Heart stroke

December 24, 2022

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix , classification_report
import tensorflow as tf
from tensorflow import keras
import pickle
%matplotlib inline
```

0.1 Reading data from file

[2]:
```
df = pd.read_csv('Heart stroke.csv')
df.head(10)
```

[2]:
```
      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
5  56669    Male  81.0             0              0          Yes
6  53882    Male  74.0             1              1          Yes
7  10434  Female  69.0             0              0           No
8  27419  Female  59.0             0              0          Yes
9  60491  Female  78.0             0              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
5        Private          Urban            186.21  29.0  formerly smoked
6        Private          Rural             70.09  27.4     never smoked
7        Private          Urban             94.39  22.8     never smoked
8        Private          Rural             76.15   NaN          Unknown
9        Private          Urban             58.57  24.2          Unknown

   stroke
0       1
1       1
2       1
3       1
4       1
5       1
6       1
7       1
8       1
9       1
```

[3]:

```
df.describe()
```

[3]:

```
                 id          age  hypertension  heart_disease  \
count   5110.000000  5110.000000   5110.000000    5110.000000
mean   36517.829354    43.226614      0.097456       0.054012
std    21161.721625    22.612647      0.296607       0.226063
min       67.000000     0.080000      0.000000       0.000000
25%    17741.250000    25.000000      0.000000       0.000000
50%    36932.000000    45.000000      0.000000       0.000000
75%    54682.000000    61.000000      0.000000       0.000000
max    72940.000000    82.000000      1.000000       1.000000

       avg_glucose_level          bmi       stroke
count        5110.000000  4909.000000  5110.000000
mean          106.147677    28.893237     0.048728
std            45.283560     7.854067     0.215320
min            55.120000    10.300000     0.000000
25%            77.245000    23.500000     0.000000
50%            91.885000    28.100000     0.000000
75%           114.090000    33.100000     0.000000
max           271.740000    97.600000     1.000000
```

[4]:

```
df.isnull().sum()
```

[4]:

```
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]:
```
df.drop(columns=['id'],axis=1,inplace=True)
```

[6]:
```
df.info()
```

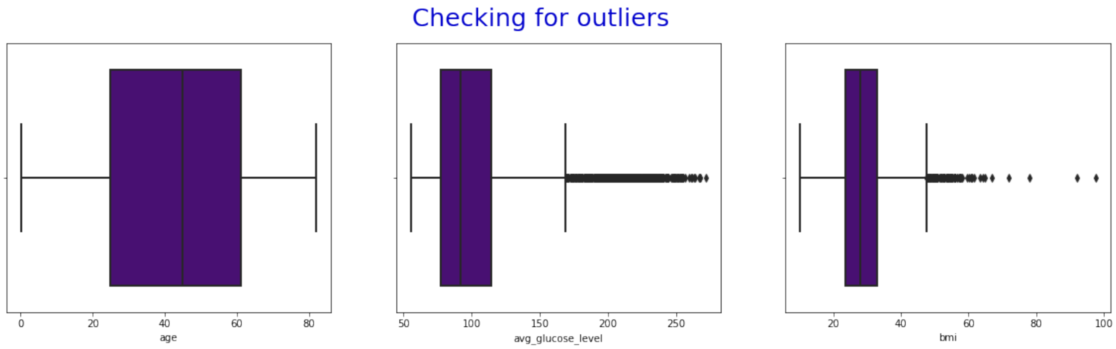
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             5110 non-null   object 
 1   age                5110 non-null   float64
 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                4909 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
```

[7]:
```
#from above we can see that hypertension, heart disease, stroke have only␣
↪classfication value so we only consider age ,bmi , glucose level as␣
↪numerical data
num_df = ['age','avg_glucose_level','bmi']
```

1 Filling missing value

[8]:
```
#checking for outliers with boxplot
fig, ax = plt.subplots(1, 3, figsize = (20, 5))
plt.suptitle('Checking for outliers', fontsize = 25, color = 'mediumblue')
i=0
for x in num_df:
    sns.boxplot(x = df[x], ax= ax[i], color= 'indigo', linewidth= 2)
    i = i+1
```

```
plt.show()
```

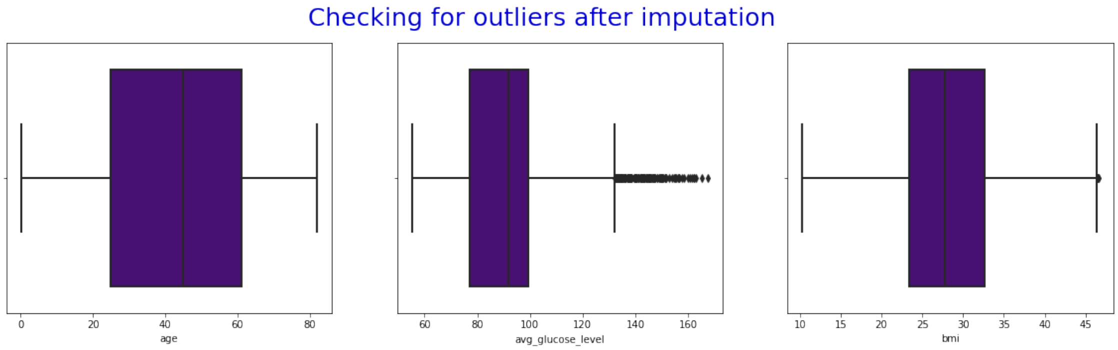


1.1 from here we can see that glucose and bmi have lots of outliers

2 dealing with avg_glucose_level and bmi column. I will apply median imputation to these columns

[10]:
```
col_out = ['avg_glucose_level','bmi']
for col in col_out:
    for i in df[col]:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)
        lqr = q3-q1
        lower_tail = q1 - 1.5*lqr
        uper_tail = q3 + 1.5*lqr
        if i > uper_tail or i<lower_tail:
            df[col] = df[col].replace(i,np.median(df[col]))
```

[11]:
```
fig, ax = plt.subplots(1, 3, figsize = (20, 5))
plt.suptitle('Checking for outliers after imputation', fontsize = 25, color =␣
 ↪'mediumblue')
i=0
for x in num_df:
    sns.boxplot(x = df[x], ax= ax[i], color= 'indigo', linewidth= 2)
    i = i+1
plt.show()
```



[12]:
```
df.describe()
```

[12]:
```
               age  hypertension  heart_disease  avg_glucose_level  \
count  5110.000000   5110.000000    5110.000000        5110.000000
mean     43.226614      0.097456       0.054012          90.000564
std      22.612647      0.296607       0.226063          18.777279
min       0.080000      0.000000       0.000000          55.120000
25%      25.000000      0.000000       0.000000          77.245000
50%      45.000000      0.000000       0.000000          91.880000
75%      61.000000      0.000000       0.000000          99.157500
max      82.000000      1.000000       1.000000         167.410000

               bmi       stroke
count  4789.000000  5110.000000
mean     28.274107     0.048728
std       6.793541     0.215320
min      10.300000     0.000000
25%      23.400000     0.000000
50%      27.800000     0.000000
75%      32.600000     0.000000
max      46.600000     1.000000
```

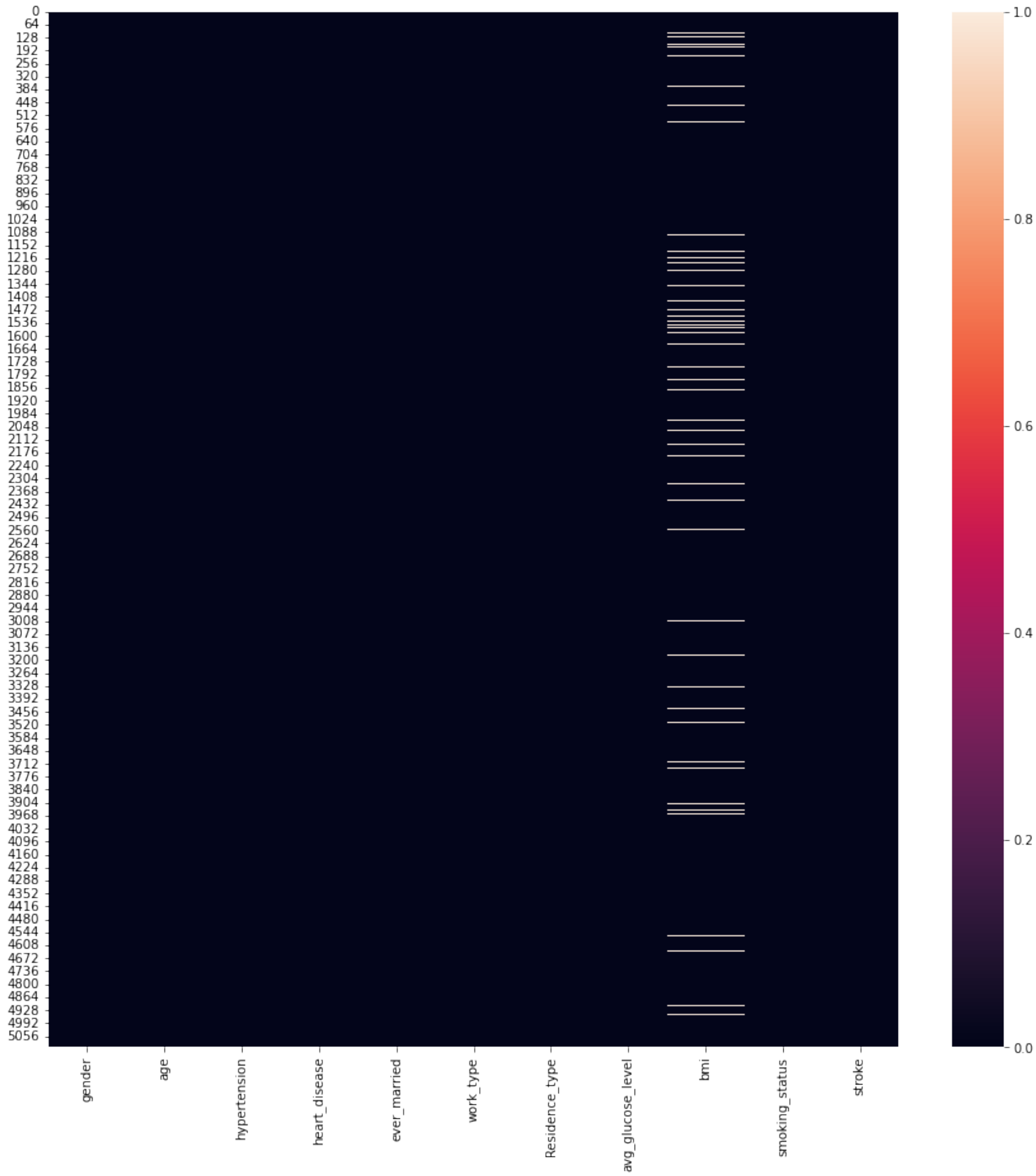
[13]:
```
df.isnull().sum()
```

[13]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  321
smoking_status         0
```

```
stroke                 0
dtype: int64
```

[14]:
```
plt.figure(figsize=(15,15))
sns.heatmap(df.isnull())
```

[14]: <AxesSubplot:>



2.0.1 white line against bmi indicates NAN values

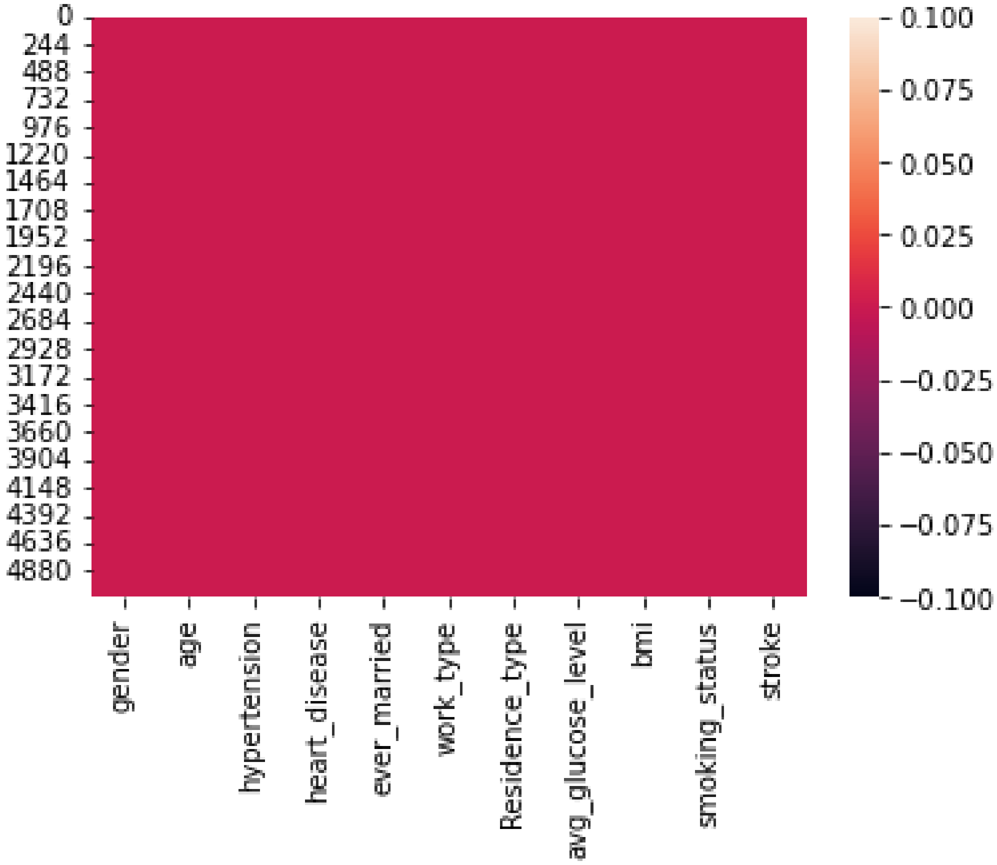
[15]:
```
df[df.bmi.isna()]
```

[15]:
```
      gender   age  hypertension  heart_disease ever_married      work_type  \
1     Female  61.0             0              0          Yes  Self-employed
8     Female  59.0             0              0          Yes        Private
13      Male  78.0             0              1          Yes        Private
19      Male  57.0             0              1           No       Govt_job
21    Female  52.0             1              0          Yes  Self-employed
…        …     …           …              …          …            …
5057  Female  49.0             0              0          Yes       Govt_job
5093  Female  45.0             1              0          Yes       Govt_job
5099    Male  40.0             0              0          Yes        Private
5103  Female  18.0             0              0           No        Private
5105  Female  80.0             1              0          Yes        Private

     Residence_type  avg_glucose_level  bmi smoking_status  stroke
1             Rural          91.882500  NaN   never smoked       1
8             Rural          76.150000  NaN        Unknown       1
13            Urban          91.880156  NaN        Unknown       1
19            Urban          91.880010  NaN        Unknown       1
21            Urban          91.880002  NaN   never smoked       1
…               …                  …   …            …     …
5057          Urban          69.920000  NaN   never smoked       0
5093          Rural          95.020000  NaN         smokes       0
5099          Rural          83.940000  NaN         smokes       0
5103          Urban          82.850000  NaN        Unknown       0
5105          Urban          83.750000  NaN   never smoked       0

[321 rows x 11 columns]
```

[16]:
```
df['bmi'].fillna(df["bmi"].mean(), inplace=True)
```

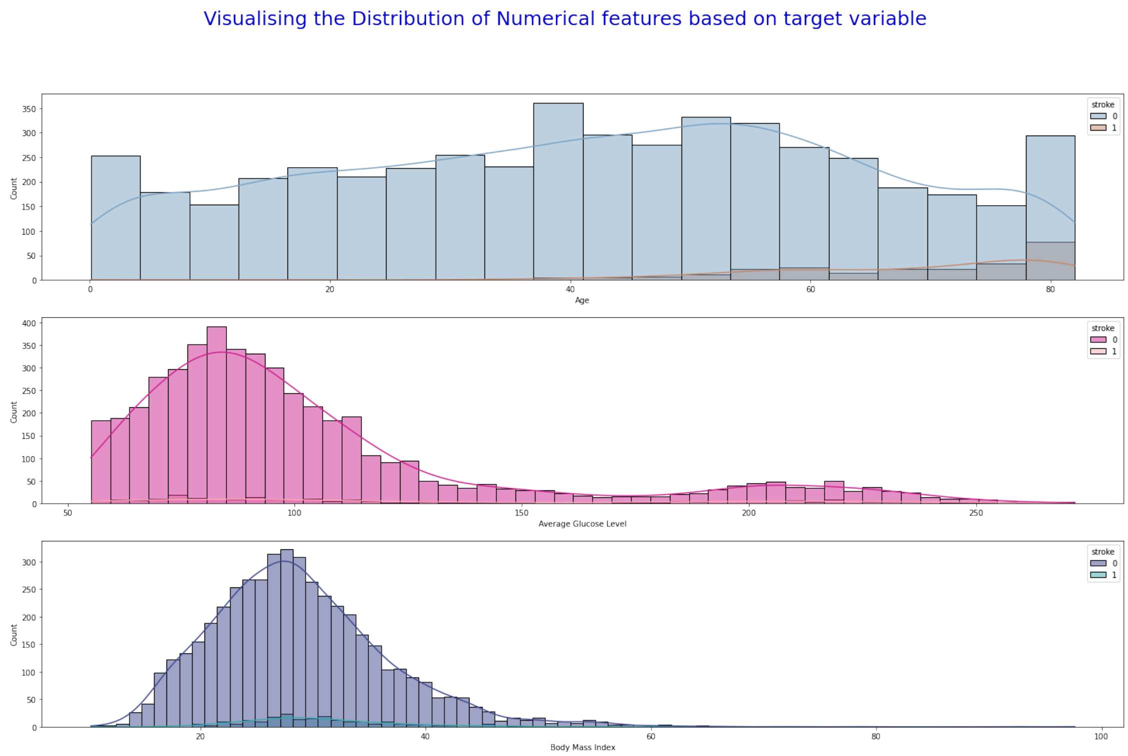
[17]:
```
sns.heatmap(df.isnull())
```

[17]: <AxesSubplot:>



3 EDA

[9]:
```
fig, ax = plt.subplots(3, 1, figsize = (25, 15))
plt.suptitle('Visualising the Distribution of Numerical features based on␣
↪target variable', fontsize = 25, color = 'mediumblue')
sns.histplot(x = df['age'], hue= df['stroke'], kde= True, ax= ax[0], palette =␣
↪'twilight_shifted')
ax[0].set(xlabel = 'Age')
sns.histplot(x = df['avg_glucose_level'], hue= df['stroke'], kde= True, ax=␣
↪ax[1], palette = 'RdPu_r')
ax[1].set(xlabel = 'Average Glucose Level')
sns.histplot(x = df['bmi'], hue= df['stroke'], kde= True, ax= ax[2], palette =␣
↪'mako')
ax[2].set(xlabel = 'Body Mass Index')
plt.show()
```

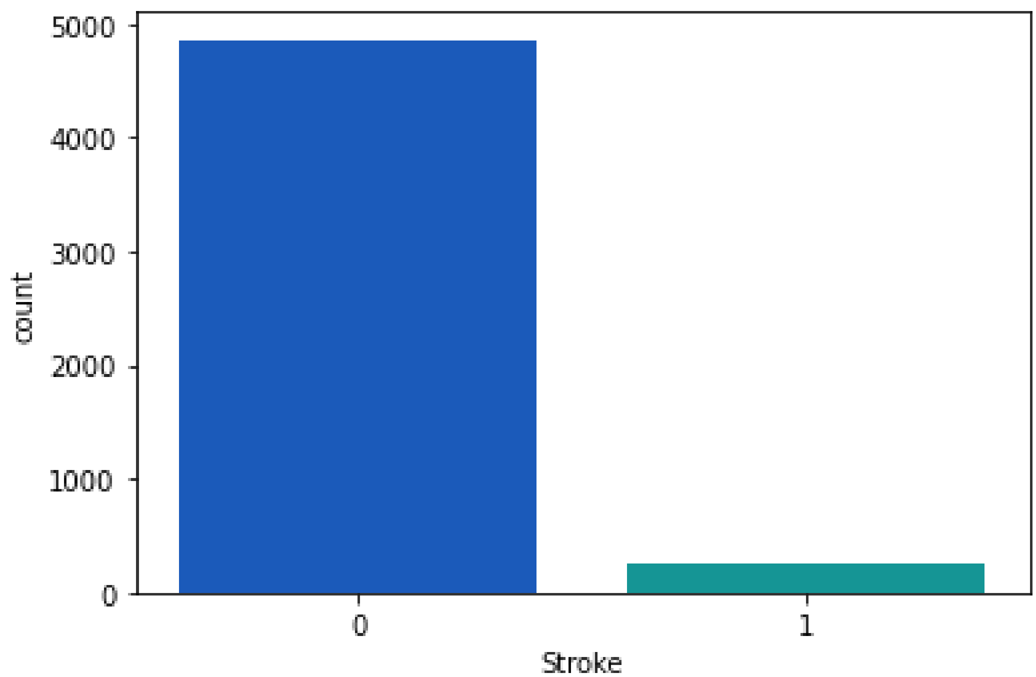


4 from this we can conclude that as ages increases chances of heart stroke also increases

[18]:
```
df['stroke'].value_counts()
```

[18]:
```
0    4861
1     249
Name: stroke, dtype: int64
```

[19]:
```
sns.countplot(x = df['stroke'], palette= 'winter')
plt.xlabel('Stroke');
```

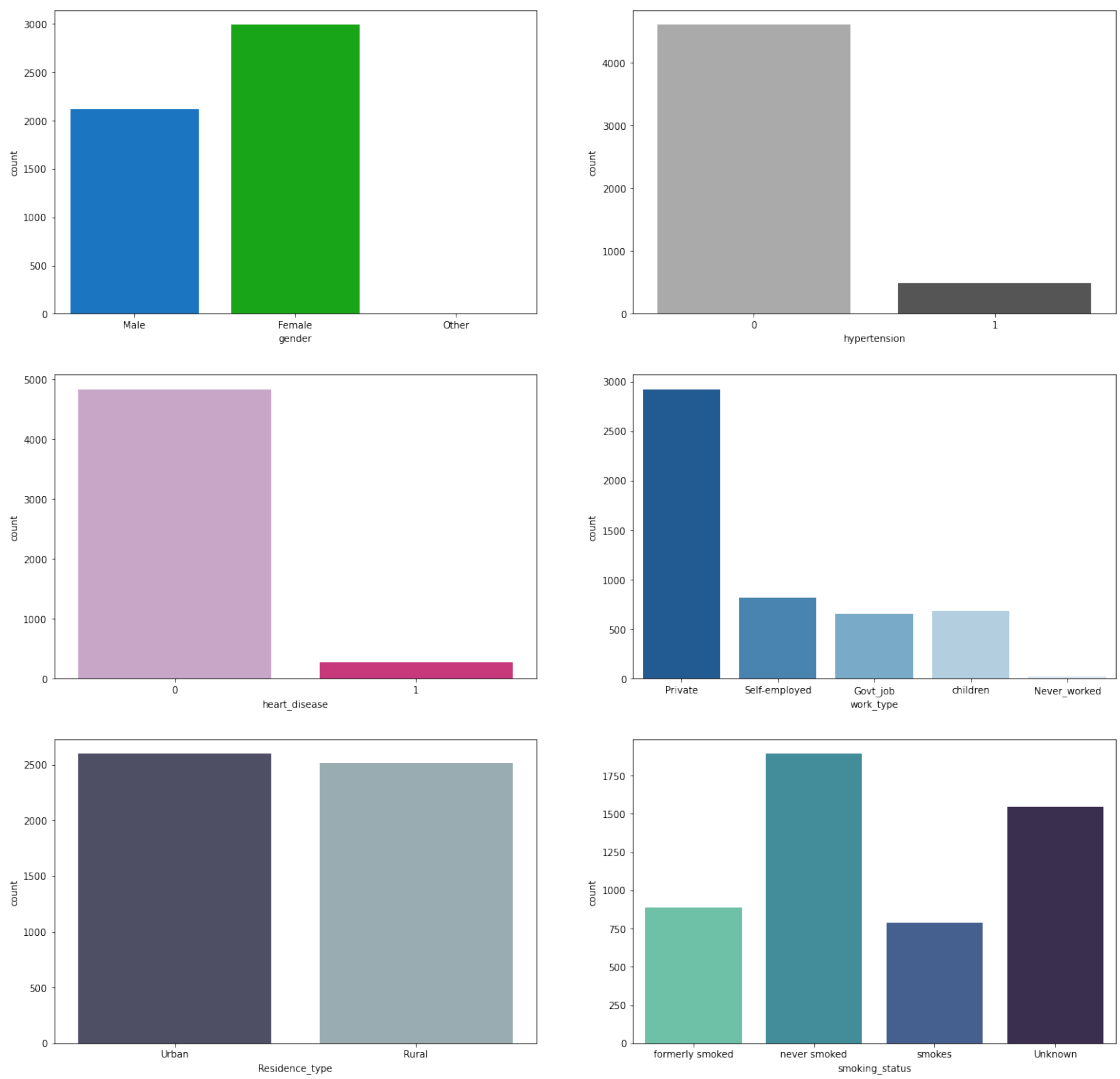


4.0.1 very less value of stroke patient which need to be handle during spliting of dataset

[20]:
```
fig, ax = plt.subplots(3, 2, figsize = (20, 20))
plt.suptitle('Visualising the classification␣
 ↪dataset',color='mediumblue',fontsize=25)
sns.countplot(x=df['gender'],ax=ax[0,0],palette='nipy_spectral')
sns.countplot(x=df['hypertension'],ax=ax[0,1],palette='gist_yarg')
sns.countplot(x=df['heart_disease'],ax=ax[1,0],palette='PuRd')
sns.countplot(x=df['work_type'],ax=ax[1,1],palette='Blues_r')
sns.countplot(x=df['Residence_type'],ax=ax[2,0],palette='bone')
sns.countplot(x=df['smoking_status'],ax=ax[2,1],palette='mako_r')
```

[20]: <AxesSubplot:xlabel='smoking_status', ylabel='count'>

Visualising the classification dataset




```
[21]: df.gender[df['gender']=='Other'].count()
```

[21]: 1

```
[22]: df[df['gender']=='Other'].index[0]
```

[22]: 3116

here we can see that only one value with other gender. we simply drop this value to improve our model accuracy

[23]:
```
df.drop(df[df['gender']=='Other'].index,axis = 0,inplace=True)
```

[24]:
```
df.gender[df['gender']=='Other'].count()
```

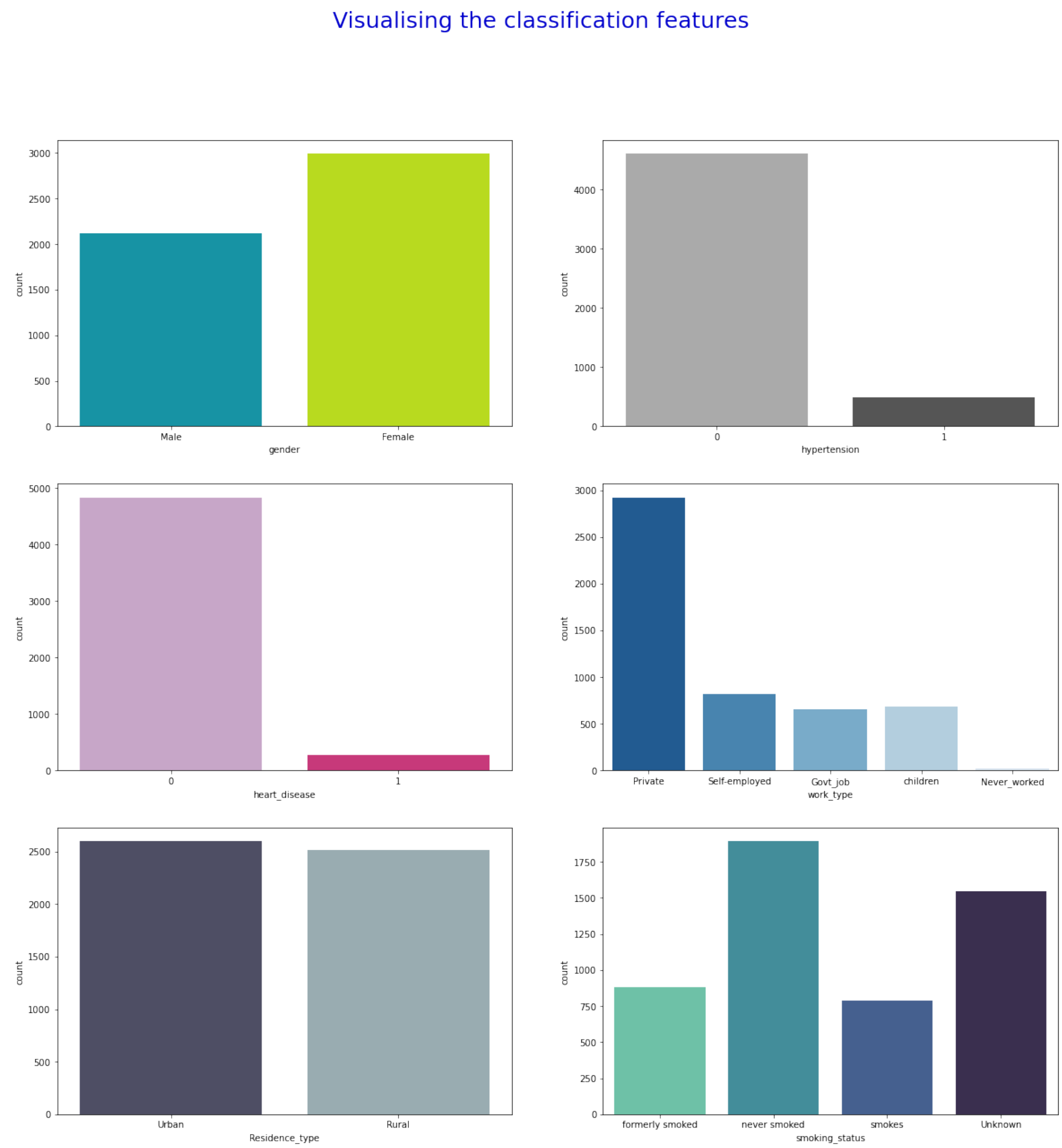
[24]: 0

[25]:
```
df.shape
```

[25]: (5109, 11)

[26]:
```
fig, ax = plt.subplots(3, 2, figsize = (20, 20))
plt.suptitle('Visualising the classification␣
  ↪features',color='mediumblue',fontsize=25)
sns.countplot(x=df['gender'],ax=ax[0,0],palette='nipy_spectral')
sns.countplot(x=df['hypertension'],ax=ax[0,1],palette='gist_yarg')
sns.countplot(x=df['heart_disease'],ax=ax[1,0],palette='PuRd')
sns.countplot(x=df['work_type'],ax=ax[1,1],palette='Blues_r')
sns.countplot(x=df['Residence_type'],ax=ax[2,0],palette='bone')
sns.countplot(x=df['smoking_status'],ax=ax[2,1],palette='mako_r')
```

[26]: <AxesSubplot:xlabel='smoking_status', ylabel='count'>

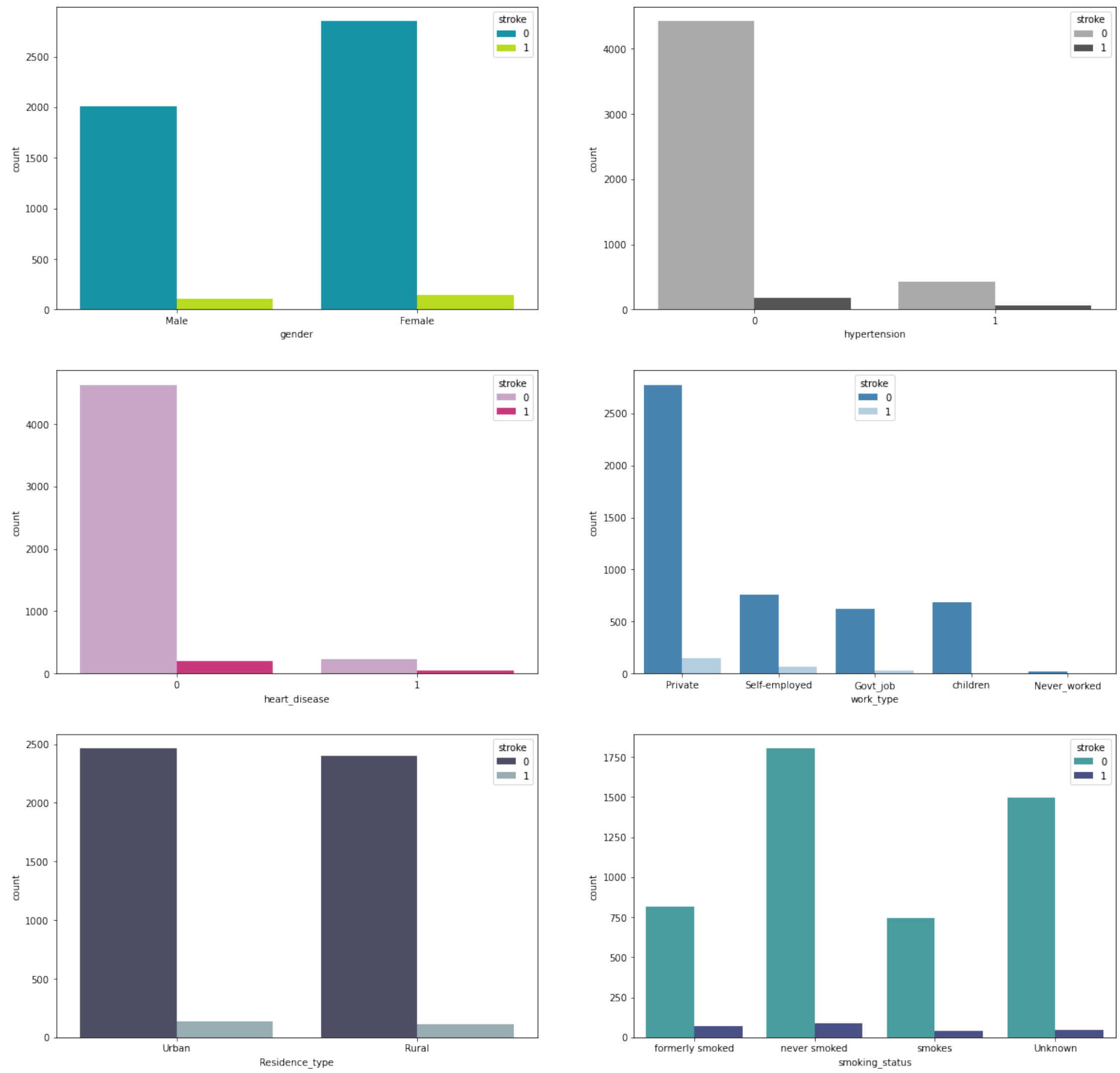


[27]:
```
fig, ax = plt.subplots(3, 2, figsize = (20, 20))
plt.suptitle('Visualising the classification columns on target␣
↪variable',color='mediumblue',fontsize=25)
sns.
↪countplot(x=df['gender'],ax=ax[0,0],palette='nipy_spectral',hue=df['stroke'])
sns.
↪countplot(x=df['hypertension'],ax=ax[0,1],palette='gist_yarg',hue=df['stroke'])
sns.countplot(x=df['heart_disease'],ax=ax[1,0],palette='PuRd',hue=df['stroke'])
sns.countplot(x=df['work_type'],ax=ax[1,1],palette='Blues_r',hue=df['stroke'])
sns.countplot(x=df['Residence_type'],ax=ax[2,0],palette='bone',hue=df['stroke'])
```

```
sns.
↪countplot(x=df['smoking_status'],ax=ax[2,1],palette='mako_r',hue=df['stroke'])
```

[27]: <AxesSubplot:xlabel='smoking_status', ylabel='count'>

Visualising the classification columns on target variable

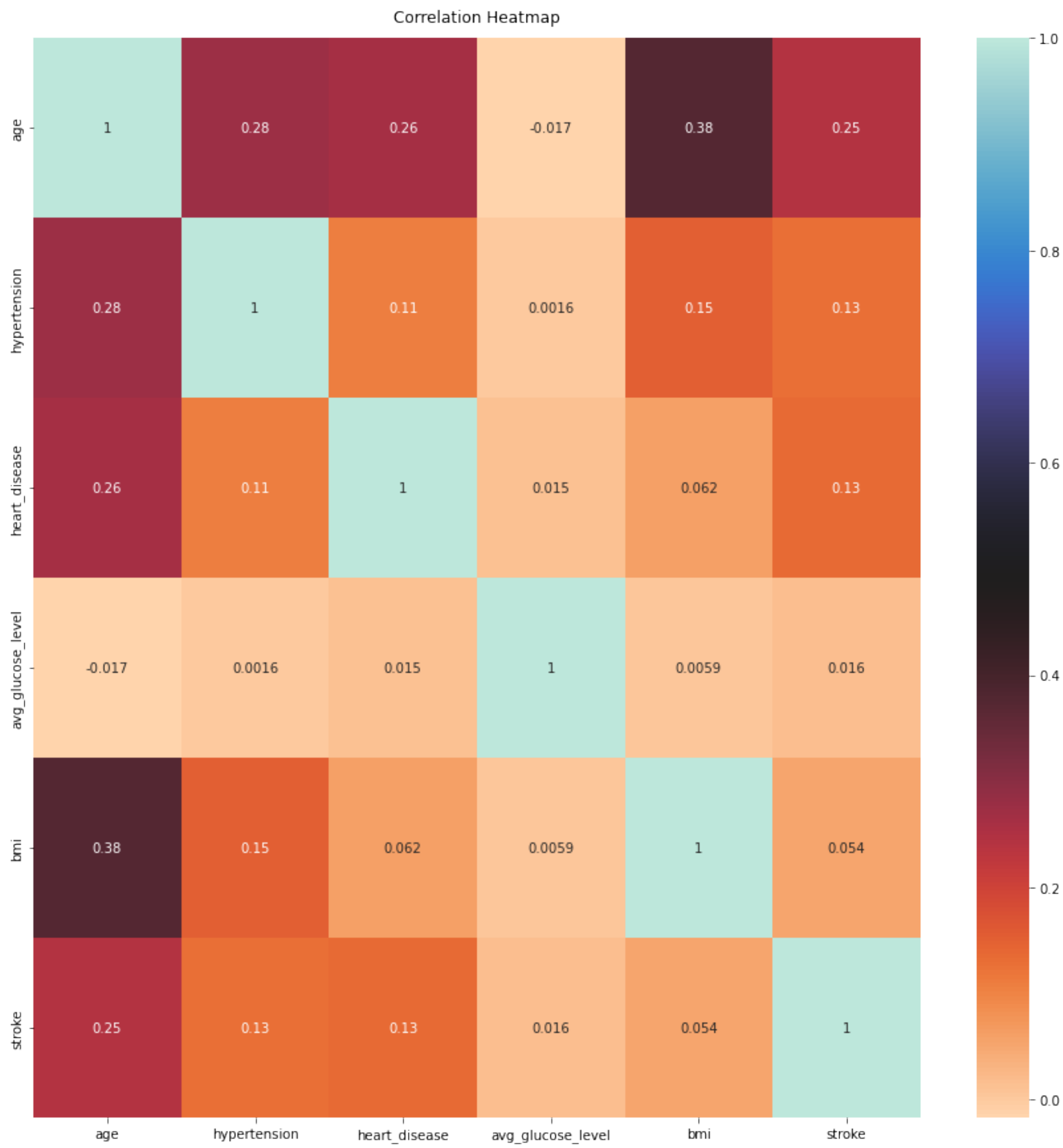


[28]:
```
#getting unique value for classifiaction features

for col in df:
    if col not in num_df:
        print(f'{col} : {df[col].unique()}')
```

```
gender : ['Male' 'Female']
hypertension : [0 1]
heart_disease : [1 0]
ever_married : ['Yes' 'No']
work_type : ['Private' 'Self-employed' 'Govt_job' 'children' 'Never_worked']
Residence_type : ['Urban' 'Rural']
smoking_status : ['formerly smoked' 'never smoked' 'smokes' 'Unknown']
stroke : [1 0]
```

4.1 Creating a correlation matrix to understand the faeature between various column

[29]:
```
plt.figure(figsize=(15,15))
heatmap = sns.heatmap(df.corr(),annot=True,cmap="icefire_r");
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



4.1.1 Encoding

[30]:
```
#replacing male = 1, Yes =1, Urban = 1

df['gender'].replace({'Male':1,'Female':0},inplace=True)
df.replace({'Yes':1,'No':0},inplace=True)
df['Residence_type'].replace({'Urban':1,'Rural':0},inplace=True)
```

[31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 5109 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5109 non-null   int64
 1   age                5109 non-null   float64
 2   hypertension       5109 non-null   int64
 3   heart_disease      5109 non-null   int64
 4   ever_married       5109 non-null   int64
 5   work_type          5109 non-null   object
 6   Residence_type     5109 non-null   int64
 7   avg_glucose_level  5109 non-null   float64
 8   bmi                5109 non-null   float64
 9   smoking_status     5109 non-null   object
 10  stroke             5109 non-null   int64
dtypes: float64(3), int64(6), object(2)
memory usage: 479.0+ KB
```

[32]:

```
#Doing one hot encoding for work_type,smoking_Status
df_encoded = pd.get_dummies(df,drop_first=True)
```

[33]:

```
df_encoded
```

[33]:

```
      gender   age  hypertension  heart_disease  ever_married  Residence_type  \
0          1  67.0             0              1             1               1
1          0  61.0             0              0             1               0
2          1  80.0             0              1             1               0
3          0  49.0             0              0             1               1
4          0  79.0             1              0             1               0
…        …    …            …              …             …               …
5105       0  80.0             1              0             1               1
5106       0  81.0             0              0             1               1
5107       0  35.0             0              0             1               0
5108       1  51.0             0              0             1               0
5109       0  44.0             0              0             1               1

      avg_glucose_level        bmi  stroke  work_type_Never_worked  \
0             91.885000  36.600000       1                       0
1             91.882500  28.274107       1                       0
2            105.920000  32.500000       1                       0
3             91.881250  34.400000       1                       0
4             91.880625  24.000000       1                       0
…                   …          …      …                       …
5105          83.750000  28.274107       0                       0
5106         125.200000  40.000000       0                       0
5107          82.990000  30.600000       0                       0
5108          91.880000  25.600000       0                       0
```

```
5109          85.280000  26.200000       0                       0

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

      smoking_status_formerly smoked  smoking_status_never smoked  \
0                                  1                            0
1                                  0                            1
2                                  0                            1
3                                  0                            0
4                                  0                            1
…                              …                          …
5105                               0                            1
5106                               0                            1
5107                               0                            1
5108                               1                            0
5109                               0                            0

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

[5109 rows x 16 columns]
```

4.2 Scaling

[34]:
```
scaler = StandardScaler()
df_encoded[num_df]=scaler.fit_transform(df_encoded[num_df])
```

[35]:
```
df_encoded.describe()
```

[35]:

	gender	age	hypertension	heart_disease	ever_married \
count	5109.000000	5.109000e+03	5109.000000	5109.000000	5109.000000
mean	0.413975	3.077275e-16	0.097475	0.054022	0.656293
std	0.492592	1.000098e+00	0.296633	0.226084	0.474991
min	0.000000	-1.908332e+00	0.000000	0.000000	0.000000
25%	0.000000	-8.062312e-01	0.000000	0.000000	0.000000
50%	0.000000	7.827984e-02	0.000000	0.000000	1.000000
75%	1.000000	7.858887e-01	0.000000	0.000000	1.000000
max	1.000000	1.714625e+00	1.000000	1.000000	1.000000

	Residence_type	avg_glucose_level	bmi	stroke \
count	5109.000000	5.109000e+03	5.109000e+03	5109.000000
mean	0.508123	-5.001219e-16	1.075237e-16	0.048738
std	0.499983	1.000098e+00	1.000098e+00	0.215340
min	0.000000	-1.858506e+00	-2.733403e+00	0.000000
25%	0.000000	-6.795556e-01	-6.805289e-01	0.000000
50%	1.000000	1.007264e-01	-1.748375e-04	0.000000
75%	1.000000	4.882025e-01	5.968151e-01	0.000000
max	1.000000	4.126321e+00	2.786548e+00	1.000000

	work_type_Never_worked	work_type_Private	work_type_Self-employed \
count	5109.000000	5109.000000	5109.000000
mean	0.004306	0.572323	0.160305
std	0.065486	0.494790	0.366925
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000

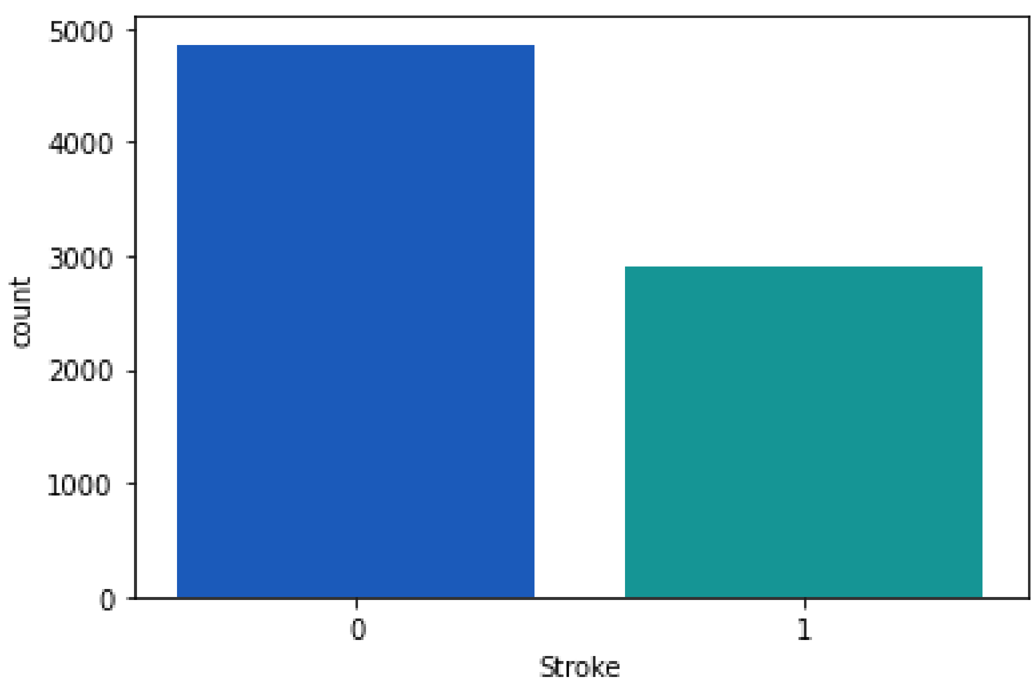
	work_type_children	smoking_status_formerly smoked \
count	5109.000000	5109.000000
mean	0.134469	0.173028
std	0.341188	0.378308
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1.000000	1.000000

```
       smoking_status_never smoked  smoking_status_smokes
count                  5109.000000            5109.000000
mean                      0.370327               0.154433
std                       0.482939               0.361399
min                       0.000000               0.000000
25%                       0.000000               0.000000
50%                       0.000000               0.000000
75%                       1.000000               0.000000
max                       1.000000               1.000000
```

4.3 data splitting

[36]:
```python
#Doing over sampling for minority classes
sampling_strategy = 0.6
oversample = RandomOverSampler(sampling_strategy=sampling_strategy)
X=df_encoded.drop(['stroke'],axis=1)
y=df_encoded['stroke']
X_over, y_over = oversample.fit_resample(X, y)
```

[37]:
```python
sns.countplot(x = y_over, palette= 'winter')
plt.xlabel('Stroke');
```



4.3.1 Splitting into train and test set

[38]:
```
X_train,X_test,y_train,y_test =␣
↪train_test_split(X_over,y_over,random_state=65,test_size=0.2)
```

[39]:
```
print('Train data set size is', X_train.shape, 'and test size is ', X_test.
↪shape)
```

```
Train data set size is (6220, 15) and test size is  (1556, 15)
```

5 ANN

[40]:
```
model = keras.Sequential([
    keras.layers.Dense(30,input_shape = (15,),activation = 'relu'),
    keras.layers.Dense(25,activation = 'relu'),
    keras.layers.Dense(20,activation = 'relu'),
    keras.layers.Dense(15,activation = 'relu'),
    keras.layers.Dense(1,activation = 'sigmoid')
])
model.compile(optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

[41]:
```
model.fit(X_train,y_train,epochs=250)
```

```
Epoch 1/250
195/195 [==============================] - 1s 884us/step - loss: 0.5338 -
accuracy: 0.7386
Epoch 2/250
195/195 [==============================] - 0s 2ms/step - loss: 0.4424 -
accuracy: 0.7805
Epoch 3/250
195/195 [==============================] - 0s 944us/step - loss: 0.4253 -
accuracy: 0.7912
Epoch 4/250
195/195 [==============================] - 0s 1ms/step - loss: 0.4075 -
accuracy: 0.7958
Epoch 5/250
195/195 [==============================] - 0s 2ms/step - loss: 0.3889 -
accuracy: 0.8092
Epoch 6/250
195/195 [==============================] - 0s 968us/step - loss: 0.3717 -
accuracy: 0.8207
Epoch 7/250
195/195 [==============================] - 0s 2ms/step - loss: 0.3583 -
accuracy: 0.8257
Epoch 8/250
195/195 [==============================] - 0s 862us/step - loss: 0.3391 -
```

```
accuracy: 0.8383
Epoch 9/250
195/195 [==============================] - 0s 2ms/step - loss: 0.3281 -
accuracy: 0.8502
Epoch 10/250
195/195 [==============================] - 0s 859us/step - loss: 0.3157 -
accuracy: 0.8519
Epoch 11/250
195/195 [==============================] - 0s 874us/step - loss: 0.3038 -
accuracy: 0.8627
Epoch 12/250
195/195 [==============================] - 0s 905us/step - loss: 0.2908 -
accuracy: 0.8715
Epoch 13/250
195/195 [==============================] - 0s 936us/step - loss: 0.2820 -
accuracy: 0.8720
Epoch 14/250
195/195 [==============================] - 0s 864us/step - loss: 0.2730 -
accuracy: 0.8814
Epoch 15/250
195/195 [==============================] - 0s 920us/step - loss: 0.2618 -
accuracy: 0.8876
Epoch 16/250
195/195 [==============================] - 0s 888us/step - loss: 0.2531 -
accuracy: 0.8915
Epoch 17/250
195/195 [==============================] - 0s 920us/step - loss: 0.2455 -
accuracy: 0.8973
Epoch 18/250
195/195 [==============================] - 0s 905us/step - loss: 0.2349 -
accuracy: 0.9026
Epoch 19/250
195/195 [==============================] - 0s 1ms/step - loss: 0.2290 -
accuracy: 0.9085
Epoch 20/250
195/195 [==============================] - 0s 925us/step - loss: 0.2220 -
accuracy: 0.9095
Epoch 21/250
195/195 [==============================] - 0s 936us/step - loss: 0.2163 -
accuracy: 0.9130
Epoch 22/250
195/195 [==============================] - 0s 879us/step - loss: 0.2133 -
accuracy: 0.9154
Epoch 23/250
195/195 [==============================] - 0s 895us/step - loss: 0.2027 -
accuracy: 0.9212
Epoch 24/250
195/195 [==============================] - 0s 874us/step - loss: 0.1990 -
```

```
accuracy: 0.9198
Epoch 25/250
195/195 [==============================] - 0s 869us/step - loss: 0.1948 -
accuracy: 0.9233
Epoch 26/250
195/195 [==============================] - 0s 915us/step - loss: 0.1862 -
accuracy: 0.9302
Epoch 27/250
195/195 [==============================] - 0s 869us/step - loss: 0.1865 -
accuracy: 0.9296
Epoch 28/250
195/195 [==============================] - 0s 874us/step - loss: 0.1758 -
accuracy: 0.9330
Epoch 29/250
195/195 [==============================] - 0s 910us/step - loss: 0.1772 -
accuracy: 0.9317
Epoch 30/250
195/195 [==============================] - 0s 889us/step - loss: 0.1712 -
accuracy: 0.9323
Epoch 31/250
195/195 [==============================] - 0s 910us/step - loss: 0.1702 -
accuracy: 0.9336
Epoch 32/250
195/195 [==============================] - 0s 932us/step - loss: 0.1657 -
accuracy: 0.9362
Epoch 33/250
195/195 [==============================] - 0s 884us/step - loss: 0.1558 -
accuracy: 0.9434
Epoch 34/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1511 -
accuracy: 0.9447
Epoch 35/250
195/195 [==============================] - 0s 892us/step - loss: 0.1512 -
accuracy: 0.9412
Epoch 36/250
195/195 [==============================] - 0s 920us/step - loss: 0.1528 -
accuracy: 0.9431
Epoch 37/250
195/195 [==============================] - 0s 977us/step - loss: 0.1459 -
accuracy: 0.9458
Epoch 38/250
195/195 [==============================] - 0s 910us/step - loss: 0.1432 -
accuracy: 0.9481
Epoch 39/250
195/195 [==============================] - 0s 913us/step - loss: 0.1414 -
accuracy: 0.9479
Epoch 40/250
195/195 [==============================] - 0s 922us/step - loss: 0.1436 -
```

```
accuracy: 0.9473
Epoch 41/250
195/195 [==============================] - 0s 920us/step - loss: 0.1358 -
accuracy: 0.9495
Epoch 42/250
195/195 [==============================] - 0s 898us/step - loss: 0.1299 -
accuracy: 0.9531
Epoch 43/250
195/195 [==============================] - 0s 920us/step - loss: 0.1347 -
accuracy: 0.9502
Epoch 44/250
195/195 [==============================] - 0s 959us/step - loss: 0.1225 -
accuracy: 0.9564
Epoch 45/250
195/195 [==============================] - 0s 956us/step - loss: 0.1378 -
accuracy: 0.9473
Epoch 46/250
195/195 [==============================] - 0s 910us/step - loss: 0.1318 -
accuracy: 0.9523
Epoch 47/250
195/195 [==============================] - 0s 879us/step - loss: 0.1235 -
accuracy: 0.9543
Epoch 48/250
195/195 [==============================] - 0s 997us/step - loss: 0.1189 -
accuracy: 0.9576
Epoch 49/250
195/195 [==============================] - 0s 946us/step - loss: 0.1198 -
accuracy: 0.9569
Epoch 50/250
195/195 [==============================] - 0s 910us/step - loss: 0.1190 -
accuracy: 0.9584
Epoch 51/250
195/195 [==============================] - 0s 941us/step - loss: 0.1132 -
accuracy: 0.9592
Epoch 52/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1081 -
accuracy: 0.9613
Epoch 53/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1166 -
accuracy: 0.9566
Epoch 54/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1141 -
accuracy: 0.9605
Epoch 55/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1069 -
accuracy: 0.9635
Epoch 56/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1078 -
```

```
accuracy: 0.9633
Epoch 57/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1031 -
accuracy: 0.9653
Epoch 58/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1104 -
accuracy: 0.9622
Epoch 59/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1042 -
accuracy: 0.9643
Epoch 60/250
195/195 [==============================] - 0s 1ms/step - loss: 0.1015 -
accuracy: 0.9646
Epoch 61/250
195/195 [==============================] - 0s 931us/step - loss: 0.1036 -
accuracy: 0.9645
Epoch 62/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0947 -
accuracy: 0.9683
Epoch 63/250
195/195 [==============================] - 0s 910us/step - loss: 0.0997 -
accuracy: 0.9650
Epoch 64/250
195/195 [==============================] - 0s 915us/step - loss: 0.0942 -
accuracy: 0.9712
Epoch 65/250
195/195 [==============================] - 0s 876us/step - loss: 0.1000 -
accuracy: 0.9675
Epoch 66/250
195/195 [==============================] - 0s 879us/step - loss: 0.1013 -
accuracy: 0.9641
Epoch 67/250
195/195 [==============================] - 0s 951us/step - loss: 0.0946 -
accuracy: 0.9667
Epoch 68/250
195/195 [==============================] - 0s 943us/step - loss: 0.0896 -
accuracy: 0.9704
Epoch 69/250
195/195 [==============================] - 0s 890us/step - loss: 0.0982 -
accuracy: 0.9664
Epoch 70/250
195/195 [==============================] - 0s 856us/step - loss: 0.0866 -
accuracy: 0.9715
Epoch 71/250
195/195 [==============================] - 0s 874us/step - loss: 0.0810 -
accuracy: 0.9736
Epoch 72/250
195/195 [==============================] - 0s 848us/step - loss: 0.0886 -
```

```
accuracy: 0.9717
Epoch 73/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0806 -
accuracy: 0.9752
Epoch 74/250
195/195 [==============================] - 0s 915us/step - loss: 0.0803 -
accuracy: 0.9757
Epoch 75/250
195/195 [==============================] - 0s 901us/step - loss: 0.1007 -
accuracy: 0.9690
Epoch 76/250
195/195 [==============================] - 0s 853us/step - loss: 0.1019 -
accuracy: 0.9641
Epoch 77/250
195/195 [==============================] - 0s 868us/step - loss: 0.0878 -
accuracy: 0.9699
Epoch 78/250
195/195 [==============================] - 0s 853us/step - loss: 0.0828 -
accuracy: 0.9744
Epoch 79/250
195/195 [==============================] - 0s 864us/step - loss: 0.0761 -
accuracy: 0.9762
Epoch 80/250
195/195 [==============================] - 0s 856us/step - loss: 0.0810 -
accuracy: 0.9757
Epoch 81/250
195/195 [==============================] - 0s 874us/step - loss: 0.0728 -
accuracy: 0.9793
Epoch 82/250
195/195 [==============================] - 0s 874us/step - loss: 0.0788 -
accuracy: 0.9762
Epoch 83/250
195/195 [==============================] - 0s 874us/step - loss: 0.0834 -
accuracy: 0.9748
Epoch 84/250
195/195 [==============================] - 0s 855us/step - loss: 0.0766 -
accuracy: 0.9765
Epoch 85/250
195/195 [==============================] - 0s 879us/step - loss: 0.0736 -
accuracy: 0.9767
Epoch 86/250
195/195 [==============================] - 0s 869us/step - loss: 0.0712 -
accuracy: 0.9794
Epoch 87/250
195/195 [==============================] - 0s 889us/step - loss: 0.0689 -
accuracy: 0.9799
Epoch 88/250
195/195 [==============================] - 0s 869us/step - loss: 0.0895 -
```

```
accuracy: 0.9696
Epoch 89/250
195/195 [==============================] - 0s 853us/step - loss: 0.0948 -
accuracy: 0.9693
Epoch 90/250
195/195 [==============================] - 0s 874us/step - loss: 0.0775 -
accuracy: 0.9764
Epoch 91/250
195/195 [==============================] - 0s 848us/step - loss: 0.0792 -
accuracy: 0.9759
Epoch 92/250
195/195 [==============================] - 0s 884us/step - loss: 0.0637 -
accuracy: 0.9807
Epoch 93/250
195/195 [==============================] - 0s 879us/step - loss: 0.0766 -
accuracy: 0.9754
Epoch 94/250
195/195 [==============================] - 0s 843us/step - loss: 0.0712 -
accuracy: 0.9764
Epoch 95/250
195/195 [==============================] - 0s 848us/step - loss: 0.0710 -
accuracy: 0.9778
Epoch 96/250
195/195 [==============================] - 0s 905us/step - loss: 0.0568 -
accuracy: 0.9852
Epoch 97/250
195/195 [==============================] - 0s 879us/step - loss: 0.0630 -
accuracy: 0.9797
Epoch 98/250
195/195 [==============================] - 0s 890us/step - loss: 0.0595 -
accuracy: 0.9817
Epoch 99/250
195/195 [==============================] - 0s 889us/step - loss: 0.0660 -
accuracy: 0.9797
Epoch 100/250
195/195 [==============================] - 0s 869us/step - loss: 0.0812 -
accuracy: 0.9732
Epoch 101/250
195/195 [==============================] - 0s 838us/step - loss: 0.0584 -
accuracy: 0.9834
Epoch 102/250
195/195 [==============================] - 0s 879us/step - loss: 0.0544 -
accuracy: 0.9838
Epoch 103/250
195/195 [==============================] - 0s 874us/step - loss: 0.0576 -
accuracy: 0.9826
Epoch 104/250
195/195 [==============================] - 0s 884us/step - loss: 0.0646 -
```

```
accuracy: 0.9794
Epoch 105/250
195/195 [==============================] - 0s 853us/step - loss: 0.0653 -
accuracy: 0.9796
Epoch 106/250
195/195 [==============================] - 0s 874us/step - loss: 0.0520 -
accuracy: 0.9844
Epoch 107/250
195/195 [==============================] - 0s 853us/step - loss: 0.0578 -
accuracy: 0.9812
Epoch 108/250
195/195 [==============================] - 0s 900us/step - loss: 0.0822 -
accuracy: 0.9748
Epoch 109/250
195/195 [==============================] - 0s 944us/step - loss: 0.0775 -
accuracy: 0.9765
Epoch 110/250
195/195 [==============================] - 0s 905us/step - loss: 0.0572 -
accuracy: 0.9854
Epoch 111/250
195/195 [==============================] - 0s 864us/step - loss: 0.0483 -
accuracy: 0.9854
Epoch 112/250
195/195 [==============================] - 0s 869us/step - loss: 0.0524 -
accuracy: 0.9841
Epoch 113/250
195/195 [==============================] - 0s 848us/step - loss: 0.0755 -
accuracy: 0.9773
Epoch 114/250
195/195 [==============================] - 0s 915us/step - loss: 0.0540 -
accuracy: 0.9831
Epoch 115/250
195/195 [==============================] - 0s 889us/step - loss: 0.0501 -
accuracy: 0.9847
Epoch 116/250
195/195 [==============================] - 0s 915us/step - loss: 0.0485 -
accuracy: 0.9862
Epoch 117/250
195/195 [==============================] - 0s 843us/step - loss: 0.0706 -
accuracy: 0.9757
Epoch 118/250
195/195 [==============================] - 0s 848us/step - loss: 0.0817 -
accuracy: 0.9748
Epoch 119/250
195/195 [==============================] - 0s 920us/step - loss: 0.0594 -
accuracy: 0.9804
Epoch 120/250
195/195 [==============================] - 0s 884us/step - loss: 0.0491 -
```

```
accuracy: 0.9838
Epoch 121/250
195/195 [==============================] - 0s 843us/step - loss: 0.0444 -
accuracy: 0.9873
Epoch 122/250
195/195 [==============================] - 0s 930us/step - loss: 0.0551 -
accuracy: 0.9820
Epoch 123/250
195/195 [==============================] - 0s 879us/step - loss: 0.0433 -
accuracy: 0.9875
Epoch 124/250
195/195 [==============================] - 0s 848us/step - loss: 0.0441 -
accuracy: 0.9878
Epoch 125/250
195/195 [==============================] - 0s 848us/step - loss: 0.0432 -
accuracy: 0.9868
Epoch 126/250
195/195 [==============================] - 0s 864us/step - loss: 0.0719 -
accuracy: 0.9770
Epoch 127/250
195/195 [==============================] - 0s 859us/step - loss: 0.0493 -
accuracy: 0.9849
Epoch 128/250
195/195 [==============================] - 0s 874us/step - loss: 0.0497 -
accuracy: 0.9859
Epoch 129/250
195/195 [==============================] - 0s 864us/step - loss: 0.0594 -
accuracy: 0.9801
Epoch 130/250
195/195 [==============================] - 0s 828us/step - loss: 0.0470 -
accuracy: 0.9857
Epoch 131/250
195/195 [==============================] - 0s 958us/step - loss: 0.0467 -
accuracy: 0.9854
Epoch 132/250
195/195 [==============================] - 0s 895us/step - loss: 0.0390 -
accuracy: 0.9887
Epoch 133/250
195/195 [==============================] - 0s 910us/step - loss: 0.0469 -
accuracy: 0.9855
Epoch 134/250
195/195 [==============================] - 0s 937us/step - loss: 0.0449 -
accuracy: 0.9867
Epoch 135/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0419 -
accuracy: 0.9875
Epoch 136/250
195/195 [==============================] - 0s 874us/step - loss: 0.0455 -
```

```
accuracy: 0.9857
Epoch 137/250
195/195 [==============================] - 0s 920us/step - loss: 0.0421 -
accuracy: 0.9873
Epoch 138/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0778 -
accuracy: 0.9777
Epoch 139/250
195/195 [==============================] - 0s 925us/step - loss: 0.0406 -
accuracy: 0.9876
Epoch 140/250
195/195 [==============================] - 0s 853us/step - loss: 0.0350 -
accuracy: 0.9905
Epoch 141/250
195/195 [==============================] - 0s 915us/step - loss: 0.0471 -
accuracy: 0.9857
Epoch 142/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0362 -
accuracy: 0.9892
Epoch 143/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0398 -
accuracy: 0.9871
Epoch 144/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0418 -
accuracy: 0.9863
Epoch 145/250
195/195 [==============================] - 0s 2ms/step - loss: 0.0360 -
accuracy: 0.9899
Epoch 146/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0611 -
accuracy: 0.9785
Epoch 147/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0582 -
accuracy: 0.9818
Epoch 148/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0451 -
accuracy: 0.9865
Epoch 149/250
195/195 [==============================] - 0s 2ms/step - loss: 0.0482 -
accuracy: 0.9879
Epoch 150/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0552 -
accuracy: 0.9817
Epoch 151/250
195/195 [==============================] - 0s 925us/step - loss: 0.0347 -
accuracy: 0.9900
Epoch 152/250
195/195 [==============================] - 0s 931us/step - loss: 0.0392 -
```

```
accuracy: 0.9879
Epoch 153/250
195/195 [==============================] - 0s 925us/step - loss: 0.0356 -
accuracy: 0.9886
Epoch 154/250
195/195 [==============================] - 0s 941us/step - loss: 0.0325 -
accuracy: 0.9905
Epoch 155/250
195/195 [==============================] - 0s 864us/step - loss: 0.0402 -
accuracy: 0.9871
Epoch 156/250
195/195 [==============================] - 0s 855us/step - loss: 0.0364 -
accuracy: 0.9891
Epoch 157/250
195/195 [==============================] - 0s 905us/step - loss: 0.0468 -
accuracy: 0.9859
Epoch 158/250
195/195 [==============================] - 0s 843us/step - loss: 0.0442 -
accuracy: 0.9855
Epoch 159/250
195/195 [==============================] - 0s 853us/step - loss: 0.0326 -
accuracy: 0.9899
Epoch 160/250
195/195 [==============================] - 0s 869us/step - loss: 0.0398 -
accuracy: 0.9870
Epoch 161/250
195/195 [==============================] - 0s 848us/step - loss: 0.0324 -
accuracy: 0.9899
Epoch 162/250
195/195 [==============================] - 0s 879us/step - loss: 0.0507 -
accuracy: 0.9842
Epoch 163/250
195/195 [==============================] - 0s 910us/step - loss: 0.0300 -
accuracy: 0.9910
Epoch 164/250
195/195 [==============================] - 0s 859us/step - loss: 0.0242 -
accuracy: 0.9928
Epoch 165/250
195/195 [==============================] - 0s 915us/step - loss: 0.0544 -
accuracy: 0.9823
Epoch 166/250
195/195 [==============================] - 0s 850us/step - loss: 0.0766 -
accuracy: 0.9780
Epoch 167/250
195/195 [==============================] - 0s 838us/step - loss: 0.0348 -
accuracy: 0.9887
Epoch 168/250
195/195 [==============================] - 0s 853us/step - loss: 0.0301 -
```

```
accuracy: 0.9902
Epoch 169/250
195/195 [==============================] - 0s 878us/step - loss: 0.0326 -
accuracy: 0.9897
Epoch 170/250
195/195 [==============================] - 0s 838us/step - loss: 0.0329 -
accuracy: 0.9891
Epoch 171/250
195/195 [==============================] - 0s 853us/step - loss: 0.0299 -
accuracy: 0.9905
Epoch 172/250
195/195 [==============================] - 0s 864us/step - loss: 0.0575 -
accuracy: 0.9799
Epoch 173/250
195/195 [==============================] - 0s 838us/step - loss: 0.0338 -
accuracy: 0.9892
Epoch 174/250
195/195 [==============================] - 0s 859us/step - loss: 0.0558 -
accuracy: 0.9838
Epoch 175/250
195/195 [==============================] - 0s 886us/step - loss: 0.0400 -
accuracy: 0.9871
Epoch 176/250
195/195 [==============================] - 0s 854us/step - loss: 0.0352 -
accuracy: 0.9873
Epoch 177/250
195/195 [==============================] - 0s 838us/step - loss: 0.0284 -
accuracy: 0.9910
Epoch 178/250
195/195 [==============================] - 0s 848us/step - loss: 0.0272 -
accuracy: 0.9905
Epoch 179/250
195/195 [==============================] - 0s 843us/step - loss: 0.0273 -
accuracy: 0.9908
Epoch 180/250
195/195 [==============================] - 0s 874us/step - loss: 0.0247 -
accuracy: 0.9941
Epoch 181/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0313 -
accuracy: 0.9886
Epoch 182/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0336 -
accuracy: 0.9887
Epoch 183/250
195/195 [==============================] - 0s 963us/step - loss: 0.0439 -
accuracy: 0.9859
Epoch 184/250
195/195 [==============================] - 0s 985us/step - loss: 0.0631 -
```

```
accuracy: 0.9794
Epoch 185/250
195/195 [==============================] - 0s 848us/step - loss: 0.0513 -
accuracy: 0.9838
Epoch 186/250
195/195 [==============================] - 0s 900us/step - loss: 0.0379 -
accuracy: 0.9876
Epoch 187/250
195/195 [==============================] - 0s 833us/step - loss: 0.0291 -
accuracy: 0.9921
Epoch 188/250
195/195 [==============================] - 0s 853us/step - loss: 0.0268 -
accuracy: 0.9915
Epoch 189/250
195/195 [==============================] - 0s 843us/step - loss: 0.0207 -
accuracy: 0.9939
Epoch 190/250
195/195 [==============================] - 0s 848us/step - loss: 0.0206 -
accuracy: 0.9941
Epoch 191/250
195/195 [==============================] - 0s 843us/step - loss: 0.0295 -
accuracy: 0.9915
Epoch 192/250
195/195 [==============================] - 0s 892us/step - loss: 0.0443 -
accuracy: 0.9855
Epoch 193/250
195/195 [==============================] - 0s 833us/step - loss: 0.0331 -
accuracy: 0.9900
Epoch 194/250
195/195 [==============================] - 0s 885us/step - loss: 0.0352 -
accuracy: 0.9892
Epoch 195/250
195/195 [==============================] - 0s 853us/step - loss: 0.0549 -
accuracy: 0.9828
Epoch 196/250
195/195 [==============================] - 0s 881us/step - loss: 0.0338 -
accuracy: 0.9900
Epoch 197/250
195/195 [==============================] - 0s 833us/step - loss: 0.0281 -
accuracy: 0.9915
Epoch 198/250
195/195 [==============================] - 0s 910us/step - loss: 0.0239 -
accuracy: 0.9926
Epoch 199/250
195/195 [==============================] - 0s 972us/step - loss: 0.0249 -
accuracy: 0.9918
Epoch 200/250
195/195 [==============================] - 0s 859us/step - loss: 0.0322 -
```

```
accuracy: 0.9894
Epoch 201/250
195/195 [==============================] - 0s 859us/step - loss: 0.0227 -
accuracy: 0.9929
Epoch 202/250
195/195 [==============================] - 0s 843us/step - loss: 0.0240 -
accuracy: 0.9921
Epoch 203/250
195/195 [==============================] - 0s 848us/step - loss: 0.0468 -
accuracy: 0.9868
Epoch 204/250
195/195 [==============================] - 0s 900us/step - loss: 0.0342 -
accuracy: 0.9884
Epoch 205/250
195/195 [==============================] - 0s 869us/step - loss: 0.0356 -
accuracy: 0.9889
Epoch 206/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0405 -
accuracy: 0.9862
Epoch 207/250
195/195 [==============================] - 0s 848us/step - loss: 0.0277 -
accuracy: 0.9918
Epoch 208/250
195/195 [==============================] - 0s 2ms/step - loss: 0.0277 -
accuracy: 0.9912
Epoch 209/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0402 -
accuracy: 0.9870
Epoch 210/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0402 -
accuracy: 0.9857
Epoch 211/250
195/195 [==============================] - 0s 833us/step - loss: 0.0292 -
accuracy: 0.9908
Epoch 212/250
195/195 [==============================] - 0s 934us/step - loss: 0.0276 -
accuracy: 0.9913
Epoch 213/250
195/195 [==============================] - 0s 941us/step - loss: 0.0299 -
accuracy: 0.9916
Epoch 214/250
195/195 [==============================] - 0s 925us/step - loss: 0.0164 -
accuracy: 0.9961
Epoch 215/250
195/195 [==============================] - 0s 869us/step - loss: 0.0265 -
accuracy: 0.9918
Epoch 216/250
195/195 [==============================] - 0s 869us/step - loss: 0.0481 -
```

```
accuracy: 0.9846
Epoch 217/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0277 -
accuracy: 0.9912
Epoch 218/250
195/195 [==============================] - 0s 941us/step - loss: 0.0284 -
accuracy: 0.9913
Epoch 219/250
195/195 [==============================] - 0s 956us/step - loss: 0.0240 -
accuracy: 0.9928
Epoch 220/250
195/195 [==============================] - 0s 874us/step - loss: 0.0306 -
accuracy: 0.9907
Epoch 221/250
195/195 [==============================] - 0s 843us/step - loss: 0.0468 -
accuracy: 0.9867
Epoch 222/250
195/195 [==============================] - 0s 900us/step - loss: 0.0260 -
accuracy: 0.9912
Epoch 223/250
195/195 [==============================] - 0s 925us/step - loss: 0.0180 -
accuracy: 0.9944
Epoch 224/250
195/195 [==============================] - 0s 983us/step - loss: 0.0380 -
accuracy: 0.9878
Epoch 225/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0649 -
accuracy: 0.9817
Epoch 226/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0335 -
accuracy: 0.9905
Epoch 227/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0289 -
accuracy: 0.9910
Epoch 228/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0185 -
accuracy: 0.9944
Epoch 229/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0172 -
accuracy: 0.9947
Epoch 230/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0261 -
accuracy: 0.9916
Epoch 231/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0193 -
accuracy: 0.9947
Epoch 232/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0228 -
```

```
accuracy: 0.9923
Epoch 233/250
195/195 [==============================] - 0s 884us/step - loss: 0.0293 -
accuracy: 0.9910
Epoch 234/250
195/195 [==============================] - 0s 903us/step - loss: 0.0670 -
accuracy: 0.9830
Epoch 235/250
195/195 [==============================] - 0s 871us/step - loss: 0.0209 -
accuracy: 0.9932
Epoch 236/250
195/195 [==============================] - 0s 966us/step - loss: 0.0225 -
accuracy: 0.9928
Epoch 237/250
195/195 [==============================] - 0s 889us/step - loss: 0.0250 -
accuracy: 0.9924
Epoch 238/250
195/195 [==============================] - 0s 843us/step - loss: 0.0515 -
accuracy: 0.9863
Epoch 239/250
195/195 [==============================] - 0s 874us/step - loss: 0.0370 -
accuracy: 0.9887
Epoch 240/250
195/195 [==============================] - 0s 853us/step - loss: 0.0202 -
accuracy: 0.9945
Epoch 241/250
195/195 [==============================] - 0s 889us/step - loss: 0.0172 -
accuracy: 0.9955
Epoch 242/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0167 -
accuracy: 0.9957
Epoch 243/250
195/195 [==============================] - 0s 920us/step - loss: 0.0224 -
accuracy: 0.9936
Epoch 244/250
195/195 [==============================] - 0s 905us/step - loss: 0.0196 -
accuracy: 0.9932
Epoch 245/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0348 -
accuracy: 0.9870
Epoch 246/250
195/195 [==============================] - 0s 941us/step - loss: 0.0807 -
accuracy: 0.9767
Epoch 247/250
195/195 [==============================] - 0s 966us/step - loss: 0.0254 -
accuracy: 0.9926
Epoch 248/250
195/195 [==============================] - 0s 859us/step - loss: 0.0157 -
```

```
accuracy: 0.9957
Epoch 249/250
195/195 [==============================] - 0s 987us/step - loss: 0.0149 -
accuracy: 0.9952
Epoch 250/250
195/195 [==============================] - 0s 1ms/step - loss: 0.0229 -
accuracy: 0.9924
```

[41]: <keras.callbacks.History at 0x2300c1ffc70>

[42]:
```
model.evaluate(X_test,y_test)
```

```
49/49 [==============================] - 0s 879us/step - loss: 0.1563 -
accuracy: 0.9640
```

[42]: [0.15632732212543488, 0.9640102982521057]

[43]:
```
model.evaluate(X_train,y_train)
```

```
195/195 [==============================] - 0s 715us/step - loss: 0.0129 -
accuracy: 0.9968
```

[43]: [0.012905284762382507, 0.9967845678329468]

[44]:
```
y_train
```

[44]:
```
849     0
2575    0
187     1
588     0
1130    0
       ..
3399    0
2773    0
296     0
575     0
2165    0
Name: stroke, Length: 6220, dtype: int64
```

[45]:
```
y_test_predict = model.predict(X_test)
y_train_predict = model.predict(X_train)
```

```
49/49 [==============================] - 0s 737us/step
195/195 [==============================] - 0s 637us/step
```

[46]:
```
X_train
```

[46]:
```
      gender       age  hypertension  heart_disease  ever_married  \
849        0  1.183919             1              0             1
2575       1 -0.717780             0              0             0
```

```
187       0  1.714625            1             1             1
588       0 -0.850457            0             0             1
1130      1  1.626174            0             0             1
…      …       …           …            …           …
3399      0  1.714625            1             1             1
2773      1 -0.762006            0             0             1
296       0  1.007016            0             0             1
575       1  0.078280            0             0             1
2165      0 -1.381163            0             0             0

      Residence_type  avg_glucose_level       bmi  work_type_Never_worked  \
849                1           0.512187  0.916151                       0
2575               1          -0.926325  1.554823                       0
187                1           0.100726 -0.057063                       0
588                1          -0.000007 -0.589290                       0
1130               1          -0.597477 -0.650116                       0
…               …                  …       …                     …
3399               1          -0.895412  0.794499                       0
2773               1           0.226510  0.140621                       0
296                0           2.731513  0.034175                       0
575                0           0.531907  1.113835                       0
2165               1          -0.405071 -1.699363                       0

      work_type_Private  work_type_Self-employed  work_type_children  \
849                   0                        1                   0
2575                  1                        0                   0
187                   0                        0                   0
588                   1                        0                   0
1130                  1                        0                   0
…                  …                      …                 …
3399                  1                        0                   0
2773                  0                        0                   0
296                   1                        0                   0
575                   1                        0                   0
2165                  0                        0                   1

      smoking_status_formerly smoked  smoking_status_never smoked  \
849                                1                            0
2575                               0                            1
187                                1                            0
588                                1                            0
1130                               1                            0
…                               …                          …
3399                               0                            1
2773                               1                            0
296                                0                            1
575                                0                            1
```

```
2165                               0                            1

      smoking_status_smokes
849                       0
2575                      0
187                       0
588                       0
1130                      0
…                     …
3399                      0
2773                      0
296                       0
575                       0
2165                      0

[6220 rows x 15 columns]
```

[47]:

```
y_test_predict.shape
```

[47]: (1556, 1)

[48]:

```
y_test_pred = []
for ele in  y_test_predict:
    if ele >0.5:
        y_test_pred.append(1)
    else:
        y_test_pred.append(0)
```

[49]:

```
y_train_pred = []
for ele in  y_train_predict:
    if ele >0.5:
        y_train_pred.append(1)
    else:
        y_train_pred.append(0)
```

5.1 Checking the accuracy of the model

[50]:

```
print('Accuracy for test data:', accuracy_score(y_test, y_test_pred))
print('Accuracy for train data:', accuracy_score(y_train, y_train_pred))
```

```
Accuracy for test data: 0.9640102827763496
Accuracy for train data: 0.9967845659163987
```

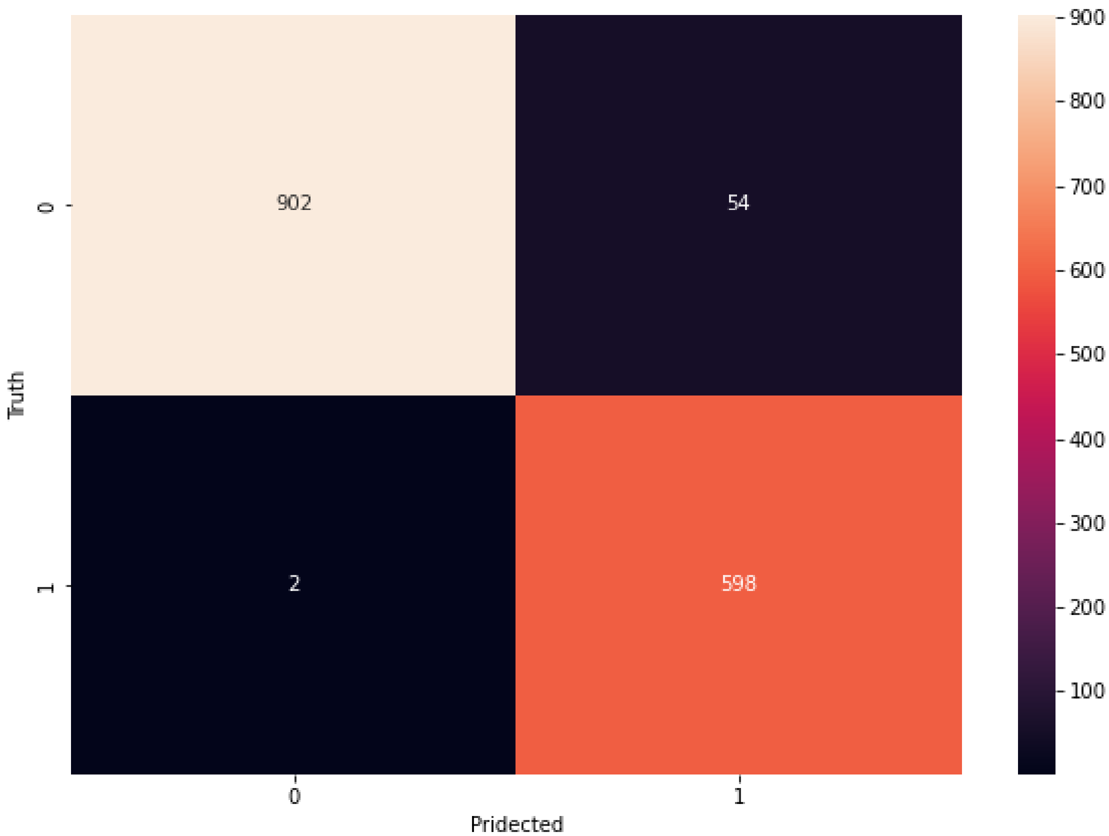
[51]:

```
print(classification_report(y_test,y_test_pred))
cm= tf.math.confusion_matrix(labels = y_test,predictions=y_test_pred)

plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,fmt='d')
```

```
plt.xlabel('Pridected')
plt.ylabel('Truth')
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       956
           1       0.92      1.00      0.96       600

    accuracy                           0.96      1556
   macro avg       0.96      0.97      0.96      1556
weighted avg       0.97      0.96      0.96      1556
```

[51]: Text(69.0, 0.5, 'Truth')



6 From here we conclude that the given model has recall for 1 has 100%. which means whenever any patient has chances of stroke our model will report it accurately.

[52]:
```
input_data = (1,49,0,0,0,0,104.86,31.9,0,1,0,0,0,0,1)

# changing the input_data to numpy array
input_data = np.asarray(input_data)


# reshape the array as we are predicting for one instance
input_data = input_data.reshape(1,-1)

#standarised the data
print(input_data[0,1],input_data[0,10],input_data[0,11])
lst=scaler.transform([[input_data[0,1],input_data[0,6],input_data[0,7]]])
lst
input_data[0,1],input_data[0,6],input_data[0,7] = lst[0,0],lst[0,1],lst[0,2]

print(input_data)

prediction = model.predict(input_data)
print(prediction)
if prediction[0] >= 0.5:
    print('The patient has Stroke')
else:
    print('The patient has not strokee')
#     1        1.051242        0        1        1        1        0.
 ↪100993        1.
 ↪265900        0        1        0        0        1        0        0
```

```
49.0 0.0 0.0
[[1.         0.25518205 0.         0.         0.         0.
  0.79253384 0.55119568 0.         1.         0.         0.
  0.         0.         1.        ]]
1/1 [==============================] - 0s 47ms/step
[[0.9999907]]
The patient has Stroke
```

```
C:\Users\01abn\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but StandardScaler was fitted with feature
names
  warnings.warn(
```

6.1 Saving the model

[53]:
```
#Saving the scaler and model
filename = 'heart_stroke.sav'
pickle.dump(model, open(filename, 'wb'))
scalerfile = 'scaler.sav'
pickle.dump(scaler, open(scalerfile, 'wb'))
```

```
Keras weights file (<HDF5 file "variables.h5" (mode r+)>) saving:
…layers\dense
…vars
…0
…1
…layers\dense_1
…vars
…0
…1
…layers\dense_2
…vars
…0
…1
…layers\dense_3
…vars
…0
…1
…layers\dense_4
…vars
…0
…1
…metrics\mean
…vars
…0
…1
…metrics\mean_metric_wrapper
…vars
…0
…1
…optimizer
…vars
…0
…1
…10
…11
…12
…13
…14
…15
…16
```

```
…17
…18
…19
…2
…20
…3
…4
…5
…6
…7
…8
…9
…vars
Keras model archive saving:
File Name                                             Modified             Size
config.json                                    2022-12-24 02:53:32         2672
metadata.json                                  2022-12-24 02:53:32           64
variables.h5                                   2022-12-24 02:53:32        56424
```

[54]:

```
# loading the saved model
load_model = pickle.load(open('heart_stroke.sav', 'rb'))
load_scaler = pickle.load(open('scaler.sav','rb'))
```

```
Keras model archive loading:
File Name                                             Modified             Size
config.json                                    2022-12-24 02:53:32         2672
metadata.json                                  2022-12-24 02:53:32           64
variables.h5                                   2022-12-24 02:53:32        56424
Keras weights file (<HDF5 file "variables.h5" (mode r)>) loading:
…layers\dense
…vars
…0
…1
…layers\dense_1
…vars
…0
…1
…layers\dense_2
…vars
…0
…1
…layers\dense_3
…vars
…0
…1
…layers\dense_4
…vars
…0
```

```
…1
…metrics\mean
…vars
…0
…1
…metrics\mean_metric_wrapper
…vars
…0
…1
…optimizer
…vars
…0
…1
…10
…11
…12
…13
…14
…15
…16
…17
…18
…19
…2
…20
…3
…4
…5
…6
…7
…8
…9
…vars
```

[55]:

```
input_data = (1,49,0,0,0,0,104.86,31.9,0,1,0,0,0,0,1)

# changing the input_data to numpy array
input_data = np.asarray(input_data)


# reshape the array as we are predicting for one instance
input_data = input_data.reshape(1,-1)

#standarised the data
print(input_data[0,1],input_data[0,10],input_data[0,11])
lst=load_scaler.transform([[input_data[0,1],input_data[0,6],input_data[0,7]]])
# lst
```

```
input_data[0,1],input_data[0,6],input_data[0,7] = lst[0,0],lst[0,1],lst[0,2]

# print(input_data)

prediction = load_model.predict(input_data)
print(prediction)
if prediction[0] >= 0.5:
    print('The patient has Stroke')
else:
    print('The patient has not strokee')
```

```
49.0 0.0 0.0
1/1 [==============================] - 0s 55ms/step
[[0.9999907]]
The patient has Stroke
```

```
C:\Users\01abn\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but StandardScaler was fitted with feature
names
  warnings.warn(
```

[56]:

```
for col in X:
    print(col)
```

```
gender
age
hypertension
heart_disease
ever_married
Residence_type
avg_glucose_level
bmi
work_type_Never_worked
work_type_Private
work_type_Self-employed
work_type_children
smoking_status_formerly smoked
smoking_status_never smoked
smoking_status_smokes
```

[ ]: