In [1]:
```
import numpy as np
labels = [b'NO', b'DH', b'SL']
data = np.loadtxt("../data/spine-data.txt", converters={6: lambda s : labels.index(s)}
```

In [2]:
```
tr = data[:250, :-1]
te = data[250:, :-1]
tr_y = data[:250, -1]
te_y = data[250:, -1]

print(tr.shape, tr_y.shape)
```

```
(250, 6) (250,)
```

In [3]:
```
# copied from example code
def squared_dist(x,y):
  v = np.sum(np.square(x - y))
  return v

def l1_norm(x, y):
  return np.sum(np.abs(x - y))

def find_NN(x, dist_fn):
  # Compute distances from x to every row in train_data
  distances = [dist_fn(x,tr[i,]) for i in range(len(tr_y))]
  # Get the index of the smallest distance
  return np.argmin(distances)
def NN_classifier(x, dist_fn):
  # Get the index of the the nearest neighbor
  index = find_NN(x, dist_fn)
  # Return its class
  return tr_y[index]
```

In [4]:
```
def experiment(dist_fn):
  test_predictions = [NN_classifier(te[i,], dist_fn) for i in range(len(te_y))]
  err_positions = np.not_equal(test_predictions, te_y)
  error = float(np.sum(err_positions))/len(te_y)

  mat = np.zeros((3, 3))
  for lab, pred in zip(te_y, test_predictions):
    x, y = int(lab), int(pred)
    mat[x, y] += 1

  return error, mat
```

In [5]:
```
print("=== L1 Norm ===")
l1_e, l1_mat = experiment(l1_norm)
print(f"Error: {l1_e:0.4f}")
print(l1_mat.astype(int))
print("=== L2 Norm ===")
l2_e, l2_mat = experiment(squared_dist)
print(f"Error: {l2_e:0.4f}")
print(l2_mat.astype(int))
```

```
=== L1 Norm ===
Error: 0.2167
[[14  0  2]
 [ 9  9  0]
 [ 1  1 24]]
=== L2 Norm ===
Error: 0.2333
[[12  1  3]
 [ 9  9  0]
 [ 1  0 25]]
```