Name : Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

```
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()
```

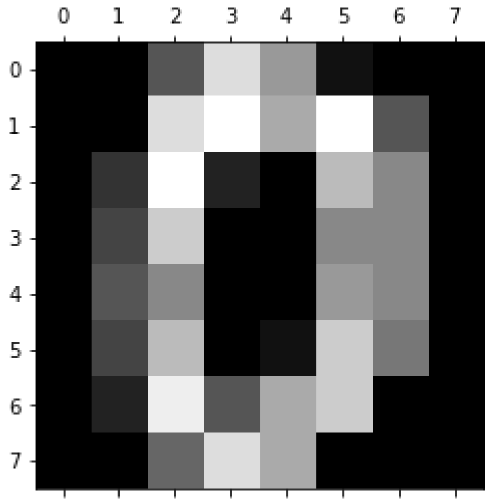
```
dir(digits)
```

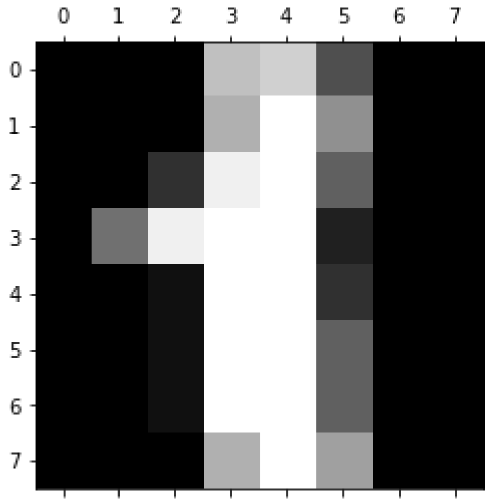
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

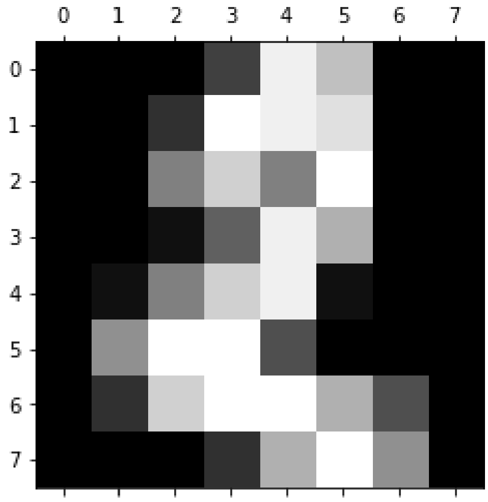
```
%matplotlib inline
import matplotlib.pyplot as plt
```

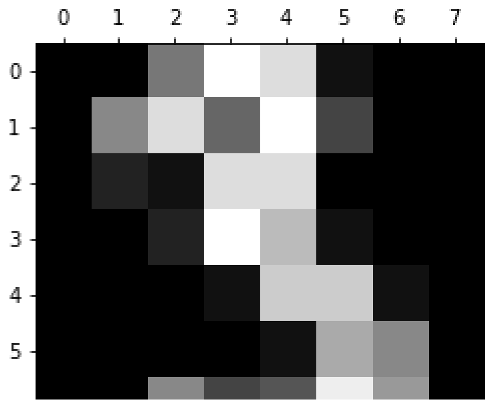
```
plt.gray()
for i in range(10):
  plt.matshow(digits.images[i])
```

<Figure size 432x288 with 0 Axes>









```
df = pd.DataFrame(digits.data)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0	1
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0	1
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0	1
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	7.0	13.0	1
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2.0	1

5 rows � 64 columns

```
df['target'] = digits.target
```

```
df[0:12]
```

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	13.0	1
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	1
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	1
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	1
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	1
5	0.0	0.0	12.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	9.0	16.0	1
6	0.0	0.0	0.0	12.0	13.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	9.0	1
7	0.0	0.0	7.0	8.0	13.0	16.0	15.0	1.0	0.0	0.0	...	0.0	0.0	0.0	13.0	5.0	
8	0.0	0.0	9.0	14.0	8.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	11.0	16.0	1
9	0.0	0.0	11.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	9.0	12.0	1
10	0.0	0.0	1.0	9.0	15.0	11.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	10.0	1
11	0.0	0.0	0.0	0.0	14.0	13.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	1

12 rows � 65 columns

```
## Train the model and prediction
```

```
X = df.drop('target',axis = 'columns')
y = df.target
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1)
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=30)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=30)
```

```
model.score(X_test, y_test)
```

```
0.9666666666666667
```

```
y_predicted = model.predict(X_test)
```

```
from sklearn.datasets import make_classification
```

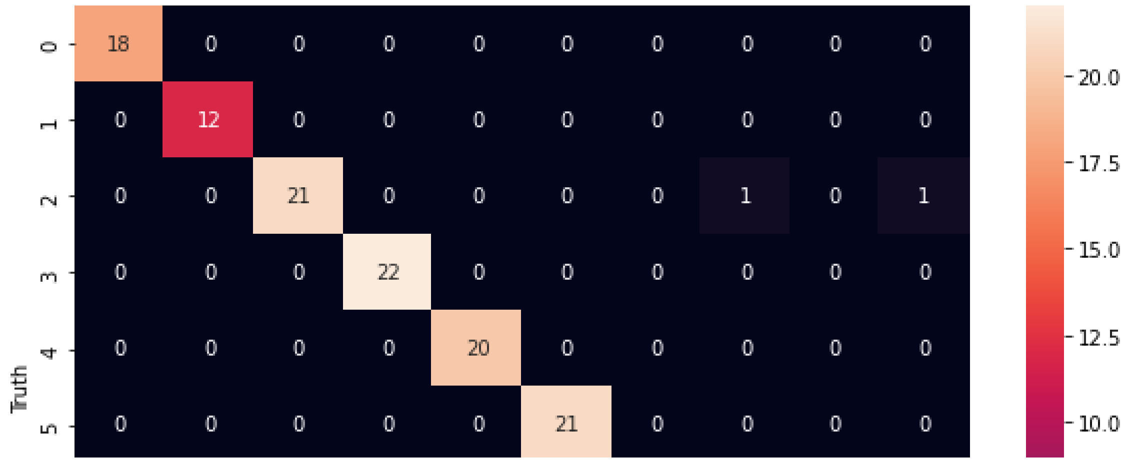
Confusion Matrix

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

```
array([[18,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 12,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 21,  0,  0,  0,  0,  1,  0,  1],
       [ 0,  0,  0, 22,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 20,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 21,  0,  0,  0,  0],
       [ 0,  1,  0,  0,  0,  0, 13,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 16,  0,  1],
       [ 0,  1,  0,  0,  0,  0,  0,  1, 15,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 16]])
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(69.0, 0.5, 'Truth')
```




```
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
```

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


model = GaussianNB()
model.fit(dataset.data, dataset.target)
```

```
GaussianNB()
```

```
## Predictions


expected = dataset.target
predicted = model.predict(dataset.data)


print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       178
           1       0.83      0.85      0.84       182
           2       0.98      0.64      0.77       177
           3       0.94      0.79      0.86       183
           4       0.98      0.84      0.90       181
           5       0.91      0.93      0.92       182
           6       0.96      0.99      0.98       181
           7       0.72      0.99      0.83       179
           8       0.58      0.86      0.69       174
           9       0.94      0.71      0.81       180

    accuracy                           0.86      1797
   macro avg       0.88      0.86      0.86      1797
```

```
weighted avg       0.89      0.86      0.86      1797

[[176   0   0   0   1   0   0   1   0   0]
 [  0 154   0   0   0   0   3   5  14   6]
 [  0  13 113   0   0   1   1   0  49   0]
 [  0   2   2 145   0   6   0   7  20   1]
 [  1   1   0   0 152   1   2  21   3   0]
 [  0   0   0   3   0 169   1   6   2   1]
 [  0   1   0   0   0   1 179   0   0   0]
 [  0   0   0   0   1   1   0 177   0   0]
 [  0   8   0   1   0   3   0  12 150   0]
 [  1   6   0   5   1   3   0  17  20 127]]
```

```
# Multinomial Naive Bayes

from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()


model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       178
           1       0.87      0.75      0.81       182
           2       0.90      0.90      0.90       177
           3       0.99      0.87      0.93       183
           4       0.96      0.96      0.96       181
           5       0.97      0.86      0.91       182
           6       0.98      0.97      0.98       181
           7       0.89      0.99      0.94       179
           8       0.78      0.89      0.83       174
           9       0.76      0.88      0.82       180

    accuracy                           0.91      1797
   macro avg       0.91      0.91      0.91      1797
weighted avg       0.91      0.91      0.91      1797

[[175   0   0   0   3   0   0   0   0   0]
 [  0 137  14   0   0   1   2   0  13  15]
 [  0   7 160   0   0   0   0   0   8   2]
 [  0   0   2 159   0   2   0   5   8   7]
 [  1   0   0   0 173   0   0   4   3   0]
 [  0   0   0   0   1 157   1   1   2  20]
 [  0   2   0   0   1   1 176   0   1   0]
 [  0   0   0   0   0   0   0 178   1   0]
 [  0  11   1   0   1   0   1   1 154   5]
 [  0   1   0   1   1   1   0  11   7 158]]
```

```
## Bernoulli Naive bayes
from sklearn.naive_bayes import BernoulliNB
model = BernoulliNB()
```

```
model.fit(dataset.data, dataset.target)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       178
           1       0.76      0.62      0.68       182
           2       0.86      0.86      0.86       177
           3       0.91      0.86      0.88       183
           4       0.91      0.95      0.93       181
           5       0.93      0.82      0.87       182
           6       0.97      0.94      0.96       181
           7       0.88      0.98      0.93       179
           8       0.70      0.82      0.75       174
           9       0.76      0.81      0.78       180

    accuracy                           0.86      1797
   macro avg       0.87      0.86      0.86      1797
weighted avg       0.87      0.86      0.86      1797

[[175   1   0   0   2   0   0   0   0   0]
 [  0 112  21   0   3   1   1   1  32  11]
 [  0   6 153   6   0   0   0   1  11   0]
 [  1   1   3 157   0   2   0   3   7   9]
 [  0   1   0   0 172   0   0   7   1   0]
 [  2   3   0   2   1 149   2   0   3  20]
 [  0   5   0   0   2   2 171   0   1   0]
 [  0   0   0   0   3   0   0 175   1   0]
 [  0  13   1   4   0   3   2   2 142   7]
 [  0   6   0   3   7   3   0   9   6 146]]
```

```
##

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline


def Naive_bayes(Model_Type):
        # import some data to play with
        iris = datasets.load_iris()
        X = iris.data[:, :2]  # we only take the first two features.
        Y = iris.target
        h = .02  # step size in the mesh
        # we create an instance of Neighbours Classifier and fit the data.
        if(Model_Type=='Gaussian'):
            model =  GaussianNB()
        elif (Model_Type=='Multinomial'):
                model =  MultinomialNB()
        else:
                model =  BernoulliNB()
```

```
        model.fit(X, Y)
        # Plot the decision boundary. For that, we will assign a color to each
        # point in the mesh [x_min, m_max]x[y_min, y_max].
        x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
        y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
        Z = model.predict(np.c_[xx.ravel(), yy.ravel()])

        # Put the result into a color plot
        Z = Z.reshape(xx.shape)
        plt.figure(1, figsize=(4, 3))
        plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

        # Plot also the training points
        plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
        plt.xlabel('Sepal length')
        plt.ylabel('Sepal width')
        plt.xlim(xx.min(), xx.max())
        plt.ylim(yy.min(), yy.max())
        plt.xticks(())
        plt.yticks(())
        plt.show()

        model.fit(dataset.data, dataset.target)
        expected = dataset.target
        predicted = model.predict(dataset.data)
        print(metrics.classification_report(expected, predicted))
        print(metrics.confusion_matrix(expected, predicted))


from IPython.html import widgets
from IPython.html.widgets import interact
from IPython.display import display
import warnings
warnings.filterwarnings('ignore')



i = interact(Naive_bayes, Model_Type=['Gaussian','Multinomial','Bernoulli'])
```