In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

In [2]:

```
data = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/iris-
```

In [3]:

```
data.shape
```

Out[3]:

```
(150, 5)
```

In [4]:

```
data.head()
```

Out[4]:

	sepal length	sepal width	petal length	petal width	class
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

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal length  150 non-null    float64
 1   sepal width   150 non-null    float64
 2   petal length  150 non-null    float64
 3   petal width   150 non-null    float64
 4   class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:

```
data.describe()
```

Out[6]:

	sepal length	sepal width	petal length	petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
sepal length    0
sepal width     0
petal length    0
petal width     0
class           0
dtype: int64
```

In [8]:

```
X = data.drop(['class'], axis=1)
y = data.drop(['sepal length', 'sepal width', 'petal length', 'petal width'], axis
```

In [9]:

```
 print(X.shape)
print(y.shape)
```

```
(150, 4)
(150, 1)
```

In [12]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=T
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(120, 4)
(30, 4)
(120, 1)
(30, 1)
```

In [13]:

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

Out[13]:

```
GaussianNB()
```

In [14]:

```
y_pred = model.predict(X_test)
model.score(X_test,y_test)
```

Out[14]:

```
0.9666666666666667
```

In [15]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDispl
print(accuracy_score(y_test, y_pred))
```

```
0.9666666666666667
```

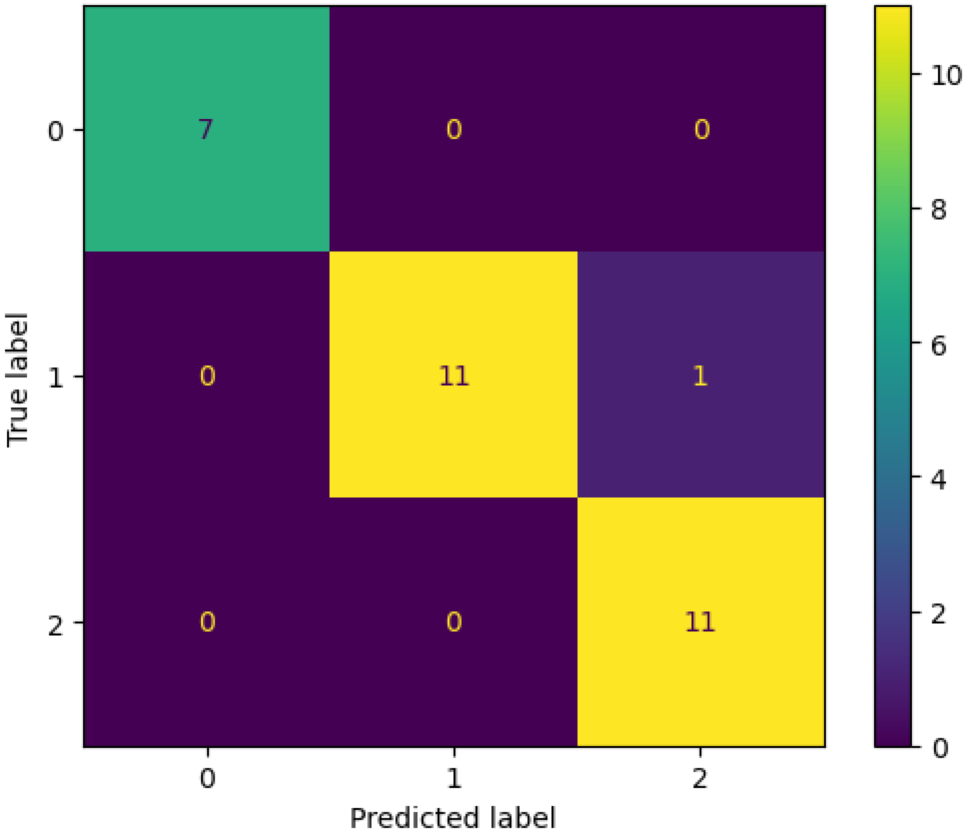
In [16]:

```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = cm)
print("Confusion matrix:")
print(cm)
```

```
Confusion matrix:
[[ 7  0  0]
 [ 0 11  1]
 [ 0  0 11]]
```

In [17]:

```
disp.plot()
plt.show()
```



In [19]:

```
def get_confusion_matrix_values(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    return(cm[0][0], cm[0][1], cm[1][0], cm[1][1])

TP, FP, FN, TN = get_confusion_matrix_values(y_test, y_pred)
print("TP: ", TP)
print("FP: ", FP)
print("FN: ", FN)
print("TN: ", TN)
```

```
TP:  7
FP:  0
FN:  0
TN:  11
```

In [20]:

```
print("The Accuracy is ", (TP+TN)/(TP+TN+FP+FN))
print("The precision is ", TP/(TP+FP))
print("The recall is ", TP/(TP+FN))
```

```
The Accuracy is  1.0
The precision is  1.0
The recall is  1.0
```

In [ ]: