



```
In [8]: import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
```

```
In [9]: # Load the digits dataset
digits = datasets.load_digits()
```

```
In [10]: # Split the data into features (X) and labels (Y)
x = digits.data
y = digits.target
print("X: \n", x)
print()
print("Y: ", y)
```
















```
X:
[[ 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.]]
```

```
Y: [0 1 2 ... 8 9 8]
```

```
In [11]: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
In [12]: # Create an SVM classifier
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, y_train)
```

Out[12]:

SVC		
Parameters		
	C	1.0
	kernel	'linear'
	degree	3
	gamma	'scale'
	coef0	0.0
	shrinking	True
	probability	False
	tol	0.001
	cache_size	200
	class_weight	None
	verbose	False
	max_iter	-1
	decision_function_shape	'ovr'
	break_ties	False
	random_state	None

```
In [13]: y_pred = classifier.predict(X_test)
```

```
In [14]: # Calculate accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
```

Accuracy: 0.9777777777777777

```
In [15]: # Confusion matrix
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
print("Confusion Matrix: ")
print(confusion_matrix)
```

Confusion Matrix:

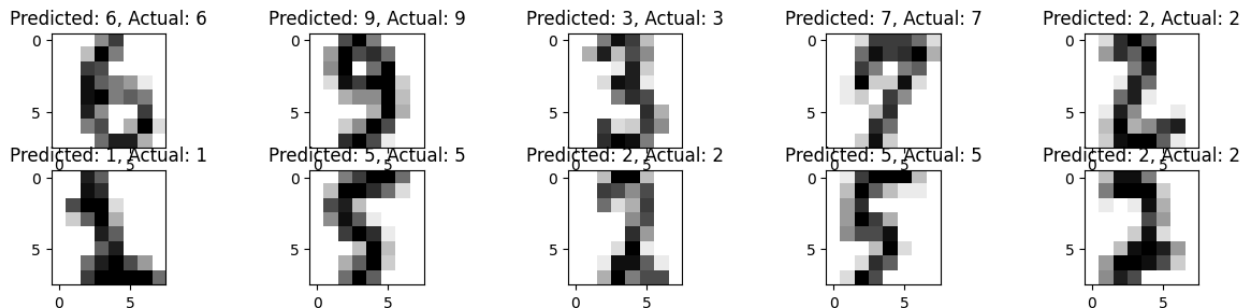
```
[[33  0  0  0  0  0  0  0  0  0]
 [ 0 28  0  0  0  0  0  0  0  0]
 [ 0  0 33  0  0  0  0  0  0  0]
 [ 0  0  0 32  0  1  0  0  0  1]
 [ 0  1  0  0 45  0  0  0  0  0]
 [ 0  0  0  0  0 47  0  0  0  0]
 [ 0  0  0  0  0  0 35  0  0  0]
 [ 0  0  0  0  0  0  0 33  0  1]
 [ 0  0  0  0  0  1  0  0 29  0]
 [ 0  0  0  1  1  0  0  1  0 37]]
```

```
In [16]: # Classification report
classification_report = metrics.classification_report(y_test, y_pred)
print("Classification Report: ")
print(classification_report)
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	33
1	0.97	1.00	0.98	28
2	1.00	1.00	1.00	33
3	0.97	0.94	0.96	34
4	0.98	0.98	0.98	46
5	0.96	1.00	0.98	47
6	1.00	1.00	1.00	35
7	0.97	0.97	0.97	34
8	1.00	0.97	0.98	30
9	0.95	0.93	0.94	40
accuracy			0.98	360
macro avg	0.98	0.98	0.98	360
weighted avg	0.98	0.98	0.98	360

```
In [21]: # Visualize the images
plt.figure(figsize=(15,8))
for i in range(10):
    plt.subplot(5, 5, i + 1)
    plt.imshow(X_test[i].reshape(8, 8), cmap=plt.cm.gray_r)
    plt.title(f"Predicted: {y_pred[i]}, Actual: {y_test[i]}")
    plt.axis('on')
```



In []: