In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
df = pd.read_csv("C:/Users/ameya/OneDrive/Desktop/DSBDAL/Iris.csv")
```

In [4]:

```
df.head()
```

Out[4]:

| | 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 [5]:

```
df.shape
```

Out[5]:

```
(150, 6)
```

In [6]:

```
df.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
```

In [7]:

```
df.describe()
```

Out[7]:

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

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [9]:

```
df['Species'].unique()
```

Out[9]:

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

In [10]:

```
x = df.drop(['Species'],axis=1)
y = df['Species']
```

In [11]:

```
x
```

Out[11]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 |
| **...** | ... | ... | ... | ... | ... |
| **145** | 146 | 6.7 | 3.0 | 5.2 | 2.3 |
| **146** | 147 | 6.3 | 2.5 | 5.0 | 1.9 |
| **147** | 148 | 6.5 | 3.0 | 5.2 | 2.0 |
| **148** | 149 | 6.2 | 3.4 | 5.4 | 2.3 |
| **149** | 150 | 5.9 | 3.0 | 5.1 | 1.8 |

150 rows × 5 columns

In [12]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
x_scaled = scalar.fit_transform(x)
```

In [13]:

```
x_scaled
```

Out[13]:

```
array([[-1.72054204e+00, -9.00681170e-01,  1.03205722e+00,
        -1.34127240e+00, -1.31297673e+00],
       [-1.69744751e+00, -1.14301691e+00, -1.24957601e-01,
        -1.34127240e+00, -1.31297673e+00],
       [-1.67435299e+00, -1.38535265e+00,  3.37848329e-01,
        -1.39813811e+00, -1.31297673e+00],
       [-1.65125846e+00, -1.50652052e+00,  1.06445364e-01,
        -1.28440670e+00, -1.31297673e+00],
       [-1.62816394e+00, -1.02184904e+00,  1.26346019e+00,
        -1.34127240e+00, -1.31297673e+00],
       [-1.60506942e+00, -5.37177559e-01,  1.95766909e+00,
        -1.17067529e+00, -1.05003079e+00],
       [-1.58197489e+00, -1.50652052e+00,  8.00654259e-01,
        -1.34127240e+00, -1.18150376e+00],
       [-1.55888037e+00, -1.02184904e+00,  8.00654259e-01,
        -1.28440670e+00, -1.31297673e+00],
       [-1.53578584e+00, -1.74885626e+00, -3.56360566e-01,
        -1.34127240e+00. -1.31297673e+00].
```

In [14]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)
```

In [15]:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train,y_train)
```

Out[15]:

```
GaussianNB()
```

In [16]:

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

In [17]:

```
y_pred
```

Out[17]:

```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setos
a',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor'], dtype='<U15')
```

In [21]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[21]:

```
array([[13,  0,  0],
       [ 0, 16,  0],
       [ 0,  0,  9]], dtype=int64)
```

In [22]:

```
#For Setosa Class
tp=cm[0][0]
fn=(cm[0][1])+(cm[0][2])
tn=(cm[1][1])+(cm[1][2])+(cm[2][1])+(cm[2][2])
fp=(cm[1][0])+(cm[2][0])
print('true positive: ',tp)
print('false positive: ',fp)
print('true negative: ',tn)
print('false negative: ',fn)
error_rate=(fp+fn)/(tp+tn+fp+fn)
print('error rate:', error_rate )
```

```
true positive:  13
false positive:  0
true negative:  25
false negative:  0
error rate: 0.0
```

In [23]:

```
#For Versicolor Class
tp=cm[1][1]
fn=(cm[1][0])+(cm[1][2])
tn=(cm[0][0])+(cm[0][2])+(cm[2][0])+(cm[2][2])
fp=(cm[0][1])+(cm[2][1])
print('true positive: ',tp)
print('false positive: ',fp)
print('true negative: ',tn)
print('false negative: ',fn)
error_rate=(fp+fn)/(tp+tn+fp+fn)
print('error rate:', error_rate )
```

```
true positive:  16
false positive:  0
true negative:  22
false negative:  0
error rate: 0.0
```

In [24]:

```
#For Virginca Class
tp=cm[1][2]
fn=(cm[2][0])+(cm[2][1])
tn=(cm[0][0])+(cm[0][1])+(cm[1][0])+(cm[1][1])
fp=(cm[0][2])+(cm[1][2])
print('true positive: ',tp)
print('false positive: ',fp)
print('true negative: ',tn)
print('false negative: ',fn)
error_rate=(fp+fn)/(tp+tn+fp+fn)
print('error rate:', error_rate )
```

```
true positive:  0
false positive:  0
true negative:  29
false negative:  0
error rate: 0.0
```

In [ ]:

```
```

In [23]:

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

Out[23]:

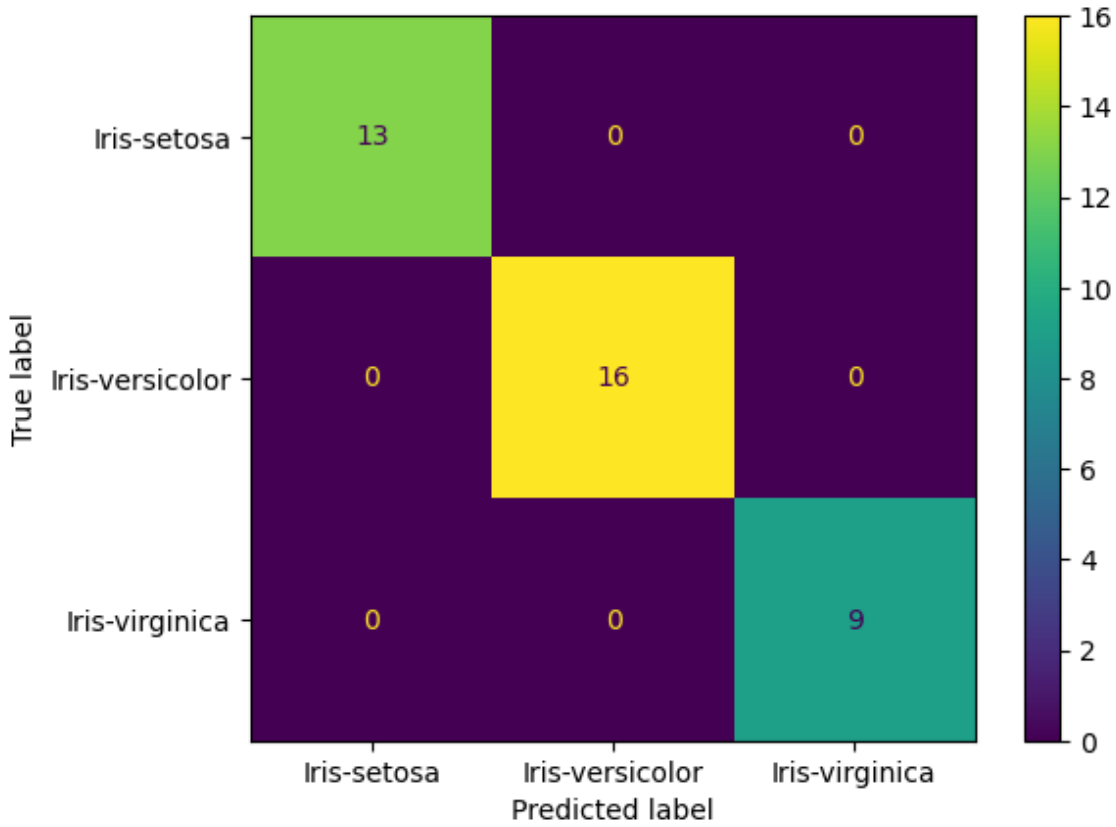
```
1.0
```

In [24]:

```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

Out[24]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x23351
3363d0>
```



In [27]:

```
from sklearn.metrics import classification_report
cr = classification_report(y_test,y_pred)
print(cr)
```

```
                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 [ ]:

```

```