In [1]:

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

# Used for Confusion Matrix
from sklearn import metrics

%matplotlib inline
```

In [2]:

```
digits = load_digits()
print(digits.target_names)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

In [3]:

```
digits.data.shape
print(digits.keys())
```

```
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
```

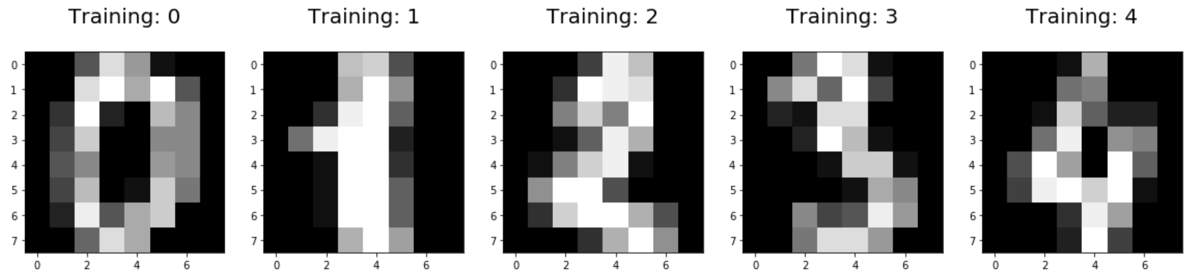
In [4]:

```
print(digits.images)
print(digits.DESCR)
digits.target.shape
```

...

In [5]:

```
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)
```



Training: 0 Training: 1 Training: 2 Training: 3 Training: 4

In [6]:

```
# test_size: what proportion of original data is used for test set
x_train, x_test, y_train, y_test = train_test_split(
    digits.data, digits.target, test_size=0.25, random_state=0)
```

In [7]:

```
print(x_train.shape)
```

```
...
```

In [8]:

```
print(y_train.shape)
```

```
(1347,)
```

In [9]:

```
print(x_test.shape)
```

```
(450, 64)
```

In [10]:

```
print(y_test.shape)
```

```
(450,)
```

In [11]:

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

In [12]:

```
logisticRegr = LogisticRegression()
```

In [13]:

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

```
...
```

In [14]:

```
# Returns a NumPy Array
# Predict for One Observation (image)
logisticRegr.predict(x_test[0].reshape(1,-1))
```

Out[14]:

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

In [15]:

```
# Predict for Multiple Observations (images) at Once
logisticRegr.predict(x_test[0:10])
```

Out[15]:

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

In [16]:

```
# Make predictions on entire test data
predictions = logisticRegr.predict(x_test)
```

In [17]:

```
predictions.shape
```

Out[17]:

```
(450,)
```

In [18]:

```
# Use score method to get accuracy of model
score = logisticRegr.score(x_test, y_test)
print(score)
```

```
0.9533333333333334
```

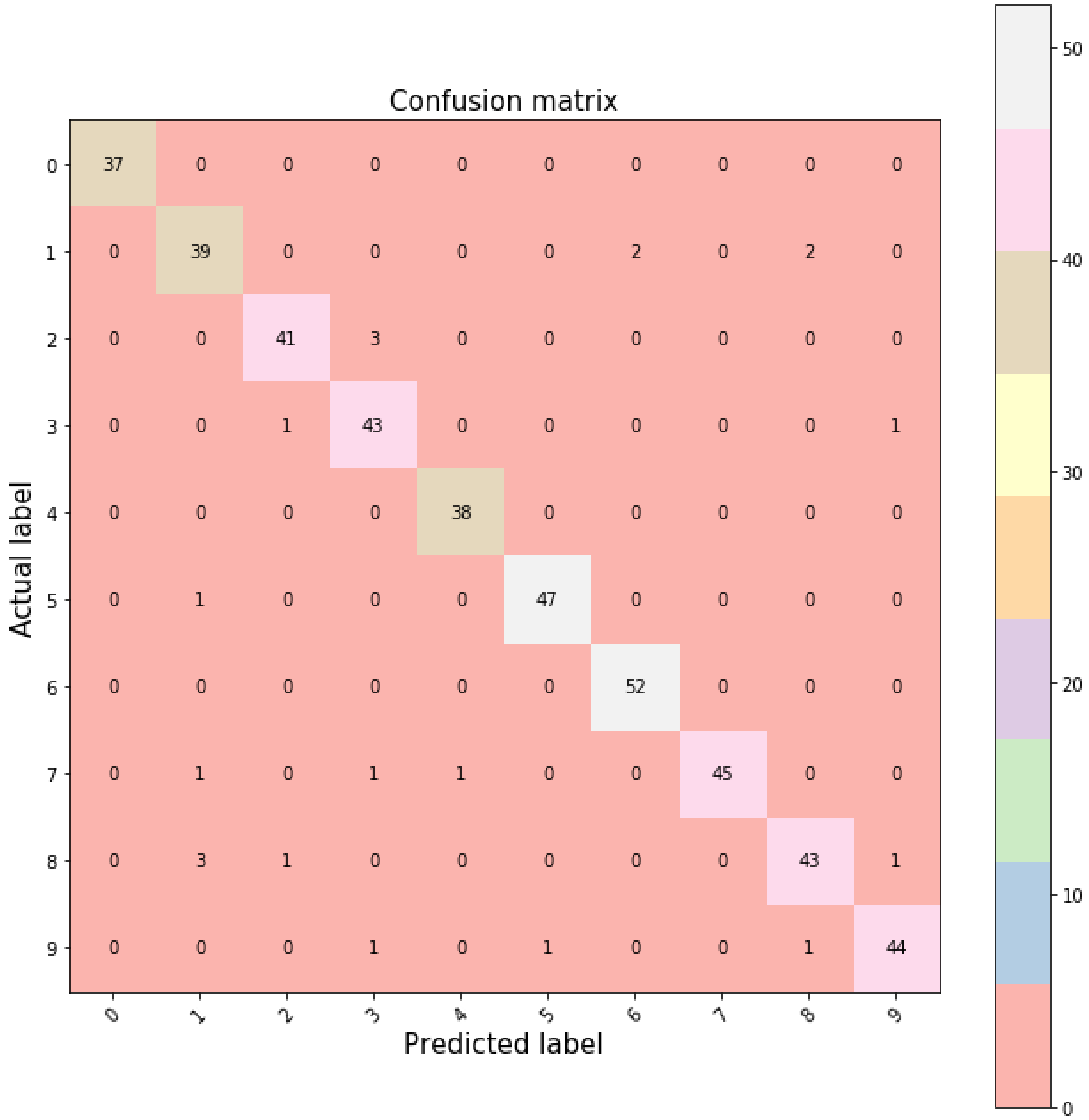
In [19]:

```
def plot_confusion_matrix(cm, title='Confusion matrix', cmap='Pastel1'):
    plt.figure(figsize=(9,9))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, size = 15)
    plt.colorbar()
    tick_marks = np.arange(10)
    plt.xticks(tick_marks, ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"], rotation=45,
    plt.yticks(tick_marks, ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"], size = 10)
    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')
```

In [20]:

```
# confusion matrix
confusion = metrics.confusion_matrix(y_test, predictions)
print('Confusion matrix')
print(confusion)
plt.figure()
plot_confusion_matrix(confusion);
plt.show();
```

```
Confusion matrix
[[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]]

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



In [21]:

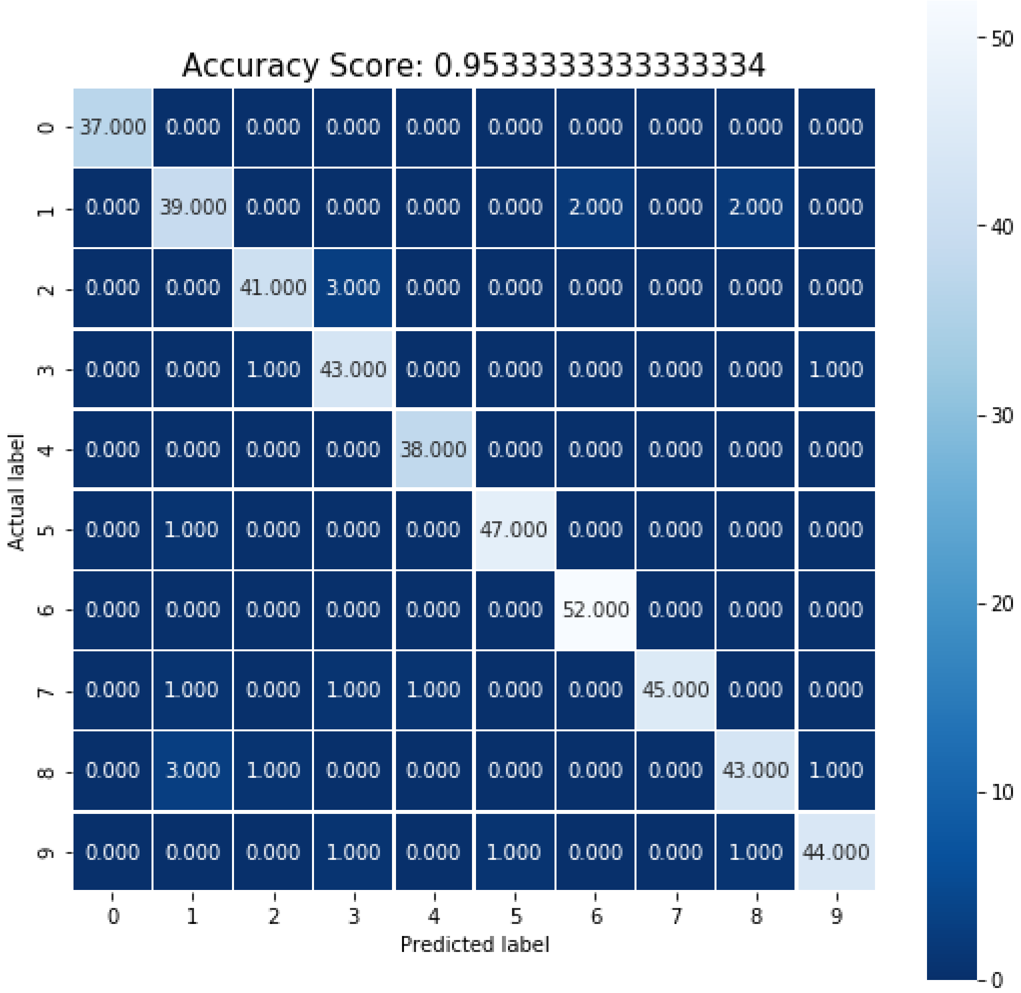
```
# Make predictions on test data
predictions = logisticRegr.predict(x_test)
```

In [22]:

```
cm = metrics.confusion_matrix(y_test, predictions)
#cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

In [23]:

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



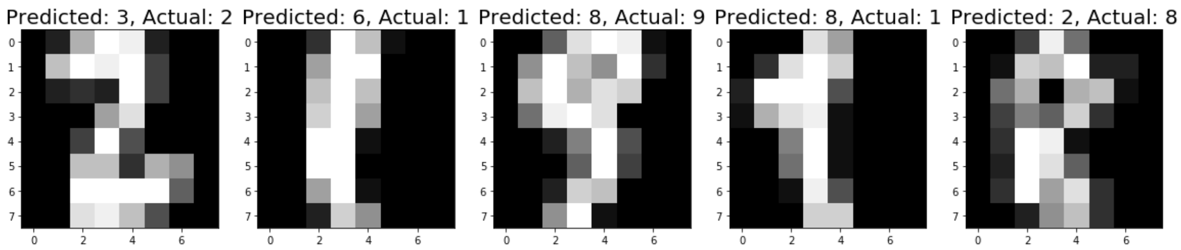
	0	1	2	3	4	5	6	7	8	9
0	37.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	39.000	0.000	0.000	0.000	0.000	2.000	0.000	2.000	0.000
2	0.000	0.000	41.000	3.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	1.000	43.000	0.000	0.000	0.000	0.000	0.000	1.000
4	0.000	0.000	0.000	0.000	38.000	0.000	0.000	0.000	0.000	0.000
5	0.000	1.000	0.000	0.000	0.000	47.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.000	52.000	0.000	0.000	0.000
7	0.000	1.000	0.000	1.000	1.000	0.000	0.000	45.000	0.000	0.000
8	0.000	3.000	1.000	0.000	0.000	0.000	0.000	0.000	43.000	1.000
9	0.000	0.000	0.000	1.000	0.000	1.000	0.000	0.000	1.000	44.000

In [24]:

```
index = 0
misclassifiedIndex = []
for predict, actual in zip(predictions, y_test):
    if predict != actual:
        misclassifiedIndex.append(index)
    index +=1
```

In [25]:

```
plt.figure(figsize=(20,4))
for plotIndex, wrong in enumerate(misclassifiedIndex[10:15]):
    plt.subplot(1, 5, plotIndex + 1)
    plt.imshow(np.reshape(x_test[wrong], (8,8)), cmap=plt.cm.gray)
    plt.title('Predicted: {}, Actual: {}'.format(predictions[wrong], y_test[wrong]), fontsi
```



In [26]:

```
digits
```

```
...
```

In [27]:

```
#K-Neighbors Classifier

from sklearn.neighbors import KNeighborsClassifier
clf=KNeighborsClassifier(n_neighbors=3).fit(x_train,y_train)
predictions1 =clf.predict(x_test)
```

In [ ]:

In [28]:

```
from sklearn.metrics import accuracy_score
print("accuracy found is")
acc1=accuracy_score(y_test,clf.predict(x_test))
print(acc1)
```

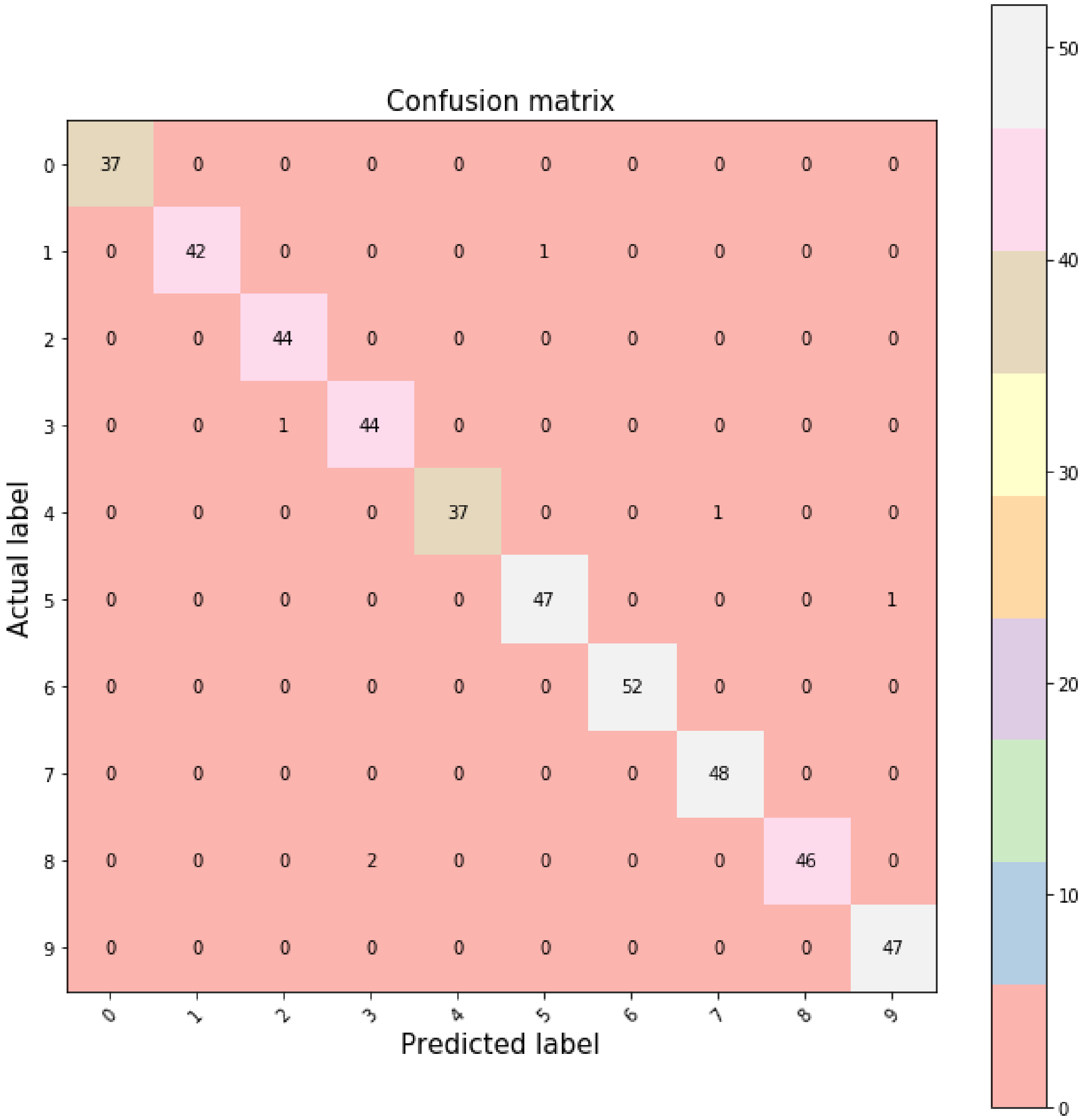
```
accuracy found is
0.9866666666666667
```

