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

```
from sklearn.datasets import load_digits
```

```
df=load_digits()
```

```
df.images.shape
```

```
(1797, 8, 8)
```

```
df.images[0]
```

```
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.]])
```

```
df.images[0].shape
```

```
(8, 8)
```

```
type(df.images)
```

```
numpy.ndarray
```

#Flatten

```
data=df.images.reshape(len(df.images),-1)
```

```
data[0]
```

```
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.])
```

```
data[0].shape
```

```
(64,)
```

```
data.shape
```

```
(1797, 64)
```

#Scaling Data

```
data.min()
```

```
0.0
```

```
data.max()
```

```
16.0
```

```
data=data/16
```

```
data[0]
```

```
array([0.    , 0.    , 0.3125, 0.8125, 0.5625, 0.0625, 0.    , 0.    ,
       0.    , 0.    , 0.8125, 0.9375, 0.625 , 0.9375, 0.3125, 0.    ,
       0.    , 0.1875, 0.9375, 0.125 , 0.    , 0.6875, 0.5   , 0.    ,
       0.    , 0.25  , 0.75  , 0.    , 0.    , 0.5   , 0.5   , 0.    ,
       0.    , 0.3125, 0.5   , 0.    , 0.    , 0.5625, 0.5   , 0.    ,
       0.    , 0.25  , 0.6875, 0.    , 0.0625, 0.75  , 0.4375, 0.    ,
       0.    , 0.125 , 0.875 , 0.3125, 0.625 , 0.75  , 0.    , 0.    ,
       0.    , 0.    , 0.375 , 0.8125, 0.625 , 0.    , 0.    ,
0.    ])
```

```
data.min()
```

```
0.0
```

```
data.max()
```

```
1.0
```

#Train Test Split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test
=train_test_split(data,df.target,test_size=0.3,random_state=234)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((1257, 64), (540, 64), (1257,), (540,))
```

#Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
```

```
rf.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
y_pred = rf.predict(X_test)
```

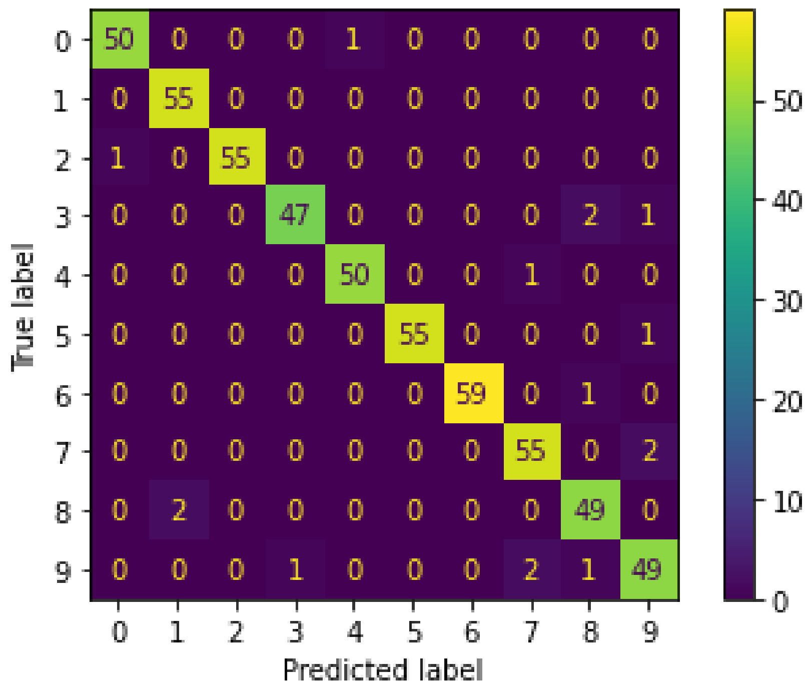
```
y_pred
```

```
array([4, 2, 1, 5, 2, 9, 7, 2, 8, 2, 6, 3, 3, 4, 4, 8, 7, 6, 8, 3, 9,
2,
       4, 1, 5, 1, 5, 3, 8, 0, 1, 0, 5, 0, 7, 8, 6, 4, 0, 2, 1, 7, 3,
2,
       0, 8, 5, 7, 5, 0, 4, 6, 9, 7, 4, 1, 5, 4, 8, 9, 9, 2, 7, 1, 5,
2,
       7, 6, 3, 9, 5, 2, 1, 4, 2, 3, 0, 9, 0, 1, 9, 8, 8, 5, 8, 9, 8,
9,
       6, 1, 3, 6, 6, 3, 8, 1, 5, 6, 5, 6, 7, 4, 2, 3, 0, 9, 9, 6, 7,
3,
       1, 8, 3, 2, 8, 6, 0, 0, 2, 3, 0, 6, 0, 6, 4, 7, 1, 7, 7, 3, 5,
5,
       3, 3, 7, 4, 7, 0, 5, 7, 2, 2, 8, 2, 6, 1, 4, 4, 1, 4, 8, 0, 3,
3,
       7, 4, 4, 5, 1, 4, 3, 6, 3, 5, 0, 3, 6, 1, 7, 0, 0, 1, 7, 9, 1,
2,
       9, 3, 8, 5, 6, 1, 7, 9, 4, 1, 5, 6, 7, 1, 2, 2, 3, 8, 3, 1, 2,
2,
       4, 4, 7, 4, 8, 5, 4, 4, 6, 8, 9, 4, 1, 7, 8, 6, 1, 4, 8, 4, 9,
5,
       2, 0, 6, 1, 4, 3, 0, 3, 5, 4, 7, 3, 7, 1, 9, 6, 0, 8, 6, 8, 7,
7,
       9, 0, 0, 1, 9, 1, 1, 0, 9, 5, 5, 8, 5, 4, 0, 3, 6, 4, 7, 7, 6,
8,
       0, 6, 6, 6, 6, 9, 0, 8, 3, 8, 0, 1, 0, 4, 2, 6, 2, 4, 0, 6, 2,
1,
       5, 1, 2, 3, 5, 1, 2, 1, 6, 3, 5, 8, 9, 2, 1, 0, 7, 9, 0, 3, 4,
9,
       9, 5, 6, 6, 0, 5, 8, 7, 0, 2, 6, 8, 8, 4, 8, 8, 2, 7, 5, 1, 7,
6,
       5, 5, 6, 0, 7, 2, 8, 9, 9, 6, 5, 1, 9, 5, 7, 9, 2, 2, 0, 0, 1,
5,
       5, 2, 6, 8, 7, 6, 7, 6, 9, 5, 3, 0, 3, 7, 8, 2, 1, 7, 9, 2, 8,
0,
       7, 1, 2, 9, 6, 2, 1, 7, 4, 9, 9, 9, 5, 6, 2, 3, 3, 0, 9, 4, 0,
2,
       4, 3, 4, 4, 1, 0, 1, 8, 0, 7, 3, 7, 2, 3, 5, 9, 1, 6, 8, 0, 9,
9,
       4, 7, 9, 1, 5, 8, 3, 7, 4, 2, 8, 5, 9, 8, 9, 2, 5, 7, 1, 6, 4,
2,
       6, 3, 6, 3, 8, 6, 4, 8, 6, 7, 2, 4, 5, 7, 7, 1, 2, 5, 5, 2, 6,
7,
       4, 0, 7, 2, 5, 9, 1, 6, 5, 1, 9, 5, 8, 9, 6, 9, 7, 7, 7, 4, 9,
2,
       3, 0, 6, 9, 8, 7, 5, 8, 6, 8, 5, 8, 0, 8, 2, 3, 3, 9, 3, 5, 5,
1,
       4, 8, 1, 6, 6, 0, 5, 6, 1, 7, 1, 9, 1, 7, 4, 9, 0, 7, 0, 0, 1,
2,
       3, 6, 2, 2, 4, 9, 1, 3, 5, 8, 2, 4])
```

