Step 1: Data Preprocessing

In [1]:

```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings


# ignore all warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
#importing our cancer dataset
dataset = pd.read_csv('dataset.csv')
X = dataset.iloc[:, 1:31].values
Y = dataset.iloc[:, 31].values
```

In [3]:

```
print("Cancer data set dimensions : {}".format(dataset.shape))
```

```
Cancer data set dimensions : (568, 32)
```

Missing or Null Data points

In [4]:

```
dataset.isnull().sum()
dataset.isna().sum()
```

Out[4]:

```
842302      0
17.99       0
10.38       0
122.8       0
1001        0
0.1184      0
0.2776      0
0.3001      0
0.1471      0
0.2419      0
0.07871     0
1.095       0
0.9053      0
8.589       0
153.4       0
0.006399    0
0.04904     0
0.05373     0
0.01587     0
0.03003     0
0.006193    0
25.38       0
17.33       0
184.6       0
2019        0
0.1622      0
0.6656      0
0.7119      0
0.2654      0
0.4601      0
0.1189      0
M           0
dtype: int64
```

In [5]:

```
#Encoding categorical data values
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
Y = labelencoder_Y.fit_transform(Y)
```

In [6]:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
```

Step 2: Feature Scaling

In [7]:

```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step 3: Fitting supervised machine learning algorithm to the Training set

In [8]:

```
#Using KNeighborsClassifier Method of neighbors class to use Nearest Neighbor algorithm
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, Y_train)
```

Out[8]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

Step 4: Predecting the Result

In [9]:

```
predit_classifier_KNN = classifier.predict(X_test)
#print result
print("Prediction results of K Nearest Neighbor algorithm : \n{}".format(predit_classifier_KNN))
```

```
Prediction results of K Nearest Neighbor algorithm :
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0
 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0
 0 0 0 0 1 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0
 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1]
```

Step 5: Making the Confusion Matrix & Classfication Report

In [10]:

```
#import classification report & confusion_matrix & Accuracy score
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

print("Classification Report of K Nearest Neighbor algorithm : \n{}"
      .format(classification_report(Y_test, predit_classifier_KNN)))

print("Confusion Matrix of K Nearest Neighbor algorithm : \n{}".format(confusion_matrix(Y_test, predit_classifier_KNN)))

print('\nAccuracy is :- ',accuracy_score(predit_classifier_KNN,Y_test))
```

```
Classification Report of K Nearest Neighbor algorithm :
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        92
           1       1.00      0.88      0.94        50

   micro avg       0.96      0.96      0.96       142
   macro avg       0.97      0.94      0.95       142
weighted avg       0.96      0.96      0.96       142

Confusion Matrix of K Nearest Neighbor algorithm :
[[92  0]
 [ 6 44]]

Accuracy is :-  0.9577464788732394
```