In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [47]:
```
from sklearn.datasets import load_digits
digits = load_digits()
```

In [48]:
```
dir(digits)
```

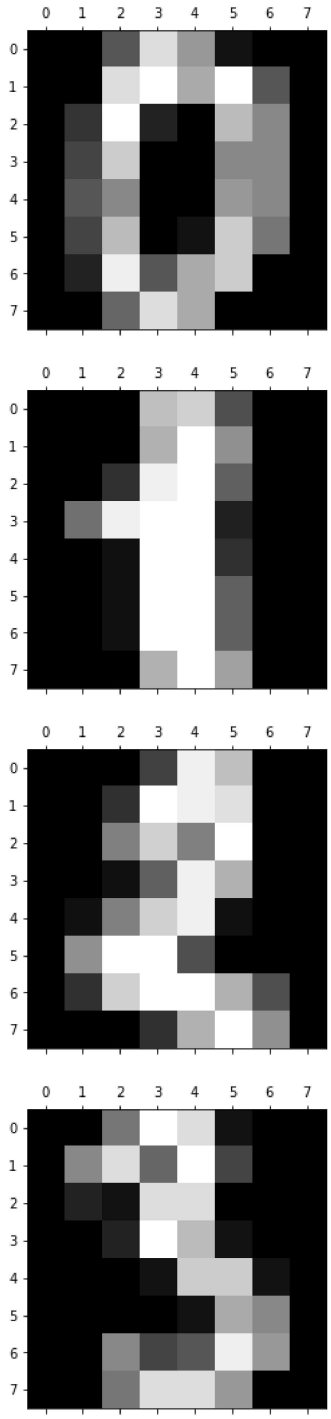
Out[48]:
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

In [49]:
```
digits['images'][:5]
```

Out[49]:
```
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.]],

       [[ 0.,  0.,  0., 12., 13.,  5.,  0.,  0.],
        [ 0.,  0.,  0., 11., 16.,  9.,  0.,  0.],
        [ 0.,  0.,  3., 15., 16.,  6.,  0.,  0.],
        [ 0.,  7., 15., 16., 16.,  2.,  0.,  0.],
        [ 0.,  0.,  1., 16., 16.,  3.,  0.,  0.],
        [ 0.,  0.,  1., 16., 16.,  6.,  0.,  0.],
        [ 0.,  0.,  1., 16., 16.,  6.,  0.,  0.],
        [ 0.,  0.,  0., 11., 16., 10.,  0.,  0.]],

       [[ 0.,  0.,  0.,  4., 15., 12.,  0.,  0.],
        [ 0.,  0.,  3., 16., 15., 14.,  0.,  0.],
        [ 0.,  0.,  8., 13.,  8., 16.,  0.,  0.],
        [ 0.,  0.,  1.,  6., 15., 11.,  0.,  0.],
        [ 0.,  1.,  8., 13., 15.,  1.,  0.,  0.],
        [ 0.,  9., 16., 16.,  5.,  0.,  0.,  0.],
        [ 0.,  3., 13., 16., 16., 11.,  5.,  0.],
        [ 0.,  0.,  0.,  3., 11., 16.,  9.,  0.]],

       [[ 0.,  0.,  7., 15., 13.,  1.,  0.,  0.],
        [ 0.,  8., 13.,  6., 15.,  4.,  0.,  0.],
        [ 0.,  2.,  1., 13., 13.,  0.,  0.,  0.],
        [ 0.,  0.,  2., 15., 11.,  1.,  0.,  0.],
        [ 0.,  0.,  0.,  1., 12., 12.,  1.,  0.],
        [ 0.,  0.,  0.,  0.,  1., 10.,  8.,  0.],
        [ 0.,  0.,  8.,  4.,  5., 14.,  9.,  0.],
        [ 0.,  0.,  7., 13., 13.,  9.,  0.,  0.]],

       [[ 0.,  0.,  0.,  1., 11.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  7.,  8.,  0.,  0.,  0.],
        [ 0.,  0.,  1., 13.,  6.,  2.,  2.,  0.],
        [ 0.,  0.,  7., 15.,  0.,  9.,  8.,  0.],
        [ 0.,  5., 16., 10.,  0., 16.,  6.,  0.],
        [ 0.,  4., 15., 16., 13., 16.,  1.,  0.],
        [ 0.,  0.,  0.,  3., 15., 10.,  0.,  0.],
        [ 0.,  0.,  0.,  2., 16.,  4.,  0.,  0.]]])
```

In [50]:
```
plt.gray()
for i in range(4):
    plt.matshow(digits.images[i])
```

```
<Figure size 432x288 with 0 Axes>
```



In [51]:
```
df=pd.DataFrame(digits.data)
df.head()
```

Out[51]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60	61	62	63
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0	10.0	0.0	0.0	0.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0	16.0	10.0	0.0	0.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0	11.0	16.0	9.0	0.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	7.0	13.0	13.0	9.0	0.0	0.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2.0	16.0	4.0	0.0	0.0

5 rows × 64 columns

In [52]:
```
df['target']=digits.target
```

In [53]:
```
X= df.drop('target', axis='columns')
y=df.target
```

In [54]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)
```

In [55]:
```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
```

Out[55]:
```
RandomForestClassifier(n_estimators=20)
```

In [56]:
```
model.score(X_test,y_test)
```

Out[56]:
```
0.9694444444444444
```

In [57]:
```
y_predicted =model.predict(X_test)
```

In [58]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_predicted)
```

In [59]:
```
cm
```

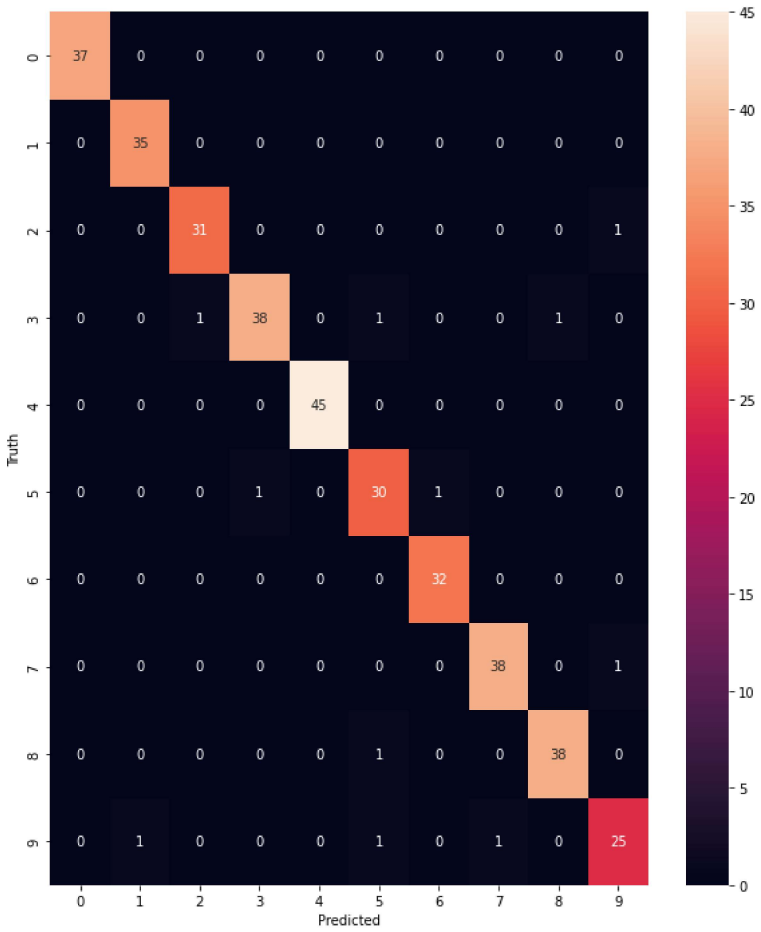
Out[59]:
```
array([[37,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 35,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 31,  0,  0,  0,  0,  0,  0,  1],
       [ 0,  0,  1, 38,  0,  1,  0,  0,  1,  0],
       [ 0,  0,  0,  0, 45,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  1,  0, 30,  1,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 32,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 38,  0,  1],
       [ 0,  0,  0,  0,  0,  1,  0,  0, 38,  0],
       [ 0,  1,  0,  0,  0,  1,  0,  1,  0, 25]], dtype=int64)
```

In [60]:
```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_predicted)
print("Accuracy: ", acc)
```

```
Accuracy:  0.9694444444444444
```

In [61]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.figure(figsize=(10,12))
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[61]:
```
Text(69.0, 0.5, 'Truth')
```



Exercise

In [62]:
```
from sklearn.datasets import load_iris
iris=load_iris()
dir(iris)
```

Out[62]:
```
['DESCR',
 'data',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

In [63]:
```
help(dir)
```

```
Help on built-in function dir in module builtins:

dir(...)
    dir([object]) -> list of strings

    If called without an argument, return the names in the current scope.
    Else, return an alphabetized list of names comprising (some of) the attributes
    of the given object, and of attributes reachable from it.
    If the object supplies a method named __dir__, it will be used; otherwise
    the default dir() logic is used and returns:
      for a module object: the module's attributes.
      for a class object:  its attributes, and recursively the attributes
        of its bases.
      for any other object: its attributes, its class's attributes, and
        recursively the attributes of its class's base classes.
```

In [64]:
```
y=iris['target']
```

In [ ]:
```

```

In [20]:
```
y
```

Out[20]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [21]:
```
X=pd.DataFrame(iris.data, columns=iris.feature_names)
```

In [22]:
```
X.head(1)
```

Out[22]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2

In [23]:
```
type(X.head(1))
```

Out[23]:
```
pandas.core.frame.DataFrame
```

In [24]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.25 )
```

In [25]:
```
len(X_train)
```

Out[25]:
```
112
```

In [26]:
```
len(X_test)
```

Out[26]:
```
38
```

In [27]:
```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=40)
model.fit(X_train, y_train)
```

Out[27]:
```
RandomForestClassifier(n_estimators=40)
```

In [28]:
```
model.score(X_test,y_test)
```

Out[28]:
```
0.9210526315789473
```

In [29]:
```
model.predict(X_test)
```

Out[29]:
```
array([1, 0, 0, 0, 0, 1, 0, 2, 2, 1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 2, 0, 1, 1, 1, 2, 2, 0, 1, 1])
```

In [30]:
```
y_predicted=model.predict(X_test)
```

In [38]:
```
type(X_test)
```

Out[38]:
```
pandas.core.frame.DataFrame
```

In [40]:
```
y_test
```

Out[40]:
```
array([1, 0, 0, 0, 0, 1, 0, 2, 2, 2, 0, 2, 1, 1, 0, 2, 2, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 2, 2, 0, 1, 1, 1, 2, 2, 0, 1, 1])
```

In [41]:
```
y_predicted
```

Out[41]:
```
array([1, 0, 0, 0, 0, 1, 0, 2, 2, 1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 2, 0, 1, 1, 1, 2, 2, 0, 1, 1])
```

In [43]:
```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_predicted)
print("Accuracy: ", acc)
```

```
Accuracy:  0.9210526315789473
```

In [ ]:
```

```