DSBDA Lab Assignment No. 6

Name: Akash Ganesh Padir
Roll No.: TEB04

In [44]:
```
#Step 1: Importing the Libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [45]:
```
#Step 2: Importing the dataset

dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Classification/master/IrisDataset.csv')
```

In [46]:
```
dataset.head()
```

Out[46]:

	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

In [47]:
```
gk=dataset.groupby('species')
```

In [48]:
```
gk.first()
```

Out[48]:

species	sepal_length	sepal_width	petal_length	petal_width
setosa	5.1	3.5	1.4	0.2
versicolor	7.0	3.2	4.7	1.4
virginica	6.3	3.3	6.0	2.5

In [49]:
```
#Step 3:

X=dataset.iloc[:,:4].values
y=dataset['species'].values
```

In [50]:
```
#Step 4:

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)
```

In [51]:
```
#Step 5:

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [52]:
```
#Step 6: Training the Naive Bayes Classification model

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

Out[52]:
```
GaussianNB()
```

In [53]:
```
#Step 7:

y_pred=classifier.predict(X_test)
y_pred
```

Out[53]:
```
array(['virginica', 'virginica', 'virginica', 'virginica', 'virginica',
       'versicolor', 'virginica', 'virginica', 'virginica', 'setosa',
       'setosa', 'versicolor', 'setosa', 'virginica', 'setosa', 'setosa',
       'setosa', 'versicolor', 'setosa', 'versicolor', 'versicolor',
       'virginica', 'setosa', 'setosa', 'virginica', 'versicolor',
       'virginica', 'virginica', 'virginica', 'setosa'], dtype='<U10')
```

In [54]:
```
#Step 8: Confusion Matrix and Accuracy

from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test,y_pred))
cm
```

```
Accuracy :  0.9333333333333333
```

Out[54]:
```
array([[10,  0,  0],
       [ 0,  5,  1],
       [ 0,  1, 13]], dtype=int64)
```

In [55]:
```
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[55]:
```
array([[10,  0,  0],
       [ 0,  5,  1],
       [ 0,  1, 13]], dtype=int64)
```

New Method

In [56]:
```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
dataset1 = datasets.load_iris()
```

In [57]:
```
dataset1
```

Out[57]:
```
{'data': 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],
        [5.1, 3.8, 1.5, 0.3],
```

In [15]:
```
model = GaussianNB()
model.fit(dataset1.data, dataset1.target)
```

Out[15]:
```
GaussianNB()
```

In [16]:
```
#Making Predictions
expected = dataset1.target
predicted = model.predict(dataset1.data)
```

In [17]:
```
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.94      0.94      0.94        50
           2       0.94      0.94      0.94        50

    accuracy                           0.96       150
   macro avg       0.96      0.96      0.96       150
weighted avg       0.96      0.96      0.96       150

[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
```

In [20]:
```
###Importing Libraries
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split as tts
```

In [24]:
```
###Importing Dataset

iris = datasets.load_iris()

data = pd.DataFrame({"sl":iris.data[:,0], "sw":iris.data[:,1], "pl":iris.data[:,2], "pw":iris.data[:,3], 'species': iris.t

###Splitting train/test data
from sklearn.model_selection import train_test_split
X=data[['sl','sw','pl','pw']]
y=data["species"]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=0)
```

In [25]:
```
X_train
```

Out[25]:

	sl	sw	pl	pw
60	5.0	2.0	3.5	1.0
116	6.5	3.0	5.5	1.8
144	6.7	3.3	5.7	2.5
119	6.0	2.2	5.0	1.5
108	6.7	2.5	5.8	1.8
...	...	...	...	...
9	4.9	3.1	1.5	0.1
103	6.3	2.9	5.6	1.8
67	5.8	2.7	4.1	1.0
117	7.7	3.8	6.7	2.2
47	4.6	3.2	1.4	0.2

105 rows × 4 columns

In [26]:
```
X_test
```

Out[26]:

	sl	sw	pl	pw
114	5.8	2.8	5.1	2.4
62	6.0	2.2	4.0	1.0
33	5.5	4.2	1.4	0.2
107	7.3	2.9	6.3	1.8
7	5.0	3.4	1.5	0.2
100	6.3	3.3	6.0	2.5
40	5.0	3.5	1.3	0.3
86	6.7	3.1	4.7	1.5
76	6.8	2.8	4.8	1.4
71	6.1	2.8	4.0	1.3
134	6.1	2.6	5.6	1.4
51	6.4	3.2	4.5	1.5
73	6.1	2.8	4.7	1.2
54	6.5	2.8	4.6	1.5
63	6.1	2.9	4.7	1.4
37	4.9	3.6	1.4	0.1
78	6.0	2.9	4.5	1.5
90	5.5	2.6	4.4	1.2
45	4.8	3.0	1.4	0.3
16	5.4	3.9	1.3	0.4
121	5.6	2.8	4.9	2.0
66	5.6	3.0	4.5	1.5
24	4.8	3.4	1.9	0.2
8	4.4	2.9	1.4	0.2
126	6.2	2.8	4.8	1.8
22	4.6	3.6	1.0	0.2
44	5.1	3.8	1.9	0.4
97	6.2	2.9	4.3	1.3
93	5.0	2.3	3.3	1.0
26	5.0	3.4	1.6	0.4
137	6.4	3.1	5.5	1.8
84	5.4	3.0	4.5	1.5
27	5.2	3.5	1.5	0.2
127	6.1	3.0	4.9	1.8
132	6.4	2.8	5.6	2.2
59	5.2	2.7	3.9	1.4
18	5.7	3.8	1.7	0.3
83	6.0	2.7	5.1	1.6
61	5.9	3.0	4.2	1.5
92	5.8	2.6	4.0	1.2
112	6.8	3.0	5.5	2.1
2	4.7	3.2	1.3	0.2
141	6.9	3.1	5.1	2.3
43	5.0	3.5	1.6	0.6
10	5.4	3.7	1.5	0.2

In [27]:
```
y_train
```

Out[27]:
```
60     1
116    2
144    2
119    2
108    2
      ..
9      0
103    2
67     1
117    2
47     0
Name: species, Length: 105, dtype: int32
```

In [28]:
```
y_test
```

Out[28]:
```
114    2
62     1
33     0
107    2
7      0
100    2
40     0
86     1
76     1
71     1
134    2
51     1
73     1
54     1
63     1
37     0
78     1
90     1
45     0
16     0
121    2
66     1
24     0
8      0
126    2
22     0
44     0
97     1
93     1
26     0
137    2
84     1
27     0
127    2
132    2
59     1
18     0
83     1
61     1
92     1
112    2
2      0
141    2
43     0
10     0
Name: species, dtype: int32
```

In [29]:
```
#Naive Bayes The technique is easiest to understand when described using binary or
#categorical input values.Gaussian Naive Bayes:
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

```
Confusion matrix for Naive Bayes
 [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
accuracy_Naive Bayes: 1.000
precision_Naive Bayes: 1.000
recall_Naive Bayes: 1.000
f1-score_Naive Bayes : 1.000
```

In [30]:
```
#Multinomial Naive Bayes:
MNB = MultinomialNB(alpha=0.6)
MNB.fit(X_train, y_train)
Y_pred = MNB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_MNB = round(MNB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

```
Confusion matrix for Naive Bayes
 [[16  0  0]
 [ 0  0 18]
 [ 0  0 11]]
accuracy_Naive Bayes: 0.600
precision_Naive Bayes: 0.600
recall_Naive Bayes: 0.600
f1-score_Naive Bayes : 0.600
```

In [31]:
```
from sklearn.naive_bayes import BernoulliNB
BNB = BernoulliNB(fit_prior = False)
BNB.fit(X_train, y_train)
Y_pred = BNB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_MNB = round(BNB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

```
Confusion matrix for Naive Bayes
 [[ 0  0 16]
 [ 0  0 18]
 [ 0  0 11]]
accuracy_Naive Bayes: 0.244
precision_Naive Bayes: 0.244
recall_Naive Bayes: 0.244
f1-score_Naive Bayes : 0.244
```

In [32]:
```
#Complement Naive Bayes
from sklearn.naive_bayes import ComplementNB
CNB = ComplementNB(norm = True)
CNB.fit(X_train, y_train)
Y_pred = CNB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_MNB = round(CNB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

```
Confusion matrix for Naive Bayes
 [[16  0  0]
 [18  0  0]
 [ 8  3  0]]
accuracy_Naive Bayes: 0.356
precision_Naive Bayes: 0.356
recall_Naive Bayes: 0.356
f1-score_Naive Bayes : 0.356
```

In [36]:
```
###Importing Libraries
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split as tts
###Importing Dataset
iris = datasets.load_iris()
data = pd.DataFrame({"sl":iris.data[:,0], "sw":iris.data[:,1], "pl":iris.data[:,2], "pw":iris.data[:,3], 'species': iris.ta
###Splitting train/test data
from sklearn.model_selection import train_test_split
X=data[['sl','sw','pl','pw']]
y=data["species"]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=0)
X_train
X_test
y_train
y_test
#Naive Bayes The technique is easiest to understand when described using binary or
#categorical input values.
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
MNB = MultinomialNB(alpha=0.6)
MNB.fit(X_train, y_train)
Y_pred = MNB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_MNB = round(MNB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
from sklearn.naive_bayes import BernoulliNB
BNB = BernoulliNB(fit_prior = False)
BNB.fit(X_train, y_train)
Y_pred = BNB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_MNB = round(BNB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
from sklearn.naive_bayes import ComplementNB
CNB = ComplementNB(norm = True)
CNB.fit(X_train, y_train)
Y_pred = CNB.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_MNB = round(CNB.score(X_train, y_train) * 100, 2)
cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall = recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
```

```
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

```
Confusion matrix for Naive Bayes
 [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]
accuracy_Naive Bayes: 1.000
precision_Naive Bayes: 1.000
recall_Naive Bayes: 1.000
f1-score_Naive Bayes : 1.000
Confusion matrix for Naive Bayes
 [[16  0  0]
 [ 0  0 18]
 [ 0  0 11]]
accuracy_Naive Bayes: 0.600
precision_Naive Bayes: 0.600
recall_Naive Bayes: 0.600
f1-score_Naive Bayes : 0.600
Confusion matrix for Naive Bayes
 [[ 0  0 16]
 [ 0  0 18]
 [ 0  0 11]]
accuracy_Naive Bayes: 0.244
precision_Naive Bayes: 0.244
recall_Naive Bayes: 0.244
f1-score_Naive Bayes : 0.244
Confusion matrix for Naive Bayes
 [[16  0  0]
 [18  0  0]
 [ 8  3  0]]
accuracy_Naive Bayes: 0.356
precision_Naive Bayes: 0.356
recall_Naive Bayes: 0.356
f1-score_Naive Bayes : 0.356
```

In [37]:

```
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(X_test)
# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 95.0
```