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 SVC method of svm class to use Support Vector Machine Algorithm
from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(X_train, Y_train)
```

Out[8]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

Step 4: Predecting the Result

In [9]:
```
predit_classifier = classifier.predict(X_test)
#print result
print("Prediction results of Support Vector Machine Alogorithm : \n{}".format(predit_classifier))
```

```
Prediction results of Support Vector Machine Alogorithm :
[0 0 0 0 0 1 0 0 0 0 0 1 1 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 0 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 1 1 1 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0
 1 1 1 1 0 0 0 0 1 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 Support Vector Machine Alogorith : \n{}"
      .format(classification_report(Y_test, predit_classifier)))

print("Confusion Matrix of Support Vector Machine Alogorith : \n{}".format(confusion_matrix(Y_test, predit_classifier)))

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

```
Classification Report of Support Vector Machine Alogorith :
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        92
           1       0.98      0.94      0.96        50

   micro avg       0.97      0.97      0.97       142
   macro avg       0.97      0.96      0.97       142
weighted avg       0.97      0.97      0.97       142

Confusion Matrix of Support Vector Machine Alogorith :
[[91  1]
 [ 3 47]]

Accuracy is :-  0.971830985915493
```