IMPORTING MODULES

In [1]:

```
import numpy as np

# dataa split
from sklearn.model_selection import train_test_split

# model Evaluation
from sklearn import metrics

#navie bayesian and accuracy
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

#pandas
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn')
```

In [2]:

```
import seaborn as sns
import plotly.express as px

# SMOTE
from imblearn.over_sampling import SMOTE

# scaling
from sklearn.preprocessing import StandardScaler

# tune
from sklearn.model_selection import RandomizedSearchCV
```

READING THE DATASET

In [3]:

```
data=pd.read_csv('D1.csv')
data
```

Out[3]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
0	9046	Male	67	0	1	Yes	Private	Urban
1	31112	Male	80	0	1	Yes	Private	Rural
2	60182	Female	49	0	0	Yes	Private	Urban
3	1665	Female	79	1	0	Yes	Self-employed	Rural
4	56669	Male	81	0	0	Yes	Private	Urban
...	...	...	...	...	...	...	...	...
755	15220	Female	53	1	0	Yes	Private	Urban
756	4813	Male	27	0	0	No	Private	Urban
757	31166	Female	36	0	0	Yes	Govt_job	Rural
758	9051	Female	50	0	0	Yes	Private	Urban

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
759	59894	Female	58	0	0	Yes	Govt_job	Rural

760 rows × 12 columns

DATA ENCODING (PREPARATION)

In [4]:

```
# convert string to numeric using map

# gender
data['gender'] = data['gender'].map({
'Male': int(0),
'Female':int(1),
'Other':int(2)})

# ever_married
data['ever_married'] =data['ever_married'].map({
'Yes':int(1),
'No':int(0)})

# work_type
data['work_type'] = data['work_type'].map({
'Private':int(3),
'Self-employed':int(4),
'Govt_job':int(2),
'children':int(1),
'Never_worked':int(0)})

# Residence_type
data['Residence_type'] = data['Residence_type'].map({
'Urban':int(2),
'Rural':int(1)})

# smoking_status
data['smoking_status'] = data['smoking_status'].map({
'formerly smoked':int(1),
'never smoked':int(2),
'smokes':int(3),
'Unknown':int(0)})
```

attributes used in the classification

In [5]:

```
x=data[['gender','age','hypertension','heart_disease','ever_married','work_ty
y=data[['stroke']]
x=x.values
y=y.values
```

SPLIT DATASET

In [6]:

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20,ra
```

In [8]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(608, 10)
(152, 10)
```

In [9]:
```
y_train=y_train.flatten()
y_test=y_test.flatten()
print(y_train.shape)
print(y_test.shape)
```

```
(608,)
(152,)
```

NAVIE'S BAYER

In [10]:
```
from sklearn.naive_bayes import GaussianNB
```

In [11]:
```
model = GaussianNB()
model.fit(X_train,y_train)
predictions=model.predict(X_test)
```

In [12]:
```
print(np.unique(predictions))
```

```
[0 1]
```

In [13]:
```
print('1. CONFUSION MATRIX\n',metrics.confusion_matrix(y_test, predictions))


print("\n2. F1 SCORE")
print('F1-score on Test set:\t',metrics.f1_score(y_test,predictions))

print('\n3. OTHER METRICS')
print(metrics.classification_report(y_test, predictions))



# accuracy score
train_score =model.score(X_train,y_train)
test_score = model.score(X_test,y_test)

print("\n4. TRAINING AND TEST ERROS")
print('Accuracy on Train set\t',train_score)
print('Error on Train set\t',1-train_score)
print('Accuracy on Test set\t',test_score)
print('Error on Test set\t',1-test_score)
```

```
1. CONFUSION MATRIX
 [[103  16]
 [ 15  18]]
```

```
2. F1 SCORE
F1-score on Test set:    0.5373134328358209

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.87      0.87      0.87       119
           1       0.53      0.55      0.54        33

    accuracy                           0.80       152
   macro avg       0.70      0.71      0.70       152
weighted avg       0.80      0.80      0.80       152


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.7532894736842105
Error on Train set       0.2467105263157895
Accuracy on Test set     0.7960526315789473
Error on Test set        0.20394736842105265
```

DECISION TREE

In [14]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [15]:

```
model= DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

In [16]:

```
print(np.unique(predictions))
```

```
[0 1]
```

In [17]:

```
print('1. CONFUSION MATRIX\n',metrics.confusion_matrix(y_test, predictions))


print("\n2. F1 SCORE")
print('F1-score on Test set:\t',metrics.f1_score(y_test,predictions))

print('\n3. OTHER METRICS')
print(metrics.classification_report(y_test, predictions))



# accuracy score
train_score =model.score(X_train,y_train)
test_score = model.score(X_test,y_test)

print("\n4. TRAINING AND TEST ERROS")
print('Accuracy on Train set\t',train_score)
print('Error on Train set\t',1-train_score)
print('Accuracy on Test set\t',test_score)
print('Error on Test set\t',1-test_score)
```

```
1. CONFUSION MATRIX
 [[98 21]
```

```
 [16 17]]

2. F1 SCORE
F1-score on Test set:    0.4788732394366197

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.86      0.82      0.84       119
           1       0.45      0.52      0.48        33

    accuracy                           0.76       152
   macro avg       0.65      0.67      0.66       152
weighted avg       0.77      0.76      0.76       152


4. TRAINING AND TEST ERROS
Accuracy on Train set    1.0
Error on Train set       0.0
Accuracy on Test set     0.756578947368421
Error on Test set        0.24342105263157898
```

KNN KNeighborsClassifier

In [18]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

Here we took k=4.

This model will use the four nearest neighbors to predict the value of a future data point.

In [19]:
```python
model = KNeighborsClassifier(n_neighbors = 4)
model.fit(X_train, y_train.ravel())
predictions = model.predict(X_test)
```

In [20]:
```python
print(np.unique(predictions))
```

```
[0 1]
```

In [21]:
```python
print('1. CONFUSION MATRIX\n',metrics.confusion_matrix(y_test, predictions))


print("\n2. F1 SCORE")
print('F1-score on Test set:\t',metrics.f1_score(y_test,predictions))

print('\n3. OTHER METRICS')
print(metrics.classification_report(y_test, predictions))



# accuracy score
train_score =model.score(X_train,y_train)
test_score = model.score(X_test,y_test)

print("\n4. TRAINING AND TEST ERROS")
print('Accuracy on Train set\t',train_score)
print('Error on Train set\t',1-train_score)
print('Accuracy on Test set\t',test_score)
print('Error on Test set\t',1-test_score)
```

```
1. CONFUSION MATRIX
 [[109  10]
 [ 24   9]]

2. F1 SCORE
F1-score on Test set:    0.34615384615384615

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.82      0.92      0.87       119
           1       0.47      0.27      0.35        33

    accuracy                           0.78       152
   macro avg       0.65      0.59      0.61       152
weighted avg       0.74      0.78      0.75       152


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.8256578947368421
Error on Train set       0.17434210526315785
Accuracy on Test set     0.7763157894736842
Error on Test set        0.22368421052631582
```

ANN Artifical Neural Networks

In [22]:

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
```

In [23]:

```
# define keras model
model=tf.keras.Sequential()

model.add(tf.keras.layers.Dense(units=25,activation='relu'))

model.add(tf.keras.layers.Dense(units=25,activation='relu'))

model.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))


#compile keras model
model.compile('adam','binary_crossentropy',metrics=['accuracy'])
```

Training ANN Model

In [24]:

```
#fitting ANN to training set
model.fit(X_train,y_train,epochs=5)


#accuracy
accuracy=model.evaluate(X_train,y_train)
```

```
Epoch 1/5
19/19 [==============================] - 0s 520us/step - loss: 2.3732 - accur
acy: 0.7254
Epoch 2/5
```

```
19/19 [==============================] - 0s 824us/step - loss: 0.6559 - accur
acy: 0.6621
Epoch 3/5
19/19 [==============================] - 0s 911us/step - loss: 0.5183 - accur
acy: 0.7625
Epoch 4/5
19/19 [==============================] - 0s 1ms/step - loss: 0.5035 - accurac
y: 0.7758
Epoch 5/5
19/19 [==============================] - 0s 865us/step - loss: 0.4836 - accur
acy: 0.7924
19/19 [==============================] - 0s 433us/step - loss: 0.4825 - accur
acy: 0.7648
```

In [25]:

```
#predictions
predictions = model.predict(X_test)
predictions = (predictions > 0.5)
```

In [26]:

```
print('1. CONFUSION MATRIX\n',metrics.confusion_matrix(y_test, predictions))

print("\n2. F1 SCORE")
print('F1-score on Test set:\t',metrics.f1_score(y_test,predictions))

print('\n3. OTHER METRICS')
print(metrics.classification_report(y_test, predictions))
print("\n 4.ACCURACY")
print(accuracy)
```

```
1. CONFUSION MATRIX
 [[102  17]
 [ 18  15]]

2. F1 SCORE
F1-score on Test set:    0.46153846153846156

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.85      0.86      0.85       119
           1       0.47      0.45      0.46        33

    accuracy                           0.77       152
   macro avg       0.66      0.66      0.66       152
weighted avg       0.77      0.77      0.77       152


 4.ACCURACY
[0.4825453460216522, 0.7648026347160339]
```