**Recognizing Handwritten Digits with scikit-learn**

**Load Dataset**

In [1]:

**from** sklearn **import** datasets

digits **=** datasets**.**load\_digits()

**Full description of the dataset**

In [2]:

print(digits**.**DESCR)

.. \_digits\_dataset:

Optical recognition of handwritten digits dataset

--------------------------------------------------

\*\*Data Set Characteristics:\*\*

:Number of Instances: 5620

:Number of Attributes: 64

:Attribute Information: 8x8 image of integer pixels in the range 0..16.

:Missing Attribute Values: None

:Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)

:Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets

https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

The data set contains images of hand-written digits: 10 classes where

each class refers to a digit.

Preprocessing programs made available by NIST were used to extract

normalized bitmaps of handwritten digits from a preprinted form. From a

total of 43 people, 30 contributed to the training set and different 13

to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of

4x4 and the number of on pixels are counted in each block. This generates

an input matrix of 8x8 where each element is an integer in the range

0..16. This reduces dimensionality and gives invariance to small

distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.

T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.

L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,

1994.

.. topic:: References

- C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their

Applications to Handwritten Digit Recognition, MSc Thesis, Institute of

Graduate Studies in Science and Engineering, Bogazici University.

- E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.

- Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.

Linear dimensionalityreduction using relevance weighted LDA. School of

Electrical and Electronic Engineering Nanyang Technological University.

2005.

- Claudio Gentile. A New Approximate Maximal Margin Classification

Algorithm. NIPS. 2000.

The numerical values represented by images, i.e., the targets, are contained in the digit.targets array.

In [3]:

digits**.**target

Out[3]:

array([0, 1, 2, ..., 8, 9, 8])

**Shape of the dataset**

In [4]:

digits**.**data**.**shape

Out[4]:

(1797, 64)

**Images stored in the form of array**

The images of the handwritten digits are contained in a digits.images array. Each element of this array is an image that is represented by an 8x8 matrix of numerical values that correspond to a grayscale from white, with a value of 0, to black, with the value 15

In [5]:

digits**.**images[0]

Out[5]:

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.]])

The images of the handwritten digits are contained in a digits.images array

**Visualizing an array**

In [6]:

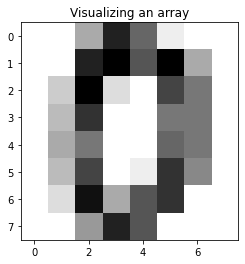
**import** matplotlib.pyplot **as** plt

plt**.**imshow(digits**.**images[0], cmap**=**plt**.**cm**.**gray\_r, interpolation**=**'nearest')

plt**.**title('Visualizing an array')

*# save the figure*

plt**.**savefig('plot2.png', dpi**=**100, bbox\_inches**=**'tight')



**Visualization of digits**

In [61]:

**import** numpy **as** np

plt**.**figure(figsize**=**(15,4))

plt**.**subplots\_adjust(hspace**=**0.8)

**for** index, (image, label) **in** enumerate(zip(digits**.**data[0:10], digits**.**target[0:10])):

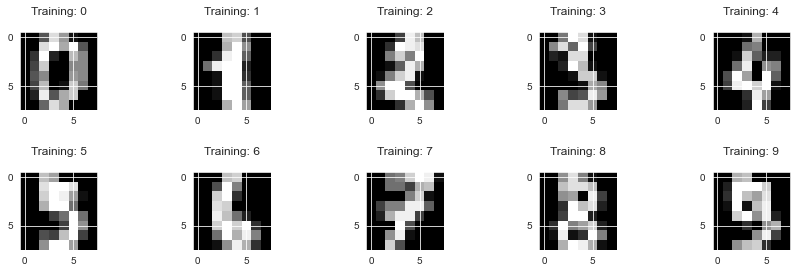
plt**.**subplot(2, 5, index **+** 1)

plt**.**imshow(np**.**reshape(image, (8,8)), cmap**=**plt**.**cm**.**gray)

plt**.**title('Training: %i\n' **%** label, fontsize **=**12)

*# save the figure*

plt**.**savefig('plot1.png', dpi**=**300, bbox\_inches**=**'tight')



**Split the dataset**

**Size of the training set**

It was reported that the dataset is a training set consisting of 1,797 images. we can determine if that is true.

In [62]:

digits**.**target**.**size

Out[62]:

1797

In [63]:

*# flatten the images*

n\_samples **=** len(digits**.**images)

data **=** digits**.**images**.**reshape((n\_samples, **-**1))

In [64]:

**from** sklearn.model\_selection **import** train\_test\_split

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(data, digits**.**target, test\_size**=**0.2, random\_state**=**0)

**Training and Prediction**

**1.Support Vector Classifier**

In [65]:

**from** sklearn **import** svm

svc **=** svm**.**SVC(gamma**=**0.001, C**=**100.)

In [66]:

svc**.**fit(x\_train, y\_train)

Out[66]:

SVC(C=100.0, gamma=0.001)

In [67]:

y\_pred **=** svc**.**predict(x\_test)

**4 test samples and their predicted digit value**

In [68]:

\_, axes **=** plt**.**subplots(nrows**=**1, ncols**=**4, figsize**=**(10, 3))

**for** ax, image, prediction **in** zip(axes, x\_test, y\_pred):

ax**.**set\_axis\_off()

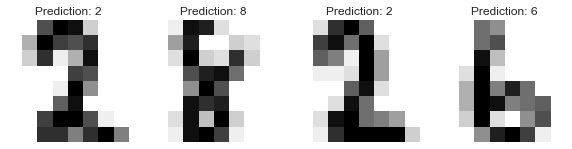
image **=** image**.**reshape(8, 8)

ax**.**imshow(image, cmap**=**plt**.**cm**.**gray\_r, interpolation**=**'nearest')

ax**.**set\_title(f'Prediction: {prediction}')

*# save the figure*

plt**.**savefig('plot7.png', dpi**=**300, bbox\_inches**=**'tight')



In [69]:

score **=** svc**.**score(x\_test, y\_test)

print('Accuracy Score: {0}'**.**format(score))

Accuracy Score: 0.9916666666666667

In [70]:

**from** sklearn.metrics **import** confusion\_matrix,classification\_report,accuracy\_score

**import** seaborn **as** sns

**import** pandas **as** pd

labels**=**['0','1','2', '3','4','5','6','7','8','9']

f, ax **=** plt**.**subplots(figsize**=**(10,10))

cm**=**confusion\_matrix(y\_test,y\_pred)

sns**.**heatmap(cm, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

*#labels, title and ticks*

ax**.**set\_xlabel('Predicted labels')

ax**.**set\_ylabel('True labels')

ax**.**set\_title('Accuracy Score: {0} \n Confusion Matrix'**.**format(np**.**round(score,2)))

ax**.**xaxis**.**set\_ticklabels(labels)

ax**.**yaxis**.**set\_ticklabels(labels)

plt**.**savefig('plot3.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()

f, ax **=** plt**.**subplots(figsize**=**(6,6))

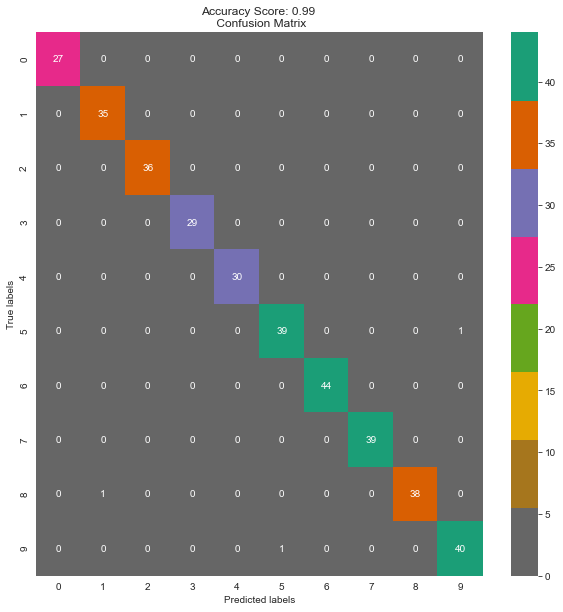
class\_report**=**classification\_report(y\_test,y\_pred,target\_names**=**labels, output\_dict**=True**)

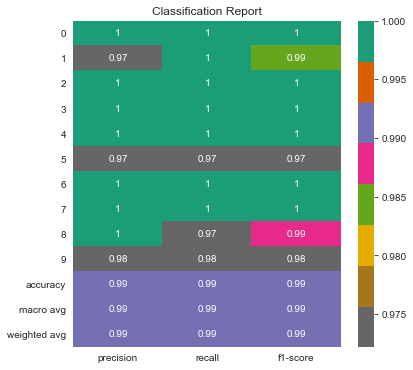
sns**.**heatmap(pd**.**DataFrame(class\_report)**.**iloc[:**-**1, :]**.**T, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

ax**.**set\_title('Classification Report')

plt**.**savefig('plot4.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()





**2.Logistic Regression**

In [71]:

**from** sklearn.linear\_model **import** LogisticRegression

logisticRegr **=** LogisticRegression()

In [72]:

**import** warnings

warnings**.**filterwarnings("ignore")

logisticRegr**.**fit(x\_train, y\_train)

Out[72]:

LogisticRegression()

In [73]:

y\_pred**=**logisticRegr**.**predict(x\_test)

In [74]:

*# Use score method to get accuracy of model*

score **=** logisticRegr**.**score(x\_test, y\_test)

print('Accuracy Score: {0}'**.**format(score))

Accuracy Score: 0.9666666666666667

In [75]:

**from** sklearn.metrics **import** confusion\_matrix,classification\_report,accuracy\_score

**import** seaborn **as** sns

**import** pandas **as** pd

labels**=**['0','1','2', '3','4','5','6','7','8','9']

f, ax **=** plt**.**subplots(figsize**=**(10,10))

cm**=**confusion\_matrix(y\_test,y\_pred)

sns**.**heatmap(cm, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

*#labels, title and ticks*

ax**.**set\_xlabel('Predicted labels')

ax**.**set\_ylabel('True labels')

ax**.**set\_title('Accuracy Score: {0} \n Confusion Matrix'**.**format(np**.**round(score,2)))

ax**.**xaxis**.**set\_ticklabels(labels)

ax**.**yaxis**.**set\_ticklabels(labels)

plt**.**savefig('plot5.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()

f, ax **=** plt**.**subplots(figsize**=**(6,6))

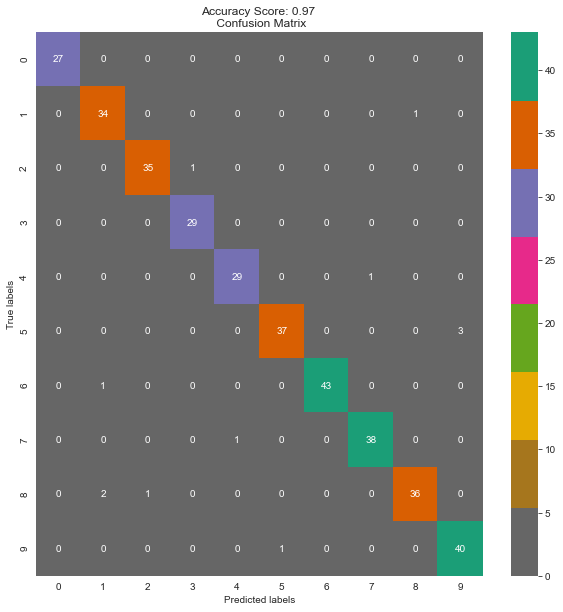
class\_report**=**classification\_report(y\_test,y\_pred,target\_names**=**labels, output\_dict**=True**)

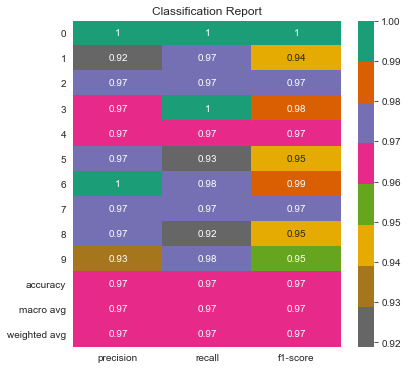
sns**.**heatmap(pd**.**DataFrame(class\_report)**.**iloc[:**-**1, :]**.**T, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

ax**.**set\_title('Classification Report')

plt**.**savefig('plot6.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()





**3.Decision Tree Classifier**

In [76]:

**from** sklearn.tree **import** DecisionTreeClassifier

dt **=** DecisionTreeClassifier(criterion **=** 'gini')

In [77]:

dt**.**fit(x\_train, y\_train)

Out[77]:

DecisionTreeClassifier()

In [78]:

y\_pred**=**dt**.**predict(x\_test)

In [79]:

*# Use score method to get accuracy of model*

score **=** dt**.**score(x\_test, y\_test)

print('Accuracy Score: {0}'**.**format(score))

Accuracy Score: 0.8444444444444444

In [80]:

**from** sklearn.metrics **import** confusion\_matrix,classification\_report,accuracy\_score

**import** seaborn **as** sns

**import** pandas **as** pd

labels**=**['0','1','2', '3','4','5','6','7','8','9']

f, ax **=** plt**.**subplots(figsize**=**(10,10))

cm**=**confusion\_matrix(y\_test,y\_pred)

sns**.**heatmap(cm, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

*#labels, title and ticks*

ax**.**set\_xlabel('Predicted labels')

ax**.**set\_ylabel('True labels')

ax**.**set\_title('Accuracy Score: {0} \n Confusion Matrix'**.**format(np**.**round(score,2)))

ax**.**xaxis**.**set\_ticklabels(labels)

ax**.**yaxis**.**set\_ticklabels(labels)

plt**.**savefig('plot7.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()

f, ax **=** plt**.**subplots(figsize**=**(6,6))

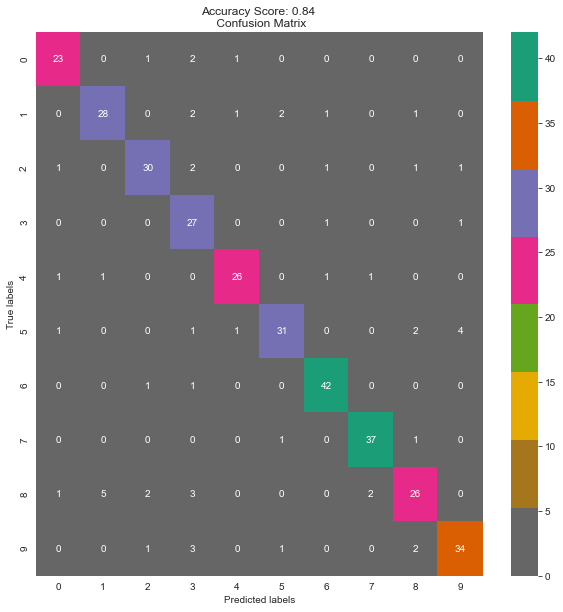
class\_report**=**classification\_report(y\_test,y\_pred,target\_names**=**labels, output\_dict**=True**)

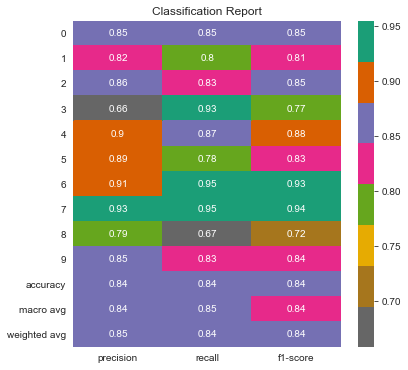
sns**.**heatmap(pd**.**DataFrame(class\_report)**.**iloc[:**-**1, :]**.**T, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

ax**.**set\_title('Classification Report')

plt**.**savefig('plot8.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()





**4. Random Forest Classifier**

In [81]:

**from** sklearn.ensemble **import** RandomForestClassifier

rc **=** RandomForestClassifier(n\_estimators **=** 150)

In [82]:

rc**.**fit(x\_train, y\_train)

Out[82]:

RandomForestClassifier(n\_estimators=150)

In [83]:

y\_pred**=**rc**.**predict(x\_test)

In [84]:

*# Use score method to get accuracy of model*

score **=** rc**.**score(x\_test, y\_test)

print('Accuracy Score: {0}'**.**format(score))

Accuracy Score: 0.975

In [85]:

**from** sklearn.metrics **import** confusion\_matrix,classification\_report,accuracy\_score

**import** seaborn **as** sns

**import** pandas **as** pd

labels**=**['0','1','2', '3','4','5','6','7','8','9']

f, ax **=** plt**.**subplots(figsize**=**(10,10))

cm**=**confusion\_matrix(y\_test,y\_pred)

sns**.**heatmap(cm, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

*#labels, title and ticks*

ax**.**set\_xlabel('Predicted labels')

ax**.**set\_ylabel('True labels')

ax**.**set\_title('Accuracy Score: {0} \n Confusion Matrix'**.**format(np**.**round(score,2)))

ax**.**xaxis**.**set\_ticklabels(labels)

ax**.**yaxis**.**set\_ticklabels(labels)

plt**.**savefig('plot9.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()

f, ax **=** plt**.**subplots(figsize**=**(6,6))

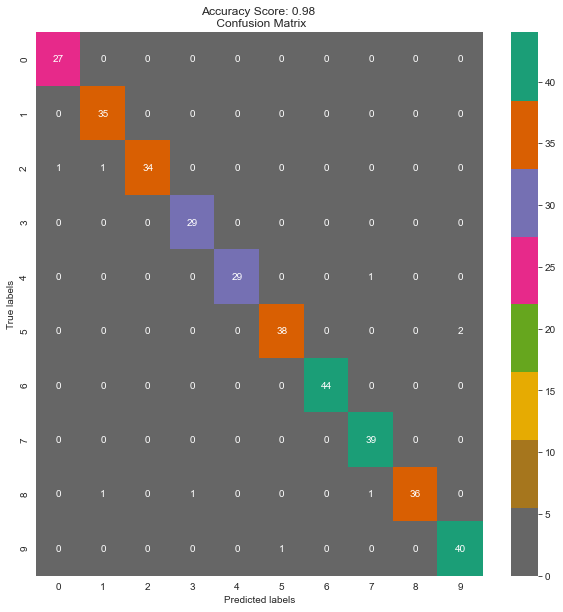
class\_report**=**classification\_report(y\_test,y\_pred,target\_names**=**labels, output\_dict**=True**)

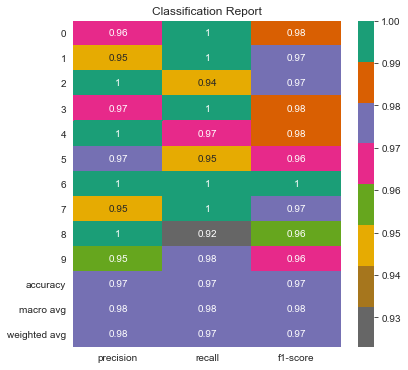
sns**.**heatmap(pd**.**DataFrame(class\_report)**.**iloc[:**-**1, :]**.**T, annot**=True**,ax**=**ax,cmap**=**"Dark2\_r")

ax**.**set\_title('Classification Report')

plt**.**savefig('plot10.png', dpi**=**300, bbox\_inches**=**'tight')

plt**.**show()





**Conclusion:**

This dataset predicts the digit accurately 95% of the times.