Heart Stroke Prediction

In [ ]:

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [ ]:

Importing Data Set

In [2]:
```
data = pd.read_csv('Heart Stroke Merged Data.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	ID	Gender	Age	Hypertension	Heart_Disease	Ever_Married	Work_Type	Residence_Type	A
0	10001	Male	3.0	0	0	No	children	Rural	
1	10002	Male	58.0	1	0	Yes	Private	Urban	
2	10003	Female	8.0	0	0	No	Private	Urban	
3	10004	Female	70.0	0	0	Yes	Private	Rural	
4	10005	Male	14.0	0	0	No	Never_worked	Rural	

In [ ]:

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48510 entries, 0 to 48509
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   ID                 48510 non-null  int64  
 1   Gender             48510 non-null  object 
 2   Age                48510 non-null  float64
 3   Hypertension       48510 non-null  int64  
 4   Heart_Disease      48510 non-null  int64  
 5   Ever_Married       48510 non-null  object 
 6   Work_Type          48510 non-null  object 
 7   Residence_Type     48510 non-null  object 
 8   Avg_Glucose_Level  48510 non-null  float64
 9   BMI                46847 non-null  float64
 10  Smoking_Status     33674 non-null  object 
 11  Stroke             48510 non-null  int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 4.4+ MB
```

In [ ]:

Statistical Description

In [5]:
```
data.describe()
```

Out[5]:

	ID	Age	Hypertension	Heart_Disease	Avg_Glucose_Level	BMI	
count	48510.000000	48510.000000	48510.000000	48510.000000	48510.000000	46847.000000	48
mean	34255.500000	42.324152	0.093981	0.048196	104.658132	28.635238	
std	14003.775116	22.531358	0.291805	0.214183	43.348186	7.779286	
min	10001.000000	0.080000	0.000000	0.000000	55.000000	10.100000	
25%	22128.250000	24.000000	0.000000	0.000000	77.520000	23.300000	
50%	34255.500000	44.000000	0.000000	0.000000	91.600000	27.800000	
75%	46382.750000	60.000000	0.000000	0.000000	112.227500	32.900000	
max	58510.000000	82.000000	1.000000	1.000000	291.050000	97.600000	

In [ ]:

Checking the Null Values

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
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                 1663
Smoking_Status     14836
Stroke                 0
dtype: int64
```

In [ ]:
```
```

Droping the Unnecessary Column

In [7]:
```
data.drop(['ID'], axis = 1, inplace = True)
```

In [8]:
```
data.head()
```

Out[8]:

	Gender	Age	Hypertension	Heart_Disease	Ever_Married	Work_Type	Residence_Type	Avg_Gluc
0	Male	3.0	0	0	No	children	Rural	
1	Male	58.0	1	0	Yes	Private	Urban	
2	Female	8.0	0	0	No	Private	Urban	
3	Female	70.0	0	0	Yes	Private	Rural	
4	Male	14.0	0	0	No	Never_worked	Rural	

In [ ]:
```
```

Filling the Null Values

In [9]:
```
data['BMI'].fillna(data['BMI'].median(), inplace = True)
```

In [10]:
```
data['Smoking_Status'].fillna(data['Smoking_Status'].mode()[0], inplace = True)
```

In [11]:
```
data.isna().sum()
```

Out[11]:
```
Gender               0
Age                  0
Hypertension         0
Heart_Disease        0
Ever_Married         0
Work_Type            0
Residence_Type       0
Avg_Glucose_Level    0
BMI                  0
Smoking_Status       0
Stroke               0
dtype: int64
```

In [12]:
```
data['Age'] = data['Age'].astype(int)
```

In [13]:
```
data.head()
```

Out[13]:

	Gender	Age	Hypertension	Heart_Disease	Ever_Married	Work_Type	Residence_Type	Avg_Gluc
0	Male	3	0	0	No	children	Rural	
1	Male	58	1	0	Yes	Private	Urban	
2	Female	8	0	0	No	Private	Urban	
3	Female	70	0	0	Yes	Private	Rural	
4	Male	14	0	0	No	Never_worked	Rural	

In [ ]:
```

```

Exporting the data

In [14]:
```
data.to_csv("E:\FireBlaze/Cleaned_data.csv")
```

In [ ]:
```

```

giving the numerical labels to the Categorical column

In [15]:
```
from sklearn.preprocessing import LabelEncoder
```

In [16]:
```
LE=LabelEncoder()
```

In [17]:
```
data.head(11)
```

Out[17]:

	Gender	Age	Hypertension	Heart_Disease	Ever_Married	Work_Type	Residence_Type	Avg_Glu
0	Male	3	0	0	No	children	Rural	
1	Male	58	1	0	Yes	Private	Urban	
2	Female	8	0	0	No	Private	Urban	
3	Female	70	0	0	Yes	Private	Rural	
4	Male	14	0	0	No	Never_worked	Rural	
5	Female	47	0	0	Yes	Private	Urban	
6	Female	52	0	0	Yes	Private	Urban	
7	Female	75	0	1	Yes	Self-employed	Rural	
8	Female	32	0	0	Yes	Private	Rural	
9	Female	74	1	0	Yes	Self-employed	Urban	
10	Female	79	0	0	Yes	Govt_job	Urban	

In [18]:
```
data['Gender'] = LE.fit_transform(data['Gender'].astype(str))

#1 - Male
#0 - Female
```

In [20]:
```
data['Ever_Married'] = LE.fit_transform(data['Ever_Married'].astype(str))

#1 - Yes
#0 - No
```

In [22]:
```
data['Work_Type'] = LE.fit_transform(data['Work_Type'].astype(str))

#0 - Govt_job
#1 - Never_worked
#2 - Private
#3 - Self_employed
#4 - Children
```

In [24]:
```
data['Residence_Type'] = LE.fit_transform(data['Residence_Type'].astype(str))

#0 - Rural
#1 - Urban
```

In [26]:
```
data['Smoking_Status'] = LE.fit_transform(data['Smoking_Status'].astype(str))

#0 - Formerly Smoked
#1 - never smoked
#2 - smokes
```

In [29]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48510 entries, 0 to 48509
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             48510 non-null  int32  
 1   Age                48510 non-null  int32  
 2   Hypertension       48510 non-null  int64  
 3   Heart_Disease      48510 non-null  int64  
 4   Ever_Married       48510 non-null  int32  
 5   Work_Type          48510 non-null  int32  
 6   Residence_Type     48510 non-null  int32  
 7   Avg_Glucose_Level  48510 non-null  float64
 8   BMI                48510 non-null  float64
 9   Smoking_Status     48510 non-null  int32  
 10  Stroke             48510 non-null  int64  
dtypes: float64(2), int32(6), int64(3)
memory usage: 3.0 MB
```

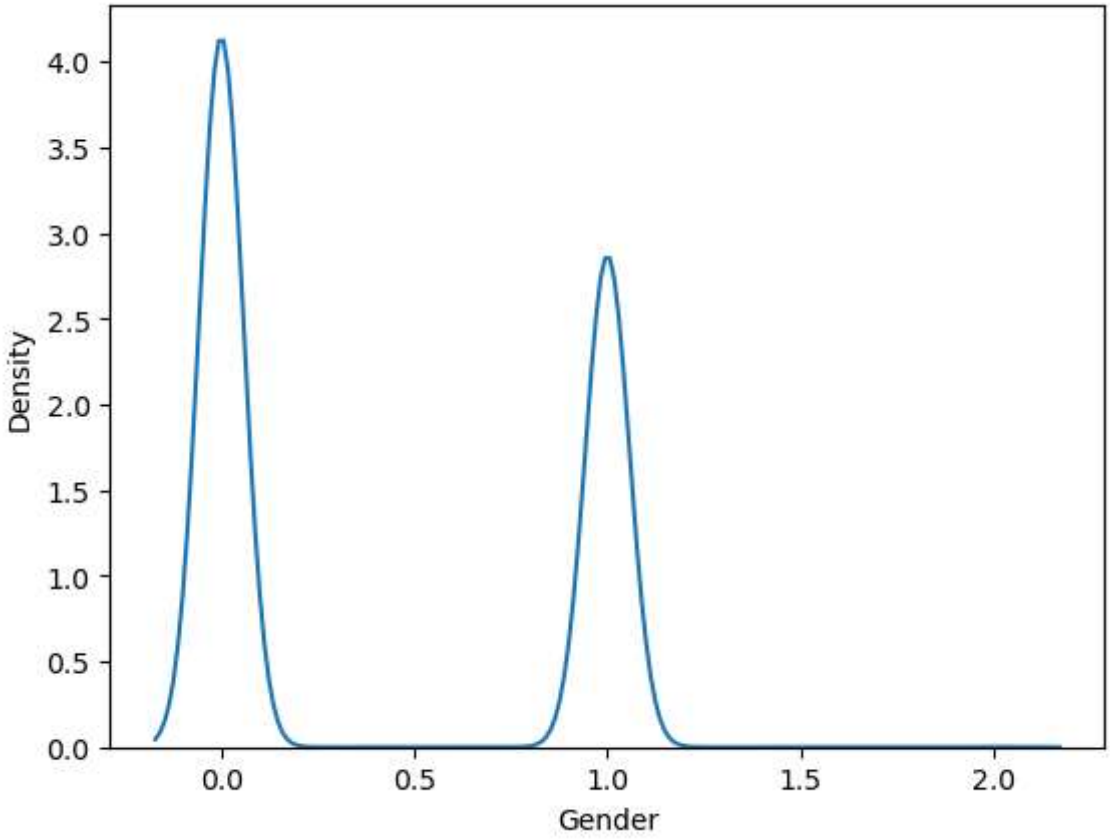
In [ ]:

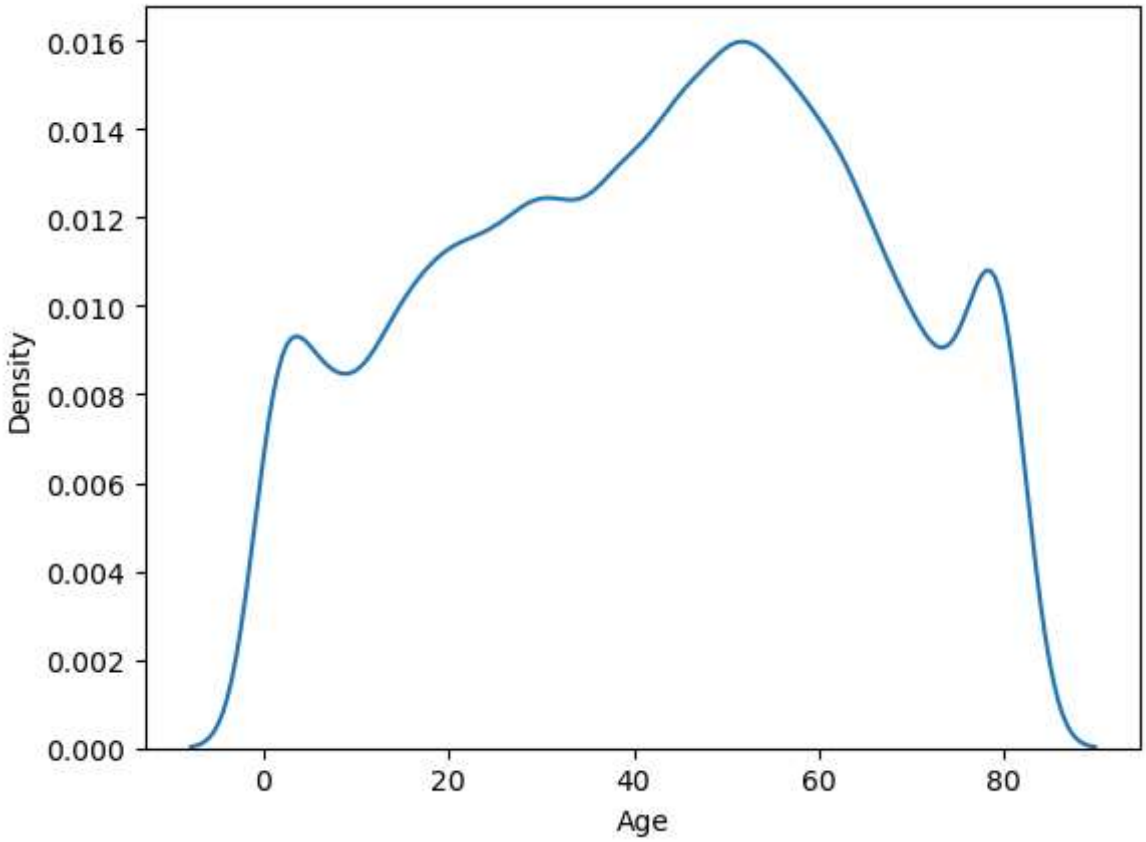
Checking for skewness

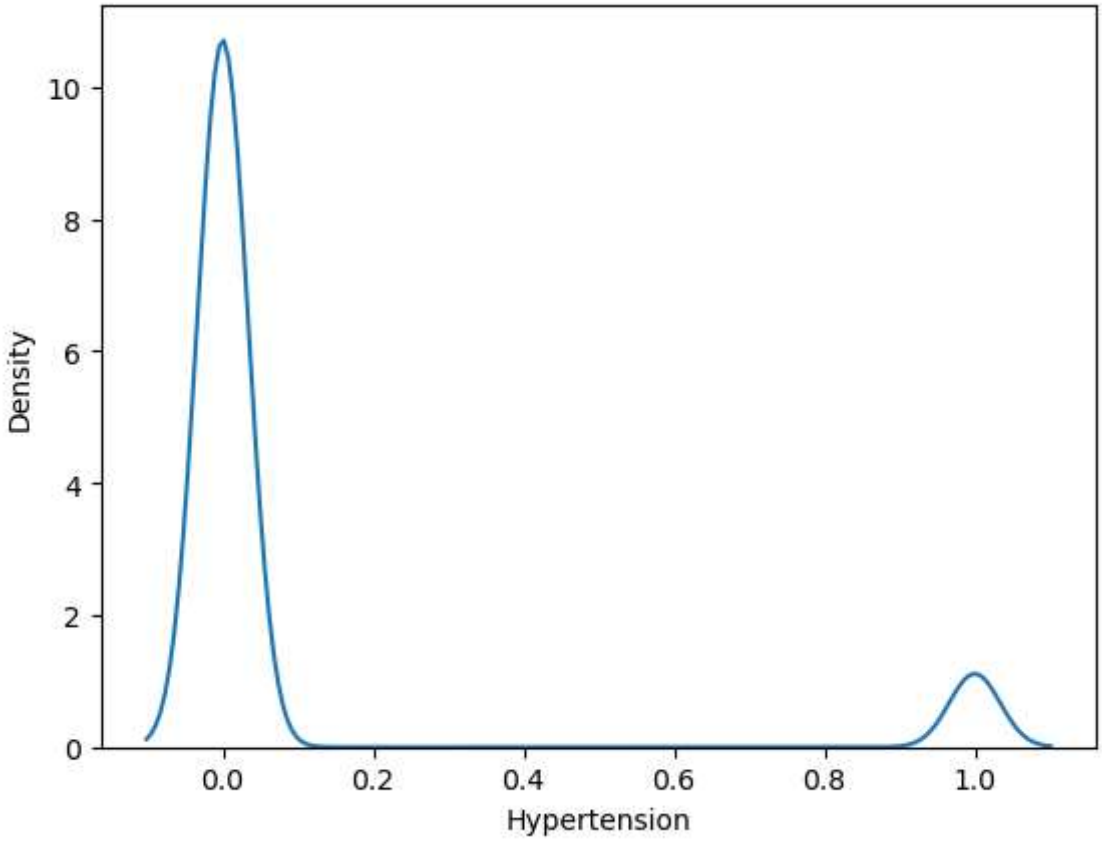
In [30]:
```
data.skew()
```

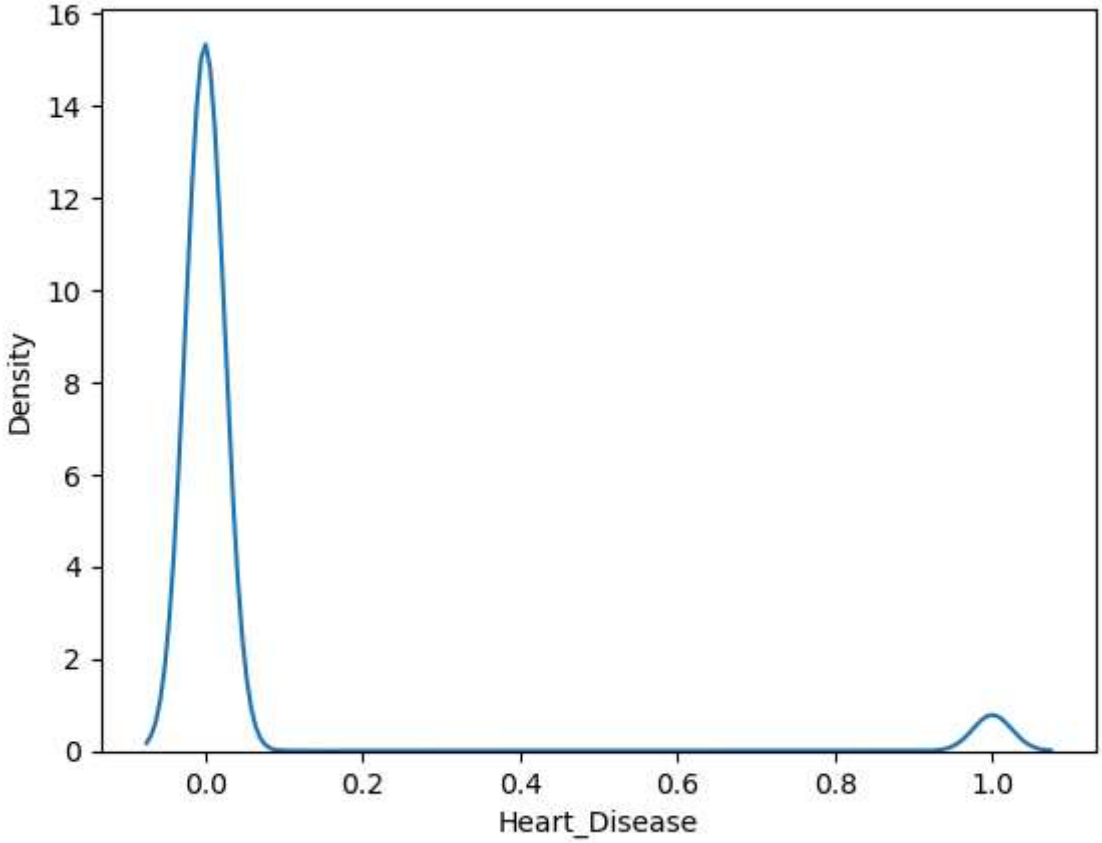
Out[30]:
```
Gender              0.374473
Age                -0.109716
Hypertension        2.782930
Heart_Disease       4.219033
Ever_Married       -0.606255
Work_Type          -0.297333
Residence_Type     -0.007999
Avg_Glucose_Level   1.664340
BMI                 0.942395
Smoking_Status     -0.003237
Stroke              6.635531
dtype: float64
```

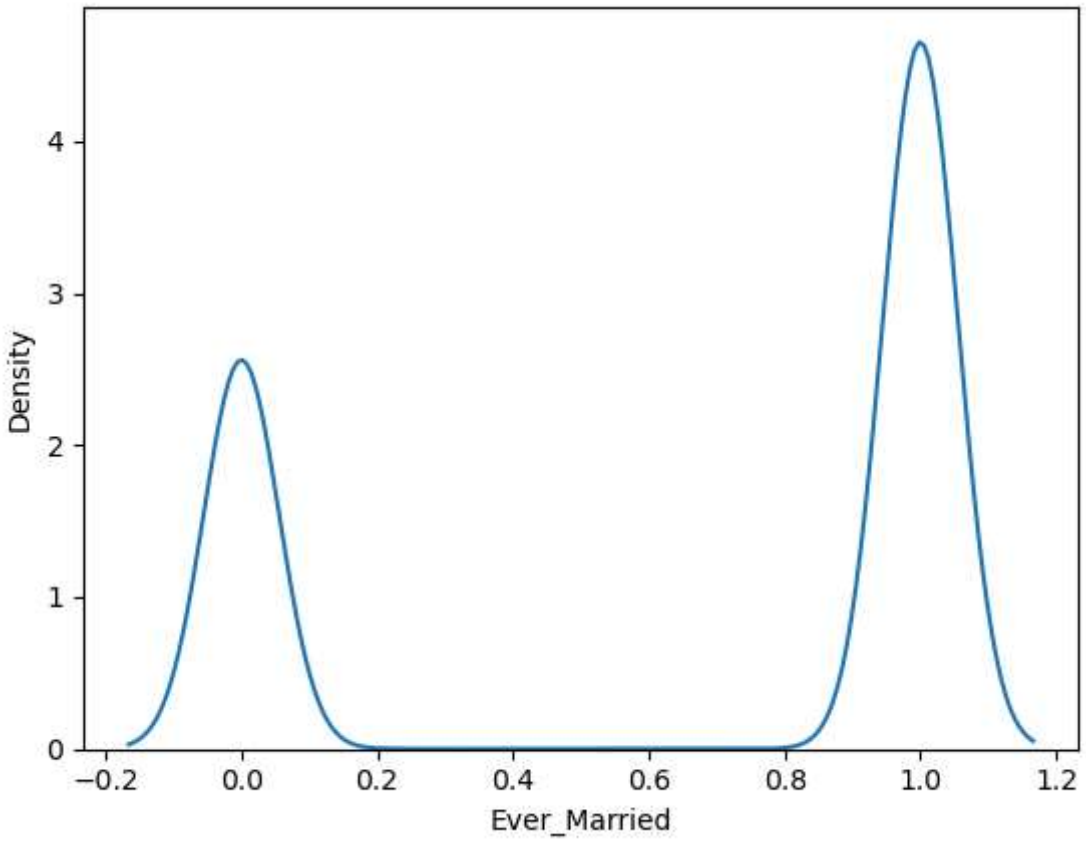
In [31]:
```
for i in data:
    sns.kdeplot(data=data,x=i)
    plt.show()
```

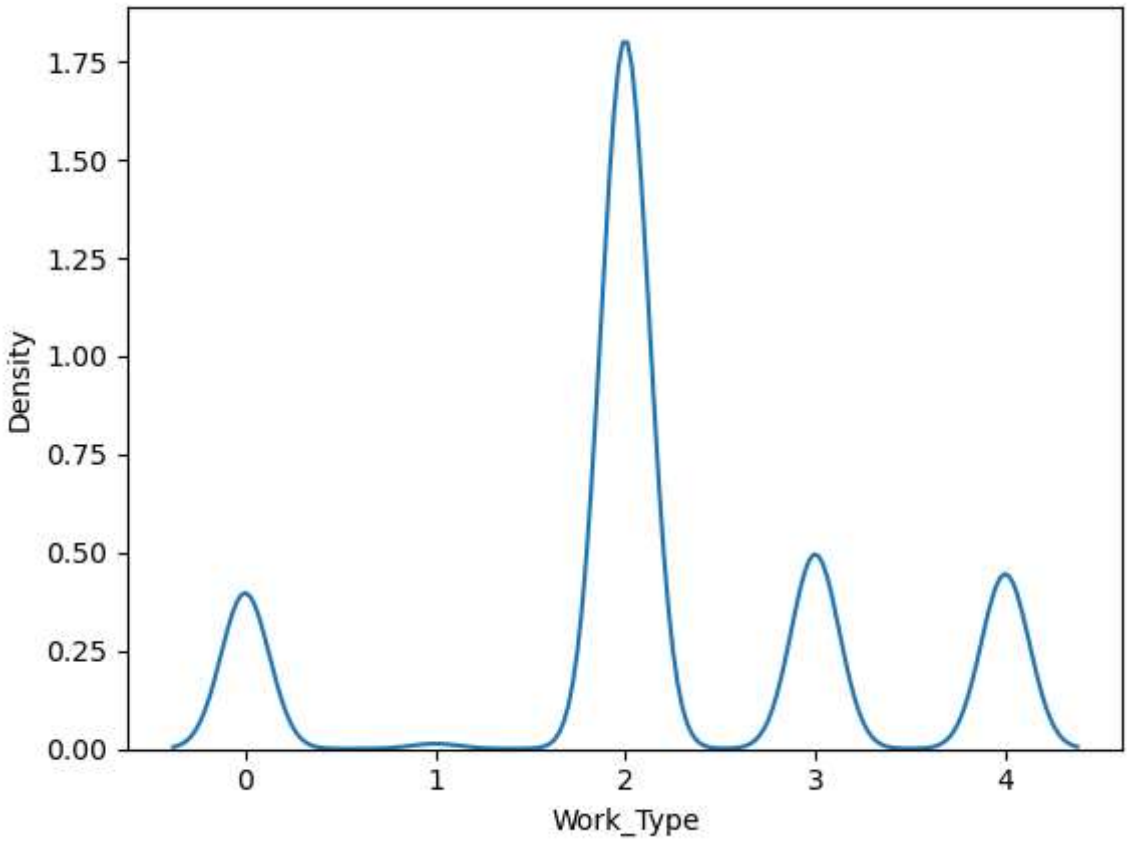

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
0.0
0.5
1.0
1.5
2.0
Gender

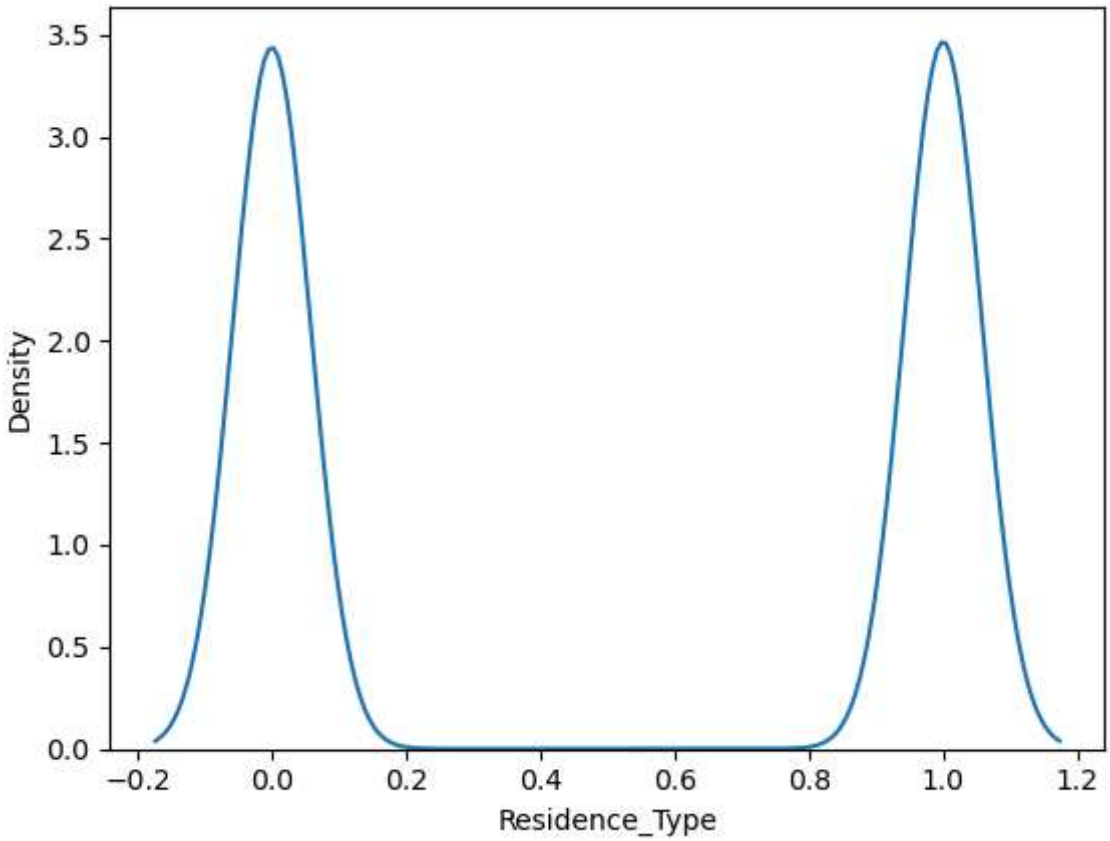

0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
0
20
40
60
80
Age

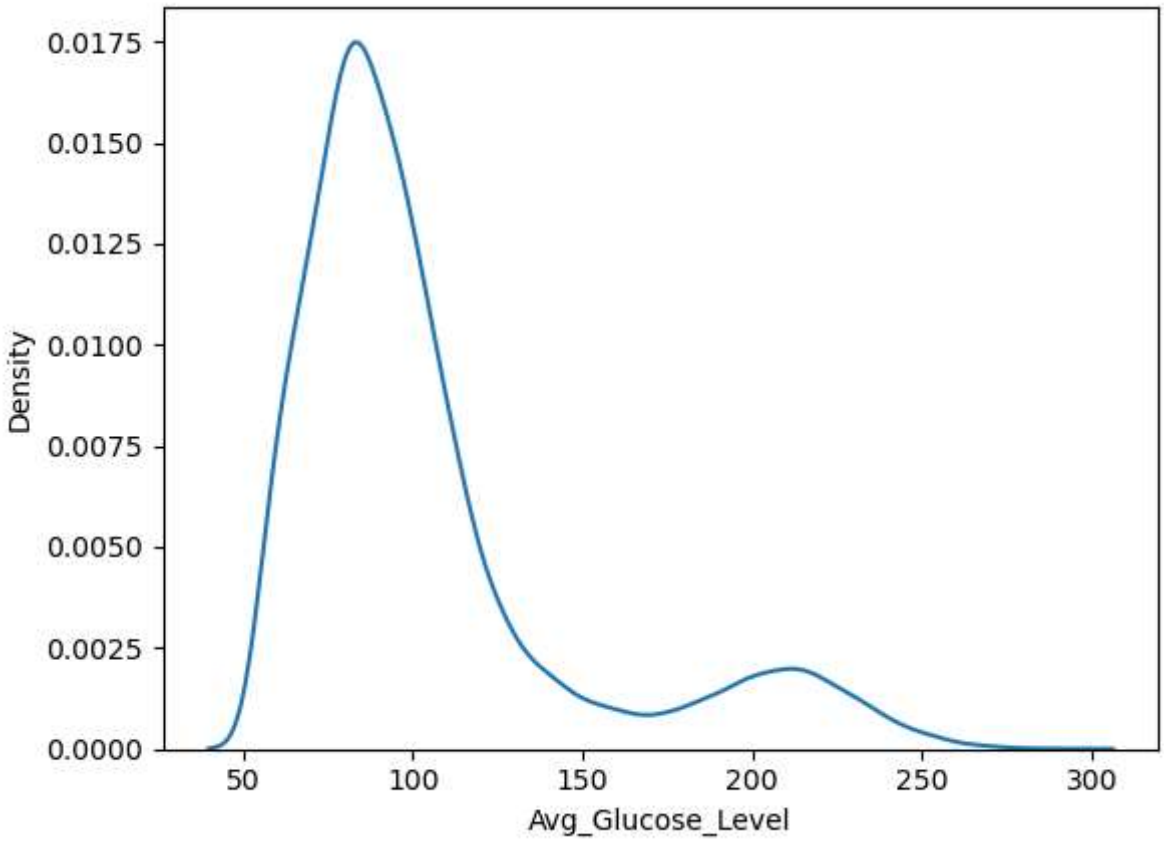

Density
Hypertension
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0

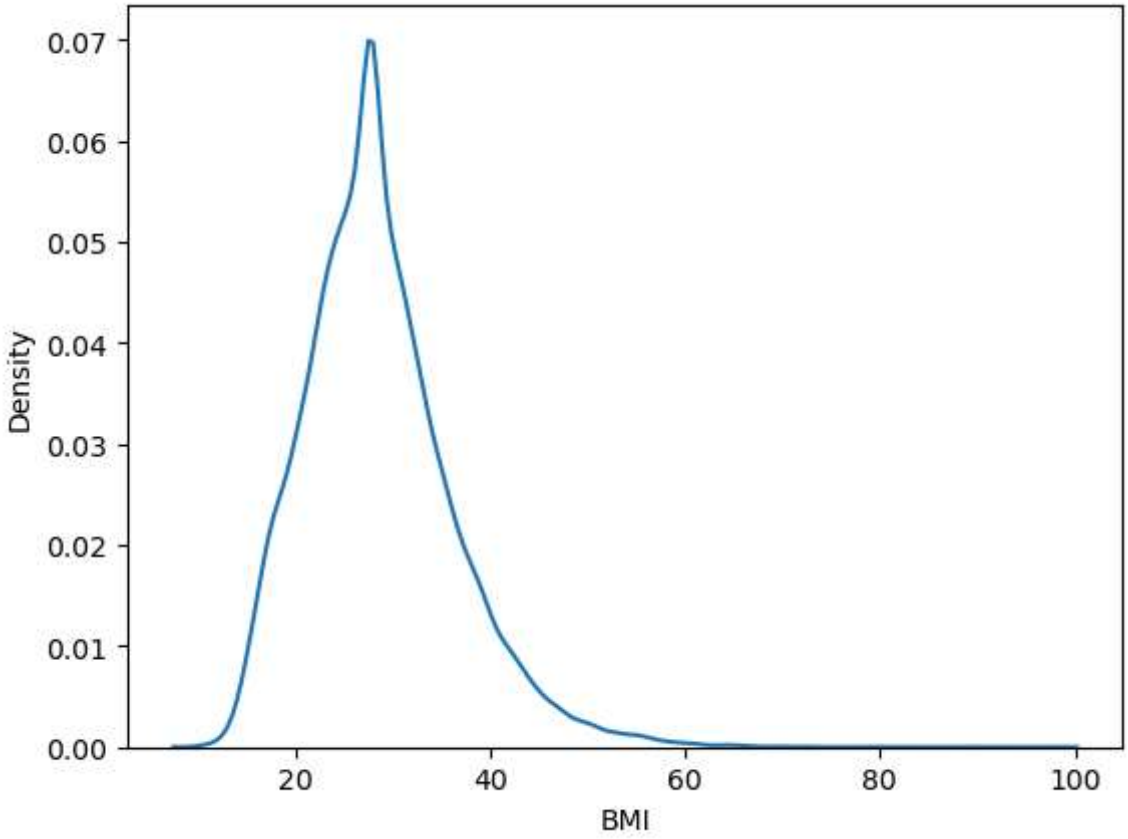

Density
Heart_Disease
0
2
4
6
8
10
12
14
16
0.0
0.2
0.4
0.6
0.8
1.0

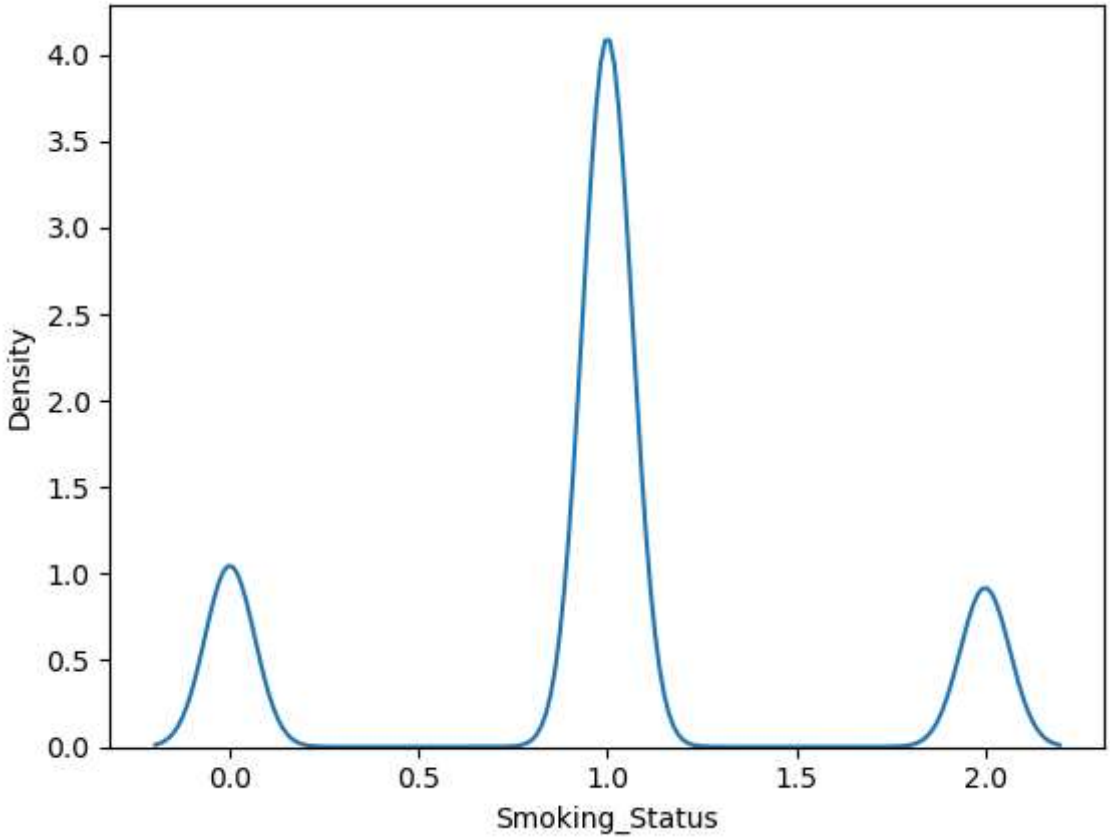

Density
Ever_Married
0
1
2
3
4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

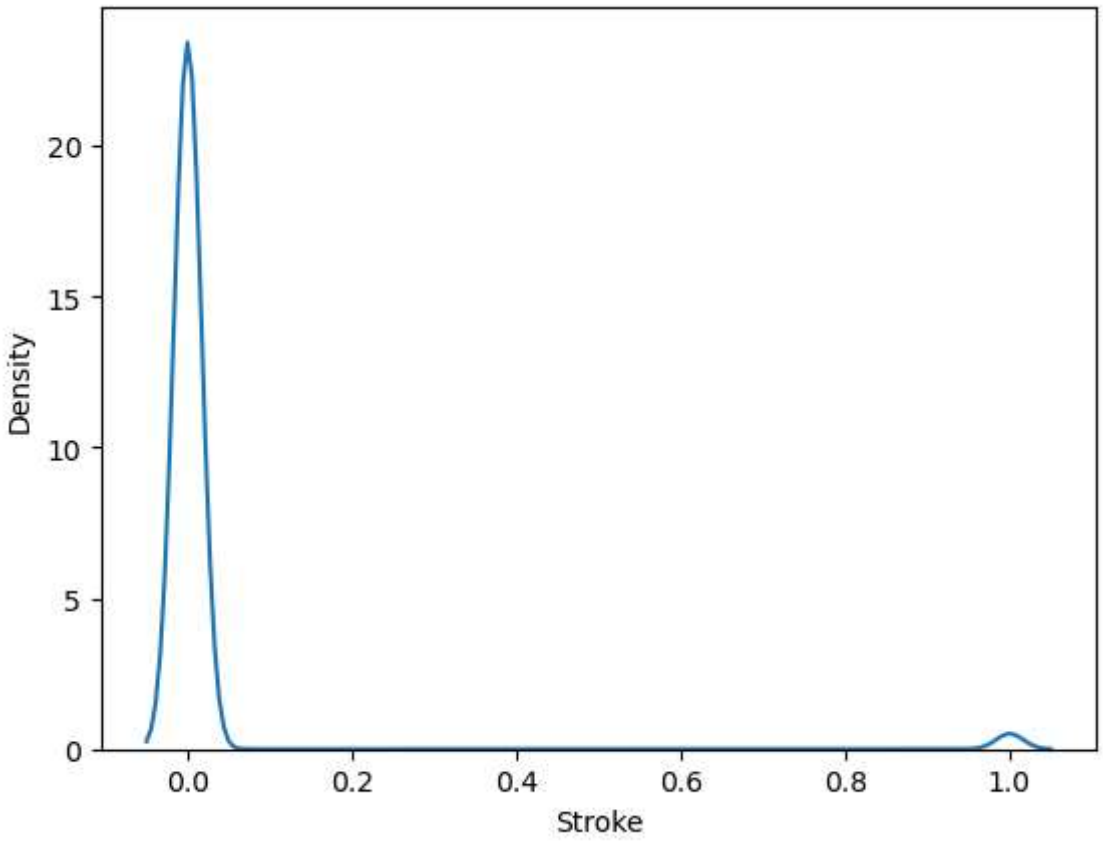

Density
Work_Type
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0
1
2
3
4


Density
Residence_Type
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Density
Avg_Glucose_Level
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
50
100
150
200
250
300


0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
20
40
60
80
100
BMI


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
0.0
0.5
1.0
1.5
2.0
Smoking_Status



In [ ]:

Separating Input Feature & Target Variable

In [32]:
```
data.columns
```

Out[32]:
```
Index(['Gender', 'Age', 'Hypertension', 'Heart_Disease', 'Ever_Married',
       'Work_Type', 'Residence_Type', 'Avg_Glucose_Level', 'BMI',
       'Smoking_Status', 'Stroke'],
      dtype='object')
```

In [33]:
```
X = data[['Gender', 'Age', 'Hypertension', 'Heart_Disease', 'Ever_Married',
       'Work_Type', 'Residence_Type', 'Avg_Glucose_Level', 'BMI',
       'Smoking_Status']]

y = data.Stroke
```

In [ ]:

Dividing the data into training & testing data

In [34]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.25, random_state =
```

In [ ]:

Importing the model

In [35]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [36]:
```
LR=LogisticRegression()
```

In [ ]:
```

```

Training the Model

In [37]:
```
LR.fit(x_train,y_train)
```

```
D:\Python\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarnin
g: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[37]:
```
LogisticRegression()
```

In [ ]:
```

```

Testing the Model

In [38]:
```
y_pred= LR.predict(x_test)
```

In [ ]:
```

```

Checking the Accuracy of the model

In [39]:
```
accuracy_score(y_test,y_pred)
```

Out[39]:
```
0.9790567282321899
```

In [ ]:
```

```

Importing the Confusion Matrix to check in how many the model get confused

In [40]:
```
from sklearn.metrics import confusion_matrix
```

In [41]:
```
confusion_matrix(y_test,y_pred)
```

Out[41]:
```
array([[11874,     0],
       [  254,     0]], dtype=int64)
```

In [ ]: