Importing Liabraries

In [1]:
```
import pandas as pd
import numpy as np
```

Loading Dataset

In [3]:
```
dataset=pd.read_csv(r'D:\csv files\Brain Stroke.csv')
dataset
```

Out[3]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_gl
0	Male	67.0	0	1	Yes	Private	Urban	
1	Male	80.0	0	1	Yes	Private	Rural	
2	Female	49.0	0	0	Yes	Private	Urban	
3	Female	79.0	1	0	Yes	Self-employed	Rural	
4	Male	81.0	0	0	Yes	Private	Urban	
...	...	...	...	...	...	...	...	
4976	Male	41.0	0	0	No	Private	Rural	
4977	Male	40.0	0	0	Yes	Private	Urban	
4978	Female	45.0	1	0	Yes	Govt_job	Rural	
4979	Male	40.0	0	0	Yes	Private	Rural	
4980	Female	80.0	1	0	Yes	Private	Urban	

4981 rows × 11 columns

Preprocessing

In [6]:
```
from sklearn.preprocessing import LabelEncoder
l1=LabelEncoder()
```

In [7]:
```
dataset['gender']=l1.fit_transform(dataset['gender'])
dataset['ever_married']=l1.fit_transform(dataset['ever_married'])
dataset['work_type']=l1.fit_transform(dataset['work_type'])
dataset['Residence_type']=l1.fit_transform(dataset['Residence_type'])
dataset['smoking_status']=l1.fit_transform(dataset['smoking_status'])
```

In [8]:
```
dataset
```

Out[8]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_gl
0	1	67.0	0	1	1	1	1	
1	1	80.0	0	1	1	1	0	
2	0	49.0	0	0	1	1	1	
3	0	79.0	1	0	1	2	0	
4	1	81.0	0	0	1	1	1	
...	...	...	...	...	...	...	...	
4976	1	41.0	0	0	0	1	0	
4977	1	40.0	0	0	1	1	1	
4978	0	45.0	1	0	1	0	0	
4979	1	40.0	0	0	1	1	0	
4980	0	80.0	1	0	1	1	1	

4981 rows × 11 columns

In [9]:
```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

In [10]:
```
x
```

Out[10]:
```
array([[  1.  ,  67.  ,   0.  , ..., 228.69,  36.6 ,   1.  ],
       [  1.  ,  80.  ,   0.  , ..., 105.92,  32.5 ,   2.  ],
       [  0.  ,  49.  ,   0.  , ..., 171.23,  34.4 ,   3.  ],
       ...,
       [  0.  ,  45.  ,   1.  , ...,  95.02,  31.8 ,   3.  ],
       [  1.  ,  40.  ,   0.  , ...,  83.94,  30.  ,   3.  ],
       [  0.  ,  80.  ,   1.  , ...,  83.75,  29.1 ,   2.  ]])
```

In [11]:
```
y
```

Out[11]:
```
array([1, 1, 1, ..., 0, 0, 0], dtype=int64)
```

In [13]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
scale=sc.fit_transform(x)
```

In [14]:
```
scale
```

Out[14]:
```
array([[ 1.18390850e+00,  1.04058433e+00, -3.26185770e-01, ...,
         2.72341090e+00,  1.19323816e+00, -3.53933192e-01],
       [ 1.18390850e+00,  1.61427033e+00, -3.26185770e-01, ...,
        -5.22766599e-04,  5.89389611e-01,  5.78839946e-01],
       [-8.44659868e-01,  2.46249882e-01, -3.26185770e-01, ...,
         1.44852918e+00,  8.69221866e-01,  1.51161308e+00],
       ...,
       [-8.44659868e-01,  6.97311148e-02,  3.06573766e+00, ...,
        -2.42364234e-01,  4.86293516e-01,  1.51161308e+00],
       [ 1.18390850e+00, -1.50917344e-01, -3.26185770e-01, ...,
        -4.88199415e-01,  2.21189274e-01,  1.51161308e+00],
       [-8.44659868e-01,  1.61427033e+00,  3.06573766e+00, ...,
        -4.92415000e-01,  8.86371531e-02,  5.78839946e-01]])
```

In [15]:
```
dataset['stroke'].value_counts()
```

Out[15]:
```
0    4733
1     248
Name: stroke, dtype: int64
```

In [16]:
```
from imblearn.over_sampling import RandomOverSampler
ros=RandomOverSampler()
X,Y=ros.fit_resample(scale,y)
```

In [17]:
```
from collections import Counter
print(Counter(Y))
```

```
Counter({1: 4733, 0: 4733})
```

Splitting data into train and test set

In [18]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=9)
```

Logistic Regression

In [19]:
```
from sklearn.linear_model import LogisticRegression
lg=LogisticRegression()
lg.fit(x_train,y_train)
```

Out[19]:
```
LogisticRegression()
```

In [20]:
```
y_pred=lg.predict(x_test)
y_pred
```

Out[20]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Accuracy of Logistic Regression

In [23]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred,y_test)*100)
```

```
75.55438225976768
```

KNN

In [25]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
knn.fit(x_train,y_train)
```

Out[25]:
```
KNeighborsClassifier()
```

In [26]:
```
y1_pred=knn.predict(x_test)
y1_pred
```

Out[26]:
```
array([1, 1, 1, ..., 0, 1, 1], dtype=int64)
```

Accuracy of KNN

In [27]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y1_pred,y_test)*100)
```

```
92.18585005279832
```

In [ ]:
```

```