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('D2.csv')
```

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)
```

```
(1323, 10)
(331, 10)
```

In [9]:

```
y_train=y_train.flatten()
y_test=y_test.flatten()
print(y_train.shape)
print(y_test.shape)
```

```
(1323,)
(331,)
```

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
 [[264  31]
 [ 25  11]]

2. F1 SCORE
F1-score on Test set:    0.2820512820512821

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       295
           1       0.26      0.31      0.28        36

    accuracy                           0.83       331
   macro avg       0.59      0.60      0.59       331
weighted avg       0.84      0.83      0.84       331


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.8261526832955405
Error on Train set       0.17384731670445952
Accuracy on Test set     0.8308157099697885
Error on Test set        0.16918429003021151
```

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
 [[268  27]
 [ 26  10]]

2. F1 SCORE
F1-score on Test set:    0.273972602739726

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.91      0.91      0.91       295
           1       0.27      0.28      0.27        36

    accuracy                           0.84       331
   macro avg       0.59      0.59      0.59       331
weighted avg       0.84      0.84      0.84       331


4. TRAINING AND TEST ERROS
Accuracy on Train set    1.0
Error on Train set       0.0
Accuracy on Test set     0.8398791540785498
Error on Test set        0.16012084592145015
```

KNN KNeighborsClassifier

In [18]:

```
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]:

```
model = KNeighborsClassifier(n_neighbors = 4)
model.fit(X_train, y_train.ravel())
predictions = model.predict(X_test)
```

In [20]:

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

```
[0 1]
```

In [21]:

```
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
 [[290   5]
 [ 35   1]]

2. F1 SCORE
F1-score on Test set:    0.047619047619047616

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.89      0.98      0.94       295
           1       0.17      0.03      0.05        36

    accuracy                           0.88       331
   macro avg       0.53      0.51      0.49       331
weighted avg       0.81      0.88      0.84       331


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.9062736205593348
Error on Train set       0.0937263794406652
Accuracy on Test set     0.879154078549849
Error on Test set        0.12084592145015105
```

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
42/42 [==============================] - 0s 614us/step - loss: 1.1907 - accur
acy: 0.8254
Epoch 2/5
42/42 [==============================] - 0s 660us/step - loss: 0.5757 - accur
acy: 0.8508
Epoch 3/5
42/42 [==============================] - 0s 792us/step - loss: 0.5023 - accur
acy: 0.8578
Epoch 4/5
42/42 [==============================] - 0s 729us/step - loss: 0.4923 - accur
acy: 0.8577
Epoch 5/5
42/42 [==============================] - 0s 804us/step - loss: 0.3739 - accur
acy: 0.8661
42/42 [==============================] - 0s 563us/step - loss: 0.3991 - accur
acy: 0.8496
```

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
 [[270  25]
 [ 28   8]]
```

```
2. F1 SCORE
F1-score on Test set:    0.2318840579710145

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.91      0.92      0.91       295
           1       0.24      0.22      0.23        36

    accuracy                           0.84       331
   macro avg       0.57      0.57      0.57       331
weighted avg       0.83      0.84      0.84       331


 4.ACCURACY
[0.3990599513053894, 0.8495842814445496]
```