In [ ]:
```
'''
Data Analytics III

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

In [1]:
```
import pandas as pd

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

In [19]:
```
df = sns.load_dataset('iris')
```

In [20]:
```
df
```

Out[20]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [22]:
```
df.describe()
```

Out[22]:

	sepal_length	sepal_width	petal_length	petal_width
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 [23]:
```
df.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   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [24]:
```
df.dtypes
```

Out[24]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [25]:
```
np.unique(df['species'])
```

Out[25]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [27]:
```
X = df.iloc[:,0:4].values
y= df.iloc[:,4].values
```

In [28]:
```
df.iloc[:,0:4]
```

Out[28]:

	sepal_length	sepal_width	petal_length	petal_width
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 [30]:
```
df.iloc[:,0:4].values
```

Out[30]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

Test-size = 0.25

In [32]:
```
# Splitting the dataset into the Training set and Test set
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 = 42)
```

In [36]:
```
from sklearn.naive_bayes import GaussianNB

NB = GaussianNB()
NB.fit(X_train, y_train)
```

Out[36]:
```
▾ GaussianNB
GaussianNB()
```

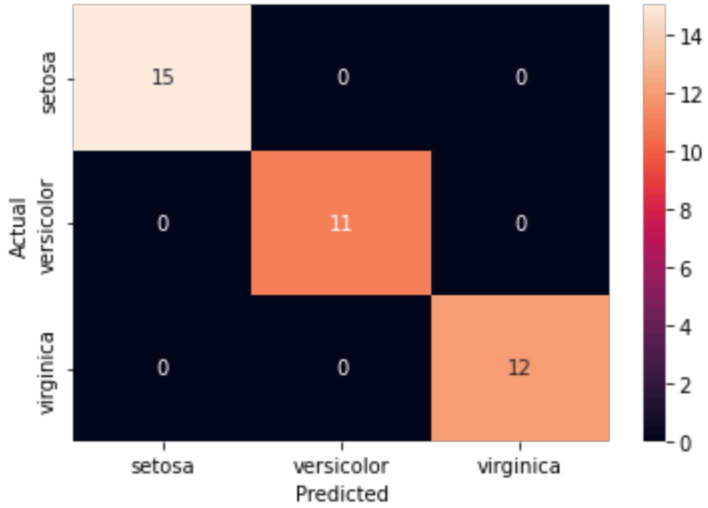
In [37]:
```
Y_pred = NB.predict(X_test)
```

In [38]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, Y_pred)
```

In [39]:
```
df_cm = pd.DataFrame(cm, columns=np.unique(y_test), index=np.unique(y_test))
```

In [40]:
```
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'

sns.heatmap(df_cm, annot=True)
plt.show()
```



In [41]:
```
print(cm)
# TP FP
# FN TN
```

```
[[15  0  0]
 [ 0 11  0]
 [ 0  0 12]]
```

In [44]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      1.00      1.00        11
   virginica       1.00      1.00      1.00        12

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38
```

In [46]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, Y_pred)
accuracy
```

Out[46]: 1.0

In [47]:
```
error_rate = 1-accuracy
error_rate
```

Out[47]: 0.0

Test-size = 0.2

In [48]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 20)
```

In [49]:
```
NB = GaussianNB()
NB.fit(X_train, y_train)
```

Out[49]:
```
▾ GaussianNB
GaussianNB()
```

In [50]:
```
Y_pred = NB.predict(X_test)
```

In [51]:
```
cm = confusion_matrix(y_test, Y_pred)
```

In [52]:
```
df_cm = pd.DataFrame(cm, columns=np.unique(y_test), index=np.unique(y_test))
```

In [53]:
```
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'

sns.heatmap(df_cm, annot=True)
plt.show()
```

In [54]:
```
print(classification_report(y_test,Y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         8
  versicolor       0.85      1.00      0.92        11
   virginica       1.00      0.82      0.90        11

    accuracy                           0.93        30
   macro avg       0.95      0.94      0.94        30
weighted avg       0.94      0.93      0.93        30
```

In [55]:
```
accuracy = accuracy_score(y_test, Y_pred)
accuracy
```

Out[55]: 0.9333333333333333

In [56]:
```
error_rate = 1-accuracy
error_rate
```

Out[56]: 0.06666666666666665

In [ ]: