

Logistic Regression for Classification - target takes specific values only

```
In [23]: from sklearn.datasets import load_digits
```

```
In [24]: digits = load_digits()
```

```
In [25]: print(digits.DESCR)
```

```
.. _digits_dataset:
```

Optical recognition of handwritten digits dataset

****Data Set Characteristics:****

```
:Number of Instances: 1797
:Number of Attributes: 64
:Attribute Information: 8x8 image of integer pixels in the 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 digits datasets <https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits> (<https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>)

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

Preprocessing programs made available by NIST were used to extract normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.

For info on NIST preprocessing routines, see M. D. Garriss, J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469, 1994.

.. topic:: References

- C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition, MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.
- E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
- Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin. Linear dimensionality reduction using relevance weighted LDA. School of Electrical and Electronic Engineering Nanyang Technological University. 2005.
- Claudio Gentile. A New Approximate Maximal Margin Classification Algorithm. NIPS. 2000.

```
In [26]: import matplotlib.pyplot as plt
```

```
In [27]: #shows 2d data, each row contains multiple columns with meaning  
digits.data
```

```
Out[27]: 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 [28]: #dimension of actual data  
digits.data.shape
```

```
Out[28]: (1797, 64)
```

```
In [29]: d=digits.data[0:500]
```

```
In [30]: d.shape
```

```
Out[30]: (500, 64)
```

```
In [31]: #target or label dimension  
digits.target.shape
```

```
Out[31]: (1797,)
```

```
In [32]: #taking only the first image pixel data  
image = digits.data[10]
```

```
In [33]: #image is represented as collection of 64 pixel values  
print(image)
```

```
[ 0.  0.  1.  9. 15. 11.  0.  0.  0.  0. 11. 16.  8. 14.  6.  0.  0.  2.  
 16. 10.  0.  9.  9.  0.  0.  1. 16.  4.  0.  8.  8.  0.  0.  4. 16.  4.  
  0.  8.  8.  0.  0.  1. 16.  5.  1. 11.  3.  0.  0.  0. 12. 12. 10. 10.  
  0.  0.  0.  0.  1. 10. 13.  3.  0.  0.]
```

```
In [34]: #determine class of the 64 pixels stored in image  
digits.target[10]
```

```
Out[34]: 0
```

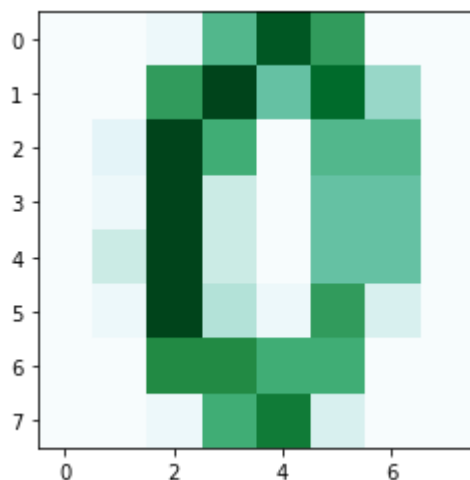
```
In [35]: import numpy as np
```

```
In [36]: np.reshape(image,(8,8))
```

```
Out[36]: array([[ 0.,  0.,  1.,  9., 15., 11.,  0.,  0.],  
               [ 0.,  0., 11., 16.,  8., 14.,  6.,  0.],  
               [ 0.,  2., 16., 10.,  0.,  9.,  9.,  0.],  
               [ 0.,  1., 16.,  4.,  0.,  8.,  8.,  0.],  
               [ 0.,  4., 16.,  4.,  0.,  8.,  8.,  0.],  
               [ 0.,  1., 16.,  5.,  1., 11.,  3.,  0.],  
               [ 0.,  0., 12., 12., 10., 10.,  0.,  0.],  
               [ 0.,  0.,  1., 10., 13.,  3.,  0.,  0.]])
```

```
In [37]: plt.imshow(np.reshape(image, (8,8)), cmap='BuGn')
```

```
Out[37]: <matplotlib.image.AxesImage at 0x7fe9324cbca0>
```



```
In [38]: from sklearn.model_selection import train_test_split
```

```
In [39]: #split the dataset using train_test_split function, pass train data, labels, and  
x_train, x_test, y_train, y_test = train_test_split(digits.data,digits.target,tes
```

```
In [40]: from sklearn.linear_model import LogisticRegression
```

```
In [41]: logReg1 = LogisticRegression(max_iter=10000)
```

```
In [42]: logReg1.fit(x_train, y_train)
```

```
Out[42]: LogisticRegression(max_iter=10000)
```

```
In [43]: y_pred=logReg1.predict(x_test)
```

```
In [44]: logReg1.score(x_test,y_test)
```

```
Out[44]: 0.9666666666666667
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```