In [1]:
```
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

import numpy as np
import math
```

In [2]:
```
bankdataset=pd.read_csv("Churn_Modelling.csv")
bankdataset.head()
```

Out[2]:

| | RowNumber | CustomerId | Surname | CreditScore | Geography | Gender | Age | Tenure | Ba |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 15634602 | Hargrave | 619 | France | Female | 42 | 2 | |
| **1** | 2 | 15647311 | Hill | 608 | Spain | Female | 41 | 1 | 83 |
| **2** | 3 | 15619304 | Onio | 502 | France | Female | 42 | 8 | 159 |
| **3** | 4 | 15701354 | Boni | 699 | France | Female | 39 | 1 | |
| **4** | 5 | 15737888 | Mitchell | 850 | Spain | Female | 43 | 2 | 125 |

In [3]:
```
bankdataset.shape
```

Out[3]: (10000, 14)

In [4]:
```
le=preprocessing.LabelEncoder()
bankdataset[["Gender"]] = le.fit_transform(bankdataset["Gender"])
bankdataset.loc[:,["CreditScore","Gender","Age","Tenure","Balance",
```

Out[4]:

| | CreditScore | Gender | Age | Tenure | Balance | NumOfProducts | HasCrCard | IsActiveMem |
|---|---|---|---|---|---|---|---|---|
| **0** | 619 | 0 | 42 | 2 | 0.00 | 1 | 1 | |
| **1** | 608 | 0 | 41 | 1 | 83807.86 | 1 | 0 | |
| **2** | 502 | 0 | 42 | 8 | 159660.80 | 3 | 1 | |
| **3** | 699 | 0 | 39 | 1 | 0.00 | 2 | 0 | |
| **4** | 850 | 0 | 43 | 2 | 125510.82 | 1 | 1 | |

In [5]:
```
X = bankdataset.loc[:,["CreditScore","Gender","Age","Tenure","Balan
y = bankdataset["Exited"]
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
```

In [7]:
```
class gaussain_Bank:
    def fit(self,x_train, y_train):
        self.x = x_train
        self.y = y_train

        X = X_train.join(y_train)
        self.x1 = X.loc[X.Exited == 1]
        self.x2 = X.loc[X.Exited == 0]

        self.y1 = self.x1.Exited
        self.y2 = self.x2.Exited

        self.x1 = self.x1.drop(columns=["Exited"])
        self.x2 = self.x2.drop(columns=["Exited"])

        self.meanX1 = self.x1.mean()
        self.meanX2 = self.x2.mean()

        self.sdX1 = self.x1.std()
        self.sdX2 = self.x2.std()

    def gaussainY1(self,x):
        gaussainValue = 1
        for Mean , SD , X in zip(self.meanX1, self.sdX1, x):
            gaussainValue *= (1/SD*(2*math.pi))*(math.e**((-1/2)*((
        return gaussainValue

    def gaussainY2(self,x):
        gaussainValue = 1
        for Mean , SD , X in zip(self.meanX2, self.sdX2, x):
            gaussainValue *= (1/SD*(2*math.pi))*(math.e**((-1/2)*((
        return gaussainValue

    def predict(self,x):
        y_predict = []
        for xi in x.index:
            gausY1 = self.gaussainY1(np.array(X_test.loc[[xi]])[0])
            gausY2 = self.gaussainY2(np.array(X_test.loc[[xi]])[0])
            if(gausY1 > gausY2):
                y_predict.append(1)
            else:
                y_predict.append(0)
        return np.array(y_predict)

    def score(self,x,y):
        y_predict = self.predict(x)
        counting = 0
        for pi , yi in zip(y_predict,y):
            if(pi == yi):
                counting += 1
        return counting/len(y_predict)
```

In [8]:
```
h = gaussain_Bank()
h.fit(X_train, y_train)
h.score(X_test, y_test)
```

Out[8]:
```
0.7515
```

In [9]:
```
h.predict(X_test)
```

Out[9]:
```
array([1, 0, 0, ..., 0, 0, 0])
```

In [10]:
```
model = GaussianNB()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Out[10]:
```
0.795
```

In [11]:
```
model.predict(X_test)
```

Out[11]:
```
array([1, 0, 0, ..., 0, 0, 0])
```