Get Model Evaluation

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, plot_confusion_matrix

accuracy_score(y_test,y_pred)

0.9703703703703703

confusion_matrix(y_test,y_pred)

array([[50,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 55,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 1,  0, 55,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 47,  0,  0,  0,  0,  2,  1],
       [ 0,  0,  0,  0, 50,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 55,  0,  0,  0,  1],
       [ 0,  0,  0,  0,  0,  0, 59,  0,  1,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 55,  0,  2],
       [ 0,  2,  0,  0,  0,  0,  0,  0, 49,  0],
       [ 0,  0,  0,  1,  0,  0,  0,  2,  1, 49]])

print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.98      0.98      0.98        51
           1       0.96      1.00      0.98        55
           2       1.00      0.98      0.99        56
           3       0.98      0.94      0.96        50
           4       0.98      0.98      0.98        51
           5       1.00      0.98      0.99        56
           6       1.00      0.98      0.99        60
           7       0.95      0.96      0.96        57
           8       0.92      0.96      0.94        51
           9       0.92      0.92      0.92        53

    accuracy                           0.97       540
   macro avg       0.97      0.97      0.97       540
weighted avg       0.97      0.97      0.97       540


plot_confusion_matrix(rf,X_test,y_test)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: Function plot_confusion_matrix is
deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and
will be removed in 1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7ff1da0fc1d0>
```



	0	1	2	3	4	5	6	7	8	9
0	50	0	0	0	1	0	0	0	0	0
1	0	55	0	0	0	0	0	0	0	0
2	1	0	55	0	0	0	0	0	0	0
3	0	0	0	47	0	0	0	0	2	1
4	0	0	0	0	50	0	0	1	0	0
5	0	0	0	0	0	55	0	0	0	1
6	0	0	0	0	0	0	59	0	1	0
7	0	0	0	0	0	0	0	55	0	2
8	0	2	0	0	0	0	0	0	49	0
9	0	0	0	1	0	0	0	2	1	49

```
rf.predict_proba(X_test)
```

```
array([[0.03, 0.  , 0.  , ..., 0.02, 0.01, 0.01],
       [0.  , 0.01, 0.91, ..., 0.  , 0.01, 0.02],
       [0.  , 0.93, 0.  , ..., 0.02, 0.03, 0.01],
       ...,
       [0.03, 0.08, 0.03, ..., 0.36, 0.39, 0.  ],
       [0.01, 0.04, 0.73, ..., 0.03, 0.08, 0.01],
       [0.  , 0.04, 0.04, ..., 0.02, 0.1 , 0.  ]])
```