In [29]:

```
confusion1 = metrics.confusion_matrix(y_test, predictions1)
print('Confusion matrix')
print(confusion1)
plt.figure()
plot_confusion_matrix(confusion1);
plt.show();
```

```
Confusion matrix
[[37  0  0  0  0  0  0  0  0  0]
 [ 0 42  0  0  0  1  0  0  0  0]
 [ 0  0 44  0  0  0  0  0  0  0]
 [ 0  0  1 44  0  0  0  0  0  0]
 [ 0  0  0  0 37  0  0  1  0  0]
 [ 0  0  0  0  0 47  0  0  0  1]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  0  0  0  0  0  0 48  0  0]
 [ 0  0  0  2  0  0  0  0 46  0]
 [ 0  0  0  0  0  0  0  0  0 47]]
```

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



In [ ]:

```

```

In [30]:

```
digits.images=digits.images.reshape(digits.images.shape[0],digits.images.shape[1]*digits.im

#now lets print new matrix dimensions
print(digits.images.shape)



#lets try to print features and labels we have in the dataset
print(digits.images)
print(digits.target)

#lets also see the dimension of data we have

print(digits.images.shape)
print(digits.target.shape)
```

...

In [31]:

```
#Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(x_train,y_train)
```

Out[31]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [32]:

```
pred=rf.predict(x_test)
```

In [33]:

```
pred
```

```
...
```

In [34]:

```
acc2=accuracy_score(y_test,rf.predict(x_test))
print(acc2)
```

```
0.9733333333333334
```

In [35]:

```
#Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier
```

In [36]:

```
clf = DecisionTreeClassifier()
```

In [37]:

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

```
...
```

In [38]:

```
pred=rf.predict(x_test)
```

In [39]:

```
pred
```

```
...
```

In [40]:

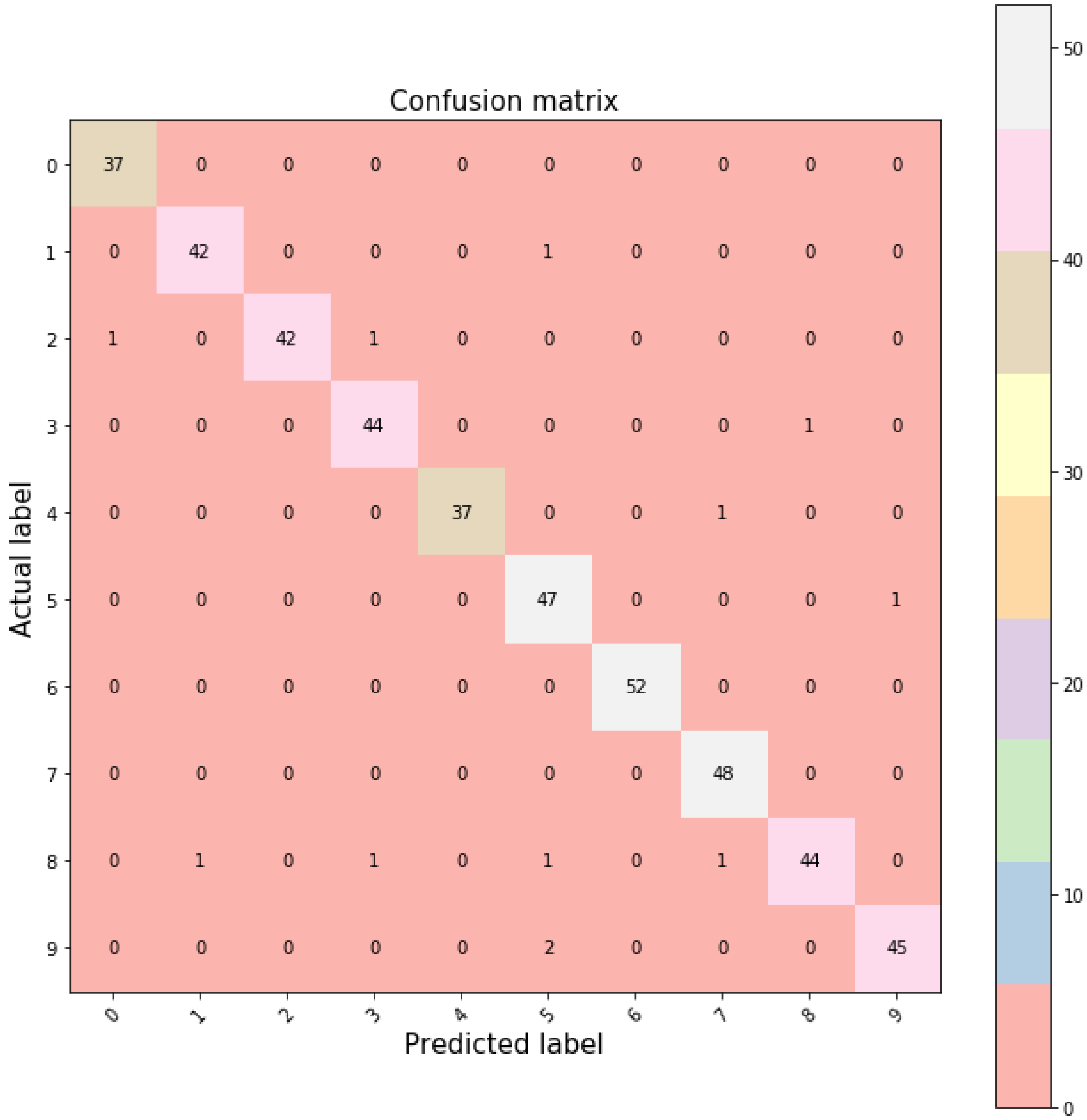
```
acc3=accuracy_score(y_test,clf.predict(x_test))
print(acc3)
```

```
0.8444444444444444
```

In [41]:

```
confusion2 = metrics.confusion_matrix(y_test, pred)
print('Confusion matrix')
print(confusion2)
plt.figure()
plot_confusion_matrix(confusion2);
plt.show();
```

```
Confusion matrix
[[37  0  0  0  0  0  0  0  0  0]
 [ 0 42  0  0  0  1  0  0  0  0]
 [ 1  0 42  1  0  0  0  0  0  0]
 [ 0  0  0 44  0  0  0  0  1  0]
 [ 0  0  0  0 37  0  0  1  0  0]
 [ 0  0  0  0  0 47  0  0  0  1]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  0  0  0  0  0  0 48  0  0]
 [ 0  1  0  1  0  1  0  1 44  0]
 [ 0  0  0  0  0  2  0  0  0 45]]

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



In [42]:

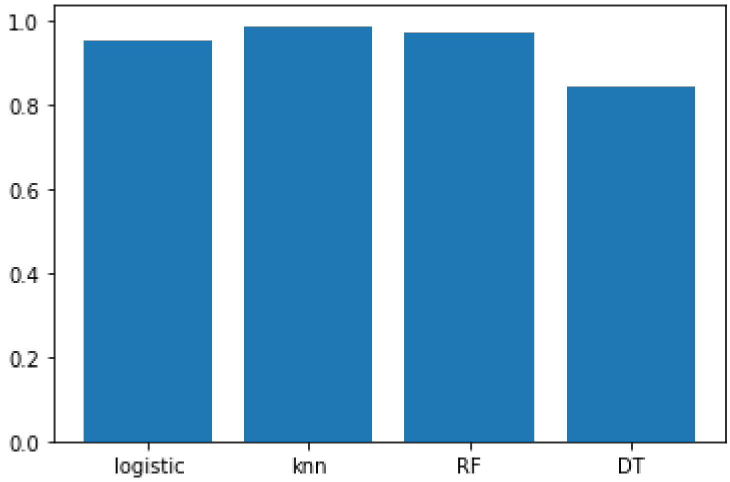
```
lable=['logistic','knn','RF','DT']
accuracy=[
    score,acc1,acc2,acc3
]
```

In [43]:

```
index=np.arange(len(lable))
plt.bar(index,accuracy)
plt.xticks(index,lable)
```

Out[43]:

```
([<matplotlib.axis.XTick at 0x2130fae40f0>,
  <matplotlib.axis.XTick at 0x2130fab59e8>,
  <matplotlib.axis.XTick at 0x2130fab5978>,
  <matplotlib.axis.XTick at 0x2130fafeb70>],
 <a list of 4 Text xticklabel objects>)
```



In [ ]:

In [ ]: