Logistic Regression: Multiclass Classification

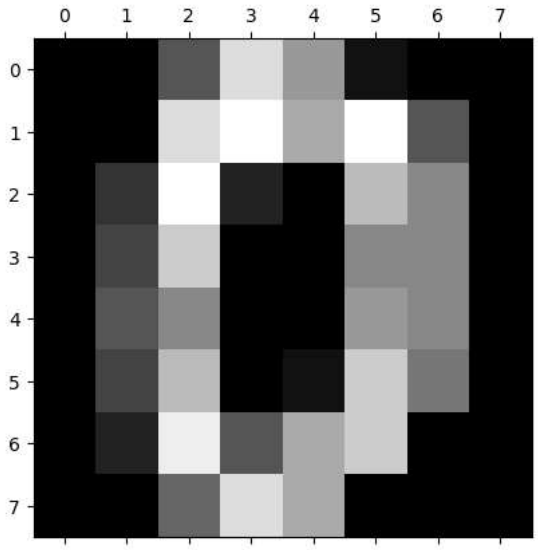
The Digit Dataset

This dataset is made up of 1797 8x8 images. Each image, like the one shown below, is of a hand-written digit. In order to utilize an 8x8 figure like this, we'd have to first transform it into a feature vector with length 64.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.datasets import load_digits
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
digits = load_digits()
```

```
plt.gray()
plt.matshow(digits.images[0])
```

```
<matplotlib.image.AxesImage at 0x7f0bac27ec90>
<Figure size 640x480 with 0 Axes>
```



digits.data - A 2D array of shape (n_samples, n_features) — the flattened pixel values (64 features for 8x8 images).
digits.target - The actual digit labels (0 to 9) for each image (used as class labels).
digits.images - A 3D array of shape (n_samples, 8, 8) — the original 8x8 grayscale images.
digits.target_names - An array: [0, 1, 2, ..., 9] — class names (digits).

```
dir(digits)
```

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

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

```
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.]])
```

```
model = LogisticRegression(max_iter=2000)
```

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

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

```
LogisticRegression
LogisticRegression(max_iter=2000)
```

Measure accuracy of our model

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

```
0.9527777777777777
```

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

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

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

Confusion Matrix

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

```
array([[39,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 32,  0,  0,  0,  1,  0,  0,  2,  0],
       [ 0,  3, 34,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 30,  0,  0,  0,  0,  0,  1],
       [ 0,  0,  0,  0, 29,  0,  0,  0,  1,  0],
       [ 0,  0,  0,  0,  0, 32,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1, 40,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  1,  0, 44,  0,  0],
       [ 0,  1,  0,  1,  0,  0,  0,  0, 37,  3],
       [ 0,  0,  0,  0,  0,  0,  0,  1,  0, 26]])
```

```
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(95.72222222222221, 0.5, 'Truth')
```