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

In [2]:

```
df = pd.read_csv("iris.csv")
df.head(10)
```

Out[2]:

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

In [3]:

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

Out[3]:

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

In [4]:

```
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_state=1)
X_test
```

Out[4]:

	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 [5]:

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

y
```

Out[5]:

```
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 [6]:

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

Out[6]:

```
GaussianNB()
```

In [7]:

```
y_predicted = model.predict(X_test)
```

In [8]:

```
y_predicted
```

Out[8]:

```
array(['setosa', 'versicolor', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'setosa', 'setosa', 'virginica',
       'versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor',
       'setosa', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'versicolor', 'versicolor', 'virginica', 'setosa', 'virginica',
       'versicolor', 'setosa', 'setosa', 'versicolor', 'virginica'],
      dtype='<U10')
```

In [9]:

```
model.score(X_test,y_test)
```

Out[9]:

```
0.9666666666666667
```

In [10]:

```
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(y_test, y_predicted)
```

In [11]:

```
cm
```

Out[11]:

```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]])
```

In [12]:

```
# classification report for precision, recall f1-score and accuracy
cl_report=classification_report(y_test,y_predicted)
```

In [13]:

```
cl_report
```

Out[13]:

```
'              precision    recall  f1-score   support\n\n      setosa
1.00      1.00      1.00        11\n  versicolor       1.00      0.92
0.96        13\n   virginica       0.86      1.00      0.92         6\n\n
accuracy                           0.97        30\n   macro avg       0.95
0.97      0.96        30\nweighted avg       0.97      0.97      0.97
30\n'
```

In [20]:

```
#          precision    recall  f1-score   support\n\n
#Setosa       1.00      1.00      1.00        11\n
#Versicolor   1.00      0.92      0.96        13\n
#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
#weighted avg    0.97      0.97      0.97        30\n
```

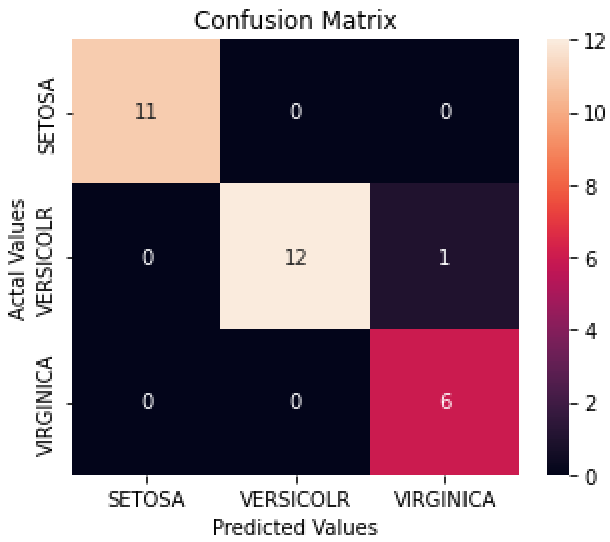
https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/
(https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/)

In [16]:

```
# Creating a dataframe for a array-formatted Confusion matrix,so it will be easy for pl
otting.
cm_df = pd.DataFrame(cm,
                     index = ['SETOSA','VERSICOLR','VIRGINICA'],
                     columns = ['SETOSA','VERSICOLR','VIRGINICA'])
```

In [17]:

```
#Plotting the confusion matrix
import seaborn as sns
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()
```



In [18]:

```
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 [19]:

```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("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

Accuracy    :  0.9666666666666667
Precision   :  0.8571428571428571
Recall      :  1.0
F1-Score    :  1.0769230769230769
Error rate  :  0.033333333333333326
```