#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: Breast Cancer Dataset

##Importing Libraries and Dataset

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.naive_bayes import GaussianNB
```

In [2]:

```
cancer = load_breast_cancer()
data = pd.DataFrame(cancer.data, columns=[cancer.feature_names])
data['Target'] = pd.Series(data=cancer.target, index=data.index)
data.sample(10)
```

Out[2]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmet
239	17.460	39.28	113.40	920.6	0.09812	0.12980	0.14170	0.08811	0.18
149	13.740	17.91	88.12	585.0	0.07944	0.06376	0.02881	0.01329	0.14
402	12.960	18.29	84.18	525.2	0.07351	0.07899	0.04057	0.01883	0.18
312	12.760	13.37	82.29	504.1	0.08794	0.07948	0.04052	0.02548	0.16
416	9.405	21.70	59.60	271.2	0.10440	0.06159	0.02047	0.01257	0.20
286	11.940	20.76	77.87	441.0	0.08605	0.10110	0.06574	0.03791	0.15
510	11.740	14.69	76.31	426.0	0.08099	0.09661	0.06726	0.02639	0.14
537	11.690	24.44	76.37	406.4	0.12360	0.15520	0.04515	0.04531	0.21
304	11.460	18.16	73.59	403.1	0.08853	0.07694	0.03344	0.01502	0.14
251	11.500	18.45	73.28	407.4	0.09345	0.05991	0.02638	0.02069	0.18

##Data Preprocessing

In [3]:

```
x,y = load_breast_cancer(return_X_y=True)
```

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: 93.71%
```

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

           0       0.92      0.91      0.91        53
           1       0.95      0.96      0.95        90

    accuracy                           0.94       143
   macro avg       0.93      0.93      0.93       143
weighted avg       0.94      0.94      0.94       143
```

In [8]:

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

```
Confusion Matrix:
 [[48  5]
 [ 4 86]]
```

#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: 91.61%
```

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

           0       0.89      0.89      0.89        53
           1       0.93      0.93      0.93        90

    accuracy                           0.92       143
   macro avg       0.91      0.91      0.91       143
weighted avg       0.92      0.92      0.92       143
```

In [13]:

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

```
Confusion Matrix:
 [[47  6]
 [ 6 84]]
```

#Accuracy Comparison with and without feature scaling

In [14]:

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

```
 Accuracy without Feature Scaling:  93.7062937062937
 Accuracy with Feature Scaling:  91.6083916083916
```