In [2]:
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

In [3]:
```python
digits = datasets.load_digits()
X = digits.images
y = digits.target
```

In [4]:
```python
n_samples = len(X)
X = X.reshape((n_samples, -1))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)
```

In [5]:
```python
model = MLPClassifier(hidden_layer_sizes=(50,), max_iter=1000)
model.fit(X_train, y_train)
```

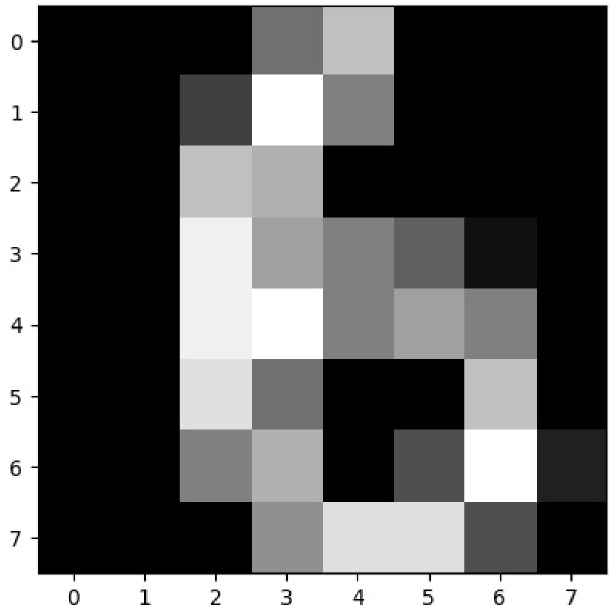
Out[5]:
```
▾                MLPClassifier
MLPClassifier(hidden_layer_sizes=(50,), max_iter=1000)
```

In [6]:
```python
y_pred = model.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[82  0  0  0  0  0  0  0  0  0]
 [ 0 87  1  0  0  0  0  0  1  0]
 [ 0  0 82  0  0  0  0  0  1  0]
 [ 1  0  1 81  0  1  0  5  4  0]
 [ 0  0  0  0 92  0  1  0  0  0]
 [ 0  1  0  0  0 92  1  0  0  5]
 [ 0  0  0  0  2  0 96  0  0  0]
 [ 0  0  0  0  0  1  0 86  0  0]
 [ 0  3  0  0  0  1  0  0 78  1]
 [ 0  0  0  1  0  0  0  0  1 90]]

Classification Report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99        82
           1       0.96      0.98      0.97        89
           2       0.98      0.99      0.98        83
           3       0.99      0.87      0.93        93
           4       0.98      0.99      0.98        93
           5       0.97      0.93      0.95        99
           6       0.98      0.98      0.98        98
           7       0.95      0.99      0.97        87
           8       0.92      0.94      0.93        83
           9       0.94      0.98      0.96        92

    accuracy                           0.96       899
   macro avg       0.96      0.96      0.96       899
weighted avg       0.96      0.96      0.96       899
```

In [8]:
```python
# Example prediction using a single image
example_image = X_test[0]
predicted_digit = model.predict([example_image])
print("Predicted Digit:", predicted_digit[0])

# Display the image
plt.imshow(example_image.reshape(8, 8), cmap=plt.cm.gray)
plt.show()
```

```
Predicted Digit: 6
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js