BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC UEH - TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

CHUYÊN NGÀNH KHOA HỌC DỮ LIỆU

KHÓA LUẬN TỐT NGHIỆP

Đề tài:

XÂY DỰNG MÔ HÌNH KẾT LUẬN TÌNH TRẠNG ĐỘT QUỴ

Họ tên sinh viên: BẠCH NGỌC MINH TRÚC

Mã sinh viên: 31201024526

Lớp: DS001 Khóa: 46

Họ tên giáo viên hướng dẫn: HUỲNH VĂN ĐỨC

Niên khóa: 2020 - 2023

Tp Hồ Chí Minh, ngày 20 tháng 10 năm 2023

Lời cám ơn

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến Đại học Kinh tế Thành phố Hồ Chí Minh nói chung và thầy/cô khoa Công nghệ thông tin kinh doanh (BIT) nói riêng, các thầy/cô đã nhìn thấy triển vọng và nhu cầu của ngành Khoa học dữ liệu (Data Science) trong tương lai nên đã đưa môn học này vào chương trình giảng dạy.

Đặc biệt, em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn – Thầy Huỳnh Văn Đức đã tận tình truyền đạt và bổ sung những kiến thức quý giá cho em trong suốt thời gian thực hiện đề tài. Trong thời gian này, em đã trau dồi cho bản thân nhiều kiến thức bổ ích, tinh thần học tập nghiêm túc và hiệu quả. Đây chắc chắn sẽ là những kiến thức có giá trị sâu sắc, là hành trang để em vững bước sau này.

Em xin chân thành cảm ơn!

Tóm tắt

Mục đích của đề tài **Xây Dựng Mô Hình Kết Luận Tình Trạng Đột Quỵ** là đưa ra dự báo phân loại tình trạng không hoặc có bị đột quỵ (0 đại diện cho "không"; 1 đại diện cho "có"). Vì vậy, đây là bài toán học có giám sát phân lớp nhị phân và tìm ra mô hình phân loại tốt nhất.

Để giải quyết vấn đề trên, em đã sử dụng quy trình làm việc ngành Khoa học dữ liệu từ bước tìm hiểu vấn đề, nhu cầu của đề tài (chương 1); làm sạch dữ liệu – chuyển đổi tập dữ liệu thô sang tập dữ liệu sẵn sàng để huấn luyện cho mô hình (mục 3.2); tính ra hàm số thể hiện được mối liên kết giữa các biến độc lập với biến phụ thuộc, xây dựng và kiểm định các mô hình học máy liên quan đến hàm số trên và kết luận ra một mô hình tốt nhất (mục 3.2); đến bước truyền đạt kết quả phân tích – trên cơ sở có được mô hình cuối cùng, em đã tìm xác suất ở từng chân trị 0 và 1 của từng bệnh nhân (mục 3.3). Bên cạnh đó, em dùng chương 2 để tập hợp các cơ sở lý thuyết mà hỗ trợ em trong suốt bài toán này.

Cụ thể, đầu tiên em đã kiểm tra tập dữ liệu có cân bằng không và khắc phục cho nó bằng Up-sample Minority Class. Nhận thấy mối liên kết giữa chỉ số BMI và tuổi tác nên em đã thay thế các giá trị NaN của biến BMI theo các nhóm tuổi. Sau khi kiểm định hoàn tất các biến độc lập nào gây ảnh hưởng lên biến phụ thuộc bằng các phương pháp kiểm định thống kê, em đã xây được mô hình giả định giữa chúng. Từ đó, em phát triển hàm hồi quy phi tuyến tính Logistic (còn gọi là hàm Sigmoid) và xem nó như tiền đề để xây dựng các mô hình dự báo. Mô hình dự báo tốt nhất mà em có được là Hồi quy Logistic (Logistic Regression) có mức accuracy khoảng 78%.

Mục lục

Danh mục các hình ảnh, biểu đồ

Danh mục các bảng biểu

Chương 1: Tổng quan

1.1 Lý do chọn đề tài

Nhằm đáp ứng được nhu cầu sức khỏe của con người ngày càng cao, các bệnh viện đang tích cực tìm tòi, nghiên cứu các thiết bị, máy móc xét nghiệm, chuẩn đoán mới; chuẩn xác và tuyển dụng đội ngũ y bác sĩ tài năng, kinh nghiệm dồi dào để khám/chữa bệnh cho người dân. Đối với ngành Khoa học dữ liệu, ngành này cũng có thể góp sức vào việc hỗ trợ các bác sĩ và bệnh viện để tránh sai xót xuống tối thiểu và gia tăng sự chính xác trong quá trình khám/chữa lên tối đa.

Nhờ sức mạnh của khoa học - công nghệ tiên tiến nên không chỉ bác sĩ – người điều trị trực tiếp cho bệnh nhân mà còn thiết bị, máy móc trong y tế – sản phẩm của nhóm ngành kĩ thuật – đã giúp cho trình độ y học trong nước được bắt kịp với thế giới.

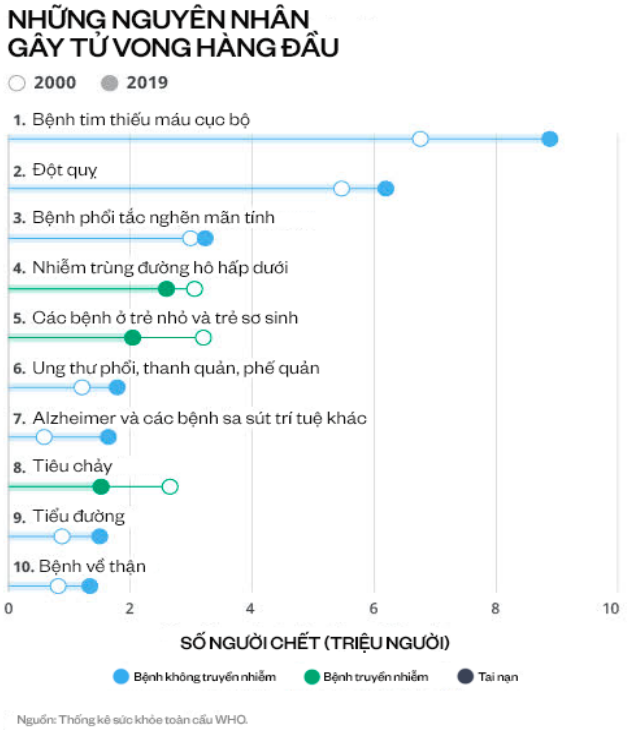
1.2 Mục tiêu

Có hai mục tiêu lớn trong đề tài này, chính là:

- Đối với bác sĩ, mô hình này giúp đỡ, giảm nhẹ gánh nặng trong quá trình chuẩn đoán khả năng đột quỵ của bệnh nhân.
- Đối với người dân, mô hình này là một ứng dụng hữu ích hỗ trợ họ kiểm soát tình trạng sức khỏe của bản thân một cách trực tiếp và thuận tiện.

1.3 Đối tượng nghiên cứu

Đặt vấn đề con người là cốt lõi nên sức khỏe của họ được đặt lên hàng đầu trong bài nghiên cứu này. Gần đây nhất chính là bài nghiên cứu Top 10 nguyên nhân gây tử vong hàng đầu cho con người trong gần hai thập kỷ (2000 – 2019) của Tổ chức Y tế Thế giới (WHO) đã được công bố vào tháng 10/2020.



Hình 1. Top 10 nguyên nhân gây tử vong 2000-2019

Trong đó, WHO đã thống kê được đột quỵ chiếm vị trí thứ hai và trong gần 20 năm, tỉ lệ nạn nhân tử vong vì đột quỵ vẫn đang tăng. Đột quỵ – còn gọi là tai biến mạch máu não, xảy ra khi mạch máu cung cấp oxy và dinh dưỡng cho não bị tắc nghẽn.

1.4 Phạm vi nghiên cứu

Đề tài được thực hiện với tư cách là luận án tốt nghiệp cử nhân Đại học Kinh tế TP.HCM và được diễn ra trong 11 tuần, tính từ ngày 7 đến ngày 20 /10/2023.

1.5 Thực hiện

Link kết quả chạy code trên Jupyter.

Link kết quả chạy code trên Github.

Link bộ dữ liệu trên Github.

Link mã nguồn và bộ dữ liệu trên Google Drive.

Chương 2: Cơ sở lý thuyết

2.1 Định nghĩa mất cân bằng dữ liệu

Bộ dữ liệu mất cân bằng (imbalanced dataset) là một bộ dữ liệu có sự khác biệt lớn trong việc phân phối các lớp – kết quả của biến phụ thuộc y. Điều này dẫn đến kết quả dự báo sẽ thiên về một lớp trong bộ dữ liệu bởi vì bộ dữ liệu thiên về một lớp, một thuật toán được huấn luyện trên cùng một dữ liệu sẽ thiên về cùng một lớp.

2.2 Chỉ số BMI có mối liên hệ thế nào với tuổi tác

BMI (Body Mass Index) là chỉ số khối cơ thể con người dựa trên công thức BMI = $\frac{Trọng\ lượng\ cơ\ thể\ (kg)}{Chiều\ cao^2\ (m^2)}$, được dùng để đánh giá mức độ gầy hay béo của một người. Chỉ số này do nhà bác học người Bỉ đưa ra vào năm 1832. Nhược điểm duy nhất của chỉ số BMI là nó không thể tính toán được lượng chất béo trong cơ thể – yếu tố tiềm ẩn các nguy cơ liên quan đến sức khỏe trong tương lai.

Tuy nhiên thang đo BMI lại được đo lường theo 2 nhóm chính: người lớn trên 20 tuổi và trẻ em. Nguyên nhân là vì cấu trúc bên trong cơ thể, các tế bào của trẻ em chưa phát triển hoàn toàn nên khả năng hấp thụ các chất dinh dưỡng của trẻ em sẽ thấp hơn nên chỉ số BMI cũng sẽ có mức quy đổi khác so với người lớn. Thêm vào đó, Tổ chức Y tế Thế giới (WHO) đã đưa ra kết quả khảo sát các mức đánh giá độ béo phì bằng BMI theo từng nhóm tuổi: người lớn và trẻ em. Vì vậy, ta hoàn toàn có cơ sở liên kết giá trị BMI với tuổi tác.

- Đối với người lớn:

Chỉ số BMI chuẩn đối với một người bình thường nằm trong khoảng từ 18,5 – 25. Tỷ lệ về chỉ số BMI của người trên 20 tuổi cụ thể như sau:

BMI < 18: người gầy

BMI = 18,5 – 25: người bình thường

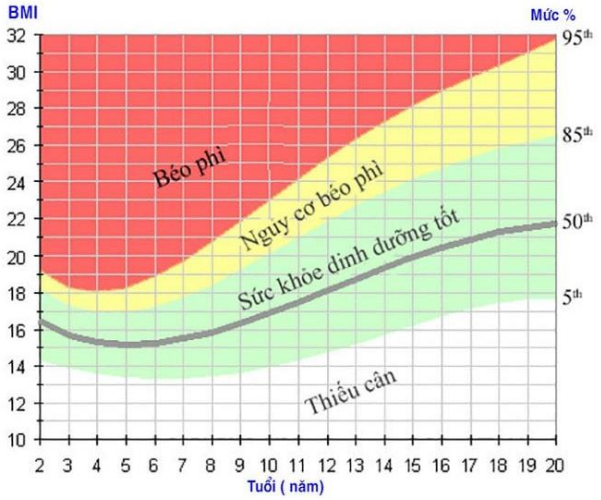
BMI = 25 – 30: người béo phì độ I

BMI = 30 – 40: người béo phì độ II

BMI > 40: người béo phì độ III

- Đối với trẻ em:

Mức độ gầy/béo sẽ được kết luận dựa trên Biểu đồ tăng trưởng BPV.



Hình 2. Biểu đồ tăng trưởng BPV

2.3 Giá trị ngoại lai

Giá trị ngoại lai hay giá trị gây nhiễu (outliers) là những giá trị dữ liệu được ghi nhận có sự khác biệt bất thường so với những giá trị dữ liệu khác, không theo một quy tắc chung nào và có thể gây ra sự sai lệch trong kết quả phân tích và việc xây dựng các thuật toán dự đoán. Phát hiện và hiểu rõ các dữ liệu ngoại lai rất quan trọng, bởi:

- Các dữ liệu ngoại lai có thể làm sai lệch và tạo ra thiên kiến tiêu cực cho toàn bộ kết quả của một phân tích (trường hợp xấu).
- Trong một số trường hợp, các giá trị ngoại lai có thể cung cấp insight cho một kết quả phân tích (trường hợp tốt).
- Loại bỏ outlier trong tập dữ liệu sẽ giúp kết quả phân tích của bạn chính xác và sát với thực tế hơn.

2.4 Các công cụ thống kê

2.4.1 Độ trải giữa (IQR)

IQR – Interquartile Range là độ trải giữa hay còn gọi là khoảng tứ phân vị của tập dữ liệu. Nó thường được sử dụng cho khoảng biến thiên vì nó loại trừ hầu hết giá trị bất thường của dữ liệu. IQR là sự khác biệt giữa tứ phân vị thứ nhất Q_1 và tứ phân

vị thứ ba Q_3: IQR = $Q_3 - Q_1$

Với Q_1 là tứ phân vị thứ nhất (chiếm 25% trong tập dữ liệu)

Q_3 là tứ phân vị thứ ba (chiếm 75% trong tập dữ liệu)

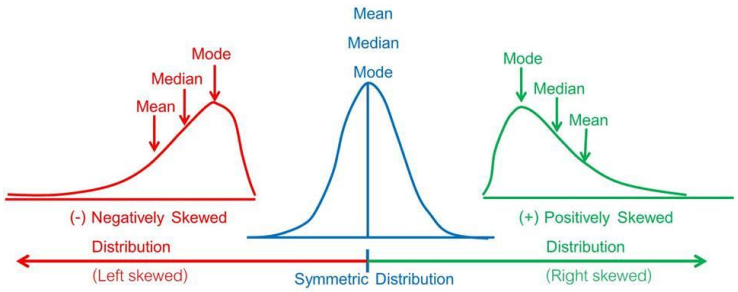
Giá trị IQR có thể sử dụng để xác định outliers bằng cách thiết lập các giá trị biên Upper/Lower như sau: Nếu ta lấy $Q_1 - 1.5 * IQR$, bất kỳ giá trị dữ liệu nào nhỏ hơn con số này được coi là giá trị outliers. Tương tự như vậy, nếu chúng lấy $Q_3 + 1.5 * IQR$, bất kỳ giá trị dữ liệu nào lớn hơn con số này được coi là outliers.

2.4.2 Độ lệch và độ nhọn

2.4.2.1 Độ lệch

Độ lệch (Skewness) là một công cụ thống kê quan trọng để đánh giá độ phân tán của các giá trị trong một tập dữ liệu, được sử dụng để mô tả sự cân bằng của phân phối.

- Biểu đồ có trọng tâm và đối xứng với cùng một hình dạng ở cả hai bên – phân phối chuẩn.
- Biểu đồ không cân đối và bị lệch sang một bên (phải hoặc trái) – phân phối lệch
  - Độ lệch nằm trong khoảng -0,5 đến 0,5, dữ liệu khá đối xứng.
  - Độ lệch nằm trong khoảng -1 đến -0,5 (độ lệch âm) hoặc từ 0,5 đến 1 (độ lệch dương), dữ liệu bị lệch vừa phải.
  - Độ lệch nhỏ hơn -1 (độ lệch âm) hoặc lớn hơn 1 (độ lệch dương), dữ liệu có độ lệch cao.



Hình 3. Các loại độ lệch

Ý nghĩa của độ lệch cho ta biết về hướng của giá trị ngoại lai nằm ở bên phải (skewness âm), nằm ở bên trái (skewness dương) hoặc ở giữa (skewness bằng 0). Dữ liệu bị lệch cao là dữ liệu không bình thường và điều đó có thể ảnh hưởng đến các bài

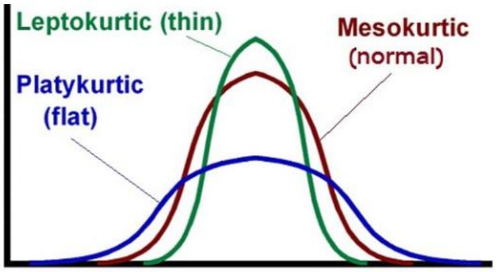
kiểm tra thống kê hoặc khả năng dự đoán của máy học. Trong những trường hợp như vậy, chúng ta cần biến đổi dữ liệu để làm cho nó bình thường. Một số kỹ thuật phổ biến được sử dụng để xử lý dữ liệu bị lệch: Chuyển đổi nhật ký, Phép biến đổi căn bậc hai, Chuyển đổi quyền lực, Phép biến đổi hàm mũ, Chuyển đổi Box-Cox, ...

2.4.2.2 Độ nhọn

Độ nhọn (Kurtosis) được sử dụng để mô tả các phân phối. Trong khi độ lệch phân biệt các giá trị cực trị ở một đuôi so với đuôi kia, thì độ nhọn đo lường các giá trị cực trị ở một trong hai đuôi.

Chọn phân phối chuẩn có Kurtosis bằng 3 làm gốc:

- Giá trị lớn hơn 3 cho biết phân phối tương đối đạt đỉnh, được gọi là leptokurtic
- Giá trị nhỏ hơn 3 cho biết phân phối tương đối bằng phẳng, được gọi là platykurtic.



Hình 4. Các loại độ nhọn

Ý nghĩa của độ nhọn:

- Phân phối có độ nhọn lớn có dữ liệu đuôi có các giá trị cực trị cao hơn đuôi của phân phối chuẩn.
- Phân phối có độ nhọn thấp có dữ liệu đuôi có các giá trị cực trị thấp hơn đuôi của phân phối chuẩn.

2.5 Biến đổi Log

Biến đổi log (Log Transform), thuộc họ power transform. Hàm này có thể được biểu diễn dưới dạng toán học như sau: $y = \log_b(x)$. Log transform rất hữu ích khi áp dụng cho các phân phối không chuẩn vì chúng có xu hướng mở rộng các giá trị nằm trong phạm vi mật độ thấp và né hoặc giảm các giá trị trong phạm vi mật độ cao. Phương pháp này giúp biến đổi phân phối bị sai lệch trở thành bình thường nhất.

2.6 Chuẩn hóa dữ liệu – StandardScaler

Các điểm dữ liệu đôi khi được đo đạc với những đơn vị khác nhau, m và feet chẳng hạn. Hoặc có hai thành phần (của vector dữ liệu) chênh lệch nhau quá lớn, một thành phần có khoảng giá trị từ 0 đến 1000, thành phần kia chỉ có khoảng giá trị từ 0 đến 1 chẳng hạn. Lúc này, chúng ta cần chuẩn hóa dữ liệu trước khi thực hiện các bước tiếp theo. Nếu đầu vào không được scaling có thể dẫn đến quá trình huấn luyện không ổn định. Đối với đề tài này, ta sử dụng phương pháp StandardScaler.

StandardScaler là một phương pháp chia tỷ lệ dựa trên trung bình. Công thức của StandardScaler là $z = \frac{x-\mu}{\sigma}$, vì vậy nó điều chỉnh giá trị trung bình về 0. StandardScaler dễ bị ảnh hưởng bởi các ngoại lệ vì các ngoại lệ ảnh hưởng đến giá trị trung bình. Nếu chúng ta có phân phối chuẩn hoặc có dữ liệu gần chuẩn, StandardScaler sẽ di chuyển dữ liệu của bạn gần hơn với phân phối chuẩn chuẩn.

2.7 Kiểm định độc lập cho giả thuyết thống kê

Kiểm định độc lập là một quá trình kiểm tra và đánh giá tính độc lập của một hoặc nhiều biến độc lập (feature - x) với một biến phụ thuộc (target - y). Đối với một bài toán phân lớp, ban đầu ta sẽ có rất nhiều biến x tham gia vào, đây là các biến mà người làm khảo sát cho rằng là nó có ảnh hưởng đến kết quả của biến y. Tại đây, ta gặp vấn đề là "Có phải tất cả biến x này đều tác động lên biến y không?". Để làm rõ câu hỏi trên, chúng ta cần phương pháp kiểm định rằng từng biến x trong khảo sát có thực sự gây ảnh hưởng lên biến y không hay chúng hoàn toàn không gây tác động và độc lập hoàn toàn đối với biến y.

2.7.1 Kiểm định T-Test

Kiểm định T-Test là phương pháp được ứng dụng thường xuyên để kiểm tra sự khác nhau của giá trị trung bình biến đơn với giá trị cho trước hoặc giá trị trung bình tổng thể. Nó phù hợp trong so sánh giữa 2 biến định lượng.

Kiểm định gồm 2 giả thuyết:

- Giả thuyết vô hiệu là H_0: $\mu_1 - \mu_2 = 0$; điều này tương đương với $\mu_1 = \mu_2$. Nói cách khác, điều này nói rằng trung bình của 2 mẫu là giống nhau, tương đương với việc nói rằng đó là một quần thể chứ không phải hai.
- Giả thuyết thay thế là H_a: $\mu_1 \neq \mu_2$

```
Independent Accreditation among Quantitative variables vs Target
    H0: μ_(x) = μ_(y) ; with x is each quantitative variable in dataset
    Ha: μ_(x) ≠ μ_(y)
```

Quy tắc bác bỏ H_0 (chọn 1 trong 2 cách sau):

- Phương pháp giá trị tới hạn, $|t| \geq t_{1-\frac{\alpha}{2}}$
- Phương pháp P-value, $p \leq \alpha$

Trong thống kê, mức ý nghĩa alpha (α) được sử dụng để đánh giá sự ý nghĩa thống kê của kết quả. Nó cho biết xác suất tối đa mà chúng ta có thể chấp nhận sai lầm loại 1 (giá trị dự đoán là "có" nhưng giá trị thực tế là "không"). Cho alpha = 0.05, tức ta chấp nhận tỉ lệ gặp sai lầm loại 1 là 5% trong kiểm định. Nếu p-value <= alpha (0.05) thì ta bác bỏ H_0, tức là chấp nhận 2 mẫu kiểm định là phụ thuộc vào nhau ở mức tin cậy 95%, ngược lại, ta bác bỏ H_0.

2.7.2 Kiểm định Chi-Square

Kiểm định độc lập Chi-Square

- Là kiểm định dùng để kiểm tra một mô hình gồm tập hợp các dữ liệu rời rạc (discrete data) có gần với phân phối chuẩn hay không, xác định sự khác biệt giữa dữ liệu quan sát và dữ liệu mong đợi là do ngẫu nhiên hay do mối quan hệ giữa các biến đang nghiên cứu.
- Là một phép thử thống kê được sử dụng để so sánh kết quả quan sát và kết quả mong đợi giữa các biến phân loại trong tổng thể dữ liệu.

Chi-Square cũng gồm có 2 giả thuyết và ý nghĩa bác bỏ giống như T-Test.

```
Independent Accreditation among Qualitative variables vs Target
    H0: μ_(x) = μ_(y) ; with x is each qualitative variable in dataset
    Ha: μ_(x) ≠ μ_(y)
```

Để chấp nhận hay bác bỏ giả thuyết H_0, ta sử dụng giá trị p. Khi p-value > α, chúng ta chấp nhận giả thuyết H_0, ngược lại, p-value ≤ α, chúng ta bác bỏ giả thuyết H_0.

2.8 Kiểm định đồng thời cho giả thuyết thống kê

2.8.1 Phương pháp OLS

Phương pháp OLS (Ordinary Least Squares – bình phương nhỏ nhất) trong thống kê là một phương pháp mạnh mẽ và phổ biến để ước tính các tham số trong mô hình hồi quy. Được sử dụng rộng rãi, OLS giúp chúng ta hiểu và dự đoán mối quan hệ giữa các biến trong một mô hình tuyến tính. Nó được sử dụng để tìm ra mối quan hệ giữa một biến phụ thuộc (biến y) và các biến độc lập (biến x). Cụ thể, mô hình này làm giảm thiểu tổng dư bình phương (the squared residuals) giữa dữ liệu quan sát và các giá trị mong đợi phù hợp với mô hình dự báo.

Nó được sử dụng để tìm ra hệ số tối ưu sao cho tổng bình phương sai số giữa giá trị dự đoán và giá trị thực tế là nhỏ nhất. Dưới đây là cách hoạt động chi tiết của phương pháp OLS:

Bước 1: Xác định mô hình hồi quy, có dạng:

$$Y = \beta_0 + \beta_1(X_1) + \beta_2(X_2) + \beta_3(X_3) + \ldots + \beta_n(X_n) + \varepsilon$$

Với $\begin{cases} \beta n \text{ là hệ số của các biến độc lập} \\ \varepsilon \text{ là sai số ngẫu nhiên không giải thích được} \end{cases}$

Bước 2: Chuẩn bị dữ liệu

Ta có một ma trận vector Y kích thước n x 1 (với n là số quan sát) và ma trận vector X kích thước n x m (với m là số biến độc lập), trong đó mỗi hàng của ma trận X tương ứng với một quan sát và mỗi cột tương ứng với một biến độc lập.

Bước 3: Ước lượng hệ số

Sử dụng phương pháp OLS, ta ước lượng các hệ số β_0, β_1, ..., β_n bằng cách tìm nghiệm tối ưu cho hàm mất mát (Loss Function) như sau:

$$\text{Loss function} = \sum(Y - X\beta)^2$$

Với $\begin{cases} \text{vector } \beta \text{ kích thước m x 1 chứa các hệ số cần ước lượng} \\ \sum \text{ là ký hiệu cho tổng các phần tử trong ma trận.} \end{cases}$

Mục tiêu là tìm giá trị β tối ưu sao cho tổng bình phương sai số là nhỏ nhất.

Bước 4: Tìm giá trị tối ưu

Để tìm giá trị β tối ưu, ta lấy đạo hàm của hàm mất mát theo β và đặt nó bằng

không, sau đó giải phương trình để tìm giá trị β. Công thức cụ thể cho việc tìm giá trị tối ưu của β là:

$$\beta = (X^T X)^{-1} (X^T Y)$$

Với X^T là ma trận chuyển vị của ma trận X

Bước 5: Kiểm định và đánh giá

Kiểm tra các giả định của mô hình và đánh giá tính phù hợp của mô hình bằng cách kiểm tra các thống kê như hệ số xác định R^2, T-test, P-value, AIC và BIC.

Ưu và khuyết điểm của OLS

Ưu điểm:

- Đơn giản và dễ hiểu: phương trình đơn giản; dễ hiểu, dễ dàng áp dụng trong nhiều bài toán.
- Hiệu quả tính toán: tính toán nhanh chóng và hiệu quả, đặc biệt là với số lượng mẫu nhỏ.
- Tính blue (Best Linear Unbiased Estimator): là ước lượng tối ưu và không chệch cho các tham số trong mô hình hồi quy tuyến tính.
- Khả năng mô hình hóa phức tạp: được áp dụng cho các mô hình hồi quy tuyến tính phức tạp, có nhiều biến độc lập và liên hệ tương quan.

Hạn chế:

- Tính nhạy cảm: nhạy cảm với các ngoại lệ (outliers) trong dữ liệu. Một ngoại lệ tương đối lớn có thể ảnh hưởng lớn đến kết quả ước lượng.
- Giả định về mô hình hồi quy tuyến tính: OLS dựa trên giả định về mô hình hồi quy tuyến tính, tức là mối quan hệ giữa biến phụ thuộc và các biến độc lập phải là một mối liên hệ tuyến tính. Khi mô hình không thỏa mãn giả định này, kết quả ước lượng OLS có thể không đáng tin cậy.
- Chống định của phương pháp OLS: là hiện tượng nội sinh (Endogeneity) cho ra ước lượng không chính xác khi các biến độc lập có quan hệ tương quan với biến sai số. Điều này cần được chú ý và xử lý để đảm bảo độ tin cậy của kết quả.
- Khả năng phân tích tương quan giữa các biến: hạn chế trong việc xác định mối

quan hệ tương quan giữa các biến độc lập. Điều này có thể ảnh hưởng đến độ tin cậy của kết quả và phân tích của mô hình.

Phương pháp OLS giúp cung cấp thông tin định lượng về mối quan hệ giữa các biến, đồng thời cho phép ước lượng và kiểm định các giả định thống kê.

2.8.2 Tiêu chuẩn AIC

Tiêu chuẩn AIC (Akaike Information Criterion) đo lường chất lượng của mô hình, gồm 2 yếu tố chính:

- Likelihood: đo lường mức độ phù hợp của tập dữ liệu khi chạy trong mô hình
- Penalty: đo kích thước mô hình, số lượng biến trong một mô hình

$$\text{AIC} = -2 \log L(\hat{\beta}, \hat{\sigma}^2) + 2p = \underbrace{n + n\log(2\pi) + n\log\left(\frac{RSS}{n}\right)}_{\text{Likelihood}} + \underbrace{2p}_{\text{Penalty}}$$

Với
- RSS (Residual sum of squares) là tổng bình phương thặng dư, đo sự chênh lệch của dự đoán so với giá trị thực tế của dữ liệu, RSS = $\sum_1^n (y_i - \hat{y}_i)^2$
- p là số lượng tham số trong mô hình

Bởi vì ta mong muốn một mô hình sẽ dự đoán chính xác, tức là sự chênh lệch giữa giá trị tính được và thực tế là nhỏ nhất nên RSS sẽ càng nhỏ càng tốt, dẫn đến likelihood sẽ nhỏ và AIC cũng phải nhỏ theo. Tóm lại, AIC càng nhỏ sẽ cho ra mô hình càng tốt.

2.8.3 Tiêu chuẩn BIC

Tiêu chuẩn BIC (Bayesian Information Criteria) cũng mang ý nghĩa tương tự AIC, BIC (cụ thể là likelihood) càng nhỏ cũng sẽ cho ra mô hình càng tốt:

$$\text{BIC} = -2 \log L(\hat{\beta}, \hat{\sigma}^2) + \log(n)p = \underbrace{n + n\log(2\pi) + n\log\left(\frac{RSS}{n}\right)}_{\text{Likelihood}} + \underbrace{\log(n)p}_{\text{Penalty}}$$

2.9 Các loại giả định hồi quy

Bất kì bài toán dự báo nào cũng cần tìm ra một hàm số f(x) phù hợp nhất. Điều này giúp làm giảm thiểu tối đa các nguồn tài nguyên của con người (thời gian, dung lượng máy tính) cho việc khởi tạo, chạy thử các mô hình dự báo và cuối cùng là chọn lọc một mô hình cho ra độ chính xác cao nhất.

2.9.1 Hồi quy tuyến tính (Linear Regression)

Hàm f(x) tuyến tính đại diện cho mối quan hệ giữa các biến độc lập và biến phụ

thuộc, giả sử như chúng có mối quan hệ tuyến tính:

$$\text{Stroke} = \beta_0 + \beta_1(\text{age}) + \beta_2(\text{hypertension}) + \ldots + \beta_9(\text{smoking\_status}) + \varepsilon$$

Nó có nghĩa là khi tuổi tăng/giảm 1 đơn vị, thì khả năng bị đột quỵ sẽ tăng/giảm β_1 lần, các biến còn lại cũng mang nghĩa tương tự. Hoặc là mô hình phi tuyến tính:

$$\text{Stroke} = \beta_0 + \beta_1(\text{age}^2) + \beta_2(\text{hypertension}) + \ldots + \beta_9(\text{smoking\_status}) + \varepsilon$$

Khi tuổi tăng/giảm 1 đơn vị, lúc này khả năng bị đột quỵ không còn tăng/giảm $_1$% nữa mà thay vào đó, khả năng bị đột quỵ sẽ tăng/giảm (số tuổi ban đầu $\pm$ 1)^2 β_1 lần.

Mỗi mô hình khác nhau sẽ cho ra ý nghĩa kết quả và cách diễn giải dự báo khác nhau. Do đó, việc tìm ra mỗi liên hệ giữa các biến độc lập đối với biến phụ thuộc bằng hàm f(x) sẽ giúp người thực hiện mô hình giảm đi chi phí, thời gian, tài nguyên máy tính một cách hiệu quả. Đây là 2 cách kiểm định chất lượng mô hình hồi quy tuyến tính:

- MSE (Mean Squared Error): trung bình của bình phương sai số giữa giá trị dự đoán và giá trị thực tế.
- R^2: cho biết mức độ phù hợp của mô hình nghiên cứu với ý nghĩa là các nhân tố (hay còn gọi là các biến). Đồng thời, hệ số này giải thích nhân tố phụ thuộc đó đạt bao nhiêu phần trăm trong quá trình nghiên cứu.

Mục đích của hồi quy tuyến tính là ước tính giá trị cho các hệ số mô hình và áp dụng mô hình vào dữ liệu huấn luyện sao cho giá trị của MSE tối thiểu và R^2 tối đa, cuối cùng là dự đoán đầu ra y.

2.9.2 Hồi quy Logistic (Logistic Regression)

Bài toán của đề tài trên có kết quả là 0/1 tương đương với không/có nguy cơ bị đột quỵ - giá trị đầu ra của bài toán có tính nhị phân. Vì vậy, ta không thể tính toán được giá trị dự đoán chính xác bằng 0 hoặc 1, tuy nhiên, ta có thể tìm ra xác xuất để kết luận là 0 hoặc 1. Dựa vào xác xuất thống kê, ta có: $\hat{y}$ = P (y = 1 | x). Nó có nghĩa là kết quả xác xuất để dự báo rằng bệnh nhân có thể mắc đột quỵ (y) dựa theo các thông tin từ các biến độc lập (x). Để tính toán ra được xác xuất trên, ta sẽ phải giải bài toán bằng phương pháp hồi quy Logistic.

Hồi quy Logistic là một loại thuật toán học giám sát, tính toán mối quan hệ giữa

các biến độc lập trong đầu vào và đầu ra dựa trên hàm Logistic (còn gọi là hàm Sigmoid). Mặc dù gọi là hồi quy Logistic nhưng thuật toán này không dự đoán ra giá trị thực như các thuật toán hồi quy khác, hồi quy Logistic được dùng để dự đoán ra một kết quả nhị phân (với giá trị 0/1 hay -1/1 hay True/False) dựa vào các giá trị đầu vào của nó. Nhưng hồi quy Logistic cũng có một chút giống với hồi quy tuyến tính trong quá trình xây dựng model.

Hồi quy Logistic cũng có mục đích tương tự với hồi quy tuyến tính nhưng với một bổ sung - nó chạy kết quả thông qua một hàm phi tuyến tính (non-linear) đặc biệt được gọi là hàm Logistic hoặc hàm Sigmoid để tạo ra đầu ra là một xác suất P.

Công thức hồi quy của mô hình hồi quy Logistic:

Bước 1: Tìm hàm f(x)

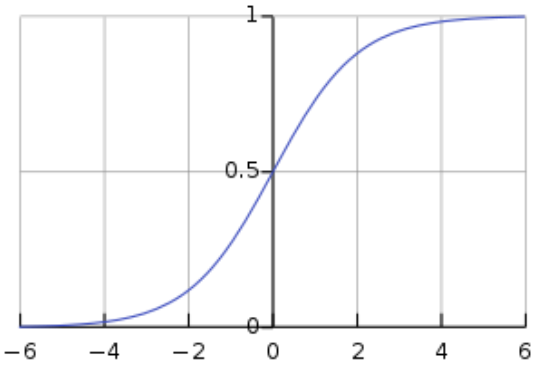
$$f(x) = B*X + \beta_0$$

Với $\begin{cases} \text{B là tập hợp } \{\beta_1, \dots \beta_n\} \\ \text{X là tập các giá trị quan sát của một bệnh nhân} \end{cases}$

Đối với một dataframe, phép nhân ở đây chính là phép nhân ma trận giữa B và X.

Bước 2: Tìm hàm Logistic

Hàm Logistic, hoặc hàm Sigmoid, được ký hiệu là S(x) với đầu ra là một số có giá trị từ 0 đến 1 được định nghĩa với công thức: $s(x) = \frac{1}{1+e^{-f(x)}}$



Hình 5. Hàm Sigmoid

Giá trị hàm Sigmoid	Giá trị hàm f(x)	Kết quả dự báo
s(x) < 0.5	khi f(x) < 0	y = 0
s(x) ≥ 0.5	khi f(x) ≥ 0	y = 1

Bảng 1. Ý nghĩa của hàm Sigmoid

2.10 Các mô hình học máy phân lớp

2.10.1 K-Nearest Neighbors

Mô hình KNN tính toán độ tương tự bằng cách sử dụng khoảng cách giữa hai điểm trên biểu đồ. Khoảng cách giữa các điểm càng lớn thì chúng càng ít giống nhau. Có nhiều cách tính khoảng cách giữa các điểm, nhưng thước đo khoảng cách phổ biến nhất chỉ là khoảng cách Euclide (khoảng cách giữa hai điểm trên một đường thẳng).

KNN là một thuật toán học có giám sát, nghĩa là các ví dụ trong tập dữ liệu phải được gán nhãn cho chúng/các lớp của chúng phải được biết đến. Có hai điều quan trọng khác cần biết về KNN. Đầu tiên, KNN là một thuật toán phi tham số. Điều này có nghĩa là không có giả định nào về tập dữ liệu được đưa ra khi mô hình được sử dụng. Thay vào đó, mô hình được xây dựng hoàn toàn từ dữ liệu được cung cấp. Thứ hai, không có sự phân chia tập dữ liệu thành tập huấn luyện và tập kiểm tra khi sử dụng KNN. KNN không khái quát hóa giữa tập huấn luyện và tập kiểm tra, vì vậy tất cả dữ liệu huấn luyện cũng được sử dụng khi mô hình được yêu cầu đưa ra dự đoán.

Thuật toán KNN trải qua ba giai đoạn chính khi nó được thực hiện:

1. Đặt K thành số "hàng xóm" đã chọn.
2. Tính toán khoảng cách giữa một ví dụ được cung cấp/thử nghiệm và các ví dụ về tập dữ liệu.
3. Sắp xếp các khoảng cách được tính toán.
4. Lấy nhãn của K mục hàng đầu.
5. Trả về một dự đoán về ví dụ thử nghiệm.

Trong bước đầu tiên, K được người dùng chọn và nó cho thuật toán biết có bao nhiêu hàng xóm (bao nhiêu điểm dữ liệu xung quanh) nên được xem xét khi đưa ra phán đoán điểm dữ liệu mới thuộc lớp nào. Trong bước thứ hai, lưu ý rằng mô hình sẽ kiểm tra khoảng cách giữa ví dụ mục tiêu và mọi ví dụ trong tập dữ liệu. Các khoảng cách sau đó được thêm vào một danh sách và được sắp xếp. Sau đó, danh sách đã sắp xếp được kiểm tra và các nhãn cho K phần tử trên cùng được trả về. Nói cách khác, nếu K được đặt thành 5, mô hình sẽ kiểm tra nhãn của 5 điểm dữ liệu gần nhất trên cùng với điểm dữ liệu đích. Khi hiển thị dự đoán về điểm dữ liệu đích, điều quan trọng là tác vụ đó là một hồi quy or phân loại nhiệm vụ. Đối với tác vụ hồi quy, giá trị trung

bình của K nhãn trên cùng được sử dụng, trong khi chế độ của K nhãn trên cùng được sử dụng trong trường hợp phân loại

2.10.2 Cây quyết định (Decision Tree)

Decision Tree là một trong những mô hình có khả năng diễn giải cao và có thể thực hiện cả nhiệm vụ classification và regression. Để thực hiện các mô hình tuyến tính cổ điển, người dùng phải đảm bảo dữ liệu được sử dụng để đào tạo mô hình không có các bất thường như giá trị ngoại lệ cần được xử lý, giá trị bị thiếu, đa cộng tuyến cần được giải quyết.

Chúng ta không cần phải thực hiện bất kỳ loại xử lý trước dữ liệu nào trước đó. Decision Tree đủ mạnh để xử lý tất cả các loại vấn đề như vậy để đi đến quyết định. Bên cạnh đó, Decision Tree có khả năng xử lý dữ liệu phi tuyến mà các mô hình tuyến tính cổ điển không xử lý được.

Decision Trees gồm 3 phần chính: node gốc (root node), node lá (leaf nodes) và các nhánh nhỏ (branches). Trong đó node gốc là điểm bắt đầu của Decision Trees và cả hai node gốc và node chứa câu hỏi hoặc tiêu chí để được trả lời. Các nhánh nhỏ thể hiện kết quả của kiểm tra trên nút.

2.10.3 Bernoulli Naive Bayes

Naive Bayes Classification (NBC) là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes. Theo định lý Bayes, ta có công thức tính xác suất ngẫu nhiên của y khi biết x như sau: $P(y \mid x) = \frac{P(x \mid y)\, P(y)}{P(x)}$

Cụ thể ở mô hình Bernoulli, biến phụ thuộc y cho các giá trị nhị phân 0, 1. Trong đó 0 thể hiện "không", 1 thể hiện "có" xác suất xảy ra đột quỵ.

$$\text{Xác suất } P(x_i \mid y) = P(i \mid y)\, x_i + (1 - P(i \mid y))\,(1 - x_i)$$

Với $P(i \mid y)$ là tỉ lệ số lần từ x_i xuất hiện trong toàn bộ tập training data có nhãn y.

2.11 Các phương pháp đánh giá mô hình phân lớp

Trong quá trình xây dựng một mô hình Machine Learning, một phần không thể thiếu để xét xem mô hình có chất lượng tốt hay không chính là đánh giá mô hình. Đánh giá mô hình giúp chúng ta chọn lựa được các mô hình phù hợp với bài toán cụ

thể. Để có thể áp dụng đúng thước đo đánh giá mô hình phù hợp, chúng ta cần hiểu bản chất, ý nghĩa cũng như các trường hợp sử dụng nó.

2.11.1 Kiểm chứng chéo (Cross validation)

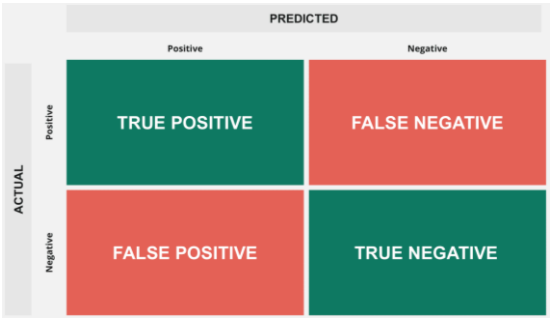
Cross validation là một kỹ thuật lấy mẫu để đánh giá mô hình học máy. Tham số quan trọng trong kỹ thuật này là k, đại diện cho số nhóm mà dữ liệu sẽ được chia ra. Vì lý do đó, nó được mang tên k-fold cross-validation. Khi giá trị của k được lựa chọn, người ta sử dụng trực tiếp giá trị đó trong tên của phương pháp đánh giá. Ví dụ với k=10, phương pháp sẽ mang tên 10-fold cross-validation.

Kỹ thuật này thường bao gồm các bước như sau:

1. Xáo trộn dataset một cách ngẫu nhiên
2. Chia dataset thành k nhóm
3. Với mỗi nhóm:
    - 3.1. Sử dụng nhóm hiện tại để đánh giá hiệu quả mô hình
    - 3.2. Các nhóm còn lại được sử dụng để huấn luyện mô hình
    - 3.3. Huấn luyện mô hình
    - 3.4. Đánh giá và sau đó hủy mô hình
4. Tổng hợp hiệu quả của mô hình dựa từ các số liệu đánh giá

2.11.2 Ma trận nhầm lẫn (Confusion matrix)

Confusion matrix là tổng kết giữa nhãn thực tế của dữ liệu với nhãn do mô hình dự đoán ra.



Hình 6. Confusion Matrix

Với
- True Positive (TP) là thực tế và dự đoán đều cho ra kết quả "có"
- True Negative (TN) là thực tế và dự đoán đều cho ra kết quả "không"
- False Negative (FN) là sai lầm loại I, thực tế là "không" nhưng dự đoán "có"
- False Positive (FP) là sai lầm loại II, thực tế là "có" nhưng dự đoán "không"

Có một cách dễ nhớ hơn như sau:

- True/False chỉ những gì ta đã dự đoán là đúng hay chưa
- Positive/Negative tương đương với kết quả thực tế dương tính hay âm tính

Nói cách khách, nếu thấy chữ True tức là dự đoán là đúng còn False thì ngược lại.

2.11.3 Báo cáo tổng hợp (Classification report)

Classification report gồm Precision, Recall, F1-Score:

- **Precision** là tỷ lệ giữa số lượng quan sát dự đoán được tính là True Positive (TP) với tổng số quan sát được phân loại là Positive

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

- **Recall** đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ các mẫu thuộc nhóm Positive.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

- **F1-score** là một metric kết hợp cả Recall và Precision.

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

2.11.4 AUROC (Area Under The Receiver Operating Characteristics)

ROC Curve (The receiver operating characteristic curve) là một đường cong biểu diễn hiệu suất phân loại của một mô hình phân loại tại các ngưỡng threshold. Về cơ bản, nó hiển thị True Positive Rate (TPR) so với False Positive Rate (FPR) đối với các giá trị ngưỡng khác nhau. Các giá trị TPR, FPR được tính như sau:

$$\text{TPR} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\text{FPR} = \frac{\text{False Positive}}{\text{True Negative + False Negative}}$$

ROC tìm ra TPR và FPR ứng với các giá trị ngưỡng khác nhau và vẽ biểu đồ để dễ dàng quan sát TPR so với FPR.

AUC là chỉ số được tính toán dựa trên đường cong ROC nhằm đánh giá khả năng phân loại của mô hình tốt như thê nào. Phần diện tích nằm dưới đường cong ROC và trên trục hoành chính là AUC, có giá trị nằm trong đoạn [0, 1]. Khi diện tích này càng lớn, đường cong này sẽ dần tiệm cận với đường thẳng y = 1 tương đương với

khả năng phân loại của mô hình càng tốt. Còn khi đường cong ROC nằm sát với đường chéo đi qua hai điểm (0, 0) và (1, 1), mô hình sẽ tương đương với một phân loại ngẫu nhiên.

Chương 3: Xây dựng mô hình kết luận

3.1 Tìm hiểu bộ dữ liệu

Bộ dữ liệu thô ban đầu có 5110 dòng và 12 cột:

```
df.head(10)
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
6	53882	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
7	10434	Female	69.0	0	0	No	Private	Urban	94.39	22.8	never smoked	1
8	27419	Female	59.0	0	0	Yes	Private	Rural	76.15	NaN	Unknown	1
9	60491	Female	78.0	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1

```
df.shape
```

```
(5110, 12)
```

Hình 7. Bộ dự liệu thô

Sau khi kiểm tra và loại bỏ các giá trị trùng lặp, bộ dữ liệu không thay đổi gì, tức là mỗi dòng dữ liệu đại diện cho một bệnh nhân (một quan sát) độc lập.

```
pd.DataFrame((df.shape, df.drop_duplicates().shape), columns = ['Row', 'Column'],
             index = ['Before deleting dulicated values', 'After deleting dulicated values'])
```

	Row	Column
Before deleting dulicated values	5110	12
After deleting dulicated values	5110	12

Hình 8. Kiểm tra giá trị quan sát trùng lặp

Các cột dữ liệu gồm:

STT	Tên cột	Diễn giải	Số lượng giá trị	Kiểu dữ liệu
0	Id	Mã id của bệnh nhân	5110	Int64
1	Gender	Giới tính - Male (nam) - Female (nữ) - Other (khác)	5110	Object
2	Age	Tuổi	5110	Float64

3	Hypertension	Cao huyết áp (0, 1)	5110	Int64
4	Heart_disease	Bệnh tim (0, 1)	5110	Int64
5	Ever_married	Kết hôn - Yes / No	5110	Object
6	Work_type	Nơi làm việc - Private (công ty tư nhân) - Self-employed (tự kinh doanh) - Govt_job (nhà nước) - Children (trẻ em) - Never_worked (không đi làm)	5110	Object
7	Residence_type	Nơi ở - Urban (thành phố) - Rural (nông thôn)	5110	Object
8	Avg_glucose_level	Mức độ đường trung bình (mg/dl)	5110	Float64
9	Bmi	Chỉ số sức khỏe BMI	4909	Float64
10	Smoking_status	Hút thuốc - Formerly Smoked (từng hút) - Never Smoked (không hút) - Smokes (đang hút) - Unknown (chưa xác định)	5110	Object
11	Stroke	Đột quỵ (0, 1)	5110	Int64

Bảng 2. Thông tin bộ dữ liệu

3.2 Tiền xử lí dữ liệu

3.2.1 Kiểm tra kiểu dữ liệu và số lượng dữ liệu từng cột

Từ bộ dữ liệu, ta biết rằng:

- Biến độc lập id, gender, age, avg_glucose_level có kiểu dữ liệu định lượng.
- Biến độc lập hypertension, heart_disease, ever_married, work_type, Residence_type, smoking_status và biến phụ thuộc stroke có kiểu dữ liệu định tính.

Nhưng từ df.info() hoặc tra bảng 1 ta có biến age, hypertension, heart_disease và biến

phụ thuộc stroke cần được thay đổi kiểu dữ liệu; biến bmi cần được xử lí giá trị bị khuyết (mục 3.2.3); và xóa bỏ cột id không cần thiết.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

Hình 9. Các biến cần được xử lí kiểu và số lượng giá trị

Thay đổi kiểu dữ liệu bằng astype() và xóa bỏ cột id:

```
df['age'] = df['age'].astype(int)
df['hypertension'] = df['hypertension'].astype(str)
df['heart_disease'] = df['heart_disease'].astype(str)
df['stroke'] = df['stroke'].astype(str)

# Drop unecessary column
df.drop(['id'], axis = 1, inplace = True)
```

Kết quả:

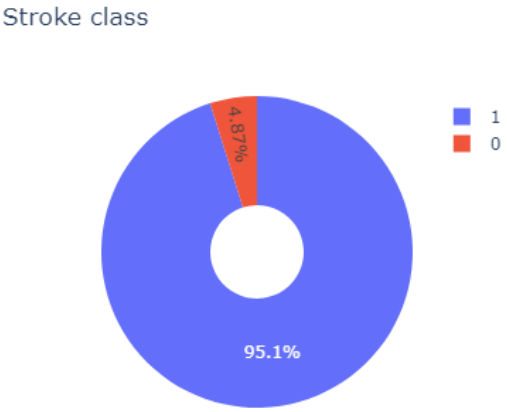
```
df.dtypes
```

```
gender                object
age                    int32
hypertension          object
heart_disease         object
ever_married          object
work_type             object
Residence_type        object
avg_glucose_level    float64
bmi                  float64
smoking_status        object
stroke                object
dtype: object
```

Hình 10. Sau khi xử lí đúng về kiểu dữ liệu

3.2.2 Kiểm tra tính cân bằng trong bộ dữ liệu

```
fig = go.Figure()
fig = px.pie(df['stroke'],
        values = df['stroke'].value_counts(),
        names = df['stroke'].unique(),
        hole = 0.3,
        title = 'Stroke class')
fig.update_layout(autosize = False, width = 400, height = 400)
fig.show()
```



Hình 11. Tỉ lệ phân lớp

```
print('The amount of each class of stroke:')
print(df['stroke'].value_counts())

class0 = len(df[df['stroke'] == '0'])
class1 = len(df[df['stroke'] == '1'])
print(f'⇒ The percentage of balance of data = {round(class1 / class0, 4) * 100}%')
```

Kết quả:

```
The amount of each class of stroke:
0    4861
1     249
Name: stroke, dtype: int64
⇒ The percentage of balance of data = 5.12%
```

- 4861 bệnh nhân không bị đột quỵ (stroke = 0)
- 249 bệnh nhân bị đột quỵ (stroke = 1)

Điều này có nghĩa mức cân bằng của bộ dữ liệu này chỉ đạt 5.12%.

Ta sẽ xử lí vấn đề mất cân bằng trên bằng kĩ thuật tăng số lượng nhóm stroke = 1 (Up-sample Minority Class) sao cho nó bằng số lượng của nhóm còn lại. Đây là quá trình nhân bản tự động các bệnh nhân của nhóm stroke = 1 để nó cân bằng với số lượng của nhóm còn lại (các quan sát từ 249 sẽ được lặp tự động lên đến 4861).

```
# Separate majority and minority classes
df_majority = df[df['stroke'] == '0']
df_minority = df[df['stroke'] == '1']

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                        replace = True,       # sample with replacement
                        n_samples = class0,   # to match majority class
                        random_state = 42)    # reproducible results

# Combine majority class with upsampled minority class
df = pd.concat([df_majority, df_minority_upsampled])

# Display new class counts
print('The amount of each class of stroke (after upsample)')
print(df['stroke'].value_counts())
print('The shape of df:', df.shape)
```

Kết quả:

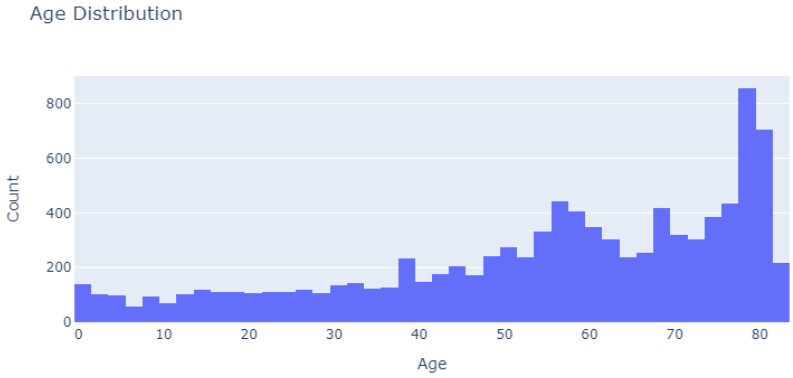
```
The amount of each class of stroke (after upsample)
0    4861
1    4861
Name: stroke, dtype: int64
The shape of df: (9722, 11)
```

3.2.3 Xử lí cột bmi bị khuyết giá trị

Để xử lí cột bmi bị khuyết giá trị một cách liên kết với tập dữ liệu, chúng ta tận dụng lại cơ sở lý thuyết về mối liên hệ giữa BMI và tuổi tác. Trước tiên, ta sẽ chia độ tuổi thành 3 nhóm dựa theo nhân khẩu học của người Việt Nam: trẻ em (dưới 16 tuổi), người trưởng thành (từ 16 đến 64 tuổi), người lớn tuổi (đủ 65 tuổi trở lên). Tìm ra giá trị BMI trung bình của từng nhóm tuổi trên và điền khuyết cột bmi bằng giá trị trung bình tương ứng với từng nhóm tuổi.

Bước 1: Xác định phân phối tuổi

```
fig = go.Figure()
fig = px.histogram(x = df['age'], title = 'Age Distribution')
fig.update_layout(autosize = False, width = 800, height = 400,
                  xaxis = go.layout.XAxis(linewidth = 0.2, mirror = True, title ='Age'),
                  yaxis = go.layout.YAxis(linewidth = 0.2, mirror = True, title ='Count'))
fig.show()
```



Hình 12. Biểu đồ phân phối tuổi

Bước 2: Phân chia ra 3 nhóm tuổi

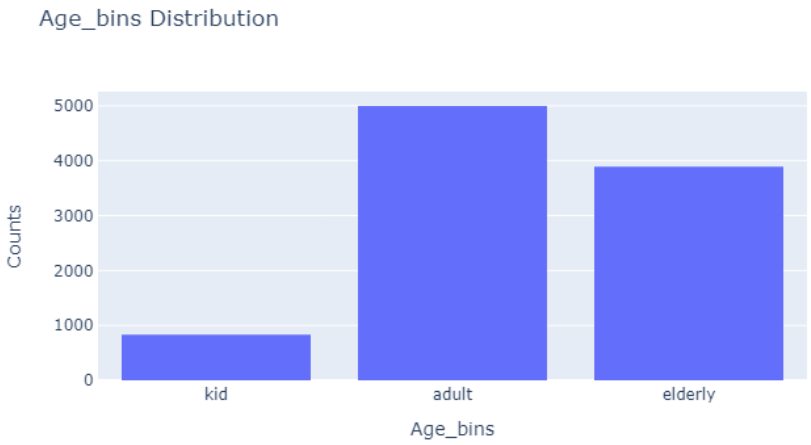
```
# Seperate Age into 3 groups
age_bins = [0, 16, 65, 100]
age_names = ('kid', 'adult', 'elderly')
df['age_bins'] = pd.cut(df['age'], age_bins, labels = age_names, include_lowest = True)

df['age_bins'].value_counts()
```

Kết quả:

```
adult      4998
elderly    3891
kid         833
Name: age_bins, dtype: int64
```

Đây là biểu đồ thể hiện số lượng của từng nhóm tuổi

```
fig = go.Figure()
fig = px.histogram(x = df['age_bins'], title = 'Age_bins Distribution')
fig.update_layout(autosize = False, width = 700, height = 400,
                  xaxis = go.layout.XAxis(linewidth = 1, mirror = True, title = 'Age_bins'),
                  yaxis = go.layout.YAxis(linewidth = 1, mirror = True, title = 'Counts'))
fig.show()
```



Hình 13. Biểu đồ thể hiện số lượng của từng nhóm tuổi

Bước 3: Tính giá trị trung bình (median) cho từng nhóm tuổi

Tạo 1 dataframe mới (df_age_treat) gồm age, age_bins, bmi và df_age_gr để

nhóm theo age_bins của df_age_treat bằng groupby để tính giá trị trung bình cho 2 cột age và bmi.

```
# Create a new dataframe with age, age_bins, bmi
df_age_treat = df[['age','age_bins', 'bmi']]
df_age_treat.head(5)

# Calculate median value for bmi basing on each age group
df_age_gr = df_age_treat.groupby(['age_bins'], as_index = False).median()
df_age_gr
```

	age	age_bins	bmi
249	3	kid	18.0
250	58	adult	39.2
251	8	kid	17.6
252	70	elderly	35.9
253	14	kid	19.1

	age_bins	age	bmi
0	kid	8.0	19.6
1	adult	50.0	30.0
2	elderly	77.0	28.3

Hình 14. Giá trị trung bình của từng nhóm tuổi

- Đối với nhóm trẻ em (kid), chúng có bmi trung bình là 19.6
- Đối với nhóm người trưởng thành (adult), họ có bmi trung bình là 30.0
- Đối với nhóm người lớn tuổi (elderly), họ có bmi trung bình là 28.3

Tạo thêm một dataframe giả (df_dummy) từ cột age_bins. Cột age_bins gồm 3 giá trị: kid, adult, elderly nên df_dummy sẽ tạo ra 3 cột mới, mỗi cột tương ứng với 1 giá trị của age_bins. Giá trị của df_dummy sẽ được gán 0 hoặc 1 theo giá trị của age_bins. Thay thế giá trị 1 của từng cột trong df_dummy bằng giá trị trung bình của bmi đã tính ở df_age_gr.

Vd: Tại dòng index 249, bệnh nhân này thuộc nhóm kid thì ở df_dummy, cột kid sẽ có giá trị bằng 1, 2 cột còn lại sẽ bằng 0.

```
# Create a new dummy dataframe for age_bins
df_dummy = pd.get_dummies(df['age_bins'])

# Replace value = 1 of each row with the median value of bmi, basing on age group
df_dummy['kid'].replace(1, df_age_gr['bmi'].loc[0], inplace = True)
df_dummy['adult'].replace(1, df_age_gr['bmi'].loc[1], inplace = True)
df_dummy['elderly'].replace(1, df_age_gr['bmi'].loc[2], inplace = True)
df_dummy
```

	kid	adult	elderly
249	1	0	0
250	0	1	0
251	1	0	0
252	0	0	1
253	1	0	0
...	...	...	...
178	0	0	1
174	0	0	1
238	0	1	0
3	0	1	0
134	0	0	1

9722 rows × 3 columns

	kid	adult	elderly
249	19.6	0	0.0
250	0.0	30	0.0
251	19.6	0	0.0
252	0.0	0	28.3
253	19.6	0	0.0
...	...	...	...
178	0.0	0	28.3
174	0.0	0	28.3
238	0.0	30	0.0
3	0.0	30	0.0
134	0.0	0	28.3

9722 rows × 3 columns

Hình 15. df_dummy trước và sau khi thay giá trị trung bình

Bước 4: Điền vào giá trị bị khuyết của cột bmi

Phải ghép 2 dataframe df và df_dummy thành một df. Tại mỗi giá trị bị khuyết của cột bmi: khi age_bins là 1 trong 3 giá trị kid; adult; elderly thì 1 trong 3 cột cuối tương ứng của dataframe sẽ thay thế giá trị của mình vào giá trị NaN của bmi. Sau khi hoàn thành lấp đầy giá trị rỗng của bmi, ta sẽ xóa bỏ 4 cột vừa tạo: kid, adult, elderly, age_bins và làm mới lại các chỉ số index của dataframe.

```
# Concat the dataframe with the dummy
df = pd.concat([df, df_dummy], axis = 1)

df['bmi'] = np.where(df['age_bins'] == 'kid', df['bmi'].fillna(df['kid']), df['bmi'])
df['bmi'] = np.where(df['age_bins'] == 'adult', df['bmi'].fillna(df['adult']), df['bmi'])
df['bmi'] = np.where(df['age_bins'] == 'elderly', df['bmi'].fillna(df['elderly']), df['bmi'])
df['bmi']
```

```
249    18.0
250    39.2
251    17.6
252    35.9
253    19.1
       ...
178    28.3
174    28.3
238    28.4
3      34.4
134    26.7
Name: bmi, Length: 9722, dtype: float64
```

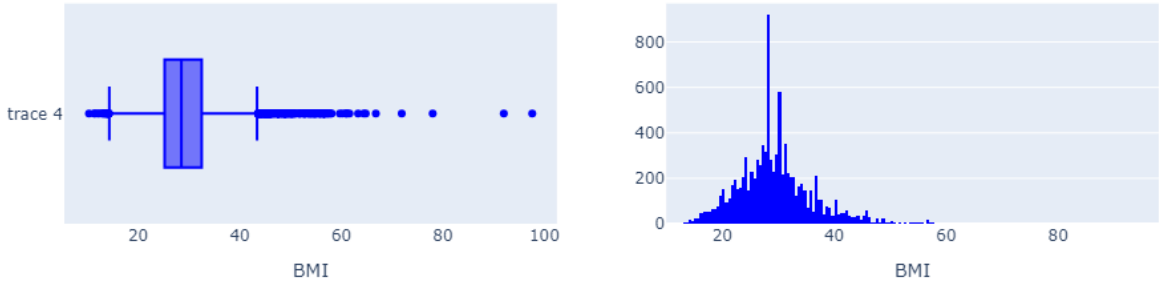
Hình 16. Cột BMI sau khi xử lí các giá trị bị khuyết

3.2.4 Làm sạch và chọn lọc biến định lượng

3.2.4.1 Kiểm tra giá trị ngoại lai

Vẽ biểu đồ hộp (box plot) và biểu đồ phân phối (histogram) để làm rõ sự phân phối của các giá trị trong từng biến.

```
fig = make_subplots(rows = 3, cols  = 2)
fig.add_trace(go.Box(x = df['age'], marker_color = 'green'), row = 1, col = 1)
fig.add_trace(go.Histogram(x = df['age'], marker_color = 'green'), row = 1, col = 2)
fig.add_trace(go.Box(x = df['avg_glucose_level'], marker_color = 'goldenrod'), row = 2, col = 1)
fig.add_trace(go.Histogram(x = df['avg_glucose_level'], marker_color = 'goldenrod'), row = 2, col = 2)
fig.add_trace(go.Box(x = df['bmi'], marker_color = 'blue'), row = 3, col = 1)
fig.add_trace(go.Histogram(x = df['bmi'], marker_color = 'blue'), row = 3, col = 2)

fig.update_xaxes(title_text = 'Age', showgrid = False, row = 1, col = 1)
fig.update_xaxes(title_text = 'Age', row = 1, col = 2)
fig.update_xaxes(title_text = 'Avg glucose', showgrid = False, row = 2, col = 1)
fig.update_xaxes(title_text = 'Avg glucose', row = 2, col = 2)
fig.update_xaxes(title_text = 'BMI', showgrid = False, row = 3, col = 1)
fig.update_xaxes(title_text = 'BMI', row = 3, col = 2)

fig.update_layout(showlegend = False, title_text = 'Qualitative variables with Box plot and Distribution plot',
                  height = 800)
fig.show()
```



Hình 17. Biểu diễn các biến định tính dưới dạng Biểu đồ Hộp và Phân phối

Từ kết quả trực quan trên, biến age không có giá trị ngoại lai, ngược lại, biến avg_glucose_level và bmi thì xuất hiện giá trị ngoại lai. Đối với 2 biến cần loại bỏ giá trị nhiễu, bởi vì các giá trị của chúng không có dạng phân phối chuẩn, nên ta sẽ dùng phương pháp thống kê IQR.

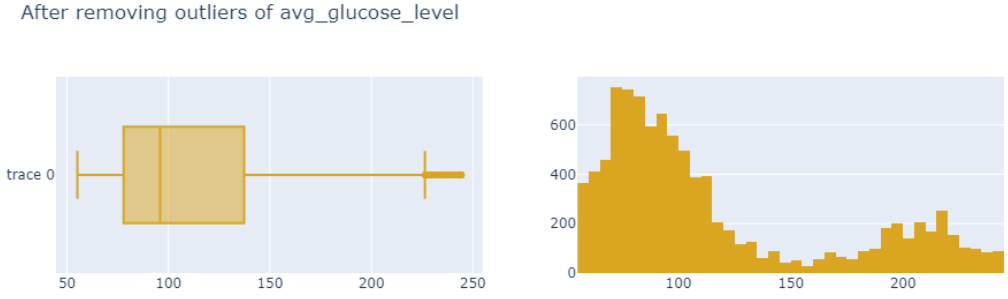
```
def remove_outlier(df, var, color):
    q3, q1 = np.percentile(df[var], [75 ,25])
    iqr = q3 - q1
    upperlimit = q3 + iqr * 1.5
    lowerlimit = q1 - iqr * 1.5
    print(f'The IQR of {var}:' , iqr)
    print(f'The upper limit of {var}:' , upperlimit)
    print(f'The lower limit of {var}:' , lowerlimit)
    df = df.drop(df[df[var] > upperlimit].index | df[df[var] < lowerlimit].index)

    fig = make_subplots(rows = 1, cols = 2)
    fig.add_trace(go.Box(x = df[var], marker_color = color), row = 1, col = 1)
    fig.add_trace(go.Histogram(x = df[var], marker_color = color), row = 1, col = 2)
    fig.update_layout(showlegend = False, title_text = f'After removing outliers of {var}', height=350)
    fig.show()
```

1. Biến avg_glucose_level

```
remove_outlier(df, 'avg_glucose_level', 'goldenrod')
```

Kết quả:

```
The IQR of avg_glucose_level: 66.78750000000001
The upper limit of avg_glucose_level: 245.08125
The lower limit of avg_glucose_level: -22.06875000000001
```



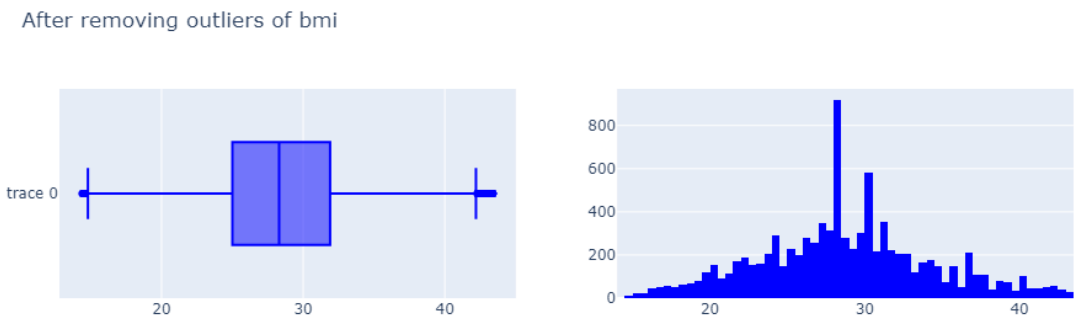
Hình 18. Sau khi loại bỏ nhiễu cho Glucose

2. Biến bmi

remove_outlier(df, 'bmi', 'blue')

Kết quả:

```
The IQR of bmi: 7.274999999999999
The upper limit of bmi: 43.412499999999994
The lower limit of bmi: 14.312500000000004
```



Hình 19. Sau khi loại bỏ nhiễu cho BMI

3.2.4.2 Kiểm tra độ lệch và độ nhọn

```
# Calculate skewness and kurtosis for 3 variables
age_s = skew(df['age'], axis = 0, bias = True)
age_k = kurtosis(df['age'], axis = 0, bias = True)
glu_s = skew(df['avg_glucose_level'], axis = 0, bias = True)
glu_k = kurtosis(df['avg_glucose_level'], axis = 0, bias = True)
bmi_s = skew(df['bmi'], axis = 0, bias = True)
bmi_k = kurtosis(df['bmi'], axis = 0, bias = True)

# Create a def to validate skewness and kurtosis
def skew(val_s):
    if val_s > -0.5 and val_s < 0.5:
        return(' Data is quite symmetry')
    elif val_s > -1 and val_s < 1:
        return(' Data is moderately deviated')
    else:
        return(' Data is highly deviated')

def kurt(val_k):
    if val_k == 3:
        return('Mesokurtic - Data is quite symmetry')
    elif val_k > 3:
        return('Leptokurtic - Data with a high probability of outliers')
    else:
        return('Platykurtic - Data with a low probability of outliers')

pd.DataFrame([[age_s, skew(age_s), age_k, kurt(age_k)],
              [glu_s, skew(glu_s), glu_k, kurt(glu_k)],
              [bmi_s, skew(bmi_s), bmi_k, kurt(bmi_k)]], index=['Age', 'Avg glucose', 'BMI'],
              columns=['Skewness', 'Meaning of skewness', 'Kurtosis', 'Meaning of kurtosis'])
```

	Skewness	Meaning of skewness	Kurtosis	Meaning of kurtosis
Age	-0.774139	Data is moderately deviated	-0.374816	Platykurtic - Data with a low probability of o...
Avg glucose	1.036868	Data is highly deviated	-0.271317	Platykurtic - Data with a low probability of o...
BMI	0.989251	Data is moderately deviated	3.510839	Leptokurtic - Data with a high probability of ...

Bảng 3. Đánh giá độ nhọn và độ lệch của biến định tính

Bảng kết quả trên có nghĩa:

- Biến age, skewness ≈ -0.77 ∈ (-1; -0.5), phân phối dữ liệu bị lệch nhẹ sang trái; kurtosis ≈ -0.37 (< 3), dữ liệu ít gặp các biến ngoại lai.
- Biến avg_glucose_level, skewness ≈ 1.04 ∈ (1; +∞), dữ liệu bị lệch nhiều sang phải; kurtosis ≈ -0.27 (< 3), dữ liệu ít gặp các biến ngoại lai.
- Biến bmi, skewness ≈ 0.99 ∈ (0.5; 1), dữ liệu bị lệch không đáng kể sang phải; kurtosis ≈ 3.51 (> 3), dữ liệu thường bắt gặp các biến ngoại lai.

Ta thấy biến avg_glucose_level có phân phối bị lệch nhiều sang phải, như đã nói ở trên, khi dữ liệu bị lệch nhiều tức là dữ liệu này đang không bình thường và sẽ gây ảnh hưởng đến kết quả phân lớp cuối cùng. Do vậy, chúng ta chọn phương pháp biến đổi Log để giải quyết độ lệch này.

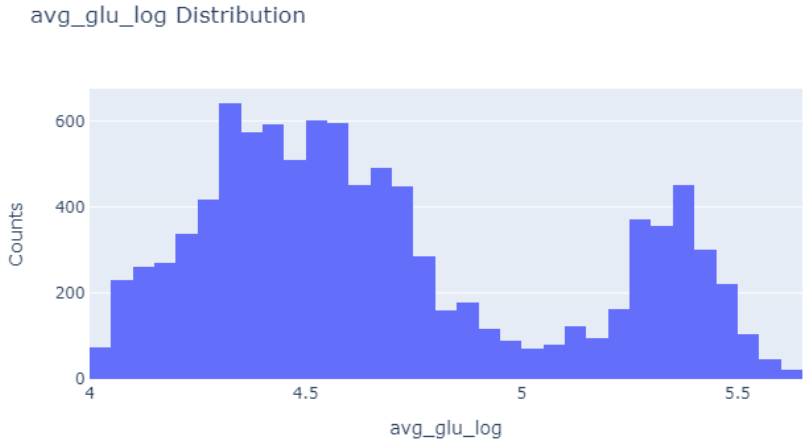
Cho b = e, tức avg_glu_log = $\log_e$(avg_glucose_level).

```
# Regular Avg gulcose
df['avg_glu_log'] = np.log((1 + df['avg_glucose_level']))
col1 = df.pop('avg_glu_log')
df.insert(8, col1.name, col1)
df.head(5)
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	avg_glu_log	bmi	smoking_status	stroke
0	Male	3	0	0	No	children	Rural	95.12	4.565597	18.0	Unknown	0
1	Male	58	1	0	Yes	Private	Urban	87.96	4.488187	39.2	never smoked	0
2	Female	8	0	0	No	Private	Urban	110.89	4.717516	17.6	Unknown	0
3	Female	70	0	0	Yes	Private	Rural	69.04	4.249067	35.9	formerly smoked	0
4	Male	14	0	0	No	Never_worked	Rural	161.28	5.089323	19.1	Unknown	0

Cột avg_glucose_level là dữ liệu ban đầu, sau khi chuyển sang dạng log – cột avg_glu_log – giá trị được làm nhỏ lại rất nhiều.

```
fig = go.Figure()
fig = px.histogram(x = df['avg_glu_log'], title = 'avg_glu_log Distribution')
fig.update_layout(autosize = False, width = 700, height = 400,
                  xaxis = go.layout.XAxis(linewidth = 1, mirror = True, title = 'avg_glu_log'),
                  yaxis = go.layout.YAxis(linewidth = 1, mirror = True, title = 'Counts'))
fig.show()
```



Hình 20. Biến đổi Log cho biến glucos

Về nghĩa đen, điều này giúp phân phối ban đầu sẽ hẹp dần theo chiều cao và rộng.

```
from scipy.stats import skew
glu_log_s = skew(df['avg_glu_log'], axis=0, bias=True)
glu_log_k = kurtosis(df['avg_glu_log'], axis=0, bias=True)

def skew(val_s):
   if val_s > -0.5 and val_s < 0.5:
      return('Data is quite symmetry')
   elif val_s > -1 and val_s < 1:
      return('Data is moderately deviated')
   else:
      return('Data is highly deviated')

pd.DataFrame([[glu_log_s, skew(glu_log_s), glu_log_k, kurt(glu_log_k)]],
                  index = ['Avg glucose log'],
                  columns = ['Skewness', 'Meaning of skewness', 'Kurtosis', 'Meaning of kurtosis'])
```

	Skewness	Meaning of skewness	Kurtosis	Meaning of kurtosis
Avg glucose log	0.592367	Data is moderately deviated	-0.848473	Platykurtic - Data with a low probability of o...

Bảng 4. Đánh giá độ lệch và độ nhọn cho glu_log

Về nghĩa bóng, biến đổi log đã giúp chúng ta làm giảm độ lệch (skewness) cho cột avg_glucose_level và biến nó trở thành phân phối lệch nhẹ sang phải – không gây ảnh hưởng tiêu cực (thiên vị) đến bài toán phân lớp; kurtosis -0.85 cho thấy avg_glu_log ít gặp các giá trị nhiễu.

3.2.4.3 Kiểm định độc lập

Trước tiên, ta sẽ ép kiểu của stroke từ object về int để phù hợp với kiểm định T-Test. Tạo một dataframe mới chứa 3 cột biến định lượng và 1 biến phụ thuộc; và chuẩn hóa dataframe trên để vẽ biểu đồ trực quan về phân phối của từng biến x và y có trùng nhau không. Nếu có, thì 2 mẫu này phụ thuộc vào nhau; ngược lại thì chứng độc lập với nhau.

```
df_qualitative_tar = df[['age', 'avg_glu_log', 'bmi', 'stroke']]
df_qualitative_tar = StandardScaler().fit_transform(df_qualitative_tar)
df_qualitative_tar = pd.DataFrame(df_qualitative_tar)

df_qualitative_tar.rename(columns = {0:'age', 1:'avg_glu_log', 2:'bmi', 3:'stroke'}, inplace = True)
df_qualitative_tar.head()
```

	age	avg_glu_log	bmi	stroke
0	-2.332673	-0.295530	-1.666175	-1.0
1	0.138850	-0.480691	1.427882	-1.0
2	-2.107989	0.067849	-1.724554	-1.0
3	0.678092	-1.052650	0.946260	-1.0
4	-1.838368	0.957185	-1.505635	-1.0

Hình 21. df_qualitative_tar

Sử dụng stats.ttest_ind của thư viện spicy để tính ra t-value và p-value. Ta chọn p-value và so sánh với alpha để xác định biến độc lập gây ảnh hưởng lên biến y.

```
def ttest(var, tar):
  t, p = stats.ttest_ind(df[var], df[tar])
  print('   t-value = %f'%float("{:.6f}".format(t)))
  print('   p-value for two tailed test = %f'%p)
  alpha = 0.05
  if p <= alpha:
    print(f'''\033[92mBecause p-value = {p} <= alpha({alpha})
⇒ Reject H0: these 2 means of {var} & {tar} are not different, they are interdependent. \033[0;0m''')
  else:
    print(f'''\033[91mBecause p-value = {p} > alpha({alpha})
⇒ Not reject H0: these 2 means of {var} & {tar} are different. \033[0;0m''')
  # Plot these 2 means
  fig = ff.create_distplot(hist_data = [df_qualitative_tar[var], df_qualitative_tar[tar]],
                           group_labels = [var, tar],
                           bin_size = .15,
                           curve_type = 'normal', # override default 'kde'
                           colors = ['#2BCDC1', '#F66095'])
  fig.update_layout(title_text = f'Distplot with Normal Distribution of {var} vs {tar}')
  fig.show()
```

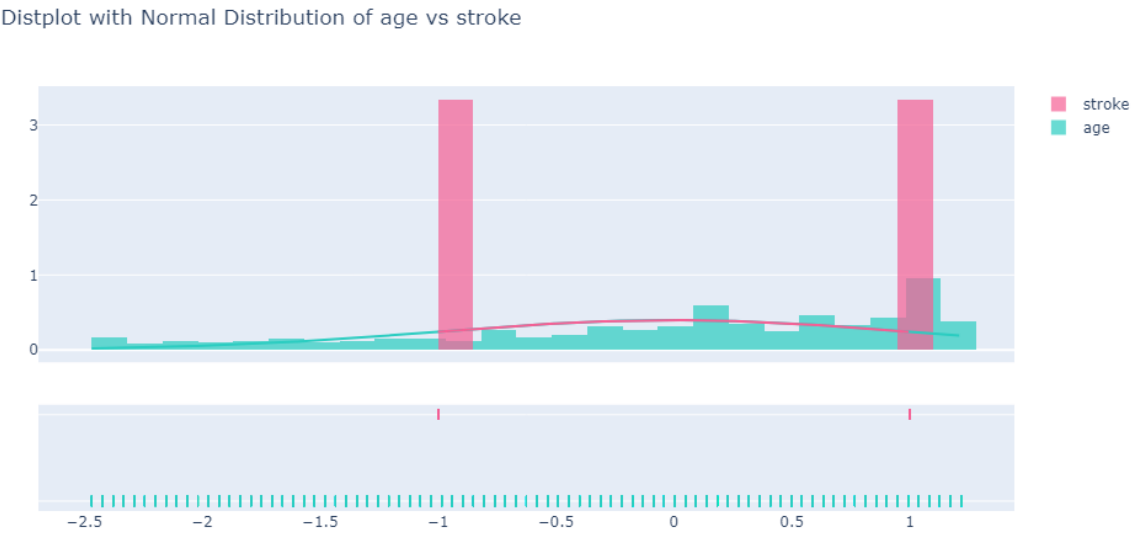
1. With x is age

t-value = 241.005734

p-value for two tailed test = 0.000000

Because p-value = 0.0 <= alpha(0.05)

⇒ Reject H0: these 2 means of age & stroke are not different, they are interdependent.



Hình 22. Kiểm định độc lập giữa age và stroke

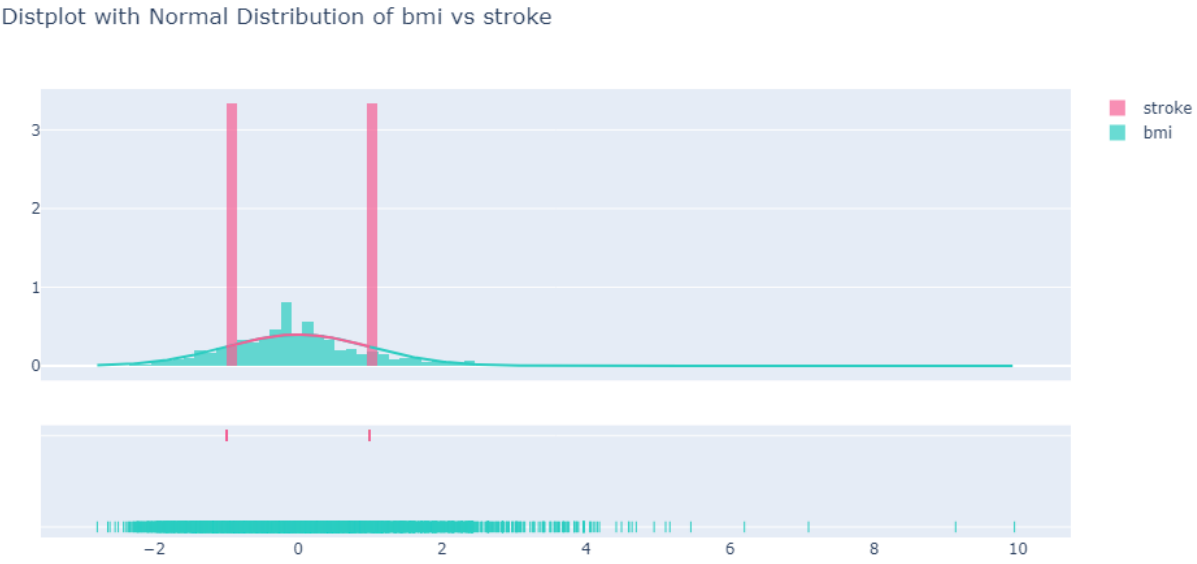
2. With x is avg_glu_log

t-value = 633.719511

p-value for two tailed test = 0.000000

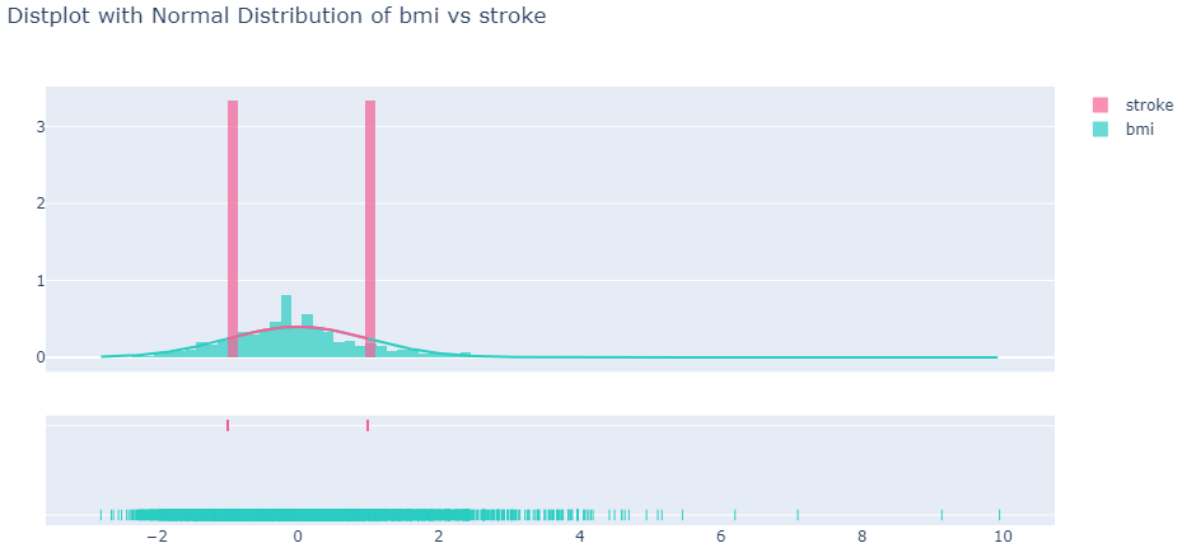
Because p-value = 0.0 <= alpha(0.05)

⇒ Reject H0: these 2 means of avg_glu_log & stroke are not different, they are interdependent.



Hình 23. Kiểm định độc lập giữa glu_log và stroke

```
3. With x is bmi
   t-value = 414.990885
   p-value for two tailed test = 0.000000
Because p-value = 0.0 <= alpha(0.05)
⇒ Reject H0: these 2 means of bmi & stroke are not different, they are interdependent.
```



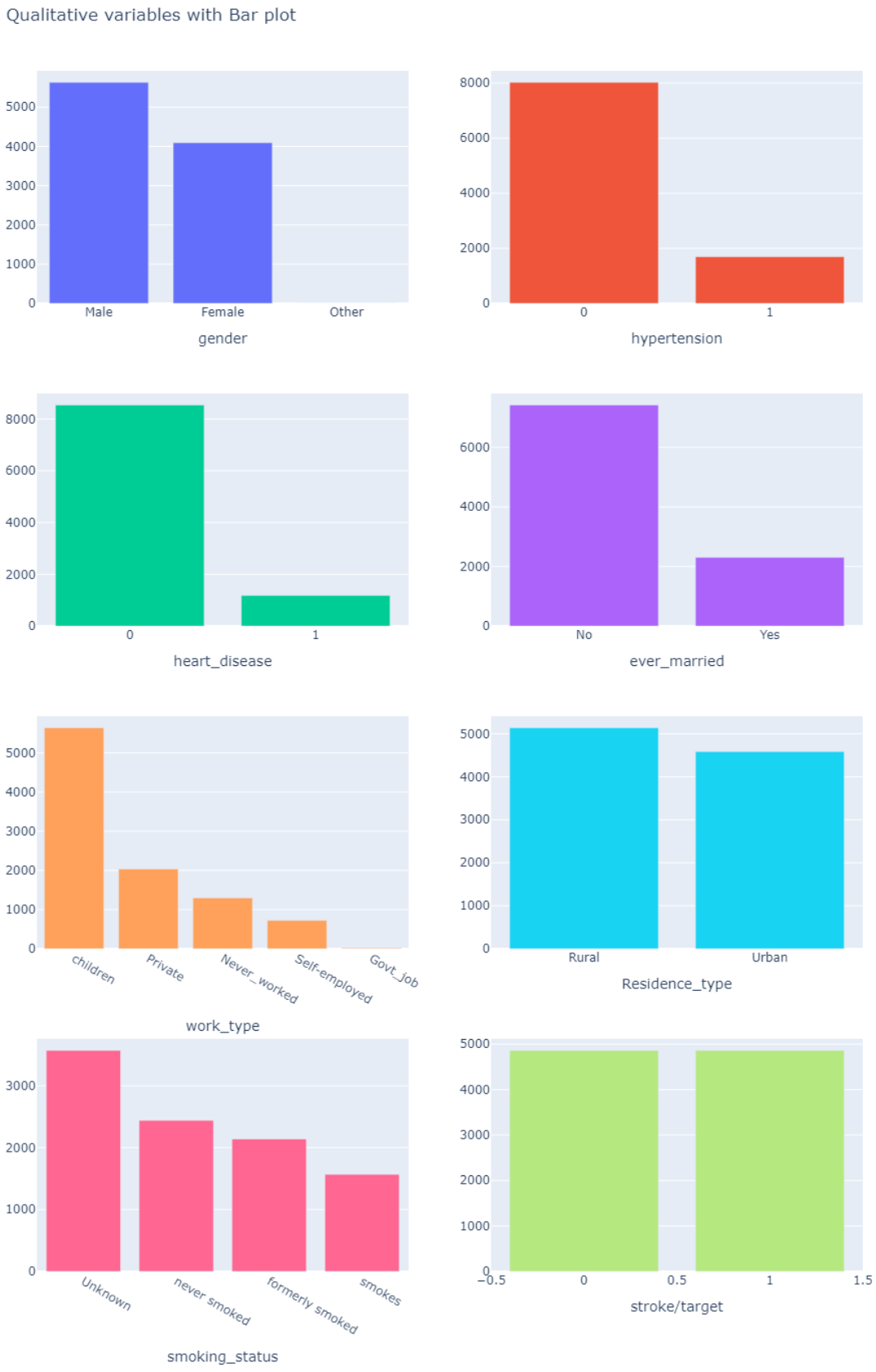
Hình 24. Kiểm định độc lập giữa bmi và stroke

Từ kiểm định T-Test, cả 3 kết quả đều bác bỏ H_0, nên ta tạm thời kết luận rằng cả 3 biến định lượng trên đều gây ra ảnh hưởng đến biến stroke.

3.2.5 Chọn lọc biến định danh

Trực quan các giá trị của các biến định danh bằng biểu đồ cột (bar chart).

```
fig = make_subplots(rows = 4, cols = 2)
fig.add_trace(go.Bar(x = df['gender'].unique(), y = df['gender'].value_counts()), row = 1, col = 1)
fig.add_trace(go.Bar(x = df['hypertension'].unique(), y = df['hypertension'].value_counts()), row = 1, col = 2)
fig.add_trace(go.Bar(x = df['heart_disease'].unique(), y = df['heart_disease'].value_counts()), row = 2, col = 1)
fig.add_trace(go.Bar(x = df['ever_married'].unique(), y = df['ever_married'].value_counts()), row = 2, col = 2)
fig.add_trace(go.Bar(x = df['work_type'].unique(), y = df['work_type'].value_counts()), row = 3, col = 1)
fig.add_trace(go.Bar(x = df['Residence_type'].unique(), y = df['Residence_type'].value_counts()), row = 3, col = 2)
fig.add_trace(go.Bar(x = df['smoking_status'].unique(), y = df['smoking_status'].value_counts()),row = 4, col = 1)
fig.add_trace(go.Bar(x = df['stroke'].unique(), y = df['stroke'].value_counts()), row = 4, col = 2)
fig.update_xaxes(title_text = 'gender', row = 1, col = 1)
fig.update_xaxes(title_text = 'hypertension', row = 1, col = 2)
fig.update_xaxes(title_text = 'heart_disease', row = 2, col = 1)
fig.update_xaxes(title_text = 'ever_married', row = 2, col = 2)
fig.update_xaxes(title_text = 'work_type', row = 3, col = 1)
fig.update_xaxes(title_text = 'Residence_type', row = 3, col = 2)
fig.update_xaxes(title_text = 'smoking_status', row = 4, col = 1)
fig.update_xaxes(title_text = 'stroke/target',  row = 4, col = 2)
fig.update_layout(showlegend = False, title_text = 'Qualitative variables with Bar plot', height = 1400)
fig.show()
```



Hình 25. Các biến định danh

Bởi vì chúng ta đã thực hiện cân bằng dữ liệu ở mục 3.2.2 nên khi biểu diễn

stroke (biến y), số lượng của 2 chân trị 0 và 1 bằng nhau. Các đồ thị còn lại chính là biểu diễn cho dữ liệu định danh sau khi đã xử lí cân bằng. Khác với biến định tính (sử dùng kiểm định T-Test), để xác định từng biến định danh có tác động đến biến y không, ta sẽ dùng kiểm định Chi-Square (hay còn gọi là Chi bình phương).

```
# Independent testing - categorize vs target (Chi-square)
print('''\033[1mIndependent Accreditation among Qualitative variables vs Target \033[0;0m
   H0: μ_(x) = μ_(y) ; with x is each qualitative variable in dataset
   Ha: μ_(x) ≠ μ_(y)''')

def chisqrt(var, tar):
   chi2 = pd.crosstab(df[var], df[tar], margins = True)
   stat, p, dof, expected = chi2_contingency(chi2)
   print('   * Observed values:')
   print(chi2)
   print('\n   * Expected values:')
   print(expected)
   print('\n   * Chi-Square tests:')
   print(pd.DataFrame([[stat, p, dof]], columns = ['Stat-value', 'p-value', 'Df'], index = ['Result']))

   alpha = 0.05
   if p <= alpha:
      print(f'''\033[92mBecause p-value = {str(round(p, 4))} <= alpha({alpha})
⇒ Reject H0: these 2 means of {var} and {tar} are not different, they are interdependent. \033[0;0m''')
   else:
      print(f'''\033[91mBecause p-value = {str(round(p, 4))} > alpha({alpha})
⇒ Not reject H0: these 2 means of {var} and {tar} are different. \033[0;0m''')
```

Kết quả:

```
4. With x is gender
   * Observed values:
stroke     0     1   All
gender
Female  2853  2775  5628
Male    2007  2086  4093
Other      1     0     1
All     4861  4861  9722

   * Expected values:
[[2.8140e+03 2.8140e+03 5.6280e+03]
 [2.0465e+03 2.0465e+03 4.0930e+03]
 [5.0000e-01 5.0000e-01 1.0000e+00]
 [4.8610e+03 4.8610e+03 9.7220e+03]]

   * Chi-Square tests:
        Stat-value   p-value  Df
Result    3.605822  0.729841   6
Because p-value = 0.7298 > alpha(0.05)
⇒ Not reject H0: these 2 means of gender and stroke are different.
```

Vì p-value = 0.73 > alpha (0.05), nên ta sẽ bác bỏ giả thuyết H_0, tức là ta bác bỏ 2 biến này có phân phối trùng nhau, phụ thuộc lẫn nhau.

5. With x is hypertension

```
   * Observed values:
stroke           0     1   All
hypertension
0             4429  3593  8022
1              432  1268  1700
All           4861  4861  9722

   * Expected values:
[[4011. 4011. 8022.]
 [ 850.  850. 1700.]
 [4861. 4861. 9722.]]

   * Chi-Square tests:
        Stat-value        p-value  Df
Result  498.237707  1.611398e-106   4
Because p-value = 0.0 <= alpha(0.05)
⇒ Reject H0: these 2 means of hypertension and stroke are not different, they are interdependent.
```

6. With x is heart_disease

```
   * Observed values:
stroke            0     1   All
heart_disease
0              4632  3903  8535
1               229   958  1187
All            4861  4861  9722

   * Expected values:
[[4267.5 4267.5 8535. ]
 [ 593.5  593.5 1187. ]
 [4861.  4861.  9722. ]]

   * Chi-Square tests:
        Stat-value        p-value  Df
Result  509.983857  4.641292e-109   4
Because p-value = 0.0 <= alpha(0.05)
⇒ Reject H0: these 2 means of heart_disease and stroke are not different, they are interdependent.
```

7. With x is ever_married

```
   * Observed values:
stroke           0     1   All
ever_married
No            1728   579  2307
Yes           3133  4282  7415
All           4861  4861  9722

   * Expected values:
[[1153.5 1153.5 2307. ]
 [3707.5 3707.5 7415. ]
 [4861.  4861.  9722. ]]

   * Chi-Square tests:
        Stat-value        p-value  Df
Result  750.303417  4.457036e-161   4
Because p-value = 0.0 <= alpha(0.05)
⇒ Reject H0: these 2 means of ever_married and stroke are not different, they are interdependent.
```

8. With x is work_type

```
   * Observed values:
stroke            0     1   All
work_type
Govt_job        624   673  1297
Never_worked     22     0    22
Private        2776  2866  5642
Self-employed   754  1283  2037
children        685    39   724
All            4861  4861  9722

   * Expected values:
[[ 648.5  648.5 1297. ]
 [  11.    11.    22. ]
 [2821.  2821.  5642. ]
 [1018.5 1018.5 2037. ]
 [ 362.   362.   724. ]
 [4861.  4861.  9722. ]]

   * Chi-Square tests:
        Stat-value        p-value  Df
Result   739.06916  2.560318e-152  10
```

Because p-value = 0.0 <= alpha(0.05)
⇒ Reject H0: these 2 means of work_type and stroke are not different, they are interdependent.

9. With x is Residence_type

```
   * Observed values:
stroke             0     1   All
Residence_type
Rural           2400  2184  4584
Urban           2461  2677  5138
All             4861  4861  9722

   * Expected values:
[[2292. 2292. 4584.]
 [2569. 2569. 5138.]
 [4861. 4861. 9722.]]

   * Chi-Square tests:
        Stat-value   p-value  Df
Result   19.258587  0.000699   4
```

Because p-value = 0.0007 <= alpha(0.05)
⇒ Reject H0: these 2 means of Residence_type and stroke are not different, they are interdependent.

10. With x is smoking_status

```
   * Observed values:
stroke              0     1   All
smoking_status
Unknown          1497   944  2441
formerly smoked   815  1325  2140
never smoked     1802  1770  3572
smokes            747   822  1569
All              4861  4861  9722

   * Expected values:
[[1220.5 1220.5 2441. ]
 [1070.  1070.  2140. ]
 [1786.  1786.  3572. ]
 [ 784.5  784.5 1569. ]
 [4861.  4861.  9722. ]]

   * Chi-Square tests:
        Stat-value       p-value  Df
Result  250.694029  1.227747e-49   8
```

Because p-value = 0.0 <= alpha(0.05)
⇒ Reject H0: these 2 means of smoking_status and stroke are not different, they are interdependent.

Các biến định danh còn lại gồm hypertension, heart_disease, ever_married, work_type, Residence_type, smoking_status thì ta không thể bác bỏ H_0, tức ta cho rằng chúng có phân phối tương đối trùng và tác động lên biến stroke ở mức tin cậy 95%. Mặc dù, khi xem xét độc lập các biến x với biến y thì ta có thể bác bỏ biến gender. Nhưng khi biến gender đứng chung với các biến định lượng và định tính còn lại, thì gender có gây ảnh hưởng lên chúng không. Phần tiếp theo, ta sẽ xác định điều này.

3.2.6 Kiểm định đồng thời

Để xử lí mối quan hệ giữa gender với các biến độc lập còn lại; các biến độc lập với biến phụ thuộc, ta giả định chúng có mối quan hệ tuyến tính và sử dụng phương pháp OLS trên môi trường Python bằng thư viện statsmodels.api.

3.2.6.1 Ý nghĩa thống kê của biến gender đối với các biến độc lập khác

Cần xác định các biến định danh (có dtypes là object) và ép kiểu chúng về dạng định lượng (có dtypes là int). Để quá trình kiểm định trên không làm ảnh hưởng đến bộ dữ liệu gốc, ta sẽ tạo bản sao cho df thành df1 và bỏ cột stroke.

```
s = (df.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables in the dataset: \n", object_cols)

# Convert object type to numeric type
LE = LabelEncoder()
for i in object_cols:
   df[i] = df[[i]].apply(LE.fit_transform)
```

Kết quả:

```
Categorical variables in the dataset:
 ['gender', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']
```

	Gender	Age	Hypertension	Married	Work	Resident	Glu_log	Bmi	Smoking
Min	0	0	0	0	0	0	4.02	10.30	0
Max	2	82	1	1	4	1	5.60	97.60	3

Bảng 5. Giá trị biến thiên của biến độc lập

Vì biến định lượng có sự giao động giá trị tương đối lớn so với các biến định

danh nên ta sẽ chuẩn hóa (StandarScaler) các giá trị này về một khoảng giao động giống nhau để dễ dàng so sánh hơn.

```
# Scale dataset
df1 = StandardScaler().fit_transform(df1)
df1 = pd.DataFrame(df1)

df1.rename(columns = {0:'gender', 1:'age', 2:'hypertension', 3:'heart_disease', 4:'ever_married',
                      5:'work_type', 6:'Residence_type', 7:'avg_glu_log', 8:'bmi', 9:'smoking_status'}, inplace = True)
df1.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glu_log	bmi	smoking_status
0	1.171732	-2.332673	-0.460345	-0.372927	-1.792800	1.877747	-1.058704	-0.295530	-1.666175	-1.390442
1	1.171732	0.138850	2.172285	-0.372927	0.557787	-0.087847	0.944551	-0.480691	1.427882	0.541925
2	-0.852718	-2.107989	-0.460345	-0.372927	-1.792800	-0.087847	0.944551	0.067849	-1.724554	-1.390442
3	-0.852718	0.678092	-0.460345	-0.372927	0.557787	-0.087847	-1.058704	-1.052650	0.946260	-0.424258
4	1.171732	-1.838368	-0.460345	-0.372927	-1.792800	-1.070644	-1.058704	0.957185	-1.505635	-1.390442

Xem xét ý nghĩa thống kê của gender với các biến độc lập còn lại.

```
print('\033[1mgender ~ age + hypertension + heart_disease + ever_married + work_type + Residence_type +
avg_glu_log + bmi + smoking_status \033[0;0m')

model1 = sm.OLS.from_formula('gender ~ age + hypertension + heart_disease + ever_married + work_type +
Residence_type + avg_glu_log + bmi + smoking_status', data=df1).fit()
print(model1.summary())
```

```
gender ~ age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status
                            OLS Regression Results
==============================================================================
Dep. Variable:                 gender   R-squared:                       0.032
Model:                            OLS   Adj. R-squared:                  0.031
Method:                 Least Squares   F-statistic:                     36.11
Date:                Thu, 05 Oct 2023   Prob (F-statistic):           1.64e-63
Time:                        11:04:52   Log-Likelihood:                -13635.
No. Observations:                9722   AIC:                          2.729e+04
Df Residuals:                    9712   BIC:                          2.736e+04
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   1.826e-16      0.010   1.83e-14      1.000      -0.020       0.020
age            -0.0633      0.013     -4.797      0.000      -0.089      -0.037
hypertension   -0.0267      0.011     -2.535      0.011      -0.047      -0.006
heart_disease   0.1159      0.011     10.908      0.000       0.095       0.137
ever_married    0.0686      0.013      5.441      0.000       0.044       0.093
work_type       0.0433      0.011      4.091      0.000       0.023       0.064
Residence_type -0.0145      0.010     -1.450      0.147      -0.034       0.005
avg_glu_log     0.0973      0.011      9.058      0.000       0.076       0.118
bmi            -0.0049      0.011     -0.457      0.648      -0.026       0.016
smoking_status -0.0593      0.010     -5.687      0.000      -0.080      -0.039
==============================================================================
Omnibus:                    39781.284   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1420.539
Skew:                           0.315   Prob(JB):                    3.42e-309
Kurtosis:                       1.237   Cond. No.                         2.51
==============================================================================
```

Hình 26. Ý nghĩa thống kê của gender với các biến độc lập

Chỉ có 2 biến Resident_type và bmi là không có ý nghĩa thống kê đối với biến gender vì p-value lần lượt của chúng là 0.147 và 0.648 lớn hơn mức ý nghĩa alpha (0.05). Số

lượng biến không bị ảnh hưởng bởi gender quá ít nên ta hoàn toàn có thể giữa lại gender.

3.2.6.2 Ý nghĩa thống kê của các biến độc lập đối với biến phụ thuộc storke

Ta cũng sẽ làm tương tự phương pháp luận ở trên: tạo thêm df2 và chuẩn hóa dữ liệu bằng StandarScaler() và xem xét tính phụ thuộc giữa các biến x và biến y.

```
print('\033[1mstroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status \033[0;0m')

model2 = sm.OLS.from_formula('stroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status', data=df2).fit()
print(model2.summary())
```

stroke ~ gender + age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status

OLS Regression Results

Dep. Variable:	stroke	R-squared:	0.364
Model:	OLS	Adj. R-squared:	0.363
Method:	Least Squares	F-statistic:	555.3
Date:	Thu, 05 Oct 2023	Prob (F-statistic):	0.00
Time:	11:16:54	Log-Likelihood:	-4857.9
No. Observations:	9722	AIC:	9738.
Df Residuals:	9711	BIC:	9817.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.6660	0.050	-13.316	0.000	-0.764	-0.568
gender	-0.0002	0.008	-0.023	0.982	-0.017	0.016
age	0.0130	0.000	54.081	0.000	0.013	0.013
hypertension	0.0893	0.011	7.931	0.000	0.067	0.111
heart_disease	0.0928	0.013	7.006	0.000	0.067	0.119
ever_married	-0.0645	0.012	-5.364	0.000	-0.088	-0.041
work_type	0.0240	0.004	5.668	0.000	0.016	0.032
Residence_type	0.0238	0.008	2.917	0.004	0.008	0.040
avg_glu_log	0.1040	0.010	9.936	0.000	0.083	0.124
bmi	-0.0019	0.001	-3.001	0.003	-0.003	-0.001
smoking_status	-0.0146	0.004	-3.557	0.000	-0.023	-0.007

Omnibus:	303.403	Durbin-Watson:	0.726
Prob(Omnibus):	0.000	Jarque-Bera (JB):	240.590
Skew:	-0.304	Prob(JB):	5.71e-53
Kurtosis:	2.526	Cond. No.	829.

Hình 27. Ý nghĩa thống kê đối với biếntroke

Ta thấy p-value của biến gender = 0.982 lớn hơn mức tin cậy alpha (0.05), do đó ta sẽ loại bỏ biến này ra khỏi mô hình và tiếp tục thực hiện kiểm định này đối với các biến còn lại.

```
print('\033[1mstroke ~ age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status \033[0;0m')

model2_ = sm.OLS.from_formula('stroke ~ age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status', data=df2).fit()
print(model2_.summary())
```

```
stroke ~ age + hypertension + heart_disease + ever_married + work_type + Residence_type + avg_glu_log + bmi + smoking_status
                            OLS Regression Results
==============================================================================
Dep. Variable:                 stroke   R-squared:                       0.364
Model:                            OLS   Adj. R-squared:                  0.363
Method:                 Least Squares   F-statistic:                     617.0
Date:                Thu, 05 Oct 2023   Prob (F-statistic):               0.00
Time:                        11:22:00   Log-Likelihood:                -4857.9
No. Observations:                9722   AIC:                             9736.
Df Residuals:                    9712   BIC:                             9808.
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.6659      0.050    -13.318      0.000      -0.764      -0.568
age            0.0130      0.000     54.149      0.000       0.013       0.013
hypertension   0.0893      0.011      7.934      0.000       0.067       0.111
heart_disease  0.0927      0.013      7.046      0.000       0.067       0.119
ever_married  -0.0645      0.012     -5.374      0.000      -0.088      -0.041
work_type      0.0239      0.004      5.672      0.000       0.016       0.032
Residence_type 0.0238      0.008      2.918      0.004       0.008       0.040
avg_glu_log    0.1040      0.010      9.976      0.000       0.084       0.124
bmi           -0.0019      0.001     -3.001      0.003      -0.003      -0.001
smoking_status -0.0146     0.004     -3.562      0.000      -0.023      -0.007
==============================================================================
Omnibus:                      303.308   Durbin-Watson:                   0.726
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              240.534
Skew:                          -0.304   Prob(JB):                     5.87e-53
Kurtosis:                       2.527   Cond. No.                         829.
==============================================================================
```

Hình 28. Ý nghĩa thống kê đối với biến stroke (sau khi loại gender)

Đến đây, không còn thấy biến nào có thể loại bỏ nữa, bởi vì p-value của chúng đều nhỏ hơn hoặc bằng alpha (0.05). Ngoài ra, ta có thể so sánh kết quả AIC và BIC của cả 2 mô hình trước và sau khi loại bỏ biến gender:

	AIC	**BIC**
Trước	9738	9817
Sau	9736	9808

Chỉ số AIC và BIC đều giảm sau khi loại bỏ biến. Đây là dấu hiệu tốt cho thấy ta đã loại bỏ thành công biến độc lập không có ý nghĩa với biến stroke nói riêng và mô hình nói chung.

Sau khi thực hiện điểm định độc lập và kiểm định đồng thời đối với các biến độc lập với nhau và các biến độc lập với biến phụ thuộc, ta đã có đủ cơ sở để loại bỏ biến gender ra khỏi mô hình. Hơn thế nữa, ta có thể định lượng cường độ của mối quan hệ tương quan giữa các biến độc lập với biến phụ thuộc như thế nào. Cụ thể, ta sẽ vẽ biểu đồ nhiệt (heatmap).

```
# Correlation
correlation = df.corr()

# Delete the last row and column in dataset
adjusted_mask_dataset_corr = correlation.iloc[1: , :-1]

fig = px.imshow(adjusted_mask_dataset_corr, color_continuous_scale = px.colors.sequential.GnBu, text_auto = Tru
fig.update_layout(autosize = False, width = 900, height = 850)
fig.show()
```

Hình 29. Biểu đồ nhiệt của các biến

```
# Sort
correlation["stroke"].sort_values(ascending=False)
```

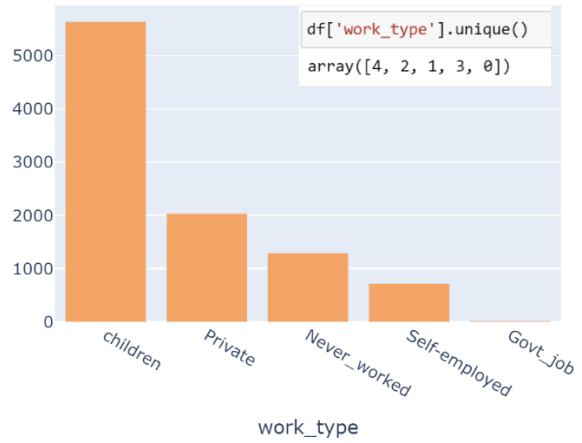
```
stroke            1.000000
age               0.581950
ever_married      0.277805
heart_disease     0.229034
avg_glu_log       0.228476
hypertension      0.226381
bmi               0.088609
smoking_status    0.066685
Residence_type    0.044508
work_type        -0.084815
Name: stroke, dtype: float64
```

Hình 30. Thứ tự tương quan của các biến x với y

Từ thứ tự tương quan giảm dần phía trên, ta kết luận rằng:

- Tuổi tác (biến age) càng cao càng dễ mắc đột quỵ và tuổi tác quy định tới 58.19% khả năng bị đột quỵ (biến age có tỉ lệ tương quan thuận với stroke là 0.5819)
- Xếp sau đó lần lượt là tình trạng hôn nhân, một người đã lập gia đình có khả năng bị đột quỵ là 27.78%
- Bệnh về tim, một người có bệnh về tim mạch có khả năng bị đột quỵ là 22.90%.
- Nồng độ đường, một người có mức đường cao có khả năng bị đột quỵ là 22.84%.
- Cao huyết áp, một người bị cao huyết áp có khả năng bị đột quỵ là 22.63%.
- Các biến bmi, tình trạng hút thuốc, nơi ở (nông thôn, thành thị) ảnh hưởng không đáng kể, rất ít khả năng dẫn đến đột quỵ, lần lượt chiếm 8.86%, 6.66%, 4.45%.

Ngược lại, môi trường làm việc (nhà nước, tư nhân, chưa từng đi làm, …) có mối tương quan nghịch với tình trạng đột quỵ. Môi trường làm việc quy định 8.48% khả năng đột quỵ.



Nó có nghĩa là một người làm việc cho nhà nước hoặc chính phủ có nguy cơ mắc đột quỵ cao nhất, sau đó là người chưa từng đi làm, và tư nhân. Trẻ em và tự kinh doanh là 2 trường hợp ít bị đột quỵ nhất.

3.3 Đề xuất giả thuyết

3.3.1 Hồi quy tuyến tính

Hàm số dễ áp dụng nhất đối với tất cả các mô hình dự báo phân lớp chính là mô hình hồi quy tuyến tính. Dựa vào kết quả hồi quy OLS cuối cùng ở mục 3.2.6.2, ta dễ dàng có được mô hình tuyến tính. Với môi trường Python, ta sử dụng thư viện

sklearn.linear_model, cài đặt LinearRegression() để có kết quả hồi quy tuyến tính.

```
x = df.drop('stroke', axis = 1)
y = df['stroke']

lr = LinearRegression().fit(x, y)
yhat_lr = lr.predict(x)
b0 = round(lr.intercept_,4)
B = np.round(lr.coef_,4)
print(f'''MLR: yhat = {b0} + ({B[0]}){x.columns[0]} + ({B[1]}){x.columns[1]} + ({B[2]}){x.columns[2]} +
      ({B[3]}){x.columns[3]} + ({B[4]}){x.columns[4]} + ({B[5]}){x.columns[5]} + ({B[6]}){x.columns[6]} +
      ({B[7]}){x.columns[7]} + ({B[8]}){x.columns[8]}''')
print('     MSE =', round(mean_squared_error(y, yhat_lr), 4))
print('      R2 =', round(r2_score(y, yhat_lr), 4), f'-> {round(r2_score(y, yhat_lr)*100,2)}% of variation of
      Stroke variable is explained by MLR model.')
```

Kết quả:

```
MLR: yhat = -0.6659 + (0.013)age + (0.0893)hypertension + (0.0927)heart_disease + (-0.0645)ever_married + (0.0239)work_type
                    + (0.0238)Residence_type + (0.104)avg_glu_log + (-0.0019)bmi + (-0.0146)smoking_status
      MSE = 0.1591
       R2 = 0.3638 -> 36.38% of variation of Stroke variable is explained by MLR model.
```

Ta kiểm tra lại chất lượng mô hình bằng:

- MSE = 0.1591 nên sự chênh lệch giữa giá trị dự đoán và thực tế là gần 16%.
- R^2 = 0.3638 nên khoảng 36% giá trị dự báo được giải thích bởi mô hình hồi quy tuyến tính, tức là chỉ có mô hình trên chỉ dự báo đúng 36%.

Kết quả dự báo của mô hình là có dạng số thập phân, nên ta sẽ chuyển đổi chúng một chút: giá trị âm, ta sẽ gán nó dự báo bằng 0 (không bị đột quỵ), ngược lại bằng 1 (bị đột quỵ).

Hình 31. Biểu đồ phân phối giữa giá trị dự đoán và thực tế bằng LR

Vì vậy, ta phải tìm thêm một hàm khác dự báo kết quả tốt hơn.

3.3.2 *Hồi quy phi tuyến tính*

Xây dựng hàm Sigmoid để tính ra xác suất của từng lớp và kiểm tra lại chất lượng của mô hình Logistic bằng biểu đồ.

```
# Find a f(x)
f = np.dot(x, B) + b0

# Find a s(x) - sigmoid function
s = np.round(1/(1 + math.e**(-f)), 4)
s
```

Kết quả phân lớp của từng quan sát:

```
array([0.4773, 0.6336, 0.4917, ..., 0.6171, 0.5998, 0.6877])
```

```
fig = go.Figure()
fig.add_trace(go.Histogram(x = df_temp2['s(x)'], name = 's(x)'))
fig.add_trace(go.Histogram(x = df_temp2['stroke'], name = 'y true'))
fig.update_xaxes(title_text = 'y value')
fig.update_layout(title_text = 'Predict stroke with Logistic Regression', width = 600, height = 450)
fig.show()
```

Hình 32. Biểu đồ phân phối giữa giá trị dự đoán và thực tế bằng s(x)

```
pricise_rate = round(1 - sum(abs(s) - abs(y_true)) / len(df.index), 4) * 100
print(f'Precise rate of Logistic function = {pricise_rate}%')

loss = round(log_loss(y_true, s),4)
print(f'The difference between y_pred and y_true = {loss}')
```

Kết quả:

```
Precise rate of Logistic function = 56.16%
The difference between y_pred and y_true = 0.6438
```

Tỉ lệ chính xác của mô hình này là 56.96% (tốt hơn so với 36.38% của mô hình hồi quy tuyến tính) và chênh lệch giữa giá trị dự đoán và thực tế chỉ chiếm 0.16%.

3.4 Huấn luyện mô hình

Ta chia bộ dữ liệu thành 2 phần: 1 phần để tiến hành huấn luyện (training) cho mô hình dự báo, phần còn lại để kiểm tra (testing) chất lượng mô hình vừa rồi, nó dự đoán đúng bao nhiêu phần trăm, nó xảy ra các loại sai lầm gì. Với bài toán này, ta chọn tỉ lệ 70% train - 30% test.

```
x = StandardScaler().fit_transform(x)
x = pd.DataFrame(x)
x.rename(columns = {0:'age', 1:'hypertension', 2:'heart_disease', 3:'ever_married', 4:'work_type',
            5:'Residence_type', 6:'avg_glu_log', 7:'bmi', 8:'smoking_status'}, inplace = True)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)
pd.DataFrame([[x_train.shape, x_test.shape],
                    [y_train.shape, y_test.shape]],
                    index = ['x','y'], columns = ['train', 'test'])
```

	train	test
x	(6805, 9)	(2917, 9)
y	(6805,)	(2917,)

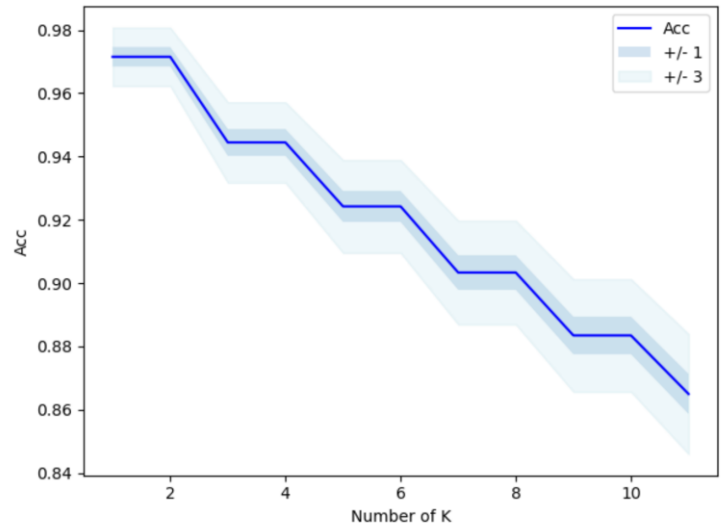
Hình 33. Kết quả chia tập train – test

3.4.1 Xây dựng mô hình

3.4.1.1 K-nearest neighbor (KNN)

Đầu tiên, chúng ta sẽ tìm ra số lượng "hàng xóm" gần nhất với một quan sát cần dự báo.

```
k_s = 12
mean_acc = np.zeros((k_s -1))
std_acc = np.zeros((k_s -1))

for n in range (1, k_s):
   neigh = KNeighborsClassifier(n_neighbors = n).fit(x_train, y_train)
   yhat_knn = neigh.predict(x_test)
   mean_acc[n-1] = metrics.accuracy_score(y_test, yhat_knn)
   std_acc[n-1] = np.std(yhat_knn == y_test) / np.sqrt(yhat_knn.shape[0])

plt.plot(range(1, k_s), mean_acc, 'b')
plt.fill_between(range(1, k_s), mean_acc - 1*std_acc, mean_acc + 1*std_acc, alpha = 0.2)
plt.fill_between(range(1, k_s), mean_acc - 3*std_acc, mean_acc + 3*std_acc, alpha = 0.2, color = 'lightblue')
plt.legend(('Acc', '+/- 1', '+/- 3'))
plt.ylabel('Acc')
plt.xlabel('Number of K')
plt.tight_layout()
plt.show()
```




```
The best Acc is  0.9715461090161125 with K = 1
```

Hình 34. Kết quả chọn K

Với K = 1, mô hình KNN đã có thể phân lớp chính xác tập dữ liệu lên đến 97.15%. Tiếp theo, ta sẽ cho dữ liệu từ 70% train và 30% test lần lượt chạy qua mô hình K-NN.

```
neigh = KNeighborsClassifier(n_neighbors = mean_acc.argmax() + 1).fit(x_train, y_train)
yhat_knn = neigh.predict(x_test)
print('Train set acc: ', metrics.accuracy_score(y_train, neigh.predict(x_train)))
print('Test set acc: ', metrics.accuracy_score(y_test, yhat_knn))
```

Kết quả:
```
Train set acc:  1.0
Test set acc:  0.9715461090161125
```

3.4.1.2 Decision tree

```
dtree = DecisionTreeClassifier(criterion = 'entropy', max_depth = 2).fit(x_train, y_train)
yhat_dt = dtree.predict(x_test)
print('Train set acc: ', metrics.accuracy_score(y_train, dtree.predict(x_train)))
print('Test set acc: ', metrics.accuracy_score(y_test, yhat_dt))
```

Kết quả:
```
Train set acc:  0.7748714180749449
Test set acc:  0.7709976002742543
```

3.4.1.3 Logistics regression

Đây là mô hình phân lớp dựa theo ý nghĩa và cách thức hoạt động của hàm Sigmoid –mô hình hồi quy phi tuyến tính (mục 3.3.2).

```
lgr = LogisticRegression(C = 0.01, solver = 'liblinear').fit(x_train, y_train)
yhat_lgr = lgr.predict(x_test)
print('Train set acc: ', metrics.accuracy_score(y_train, lgr.predict(x_train)))
print('Test set acc: ', metrics.accuracy_score(y_test, yhat_lgr))
```

Kết quả:
```
Train set acc:  0.7814842027920647
Test set acc:  0.7757970517655125
```

3.4.1.4 Bernoulli Naive Bayes

```
nb = BernoulliNB().fit(x_train, y_train)
yhat_nb = nb.predict(x_test)
print('Train set acc: ', metrics.accuracy_score(y_train, nb.predict(x_train)))
print('Test set acc: ', metrics.accuracy_score(y_test, yhat_nb))
```

Kết quả:

```
Train set acc:  0.7545922116091109
Test set acc:  0.762427151182722
```

3.4.2 Đánh giá các mô hình phân lớp

```
neigh_val = cross_val_score(neigh, x_train, y_train, cv = 5, scoring = "accuracy")
dtree_val = cross_val_score(dtree, x_train, y_train, cv = 5, scoring = "accuracy")
lgr_val = cross_val_score(lgr, x_train, y_train, cv = 5, scoring="accuracy")
nb_val = cross_val_score(nb, x_train, y_train, cv = 5, scoring="accuracy")

kfold = pd.DataFrame((neigh_val, dtree_val, lgr_val, nb_val),
                     index = ['K-NN', 'Decision tree', 'Log reg', 'Naive Bayes'],
                     columns = ['Fold = 1', 'Fold = 2', 'Fold = 3', 'Fold = 4', 'Fold = 5'])
mean_kfold = pd.DataFrame((statistics.mean(neigh_val), statistics.mean(dtree_val),
statistics.mean(lgr_val), statistics.mean(nb_val)),
                          index = ['K-NN', 'Decision tree', 'Log reg', 'Naive Bayes'],
                          columns = ['Mean'])
pd.concat([kfold, mean_kfold], axis = 1)
```

	Fold = 1	Fold = 2	Fold = 3	Fold = 4	Fold = 5	Mean
K-NN	0.963997	0.957384	0.963262	0.963997	0.965467	0.962821
Decision tree	0.756796	0.781043	0.754592	0.756062	0.788391	0.767377
Log reg	0.757531	0.792799	0.756062	0.784717	0.789860	0.776194
Naive Bayes	0.734754	0.756062	0.739162	0.767083	0.772961	0.754004

Hình 35. Kết quả 5-fold

Bộ tập dữ liệu được chia thành 5 phần bằng nhau, lấy bất kì 4 phần để huấn luyện cho 4 mô hình, gồm K-NN, Decision tree, Logistic regression, Naive Bayes và 1 phần còn lại để kiểm định mô hình và đưa ra tỉ lệ chính xác accuracy, 4 phần train và 1 phần test cứ thế lặp lại cho đến hết. Cột mean chính là giá trị trung bình của 5-fold hay chính là trung bình của 5 tỉ lệ accuracy theo từng fold.

	Accuracy of training set	Accuracy of testing set
K-NN	1.00	0.97
Decision tree	0.77	0.77
Log reg	0.78	0.78
Naive Bayes	0.75	0.76

Hình 36. Accuracy của tập train và test

Tổng hợp lại accuracy train và test của 4 mô hình dự báo, ta rút ra kết luận sau: Thứ nhất, độ chính xác (Accuracy) bằng 1 không nhất thiết dẫn đến hiện tượng mô hình dự đoán quá khớp trên tập huấn luyện (overfitting). Tuy nhiên, nếu tập huấn luyện (training set) của một mô hình có accuracy là 1 mà tập kiểm định (testing set) khác 1, đây có thể là dấu hiệu của overfitting. Thứ hai, k = 1 (quá nhỏ so với số lượng các quan sát = 9722) cũng dẫn đến overfitting. Thứ ba, khi so sánh giá trị trung bình (mean) của 5-fold validation với tập huấn luyện (accuracy of training set), chỉ có K-NN cho ra 2 giá trị có sự chênh lệnh cao nhất so với 3 mô hình dự báo còn lại.

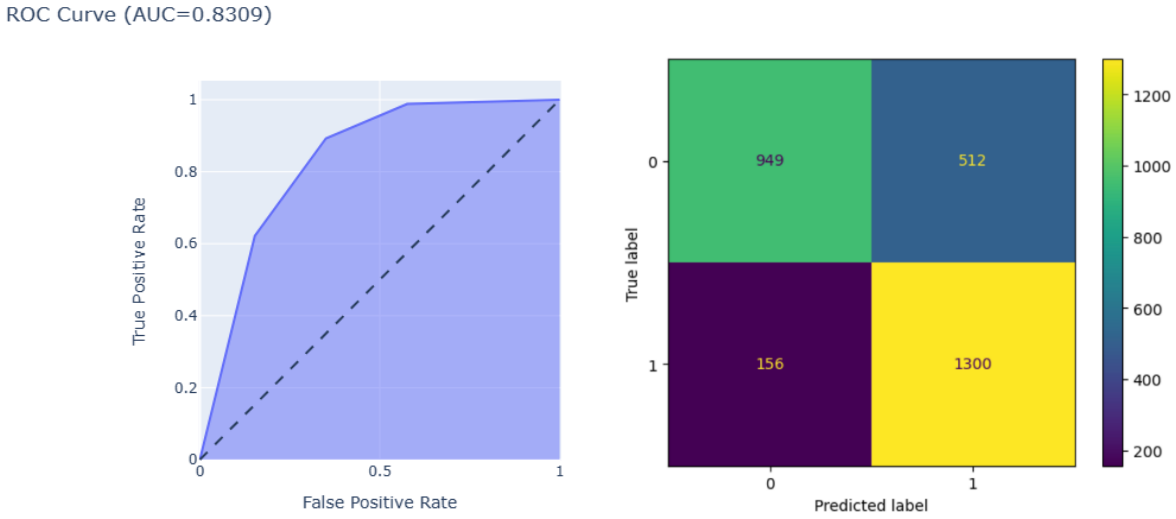
Từ ba dấu hiệu trên, ta có thể khẳng định rằng mô hình K-NN đang gặp tình huống overfitting, như vậy nó sẽ cho ra kết quả dự báo hoàn toàn tệ. Ta loại bỏ mô hình K-NN ra khỏi bài toán này.

```
def Model(model):
   model.fit(x_train,y_train)
   yhat = model.predict(x_test)
   cm = confusion_matrix(y_test,yhat)

   print(classification_report(y_test, yhat))
   yhat_model_proba = model.predict_proba(x_test)
   print('Log loss', log_loss(y_test, yhat_model_proba))

   disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = model.classes_)
   disp.plot()

   y_score = yhat_model_proba[:, 1]
   fpr, tpr, thresholds = metrics.roc_curve(y_test, y_score)
   fig = px.area(x = fpr, y = tpr,
                 title=f'ROC Curve (AUC={metrics.auc(fpr, tpr):.4f})',
                 labels = dict(x = 'False Positive Rate', y = 'True Positive Rate'),
                 width = 700, height = 500)
   fig.add_shape(type = 'line', line = dict(dash = 'dash'), x0 = 0, x1 = 1, y0 = 0, y1 = 1)
   fig.update_yaxes(scaleanchor = "x", scaleratio = 1)
   fig.update_xaxes(constrain = 'domain')
   fig.show()
```

- **Đánh giá mô hình phân lớp Decision tree**

Model(dtree)

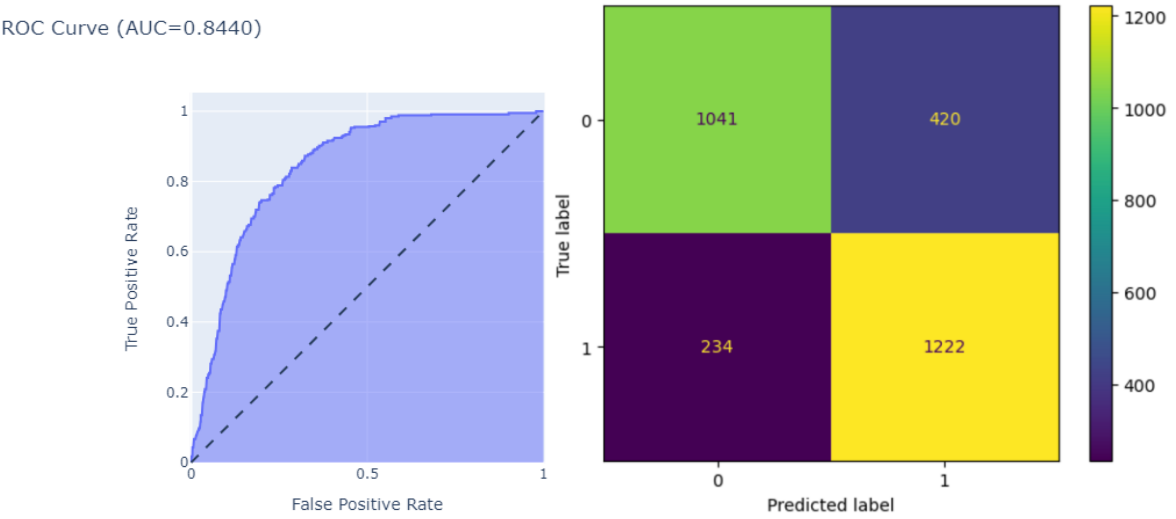
	precision	recall	f1-score	support
0	0.86	0.65	0.74	1461
1	0.72	0.89	0.80	1456
accuracy			0.77	2917
macro avg	0.79	0.77	0.77	2917
weighted avg	0.79	0.77	0.77	2917



- **Đánh giá mô hình phân lớp Logstic regression**

Model(lgr)

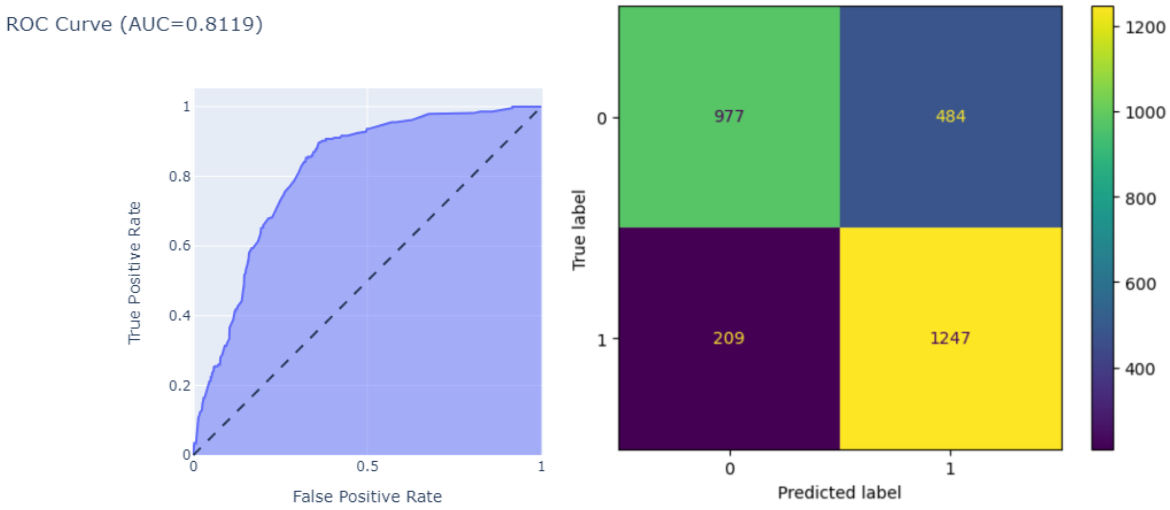
	precision	recall	f1-score	support
0	0.82	0.71	0.76	1461
1	0.74	0.84	0.79	1456
accuracy			0.78	2917
macro avg	0.78	0.78	0.77	2917
weighted avg	0.78	0.78	0.77	2917



- **Đánh giá mô hình phân lớp Bernoulli Naive Bayes**

Model(nb)

```
              precision    recall  f1-score   support

           0       0.82      0.67      0.74      1461
           1       0.72      0.86      0.78      1456

    accuracy                           0.76      2917
   macro avg       0.77      0.76      0.76      2917
weighted avg       0.77      0.76      0.76      2917

Log loss 0.5450762027566832
```



3.4.3 Lựa chọn mô hình

Sau khi thực hiện các bước kiểm định mô hình từ 5-fold-cross-validation, confusion matrix đến classification report, ta có thể tóm tắt như sau:

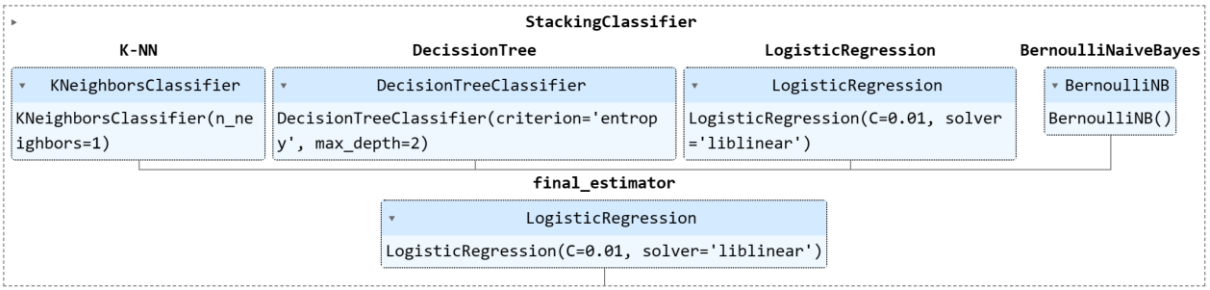
	Mean 5-fold	**F1-score**	**AUC**	**Sai lầm I (FN)**	**Sai lầm II (FP)**
Decision tree	0.7673	0.77	0.8309	512	156
Logistic regression	0.7761	0.78	0.8440	420	234
Naive Bayes	0.7540	0.76	0.8119	484	209

Bảng 6. Tóm tắt đánh giá mô hình

Mô hình học có giám sát Logistic Regression cho ra kết quả vượt trội nhất. Thứ nhất, trung bình của 5-fold là 0.77; thứ hai, F1-score bằng 0.78, tức mô hình này có thể dự đoán chính xác 1 quan sát (1 bệnh nhân) mới đạt 78%; thứ ba, AUC cho biết mô hình này có thể nhận biết và phân biệt được sự khác nhau giữa giữa 2 chân trị 0/1 là hơn 84%. Tuy nhiên mô hình Logistic Regression chưa đưa kết quả sai lầm loại I và II

tốt nhất nhưng dựa vào F1 và AUC – đây là khả năng mà mô hình phân biệt được chân trị 0/1 – thì mô hình hồi quy Logistic hoàn toàn có thể phát triển tốt hơn so với 2 mô hình còn lại trong tương lai.

Biểu diễn mô hình phân lớp được chọn.



Hình 37. Mô hình Logistic được chọn

3.4.4 Áp dụng mô hình dự báo cho bệnh nhân mới

Chúng ta sẽ cho người dùng nhập các thông tin như tuổi (age), tình trạng cao huyết áp (hypertension), tình trạng hôn nhân (married_status), … tương ứng với 9 biến độc lập trong mô hình dự báo.

```
x0 = int(eval(input('Age                                                                                    = ')))
x1 = int(eval(input('Hypertension (0/1)                                                                     = ')))
x2 = int(eval(input('Heart disease (0/1)                                                                    = ')))
x3 = int(eval(input('Marry status (0/1)                                                                     = ')))
x4 = int(eval(input('Work type (Govt_job=0; Never_worked=1; Private=2; Self-employed=3; children=4)  = ')))
x5 = int(eval(input('Residence type (Rural=0; Urban=1)                                                      = ')))
x6 = float(eval(input('Glucose level(mg/dl) (18*(mmol/l) = mg/dl)                                           = ')))
x6 = np.log(x6)
x7_1 = float(eval(input('Weight(kg)                                                                         = ')))
x7_2 = float(eval(input('Height(m)                                                                          = ')))
x7 = x7_1 / (x7_2)**2
x8 = int(eval(input('Smoking status (Unknow=0; formerly smoked=1; never smoked=2; smokes=3)         = ')))
```

```
Age                                                                    = 28
Hypertension (0/1)                                                     = 0
Heart disease (0/1)                                                    = 0
Marry status (0/1)                                                     = 0
Work type (Govt_job=0; Never_worked=1; Private=2; Self-employed=3; children=4) = 2
Residence type (Rural=0; Urban=1)                                      = 1
Glucose level(mg/dl) (18*(mmol/l) = mg/dl)                             = 18*4.51
Weight(kg)                                                             = 44
Height(m)                                                              = 1.48
Smoking status (Unknow=0; formerly smoked=1; never smoked=2; smokes=3)         = 2
```

Hình 38. Mẫu của người dùng mới

Thực hiện kết quả phân lớp bằng mô hình hồi quy Logistic.

```
print('Prediction for a new patient is class', str(lgr.predict(patient)))
print('Probability (%) of a new patient in each class:')
pd.DataFrame(np.round(lgr.predict_proba(patient),4)*100, columns = ['Class 0', 'Class 1'])
```

Kết quả:

```
Prediction for a new patient is class [0]
Probability (%) of a new patient in each class:
```

	Class 0	Class 1
0	87.96	12.04

Kết quả phân lớp của bệnh nhân mới là 0, tức người này không bị đột quỵ với xác suất rơi vào lớp 0 là 87.96% và lớp 1 là 12.04%.

Kết luận và hướng phát triển của đề tài

Kết luận

Đứng trước nhu cầu về sức khỏe đời sống của người dân ngày càng cao, các bệnh viện và bác sĩ chuyên môn đang tích cực hỗ trợ người dân trong khám và điều trị các ca bệnh tình từ mức sơ cấp đến nguy kịch nhiều nhất có thể. Với đề tài **Xây Dựng Mô Hình Kết Luận Tình Trạng Đột Quỵ**, đây được xem là giải pháp giúp các bệnh viện và bác sĩ giảm nhẹ gánh nặng từ số lượng người dân đến khám, giảm thiểu tối đa các trường hợp chuẩn đoán sai từ bác sĩ. Mặt khác, giúp đỡ người dân nâng cao nhận thức về bệnh đột quỵ và có phương pháp ngăn chặn căn bệnh hiệu quả

Dựa vào những kiến thức đã học và quá trình nghiên cứu, mục tiêu mà em đạt được trong bài báo cáo này là:

- Tiếp thu thêm những kiến thức bên ngoài nhà trường về mặt kĩ thuật phân tích mẫu dữ liệu thô cho đến bộ dữ liệu hoàn chỉnh, có chất lượng tốt để sử dụng trong suốt quá trình xây dựng mô hình học máy dự báo tình trạng bị đột quỵ.
- Tổng kết, thống kê lại những kiến thức trong và ngoài trường học để tạo ra phương pháp luận, cách thức triển khai từng bước, giai đoạn trong phân tích dữ liệu.
- Xây dựng ra mô hình học máy có khả năng dự đoán tốt và có triển vọng đóng góp thực tế vào tương lai.

Hướng phát triển của đề tài

Trong tương lai, mô hình học máy hồi quy Logistic và bài toán phân lớp khả năng bị đột quỵ cho bệnh nhân nói riêng và người sử dụng khác nói chung hoàn toàn có thể đem ra ngoài thực tế. Cụ thể, nhà lập trình có thể xây dựng thuật toán trên thành một thiết bị hoặc phần mềm dự báo dùng trong bệnh viện, hoặc xây dựng chúng thành một ứng dụng trên điện thoại. Qua đó, một người dùng sẽ được tự đánh giá, kiểm tra sức khỏe của mình một cách nhanh chóng và thuận tiện. Nếu người dùng thấy có dấu hiệu bị đột quỵ, họ sẽ tự đến các bệnh viện uy tín để kiểm tra lại bằng máy móc, thiết bị chuyên dụng và kinh nghiệm thực chiến của đội ngũ bác sĩ.

- Xây dựng thuật toán trên cơ sở máy móc bệnh viện

Đối với bệnh viện và bác sĩ, việc sử dụng thêm máy móc, phần mềm hỗ trợ dự báo khả năng bị đột quỵ trong quá trình khám/chữa bệnh sẽ giúp gia tăng sự tin cậy hơn vào kết quả chuẩn đoán của bác sĩ. Từ đó, sẽ làm giảm đáng kể sai sót xảy ra sai lầm loại II (bệnh nhân có thể bị đột quỵ, nhưng chuẩn đoán thì nói không). Ngoài ra bệnh viện cũng xây dựng được hình ảnh uy tín của mình về trang thiết bị tiên tiến, bắt kịp nhu cầu khám/chữa bệnh của người dân.

Đối với bệnh nhân đi khám, họ sẽ gia tăng sự tin tưởng, an tâm vào bệnh viện hơn. Hơn thế nữa, nếu kết quả dự báo đột quỵ là có, bệnh nhân sẽ được kịp thời chữa trị sớm để tránh xảy ra trường hợp đáng tiếc về sau.

- Xây dựng thuật toán trên cơ sở ứng dụng trên điện thoại

Đối với người dân, họ không cần phải đến trực tiếp bệnh viện, dù là đang sinh sống ở thành thị, nông thôn hay vùng ven; không cần phải tốn thời gian, chi phí đi lại, họ hoàn toàn có thể tự chăm sóc cho sức khỏe của bản thân mình thông qua ứng dụng này. Ngoài ra, họ có thể kiểm tra cho chính mình nhiều lần, bất cứ lúc nào, bất cứ nơi đâu; sâu xa hơn, họ có thể giúp đỡ và chia sẻ ứng dụng trên với nhiều người khác nữa. Lúc này sẽ có 2 tình huống xảy ra:

1. Một là kết quả dự đoán đột quỵ là "không", người dân có thể vơi bớt sự lo lắng và tạm thời an tâm về chất lượng cuộc sống của bản thân ở thời điểm hiện tại hoặc họ có thể đến bệnh viện để được kiểm tra lại.
2. Hai là kết quả dự đoán nói "có", người dân sẽ đến các bệnh viện đáng tin cậy để được chuẩn đoán lại và đưa ra kết luận chính xác nhất từ bác sĩ chuyên môn.

Tài liệu tham khảo

Tran Van Nhat (2017). Sơ lược về module python

Dictionary4it. Định nghĩa Tập dữ liệu mất cân bằng

Tomorrow Marketers (2023). Dữ liệu ngoại lai (outlier) là gì và xử lý những dữ liệu này như nào trong phân tích?

Cuong Sai (2020). Sử dụng thống kê để xác định và loại bỏ dữ liệu ngoại lai cho machine learning trong R và Python

Nhóm MBA Bách Khoa (2021). Chỉ số skewness và kurtosis là gì và dùng để làm gì?

CEDSEDU (2021). Đô Lệch là gì

Lê Thảo (2019). Phân phối Leptokurtic

Cao Minh Ngoc (2019). Feature Engineering

Nghiên cứu giáo dục (2023). T-test độc lập

Data Science Basic (2020). Kiểm định giả thuyết và ý nghĩa thống kê

Huỳnh Công Thịnh (2022). Kiểm Định Chi Square

Daniel Nelson (2020). KNN (K-Hàng xóm gần nhất) là gì?

Meey Land (2022). Decision Tree là gì?

1UP Note (2018). Naive Bayes Classification (NBC) là gì?

Trí tuệ nhân tạo (2020). Giới thiệu về k-fold cross-validation

Trung Đức (2021). Đánh giá các mô hình học máy

Codecademy (2023). K-Nearest Neighbors Underfitting and Overfitting

Phụ lục

Các thư viện được sử dụng trong bài toán:

Cài đặt module sys – module quan trọng được cài sẵn vào trình thông dịch Python, nó giúp cho mã code chạy được trên mọi phiên bản của Python mà không xuất hiện cảnh báo.

```
# Ignore warnings
import sys
if not sys.warnoptions:
   import warnings
   warnings.simplefilter("ignore")
```

Cài đặt các thư viện cho bài toán phân lớp, gồm:

```
import pandas                    as pd                    # Import library
import numpy                     as np
import matplotlib.pyplot         as plt                   # Library for data visualization
import plotly.graph_objects      as go
import plotly.express            as px
from plotly.subplots import make_subplots
import plotly.figure_factory     as ff
from sklearn.utils import resample                        # Library for regulating imbalance data
from scipy.stats import skew                              # Library for skewness and kurtosis
from scipy.stats import kurtosis
from scipy import stats                                   # Library for T-test and Chi-square
from scipy.stats import chi2_contingency
import statsmodels.api           as sm                    # Library for OLS
from sklearn.linear_model import LinearRegression         # Library for Linear Reg
from sklearn.metrics import r2_score, mean_squared_error
import math                                               # Library for Logistic Reg
from sklearn.metrics import log_loss
from sklearn.preprocessing import LabelEncoder, StandardScaler # Library for spliting dataset
from sklearn.model_selection import train_test_split
from sklearn import metrics                               # Library for training model
from sklearn.metrics import accuracy_score, log_loss
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import cross_val_score       # Library for testing model
import statistics
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import StackingClassifier
```