In [84]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [85]:
```
df=pd.read_csv('Iris.csv')
```

In [86]:
```
df
```

Out[86]:

	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

In [87]:
```
df['Species'].unique()
```

Out[87]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [88]:
```
df.isnull().sum()
```

Out[88]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [89]:
```
df1=df.drop(columns='Id')
```

In [90]:
```
df1
```

Out[90]:

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

150 rows × 5 columns

In [91]:
```
#x is dependant ,y is independant
x=df1.drop(columns='Species')
y=df1['Species']
```

In [92]:
```
#Splitting dataset into Training and Test
from sklearn.model_selection import train_test_split
Xtrain,xtest,Ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)
```

In [93]:
```
xtest.head()
```

Out[93]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

In [94]:
```
#Scaling to interpret these features on the same scale
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
```

In [95]:
```
xtest=std.fit_transform(xtest)
Xtrain=std.fit_transform(Xtrain)
```

In [96]:
```
#Building model
from sklearn.naive_bayes import GaussianNB
model1=GaussianNB()
```

In [97]:
```
#fitting data to the model
model1.fit(Xtrain,Ytrain)
```

Out[97]:
▾ GaussianNB

GaussianNB()

In [98]:
```
#Predict dataset
y_predict=model1.predict(xtest)
y_predict
```

Out[98]:
```
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', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor'], dtype='<U15')
```

In [99]:
```
#Calculating Accuracy Score and confusion matrix
from sklearn.metrics import confusion_matrix,recall_score,precision_score,f1_score,accuracy_s
from sklearn import metrics
```

In [100…
```
accuracy=accuracy_score(y_predict,ytest)
```

In [101…
```
accuracy*100
```

Out[101]:
```
95.55555555555556
```

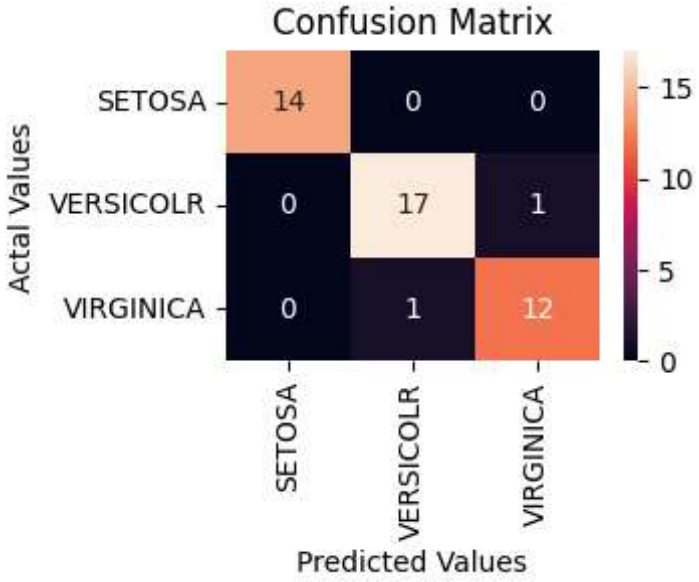
In [102…
```
con=confusion_matrix(y_predict,ytest)
```

In [103…
```
con
```

Out[103]:
```
array([[14,  0,  0],
       [ 0, 17,  1],
       [ 0,  1, 12]], dtype=int64)
```

In [104…
```
cm_df = pd.DataFrame(con,
                     index = ['SETOSA','VERSICOLR','VIRGINICA'],
                     columns = ['SETOSA','VERSICOLR','VIRGINICA'])
```

In [108…
```
plt.figure(figsize=(3,2))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()
```



In [109…
```
#Predicting new data
predict_new=model1.predict([[6.3,2.8,5.6,1.7],[4.6,3.1,1.4,0.2]])
```

In [110…
```
predict_new
```

Out[110]:
```
array(['Iris-virginica', 'Iris-virginica'], dtype='<U15')
```

In [ ]: