Naive Bayes

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
data = pd.read_csv('mushrooms.csv')
```

Defining accuracy function

In [2]:
```
correct = 0
def accuracy(a,b):
    correct = 0
    for i in range(len(a)):
        if(a[i] == b[i]):
              correct = correct+1

    accu = correct/len(a) * 100
    print("Accuracy is", + accu)


```

Defining sensitivity function

In [3]:
```

def Sensitivity():

    sensi = (TP / (TP+FN))*100
    print("Sensitivity is", + sensi)
```

Defining specificity function

In [4]:
```
def Specificity():
    speci  = (TN/(TN + FP))*100
    print("Specificity is", + speci)
```

In [5]:
```
data
```

Out[5]:

	cap_shape	Cap_surface	bruises	category
0	Convex	Smooth	Bruises	Poisonous
1	Convex	Smooth	Bruises	Edible
2	Bell	Smooth	Bruises	Edible
3	Convex	Scaly	Bruises	Poisonous
4	Convex	Smooth	NoBruises	Edible
...	...	...	...	...
8119	Knobbed	Smooth	NoBruises	Edible
8120	Convex	Smooth	NoBruises	Edible
8121	Flat	Smooth	NoBruises	Edible
8122	Knobbed	Scaly	NoBruises	Poisonous
8123	Convex	Smooth	NoBruises	Edible

8124 rows × 4 columns

In [6]:
```
y = data.category
X = data
#X = data.drop('category', axis=1)
#X=X[1:200]
#y=y[1:100]
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3)

print(y_train.shape)
```

```
(5686,)
```

In [ ]:
```

```

In [8]:
```
y_train.replace({'Poisonous': 'Negative', 'Edible' : 'Positive'}, inplace = True)#------------replacing values of bi
y_test.replace({'Poisonous': 'Negative', 'Edible' : 'Positive'}, inplace = True)
X_train['category'].replace({'Poisonous': 'Negative', 'Edible' : 'Positive'}, inplace = True)#------------replacing
X_test['category'].replace({'Poisonous': 'Negative', 'Edible' : 'Positive'}, inplace = True)



```

```
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\pandas\core\generic.py:6786: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y)
  self._update_inplace(new_data)
```

Defining probability function using m-estimate

In [9]:
```


def probability(a,b,m):#-------------defininf probility function with m-estimate

    p=(1/2)
    k = ((a+(m*p))/(b+m))
    return k

```

In [ ]:
```
1
```

In [10]:
```
1 y_train[1:10]
```

Out[10]:
```
3232    Negative
6629    Negative
4880    Negative
3703    Negative
7914    Negative
1365    Positive
5398    Negative
5435    Negative
1137    Positive
Name: category, dtype: object
```

In [11]:
```
1 print(y_train[y_train == 'Positive'].shape[0])
2 print(y_train[y_train == 'Negative'].shape[0])
```

```
2967
2719
```

In [12]:
```
1 X_train[X_train.bruises == 'Bruises'][X_train.Cap_surface== 'Smooth']
```

```
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.
  """Entry point for launching an IPython kernel.
```

Out[12]:

	cap_shape	Cap_surface	bruises	category
5435	Convex	Smooth	Bruises	Negative
1739	Flat	Smooth	Bruises	Negative
5499	Convex	Smooth	Bruises	Negative
5550	Convex	Smooth	Bruises	Negative
5462	Convex	Smooth	Bruises	Negative
...	...	...	...	...
5740	Convex	Smooth	Bruises	Negative
2241	Flat	Smooth	Bruises	Negative
844	Convex	Smooth	Bruises	Positive
984	Convex	Smooth	Bruises	Negative
5657	Flat	Smooth	Bruises	Negative

611 rows × 4 columns

Initial probabillity of dataset(class)

In [13]:
```
#--probability of positive and negative

Prob_positive = probability(X_train[X_train.category == 'Positive'].shape[0],X_train.shape[0],2)
Prob_negative = probability(X_train[X_train.category == 'Negative'].shape[0],X_train.shape[0],2)
print(Prob_positive)
print(Prob_negative)

```

```
0.5218002812939522
0.4781997187060478
```

In [ ]:
```

```

In [14]:
```
print(X_train.cap_shape.unique())#-----------getting unique values of each feature
print(X_train.Cap_surface.unique())
print(X_train.bruises.unique())
```

```
['Flat' 'Convex' 'Knobbed' 'Bell' 'Sunken' 'Conical']
['Scaly' 'Smooth' 'Fiberous' 'Grooves']
['Bruises' 'NoBruises']
```

In [15]:
```
#---------starting for feature 1 (cap_shape)
Prob_convex_pos = probability(X_train[X_train.cap_shape == 'Convex'][X_train.category == 'Positive'].shape[0],X_trai
Prob_convex_neg = probability(X_train[X_train.cap_shape == 'Convex'][X_train.category == 'Negative'].shape[0],X_trai


Prob_Conical_pos = probability(X_train[X_train.cap_shape == 'Conical'][X_train.category == 'Positive'].shape[0],X_tr
Prob_Conical_neg = probability(X_train[X_train.cap_shape == 'Conical'][X_train.category == 'Negative'].shape[0],X_tr


Prob_Bell_pos = probability(X_train[X_train.cap_shape == 'Bell'][X_train.category == 'Positive'].shape[0],X_train[y_
Prob_Bell_neg = probability(X_train[X_train.cap_shape == 'Bell'][X_train.category == 'Negative'].shape[0],X_train[y_

Prob_Flat_pos = probability(X_train[X_train.cap_shape == 'Flat'][X_train.category == 'Positive'].shape[0],X_train[y_
Prob_Flat_neg = probability(X_train[X_train.cap_shape == 'Flat'][X_train.category == 'Negative'].shape[0],X_train[y_


Prob_Knobbed_pos = probability(X_train[X_train.cap_shape == 'Knobbed'][X_train.category == 'Positive'].shape[0],X_tr
Prob_Knobbed_neg = probability(X_train[X_train.cap_shape == 'Knobbed'][X_train.category == 'Negative'].shape[0],X_tr

Prob_Sunken_pos = probability(X_train[X_train.cap_shape == 'Sunken'][X_train.category == 'Positive'].shape[0],X_trai
Prob_Sunken_neg = probability(X_train[X_train.cap_shape == 'Sunken'][X_train.category == 'Negative'].shape[0],X_trai

print(Prob_Sunken_pos)
print(Prob_Sunken_neg)
```

```
0.008409014463504878
0.0011009174311926607

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:6: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:7: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.
  import sys
```

```
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:10: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
  # Remove the CWD from sys.path while we load stuff.
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:11: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
  # This is added back by InteractiveShellApp.init_path()
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:13: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
  del sys.path[0]
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:14: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:17: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:18: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:20: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:21: UserWarning: Boolean
Series key will be reindexed to match DataFrame index.
```

In [16]:
```
#---------starting for feature 2 (cap_shape)
Prob_Fiberous_pos = probability(X_train[X_train.Cap_surface == 'Fiberous'][X_train.category == 'Positive'].shape[0],
Prob_Fiberous_neg = probability(X_train[X_train.Cap_surface == 'Fiberous'][X_train.category == 'Negative'].shape[0],


Prob_Smooth_pos = probability(X_train[X_train.Cap_surface == 'Smooth'][X_train.category == 'Positive'].shape[0],X_tr
Prob_Smooth_neg = probability(X_train[X_train.Cap_surface == 'Smooth'][X_train.category == 'Negative'].shape[0],X_tr


Prob_Scaly_pos = probability(X_train[X_train.Cap_surface == 'Scaly'][X_train.category == 'Positive'].shape[0],X_trai
Prob_Scaly_neg = probability(X_train[X_train.Cap_surface == 'Scaly'][X_train.category == 'Negative'].shape[0],X_trai

Prob_Grooves_pos = probability(X_train[X_train.Cap_surface == 'Grooves'][X_train.category == 'Positive'].shape[0],X_
Prob_Grooves_neg = probability(X_train[X_train.Cap_surface == 'Grooves'][X_train.category == 'Negative'].shape[0],X_



print(Prob_Fiberous_pos)
print(Prob_Fiberous_neg)
```

```
0.3779872096937058
0.1883951524054352

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Boolea
n Series key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Boolea
n Series key will be reindexed to match DataFrame index.
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:6: UserWarning: Boolea
n Series key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:7: UserWarning: Boolea
n Series key will be reindexed to match DataFrame index.
  import sys
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:10: UserWarning: Boole
an Series key will be reindexed to match DataFrame index.
  # Remove the CWD from sys.path while we load stuff.
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:11: UserWarning: Boole
an Series key will be reindexed to match DataFrame index.
```

```
  # This is added back by InteractiveShellApp.init_path()
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:13: UserWarning: Boole
an Series key will be reindexed to match DataFrame index.
  del sys.path[0]
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:14: UserWarning: Boole
an Series key will be reindexed to match DataFrame index.
```

In [17]:

```
#---------------for feature 3(bruises)
Prob_Bruises_pos = probability(X_train[X_train.bruises == 'Bruises'][X_train.category == 'Positive'].shape[0],X_trai
Prob_Bruises_neg = probability(X_train[X_train.bruises == 'Bruises'][X_train.category == 'Negative'].shape[0],X_trai


Prob_NoBruises_pos = probability(X_train[X_train.bruises == 'NoBruises'][X_train.category == 'Positive'].shape[0],X_
Prob_NoBruises_neg = probability(X_train[X_train.bruises == 'NoBruises'][X_train.category == 'Negative'].shape[0],X_

print(Prob_Bruises_pos)
```

```
c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:2: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.
  This is separate from the ipykernel package so we can avoid doing imports until

0.6611653755473224

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:6: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.

c:\users\alouk\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:7: UserWarning: Boolean S
eries key will be reindexed to match DataFrame index.
  import sys
```

In [18]:

```
X_train1 = X_test.to_numpy()#-------------x_train
X_train1 = np.delete(X_train1, np.s_[3], axis=1)
```

Predicting probability of combinations of instances

In [19]:

```
predicted = np.zeros((len(X_train1),1))
def prediction(a):
     ##----------testing
    for i in range(len(a)):
        if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_convex_pos * Prob_Scaly_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_convex_neg * Prob_Scaly_neg * Prob_NoBruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative


        if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Bell_pos * Prob_Scaly_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_Bell_neg * Prob_Scaly_neg * Prob_NoBruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative


        if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Flat_pos * Prob_Scaly_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_Flat_neg * Prob_Scaly_neg * Prob_NoBruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative

        if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Knobbed_pos * Prob_Scaly_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_Knobbed_neg * Prob_Scaly_neg * Prob_NoBruises_neg
```

```
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative

        if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Sunken_pos * Prob_Scaly_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_Sunken_neg * Prob_Scaly_neg * Prob_NoBruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative


        if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Conical_pos * Prob_Scaly_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_Conical_neg * Prob_Scaly_neg * Prob_NoBruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative




            #----cahnging fibrous



        if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_convex_pos * Prob_Fiberous_pos * Prob_NoBruises_pos
            negative_prob1 = Prob_convex_neg * Prob_Fiberous_neg * Prob_NoBruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative
```

```
84
85
86          if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
87              positive_prob1 = 0
88              negative_prob1 = 0
89              positive_prob1 = Prob_Bell_pos * Prob_Fiberous_pos * Prob_NoBruises_pos
90              negative_prob1 = Prob_Bell_neg * Prob_Fiberous_neg * Prob_NoBruises_neg
91              if(positive_prob1 >= negative_prob1):
92                  predicted[i] = 1 #---positive
93              else:
94                  predicted[i] = 0 #-----negative
95
96
97          if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
98              positive_prob1 = 0
99              negative_prob1 = 0
100             positive_prob1 = Prob_Flat_pos * Prob_Fiberous_pos * Prob_NoBruises_pos
101             negative_prob1 = Prob_Flat_neg * Prob_Fiberous_neg * Prob_NoBruises_neg
102             if(positive_prob1 >= negative_prob1):
103                 predicted[i] = 1 #---positive
104             else:
105                 predicted[i] = 0 #-----negative
106
107         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
108             positive_prob1 = 0
109             negative_prob1 = 0
110             positive_prob1 = Prob_Knobbed_pos * Prob_Fiberous_pos * Prob_NoBruises_pos
111             negative_prob1 = Prob_Knobbed_neg * Prob_Fiberous_neg * Prob_NoBruises_neg
112             if(positive_prob1 >= negative_prob1):
113                 predicted[i] = 1 #---positive
114             else:
115                 predicted[i] = 0 #-----negative
116
117         if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
118             positive_prob1 = 0
119             negative_prob1 = 0
120             positive_prob1 = Prob_Sunken_pos * Prob_Fiberous_pos * Prob_NoBruises_pos
121             negative_prob1 = Prob_Sunken_neg * Prob_Fiberous_neg * Prob_NoBruises_neg
122             if(positive_prob1 >= negative_prob1):
123                 predicted[i] = 1 #---positive
124             else:
125                 predicted[i] = 0 #-----negative
```

