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('D3.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
...	...	...	...	...	...	...	...	...
2538	5006	Female	46	0	0	Yes	Self-employed	Rural
2539	11250	Male	78	0	0	Yes	Self-employed	Rural
2540	41858	Female	63	0	1	Yes	Private	Rural

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
2541	34965	Female	18	0	0	No	Private	Urban
2542	65748	Female	46	0	0	Yes	Private	Urban

2543 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)
```

```
(2034, 10)
(509, 10)
```

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

```
(2034,)
(509,)
```

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
```

```
 [[427  49]
 [ 23  10]]

2. F1 SCORE
F1-score on Test set:    0.21739130434782608

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.95      0.90      0.92       476
           1       0.17      0.30      0.22        33

    accuracy                           0.86       509
   macro avg       0.56      0.60      0.57       509
weighted avg       0.90      0.86      0.88       509


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.8500491642084562
Error on Train set       0.14995083579154378
Accuracy on Test set     0.8585461689587426
Error on Test set        0.14145383104125742
```

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
 [[447  29]
 [ 29   4]]

2. F1 SCORE
F1-score on Test set:    0.12121212121212122

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       476
           1       0.12      0.12      0.12        33

    accuracy                           0.89       509
   macro avg       0.53      0.53      0.53       509
weighted avg       0.89      0.89      0.89       509


4. TRAINING AND TEST ERROS
Accuracy on Train set    1.0
Error on Train set       0.0
Accuracy on Test set     0.8860510805500982
Error on Test set        0.11394891944990182
```

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
 [[473   3]
 [ 33   0]]

2. F1 SCORE
F1-score on Test set:    0.0

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       476
           1       0.00      0.00      0.00        33

    accuracy                           0.93       509
   macro avg       0.47      0.50      0.48       509
weighted avg       0.87      0.93      0.90       509


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.9321533923303835
Error on Train set       0.06784660766961648
Accuracy on Test set     0.9292730844793713
Error on Test set        0.07072691552062871
```

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
64/64 [==============================] - 0s 612us/step - loss: 4.2688 - accur
acy: 0.5320
```

```
Epoch 2/5
64/64 [==============================] - 0s 717us/step - loss: 0.4407 - accur
acy: 0.9286
Epoch 3/5
64/64 [==============================] - 0s 730us/step - loss: 0.2819 - accur
acy: 0.9264
Epoch 4/5
64/64 [==============================] - 0s 685us/step - loss: 0.2545 - accur
acy: 0.9315
Epoch 5/5
64/64 [==============================] - 0s 681us/step - loss: 0.2528 - accur
acy: 0.9288
64/64 [==============================] - 0s 429us/step - loss: 0.2446 - accur
acy: 0.9267
```

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
 [[476   0]
 [ 33   0]]

2. F1 SCORE
F1-score on Test set:    0.0

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       476
           1       0.00      0.00      0.00        33

    accuracy                           0.94       509
   macro avg       0.47      0.50      0.48       509
weighted avg       0.87      0.94      0.90       509


 4.ACCURACY
[0.24461983144283295, 0.9267453551292419]
```

```
/home/pandu/my_project_dir/my_project_env/lib/python3.8/site-packages/sklear
n/metrics/_classification.py:1248: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/pandu/my_project_dir/my_project_env/lib/python3.8/site-packages/sklear
n/metrics/_classification.py:1248: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/pandu/my_project_dir/my_project_env/lib/python3.8/site-packages/sklear
n/metrics/_classification.py:1248: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```