In [1]:

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
```

In [2]:

```
digits=load_digits()
dir(digits)
```

Out[2]:

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_na
mes']
```

In [3]:

```
digits.data[0]
```

Out[3]:

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

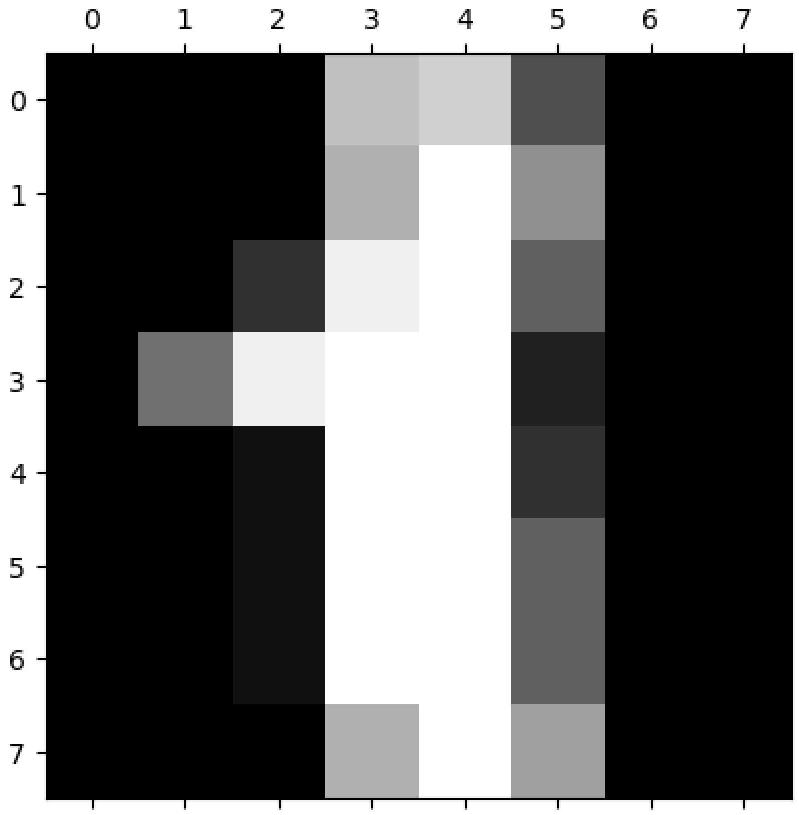
In [4]:

```
plt.gray()
plt.matshow(digits.images[1])
```

Out[4]:

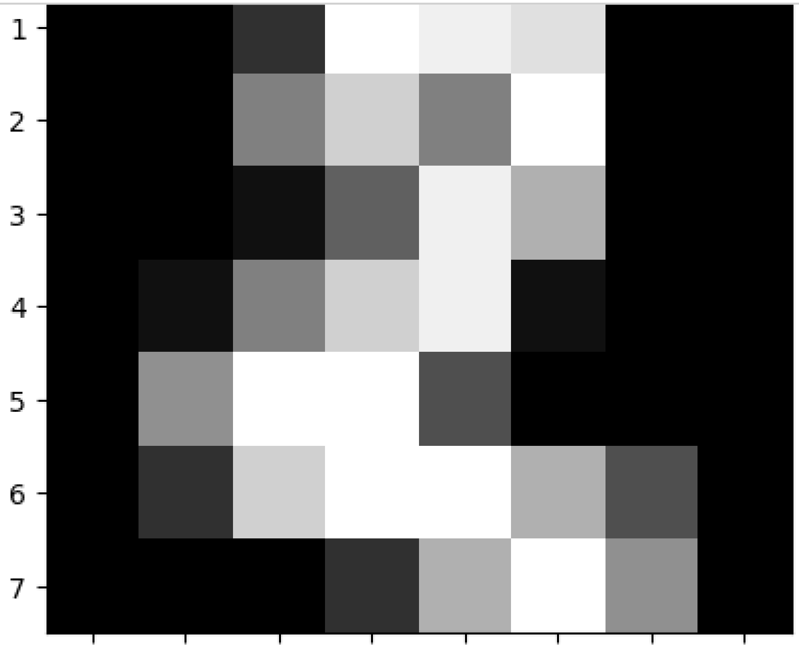
```
<matplotlib.image.AxesImage at 0x22673e52730>
```

```
<Figure size 640x480 with 0 Axes>
```



In [5]:

```
for i in range(5):
    plt.matshow(digits.images[i])
```



In [7]:

```
digits.target[0:5]
```

Out[7]:

```
array([0, 1, 2, 3, 4])
```

In [10]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.2)
```

In [12]:

```
len(X_train)
```

Out[12]:

```
1437
```

In [13]:

```
len(X_test)
```

Out[13]:

```
360
```

In [14]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [15]:

```
model.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[15]:

```
LogisticRegression()
```

In [16]:

```
model.predict(X_test)
```

Out[16]:

```
array([2, 1, 7, 0, 5, 5, 4, 6, 2, 1, 9, 5, 7, 3, 1, 0, 9, 9, 8, 5, 1, 1,
       6, 9, 5, 3, 0, 3, 5, 3, 8, 6, 8, 8, 2, 1, 8, 6, 2, 1, 9, 5, 2, 6,
       6, 2, 8, 2, 0, 6, 7, 2, 8, 5, 2, 7, 9, 6, 5, 5, 4, 9, 6, 7, 8, 1,
       2, 8, 0, 3, 1, 3, 6, 3, 2, 1, 7, 8, 5, 7, 6, 6, 6, 9, 0, 2, 1, 1,
       4, 5, 4, 5, 0, 5, 5, 1, 5, 3, 3, 4, 3, 4, 7, 5, 5, 0, 0, 1, 7, 0,
       3, 1, 5, 6, 3, 4, 1, 9, 9, 6, 8, 5, 1, 4, 8, 9, 9, 1, 8, 6, 9, 7,
       2, 2, 8, 6, 0, 5, 8, 7, 7, 1, 0, 4, 5, 3, 7, 1, 3, 1, 9, 4, 1, 9,
       6, 5, 6, 1, 4, 9, 3, 4, 1, 6, 5, 1, 3, 5, 7, 5, 8, 1, 5, 4, 1, 0,
       1, 3, 4, 2, 1, 3, 9, 9, 2, 2, 8, 4, 5, 7, 0, 0, 7, 5, 4, 3, 6, 0,
       6, 5, 7, 0, 7, 5, 2, 8, 4, 3, 0, 9, 6, 8, 4, 6, 4, 6, 0, 1, 0, 4,
       0, 0, 7, 0, 9, 7, 3, 5, 8, 2, 0, 6, 7, 9, 9, 4, 1, 6, 0, 9, 9, 6,
       5, 9, 4, 6, 8, 0, 8, 5, 2, 0, 3, 0, 0, 7, 8, 7, 7, 0, 9, 2, 4, 7,
       0, 4, 2, 5, 6, 2, 2, 1, 8, 4, 6, 5, 3, 5, 3, 0, 3, 4, 1, 1, 4, 5,
       8, 6, 7, 6, 8, 6, 9, 5, 6, 5, 9, 6, 4, 7, 7, 0, 2, 6, 5, 5, 7, 3,
       6, 2, 8, 7, 0, 7, 2, 0, 4, 3, 7, 1, 1, 8, 8, 3, 9, 5, 6, 3, 7, 2,
       2, 6, 4, 7, 4, 4, 5, 1, 7, 3, 9, 5, 9, 9, 4, 7, 1, 1, 4, 1, 9, 4,
       3, 6, 9, 9, 4, 2, 0, 2])
```

In [17]:

```
y_test
```

Out[17]:

```
array([8, 1, 7, 0, 5, 5, 4, 6, 2, 1, 9, 8, 7, 3, 1, 0, 9, 9, 8, 5, 1, 1,
       6, 9, 5, 3, 0, 3, 5, 3, 8, 6, 9, 8, 2, 1, 8, 6, 2, 1, 9, 5, 2, 6,
       6, 2, 8, 2, 0, 6, 7, 2, 8, 5, 2, 7, 9, 6, 5, 5, 4, 9, 6, 7, 8, 1,
       2, 8, 0, 3, 1, 3, 6, 3, 2, 1, 7, 8, 5, 7, 6, 6, 6, 9, 0, 2, 1, 1,
       4, 5, 4, 5, 0, 5, 5, 1, 5, 3, 3, 4, 3, 4, 7, 5, 5, 0, 0, 1, 7, 0,
       3, 1, 5, 6, 3, 4, 1, 9, 9, 6, 8, 5, 1, 4, 8, 9, 9, 1, 8, 6, 9, 7,
       2, 2, 8, 6, 0, 5, 8, 7, 7, 1, 0, 4, 5, 3, 7, 1, 3, 1, 9, 4, 1, 9,
       6, 5, 6, 1, 4, 9, 3, 4, 1, 6, 5, 1, 3, 5, 7, 5, 8, 1, 5, 4, 1, 0,
       1, 1, 4, 2, 1, 8, 9, 9, 2, 2, 8, 4, 5, 7, 0, 0, 7, 8, 4, 3, 6, 0,
       6, 5, 7, 0, 7, 5, 2, 8, 4, 3, 0, 9, 6, 8, 4, 6, 4, 6, 0, 1, 0, 4,
       0, 0, 7, 0, 9, 7, 3, 5, 8, 2, 0, 6, 7, 9, 9, 4, 1, 6, 0, 9, 9, 6,
       5, 9, 4, 6, 8, 0, 8, 5, 2, 0, 3, 0, 0, 7, 8, 7, 7, 0, 9, 5, 4, 7,
       0, 4, 2, 5, 6, 2, 2, 1, 8, 4, 6, 5, 3, 5, 3, 0, 3, 4, 1, 1, 4, 5,
       8, 6, 7, 6, 8, 6, 9, 5, 6, 5, 9, 6, 4, 7, 7, 0, 2, 6, 5, 8, 7, 3,
       6, 2, 8, 7, 0, 7, 2, 0, 4, 3, 7, 1, 1, 8, 9, 3, 9, 5, 6, 3, 7, 2,
       2, 6, 4, 7, 4, 4, 5, 1, 7, 3, 9, 5, 9, 5, 4, 7, 1, 4, 4, 1, 9, 4,
       3, 6, 9, 4, 4, 2, 0, 2])
```

In [18]:

```
model.score(X_test,y_test)
```

Out[18]:

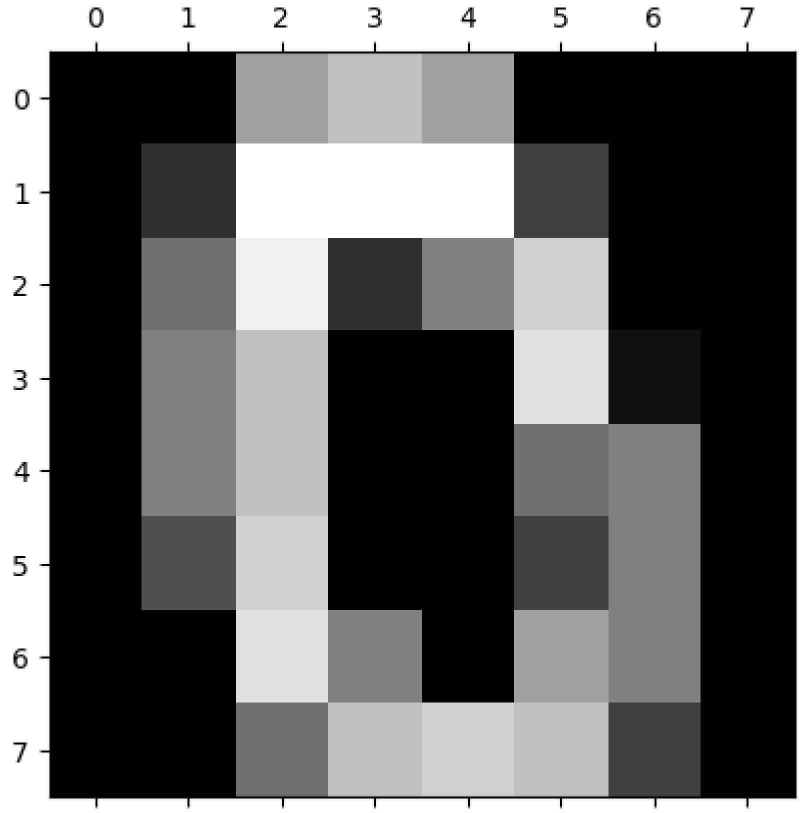
```
0.9666666666666667
```

In [19]:

```
plt.matshow(digits.images[78])
```

Out[19]:

```
<matplotlib.image.AxesImage at 0x22673ef8c40>
```



In [21]:

```
model.predict([digits.data[78]])
```

Out[21]:

```
array([0])
```

In [22]:

```
digits.target[78]
```

Out[22]:

```
0
```

In [23]:

```
model.predict(digits.data[0:5])
```

Out[23]:

```
array([0, 1, 2, 3, 4])
```

In [24]:

```
y_predicted=model.predict(X_test)
y_predicted
```

Out[24]:

```
array([2, 1, 7, 0, 5, 5, 4, 6, 2, 1, 9, 5, 7, 3, 1, 0, 9, 9, 8, 5, 1, 1,
       6, 9, 5, 3, 0, 3, 5, 3, 8, 6, 8, 8, 2, 1, 8, 6, 2, 1, 9, 5, 2, 6,
       6, 2, 8, 2, 0, 6, 7, 2, 8, 5, 2, 7, 9, 6, 5, 5, 4, 9, 6, 7, 8, 1,
       2, 8, 0, 3, 1, 3, 6, 3, 2, 1, 7, 8, 5, 7, 6, 6, 6, 9, 0, 2, 1, 1,
       4, 5, 4, 5, 0, 5, 5, 1, 5, 3, 3, 4, 3, 4, 7, 5, 5, 0, 0, 1, 7, 0,
       3, 1, 5, 6, 3, 4, 1, 9, 9, 6, 8, 5, 1, 4, 8, 9, 9, 1, 8, 6, 9, 7,
       2, 2, 8, 6, 0, 5, 8, 7, 7, 1, 0, 4, 5, 3, 7, 1, 3, 1, 9, 4, 1, 9,
       6, 5, 6, 1, 4, 9, 3, 4, 1, 6, 5, 1, 3, 5, 7, 5, 8, 1, 5, 4, 1, 0,
       1, 3, 4, 2, 1, 3, 9, 9, 2, 2, 8, 4, 5, 7, 0, 0, 7, 5, 4, 3, 6, 0,
       6, 5, 7, 0, 7, 5, 2, 8, 4, 3, 0, 9, 6, 8, 4, 6, 4, 6, 0, 1, 0, 4,
       0, 0, 7, 0, 9, 7, 3, 5, 8, 2, 0, 6, 7, 9, 9, 4, 1, 6, 0, 9, 9, 6,
       5, 9, 4, 6, 8, 0, 8, 5, 2, 0, 3, 0, 0, 7, 8, 7, 7, 0, 9, 2, 4, 7,
       0, 4, 2, 5, 6, 2, 2, 1, 8, 4, 6, 5, 3, 5, 3, 0, 3, 4, 1, 1, 4, 5,
       8, 6, 7, 6, 8, 6, 9, 5, 6, 5, 9, 6, 4, 7, 7, 0, 2, 6, 5, 5, 7, 3,
       6, 2, 8, 7, 0, 7, 2, 0, 4, 3, 7, 1, 1, 8, 8, 3, 9, 5, 6, 3, 7, 2,
       2, 6, 4, 7, 4, 4, 5, 1, 7, 3, 9, 5, 9, 9, 4, 7, 1, 1, 4, 1, 9, 4,
       3, 6, 9, 9, 4, 2, 0, 2])
```

In [27]:

```
from sklearn.metrics import confusion_matrix
```

In [28]:

```
cm=confusion_matrix(y_test,y_predicted)
cm
```

Out[28]:

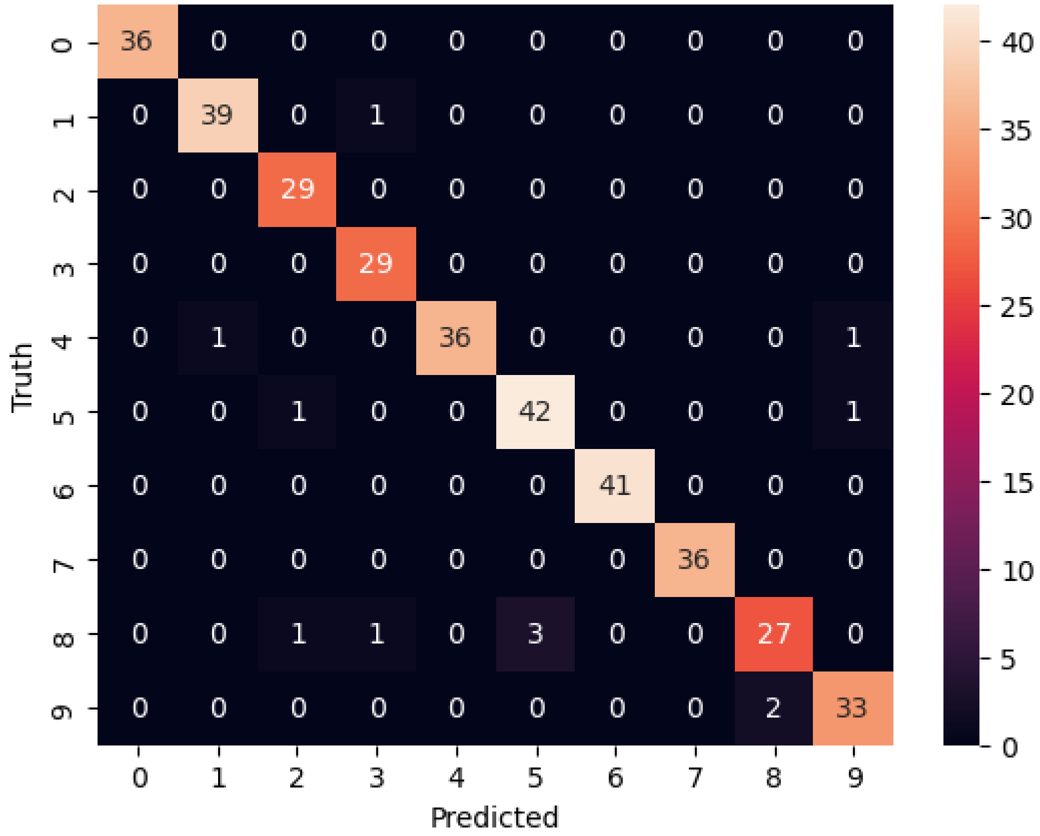
```
array([[36,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 39,  0,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 29,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 29,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0,  0, 36,  0,  0,  0,  0,  1],
       [ 0,  0,  1,  0,  0, 42,  0,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 41,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 36,  0,  0],
       [ 0,  0,  1,  1,  0,  3,  0,  0, 27,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  2, 33]], dtype=int64)
```

In [29]:

```
import seaborn as sn
sn.heatmap(cm,annot=True)
plt.xlabel("Predicted")
plt.ylabel("Truth")
```

Out[29]:

```
Text(50.722222222222214, 0.5, 'Truth')
```



In [30]:

```
model.classification_report_
```

```
------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_20136\1880585890.py in <module>
----> 1 model.classification_report_

AttributeError: 'LogisticRegression' object has no attribute 'classificati
on_report_'
```

In [31]:

```
from sklearn.metrics import classification_report
```

In [32]:

```
print(classification_report(y_test,y_predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        36
           1       0.97      0.97      0.97        40
           2       0.94      1.00      0.97        29
           3       0.94      1.00      0.97        29
           4       1.00      0.95      0.97        38
           5       0.93      0.95      0.94        44
           6       1.00      1.00      1.00        41
           7       1.00      1.00      1.00        36
           8       0.93      0.84      0.89        32
           9       0.94      0.94      0.94        35

    accuracy                           0.97       360
   macro avg       0.97      0.97      0.97       360
weighted avg       0.97      0.97      0.97       360
```

In [ ]:

In [ ]:

In [ ]: