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('dataset.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
...	...	...	...	...	...	...	...	...
3421	68398	Male	82	1	0	Yes	Self-employed	Rural
3422	45010	Female	57	0	0	Yes	Private	Rural
3423	44873	Female	81	0	0	Yes	Self-employed	Urban

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
3424	19723	Female	35	0	0	Yes	Self-employed	Rural
3425	37544	Male	51	0	0	Yes	Private	Rural

3426 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

The dataset has been splitted into training set and test set.

1. Training set: 80%
2. Test set: 20%

Split arrays or matrices into random train and test subsets

parameters

1. *arrayssequence of indexables with same length / shape[0]
2. test_sizefloat or int, default=None
3. train_sizefloat or int, default=None
4. random_stateint, RandomState instance or None, default=None
5. shufflebool, default=True
6. stratifyarray-like, default=None

returns

1. splittinglist, length=2 * len(arrays)

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)
```

```
(2740, 10)
(686, 10)
```

In [9]:

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

```
(2740,)
(686,)
```

NAVIE'S BAYER

prototype: class sklearn.naive_bayes.GaussianNB(*, priors=None, var_smoothing=1e-09)

Parameters

1. priorsarray-like of shape (n_classes,)
2. var_smoothingfloat, default=1e-9

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
 [[567  75]
 [ 25  19]]

2. F1 SCORE
F1-score on Test set:    0.2753623188405797

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.96      0.88      0.92       642
           1       0.20      0.43      0.28        44

    accuracy                           0.85       686
   macro avg       0.58      0.66      0.60       686
weighted avg       0.91      0.85      0.88       686


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.8766423357664234
Error on Train set       0.12335766423357664
Accuracy on Test set     0.8542274052478134
Error on Test set        0.14577259475218662
```

DECISION TREE

Prototype: class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

Parameters

1. criterion{“gini”, “entropy”}, default=”gini”
2. splitter{“best”, “random”}, default=”best”
3. max_depthint, default=None
4. min_samples_splitint or float, default=2
5. min_samples_leafint or float, default=1
6. min_weight_fraction_leaffloat, default=0.0
7. max_featuresint, float or {“auto”, “sqrt”, “log2”}, default=None
8. random_stateint, RandomState instance or None, default=None
9. max_leaf_nodesint, default=None
10. min_impurity_decreasefloat, default=0.0
11. min_impurity_splitfloat, default=0
12. class_weightdict, list of dict or “balanced”, default=None
13. ccp_alphanon-negative float, default=0.0

In [14]:

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

In [15]:

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

In [16]:

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

```
[0 1]
```

In [17]:

```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
 [[600  42]
 [ 35   9]]

2. F1 SCORE
F1-score on Test set:    0.18947368421052632

3. OTHER METRICS
              precision    recall  f1-score   support
```

```
           0       0.94      0.93      0.94       642
           1       0.18      0.20      0.19        44

    accuracy                           0.89       686
   macro avg       0.56      0.57      0.56       686
weighted avg       0.90      0.89      0.89       686


4. TRAINING AND TEST ERROS
Accuracy on Train set    1.0
Error on Train set       0.0
Accuracy on Test set     0.8877551020408163
Error on Test set        0.11224489795918369
```

KNN KNeighborsClassifier

Protype: class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs)

Parameters

1. n_neighborsint, default=5
2. weights{‘uniform’, ‘distance’} or callable, default=’uniform’
3. algorithm{‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’}, default=’auto’
4. leaf_sizeint, default=30
5. pint, default=2
6. metricstr or callable, default=’minkowski’
7. metric_paramsdict, default=None
8. n_jobsint, default=None

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
 [[636   6]
 [ 43   1]]

2. F1 SCORE
F1-score on Test set:    0.0392156862745098

3. OTHER METRICS
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       642
           1       0.14      0.02      0.04        44

    accuracy                           0.93       686
   macro avg       0.54      0.51      0.50       686
weighted avg       0.89      0.93      0.90       686


4. TRAINING AND TEST ERROS
Accuracy on Train set    0.9529197080291971
Error on Train set       0.04708029197080288
Accuracy on Test set     0.9285714285714286
Error on Test set        0.0714285714285714
```

ANN Artifical Neural Networks

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

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='tanh'))

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
86/86 [==============================] - 1s 969us/step - loss: 1.8740 - accur
acy: 0.1874
Epoch 2/5
86/86 [==============================] - 0s 860us/step - loss: 0.2016 - accur
acy: 0.9503
Epoch 3/5
86/86 [==============================] - 0s 687us/step - loss: 0.1920 - accur
acy: 0.9500
Epoch 4/5
86/86 [==============================] - 0s 871us/step - loss: 0.1813 - accur
acy: 0.9526
Epoch 5/5
86/86 [==============================] - 0s 793us/step - loss: 0.1700 - accur
acy: 0.9565
86/86 [==============================] - 0s 644us/step - loss: 0.1844 - accur
acy: 0.9504
```

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
 [[642   0]
 [ 44   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       642
           1       0.00      0.00      0.00        44

    accuracy                           0.94       686
   macro avg       0.47      0.50      0.48       686
weighted avg       0.88      0.94      0.90       686


 4.ACCURACY
[0.1843763291835785, 0.9503649473190308]
```

```
/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))
```