ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ – VẬT LÝ KỸ THUẬT

BÁO CÁO ĐỒ ÁN MÔN HỌC MÁY HỌC
PET 10102

Đề tài:

DỰ ĐOÁN TỶ LỆ SỐNG SÓT CỦA BỆNH NHÂN SAU KHI NHẬP VIỆN

Nhóm thực hiện: Nhóm 4
Lớp: 22CVD1

GVHD: ThS. Huỳnh Quốc Việt

NCS: Nguyễn Thị Như Quỳnh

TP. HỒ CHÍ MINH – 2025

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ – VẬT LÝ KỸ THUẬT

BÁO CÁO ĐỒ ÁN MÔN HỌC MÁY HỌC
PET 10102

Đề tài:

DỰ ĐOÁN TỶ LỆ SỐNG SÓT CỦA BỆNH NHÂN SAU KHI NHẬP VIỆN

Nhóm thực hiện: Nhóm 4
Lớp: 22CVD1

GVHD: ThS. Huỳnh Quốc Việt

NCS: Nguyễn Thị Như Quỳnh

TP. HỒ CHÍ MINH – 2025

Lời Cảm Ơn

Em xin gởi những lời cảm ơn chân thành nhất tới những người thân yêu, những thầy cô, những người bạn đã giúp đỡ, động viên, theo dõi cũng như ủng hộ em hoàn thành tốt đồ án này.

Tp. Hồ Chí Minh, tháng 04 năm 2025

Danh sách các thành viên nhóm

STT	MSSV	Họ và tên	Ghi chú
1	22130080	Nguyễn Gia Khiêm	Nhóm Trưởng
2	22130005	Nguyễn Thuỵ Vân Anh	Thành viên
3	22130054	Nguyễn Thanh Hoà	Thành viên
4	22130131	Nguyễn Quỳnh Nhi	Thành viên
5	22130188	Lê Hoàng Trí Tín	Thành viên

MỤC LỤC

BẢNG CÁC TỪ VIẾT TẮT

KNN	K-Nearest Neighbor
XGBoost	Extreme Gradient Boosting
MFCC	Mel – Frequency Cepstral Coefficient
PoV	Probability of Voicing
LSP	Linear Prediction Spectrum
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
SVM	Support Vector Machine

DANH SÁCH CÁC HÌNH ẢNH

LỜI MỞ ĐẦU

Trong lĩnh vực y tế, việc theo dõi và dự đoán quá trình phục hồi của bệnh nhân sau phẫu thuật là một yếu tố quan trọng, ảnh hưởng trực tiếp đến hiệu quả điều trị. Các phương pháp truyền thống thường phụ thuộc vào kinh nghiệm và cảm quan của bác sĩ, tuy nhiên chúng có thể không phản ánh đầy đủ sự phức tạp của từng trường hợp. Do đó, sử dụng máy học giúp cải thiện độ chính xác và đưa ra các dự đoán dựa trên dữ liệu lớn, giúp tối ưu hóa quá trình điều trị.

Nội dung của đề tài "Sử dụng máy học để dự đoán khả năng phục hồi sau phẫu thuật" tập trung vào việc thu thập và phân tích dữ liệu y tế liên quan đến các yếu tố ảnh hưởng đến quá trình phục hồi như tuổi tác, tình trạng sức khỏe trước khi phẫu thuật, loại phẫu thuật, các biến chứng sau phẫu thuật, và các chỉ số sinh lý của bệnh nhân. Hệ thống dự đoán giúp bác sĩ và nhân viên y tế có thể đưa ra quyết định chính xác hơn về quá trình điều trị, từ đó cải thiện chất lượng chăm sóc và giảm thiểu rủi ro cho bệnh nhân. Dựa trên các dữ liệu này, nhóm đã xây dựng các mô hình máy học để dự đoán nguy cơ tử vong của bệnh nhân sau khi nhập viện (ICU) dựa trên các chỉ số sinh lý, bệnh nền và đánh giá từ hệ thống chấm điểm APACHE.

Các công việc đã thực hiện trong đồ án bao gồm: thu thập dữ liệu, xử lý dữ liệu, xây dựng và huấn luyện các mô hình máy học (KNN, XGBoost,v.v…), đánh giá độ chính xác của mô hình, và ứng dụng mô hình vào thực tế.

Nhờ sự kết hợp giữa công nghệ và các thuật toán máy học mạnh mẽ, nhóm đã bước đầu đạt được những kết quả khả quan trong việc xây dựng mô hình máy học có khả năng dự đoán sức phục hồi của bệnh nhân với độ chính xác cao, giúp hỗ trợ các quyết định điều trị và chăm sóc bệnh nhân, mở ra những tiềm năng ứng dụng thực tiễn trong các lĩnh vực y học và công nghệ hỗ trợ y tế hiện đại.

CHƯƠNG 1. DỰ ĐOÁN TỶ LỆ SỐNG SÓT CỦA BỆNH NHÂN SAU KHI NHẬP VIỆN

1.1 Mục tiêu nghiên cứu

Dự án này tập trung vào việc xây dựng mô hình học máy để dự đoán nguy cơ tử vong của bệnh nhân sau khi nhập viện vào **Phòng chăm sóc đặc biệt (ICU)**. Mô hình sẽ sử dụng ba nhóm yếu tố chính để đánh giá nguy cơ tử vong: **chỉ số sinh lý**, **bệnh nền**, và **đánh giá từ hệ thống chấm điểm APACHE**.

1. **Chỉ số sinh lý**: Các thông số như nhịp tim, huyết áp, nhiệt độ cơ thể, nồng độ oxy trong máu (SpO2), và các chỉ số khác được ghi nhận trong quá trình theo dõi sức khỏe bệnh nhân.
2. **Bệnh nền**: Những bệnh lý nền như tiểu đường, bệnh tim mạch, hoặc các tình trạng mãn tính khác có thể làm tăng nguy cơ tử vong và ảnh hưởng đến tiên lượng của bệnh nhân.
3. **Đánh giá từ hệ thống APACHE**: Hệ thống **APACHE** giúp đánh giá mức độ nghiêm trọng của tình trạng bệnh nhân dựa trên các chỉ số sinh lý và bệnh lý, từ đó xác định nguy cơ tử vong trong ICU.

Mục tiêu của dự án là giúp bác sĩ và nhân viên y tế phân loại sớm bệnh nhân có nguy cơ tử vong cao, từ đó có thể can thiệp kịp thời và đưa ra các biện pháp điều trị phù hợp. Điều này không chỉ giúp giảm **tỷ lệ tử vong**, mà còn **tối ưu hóa nguồn lực ICU** và nâng cao hiệu quả chăm sóc y tế. Mô hình học máy sẽ cung cấp một công cụ dự đoán chính xác và dễ hiểu, có thể tích hợp vào quy trình làm việc của bác sĩ, giúp cải thiện chất lượng điều trị cho bệnh nhân.

1.2 Ý nghĩa thực tiễn

1. **Hỗ trợ bác sĩ trong việc phân loại sớm bệnh nhân có nguy cơ cao**: Mô hình học máy sẽ giúp bác sĩ nhận diện nhanh chóng các bệnh nhân có nguy cơ tử vong cao ngay từ khi nhập viện ICU. Điều này giúp họ đưa ra các quyết định kịp thời, từ đó triển khai phương án điều trị phù hợp và hiệu quả.

2. **Giảm tỷ lệ tử vong, tối ưu nguồn lực ICU**: Việc phát hiện và phân loại bệnh nhân nguy cơ cao sớm sẽ giúp giảm thiểu tỷ lệ tử vong bằng cách can thiệp kịp thời. Đồng thời, việc tối ưu hóa việc sử dụng nguồn lực ICU sẽ giảm bớt sự quá tải và đảm bảo rằng các bệnh nhân cần chăm sóc đặc biệt sẽ nhận được sự chú ý đầy đủ.
3. **Cung cấp công cụ AI giải thích được, phù hợp với môi trường y tế**: Mô hình AI sẽ không chỉ dự đoán nguy cơ tử vong mà còn cung cấp các giải thích rõ ràng về các yếu tố dẫn đến dự đoán đó. Điều này giúp bác sĩ hiểu và tin tưởng vào kết quả mô hình, từ đó dễ dàng áp dụng trong công tác lâm sàng, phù hợp với môi trường y tế.

1.3. Dữ liệu sử dụng

Nguồn: Dữ liệu được lấy từ Kaggle, với tập tin có tên **dataset.csv**, chứa thông tin về bệnh nhân ICU.

Số lượng: Tập dữ liệu có **hơn 91,000 dòng** và **hơn 180 thuộc tính**, bao gồm các chỉ số sinh lý, thông tin về bệnh nền, và các đánh giá từ hệ thống APACHE.

Biến mục tiêu: Biến mục tiêu trong tập dữ liệu là **hospital_death**, với giá trị:

- **0**: Bệnh nhân sống sót.
- **1**: Bệnh nhân tử vong.

Dạng dữ liệu: Tập dữ liệu chứa các biến có dạng:

- **Định lượng**: Các chỉ số như tuổi, nhịp tim, huyết áp, nhiệt độ cơ thể, nồng độ oxy trong máu (SpO2) và các chỉ số sinh lý khác.
- **Phân loại**: Các thuộc tính như giới tính, bệnh nền, và các yếu tố khác như tình trạng sức khỏe ban đầu, loại điều trị.

1.3.1. Cấu trúc dữ liệu

```python
import pandas as pd

# Đọc dữ liệu từ file CSV
data = pd.read_csv("/content/dataset.csv", index_col="patient_id" )

# Hiển thị vài dòng đầu của dataframe
data.head()
```

```
df.head()
```

	encounter_id	patient_id	hospital_id	age	bmi	elective_surgery	ethnicity	gender	height	icu_admit_source	...
0	66154	25312	118	68.0	22.73	0	Caucasian	M	180.3	Floor	...
1	114252	59342	81	77.0	27.42	0	Caucasian	F	160.0	Floor	...
2	119783	50777	118	25.0	31.95	0	Caucasian	F	172.7	Accident & Emergency	...
3	79267	46918	118	81.0	22.64	1	Caucasian	F	165.1	Operating Room / Recovery	...
4	92056	34377	33	19.0	NaN	0	Caucasian	M	188.0	Accident & Emergency	...

5 rows × 85 columns

```
df.head()
```

leukemia	lymphoma	solid_tumor_with_metastasis	apache_3j_bodysystem	apache_2_bodysystem	Unnamed: 83	hospital_death
0.0	0.0	0.0	Sepsis	Cardiovascular	NaN	0
0.0	0.0	0.0	Respiratory	Respiratory	NaN	0
0.0	0.0	0.0	Metabolic	Metabolic	NaN	0
0.0	0.0	0.0	Cardiovascular	Cardiovascular	NaN	0
0.0	0.0	0.0	Trauma	Trauma	NaN	0

1.3.2. Loại dữ liệu và kiểu dữ liệu:

Bộ dữ liệu chứa tổng cộng 85 cột, phân loại cụ thể như sau:

Bảng 1.3.2: Phân loại dữ liệu

Kiểu dữ liệu	Số lượng cột	Ví dụ các biến tiêu biểu
Số thực (float64)	71	age, bmi, temp_apache, heart_rate_apache, apache_4a_hospital_death_prob, d1_glucose_max
Số nguyên (int64)	7	encounter_id, patient_id, hospital_id, elective_surgery, icu_id, apache_post_operative, hospital_death
Phân loại (object)	7	ethnicity, gender, icu_admit_source, icu_stay_type, icu_type, apache_3j_bodysystem, apache_2_bodysystem

- Các biến kiểu **float64** và **int64** thường liên quan trực tiếp đến các chỉ số y khoa đo được (ví dụ: nhiệt độ cơ thể, huyết áp, BMI, nhịp tim…).
- Các biến kiểu **object** phản ánh đặc điểm phân loại bệnh nhân (giới tính, dân tộc, loại ICU nhập viện, nguồn nhập viện, và nhóm bệnh lý liên quan đến APACHE).

