In [2]:

```
from IPython.display import Image    ## to display images
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler ## for scaling the input data
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_digits       ## mnist data
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

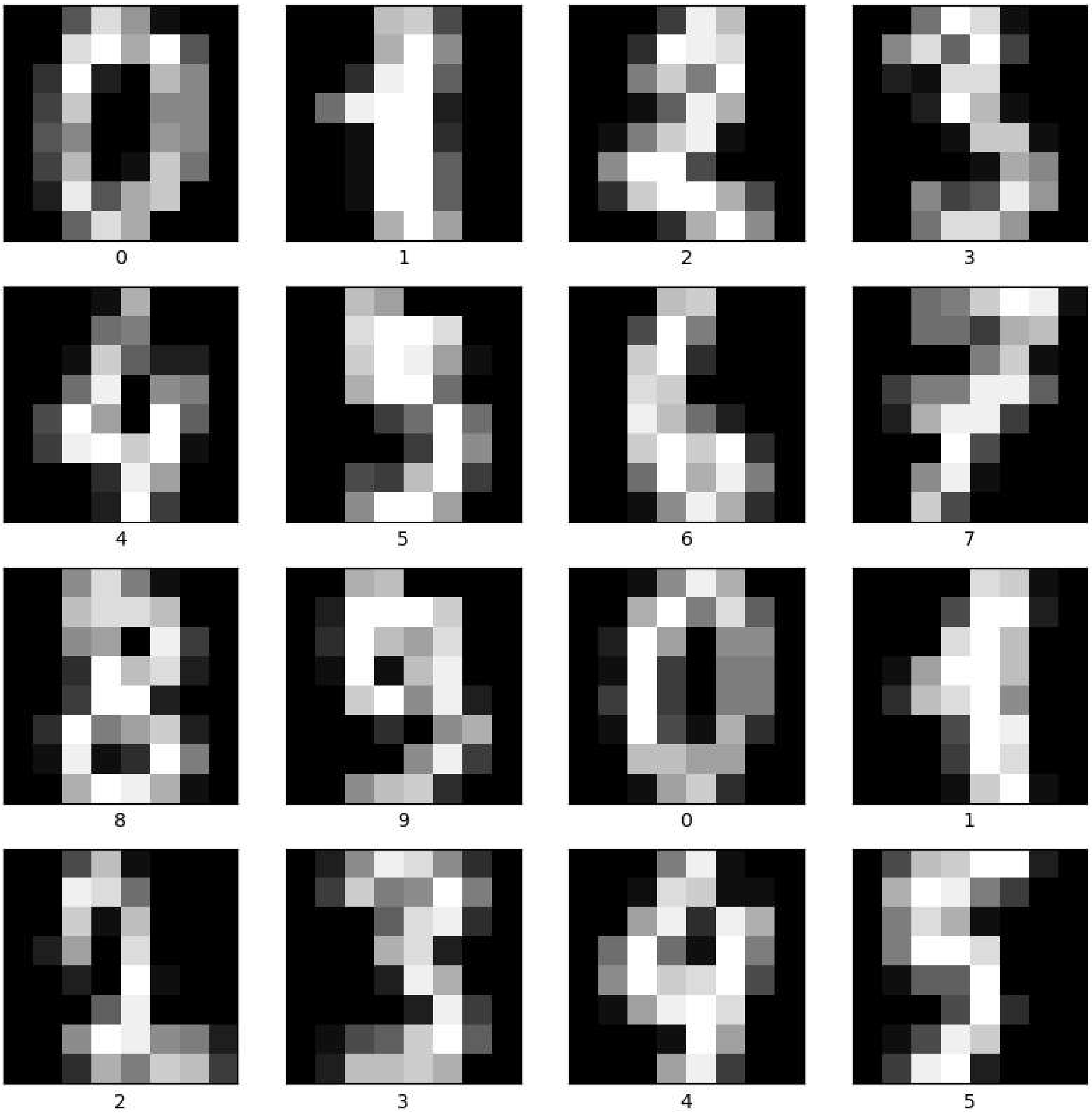
```
## dataset
digits = load_digits()
```

In [5]:

```
data = digits.images ## features from digits
target = digits.target ##labels from digits
```

In [6]:

```
## plotting some images
plt.figure(figsize=(10,10))
for i in range(16):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(data[i], cmap='gray')
    plt.xlabel(target[i])
plt.show()
```



In [7]:

```
print("Data shape is: ",data.shape)
print("Target shape is: ",target.shape)
```

```
Data shape is:  (1797, 8, 8)
Target shape is:  (1797,)
```

In [8]:

```
data = data.reshape((1797,64)) ## reshaping the input befor passing the input to MinMaxScale
```

In [9]:

```
## scaling the input


min_max_sc = MinMaxScaler()
X = min_max_sc.fit_transform(data)
```

In [10]:

```
X_train,X_test,y_train,y_test = train_test_split(X,target,test_size=0.25,random_state=42)
```

In [11]:

```
print("X_train shape: ", X_train.shape)
print("X_test shape: ", X_test.shape)
print("y_train shape: ", y_train.shape)
print("y_test shape: ", y_test.shape)
```

```
X_train shape:  (1347, 64)
X_test shape:  (450, 64)
y_train shape:  (1347,)
y_test shape:  (450,)
```

In [12]:

```
## logistic Regression

lg = LogisticRegression()

## training
lg.fit(X_train,y_train)
```

Out[12]:

```
▾  LogisticRegression  ⓘ ?
LogisticRegression()
```

(https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html)

In [13]:

```
## prediction

pred = lg.predict(X_test)
```

In [14]:

```
print("----------------------------------------------CLassification Report-------------------
print(classification_report(y_test,pred))


print("-------------------------------------------Accuracy Score ---------------------------
print(accuracy_score(y_test,pred))

print("-----------------------------------------------Confusion Matrix ---------------------
plt.figure(figsize=(10,10))
sns.heatmap(confusion_matrix(y_test,pred))
```

```
----------------------------------------------CLassification Report-----------------------
-----------------------
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       0.97      0.95      0.96        37
           2       0.97      1.00      0.99        38
           3       1.00      0.93      0.97        46
           4       1.00      0.98      0.99        55
           5       0.93      0.95      0.94        59
           6       0.98      0.98      0.98        45
           7       1.00      0.98      0.99        41
           8       0.93      0.97      0.95        38
           9       0.92      0.96      0.94        48

    accuracy                           0.97       450
   macro avg       0.97      0.97      0.97       450
weighted avg       0.97      0.97      0.97       450
```

In [15]:

```
pd.DataFrame({'Actual':y_test, 'Predicted':pred}).head(50)
```

Out[15]:

	Actual	Predicted
0	6	6
1	9	9
2	3	3
3	7	7
4	2	2
5	1	2
6	5	5
7	2	2
8	5	5
9	2	2
10	1	1
11	9	9
12	4	4
13	0	0
14	4	4
15	2	2
16	3	3
17	7	7
18	8	8
19	8	8
20	4	4
21	3	3
22	9	9
23	7	7
24	5	5
25	6	6
26	3	3
27	5	5
28	6	6
29	3	3
30	4	4
31	9	9
32	1	1
33	4	4
34	4	4
35	6	6
36	9	9
37	4	4
38	7	7
39	6	6

	Actual	Predicted
40	6	6
41	9	9
42	1	1
43	3	3
44	6	6
45	1	1
46	3	3
47	0	0
48	6	6
49	5	5

```
In [ ]:
```