ASSIGNMENT 6

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 numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:
```
iris = load_iris()
iris.keys()
```

Out[2]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'f
ilename', 'data_module'])
```

In [3]:
```
x = pd.DataFrame(iris['data'], columns=iris['feature_names'])
y = pd.DataFrame(iris['target'], columns=['target'])
```

In [4]:
```
x.head()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

In [5]:
```
x.tail()
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

In [6]:
```
x.sample(5)
```

Out[6]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
2	4.7	3.2	1.3	0.2
32	5.2	4.1	1.5	0.1
128	6.4	2.8	5.6	2.1
10	5.4	3.7	1.5	0.2
7	5.0	3.4	1.5	0.2

In [7]:
```
x.shape, y.shape
```

Out[7]:
```
((150, 4), (150, 1))
```

In [8]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [9]:
```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   target  150 non-null    int64
dtypes: int64(1)
memory usage: 1.3 KB
```

In [10]:
```
x.describe()
```

Out[10]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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

Data preparation

In [11]:
```
scaler = StandardScaler()
x = scaler.fit_transform(x.values)
```

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y.values, test_size=0.3, ran
```

In [13]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[13]:
```
((105, 4), (45, 4), (105, 1), (45, 1))
```

Model building

In [14]:
```
model = GaussianNB()
```

In [15]:
```
model.fit(x_train, y_train)
```

Out[15]:

▾ GaussianNB

```
GaussianNB()
```

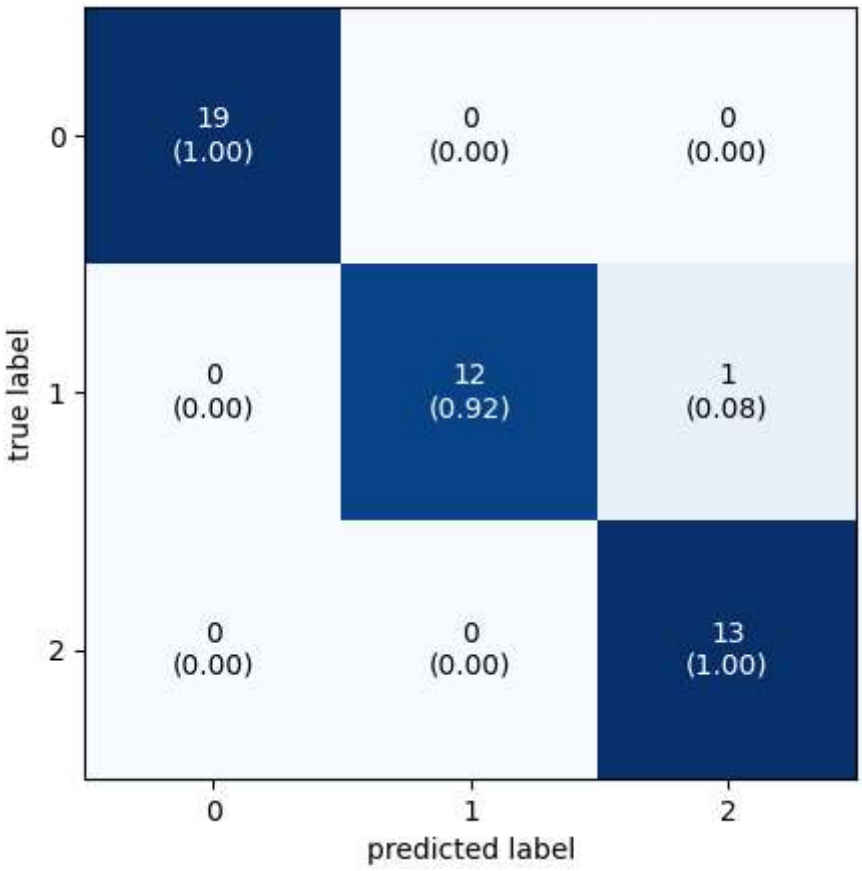
In [16]:
```
y_pred = model.predict(x_test)
```

Confusion Matrix

In [17]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
```

In [18]:
```
plot_confusion_matrix(conf_mat=cm, figsize=(5,5), show_normed=True)
plt.show()
```



In [19]:
```
print(f"TP value is {cm[0,0]}")
print(f"TN value is {cm[1,1] + cm[2,2]}")
print(f"FP value is {cm[0,1] + cm[0,2]}")
print(f"FN value is {cm[1,0] + cm[2,0]}")
```

```
TP value is 19
TN value is 25
FP value is 0
FN value is 0
```

In [20]:
```
tp = 19
tn = 25
fp = 0
fn = 0
```

In [21]:
```
print('Accuracy score is :',(tn+tp)/(tn+fp+fn+tp))
```

```
Accuracy score is : 1.0
```

In [22]:
```
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
```

```
Error Rate:  0.0
```

In [23]:
```
print('Precision :',tp/(tp+fp))
```

```
Precision : 1.0
```

In [24]:
```
print('Recall :',tp/(tp+fn))
```

```
Recall : 1.0
```