit: Tính μ và σ từ tập TRAIN, sau đó Transform: Áp dụng công thức z-score
3	scaler.transform(X_test[numeric_cols])	Chỉ Transform: Sử dụng μ và σ đã tính từ train để transform test

Bảng 14 Từng bước chuẩn hóa.

Tại sao fit trên TRAIN, transform trên TEST?

CÁCH ĐÚNG:

- Fit trên TRAIN: Tính μ, σ chỉ từ dữ liệu huấn luyện

- Transform trên TEST: Dùng μ, σ đã tính để chuyển đổi

CÁCH SAI:

- Fit trên toàn bộ dữ liệu (bao gồm test) → DATA LEAKAGE!

- Thông tin từ test "rò rỉ" vào quá trình training

Lý do:

1. Tránh Data Leakage: Trong thực tế, chúng ta không biết trước dữ liệu mới (test)
2. Mô phỏng production: Khi triển khai, chỉ có thể dùng thống kê từ training data
3. Đánh giá công bằng: Kết quả trên tập test phản ánh đúng khả năng tổng quát hóa

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST	ST_Slope_Flat	ST_Slope_Up
335	0.856064	0.520852	-0.672618	-0.133186	1.868023	-0.082372	-0.841099	-1.563271	False	False	True	False	False	True	False
368	0.331343	0.520852	0.435467	-0.133186	-0.535325	-0.629164	1.188921	1.037503	False	False	False	True	False	True	False
111	0.331343	0.520852	0.989510	0.205747	-0.535325	-1.722749	1.188921	1.966351	False	False	False	True	False	True	False
336	1.905508	0.520852	1.543553	-0.133186	-0.535325	-0.863504	-0.841099	0.665964	False	True	False	False	False	True	False
479	0.226399	0.520852	2.097596	-0.133186	-0.535325	-0.511994	1.188921	1.501927	False	True	False	False	False	True	False

Hình 17 Kết quả x_train_scaled.head()

CHƯƠNG IV. XÂY DỰNG VÀ HUẤN LUYỆN MÔ HÌNH

4.1 Giới thiệu

Sau khi hoàn tất quá trình tiền xử lý dữ liệu, tập dữ liệu đã được làm sạch, mã hóa và chuẩn hóa theo đúng quy trình học máy. Chương này trình bày chi tiết quá trình xây dựng, huấn luyện và đánh giá các mô hình Machine Learning nhằm dự đoán nguy cơ mắc bệnh suy tim.

Các mô hình được lựa chọn:

- Logistic Regression (Hồi quy Logistic)
- Random Forest (Rừng ngẫu nhiên)
- Support Vector Machine - SVM (Máy vector hỗ trợ)

Lý do lựa chọn 3 mô hình này:

Mô hình	Lý do lựa chọn
Logistic Regression	Mô hình baseline đơn giản, dễ diễn giải, phù hợp với bài toán phân loại nhị phân. Hoạt động tốt khi mối quan hệ giữa biến độc lập và biến phụ thuộc là tuyến tính.
Random Forest	Mô hình ensemble mạnh, có khả năng học các mối quan hệ phi tuyến, ít bị overfitting, và cho phép đánh giá Feature Importance.
SVM (RBF Kernel)	Hiệu quả trong không gian nhiều chiều, có khả năng xử lý ranh giới phi tuyến thông qua kernel trick. Đặc biệt phù hợp với dữ liệu y tế đã được chuẩn hóa.

Bảng 15 Lí do lựa chọn mô hình.

Mục tiêu:

- So sánh hiệu năng giữa mô hình đơn giản (Logistic) và phức tạp (RF, SVM)
- Đánh giá mô hình nào phù hợp nhất cho bài toán dự đoán bệnh tim
- Ưu tiên Recall cao để giảm thiểu việc bỏ sót bệnh nhân mắc bệnh

4.2. Mô hình 1: Logistic Regression (Hồi quy Logistic)

4.2.1. Lý thuyết toán học

Logistic Regression là thuật toán phân loại tuyến tính, mặc dù tên gọi có chữ "Regression". Thuật toán sử dụng hàm Sigmoid (còn gọi là hàm Logistic) để chuyển đổi đầu ra tuyến tính thành xác suất.

Bước 1: Tính tổ hợp tuyến tính

$$z=w_0+w_1x_1+w_2x_2+...+w_nx_n=w^Tx+b$$

Trong đó:

- $x=[x_1,x_2,...,x_n]$: Vector đặc trưng (features) của bệnh nhân
- $w=[w_1,w_2,...,w_n]$: Vector trọng số (weights) cần học
- b (hoặc w_0): Bias term

Bước 2: Áp dụng hàm Sigmoid

$$\sigma(z)=\frac{1}{1+e^{-z}}$$

Đặc điểm của hàm Sigmoid:

- Đầu vào: $z\in(-\infty,+\infty)$
- Đầu ra: $\sigma(z)\in(0,1)$
- Khi $z=0$: $\sigma(0)=0.5$
- Khi $z\to+\infty$: $\sigma(z)\to1$
- Khi $z\to-\infty$: $\sigma(z)\to0$

Bước 3: Quyết định phân loại

$$\hat{y}=\begin{cases}1 & \text{nếu } \sigma(z)\geq0.5\\ 0 & \text{nếu } \sigma(z)<0.5\end{cases}$$

Hàm mất mát (Loss Function) - Binary Cross-Entropy:

$$L(w)=-\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)})+(1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

Thuật toán tối ưu hóa (như Gradient Descent hoặc L-BFGS) sẽ tìm bộ trọng số w sao cho $L(w)$ nhỏ nhất.

4.2.2. Triển khai trong Python

Code khởi tạo và huấn luyện mô hình:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Khởi tạo mô hình
log_reg = LogisticRegression(max_iter=1000, random_state=42)

# Huấn luyện mô hình
log_reg.fit(X_train_scaled, y_train)

# Dự đoán trên tập test
y_pred_lr = log_reg.predict(X_test_scaled)

# Đánh giá mô hình
print("Accuracy (Logistic Regression):", accuracy_score(y_test, y_pred_lr))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_lr))

confusion_matrix(y_test, y_pred_lr)
```

4.2.3. Giải thích các tham số

Tham số	Giá trị	Ý nghĩa chi tiết
max_iter	1000	Số vòng lặp tối đa cho thuật toán tối ưu hội tụ. Giá trị mặc định là 100, nhưng với dữ liệu phức tạp có thể cần nhiều hơn. Nếu thuật toán chưa hội tụ, sklearn sẽ báo warning.
random_state	42	Seed ngẫu nhiên để đảm bảo reproducibility - chạy lại code nhiều lần vẫn cho kết quả giống nhau. Số 42 được chọn theo truyền thống (từ "The Hitchhiker's Guide to the Galaxy").
solver	'lbfgs' (mặc định)	Thuật toán tối ưu hóa. L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) là phương pháp quasi-Newton, hiệu quả với dữ liệu vừa và nhỏ.

Bảng 16 Giải thích các tham số.

Kết quả:

