Shivam Wagh

DSBDA Practical A-6: Implement Simple Naïve Bayes classification algorithm using Python/R on iris

Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

In [1]:
```
# https://www.kaggle.com/datasets/uciml/iris
```

In [2]:
```
import pandas as pd
```

In [8]:
```
df = pd.read_excel(r"C:\Users\Admin\Desktop\TEB56\Iris.xlsx")
```

In [9]:
```
df.head()
```

Out[9]:

	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

In [11]:
```
df.tail()
```

Out[11]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

In [12]:
```
df.shape
```

Out[12]:
```
(150, 6)
```

In [13]:
```
df.describe()
```

Out[13]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [14]:
```
x=df.drop(['Id', 'Species'], axis=1)
```

In [15]:
```
x.head()
```

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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 [16]:
```
x.shape
```

Out[16]:
```
(150, 4)
```

In [17]:
```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
df['Species']=label.fit_transform(df['Species'])
```

In [18]:
```
y=df['Species']
y.head()
```

Out[18]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int32
```

In [19]:
```
print(y.shape)
```

```
(150,)
```

In [20]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest=train_test_split(x,y,test_size=0.2, random_state=0)
```

In [21]:
```
xtrain.shape
```

Out[21]:
```
(120, 4)
```

In [22]:
```
xtest.shape
```

Out[22]:
```
(30, 4)
```

In [23]:
```
ytrain.shape
```

Out[23]:
```
(120,)
```

In [24]:
```
ytest.shape
```

Out[24]:
```
(30,)
```

In [26]:
```
from sklearn.naive_bayes import GaussianNB
```

In [27]:
```
model=GaussianNB()
```

In [28]:
```
model.fit(xtrain, ytrain)  # Train the model
```

Out[28]:
```
▾ GaussianNB

GaussianNB()
```

In [29]:
```
ypred=model.predict(xtest) # Predict on test data
```

In [30]:
```
model.score(xtest, ytest)
```

Out[30]:
```
0.9666666666666667
```

In [31]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [32]:
```
accuracy_score(ytest, ypred)
```

Out[32]:
```
0.9666666666666667
```

In [33]:
```
cm=confusion_matrix(ytest, ypred)
print(cm)
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

In [35]:
```
print(classification_report(ytest, ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.93      1.00      0.96        13
           2       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

In [ ]:

In [ ]: