Import

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import confusion_matrix, accuracy_score, classification_r
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [4]:
```
data=pd.read_csv('Iris.csv')
```

In [5]:
```
data
```

Out[5]:

	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
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Load Data

In [13]:
```
iris = load_iris()
iris.keys()
```

Out[13]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_name
s', 'filename', 'data_module'])
```

In [14]:
```
x = pd.DataFrame(iris['data'], columns=iris['feature_names'])
y = pd.DataFrame(iris['target'], columns=['target'])
```

In [17]:
```
x
```

Out[17]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [18]:
```
y
```

Out[18]:

	target
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

150 rows × 1 columns

Basic Stats

In [15]:
```
x.head()
```

Out[15]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [16]:
```
y.head()
```

Out[16]:

	target
0	0
1	0
2	0
3	0
4	0

In [19]:
```
x.head(10)
```

Out[19]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

In [20]:
```
y.head(10)
```

Out[20]:

	target
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

In [21]:
```
x.tail()
```

Out[21]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

In [22]:
```
y.tail()
```

Out[22]:

	target
145	2
146	2
147	2
148	2
149	2

In [23]:
```
x.tail(10)
```

Out[23]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

In [24]:
```
y.tail(10)
```

Out[24]:

	target
140	2
141	2
142	2
143	2
144	2
145	2
146	2
147	2
148	2
149	2

In [25]:
```
x.shape, y.shape
```

Out[25]:
```
((150, 4), (150, 1))
```

In [26]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [27]:
```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   target  150 non-null    int32
dtypes: int32(1)
memory usage: 728.0 bytes
```

In [28]:
```
x.describe()
```

Out[28]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
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 [29]:
```
y.describe()
```

Out[29]:

	target
count	150.000000
mean	1.000000
std	0.819232
min	0.000000
25%	0.000000
50%	1.000000
75%	2.000000
max	2.000000

Data Preparation

In [30]:
```
scaler = StandardScaler()
x = scaler.fit_transform(x.values)
```

In [31]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y.values, test_size=0.2
```

In [32]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[32]:
```
((120, 4), (30, 4), (120, 1), (30, 1))
```

Model Building

In [33]:
```
model = GaussianNB()
```

In [34]:
```
model.fit(x_train, y_train)
```

Out[34]:
```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
y_pred = model.predict(x_test)
```

Evaluation

In [36]:

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [38]:

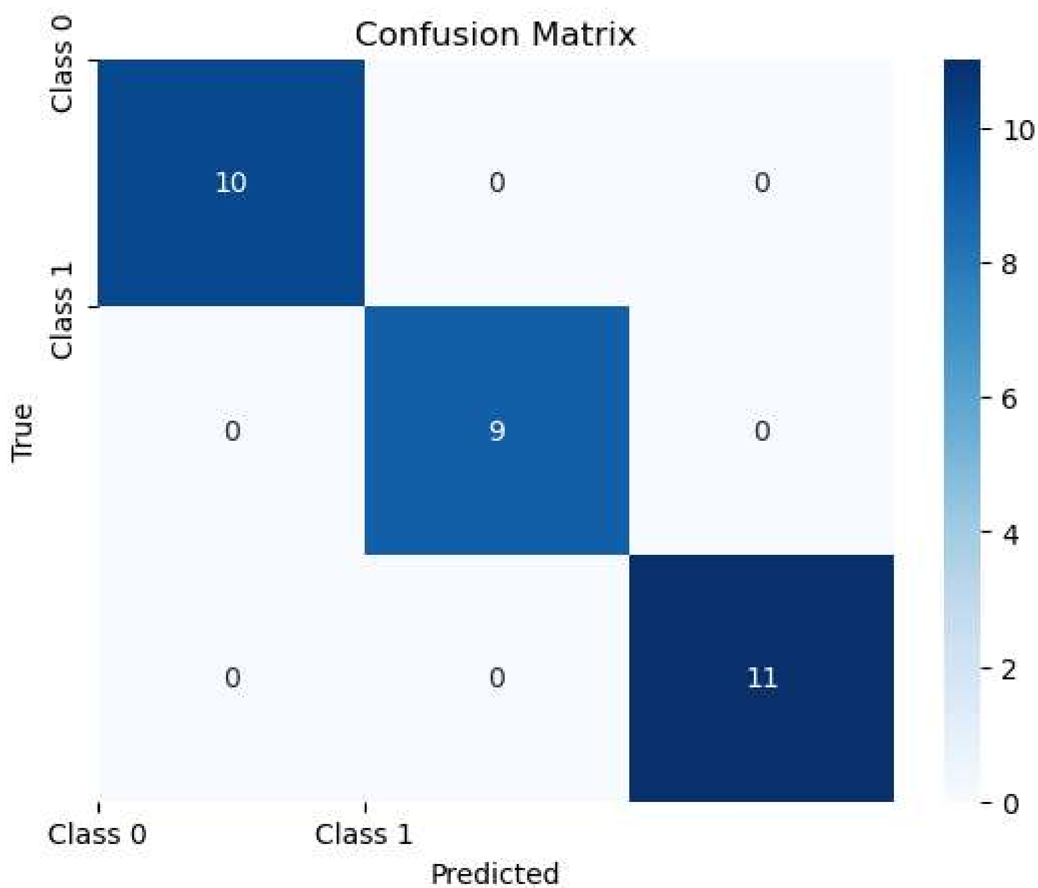
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you have already computed the confusion matrix `cm`

# Plot the confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

# Set labels, title, and axis ticks
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.xticks(ticks=[0, 1], labels=['Class 0', 'Class 1'])
plt.yticks(ticks=[0, 1], labels=['Class 0', 'Class 1'])

# Display the plot
plt.show()
```



In [39]:
```
print(f"TP value is {cm[0,0]}")
print(f"TN value is {cm[1,1] + cm[2,2]}")
print(f"FP value is {cm[0,1] + cm[0,2]}")
print(f"FN value is {cm[1,0] + cm[2,0]}")
```

```
TP value is 10
TN value is 20
FP value is 0
FN value is 0
```

In [40]:
```
print(f"Accuracy score is {accuracy_score(y_test, y_pred)}")
```

```
Accuracy score is 1.0
```

In [41]:
```
print(f"Error rate is {1 - accuracy_score(y_test, y_pred)}")
```

```
Error rate is 0.0
```

In [42]:
```
print(f"Precision score is {precision_score(y_test, y_pred, average='macro')}"
```

```
Precision score is 1.0
```

In [43]:
```
print(f"Recall score is {recall_score(y_test, y_pred, average='macro')}")
```

```
Recall score is 1.0
```

In [44]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [ ]: