Implementation of principal component analysis

In [5]:
```
from sklearn.datasets import load_digits
import pandas as pd
dataset=load_digits()
dataset.keys()
```

Out[5]:
```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'image
s', 'DESCR'])
```

In [6]:
```
dataset.data.shape
```

Out[6]:
```
(1797, 64)
```

In [7]:
```
dataset.data[0]
```

Out[7]:
```
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.])
```

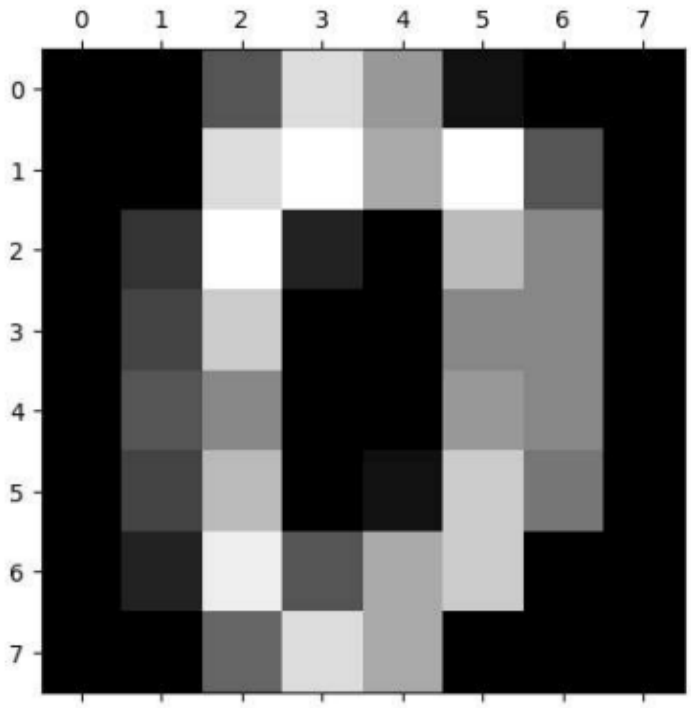
In [8]:
```
dataset.data[0].reshape(8,8)
```

Out[8]:
```
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 [13]:
```
from matplotlib import pyplot as plt
%matplotlib inline
plt.gray()
plt.matshow(dataset.data[0].reshape(8,8))
```

Out[13]:
```
<matplotlib.image.AxesImage at 0x23b84aa9ff0>

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



In [14]:
```
dataset.target[:5]
```

Out[14]:
```
array([0, 1, 2, 3, 4])
```

In [16]:
```
df=pd.DataFrame(dataset.data,columns=dataset.feature_names)
df.head()
```

Out[16]:

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_(
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.(
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.(
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.(
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.(
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.(

5 rows × 64 columns

In [17]:
```
dataset.target
```

Out[17]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [18]:
```
df.describe()
```

Out[18]:

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6
count	1797.0	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000
mean	0.0	0.303840	5.204786	11.835838	11.848080	5.781859	1.362270
std	0.0	0.907192	4.754826	4.248842	4.287388	5.666418	3.325775
min	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.0	0.000000	1.000000	10.000000	10.000000	0.000000	0.000000
50%	0.0	0.000000	4.000000	13.000000	13.000000	4.000000	0.000000
75%	0.0	0.000000	9.000000	15.000000	15.000000	11.000000	0.000000
max	0.0	8.000000	16.000000	16.000000	16.000000	16.000000	16.000000

8 rows × 64 columns

In [19]:
```
X=df
y=dataset.target
```

In [20]:
```
from sklearn.preprocessing import StandardScaler
```

In [22]:
```
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X_scaled
```

Out[22]:
```
array([[ 0.        , -0.33501649, -0.04308102, ..., -1.14664746,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, ...,  0.54856067,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, ...,  1.56568555,
         1.6951369 , -0.19600752],
       ...,
       [ 0.        , -0.33501649, -0.88456568, ..., -0.12952258,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649, -0.67419451, ...,  0.8876023 ,
        -0.5056698 , -0.19600752],
       [ 0.        , -0.33501649,  1.00877481, ...,  0.8876023 ,
        -0.26113572, -0.19600752]])
```

In [23]:
```
from sklearn.model_selection import train_test_split
```

In [24]:
```
X_train, X_test, y_train, y_test=train_test_split(X_scaled,y,test_size=0.2,ran
```

In [25]:
```
from sklearn.linear_model import LogisticRegression
```

In [27]:
```
model=LogisticRegression()
model.fit(X_train,y_train)
model.score(X_test,y_test)
```

Out[27]:
```
0.9722222222222222
```

In [28]:
```
from sklearn.decomposition import PCA
```

In [29]:
```
pca=PCA(0.95)
X_pca=pca.fit_transform(X)
X_pca.shape
```

Out[29]:
```
(1797, 29)
```

In [35]:
```
X_train_pca, X_test_pca, y_train, y_test=train_test_split(X_pca,y,test_size=0.
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
```

In [38]:
```
model=LogisticRegression(max_iter=1000)
model.fit(X_train_pca,y_train)
model.score(X_test_pca,y_test)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
458: 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 in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[38]:
```
0.9694444444444444
```

In [39]:
```
pca=PCA(n_components=2)
X_pca=pca.fit_transform(X)
X_pca.shape
```

Out[39]:
```
(1797, 2)
```

In [40]:
```
X_train_pca, X_test_pca, y_train, y_test=train_test_split(X_pca,y,test_size=0.
```

In [41]:
```
model=LogisticRegression(max_iter=1000)
model.fit(X_train_pca,y_train)
model.score(X_test_pca,y_test)
```

Out[41]:
```
0.6083333333333333
```