In [13]:
```
#import modules
import os
import numpy as np
from numpy import asarray
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from PIL import Image, ImageFilter
```

In [14]:
```
#define folders and # of classes
folder_training = 'Lego_dataset_2/training/'
folder_testing = 'Lego_dataset_2/testing/'
classes = [i for i in range (3)]
```

In [15]:
```
#Creating a list for classes, and getting data from folders
def get_data (folder,im_width):
    File_names = os.listdir(folder)
    folder_length = len(File_names)
    x = np.empty ((folder_length,im_width**2))
    y = np.empty ((folder_length,1))
    for i in range(folder_length):
        path = folder + File_names[i]
        im = Image.open(path).convert('L')
        im = im.resize((im_width,im_width))
        im_array = asarray(im)
        x[i ,:] = im_array.reshape(1,-1)
        if File_names[i][0:3].startswith("cir"):
            y[i,0] = classes[0]
        elif File_names[i][0:3].startswith("rec"):
            y[i,0] = classes[1]
        else:
            y[i,0] = classes[2]
    return x,y
```

In [16]:
```
#training data
im_width = 64
im_length = im_width
File_names = os.listdir(folder_training)
folder_length = len(File_names)
P_per_class = 24;
x_train = np.empty((folder_length,im_width**2))
y_train = np.empty((folder_length,1))
print(x_train.shape,y_train.shape)
x_train,y_train = get_data(folder_training, im_width)
P_train = len(x_train[:,0])


def count_max_horizontal(image_array, max_horizontal):
    for i in range(len(image_array[:,0])):
        if(np.count_nonzero(image_array[:,i]) == 0):
            continue
        max_count = np.count_nonzero(image_array[:,i])
        if(max_count > max_horizontal):
            max_horizontal = max_count

    return max_horizontal

def count_max_vertical(image_array, max_vertical):
    for i in range(len(image_array[0,:])):
        if(np.count_nonzero(image_array[i,:]) == 0):
            continue
        max_count = np.count_nonzero(image_array[i,:])
        if(max_count > max_vertical):
            max_vertical = max_count

    return max_vertical
```

```
(81, 4096) (81, 1)
```

In [17]:
```
#Calculating max nonzeros horizontally and vertically in all images
def process_image(image_array,edge_thresh,label):
    image_array = (image_array-np.min(image_array))*255/(np.max(image_array)-np.min(image_array)) #greyscale 0-255 for contrast
    im = Image.fromarray(image_array.reshape(im_width,im_length)).convert('L')
    edges_image = im.filter(ImageFilter.FIND_EDGES)
    edges_array = np.asarray(edges_image)
    edges_array_scaled = edges_array.copy()[1:im_width-1,1:im_length-1]
    edges_array_scaled[edges_array_scaled < edge_thresh,] = 0

    max_horizontal = 0
    max_vertical = 0

    if(label == 1):
        img = Image.fromarray(edges_array_scaled)
        count = 0
        while(max_horizontal < 10 or count <= 25):
            img = img.rotate(5)
            new_edges = np.asarray(img)
            max_horizontal = count_max_horizontal(new_edges, max_horizontal)
            count = count + 1



    if(label == 2):
        img = Image.fromarray(edges_array_scaled)
        count = 0
        while(max_horizontal < 10 or count <= 25):
            img = img.rotate(5)
            new_edges = np.asarray(img)
            max_horizontal = count_max_horizontal(new_edges, max_horizontal)
            count = count + 1


    x = np.array([max_horizontal,max_vertical]).reshape(1,-1)
    return x
```

In [ ]:

In [18]:
```
#Image processing implemented
ET = 45
```

```
PT = 2
edge_features_train = np.empty((P_train,2)) #training features
for i in range(P_train):
    edge_features_train[i,:] = process_image(x_train[i,:],ET,PT)
```

In [19]:
```
#fitting the model
model = LogisticRegression()
model.fit(edge_features_train,y_train)
```

Out[19]:
```
LogisticRegression()
```

In [20]:
```
# testing data
P_per_class = 2500;
x_test,y_test = get_data(folder_testing,im_width)
P_test = len(x_test[:,0])
print(x_test.shape,y_test.shape)
```

```
(81, 4096) (81, 1)
```

In [21]:
```
edge_features_test = np.empty((P_test,2))
for i in range(P_test):edge_features_test[i,:] = process_image(x_test[i,:],ET,PT)
```

In [22]:
```
#displaying the results
y_pred = model.predict(edge_features_test)
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
0.8518518518518519
[[21  1  5]
 [ 0 26  1]
 [ 3  2 22]]
```

In [23]:
```
def test_function(path,im_width,ET,PT,model):

    x_test,y_test=get_data(path,im_width)
    P_test = len(x_test[:,0])

    edge_features_test = np.empty((P_test,2))

    for i in range(P_test):
        edge_features_test[i,:] = process_image(x_test[i,:],ET,PT)

    y_pred=model.predict(edge_features_test)
    print(accuracy_score(y_test,y_pred))
    print(confusion_matrix(y_test,y_pred))
```

In [24]:
```
test_function(folder_testing,im_width,ET,PT,model)
```

```
0.8518518518518519
[[21  1  5]
 [ 0 26  1]
 [ 3  2 22]]
```