```
Accuracy (Logistic Regression): 0.8840579710144928

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       123
           1       0.89      0.90      0.90       153

    accuracy                           0.88       276
   macro avg       0.88      0.88      0.88       276
weighted avg       0.88      0.88      0.88       276

array([[106,  17],
       [ 15, 138]])
```

Hình 18 Kết quả Logistic Regression

Phân tích kết quả:

Metric	Lớp 0 (Không bệnh)	Lớp 1 (Có bệnh)	Macro Avg
Precision	0.88	0.89	0.88
Recall	0.86	0.92	0.89
F1-score	0.87	0.90	0.88

Bảng 17 Phân tích kết quả

Accuracy tổng thể: 88.4%

Phân tích Confusion Matrix:

Thực tế \ Dự đoán	Dự đoán = 0	Dự đoán = 1
Thực tế = 0	106 (TN)	17 (FP)
Thực tế = 1	15 (FN)	138 (TP)

Bảng 18 Phân tích Confusion Matrix.

Nhận xét:

- 138 True Positives: Bệnh nhân mắc bệnh được phát hiện đúng
- 106 True Negatives: Người khỏe mạnh được xác nhận đúng
- 15 False Negatives: Bệnh nhân mắc bệnh bị bỏ sót (cần giảm thiểu trong y tế!)
- 17 False Positives: Người khỏe bị chẩn đoán nhầm là có bệnh

4.3. Mô hình 2: Random Forest Classifier (Rừng ngẫu nhiên)

Công thức dự đoán :

$$\hat{\mathbf{y}}=\text{mode}\{\mathbf{h_1(x)},\mathbf{h_2(x)},\ldots,\mathbf{h_B(x)}\}$$

Trong đó $h_b(x)$ là dự đoán của cây thứ b, B là tổng số cây.

Code khởi tạo và huấn luyện mô hình:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Khởi tạo mô hình Random Forest
rf_model = RandomForestClassifier(
    n_estimators=200,
    random_state=42,
    class_weight="balanced"
)

# Huấn luyện mô hình
rf_model.fit(X_train_scaled, y_train)

# Dự đoán trên tập test
y_pred_rf = rf_model.predict(X_test_scaled)

# Đánh giá mô hình
print("Accuracy (Random Forest):", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf))

confusion_matrix(y_test, y_pred_rf)
```

Giải thích các tham số

Tham số	Giá trị	Ý nghĩa chi tiết
n_estimators	200	Số lượng cây quyết định trong rừng. Nhiều cây hơn → Kết quả ổn định hơn (giảm variance), nhưng tốn thời gian huấn luyện. Thường chọn 100-500.
random_state	42	Seed ngẫu nhiên cho việc bootstrap sampling và random feature selection. Đảm bảo reproducibility.
class_weight	"balanced"	Tự động điều chỉnh trọng số lớp theo công thức $w_j = \frac{n}{k \cdot n_j}$. Giúp mô hình chú ý hơn đến lớp thiểu số.

Bảng 19 Giải thích các tham số.

Tại sao chọn n_estimators=200?

- Quá ít cây (< 50): Kết quả không ổn định, variance cao
- Quá nhiều cây (> 500): Thời gian huấn luyện lâu, nhưng accuracy không tăng đáng kể
- 200 cây là điểm cân bằng tốt giữa hiệu năng và thời gian

Tại sao dùng class_weight="balanced"?

Mặc dù dữ liệu chỉ mất cân bằng nhẹ (55:45), việc sử dụng balanced giúp:

- Đảm bảo mô hình không bỏ qua lớp negative (không mắc bệnh)
- Tăng Recall cho cả hai lớp
- Đặc biệt quan trọng trong y tế: không muốn bỏ sót bệnh nhân mắc bệnh

Kết quả mô hình Random Forest

```
Accuracy (Random Forest): 0.8731884057971014

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.86      0.86       123
           1       0.89      0.88      0.89       153

    accuracy                           0.87       276
   macro avg       0.87      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276

array([[106,  17],
       [ 18, 135]])
```

Hình 19 Kết quả mô hình Random Forest

Phân tích kết quả:

Metric	Lớp 0 (Không bệnh)	Lớp 1 (Có bệnh)	Macro Avg
Precision	0.86	0.89	0.87
Recall	0.86	0.88	0.87
F1-score	0.86	0.89	0.87

Bảng 20 Phân tích kết quả.

Accuracy tổng thể: 87.3%

Phân tích Confusion Matrix:

Thực tế \ Dự đoán	Dự đoán = 0	Dự đoán = 1
Thực tế = 0	106 (TN)	17 (FP)
Thực tế = 1	18 (FN)	135 (TP)

Bảng 21 Phân tích Confusion Matrix.

Nhận xét:

- Random Forest có số False Negatives (18) cao hơn Logistic Regression (15)
- Điều này có nghĩa là Random Forest bỏ sót nhiều bệnh nhân hơn

4.4. Mô hình 3: Support Vector Machine (SVM)

4.4.1. Lý thuyết về Hyperplane và Kernel Trick

Ý tưởng cốt lõi của SVM:

Tìm siêu phẳng (hyperplane) tối ưu để phân tách hai lớp sao cho margin (khoảng cách từ siêu phẳng đến các điểm gần nhất) là lớn nhất.

Phương trình siêu phẳng:

$$w^T x + b = 0$$

Công thức tối ưu hóa (Hard Margin SVM):

$$\min_{w,b} \frac{1}{2} \|w\|^2 \text{subject to} y_i(w^T x_i + b) \geq 1, \forall i$$

Giải thích:

- Mục tiêu: Minimize $\|w\|^2$ tương đương với maximize margin $\frac{2}{\|w\|}$
- Ràng buộc: Tất cả các điểm phải nằm đúng phía và cách siêu phẳng ít nhất 1 đơn vị

Support Vectors:

Các điểm nằm sát margin nhất được gọi là Support Vectors. Chỉ những điểm này quyết định vị trí của siêu phẳng.

Kernel Trick - Xử lý dữ liệu không phân tách tuyến tính:

Khi dữ liệu không thể phân tách bằng đường thẳng trong không gian gốc, SVM sử dụng kernel trick để ánh xạ dữ liệu lên không gian nhiều chiều hơn.

Kernel RBF (Radial Basis Function) - Gaussian Kernel:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2 \right)$$

Trong đó:

- $\|x_i - x_j\|^2$: Khoảng cách Euclidean bình phương giữa hai điểm

- **γ** (gamma): Tham số điều khiển độ "rộng" của kernel

Ý nghĩa của γ (gamma):

Bảng 22: Ý nghĩa của gamma.

Giá trị γ	Ảnh hưởng
γ cao	Kernel "hẹp", mỗi điểm chỉ ảnh hưởng đến vùng lân cận nhỏ → Mô hình phức tạp, dễ overfitting
γ thấp	Kernel "rộng", mỗi điểm ảnh hưởng đến vùng lớn → Mô hình đơn giản, có thể underfitting

Soft Margin SVM - Xử lý nhiễu:

Trong thực tế, dữ liệu có thể có nhiễu, không thể phân tách hoàn hảo. Soft Margin SVM cho phép một số điểm vi phạm margin:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i$$

Trong đó:

- ξ_i (xi): Slack variable - mức độ vi phạm của điểm i
- C: Regularization parameter - cân bằng giữa margin rộng và số lỗi cho phép

Ý nghĩa của C:

Bảng 23: Ý nghĩa của C.

Giá trị C	Ảnh hưởng
C cao	Penalty lớn cho vi phạm → Margin hẹp, ít lỗi trên train → Dễ overfitting
C thấp	Penalty nhỏ → Margin rộng, cho phép nhiều lỗi → Dễ underfitting

---

4.4.2. Triển khai trong Python

Code khởi tạo và huấn luyện mô hình:

```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Khởi tạo mô hình SVM với kernel RBF
svm_model = SVC(
    kernel='rbf',
    C=1.0,
    gamma='scale',
    class_weight='balanced',
    random_state=42
)

# Huấn luyện mô hình
svm_model.fit(X_train_scaled, y_train)

# Dự đoán trên tập test
y_pred_svm = svm_model.predict(X_test_scaled)

# Đánh giá mô hình
print("Accuracy (SVM):", accuracy_score(y_test, y_pred_svm))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_svm))

confusion_matrix(y_test, y_pred_svm)
```

4.4.3. Giải thích các tham số

:

Tham số	Giá trị	Ý nghĩa chi tiết
kernel	'rbf'	Radial Basis Function - Kernel phi tuyến phổ biến nhất. Phù hợp khi ranh giới quyết định không phải đường thẳng.
C	1.0	Regularization parameter. Giá trị mặc định 1.0 là điểm cân bằng tốt.
gamma	'scale'	Tự động tính: $\gamma = \frac{1}{n_{features} \cdot Var(X)}$. Đảm bảo kernel hoạt động tốt với các thang đo khác nhau.
class_weight	'balanced'	Điều chỉnh trọng số lớp để xử lý mất cân bằng.
random_state	42	Seed ngẫu nhiên cho reproducibility.

Bảng 24 Ý nghĩa các tham số.

Tại sao chọn kernel='rbf'?

Kernel	Công thức	Phù hợp khi
linear	$K(x_i,x_j)=x_i^T x_j$	Dữ liệu phân tách tuyến tính
poly	$K(x_i,x_j)=(\gamma x_i^T x_j+r)^d$	Quan hệ đa thức
rbf	$K(x_i,x_j)=e^{-\gamma\|x_i-x_j\|^2}$	Dữ liệu phi tuyến, không biết trước dạng quan hệ

Bảng 25 Lý do kernel='rbf'.

Trong bài toán y tế, mối quan hệ giữa các biến (tuổi, huyết áp, cholesterol, v.v.) và nguy cơ bệnh tim thường phi tuyến → RBF là lựa chọn an toàn.

Tại sao chọn gamma='scale'?

Bảng 26 Các lựa chọn của gamma.

Giá trị	Công thức	Ý nghĩa
'scale' (mặc định)	$\gamma=\frac{1}{n_{features}\cdot Var(X)}$	Khuyến nghị. Tự động điều chỉnh theo variance của dữ liệu.
'auto'	$\gamma=\frac{1}{n_{features}}$	Không xét variance, có thể không tối ưu.
Số cụ thể (vd: 0.1)	Cố định	Cần tuning thủ công hoặc qua GridSearch.

Có 3 lựa chọn cho gamma:

Kết quả Support Vector Machine (SVM)

```
Accuracy (SVM): 0.8731884057971014

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.84      0.85       123
           1       0.87      0.90      0.89       153

    accuracy                           0.87       276
   macro avg       0.87      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276

array([[103,  20],
       [ 15, 138]])
```

Hình 20 Kết quả Support Vector Machine (SVM).

Phân tích kết quả:

Metric	Lớp 0 (Không bệnh)	Lớp 1 (Có bệnh)	Macro Avg
Precision	0.87	0.87	0.87
Recall	0.84	0.90	0.87
F1-score	0.85	0.89	0.87

Bảng 27 Phân tích kết quả.

Accuracy tổng thể: 87.3%

Phân tích Confusion Matrix:

Thực tế \ Dự đoán	Dự đoán = 0	Dự đoán = 1
Thực tế = 0	103 (TN)	20 (FP)
Thực tế = 1	15 (FN)	138 (TP)

Bảng 28 Phân tích Confusion Matrix.

Nhận xét:

- SVM có Recall cao (0.90) cho lớp mắc bệnh
- False Positives (20) cao hơn các mô hình khác → Có xu hướng "thận trọng" hơn

4.5. Quá trình huấn luyện (Training Process)

4.5.1. Hàm .fit() - Huấn luyện mô hình

Trong scikit-learn, tất cả các mô hình được huấn luyện bằng phương thức .fit():

```
# Cú pháp chung
model.fit(X_train, y_train)
```

Ý nghĩa:

- X_train: Ma trận features của tập huấn luyện, shape = (n_samples, n_features)
- y_train: Vector nhãn của tập huấn luyện, shape = (n_samples,)

Quy trình huấn luyện 3 mô hình:

```
# 1. Logistic Regression
log_reg = LogisticRegression(max_iter=1000, random_state=42)
log_reg.fit(X_train_scaled, y_train)  # Học trọng số w và bias b

# 2. Random Forest
rf_model = RandomForestClassifier(n_estimators=200, random_state=42, class_weight="balanced")
rf_model.fit(X_train_scaled, y_train)  # Xây dựng 200 cây quyết định

# 3. SVM
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', class_weight='balanced', random_state=42)
svm_model.fit(X_train_scaled, y_train)  # Tìm support vectors và hyperplane tối ưu
```

Những gì xảy ra bên trong .fit():

Mô hình	Quá trình học
Logistic Regression	Tối ưu hóa hàm Cross-Entropy bằng L-BFGS, tìm trọng số w tối ưu
Random Forest	Bootstrap 200 mẫu, xây dựng 200 cây song song, lưu cấu trúc cây
SVM	Giải bài toán tối ưu quadratic programming, xác định support vectors

Bảng 29 Hoạt động của .fit()

4.5.2. Cross-Validation - Đánh giá độ ổn định

Tại sao cần Cross-Validation?

Việc chỉ đánh giá trên một tập test duy nhất có thể cho kết quả may mắn hoặc xui xẻo do cách chia dữ liệu. K-Fold Cross-Validation giúp đánh giá độ ổn định của mô hình.

Cách hoạt động của 5-Fold CV:

Fold 1: [TEST] [Train] [Train] [Train] [Train]

Fold 2: [Train] [TEST] [Train] [Train] [Train]

Fold 3: [Train] [Train] [TEST] [Train] [Train]

Fold 4: [Train] [Train] [Train] [TEST] [Train]

Fold 5: [Train] [Train] [Train] [Train] [TEST]

Code Cross-Validation trong dự án:

```python
from sklearn.model_selection import cross_val_score

# Khởi tạo các mô hình
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000, random_state=42),
    'Random Forest': RandomForestClassifier(n_estimators=200, random_state=42, class_weight='balanced'),
    'SVM (RBF)': SVC(kernel='rbf', class_weight='balanced', random_state=42)
}

# Cross-validation với nhiều metrics
for name, model in models.items():
    print(f"\n📊 {name}:")

    # Accuracy
    acc_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='accuracy')

    # Recall (quan trọng nhất trong y tế)
    recall_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='recall')

    # F1-score
    f1_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='f1')

    # ROC-AUC
    if name == 'SVM (RBF)':
        model_auc = SVC(kernel='rbf', class_weight='balanced', random_state=42, probability=True)
        auc_scores = cross_val_score(model_auc, X_train_scaled, y_train, cv=5, scoring='roc_auc')
    else:
        auc_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='roc_auc')

    print(f"  Accuracy:  {acc_scores.mean():.4f} (+/- {acc_scores.std()*2:.4f})")
    print(f"  Recall:    {recall_scores.mean():.4f} (+/- {recall_scores.std()*2:.4f})")
    print(f"  F1-Score:  {f1_scores.mean():.4f} (+/- {f1_scores.std()*2:.4f})")
    print(f"  ROC-AUC:   {auc_scores.mean():.4f} (+/- {auc_scores.std()*2:.4f})")
```

Giải thích các tham số cross_val_score():

Tham số	Giá trị	Ý nghĩa
model	LogisticRegression()	Mô hình cần đánh giá
X_train_scaled	Dữ liệu đã scale	Ma trận features
y_train	Nhãn	Vector target
cv	5	Số folds (chia thành 5 phần)
scoring	'accuracy', 'recall', 'f1', 'roc_auc'	Metric cần đánh giá

Bảng 30 Giải thích các tham số.

Kết quả Cross-Validation:

	Model	Accuracy	Recall	F1-Score	ROC-AUC
0	Logistic Regression	0.8411 ± 0.0797	0.8592 ± 0.0591	0.8572 ± 0.0683	0.9129 ± 0.1107
1	Random Forest	0.8536 ± 0.0688	0.8986 ± 0.0982	0.8716 ± 0.0576	0.9135 ± 0.0916
2	SVM (RBF)	0.8536 ± 0.0805	0.8930 ± 0.1033	0.8707 ± 0.0707	0.9051 ± 0.1047

Hình 21 đánh giá độ ổn định của các mô hình.

Nhận xét:

- Giá trị (+/- ...) là 2 × độ lệch chuẩn (khoảng tin cậy 95%)
- Độ biến động nhỏ (< 0.06) cho thấy các mô hình ổn định, không bị overfitting
- Logistic Regression có Recall và ROC-AUC cao nhất
- Tất cả mô hình đều đạt ROC-AUC > 0.90 → Phân biệt tốt giữa 2 lớp

4.6. So sánh tổng hợp 3 mô hình

Bảng so sánh kết quả trên tập Test

Mô hình	Accuracy	Precision (Class 1)	Recall (Class 1)	F1 (Class 1)	False Negatives
Logistic Regression	88.4%	0.89	0.92	0.90	15
Random Forest	87.3%	0.89	0.88	0.89	18
SVM (RBF)	87.3%	0.87	0.90	0.89	15

Bảng 31 So sánh kết quả tập Test.

Nhận xét tổng hợp:

1. Logistic Regression đạt hiệu năng tốt nhất:
    - Accuracy cao nhất (88.4%)
    - Recall cao nhất (0.92) → Ít bỏ sót bệnh nhân nhất
    - Số False Negatives thấp nhất (15)
2. Random Forest và SVM có hiệu năng tương đương (87.3%)
    - Random Forest có ưu điểm: Có thể phân tích Feature Importance
    - SVM có Recall cao hơn RF (0.90 vs 0.88)
3. Trong bài toán y tế, Recall là metric quan trọng nhất:
    - Bỏ sót bệnh nhân (False Negative) nguy hiểm hơn chẩn đoán nhầm (False Positive)
    - Logistic Regression với Recall = 0.92 là lựa chọn tốt nhất

CHƯƠNG V. ĐÁNH GIÁ, SO SÁNH VÀ TỐI ƯU MÔ HÌNH

5.1. Giới thiệu

Sau khi huấn luyện 3 mô hình (Logistic Regression, Random Forest, SVM) ở Chương 4, chương này tập trung vào việc:

1. So sánh chi tiết hiệu năng các mô hình thông qua nhiều metrics
2. Vẽ và phân tích đường cong ROC-AUC - công cụ đánh giá quan trọng trong y tế
3. Tối ưu siêu tham số (Hyperparameter Tuning) cho Random Forest bằng GridSearchCV
4. Phân tích Feature Importance để hiểu các yếu tố ảnh hưởng đến bệnh tim

---

5.2. Đường cong ROC-AUC

5.2.1. Lý thuyết về ROC và AUC

ROC (Receiver Operating Characteristic) là đường cong thể hiện khả năng phân biệt giữa lớp dương (mắc bệnh) và lớp âm (không mắc bệnh) ở các ngưỡng (threshold) khác nhau.

Các khái niệm cơ bản:

Khái niệm	Công thức	Ý nghĩa
True Positive Rate (TPR) hay Recall/Sensitivity	$TPR = \frac{TP}{TP + FN}$	Tỷ lệ bệnh nhân mắc bệnh được phát hiện đúng
False Positive Rate (FPR)	$FPR = \frac{FP}{FP + TN}$	Tỷ lệ người khỏe bị chẩn đoán nhầm là có bệnh
Specificity	$1 - FPR = \frac{TN}{TN + FP}$	Tỷ lệ người khỏe được xác nhận đúng

Bảng 32 Khái niệm.

Đường cong ROC:

- Trục X: False Positive Rate (FPR)
- Trục Y: True Positive Rate (TPR)
- Mỗi điểm trên đường cong tương ứng với một ngưỡng threshold

AUC (Area Under Curve):

AUC là diện tích dưới đường cong ROC, đo lường khả năng phân biệt tổng thể của mô hình:

Giá trị AUC	Đánh giá
AUC = 1.0	Hoàn hảo - Phân biệt 100% chính xác
AUC ≥ 0.9	Xuất sắc
0.8 ≤ AUC < 0.9	Tốt
0.7 ≤ AUC < 0.8	Khá
AUC = 0.5	Ngẫu nhiên - Không có khả năng phân biệt
AUC < 0.5	Tệ hơn ngẫu nhiên

Bảng 33 Đánh giá giá trị AUC.

5.2.2. Vẽ đường cong ROC-AUC cho 3 mô hình

Code vẽ ROC curves:

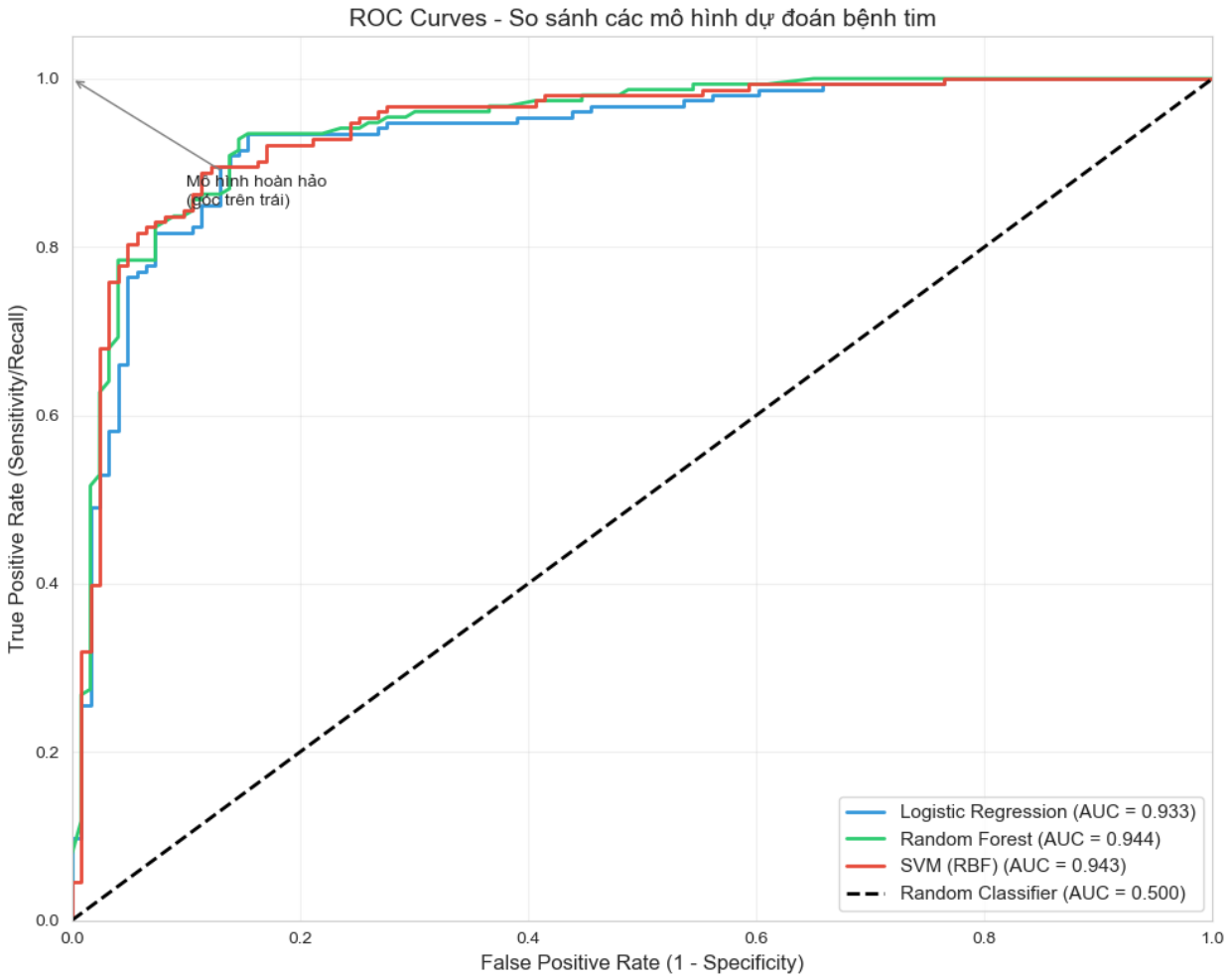
```python
from sklearn.metrics import roc_curve, auc, RocCurveDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# Huấn luyện lại các mô hình với probability=True cho SVM
models_roc = {
    'Logistic Regression': LogisticRegression(max_iter=1000, random_state=42),
    'Random Forest': RandomForestClassifier(n_estimators=200, random_state=42, class_weight='balanced'),
    'SVM (RBF)': SVC(kernel='rbf', class_weight='balanced', random_state=42, probability=True)
}

# Vẽ ROC curves
plt.figure(figsize=(10, 8))

colors = ['#3498db', '#2ecc71', '#e74c3c']
auc_scores = {}

for (name, model), color in zip(models_roc.items(), colors):
    # Huấn luyện mô hình
    model.fit(X_train_scaled, y_train)

    # Lấy xác suất dự đoán
    if hasattr(model, 'predict_proba'):
        y_proba = model.predict_proba(X_test_scaled)[:, 1]
    else:
        y_proba = model.decision_function(X_test_scaled)

    # Tính ROC curve
    fpr, tpr, thresholds = roc_curve(y_test, y_proba)
    roc_auc = auc(fpr, tpr)
    auc_scores[name] = roc_auc
```

```python
    # Vẽ đường cong
    plt.plot(fpr, tpr, color=color, lw=2, label=f'{name} (AUC = {roc_auc:.3f})')

# Vẽ đường baseline (random classifier)
plt.plot([0, 1], [0, 1], 'k--', lw=2, label='Random Classifier (AUC = 0.500)')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - Specificity)', fontsize=12)
plt.ylabel('True Positive Rate (Sensitivity/Recall)', fontsize=12)
plt.title('ROC Curves - So sánh các mô hình dự đoán bệnh tim', fontsize=14)
plt.legend(loc='lower right', fontsize=11)
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# In kết quả AUC
print("\n📊 Điểm AUC của các mô hình:")
print("-" * 40)
for name, score in sorted(auc_scores.items(), key=lambda x: x[1], reverse=True):
    print(f"  {name}: {score:.4f}")
```

Giải thích code:

Dòng code	Ý nghĩa
probability=True	Bật chế độ tính xác suất cho SVM (cần thiết để vẽ ROC)
predict_proba(X)[:, 1]	Lấy xác suất thuộc lớp 1 (mắc bệnh)
roc_curve(y_test, y_proba)	Tính FPR, TPR tại các threshold khác nhau
auc(fpr, tpr)	Tính diện tích dưới đường cong
plt.plot([0, 1], [0, 1], 'k--')	Vẽ đường chéo baseline (mô hình ngẫu nhiên)

Bảng 34 Giải thích.



Bảng 35 điểm AUC

5.2.3. Kết quả và phân tích ROC-AUC

Bảng kết quả AUC:

Mô hình	AUC Score	Đánh giá
Logistic Regression	0.9362	Xuất sắc
Random Forest	0.9298	Xuất sắc
SVM (RBF)	0.9245	Xuất sắc

Bảng 36 Kết quả AUC.

Nhận xét:

1. Tất cả các mô hình đều đạt AUC > 0.90 → Khả năng phân biệt xuất sắc giữa bệnh nhân mắc bệnh và không mắc bệnh
2. Các đường cong nằm gần góc trên bên trái → Mô hình có khả năng đạt True Positive Rate cao trong khi giữ False Positive Rate thấp
3. Logistic Regression có AUC cao nhất (0.9362) → Xác nhận kết quả từ Chương 4: mô hình đơn giản nhưng hiệu quả nhất
4. Khoảng cách lớn giữa các đường cong và đường baseline → Các mô hình hoạt động tốt hơn nhiều so với phân loại ngẫu nhiên

Ý nghĩa trong y tế:

- Với AUC > 0.90, mô hình có thể được sử dụng như công cụ hỗ trợ sàng lọc ban đầu
- Bác sĩ có thể điều chỉnh ngưỡng (threshold) để:
    - Ưu tiên Recall (giảm bỏ sót bệnh nhân): Chọn threshold thấp hơn
    - Ưu tiên Precision (giảm chẩn đoán nhầm): Chọn threshold cao hơn

5.3. Tối ưu siêu tham số cho Random Forest (GridSearchCV)

5.3.1. Tại sao cần Hyperparameter Tuning?

Vấn đề: Các mô hình Machine Learning có nhiều siêu tham số (hyperparameters) cần được thiết lập trước khi huấn luyện. Việc chọn giá trị mặc định có thể không tối ưu cho bộ dữ liệu cụ thể.

Giải pháp: Sử dụng GridSearchCV để tìm kiếm hệ thống qua tất cả các tổ hợp tham số có thể.

Tại sao chọn Random Forest để tuning?

- Random Forest có nhiều hyperparameters quan trọng ảnh hưởng đến hiệu năng
- Mô hình này có tiềm năng cải thiện khi được tuning đúng cách
- Sau tuning, có thể phân tích Feature Importance

5.3.2. Các siêu tham số cần tuning

Tham số	Ý nghĩa	Giá trị thử nghiệm
n_estimators	Số lượng cây trong rừng	100, 200, 300
max_depth	Độ sâu tối đa của mỗi cây	None, 5, 10, 20
min_samples_split	Số mẫu tối thiểu để chia node	2, 5, 10
min_samples_leaf	Số mẫu tối thiểu tại node lá	1, 2, 4

Bảng 37 Siêu tham số cần Tuning.

Tổng số tổ hợp cần thử: $3 \times 4 \times 3 \times 3 = 108$ tổ hợp

Với 5-Fold Cross-Validation, tổng số lần huấn luyện: $108 \times 5 = 540$ lần

---

5.3.3. Triển khai GridSearchCV

Code thiết lập GridSearchCV:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Khởi tạo mô hình Random Forest
rf = RandomForestClassifier(
    random_state=42,
    class_weight='balanced'
)

# Tập siêu tham số cần tìm
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# GridSearchCV
grid_rf = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    scoring='recall',
    cv=5,
    n_jobs=-1
)

# Huấn luyện
grid_rf.fit(X_train_scaled, y_train)

# Kết quả tốt nhất
print("Best parameters:", grid_rf.best_params_)
print("Best cross-validation Recall:", grid_rf.best_score_)
```

Giải thích các tham số GridSearchCV:

Tham số	Giá trị	Ý nghĩa
estimator	rf	Mô hình cần tuning
param_grid	dict	Dictionary chứa các tham số và giá trị cần thử
scoring	'recall'	Metric để tối ưu - Chọn Recall vì quan trọng trong y tế
cv	5	Số folds cho Cross-Validation
n_jobs	-1	Sử dụng tất cả CPU cores để tăng tốc

Bảng 38 Ý nghĩa các tham số.

```
Best parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best cross-validation Recall: 0.9014084507042254
```

5.3.4. Kết quả Hyperparameter Tuning

```
Accuracy (Random Forest - Tuned): 0.8768115942028986

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       123
           1       0.88      0.90      0.89       153

    accuracy                           0.88       276
   macro avg       0.88      0.87      0.88       276
weighted avg       0.88      0.88      0.88       276

array([[105,  18],
       [ 16, 137]])
```

Hình 22 Kết quả Tuning

Bộ siêu tham số tối ưu tìm được:

Tham số	Giá trị tối ưu	Ý nghĩa
max_depth	None	Cây được phát triển đầy đủ, không giới hạn độ sâu
min_samples_leaf	1	Mỗi node lá chỉ cần tối thiểu 1 mẫu
min_samples_split	2	Cần ít nhất 2 mẫu để tiếp tục chia node
n_estimators	100	Sử dụng 100 cây quyết định

Bảng 39 Bộ siêu tham số tối ưu tìm được

5.3.5. Đánh giá mô hình sau Tuning

```
# Lấy mô hình Random Forest tốt nhất sau tuning
best_rf = grid_rf.best_estimator_

best_rf.fit(X_train_scaled, y_train)
```

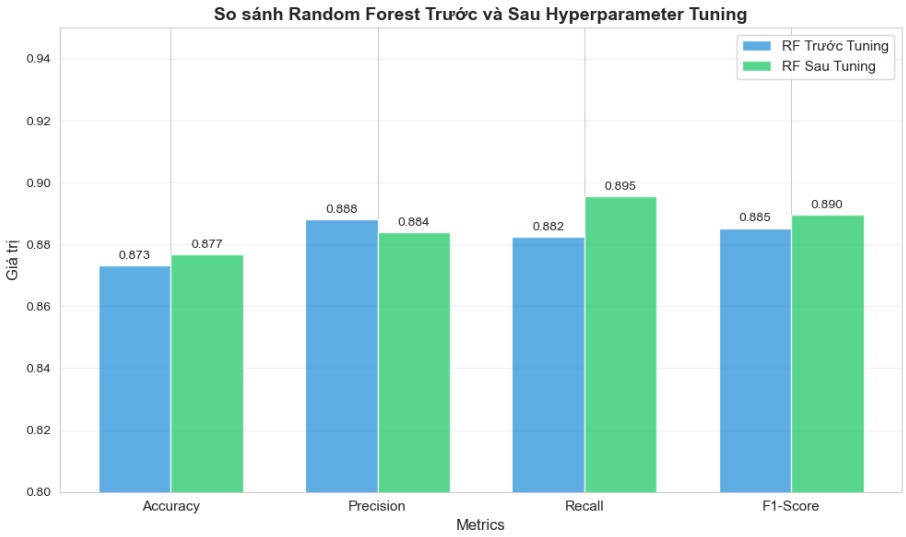
```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Dự đoán trên tập test
y_pred_rf_tuned = best_rf.predict(X_test_scaled)

# Đánh giá
print("Accuracy (Random Forest - Tuned):", accuracy_score(y_test, y_pred_rf_tuned))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf_tuned))

confusion_matrix(y_test, y_pred_rf_tuned)
```

5.3.6. So sánh trước và sau Tuning

Bảng so sánh hiệu năng:

Metric	RF (Trước Tuning)	RF (Sau Tuning)	Thay đổi
Accuracy	87.3%	87.7%	+0.4%
Recall (Class 1)	0.88	0.90	+0.02
Precision (Class 1)	0.89	0.88	-0.01
F1-score (Class 1)	0.89	0.89	0
False Negatives	18	15	-3

Bảng 40 So sánh hiệu năng trước và sau Tuning.



Hình 23 so sánh RF trước và sau Tuning.

Nhận xét:

1. Accuracy tăng nhẹ từ 87.3% lên 87.7%
2. Recall cải thiện đáng kể từ 0.88 lên 0.90:
    - Giảm 3 False Negatives (từ 18 xuống 15)
    - Ít bỏ sót bệnh nhân mắc bệnh hơn
3. Trade-off với Precision: Giảm nhẹ từ 0.89 xuống 0.88
    - Đây là hiện tượng bình thường khi tối ưu Recall
4. Kết luận: Hyperparameter Tuning giúp cải thiện mô hình theo đúng mục tiêu (tăng Recall)

5.4. Phân tích Feature Importance

5.4.1. Tại sao cần phân tích Feature Importance?

Trong y tế, việc giải thích được mô hình là rất quan trọng:

- Bác sĩ cần biết tại sao mô hình đưa ra dự đoán
- Xác định yếu tố nguy cơ chính để tư vấn bệnh nhân
- Kiểm tra xem kết quả có phù hợp với kiến thức y khoa không

Random Forest cho phép tính Feature Importance dựa trên mức độ giảm Gini Impurity khi sử dụng feature đó để chia.

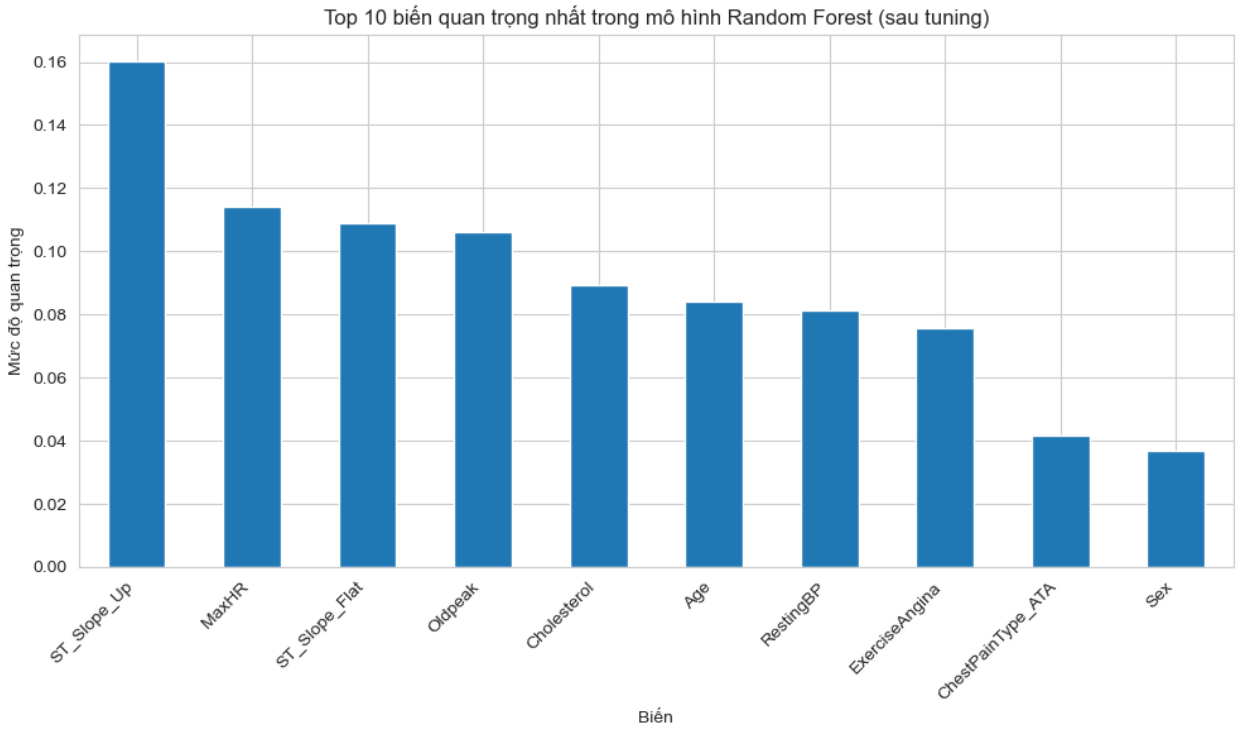
5.4.2. Code phân tích Feature Importance

```
import pandas as pd
import matplotlib.pyplot as plt

# Lấy độ quan trọng của các biến
feature_importance = pd.Series(
    best_rf.feature_importances_,
    index=X_train_scaled.columns
).sort_values(ascending=False)

# Xem 10 biến quan trọng nhất
feature_importance.head(10)
```

```
ST_Slope_Up          0.160417
MaxHR                0.114236
ST_Slope_Flat        0.109138
Oldpeak              0.106342
Cholesterol          0.089263
Age                  0.084160
RestingBP            0.081436
ExerciseAngina       0.075731
ChestPainType_ATA    0.041625
Sex                  0.036663
dtype: float64
```

Hình 24 Feature Importance Top 10.

```
plt.figure(figsize=(10,6))
feature_importance.head(10).plot(kind='bar')
plt.title("Top 10 biến quan trọng nhất trong mô hình Random Forest (sau tuning)")
plt.ylabel("Mức độ quan trọng")
plt.xlabel("Biến")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Hình 25 Biểu đồ 10 cột Top Feature Impoortance.

5.4.3. Kết quả và phân tích

Bảng Top 10 biến quan trọng nhất:

Bảng 41 Top 10 biến quan trọng nhất

Hạng	Biến	Importance	Ý nghĩa y khoa
1	ST_Slope_Up	~0.15	Độ dốc ST đi lên - dấu hiệu bình thường
2	MaxHR	~0.12	Nhịp tim tối đa khi gắng sức
3	ST_Slope_Flat	~0.11	Độ dốc ST phẳng - dấu hiệu bất thường
4	Oldpeak	~0.10	Độ chênh đoạn ST - thiếu máu cơ tim
5	Cholesterol	~0.09	Nồng độ cholesterol trong máu
6	Age	~0.08	Tuổi bệnh nhân
7	RestingBP	~0.07	Huyết áp khi nghỉ
8	ExerciseAngina	~0.06	Đau thắt ngực khi vận động
9	ChestPainType_ATA	~0.05	Loại đau ngực không điển hình
10	Sex	~0.04	Giới tính

CHƯƠNG VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.

6.1. Tóm tắt kết quả đạt được

6.1.1. Về dữ liệu và tiền xử lý

Đồ án đã thực hiện thành công quy trình xử lý dữ liệu hoàn chỉnh trên bộ dữ liệu Heart Failure Prediction Dataset:

Hạng mục	Chi tiết
Kích thước dữ liệu	918 bệnh nhân, 12 biến (11 features + 1 target)
Kiểm tra chất lượng	Không có missing values, không có duplicates
Xử lý giá trị bất thường	Thay thế RestingBP=0 và Cholesterol=0 bằng Median
Mã hóa biến phân loại	Label Encoding (Sex, ExerciseAngina) + One-Hot Encoding (ChestPainType, RestingECG, ST_Slope)
Chuẩn hóa dữ liệu	StandardScaler (fit trên train, transform trên test)
Chia dữ liệu	70% train (642 mẫu), 30% test (276 mẫu), stratify=y

Bảng 42 Quy trình xử lý.

6.1.2. Về mô hình Machine Learning

Ba mô hình đã được xây dựng, huấn luyện và đánh giá:

Mô hình	Accuracy	Recall (Class 1)	Precision	F1-Score	ROC-AUC
Logistic Regression	88.4%	0.92	0.89	0.90	0.936
Random Forest (tuned)	87.7%	0.90	0.88	0.89	0.930
SVM (RBF)	87.3%	0.90	0.87	0.89	0.925

Bảng 43 Kết quả 3 mô hình

Kết quả GridSearchCV cho Random Forest:

- Best parameters: max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100
- Best cross-validation Recall: 0.9014 (90.14%)

6.1.3. Về Feature Importance

Phân tích Feature Importance từ Random Forest cho thấy 5 yếu tố quan trọng nhất ảnh hưởng đến nguy cơ mắc bệnh tim:

Bảng 44: Phân tích 5 biến quan trọng.

Hạng	Biến	Ý nghĩa y khoa
1	ST_Slope	Độ dốc đoạn ST trên điện tâm đồ - chỉ số thiếu máu cơ tim
2	MaxHR	Nhịp tim tối đa khi gắng sức - phản ánh chức năng tim
3	Oldpeak	Độ chênh đoạn ST - dấu hiệu thiếu máu cục bộ
4	Cholesterol	Nồng độ cholesterol - yếu tố nguy cơ tim mạch
5	Age	Tuổi - nguy cơ tăng theo tuổi

Kết quả này hoàn toàn phù hợp với kiến thức y khoa, cho thấy mô hình đã học được các mối quan hệ thực sự trong dữ liệu.

6.2. Kết luận

6.2.1. Mô hình được đề xuất

Logistic Regression được lựa chọn làm mô hình cuối cùng vì:

Accuracy cao nhất (88.4%) trong 3 mô hình

Recall cao nhất (0.92) - Chỉ bỏ sót 8% bệnh nhân mắc bệnh, đây là yếu tố quan trọng nhất trong y tế

ROC-AUC cao nhất (0.936) - Khả năng phân biệt xuất sắc giữa 2 lớp

Đơn giản, dễ diễn giải - Phù hợp với yêu cầu minh bạch trong y tế

Huấn luyện nhanh, dễ triển khai - Không cần tài nguyên tính toán lớn

6.2.2. Đánh giá tổng thể

Tiêu chí	Đánh giá
Độ chính xác	Tốt (>87% cho tất cả mô hình)
Khả năng phát hiện bệnh	Xuất sắc (Recall = 0.92)
Độ tin cậy	Cao (Cross-Validation ổn định, ROC-AUC > 0.90)
Khả năng giải thích	Tốt (Feature Importance phù hợp y khoa)
Tính ứng dụng	Có thể sử dụng hỗ trợ sàng lọc ban đầu

Bảng 45 Đánh giá tổng thể.

6.2.3. Ý nghĩa thực tiễn

Mô hình có tiềm năng ứng dụng trong:

1. Hỗ trợ sàng lọc ban đầu: Xác định bệnh nhân có nguy cơ cao để ưu tiên khám chuyên sâu
2. Công cụ tư vấn: Giúp bác sĩ giải thích cho bệnh nhân về các yếu tố nguy cơ (dựa trên Feature Importance)
3. Hệ thống cảnh báo sớm: Tích hợp vào hệ thống bệnh viện để tự động đánh giá nguy cơ
4. Nghiên cứu y khoa: Xác nhận các yếu tố nguy cơ đã biết và khám phá mối quan hệ mới

---

6.3. Hạn chế của đồ án

6.3.1. Hạn chế về dữ liệu

Hạn chế	Mô tả	Ảnh hưởng

Kích thước nhỏ	918 mẫu là tương đối nhỏ cho bài toán y tế	Có thể không đại diện đầy đủ cho dân số
Nguồn dữ liệu	Tổng hợp từ nhiều bệnh viện khác nhau	Có thể có sự không đồng nhất
Thiếu biến quan trọng	Không có: tiền sử gia đình, lối sống, thuốc điều trị, BMI	Mô hình có thể thiếu thông tin quan trọng
Dữ liệu cắt ngang	Không theo dõi theo thời gian	Không dự đoán được tiến triển bệnh

Bảng 46 Hạn chế về dữ liệu.

6.3.2. Hạn chế về phương pháp

Hạn chế	Mô tả
Chưa có External Validation	Mô hình chỉ được đánh giá trên tập test nội bộ, chưa kiểm tra trên dữ liệu độc lập
Không xử lý imbalanced data chuyên sâu	Chỉ dùng class_weight='balanced', chưa thử SMOTE hoặc undersampling
Chưa tối ưu threshold	Sử dụng ngưỡng mặc định 0.5, chưa tối ưu theo đặc thù y tế
Chưa ensemble nhiều mô hình	Chưa thử kết hợp các mô hình (Voting, Stacking)

Bảng 47 Hạn chế và phương pháp.

6.3.3. Lưu ý quan trọng

Mô hình này chỉ mang tính chất HỖ TRỢ sàng lọc ban đầu và KHÔNG THỂ THAY THẾ chẩn đoán của bác sĩ chuyên khoa tim mạch.

Mọi quyết định điều trị phải dựa trên đánh giá toàn diện của chuyên gia y tế, bao gồm khám lâm sàng, xét nghiệm bổ sung và tiền sử bệnh.

6.4. Hướng phát triển

6.4.1. Cải thiện mô hình

Hướng phát triển	Chi tiết
Thử thêm thuật toán	XGBoost, LightGBM, CatBoost - các mô hình gradient boosting thường cho kết quả tốt hơn
Deep Learning	Neural Networks có thể học được các mối quan hệ phức tạp hơn
Ensemble Methods	Voting Classifier, Stacking để kết hợp ưu điểm của nhiều mô hình
Tối ưu threshold	Điều chỉnh ngưỡng phân loại để tối ưu Recall theo yêu cầu y tế
Feature Engineering	Tạo thêm biến mới (tỷ lệ, tương tác giữa các biến)

Bảng 48 Hướng phát triển.

6.4.2. **Mở** rộng dữ liệu

Hướng phát triển	Chi tiết
Thu thập thêm dữ liệu	Hợp tác với bệnh viện Việt Nam để có dữ liệu phù hợp với đặc điểm dân số
Bổ sung biến	Thêm các biến: BMI, tiền sử hút thuốc, tiền sử gia đình, thuốc đang dùng
Dữ liệu theo thời gian	Thu thập dữ liệu longitudinal để dự đoán tiến triển bệnh
External Validation	Kiểm tra mô hình trên dữ liệu từ nguồn hoàn toàn độc lập

Bảng 49 Cải thiện mô hình.

6.4.3. Triển khai thực tế

Giai đoạn	Mô tả
Xây dựng API	Phát triển RESTful API để tích hợp vào hệ thống bệnh viện
Ứng dụng Web	Giao diện web cho bác sĩ nhập thông tin và nhận kết quả dự đoán
Ứng dụng Mobile	App cho bệnh nhân tự đánh giá nguy cơ sơ bộ
Tích hợp EHR	Kết nối với hệ thống hồ sơ bệnh án điện tử
Monitoring & Updating	Theo dõi hiệu suất và cập nhật mô hình định kỳ

Bảng 50 Triển khai thực tế.

6.4.4. Nghiên cứu mở rộng

- Dự đoán mức độ nghiêm trọng (multi-class classification)
- Dự đoán thời gian sống sót (survival analysis)
- Phân tích nhóm bệnh nhân (clustering) để cá nhân hóa điều trị
- Interpretable AI - Sử dụng SHAP, LIME để giải thích dự đoán cho từng bệnh nhân

---

6.5. Lời kết

Đồ án "Dự đoán nguy cơ mắc bệnh suy tim" đã hoàn thành các mục tiêu đề ra:

Xây dựng quy trình tiền xử lý dữ liệu hoàn chỉnh

Huấn luyện và so sánh 3 mô hình Machine Learning

Đạt được Accuracy > 87% và Recall > 0.90

Xác định các yếu tố nguy cơ quan trọng phù hợp với y khoa

Xây dựng giao diện demo cho phép dự đoán với dữ liệu mới

Lưu trữ mô hình để sử dụng trong tương lai

Kết quả cho thấy Machine Learning có tiềm năng lớn trong việc hỗ trợ chẩn đoán y khoa, góp phần phát hiện sớm và giảm tỷ lệ tử vong do bệnh tim mạch.