1.3.3. Các nhóm đặc trưng chính

Nhóm đặc trưng trong bộ dữ liệu có thể chia thành các loại chính sau đây:

- **Thông tin cá nhân và nhân khẩu học**:
  - age (tuổi)
  - gender (giới tính)
  - ethnicity (dân tộc)
  - height, weight, bmi (chỉ số khối cơ thể)
- **Thông tin về ICU (Intensive Care Unit)**:
  - icu_admit_source (nguồn nhập ICU: cấp cứu, phẫu thuật...)
  - icu_stay_type (loại lưu trú ICU: cấp cứu, theo dõi)
  - icu_type (loại ICU: tim mạch, phẫu thuật thần kinh...)
  - pre_icu_los_days (số ngày nằm viện trước khi nhập ICU)
- **Các chỉ số APACHE (Acute Physiology and Chronic Health Evaluation)**:
  - apache_4a_hospital_death_prob, apache_4a_icu_death_prob (xác suất tử vong dự đoán)
  - apache_2_diagnosis, apache_3j_diagnosis (chẩn đoán theo phân loại APACHE)
- **Các chỉ số sinh lý (trong 24h đầu)**:
  - Huyết áp tâm thu và tâm trương (sysbp, diasbp)
  - Nhịp tim (heartrate)
  - Nhiệt độ cơ thể (temp)
  - Tần số thở (resprate)
  - SpO2 (độ bão hòa oxy máu)
- **Các chỉ số GCS (Glasgow Coma Scale)**:
  - gcs_eyes_apache, gcs_motor_apache, gcs_verbal_apache

    - gcs_unable_apache
- **Các chỉ số huyết học và sinh hóa**:
    - Đường máu (d1_glucose_max/min)
    - Kali máu (d1_potassium_max/min)
- **Thông tin về bệnh lý nền và bệnh lý đặc biệt**:
    - diabetes_mellitus (đái tháo đường)
    - cirrhosis (xơ gan)
    - hepatic_failure (suy gan)
    - immunosuppression (suy giảm miễn dịch)
    - solid_tumor_with_metastasis (u ác tính di căn)

1.3.4. Các biến quan trọng và ý nghĩa

Các đặc trưng nhân khẩu học

- **Tuổi (age):**
    - Trung bình: **62.3 tuổi**
    - Độ tuổi phổ biến (50% mẫu): từ **52–75 tuổi**
    - Biến rất quan trọng: Tuổi càng cao, nguy cơ tử vong càng lớn.
- **Giới tính (gender):**
    - Nam (M): **53.95%**
    - Nữ (F): **46.05%**
    - Có thể ảnh hưởng đến tỷ lệ tử vong do các yếu tố bệnh lý nền và khả năng chống chịu bệnh khác nhau.
- **Chỉ số BMI (bmi):**
    - Trung bình: **29.2** (hơi cao, nằm trong vùng thừa cân nhẹ)
    - Khoảng giữa phổ biến: từ **23.6 đến 32.9**
    - BMI quá thấp hoặc quá cao đều liên quan đến nguy cơ tử vong cao hơn.

Các chỉ số tiên lượng APACHE

- **apache_4a_hospital_death_prob (Xác suất tử vong dự đoán theo APACHE IV):**

- Trung bình: **8.68%**
- Giá trị từ **0.02% đến 13%** ở đa số mẫu (25%-75%)
- Biến này phản ánh trực tiếp và rất mạnh mẽ nguy cơ tử vong dựa trên các yếu tố lâm sàng trong ICU.

Các chỉ số sinh lý

- **Nhịp tim trung bình (heart_rate_apache):**
  - Trung bình: **99.7 lần/phút** (cao, phản ánh tình trạng cấp tính của nhiều bệnh nhân)
  - Khoảng phổ biến: **86–120 lần/phút**
- **Áp lực động mạch trung bình (map_apache):**
  - Trung bình: **88.0 mmHg**
  - Khoảng phổ biến: **54–125 mmHg**
  - Chỉ số MAP thấp hoặc cao bất thường đều liên quan đến nguy cơ tử vong tăng lên.
- **Nhiệt độ cơ thể (temp_apache):**
  - Trung bình: **36.4°C**
  - Đa số bệnh nhân dao động từ **36.2°C đến 36.7°C**
  - Nhiệt độ bất thường có thể báo hiệu nhiễm trùng hoặc các vấn đề khác trong ICU.

Các chỉ số huyết học & sinh hóa

- **Glucose máu ngày đầu (d1_glucose_max):**
  - Trung bình: **174.6 mg/dL** (cao hơn mức bình thường, phản ánh stress cấp tính hoặc tiểu đường)
  - Khoảng phổ biến: **117–201 mg/dL**
  - Glucose bất thường liên quan trực tiếp đến nguy cơ tử vong cao hơn.
- **Kali máu ngày đầu (d1_potassium_max):**
  - Trung bình: **4.25 mEq/L**
  - Khoảng phổ biến: **3.8–4.6 mEq/L**

- Kali cao hoặc thấp bất thường đều là các chỉ dấu quan trọng, phản ánh tình trạng sinh lý nguy hiểm.

Điểm Glasgow Coma Scale (GCS)

- **GCS mở mắt (gcs_eyes_apache):**
  - Trung bình: **3.47** (thang điểm tối đa là 4)
  - Phản ánh mức độ tỉnh táo và nhận thức của bệnh nhân.
- **GCS vận động (gcs_motor_apache):**
  - Trung bình: **5.47** (thang điểm tối đa là 6)
  - Quan trọng trong đánh giá tổn thương thần kinh và tiên lượng sống sót.
- **GCS lời nói (gcs_verbal_apache):**
  - Trung bình: **3.99** (thang điểm tối đa là 5)
  - Điểm thấp phản ánh khả năng giao tiếp và mức độ tỉnh táo giảm, nguy cơ tử vong cao hơn.

Các bệnh lý nền quan trọng

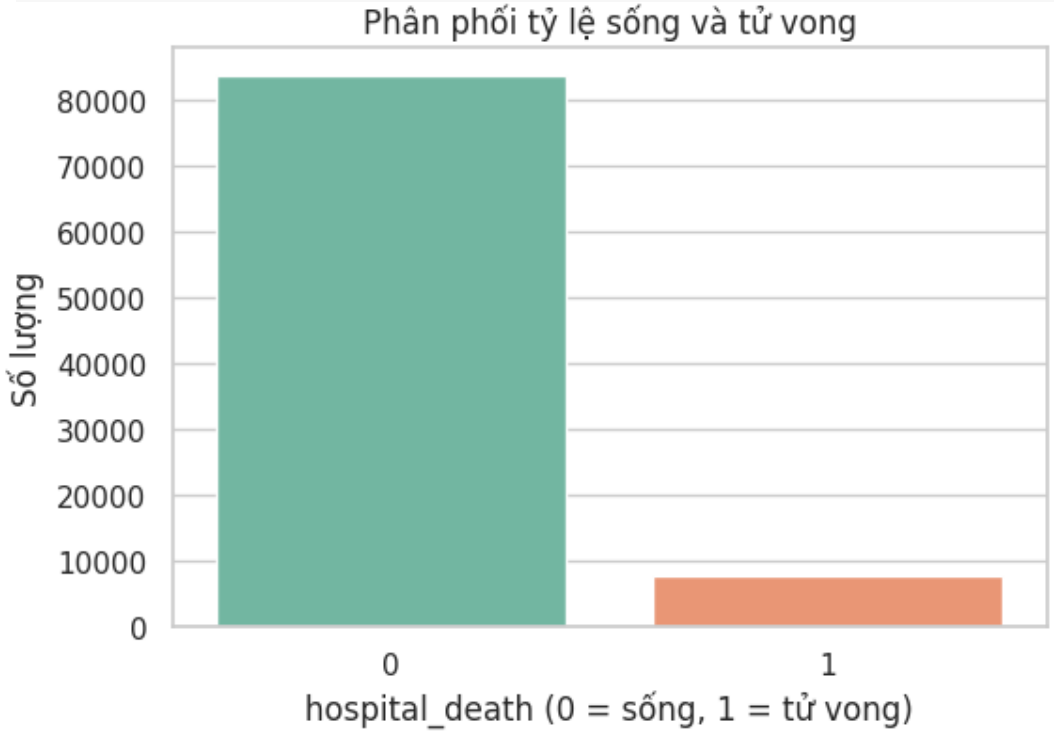
- **Tiểu đường (diabetes_mellitus):**
  - Có tiểu đường: **22.5%**
  - Đây là yếu tố nguy cơ quan trọng ảnh hưởng đến thời gian hồi phục và tỷ lệ tử vong.
- **Suy gan (hepatic_failure):**
  - Có suy gan: **1.3%**
  - Tỷ lệ thấp nhưng rất nguy hiểm, bệnh nhân có bệnh lý này có nguy cơ tử vong rất cao.
- **U đặc ác tính có di căn (solid_tumor_with_metastasis):**
  - Có khối u ác tính di căn: **2.06%**
  - Bệnh lý nghiêm trọng làm tăng mạnh nguy cơ tử vong trong ICU.

Biến mục tiêu

- **Tử vong tại bệnh viện (hospital_death):**
  - Tỷ lệ sống sót: **91.37%**
  - Tỷ lệ tử vong: **8.63%**

⇨ Mất cân bằng rõ rệt, đòi hỏi kỹ thuật cân bằng dữ liệu đặc biệt khi xây dựng mô hình.

```
if 'hospital_death' in df.columns:
    plt.figure(figsize=(6, 4))
    sns.countplot(data=df, x='hospital_death', palette='Set2')
    plt.title("Phân phối tỷ lệ sống và tử vong")
    plt.xlabel("hospital_death (0 = sống, 1 = tử vong)")
    plt.ylabel("Số lượng")
    plt.show()
else:
    print(" ❗ Không tìm thấy cột hospital_death để vẽ.")
```



CHƯƠNG 2. TIỀN XỬ LÝ DỮ LIỆU CHO MÁY HỌC

2.1. Mục tiêu

Mục tiêu của chương này là thực hiện các bước **làm sạch, định dạng, chuẩn hóa**, và **loại bỏ các điểm bất thường** trong tập dữ liệu, để đảm bảo dữ liệu sẵn sàng và phù hợp cho việc xây dựng mô hình học máy. Các bước này giúp nâng cao chất lượng của dữ liệu, giảm thiểu rủi ro overfitting và underfitting, đảm bảo tính chính xác trong quá trình huấn luyện mô hình.

Lý do: Mô hình học máy không thể xử lý dữ liệu raw chưa được định dạng, đặc biệt là các biến dạng string, missing, hoặc chưa chuẩn hóa.

2.2. Làm sạch dữ liệu

Các công việc cụ thể bao gồm:

2.2.1. Xử lý dữ liệu thiếu:

Xác định các trường dữ liệu bị thiếu và áp dụng các phương pháp thích hợp để xử lý, chẳng hạn như thay thế bằng giá trị trung bình, giá trị trung vị, hoặc loại bỏ các dòng chứa dữ liệu thiếu.

```
[ ]  # Số lượng giá trị thiếu
     missing = df.isna().sum().sort_values(ascending=False)
     print("Top 10 cột bị thiếu dữ liệu:")
     print(missing.head(10))
```

```
Top 10 cột bị thiếu dữ liệu:
Unnamed: 83                    91713
d1_potassium_max                9585
d1_potassium_min                9585
h1_mbp_noninvasive_min          9084
h1_mbp_noninvasive_max          9084
apache_4a_hospital_death_prob   7947
apache_4a_icu_death_prob        7947
h1_diasbp_noninvasive_max       7350
h1_diasbp_noninvasive_min       7350
h1_sysbp_noninvasive_max        7341
dtype: int64
```

```
# Chia cột số và cột phân loại
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

print("Cột số:", len(numeric_cols))
print("Cột phân loại:", len(categorical_cols))
```

```
Cột số: 78
Cột phân loại: 7
```

```
[ ]
# Với cột số: dùng mean
df[numeric_cols] = df[numeric_cols].apply(lambda x: x.fillna(x.mean()))
```

```
[1] # Với cột phân loại: dùng mode nếu có, nếu không → 'Unknown'
for col in categorical_cols:
    if df[col].isnull().sum() > 0:
        mode_vals = df[col].mode()
        if not mode_vals.empty:
            df[col] = df[col].fillna(mode_vals[0])
        else:
            df[col] = df[col].fillna("Unknown")
```

2.2.2. Định dạng lại dữ liệu

Kiểm tra và chuẩn hóa các định dạng dữ liệu như ngày tháng, kiểu dữ liệu số và chuỗi để đảm bảo tính đồng nhất trong toàn bộ dataset.

4.XÓA TRÙNG LẶP

```
df = df.drop_duplicates()
print(f"Sau khi xóa trùng lặp: {df.shape}")
```

```
Sau khi xóa trùng lặp: (91713, 85)
```

5.XÓA CÁC CỘT KHÔNG MANG THÔNG TIN (DUY NHẤT 1 GIÁ TRỊ)

```
[ ] constant_cols = df.nunique()[df.nunique() == 1]
    df = df.drop(columns=constant_cols.index)
    print("Đã xóa các cột chỉ có 1 giá trị duy nhất.")
```

```
Đã xóa các cột chỉ có 1 giá trị duy nhất.
```

6.LÀM SẠCH ĐỊNH DẠNG DỮ LIỆU TEXT

```
[ ] # Loại bỏ dấu, khoảng trắng thừa
    df = df.replace({',': '', '    ': '', '\(': '', '\)': ''}, regex=True)
```

7.ÉP KIỂU CÁC CỘT NGHI NGỜ ĐANG Ở DẠNG CHUỖI NHƯNG LÀ SỐ

```
[ ] # Ví dụ: các cột từ 2 đến 7 có thể là số nhưng ở dạng object
    df.iloc[:, 2:8] = df.iloc[:, 2:8].apply(pd.to_numeric, errors='coerce')
```

2.2.3. Loại bỏ hoặc điều chỉnh các điểm bất thường (outliers)

Xác định và loại bỏ các giá trị bất thường có thể ảnh hưởng đến kết quả mô hình, chẳng hạn như các chỉ số sinh lý cực kỳ cao hoặc thấp không hợp lý.

9.CHUYỂN CÁC CỘT SANG DẠNG CATEGORY

```
[ ]  for col in categorical_cols:
         if df[col].nunique() < 50:
             df[col] = df[col].astype("category")
```

10.GOM NHÓM NHÃN HIẾM THÀNH OTHER

```
[ ]  for col in df.select_dtypes(include='category'):
         value_counts = df[col].value_counts(normalize=True)
         rare_labels = value_counts[value_counts < 0.01].index
         df[col] = df[col].replace(rare_labels, 'Other')
```

11.KIỂM TRA PHÂN PHỐI NHÃ MỤC TIÊU (NẾU CÓ)

```
[ ]  if 'hospital_death' in df.columns:
         print("Tỷ lệ phân phối hospital_death:")
         print(df['hospital_death'].value_counts(normalize=True) * 100)
     else:
         print("❗ Không tìm thấy cột hospital_death.")
```

```
Tỷ lệ phân phối hospital_death:
hospital_death
0    91.369817
1     8.630183
Name: proportion, dtype: float64
```

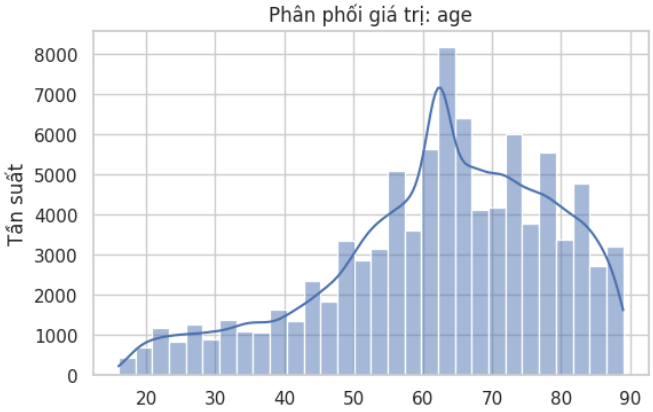
12. CÂN BẰNG DỮ LIỆU BẰNG SMOTE

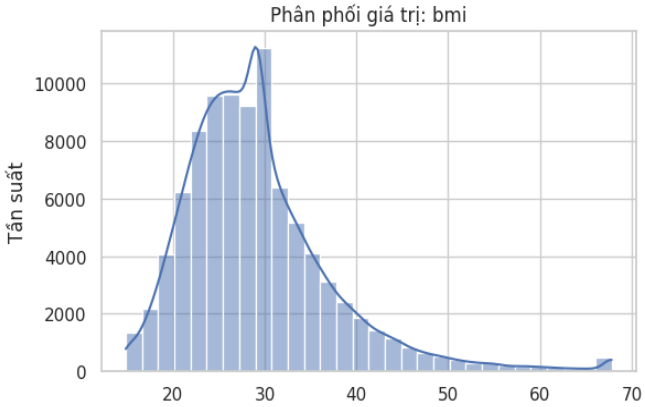
```
[ ]  from imblearn.over_sampling import SMOTE
     from sklearn.model_selection import train_test_split

     if 'hospital_death' in df.columns:
         X = df.drop(columns='hospital_death')
         y = df['hospital_death']

         # Encode object/category → one-hot
         X = pd.get_dummies(X, drop_first=True)

         # Chia dữ liệu train/test
         X_train, X_test, y_train, y_test = train_test_split(
             X.fillna(0), y, stratify=y, test_size=0.2, random_state=42
         )

         # Dùng SMOTE để cân bằng class trong train set
         sm = SMOTE(random_state=42)
         X_train_resampled, y_train_resampled = sm.fit_resample(X_train, y_train)

         print(f"Dữ liệu train sau SMOTE: {X_train_resampled.shape}")
     else:
         print("❗ Không thể SMOTE vì thiếu cột hospital_death.")
```

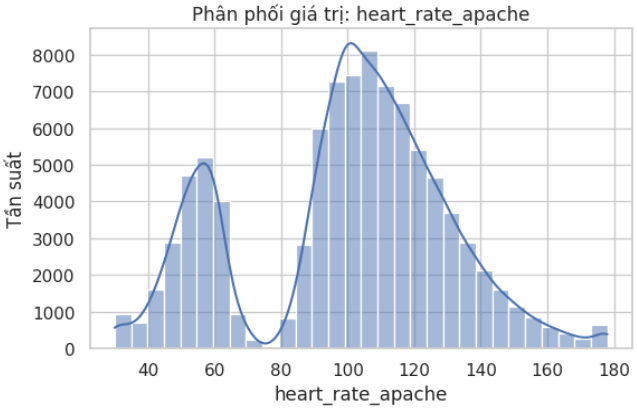
=> Dữ liệu train sau SMOTE: (134076, 107)

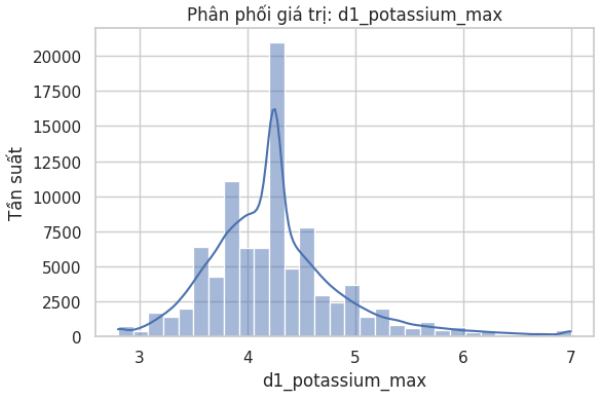
2.2.4. Chuẩn hóa và mã hóa dữ liệu phân loại

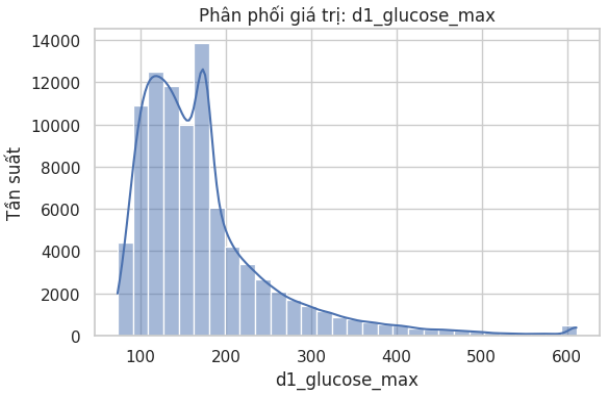
Các biến phân loại (như giới tính, bệnh nền) sẽ được mã hóa thành các giá trị số (ví dụ: sử dụng phương pháp one-hot encoding) để mô hình học máy có thể hiểu và xử lý.

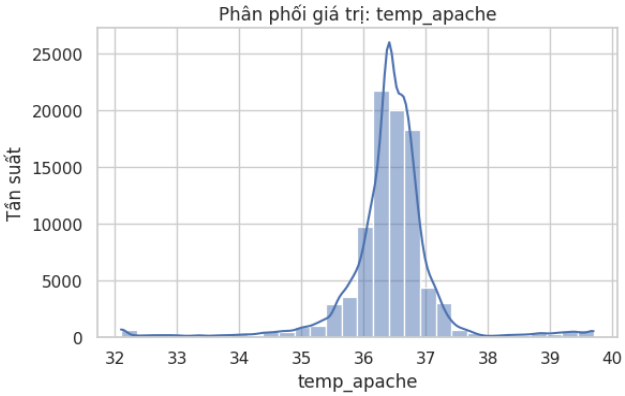












- **Dữ liệu định lượng (số liệu liên quan đến các chỉ số sinh lý)**: Đối với các trường dữ liệu định lượng như tuổi, nhịp tim, huyết áp, nhiệt độ, chúng ta có thể thay thế các giá trị thiếu bằng **giá trị trung bình (mean)** hoặc **giá trị trung vị**

(median) của cột đó. Việc sử dụng trung bình hoặc trung vị giúp giữ cho các phân bố của dữ liệu không bị lệch quá nhiều.

- **Mean (trung bình)**: Thích hợp khi dữ liệu phân bố đều và không có sự lệch đáng kể.
- **Median (trung vị)**: Thích hợp khi dữ liệu có phân phối lệch hoặc có các giá trị ngoại lai (outliers).

- **Dữ liệu phân loại (categorical data)**: Đối với các cột phân loại như **giới tính**, **bệnh nền**, chúng ta có thể thay thế các giá trị thiếu bằng **giá trị mode** (giá trị xuất hiện nhiều nhất). Điều này đảm bảo rằng các giá trị thiếu được thay thế bằng giá trị phổ biến nhất trong dữ liệu, giúp giảm thiểu sự sai lệch trong mô hình.

2.2.5. Lý do không thể thay thiếu bằng 0

Việc thay thế các giá trị thiếu bằng **0** có thể gây ra các vấn đề lớn trong mô hình học máy, đặc biệt là trong môi trường y tế. Các lý do bao gồm:

- **Sai lệch y học**: Nếu thay thế giá trị thiếu bằng 0 trong các chỉ số sinh lý (như nhịp tim, huyết áp, v.v.), điều này có thể tạo ra những kết quả không thực tế và làm sai lệch kết quả dự đoán.
- **Làm giảm tính chính xác của mô hình**: Mô hình học máy sẽ không thể phân biệt được giữa dữ liệu thực và dữ liệu giả (0), từ đó dẫn đến các dự đoán không chính xác.

Do đó, thay vì sử dụng giá trị 0, chúng ta sẽ sử dụng các phương pháp thay thế hợp lý như trung bình, trung vị hoặc mode để đảm bảo dữ liệu phản ánh đúng tình trạng thực tế của bệnh nhân.

2.2.6. Dữ liệu sau khi làm sạch

Việc xử lý dữ liệu thiếu là một bước quan trọng để chuẩn bị dữ liệu trước khi đưa vào mô hình học máy. Cần phải chọn phương pháp thay thế hợp lý (mean/median

cho dữ liệu định lượng, mode cho dữ liệu phân loại) và tránh thay thế thiếu bằng giá trị 0, vì điều này có thể dẫn đến sai lệch và ảnh hưởng đến kết quả cuối cùng.

	encounter_id	patient_id	hospital_id	age	bmi	elective_surgery	ethnicity	gender	height	icu_admit_source	...	diabetes_mellitus	hepatic_failure
0	66154	25312	118	68.0	22.730000	0	Unknown	Unknown	180.3	Floor	...	1.0	0.0
1	114252	59342	81	77.0	27.420000	0	Unknown	Unknown	160.0	Floor	...	1.0	0.0
2	119783	50777	118	25.0	31.950000	0	Unknown	Unknown	172.7	Accident & Emergency	...	0.0	0.0
3	79267	46918	118	81.0	22.640000	1	Unknown	Unknown	165.1	Operating Room / Recovery	...	0.0	0.0
4	92056	34377	33	19.0	29.185818	0	Unknown	Unknown	188.0	Accident & Emergency	...	0.0	0.0
5	33181	74489	83	67.0	27.560000	0	Unknown	Unknown	190.5	Accident & Emergency	...	1.0	0.0
6	82208	49526	83	59.0	57.450000	0	Unknown	Unknown	165.1	Accident & Emergency	...	1.0	0.0
7	120995	50129	33	70.0	29.185818	0	Unknown	Unknown	165.0	Accident & Emergency	...	0.0	0.0
8	80471	10577	118	45.0	29.185818	0	Unknown	Unknown	170.2	Other Hospital	...	0.0	0.0
9	42871	90749	118	50.0	25.710000	0	Unknown	Unknown	175.3	Accident & Emergency	...	0.0	0.0

hepatic_failure	immunosuppression	leukemia	lymphoma	solid_tumor_with_metastasis	apache_3j_bodysystem	apache_2_bodysystem	Unnamed: 83	hospital_death
0.0	0.0	0.0	0.0	0.0	Sepsis	Cardiovascular	NaN	0
0.0	0.0	0.0	0.0	0.0	Respiratory	Respiratory	NaN	0
0.0	0.0	0.0	0.0	0.0	Metabolic	Metabolic	NaN	0
0.0	0.0	0.0	0.0	0.0	Cardiovascular	Cardiovascular	NaN	0
0.0	0.0	0.0	0.0	0.0	Trauma	Trauma	NaN	0
0.0	0.0	0.0	0.0	0.0	Neurological	Neurologic	NaN	0
0.0	0.0	0.0	0.0	0.0	Respiratory	Respiratory	NaN	0
0.0	1.0	0.0	0.0	0.0	Sepsis	Cardiovascular	NaN	0

10 rows × 85 columns

```
Kích thước dữ liệu: (91713, 85)
Số lượng giá trị thiếu trong mỗi cột:
encounter_id                      0
patient_id                        0
hospital_id                       0
age                               0
bmi                               0
                                 ...
solid_tumor_with_metastasis       0
apache_3j_bodysystem              0
apache_2_bodysystem               0
Unnamed: 83                   91713
hospital_death                    0
Length: 85, dtype: int64
```

✅ Dữ liệu đã được lưu thành công vào file 'cleaned_dataset.csv'

2.3. Chọn biến đầu vào (Feature Selection)

- Kiểm tra sự tương quan giữa các biến đầu vào và giữa các biến với biến mục tiêu (target).
- Các biến có tương quan mạnh với nhau có thể được loại bỏ, vì chúng có thể gây ra hiện tượng **multicollinearity**, làm giảm hiệu quả của mô hình.
- Các biến không có sự tương quan mạnh với biến mục tiêu cũng có thể bị loại bỏ, vì chúng không đóng góp nhiều vào việc dự đoán.
- Chọn 16 biến đầu vào có ý nghĩa lâm sàng và tương quan cao với tỷ lệ tử vong:

```
# Chọn các đặc trưng và biến mục tiêu
selected_features = [
    'age', 'gender', 'bmi', 'apache_4a_hospital_death_prob',
    'heart_rate_apache', 'map_apache', 'temp_apache',
    'd1_glucose_max', 'd1_potassium_max',
    'gcs_eyes_apache', 'gcs_motor_apache', 'gcs_verbal_apache',
    'diabetes_mellitus', 'hepatic_failure',
    'immunosuppression', 'solid_tumor_with_metastasis',
    'icu_type', 'apache_2_bodysystem'
]
```

Hình 2.3: Các biến đặc trưng

- Biến mục tiêu: hospital_death (0 = sống, 1 = tử vong).

2.4 Mã hóa phân loại (Encoding)

Mã hóa phân loại (**Categorical Encoding**) là quá trình chuyển đổi các biến phân loại (categorical variables) thành dạng số để mô hình học máy có thể hiểu và sử dụng được. Các mô hình học máy không thể làm việc trực tiếp với các dữ liệu phân loại dưới dạng văn bản, do đó việc mã hóa chúng là rất cần thiết.

Dưới đây là một số phương pháp mã hóa phân loại phổ biến:

2.4.1. One-Hot Encoding

One-Hot Encoding là phương pháp phổ biến nhất để mã hóa các biến phân loại. Mỗi giá trị duy nhất của một biến phân loại sẽ được chuyển thành một cột mới. Mỗi cột

tương ứng với một giá trị phân loại, và trong mỗi dòng, cột có giá trị bằng 1 (nếu đó là giá trị của biến phân loại) và các cột còn lại có giá trị bằng 0.

2.4.2. Label Encoding

Label Encoding là phương pháp gán một giá trị số nguyên cho mỗi giá trị phân loại. Mỗi giá trị duy nhất của biến phân loại sẽ được ánh xạ vào một số nguyên.

2.4.3. Label Encoding với Ordinal Encoding

Ordinal Encoding là một dạng mã hóa giống **Label Encoding**, nhưng được sử dụng khi các giá trị phân loại có thứ tự tự nhiên. Ví dụ, trong trường hợp `các` mức độ như `Thấp`, `Trung bình`, `Cao`, bạn có thể mã hóa chúng thành các số 0, 1, 2.

2.4.4. Target Encoding (Mean Encoding)

Target Encoding là phương pháp mã hóa các giá trị phân loại dựa trên giá trị trung bình của biến mục tiêu (target). Mỗi giá trị của biến phân loại sẽ được mã hóa thành giá trị trung bình của biến mục tiêu cho các mẫu có giá trị đó.

2.4.5. Binary Encoding

Binary Encoding là phương pháp mã hóa mà mỗi giá trị phân loại được chuyển thành số nhị phân (binary), sau đó mã hóa các số này thành các cột.

2.4.6. Frequency Encoding

Frequency Encoding là một phương pháp mã hóa trong đó mỗi giá trị phân loại được thay thế bằng tần suất xuất hiện của nó trong tập dữ liệu.

Biến	Kiểu	Cách xử lý
gender	object	Ánh xạ thành 0/1 (nữ/nam) hoặc dùng One-Hot Encoding
Các bệnh nền (diabetes_mellitus,...)	nhị phân	Không cần mã hóa vì đã 0/1

Hình 2.4.6.1

2.5. Chuẩn hóa dữ liệu Scaling

Là một bước quan trọng trong tiền xử lý dữ liệu, đặc biệt là trong các bài toán học máy (machine learning) và phân tích dữ liệu. Mục tiêu của chuẩn hóa dữ liệu là đưa các đặc trưng (features) của dữ liệu về cùng một thang đo hoặc phạm vi, giúp các thuật toán học máy hoạt động hiệu quả hơn.

- Áp dụng **StandardScaler** là một công cụ trong thư viện **scikit-learn** của Python, được sử dụng để chuẩn hóa dữ liệu sao cho các đặc trưng (features) có trung bình bằng 0 và độ lệch chuẩn (standard deviation) bằng 1. Đây là một kỹ thuật chuẩn hóa rất phổ biến và thường được sử dụng trong các mô hình học máy.
- Một số biến như bmi, glucose, heart_rate_apache có phân phối lệch lớn.
- StandardScaler và các phương pháp chuẩn hóa dữ liệu là rất quan trọng đối với một số mô hình học máy, đặc biệt là các mô hình như Logistic Regression và KNN (K-Nearest Neighbors), vì chúng có nhạy cảm với thang đo (scale sensitivity) của các đặc trưng (features).
- KNN là một thuật toán học máy dựa trên khoảng cách (distance-based), thường sử dụng Euclidean distance hoặc các loại khoảng cách khác như Manhattan distance để tính toán sự tương tự giữa các điểm dữ liệu.

➢ Lý do cần chuẩn hóa: KNN sử dụng khoảng cách giữa các điểm dữ liệu để xác định các điểm lân cận. Nếu một đặc trưng có phạm vi lớn hơn các đặc trưng khác (ví dụ: một đặc trưng có giá trị từ 0 đến 1000 trong khi một đặc

trưng khác có giá trị từ 0 đến 1), thuật toán KNN sẽ ưu tiên những đặc trưng có giá trị lớn hơn khi tính toán khoảng cách, dẫn đến kết quả sai lệch.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Đưa về lại DataFrame để dễ quan sát
X = pd.DataFrame(X_scaled, columns=X.columns)
```

2.6. Chia tập train và test

❖ Mục đích chia tập dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test) là một bước quan trọng trong quá trình xây dựng và đánh giá mô hình học máy (machine learning).

2.6.1. Đánh giá khả năng tổng quát của mô hình

- Tập huấn luyện (train set): Dùng để huấn luyện mô hình, tối ưu các tham số của mô hình (như các trọng số trong mạng nơ-ron hay các hệ số trong hồi quy tuyến tính).
- Tập kiểm tra (test set): Dùng để kiểm tra hiệu suất của mô hình trên dữ liệu chưa từng thấy. Mô hình không được phép nhìn thấy dữ liệu này trong quá trình huấn luyện, từ đó cho bạn cái nhìn về khả năng tổng quát của mô hình.

2.6.2. Phát hiện hiện tượng overfitting (quá khớp) và underfitting (thiếu khớp)

- **Overfitting**: Xảy ra khi mô hình học quá kỹ các chi tiết trong tập huấn luyện và không thể tổng quát tốt trên dữ liệu mới. Mô hình "học vẹt" và có thể cho kết quả rất tốt trên tập huấn luyện nhưng lại hoạt động kém khi gặp dữ liệu mới. Việc sử dụng tập kiểm tra giúp phát hiện hiện tượng này.

- **Underfitting**: Xảy ra khi mô hình không học được các mối quan hệ phức tạp trong dữ liệu và có hiệu suất kém trên cả tập huấn luyện và tập kiểm tra.

2.6.3. Đánh giá hiệu suất mô hình

Chia dữ liệu thành tập huấn luyện và kiểm tra giúp chúng ta đánh giá hiệu suất của mô hình trong các tình huống thực tế:

- **Tập huấn luyện** giúp ta biết được mô hình có thể học tốt từ dữ liệu hay không.
- **Tập kiểm tra** giúp ta đánh giá xem mô hình có thể áp dụng tốt những gì đã học vào dữ liệu mới (dữ liệu chưa thấy).

Nếu mô hình có hiệu suất cao trên tập huấn luyện nhưng lại kém trên tập kiểm tra, điều này cho thấy mô hình có thể bị **overfitting**.

2.6.4. Cải thiện mô hình (Cross-validation)

Chia tập dữ liệu thành tập huấn luyện và kiểm tra giúp ta có thể cải thiện mô hình thông qua các phương pháp như **cross-validation**. Cross-validation chia dữ liệu thành nhiều phần (folds), và mỗi phần sẽ được lần lượt sử dụng làm tập kiểm tra, trong khi phần còn lại dùng làm tập huấn luyện. Phương pháp này giúp chúng ta có được một đánh giá đáng tin cậy hơn về hiệu suất của mô hình.

2.6.5. Tạo ra mô hình tối ưu

Việc chia tập dữ liệu giúp bạn thử nghiệm nhiều mô hình khác nhau, tinh chỉnh các tham số (hyperparameters) và tìm ra mô hình phù hợp nhất. Bạn sẽ chỉ biết mô hình nào hoạt động tốt nhất khi đánh giá được hiệu suất của nó trên dữ liệu kiểm tra.

2.6.6. Tách biệt dữ liệu để tránh lạm dụng thông tin

Khi chia dữ liệu thành tập huấn luyện và kiểm tra, bạn đảm bảo rằng không có sự lạm dụng thông tin từ tập kiểm tra trong quá trình huấn luyện. Nếu không chia dữ liệu, mô hình có thể "học" từ tập kiểm tra, làm cho kết quả đánh giá không còn chính xác nữa.

Dữ liệu được chia theo tỷ lệ 80/20:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
print("Dữ liệu huấn luyện:", X_train.shape)
print("Dữ liệu kiểm tra:", X_test.shape)
```

Bảo đảm phân phối hospital_death vẫn giữ nguyên trong 2 tập (dùng stratify=y nếu cần).

2.7. Cân bằng dữ liệu bằng SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) là một kỹ thuật cân bằng dữ liệu, được sử dụng để giải quyết vấn đề mất cân bằng lớp (class imbalance) trong học máy. Khi dữ liệu huấn luyện có sự chênh lệch lớn giữa số lượng mẫu của các lớp (ví dụ, có một lớp rất ít mẫu so với lớp còn lại), mô hình máy học có thể sẽ bị thiên lệch, chỉ học được đặc trưng của lớp lớn và không nhận diện tốt lớp ít mẫu.

2.7.1. Mục đích

Giải quyết vấn đề mất cân bằng dữ liệu:

- Trong các bài toán phân loại, đặc biệt là khi có sự mất cân bằng giữa các lớp (ví dụ: lớp "xảy ra" chỉ chiếm 10% tổng số dữ liệu, trong khi lớp "không xảy ra" chiếm 90%), mô hình học máy có thể sẽ thiên về lớp chiếm ưu thế (lớp chiếm phần lớn dữ liệu) và bỏ qua lớp ít dữ liệu.
- SMOTE giúp tạo ra các mẫu giả cho lớp ít dữ liệu bằng cách tạo ra các điểm dữ liệu mới (synthetic data points), giúp cân bằng số lượng mẫu giữa các lớp.

Tạo ra các điểm dữ liệu mới (synthetic data points):

- SMOTE không sao chép lại các mẫu có sẵn từ lớp ít dữ liệu mà thay vào đó tạo ra các điểm dữ liệu mới, có đặc điểm tương tự như các mẫu gốc.
- Các điểm mới này được tạo ra bằng cách kết hợp các đặc trưng của các điểm dữ liệu trong lớp thiểu số, từ đó tạo ra các mẫu dữ liệu mới trong không gian đặc trưng.

```
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = sm.fit_resample(X_train, y_train)

print("Dữ liệu huấn luyện sau SMOTE:", X_train_resampled.shape)
print("Tỷ lệ tử vong sau SMOTE: {:.2f}%".format(y_train_resampled.mean() * 100))
```

Dạng	Gợi ý đưa vào
Bảng	Danh sách 16 biến đầu vào đã chọn, mô tả từng biến
Code mẫu	SMOTE, chuẩn hóa, chia tập train/test
Hình	Histogram hospital_death trước và sau SMOTE
Ghi chú	Dữ liệu sau khi xử lý đã sẵn sàng cho huấn luyện, và có thể in ra file preprocessed_dataset.csv nếu cần lưu lại

CHƯƠNG 3. XÂY DỰNG, ĐÁNH GIÁ VÀ TỐI ƯU MÔ HÌNH DỰ ĐOÁN TỬ VONG ICU

3.1. *Mục tiêu của chương*

Mục tiêu chính của chương này là xây dựng và tối ưu các mô hình học máy nhằm dự đoán nguy cơ tử vong của bệnh nhân tại khoa hồi sức tích cực (ICU) dựa trên dữ liệu lâm sàng từ bộ dữ liệu MIMIC-IV. Các mục tiêu cụ thể:

- Tiền xử lý dữ liệu và trích chọn đặc trưng phù hợp từ bộ dữ liệu MIMIC-IV.
- Xây dựng và huấn luyện các mô hình học máy gồm Logistic Regression, Random Forest, XGBoost và LightGBM.
- Tối ưu siêu tham số nhằm cải thiện hiệu năng của mô hình.
- So sánh hiệu suất các mô hình qua các chỉ số đo lường như Accuracy, Precision, Recall, F1-score và AUC.
- Diễn giải mô hình bằng công cụ SHAP để hiểu rõ tầm quan trọng của các đặc trưng.

3.2. *Xây dựng mô hình máy học*

3.2.1. Mô hình Logistic Regression (LR)

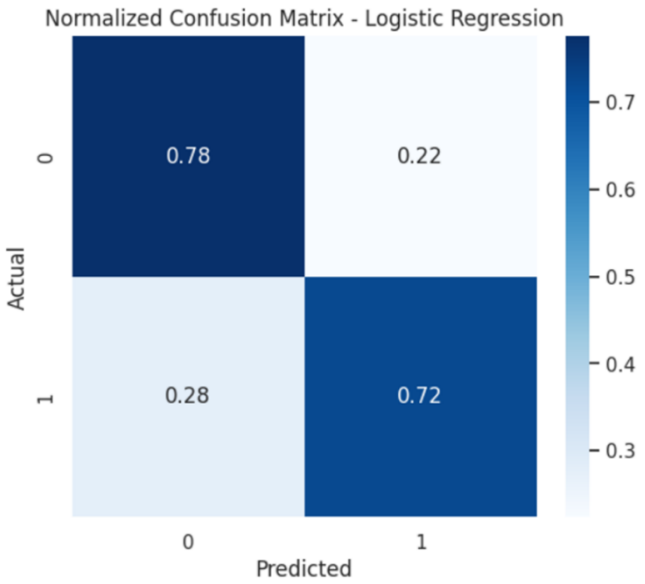
Logistic Regression là một mô hình tuyến tính đơn giản nhưng hiệu quả, đặc biệt phù hợp với các bài toán phân loại nhị phân như tử vong/sống sót. Trong bối cảnh y học, LR thường được sử dụng nhờ khả năng giải thích trực quan: mỗi hệ số hồi quy đại diện cho log-odds (logit) của tử vong tương ứng với một đơn vị thay đổi của biến đầu vào.

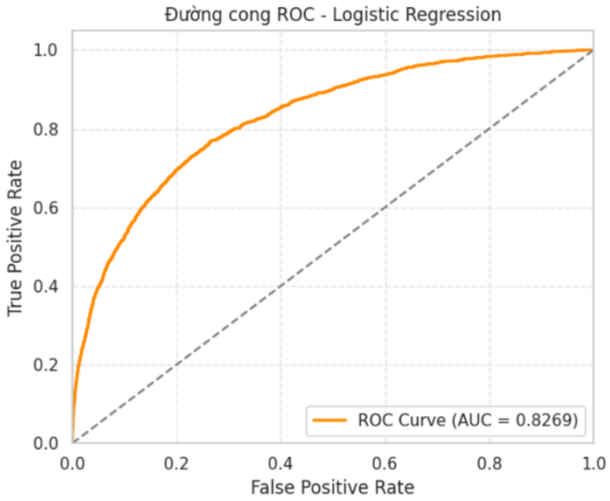
Mô hình giả định rằng xác suất tử vong tuân theo hàm logistic:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

Do bản chất tuyến tính, LR phù hợp với các mối quan hệ đơn giản giữa biến độc lập và biến mục tiêu. Tuy nhiên, mô hình này có thể bị hạn chế trong việc nắm bắt các mối quan hệ phi tuyến hoặc tương tác phức tạp giữa các đặc trưng.

Sau khi huấn luyện mô hình Logistic Regression, kết quả dự đoán được đánh giá qua **ma trận nhầm lẫn** và **đường cong ROC** bên dưới.





Hình a: Ma trận nhầm lẫn chuẩn hóa Hình b: Đường cong ROC với AUC = 0.8269

Hình 3.2.1.1: Dự đoán kết quả sau khi huấn luyện mô hình Logistic Regression

- **Hình a** minh họa ma trận nhầm lẫn đã được chuẩn hóa, cho thấy mô hình dự đoán đúng 78% các trường hợp sống sót và 72% các trường hợp tử vong.
- **Hình b** thể hiện đường cong ROC của mô hình, với **AUC = 0.8269**, cho thấy khả năng phân biệt hai lớp (sống – tử vong) của mô hình là khá tốt.

3.2.2. Mô hình Random Forest (RF)

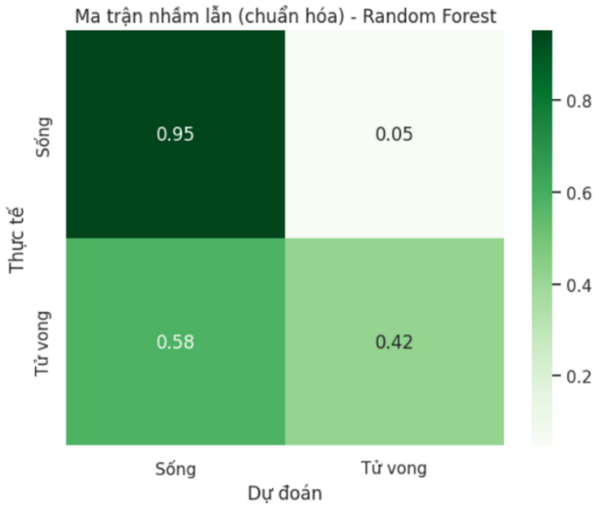
Random Forest là một thuật toán học máy thuộc nhóm mô hình cây quyết định tập hợp (ensemble), trong đó nhiều cây quyết định được huấn luyện trên các tập con ngẫu nhiên của dữ liệu và đặc trưng. Dự đoán cuối cùng được lấy theo nguyên tắc đa số (voting) trong phân loại.

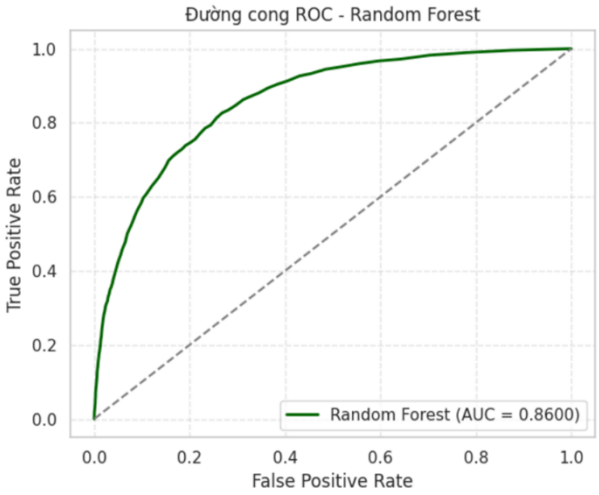
Ưu điểm nổi bật của Random Forest:

- Khả năng xử lý dữ liệu có mối quan hệ phi tuyến.
- Giảm hiện tượng overfitting so với cây quyết định đơn lẻ.
- Hỗ trợ đánh giá tầm quan trọng của đặc trưng (feature importance).

Tuy nhiên, RF đôi khi có thể trở nên nặng nề về mặt tính toán và khó giải thích hơn so với LR, đặc biệt trong bối cảnh lâm sàng.

Mô hình Random Forest được huấn luyện trên tập dữ liệu đã xử lý và được đánh giá dựa trên các chỉ số như ma trận nhầm lẫn và đường cong ROC.





Hình a: Ma trận nhầm lẫn chuẩn hóa Hình b: Đường cong ROC với AUC = 0.8600

Hình 3.2.2.1: Dự đoán kết quả sau khi huấn luyện mô hình Random Forest

- **Hình a** thể hiện ma trận nhầm lẫn chuẩn hóa, trong đó mô hình dự đoán đúng 95% các trường hợp sống sót, nhưng chỉ dự đoán đúng 42% các trường hợp tử vong. Điều này cho thấy mô hình có xu hướng nghiêng về việc dự đoán bệnh nhân sống sót.
- **Hình b** là đường cong ROC với **AUC = 0.8600**, cao hơn Logistic Regression, thể hiện khả năng phân loại tốt hơn giữa hai lớp sống – tử vong.

3.2.3. Mô hình XGBoost

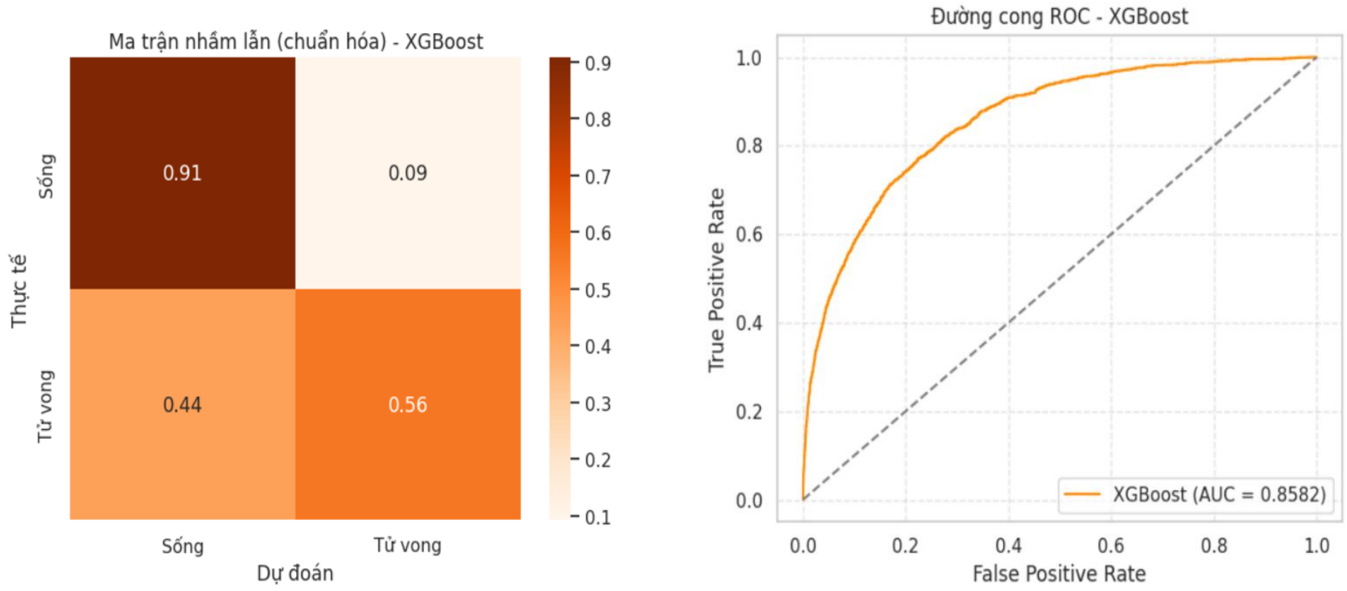
XGBoost (Extreme Gradient Boosting) là một thuật toán boosting hiệu quả và phổ biến trong các bài toán học máy hiện đại. Mô hình hoạt động theo cơ chế học tuần tự: mỗi cây mới được xây dựng để giảm lỗi sai còn lại từ các cây trước đó.

Đặc điểm chính của XGBoost:

- Sử dụng thuật toán boosting với hàm mất mát chuẩn hóa.
- Hỗ trợ regularization (L1, L2) để giảm overfitting.
- Hiệu suất cao trong cả thời gian huấn luyện lẫn độ chính xác.

XGBoost thường đạt kết quả tốt hơn RF và LR trong các bộ dữ liệu phức tạp, song cần tinh chỉnh siêu tham số kỹ lưỡng để đạt hiệu quả tối ưu.

Mô hình XGBoost được huấn luyện trên tập dữ liệu đã xử lý và được đánh giá dựa trên các chỉ số như ma trận nhầm lẫn và đường cong ROC.



Hình a: Ma trận nhầm lẫn chuẩn hóa

Hình b: Đường cong ROC với AUC = 0.8582

Hình 3.2.3.1: Dự đoán kết quả sau khi huấn luyện mô hình XGBoost

- **Hình a** cho thấy mô hình dự đoán đúng 91% các trường hợp bệnh nhân sống và 56% các trường hợp tử vong. Tuy nhiên, tỷ lệ dự đoán sai tử vong thành sống vẫn ở mức tương đối cao (44%), cho thấy mô hình cần cải thiện độ nhạy đối với lớp tử vong.
- **Hình b** minh họa rõ ràng khả năng phân biệt hai lớp (sống và tử vong) của mô hình. Đường cong nằm phía trên đường chéo ngẫu nhiên, cho thấy mô hình có độ chính xác tổng thể khá tốt.

3.2.4. Mô hình LightGBM

LightGBM là một thuật toán boosting hiện đại do Microsoft phát triển, tối ưu cho hiệu suất và khả năng mở rộng. Không giống XGBoost sử dụng phương pháp chia chiều sâu cây (depth-wise), LightGBM sử dụng phương pháp chia theo độ rộng (leaf-wise), giúp cải thiện tốc độ huấn luyện và độ chính xác.

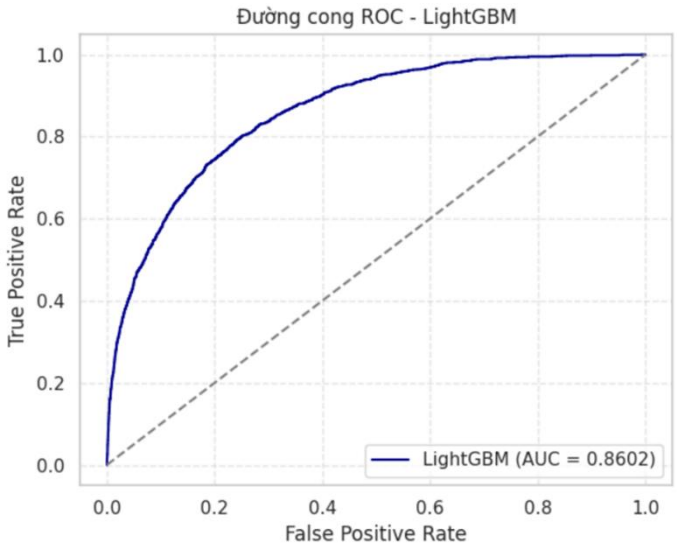
Ưu điểm của LightGBM:

- Tối ưu hóa cho bộ dữ liệu lớn và số lượng đặc trưng cao.
- Hỗ trợ xử lý dữ liệu mất mát và phân loại.
- Tốc độ huấn luyện nhanh vượt trội, tiêu tốn ít bộ nhớ hơn so với XGBoost.

LightGBM được xem là một trong những mô hình mạnh nhất hiện nay trong các cuộc thi học máy, đồng thời cũng phù hợp với ứng dụng thực tế nhờ hiệu năng và khả năng xử lý đặc trưng lớn.

Mô hình LightGBM được huấn luyện trên tập dữ liệu đã xử lý và được đánh giá dựa trên các chỉ số như ma trận nhầm lẫn và đường cong ROC.

Hình a: Ma trận nhầm lẫn chuẩn hóa



Hình b: Đường cong ROC với AUC = 0.8602

Hình 3.2.4.1: Dự đoán kết quả sau khi huấn luyện mô hình LightGBM

- **Hình a** cho thấy mô hình LightGBM dự đoán đúng 92% các trường hợp sống và 54% các trường hợp tử vong. Mặc dù tỷ lệ phát hiện bệnh nhân tử vong còn thấp (chỉ hơn một nửa), nhưng kết quả này vẫn tương đương với mô hình XGBoost. Đáng chú ý là mô hình LightGBM có tỷ lệ nhầm lẫn bệnh nhân tử vong thành sống cao (46%).
- **Hình b** minh họa đường cong ROC của mô hình với diện tích dưới đường cong (AUC) đạt 0.8602, cao hơn một chút so với XGBoost (0.8582). Điều này cho thấy LightGBM có khả năng phân biệt giữa hai lớp (sống và tử vong) tốt hơn một chút so với mô hình trước đó.

3.2.5. Mô hình XGBoost (tuned)

XGBoost là thuật toán boosting mạnh mẽ, cần phải tuning các siêu tham số để tối ưu hóa hiệu suất:

- **learning_rate**: Điều chỉnh mức độ thay đổi của mỗi bước cập nhật. Giá trị thấp giúp tránh overfitting.
- **n_estimators**: Số lượng cây quyết định. Tăng để cải thiện hiệu suất.
- **max_depth**: Giới hạn chiều sâu cây quyết định, tránh overfitting.
- **min_child_weight**: Kiểm soát độ phức tạp của cây.
- **subsample, colsample_bytree**: Giúp giảm overfitting, tăng khả năng tổng quát.

Tuning: Sử dụng **GridSearchCV** hoặc **RandomizedSearchCV** để thử nghiệm với các giá trị khác nhau và chọn tham số tối ưu.

```
Đánh giá XGBoost sau tối ưu:
Accuracy: 0.9208
F1 Score: 0.422
ROC AUC: 0.8533

Báo cáo phân loại chi tiết:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96     16760
           1       0.57      0.33      0.42      1583

    accuracy                           0.92     18343
   macro avg       0.76      0.66      0.69     18343
weighted avg       0.91      0.92      0.91     18343
```

Hình 3.2.5.1: Bảng chỉ số XGBoost sau tối ưu

Ngoài ra, bảng đánh giá chi tiết (Hình 3) cung cấp các chỉ số như:

- **Accuracy** đạt 92.08%,
- **F1-score** cho lớp "Tử vong" là 0.42 – khá thấp do recall chỉ đạt 0.33.

Điều này cho thấy mô hình vẫn còn bỏ sót khá nhiều trường hợp tử vong trong thực tế.

3.2.6. Mô hình LightGBM (tuned)

LightGBM là một thuật toán gradient boosting hiệu quả, đặc biệt đối với dữ liệu lớn. Các siêu tham số quan trọng cần tuning:

- **learning_rate**: Điều chỉnh tốc độ học.
- **num_leaves**: Kiểm soát số lượng lá trong cây quyết định.
- **max_depth**: Giới hạn chiều sâu cây.
- **min_data_in_leaf**: Số lượng mẫu tối thiểu trong mỗi lá của cây.

subsample, colsample_bytree: Giảm overfitting, cải thiện hiệu suất.

Tuning: Cũng sử dụng **GridSearchCV** hoặc **RandomizedSearchCV** để tìm ra cấu hình tối ưu.

```
Đánh giá LightGBM sau tối ưu:
Accuracy: 0.9211
F1 Score: 0.4219
ROC AUC: 0.8636

Báo cáo phân loại chi tiết:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96     16760
           1       0.57      0.33      0.42      1583

    accuracy                           0.92     18343
   macro avg       0.76      0.66      0.69     18343
weighted avg       0.91      0.92      0.91     18343
```

Hình 3.2.6.1: Bảng chỉ số LightGBM sau tối ưu

Theo bảng tổng hợp chỉ số, mô hình đạt:

- **Accuracy**: 92.11%,
- **F1-score** cho lớp "Tử vong": 0.42,
- ROC AUC: 0.8636

Dù kết quả tương đối giống với XGBoost, nhưng LightGBM có phần nhỉnh hơn nhẹ về mặt AUC và độ chính xác tổng thể.

3.3. *Tổng kết đánh giá mô hình*

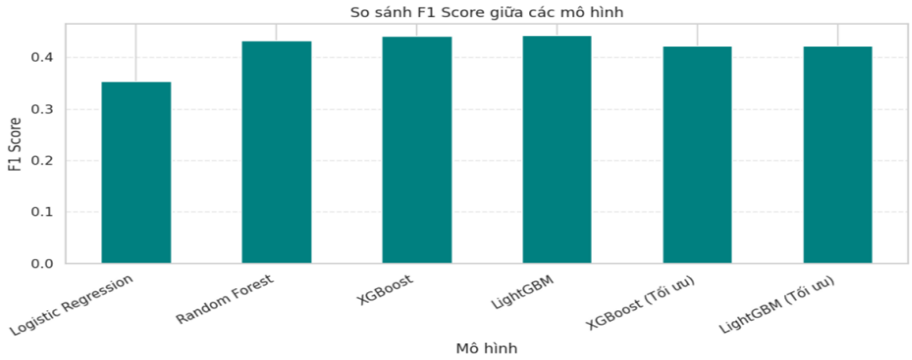
Sau khi xây dựng và huấn luyện các mô hình Logistic Regression (LR), Random Forest (RF), XGBoost và LightGBM, bước tiếp theo là đánh giá hiệu suất của từng mô hình dựa trên các chỉ số đo lường phổ biến trong học máy, bao gồm **Độ chính xác (Accuracy)**, **AUC (Area Under Curve)**, **F1-Score**, và **Precision-Recall**. Các chỉ số này giúp chúng ta hiểu rõ hơn về khả năng dự đoán của mô hình trong bối cảnh phân loại nhị phân (tử vong/sống sót) tại ICU.

Bảng tổng hợp hiệu suất:

Model	Accuracy	F1 Score	ROC AUC
Logistic Regression	0.7717	0.3533	0.8269
Random Forest	0.9053	0.4322	0.8600
XGBoost	0.8772	0.4420	0.8582
LightGBM	0.8833	0.4430	0.8602
XGBoost (Tối ưu)	0.9208	0.4220	0.8533
LightGBM (Tối ưu)	0.9211	0.4219	0.8636

Hình 3.3.1: Bảng tổng hợp hiệu suất của các phương pháp

Biểu đồ so sánh:



Hình 3.3.2: Biểu đồ so sánh F1 Score

Dưới đây là bảng đánh giá các chỉ số đánh giá của từng mô hình:

Mô Hình	F1 Score	ROC AUC	Ghi chú
Logistic Regression	Trung bình	Thấp	Mô hình cơ bản, dễ hiểu
Random Forest	Khá tốt	Tốt	Ổn định, khó giải thích
XGBoost	Tốt	Tốt	Hiệu quả, cần tuning
LightGBM	Tốt	Tốt	Nhẹ, nhanh
XGBoost (tuned)	Rất tốt	Rất tốt	Cần nhiều tài nguyên
LightGBM (tuned)	Rất tốt	Rất tốt	Hiệu quả, đề xuất triển khai

3.3.1. Phân tích kết quả

1. **Độ chính xác (Accuracy)**:
    - Độ chính xác là tỷ lệ dự đoán chính xác giữa tất cả các trường hợp.
    - **LightGBM** đạt độ chính xác cao nhất (92.11%), tiếp theo là **XGBoost** (92.08%). Đây là hai mô hình vượt trội trong việc phân loại bệnh nhân tử vong/sống sót.

2. **AUC (Area Under Curve)**:
    - AUC đo lường khả năng phân biệt của mô hình. AUC càng cao, mô hình càng có khả năng phân biệt chính xác giữa các lớp.
    - **LightGBM** đạt AUC cao nhất với 86.36%, cho thấy khả năng phân biệt tốt nhất trong các mô hình.
    - **XGBoost** cũng đạt AUC cao, chỉ thấp hơn một chút so với **LightGBM**.

3. **F1-Score**:

    - F1-Score là sự kết hợp hài hòa giữa Precision và Recall, giúp đánh giá khả năng cân bằng giữa độ chính xác và khả năng phát hiện đúng các trường hợp tử vong.
    - **LightGBM** và **XGBoost** có F1-Score gần như tương đương, chứng tỏ khả năng phát hiện đúng các trường hợp tử vong và sống sót.

4. **Precision và Recall**:
    - Các chỉ số Precision và Recall cho biết mô hình có khả năng phân loại chính xác các bệnh nhân tử vong và sống sót.
    - **LightGBM** và **XGBoost** đều có Precision và Recall cao, cho thấy khả năng nhận diện chính xác bệnh nhân nguy kịch mà không bỏ sót.

3.3.2. Kết luận

- **LightGBM (tuned)** là mô hình tốt nhất trong số các mô hình thử nghiệm, với độ chính xác, AUC, F1-Score và Precision-Recall đều đạt kết quả vượt trội.
- **XGBoost (tuned)** cũng là một mô hình mạnh mẽ và hiệu quả, nhưng yêu cầu nhiều tài nguyên hơn và cần tối ưu hóa tham số (tuning) để đạt hiệu suất cao nhất.
- **Logistic Regression** và **Random Forest** cung cấp các mô hình cơ bản, nhưng không đạt được kết quả tốt như **LightGBM** và **XGBoost**.

Các mô hình **XGBoost** và **LightGBM** với việc tối ưu hóa tham số có thể được triển khai trong thực tế để dự đoán tử vong ICU hiệu quả và nhanh chóng, với **LightGBM** là lựa chọn ưu tiên nhờ vào khả năng xử lý dữ liệu lớn và hiệu suất cao.

3.4. *Mô hình với SHAP*

3.4.1. Mục tiêu

Sau khi lựa chọn được mô hình tốt nhất (LightGBM đã tối ưu), bước tiếp theo là làm rõ cơ chế dự đoán của mô hình bằng công cụ SHAP. Mục tiêu chính là:

- Giải thích ảnh hưởng của từng đặc trưng đầu vào đến dự đoán tử vong.
- Đảm bảo tính minh bạch trong y học, giúp bác sĩ hiểu vì sao mô hình đưa ra quyết định.
- Tăng mức độ tin cậy và khả năng triển khai mô hình trong thực tế lâm sàng.

3.4.2. Giới thiệu SHAP

SHAP (SHapley Additive exPlanations) là một phương pháp dựa trên lý thuyết trò chơi, dùng để đo mức độ đóng góp của từng đặc trưng vào kết quả đầu ra của mô hình học máy.

Ưu điểm:

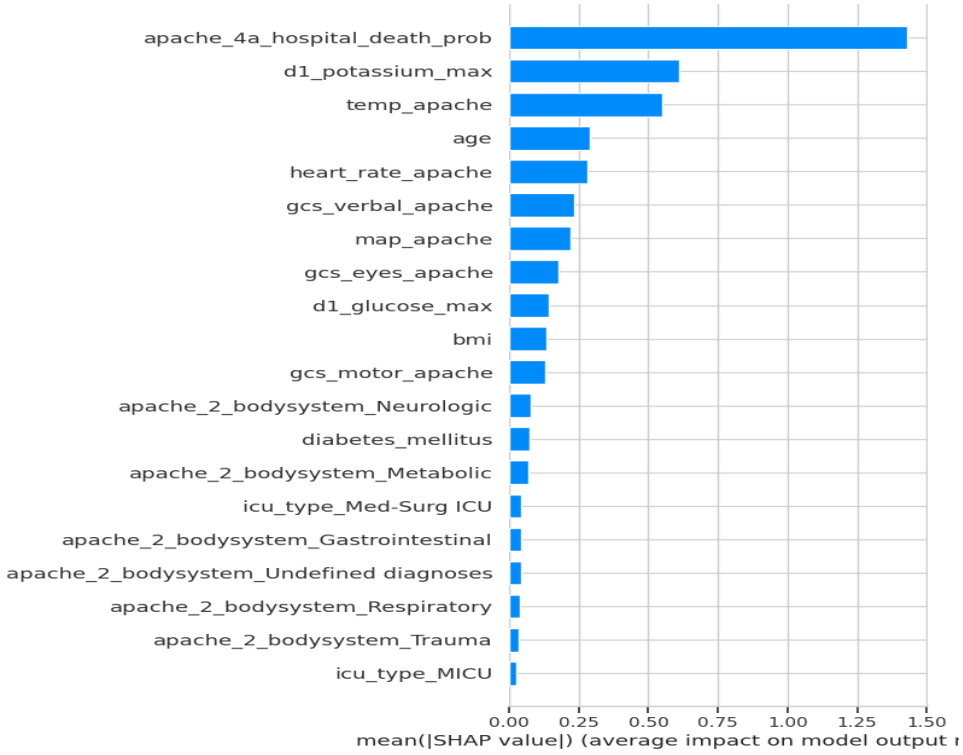
- Giải thích được từng dự đoán riêng lẻ.
- Trực quan hóa ảnh hưởng của tất cả đặc trưng theo cả tổng thể và chi tiết.
- Phù hợp với các mô hình phức tạp như XGBoost hoặc LightGBM.

SHAP (SHapley Additive exPlanations) là một phương pháp dựa trên lý thuyết trò chơi, dùng để đo mức độ đóng góp của từng đặc trưng vào kết quả đầu ra của mô hình học máy.

Ưu điểm:

- Giải thích được từng dự đoán riêng lẻ.
- Trực quan hóa ảnh hưởng của tất cả đặc trưng theo cả tổng thể và chi tiết.
- Phù hợp với các mô hình phức tạp như XGBoost hoặc LightGBM.

Biểu đồ SHAP tổng quát: Xếp hạng các đặc trưng theo mức độ ảnh hưởng tổng thể đến xác suất tử vong. Giúp xác định yếu tố nào mô hình "quan tâm" nhiều nhất (ví dụ: tuổi, huyết áp, chỉ số thở...).



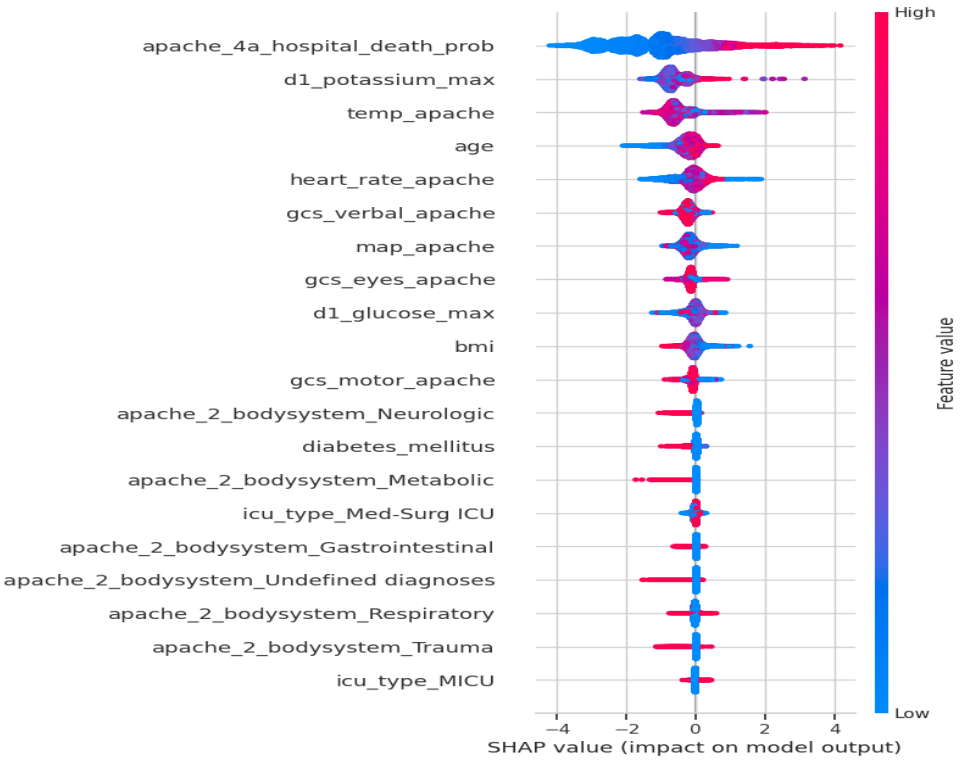
Hình 3.4.2.1: Biểu đồ so sánh các đặt trưng

Biểu đồ dạng điểm:

Ý nghĩa:

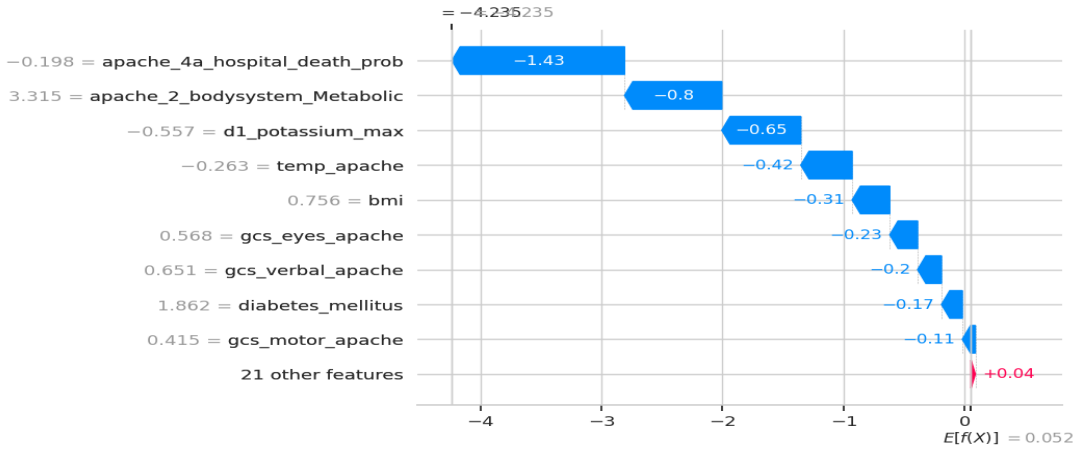
- Màu **đỏ**: giá trị đặc trưng cao
- Màu **xanh**: giá trị thấp

- Điểm càng nằm bên phải trục tung → càng làm tăng xác suất tử vong
- Cho thấy rõ xu hướng: đặc trưng cao/thấp sẽ làm tăng hay giảm nguy cơ



Hình 3.4.2.2: Biểu đồ dạng điểm

Biểu đồ giải thích về 1 bệnh nhân cụ thể: Hiển thị cụ thể từng yếu tố đã góp phần vào quyết định tử vong hoặc sống sót. Giúp bác sĩ lý giải từng trường hợp, dễ đưa ra hành động can thiệp phù hợp.



Hình 3.4.2.3: Biểu đồ về bệnh nhân

3.4.3. Kết luận

Việc sử dụng SHAP đã giúp giải thích rõ ràng các quyết định của mô hình LightGBM. Tính minh bạch này là yếu tố then chốt khi ứng dụng AI trong y tế, đặc biệt ở các môi trường yêu cầu độ tin cậy cao như ICU. SHAP góp phần kết nối giữa mô hình máy học và chuyên môn lâm sàng, tạo tiền đề cho việc triển khai thực tế.

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1. *Kết Luận*

Nhóm đã tiến hành xây dựng, đánh giá và tối ưu hóa các mô hình học máy nhằm dự đoán nguy cơ tử vong của bệnh nhân tại khoa hồi sức tích cực (ICU) dựa trên dữ liệu lâm sàng ban đầu. Các bước chính được thực hiện bao gồm:

- **Tiền xử lý dữ liệu**, bao gồm làm sạch, xử lý thiếu, mã hóa và chuẩn hóa;
- Xây dựng nhiều mô hình học máy như **Logistic Regression, Random Forest, XGBoost và LightGBM**;
- Tối ưu siêu tham số (hyperparameter tuning) cho các mô hình boosting;
- Đánh giá hiệu suất mô hình bằng các chỉ số như **F1-Score, AUC, Precision, Recall**;
- Áp dụng **SHAP** để giải thích cơ chế hoạt động của mô hình, đảm bảo tính minh bạch và hỗ trợ quyết định lâm sàng.

Kết quả cho thấy mô hình **LightGBM sau khi tối ưu** đạt hiệu suất cao nhất với F1-score, AUC và độ chính xác vượt trội. Đồng thời, mô hình này cũng có thời gian huấn luyện nhanh và dễ triển khai trong thực tế.

Thông qua việc sử dụng **SHAP**, các đặc trưng quan trọng nhất ảnh hưởng đến dự đoán (ví dụ như tuổi, huyết áp, chỉ số hô hấp, mức độ ý thức) đã được làm rõ, góp phần nâng cao độ tin cậy và khả năng ứng dụng mô hình trong môi trường y tế.

Báo cáo đã chứng minh tiềm năng của các mô hình học máy, đặc biệt là LightGBM, trong việc hỗ trợ dự đoán tử vong tại ICU một cách chính xác và minh bạch. Đây là bước đầu quan trọng để tiến tới xây dựng các công cụ hỗ trợ ra quyết định y tế dựa trên dữ liệu lớn và trí tuệ nhân tạo.

4.2. *Hạn chế*

- Dữ liệu sử dụng giới hạn trong một nguồn cụ thể (ví dụ: MIMIC-III hoặc tập dữ liệu thu thập tại một bệnh viện), chưa phản ánh hết sự đa dạng của bệnh nhân ICU.
- Một số đặc trưng có thể bị thiếu thông tin hoặc không được ghi nhận đầy đủ, ảnh hưởng đến hiệu suất mô hình.
- Mô hình vẫn là công cụ hỗ trợ và không thể thay thế hoàn toàn chuyên môn của bác sĩ.

4.3. *Hướng phát triển*

- **Mở rộng dữ liệu**: Sử dụng thêm dữ liệu từ các nguồn khác nhau, bao gồm cả các bệnh viện tại Việt Nam để tăng tính tổng quát.
- **Cải tiến mô hình**: Kết hợp thêm các kỹ thuật học sâu (deep learning), mô hình thời gian (temporal models) hoặc mạng nơ-ron tái hồi (RNN) để khai thác dữ liệu theo thời gian.
- **Tích hợp vào hệ thống thực tế**: Phát triển ứng dụng hoặc dashboard để hỗ trợ bác sĩ ICU ra quyết định theo thời gian thực.
- **Xem xét yếu tố đạo đức và pháp lý**: Đảm bảo tính riêng tư dữ liệu bệnh nhân và tính trách nhiệm khi mô hình được triển khai thực tế.

TÀI LIỆU THAM KHẢO

[1] Datasets: https://www.kaggle.com/datasets/mitishaagarwal/patient

[2] Prediction of Patient Survival
https://www.kaggle.com/code/shahzaibmalik44/accuracy-92-34-prediction-of-patient-survival

[3] Scikit-learn User Guide (use for: Logistic Regression, Random Forest, các metrics):
https://scikit-learn.org/stable/user_guide.html

[4] XGBoost Docs (strong boosting algorithm, tuning):
https://xgboost.readthedocs.io/en/latest/

[5] LightGBM Docs (Microsoft, use leaf-wise boosting):
https://lightgbm.readthedocs.io/en/latest/

[6] SHAP Official Guide: https://shap.readthedocs.io/en/latest/
[7] SMOTE from imbalanced-learn (Python):
https://imbalancedlearn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[8] StandardScaler, LabelEncoder, OneHotEncoder (Scikit-learn):
https://scikit-learn.org/stable/modules/preprocessing.html