

```
In [2]: import pandas as pd
```

```
In [6]: df = pd.read_csv('iris.csv')
```

```
In [7]: df.shape
```

```
Out[7]: (150, 5)
```

```
In [9]: x=df.drop('species',axis=1)
        y=df['species']
```

```
In [10]: y.value_counts()
```

```
Out[10]: species
         setosa      50
         versicolor  50
         virginica   50
         Name: count, dtype: int64
```

```
In [11]: from sklearn.model_selection import train_test_split
```

```
In [12]: x_train ,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size
```

```
In [13]: x_train.shape
```

```
Out[13]: (112, 4)
```

```
In [14]: x_test.shape
```

```
Out[14]: (38, 4)
```

```
In [15]: from sklearn.naive_bayes import GaussianNB
```

```
In [16]: clf= GaussianNB()
```

```
In [17]: clf.fit(x_train,y_train)
```

```
Out[17]: ▾ GaussianNB ⓘ ?
          GaussianNB()
```

```
In [18]: y_pred=clf.predict(x_test)
```

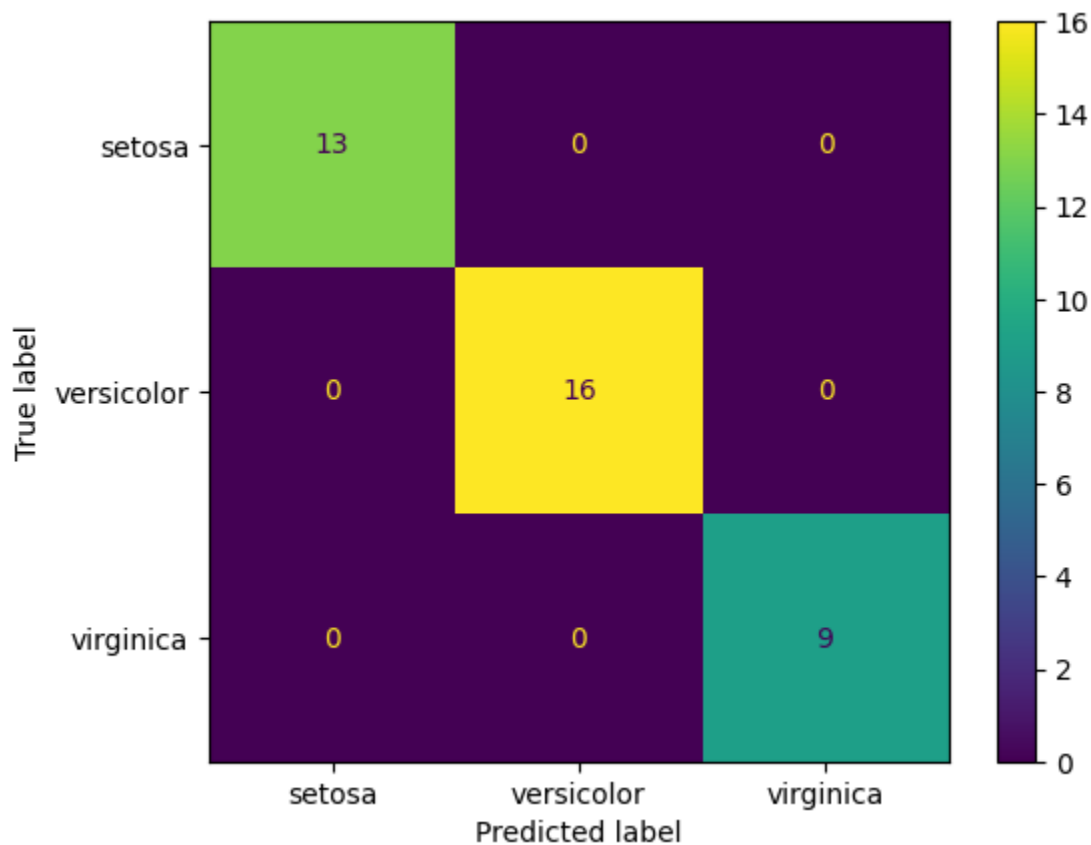
```
In [21]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import ConfusionMatrixDisplay
```

```
In [22]: confusion_matrix(y_test,y_pred)
```

```
Out[22]: array([[13,  0,  0],
                [ 0, 16,  0],
                [ 0,  0,  9]])
```

```
In [24]: ConfusionMatrixDisplay.from_estimator(clf,x_test,y_test)
```

```
Out[24]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f1e5457df10>
```



```
In [25]: accuracy_score(y_test,y_pred)
```

```
Out[25]: 1.0
```

```
In [26]: clf.predict_proba(x_test)
```

```
Out[26]: array([[2.05841140e-233, 1.23816844e-006, 9.99998762e-001],
 [1.76139943e-084, 9.9998414e-001, 1.58647449e-006],
 [1.00000000e+000, 1.48308613e-018, 1.73234612e-027],
 [6.96767669e-312, 5.33743814e-007, 9.9999466e-001],
 [1.00000000e+000, 9.33944060e-017, 1.22124682e-026],
 [4.94065646e-324, 6.57075840e-011, 1.00000000e+000],
 [1.00000000e+000, 1.05531886e-016, 1.55777574e-026],
 [2.45560284e-149, 7.80950359e-001, 2.19049641e-001],
 [4.01160627e-153, 9.10103555e-001, 8.98964447e-002],
 [1.46667004e-094, 9.99887821e-001, 1.12179234e-004],
 [5.29999917e-215, 4.59787449e-001, 5.40212551e-001],
 [4.93479766e-134, 9.46482991e-001, 5.35170089e-002],
 [5.23735688e-135, 9.98906155e-001, 1.09384481e-003],
 [4.97057521e-142, 9.50340361e-001, 4.96596389e-002],
 [9.11315109e-143, 9.87982897e-001, 1.20171030e-002],
 [1.00000000e+000, 7.81797826e-019, 1.29694954e-028],
 [3.86310964e-133, 9.87665084e-001, 1.23349155e-002],
 [2.27343573e-113, 9.9940331e-001, 5.96690955e-005],
 [1.00000000e+000, 1.80007196e-015, 9.14666201e-026],
 [1.00000000e+000, 1.30351394e-015, 8.42776899e-025],
 [4.66537803e-188, 1.18626155e-002, 9.88137385e-001],
 [1.02677291e-131, 9.92205279e-001, 7.79472050e-003],
 [1.00000000e+000, 6.61341173e-013, 1.42044069e-022],
 [1.00000000e+000, 9.98321355e-017, 3.50690661e-027],
 [2.27898063e-170, 1.61227371e-001, 8.38772629e-001],
 [1.00000000e+000, 2.29415652e-018, 2.54202512e-028],
 [1.00000000e+000, 5.99780345e-011, 5.24260178e-020],
 [1.62676386e-112, 9.99340062e-001, 6.59938068e-004],
 [2.23238199e-047, 9.99999965e-001, 3.47984452e-008],
 [1.00000000e+000, 1.95773682e-013, 4.10256723e-023],
 [3.52965800e-228, 1.15450262e-003, 9.98845497e-001],
 [3.20480410e-131, 9.93956330e-001, 6.04366979e-003],
 [1.00000000e+000, 1.14714843e-016, 2.17310302e-026],
 [3.34423817e-177, 8.43422262e-002, 9.15657774e-001],
 [5.60348582e-264, 1.03689515e-006, 9.9998963e-001],
 [7.48035097e-091, 9.99950155e-001, 4.98452400e-005],
 [1.00000000e+000, 1.80571225e-013, 1.83435499e-022],
 [8.97496247e-182, 5.65567226e-001, 4.34432774e-001]])
```

```
In [31]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	1.00	1.00	16
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 [ ]:
```