TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

BÁO CÁO CUỐI KỲ

PYTHON CHO KHOA HỌC DỮ LIỆU

Đề tài: Sử dụng Machine Learning và xây dựng web chẩn đoán đột quỵ

Giảng viên: Hà Văn Thảo

Thành viên nhóm

Nguyễn Lê Ngọc Duy	20280023
Quách Phong Dương	20280022
Nguyễn Thị Hoa	20280033
Nguyễn Thị Huyền	20280048

Thành phố Hồ Chí Minh, 12-2022

Mục lục

1 Giới thiệu đề tài.

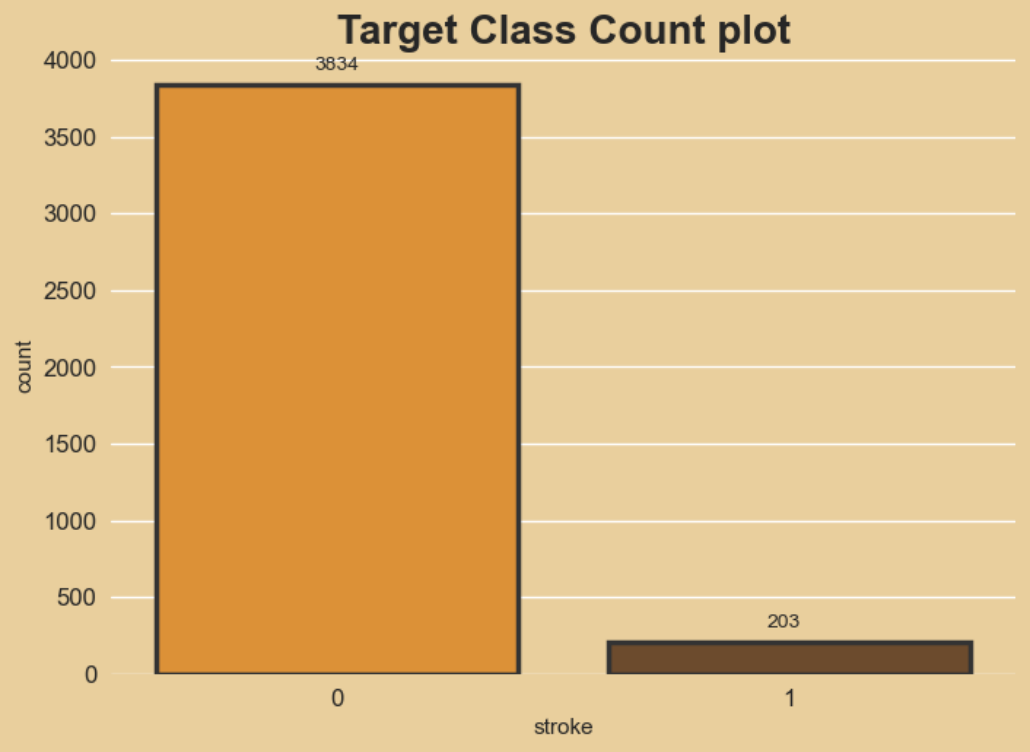
Theo thống kê của tổ chức Y Tế Thế giới (WHO), đột quỵ là nguyên nhân lớn thứ hai gây nên những ca tử vong ở trên thế giới. Hằng năm, khoảng 15 triệu người trên thế giới mắc đột quỵ, căn bệnh đã cướp đi mạng sống của 5 triệu người và khiến 5 triệu người tàn phế suốt đời.
Với mong muốn một phần nào đó giảm thiểu tỉ lệ những người phải bỏ mạng vì đột quỵ, nhóm quyết định xây dựng một web giúp chẩn đoán đột quỵ dựa trên các thông tin về bệnh nhân. Web sẽ nhận vào các thông tin về bệnh nhân và dựa vào các thuật toán Machine Learning để dự đoán xem bệnh nhân có bị đột quỵ hay không.
Bộ dữ liệu sử dụng trong đề tài được lấy từ bộ dataset của FEDESORIANO cách đây 2 năm trước và được lưu lại ở trang web dành cho các cuộc thi về dữ liệu và học máy Kaggle.
Bộ dữ liệu thu thập kết quả khảo sát của 5110 người. Các đặc trưng của bộ dữ liệu bao gồm:

- `id`: ID riêng biệt của mỗi người được khảo sát.
- `gender`: Giới tính của người được khảo sát, gồm 3 giá trị: `Male`, `Female` và `Other`.
- `age`: Tuổi của người được khảo sát.
- `hypertension`: Mang giá trị 0 nếu người được khảo sát không mắc cao huyết áp, mang giá trị 1 nếu người được khảo sát mắc cao huyết áp.
- `heart_disease`: Mang giá trị 0 nếu người được khảo sát không mắc bệnh tim, mang giá trị 1 nếu người được khảo sát mắc bệnh tim.
- `ever_married`: Mang giá trị `Yes` nếu người được khảo sát đã từng kết hôn, mang giá trị `No` nếu người được khảo sát chưa từng kết hôn.
- `work_type`: Loại công việc của người được khảo sát, gồm 5 giá trị: `Private`, `Self-employed`, `Govt_job`, `children` và `Never_worked`.
- `Residence_type`: Khu vực sinh sống của người được khảo sát, gồm 2 giá trị: `Urban` và `Rural`.
- `avg_glucose_level`: Lượng đường huyết trung bình của người được khảo sát.
- `bmi`: Chỉ số khối cơ thể (BMI) của người được khảo sát.
- `smoking_status`: Tình trạng hút thuốc của người được khảo sát, gồm 4 giá trị: `formerly smoked`, `never smoked`, `smokes` và `Unknown`. Giá trị `Unknown` trong trường hợp này có nghĩa là tình trạng hút thuốc của người đó không tồn tại.
- `stroke`: Mang giá trị 0 nếu người được khảo sát không mắc đột quỵ, mang giá trị 1 nếu người được khảo sát mắc đột quỵ. Đây cũng chính là biến mục tiêu của bài toán.

2 Xử lý ban đầu.

Từ dữ liệu thu được, bằng cách sử dụng thư viện `pandas` và `numpy`, nhóm sẽ thực hiện một số bước xử lý ban đầu như sau:

- Nhận thấy rằng `bmi` là đặc trưng duy nhất có tồn tài dữ liệu bị khuyết (201 dữ liệu), do đó nhóm quyết định thay thế các giá trị bị khuyết này bằng giá trị trung bình của tất cả các điểm dữ liệu không bị khuyết trong đặc trưng.
- Đối với `age`, nhóm quyết định chuyển kiểu dữ liệu của đặc trưng này về `int64` và thay các giá trị 0 trong đặc trưng này bằng 1.



Hình 1: Biểu đồ thể hiện tỉ lệ mắc đột quỵ của người được khảo sát.

- Đối với `smoking_status`, nhóm có tìm hiểu và nhận thấy rằng, những đứa trẻ không quá 9 tuổi có tỉ lệ hút thuốc không đáng kể (1%), do đó nhóm thay giá trị của `smoking_status` từ `Unknown` thành `never smoked` với những quan trắc nào có `age` không lớn hơn 9, còn lại sẽ được loại bỏ.
- Cuối cùng, nhóm loại bỏ những điểm dữ liệu mà thuộc tính `gender` mang giá trị `Other`.

3 Phân tích và trực quan hoá dữ liệu.

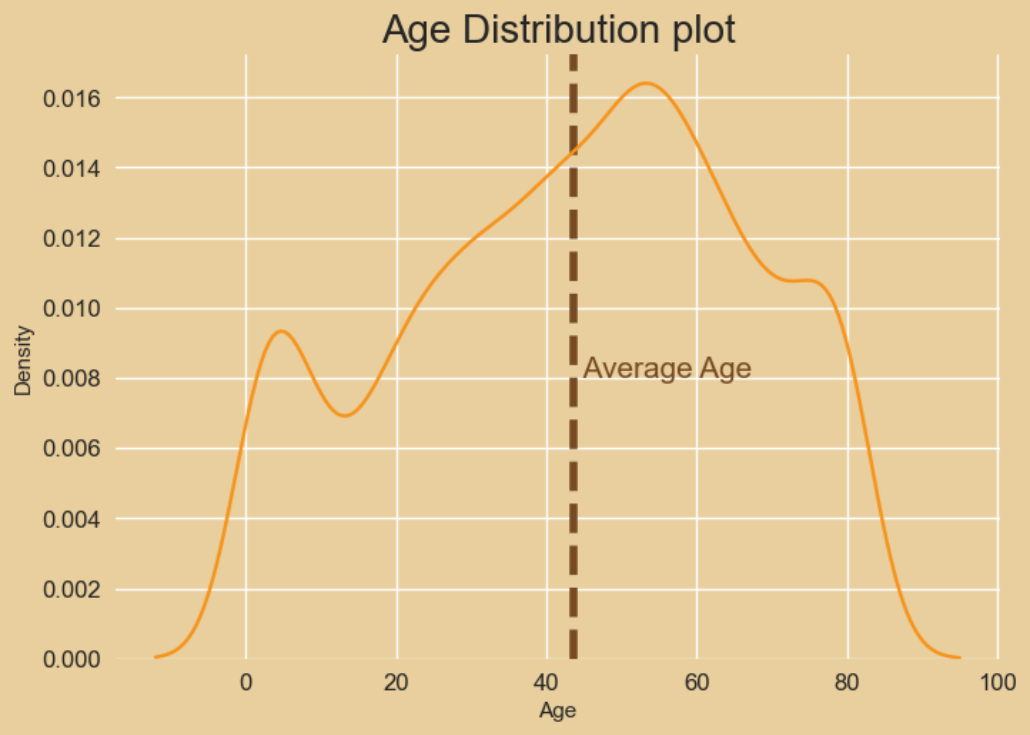
Trong phần này, nhóm sử dụng các thư viện `matplotlib` và `seaborn` để trực quan hoá dữ liệu.

3.0.1 Biến mục tiêu `stroke`.

Thông qua biểu đồ ở hình 1, ta thấy rằng có một sự mất cân bằng rất lớn ở bộ dữ liệu này, khi số lượng người khảo sát bị đột quỵ thấp hơn rất nhiều so với số lượng người không bị đột quỵ.

3.0.2 Độ tuổi `age`.

Thông qua biểu đồ ở hình 2, ta thấy rằng độ tuổi của nhóm khảo sát phân bố chủ yếu từ khoảng 0 đến 80 tuổi, trong đó có một số điểm dữ liệu có độ tuổi lớn hơn 80 tuổi. Độ tuổi trung bình của cuộc khảo sát là 43.8 tuổi.



Hình 2: Biểu đồ thể hiện phân bố độ tuổi của người được khảo sát

3.0.3 Giới tính `gender`.

Thông qua biểu đồ ở hình 3, ta thấy rằng tỷ lệ nam và nữ trong cuộc khảo sát lần lượt là 41.07% và 58.93%. Đồng thời, tỷ lệ nam và nữ bị đột quỵ lần lượt là 5.47% và 5.17%. Tuy nhiên, do tỉ lệ chênh lệch không quá vượt chội nên không thể rút ra điều gì đặc biệt về giới tính.

3.0.4 Độ tuổi `age`, lượng đường huyết trung bình `avg_glucose_level` và chỉ số BMI `bmi`.

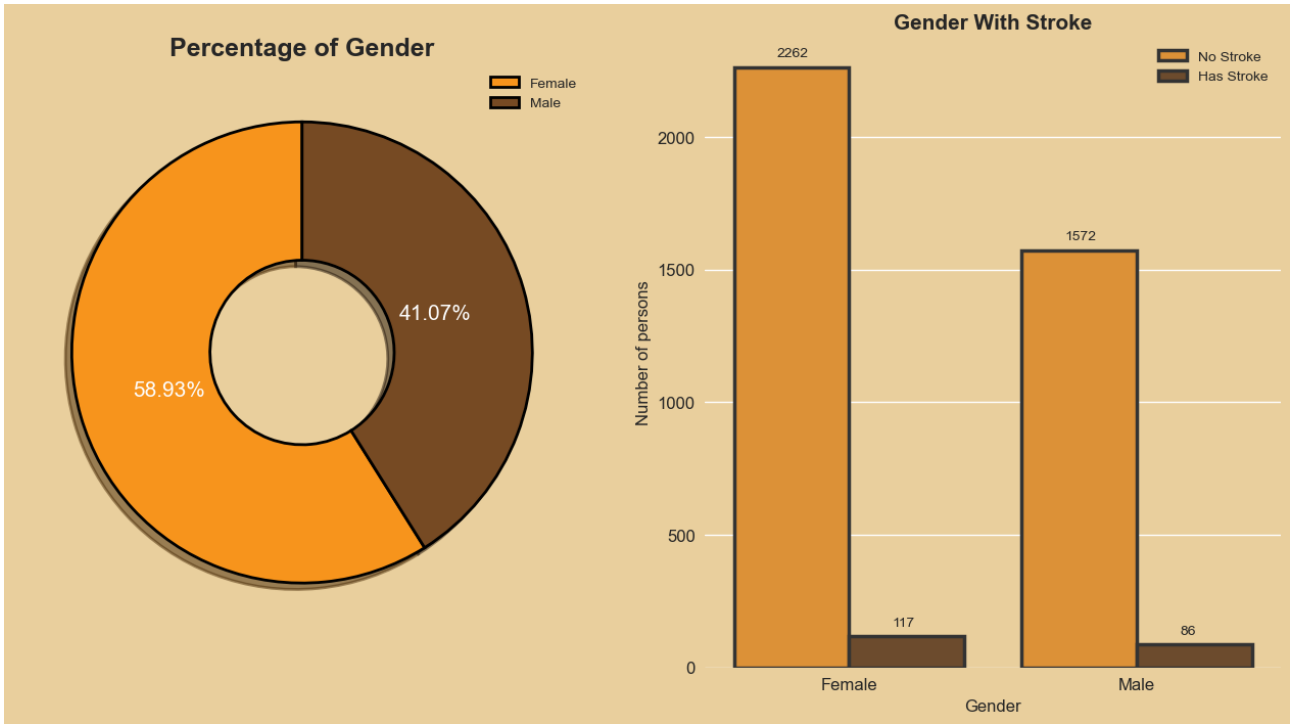
Từ các biểu đồ boxplot ở hình 4, ta nhận thấy rằng mặc dù chỉ số BMI ở cả hai nhóm đột quỵ và không đột quỵ đều như nhau, nhưng lượng đường huyết trung bình của nhóm đột quỵ (mang giá trị `stroke` là 1) thấp hơn rất nhiều so với nhóm không đột quỵ (mang giá trị `stroke` là 0). Đồng thời, độ tuổi của nhóm đột quỵ cũng thấp hơn rất nhiều so với nhóm không đột quỵ.

3.0.5 Huyết áp `hypertension`.

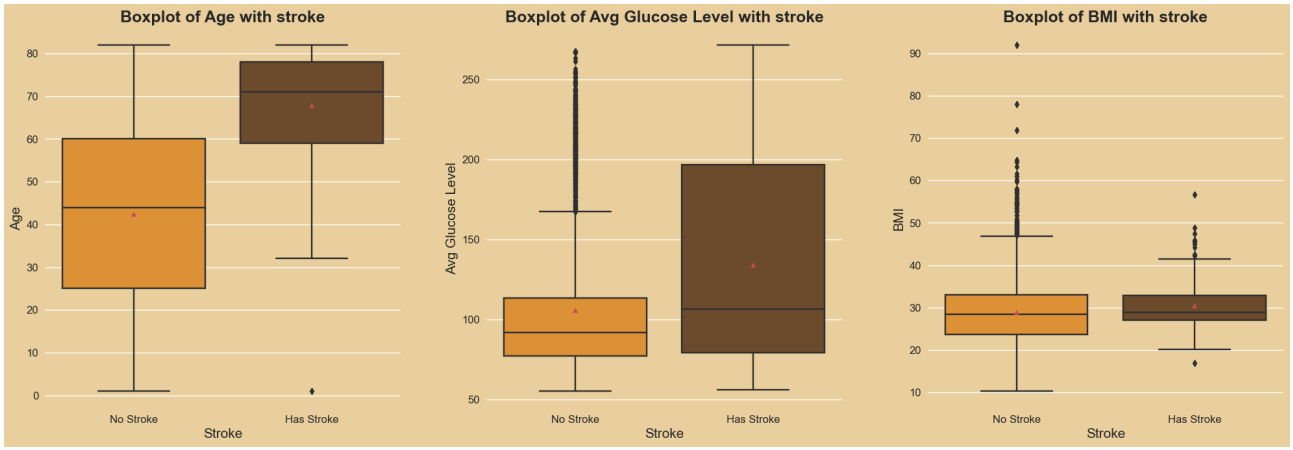
Từ các biểu đồ ở hình 5, ta nhận thấy tuy chỉ có 11% số người tham gia khảo sát bị mắc cao huyết áp, tuy nhiên tỉ lệ mắc đột quỵ của nhóm người này lại cao hơn rất nhiều so với nhóm người không mắc cao huyết áp. Do đó, ta có thể kết luận rằng cao huyết áp có thể là một yếu tố gây đột quỵ.

3.0.6 Tiền sử bệnh tim `heart_disease`.

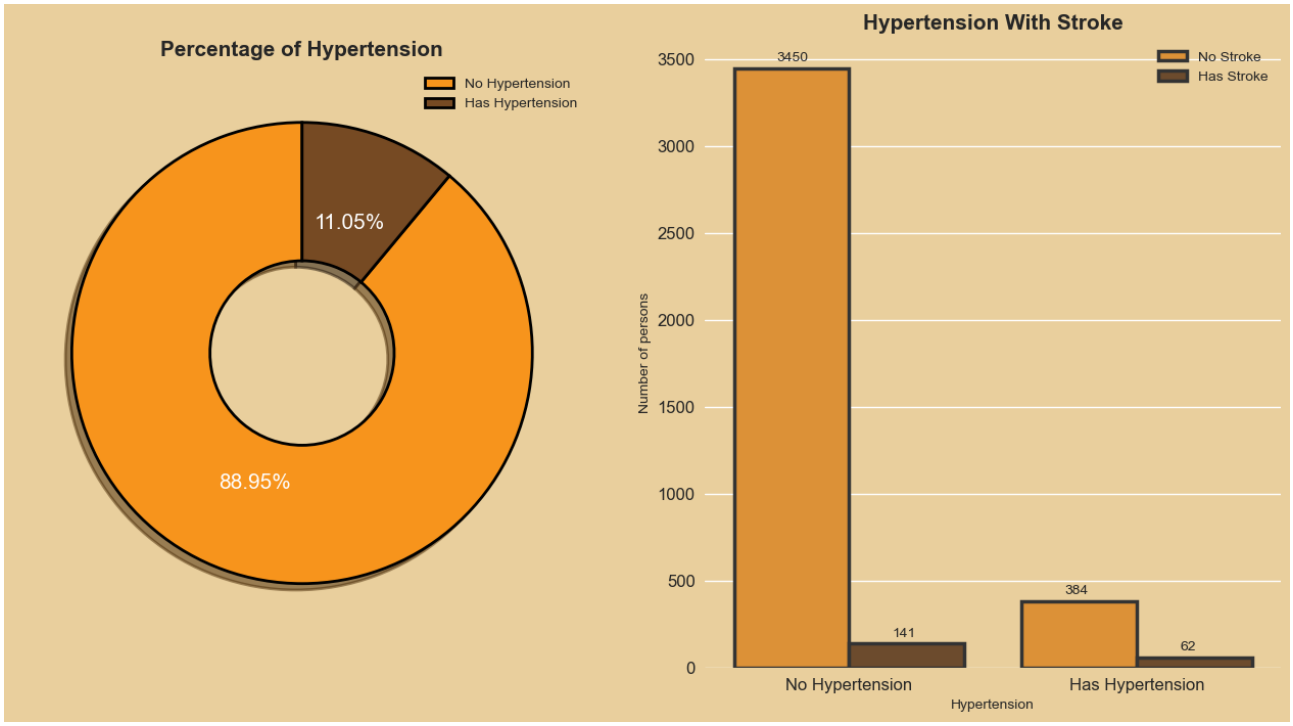
Từ các biểu đồ ở hình 6, ta nhận thấy tuy chỉ có 5.67% số người tham gia khảo sát bị mắc bệnh tim, tuy nhiên tỉ lệ mắc đột quỵ của nhóm người này lại cao hơn rất nhiều so với nhóm người không mắc



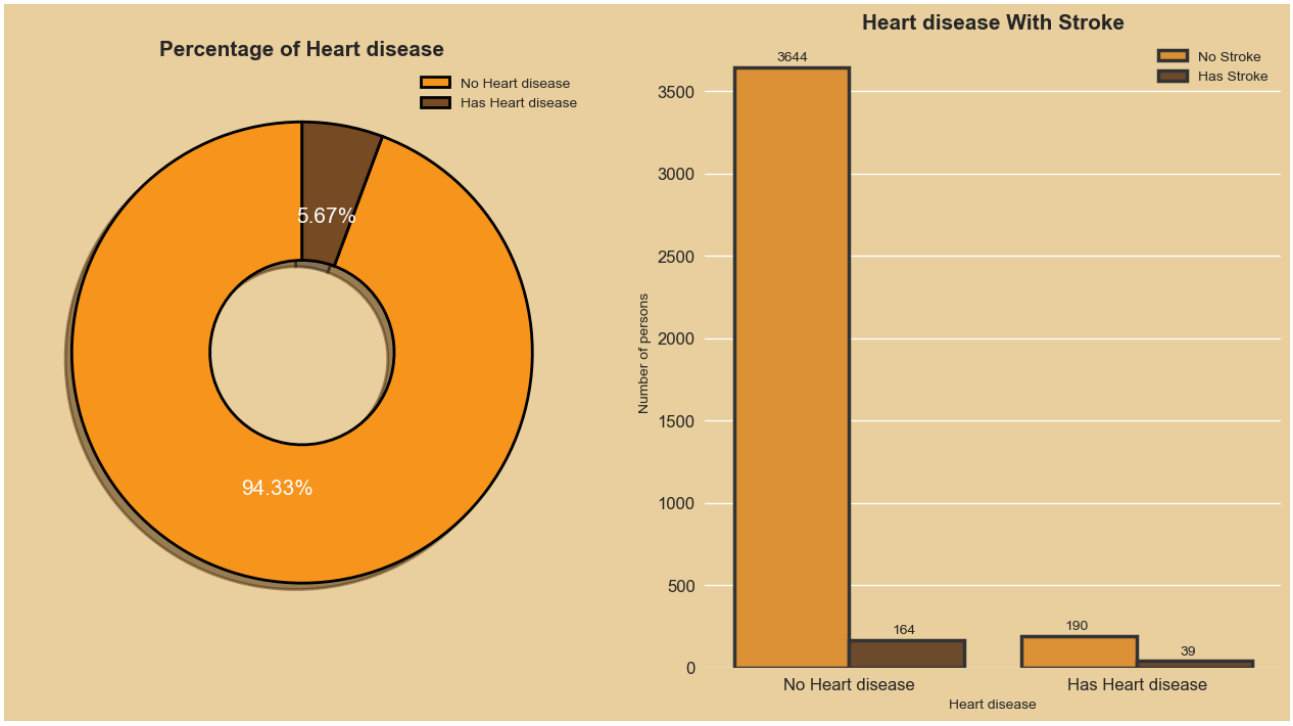
Hình 3: Biểu đồ thể hiện phân bố giới tính của người được khảo sát.



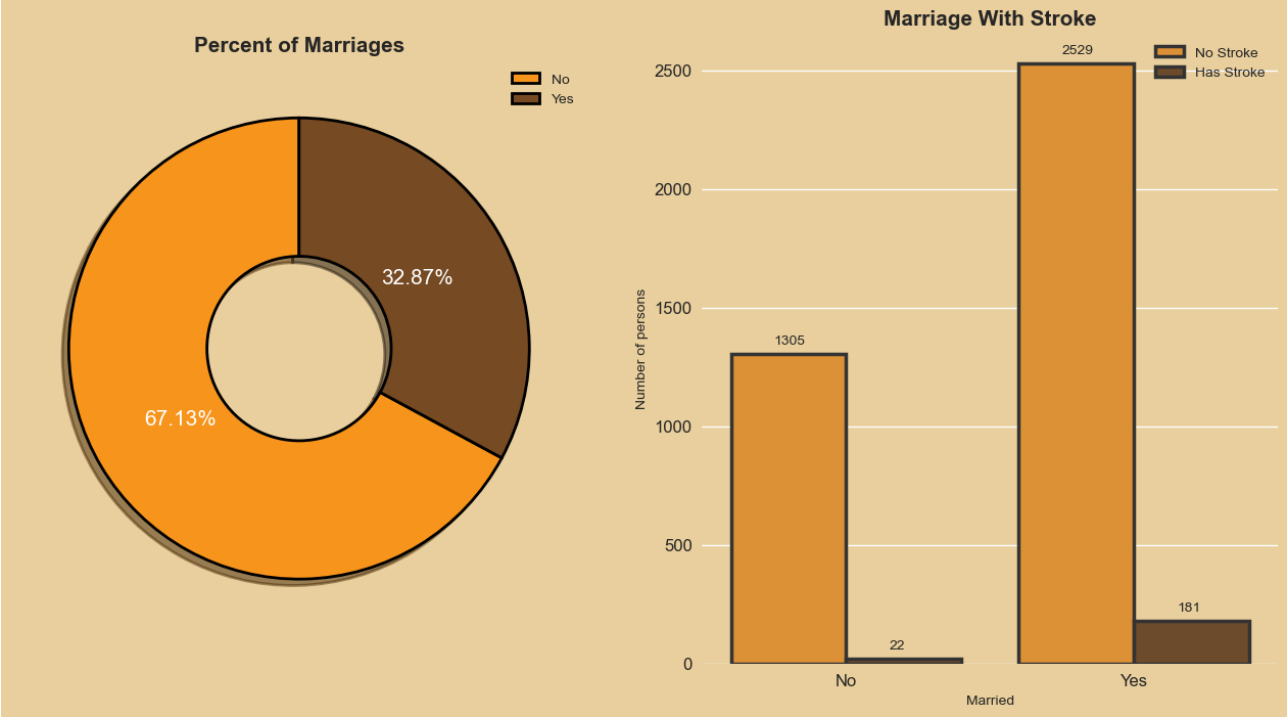
Hình 4: Biểu đồ hộp theo thứ tự: độ tuổi, lượng đường huyết trung bình và chỉ số BMI



Hình 5: Biểu đồ thể hiện tỉ lệ mắc đột quỵ theo nhóm cao huyết áp.



Hình 6: Biểu đồ thể hiện tỉ lệ mắc đột quỵ theo nhóm bệnh tim.



Hình 7: Biểu đồ thể hiện tỉ lệ mắc đột quỵ theo nhóm hôn nhân.

bệnh tim. Do đó, ta có thể kết luận rằng bệnh tim có thể là một yếu tố gây đột quỵ.

3.0.7 Tiền sử kết hôn `ever_married`.

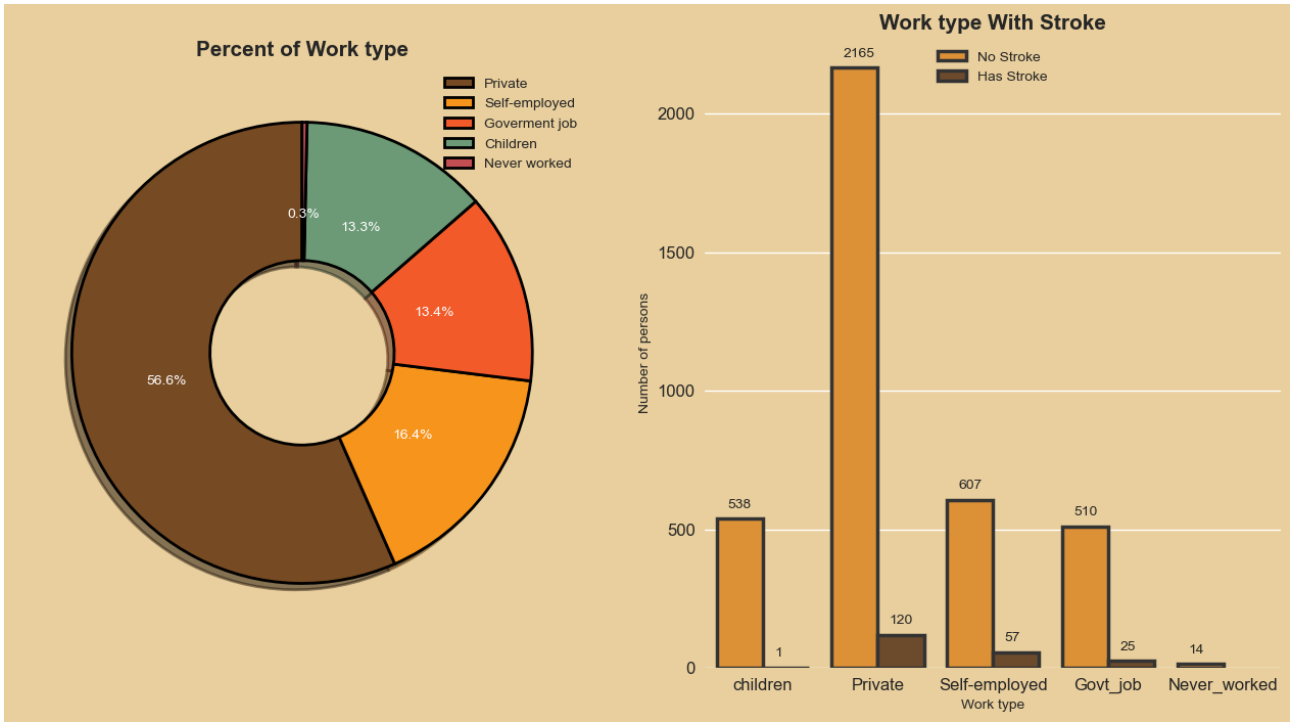
Trong số những người được tham gia khảo sát, có khoảng 1/3 số người đã từng kết hôn (hình 7), và tỉ lệ mắc đột quỵ của hai nhóm kết hôn và không kết hôn có sự chênh lệch. Những người chưa kết hôn mắc phải đột quỵ chỉ chiếm khoảng 1.6% so với tổng si61 lượng người chưa kết hôn, còn ở phân khúc đã từng kết hôn thì lại khác, khi tỉ lệ họ mắc phải đột quỵ ở mức 6.6%.

3.0.8 Loại hình công việc `work_type`.

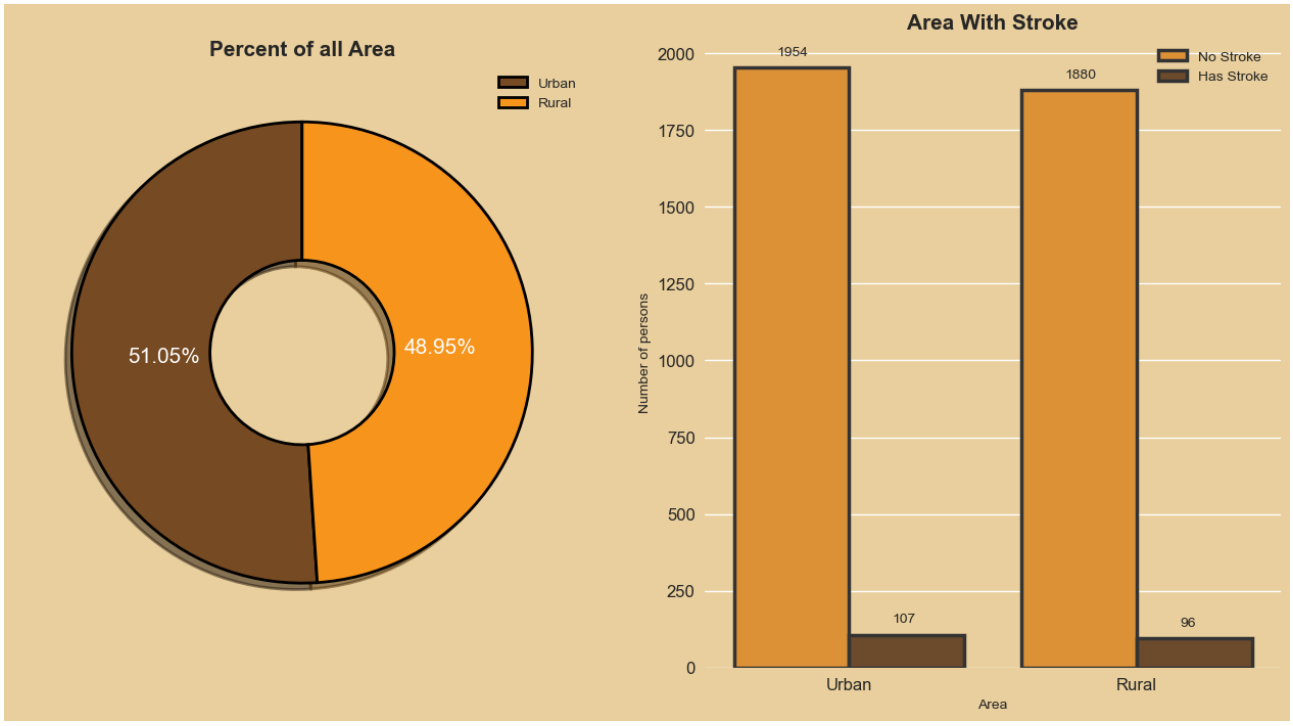
Ở lớp trẻ em, dường như là một sự miễn nhiễm so với các lứa tuổi khác khi chỉ có 1 trường hợp bị đột quỵ so với hàng trăm trường hợp. Lớp công việc private thì có thể thấy tỉ lệ khá ổn định khi chỉ có 5% người mắc đột quỵ. Ở lớp tự kinh doanh thì khá cao khi đạt 8.5% người mắc đột quỵ, nhóm công việc chính phủ thì thấp hơn khi chỉ có 4.6% người mắc đột quỵ. Còn lại ở lớp chưa từng làm việc thì kết quả là 100% người không bị đột quỵ. Ta có thể xem xét gộp nhóm chưa từng làm việc và trẻ em lại với nhau. (hình 8)

3.0.9 Nơi cư trú `Residence_type`.

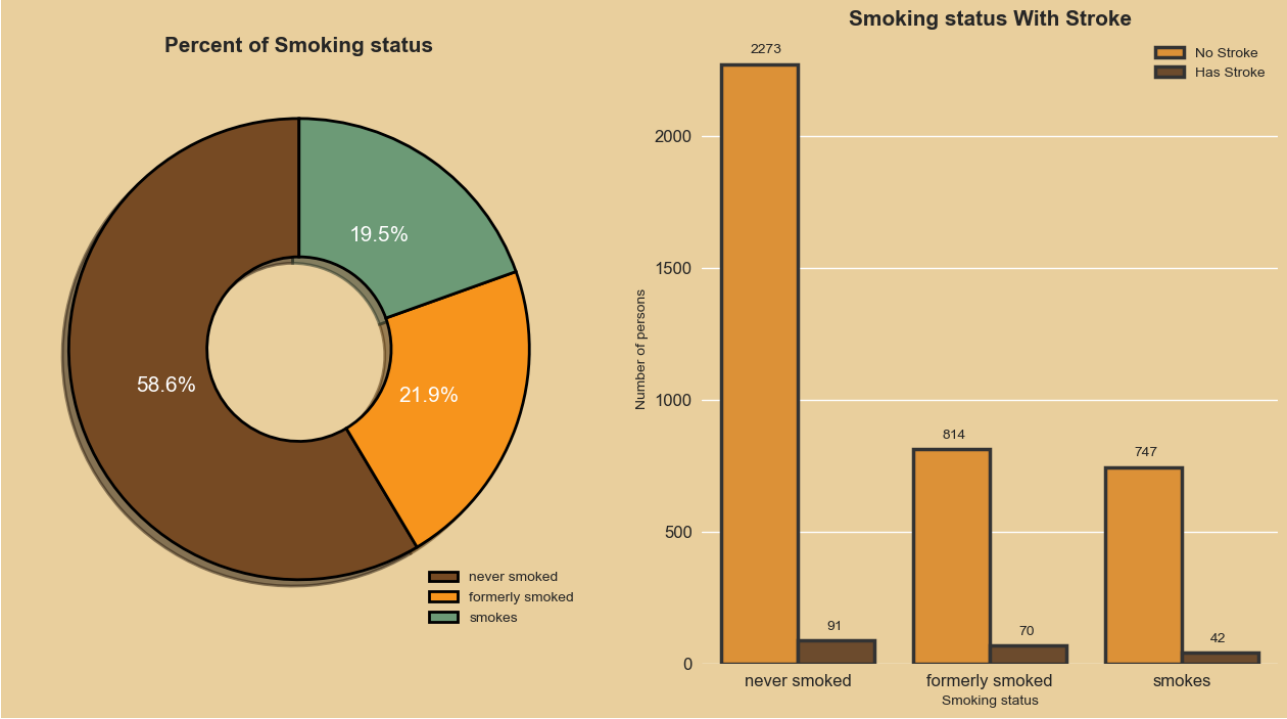
Dường như nơi cư trú không ảnh hưởng quá nhiều đến việc mắc đột quỵ. Tỉ lệ mắc đột quỵ của nhóm ở nông thôn và thành phố là gần ngang nhau (hình 9).



Hình 8: Biểu đồ thể hiện tỉ lệ mắc đột quỵ theo nhóm công việc.



Hình 9: Biểu đồ thể hiện tỉ lệ mắc đột quỵ theo nhóm cư trú.



Hình 10: Biểu đồ thể hiện tỉ lệ mắc đột quỵ theo nhóm hút thuốc.

3.0.10 Tình trạng hút thuốc `smoking_status`.

Tỉ lệ mắc đột quỵ của nhóm hút thuốc là 6%, nhóm từng hút thuốc là 8% và nhóm không hút thuốc là 3% . Tỉ lệ mắc đột quỵ của nhóm hút thuốc và từng hút thuốc là gần như bằng nhau, nhóm không hút thuốc lại cao hơn nhóm không biết hút thuốc. Điều đó cho thấy hút thuốc là một trong các yếu tố gây đột quỵ. (hình 10)

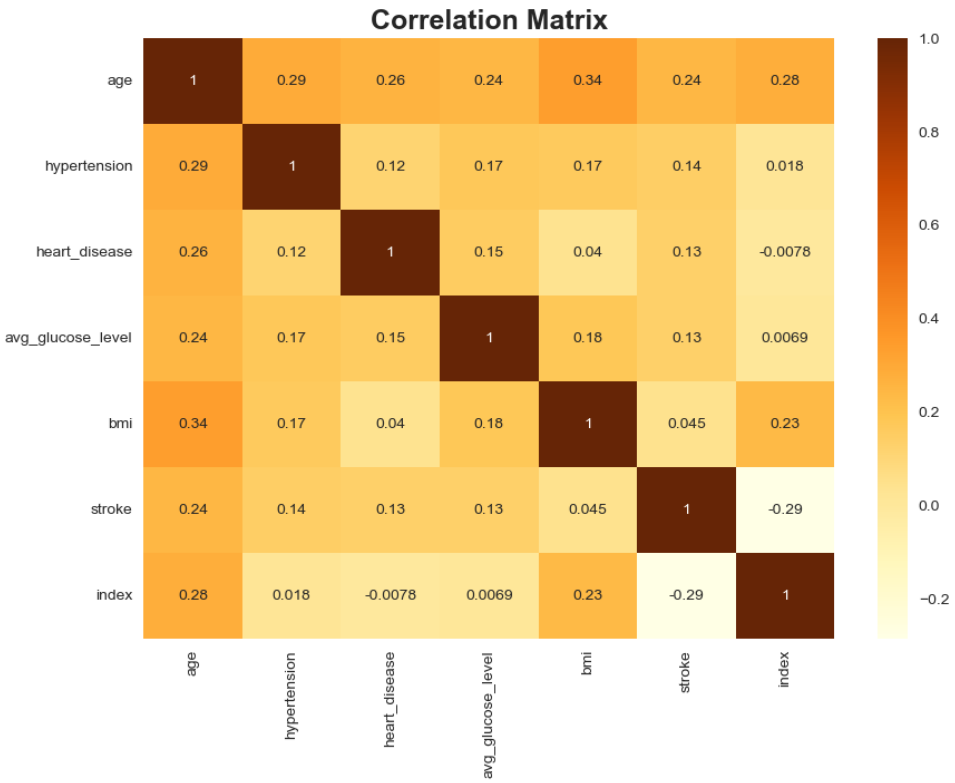
3.0.11 Ma trận tương quan giữa các biến.

Các biến trong bộ dữ liệu không có sự tương quan mạnh với nhau. Ta có thể thấy được điều này thông qua ma trận tương quan ở hình 11.

4 Xây dựng Pipeline.

Trong phần này, chúng ta sẽ xây dựng một Pipeline để giải quyết vấn đề được đặt ra phía trên với hai thư viện `scikit-learn` và `imblearn` của `Python`. Pipeline là một quy trình kết hợp các bước lại với nhau theo trình tự:

- Preprocessing: Là quá trình biến đổi dữ liệu thô ban đầu thành dữ liệu có cấu trúc tối ưu và hữu ích hơn cho việc sử dụng các mô hình học máy.
- Resampling: Do dữ liệu ban đầu không đều nhau (số lượng class 0 nhiều hơn class 1), nên chúng ta cần phải lấy mẫu lại (resample) dữ liệu trong khi huấn luyện để đảm bảo tính đồng nhất của dữ liệu được huấn luyện, từ đó các mô hình sẽ được huấn luyện tốt hơn.



Hình 11: Ma trận tương quan giữa các đặc trưng.

- Modeling: Bước cuối cùng của Pipeline chính là việc huấn luyện dữ liệu, so sánh và kiểm tra trên tập kiểm thử. Nếu hiệu năng của mô hình không cao, ta có thể phải thực hiện thêm một vài bước tối ưu mô hình để hiệu năng của mô hình được cải thiện.

4.1 Preprocessing.

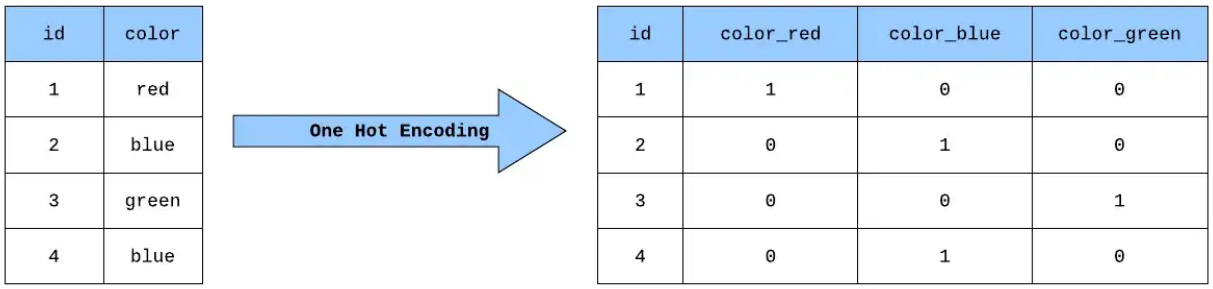
Bước đầu tiên của Pipeline là tiền xử lý dữ liệu. Trong bài toán này, chúng ta sẽ sử dụng hai kỹ thuật biến đổi dữ liệu thường được sử dụng là Min Max Scaler đối với các biến liên tục (`int64, float64, e.t.c.`) và One Hot Encoder đối với các biến rời rạc (`object`).

4.1.1 Min Max Scaler.

Các đặc trưng của dữ liệu thường không có cùng miền giá trị và độ đo, ví dụ như việc so sánh giữa độ tuổi và bmi của một nhóm người nào đó. Việc này có thể ảnh hưởng không tốt đến hiệu năng của các mô hình học máy, ví dụ như mô hình học máy có thể ưu tiên học các đặc trưng có miền giá trị cao hơn và bỏ qua các đặc trưng có miền giá trị nhỏ hơn.
Để khắc phục tình trạng này, ta có thể dùng thuật toán Min Max Scaler cho các đặc trưng có kiểu dữ liệu liên tục để đưa chúng về cùng một dạng phân phối miền giá trị nằm trong đoạn $[-1; 1]$ bằng công thức $x' = \dfrac{x - x_{\min}}{x_{\max} - x_{\min}}$ với $x_{\min}$ và $x_{\max}$ là giá trị lớn nhất và nhỏ nhất của vector $\mathbf{x}$.

4.1.2 One Hot Encoder.

One Hot Encoder là một kỹ thuật biến đổi các thuộc tính có kiểu dữ liệu rời rạc thành các ma trận gồm các con số 0 và 1 với giá trị label tương ứng. Việc biến đổi này sẽ giúp các mô hình học máy hiểu



Hình 12: Minh hoạ thuật toán One Hot Encoder

được các biến rời rạc và giúp cho việc huấn luyện trở nên dễ dàng hơn (minh hoạ như hình 12).

4.2 Resampling và thuật toán SMOTEENN.

Như đã đề cập ở trên, do số lượng các giá trị ở đầu ra là không đều nhau nên điều này sẽ ảnh hưởng không tốt đến mô hình học máy, vì mô hình sẽ ưu tiên dự đoán giá trị có số lượng cao hơn trong tập huấn luyện. Để khắc phục tình trạng này, sau khi biến đổi tập huấn luyện ở bước trên, ta sẽ tiến hành lấy mẫu lại ở tập huấn luyện để cân bằng lại số lượng các quan trắc ở mỗi lớp.
Một trong những thuật toán Resampling phổ biến chính là thuật toán **SMOTEENN**. Thuật toán này là sự kết hợp của thuật toán **SMOTE** và thuật toán Edited Nearest Neighbors (**ENN**). Thuật toán **SMOTE** sẽ tạo ra các quan trắc mới ở các lớp ít quan trắc hơn bằng cách lấy các quan trắc gần nhất của chúng và tạo ra các quan trắc mới ở giữa chúng. Thuật toán **ENN** sẽ loại bỏ các quan trắc ở các lớp nhiều quan trắc hơn nếu chúng không phải là các quan trắc gần nhất của các quan trắc ở các lớp ít quan trắc hơn.
Thuật toán **SMOTEENN** sẽ kết hợp cả hai thuật toán trên với nhau để tạo ra các quan trắc mới ở các lớp ít quan trắc hơn và loại bỏ các quan trắc ở các lớp nhiều quan trắc hơn nếu chúng không phải là các quan trắc gần nhất của các quan trắc ở các lớp ít quan trắc hơn. Thuật toán được minh hoạ ở hình 13.

4.3 Modelling.

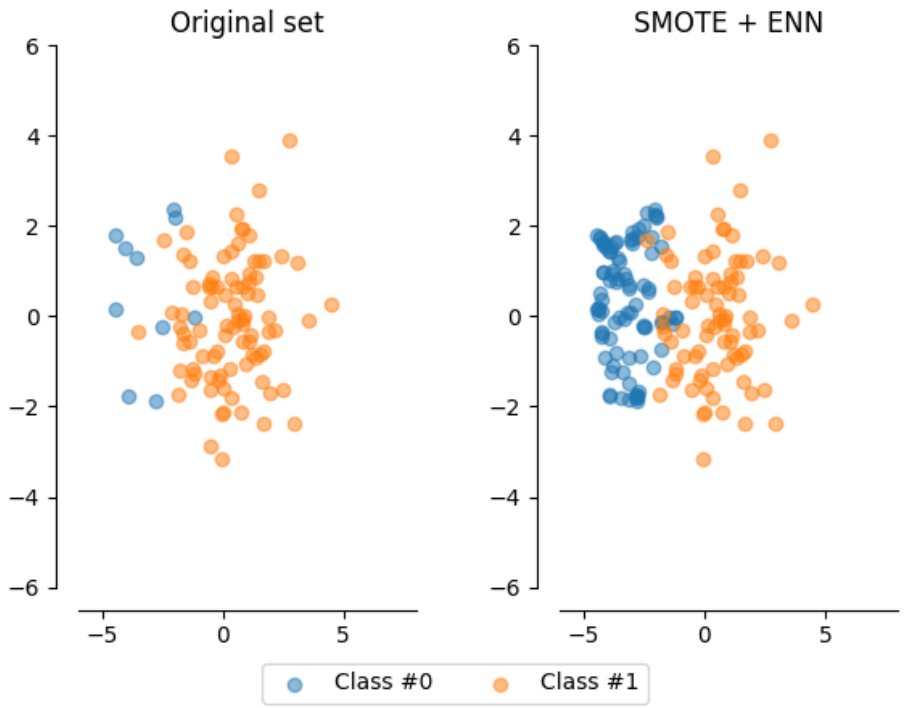
Trong phần này chúng ta sẽ cùng xây dựng các mô hình học máy để giải quyết bài toán xác định bệnh nhân đột quỵ. Chúng ta sẽ xây dựng các mô hình được liệt kê ở bên dưới.

4.3.1 Logistic Regression.

Logistic Regression là 1 mô hình được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc, cụ thể trong bài toán này là 0, 1. Thuật toán sẽ dự đoán giá trị xác suất của mỗi lớp và chọn một ngưỡng để phân loại các lớp này, thông thường ngưỡng được chọn là 0.5.

4.3.2 Support Vector Machine.

Support Vector Machine là một mô hình học máy được sử dụng để phân loại dữ liệu. Mô hình này sẽ tìm ra một đường phân chia tối ưu giữa các lớp dữ liệu. Đường phân chia này được tìm ra bằng cách tìm ra các điểm dữ liệu gần biên phân chia nhất (các điểm này được gọi là support vector) và tìm ra đường phân chia tối ưu dựa trên các điểm này.



Hình 13: Minh hoạ thuật toán SMOTEENN

4.3.3 Stochastic Gradient Descent.

Đây là một kỹ thuật tối ưu được sử dụng để học những mô hình học máy tuyến tính và không thuộc một mô hình mọc máy nào cụ thể. Theo đó, ở mỗi vòng lặp, ta tính gradient sử dụng một mẫu ngẫu nhiên và cập nhật các tham số của mô hình theo hướng âm của gradient. Điều này giúp cho việc tối ưu hóa mô hình được nhanh hơn.

4.3.4 Decision Tree.

Decision Tree là một mô hình học máy được sử dụng để phân loại dữ liệu. Mô hình này sẽ tạo ra các cây quyết định dựa trên các thuộc tính của dữ liệu để phân loại các quan trắc. Cây quyết định sẽ được tạo ra bằng cách chọn ra thuộc tính có độ phân tán cao nhất (tức là thuộc tính có độ lớn nhất của entropy) và chia tập dữ liệu thành các tập con dựa trên các giá trị của thuộc tính đó. Quá trình này sẽ được lặp lại cho đến khi đạt được một điều kiện dừng nào đó.

4.3.5 Multi-Layer Perceptron.

Multiplayer Perceptron là một mô hình học máy được sử dụng để phân loại dữ liệu. Mô hình này sẽ tạo ra một mạng nơ-ron đa tầng để phân loại các quan trắc. Mạng nơ-ron này sẽ được tạo ra bằng cách kết nối các nơ-ron với nhau. Mỗi nơ-ron sẽ được kết nối với các nơ-ron ở tầng trước và tầng sau. Mỗi nơ-ron sẽ tính toán một giá trị dựa trên các giá trị đầu vào và trọng số của nó. Sau đó, giá trị này sẽ được đưa vào một hàm kích hoạt để chuyển đổi thành một giá trị khác. Quá trình này sẽ được lặp lại cho đến khi đạt được một điều kiện dừng nào đó.

4.3.6 Độ đo dùng để đánh giá các mô hình.

Trong bài toán này, vì đây là một bài toán phân loại với phân phối các lớp không cân bằng (số lượng lớp 0 hơn rất nhiều so với số lượng lớp 1), chúng ta sẽ sử dụng các độ đo sau để đánh giá các mô hình:

- Confusion matrix (Ma trận nhầm lẫn): Đối với bài toán đang xét, ma trận nhầm lẫn là một ma trận kích thước 2x2. Ma trận này sẽ được sử dụng để đánh giá các mô hình dựa trên số lượng các dự đoán đúng và sai của mô hình. Ma trận này gồm có 4 giá trị (xem hình 14):
  - True Positive (TP): Số lượng các điểm dữ liệu mà mô hình dự đoán là lớp 1 và thực tế là lớp 1.
  - True Negative (TN): Số lượng các điểm dữ liệu mà mô hình dự đoán là lớp 1 và thực tế là lớp 0.
  - False Negative (FN): Số lượng các điểm dữ liệu mà mô hình dự đoán là lớp 0 và thực tế là lớp 1.
  - False Positive (FP): Số lượng các điểm dữ liệu mà mô hình dự đoán là lớp 0 và thực tế là lớp 0.

  Từ ma trận nhầm lẫn ta sẽ suy ra được các độ đo được liệt kê tiếp theo.
- Accuracy: Độ đo này được tính bằng tỉ lệ giữa số lượng các điểm dữ liệu được dự đoán đúng và tổng số lượng các điểm dữ liệu. Công thức tính accuracy như sau:

$$\text{Accuracy} = \frac{\text{TP} + \text{FP}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}$$

- Precision: Độ đo này được tính bằng tỉ lệ giữa số lượng các điểm dữ liệu được dự đoán là lớp 1 và thực tế là lớp 1 và tổng số lượng các điểm dữ liệu được dự đoán là lớp 1. Công thức tính precision như sau:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall: Độ đo này được tính bằng tỉ lệ giữa số lượng các điểm dữ liệu được dự đoán là lớp 1 và thực tế là lớp 1 và tổng số lượng các điểm dữ liệu thực tế là lớp 1. Công thức tính recall như sau:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F_1-score: Đây là trung bình điều hòa của precision và recall. Công thức tính F_1-score như sau:

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

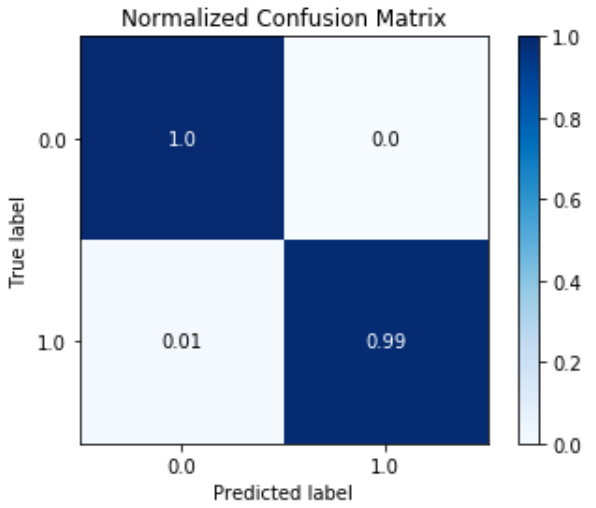
- Precision - recall curve: Đây là đồ thị biểu diễn sự thay đổi của precision và recall theo sự thay đổi của ngưỡng xác suất dự đoán (hình 15).

4.4 Quy trình xây dựng Pipeline.

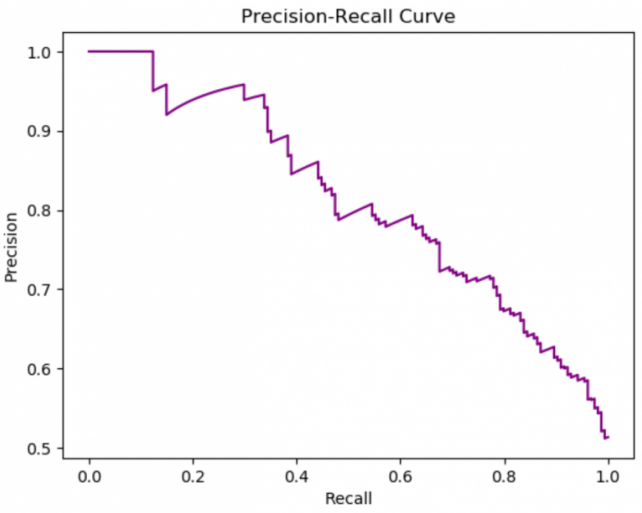
Đầu tiên, chúng ta xây dựng một `ColumnTransformer()` gồm:

- `StandardScaler()`: chuẩn hóa các cột có kiểu dữ liệu liên tục.
- `OneHotEncoder()`: mã hóa các cột có kiểu dữ liệu rời rạc.

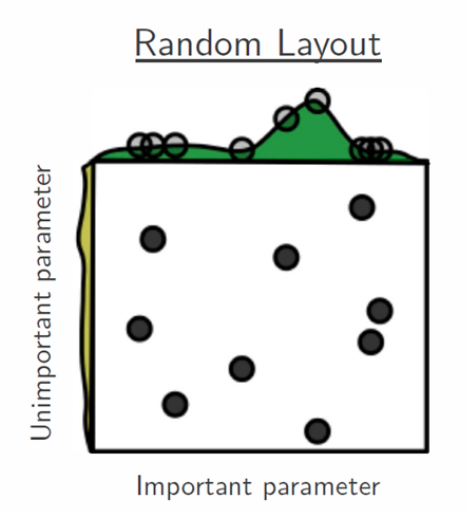
Tiếp theo đó, chúng ta tạo một `Pipeline()` gồm các bước sau:



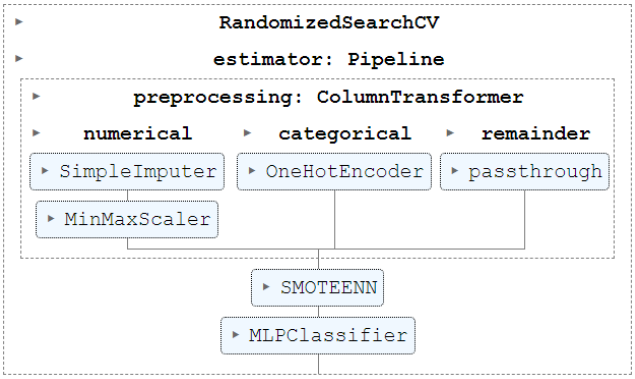
Hình 14: Ma trận nhầm lẫn



Hình 15: Đường cong Precision-Recall



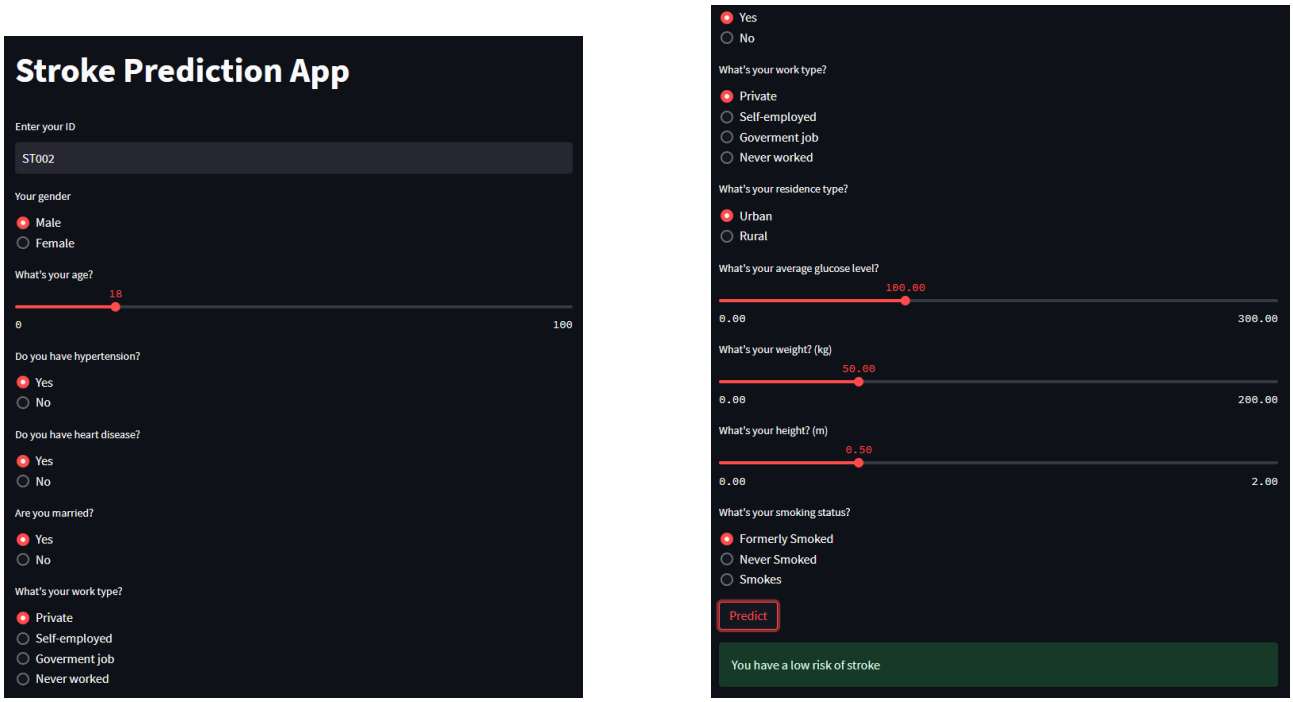
Hình 16: Minh hoạ `RandomizedSearchCV`



Hình 17: Hình ảnh **Best Pipeline**

- `ColumnTransformer()`: xử lý dữ liệu.
- `SMOTE()`: tạo ra các dữ liệu giả.
- `model`: mô hình (trong các mô hình ta liệt kê ở trên) được chọn để huấn luyện.

Chúng ta sẽ sử dụng kiến trúc pipeline này kết hợp với kỹ thuật cross-validation 10-fold bằng cách sử dụng lớp `RepeatedStratifiedKFold()`, kết hợp với `RandomizedSearchCV()` (hình 16) để tìm ra các tham số tốt nhất cho từng mô hình. Sau đó, chúng ta sẽ so sánh điểm số macro-averaged F_1 của từng mô hình và chọn ra mô hình có điểm số macro-averaged F_1 cao nhất làm mô hình chính của Pipeline. Ta thu được **Best Pipeline** (hình 17).
Cuối cùng, chúng ta sử dụng **Best Pipeline** này để huấn luyện toàn bộ dataset mà chúng ta có, và lưu lại thành một file `model.pkl`.



Hình 18: Giao diện web

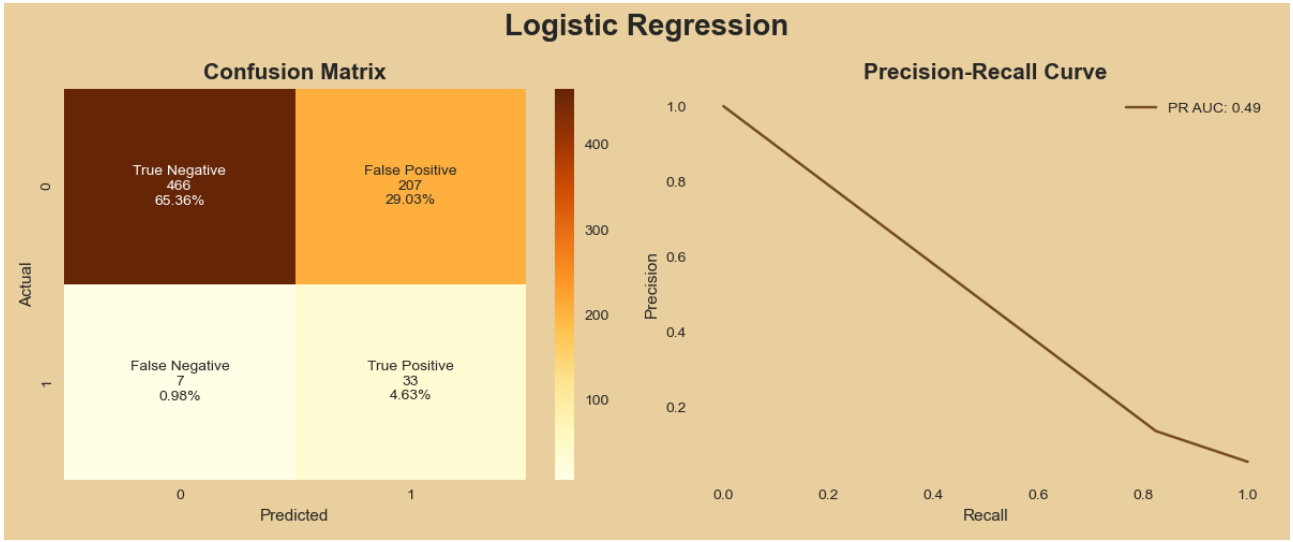
4.5 Triển khai pipeline trên web.

Sau khi có file `model.pkl`, chúng ta sẽ sử dụng file này để triển khai pipeline trên web. Chúng ta sẽ tạo một file `app.py` để chứa toàn bộ code của web sử dụng thư viện `streamlit`. Hình 18 mô tả hình ảnh về giao diện web.

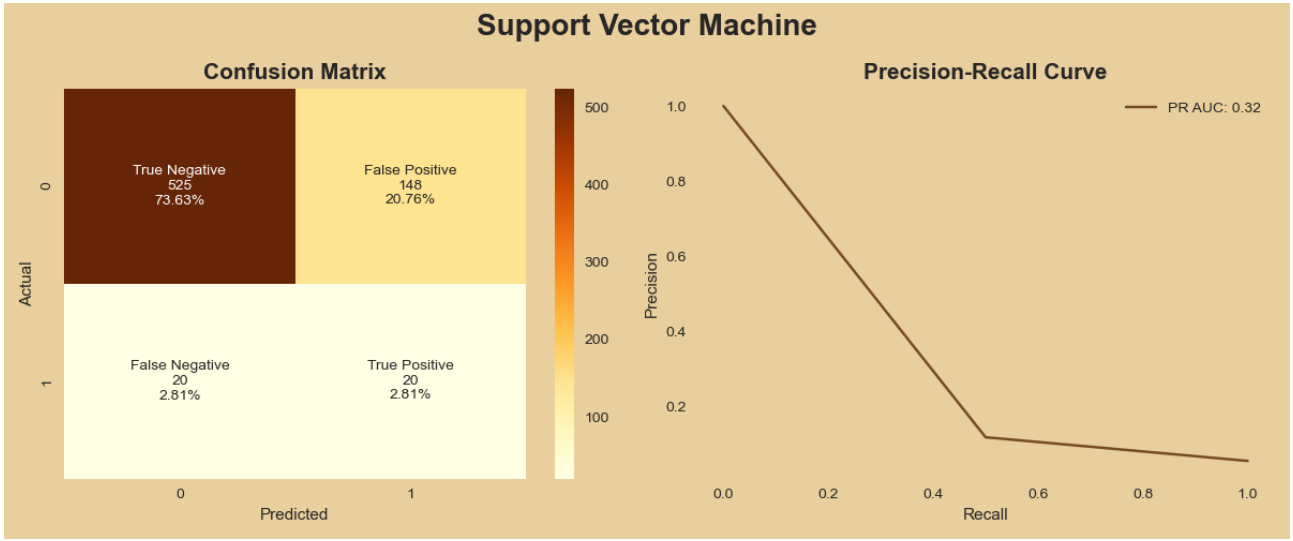
4.6 Kết quả thu được.

Sau khi áp dụng kỹ thuật xây dựng và huấn luyện pipeline với các mô hình đã liệt kê ở trên, ta thu được kết quả như hình 19 và hình 20. Ta có một số nhận xét sau:

