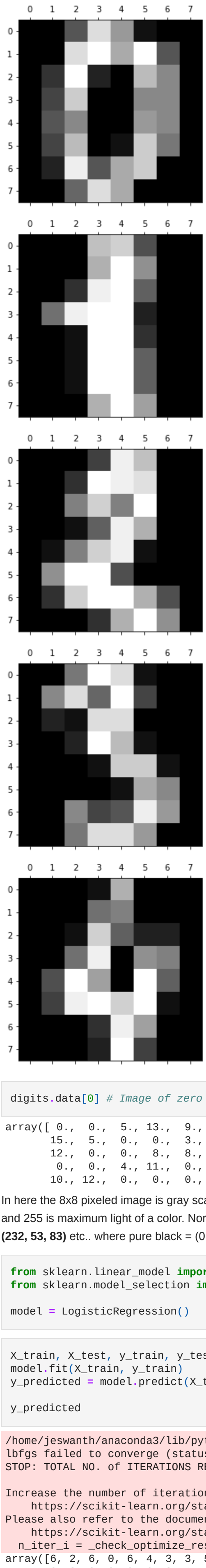
Multi-class Logistic Regression

In [1]:
```
from sklearn.datasets import load_digits
%matplotlib inline
import matplotlib.pyplot as plt
digits = load_digits()
```

In [2]:
```
plt.gray()
for i in range(5):
    plt.matshow(digits.images[i])
```

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



In [3]:
```
digits.data[0] # Image of zero matrix as pixels.
```

Out[3]:
```
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 here the 8x8 pixeled image is gray scaled so we can only get **2 - 8 bit of color**. Eg. 8 bit = 2^8. 8 bit ranges from (0-255) where 0 is black and 255 is maximum light of a color. Normally a pixel contains three colors **(Red, Green, Blue) ranges from 0-22**. **Eg. (255, 255, 255), (232, 53, 83)** etc.. where pure black = (0, 0, 0) and pure white = (255, 255, 255).

In [4]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

model = LogisticRegression()
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(digits.data,digits.target, test_size=0.1)
model.fit(X_train, y_train)
y_predicted = model.predict(X_test)

y_predicted
```

```
/home/jeswanth/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[5]:
```
array([6, 2, 6, 0, 6, 4, 3, 3, 5, 2, 2, 2, 0, 2, 2, 4, 4, 8, 3, 6, 9, 0,
       6, 2, 2, 3, 9, 6, 3, 1, 6, 6, 6, 8, 5, 9, 9, 2, 7, 6, 8, 6, 5, 6,
       1, 2, 8, 7, 1, 8, 2, 1, 3, 2, 2, 9, 0, 4, 6, 3, 1, 9, 9, 2, 9, 4,
       2, 1, 5, 5, 4, 8, 5, 4, 4, 8, 8, 3, 4, 3, 7, 9, 1, 6, 5, 0, 9, 9,
       5, 3, 1, 4, 0, 7, 7, 3, 7, 7, 1, 8, 9, 3, 2, 2, 7, 2, 0, 7, 8, 7,
       1, 6, 8, 2, 0, 3, 7, 9, 7, 5, 4, 0, 5, 0, 8, 6, 1, 3, 5, 2, 1, 0,
       6, 6, 0, 0, 2, 2, 7, 3, 8, 5, 7, 2, 4, 4, 0, 7, 4, 5, 8, 4, 9, 8,
       3, 4, 3, 7, 4, 0, 9, 4, 1, 9, 2, 9, 5, 1, 9, 4, 5, 9, 9, 0, 6, 2,
       6, 0, 0, 1])
```