Naive Bayes' From Scratch

In [1]:

```
import numpy as np
import pandas as pd

# LET NORMAL = 0 , TUMOR = 1
data = {'Sample':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
        'Gene A':[1,1,0,-1,-1,-1,0,1,1,-1,1,0,0,-1,1],
        'Gene B':[1,1,1,0,-1,-1,-1,0,-1,0,0,0,1,0,-1],
        'Gene C':[1,1,1,1,0,0,0,1,0,0,0,1,0,1,1],
        'Gene D':[0,1,0,0,0,1,1,0,0,0,1,1,0,1,1],
        'Class': [0,0,1,1,1,0,1,0,1,1,1,1,1,0,'']}
df = pd.DataFrame(data)
df
```

Out[1]:

	Sample	Gene A	Gene B	Gene C	Gene D	Class
0	1	1	1	1	0	0
1	2	1	1	1	1	0
2	3	0	1	1	0	1
3	4	-1	0	1	0	1
4	5	-1	-1	0	0	1
5	6	-1	-1	0	1	0
6	7	0	-1	0	1	1
7	8	1	0	1	0	0
8	9	1	-1	0	0	1
9	10	-1	0	0	0	1
10	11	1	0	0	1	1
11	12	0	0	1	1	1
12	13	0	1	0	0	1
13	14	-1	0	1	1	0
14	15	1	-1	1	1	

In [2]:

```
data = np.array(df)
```

```
P(Xi|0) and P(Xi|1)
```

In [3]:

```
train = data[0:14,1:]
test = data[14:,1:]
```

In [4]:

```
x = train[:,:4]
y = train[:,4]
```

```
P ( Gene A | 0 ) and P ( Gene B | 1 )
Lets try to find for one feature and Class NORMAL
```

In [5]:

```
normal_value_0 = 0
normal_value_1 = 0
normal_value_neg1 = 0

# FOR CLASS NORMAL OR 0

for i  in range(0,14):
    if x[i][0] == 0 and y[i] == 0:
        normal_value_0 += 1
    elif x[i][0] == 1 and y[i] == 0:
        normal_value_1 += 1
    elif x[i][0] == -1 and y[i] == 0:
        normal_value_neg1 +=1
```

In [6]:

```
normal_value_0
```

Out[6]:

```
0
```

In [7]:

```
normal_value_1
```

Out[7]:

```
3
```

In [8]:

```
normal_value_neg1
```

Out[8]:

```
2
```

```
Let's Find the PROBABILITY of both our features
```

In [9]:

```
normal = 0
tumor = 0
for i  in range(0,14):
    if y[i] == 0:
        normal += 1
    else:
        tumor += 1
P_N = normal/(normal+tumor)
P_T = tumor/(normal+tumor)
print('P(Normal)' , P_N)
print('P(Tumor)' , P_T )
```

```
P(Normal) 0.35714285714285715
P(Tumor) 0.6428571428571429
```

In [10]:

```
5/14 , 9/14 # Works fine
```

Out[10]:

```
(0.35714285714285715, 0.6428571428571429)
```

In [11]:

```
normal , tumor
```

Out[11]:

```
(5, 9)
```

```
Creating two dataframe / array { for class Normal , Tumor }
to store the the occurance of each event
with respect to each Gene Expression ( -1 , 0 , +1 )
```

In [12]:

```
# DATA FRAME FOR CLASS NORMAL
data_normal = {'Expression':[1,0,-1],
        'Gene A':[0,0,0],
        'Gene B':[0,0,0],
        'Gene C':[0,0,0],
        'Gene D':[0,0,0] }
df_normal = pd.DataFrame(data_normal)

# DATA FRAME FOR CLASS TUMOR
data_tumor = {'Expression':[1,0,-1],
        'Gene A':[0,0,0],
        'Gene B':[0,0,0],
        'Gene C':[0,0,0],
        'Gene D':[0,0,0] }
df_tumor = pd.DataFrame(data_tumor)
```

In [13]:

```
output_array_normal = np.array(df_normal)
```

In [14]:

```
output_array_normal
```

Out[14]:

```
array([[ 1,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0],
       [-1,  0,  0,  0,  0]], dtype=int64)
```

```
Let's write the code that counts the occurance of each event
```

In [15]:

```
# FOR CLASSN " NORMAL OR 0 "
for j in range(0,4):
    for i  in range(0,14):
        if x[i][j] == 1 and y[i] == 0:
            output_array_normal[0,(j+1)] += 1
        elif x[i][j] == 0 and y[i] == 0:
            output_array_normal[1,(j+1)] += 1
        elif x[i][j] == -1 and y[i] == 0:
            output_array_normal[2,(j+1)] +=1
```

In [16]:

```
output_array_normal
```

Out[16]:

```
array([[ 1,  3,  2,  4,  3],
       [ 0,  0,  2,  1,  2],
       [-1,  2,  1,  0,  0]], dtype=int64)
```

In [17]:

```
output_array_tumor = np.array(df_tumor)
```

In [18]:

```
# FOR CLASS " TUMOR OR 1 "
for j in range(0,4):
    for i  in range(0,14):
        if x[i][j] == 1 and y[i] == 1:
            output_array_tumor[0,(j+1)] += 1
        elif x[i][j] == 0 and y[i] == 1:
            output_array_tumor[1,(j+1)] += 1
        elif x[i][j] == -1 and y[i] == 1:
            output_array_tumor[2,(j+1)] +=1
```

In [19]:

```
output_array_tumor
```

Out[19]:

```
array([[ 1,  2,  2,  3,  3],
       [ 0,  4,  4,  6,  6],
       [-1,  3,  3,  0,  0]], dtype=int64)
```

```
Let's create a function that calculates the probability
for each event for all the 4 features {GENE A , GENE B , GENE C , GENE D}
and store it as an array
```

In [20]:

```
prob_normal = {}
prob_tumor = {}
def probability(*feature_num):
    list_normal = []
    list_tumor = []
    for i in range(3):
        list_normal.append(output_array_normal[i,feature_num]/normal)
        list_tumor.append(output_array_tumor[i,feature_num]/tumor)
        prob_normal = np.array(list_normal)
        prob_tumor = np.array(list_tumor)
    return (('NORMAL' , prob_normal) , ('TUMOR' ,prob_tumor))
```

In [21]:

```
probability(1,2,3,4)
```

Out[21]:

```
(('NORMAL',
  array([[0.6, 0.4, 0.8, 0.6],
         [0. , 0.4, 0.2, 0.4],
         [0.4, 0.2, 0. , 0. ]])),
 ('TUMOR',
  array([[0.22222222, 0.22222222, 0.33333333, 0.33333333],
         [0.44444444, 0.44444444, 0.66666667, 0.66666667],
         [0.33333333, 0.33333333, 0.        , 0.        ]])))
```

In [44]:

```
def testing(Gene):
    test_prob_normal = []
    test_prob_tumor = []
    for i in range(3):
        test_prob_normal.append((output_array_normal[i,Gene]/normal)*P_N)
        test_prob_tumor.append((output_array_tumor[i,Gene]/tumor)*P_T)
    if test_prob_normal > test_prob_tumor:
        return "Class Normal" , test_prob_normal
    else:
        return "Class Tumor" , test_prob_tumor
```

In [51]:

```
testing(1)
```

Out[51]:

```
('Class Normal', [0.21428571428571427, 0.0, 0.14285714285714288])
```

Naive Bayes' using SK-learn

In [47]:

```
y = y.astype('int')
x_test = test[:,:4]
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x, y)
y_pred = gnb.predict(x_test)
```

In [49]:

```
y_pred # Normal
```

Out[49]:

```
array([0])
```