In [1]:
```
import numpy as np
from sklearn.metrics import confusion_matrix
```

In [2]:
```
# Load data set and code labels as 0 = 'NO', 1 = 'DH', 2 = 'SL'
labels = [b'NO', b'DH', b'SL']
data = np.loadtxt('spine-data.txt', converters = {6: lambda s: labels.index(s)})
```

In [3]:
```
# Separate features from labels
X = data[:,0:6]
y = data[:,6]

# Split the data into a training set, consisting of the first 250 points,
# and a test set, consisting of the remaining 60 points.
train_data = X[list(range(0,250)),:]
train_label = y[list(range(0,250))]
test_data = X[list(range(250,310)),:]
test_label = y[list(range(250,310))]
```

In [4]:
```
def distance_L1(x, y):
    return np.sum(np.abs(x-y))

def square_distance_L2(x, y):
    return np.sum(np.square(x-y))

def predict(x, y):
    return y[np.argmin(x)]

def NN_Classifier_L1(trainx, trainy, testx):
    testy_L1 = []
    for i in range(len(testx)):
        distance = [distance_L1(testx[i], trainx[j]) for j in range(len(trainx))]
        test_predicted = predict(distance, trainy)
        testy_L1.append(test_predicted)
    return np.asarray(testy_L1)

def NN_Classifier_L2(trainx, trainy, testx):
    testy_L2 = []
    for i in range(len(testx)):
        distance = [square_distance_L2(testx[i], trainx[j]) for j in range(len(trainx))]
        test_predicted = predict(distance, trainy)
        testy_L2.append(test_predicted)
    return np.asarray(testy_L2)

def error_rate(testy, testy_fit):
    return float(sum(testy != testy_fit))/len(testy)
```

(a) What error rates do you get on the test set for each of the two distance functions?

In [5]:
```
test_label_L1 = NN_Classifier_L1(train_data, train_label, test_data)
test_label_L2 = NN_Classifier_L2(train_data, train_label, test_data)

error_L1 = error_rate(test_label, test_label_L1)
error_L2 = error_rate(test_label, test_label_L2)

print("Error rate of NN_Classifier_L1:\n", error_L1)
print("\nError rate of NN_Classifier_L2:\n", error_L2)
```

```
Error rate of NN_Classifier_L1:
 0.21666666666666667

Error rate of NN_Classifier_L2:
 0.23333333333333334
```

(b) For each of the two distance functions, give the confusion matrix of the NN classifier.

In [6]:
```
print('Confusion matrix of nearest neighbor classifier L1:\n', confusion_matrix(test_label, test_label_L1))
```

```
Confusion matrix of nearest neighbor classifier L1:
 [[14  0  2]
 [ 9  9  0]
 [ 1  1 24]]
```

In [7]:
```
print('Confusion matrix of nearest neighbor classifier L2:\n', confusion_matrix(test_label, test_label_L2))
```

```
Confusion matrix of nearest neighbor classifier L2:
 [[12  1  3]
 [ 9  9  0]
 [ 1  0 25]]
```

In [ ]:
```

```