#CS353 ML Lab 3

Name: K V Sumanth Reddy

Roll No: 181CO225

###Q: Write a program in python to implement the naïve Bayesian classifier and Bayesian classifier for a sample training data set. Compute the accuracy of the classifier

####Dataset Used: IRIS Dataset

##Importing Libraries and Dataset

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
```

In [2]:

```
dataset = pd.read_csv('IRIS.csv')
dataset.sample(10)
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
127	6.1	3.0	4.9	1.8	Iris-virginica
87	6.3	2.3	4.4	1.3	Iris-versicolor
24	4.8	3.4	1.9	0.2	Iris-setosa
69	5.6	2.5	3.9	1.1	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
56	6.3	3.3	4.7	1.6	Iris-versicolor
2	4.7	3.2	1.3	0.2	Iris-setosa
79	5.7	2.6	3.5	1.0	Iris-versicolor
84	5.4	3.0	4.5	1.5	Iris-versicolor
23	5.1	3.3	1.7	0.5	Iris-setosa

##Data Preprocessing

In [3]:

```
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

In [4]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_
```

##Training the Naive Bayes model on the Training set

In [5]:

```
model = GaussianNB()
model.fit(x_train,y_train)
acc1=model.score(x_test,y_test)*100
print("Accuracy: {:.2f}%".format(acc1))
```

```
Accuracy: 97.37%
```

In [6]:

```
y_pred = model.predict(x_test)
```

In [7]:

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

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      0.94      0.97        16
 Iris-virginica       0.90      1.00      0.95         9

       accuracy                           0.97        38
      macro avg       0.97      0.98      0.97        38
   weighted avg       0.98      0.97      0.97        38
```

In [8]:

```
print("Confusion Matrix:\n",confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
 [[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```

#USING FEATURE SCALING ###This is performed to normalize few columns (due to their higher values)

In [9]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [10]:

```
model = GaussianNB()
model.fit(x_train,y_train)
acc2=model.score(x_test,y_test)*100
print("Accuracy: {:.2f}%".format(acc2))
```

```
Accuracy: 97.37%
```

In [11]:

```
y_pred = model.predict(x_test)
```

In [12]:

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

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      0.94      0.97        16
 Iris-virginica       0.90      1.00      0.95         9

       accuracy                           0.97        38
      macro avg       0.97      0.98      0.97        38
   weighted avg       0.98      0.97      0.97        38
```

In [13]:

```
print("Confusion Matrix:\n",confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
 [[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```

#Accuracy Comparison with and without feature scaling

In [14]:

```
print(" Accuracy without Feature Scaling: ",acc1,"\n","Accuracy with Feature Scalin
```

```
 Accuracy without Feature Scaling:  97.36842105263158
 Accuracy with Feature Scaling:  97.36842105263158
```

In [14]:

```

```