In [1]:

```
from sklearn.datasets import load_digits
digits = load_digits()
```

In [3]:

```
print("Image Data Shape", digits.data.shape)

print("Label Data Shape", digits.target.shape)
```

```
Image Data Shape (1797, 64)
Label Data Shape (1797,)
```

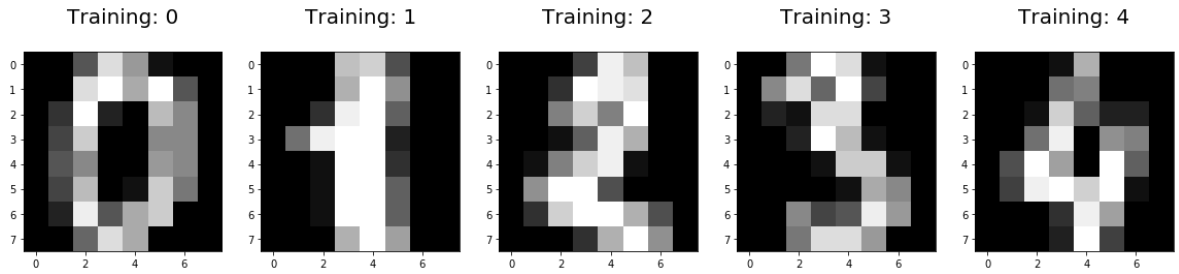
In [5]:

```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(20,4))

for index, (image, label) in enumerate(zip(digits.data[0:5], digits.target[0:5])):
    plt.subplot(1, 5, index + 1)
    plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
    plt.title('Training: %i\n' % label, fontsize = 20)

plt.show()
```



In [6]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(digits.data,
                                                    digits.target,
                                                    test_size=0.25,
                                                    random_state=0)
```

In [7]:

```
from sklearn.linear_model import LogisticRegression
```

In [8]:

```
logisticRegr = LogisticRegression()
```

In [9]:

```
logisticRegr.fit(x_train, y_train)
```

Out[9]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0
001,
          verbose=0, warm_start=False)
```

In [10]:

```
logisticRegr.predict(x_test[0].reshape(1,-1))
```

Out[10]:

```
array([2])
```

In [11]:

```
logisticRegr.predict(x_test[0:10])
```

Out[11]:

```
array([2, 8, 2, 6, 6, 7, 1, 9, 8, 5])
```

In [12]:

```
predictions = logisticRegr.predict(x_test)
```

In [13]:

```
score = logisticRegr.score(x_test, y_test)
print(score)
```

```
0.9533333333333334
```

In [14]:

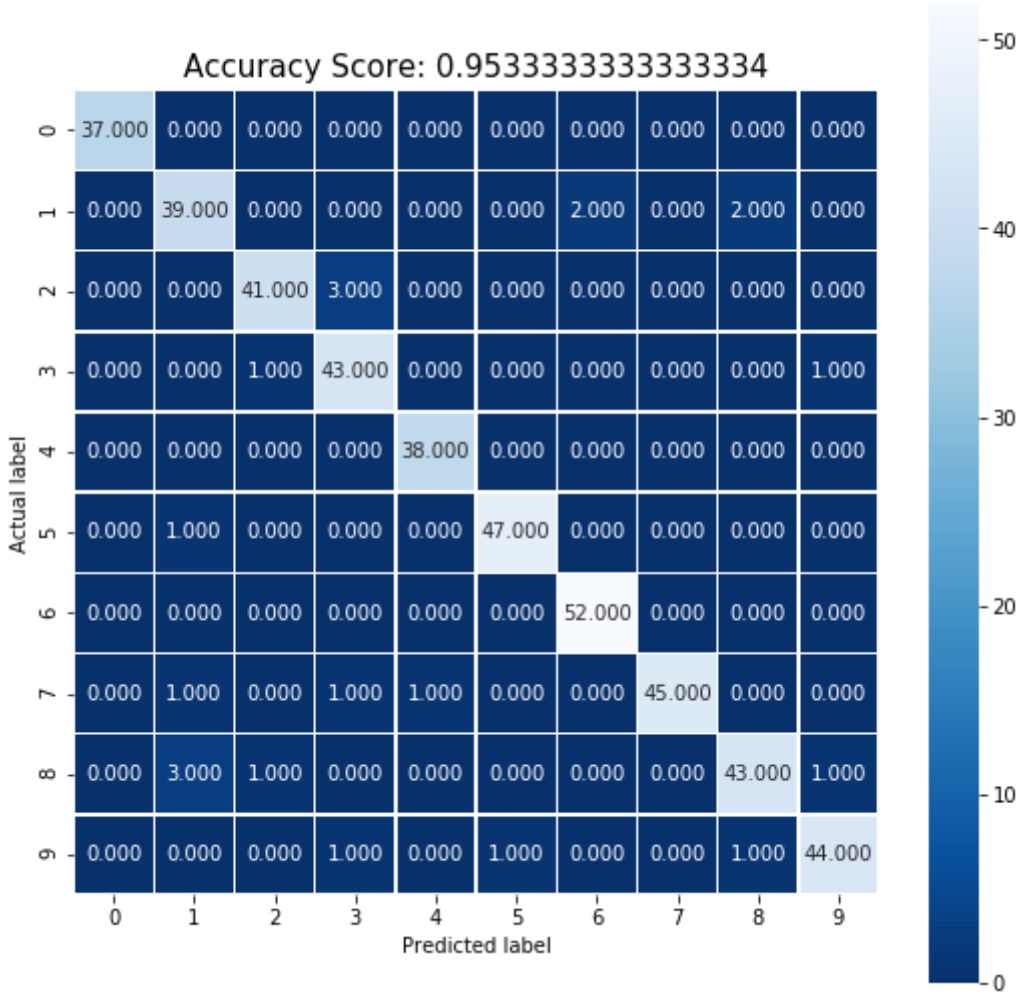
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, predictions)
print(cm)
```

```
[[37  0  0  0  0  0  0  0  0  0]
 [ 0 39  0  0  0  0  2  0  2  0]
 [ 0  0 41  3  0  0  0  0  0  0]
 [ 0  0  1 43  0  0  0  0  0  1]
 [ 0  0  0  0 38  0  0  0  0  0]
 [ 0  1  0  0  0 47  0  0  0  0]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  1  0  1  1  0  0 45  0  0]
 [ 0  3  1  0  0  0  0  0 43  1]
 [ 0  0  0  1  0  1  0  0  1 44]]
```

In [15]:

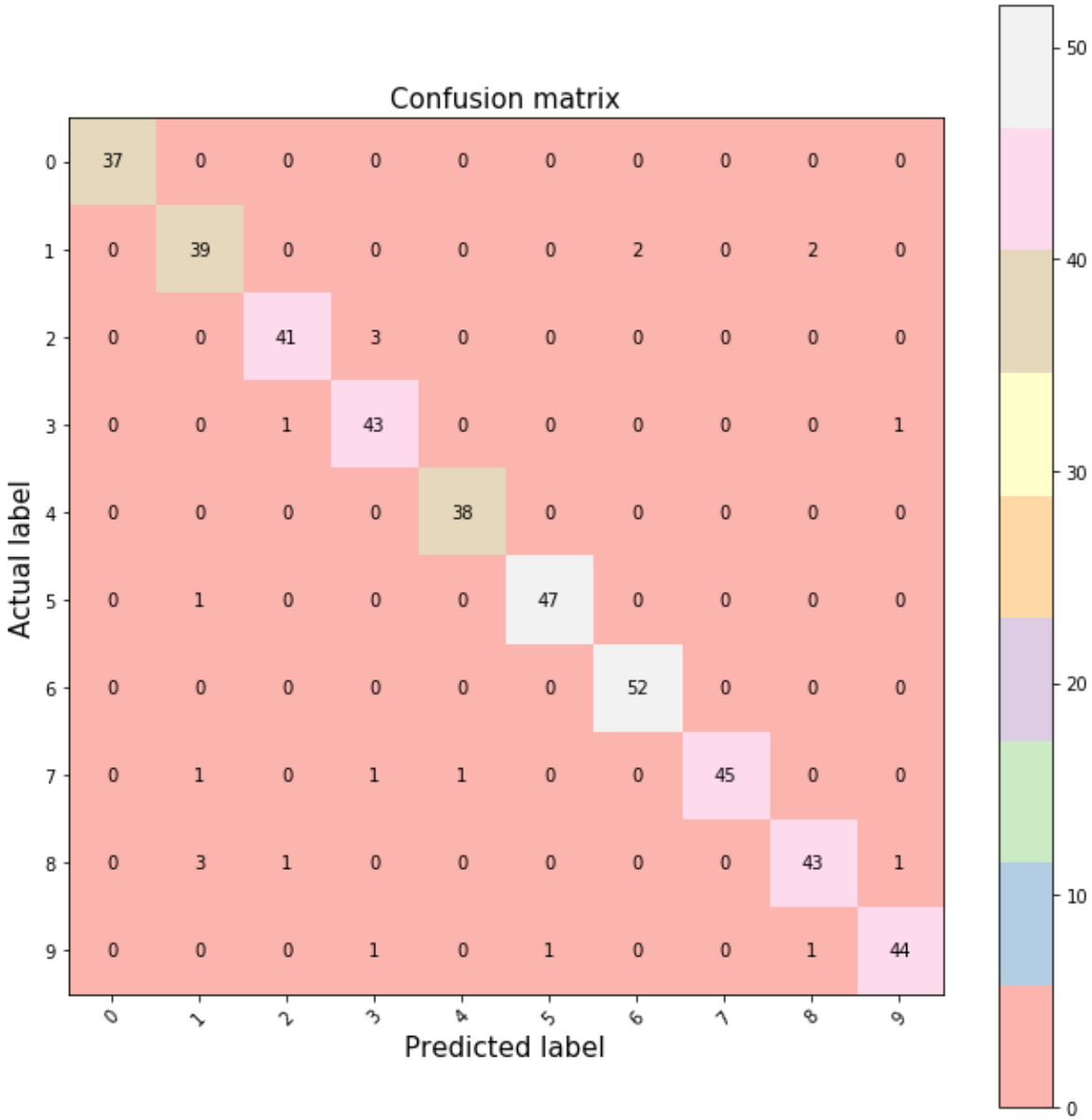
```
import seaborn as sns

plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```



In [17]:

```
plt.figure(figsize=(9,9))
plt.imshow(cm, interpolation='nearest', cmap='Pastel1')
plt.title('Confusion matrix', size = 15)
plt.colorbar()
tick_marks = np.arange(10)
plt.xticks(tick_marks, ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"], rotation
plt.yticks(tick_marks, ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"], size = 1
plt.tight_layout()
plt.ylabel('Actual label', size = 15)
plt.xlabel('Predicted label', size = 15)
width, height = cm.shape
for x in range(width):
    for y in range(height):
        plt.annotate(str(cm[x][y]), xy=(y, x),
                     horizontalalignment='center',
                     verticalalignment='center')
plt.show()
```



In [25]:

```
from tensorflow.examples.tutorials.mnist import input_data as mnist_data
```

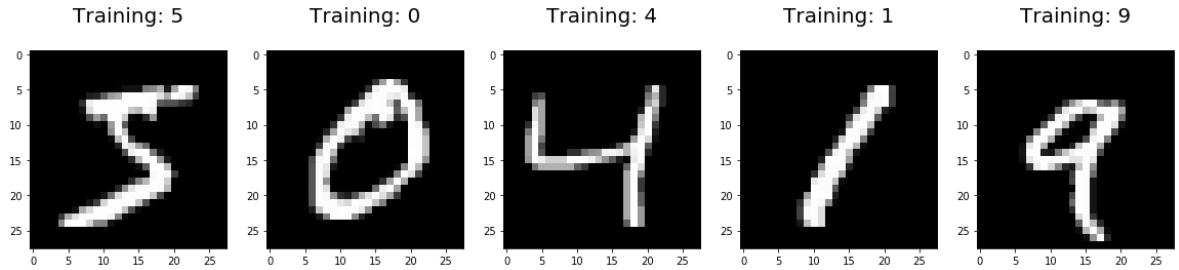
In [42]:

```
mnist = mnist_data.read_data_sets("data",
                                  one_hot=False,
                                  reshape=True,
                                  validation_size=0,
                                  source_url='file:///home/malchiodi/Documents/dida
```

```
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
```

In [43]:

```
plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(mnist.train.images[0:5], mnist.train.lab
    plt.subplot(1, 5, index + 1)
    plt.imshow(np.reshape(image, (28,28)), cmap=plt.cm.gray)
    plt.title('Training: %i\n' % label, fontsize = 20)
plt.show()
```



In [44]:

```
from sklearn.linear_model import LogisticRegression
```

In [45]:

```
logisticRegr = LogisticRegression(solver = 'lbfgs')
```

In [46]:

```
logisticRegr.fit(mnist.train.images, mnist.train.labels)
```

Out[46]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='lbfgs', tol=0.0001,
          verbose=0, warm_start=False)
```

In [48]:

```
logisticRegr.predict(mnist.test.images[0].reshape(1,-1))
```

Out[48]:

```
array([7], dtype=uint8)
```

In [49]:

```
mnist.test.labels[0]
```

Out[49]:

```
7
```

In [50]:

```
logisticRegr.predict(mnist.test.images[0:10])
```

Out[50]:

```
array([7, 2, 1, 0, 4, 1, 4, 9, 6, 9], dtype=uint8)
```

In [51]:

```
predictions = logisticRegr.predict(mnist.test.images)
```

In [52]:

```
score = logisticRegr.score(mnist.test.images, mnist.test.labels)
print(score)
```

```
0.9195
```

In [54]:

```
index = 0
misclassifiedIndexes = []
for label, predict in zip(mnist.test.labels, predictions):
    if label != predict:
        misclassifiedIndexes.append(index)
    index +=1
```

In [58]:

```
len(misclassifiedIndexes)/len(predictions)
```

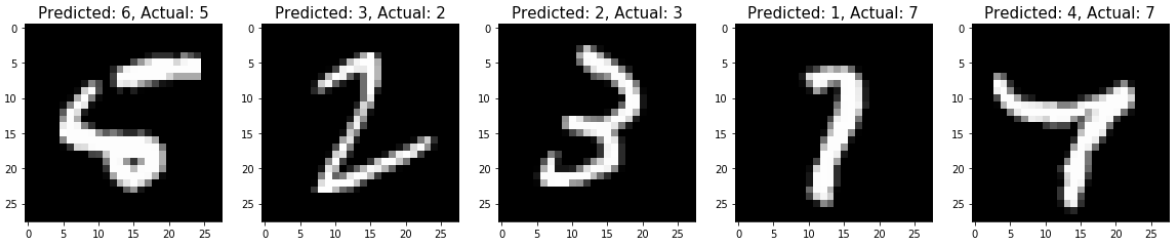
Out[58]:

```
0.0805
```

In [60]:

```
plt.figure(figsize=(20,4))
for plotIndex, badIndex in enumerate(misclassifiedIndexes[0:5]):
    plt.subplot(1, 5, plotIndex + 1)
    plt.imshow(np.reshape(mnist.test.images[badIndex], (28,28)), cmap=plt.cm.gray)
    plt.title('Predicted: {}, Actual: {}'.format(predictions[badIndex], mnist.test.

plt.show()
```



In [ ]: