#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: Social Network Ads (https://www.kaggle.com/rakeshrau/social-network-ads (https://www.kaggle.com/rakeshrau/social-network-ads))

##Importing Libraries and Dataset

In [1]:

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

In [2]:

```
dataset = pd.read_csv('Social_Network_Ads.csv')
dataset.head(10)
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0

##Data Preprocessing

In [3]:

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

In [4]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder=
x = np.array(ct.fit_transform(x))
```

In [5]:

```
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 [6]:

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

```
Accuracy: 91.00%
```

In [7]:

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

In [8]:

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

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.94        68
           1       0.93      0.78      0.85        32

    accuracy                           0.91       100
   macro avg       0.92      0.88      0.89       100
weighted avg       0.91      0.91      0.91       100
```

In [9]:

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

```
Confusion Matrix:
 [[66  2]
 [ 7 25]]
```

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

In [10]:

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

In [11]:

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

```
Accuracy: 92.00%
```

In [12]:

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

In [13]:

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

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94        68
           1       0.93      0.81      0.87        32

    accuracy                           0.92       100
   macro avg       0.92      0.89      0.90       100
weighted avg       0.92      0.92      0.92       100
```

In [14]:

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

```
Confusion Matrix:
 [[66  2]
 [ 6 26]]
```

#Accuracy Comparison with and without feature scaling

In [15]:

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

```
 Accuracy without Feature Scaling:  91.0
 Accuracy with Feature Scaling:  92.0
```

In [15]:

```

```