Importing the libraries

In [1]:

```
#import libraray
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [2]:

```
#import dataset
dataset = pd.read_csv('dataset.csv')
```

In [3]:

```
#two separate column
x = dataset.iloc[:, 1:5].values
y = dataset.iloc[:, 5].values
```

In [4]:

```
#make data frame
x = pd.DataFrame(x)
y = pd.DataFrame(y)
```

Data Clean

In [5]:

```
#import lable encoder library
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
#lable encode
x.values[:, 0] = labelencoder_X.fit_transform(x.values[:, 0])
x.values[:, 1] = labelencoder_X.fit_transform(x.values[:, 1])
x.values[:, 2] = labelencoder_X.fit_transform(x.values[:, 2])
x.values[:, 3] = labelencoder_X.fit_transform(x.values[:, 3])
y.values[:, 0] = labelencoder_X.fit_transform(y.values[:, 0])
```

Splitting the dataset into the Training set and Test set

In [6]:

```
#model split
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)
```

In [7]:

```
X_train=X_train.astype(float)
y_train=y_train.astype(float)
X_test=X_test.astype(float)
y_test=y_test.astype(float)
```

Feature Scaling

In [8]:

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

Fitting K-NN to the Training set

In [9]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 19)
classifier.fit(X_train, y_train.values.ravel())
```

Out[9]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=19, p=2,
           weights='uniform')
```

Predicting the Test set results

In [10]:

```
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[1. 1. 1. 1. 1. 1. 1.]
```

Accuracy Result

In [11]:

```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7142857142857143
```

Making the Confusion Matrix

In [12]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[0 2]
 [0 5]]
```