Title of the Project:-Brain-Stroke Prediction

Objective:-The aim of this project is to predict whether the patient is likely to get Stroke or not based on the input features in given dataset

Problem Statement :According to the World Health Organization (WHO),Brain stroke is the world's second biggest cause of death causing nearly 11% of all deaths. Despite significant advances in healthcare, Brain stroke remains to be a serious public health issue. This highlights the critical need for improved prevention, early identification, and effective precautions should be taken to minimize the world impact of Brain stroke.

Importing Libraries

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
data=pd.read_csv('healthcare-dataset-stroke-data.csv')
data.head()
```

Out[6]:

	id	gender	age	hypertension	heart_disease	Marriage_Status	work_type	Resid
0	9046	Male	67.0	0	1	Yes	Private	
1	51676	Female	61.0	0	0	Yes	Self-employed	
2	31112	Male	80.0	0	1	Yes	Private	
3	60182	Female	49.0	0	0	Yes	Private	
4	1665	Female	79.0	1	0	Yes	Self-employed	

Exploratory Data Analysis

In [8]:
```
data.shape #Rows & Columns
```

Out[8]: (5110, 12)

In [9]:
```python
data.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   Marriage_Status    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
```

In [10]:
```python
dataframe=data.drop(['id'],axis=1) #drop id column
```

In [11]:
```python
dataframe.isnull().sum()
```

Out[11]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
Marriage_Status        0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```

In [12]:
```python
dataframe.describe()
```

Out[12]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	s
count	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.0(
mean	43.226614	0.097652	0.054012	106.147677	28.893237	0.04
std	22.612647	0.296872	0.226063	45.283560	7.854067	0.2
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.0(
25%	25.000000	0.000000	0.000000	77.245000	23.500000	0.0(
50%	45.000000	0.000000	0.000000	91.885000	28.100000	0.0(
75%	61.000000	0.000000	0.000000	114.090000	33.100000	0.0(
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.0(

Handling Null Values

In [14]:
```
dataframe['bmi'].isna().sum() #before handling
```
Out[14]:
```
201
```
In [15]:
```
dataframe['bmi'].fillna(dataframe['bmi'].median(),inplace=True)
```
In [16]:
```
dataframe['bmi'].isna().sum() #after handling
```
Out[16]:
```
0
```
In [17]:
```
cols=dataframe.columns
cols
```
Out[17]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'Marriage_Status',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

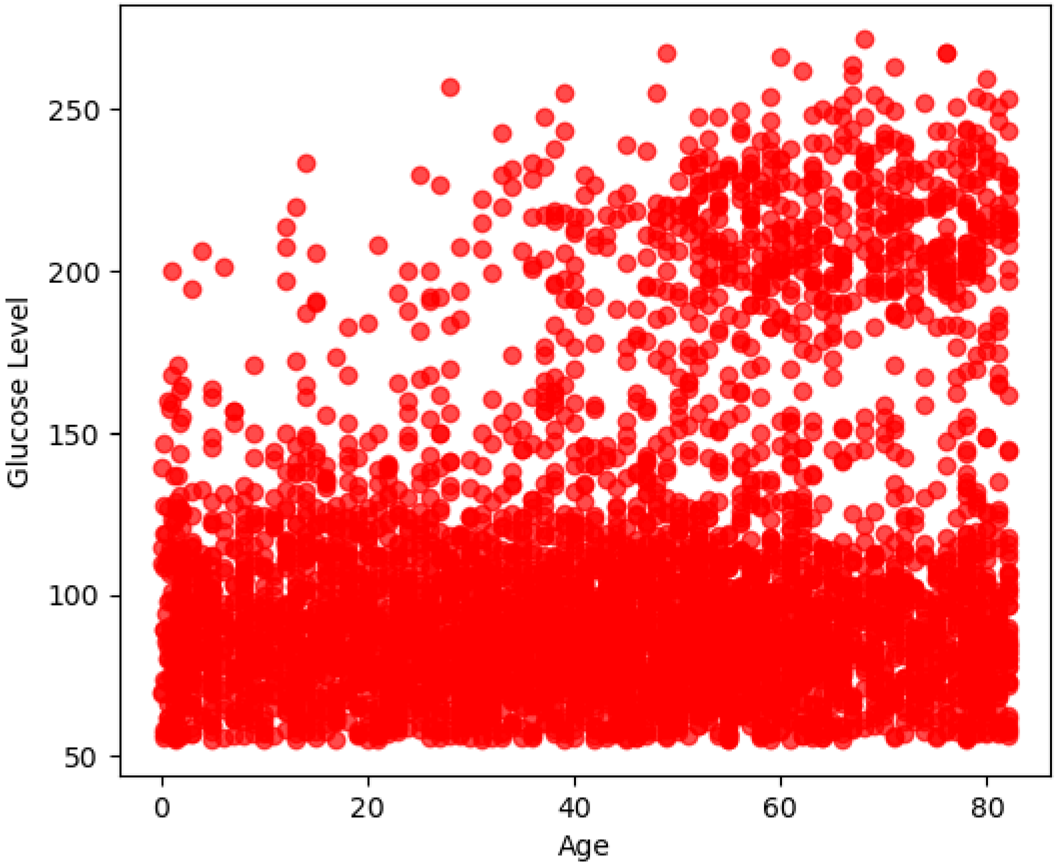
Numerical Columns

In [19]:
```
num_dataframe=dataframe.select_dtypes(exclude=object)
num_dataframe.head()
```
Out[19]:

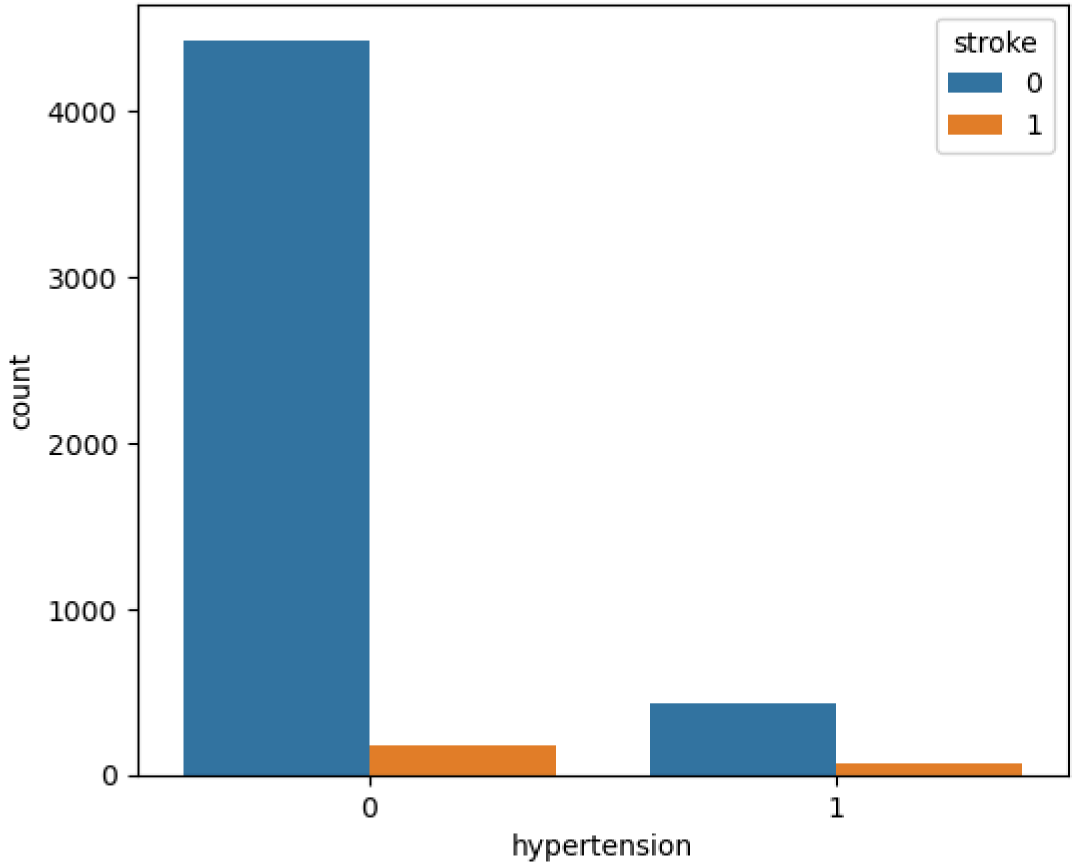
	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
0	67.0	0	1	228.69	36.6	1
1	61.0	0	0	202.21	28.1	1
2	80.0	0	1	105.92	32.5	1
3	49.0	0	0	171.23	34.4	1
4	79.0	1	0	174.12	24.0	1

Data Visualizations

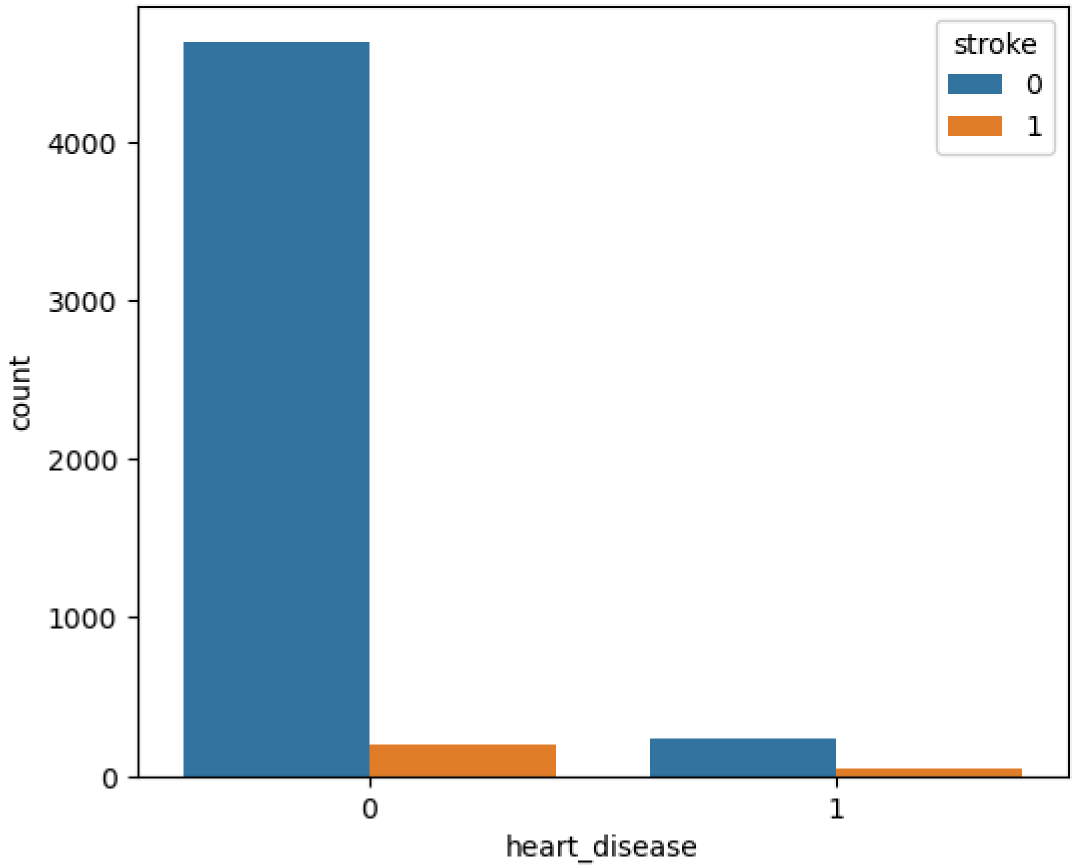
In [21]:
```
plt.figure(figsize=(6, 5))
plt.scatter(num_dataframe['age'], num_dataframe['avg_glucose_level'], alpha=0.7,
plt.xlabel('Age')
plt.ylabel('Glucose Level')
plt.show()
```



In [22]:
```
# Create the countplot
plt.figure(figsize=(6,5))
sns.countplot(x='hypertension', hue='stroke', data=num_dataframe)
plt.show()
```

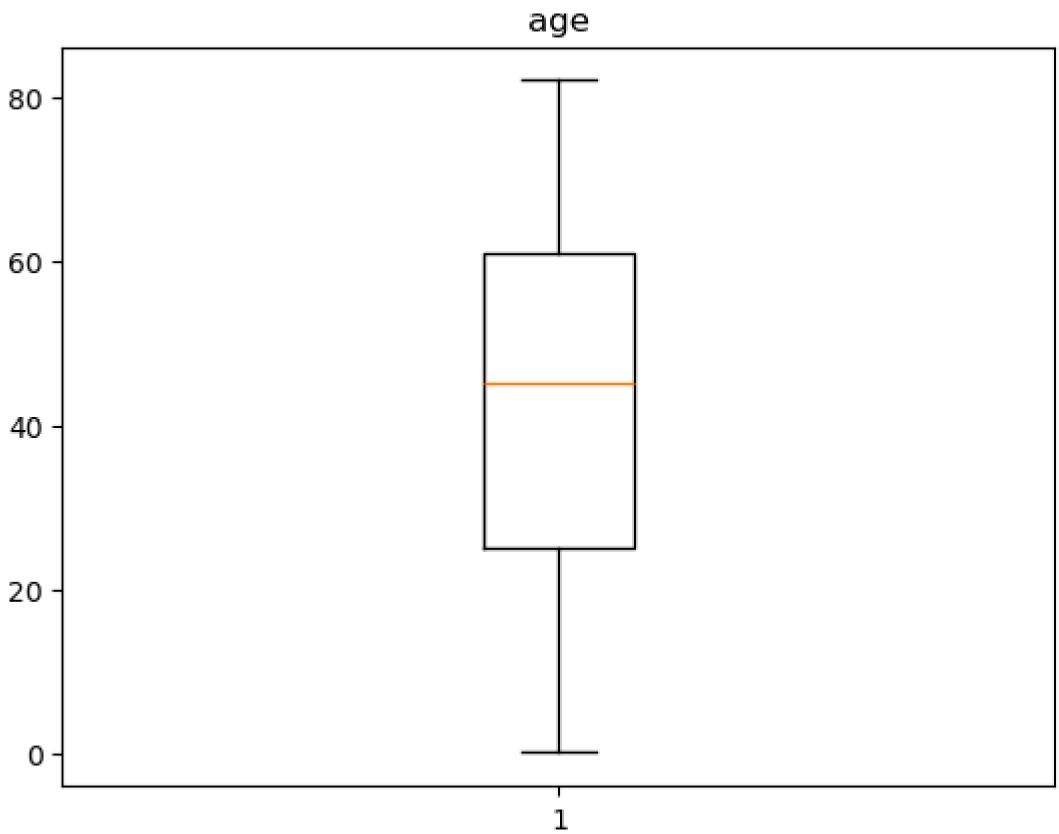


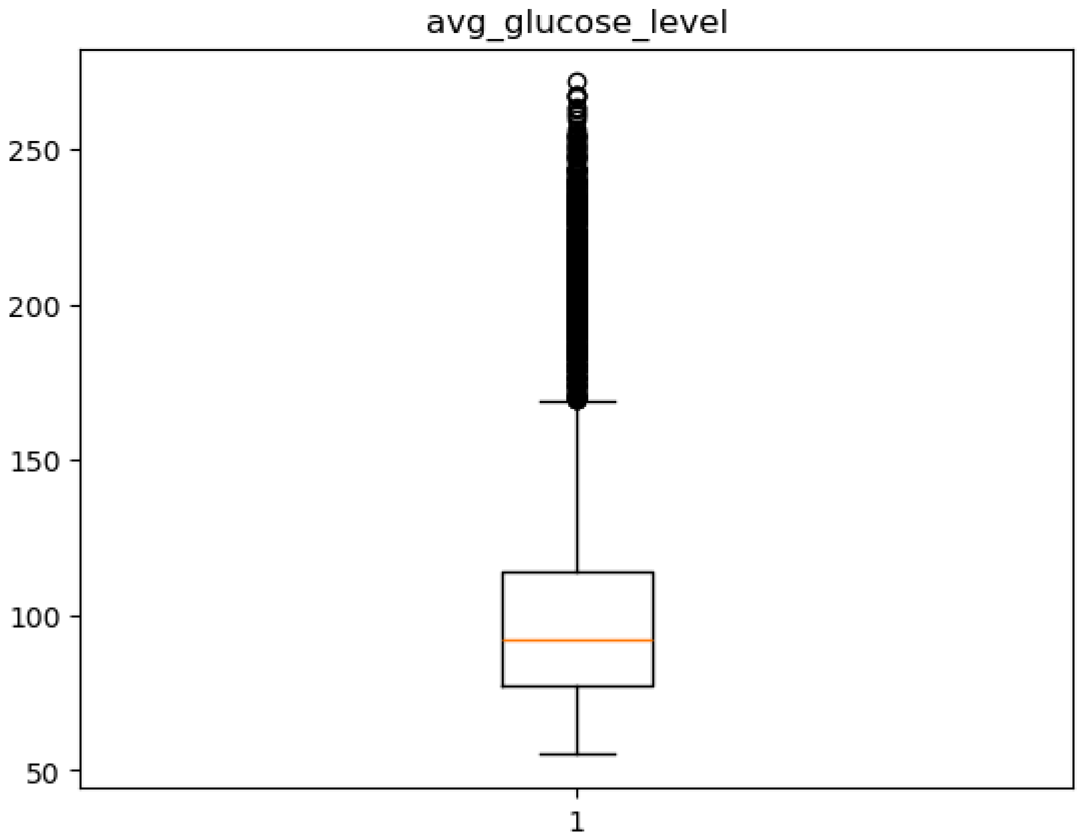
In [23]:
```
plt.figure(figsize=(6,5))
sns.countplot(x='heart_disease', hue='stroke', data=num_dataframe)
plt.show()
```

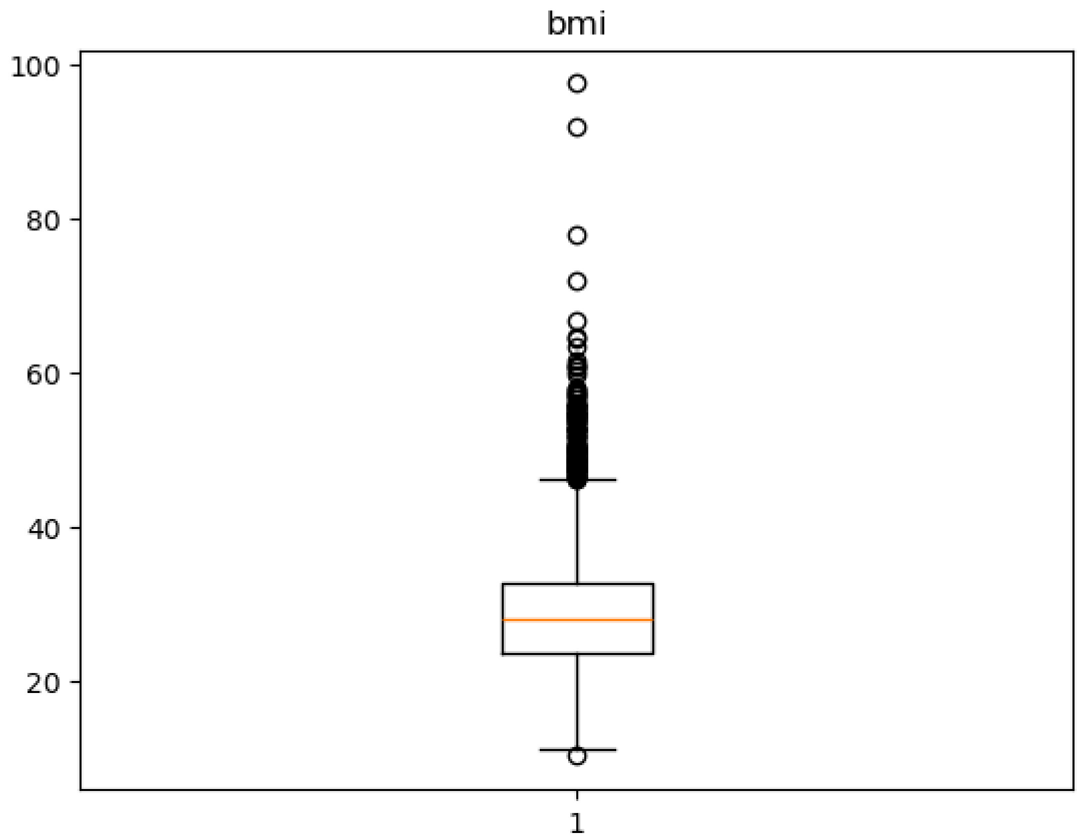


Outliers Detection

In [25]:
```
num_outliers=['age','avg_glucose_level','bmi']
for column in num_outliers:
    plt.boxplot(x=num_dataframe[column])
    plt.title(column)
    plt.show()
```







In [26]:
```
for column in num_outliers:

    print(column,":")
    q1=num_dataframe[column].quantile(0.25)
    q3=num_dataframe[column].quantile(0.75)
    iqr=q3-q1
    upper_tail=q3+1.5*iqr
    lower_tail=q1-1.5*iqr
```

```
    print("q1 --->",q1)
    print("q3 --->",q3)
    print("iqr --->",iqr)
    print("upper tail --->",upper_tail)
    print("lower_tail --->",lower_tail)
    print("-"*50)
```

```
age :
q1 ---> 25.0
q3 ---> 61.0
iqr ---> 36.0
upper tail ---> 115.0
lower_tail ---> -29.0
--------------------------------------------------
avg_glucose_level :
q1 ---> 77.245
q3 ---> 114.09
iqr ---> 36.845
upper tail ---> 169.35750000000002
lower_tail ---> 21.977500000000006
--------------------------------------------------
bmi :
q1 ---> 23.8
q3 ---> 32.8
iqr ---> 8.999999999999996
upper tail ---> 46.29999999999999
lower_tail ---> 10.300000000000006
--------------------------------------------------
```

In [27]:

```
 ### Here we are not handling outliers because for columns avg_glucose_level and
### to detect if the person is having stroke or not.Otherwise model will predict
```

Checking skewness

In [29]:

```
from scipy.stats import skew
```

In [30]:

```
num_skew=['age','avg_glucose_level','bmi']
```

In [31]:

```
for num_column in num_skew:
        print(num_column,"===>",skew(num_dataframe[num_column]))
```

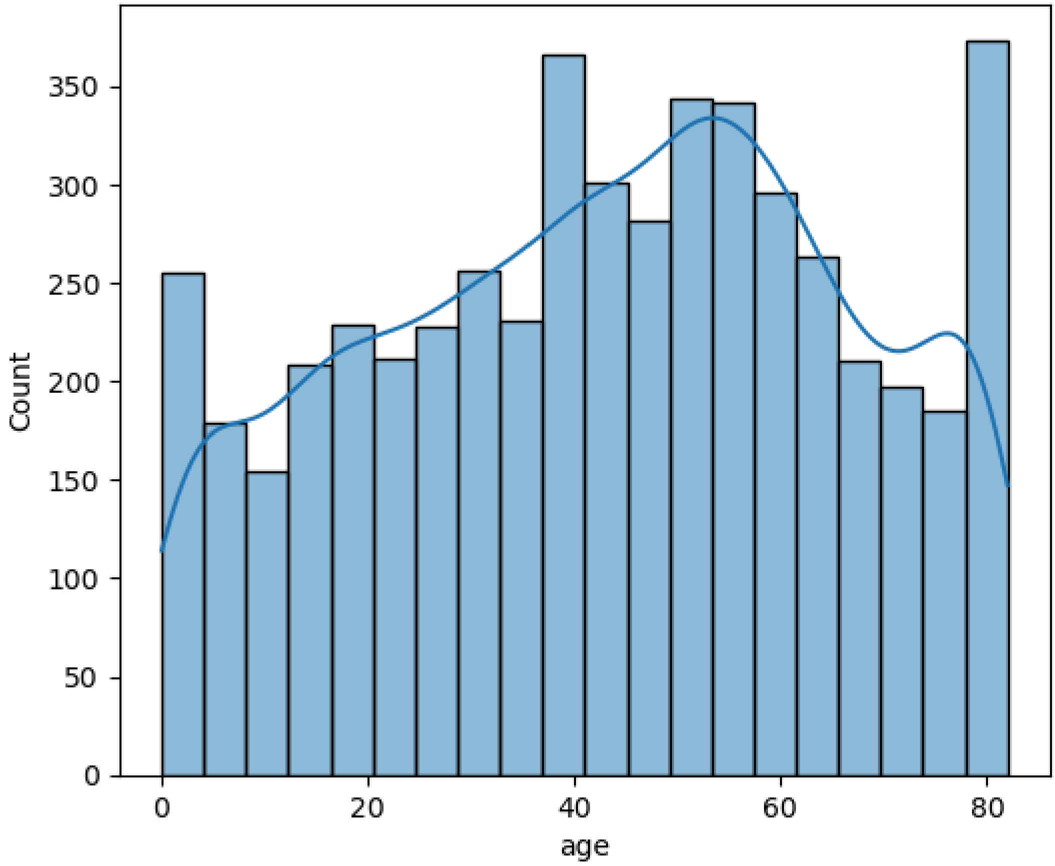
```
age ===> -0.1370190866396024
avg_glucose_level ===> 1.571822297397199
bmi ===> 1.0878677778570507
```

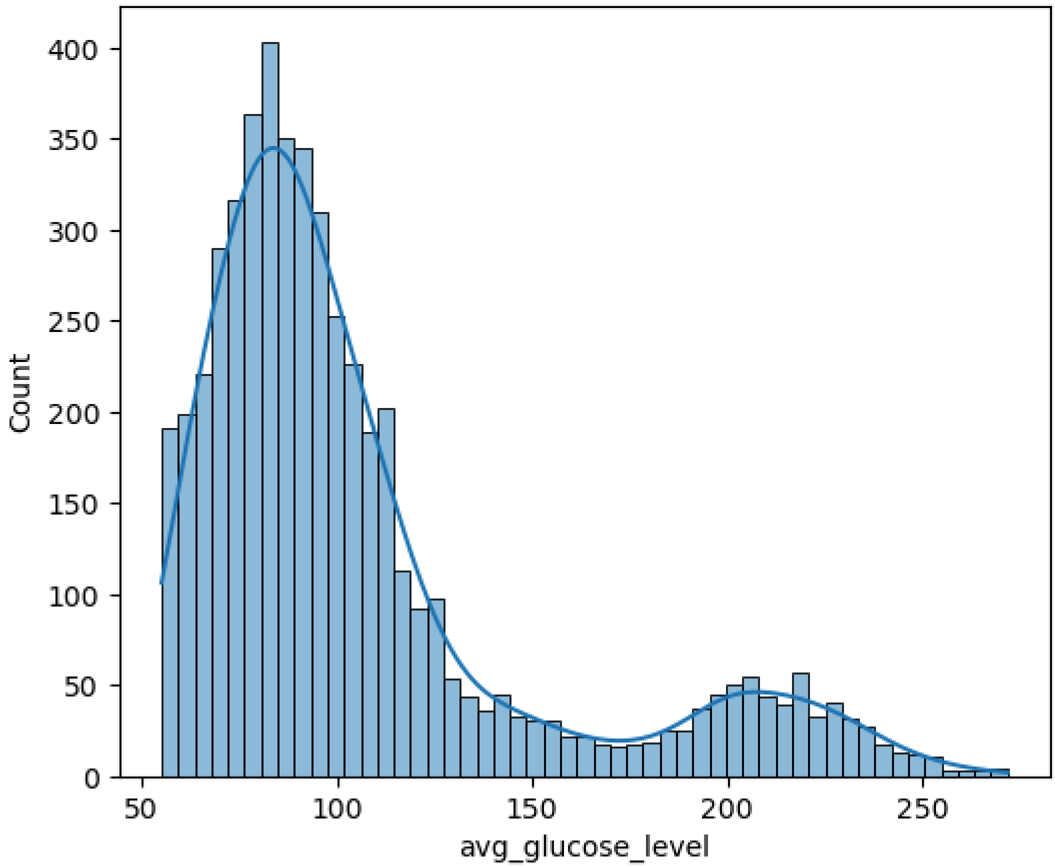
In [32]:

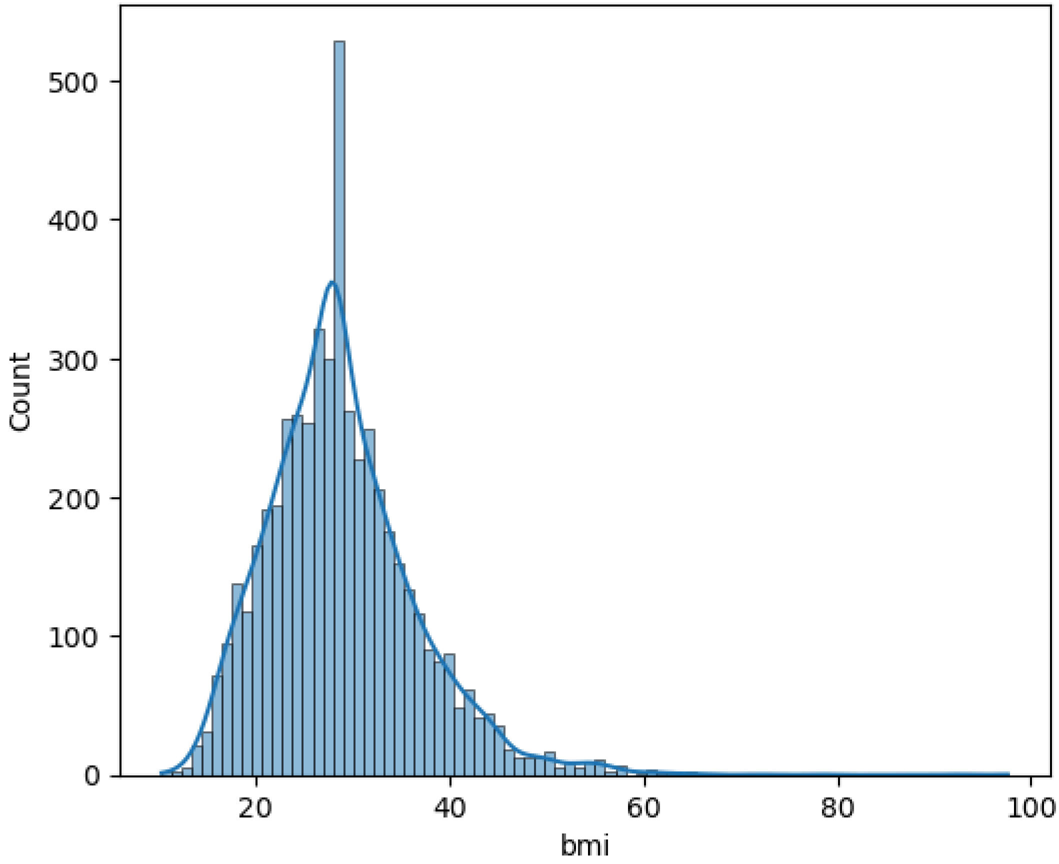
```
# graph of skewness visualization
```

In [33]:

```
for num_column in num_skew:
    plt.figure(figsize=(6,5))
    sns.histplot(num_dataframe[num_column], kde=True)
    plt.show()
```


Count
0
50
100
150
200
250
300
350
0
20
40
60
80
age


Count
0
50
100
150
200
250
300
350
400
50
100
150
200
250
avg_glucose_level



Handling Skewness

In [35]:
```
for num_column in num_dataframe:
    num_dataframe[num_column] = np.sqrt(num_dataframe[num_column])
```

In [36]:
```
for num_column in num_skew:
        print(num_column,"===>",skew(num_dataframe[num_column]))
```

```
age ===> -0.7859816333642492
avg_glucose_level ===> 1.2423570467212877
bmi ===> 0.4888921536853998
```

Object Columns

In [38]:
```
object_dataframe = dataframe.select_dtypes(include=object)
object_dataframe.head()
```

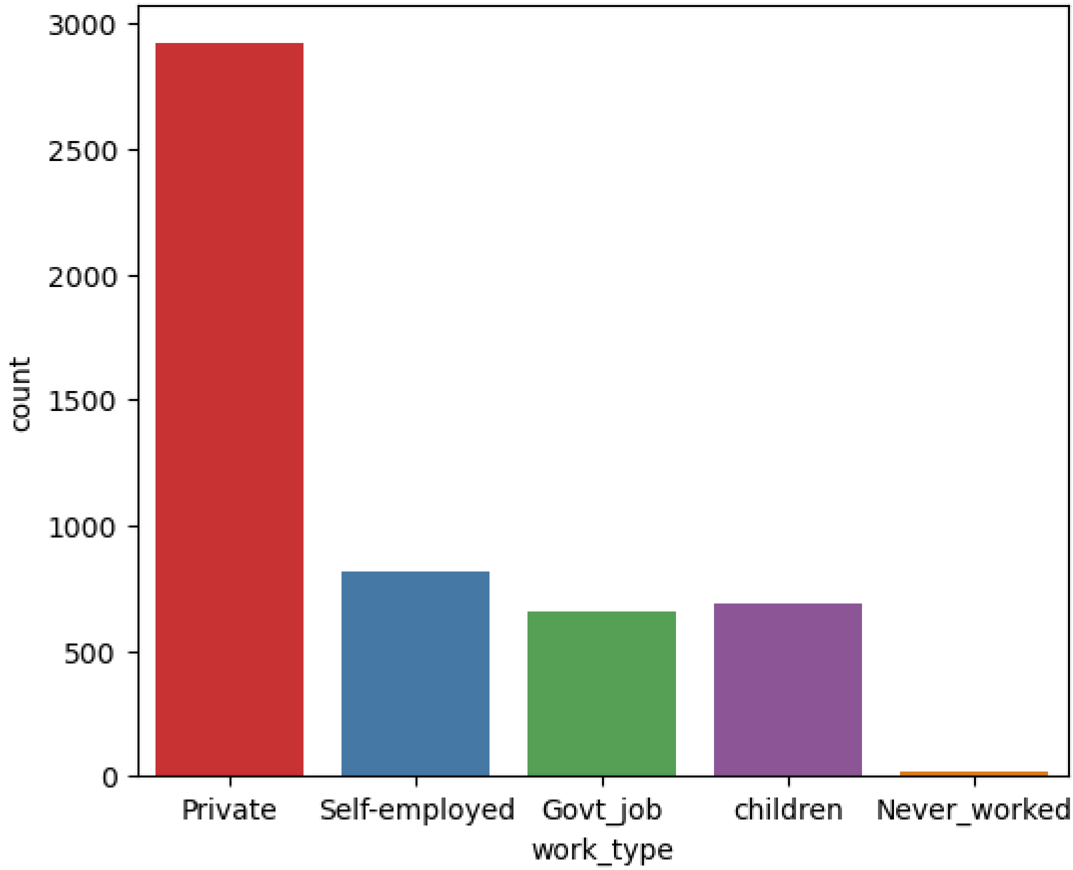
Out[38]:

	gender	Marriage_Status	work_type	Residence_type	smoking_status
0	Male	Yes	Private	Urban	formerly smoked
1	Female	Yes	Self-employed	Rural	never smoked
2	Male	Yes	Private	Rural	never smoked
3	Female	Yes	Private	Urban	smokes
4	Female	Yes	Self-employed	Rural	never smoked

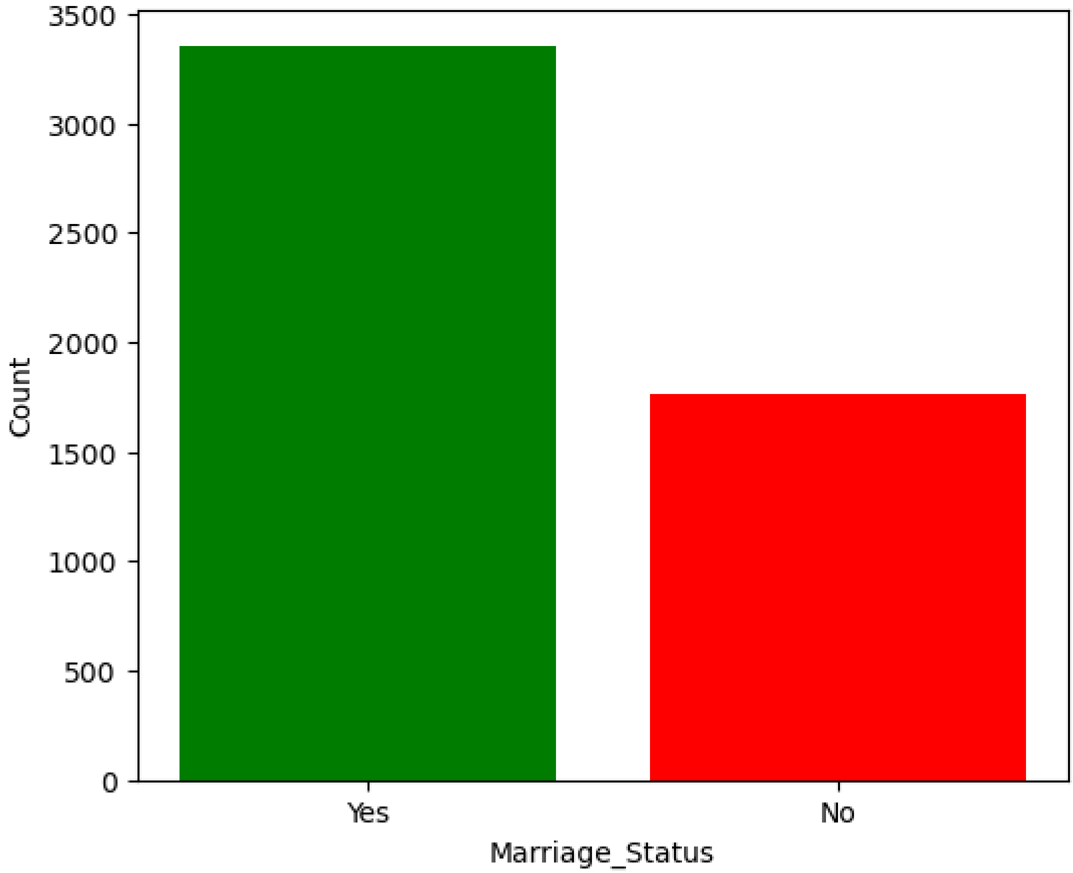
In [39]:
```
for column in object_dataframe:
    print(object_dataframe[column].value_counts())
    print("-"*50)
```

```
gender
Female    2994
Male      2116
Name: count, dtype: int64
--------------------------------------------------
Marriage_Status
Yes    3353
No     1757
Name: count, dtype: int64
--------------------------------------------------
work_type
Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: count, dtype: int64
--------------------------------------------------
Residence_type
Urban    2596
Rural    2514
Name: count, dtype: int64
--------------------------------------------------
smoking_status
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: count, dtype: int64
--------------------------------------------------
```

In [40]:
```
plt.figure(figsize=(6,5))
sns.countplot(x='work_type',data=data,palette = "Set1")
plt.show()
```



In [41]:
```
married_status_count= data['Marriage_Status'].value_counts()
plt.figure(figsize=(6,5))
plt.bar(x=married_status_count.index, height = married_status_count.values,color
plt.xlabel("Marriage_Status")
plt.ylabel("Count")
plt.show()
```



Data Encoding

In [43]:
```
#Label encoding
```

In [44]:
```
from sklearn.preprocessing import LabelEncoder
```

In [45]:
```
label_encoder=LabelEncoder()
```

In [46]:
```
Label_Encoder_columns=['gender','Marriage_Status','Residence_type']
```

In [47]:
```
for column_le in Label_Encoder_columns:
    object_dataframe[column_le] = label_encoder.fit_transform(object_dataframe[c
```

In [48]:
```
object_dataframe.head()
```

Out[48]:

	gender	Marriage_Status	work_type	Residence_type	smoking_status
0	1	1	Private	1	formerly smoked
1	0	1	Self-employed	0	never smoked
2	1	1	Private	0	never smoked
3	0	1	Private	1	smokes
4	0	1	Self-employed	0	never smoked

In [49]:
```
#one-hot Encoding(get dummies)
```

In [50]:
```
object_dataframe= pd.get_dummies(object_dataframe, columns=['work_type','smoking
```

In [51]:
```
object_dataframe.head()
```

Out[51]:

	gender	Marriage_Status	Residence_type	work_type_Govt_job	work_type_Never_work
0	1	1	1	0	
1	0	1	0	0	
2	1	1	0	0	
3	0	1	1	0	
4	0	1	0	0	

In [52]:
```
object_dataframe.columns
```

Out[52]:
```
Index(['gender', 'Marriage_Status', 'Residence_type', 'work_type_Govt_job',
       'work_type_Never_worked', 'work_type_Private',
       'work_type_Self-employed', 'work_type_children',
       'smoking_status_Unknown', 'smoking_status_formerly smoked',
       'smoking_status_never smoked', 'smoking_status_smokes'],
      dtype='object')
```

In [53]:
```
concat_dataframe = pd.concat([object_dataframe,num_dataframe], axis=1)
concat_dataframe.head()
```

Out[53]:

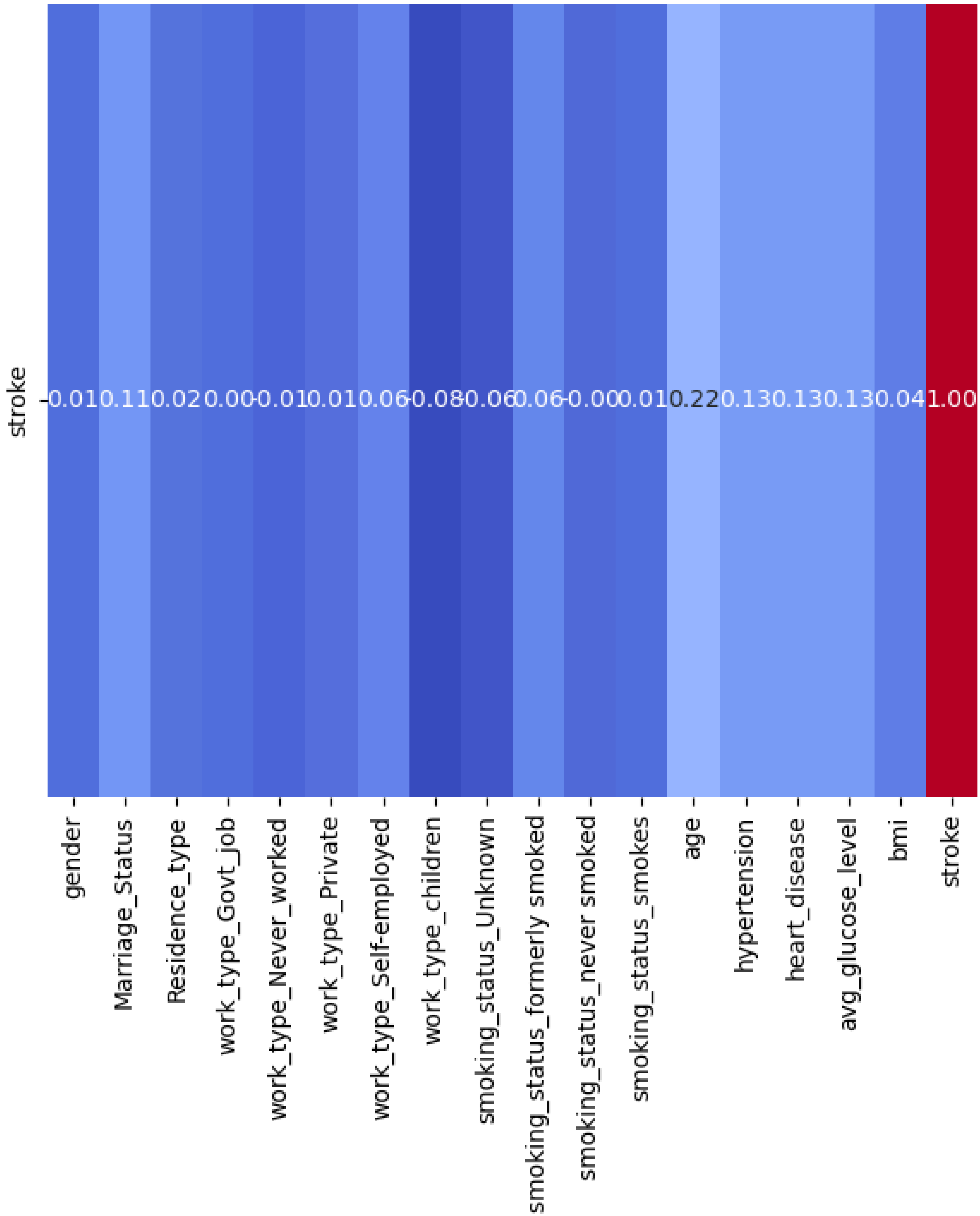
	gender	Marriage_Status	Residence_type	work_type_Govt_job	work_type_Never_work
0	1	1	1	0	
1	0	1	0	0	
2	1	1	0	0	
3	0	1	1	0	
4	0	1	0	0	

In [54]:
```
concat_dataframe.corr().tail(1)
```

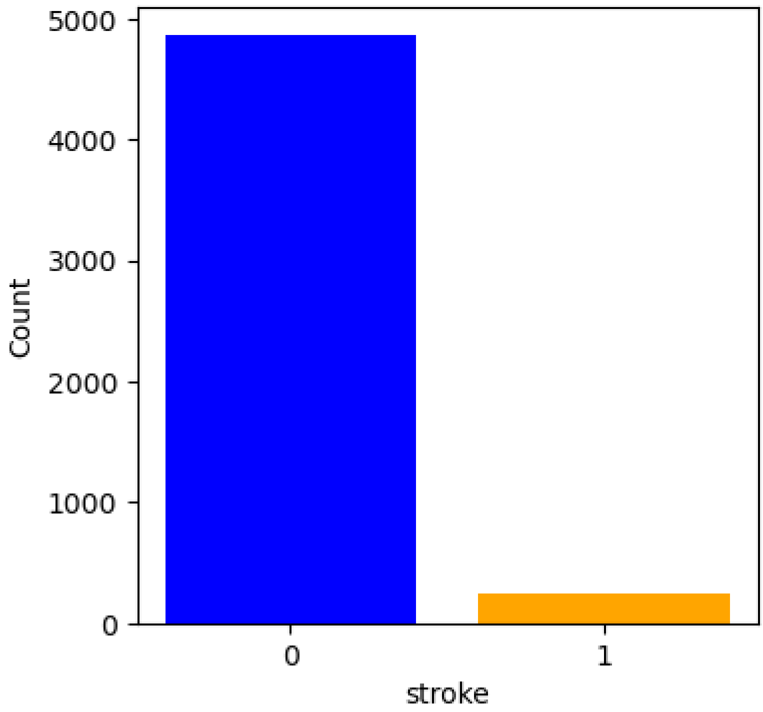
Out[54]:

	gender	Marriage_Status	Residence_type	work_type_Govt_job	work_type_Neve
stroke	0.009027	0.10834	0.015458	0.002677	

In [55]:
```
plt.figure(figsize=(7,6))
sns.heatmap(concat_dataframe.corr().tail(1),cmap = 'coolwarm',fmt = '.2f',annot
plt.show()
```



In [56]:
```
target_count = data['stroke'].value_counts()
plt.figure(figsize=(4,4))
plt.bar(x=target_count.index, height = target_count.values,color=['blue','orange
plt.xticks(ticks = [0,1])
plt.xlabel("stroke")
plt.ylabel("Count")
plt.show()
```



In [57]:
```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
```

In [58]:
```
# splitting data into features and target
x = concat_dataframe.drop('stroke', axis=1)
y = concat_dataframe['stroke']

smt = SMOTE(random_state = 10)
x_sample, y_sample = smt.fit_resample(x,y)

x = x_sample
y = y_sample
```

In [59]:
```
y.value_counts()
```

Out[59]:
```
stroke
1.0    4861
0.0    4861
Name: count, dtype: int64
```

In [60]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

Applying Algorithms

In [62]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import classification_report, confusion_matrix,accuracy_scc
```

In [63]:
```
logistic_model= LogisticRegression()
logistic_model.fit(x_sample,y_sample)
```

Out[63]:

```
▾ LogisticRegression
LogisticRegression()
```

In [64]:

```
y_pred_train1=logistic_model.predict(x_train)

cnf_matrix= confusion_matrix(y_train,y_pred_train1)
print("confusion matrix:\n",cnf_matrix)
print("-"*45)

accuracy=accuracy_score(y_train,y_pred_train1)
print("Accuracy:",accuracy)
print("-"*45)

clf_report=classification_report(y_train,y_pred_train1)
print("classification report:\n",clf_report)
```

```
confusion matrix:
 [[2746  681]
 [ 453 2925]]
---------------------------------------------
Accuracy: 0.8333578251285819
---------------------------------------------
classification report:
               precision    recall  f1-score   support

         0.0       0.86      0.80      0.83      3427
         1.0       0.81      0.87      0.84      3378

    accuracy                           0.83      6805
   macro avg       0.83      0.83      0.83      6805
weighted avg       0.83      0.83      0.83      6805
```

In [65]:

```
y_pred_test1= logistic_model.predict(x_test)

cnf_matrix= confusion_matrix(y_test,y_pred_test1)
print("confusion matrix:\n",cnf_matrix)
print("-"*45)

accuracy=accuracy_score(y_test,y_pred_test1)
print("Accuracy:",accuracy)
print("-"*45)

clf_report=classification_report(y_test,y_pred_test1)
print("classification report:\n",clf_report)
```

```
confusion matrix:
 [[1159  275]
 [ 202 1281]]
---------------------------------------------
Accuracy: 0.8364758313335618
---------------------------------------------
classification report:
               precision    recall  f1-score   support

         0.0       0.85      0.81      0.83      1434
         1.0       0.82      0.86      0.84      1483

    accuracy                           0.84      2917
   macro avg       0.84      0.84      0.84      2917
weighted avg       0.84      0.84      0.84      2917
```

RANDOM FOREST CLASSIFIER

In [67]:
```python
random_classify = RandomForestClassifier(n_estimators=100,criterion="entropy")
random_classify.fit(x_sample,y_sample)
```

Out[67]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [68]:
```python
y_pred_train2=random_classify.predict(x_train)

cnf_matrix= confusion_matrix(y_train,y_pred_train2)
print("confusion matrix:\n",cnf_matrix)
print("*"*45)

accuracy=accuracy_score(y_train,y_pred_train2)
print("Accuracy:",accuracy)
print("*"*45)

clf_report=classification_report(y_train,y_pred_train2)
print("classification report:\n",clf_report)
```

```
confusion matrix:
 [[3427    0]
 [   0 3378]]
*********************************************
Accuracy: 1.0
*********************************************
classification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      3427
         1.0       1.00      1.00      1.00      3378

    accuracy                           1.00      6805
   macro avg       1.00      1.00      1.00      6805
weighted avg       1.00      1.00      1.00      6805
```

In [69]:
```python
y_pred_test2= random_classify.predict(x_test)
```

```
cnf_matrix= confusion_matrix(y_test,y_pred_test2)
print("confusion matrix:\n",cnf_matrix)
print("-"*45)

accuracy=accuracy_score(y_test,y_pred_test2)
print("Accuracy:",accuracy)
print("-"*45)

clf_report=classification_report(y_test,y_pred_test2)
print("classification report:\n",clf_report)
```

```
confusion matrix:
 [[1434    0]
 [   0 1483]]
---------------------------------------------
Accuracy: 1.0
---------------------------------------------
classification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1434
         1.0       1.00      1.00      1.00      1483

    accuracy                           1.00      2917
   macro avg       1.00      1.00      1.00      2917
weighted avg       1.00      1.00      1.00      2917
```

AdaBoost

In [71]:
```
adb_classify=AdaBoostClassifier(random_state=10)
adb_classify.fit(x_sample,y_sample)
```

Out[71]:
```
AdaBoostClassifier
AdaBoostClassifier(random_state=10)
```

In [72]:
```
y_pred_train3 = adb_classify.predict(x_train)

accuracy = accuracy_score(y_train, y_pred_train3)

accuracy
```

Out[72]: 0.8498163115356355

In [73]:
```
y_pred_test3 = adb_classify.predict(x_test)

accuracy = accuracy_score(y_test, y_pred_test3)

accuracy
```

Out[73]: 0.8385327391155296

In [ ]:

In [ ]:

In [ ]: