

# STROKE PREDICTION

Ngô Minh Khánh  
Nhóm 4



# Mục lục

## NỘI DUNG

- 1 INTRODUCTION
- 2 IMPORT, CHECK THE DATA
- 3 CLEAN DATA
- 4 EDA
- 5 Data Preprocessing
- 6 MODELING
- 7 CONCLUSION

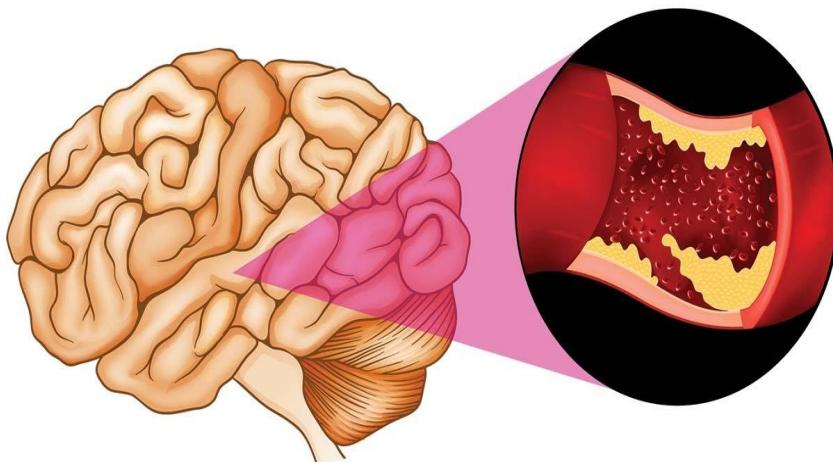
# INTRODUCTION



Theo Tổ chức Y tế Thế giới (WHO), đột quỵ là nguyên nhân gây tử vong đứng thứ 2 trên toàn cầu, chiếm khoảng 11% tổng số ca tử vong.

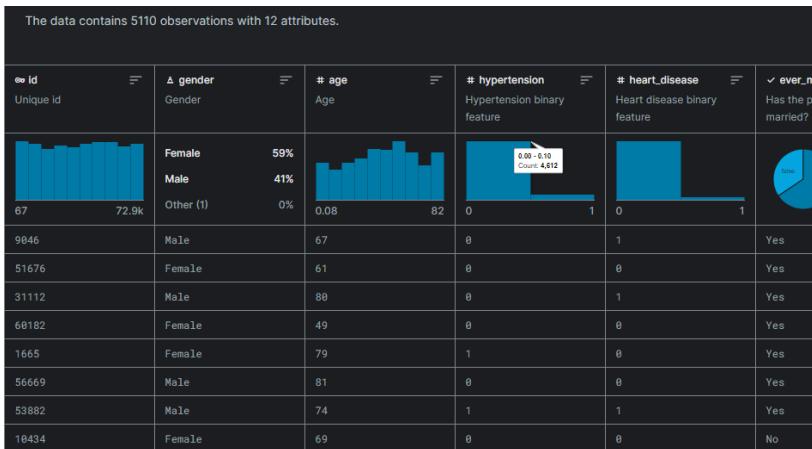
Hậu quả của đột quỵ có thể rất nghiêm trọng, ảnh hưởng đến sức khỏe, chất lượng cuộc sống và gánh nặng tài chính cho bản thân người bệnh và gia đình. Mức độ ảnh hưởng phụ thuộc vào nhiều yếu tố như vị trí, mức độ tổn thương não, thời gian điều trị và khả năng hồi phục của mỗi người.

## INTRODUCTION



Đột quỵ, còn gọi là tai biến mạch máu não, là tình trạng cấp tính nguy hiểm xảy ra khi nguồn cung cấp máu cho một phần não bị gián đoạn hoặc giảm đáng kể. Khi đó, não bộ bị thiếu oxy và dinh dưỡng, dẫn đến tổn thương hoặc chết tế bào não. Đột quỵ có thể gây ra các di chứng nghiêm trọng, thậm chí tử vong nếu không được điều trị kịp thời.

# INTRODUCTION



- Stroke Prediction Dataset là bộ dữ liệu được sử dụng để dự đoán liệu một bệnh nhân có khả năng bị đột quỵ hay không dựa trên các thông số đầu vào như giới tính, tuổi tác, các bệnh khác nhau và tình trạng hút thuốc. Mỗi hàng trong dữ liệu cung cấp thông tin liên quan về bệnh nhân.
- Bộ dữ liệu gồm 12 cột và 5110 dòng.
- Mục tiêu:
  - + Phân loại.
  - + Sử dụng kỹ thuật học máy để dự đoán.
  - + Tìm hiểu những nguyên nhân có thể dẫn đến đột quỵ.
  - + Kết luận và đưa ra những phương pháp hợp lý để phòng ngừa nguy cơ bị đột quỵ.

## 2. IMPORT, CHECK DATA

```
dataRaw.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
```

```
dataRaw.describe().T
```

	count	mean	std	min	25%	50%	75%	max
<b>id</b>	5110.0	36517.829354	21161.721625	67.00	17741.250	36932.000	54682.00	72940.00
<b>age</b>	5110.0	43.226614	22.612647	0.08	25.000	45.000	61.00	82.00
<b>hypertension</b>	5110.0	0.097456	0.296607	0.00	0.000	0.000	0.00	1.00
<b>heart_disease</b>	5110.0	0.054012	0.226063	0.00	0.000	0.000	0.00	1.00
<b>avg_glucose_level</b>	5110.0	106.147677	45.283560	55.12	77.245	91.885	114.09	271.74
<b>bmi</b>	4909.0	28.893237	7.854067	10.30	23.500	28.100	33.10	97.60
<b>stroke</b>	5110.0	0.048728	0.215320	0.00	0.000	0.000	0.00	1.00

```
[7] dataRaw.duplicated().sum()
```

0

### 3. CLEAN DATA

gender has ['Male' 'Female' 'Other'] categories.

hypertension has [0 1] categories.

heart\_disease has [1 0] categories.

ever\_married has ['Yes' 'No'] categories.

work\_type has ['Private' 'Self-employed' 'Govt\_job' 'children' 'Never\_worked'] categories.

Residence\_type has ['Urban' 'Rural'] categories.

smoking\_status has ['formerly smoked' 'never smoked' 'smokes' 'Unknown'] categories.

stroke has [1 0] categories.

```
numerical_features = []
categorical_features = []
for i in dataRaw.columns:
    if dataRaw[i].nunique() > 6:
        numerical_features.append(i)
    else:
        categorical_features.append(i)
for feats in categorical_features:
    print(f'{feats} has {dataRaw[feats].unique()} categories.\n')
```

### 3. CLEAN DATA

```
count_other = (dataRaw['gender'] == 'Other').sum()  
print("Số lượng hàng có giá trị 'Other' trong cột 'gender':", count_other)
```

Số lượng hàng có giá trị 'Other' trong cột 'gender': 1

```
[10] df = dataRaw[dataRaw['gender'] != 'Other']
```

Chỉ có 1 hàng mang giá trị "Other" ở cột "gender" nên có thể xóa giá trị này vì không có ảnh hưởng lớn tới dữ liệu.



```
# Thay thế giá trị "yes" bằng 1 và "no" bằng 0  
df['ever_married'] = df['ever_married'].replace({'Yes': 1, 'No': 0})
```

### 3. CLEAN DATA

```
for col in df.columns:  
    pct_missing = np.mean(df[col].isnull())  
    print('{} {}%'.format(col, round(pct_missing* 100)))
```

id 0%  
gender 0%  
age 0%  
hypertension 0%  
heart\_disease 0%  
ever\_married 0%  
work\_type 0%  
Residence\_type 0%  
avg\_glucose\_level 0%  
bmi 4%  
smoking\_status 0%  
stroke 0%

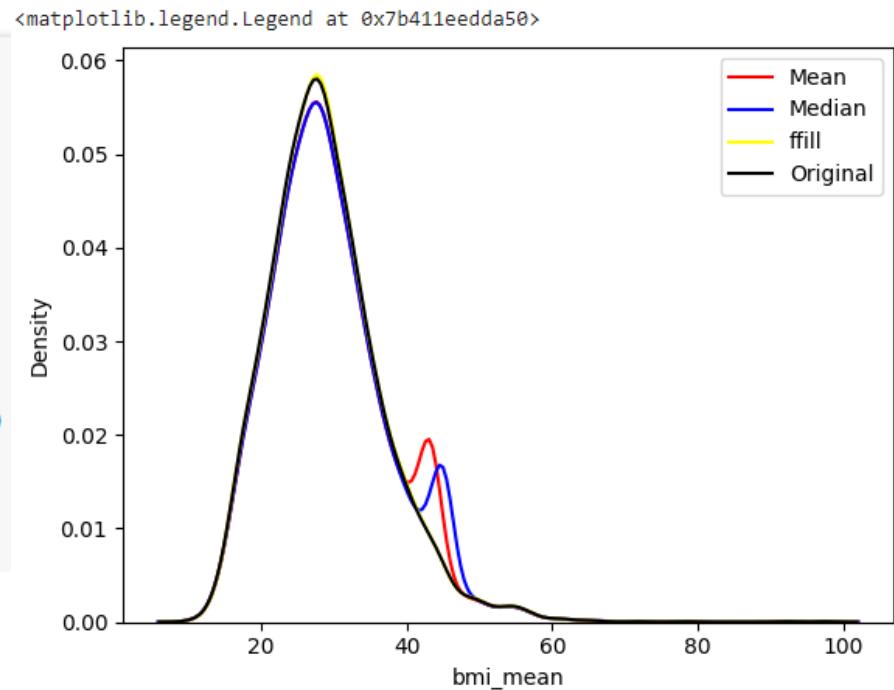
```
[44] # Đếm số hàng có cột "bmi" bị thiếu và có giá trị "1" trong cột "stroke"  
missing_bmi_stroke_1 = dataRaw[(dataRaw['bmi'].isnull() & (dataRaw['stroke'] == 1)].shape[0]  
  
print("Số hàng có cột 'bmi' bị thiếu và có giá trị '1' trong cột 'stroke':", missing_bmi_stroke_1)
```

Số hàng có cột 'bmi' bị thiếu và có giá trị '1' trong cột 'stroke': 40

### 3. CLEAN DATA

```
#Thử nghiệm thay thế các giá trị bị thiếu trong cột "bmi"
df_2 = df.copy()

bmi_mean = df_2.age.mean()
bmi_median= df_2.age.median()
df_2['bmi_mean'] = df_2.bmi.fillna(bmi_mean)
df_2['bmi_median'] = df_2.bmi.fillna(bmi_median)
df_2[ 'new_bmi'] = df_2[ 'bmi'].fillna (method="ffill")
sns.kdeplot(df_2[ 'bmi_mean'],color='red', label='Mean')
sns.kdeplot(df_2[ 'bmi_median'],color='blue', label='Median')
sns.kdeplot(df_2[ 'new_bmi'],color='yellow', label='ffill')
sns.kdeplot(df_2[ 'bmi'], color='black', label='Original')
plt.legend()
```



### 3. CLEAN DATA

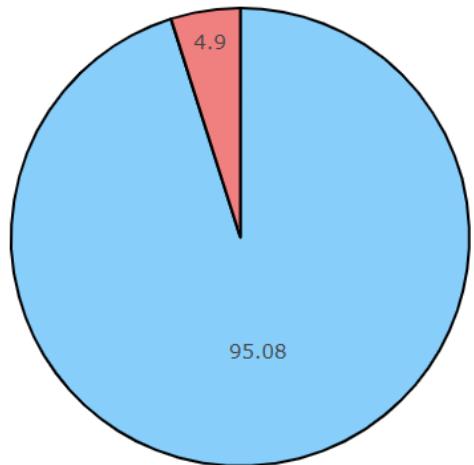
```
#Thay thế giá trị bị thiếu trong 'bmi' bằng phương pháp ffill  
df['bmi'] = df['bmi'].fillna(method='ffill')
```

id 0%  
gender 0%  
age 0%  
hypertension 0%  
heart\_disease 0%  
ever\_married 0%  
work\_type 0%  
Residence\_type 0%  
avg\_glucose\_level 0%  
bmi 0%  
smoking\_status 0%  
stroke 0%

```
#Drop cột 'id'  
df.drop("id", axis = 1, inplace=True)
```

## 4. EDA

Incidence of stroke

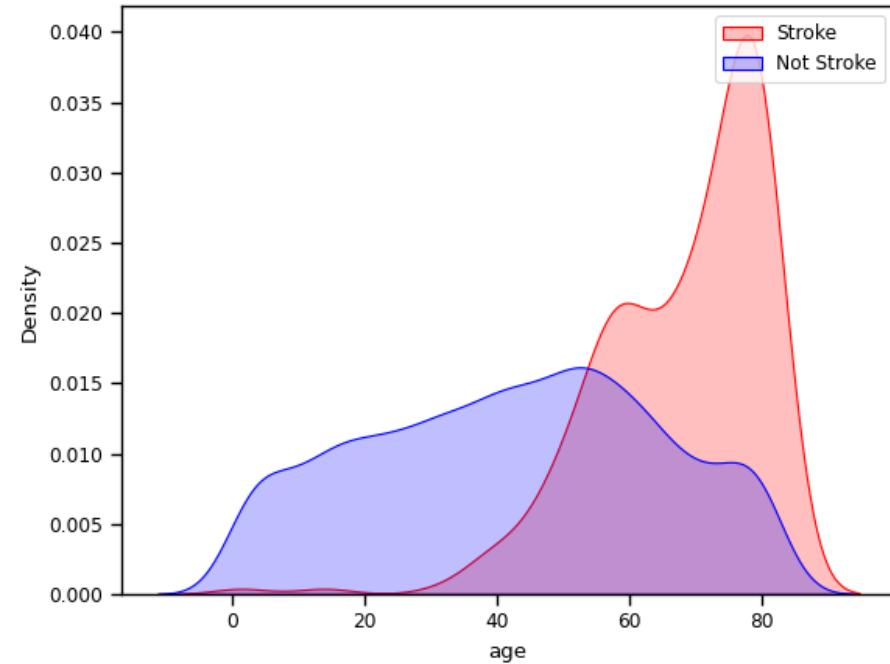


Trong tập dữ liệu này có khoảng 95.08% là người không mắc bệnh đột quỵ và 4.9% là người mắc bệnh đột quỵ.

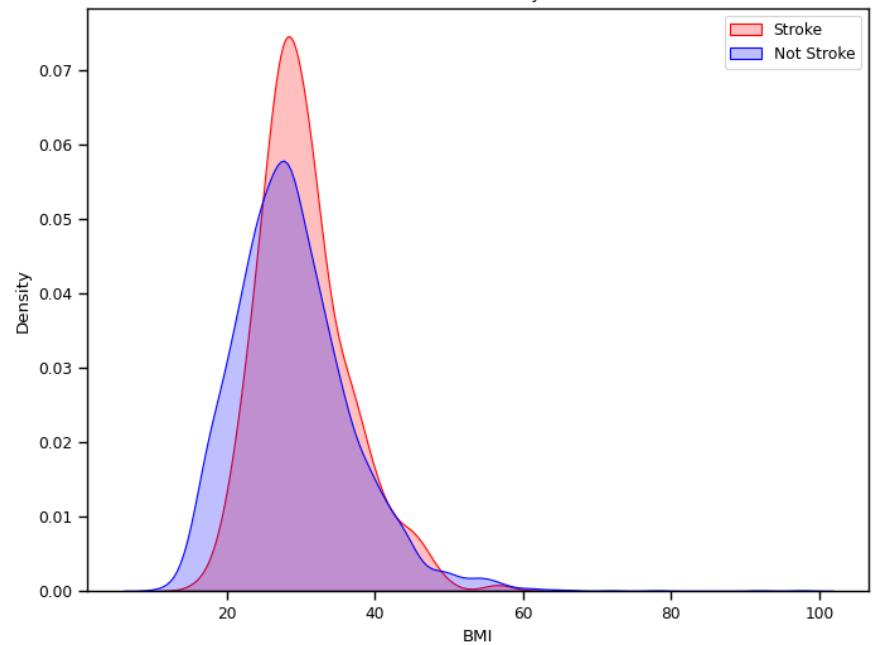
#### 4. EDA



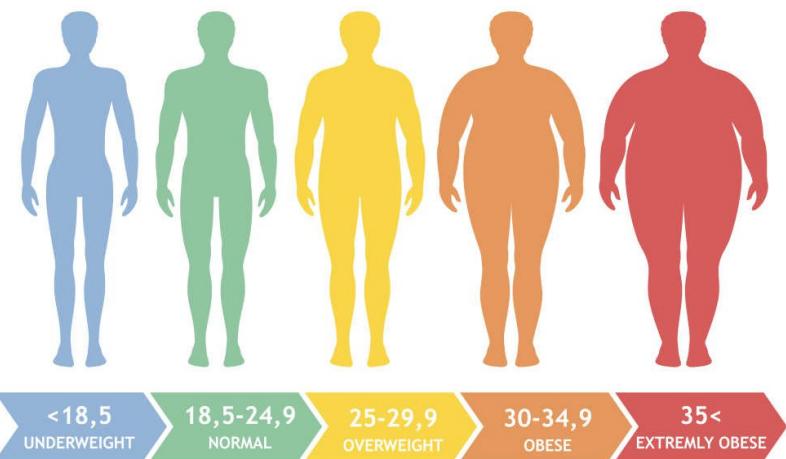
Distribution of age by stroke



Distribution of BMI by stroke



#### 4. EDA



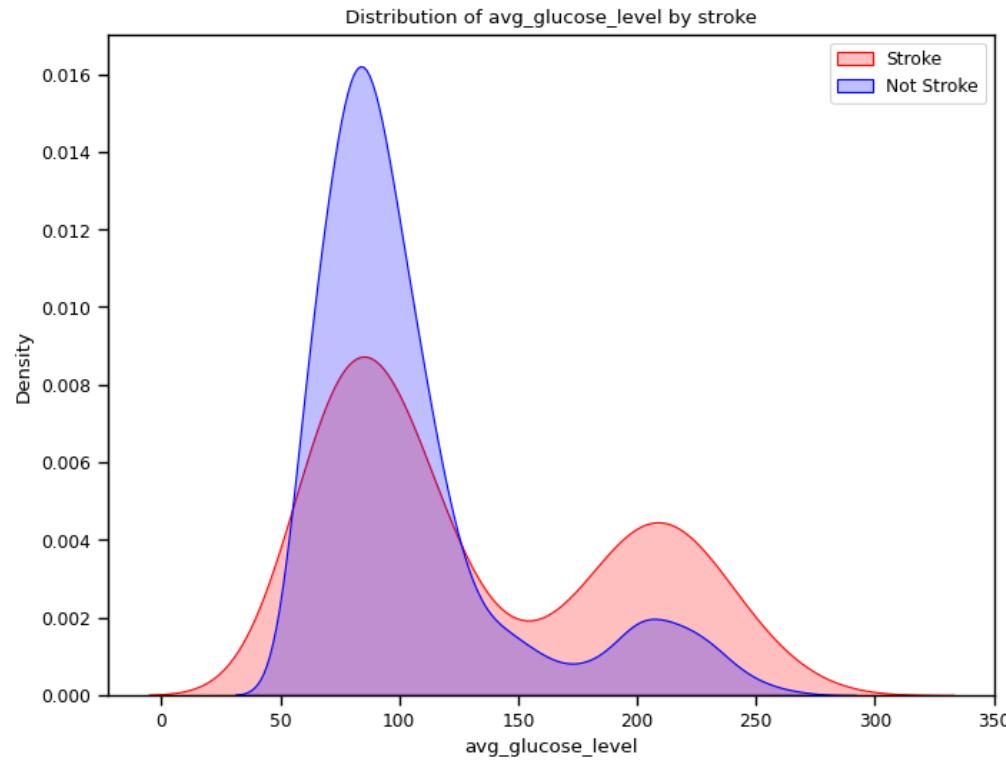
```
#Chia tinh trạng cơ thể theo BMI
df['bmi_categories'] = pd.cut(df['bmi'],
                                bins=[0, 18.4, 24.9, 29.9, np.inf],
                                labels=['Underweight', 'Normal weight',
                                        'Overweight', 'Obese'])
```

	count	stroke_count	stroke_percentage
--	-------	--------------	-------------------

### bmi\_categories

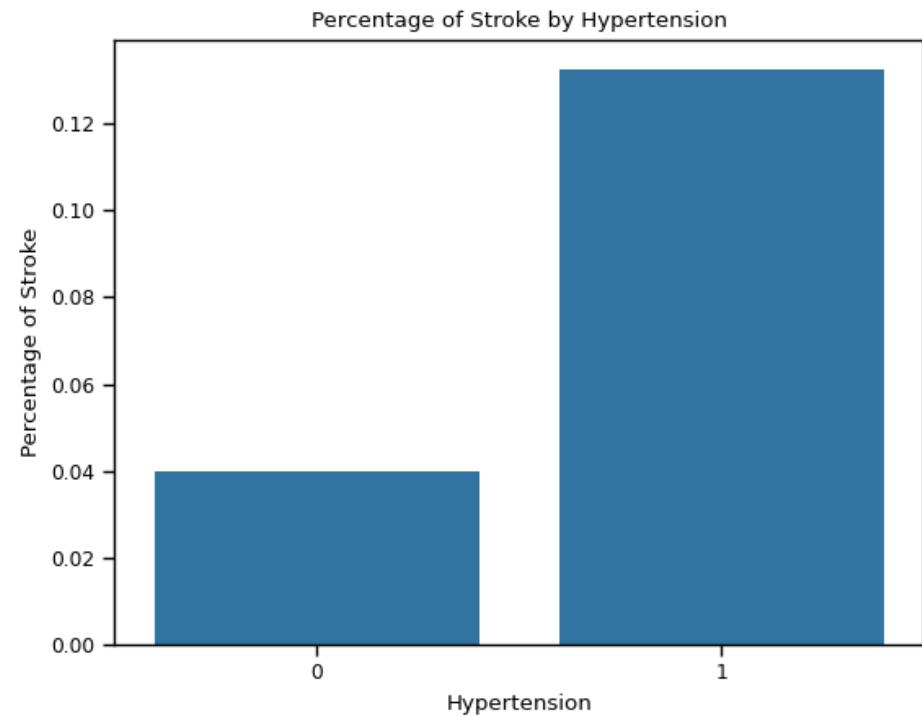
Obese	1998	113	5.655656
Overweight	1481	97	6.549629
Normal weight	1264	38	3.006329
Underweight	323	1	0.309598

## 4. EDA



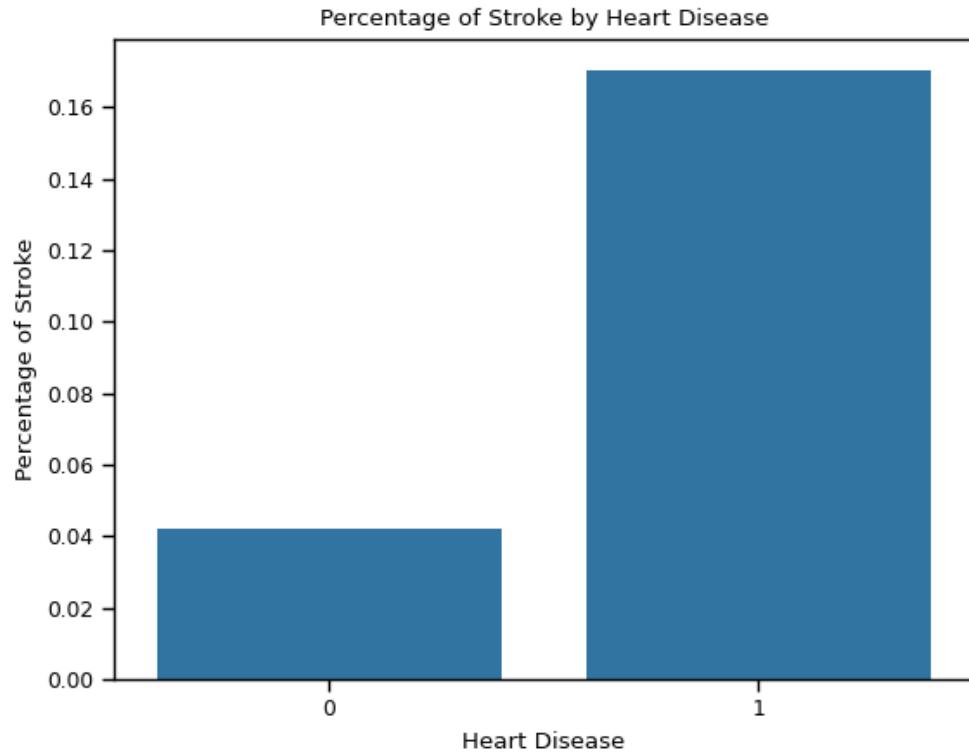
## 4. EDA

stroke	0	1
hypertension		
0	0.959939	0.040061
1	0.867470	0.132530

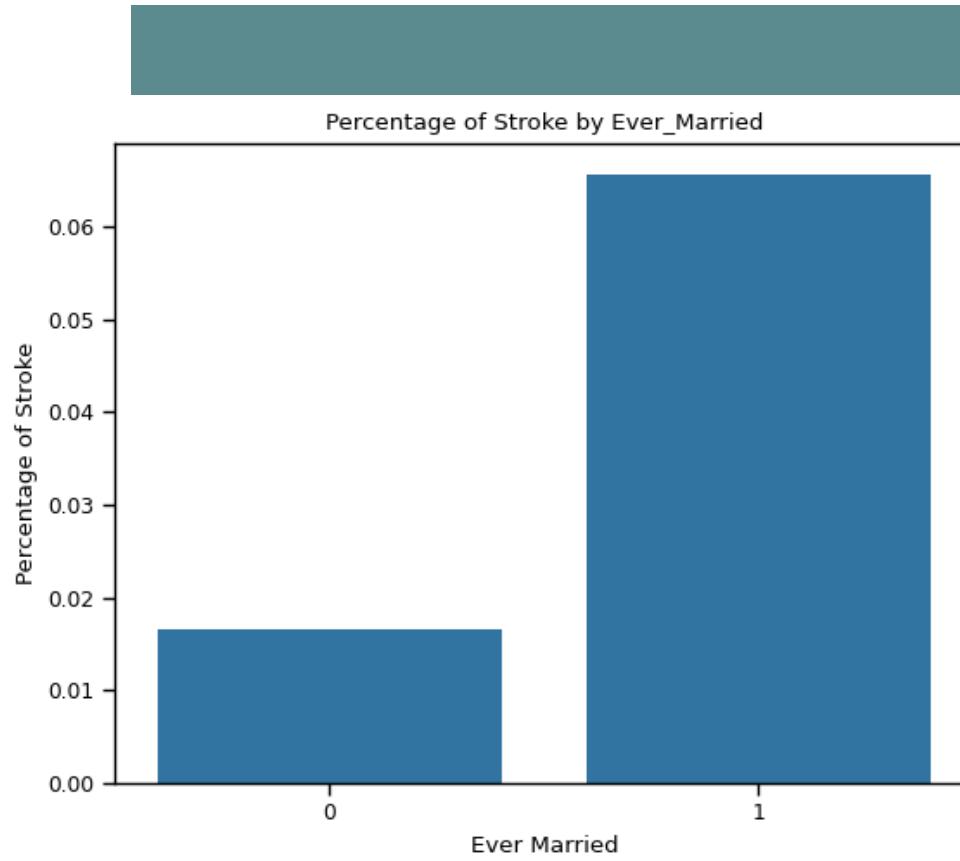


## 4. EDA

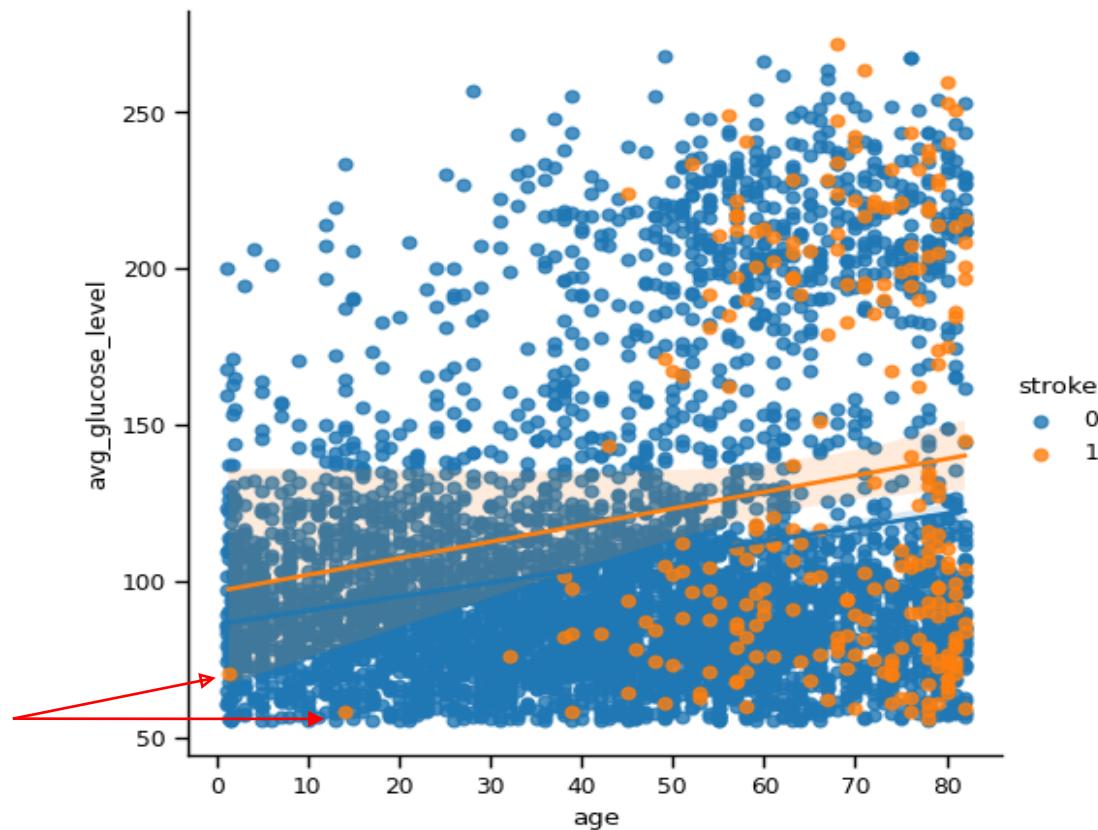
stroke	0	1
heart_disease		
0	0.957829	0.042171
1	0.829710	0.170290



## 4. EDA



#### 4. EDA



#### 4. EDA

```
[41] df.query('stroke==1 and age<=30')
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke	bmi_categories
162	Female	1.32	0	0	No	children	Urban	70.37	25.0	Unknown	1	Overweight
245	Female	14.00	0	0	No	children	Rural	57.93	30.9	Unknown	1	Overweight

#### 4. EDA

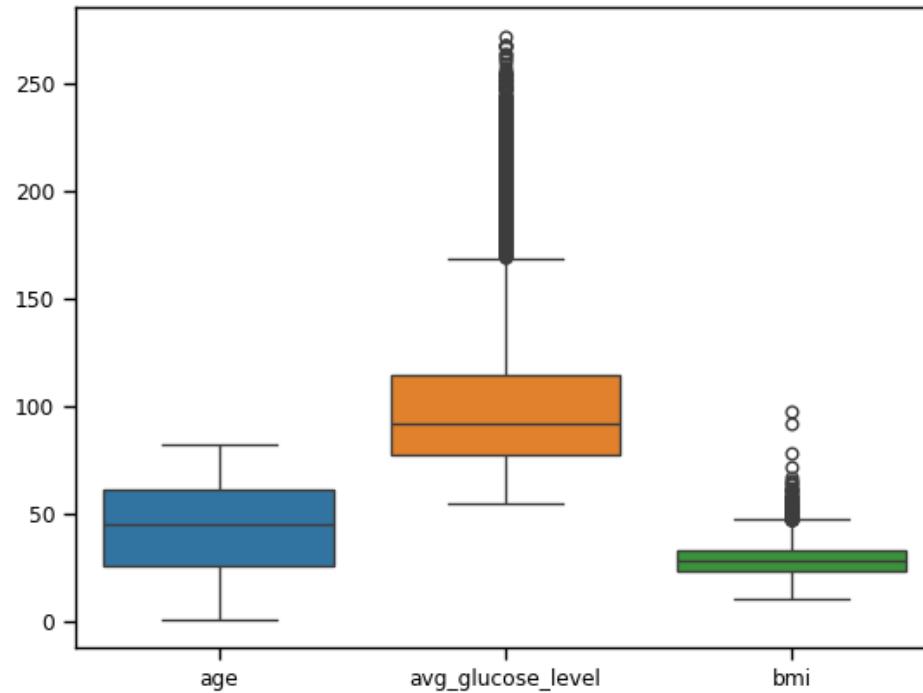
```
df[categorical_feature].describe()
```

	gender	ever_married	work_type	Residence_type	smoking_status
count	5066	5066	5066	5066	5066
unique	2	2	5	2	4
top	Female	Yes	Private	Urban	never smoked
freq	2979	3353	2924	2573	1892

## 4. EDA

```
sns.boxplot(data=df[['age', 'avg_glucose_level', 'bmi']])
```

```
<Axes: >
```



## 4. EDA

```
| threshold = 3
z_scores = (df[df['stroke'] == 1]['bmi'] - df[df['stroke'] == 1]['bmi'].mean()) / df[df['stroke'] == 1]['bmi'].std()

# Đếm số lượng giá trị outlier
outliers_count = (z_scores > threshold).sum()

print("Số lượng giá trị outlier trong cột 'bmi' với 'stroke' = 1 là:", outliers_count)
```

Số lượng giá trị outlier trong cột 'bmi' với 'stroke' = 1 là: 2

```
threshold = 3
z_scores = (df[df['stroke'] == 1]['avg_glucose_level'] - df[df['stroke'] == 1]['avg_glucose_level'].mean()) / df[df['stroke'] == 1]['avg_glucose_level'].std()

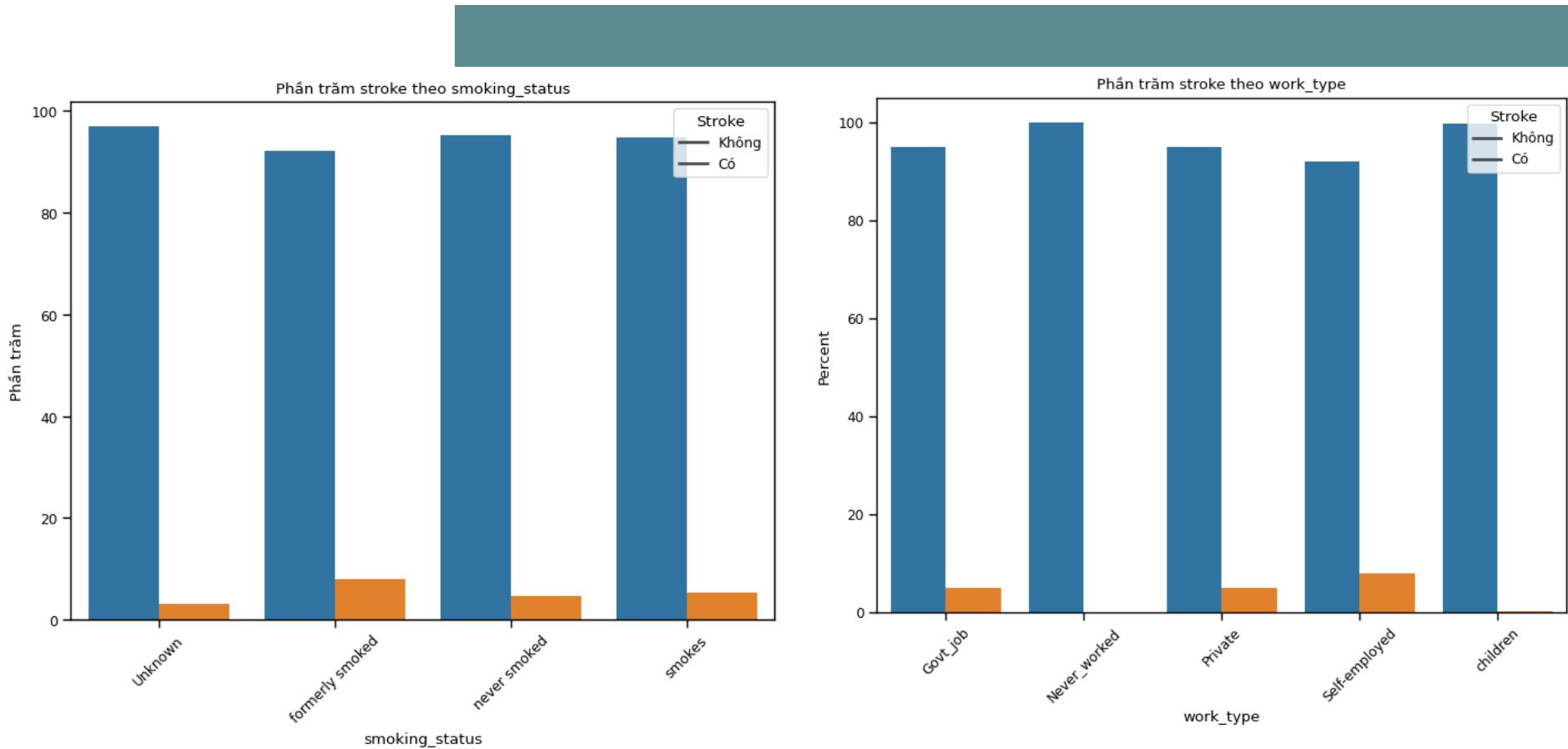
# Đếm số lượng giá trị outlier
outliers_count = (z_scores > threshold).sum()

print("Số lượng giá trị outlier trong cột 'avg_glucose_level' với 'stroke' = 1 là:", outliers_count)
```

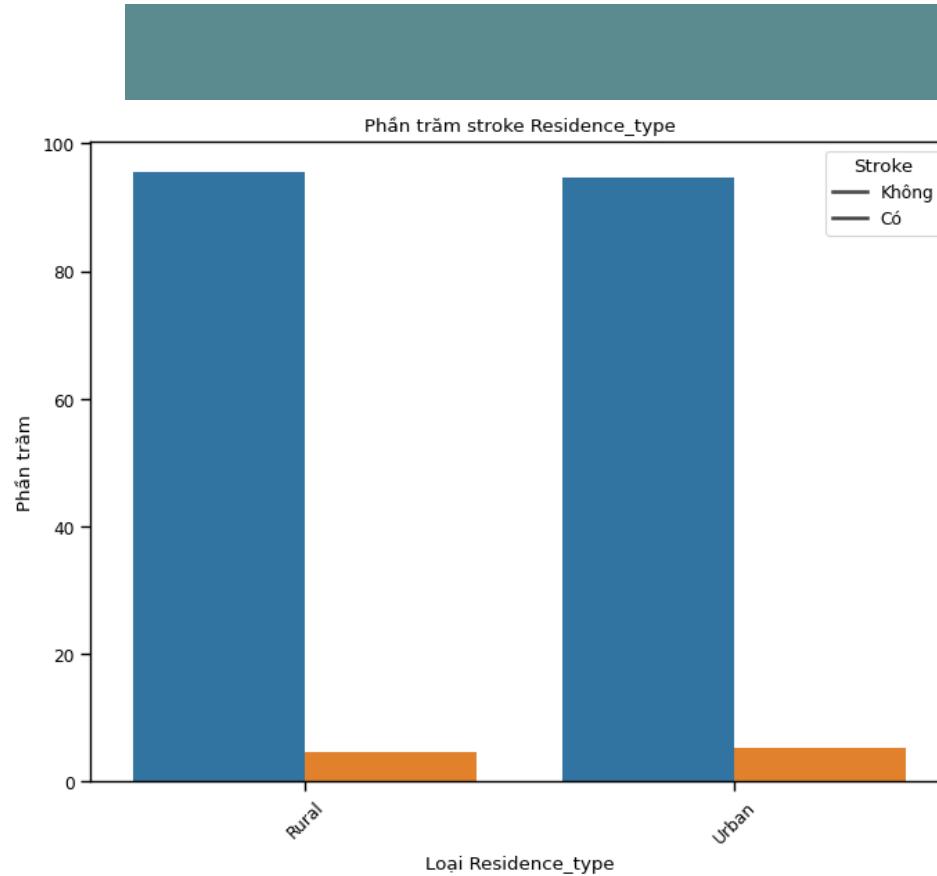
Số lượng giá trị outlier trong cột 'avg\_glucose\_level' với 'stroke' = 1 là: 0

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['age', 'avg_glucose_level', 'bmi']] = scaler.fit_transform(df[['age', 'avg_glucose_level', 'bmi']])
```

#### 4. EDA



#### 4. EDA



## 5. Data Preprocessing

```
# Định nghĩa bảng ánh xạ giữa giá trị ban đầu và giá trị mới
work_type_mapping = {
    'Private': 0,
    'Self-employed': 1,
    'Govt_job': 2,
    'children': 3,
    'Never_worked': 4
}
# Thực hiện thay đổi giá trị trong cột "Residence_type"
df['Residence_type'] = df['Residence_type'].replace(residence_type_mapping)

# Thực hiện thay đổi giá trị trong cột "work_type"
df['work_type'] = df['work_type'].replace(work_type_mapping)

] # Thay đổi giá trị "Male" thành 1 và "Female" thành 0 trong cột "gender"
df['gender'] = df['gender'].replace({'Male': 1, 'Female': 0})

# Định nghĩa bảng ánh xạ giữa giá trị ban đầu và giá trị mới
smoking_status_mapping = {
    'never smoked': 0,
    'formerly smoked': 2,
    'smokes': 1,
    'Unknown': 3
}
# Thực hiện thay đổi giá trị trong cột "smoking_status"
df['smoking_status'] = df['smoking_status'].replace(smoking_status_mapping)
```

## 5. Data Preprocessing

```
#Chia tập dữ liệu thành 2 bộ để train và test
X= df.drop(['stroke'], axis=1)

y= df['stroke']

X_train, X_test,y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

```
X_train.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
4039	7233	1	15.0	0	0	0	3	1	74.83	17.4	3
576	56179	1	29.0	0	0	0	0	0	207.58	22.8	1
4013	36388	1	44.0	1	0	1	0	1	91.28	26.5	0
4536	59405	0	68.0	1	0	1	0	0	150.74	40.3	3
1180	46643	0	62.0	0	0	1	0	1	82.57	36.0	2

## 6. Modeling

### Decision Tree Baseline Model

```
Accuracy: 0.8936725375081539
```

```
Confusion Matrix:
```

```
[[1362  82]
 [ 81   8]]
```

```
Classification Report:
```

	precision	recall	f1-score	support
0	0.94	0.94	0.94	1444
1	0.09	0.09	0.09	89
accuracy			0.89	1533
macro avg	0.52	0.52	0.52	1533
weighted avg	0.89	0.89	0.89	1533

## 6. Modeling

### Random Forest

Accuracy: 0.9419439008480104

Confusion Matrix:

```
[[1444    0]
 [ 89    0]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1444
1	0.00	0.00	0.00	89
accuracy			0.94	1533
macro avg	0.47	0.50	0.49	1533
weighted avg	0.89	0.94	0.91	1533

## 6. Modeling

### Support Vector Machines (SVM)

Accuracy: 0.9419439008480104

Confusion Matrix:

```
[[1444    0]
 [ 89    0]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1444
1	0.00	0.00	0.00	89
accuracy			0.94	1533
macro avg	0.47	0.50	0.49	1533
weighted avg	0.89	0.94	0.91	1533

## 6. Modeling

### Logistic Regression

Logistic Regression accuracy is : 0.9425962165688193

Classification Report:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1444
1	1.00	0.01	0.02	89
accuracy			0.94	1533
macro avg	0.97	0.51	0.50	1533
weighted avg	0.95	0.94	0.92	1533

## 6. Modeling

KNeighborsClassifier

KNN accuracy: 0.9419439008480104

Classification Report:

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1444
1	0.00	0.00	0.00	89
accuracy			0.94	1533
macro avg	0.47	0.50	0.49	1533
weighted avg	0.89	0.94	0.91	1533

## 7. Conclude

- Dựa trên các kết quả đánh giá đã cung cấp, không có một mô hình nào có thể được xác định là "tốt nhất" một cách tuyệt đối. Mỗi mô hình có ưu điểm và hạn chế riêng, và hiệu suất của chúng có thể phụ thuộc vào loại dữ liệu và yêu cầu cụ thể của vấn đề.
- Tuy nhiên, nếu chỉ dựa trên độ chính xác, mô hình Logistic Regression có vẻ như là một lựa chọn tốt nhất, với độ chính xác cao nhất trong số các mô hình được đề cập. Tuy nhiên, cần lưu ý rằng F1-score của lớp 1 của mô hình này vẫn thấp, vì vậy nó có thể không thực sự hiệu quả trong việc dự đoán các trường hợp của lớp 1.

## 7. Conclude

### **Phân phối dữ liệu trường hợp đột quy có cân bằng không?**

Không, phân phối dữ liệu trường hợp đột quy không cân bằng. Phần lớn dữ liệu không có tiền sử đột quy, trong khi chỉ một phần nhỏ có tiền sử đột quy. Điều này có thể nhìn thấy qua so sánh lượng dữ liệu có đột quy (lớp thiểu số) và không có đột quy (lớp đa số) trong tập dữ liệu.

### **Người bị tăng huyết áp có nguy cơ bị đột quy cao hơn người bình thường không?**

Dựa trên phân tích, người bị tăng huyết áp có nguy cơ bị đột quy cao hơn so với người bình thường. Điều này có thể thấy qua sự phân bố các đặc điểm tăng huyết áp trong dữ liệu trường hợp đột quy, trong đó tỷ lệ người bị tăng huyết áp cao hơn ở nhóm có tiền sử đột quy.

- Các đặc điểm như giới tính, tình trạng hôn nhân, nghề nghiệp và thói quen hút thuốc có sự đa dạng trong phân bố. Điều này cho thấy tầm quan trọng của các yếu tố này trong việc dự đoán nguy cơ đột quy.

## 7. Conclude

Có một số giải pháp giảm nguy cơ đột quy mà bạn có thể thực hiện. Dưới đây là một số giải pháp phổ biến:

- **Dinh dưỡng lành mạnh:** Nên ăn uống cân đối và lành mạnh, giảm tiêu thụ đồ ăn có nhiều chất béo bão hòa và cholesterol cao. Tăng cường tiêu thụ rau củ, hoa quả, ngũ cốc nguyên hạt và thực phẩm giàu omega-3 có thể giúp giảm nguy cơ đột quy.
- **Tập thể dục đều đặn:** Tập thể dục thường xuyên giúp cải thiện sức khỏe tim mạch và huyết áp, giảm cân, cải thiện sự linh hoạt và giảm nguy cơ đột quy.
- **Kiểm soát cân nặng:** Việc duy trì cân nặng ở mức lý tưởng thông qua chế độ dinh dưỡng và tập luyện có thể giảm nguy cơ đột quy.
- **Kiểm soát huyết áp:** Theo dõi và kiểm soát huyết áp là một yếu tố quan trọng trong việc giảm nguy cơ đột quy. Uống thuốc theo chỉ định của bác sĩ nếu cần thiết.
- **Hạn chế tiêu thụ rượu và thuốc lá:** Hạn chế hoặc tránh tiêu thụ rượu và hút thuốc lá có thể giúp giảm nguy cơ đột quy.

## 7. Conclude

- **Kiểm tra và điều trị các bệnh lý liên quan:** Điều trị các bệnh như tiểu đường, cao huyết áp và bệnh tim mạch có thể giúp giảm nguy cơ đột quỵ.
- **Thực hiện kiểm tra sức khỏe định kỳ:** Định kỳ kiểm tra sức khỏe để theo dõi các yếu tố nguy cơ và xác định các vấn đề sức khỏe sớm có thể giúp phòng ngừa đột quỵ.
- **Giảm căng thẳng:** Căng thẳng và lo lắng có thể gây ra các vấn đề sức khỏe, bao gồm đột quỵ. Việc thực hiện các kỹ thuật giảm căng thẳng như thiền, yoga và tập thể dục nhẹ có thể giúp giảm nguy cơ.
- **Phòng ngừa môi trường:** Tránh tiếp xúc với các yếu tố môi trường gây hại như ô nhiễm không khí, hóa chất độc hại có thể giúp giảm nguy cơ đột quỵ.

A close-up photograph of a medical professional, likely a doctor or pharmacist, wearing a white lab coat and a stethoscope around their neck. They are holding a clear plastic prescription bottle with both hands. The bottle is filled with various colored pills, including red, blue, and green ones. The doctor's hands are visible, with their fingers gripping the sides of the bottle. The background is plain and light-colored.

THANK YOU  
FOR LISTENING