Assignment N0. 6

Aim : 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.

Code :

In [1]:

```python
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```python
data = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/iris-data.csv")
data.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   class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [3]:

```python
data.shape
```

Out[3]:

```
(150, 5)
```

In [4]:

```python
data.head()
```

Out[4]:

	sepal length	sepal width	petal length	petal width	class
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

In [5]:

```python
data.tail()
```

Out[5]:

	sepal length	sepal width	petal length	petal width	class
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

In [6]:

```
data.describe()
```

Out[6]:

	sepal length	sepal width	petal length	petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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 [7]:

```
data.isnull().sum()
```

Out[7]:

```
sepal length    0
sepal width     0
petal length    0
petal width     0
class           0
dtype: int64
```

In [8]:

```
X = data.drop(['class'], axis=1)
y = data.drop(['sepal length',  'sepal width',  'petal length',  'petal width'], axis=1)
print(X)
print(y)
print(X.shape)
print(y.shape)
```

```
     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 x 4 columns]
              class
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

[150 rows x 1 columns]
(150, 4)
(150, 1)
```

In [9]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(120, 4)
(30, 4)
(120, 1)
(30, 1)
```

In [19]:

```
X_train
```

Out[19]:

	sepal length	sepal width	petal length	petal width
84	5.4	3.0	4.5	1.5
61	5.9	3.0	4.2	1.5
138	6.0	3.0	4.8	1.8
111	6.4	2.7	5.3	1.9
128	6.4	2.8	5.6	2.1
...	...	...	...	...
102	7.1	3.0	5.9	2.1
70	5.9	3.2	4.8	1.8
3	4.6	3.1	1.5	0.2
76	6.8	2.8	4.8	1.4
22	4.6	3.6	1.0	0.2

120 rows × 4 columns

In [20]:

```
X_test
```

Out[20]:

	sepal length	sepal width	petal length	petal width
17	5.1	3.5	1.4	0.3
147	6.5	3.0	5.2	2.0
12	4.8	3.0	1.4	0.1
99	5.7	2.8	4.1	1.3
58	6.6	2.9	4.6	1.3
71	6.1	2.8	4.0	1.3
144	6.7	3.3	5.7	2.5
121	5.6	2.8	4.9	2.0
110	6.5	3.2	5.1	2.0
15	5.7	4.4	1.5	0.4
2	4.7	3.2	1.3	0.2
20	5.4	3.4	1.7	0.2
100	6.3	3.3	6.0	2.5
105	7.6	3.0	6.6	2.1
91	6.1	3.0	4.6	1.4
31	5.4	3.4	1.5	0.4
32	5.2	4.1	1.5	0.1
142	5.8	2.7	5.1	1.9
127	6.1	3.0	4.9	1.8
36	5.5	3.5	1.3	0.2
93	5.0	2.3	3.3	1.0
104	6.5	3.0	5.8	2.2
119	6.0	2.2	5.0	1.5
103	6.3	2.9	5.6	1.8
101	5.8	2.7	5.1	1.9
81	5.5	2.4	3.7	1.0
11	4.8	3.4	1.6	0.2
125	7.2	3.2	6.0	1.8
149	5.9	3.0	5.1	1.8
109	7.2	3.6	6.1	2.5

In [21]:

```
y_train
```

Out[21]:

	class
84	Iris-versicolor
61	Iris-versicolor
138	Iris-virginica
111	Iris-virginica
128	Iris-virginica
...	...
102	Iris-virginica
70	Iris-versicolor
3	Iris-setosa
76	Iris-versicolor
22	Iris-setosa

120 rows × 1 columns

In [22]:

```
y_test
```

Out[22]:

	class
17	Iris-setosa
147	Iris-virginica
12	Iris-setosa
99	Iris-versicolor
58	Iris-versicolor
71	Iris-versicolor
144	Iris-virginica
121	Iris-virginica
110	Iris-virginica
15	Iris-setosa
2	Iris-setosa
20	Iris-setosa
100	Iris-virginica
105	Iris-virginica
91	Iris-versicolor
31	Iris-setosa
32	Iris-setosa
142	Iris-virginica
127	Iris-virginica
36	Iris-setosa
93	Iris-versicolor
104	Iris-virginica
119	Iris-virginica
103	Iris-virginica
101	Iris-virginica
81	Iris-versicolor
11	Iris-setosa
125	Iris-virginica
149	Iris-virginica
109	Iris-virginica

In [23]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler
```

Out[23]:

```
▾ MinMaxScaler
MinMaxScaler()
```

In [10]:

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

```
C:\Users\Welcome\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarni
ng: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_
samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[10]:

```
▾ GaussianNB
GaussianNB()
```

In [11]:

```
y_pred = model.predict(X_test)
model.score(X_test,y_test)
```

Out[11]:

```
0.9666666666666667
```

In [12]:

```
y_pred
```

Out[12]:

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

In [13]:

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

```
0.9666666666666667
```

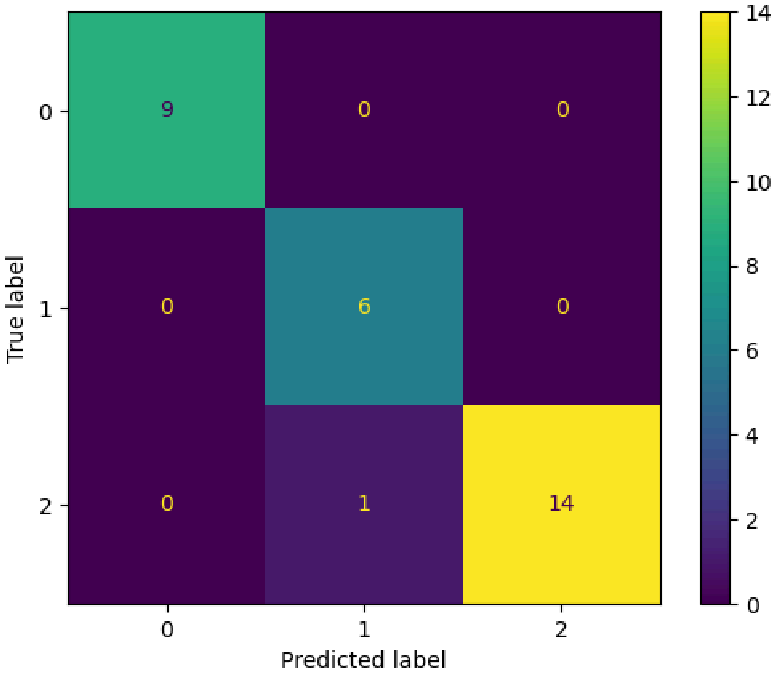
In [14]:

```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = cm)
print("Confusion matrix:")
print(cm)
```

```
Confusion matrix:
[[ 9  0  0]
 [ 0  6  0]
 [ 0  1 14]]
```

In [15]:

```
disp.plot()
plt.show()
```



In [16]:

```
def get_confusion_matrix_values(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    return(cm[0][0], cm[0][1], cm[1][0], cm[1][1])

TP, FP, FN, TN = get_confusion_matrix_values(y_test, y_pred)
print("TP: ", TP)
print("FP: ", FP)
print("FN: ", FN)
print("TN: ", TN)
```

```
TP:  9
FP:  0
FN:  0
TN:  6
```

In [17]:

```
print("The Accuracy is ", (TP+TN)/(TP+TN+FP+FN))
print("The precision is ", TP/(TP+FP))
print("The recall is ", TP/(TP+FN))
```

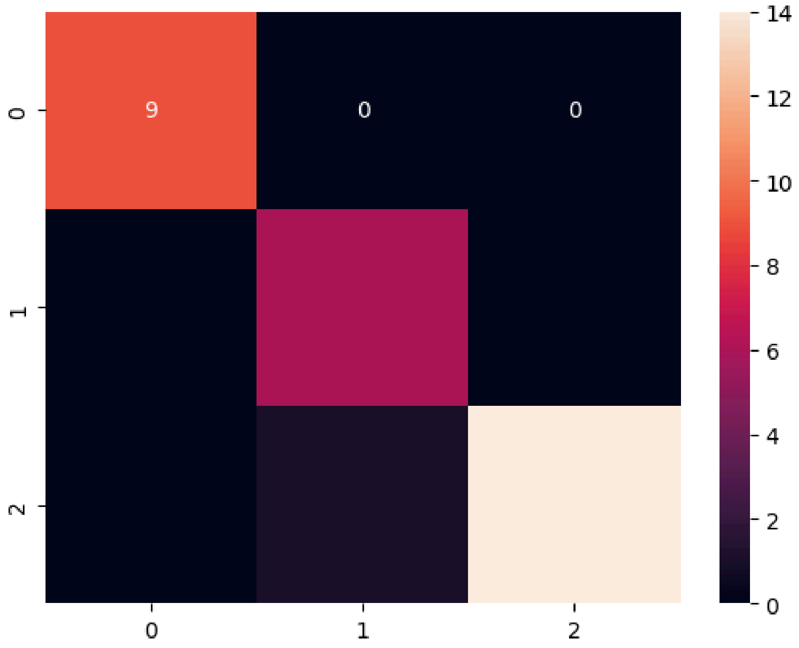
```
The Accuracy is  1.0
The precision is  1.0
The recall is  1.0
```

In [18]:

```
import seaborn as sns
sns.heatmap(cm, annot=True)
```

Out[18]:

```
<Axes: >
```



Name : Mohan Kadambande

Roll No. : 13212 (TECO-b1)