ÖZYEĞİN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

CS 454

2021 Fall

Homework1

Parametric Classification

By
Tuğcan Hoşer
S015561

Supervised By
Ethem Alpaydın

Part A

1) First of all, the data in the training.csv file in the project was separated according to their classes. There are 3 classes and each class has 50 data.

Result :

```
Number_Of_Class_1 = 50
Number_Of_Class_2 = 50
Number_Of_Class_3 = 50
```

2) The prior of each class was found.

Result :

```
Priors_Of_Class_1 = 0.3333333333333333
Priors_Of_Class_2 = 0.3333333333333333
Priors_Of_Class_3 = 0.3333333333333333
```

Priors Formula :

$$\frac{Number\ of\ Class1}{Number\ of\ Instances} = P(\ C = 1)$$

$$\frac{Number\ of\ Class2}{Number\ of\ Instances} = P(\ C = 2)$$

$$\frac{Number\ of\ Class3}{Number\ of\ Instances} = P(\ C = 3)$$

3) Mean and Standard Deviation were found

Results of Mean :

```
Mean_of_Class_1 = 24.48
Mean_of_Class_2 = 34.12
Mean_of_Class_3 = 49.44
```

Mean Formula :

$$\frac{\sum_{i=1}^{n} x_i}{n} = \mu$$

$$\Sigma x_i \rightarrow x_1 + x_2 + x_{3+\ldots} x_n$$

$$n \rightarrow Total\ number\ of\ elements\ in\ each\ class$$

$$\mu \rightarrow Mean$$

Results of Standard Deviation :

```
Standard_Deviation_Of_Class_1 = 1.992385504865963
Standard_Deviation_Of_Class_2 = 4.135891681366909 5
Standard_Deviation_Of_Class_3 = 5.091797325110258
```

Standard Deviation Formula :

$$\sigma = \sqrt{\frac{\Sigma(x-\mu)^2}{n}}$$

4) Likelihoods values were found with Gaussian distribution. Previously found mean and standard deviation were used. As input, all numbers between the smallest value and the largest value of the ages in the training.csv file were tried.

Gaussian Distribution Formula :

$$y = \frac{1}{\sigma\sqrt{2\pi}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

5) Posterior were found along with likelihoods and priors values.

For class 1 :

$$P(c=1\,|x=n) = \frac{P(x=n\,|\,c=1)*\ P(c=1)}{P(x=n\,|\,c=1)*P(c=1)+\ P(x=n\,|\,c=2)*P(c=2)+\ P(x=n\,|\,c=3)*P(c=3)}$$

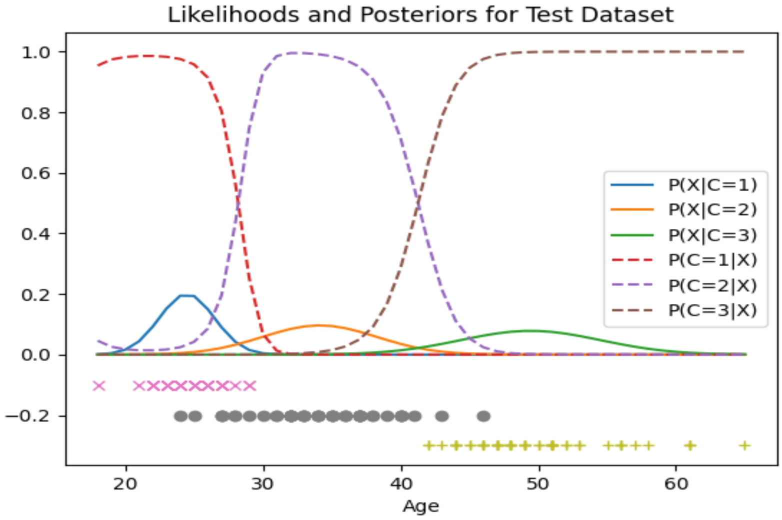
For class 2:

$$P(c=2\,|x=n) = \frac{P(x=n\,|\,c=2)*\ P(c=2)}{P(x=n\,|\,c=1)*P(c=1)+\ P(x=n\,|\,c=2)*P(c=2)+\ P(x=n\,|\,c=3)*P(c=3)}$$

For class 3:

$$P(c=3\,|x=n) = \frac{P(x=n\,|\,c=3)*\ P(c=3)}{P(x=n\,|\,c=1)*P(c=1)+\ P(x=n\,|\,c=2)*P(c=2)+\ P(x=n\,|\,c=3)*P(c=3)}$$

6) After finding all these values Likelihoods, Posteriors and, Instances are shown using matplotlib.



We assumed that the distributions of the samples were Gaussian. This distribution seems to fit when we draw the examples with the likelihood functions.

PART B

0 -1 Loss Training Part

0/1 loss training chart is drawn. Misclassifications have been identified. Shown in the 3 x 3 confusion matrix chart. Both the training set and the testing set were used.

For Training Set

```
-------------
C1_C1 = 49
C2_C1 = 1
C3_C1 = 0
-------------
C1_C2 = 5
C2_C2 = 42
C3_C2 = 3
-------------
C1_C3 = 0
C2_C3 = 2
C3_C3 = 48
--------------
```

Training Set		Actual		
		C1	C2	C3
Action	C1	49	5	0
	C2	1	42	2
	C3	0	3	48

For Test Set

```
------------
T1_T1 = 48
T2_T1 = 2
T3_T1 = 0
------------
T1_T2 = 7
T2_T2 = 41
T3_T2 = 2
------------
T1_T3 = 0
T2_T3 = 0
T3_T3 = 50
-------------
```

Test Set		Actual		
		T1	T2	T3
Action	T1	48	7	0
	T2	2	41	0
	T3	0	2	50

We see that the model makes misclassifications in both the test set and the training set. This is because of the outlier. A dataset, unlike other data, may contain outliers that are outside the expected range. These are called outliers.

Rejection and Minimum expected Risk Part

The cost (loss) of assigning an input to an incorrect class was determined of 4 and the cost of rejecting a sample was 1, the decision threshold for minimum expected risk was calculated. It was also shown in the 3 x 4 confusion matrix.

If there is a high probability of making a wrong decision, it can be said that I will refuse, I do not decide.
If rejections are less costly than misclassifications. Then rejection can be made.

Loss Table		Truth		
		C1	C2	C3
Action	α1	0	4	4
	α2	4	0	4
	α3	4	4	0
	REJECT	1	1	1

To set the threshold value formula

$$R(\alpha_1|x) = loss_{c_1\alpha_1} * P(c_1|x) + loss_{c_2\alpha_1} * P(c_2|x) + loss_{c_3\alpha_1} * P(c_3|x)$$

$$R(\alpha_2|x) = loss_{c_1\alpha_2} * P(c_1|x) + loss_{c_2\alpha_2} * P(c_2|x) + loss_{c_3\alpha_2} * P(c_3|x)$$

$$R(\alpha_3|x) = loss_{c_1\alpha_3} * P(c_1|x) + loss_{c_2\alpha_3} * P(c_2|x) + loss_{c_3\alpha_3} * P(c_3|x)$$

Result :

$$R(\alpha_1|x) = 0 * P(c_1|x) + 4 * P(c_2|x) + 4 * P(c_3|x) = 4\left(P(c_2|x) + P(c_3|x)\right) = 4\,(1 - P(c_1|x))$$

$$R(\alpha_2|x) = 4 * P(c_1|x) + 0 * P(c_2|x) + 4 * P(c_3|x) = 4\left(P(c_1|x) + P(c_3|x)\right) = 4\,(1 - P(c_2|x))$$

$$R(\alpha_3|x) = 4 * P(c_1|x) + 4 * P(c_2|x) + 0 * P(c_3|x) = 4\left(P(c_1|x) + P(c_2|x)\right) = 4\,(1 - P(c_3|x))$$

IF Choose Action $\alpha_1|c_1$

$$R(\alpha_1|x) < R(\alpha_2|x) \;\&\; R(\alpha_1|x) < R(\alpha_3|x) \;\&\; R(\alpha_1|x) < Reject$$

1) $4\left(1 - P(c_1|x)\right) < 4\,(1 - P(c_2|x))$

$P(c_1|x) > P(c_2|x)$

2) $4\left(1 - P(c_1|x)\right) < 4\,(1 - P(c_2|x))$

$P(c_1|x) > P(c_3|x)$

3) $4\left(1 - P(c_1|x)\right) < 1$

$P(c_1|x) > 3/4$

Training Set for rejection

```
------------
P1_P1 = 47
P2_P1 = 0
P3_P1 = 0
REJECT_P1 = 3
------------
P1_P2 = 5
P2_P2 = 41
P3_P2 = 1
REJECT_P2 = 3
------------
P1_P3 = 0
P2_P3 = 2
P3_P3 = 47
REJECT_P3 = 1
-------------
```

Training Set		Actual		
		P1	P2	P3
Action	P1	47	5	0
	P2	0	41	2
	P3	0	1	47
	REJECT	3	3	1

Testing Set for Rejection

```
------------
Q1_Q1 = 47
Q2_Q1 = 0
Q3_Q1 = 0
REJECT_Q1 = 3
------------
Q1_Q2 = 5
Q2_Q2 = 35
Q3_Q2 = 2
REJECT_Q2 = 8
------------
Q1_Q3 = 0
Q2_Q3 = 0
Q3_Q3 = 48
REJECT_Q3 = 2
-------------
```

Test Set		Actual		
		P1	P2	P3
Action	P1	47	5	0
	P2	0	35	0
	P3	0	2	48
	REJECT	3	8	2

As you can see in the tables, there are some rejections. This is because if the tendency to make mistakes is high, and the cost value of rejection is lower than misclassifications, rejection can be selected.

CODING PART

```
import pandas as pd
import math
from matplotlib import pyplot as plt

training_file = pd.read_csv("training.csv")
testing_file = pd.read_csv("testing.csv")

## we have 3 classes these are 1, 2, 3
## Priors = Number of Class / Number of Instances

Number_Of_1 = 0
Number_Of_2 = 0
Number_Of_3 = 0

for classType in training_file['class']:
    if classType == 1:
        Number_Of_1 += 1
    elif classType == 2:
        Number_Of_2 += 1
    else:
        Number_Of_3 += 1


print("")
print("Number_Of_Class_1 = " + str(Number_Of_1) )
print("Number_Of_Class_2 = " + str(Number_Of_1) )
print("Number_Of_Class_3 = " + str(Number_Of_1) )

Total_Number_Of_Data = Number_Of_1 + Number_Of_2 + Number_Of_3

Priors_Of_1 = Number_Of_1 / Total_Number_Of_Data
Priors_Of_2 = Number_Of_2 / Total_Number_Of_Data
Priors_Of_3 = Number_Of_3 / Total_Number_Of_Data

print("Priors_Of_Class_1 = " + str(Priors_Of_1) )
print("Priors_Of_Class_2 = " + str(Priors_Of_2) )
print("Priors_Of_Class_3 = " + str(Priors_Of_3) )


## Find Mean of Each Classes

def Find_Total_Number_Of_Age_Of_Class(n):
    count = 0
    for index, row in training_file.iterrows():
        if row['class'] == n:
            count += row['age']
    return count


Find_Total_Number_Of_Age_Of_Class(1)  # 1224
Find_Total_Number_Of_Age_Of_Class(2)  # 1706
Find_Total_Number_Of_Age_Of_Class(3)  # 2472

Mean_of_Class_1 = Find_Total_Number_Of_Age_Of_Class(1) / Number_Of_1
Mean_of_Class_2 = Find_Total_Number_Of_Age_Of_Class(2) / Number_Of_2
Mean_of_Class_3 = Find_Total_Number_Of_Age_Of_Class(3) / Number_Of_3


print("Mean_of_Class_1 = " + str(Mean_of_Class_1 ) )
print("Mean_of_Class_2 = " + str(Mean_of_Class_2 ) )
print("Mean_of_Class_3 = " + str(Mean_of_Class_3 ) )
```

```python
## Find Standard Deviation of Each Classes

def Standard_Deviation_Of_Class(n):
    count = 0
    if (n == 1):
        mean_temp = Mean_of_Class_1
        number_temp = Number_Of_1
    elif (n == 2):
        mean_temp = Mean_of_Class_2
        number_temp = Number_Of_2
    elif (n == 3):
        mean_temp = Mean_of_Class_3
        number_temp = Number_Of_3


    for index, row in training_file.iterrows():
        if row['class'] == n:
            temp = row['age'] - mean_temp
            count += math.pow(temp, 2)
    count = count / number_temp
    count = math.pow(count, 0.5)

    return count


print("Standard_Deviation_Of_Class_1 = " + str(Standard_Deviation_Of_Class(1) ) )
print("Standard_Deviation_Of_Class_2 = " + str(Standard_Deviation_Of_Class(2) ) )
print("Standard_Deviation_Of_Class_3 = " + str(Standard_Deviation_Of_Class(3) ) )

p1 = []
p2 = []
p3 = []


## Likelihoods for the first instance of the dataset

def find_likelihoods_of_data(n, class_index):
    likelihoods_for_class = 0
    # For class_1
    if class_index == 1:
        likelihoods_for_class = (1 / (Standard_Deviation_Of_Class(1)
         * math.pow((2 * math.pi), 0.5))) \
          * math.pow(math.e, (-0.5 * (math.pow(((n - Mean_of_Class_1) /
            Standard_Deviation_Of_Class(1)), 2))))
         return likelihoods_for_class

    # For class_2
    elif class_index == 2:
        likelihoods_for_class = (1 / (Standard_Deviation_Of_Class(2)
         * math.pow((2 * math.pi), 0.5))) \
            * math.pow(math.e, ( -0.5 * (math.pow(((n - Mean_of_Class_2) /
              Standard_Deviation_Of_Class(2)), 2))))
         return likelihoods_for_class

    # For class_3
    else:
        likelihoods_for_class = (1 / (Standard_Deviation_Of_Class(3)
          * math.pow((2 * math.pi), 0.5))) \
            * math.pow(math.e, (-0.5 * (math.pow(((n - Mean_of_Class_3) /
               Standard_Deviation_Of_Class(3)), 2))))
        return likelihoods_for_class

x1 =  [ N for N in range(min(testing_file['age']), max(testing_file['age'])+1)]


for N in range(min(testing_file['age']), max(testing_file['age'])+1) :
    p1.append(find_likelihoods_of_data(N,1))
    p2.append(find_likelihoods_of_data(N,2))
    p3.append(find_likelihoods_of_data(N,3))
```

```python
def find_posteriors_of_data(n, class_type):
        if class_type == 1:
            x = ((find_likelihoods_of_data(n, 1) * Priors_Of_1) /
              (((find_likelihoods_of_data(n, 1) * Priors_Of_1))
               + (find_likelihoods_of_data(n, 2) * Priors_Of_2)
                   + (findlikelihoods_of_data(n, 3) * Priors_Of_3)))
            return x

        elif class_type == 2:
            x = ((find_likelihoods_of_data(n, 2) * Priors_Of_2) /
              (((find_likelihoods_of_data(n, 1) * Priors_Of_1))
               + (find_likelihoods_of_data(n, 2) * Priors_Of_2)
                  + (find_likelihoods_of_data(n, 3) * Priors_Of_3)))
            return x

        else:
            x = ((find_likelihoods_of_data(n, 3) * Priors_Of_3) /
              (((find_likelihoods_of_data(n, 1) * Priors_Of_1))
               + (find_likelihoods_of_data(n, 2) * Priors_Of_2)
                  + (find_likelihoods_of_data(n, 3) * Priors_Of_3)))
            return x

t1 = []
t2 = []
t3 = []


x2 = [N for N in range(min(testing_file['age']), max(testing_file['age']) + 1)]

for N in range(min(testing_file['age']), max(testing_file['age']) + 1):
         t1.append(find_posteriors_of_data(N, 1))
         t2.append(find_posteriors_of_data(N, 2))
         t3.append(find_posteriors_of_data(N, 3))

plt.plot( x1, p1 )
plt.plot( x1, p2 )
plt.plot( x1, p3 )
plt.plot( x2, t1 , '--')
plt.plot( x2, t2 , '--')
plt.plot( x2, t3 , '--')
class_y = [-0.1,-0.2,-0.3]
plt.plot( testing_file[testing_file['class']==1]['age'] , [-0.1 for N in range (50)] , 'x')
plt.plot( testing_file[testing_file['class']==2]['age'], [-0.2 for N in range (50)] , 'o')
plt.plot(testing_file[testing_file['class']==3]['age'] ,[-0.3 for N in range (50)], '+' )
plt.legend(['P(X|C=1)','P(X|C=2)','P(X|C=3)','P(C=1|X)','P(C=2|X)','P(C=3|X)'])
plt.title('Likelihoods and Posteriors for Test Dataset')
plt.xlabel('Age')
plt.show()


### part b

### 0 - 1 Loss Training

C1_C1 = 0
C2_C1 = 0
C3_C1 = 0

C1_C2 = 0
C2_C2 = 0
C3_C2 = 0

C1_C3 = 0
C2_C3 = 0
C3_C3 = 0
```

```python
for index, row in training_file.iterrows():
    if row['class'] == 1:
        if((find_posteriors_of_data(row['age'],1) > find_posteriors_of_data(row['age'],2)) and
           (find_posteriors_of_data(row['age'],1)> find_posteriors_of_data(row['age'],3))):
             C1_C1 = C1_C1 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
             and (find_posteriors_of_data(row['age'], 2)
                > find_posteriors_of_data(row['age'], 3))):
             C2_C1 = C2_C1 + 1
        else:
            C3_C1 = C3_C1 + 1

    elif row['class'] == 2:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'], 2))
          and (find_posteriors_of_data(row['age'], 1) >
              find_posteriors_of_data(row['age'],3))):
            C1_C2 = C1_C2 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 2) >
               find_posteriors_of_data(row['age'], 3))):
            C2_C2 = C2_C2 + 1
        else:
            C3_C2 = C3_C2 + 1
    else:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'], 2))
           and (find_posteriors_of_data(row['age'], 1) >
               find_posteriors_of_data(row['age'], 3))):
            C1_C3 = C1_C3 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 2) >
               find_posteriors_of_data(row['age'], 3))):
            C2_C3 = C2_C3 + 1
        else:
            C3_C3 = C3_C3 + 1



print("-------------")
print("C1_C1 = " + str(C1_C1))
print("C2_C1 = " + str(C2_C1))
print("C3_C1 = " + str(C3_C1))
print("-------------")
print("C1_C2 = " + str(C1_C2))
print("C2_C2 = " + str(C2_C2))
print("C3_C2 = " + str(C3_C2))
print("-------------")
print("C1_C3 = " + str(C1_C3))
print("C2_C3 = " + str(C2_C3))
print("C3_C3 = " + str(C3_C3))
print("--------------")

T1_T1 = 0
T2_T1 = 0
T3_T1 = 0

T1_T2 = 0
T2_T2 = 0
T3_T2 = 0

T1_T3 = 0
T2_T3 = 0
T3_T3 = 0
```

```python
for index, row in testing_file.iterrows():
    if row['class'] == 1:
        if((find_posteriors_of_data(row['age'],1) > find_posteriors_of_data(row['age'],2)) and
          (find_posteriors_of_data(row['age'],1) > find_posteriors_of_data(row['age'],3))):
            T1_T1 = T1_T1 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 2) >
              find_posteriors_of_data(row['age'],3))):
            T2_T1 = T2_T1 + 1
        else:
            T3_T1 = T3_T1 + 1

    elif row['class'] == 2:
        if ((find_posteriors_of_data(row['age'],1) > find_posteriors_of_data(row['age'],2))
             and (find_posteriors_of_data(row['age'], 1)
                  > find_posteriors_of_data(row['age'], 3))):
            T1_T2 = T1_T2 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
             and (find_posteriors_of_data(row['age'], 2) >
                    find_posteriors_of_data(row['age'], 3))):
            T2_T2 = T2_T2 + 1
        else:
            T3_T2 = T3_T2 + 1
    else:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'], 2))
            and (find_posteriors_of_data(row['age'], 1)
                  > find_posteriors_of_data(row['age'], 3))):
            T1_T3 = T1_T3 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 2)
                  > find_posteriors_of_data(row['age'], 3))):
            T2_T3 = T2_T3 + 1
        else:
            T3_T3 = T3_T3 + 1

print("-------------")
print("T1_T1 = " + str(T1_T1))
print("T2_T1 = " + str(T2_T1))
print("T3_T1 = " + str(T3_T1))
print("-------------")
print("T1_T2 = " + str(T1_T2))
print("T2_T2 = " + str(T2_T2))
print("T3_T2 = " + str(T3_T2))
print("-------------")
print("T1_T3 = " + str(T1_T3))
print("T2_T3 = " + str(T2_T3))
print("T3_T3 = " + str(T3_T3))
print("--------------")


### Rejection Part

P1_P1 = 0
P2_P1 = 0
P3_P1 = 0
REJECT_P1 = 0

P1_P2 = 0
P2_P2 = 0
P3_P2 = 0
REJECT_P2 = 0

P1_P3 = 0
P2_P3 = 0
P3_P3 = 0
REJECT_P3 = 0
```

```
for index, row in training_file.iterrows():
    if row['class'] == 1:
        if((find_posteriors_of_data(row['age'],1) > find_posteriors_of_data(row['age'],2))
          and (find_posteriors_of_data(row['age'],1) >
              find_posteriors_of_data(row['age'],3)) and
                (find_posteriors_of_data(row['age'],1) > 3/4) ):
            P1_P1 = P1_P1 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],3))
              and (find_posteriors_of_data(row['age'],2) > 3/4) ):
            P2_P1 = P2_P1 + 1
        elif ((find_posteriors_of_data(row['age'], 3) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 3) >
              find_posteriors_of_data(row['age'],2)) and
                (find_posteriors_of_data(row['age'],3) > 3/4) ):
            P3_P1 = P3_P1 + 1
        else:
            REJECT_P1 = REJECT_P1 + 1

    elif row['class'] == 2:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'], 2))
           and (find_posteriors_of_data(row['age'], 1) >
             find_posteriors_of_data(row['age'], 3))
               and (find_posteriors_of_data(row['age'], 1) > 3 / 4)):
            P1_P2 = P1_P2 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
             and (find_posteriors_of_data(row['age'], 2) >
                find_posteriors_of_data(row['age'], 3)) and
                    (find_posteriors_of_data(row['age'], 2) > 3 / 4)):
            P2_P2 = P2_P2 + 1
        elif ((find_posteriors_of_data(row['age'], 3) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 3) >
                 find_posteriors_of_data(row['age'], 2)) and
                      (find_posteriors_of_data(row['age'], 3) > 3 / 4)):
            P3_P2 = P3_P2 + 1
        else:
            REJECT_P2 = REJECT_P2 + 1
    else:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'], 2))
           and (find_posteriors_of_data(row['age'], 1) >
                  find_posteriors_of_data(row['age'], 3)) and
                    (find_posteriors_of_data(row['age'], 1) > 3 / 4)):
            P1_P3 = P1_P3 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 2) >
              find_posteriors_of_data(row['age'], 3)) and
                    (find_posteriors_of_data(row['age'], 2) > 3 / 4)):
            P2_P3 = P2_P3 + 1
        elif ((find_posteriors_of_data(row['age'], 3) > find_posteriors_of_data(row['age'],1))
           and (find_posteriors_of_data(row['age'], 3) >
              find_posteriors_of_data(row['age'], 2)) and
                (find_posteriors_of_data(row['age'], 3) > 3 / 4)):
            P3_P3 = P3_P3 + 1
        else:
            REJECT_P3 = REJECT_P3 + 1

print("-------------")
print("P1_P1 = " + str(P1_P1))
print("P2_P1 = " + str(P2_P1))
print("P3_P1 = " + str(P3_P1))
print("REJECT_P1 = " + str(REJECT_P1))
print("-------------")
print("P1_P2 = " + str(P1_P2))
print("P2_P2 = " + str(P2_P2))
print("P3_P2 = " + str(P3_P2))
print("REJECT_P2 = " + str(REJECT_P2))
print("-------------")
print("P1_P3 = " + str(P1_P3))
print("P2_P3 = " + str(P2_P3))
print("P3_P3 = " + str(P3_P3))
print("REJECT_P3 = " + str(REJECT_P3))
print("--------------")
```

```python
Q1_Q1 = 0
Q2_Q1 = 0
Q3_Q1 = 0
REJECT_Q1 = 0

Q1_Q2 = 0
Q2_Q2 = 0
Q3_Q2 = 0
REJECT_Q2 = 0

Q1_Q3 = 0
Q2_Q3 = 0
Q3_Q3 = 0
REJECT_Q3 = 0

for index, row in testing_file.iterrows():
    if row['class'] == 1:
        if((find_posteriors_of_data(row['age'],1) > find_posteriors_of_data(row['age'],2)) and
           (find_posteriors_of_data(row['age'],1) >
              find_posteriors_of_data(row['age'],3)) and
                (find_posteriors_of_data(row['age'],1) > 3/4) ):
            Q1_Q1 = Q1_Q1 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
         and (find_posteriors_of_data(row['age'], 2) >
            find_posteriors_of_data(row['age'], 3)) and
             (find_posteriors_of_data(row['age'],2) > 3/4) ):
            Q2_Q1 = Q2_Q1 + 1
        elif ((find_posteriors_of_data(row['age'], 3) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 3) >
             find_posteriors_of_data(row['age'], 2)) and
               (find_posteriors_of_data(row['age'],3) > 3/4) ):
            Q3_Q1 = Q3_Q1 + 1
        else:
            REJECT_Q1 = REJECT_Q1 + 1

    elif row['class'] == 2:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'], 2))
          and (find_posteriors_of_data(row['age'], 1) >
              find_posteriors_of_data(row['age'], 3)) and
                 (find_posteriors_of_data(row['age'], 1) > 3 / 4)):
            Q1_Q2 = Q1_Q2 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 2) >
              find_posteriors_of_data(row['age'], 3)) and
                   (find_posteriors_of_data(row['age'], 2) > 3 / 4)):
            Q2_Q2 = Q2_Q2 + 1
        elif ((find_posteriors_of_data(row['age'], 3) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 3) >
              find_posteriors_of_data(row['age'], 2)) and
                   (find_posteriors_of_data(row['age'], 3) > 3 / 4)):
            Q3_Q2 = Q3_Q2 + 1
        else:
            REJECT_Q2 = REJECT_Q2 + 1
    else:
        if ((find_posteriors_of_data(row['age'], 1) > find_posteriors_of_data(row['age'],2))
          and (find_posteriors_of_data(row['age'], 1) >
             find_posteriors_of_data(row['age'], 3)) and
               (find_posteriors_of_data(row['age'], 1) > 3 / 4)):
            Q1_Q3 = Q1_Q3 + 1
        elif ((find_posteriors_of_data(row['age'], 2) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 2) >
             find_posteriors_of_data(row['age'], 3)) and
               (find_posteriors_of_data(row['age'], 2) > 3 / 4)):
           Q2_Q3 = Q2_Q3 + 1
        elif ((find_posteriors_of_data(row['age'], 3) > find_posteriors_of_data(row['age'],1))
          and (find_posteriors_of_data(row['age'], 3) >find_posteriors_of_data(row['age'],2))
            and (find_posteriors_of_data(row['age'], 3) > 3 / 4)):
            Q3_Q3 = Q3_Q3 + 1
        else:
            REJECT_Q3 = REJECT_Q3 + 1
```

```python
print("-------------")
print("Q1_Q1 = " + str(Q1_Q1))
print("Q2_Q1 = " + str(Q2_Q1))
print("Q3_Q1 = " + str(Q3_Q1))
print("REJECT_Q1 = " + str(REJECT_Q1))
print("-------------")
print("Q1_Q2 = " + str(Q1_Q2))
print("Q2_Q2 = " + str(Q2_Q2))
print("Q3_Q2 = " + str(Q3_Q2))
print("REJECT_Q2 = " + str(REJECT_Q2))
print("-------------")
print("Q1_Q3 = " + str(Q1_Q3))
print("Q2_Q3 = " + str(Q2_Q3))
print("Q3_Q3 = " + str(Q3_Q3))
print("REJECT_Q3 = " + str(REJECT_Q3))
print("--------------")
```