In [1]:

```
import pandas as pd
```

In [4]:

```
df = pd.read_csv('IRIS.csv')
```

In [5]:

```
df
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [6]:

```
df.shape
```

Out[6]:

```
(150, 5)
```

In [7]:

```
#input data
x=df.drop('species',axis=1)
#output data
y=df['species']
```

In [8]:

```
y.value_counts()
```

Out[8]:

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [9]:

```
#cross validation
from sklearn.model_selection import train_test_split
```

In [10]:

```
x_train ,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.25)
```

In [11]:

```
x_train.shape
```

Out[11]:

```
(112, 4)
```

In [12]:

```
x_test.shape
```

Out[12]:

```
(38, 4)
```

In [13]:

```
#import the class
from sklearn.naive_bayes import GaussianNB
```

In [14]:

```
#create the object
clf= GaussianNB()
```

In [15]:

```
#train the algorithm
clf.fit(x_train,y_train)
```

Out[15]:

```
▾ GaussianNB

GaussianNB()
```

In [16]:

```
y_pred=clf.predict(x_test)
```

In [18]:

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

In [20]:

```
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
```

In [21]:

```
confusion_matrix(y_test,y_pred)
```

Out[21]:

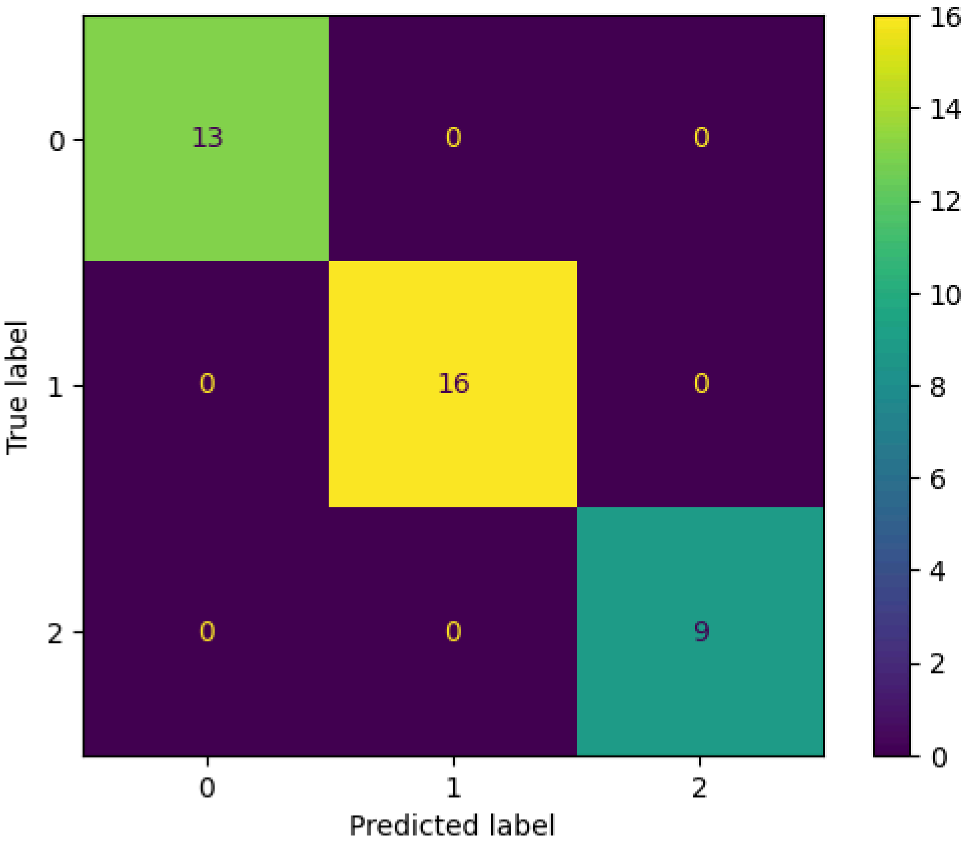
```
array([[13,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  9]])
```

In [23]:

```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

Out[23]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fe29bc90f90>
```



```
In [24]: accuracy_score(y_test,y_pred)
```

```
Out[24]: 1.0
```

```
In [25]: clf.predict_proba(x_test)
```

Out[25]:

