In [9]:

```
#classification model(logistic regression)
```

In [10]:

```
from sklearn.datasets import load_digits
```

In [11]:

```
digits = load_digits()
```

In [12]:

```
print(digits.DESCR) #description
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 5620
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in th
e range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written dig
its datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+o
f+Handwritten+Digits

The data set contains images of hand-written digits: 10 class
es where
each class refers to a digit.

Preprocessing programs made available by NIST were used to ex
tract
normalized bitmaps of handwritten digits from a preprinted fo
rm. From a
total of 43 people, 30 contributed to the training set and di
fferent 13
to the test set. 32x32 bitmaps are divided into nonoverlappin
g blocks of
4x4 and the number of on pixels are counted in each block. Th
is generates
an input matrix of 8x8 where each element is an integer in th
e range
0..16. This reduces dimensionality and gives invariance to sm
all
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J.
```

```
L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Jan
et, and C.

L. Wilson, NIST Form-Based Handprint Recognition System, NIST
IR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifier
s and Their
    Applications to Handwritten Digit Recognition, MSc Thesi
s, Institute of
    Graduate Studies in Science and Engineering, Bogazici Uni
versity.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybe
rnetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. K
ai Qin.
    Linear dimensionalityreduction using relevance weighted L
DA. School of
    Electrical and Electronic Engineering Nanyang Technologic
al University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classif
ication
    Algorithm. NIPS. 2000.
```

In [13]:

```
import matplotlib.pyplot as plt
```

In [14]:

```
digits.data
```

Out[14]:

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

In [15]:

```
digits.data.shape
```

Out[15]:

```
(1797, 64)
```

In [16]:

```
digits.target
```

Out[16]:

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

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

In [17]:

```
digits.target.shape
```

Out[17]:

```
(1797,)
```

In [18]:

```
image = digits.data[0]
```

In [19]:

```
print(image)
```

```
[ 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 [20]:

```
import numpy as np
```

In [21]:

```
np.reshape(image, (8,8))
```

Out[21]:

```
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 [22]:

```
plt.imshow(np.reshape(image, (8,8)))
```

Out[22]:

```
<matplotlib.image.AxesImage at 0x1ee057106c8>
```

In [23]:

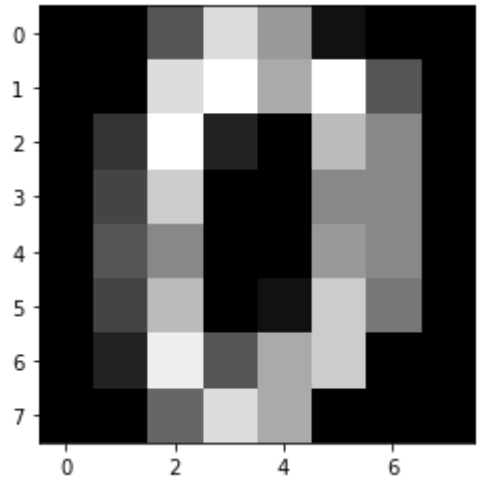
```
#zero
```

In [24]:

```
plt.imshow(np.reshape(image, (8,8)), cmap="gray")
```

Out[24]:

```
<matplotlib.image.AxesImage at 0x1ee06f33508>
```

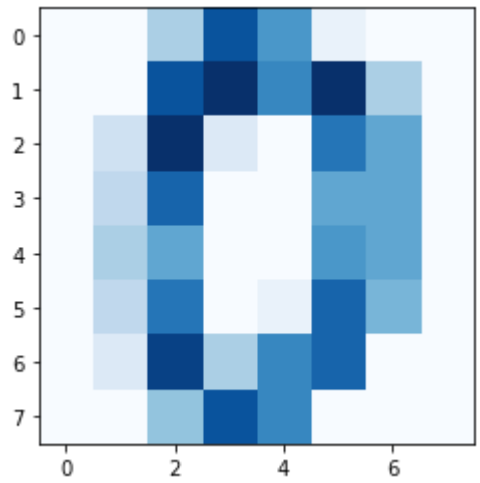


In [25]:

```
plt.imshow(np.reshape(image, (8,8)), cmap="Blues")
```

Out[25]:

```
<matplotlib.image.AxesImage at 0x1ee06f96e08>
```



In [26]:

```
digits.target[1500] #shows the digit the target value is assigned to
```

Out[26]:

```
1
```

In [27]:

```
image = digits.data[1500]
```

In [28]:

```
print(image)
```

```
[ 0.  0.  0.  3. 12. 12.  2.  0.  0.  0.  7. 15. 16. 16.  0.
0.  0.  4.
 15.  9. 14. 16.  3.  0.  0.  2.  0.  0. 14. 16.  0.  0.  0.
0.  0.  0.
 14. 16.  0.  0.  0.  0.  0.  0. 15. 13.  0.  0.  0.  0.  0.
0. 16. 14.
  1.  0.  0.  0.  0.  3. 16. 13.  2.  0.]
```

In [29]:

```
np.reshape(image, (8,8))
```

Out[29]:

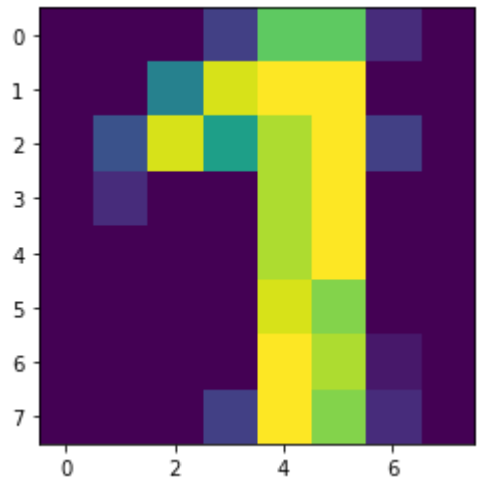
```
array([[ 0.,  0.,  0.,  3., 12., 12.,  2.,  0.],
       [ 0.,  0.,  7., 15., 16., 16.,  0.,  0.],
       [ 0.,  4., 15.,  9., 14., 16.,  3.,  0.],
       [ 0.,  2.,  0.,  0., 14., 16.,  0.,  0.],
       [ 0.,  0.,  0.,  0., 14., 16.,  0.,  0.],
       [ 0.,  0.,  0.,  0., 15., 13.,  0.,  0.],
       [ 0.,  0.,  0.,  0., 16., 14.,  1.,  0.],
       [ 0.,  0.,  0.,  3., 16., 13.,  2.,  0.]])
```

In [30]:

```
plt.imshow(np.reshape(image, (8,8)))
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x1ee07007488>
```

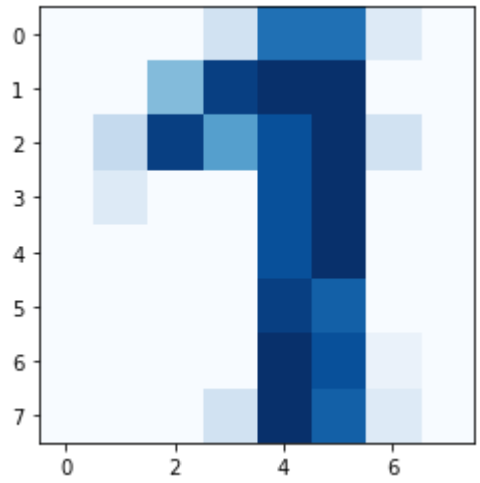


In [31]:

```
plt.imshow(np.reshape(image, (8,8)), cmap="Blues")
```

Out[31]:

```
<matplotlib.image.AxesImage at 0x1ee07068a48>
```



In [32]:

```
digits.target[1795]
```

Out[32]:

```
9
```

In [33]:

```
image = digits.data[1795]
```

In [34]:

```
print(image)
```

```
[ 0.  0.  2. 10.  7.  0.  0.  0.  0.  0. 14. 16. 16. 15.  1.
0.  0.  4.
 16.  7.  3. 16.  7.  0.  0.  5. 16. 10.  7. 16.  4.  0.  0.
0.  5. 14.
 14. 16.  4.  0.  0.  0.  0.  0.  0. 16.  2.  0.  0.  0.  4.
7.  7. 16.
  2.  0.  0.  0.  5. 12. 16. 12.  0.  0.]
```

In [35]:

```
np.reshape(image, (8,8))
```

Out[35]:

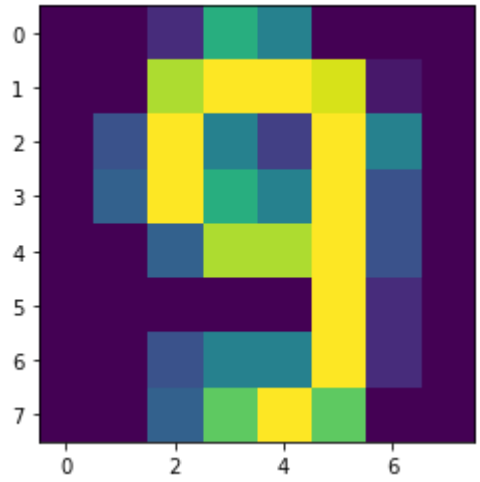
```
array([[ 0.,  0.,  2., 10.,  7.,  0.,  0.,  0.],
       [ 0.,  0., 14., 16., 16., 15.,  1.,  0.],
       [ 0.,  4., 16.,  7.,  3., 16.,  7.,  0.],
       [ 0.,  5., 16., 10.,  7., 16.,  4.,  0.],
       [ 0.,  0.,  5., 14., 14., 16.,  4.,  0.],
       [ 0.,  0.,  0.,  0.,  0., 16.,  2.,  0.],
       [ 0.,  0.,  4.,  7.,  7., 16.,  2.,  0.],
       [ 0.,  0.,  5., 12., 16., 12.,  0.,  0.]])
```

In [36]:

```
plt.imshow(np.reshape(image, (8,8)))
```

Out[36]:

```
<matplotlib.image.AxesImage at 0x1ee070d1748>
```



In [37]:

```
from sklearn.model_selection import train_test_split
```

In [38]:

```
x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.ta
rget, test_size=0.2)
```

In [39]:

```
from sklearn.linear_model import LogisticRegression
```

In [67]:

```
logReg1 =  LogisticRegression()
logReg1 = LogisticRegression(solver='lbfgs')
```

In [68]:

```
logReg1.fit(x_train, y_train)
```

```
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:469: FutureWarning: Default multi_class will
be changed to 'auto' in 0.22. Specify the multi_class option
to silence this warning.
  "this warning.", FutureWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
```

```
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)

C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
```

Out[68]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_
intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_it
er=100,
                   multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='lbfgs', tol=0.0
001, verbose=0,
                   warm_start=False)
```

In [69]:

```
logReg2 = LogisticRegression(solver='lbfgs', max_iter=1000)
```

In [70]:

```
logReg2.fit(x_train, y_train)
```

```
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:469: FutureWarning: Default multi_class will
be changed to 'auto' in 0.22. Specify the multi_class option
to silence this warning.
  "this warning.", FutureWarning)
```

Out[70]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_
intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_it
er=1000,
                   multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='lbfgs', tol=0.0
001, verbose=0,
                   warm_start=False)
```

In [71]:

```
logreg3 = LogisticRegression(solver='lbfgs', max_iter=100000, multi_class=
'auto')
```

In [72]:

```
logreg3.fit(x_train, y_train)
```

Out[72]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_
intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_it
er=100000,
                   multi_class='auto', n_jobs=None, penalty
='l2',
                   random_state=None, solver='lbfgs', tol=0.0
001, verbose=0,
                   warm_start=False)
```

In [73]:

```
logreg3.score(x_test, y_test)
```

Out[73]:

```
0.9722222222222222
```

In [74]:

```
logreg4 = LogisticRegression(max_iter=1500)
```

In [75]:

```
logreg4.fit(x_train, y_train)
```

```
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:432: FutureWarning: Default solver will be c
hanged to 'lbfgs' in 0.22. Specify a solver to silence this w
arning.
  FutureWarning)
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:469: FutureWarning: Default multi_class will
be changed to 'auto' in 0.22. Specify the multi_class option
to silence this warning.
  "this warning.", FutureWarning)
```

Out[75]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_
intercept=True,
```

```
                   intercept_scaling=1, l1_ratio=None, max_it
er=1500,
                   multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='warn', tol=0.00
01, verbose=0,
                   warm start=False)
```

In [77]:

```
logreg4.score(x_test, y_test)
```

Out[77]:

```
0.975
```

x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.3)

In [78]:

```
logreg6 = LogisticRegression(solver='lbfgs', max_iter=5000, multi_class='a
uto')
```

In [79]:

```
logreg6.fit(x_train, y_train)
```

```
C:\Users\Prerana\Anaconda3\lib\site-packages\sklearn\linear_m
odel\logistic.py:947: ConvergenceWarning: lbfgs failed to con
verge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
```

Out[79]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_
intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_it
er=5000,
                   multi_class='auto', n_jobs=None, penalty
='l2',
                   random_state=None, solver='lbfgs', tol=0.0
001, verbose=0,
                   warm_start=False)
```

In [81]:

```
logreg6.score(x_test, y_test)
```

Out[81]:

```
0.9722222222222222
```

In [ ]:

```

```