DSBDAL PRACTICAL :- 6

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,classifica
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
data =  pd.read_csv('C:/Users/HP/Downloads/Iris.csv')
data.head(5)
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [8]:
```
data.describe(include = 'all')
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

- Displaying Shape of the dataset and The Types of Species to Classify

In [10]:
```
print(data.shape)
data['Species'].unique()
```

```
(150, 6)
```

Out[10]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [12]:
```
x = data.iloc[:,1:5]
y = data.iloc[:,5:]
```

In [13]:
```
encode = LabelEncoder()
y = encode.fit_transform(y)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\preprocessing\_label.py:116: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [14]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_st
```

In [16]:
```
naive_bayes = GaussianNB()
naive_bayes.fit(x_train,y_train)
pred = naive_bayes.predict(x_test)
```

In [17]:
```
pred
```

Out[17]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
       0])
```

In [18]:
```
y_test
```

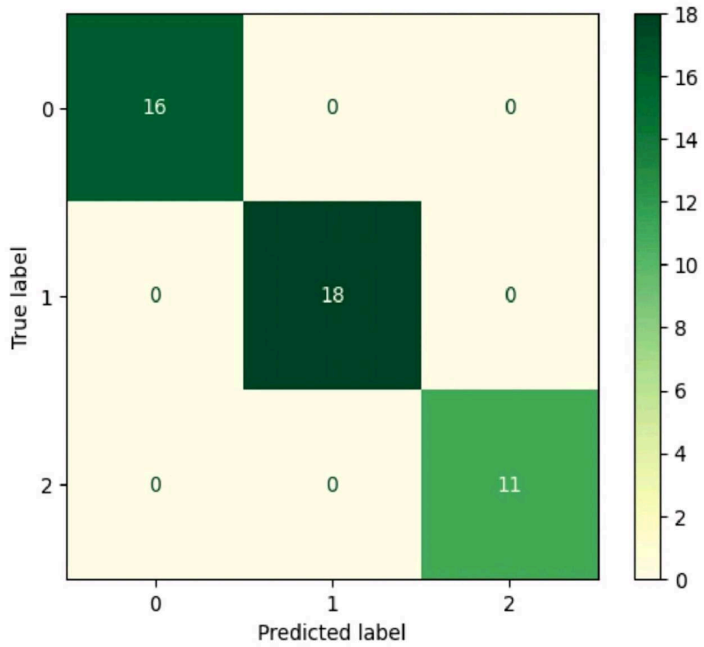
Out[18]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
       0])
```

In [19]:
```
matrix =  confusion_matrix(y_test,pred,labels = naive_bayes.classes_)
print(matrix)

tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

```
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
```

In [20]:
```
conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=na
conf_matrix.plot(cmap=plt.cm.YlGn)
plt.show()
```



In [21]:
```
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	1.00	1.00	1.00	18
2	1.00	1.00	1.00	11
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

In [22]:
```
print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```
Accuracy: 1.00
Error Rate:  0.0
Sensitivity (Recall or True positive rate) : 1.0
Specificity (True negative rate) : 1.0
Precision (Positive predictive value) : 1.0
False Positive Rate : 0.0
```