stroke-prediction-model

May 1, 2024

Author : Asad Mujeeb

Project : ***Stroke Prediction Model***

gmail : asadmujeeb@student.bzu.edu.pk

contact : +923166328876

Linkedin : Linkedin

[74]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

```
/kaggle/input/stroke-prediction-dataset/healthcare-dataset-stroke-data.csv
```

[2]:
```
# read the data
df = pd.read_csv(r"/kaggle/input/stroke-prediction-dataset/
↪healthcare-dataset-stroke-data.csv")
```

[3]:
```
# show the sample data
df.head()
```

[3]:
```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             0              0          Yes
4   1665  Female  79.0             1              0          Yes

       work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0        Private          Urban             228.69  36.6  formerly smoked
1  Self-employed          Rural             202.21   NaN     never smoked
2        Private          Rural             105.92  32.5     never smoked
3        Private          Urban             171.23  34.4           smokes
4  Self-employed          Rural             174.12  24.0     never smoked

   stroke
0       1
1       1
2       1
3       1
4       1
```

[4]:
```
# count the values

df.value_counts()
```

[4]:
```
id     gender  age   hypertension  heart_disease  ever_married  work_type
Residence_type  avg_glucose_level  bmi   smoking_status   stroke
77     Female  13.0  0             0              No            children
Rural           85.81              18.6  Unknown          0         1
49605  Male    63.0  0             0              Yes           Private
Urban           74.39              31.0  formerly smoked  0         1
49661  Male    53.0  0             0              Yes           Govt_job
Urban           85.17              29.2  never smoked     0         1
49646  Male    72.0  0             1              Yes           Self-employed
Rural           113.63             26.5  Unknown          0         1
49645  Male    58.0  0             0              No            Private
Rural           76.22              22.2  formerly smoked  0         1
                                                                ..
25138  Female  78.0  1             0              Yes           Private
Rural           91.63              33.5  smokes           0         1
25130  Female  27.0  0             0              Yes           Private
Urban           79.21              19.5  Unknown          0         1
25107  Female  47.0  0             0              Yes           Private
Urban           65.04              30.9  never smoked     0         1
```

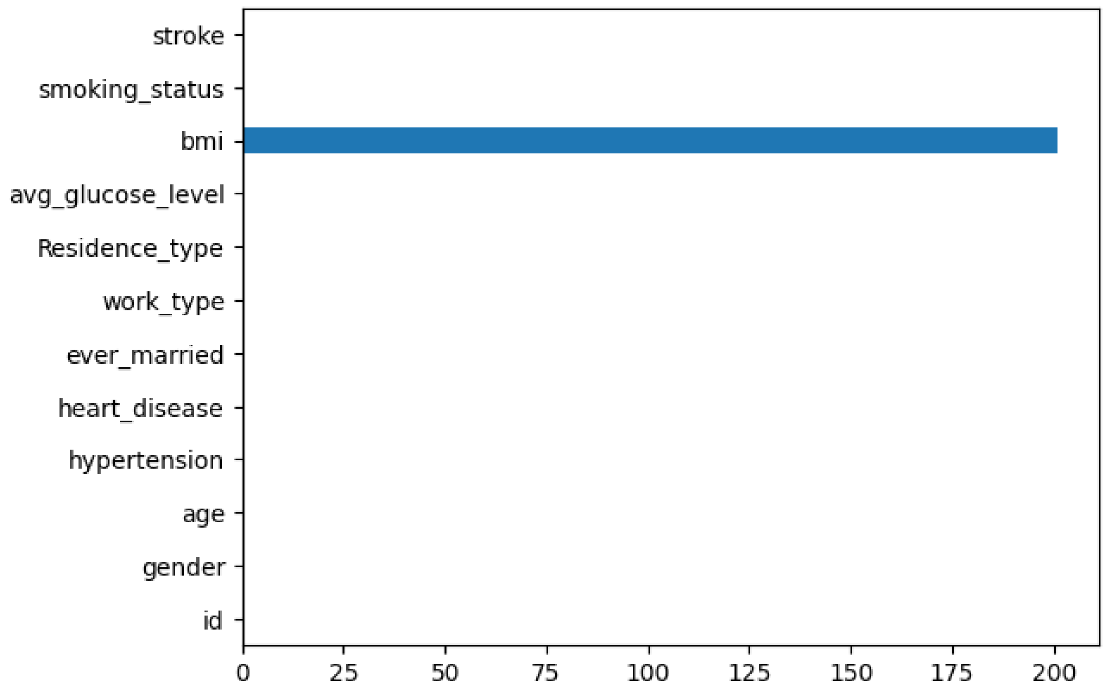
```
25102  Female  51.0  0            0            Yes           Govt_job
Urban           95.16             42.7  formerly smoked  0        1
72940  Female  2.0   0            0            No            children
Urban           102.92            17.6  Unknown          0        1
Name: count, Length: 4909, dtype: int64
```

1 Find the number of null values in each column

[5]:
```python
print(df.isna().sum())

# graphical representation of null values
df.isna().sum().plot(kind = "barh")
```

```
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                  201
smoking_status         0
stroke                 0
dtype: int64
```

[5]: <Axes: >



Found 201 NULL values in bmi column

[6]:
```
# statistical analysis

df.describe(include = "all")
```

[6]:
```
                  id  gender          age  hypertension  heart_disease  \
count    5110.000000    5110  5110.000000   5110.000000    5110.000000
unique           NaN       3          NaN           NaN            NaN
top              NaN  Female          NaN           NaN            NaN
freq             NaN    2994          NaN           NaN            NaN
mean    36517.829354     NaN    43.226614      0.097456       0.054012
std     21161.721625     NaN    22.612647      0.296607       0.226063
min        67.000000     NaN     0.080000      0.000000       0.000000
25%     17741.250000     NaN    25.000000      0.000000       0.000000
50%     36932.000000     NaN    45.000000      0.000000       0.000000
75%     54682.000000     NaN    61.000000      0.000000       0.000000
max     72940.000000     NaN    82.000000      1.000000       1.000000

       ever_married work_type Residence_type  avg_glucose_level          bmi  \
count          5110      5110           5110        5110.000000  4909.000000
unique            2         5              2                NaN          NaN
top             Yes   Private          Urban                NaN          NaN
freq           3353      2925           2596                NaN          NaN
mean            NaN       NaN            NaN         106.147677    28.893237
```

```
std               NaN         NaN            NaN          45.283560     7.854067
min               NaN         NaN            NaN          55.120000    10.300000
25%               NaN         NaN            NaN          77.245000    23.500000
50%               NaN         NaN            NaN          91.885000    28.100000
75%               NaN         NaN            NaN         114.090000    33.100000
max               NaN         NaN            NaN         271.740000    97.600000

       smoking_status       stroke
count            5110  5110.000000
unique              4          NaN
top      never smoked          NaN
freq             1892          NaN
mean              NaN     0.048728
std               NaN     0.215320
min               NaN     0.000000
25%               NaN     0.000000
50%               NaN     0.000000
75%               NaN     0.000000
max               NaN     1.000000
```

[7]:
```
# provide the datatype of all column

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
```

2 Pre-Processign + EDA

[8]:
```
df = df.drop(['id'], axis = 1)
```

3 Gender Analysis

[9]:
```
df["gender"].value_counts()
```

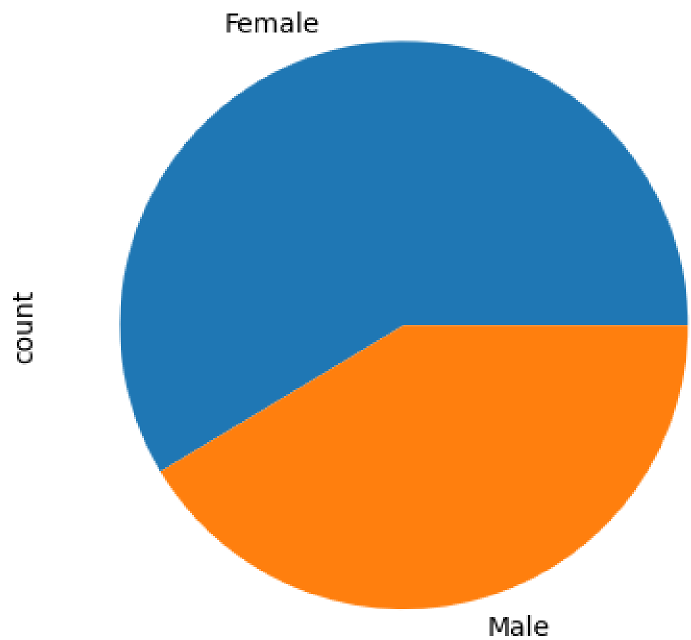
[9]:
```
gender
Female    2994
Male      2115
Other        1
Name: count, dtype: int64
```

we have a ‘other’ gender , we will remove it

[10]:
```
# remove the other gender
df["gender"] = df["gender"].replace("Other", "Female")

# plot the pie chart

print(df["gender"].value_counts())

df["gender"].value_counts().plot(kind = "pie")
```

```
gender
Female    2995
Male      2115
Name: count, dtype: int64
```

[10]:
```
<Axes: ylabel='count'>
```



4 Target Feature - Stroke

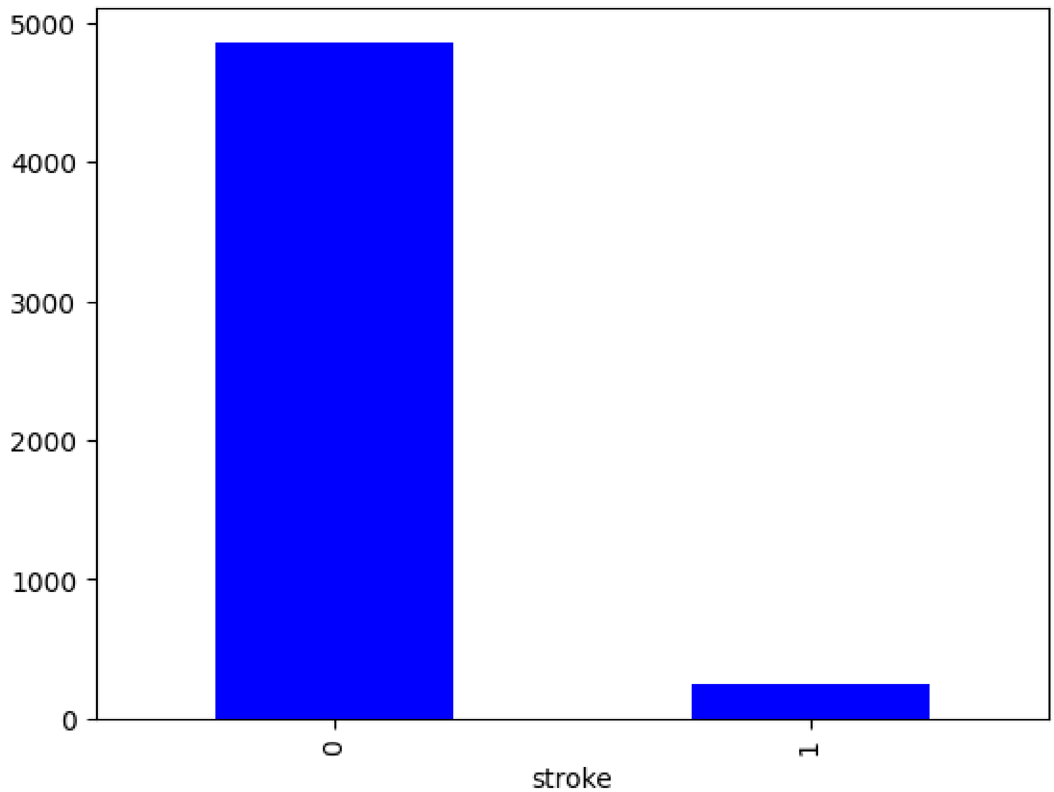
Stroke Analysis

[11]:
```
# value count in the stroke attributes

df["stroke"].value_counts()
```

[11]:
```
stroke
0    4861
1     249
Name: count, dtype: int64
```

[12]:
```
# plot the bar graph

df["stroke"].value_counts().plot(kind = "bar", color = "blue")
```

[12]: <Axes: xlabel='stroke'>



[13]:
```
print("% of people who actually got stroke ", (df["stroke"].value_counts()[1] /
    df["stroke"].value_counts().sum()).round(3) * 100)
```

```
% of people who actually got stroke  4.9
```

Only 5% of people got stroke

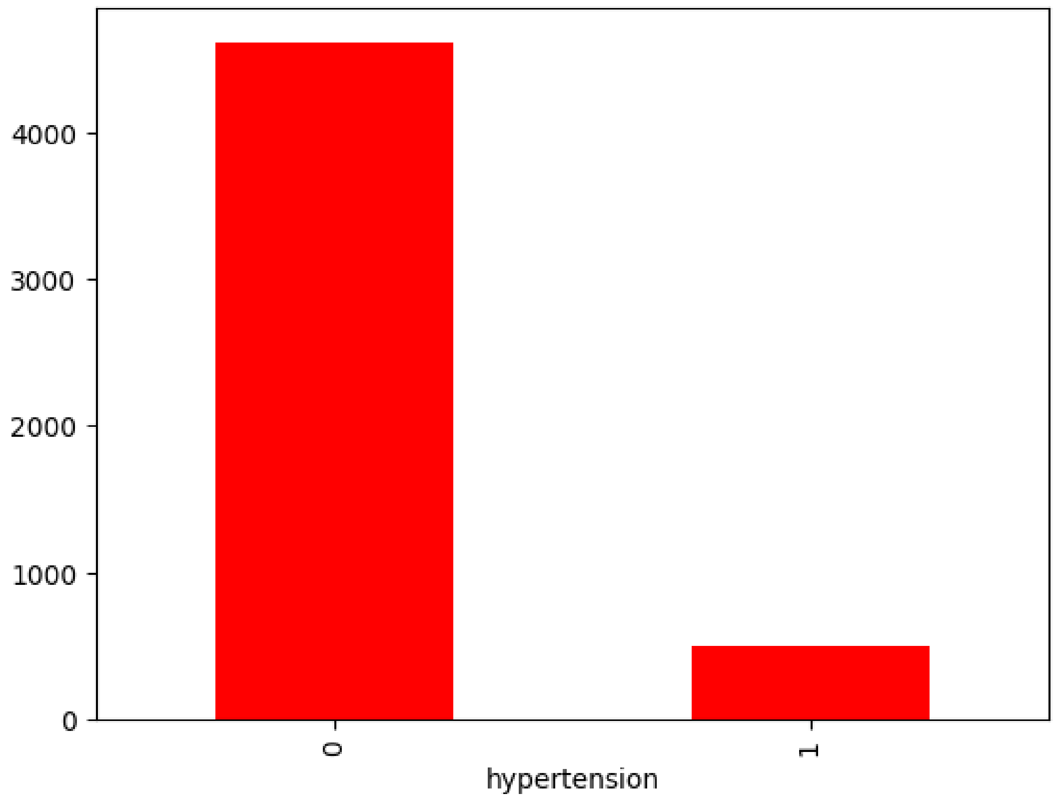
5 Hyper-tension Analysis

[14]:
```
df['hypertension'].value_counts()
```

[14]:
```
hypertension
0    4612
1     498
Name: count, dtype: int64
```

[15]:
```
df['hypertension'].value_counts().plot(kind = "bar", color = "red")
```

[15]: <Axes: xlabel='hypertension'>



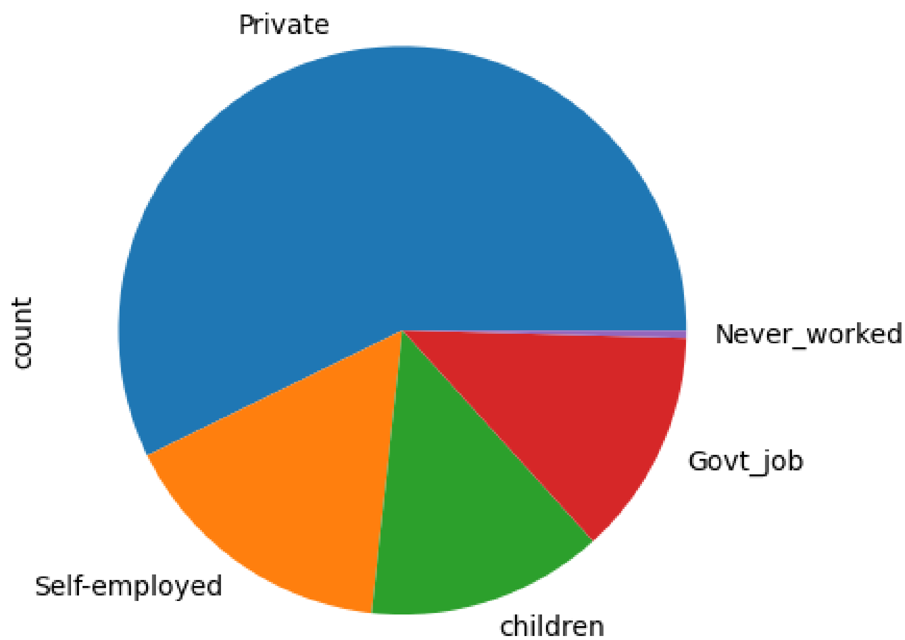
6 Work type analysis

[16]:
```
df["work_type"].value_counts()
```

[16]:
```
work_type
Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: count, dtype: int64
```

[17]:
```
# graphical representation

df['work_type'].value_counts().plot(kind = "pie")
```

[17]:
```
<Axes: ylabel='count'>
```



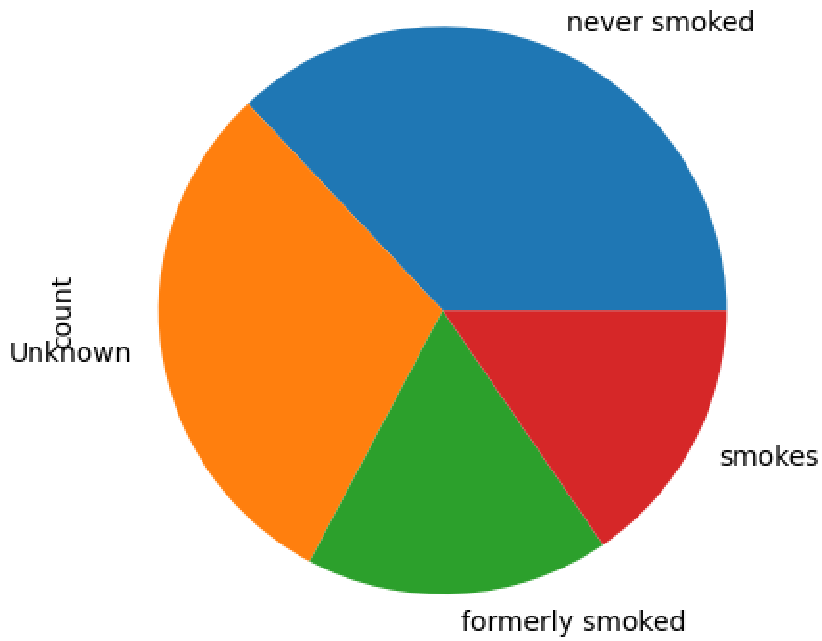
7 Smoking status Analysis

[18]:
```
df["smoking_status"].value_counts()
```

[18]:
```
smoking_status
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: count, dtype: int64
```

[19]:
```
df["smoking_status"].value_counts().plot(kind = "pie")
```

[19]:
```
<Axes: ylabel='count'>
```



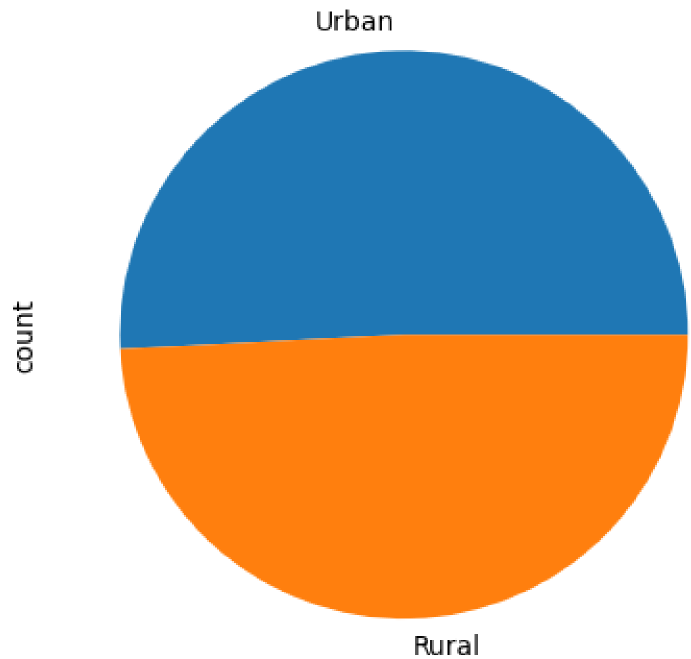
8 Residance type Analysis

[20]:
```
df["Residence_type"].value_counts()
```

[20]:
```
Residence_type
Urban    2596
Rural    2514
Name: count, dtype: int64
```

