In [55]:

```
'''1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precisior
dataset.'''
```

Out[55]:

```
'1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.c
sv dataset.\n2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error r
ate, Precision, Recall on the given\ndataset.'
```

In [56]:

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

In [57]:

```
df = pd.read_csv(r'C:\Users\ankit\Downloads\archive (2)\IRIS.csv')
df.head(10)
```

Out[57]:

	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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

In [58]:

```
X = df.iloc[:,0:4]
y = df.iloc[:,-1]
y
```

Out[58]:

```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: species, Length: 150, dtype: object
```

In [59]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size = 0.8, random_st
X_test
```

Out[59]:

	sepal_length	sepal_width	petal_length	petal_width
14	5.8	4.0	1.2	0.2
98	5.1	2.5	3.0	1.1
75	6.6	3.0	4.4	1.4
16	5.4	3.9	1.3	0.4
131	7.9	3.8	6.4	2.0
56	6.3	3.3	4.7	1.6
141	6.9	3.1	5.1	2.3
44	5.1	3.8	1.9	0.4
29	4.7	3.2	1.6	0.2
120	6.9	3.2	5.7	2.3
94	5.6	2.7	4.2	1.3
5	5.4	3.9	1.7	0.4
102	7.1	3.0	5.9	2.1
51	6.4	3.2	4.5	1.5
78	6.0	2.9	4.5	1.5
42	4.4	3.2	1.3	0.2
92	5.8	2.6	4.0	1.2
66	5.6	3.0	4.5	1.5
31	5.4	3.4	1.5	0.4
35	5.0	3.2	1.2	0.2
90	5.5	2.6	4.4	1.2
84	5.4	3.0	4.5	1.5
77	6.7	3.0	5.0	1.7
40	5.0	3.5	1.3	0.3
125	7.2	3.2	6.0	1.8
99	5.7	2.8	4.1	1.3
33	5.5	4.2	1.4	0.2
19	5.1	3.8	1.5	0.3
73	6.1	2.8	4.7	1.2
146	6.3	2.5	5.0	1.9

In [60]:

```
from sklearn.preprocessing import LabelEncoder
la_object = LabelEncoder()
y = la_object.fit_transform(y)

y
```

Out[60]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [61]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

Out[61]:
```
▾ GaussianNB

GaussianNB()
```

In [62]:
```
y_predicted = model.predict(X_test)
```

In [63]:
```
y_predicted
```

Out[63]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica'], dtype='<U15')
```

In [64]:
```
model.score(X_test, y_test)
```

Out[64]:
```
0.9666666666666667
```

In [65]:
```
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, y_predicted)
```

In [66]:
```
cm
```

Out[66]:
```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]], dtype=int64)
```

In [67]:
```
cl_report=classification_report(y_test,y_predicted)
```

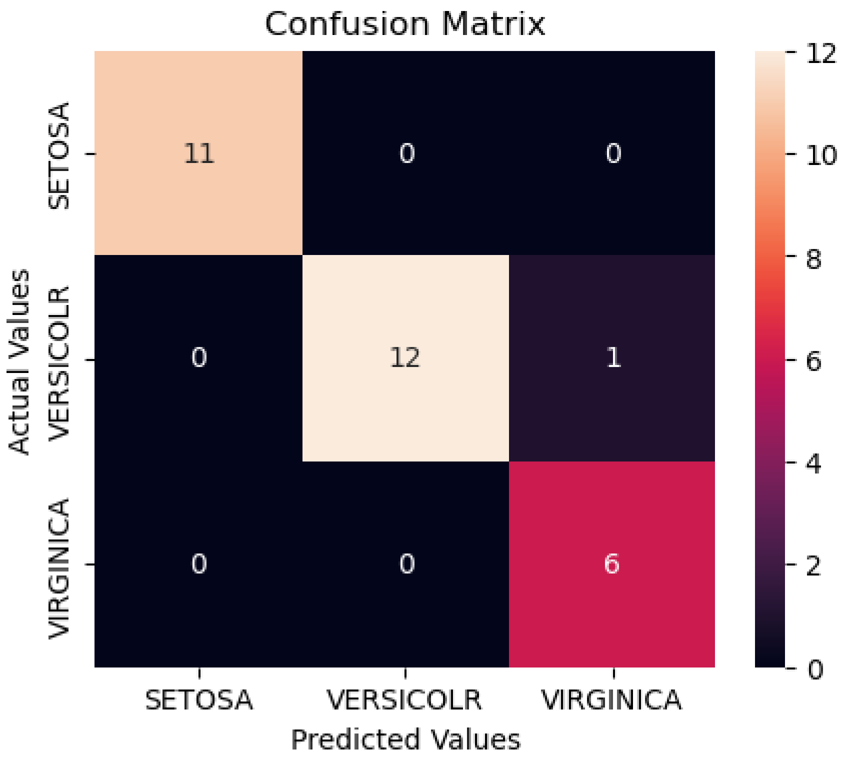
In [68]:
```
cl_report
```

Out[68]:
```
'                  precision    recall  f1-score   support\n\n    Iris-setosa
1.00      1.00      1.00        11\nIris-versicolor       1.00      0.92      0.96
13\n Iris-virginica       0.86      1.00      0.92         6\n\n       accuracy
0.97        30\n      macro avg       0.95      0.97      0.96        30\n   weigh
ted avg       0.97      0.97      0.97        30\n'
```

In [69]:
```
cm_df = pd.DataFrame(cm,
index = ['SETOSA','VERSICOLR','VIRGINICA'],
columns = ['SETOSA','VERSICOLR','VIRGINICA'])
```

In [70]:
```
import seaborn as sns
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actual Values')
```

```python
plt.xlabel('Predicted Values')
plt.show()
```



In [71]:

```python
def accuracy_cm(tp,fn,fp,tn):
    return (tp+tn)/(tp+fp+tn+fn)

def precision_cm(tp,fn,fp,tn):
    return tp/(tp+fp)

def recall_cm(tp,fn,fp,tn):
    return tp/(tp+fn)

def f1_score(tp,fn,fp,tn):
    return (2/((1/recall_cm(tp,fn,fp,tn))+precision_cm(tp,fn,fp,tn)))

def error_rate_cm(tp,fn,fp,tn):
    return 1-accuracy_cm(tp,fn,fp,tn)
```

In [73]:

```python
#For Virginica
tp = cm[2][2]
fn = cm[2][0]+cm[2][1]
fp = cm[0][2]+cm[1][2]
tn = cm[0][0]+cm[0][1]+cm[1][0]+cm[1][1]
print("For Virginica \n")
print("tp = ",tp)
print("fn = ",fn)
print("fp = ",fp)
print("tn = ",tn)
print("Accuracy : ",accuracy_cm(tp,fn,fp,tn))
print("Precision : ",precision_cm(tp,fn,fp,tn))
print("Recall : ",recall_cm(tp,fn,fp,tn))
print("F1-Score : ",f1_score(tp,fn,fp,tn))
print("Error rate : ",error_rate_cm(tp,fn,fp,tn))
```

```
For Virginica

tp =  6
fn =  0
fp =  1
tn =  23
Accuracy :  0.9666666666666667
Precision :  0.8571428571428571
Recall :  1.0
F1-Score :  1.0769230769230769
Error rate :  0.033333333333333326
```

In [ ]: