**Bộ giáo dục và đào tạo**

**Trường Đại học Ngoại ngữ - Tin học TP.HCM**



**ĐỀ TÀI: DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG**

**ĐỒ ÁN MÔN HỌC: MÁY HỌC**

**Giảng viên hướng dẫn: Th.S Vũ Đình Ái**

**Sinh viên thực hiện:**

**21DH113556 Dương Tấn Đạt**

**21DH114404 Trương Huy Hoàng**

**21DH112239 Dịp Kim Yến**

**Tháng 11 Năm 2023**

**Bộ giáo dục và đào tạo**

**Trường Đại học Ngoại ngữ - Tin học TP.HCM**



**ĐỀ TÀI: DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG**

**ĐỒ ÁN MÔN HỌC: MÁY HỌC**

**Giảng viên hướng dẫn: Th.S Vũ Đình Ái**

**Sinh viên thực hiện:**

**21DH113556 Dương Tấn Đạt**

**21DH114404 Trương Huy Hoàng**

**21DH112239 Dịp Kim Yến**

**Tháng 11 Năm 2023**

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |
| --- | --- | --- |
| Nhiệm vụ | Tên thành viên | Đánh giá |
| Thu thập dữ liệu |  |  |
| Phân tích dữ liệu | **Dương Tấn Đạt**  **Trương Huy Hoàng**  **Dịp Kim Yến** |  |
| Xây dựng mô hình | **Dương Tấn Đạt**  **Trương Huy Hoàng**  **Dịp Kim Yến** |  |
| Cài đặt và demo | **Dương Tấn Đạt**  **Trương Huy Hoàng**  **Dịp Kim Yến** |  |

**Mục Lục**

[I. Tóm tắt 1](#_Toc151674881)

[II. Giới thiệu 1](#_Toc151674882)

[III. Xây dựng mô hình 2](#_Toc151674883)

[IV. Phân tích dữ liệu 2](#_Toc151674884)

[IV.1. Giới thiệu về dữ liệu 3](#_Toc151674885)

[IV.2. Trích chọn đặc trưng 5](#_Toc151674886)

[IV.3. Chuẩn hoá dữ liệu 7](#_Toc151674887)

[IV.4. Phân chia dữ liệu 19](#_Toc151674888)

[V. Huấn luyện mô hình 23](#_Toc151674889)

[V.1. Mô hình Logistic Regression 23](#_Toc151674890)

[V.2. Mô hình Random Forest 24](#_Toc151674891)

[V.3. Mô hình K-Nearest Neighbors 25](#_Toc151674892)

[V.4. Mô hình Naive Bayes 27](#_Toc151674893)

[VI. Đánh giá mô hình 28](#_Toc151674894)

[VII. Kết luận 36](#_Toc151674895)

[Tài liệu tham khảo (IEEE) 38](#_Toc151674896)

**Mục Lục Hình Ảnh**

[Hình IV. 1. Biểu đồ Boxplot chưa loại bỏ các giá trị ngoại biên 13](#_Toc151672600)

[Hình IV. 2 Biểu đồ Boxplot đã loại bỏ các giá trị ngoại biên 17](#_Toc151672601)

[Hình IV. 3. Biểu đồ histogram của tất cả các đặt trưng trong tập dữ liệu 18](#_Toc151672602)

[Hình IV. 4. Biểu đồ histogram của đặt trưng trong tập dữ liệu train 22](#_Toc151672603)

[Hình V. 1. Cross-Validated Method for Optimal k 27](#_Toc151672645)

[Hình VI. 1 Ma trận nhần lẫn của 4 mô hình 32](#_Toc151672657)

[Hình VI. 2. Biểu đồ ROC ( Đã loại bỏ đặc trưng) 35](#_Toc151672658)

# Tóm tắt

Bệnh tiểu đường đang trở thành một vấn đề sức khỏe quan trọng toàn cầu, và khả năng dự đoán nguy cơ mắc bệnh này đóng vai trò quan trọng trong việc phát hiện sớm và can thiệp kịp thời. Trong báo cáo này, chúng tôi tập trung vào ứng dụng máy học và sử dụng các mô hình hồi quy để dự đoán bệnh tiểu đường. Máy học là một lĩnh vực nghiên cứu mạnh mẽ trong việc phân tích dữ liệu và tạo ra các mô hình dự đoán. Trong ngữ cảnh này, phương pháp hồi quy trong máy học đang trở nên quan trọng, cho phép chúng ta ước tính giá trị đầu ra dựa trên dữ liệu đầu vào. Để sao sánh và chọn ra được thuật toán tối ưu trong việc dự đoán người có nguy cơ mắc bệnh tiểu đường, nghiên cứu này sẽ áp dụng một loạt các thuật toán máy học chính, bao gồm hồi quy Logistic, hồi quy Linear, Random Forest và Naïve Bayes. Mục tiêu của việc sử dụng nhiều thuật toán là để đánh giá và so sánh hiệu suất của mỗi thuật toán trong việc dự đoán bệnh tiểu đường. Kết quả này sẽ giúp xác định thuật toán nào cung cấp kết quả tốt nhất trong việc dự đoán bệnh tiểu đường. Sự kết hợp giữa các kỹ thuật máy học và phương pháp hồi quy sẽ cung cấp cơ sở cho việc cải thiện chẩn đoán và quản lý bệnh tiểu đường, giúp giảm thiểu nguy cơ và cải thiện chất lượng cuộc sống của những người mắc bệnh.

# Giới thiệu

Tiểu đường hay còn gọi là đái tháo đường là bệnh có tình trạng lượng đường trong máu luôn cao hơn mức bình thường do cơ thể bị thiếu hụt hoặc đề kháng với insulin, gây tình trạng rối loạn chuyển hóa đường trong máu. Đây là nguyên nhân cản trở cơ thể chuyển hóa các chất bột đường thành năng lượng, gây ra hiện tượng đường tích tụ tăng dần trong máu. Lâu ngày, sự tích tụ này khiến lượng đường trong máu thường xuyên ở mức cao. Trên toàn thế giới có 415 triệu người lớn (độ tuổi 20-79) tương đương 1 trong 11 người lớn đang sống với bệnh đái tháo đường trong năm 2015. Dự đoán vào năm 2040, con số này sẽ tăng tới khoảng 642 triệu người, hay nói cách khác 1 người trong 10 người lớn sẽ có bệnh đái tháo đường. Tại Việt Nam, vào năm 2015 đã có 3,5 triệu người mắc bệnh báo cáo của Hiệp hội đái tháo đường thế giới IDF Diabetes Atlas, và con số này được dự báo sẽ tăng lên 6,1 triệu vào năm 2040. Các biểu hiện của bệnh đái tháo đường giai đoạn đầu rất khó xác định, do không điển hình và dễ nhầm lẫn với triệu chứng các bệnh khác. Việc phát hiện sớm, điều trị bệnh kịp thời và liên tục, sẽ giúp giảm nguy cơ biến chứng nguy hiểm như: Nhồi máu cơ tim, tai biến mạch máu não, bệnh lý võng mạc gây mù mắt, bệnh lý thận gây suy thận, bệnh lý mạch máu ngoại vi dẫn đến đoạn chi và các biến chứng nghiêm trọng… Với sự phát triển của trí tuệ nhân tạo và đặc biệt là máy học. Ngày nay việc áp dụng các thuật toán máy học vào quá trình phân tích và dự đoán phát hiện bệnh tiểu đường ở giai đoạn sớm sẽ giúp ích rất nhiều trong quá trình điều trị. Và trong báo cáo này nhóm sẽ áp dụng các thuật toán của máy học như hồi quy Logistic, Random Forest, Naive Bayes và Support Vector Machine để tiến hành dự đoán bệnh tiểu đường thông qua tập dữ liệu có sẵn.

# Xây dựng mô hình

Từ dữ liệu có sẵn sẽ được lưu thành file CSV, sau đó sẽ được chia thành 2 tập dữ liệu có tỉ lệ 70% cho tập huấn luyện và 30% cho tập kiểm tra , kế tiếp chương trình sẽ áp dụng các thuật toán như hồi quy Logistic, Random Forest, Naïve Bayes và K-Nearest Neighborstiến hành phân lớp thông qua tập huấn luyện. Cuối cùng sẽ dự đoán và đánh giá mô hình thông qua tập kiểm tra.

# Phân tích dữ liệu

A diagram of a logistic process

Description automatically generated

Hình . Sơ đồ huấn luyện mô hình

## Giới thiệu về dữ liệu

Dữ liệu cho đề tài "Dự đoán bệnh tiểu đường" là một bộ dữ liệu chứa thông tin về các biến độc lập (các yếu tố) và biến mục tiêu (kết quả) để dự đoán nguy cơ mắc bệnh tiểu đường. Bộ dữ liệu này được lấy từ kaggle sử dụng phát triển các mô hình học máy để dự đoán xác suất mắc bệnh tiểu đường dựa trên các yếu tố khác nhau. Dữ liệu này thường được thu thập thông qua các phương pháp điều tra lâm sàng hoặc dựa trên các bệnh nhân có sẵn trong các cơ sở y tế. Dữ liệu có thể được nhập tay từ hồ sơ bệnh nhân hoặc thu thập tự động thông qua các hệ thống hồ sơ điện tử.

Các trường thông tin của dữ liệu: Dữ liệu thường bao gồm các trường thông tin sau:

* **Gender (giới tính):** Đề cập đến giới tính sinh học của cá nhân, có thể có tác động đến khả năng mắc bệnh tiểu đường của họ. Có ba loại trong đó ‘Male’, Female’ và ‘Other’.
* **Age (tuổi tác):** Là một yếu tố quan trọng vì bệnh tiểu đường thường được chẩn đoán ở người lớn tuổi. Độ tuổi nằm trong khoảng từ 0-80 trong tập dữ liệu của chúng tôi.
* **Hypertension (tăng huyết áp):** Là một tình trạng bệnh lý trong đó huyết áp trong động mạch tăng cao liên tục. Nó có giá trị 0 hoặc 1 trong đó 0 cho biết họ không bị tăng huyết áp và với 1 có nghĩa là họ bị tăng huyết áp.
* **Heart\_disease (bệnh tim):** Là một tình trạng y tế khác có liên quan đến việc tăng nguy cơ phát triển bệnh tiểu đường. Nó có giá trị 0 hoặc 1 trong đó 0 cho biết họ không mắc bệnh tim và với 1 có nghĩa là họ mắc bệnh tim.
* **Smoking\_history (tiền sử hút thuốc):** Đây cũng được coi là một yếu tố nguy cơ của bệnh tiểu đường và có thể làm trầm trọng thêm các biến chứng liên quan đến bệnh tiểu đường. Trong tập dữ liệu của chúng tôi, chúng tôi có 5 loại tức là ‘never’, 'No Info', 'current', 'former', 'ever' và 'not current'.
* **BMI (Body Mass Index):** Là thước đo lượng mỡ trong cơ thể dựa trên cân nặng và chiều cao. Giá trị BMI cao hơn có liên quan đến nguy cơ mắc bệnh tiểu đường cao hơn. Phạm vi chỉ số BMI trong tập dữ liệu là từ 10,16 đến 71,55. BMI dưới 18,5 là thiếu cân, 18,5-24,9 là bình thường, 25-29,9 là thừa cân và từ 30 trở lên là béo phì.
* **HbA1c\_level (Mức HbA1c):** Hemoglobin A1c là thước đo lượng đường trong máu trung bình của một người trong 2-3 tháng qua. Mức độ cao hơn cho thấy nguy cơ phát triển bệnh tiểu đường cao hơn. Hầu hết mức HbA1c trên 6,5% cho thấy bệnh tiểu đường.
* **Blood\_glucose\_level (mức đường huyết):** Đề cập đến lượng glucose trong máu tại một thời điểm nhất định. Mức đường huyết cao là dấu hiệu chính của bệnh tiểu đường.
* **Diabetes (bệnh tiểu đường):** Là biến mục tiêu được dự đoán, với giá trị 1 biểu thị sự hiện diện của bệnh tiểu đường và 0 biểu thị nguy cơ mắc bệnh.

Để đọc dữ liệu từ tệp CSV chúng ta sẽ sử dụng thư viện pandas và sử dụng thư viện matplotlib để trực quan hoá dữ liệu bằng cách vẽ biểu đồ phân phối cho các biến. Điều này giúp chúng ta có cái nhìn tổng quan về dữ liệu trước khi bắt đầu xây dựng mô hình dự đoán bệnh tiểu đường.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/TH\_MH/Data\_csv/diabetes\_prediction\_dataset.csv')

df

|  | **gender** | **age** | **Hyperten**  **sion** | **heart\_**  **disease** | **Smokin\_history** | **bmi** | **HbA1c\_level** | **blood\_**  **glucose\_level** | **diabetes** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Female | 80.0 | 0 | 1 | never | 25.19 | 6.6 | 140 | 0 |
| **1** | Female | 54.0 | 0 | 0 | No Info | 27.32 | 6.6 | 80 | 0 |
| **2** | Male | 28.0 | 0 | 0 | never | 27.32 | 5.7 | 158 | 0 |
| **3** | Female | 36.0 | 0 | 0 | current | 23.45 | 5.0 | 155 | 0 |
| **4** | Male | 76.0 | 1 | 1 | current | 20.14 | 4.8 | 155 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99995** | Female | 80.0 | 0 | 0 | No Info | 27.32 | 6.2 | 90 | 0 |
| **99996** | Female | 2.0 | 0 | 0 | No Info | 17.37 | 6.5 | 100 | 0 |
| **99997** | Male | 66.0 | 0 | 0 | former | 27.83 | 5.7 | 155 | 0 |
| **99998** | Female | 24.0 | 0 | 0 | never | 35.42 | 4.0 | 100 | 0 |
| **99999** | Female | 57.0 | 0 | 0 | current | 22.43 | 6.6 | 90 | 0 |

100000 rows × 9 columns

Để hiểu cấu trúc dữ liệu, kiểm tra giá trị thiếu, đánh giá tình trạng dữ liệu, ta sẽ sử dụng phương thức info(). Xem số lượng dòng, cột và kiểu dữ liệu, phát hiện giá trị thiếu trong các cột sẽ giúp chuẩn bị cho việc xử lý các giá trị trống một cách hiệu quả trước khi phân tích. Có cái nhìn tổng quan về tình trạng của dữ liệu, giúp xác định cần thực hiện bước tiền xử lý nào để làm sạch dữ liệu trước khi phân tích.

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 100000 entries, 0 to 99999

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 gender 100000 non-null object

1 age 100000 non-null float64

2 hypertension 100000 non-null int64

3 heart\_disease 100000 non-null int64

4 smoking\_history 100000 non-null object

5 bmi 100000 non-null float64

6 HbA1c\_level 100000 non-null float64

7 blood\_glucose\_level 100000 non-null int64

8 diabetes 100000 non-null int64

dtypes: float64(3), int64(4), object(2)

memory usage: 6.9+ MB

## Trích chọn đặc trưng

Dữ liệu về bệnh tiểu đường là một tập hợp thông tin đa dạng về các yếu tố liên quan đến bệnh này, bao gồm thông tin về giới tính, độ tuổi, huyết áp, tiền sử bệnh tim mạch, lịch sử hút thuốc, chỉ số BMI, mức độ HbA1c và mức độ đường trong máu. Mỗi đặc trưng này cung cấp thông tin quan trọng về sức khỏe và yếu tố nguy cơ mắc bệnh tiểu đường.

Chúng ta không cần trích chọn đặc trưng trong trường hợp này là vì mỗi thông tin đều có thể đóng góp vào việc dự đoán bệnh tiểu đường một cách quan trọng. Tuổi có thể là một yếu tố quan trọng trong việc đánh giá nguy cơ mắc bệnh, trong khi chỉ số BMI có thể liên quan mật thiết đến việc đo lường tình trạng cơ thể. Mức độ đường trong máu và HbA1c cũng cung cấp thông tin chi tiết về kiểm soát đường huyết, một yếu tố quan trọng trong quản lý bệnh lý. Điều này sẽ mang lại những lợi ích:

* Đa dạng thông tin quan trọng: Các thông tin về sức khỏe, lối sống và yếu tố di truyền đều có thể đóng góp vào việc dự đoán nguy cơ mắc bệnh tiểu đường. Không trích chọn đặc trưng giúp giữ lại đầy đủ thông tin để mô hình có thể tận dụng mối quan hệ phức tạp giữa các yếu tố này.
* Tiềm năng ẩn của thông tin: Đôi khi, các thông tin mà ban đầu có vẻ không quan trọng có thể liên quan mật thiết đến bệnh lý. Việc loại bỏ các đặc trưng này có thể làm mất đi thông tin quan trọng trong việc dự đoán bệnh tiểu đường.
* Tránh mất mát thông tin: Trích chọn đặc trưng có thể dẫn đến mất mát thông tin quan trọng. Trong trường hợp bệnh tiểu đường, việc giữ lại tất cả thông tin có thể giúp mô hình học máy có cái nhìn tổng quan và toàn diện hơn về bệnh lý.
* Khả năng mô hình hóa tốt hơn: Một số thuật toán học máy như mạng nơ-ron sâu (deep neural networks) có thể học và tận dụng thông tin từ nhiều đặc trưng khác nhau một cách hiệu quả, do đó không cần phải trích chọn đặc trưng một cách cẩn thận.

Vì vậy, việc giữ lại tất cả các đặc trưng này giúp mô hình học máy hiểu rõ hơn về mối quan hệ phức tạp giữa chúng và bệnh tiểu đường. Điều này cung cấp cho mô hình cái nhìn toàn diện hơn về các yếu tố có thể ảnh hưởng đến bệnh lý và giúp trong việc dự đoán chính xác nguy cơ mắc bệnh, cũng như đưa ra các khuyến nghị quản lý sức khỏe hiệu quả.

## Chuẩn hoá dữ liệu

Bước tiếp theo cần thực hiện để chuẩn bị dữ liệu cho máy học là làm sạch dữ liệu đó. Làm sạch dữ liệu liên quan đến việc tìm và sửa lỗi, sự không nhất quán và các giá trị bị thiếu. Có một số cách tiếp cận để làm điều đó:

* Xử lý dữ liệu bị thiếu  
  Thiếu giá trị là một vấn đề phổ biến trong học máy. Nó có thể được xử lý bằng cách quy nạp (nghĩ: điền các giá trị còn thiếu bằng dữ liệu dự đoán hoặc ước tính), nội suy (lấy các giá trị còn thiếu từ các điểm dữ liệu xung quanh) hoặc xóa (chỉ cần xóa các hàng hoặc cột có giá trị bị thiếu khỏi tập dữ liệu.)
* Xử lý ngoại lệ  
  Ngoại lệ là các điểm dữ liệu khác biệt đáng kể so với phần còn lại của tập dữ liệu. Các ngoại lệ có thể xảy ra do lỗi đo lường, lỗi nhập dữ liệu hoặc đơn giản vì chúng đại diện cho các quan sát bất thường hoặc cực đoan. Ví dụ, trong một bộ dữ liệu về lương của nhân viên, một ngoại lệ có thể là một nhân viên kiếm được nhiều hơn hoặc ít hơn đáng kể so với những người khác. Các ngoại lệ có thể được xử lý bằng cách loại bỏ, biến đổi chúng để giảm tác động của chúng, chiến thắng (nghĩ: thay thế các giá trị cực đoan bằng các giá trị gần nhất nằm trong phạm vi phân phối thông thường) hoặc coi chúng là một loại dữ liệu riêng biệt.
* Loại bỏ trùng lặp  
  Một bước khác trong quá trình chuẩn bị dữ liệu cho máy học là loại bỏ các dữ liệu trùng lặp. Các bản sao không chỉ làm sai lệch các dự đoán ML mà còn lãng phí dung lượng lưu trữ và tăng thời gian xử lý, đặc biệt là trong các bộ dữ liệu lớn. Để loại bỏ các bản sao, các nhà khoa học dữ liệu sử dụng nhiều kỹ thuật nhận dạng trùng lặp (như khớp chính xác, khớp mờ, băm hoặc liên kết bản ghi). Sau khi được xác định, chúng có thể được loại bỏ hoặc hợp nhất. Tuy nhiên, trong các bộ dữ liệu không cân bằng, trên thực tế, các bản sao có thể được hoan nghênh để đạt được phân phối bình thường.
* Xử lý dữ liệu không liên quan  
  Dữ liệu không liên quan đề cập đến dữ liệu không hữu ích hoặc không thể áp dụng để giải quyết vấn đề. Xử lý dữ liệu không liên quan có thể giúp giảm nhiễu và cải thiện độ chính xác của dự đoán. Để xác định dữ liệu không liên quan, các nhóm dữ liệu sử dụng các kỹ thuật như phân tích thành phần chính, phân tích tương quan hoặc đơn giản là dựa vào kiến thức miền của họ. Sau khi được xác định, các điểm dữ liệu đó sẽ bị xóa khỏi tập dữ liệu.
* Xử lý dữ liệu không chính xác  
  Chuẩn bị dữ liệu cho học máy cũng phải bao gồm việc xử lý dữ liệu không chính xác và sai sót. Các kỹ thuật phổ biến để xử lý dữ liệu đó bao gồm chuyển đổi dữ liệu (thay đổi dữ liệu để dữ liệu đáp ứng các tiêu chí đã đặt) hoặc xóa hoàn toàn các điểm dữ liệu không chính xác.
* Xử lý dữ liệu mất cân bằng  
  Tập dữ liệu không cân bằng là tập dữ liệu trong đó số điểm dữ liệu trong một lớp thấp hơn đáng kể so với số điểm dữ liệu trong lớp khác. Điều này có thể dẫn đến một mô hình thiên lệch ưu tiên cho tầng lớp đa số, trong khi bỏ qua tầng lớp thiểu số. Để giải quyết vấn đề, các nhóm dữ liệu có thể sử dụng các kỹ thuật như lấy mẫu lại (lấy mẫu quá mức của lớp thiểu số hoặc lấy mẫu dưới lớp đa số để cân bằng việc phân phối dữ liệu), tạo dữ liệu tổng hợp (tạo điểm dữ liệu bổ sung cho lớp thiểu số một cách tổng hợp), chi phí -học tập nhạy cảm (gán trọng số cao hơn cho lớp thiểu số trong quá trình đào tạo), học tập đồng bộ (kết hợp nhiều mô hình được đào tạo trên các tập hợp con dữ liệu khác nhau bằng các thuật toán khác nhau) và các mô hình khác.

Các hoạt động này giúp đảm bảo rằng dữ liệu đào tạo là chính xác, đầy đủ và nhất quán. Mặc dù là một thành tích lớn nhưng vẫn chưa đủ để tạo ra một mô hình ML đáng tin cậy. Vì vậy, bước tiếp theo trong hành trình chuẩn bị dữ liệu cho máy học liên quan đến việc đảm bảo các điểm dữ liệu trong tập dữ liệu huấn luyện tuân thủ các quy tắc và tiêu chuẩn cụ thể. Và giai đoạn đó trong quy trình [quản lý dữ liệu](https://itrexgroup.com/services/data-management/?ref=hackernoon.com) được gọi là chuyển đổi dữ liệu.

#Loại bỏ dữ liệu trùng lắp :

df = df.drop\_duplicates()

#Loại bỏ dữ liệu bị null :

df = df.dropna()

df

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **gender** | **age** | **hypertension** | **heart\_disease** | **smoking\_history** | **bmi** | **HbA1c\_level** | **blood\_glucose\_level** | **diabetes** |  |
| **0** | Female | 80.0 | 0 | 1 | never | 25.19 | 6.6 | 140 | 0 |
| **1** | Female | 54.0 | 0 | 0 | No Info | 27.32 | 6.6 | 80 | 0 |
| **2** | Male | 28.0 | 0 | 0 | never | 27.32 | 5.7 | 158 | 0 |
| **3** | Female | 36.0 | 0 | 0 | current | 23.45 | 5.0 | 155 | 0 |
| **4** | Male | 76.0 | 1 | 1 | current | 20.14 | 4.8 | 155 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99994** | Female | 36.0 | 0 | 0 | No Info | 24.60 | 4.8 | 145 | 0 |
| **99996** | Female | 2.0 | 0 | 0 | No Info | 17.37 | 6.5 | 100 | 0 |
| **99997** | Male | 66.0 | 0 | 0 | former | 27.83 | 5.7 | 155 | 0 |
| **99998** | Female | 24.0 | 0 | 0 | never | 35.42 | 4.0 | 100 | 0 |
| **99999** | Female | 57.0 | 0 | 0 | current | 22.43 | 6.6 | 90 | 0 |

96146 rows × 9 columns

Sau khi loại bỏ các giá trị null và giá trị trùng lắp, ta sẽ kiểm tra lại thông tin của tập dữ liệu một lần nữa.

Hiển thị thông tin tổng quan của dữ liệu:

df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 96146 entries, 0 to 99999

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 gender 96146 non-null object

1 age 96146 non-null float64

2 hypertension 96146 non-null int64

3 heart\_disease 96146 non-null int64

4 smoking\_history 96146 non-null object

5 bmi 96146 non-null float64

6 HbA1c\_level 96146 non-null float64

7 blood\_glucose\_level 96146 non-null int64

8 diabetes 96146 non-null int64

dtypes: float64(3), int64(4), object(2)

memory usage: 7.3+ MB

Vì tập dữ liệu có các đặc trưng dạng chuỗi sẽ gây khó khăn trong việc xây dựng mô hình. Chuyển đổi dữ liệu dạng chuỗi thành dạng số bằng cách sử dụng LabelEncoder để mô hình học máy có thể hiểu được:

#Gọi thư viện LabelEncoder :

from sklearn.preprocessing import LabelEncoder

#Khởi tạo đối tượng :

label\_encoder = LabelEncoder()

#Lọc ra dữ liệu dạng chuỗi và lấy tên các cột trong có dữ liệu dạng chuỗi :

data\_string\_cols = df.select\_dtypes(exclude=[np.number]).columns

print(data\_string\_cols)

print(df[data\_string\_cols].head())

Index(['gender', 'smoking\_history'], dtype='object')

gender smoking\_history

0 Female never

1 Female No Info

2 Male never

3 Female current

4 Male current

#Dùng phương thức fit\_transform của đối tượng label\_encoder đã khai báo ở trên :

#Để thực hiện việc chuẩn hoá dữ liệu dạng chuỗi thành số :

for col in data\_string\_cols:

df[col] = label\_encoder.fit\_transform(df[col])

df

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **gender** | **age** | **hypertension** | **heart\_disease** | **smoking\_history** | **bmi** | **HbA1c\_level** | **blood\_glucose\_level** | **diabetes** |  |
| **0** | 0 | 80.0 | 0 | 1 | 4 | 25.19 | 6.6 | 140 | 0 |
| **1** | 0 | 54.0 | 0 | 0 | 0 | 27.32 | 6.6 | 80 | 0 |
| **2** | 1 | 28.0 | 0 | 0 | 4 | 27.32 | 5.7 | 158 | 0 |
| **3** | 0 | 36.0 | 0 | 0 | 1 | 23.45 | 5.0 | 155 | 0 |
| **4** | 1 | 76.0 | 1 | 1 | 1 | 20.14 | 4.8 | 155 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99994** | 0 | 36.0 | 0 | 0 | 0 | 24.60 | 4.8 | 145 | 0 |
| **99996** | 0 | 2.0 | 0 | 0 | 0 | 17.37 | 6.5 | 100 | 0 |
| **99997** | 1 | 66.0 | 0 | 0 | 3 | 27.83 | 5.7 | 155 | 0 |
| **99998** | 0 | 24.0 | 0 | 0 | 4 | 35.42 | 4.0 | 100 | 0 |
| **99999** | 0 | 57.0 | 0 | 0 | 1 | 22.43 | 6.6 | 90 | 0 |

96146 rows × 9 columns

Bay giờ thì dữ liệu đã được chuyển hết về dạng số. Tiếp theo ta sẽ kiểm tra giá trị ngoại biên và loại bỏ chúng.

Vẽ biểu đồ boxplot cho tất cả các đặc trưng để phát hiện giá trị ngoại biên.

#Thực hiện kiểm tra các giá trị ngoại biên và loại bỏ nó :

#Dùng seaborn để xem thử các giá trị ngoại biên :

#Tạo lưới subplot 1 hàng và 2 cột :

plt.figure(figsize=(16, 8))

#Biểu đồ Boxplot của df :

sns.boxplot(data=df)

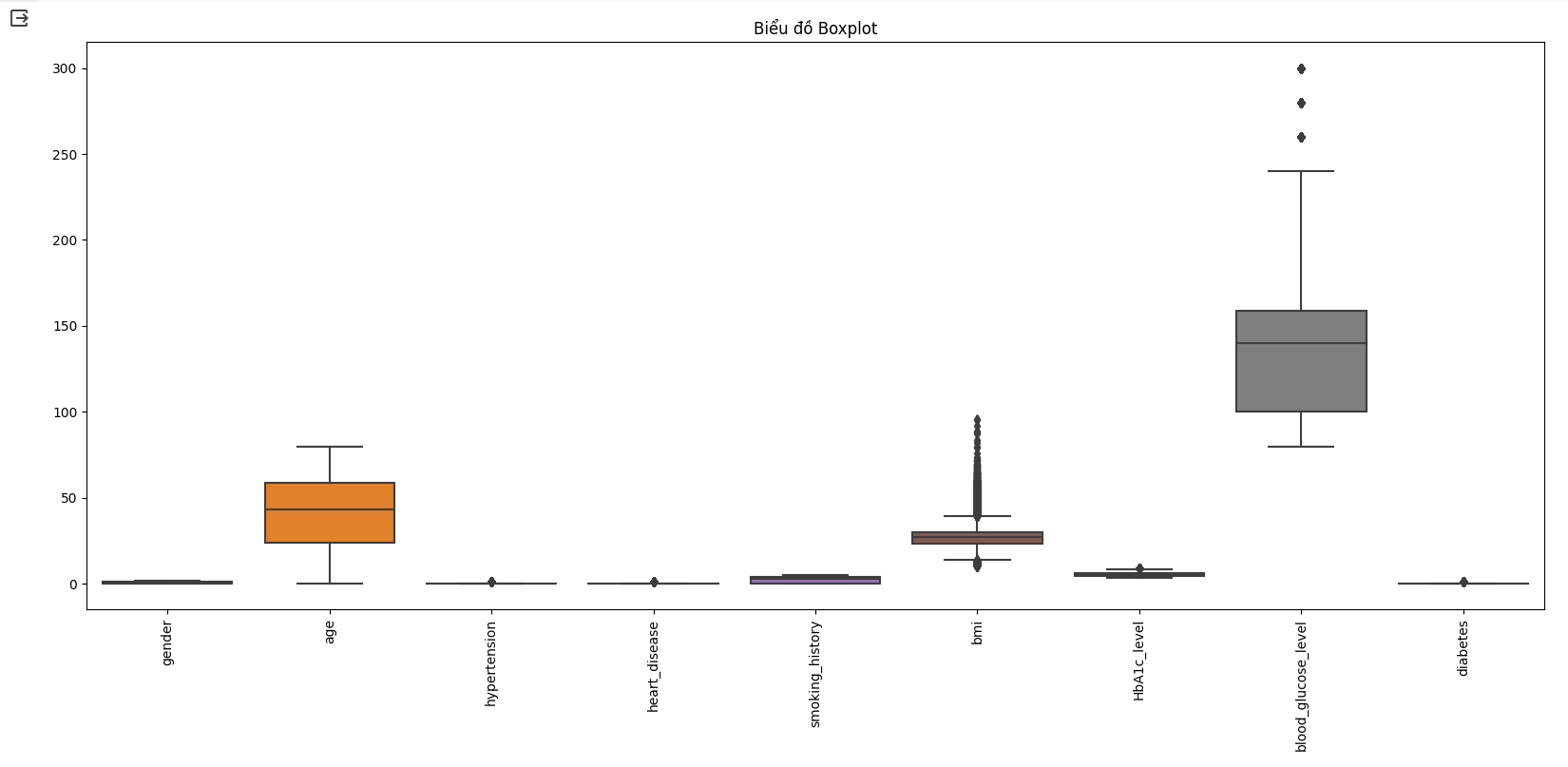
plt.xticks(range(len(df.columns)), df.columns, rotation=90)

plt.title("Biểu đồ Boxplot")

#Hiển thị biểu đồ :

plt.tight\_layout()

plt.show()

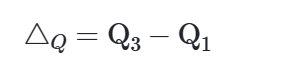


Hình IV. 1. Biểu đồ Boxplot chưa loại bỏ các giá trị ngoại biên

Đã thấy được các giá trị ngoại biên bây giờ chúng ta cần phải xử lý và lại bỏ chúng.

Đầu tiên tính toán khoảng từ phân vị (IQR) là một đại lượng cho biết mức độ phân tán của nửa giữa mẫu số liệu.

Nói cách khác, khoảng tứ phân vị là khoảng cách từ tứ phân vị thứ nhất (Q1)đến tứ phân vị thứ ba (Q3).



Cách tính khoảng tứ phân vị:

**Bước 1:** Sắp xếp mẫu số liệu theo thứ tự không giảm.

**Bước 2:** Tìm trung vị. Nếu số các số liệu của mẫu số liệu là số lẻ thì số liệu đứng chính giữa mẫu là trung vị. Nếu số các số liệu của mẫu số liệu là số chẵn thì trung vị là trung bình cộng của hai số liệu đứng chính giữa mẫu.

**Bước 3:** Tìm tứ phân vị thứ nhất (Q1). Tứ phân vị thứ nhất là trung vị của dãy số liệu nằm bên trái trung vị của mẫu (không bao gồm trung vị).

**Bước 4:** Tìm tứ phân vị thứ ba (Q3). Tứ phân vị thứ ba là trung vị của dãy số liệu nằm bên phải trung vị của mẫu (không bao gồm trung vị).

**Bước 5:** Khoảng tứ phân vị sẽ bằng hiệu Q3  - Q1 .

#Phân vị thứ nhất tương ứng với giá trị ở vị trí thứ 25% của mẫu dữ liệu :

q1 = df.quantile(0.25)

#Phân vị thứ ba tương ứng với giá trị ở vị trí thứ 75% của mẫu dữ liệu :

q3 = df.quantile(0.75)

#Tính IQR (Khoảng tứ phân vị) :

iqr = q3 - q1

iqr

gender 1.00

age 35.00

hypertension 0.00

heart\_disease 0.00

smoking\_history 4.00

bmi 6.46

HbA1c\_level 1.40

blood\_glucose\_level 59.00

diabetes 0.00

dtype: float64

in ra các giá trị không có sự biến đổi giữa phân vị thứ nhất (Q1) và phân vị thứ ba (Q3) của đặc trưng đó

df\_hypertension = df['hypertension'].unique()

df\_heart = df['heart\_disease'].unique()

df\_diabetes = df['diabetes'].unique()

print(df\_hypertension)

print(df\_heart)

print(df\_diabetes)

[0 1]

[1 0]

[0 1]

Tăng giá trị IQR (khoảng tứ phân vị) của 3 đặc trưng trên.

iqr['heart\_disease'] = iqr['heart\_disease'] + 1

iqr['hypertension'] = iqr['hypertension'] + 1

iqr['diabetes'] = iqr['diabetes'] + 1

iqr

gender 1.00

age 35.00

hypertension 1.00

heart\_disease 1.00

smoking\_history 4.00

bmi 6.46

HbA1c\_level 1.40

blood\_glucose\_level 59.00

diabetes 1.00

dtype: float64

Loại bỏ các giá trị biên

#Tính biên dưới và biên trên :

bien\_duoi = q1 - 1.5\*iqr

bien\_tren = q3 + 1.5\*iqr

#Loại bỏ các giá trị nằm phía dưới biên dưới hoặc nằm phía trên biên trên :

#Dùng mệnh đề phũ đinh ~ any để loại bỏ :

df\_no\_outlier = df[~((df < bien\_duoi) |

(df > bien\_tren))

.any(axis = 1)]

df\_no\_outlier

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **gender** | **age** | **hypertension** | **heart\_disease** | **smoking\_history** | **bmi** | **HbA1c\_level** | **blood\_glucose\_level** | **diabetes** |  |
| **0** | 0 | 80.0 | 0 | 1 | 4 | 25.19 | 6.6 | 140 | 0 |
| **1** | 0 | 54.0 | 0 | 0 | 0 | 27.32 | 6.6 | 80 | 0 |
| **2** | 1 | 28.0 | 0 | 0 | 4 | 27.32 | 5.7 | 158 | 0 |
| **3** | 0 | 36.0 | 0 | 0 | 1 | 23.45 | 5.0 | 155 | 0 |
| **4** | 1 | 76.0 | 1 | 1 | 1 | 20.14 | 4.8 | 155 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **99994** | 0 | 36.0 | 0 | 0 | 0 | 24.60 | 4.8 | 145 | 0 |
| **99996** | 0 | 2.0 | 0 | 0 | 0 | 17.37 | 6.5 | 100 | 0 |
| **99997** | 1 | 66.0 | 0 | 0 | 3 | 27.83 | 5.7 | 155 | 0 |
| **99998** | 0 | 24.0 | 0 | 0 | 4 | 35.42 | 4.0 | 100 | 0 |
| **99999** | 0 | 57.0 | 0 | 0 | 1 | 22.43 | 6.6 | 90 | 0 |

88195 rows × 9 columns

Hiển thị lại biểu đồ boxplot đã loại bỏ các giá trị ngoại biên

#Tạo lưới subplot 1 hàng và 2 cột :

plt.figure(figsize=(16, 8))

#Biểu đồ Boxplot cho df\_1 :

sns.boxplot(data=df\_no\_outlier)

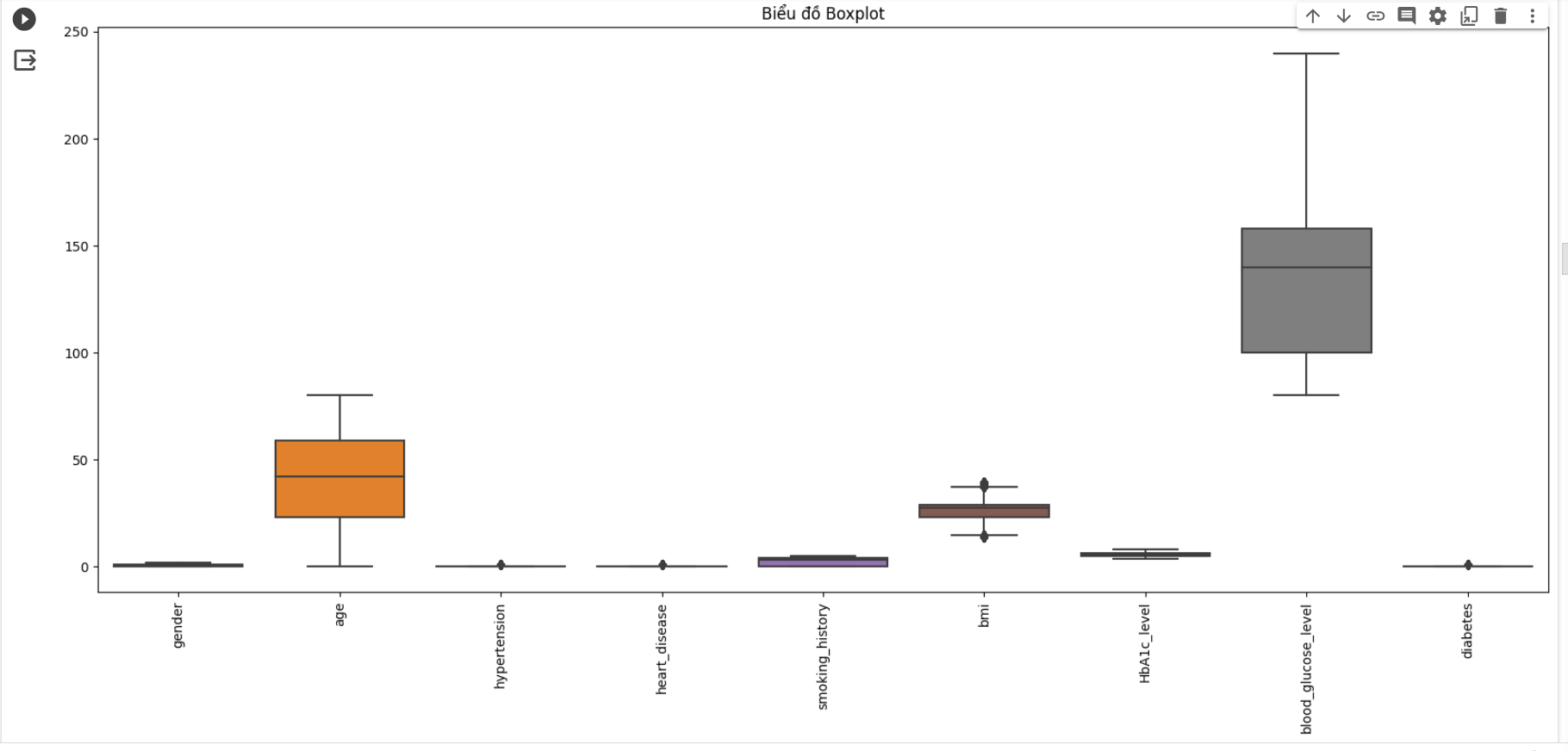
plt.xticks(range(len(df\_no\_outlier.columns)), df\_no\_outlier.columns, rotation=90)

plt.title("Biểu đồ Boxplot")

#Hiển thị biểu đồ :

plt.tight\_layout()

plt.show()



Hình IV. 2 Biểu đồ Boxplot đã loại bỏ các giá trị ngoại biên

Vẽ biểu đồ phân phối xác suất (histogram và đường phân phối chuẩn - KDE) hiển thị độ lệch, phương sai và giá trị ngoại biên của dữ liệu

#Dùng biểu đồ phân phối xác suất để xem sự phân phối tần suất của từng đặc trưng :

#Số lượng đặc trưng cần vẽ biểu đồ :

number\_features = len(df\_no\_outlier.columns)

#Thiết lập kích thước grid cho subplot :

#Tạo một grid có 2 cột (có thể thay đổi số cột tùy theo số đặc trưng) :

cols = 2

rows = number\_features // cols + 1

#Tạo một figure với kích thước phù hợp với số lượng biểu đồ :

plt.figure(figsize=(16, 16))

#Lặp qua từng đặc trưng và vẽ biểu đồ histogram và đường phân phối chuẩn (KDE) :

for i, feature in enumerate(df\_no\_outlier.columns):

plt.subplot(rows, cols, i+1)

sns.histplot(data=df\_no\_outlier, x=feature, kde=True)

sns.kdeplot(df\_no\_outlier[feature], color='black', linestyle='-', label='Đường cong phân phối')

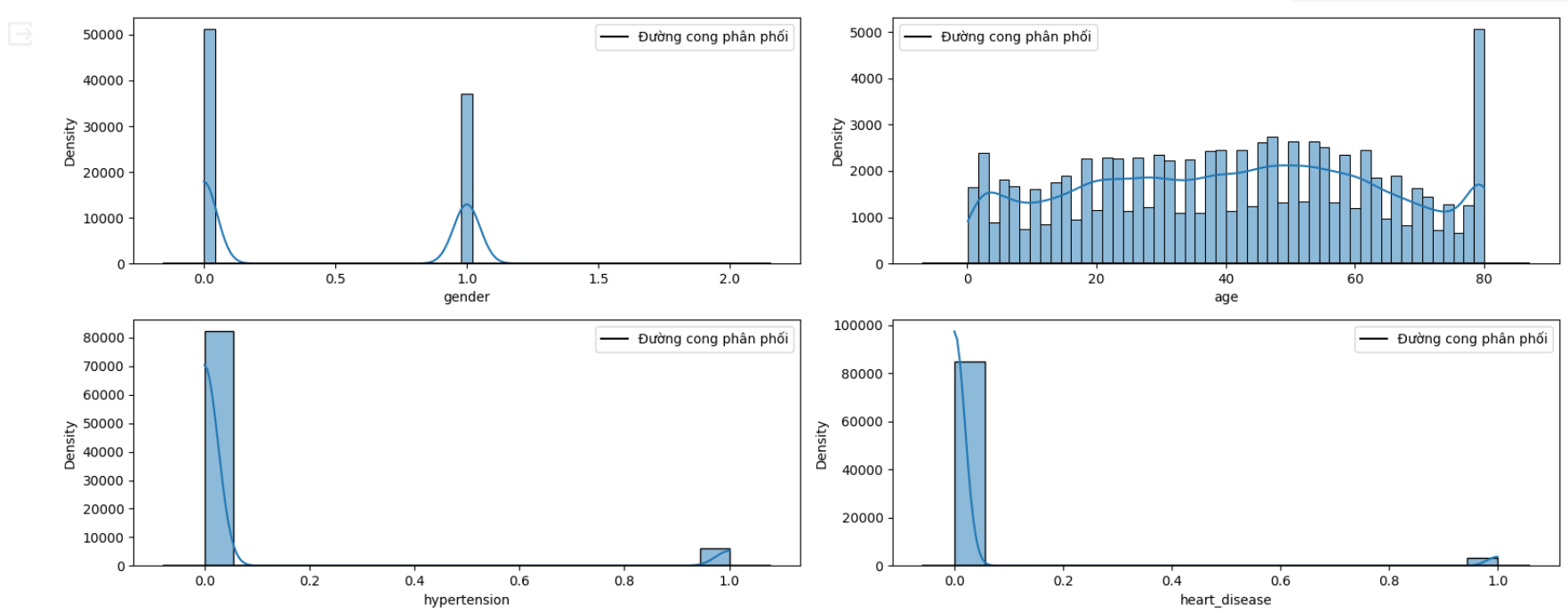
plt.xlabel(feature)

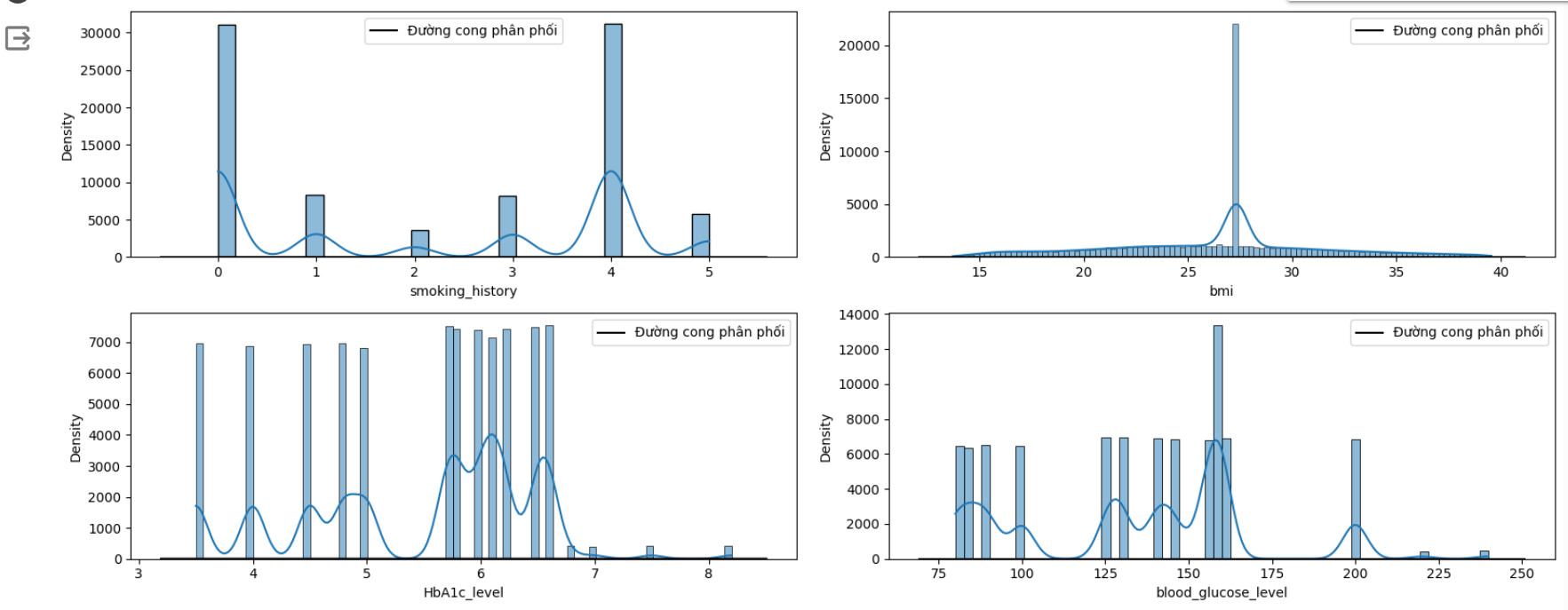
plt.ylabel('Density')

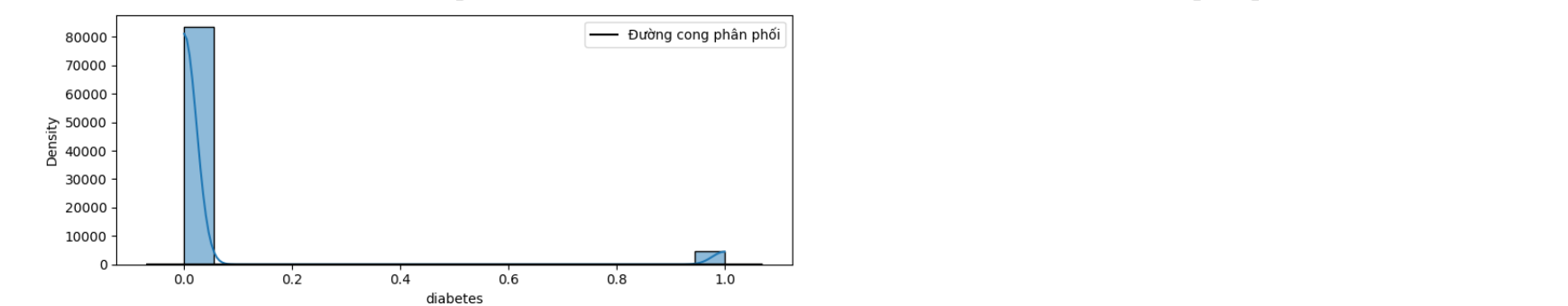
plt.legend()

plt.tight\_layout()

plt.show()







Hình IV. 3. Biểu đồ histogram của tất cả các đặt trưng trong tập dữ liệu

## Phân chia dữ liệu

Bước tiếp theo trong quá trình chuẩn bị dữ liệu cho máy học bao gồm việc chia tất cả dữ liệu đã thu thập thành các tập hợp con — quá trình này được gọi là phân tách dữ liệu. Thông thường, dữ liệu được chia nhỏ thành tập dữ liệu đào tạo, xác thực và thử nghiệm.

* Tập dữ liệu huấn luyện được sử dụng để thực sự dạy một mô hình máy học nhận dạng các mẫu và mối quan hệ giữa các biến đầu vào và mục tiêu. Bộ dữ liệu này thường là lớn nhất.
* Tập dữ liệu xác thực là một tập hợp con dữ liệu được sử dụng để đánh giá hiệu suất của mô hình trong quá trình đào tạo. Nó giúp tinh chỉnh mô hình bằng cách điều chỉnh các siêu tham số (nghĩ: các tham số của quy trình đào tạo được đặt thủ công trước khi đào tạo, như tốc độ học tập, cường độ chuẩn hóa hoặc số lớp ẩn). Tập dữ liệu xác thực cũng giúp ngăn chặn việc khớp quá mức với dữ liệu huấn luyện.
* Tập dữ liệu thử nghiệm là một tập hợp con dữ liệu được sử dụng để đánh giá hiệu suất của mô hình được đào tạo. Mục tiêu của nó là đánh giá độ chính xác của mô hình trên dữ liệu mới, chưa từng thấy. Tập dữ liệu thử nghiệm chỉ được sử dụng một lần — sau khi mô hình đã được huấn luyện và tinh chỉnh trên tập dữ liệu huấn luyện và xác thực.

Bằng cách chia tách dữ liệu, chúng tôi có thể đánh giá mức độ hiệu quả của một mô hình máy học trên dữ liệu mà nó chưa từng thấy trước đây. Nếu không có sự phân tách, rất có thể mô hình sẽ hoạt động kém trên dữ liệu mới. Điều này có thể xảy ra vì mô hình có thể chỉ ghi nhớ các điểm dữ liệu thay vì học các mẫu và khái quát hóa chúng thành dữ liệu mới.

Chia tập dữ liệu thành 2 tập train và test

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

from sklearn.preprocessing import StandardScaler

#Tách biến phụ thuộc ra khỏi tập dữ liệu :

X = df\_no\_outlier.drop('diabetes', axis=1)

y = df\_no\_outlier['diabetes']

#Chuẩn hóa dữ liệu tập (Dùng Min - Max Scaler) :

scaler = StandardScaler()

for col in X.columns :

X[[col]] = scaler.fit\_transform(X[[col]])

#Tách tập dữ liệu thành 2 phần (1 phần để train và 1 phần để test) :

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=101)

X\_train

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **gender** | **age** | **hypertension** | **heart\_disease** | **smoking\_history** | **bmi** | **HbA1c\_level** | **blood\_glucose\_level** |  |
| **36245** | 1.174372 | -1.734788 | -0.271073 | -0.195664 | -1.163514 | -1.578018 | 1.146171 | 0.577776 |
| **13228** | -0.850095 | -0.844157 | -0.271073 | -0.195664 | -1.163514 | 1.603076 | 1.045702 | 0.294133 |
| **48502** | 1.174372 | 1.103546 | -0.271073 | -0.195664 | 0.425446 | 0.303810 | -0.461345 | 0.294133 |
| **2195** | -0.850095 | -0.180167 | -0.271073 | -0.195664 | -0.633860 | -0.644769 | -1.968392 | 1.854166 |
| **3998** | 1.174372 | 1.413407 | -0.271073 | -0.195664 | 0.955100 | 1.978675 | -0.963694 | -0.244787 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **6214** | -0.850095 | -0.578561 | -0.271073 | -0.195664 | 0.425446 | -0.673514 | -0.461345 | 0.719597 |
| **82867** | -0.850095 | -1.789678 | -0.271073 | -0.195664 | 0.955100 | -2.233400 | 0.543353 | 0.691233 |
| **94219** | 1.174372 | 0.793684 | -0.271073 | -0.195664 | 0.955100 | 0.495442 | 0.342413 | -1.407720 |
| **94433** | 1.174372 | 0.882216 | -0.271073 | -0.195664 | 1.484753 | 0.140923 | -1.968392 | -1.407720 |
| **51142** | -0.850095 | 1.457673 | -0.271073 | -0.195664 | 0.425446 | -0.980125 | 0.241943 | 0.662868 |

70556 rows × 8 columns

Vẽ biểu đồ phân phối xác suất (histogram và đường phân phối chuẩn - KDE) hiển thị độ lệch, phương sai và giá trị ngoại biên của tập dữ liệu train.

#Hiển thị lại biểu đồ Histogram sau khi chuẩn hóa :

#Số lượng đặc trưng cần vẽ biểu đồ :

number\_x\_features = len(X\_train.columns)

#Thiết lập kích thước grid cho subplot :

#Tạo một grid có 2 cột (có thể thay đổi số cột tùy theo số đặc trưng) :

cols = 2

rows = number\_x\_features // cols + 1

#Tạo một figure với kích thước phù hợp với số lượng biểu đồ :

plt.figure(figsize=(16, 16))

#Lặp qua từng đặc trưng và vẽ biểu đồ histogram và đường phân phối chuẩn (KDE) :

for i, feature in enumerate(X\_train.columns):

plt.subplot(rows, cols, i+1)

sns.histplot(data=X\_train, x=feature, kde=True)

sns.kdeplot(X\_train[feature], color='black', linestyle='-', label='Đường cong phân phối')

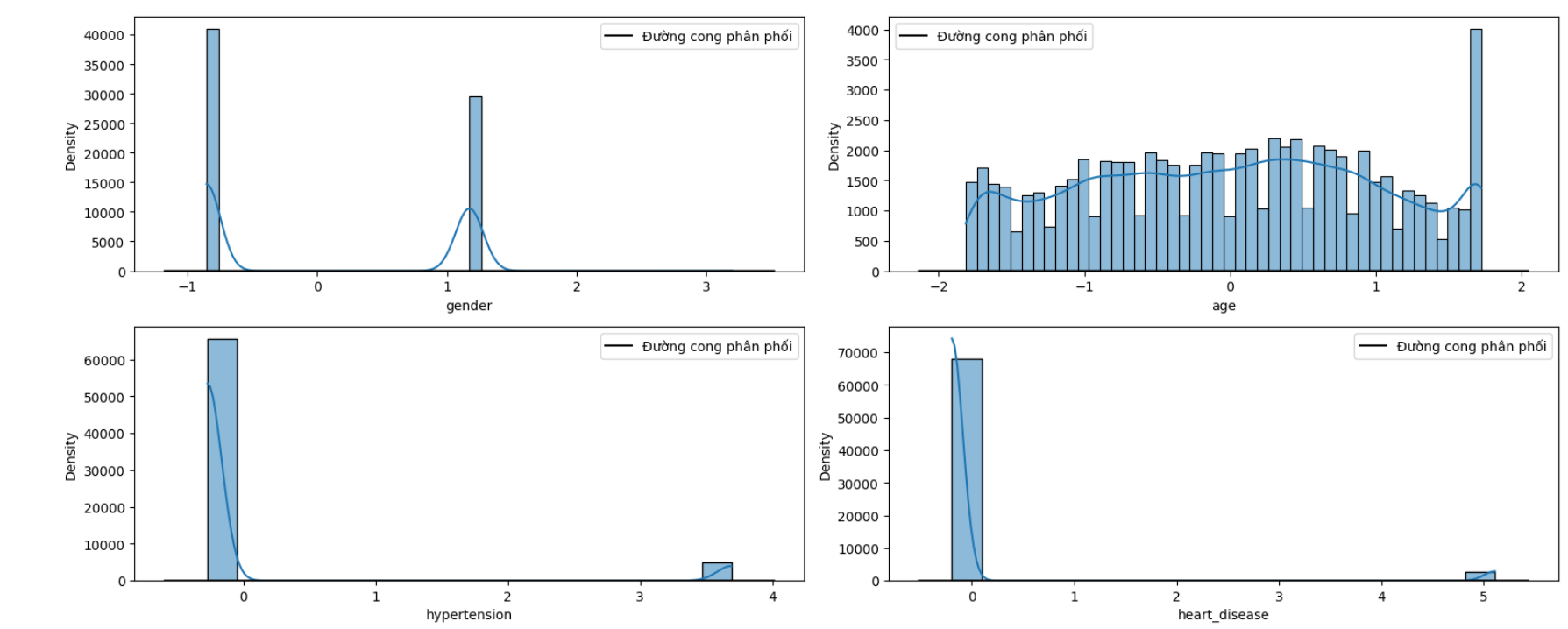
plt.xlabel(feature)

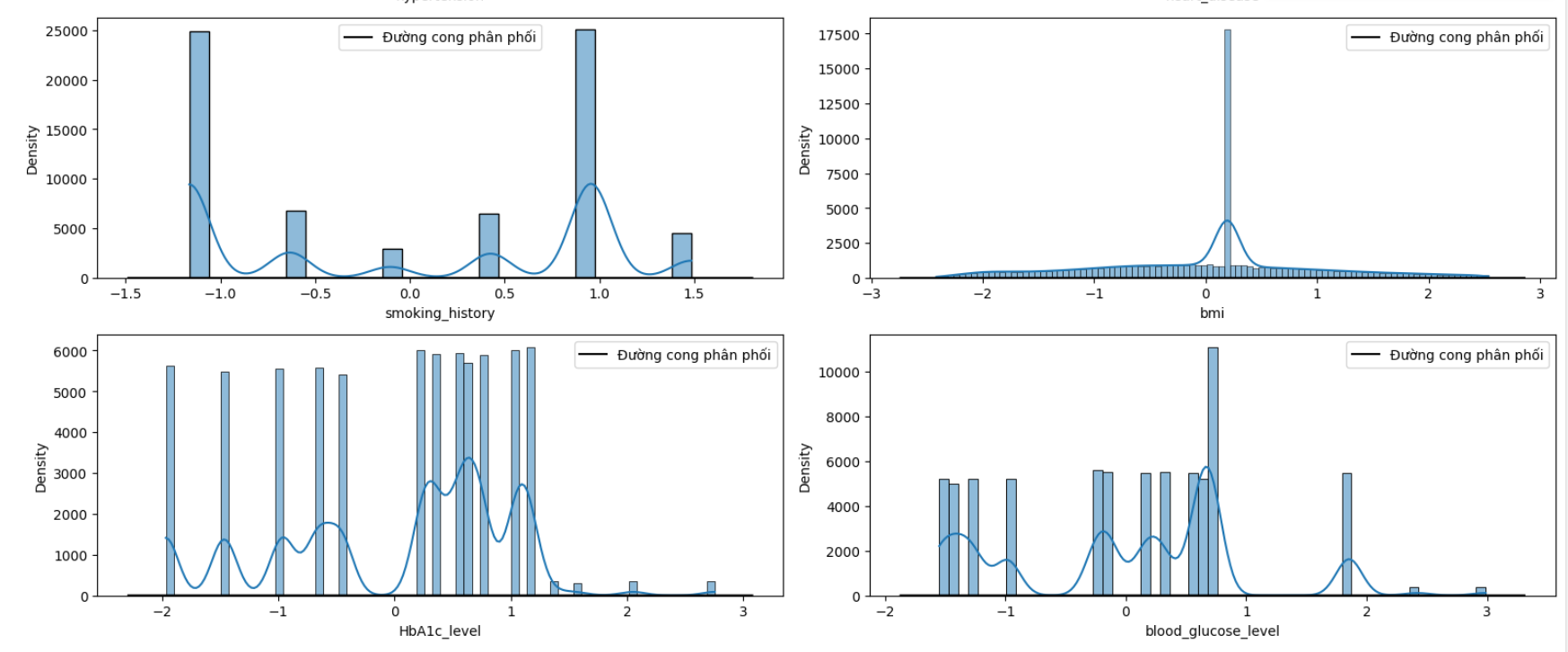
plt.ylabel('Density')

plt.legend()

plt.tight\_layout()

plt.show()





Hình IV. 4. Biểu đồ histogram của đặt trưng trong tập dữ liệu train

# Huấn luyện mô hình

Huấn luận 4 mô hình như sau:

## Mô hình Logistic Regression

Logistic Regression là một mô hình thống kê dự đoán xác suất của một sự kiện dựa trên một tập dữ liệu đầu vào. Mô hình này thường được sử dụng trong các bài toán phân loại và dự đoán.

Dùng thuật toán Logistic Regression để đưa ra dự đoán bệnh tiểu đường :

Hoạt Động của Logistic Regression:

Logistic Regression là một thuật toán phân loại có giám sát, dựa trên hàm logistic (hay còn gọi là hàm sigmoid). Hàm logistic là một hàm phi tuyến, có thể biểu diễn xác suất của một sự kiện xảy ra. Ý tưởng chính của Logistic Regression là dự đoán xác suất của một phần tử dữ liệu thuộc vào một lớp được sử dụng để giải quyết bài toán phân loại nhị phân:

Chọn hàm mất mát: Đầu tiên, chọn hàm mất mát (loss function) để đo lường sự khác biệt giữa dự đoán và thực tế. Thông thường, hàm mất mát được sử dụng là hàm entropy chéo (cross-entropy).

Tối ưu hàm mất mát: Tiếp theo, sử dụng một phương pháp tối ưu hóa (thường là gradient descent) để tìm các tham số (trọng số và độ lệch) của mô hình sao cho hàm mất mát đạt giá trị nhỏ nhất.

Đưa ra dự đoán: Khi có các tham số tối ưu, Logistic Regression sử dụng hàm logistic để tính xác suất của một phần tử dữ liệu thuộc vào lớp 1. Nếu xác suất này lớn hơn một ngưỡng (thường là 0.5), thì phần tử dữ liệu được dự đoán là thuộc lớp 1, ngược lại thì thuộc lớp 0.

from sklearn.linear\_model import LogisticRegression

#Khởi tạo mô hình Logistic Regression :

#Dùng class\_weight = 'balanced' dùng để cân bằng dự đoán giữa 2 lớp :

logistic\_model = LogisticRegression(class\_weight='balanced')

#Huấn luyện mô hình trên tập huấn luyện :

logistic\_model.fit(X\_train, y\_train)

#Dự đoán trên tập kiểm tra :

y\_pred\_logis = logistic\_model.predict(X\_test)

#Tính ma trận nhầm lẫn :

conf\_matrix\_logis = confusion\_matrix(y\_test, y\_pred\_logis)

## Mô hình Random Forest

Random Forest là một thuật toán học máy phổ biến thường được sử dụng cho cả bài toán phân loại và hồi quy. Nó kết hợp kết quả từ nhiều cây quyết định để đưa ra một kết quả duy nhất. Điều này giúp cải thiện độ chính xác và kiểm soát hiện tượng quá khớp (overfitting) mà thường gặp khi sử dụng một cây quyết định duy nhất.

Dùng thuật toán Random Forest để đưa ra dự đoán bệnh tiểu đường :

Hoạt Động của Random Forest:

* Khi có một dữ liệu mới cần được dự đoán, Random Forest thực hiện các bước sau để tạo dự đoán cuối cùng:
* Thực hiện dự đoán trên từng cây quyết định: Mô hình sử dụng mỗi cây quyết định để dự đoán kết quả trên dữ liệu mới.
* Kết hợp dự đoán từ các cây con: Dự đoán từ mỗi cây con được tính toán và sau đó kết hợp để tạo ra dự đoán cuối cùng. Trong bài toán phân loại, kết quả cuối cùng có thể dựa trên sự biểu quyết (voting) của các cây. Trong bài toán hồi quy, kết quả cuối cùng thường là trung bình của các dự đoán từ các cây.
* Kết quả dự đoán cuối cùng: Kết quả dự đoán cuối cùng được trả về.

#Import thư viện :

from sklearn.ensemble import RandomForestClassifier

#Xây dựng mô hình Random Forest (Với n\_estimators là số cây quyết định trong rừng cây (mặc định là 100 cây quyết định)) :

#Mặc định Random Forest sẽ sử dụng kỹ thuật Bagging (Bootstrap Sampling) để chia tập dữ liệu huấn luyện thành các tập con có hoàn lại (With Replacement)

#With Replacement là chia tập con và có hoàn lại mục đinh là làm tăng mức đa dạng của từng cây trong rừng

#Tức là mỗi mẫu có thể xuất hiện nhiều lần và xuất trong nhiều tập con (Để không hoàn lại thì sử dụng bootstrap=False) :

#max\_depth là độ sâu của cây quyết định (Mặc định là sẽ phân loại hoàn toàn được các trường hợp của tập huấn luyện) :

randomfs\_model = RandomForestClassifier(n\_estimators=150,max\_depth = 20,class\_weight='balanced',bootstrap=False,random\_state=101)

#Huấn luyện mô hình trên tập huấn luyện :

randomfs\_model.fit(X\_train, y\_train)

#Dự đoán trên tập kiểm tra :

y\_pred\_rf = randomfs\_model.predict(X\_test)

#Tính ma trận nhầm lẫn :

conf\_matrix\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

## Mô hình K-Nearest Neighbors

K-Nearest Neighbors (KNN): KNN là một thuật toán học máy dựa trên nguyên tắc “điểm dữ liệu giống nhau thường có nhãn giống nhau”. KNN dự đoán nhãn của một điểm dữ liệu mới bằng cách xem xét K điểm dữ liệu gần nhất trong tập huấn luyện.

Dùng thuật toán K-Nearest Neighbors để đưa ra dự đoán bệnh tiểu đường:

Hoạt Động của K-Nearest Neighbors:

* Chọn giá trị K: Đầu tiên, bạn phải chọn giá trị K, tức là số lượng láng giềng gần nhất mà bạn muốn sử dụng để đưa ra dự đoán. Giá trị K càng lớn, mô hình sẽ càng trở nên mượt mà hơn, nhưng cũng có thể làm giảm độ chính xác.
* Xác định khoảng cách: KNN sử dụng một phương pháp đo khoảng cách (thường là khoảng cách Euclidean) để xác định mức độ tương đồng giữa các điểm dữ liệu. Càng gần, khoảng cách càng nhỏ.
* Tìm K láng giềng gần nhất: Đối với mỗi điểm dữ liệu trong tập kiểm tra, KNN tìm K láng giềng gần nhất trong tập huấn luyện dựa trên khoảng cách được xác định.
* Đưa ra dự đoán: Khi có K láng giềng, giá trị hồi quy có thể được dự đoán bằng cách lấy trung bình (hoặc trọng số trung bình, tùy thuộc vào cách bạn cấu hình mô hình) của giá trị của chúng.

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import cross\_val\_score

def select\_best\_k(X\_train, y\_train, max\_k=20, cv=5):

best\_k = 0

best\_accuracy = 0

accuracies = []

k\_values = []

for k in range(1, max\_k + 1, 2):

knn = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(knn, X\_train, y\_train, cv=cv)

accuracy = np.mean(scores)

accuracies.append(accuracy)

k\_values.append(k)

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_k = k

plt.figure(figsize=(8, 6))

plt.plot(k\_values, accuracies, marker='o', linestyle='-', color='b')

plt.title('Cross-Validated Method for Optimal k')

plt.xlabel('k (Number of Neighbors)')

plt.ylabel('Cross-Validated Accuracy')

plt.grid(True)

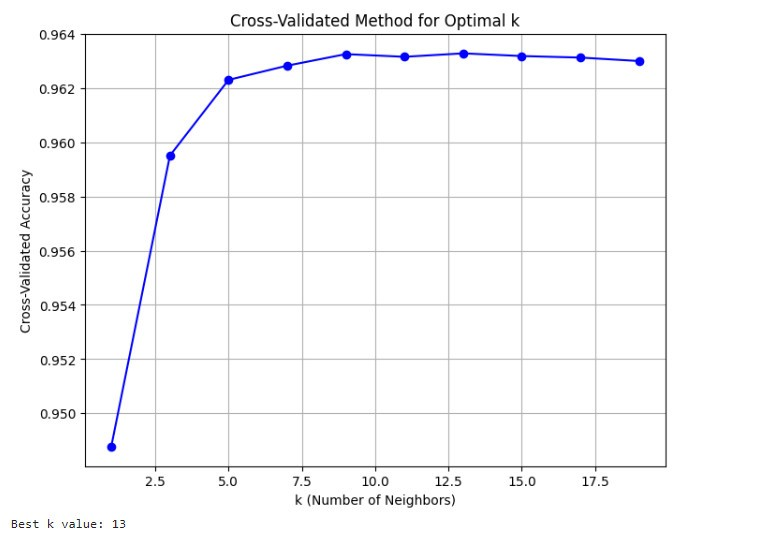
plt.show()

return best\_k

# Sử dụng hàm để chọn giá trị tối ưu của k

best\_k\_value = select\_best\_k(X\_train, y\_train)

print("Best k value:", best\_k\_value)



Hình V. 1. Cross-Validated Method for Optimal k

#Tạo mô hình KNN với giá trị tốt nhất của k :

knn\_model = KNeighborsClassifier(n\_neighbors= best\_k\_value , weights='distance', p=2)

knn\_model.fit(X\_train, y\_train)

#Đưa ra dự đoán trên tập kiểm tra :

knn\_y\_pred = knn\_model.predict(X\_test)

#Ma trận nhầm lẫn :

knn\_cnf\_matrix = confusion\_matrix(y\_test, knn\_y\_pred)

## Mô hình Naive Bayes

Naive Bayes là một thuật toán phân loại dựa trên định lý Bayes với giả định về sự độc lập giữa các đặc trưng. Mặc dù có vẻ đơn giản, nhưng Naive Bayes lại hoạt động rất hiệu quả trong nhiều tình huống thực tế.

Dùng thuật toán Naive Bayes để đưa ra dự đoán bệnh tiểu đường :

Hoạt Động của Naive Bayes:

Thuật toán phân loại có giám sát, dựa trên Định lý Bayes. Ý tưởng chính của nó là dự đoán xác suất của một phần tử dữ liệu thuộc vào một lớp được sử dụng để giải quyết bài toán phân loại:

* Chọn giả định: Đầu tiên, chọn giả định về độc lập. Naive Bayes giả định rằng tất cả các thuộc tính đầu vào đều độc lập với nhau.
* Áp dụng Định lý Bayes: Naive Bayes sử dụng Định lý Bayes để tính xác suất hậu nghiệm, tức là xác suất của một sự kiện A xảy ra khi biết sự kiện B đã xảy ra. Trong bài toán phân loại, A là một phần tử dữ liệu, B là một giả thiết về lớp mà A thuộc về.
* Tính xác suất: Đối với mỗi lớp, Naive Bayes tính xác suất của mỗi thuộc tính dữ liệu thuộc về lớp đó. Xác suất này được tính dựa trên tần suất xuất hiện của thuộc tính trong lớp.
* Đưa ra dự đoán: Khi có xác suất cho mỗi lớp, Naive Bayes dự đoán lớp của một điểm dữ liệu bằng cách chọn lớp có xác suất hậu nghiệm cao nhất.

#Import thư viện :

from sklearn.naive\_bayes import GaussianNB

#Khởi tạo mô hình Naive Bayes :

nb\_model = GaussianNB()

nb\_model.fit(X\_train, y\_train)

#Đưa ra dự đoán trên tập kiểm tra :

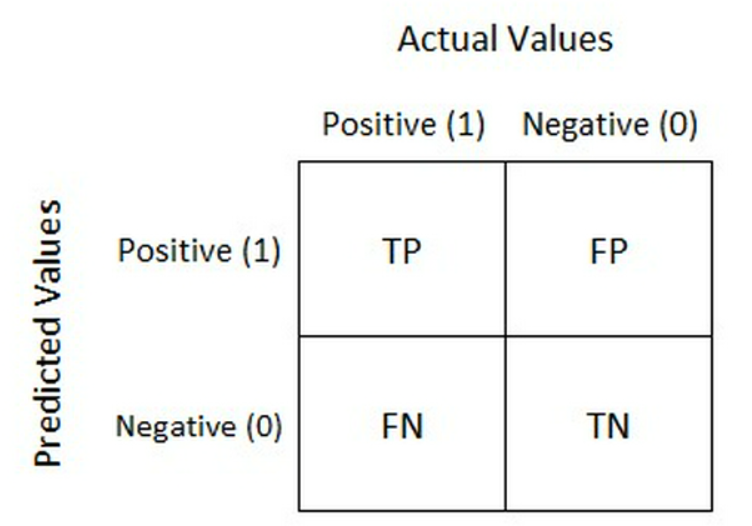
nb\_y\_pred = nb\_model.predict(X\_test)

#Ma trận nhầm lẫn :

nb\_cnf\_matrix = confusion\_matrix(y\_test, nb\_y\_pred)

# Đánh giá mô hình

Ma trận nhầm lẫn là ma trận tóm tắt hiệu suất của mô hình học máy trên một tập hợp dữ liệu thử nghiệm. Nó thường được sử dụng để đo lường hiệu suất của các mô hình phân loại, nhằm mục đích dự đoán nhãn phân loại cho từng phiên bản đầu vào. Ma trận hiển thị số lượng kết quả dương tính thật (TP), âm tính thật (TN), dương tính giả (FP) và âm tính giả (FN) do mô hình tạo ra trên dữ liệu thử nghiệm.



* TP (True Positive): Số lượng dự đoán chính xác.
* TN (True Negative): Số lương dự đoán chính xác một cách gián tiếp.
* FP (False Positive - Type 1 Error): Số lượng các dự đoán sai lệch.
* FN (False Negative - Type 2 Error): Số lượng các dự đoán sai lệch một cách gián tiếp.

Accuracy được sử dụng để đo lường hiệu suất của mô hình. Đó là tỷ lệ của Tổng số trường hợp đúng trên tổng số trường hợp.



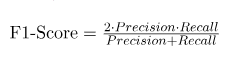
Precision là thước đo mức độ chính xác của các dự đoán tích cực của mô hình. Nó được định nghĩa là tỷ lệ dự đoán tích cực thực sự trên tổng số dự đoán tích cực được thực hiện bởi mô hình.



Recall đo lường tính hiệu quả của mô hình phân loại trong việc xác định tất cả các trường hợp có liên quan từ tập dữ liệu. Đó là tỷ lệ giữa số trường hợp dương tính thật (TP) với tổng số trường hợp dương tính thật và âm tính giả (FN).



F1-score được sử dụng để đánh giá hiệu suất tổng thể của mô hình phân loại. Đó là ý nghĩa hài hòa của Precision và Recall.



#Trực quan hoá ma trận nhầm lẫn bằng seaborn :

plt.figure(figsize=(32, 8))

#Trực quan ma trận nhầm lẫn của Logistic Regression :

plt.subplot(1,4,1)

sns.heatmap(conf\_matrix\_logis, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], yticklabels=['0', '1'])

plt.xlabel('Dự đoán')

plt.ylabel('Thực tế')

plt.title('Ma trận nhầm lẫn (Logistic Regression)')

#Trực quan ma trận nhầm lẫn của Random Forest :

plt.subplot(1,4,2)

sns.heatmap(conf\_matrix\_rf, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], yticklabels=['0', '1'])

plt.xlabel('Dự đoán')

plt.ylabel('Thực tế')

plt.title('Ma trận nhầm lẫn (Random Forest)')

#Trực quan ma trận nhầm lẫn của K-Nearest Neighbors :

plt.subplot(1,4,3)

sns.heatmap(knn\_cnf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], yticklabels=['0', '1'])

plt.xlabel('Dự đoán')

plt.ylabel('Thực tế')

plt.title('Ma trận nhầm lẫn (K-Nearest Neighbors)')

#Trực quan ma trận nhầm lẫn của Naive Bayes :

plt.subplot(1,4,4)

sns.heatmap(nb\_cnf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], yticklabels=['0', '1'])

plt.xlabel('Dự đoán')

plt.ylabel('Thực tế')

plt.title('Ma trận nhầm lẫn (Naive Bayes)')

plt.show()

print()

#Đánh giá hiệu suất mô hình :

accuracy\_logistic = accuracy\_score(y\_test, y\_pred\_logis)

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf)

accuracy\_knn = accuracy\_score(y\_test, knn\_y\_pred)

accuracy\_nb = accuracy\_score(y\_test, nb\_y\_pred)

print(f'Độ chính xác của mô hình Logistic Regression : {accuracy\_logistic : .2f}')

print(f'Độ chính xác của mô hình Random Forest : {accuracy\_rf : .2f}')

print(f'Độ chính xác của mô hình K-Nearest Neighbors : {accuracy\_knn : .2f}')

print(f'Độ chính xác của mô hình Naive Bayes : {accuracy\_nb : .2f}')

#Báo cáo phân loại của mô hình Logistic Regression :

logis\_report = classification\_report(y\_test, y\_pred\_logis, output\_dict=True)

df\_logis\_report = pd.DataFrame(logis\_report).transpose()

#Báo cáo phân loại Random Forest :

rf\_report = classification\_report(y\_test, y\_pred\_rf, output\_dict=True)

df\_rf\_report = pd.DataFrame(rf\_report).transpose()

#Báo cáo phân loại Random Forest :

knn\_report = classification\_report(y\_test, knn\_y\_pred, output\_dict=True)

df\_knn\_report = pd.DataFrame(knn\_report).transpose()

#Báo cáo phân loại Naive Bayes :

nb\_report = classification\_report(y\_test, nb\_y\_pred, output\_dict=True)

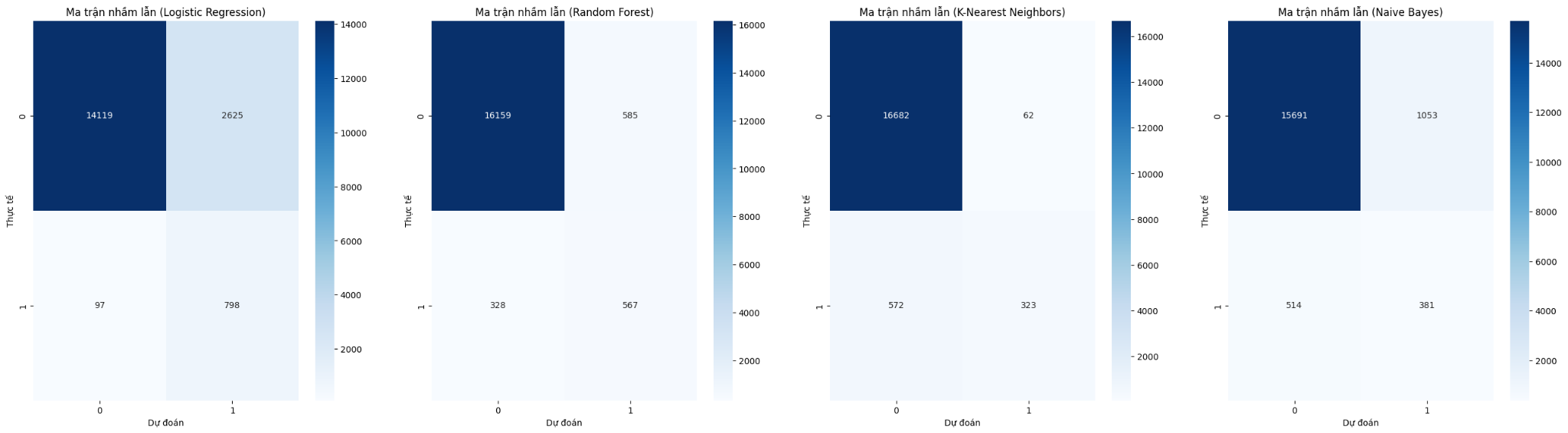
df\_nb\_report = pd.DataFrame(nb\_report).transpose()

#Kết hợp các DataFrame :

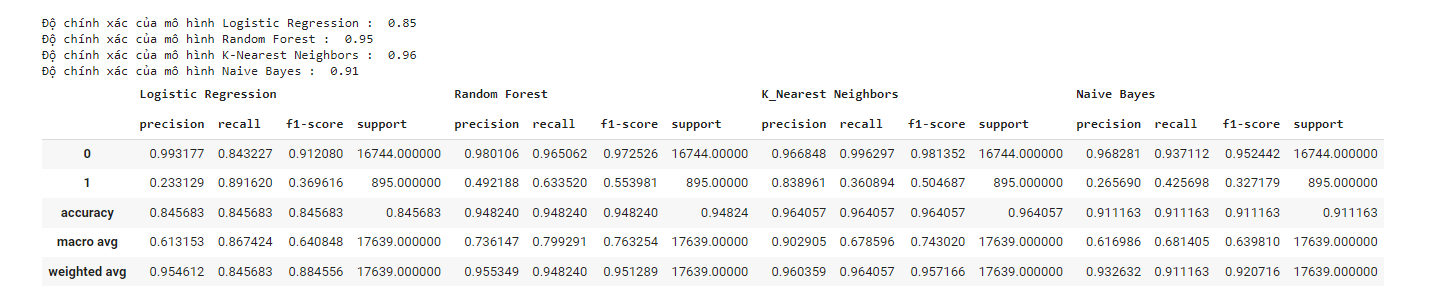
combined\_df = pd.concat([df\_logis\_report, df\_rf\_report, df\_knn\_report,df\_nb\_report], keys=['Logistic Regression', 'Random Forest', 'K\_Nearest Neighbors','Naive Bayes'], axis=1)

#Hiển thị kết quả :

combined\_df



Hình VI. 1 Ma trận nhần lẫn của 4 mô hình

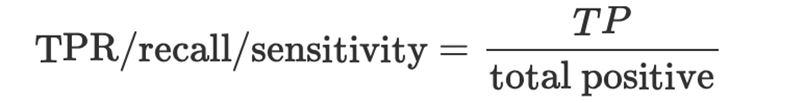


Kỹ thuật đánh giá AUC(Area Under The Curve)

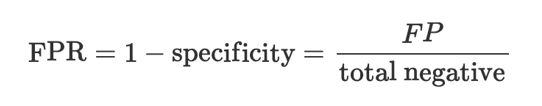
ROC là đường cong biểu diễn khả năng phân loại của một mô hình phân loại tại các ngưỡng threshold. Đường cong này dựa trên hai chỉ số :

* TPR (true positive rate): Chỉ số này chính bằng *độ phủ*. Một số tài liệu thống kê còn gọi chúng là *độ nhạy*(*sensitivity*). Đây là tỷ lệ các trường hợp phân loại đúng *dương tính* trên tổng số các trường hợp thực tế là *dương tính*. Nó có tác dụng đánh giá mức độ dự báo chính xác của mô hình trên nhóm *dương tính*. Khi giá trị của nó càng cao, mô hình dự báo càng tốt **trên nhóm *dương tính***.
* FPR (false positive rate): Tỷ lệ dự báo sai các trường hợp thực tế là *âm tính* thành thành *dương tính* trên tổng số các trường hợp thực tế là *âm tính*. Một mô hình có FPR càng thấp thì mô hình càng chuẩn xác vì sai số của nó **trên nhóm *âm tính*** càng thấp. Phần bù của FPR là *độ đặc hiệu* (*specificity*) đo lường tỷ lệ dự báo đúng các trường hợp *âm tính* trên tổng số các trường hợp thực tế là *âm tính*.

Nếu TPR=0.9, chúng ta tin rằng 90% các mẫu thuộc nhóm *dương tính* đã được mô hình phân loại đúng.



Nếu giá trị của FPR =0.1, mô hình đã dự báo sai 10% trên tổng số các trường hợp là *âm tính*.



AUC là chỉ số được tính toán dựa trên đường cong ROC (receiving operating curve) nhằm **đánh giá khả năng phân loại** của mô hình tốt như thế nào ?

Phần diện tích gạch chéo nằm dưới đường cong ROC và trên trục hoành là AUC (area under curve) có giá trị nằm trong khoảng [0, 1].

* Khi diện tích này càng lớn thì đường cong ROC có xu hướng tiệm cận đường thẳng y=1 và khả năng phân loại của mô hình càng tốt.
* Khi đường cong ROC nằm sát với đường chéo chính đi qua hai điểm (0, 0) và (1, 1), mô hình sẽ tương đương với một phân loại ngẫu nhiên.
* Trường hợp đường cong ROC nằm dưới đường chéo chính thể hiện rằng mô hình có chất lượng kém và thậm chí thua xa một dự báo ngẫu nhiên.

Vẽ biểu đồ ROC (Receiver Operating Characteristic) cho bốn mô hình học máy khác nhau: Random Forest, Logistic Regression, K-Nearest Neighbors và Naive Bayes. Biểu đồ ROC là một công cụ hữu ích để đánh giá hiệu suất của các mô hình phân loại, bằng cách so sánh tỷ lệ giữa True Positive Rate (TPR) và False Positive Rate (FPR) ở các ngưỡng khác nhau.

from sklearn.metrics import roc\_curve, roc\_auc\_score

plt.figure(figsize=(16, 6))

#Tính điểm ROC và đường cong ROC cho Logistic Regression:

y\_pred\_prob\_logis = logistic\_model.predict\_proba(X\_test)[:, 1]

fpr\_logis, tpr\_logis, thresholds\_logis = roc\_curve(y\_test, y\_pred\_prob\_logis)

roc\_auc\_logis = roc\_auc\_score(y\_test, y\_pred\_prob\_logis)

#Tính điểm ROC và đường cong ROC cho Random Forest:

y\_pred\_prob\_rf = randomfs\_model.predict\_proba(X\_test)[:, 1]

fpr\_rf, tpr\_rf, thresholds\_rf = roc\_curve(y\_test, y\_pred\_prob\_rf)

roc\_auc\_rf = roc\_auc\_score(y\_test, y\_pred\_prob\_rf)

#Tính điểm ROC và đường cong ROC cho K-Nearest Neighbors:

y\_pred\_prob\_knn = knn\_model.predict\_proba(X\_test)[:, 1]

fpr\_knn, tpr\_knn, thresholds\_knn = roc\_curve(y\_test, y\_pred\_prob\_knn)

roc\_auc\_knn = roc\_auc\_score(y\_test, y\_pred\_prob\_knn)

#Tính điểm ROC và đường cong ROC cho Naive Bayes:

y\_pred\_prob\_nb = nb\_model.predict\_proba(X\_test)[:, 1]

fpr\_nb, tpr\_nb, thresholds\_nb = roc\_curve(y\_test, y\_pred\_prob\_nb)

roc\_auc\_nb = roc\_auc\_score(y\_test, y\_pred\_prob\_nb)

#Vẽ biểu đồ ROC:

plt.subplot(1, 2, 1)

plt.plot(fpr\_rf, tpr\_rf, label=f'Random Forest (AUC = {roc\_auc\_rf:.2f})')

plt.plot(fpr\_logis, tpr\_logis, label=f'Logistic Regression (AUC = {roc\_auc\_logis:.2f})')

plt.plot(fpr\_knn, tpr\_knn, label=f'K-Nearest Neighbors (AUC = {roc\_auc\_knn:.2f})')

plt.plot(fpr\_nb, tpr\_nb, label=f'Naive Bayes (AUC = {roc\_auc\_nb:.2f})')

plt.plot([0, 1], [0, 1], '--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel('False Positive Rate')

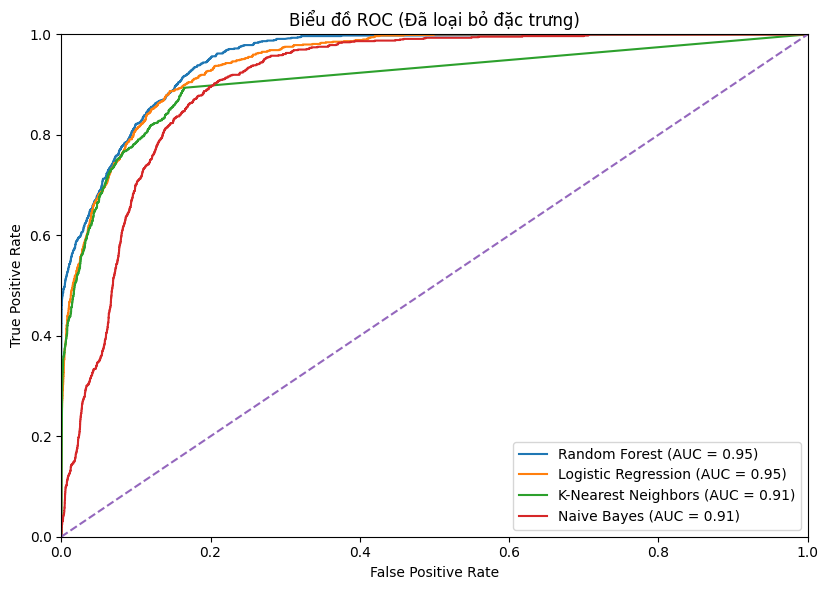
plt.ylabel('True Positive Rate')

plt.title('Biểu đồ ROC')

plt.legend(loc='lower right')

plt.tight\_layout()

plt.show()



Hình VI. 2. Biểu đồ ROC ( Đã loại bỏ đặc trưng)

Biểu đồ ROC ở trên giúp ta so sánh hiệu suất của các mô hình phân loại khác nhau, bằng cách xem xét cách thức mô hình phân biệt giữa các phần tử dữ liệu thuộc vào lớp 1 và lớp 0. Một mô hình tốt sẽ có đường cong ROC gần với góc trên bên trái, có nghĩa là TPR cao và FPR thấp.

# Kết luận

Dựa trên kết quả trên, có thể đánh giá và nhận xét 4 mô hình học máy như sau: Mô hình Logistic Regression có độ chính xác (accuracy) là 0.85, tức là nó dự đoán đúng khoảng 85% số lượng dữ liệu trong tập kiểm tra. Tuy nhiên, mô hình này có độ chuẩn xác (precision) thấp đối với lớp 1 (bệnh tiểu đường), chỉ khoảng 23%. Điều này có nghĩa là mô hình này dễ bỏ sót những người có nguy cơ mắc bệnh. Mặt khác, mô hình này có độ nhớ (recall) cao đối với lớp 1, khoảng 89%, tức là nó có thể phát hiện được hầu hết những người mắc bệnh trong tập dữ liệu. Điểm AUC của mô hình này là 0.95, cho thấy nó có khả năng phân biệt giữa hai lớp khá tốt. Mô hình Random Forest có độ chính xác (accuracy) rất cao trong 4 mô hình, là 0.95, tức là nó dự đoán đúng khoảng 95% số lượng dữ liệu trong tập kiểm tra. Mô hình này cũng có độ chuẩn xác (precision) và độ nhớ (recall) cao đối với cả hai lớp, khoảng 98% và 96% đối với lớp 0, và khoảng 49% và 63% đối với lớp 1. Điểm AUC của mô hình này là 0.95, cho thấy nó có khả năng phân biệt giữa hai lớp là rất tốt. Mô hình K-Nearest Neighbors có độ chính xác (accuracy) là 0.96 cao nhất trong 4 mô hình , tức là nó dự đoán đúng khoảng 96% số lượng dữ liệu trong tập kiểm tra. Mô hình này có độ chuẩn xác (precision) và độ nhớ (recall) cao đối với lớp 0, khoảng 97% và 100% tương ứng. Tuy nhiên, mô hình này có độ chuẩn xác (precision) và độ nhớ (recall) thấp đối với lớp 1, chỉ khoảng 84% và 36% tương ứng. Điều này cho thấy là mô hình dễ bị nhầm lẫn giữa hai lớp và bỏ sót những người mắc bệnh. Điểm AUC của mô hình này là 0.91 , cho thấy nó có khả năng phân biệt giữa hai lớp cũng khá tốt. Mô hình Naive Bayes có độ chính xác (accuracy) là 0.91, tức là nó dự đoán đúng khoảng 91% số lượng dữ liệu trong tập kiểm tra. Mô hình này có độ chuẩn xác (precision) và độ nhớ (recall) cao đối với lớp 0, khoảng 97% và 94% tương ứng. Tuy nhiên, mô hình này có độ chuẩn xác (precision) và độ nhớ (recall) thấp đối với lớp 1, chỉ khoảng 27% và 43% tương ứng. Điều này có nghĩa là mô hình này cũng dễ bị nhầm lẫn giữa hai lớp và bỏ sót những người mắc bệnh. Điểm AUC của mô hình này là 0.91, cho thấy nó có khả năng phân biệt giữa hai lớp cũng rất tốt.

Kết luận : Trong 4 mô hình , mô hình Random Forest và Logistic Regression có hiệu suất tốt , vì nó có độ chính xác (accuracy) , độ chuẩn xác (precision) , độ nhớ (recall) và điểm AUC cao đối với cả hai lớp. Mô hình K-Nearest Neighbors và Naive Bayes có hiệu suất chưa tốt, vì mặc dù chúng có độ chính xác (accuracy) , độ chuẩn xác (precision) , độ nhớ (recall) và điểm AUC cũng tương đối cao nhưng chỉ cao đối với lớp 0 (Không mắc bệnh) còn đối với lớp 1 (Có mắc bệnh) thì lại khá thấp , điều này dẫn đến mô hình sẽ dễ bị nhầm lẫn và không cân bằng được giữa 2 lớp làm cho mô hình dự đoán không tổng quát được so với 2 mô hình còn lại. Đối với việc dự đoán bệnh tiểu đường thì 2 mô hình Random Forest và Logistic Regression là 2 mô hình tốt nhất có thể được lựa chọn , mỗi cái sẽ có những ưu nhược điểm khác nhau . Về Logistic thì ưu điểm sẽ có độ nhớ (recall) của lớp 1 rất cao nhưng đổi lại thì độ chuẩn xác (precision) của nó lại rất thấp , còn về Radom Forest thì lại cân bằng hơn một chút là độ chuẩn xác (precision) và độ nhớ (recall) của lớp 1 tương đối cao và có sự chênh lệch ít hơn so với Logistic . Nên dựa vào các số liệu thống kê về độ chính xác (accuracy) , độ chuẩn xác (precision), độ nhớ (recall) và điểm AUC cùng với việc so sánh giữa các mô hình với nhau thì việc chọn mô hình Random Forest là phù hợp và tốt nhất cho việc dự đoán bệnh tiểu đường.

# Tài liệu tham khảo (IEEE)

|  |
| --- |
| GeeksforGeeks. (2023, March 21). *Confusion Matrix in machine learning*. <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/> |
| ITRex. (2023, April 13). *Chuẩn bị dữ liệu cho Machine Learning: Hướng dẫn từng bước*. HackerNoon. <https://hackernoon.com/vi/chu%E1%BA%A9n-b%E1%BB%8B-d%E1%BB%AF-li%E1%BB%87u-cho-m%C3%A1y-h%E1%BB%8Dc-h%C6%B0%E1%BB%9Bng-d%E1%BA%ABn-t%E1%BB%ABng-b%C6%B0%E1%BB%9Bc> |
| Kononenko, I. (2007). Statistical learning. In *Elsevier eBooks* (pp. 259–274). <https://doi.org/10.1533/9780857099440.259> |
| *sklearn.ensemble.RandomForestRegressor*. (n.d.). Scikit-learn.  <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html> |
| *sklearn.neighbors.KNeighborsRegressor*. (n.d.-a). Scikit-learn.  <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html> |
| *1.9. Naive Bayes*. (n.d.). Scikit-learn.  <https://scikit-learn.org/stable/modules/naive_bayes.html> |