[21]:
```
df["Residence_type"].value_counts().plot(kind = "pie")
```

[21]:
```
<Axes: ylabel='count'>
```



We have equal percentage of population who are from urban and rural areas

9 BMI Analysis

[22]:
```
# Number of  BMI- NULL values

df["bmi"].isnull().sum()
```

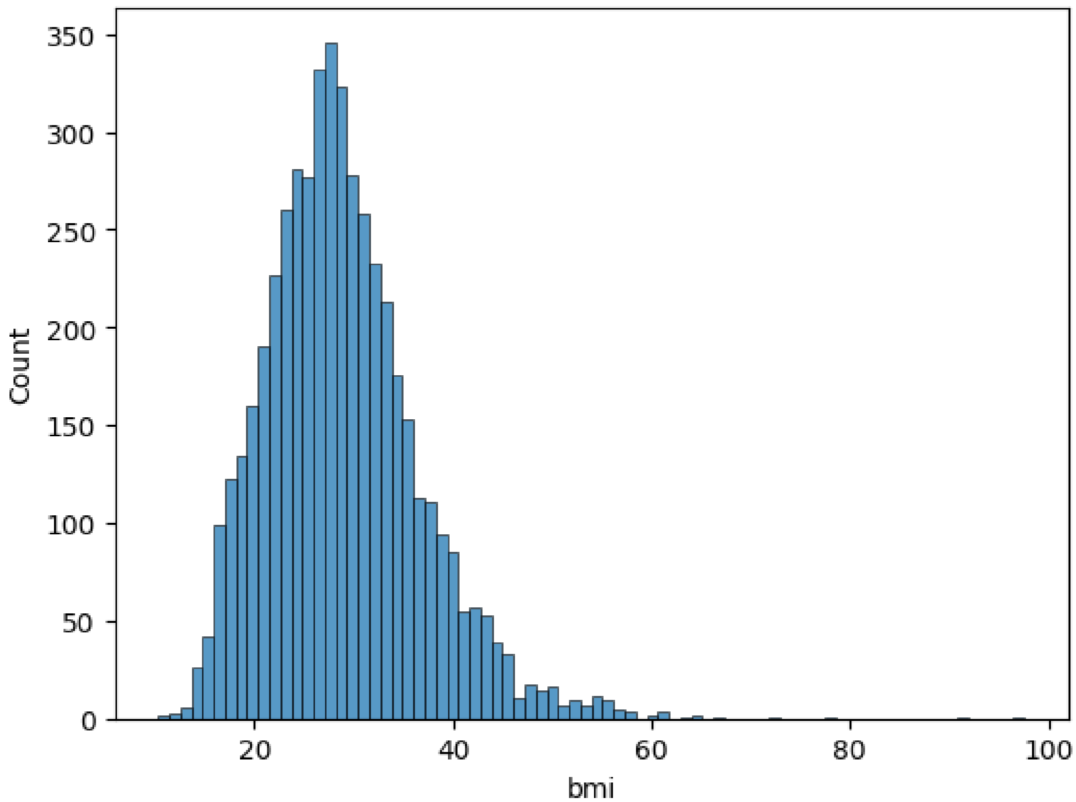
[22]: 201

we have null values in BMI column contains 201 NaN

[23]:
```
sns.histplot(data = df["bmi"])
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

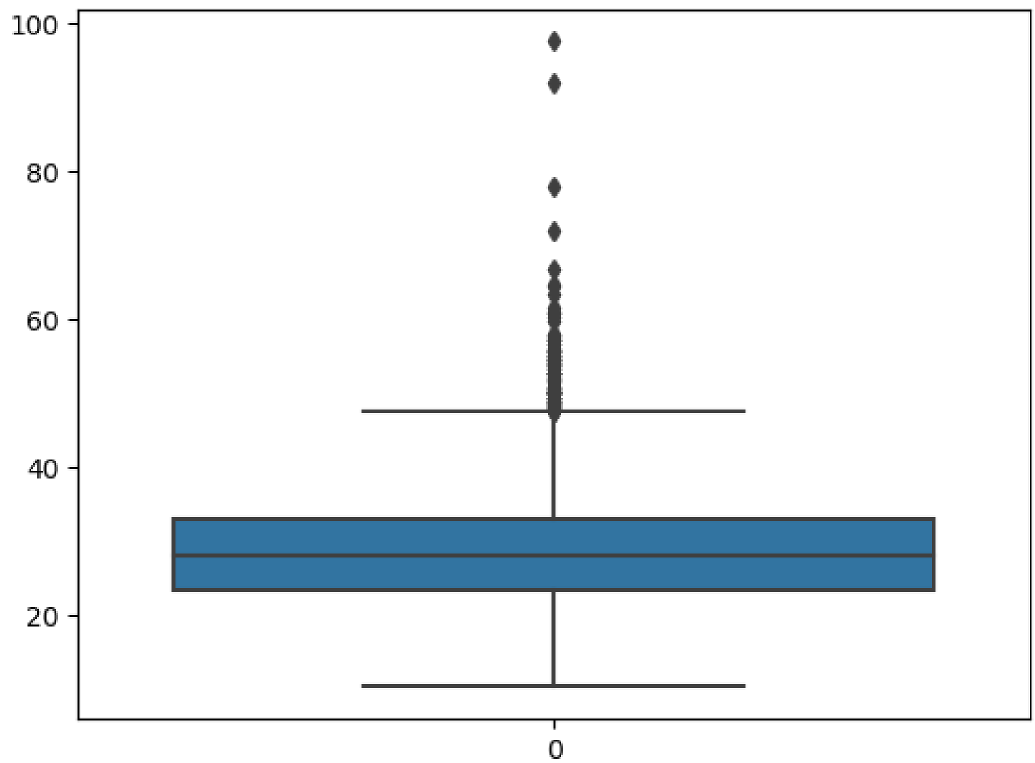
[23]: <Axes: xlabel='bmi', ylabel='Count'>



BMI is highly skewed

[24]:
```
sns.boxplot(data = df["bmi"])
```

[24]: <Axes: >



Based on histogram and distplot , there are outliers in BMI column

[25]:
```
# finding the count of ouliers
q1 = df["bmi"].quantile(0.25)
q3 = df["bmi"].quantile(0.75)
# Finding IQR
IQR = q3 - q1
da = (df["bmi"] < (q1 - 1.5 * IQR)) | (df["bmi"] > (q3 + 1.5 * IQR))
da.value_counts()
```

[25]:
```
bmi
False    5000
True      110
Name: count, dtype: int64
```

Total Outliers : 110

Total non-outliers : 5000

[26]:
```
# percentage of null values in BMI
df["bmi"].isna().sum() / len(df["bmi"]) * 100
```

[26]:
```
3.9334637964774952
```

NULL values hold 3.93% of the dataframe

[27]:
```
df_na = df.loc[df["bmi"]. isnull()]
g = df_na["stroke"].sum()
print("People who got stroke and their BMI is NA :", g)
h = df["stroke"].sum()
print("people who got stroke and their BMI is given", h)
print("percentage of people with stroke in NAN values to the overall dataset :␣
↪", g / h * 100)
```

```
People who got stroke and their BMI is NA : 40
people who got stroke and their BMI is given 249
percentage of people with stroke in NAN values to the overall dataset :
16.06425702811245
```

[28]:
```
# percentage of instance who got stroke

df["stroke"].sum() / len(df) * 100
```

[28]:
```
4.87279843444227
```

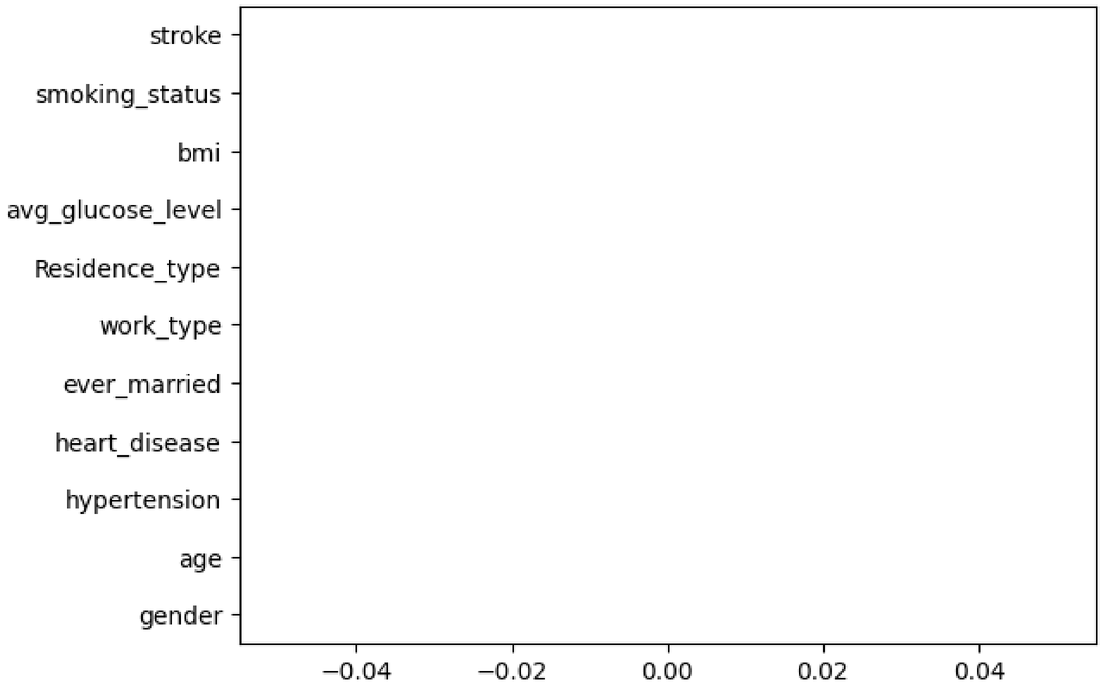
our main target is stroke and who got stroke in the minority 249 which is only 4.9%

[29]:
```
# dealing with null values in BMI

print("Median of BMI : ", df["bmi"].median())

df["bmi"] = df["bmi"].fillna(df["bmi"].median())
```

```
Median of BMI :  28.1
```

[30]:
```
df.isna().sum().plot(kind = "barh")
```

[30]:
```
<Axes: >
```

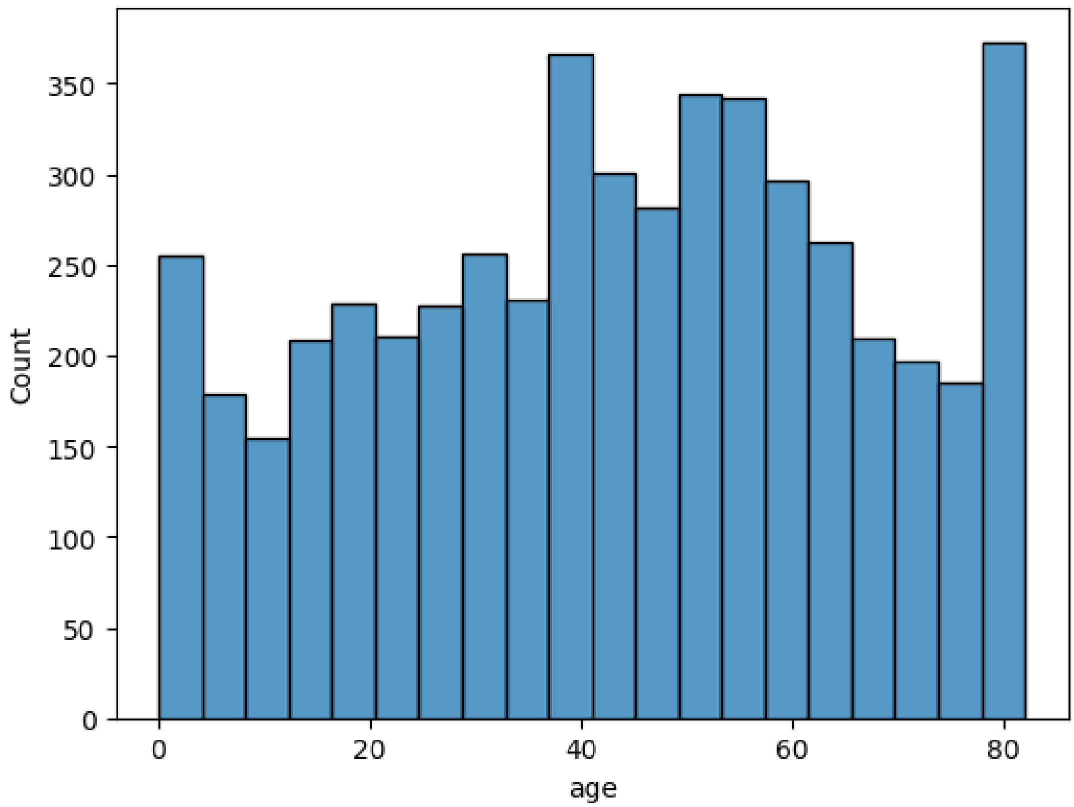


10 Age Analysis

[31]:
```
sns.histplot(data = df["age"])
```

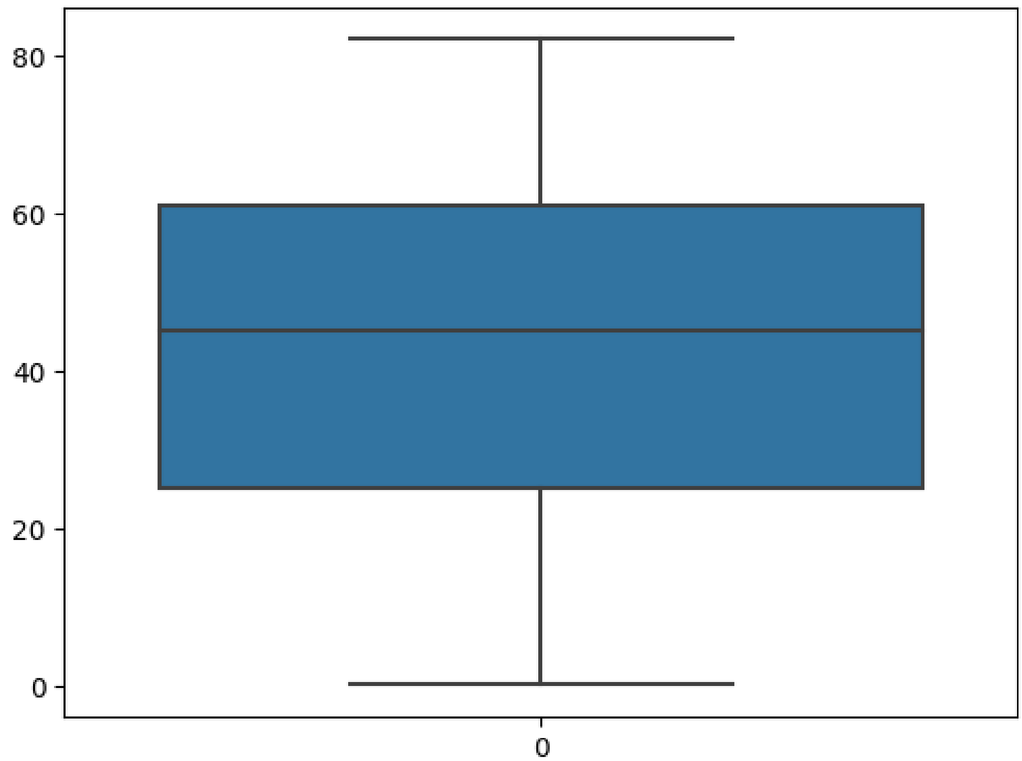
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[31]:
```
<Axes: xlabel='age', ylabel='Count'>
```



[32]:
```
sns.boxplot(data = df["age"])
```

[32]: <Axes: >



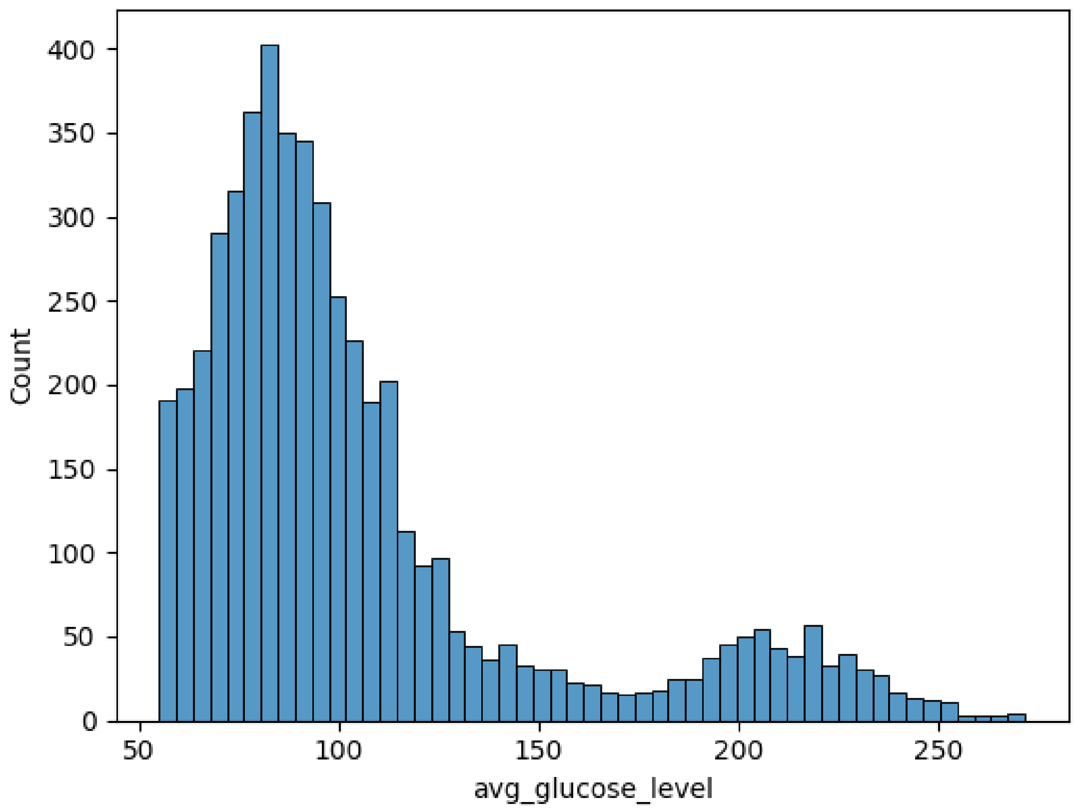
The age parameter value does not have outliers

11 Average Glucose Level

[33]:
```
sns.histplot(data = df["avg_glucose_level"])
```

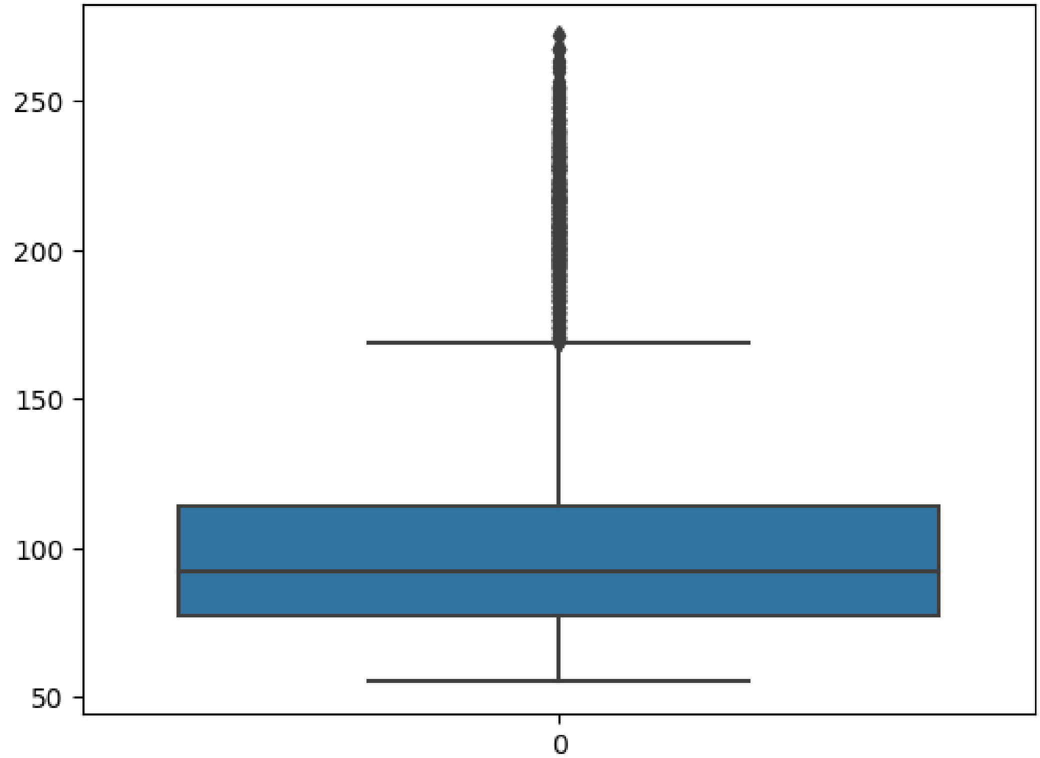
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[33]:
```
<Axes: xlabel='avg_glucose_level', ylabel='Count'>
```



[34]:
```
# boxplot
sns.boxplot(data = df["avg_glucose_level"])
```

[34]: <Axes: >



[35]:
```
# finding the count of outliers in avg_glucose_level

q1 = df["avg_glucose_level"].quantile(0.25)
q3 = df["avg_glucose_level"].quantile(0.75)
IQR = q3 - q1
da = (df["avg_glucose_level"] < (q1 - 1.5 * IQR)) | (df["avg_glucose_level"] >
↪(q3 + 1.5 * IQR))
da.value_counts()
```

[35]:
```
avg_glucose_level
False    4483
True      627
Name: count, dtype: int64
```

Total outliers in avg_glucose_level : 627

Total non-outliers in avg_glucose_level : 4483

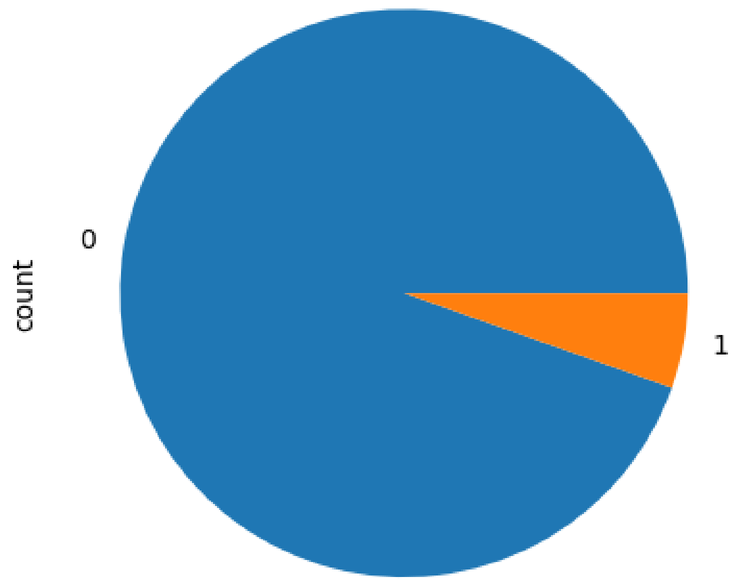
12 Heart diseases Analysis

[36]:
```
df["heart_disease"].value_counts()
```

[36]:
```
heart_disease
0    4834
1     276
Name: count, dtype: int64
```

[37]:
```
df["heart_disease"].value_counts().plot(kind = "pie")
```

[37]:
```
<Axes: ylabel='count'>
```



13 Ever married analysis with values

[38]:
```
df["ever_married"].value_counts()
```

[38]:
```
ever_married
Yes    3353
No     1757
Name: count, dtype: int64
```

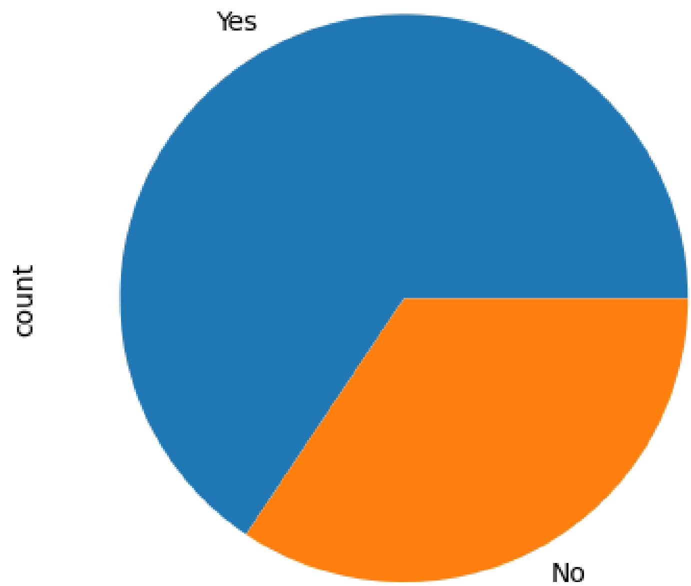
[39]:
```
3353 / len(df)
```

[39]: 0.6561643835616439

This result shows that 65.6% of people are married and 34.4 % are un-married of the population

[40]:
```
df["ever_married"].value_counts().plot(kind = "pie")
```

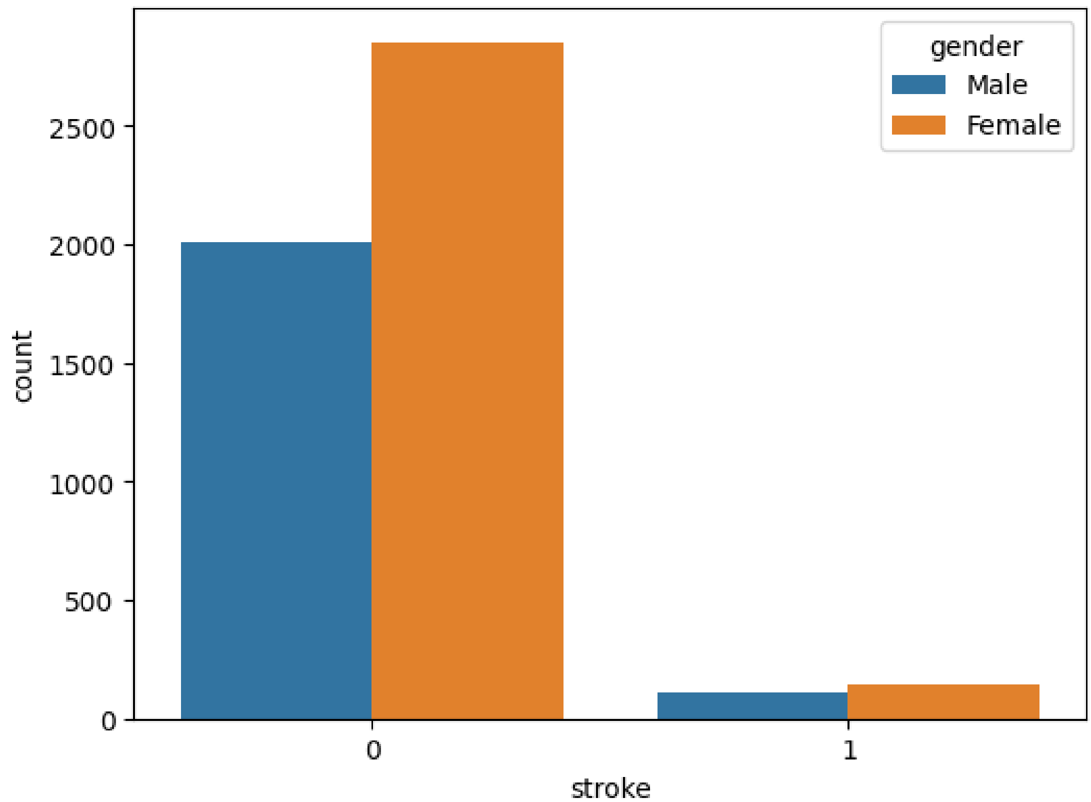
[40]: <Axes: ylabel='count'>



14 Cross Analysis - All attributes compared with target variable

[41]:
```
# comparing stroke with gender

sns.countplot(x = "stroke", hue = "gender", data = df)
```

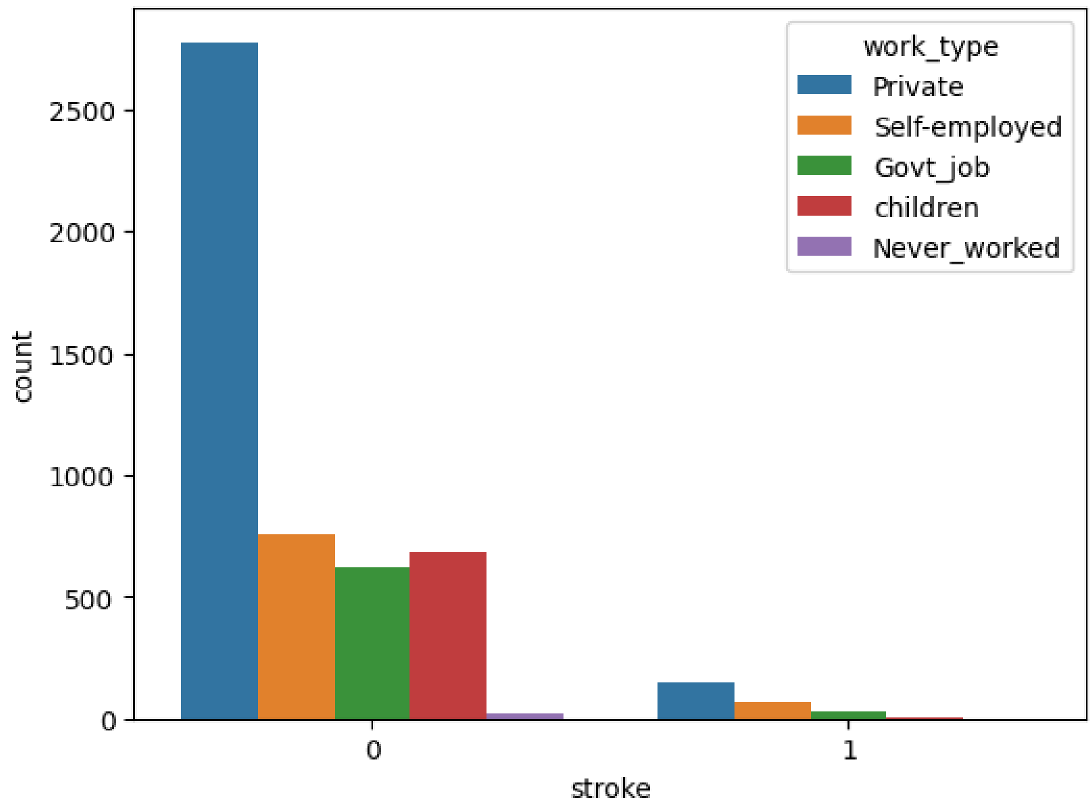
[41]: <Axes: xlabel='stroke', ylabel='count'>



15 comparing stroke with work-type

[42]:
```
sns.countplot(x = "stroke", hue = "work_type", data = df)
```

[42]: `<Axes: xlabel='stroke', ylabel='count'>`

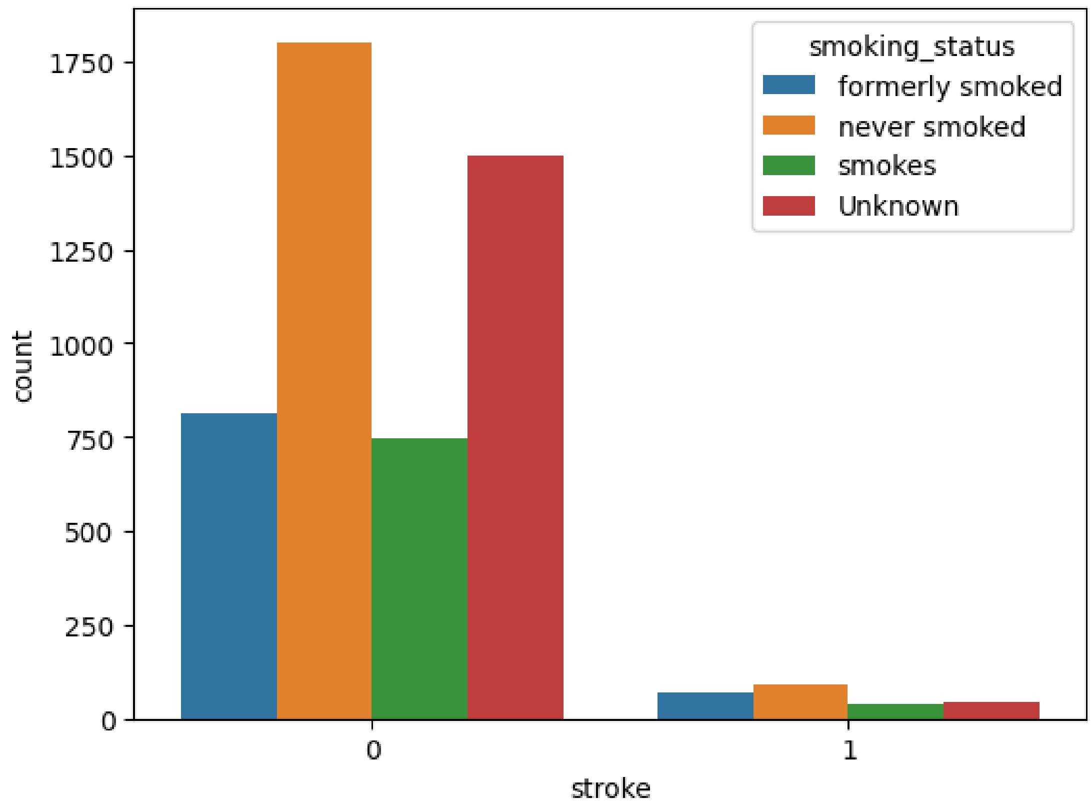


People who never worked did not face heart attack and the people who are privately employed got more heart attack

16 comparing stroke with smoking status

[43]:
```python
sns.countplot(x = "stroke", hue = "smoking_status" , data = df)
```

[43]: <Axes: xlabel='stroke', ylabel='count'>

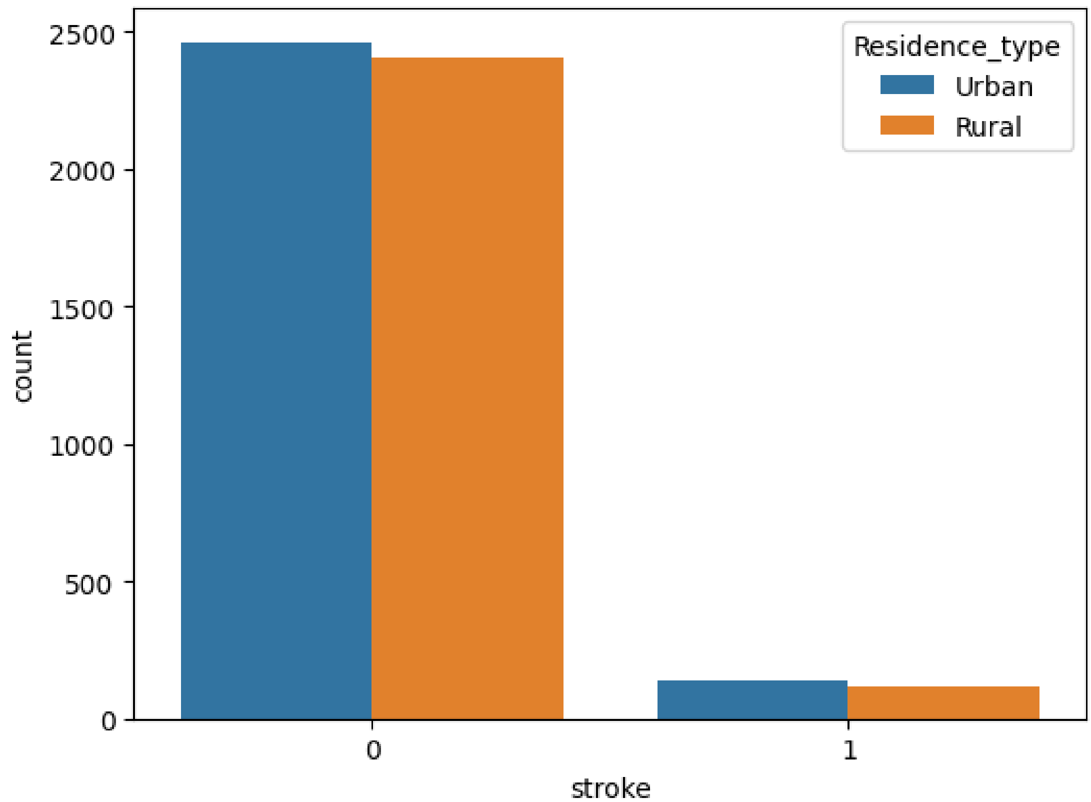


people who formely smoked got more strokes. The people who smoked and never smoked has same probability of getting stroke

17 comparing stroke with residance type

[44]:
```
sns.countplot(x = "stroke", hue = "Residence_type", data = df)
```

[44]:
```
<Axes: xlabel='stroke', ylabel='count'>
```

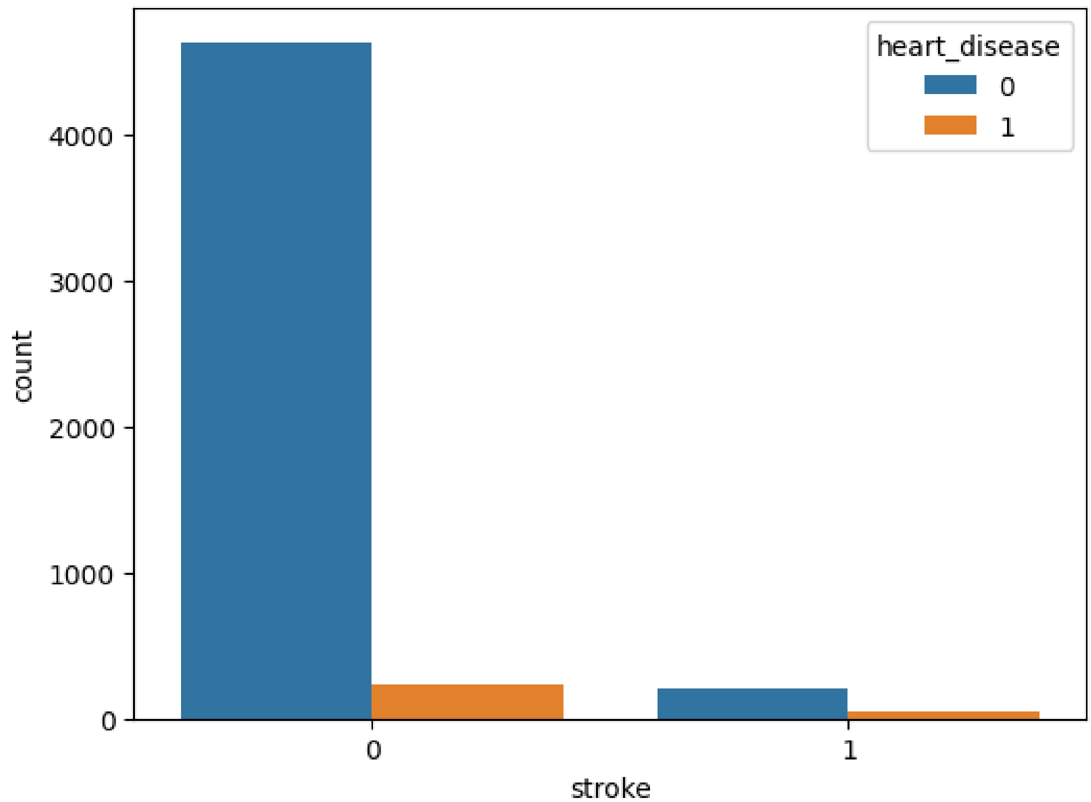


people who lived in urban areas are more like to getting stroke as compared rural areas

18 Comparing stroke with heart diseases

[45]:
```python
sns.countplot(x = "stroke", hue = "heart_disease", data = df)
```

[45]:
```
<Axes: xlabel='stroke', ylabel='count'>
```

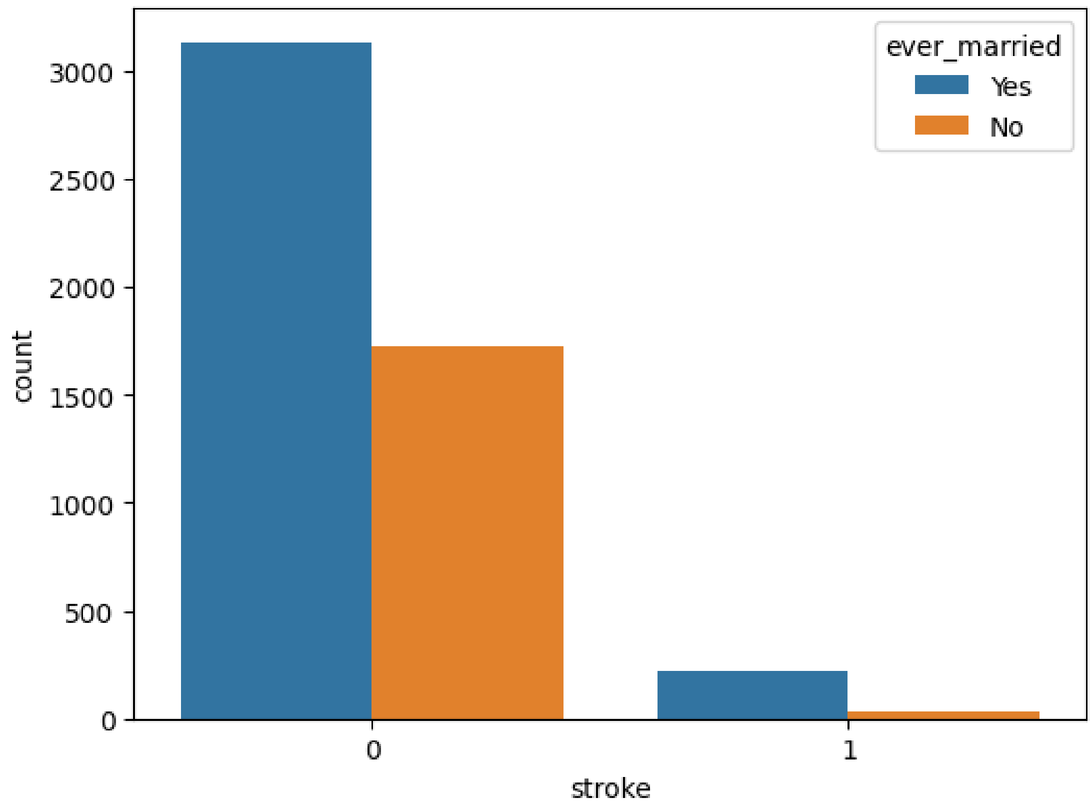


People with no heart diseases are suffering are suffering with stroke as compared to one who face heart problem

19 Comparing stroke with married status

[46]:
```
sns.countplot(x = "stroke", hue = "ever_married", data = df)
```

[46]: <Axes: xlabel='stroke', ylabel='count'>



Married people are more likely to get stroke as compared to un-married people

[47]:
```
df.info()
```

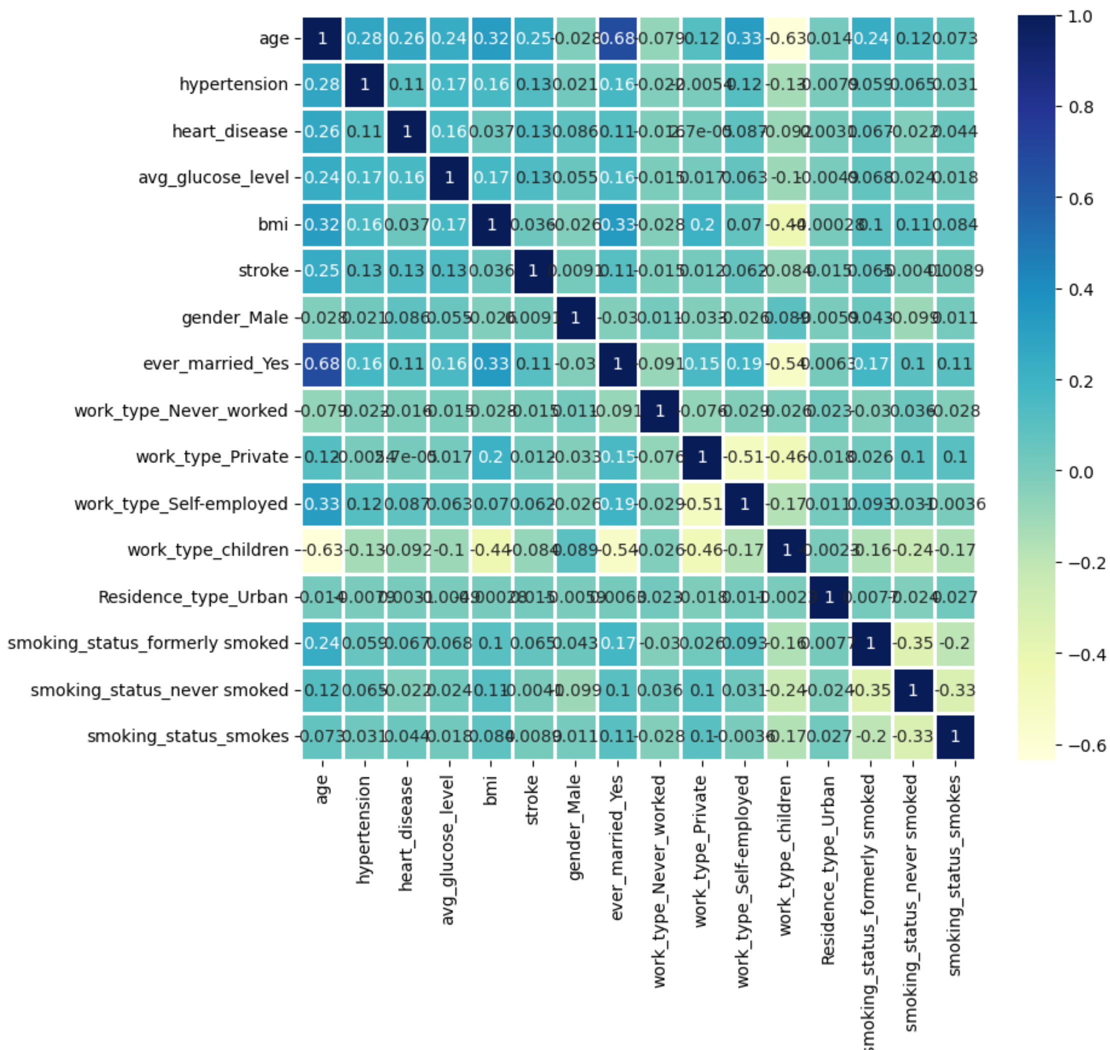
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5110 non-null   object
 1   age                5110 non-null   float64
 2   hypertension       5110 non-null   int64
 3   heart_disease      5110 non-null   int64
 4   ever_married       5110 non-null   object
 5   work_type          5110 non-null   object
 6   Residence_type     5110 non-null   object
 7   avg_glucose_level  5110 non-null   float64
 8   bmi                5110 non-null   float64
 9   smoking_status     5110 non-null   object
 10  stroke             5110 non-null   int64
dtypes: float64(3), int64(3), object(5)
memory usage: 439.3+ KB
```

20 creating dummy variable numeric binary attributes

[48]:
```
# converting numerical binary values to string

df[["hypertension", "heart_disease", "stroke"]].astype(str)

# generate the dummy attributes

df = pd.get_dummies(df, drop_first = True)
```

[49]:
```
# correlation matrix

corrmat = df.corr()
f,ax = plt.subplots(figsize = (9,8))
sns.heatmap(corrmat, ax = ax ,cmap = "YlGnBu", linewidths = 0.8, annot = True)
```

[49]: <Axes: >



[50]:
```python
# the dataframe after performing dummy attributes
df.head()
```

[50]:
```
    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

   gender_Male  ever_married_Yes  work_type_Never_worked  work_type_Private  \
0         True              True                   False               True
1        False              True                   False              False
2         True              True                   False               True
3        False              True                   False               True
4        False              True                   False              False

   work_type_Self-employed  work_type_children  Residence_type_Urban  \
0                    False               False                  True
1                     True               False                 False
2                    False               False                 False
3                    False               False                  True
4                     True               False                 False

   smoking_status_formerly smoked  smoking_status_never smoked  \
0                            True                        False
1                           False                         True
2                           False                         True
3                           False                        False
4                           False                         True

   smoking_status_smokes
0                  False
1                  False
2                  False
3                   True
4                  False
```

[51]:
```python
# use random over-sampling
from imblearn.over_sampling import RandomOverSampler
```

[52]:
```python
oversample = RandomOverSampler(sampling_strategy = "minority")

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

[53]:
```
x
```

[53]:
```
       age  hypertension  heart_disease  avg_glucose_level   bmi  gender_Male  \
0     67.0             0              1             228.69  36.6         True
1     61.0             0              0             202.21  28.1        False
2     80.0             0              1             105.92  32.5         True
3     49.0             0              0             171.23  34.4        False
4     79.0             1              0             174.12  24.0        False
…      …             …              …                  …     …           …
5105  80.0             1              0              83.75  28.1        False
5106  81.0             0              0             125.20  40.0        False
5107  35.0             0              0              82.99  30.6        False
5108  51.0             0              0             166.29  25.6         True
5109  44.0             0              0              85.28  26.2        False

      ever_married_Yes  work_type_Never_worked  work_type_Private  \
0                 True                   False               True
1                 True                   False              False
2                 True                   False               True
3                 True                   False               True
4                 True                   False              False
…                  …                      …                  …
5105              True                   False               True
5106              True                   False              False
5107              True                   False              False
5108              True                   False               True
5109              True                   False              False

      work_type_Self-employed  work_type_children  Residence_type_Urban  \
0                       False               False                  True
1                        True               False                 False
2                       False               False                 False
3                       False               False                  True
4                        True               False                 False
…                         …                  …                    …
5105                    False               False                  True
5106                     True               False                  True
5107                     True               False                 False
5108                    False               False                 False
5109                    False               False                  True

      smoking_status_formerly smoked  smoking_status_never smoked  \
0                               True                       False
1                              False                        True
2                              False                        True
3                              False                       False
4                              False                        True
```

```
…                              …                          …
5105                       False                       True
5106                       False                       True
5107                       False                       True
5108                        True                      False
5109                       False                      False

      smoking_status_smokes
0                     False
1                     False
2                     False
3                      True
4                     False
…                     …
5105                  False
5106                  False
5107                  False
5108                  False
5109                  False

[5110 rows x 15 columns]
```

[54]:
```python
y = df["stroke"]
```

[55]:
```python
y
```

[55]:
```
0       1
1       1
2       1
3       1
4       1
       ..
5105    0
5106    0
5107    0
5108    0
5109    0
Name: stroke, Length: 5110, dtype: int64
```

[56]:
```python
x_over, y_over = oversample.fit_resample(x,y)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df[["bmi", "avg_glucose_level", "age"]] = scaler.fit_transform(df[["bmi",
                                    "avg_glucose_level", "age"]])
```

[57]:
```
# create a train test split
from sklearn.model_selection import train_test_split
x_train , x_test, y_train, y_test = train_test_split(x_over,y_over, test_size =
    0.2, random_state = 42)
```

[58]:
```
# checking the shape of split

print("x_train : ", x_train.shape)
print("y_train : ", y_train.shape)
print("x_test : ", x_test.shape)
print("y_test : ", y_test.shape)
```