```
array([[3.78600393e-230, 1.23816844e-006, 9.99998762e-001],
       [3.92599599e-084, 9.99998414e-001, 1.58647449e-006],
       [1.00000000e+000, 1.49614564e-018, 1.74760052e-027],
       [3.15666262e-310, 5.33743814e-007, 9.99999466e-001],
       [1.00000000e+000, 9.42168027e-017, 1.23200067e-026],
       [2.23021233e-320, 6.57075840e-011, 1.00000000e+000],
       [1.00000000e+000, 1.08515841e-016, 1.60182246e-026],
       [2.77799567e-148, 7.80950359e-001, 2.19049641e-001],
       [3.06136988e-152, 9.10103555e-001, 8.98964447e-002],
       [7.81436720e-094, 9.99887821e-001, 1.12179234e-004],
       [4.04457884e-214, 4.59787449e-001, 5.40212551e-001],
       [5.58268067e-133, 9.46482991e-001, 5.35170089e-002],
       [2.01640272e-134, 9.98906155e-001, 1.09384481e-003],
       [5.62315541e-141, 9.50340361e-001, 4.96596389e-002],
       [6.95450261e-142, 9.87982897e-001, 1.20171030e-002],
       [1.00000000e+000, 4.12311724e-017, 2.59560830e-027],
       [4.37029216e-132, 9.87665084e-001, 1.23349155e-002],
       [8.75281574e-113, 9.99940331e-001, 5.96690955e-005],
       [1.00000000e+000, 1.85096969e-015, 9.40528745e-026],
       [1.00000000e+000, 1.32004045e-015, 8.53461992e-025],
       [6.33048950e-186, 1.18626155e-002, 9.88137385e-001],
       [1.16157655e-130, 9.92205279e-001, 7.79472050e-003],
       [1.00000000e+000, 6.67164700e-013, 1.43294857e-022],
       [1.00000000e+000, 1.00711221e-016, 3.53778714e-027],
       [1.03247801e-168, 1.61227371e-001, 8.38772629e-001],
       [1.00000000e+000, 2.31435802e-018, 2.56440926e-028],
       [1.00000000e+000, 6.07384622e-011, 5.30906978e-020],
       [8.66734148e-112, 9.99340062e-001, 6.59938068e-004],
       [4.97577242e-047, 9.99999965e-001, 3.47984452e-008],
       [1.00000000e+000, 1.98255786e-013, 4.15458137e-023],
       [1.59908962e-226, 1.15450262e-003, 9.98845497e-001],
       [3.62555857e-130, 9.93956330e-001, 6.04366979e-003],
       [1.00000000e+000, 1.15724980e-016, 2.19223857e-026],
       [1.51508632e-175, 8.43422262e-002, 9.15657774e-001],
       [2.61323399e-261, 1.03689515e-006, 9.99998963e-001],
       [5.70846678e-090, 9.99950155e-001, 4.98452400e-005],
       [1.00000000e+000, 1.85676947e-013, 1.88622210e-022],
       [1.55783636e-180, 5.65567226e-001, 4.34432774e-001]])
```

In [26]:

```
newl=[[4.5,2.9,3.1,0.4]]
clf.predict(newl)[0]
```

```
/home/rllab-09/.local/lib/python3.11/site-packages/sklearn/base.py:465: UserWarning:
X does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

Out[26]:

```
'Iris-versicolor'
```

In [27]:

```
newl=[[5.5,3.1,1.0,0.8]]
clf.predict(newl)[0]
```

```
/home/rllab-09/.local/lib/python3.11/site-packages/sklearn/base.py:465: UserWarning:
X does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

Out[27]:

```
'Iris-setosa'
```

In [28]:

```
newl=[[6.5,3.3,4.9,1.8]]
clf.predict(newl)[0]
```

```
/home/rllab-09/.local/lib/python3.11/site-packages/sklearn/base.py:465: UserWarning:
X does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

Out[28]:

```
'Iris-virginica'
```

In [29]:

```
print(classification_report(y_test,y_pred))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      1.00      1.00        16
 Iris-virginica       1.00      1.00      1.00         9

       accuracy                           1.00        38
      macro avg       1.00      1.00      1.00        38
   weighted avg       1.00      1.00      1.00        38
```

In [ ]:

```

```