Name: Kshitij V Darwhekar

Roll No: TETB19

Sub: Soft Computitng

Batch: B2

4. NAIVE BAYES

```
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 1 2	0.99 0.83 0.98	0.99 0.85 0.64	0.99 0.84 0.77	178 182 177
3	0.94	0.79	0.86	183
4 5	0.98 0.91	0.84 0.93	0.90 0.92	181 182
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 macro avg weighted avg					0.8			0.86 0.86		0.86 0.86 0.86	1797 1797 1797
[[1	76	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			on	recall f1-score				support	
			0.00							
0			0.99				9.98		0.99	178
	1			0.8			9.75		0.81	182
		2		0.9	90	(0.90		0.90	177
	3			0.9	99	(0.87		0.93	183
		4		0.9	96	(9.96		0.96	181
		5		0.9	97	(0.86		0.91	182
		6		0.9	98	(0.97		0.98	181
		7		0.8	39	(0.99		0.94	179
		8		0.7	78	(0.89		0.83	174
		9		0.7	76	0.88			0.82	180
а	nccura	асу							0.91	1797
	icro a	_		0.9	91	(0.91		0.91	1797
weighted avg			0.9	91	(0.91		0.91	1797	
[[175	0	0	0	3	0	0	0	0	0]	
[6	137	14	0	0	1	2	0	13	15]	
[6	7	160	0	0	0	0	0	8	2]	
[6	0	2	159	0	2	0	5	8	7]	
[1	. 0	0	0	173	0	0	4	3	0]	
[6		0	0	1	157	1	1	2	20]	
[6		0	0	1	1	176	0	1	0]	
[6		0	0	0	0	0	178	1	0]	
[6		1	0	1	0	1	1		5]	
[6		0	1	1	1	0	11	7		
L	_	0	_	_	_	0		,		

```
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
                         0.98
                                 0.98
                                             0.98
                                                         178

      0.76
      0.62

      0.86
      0.86

      0.91
      0.86

      0.91
      0.95

      0.93
      0.82

                1
                                              0.68
                                                         182
                2
                                                         177
                                             0.86
                                                        183
                3
                                            0.88
                                           0.93
                4
                                                         181
                5
                                            0.87
                                                         182
                6
                       0.97
                                 0.94
                                            0.96
                                                        181

      0.88
      0.98
      0.93

      0.70
      0.82
      0.75

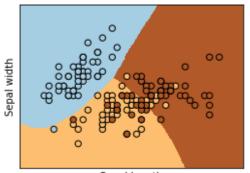
      0.76
      0.81
      0.78

                7
                                                         179
                8
                                                         174
                                                        180
                                             0.86
         accuracy
                                                    1797
                       0.87
                                 0.86
                                             0.86
                                                       1797
        macro avg
                        0.87
                                  0.86
                                              0.86
                                                       1797
     weighted avg
     [[175 1
                    0 2
                                              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
                                             91
      [ 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
                                               71
        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()

from sklearn.naive_bayes import 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'])
```



Senal	LI GO	na	th

		Sepal I							
		pred	cisio	on	re	call	f1	-score	support
0				99	(0.99		0.99	178
	1		0.8	33	(0.85		0.84	182
2			0.9	98	(0.64		0.77	177
	3		0.9	94	(0.79		0.86	183
	4		0.9	98	(0.84		0.90	181
	5		0.9	91	(0.93		0.92	182
	6		0.9	96	(0.99		0.98	181
	7		0.7	72	(0.99		0.83	179
	8		0.5	58	(0.86		0.69	174
ccura	acv							И. Xh	1/9/
			0 9	22	(3 86			1797
weighted avg									1797
	0								
0	0	0	1	0	0	1	0	0]	
154	0	0	0	0	3	5	14	6]	
13	113	0	0	1	1	0	49	0]	
2	2	145	0	6	0	7	20	1]	
1	0	0	152	1	2	21	3	0]	
0	0	3						_	
1	0	0	0	1	179	0	0	0]	
0	0	0	1	1	0	177	0	0]	
8	0	1	0	3	0			0]	
6	0	5	1	3	0	17	20	127]]	
	0 154 13 2 1 0 1	0 1 2 3 4 5 6 7 8 ccuracy cro avg ted avg 154 0 13 113 2 2 1 0 0 0 1 0 0 0 8 0	pred 0 1 2 3 4 5 6 7 8 ccuracy cro avg ted avg 0 0 13 113 0 2 145 1 0 0 0 3 1 0 0 0 0 8 0 1	0 0.9 1 0.8 2 0.9 3 0.9 4 0.9 5 0.9 6 0.9 7 0.7 8 0.9 ccuracy cro avg 0.8 ted avg 0.8 113 113 0 0 2 2 145 0 1 0 0 152 0 0 3 0 1 0 0 0 0	precision 0 0.99 1 0.83 2 0.98 3 0.94 4 0.98 5 0.91 6 0.96 7 0.72 8 0.58 ccuracy cro avg 0.88 ted avg 0.89 0 0 0 1 0 154 0 0 0 0 13 113 0 0 1 2 2 145 0 6 1 0 0 152 1 0 0 3 0 169 1 0 0 0 1 8 0 1 0 3	precision red 0 0.99 1 0.83 2 0.98 3 0.94 4 0.98 5 0.91 6 0.96 7 0.72 8 0.58 ccuracy cro avg 0.88 ted avg 0.89 0 0 0 1 0 0 154 0 0 0 0 3 13 113 0 0 1 1 2 2 145 0 6 0 1 0 0 152 1 2 0 0 3 0 169 1 1 0 0 0 1 179 0 0 0 1 1 0 8 0 1 0 3 0	precision recall 0 0.99 0.99 1 0.83 0.85 2 0.98 0.64 3 0.94 0.79 4 0.98 0.84 5 0.91 0.93 6 0.96 0.99 7 0.72 0.99 8 0.58 0.86 ccuracy cro avg 0.88 0.86 cted avg 0.89 0.86 0 0 0 1 0 0 1 154 0 0 0 0 3 5 13 113 0 0 1 1 0 2 2 145 0 6 0 7 1 0 0 152 1 2 21 0 0 3 0 169 1 6 1 0 0 0 1 179 0 0 0 0 1 1 0 177 8 0 1 0 3 0 12	precision recall f1 0 0.99 0.99 1 0.83 0.85 2 0.98 0.64 3 0.94 0.79 4 0.98 0.84 5 0.91 0.93 6 0.96 0.99 7 0.72 0.99 8 0.58 0.86 ccuracy cro avg 0.88 0.86 ccuracy cro avg 0.88 0.86 0 0 0 1 0 0 1 0 154 0 0 0 0 3 5 14 13 113 0 0 1 1 0 49 2 2 145 0 6 0 7 20 1 0 0 152 1 2 21 3 0 0 3 0 169 1 6 2 1 0 0 0 1 179 0 0 0 0 0 1 1 0 177 0 8 0 1 0 3 0 12 150	precision recall f1-score 0 0.99 0.99 0.99 1 0.83 0.85 0.84 2 0.98 0.64 0.77 3 0.94 0.79 0.86 4 0.98 0.84 0.90 5 0.91 0.93 0.92 6 0.96 0.99 0.98 7 0.72 0.99 0.83 8 0.58 0.86 0.69 ccuracy 0.88 0.86 0.86 ced avg 0.89 0.86 0.86 0 0 0 1 0 0 1 0 0 1 0 0] 154 0 0 0 0 3 5 14 6] 13 113 0 0 1 1 0 49 0] 2 2 145 0 6 0 7 20 1] 1 0 0 152 1 2 21 3 0] 0 0 3 0 169 1 6 2 1] 1 0 0 0 1 179 0 0 0] 0 0 0 1 1 779 0 0 0] 8 0 1 0 3 0 12 150 0]