```
x_train :  (7777, 15)
y_train :  (7777,)
x_test :  (1945, 15)
y_test :  (1945,)
```

21 1. Decision Tree Model*

[59]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import auc, roc_auc_score, roc_curve, precision_score,
    recall_score, f1_score
```

[60]:
```
clf = DecisionTreeClassifier()
clf = clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Accuracy : ", metrics.accuracy_score(y_test, y_pred) * 100)
print("ROC AUC Score : ", roc_auc_score(y_test, y_pred))
```

```
Accuracy :  98.04627249357326
ROC AUC Score :  0.9805128205128205
```

22 2. KNN Model

[61]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
knn = KNeighborsClassifier(n_neighbors = 2)
knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)
y_pred_prob_knn = knn.predict_proba(x_test)[:, 1]
print("Accuracy score : ", accuracy_score(y_test, y_pred) * 100)
print("ROC AUC Score : ", roc_auc_score(y_test, y_pred) * 100)
```

```
Accuracy score :  98.04627249357326
ROC AUC Score :  98.05128205128206
```

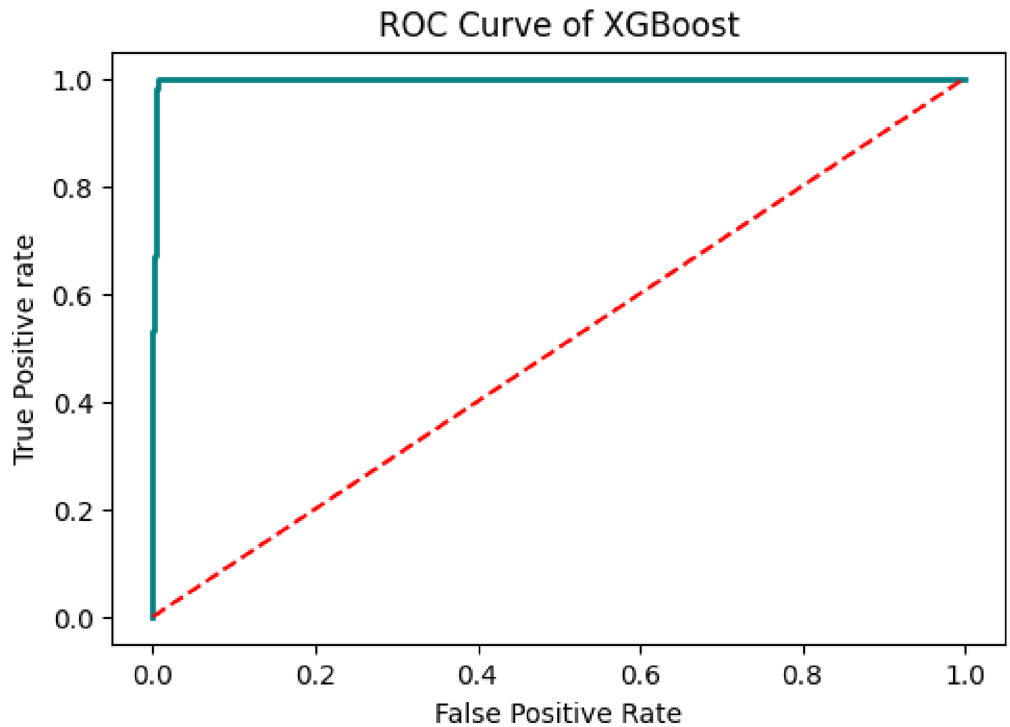
23 3- XGBoost Model

[62]:
```
from xgboost import XGBClassifier

xgb = XGBClassifier()
xgb.fit(x_train, y_train)

y_pred_xgb = xgb.predict(x_test)
y_pred_prob_xgb = xgb.predict_proba(x_test)[:, 1]

print("Accuarcy score : ", accuracy_score(y_test, y_pred))
print("ROC AUC Score : ", roc_auc_score(y_test, y_pred))
```

```
Accuarcy score :  0.9804627249357326
ROC AUC Score :  0.9805128205128205
```

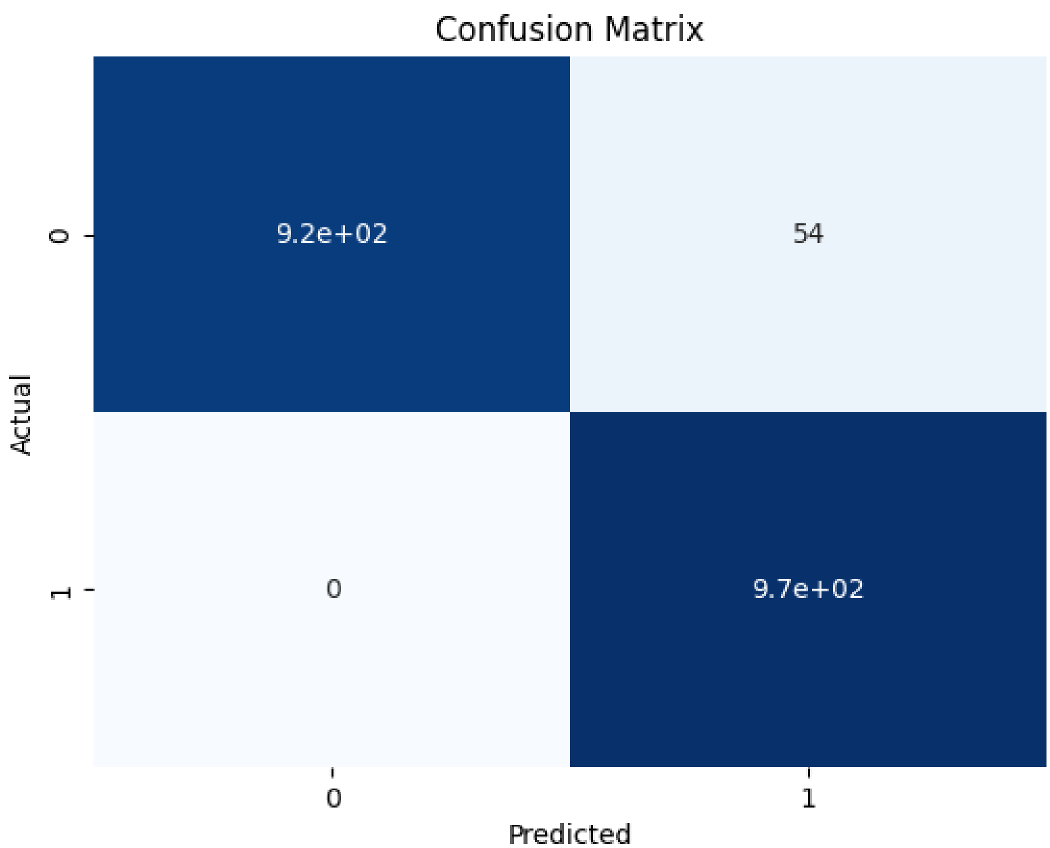
24 4- Plot ROC AUC

[63]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_xgb)
plt.figure(figsize = (6,4))
plt.plot(fpr, tpr, linewidth = 2, color = "teal")
plt.plot([0,1], [0,1], "r--")
plt.title("ROC Curve of XGBoost")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive rate")
plt.show()
```



25 Confusion Matrix

[64]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_xgb)
sns.heatmap(cm, annot = True, cmap = "Blues", cbar = False)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



[65]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
print("Accuracy Score : ", accuracy_score(y_test, y_pred_xgb))
print("precision score : ", precision_score(y_test, y_pred_xgb))
print("recall score : ", recall_score(y_test, y_pred_xgb))
print("f1 score : ", f1_score(y_test, y_pred_xgb))
```

```
Accuracy Score :  0.9722365038560411
precision score :  0.947265625
recall score :  1.0
f1 score :  0.9729187562688064
```

26 5- Random Forest Classifier Model

[66]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

rf_clf = RandomForestClassifier(n_estimators = 100)
rf_clf.fit(x_train, y_train)
```

```
y_pred_rf = rf_clf.predict(x_test)
print("Accuracy Score : ", accuracy_score(y_test, y_pred_rf) * 100)
```

```
Accuracy Score :  99.43444730077121
```

27 6- KFold Validation

[67]:
```
from sklearn import model_selection

from sklearn.model_selection import KFold

kfold = model_selection.KFold(n_splits = 20, shuffle = True)
results_kfold = model_selection.cross_val_score(rf_clf, x_over, y_over, cv =␣
 ↪kfold)

print("Accuracy : ", results_kfold.mean() * 100)
```

```
Accuracy :  99.35195747881124
```

28 7- Logistic Regression Model

[68]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, max_iter = 1000)

classifier.fit(x_train ,y_train)
y_pred_lr = classifier.predict(x_test)
print("Accuracy score : ", accuracy_score(y_test, y_pred_lr)*100)
```

```
Accuracy score :  76.70951156812339
```

[ ]:
```

```

29 8- Prediction of Model

[73]:
```
age = 75
avg_glucose_level = 300
bmi = 36.6
gender_male = 1
ever_married_Yes = 1
work_type_Never_worked = 0
work_type_Private = 1
work_type_Self_employed = 0
work_type_children = 0
Residence_type_Urban = 1
smoking_status_formerly_smoked = 1
```

```python
smoking_status_never_smoked = 0
smoking_status_smokes = 0
hypertension = 0
heart_disease = 1


input_feature = ['age', 'hypertension', 'heart_disease', 'avg_glucose_level',␣
 ↪'bmi',
        'gender_Male', 'ever_married_Yes', 'work_type_Never_worked',
       'work_type_Private', 'work_type_Self-employed', 'work_type_children',
       'Residence_type_Urban', 'smoking_status_formerly smoked',
       'smoking_status_never smoked', 'smoking_status_smokes']

feature_values = {
    'age': age,
    'hypertension': hypertension,
    'heart_disease': heart_disease,
    'avg_glucose_level': avg_glucose_level,
    'bmi': bmi,
    'gender_Male': gender_male,
    'ever_married_Yes': ever_married_Yes,
    'work_type_Never_worked': work_type_Never_worked,
    'work_type_Private': work_type_Private,
    'work_type_Self-employed': work_type_Self_employed,
    'work_type_children': work_type_children,
    'Residence_type_Urban': Residence_type_Urban,
    'smoking_status_formerly smoked': smoking_status_formerly_smoked,
    'smoking_status_never smoked': smoking_status_never_smoked,
    'smoking_status_smokes': smoking_status_smokes
}

df = pd.DataFrame(feature_values, index=[0])

prediction = rf_clf.predict(df)[0]
print(prediction)
```

```
1
```

30 Thank You

[ ]:

```python

```