```
126
127
128         if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
129             positive_prob1 = 0
130             negative_prob1 = 0
131             positive_prob1 = Prob_Conical_pos * Prob_Fiberous_pos * Prob_NoBruises_pos
132             negative_prob1 = Prob_Conical_neg * Prob_Fiberous_neg * Prob_NoBruises_neg
133             if(positive_prob1 >= negative_prob1):
134                 predicted[i] = 1 #---positive
135             else:
136                 predicted[i] = 0 #-----negative
137
138
139
140
141
142
143
144
145          #----cahnging smooth
146
147
148
149         if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'NoBruises'):
150             positive_prob1 = 0
151             negative_prob1 = 0
152             positive_prob1 = Prob_convex_pos * Prob_Smooth_pos * Prob_NoBruises_pos
153             negative_prob1 = Prob_convex_neg * Prob_Smooth_neg * Prob_NoBruises_neg
154             if(positive_prob1 >= negative_prob1):
155                 predicted[i] = 1 #---positive
156             else:
157                 predicted[i] = 0 #-----negative
158
159
160         if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'NoBruises'):
161             positive_prob1 = 0
162             negative_prob1 = 0
163             positive_prob1 = Prob_Bell_pos * Prob_Smooth_pos * Prob_NoBruises_pos
164             negative_prob1 = Prob_Bell_neg * Prob_Smooth_neg * Prob_NoBruises_neg
165             if(positive_prob1 >= negative_prob1):
166                 predicted[i] = 1 #---positive
167             else:
```

```
168                 predicted[i] = 0 #-----negative
169 
170 
171         if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'NoBruises'):
172             positive_prob1 = 0
173             negative_prob1 = 0
174             positive_prob1 = Prob_Flat_pos * Prob_Smooth_pos * Prob_NoBruises_pos
175             negative_prob1 = Prob_Flat_neg * Prob_Smooth_neg * Prob_NoBruises_neg
176             if(positive_prob1 >= negative_prob1):
177                 predicted[i] = 1 #---positive
178             else:
179                 predicted[i] = 0 #-----negative
180 
181         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'NoBruises'):
182             positive_prob1 = 0
183             negative_prob1 = 0
184             positive_prob1 = Prob_Knobbed_pos * Prob_Smooth_pos * Prob_NoBruises_pos
185             negative_prob1 = Prob_Knobbed_neg * Prob_Smooth_neg * Prob_NoBruises_neg
186             if(positive_prob1 >= negative_prob1):
187                 predicted[i] = 1 #---positive
188             else:
189                 predicted[i] = 0 #-----negative
190 
191         if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'NoBruises'):
192             positive_prob1 = 0
193             negative_prob1 = 0
194             positive_prob1 = Prob_Sunken_pos * Prob_Smooth_pos * Prob_NoBruises_pos
195             negative_prob1 = Prob_Sunken_neg * Prob_Smooth_neg * Prob_NoBruises_neg
196             if(positive_prob1 >= negative_prob1):
197                 predicted[i] = 1 #---positive
198             else:
199                 predicted[i] = 0 #-----negative
200 
201 
202         if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'NoBruises'):
203             positive_prob1 = 0
204             negative_prob1 = 0
205             positive_prob1 = Prob_Conical_pos * Prob_Smooth_pos * Prob_NoBruises_pos
206             negative_prob1 = Prob_Conical_neg * Prob_Smooth_neg * Prob_NoBruises_neg
207             if(positive_prob1 >= negative_prob1):
208                 predicted[i] = 1 #---positive
209             else:
```

```
            predicted[i] = 0 #-----negative
```

```
    #----cahnging grooves
```

```
    if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'NoBruises'):
        positive_prob1 = 0
        negative_prob1 = 0
        positive_prob1 = Prob_convex_pos * Prob_Grooves_pos * Prob_NoBruises_pos
        negative_prob1 = Prob_convex_neg * Prob_Grooves_neg * Prob_NoBruises_neg
        if(positive_prob1 >= negative_prob1):
            predicted[i] = 1 #---positive
        else:
            predicted[i] = 0 #-----negative

    if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'NoBruises'):
        positive_prob1 = 0
        negative_prob1 = 0
        positive_prob1 = Prob_Bell_pos * Prob_Grooves_pos * Prob_NoBruises_pos
        negative_prob1 = Prob_Bell_neg * Prob_Grooves_neg * Prob_NoBruises_neg
        if(positive_prob1 >= negative_prob1):
            predicted[i] = 1 #---positive
        else:
            predicted[i] = 0 #-----negative

    if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'NoBruises'):
        positive_prob1 = 0
        negative_prob1 = 0
        positive_prob1 = Prob_Flat_pos * Prob_Grooves_pos * Prob_NoBruises_pos
        negative_prob1 = Prob_Flat_neg * Prob_Grooves_neg * Prob_NoBruises_neg
        if(positive_prob1 >= negative_prob1):
            predicted[i] = 1 #---positive
        else:
            predicted[i] = 0 #-----negative
```

```
252         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'NoBruises'):
253             positive_prob1 = 0
254             negative_prob1 = 0
255             positive_prob1 = Prob_Knobbed_pos * Prob_Grooves_pos * Prob_NoBruises_pos
256             negative_prob1 = Prob_Knobbed_neg * Prob_Grooves_neg * Prob_NoBruises_neg
257             if(positive_prob1 >= negative_prob1):
258                 predicted[i] = 1 #---positive
259             else:
260                 predicted[i] = 0 #-----negative
261 
262         if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'NoBruises'):
263             positive_prob1 = 0
264             negative_prob1 = 0
265             positive_prob1 = Prob_Sunken_pos * Prob_Grooves_pos * Prob_NoBruises_pos
266             negative_prob1 = Prob_Sunken_neg * Prob_Grooves_neg * Prob_NoBruises_neg
267             if(positive_prob1 >= negative_prob1):
268                 predicted[i] = 1 #---positive
269             else:
270                 predicted[i] = 0 #-----negative
271 
272 
273         if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'NoBruises'):
274             positive_prob1 = 0
275             negative_prob1 = 0
276             positive_prob1 = Prob_Conical_pos * Prob_Grooves_pos * Prob_NoBruises_pos
277             negative_prob1 = Prob_Conical_neg * Prob_Grooves_neg * Prob_NoBruises_neg
278             if(positive_prob1 >= negative_prob1):
279                 predicted[i] = 1 #---positive
280             else:
281                 predicted[i] = 0 #-----negative
282 
283 
284             #########
285             #######
286             #########
287             ##########
288             ##########
289             ##########
290             ##############
291             ############3
292 
293 
```

```
294
295
296
297
298
299         if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'Bruises'):
300             positive_prob1 = 0
301             negative_prob1 = 0
302             positive_prob1 = Prob_convex_pos * Prob_Scaly_pos * Prob_Bruises_pos
303             negative_prob1 = Prob_convex_neg * Prob_Scaly_neg * Prob_Bruises_neg
304             if(positive_prob1 >= negative_prob1):
305                 predicted[i] = 1 #---positive
306             else:
307                 predicted[i] = 0 #-----negative
308
309
310         if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'Bruises'):
311             positive_prob1 = 0
312             negative_prob1 = 0
313             positive_prob1 = Prob_Bell_pos * Prob_Scaly_pos * Prob_Bruises_pos
314             negative_prob1 = Prob_Bell_neg * Prob_Scaly_neg * Prob_Bruises_neg
315             if(positive_prob1 >= negative_prob1):
316                 predicted[i] = 1 #---positive
317             else:
318                 predicted[i] = 0 #-----negative
319
320
321         if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'Bruises'):
322             positive_prob1 = 0
323             negative_prob1 = 0
324             positive_prob1 = Prob_Flat_pos * Prob_Scaly_pos * Prob_Bruises_pos
325             negative_prob1 = Prob_Flat_neg * Prob_Scaly_neg * Prob_Bruises_neg
326             if(positive_prob1 >= negative_prob1):
327                 predicted[i] = 1 #---positive
328             else:
329                 predicted[i] = 0 #-----negative
330
331         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'Bruises'):
332             positive_prob1 = 0
333             negative_prob1 = 0
334             positive_prob1 = Prob_Knobbed_pos * Prob_Scaly_pos * Prob_Bruises_pos
335             negative_prob1 = Prob_Knobbed_neg * Prob_Scaly_neg * Prob_Bruises_neg
```

```
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative

        if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'Bruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Sunken_pos * Prob_Scaly_pos * Prob_Bruises_pos
            negative_prob1 = Prob_Sunken_neg * Prob_Scaly_neg * Prob_Bruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative


        if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Scaly' and X_train1[i][2]== 'Bruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_Conical_pos * Prob_Scaly_pos * Prob_Bruises_pos
            negative_prob1 = Prob_Conical_neg * Prob_Scaly_neg * Prob_Bruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative




                #----cahnging fibrous



        if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'NoBruises'):
            positive_prob1 = 0
            negative_prob1 = 0
            positive_prob1 = Prob_convex_pos * Prob_Fiberous_pos * Prob_Bruises_pos
            negative_prob1 = Prob_convex_neg * Prob_Fiberous_neg * Prob_Bruises_neg
            if(positive_prob1 >= negative_prob1):
                predicted[i] = 1 #---positive
            else:
                predicted[i] = 0 #-----negative
```

```
378 
379 
380         if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'Bruises'):
381             positive_prob1 = 0
382             negative_prob1 = 0
383             positive_prob1 = Prob_Bell_pos * Prob_Fiberous_pos * Prob_Bruises_pos
384             negative_prob1 = Prob_Bell_neg * Prob_Fiberous_neg * Prob_Bruises_neg
385             if(positive_prob1 >= negative_prob1):
386                 predicted[i] = 1 #---positive
387             else:
388                 predicted[i] = 0 #-----negative
389 
390 
391         if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'Bruises'):
392             positive_prob1 = 0
393             negative_prob1 = 0
394             positive_prob1 = Prob_Flat_pos * Prob_Fiberous_pos * Prob_Bruises_pos
395             negative_prob1 = Prob_Flat_neg * Prob_Fiberous_neg * Prob_Bruises_neg
396             if(positive_prob1 >= negative_prob1):
397                 predicted[i] = 1 #---positive
398             else:
399                 predicted[i] = 0 #-----negative
400 
401         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'Bruises'):
402             positive_prob1 = 0
403             negative_prob1 = 0
404             positive_prob1 = Prob_Knobbed_pos * Prob_Fiberous_pos * Prob_Bruises_pos
405             negative_prob1 = Prob_Knobbed_neg * Prob_Fiberous_neg * Prob_Bruises_neg
406             if(positive_prob1 >= negative_prob1):
407                 predicted[i] = 1 #---positive
408             else:
409                 predicted[i] = 0 #-----negative
410 
411         if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'Bruises'):
412             positive_prob1 = 0
413             negative_prob1 = 0
414             positive_prob1 = Prob_Sunken_pos * Prob_Fiberous_pos * Prob_Bruises_pos
415             negative_prob1 = Prob_Sunken_neg * Prob_Fiberous_neg * Prob_Bruises_neg
416             if(positive_prob1 >= negative_prob1):
417                 predicted[i] = 1 #---positive
418             else:
419                 predicted[i] = 0 #-----negative
```

```
420 
421 
422         if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Fiberous' and X_train1[i][2]== 'Bruises'):
423             positive_prob1 = 0
424             negative_prob1 = 0
425             positive_prob1 = Prob_Conical_pos * Prob_Fiberous_pos * Prob_Bruises_pos
426             negative_prob1 = Prob_Conical_neg * Prob_Fiberous_neg * Prob_Bruises_neg
427             if(positive_prob1 >= negative_prob1):
428                 predicted[i] = 1 #---positive
429             else:
430                 predicted[i] = 0 #-----negative
431 
432 
433 
434 
435 
436 
437 
438 
439          #----cahnging smooth
440 
441 
442 
443         if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'Bruises'):
444             positive_prob1 = 0
445             negative_prob1 = 0
446             positive_prob1 = Prob_convex_pos * Prob_Smooth_pos * Prob_Bruises_pos
447             negative_prob1 = Prob_convex_neg * Prob_Smooth_neg * Prob_Bruises_neg
448             if(positive_prob1 >= negative_prob1):
449                 predicted[i] = 1 #---positive
450             else:
451                 predicted[i] = 0 #-----negative
452 
453 
454         if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'Bruises'):
455             positive_prob1 = 0
456             negative_prob1 = 0
457             positive_prob1 = Prob_Bell_pos * Prob_Smooth_pos * Prob_Bruises_pos
458             negative_prob1 = Prob_Bell_neg * Prob_Smooth_neg * Prob_Bruises_neg
459             if(positive_prob1 >= negative_prob1):
460                 predicted[i] = 1 #---positive
461             else:
```

```
462                 predicted[i] = 0 #-----negative
463 
464 
465         if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'Bruises'):
466             positive_prob1 = 0
467             negative_prob1 = 0
468             positive_prob1 = Prob_Flat_pos * Prob_Smooth_pos * Prob_Bruises_pos
469             negative_prob1 = Prob_Flat_neg * Prob_Smooth_neg * Prob_Bruises_neg
470             if(positive_prob1 >= negative_prob1):
471                 predicted[i] = 1 #---positive
472             else:
473                 predicted[i] = 0 #-----negative
474 
475         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'Bruises'):
476             positive_prob1 = 0
477             negative_prob1 = 0
478             positive_prob1 = Prob_Knobbed_pos * Prob_Smooth_pos * Prob_Bruises_pos
479             negative_prob1 = Prob_Knobbed_neg * Prob_Smooth_neg * Prob_Bruises_neg
480             if(positive_prob1 >= negative_prob1):
481                 predicted[i] = 1 #---positive
482             else:
483                 predicted[i] = 0 #-----negative
484 
485         if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'Bruises'):
486             positive_prob1 = 0
487             negative_prob1 = 0
488             positive_prob1 = Prob_Sunken_pos * Prob_Smooth_pos * Prob_Bruises_pos
489             negative_prob1 = Prob_Sunken_neg * Prob_Smooth_neg * Prob_Bruises_neg
490             if(positive_prob1 >= negative_prob1):
491                 predicted[i] = 1 #---positive
492             else:
493                 predicted[i] = 0 #-----negative
494 
495 
496         if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Smooth' and X_train1[i][2]== 'Bruises'):
497             positive_prob1 = 0
498             negative_prob1 = 0
499             positive_prob1 = Prob_Conical_pos * Prob_Smooth_pos * Prob_Bruises_pos
500             negative_prob1 = Prob_Conical_neg * Prob_Smooth_neg * Prob_Bruises_neg
501             if(positive_prob1 >= negative_prob1):
502                 predicted[i] = 1 #---positive
503             else:
```

```
504                 predicted[i] = 0 #-----negative
505 
506 
507 
508 
509 
510         #----cahnging grooves
511 
512 
513 
514         if(X_train1[i][0] == 'Convex' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'Bruises'):
515             positive_prob1 = 0
516             negative_prob1 = 0
517             positive_prob1 = Prob_convex_pos * Prob_Grooves_pos * Prob_Bruises_pos
518             negative_prob1 = Prob_convex_neg * Prob_Grooves_neg * Prob_Bruises_neg
519             if(positive_prob1 >= negative_prob1):
520                 predicted[i] = 1 #---positive
521             else:
522                 predicted[i] = 0 #-----negative
523 
524 
525         if(X_train1[i][0] == 'Bell' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'Bruises'):
526             positive_prob1 = 0
527             negative_prob1 = 0
528             positive_prob1 = Prob_Bell_pos * Prob_Grooves_pos * Prob_Bruises_pos
529             negative_prob1 = Prob_Bell_neg * Prob_Grooves_neg * Prob_Bruises_neg
530             if(positive_prob1 >= negative_prob1):
531                 predicted[i] = 1 #---positive
532             else:
533                 predicted[i] = 0 #-----negative
534 
535 
536         if(X_train1[i][0] == 'Flat' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'Bruises'):
537             positive_prob1 = 0
538             negative_prob1 = 0
539             positive_prob1 = Prob_Flat_pos * Prob_Grooves_pos * Prob_Bruises_pos
540             negative_prob1 = Prob_Flat_neg * Prob_Grooves_neg * Prob_Bruises_neg
541             if(positive_prob1 >= negative_prob1):
542                 predicted[i] = 1 #---positive
543             else:
544                 predicted[i] = 0 #-----negative
545 
```

```
546         if(X_train1[i][0] == 'Knobbed' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'Bruises'):
547             positive_prob1 = 0
548             negative_prob1 = 0
549             positive_prob1 = Prob_Knobbed_pos * Prob_Grooves_pos * Prob_Bruises_pos
550             negative_prob1 = Prob_Knobbed_neg * Prob_Grooves_neg * Prob_Bruises_neg
551             if(positive_prob1 >= negative_prob1):
552                 predicted[i] = 1 #---positive
553             else:
554                 predicted[i] = 0 #-----negative
555 
556         if(X_train1[i][0] == 'Sunken' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'Bruises'):
557             positive_prob1 = 0
558             negative_prob1 = 0
559             positive_prob1 = Prob_Sunken_pos * Prob_Grooves_pos * Prob_Bruises_pos
560             negative_prob1 = Prob_Sunken_neg * Prob_Grooves_neg * Prob_Bruises_neg
561             if(positive_prob1 >= negative_prob1):
562                 predicted[i] = 1 #---positive
563             else:
564                 predicted[i] = 0 #-----negative
565 
566 
567         if(X_train1[i][0] == 'Conical' and X_train1[i][1] == 'Grooves' and X_train1[i][2]== 'Bruises'):
568             positive_prob1 = 0
569             negative_prob1 = 0
570             positive_prob1 = Prob_Conical_pos * Prob_Grooves_pos * Prob_Bruises_pos
571             negative_prob1 = Prob_Conical_neg * Prob_Grooves_neg * Prob_Bruises_neg
572             if(positive_prob1 >= negative_prob1):
573                 predicted[i] = 1 #---positive
574             else:
575                 predicted[i] = 0 #-----negative
576 
577 
578 
579 
580 
581 
582 
583 
584 
585 
586 
587 
```

```
588
```

In [20]:

```
predicted = np.zeros((len(X_test),1))##----------testing
prediction(X_test)#------------calling the main function


y_test_acc = y_test.replace({"Negative": 0, "Positive" : 1}, inplace = True)
y_test = y_test.to_numpy()


```

Finding Values for confusion matrix

In [21]:
```
L = []
for i in range(len(y_test)):
    if(y_test[i] == 1 and predicted[i] == 1):#------------TRUE POSITIVE VALUE
        L.append(y_test[i])
TP=len(L)
print(TP)


L1 = []
for i in range(len(y_test)):
    if(y_test[i] == 0 and predicted[i] == 0):#------------FALSE POSITIVE VALUE
        L1.append(y_test[i])
FP=len(L1)
print(FP)



L2 = []
for i in range(len(y_test)):
    if(y_test[i] != 1 and predicted[i] == 1):#------------TRUE NEGATIVE VALUE
        L2.append(y_test[i])
TN=len(L2)
print(TN)


L3 = []
for i in range(len(y_test)):
    if(y_test[i] == 0 and predicted[i] != 0):#------------FALSE NEGATIVE VALUE
        L3.append(y_test[i])
FN=len(L2)
print(FN)

```

```
800
863
334
334
```

The final Accuracy, Sensitivity and Specificity

In [22]:
```
accuracy(y_test,predicted)#---------------finding the accuracy
Sensitivity()
Specificity()

```

```
Accuracy is 68.21164889253485
Sensitivity is 70.54673721340387
Specificity is 27.903091060985798
```

In [23]:
```
predicted_values = pd.DataFrame({'Predicted': predicted[:, 0]})
predicted_values.replace({0.0: 'Negative', 1.0 : 'Positive'}, inplace = True)
```

Creating modified database which contains new coloumn "predicted_values"

In [24]:
```
X_test = X_test.assign(predicted_values=predicted_values.values)
```

In [25]:
```
1 X_test
```

Out[25]:

	cap_shape	Cap_surface	bruises	category	predicted_values
7099	Knobbed	Scaly	NoBruises	Negative	Negative
6901	Knobbed	Smooth	NoBruises	Negative	Negative
5004	Flat	Scaly	NoBruises	Negative	Negative
662	Convex	Scaly	Bruises	Negative	Positive
2229	Convex	Fiberous	Bruises	Positive	Negative
...	...	...	...	...	...
2917	Flat	Fiberous	Bruises	Positive	Positive
7313	Knobbed	Scaly	NoBruises	Negative	Negative
3820	Flat	Scaly	Bruises	Positive	Positive
4749	Convex	Fiberous	NoBruises	Negative	Positive
2913	Convex	Scaly	Bruises	Positive	Positive

2438 rows × 5 columns

Exporting database to csv file

In [26]:
```
1 export_csv = X_test.to_csv (r'W:\Lakehead Study material\Big data\Assignment 2\Naive bayes\Work directory\prediction
```

In [27]:
```
1 accuracy(y_test,predicted)#---------------finding the accuracy
2 Sensitivity()
3 Specificity()
```

```
Accuracy is 68.21164889253485
Sensitivity is 70.54673721340387
Specificity is 27.903091060985798
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```