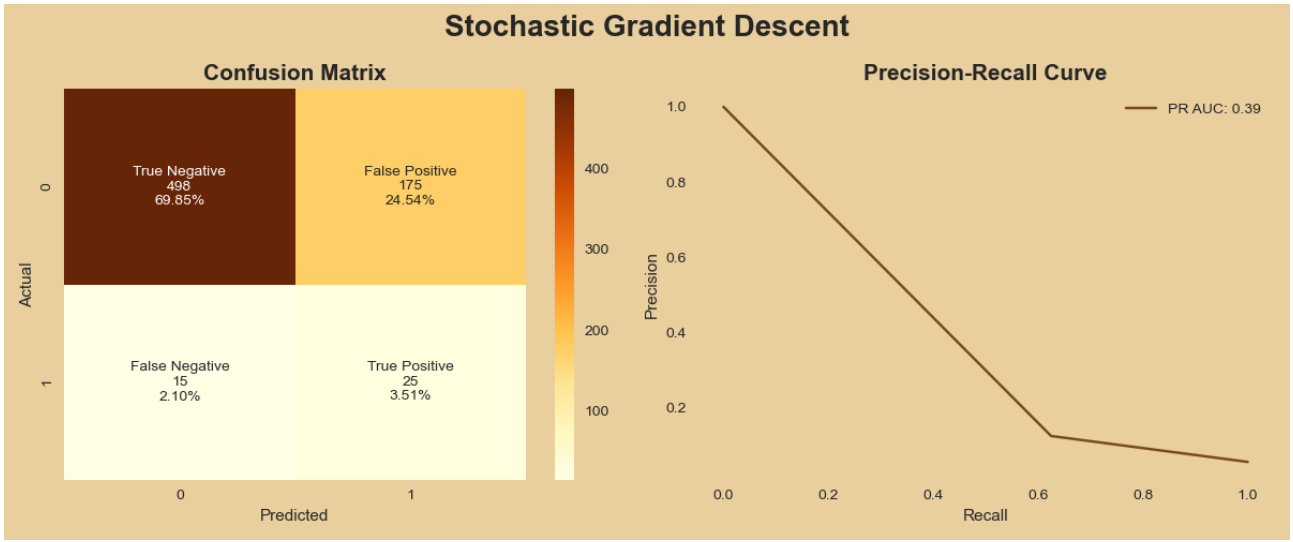
- Mô hình **Logistic Regression** có độ chính xác, điểm số Recall và điểm số weighted-F_1 thấp nhất. Mặt khác, nhìn vào ma trận nhầm lẫn ta cũng có thể phát hiện số lượng quan trắc mà mô hình này dự đoán sai là cao nhất, dù cho việc huấn luyện và tìm tham số tối ưu cho mô hình này diễn ra khá nhanh. Điều đó cho thấy mô hình này không phải là một mô hình tốt để chúng ta có thể sử dụng.
- Bốn mô hình còn lại, gồm **Support Vector Machine, Stochastic Gradient Descent, Decision Tree, Multi-Layer Perceptron** đều có độ chính xác, điểm số Recall và điểm số weighted-F_1 cao hơn mô hình **Logistic Regression**.
- Nhìn vào ma trận nhầm lẫn, ta thấy mô hình **Decision Tree** và **Multi-Layer Perceptron** có số lượng quan trắc mà mô hình dự đoán sai là nhỏ nhất. Điều đó cho thấy cả hai mô hình này đều là những mô hình tốt để chúng ta có thể sử dụng. Tuy nhiên, thời gian huấn luyện của hai mô hình này khá là chậm và tốn rất nhiều tài nguyên.
- Vì chúng ta lựa chọn mô hình với điểm số weighted-averaged F_1 cao nhất, nên dựa vào hình 8, ta có thể chọn mô hình **Multi-Layer Perceptron** làm mô hình chính để sử dụng.



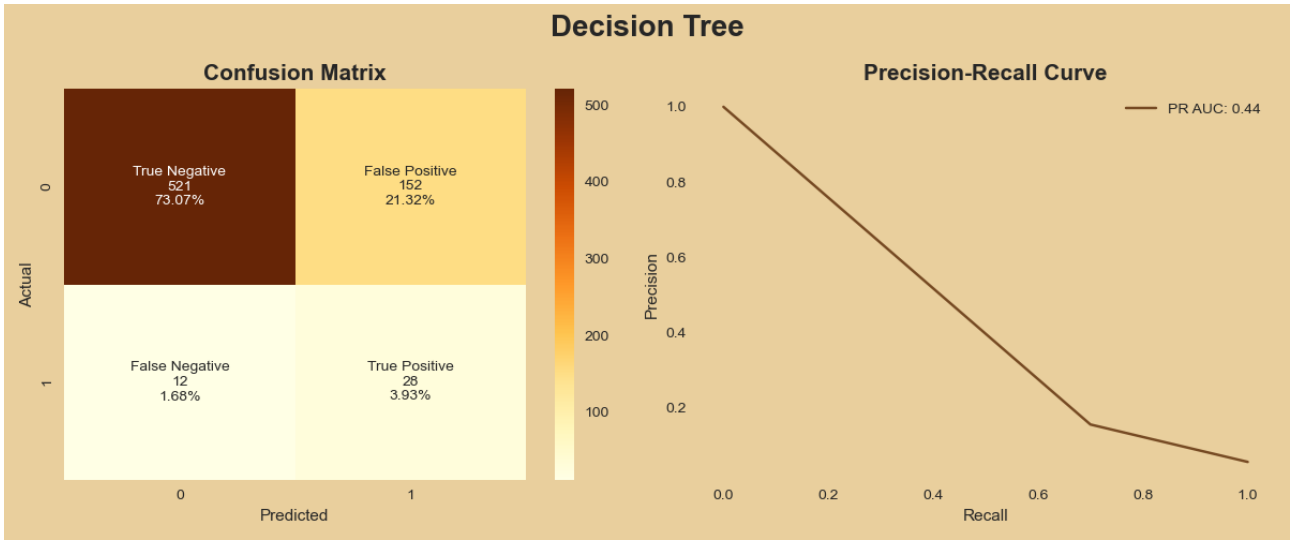
(a) Logistic Regression



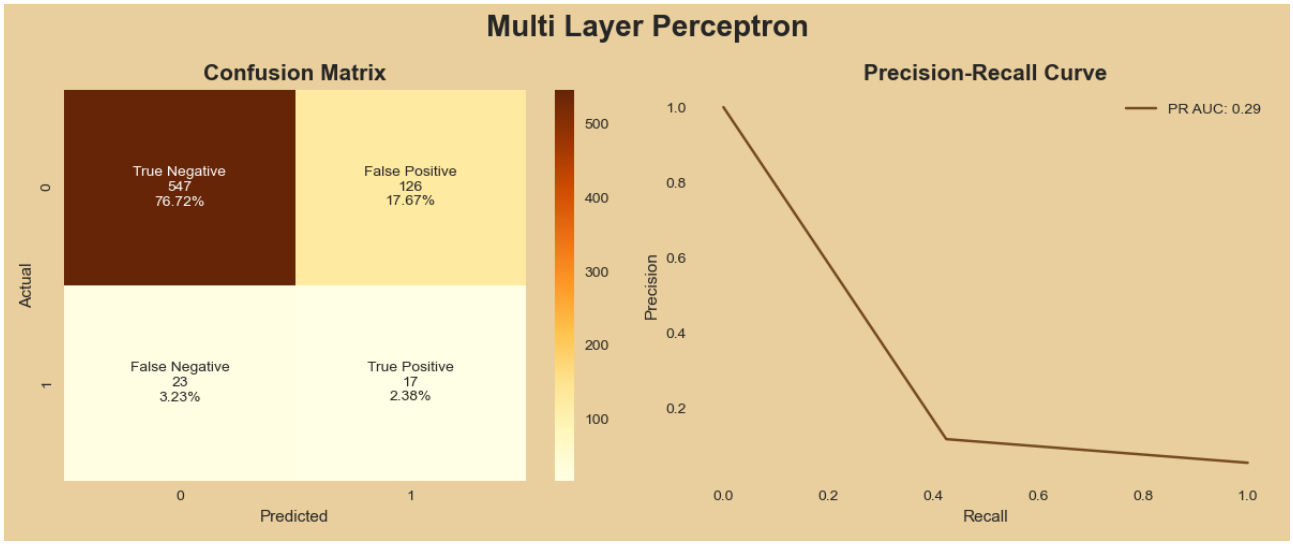
(b) Support Vector Machine



(c) Stochastic Gradient Descent



(d) Decision Tree Classifier



(e) Multi-Layer Perceptron

Hình 19: Ma trận nhầm lẫn và đường cong Precision-Recall của các mô hình

	Model	Accuracy	Weighted Precision	Weighted Recall	Weighted F1
1	Logistic Regression	0.700	0.938	0.700	0.781
2	Support Vector Machine	0.764	0.916	0.764	0.824
3	Stochastic Gradient Descent	0.734	0.923	0.734	0.804
4	Decision Tree	0.770	0.931	0.770	0.830
5	Multi Layer Perceptron	0.791	0.912	0.791	0.841

Hình 20: Kết quả so sánh các thuật toán

5 Đánh giá.

Trong bài báo cáo này, chúng ta đã cùng giải quyết bài toán phân loại bệnh nhân đột quỵ. Dựa vào những phân tích bộ Dataset có ở trên Kaggle, chúng ta đã xây dựng được một pipeline có thể giúp chẩn đoán bệnh nhân đột quỵ. Chúng ta đã áp dụng kỹ thuật cross-validation 10-fold và tìm ra các tham số tốt nhất cho từng mô hình. Sau đó, chúng ta đã so sánh điểm số weighted-average F_1 của từng mô hình và chọn ra mô hình có điểm số weighted-average F_1 cao nhất làm mô hình chính của Pipeline, qua đó thu được **Best Pipeline**. **Best Pipeline** này được sử dụng để huấn luyện lại toàn bộ dataset mà chúng ta có, và mô hình sau đó được lưu lại thành một file `model.pkl`. Cuối cùng, chúng ta đã triển khai **Best Pipeline** này lên web bằng thư viện Streamlit.
Mặc dù web được tạo nên có giao diện đẹp và dễ sử dụng, tuy nhiên mô hình dự báo của chúng ta vẫn chưa thực sự chính xác. Điều này có thể là do bộ dataset mà chúng ta có không đủ lớn, hoặc do mô hình chúng ta chọn không phù hợp. Để có thể cải thiện được độ chính xác của mô hình, chúng ta có thể thực hiện các bước sau:

- Thu thập thêm nhiều dữ liệu hơn cho bộ dataset huấn luyện.
- Tìm hiểu thêm về các mô hình khác như Gradient Boosting, Random Forest, XGBoost, LightGBM,...
- Tìm hiểu thêm về các kỹ thuật xử lý dữ liệu khác như loại bỏ điểm ngoại lai, xử lý dữ liệu thiếu, trích chọn đặc trưng, ...

6 Tham khảo.

- Bộ dataset được sử dụng: https://www.kaggle.com/ronitf/heart-disease-uci
- Thư viện `numpy`: https://numpy.org/
- Thư viện `pandas`: https://pandas.pydata.org/
- Thư viện `matplotlib`: https://matplotlib.org/
- Thư viện `seaborn`: https://seaborn.pydata.org/
- Thư viện `scikit-learn`: https://scikit-learn.org/stable/
- Thư viện `imbalanced-learn`: https://imbalanced-learn.org/stable/
- Thư viện `streamlit`: https://streamlit.io/
- Learn about stroke: https://www.world-stroke.org/world-stroke-day-campaign/why-stroke-matters/learn-about-stroke