**Özyeğin University**

C.S. 554.A

Homework I
Fall 2020

**Submitted by:** M. Furkan Oruc (MSc. Student at CE Department)
**Student ID:** S025464
**Submission Date:** 19/10/2020

## Content

## 1. Estimation of parameters based on Training Set

Based on the training dataset, following unknowns are calculated:

1. Priors

In order to find the priors P(C = +) and p(C = -); number of each class is counted and divided by the total number of instances.

In the training set, 90 instances were provided with 30 Positive and 60 Negative instances.

2. Mean and Variance

Since class densities are assumed to be Gaussian distributions as followed:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

After reaching out to log likelihood, the maximum likelihood estimation that we find is be taking partial derivatives of likelihood and setting them to 0, where mean and variance are the estimators to be utilized.

$$m = \frac{\sum t\, x^t}{N}$$

$$s^2 = \frac{\sum t\, (x^t - m)^2}{N}$$

Then, after likelihoods for every instance and posterior probabilities are calculated for each class, model is tested with testing dataset. Following are the outcomes in the next section.

## 2. Outcomes

### a. Training Dataset Results

$$mi(for\ negative\ instances\ \ P(C=-)=39.45$$
$$mi\ (for\ positive\ instances, P(C=\ +)=26.56$$
$$P\ (C=\ +)(Prior)=0.3333$$
$$P\ (C=\ -)(Prior)=0.6666$$
$$Standard\ Deviation\ (for\ negative\ instances, P\ (C=\ -)=5.0318$$
$$Standard\ Deviation\ (for\ positive\ instances, P\ (C=\ +)=3.478$$

Total False Positive from Training Dataset: 3

Total False Negative from Training Dataset: 2

As the calculations and background will be provided in next pages, the model practiced 3 false positives and 2 false negatives through the training period.

**Initial Calculation for the Risk Function:**

$$\boldsymbol{Assume\ the\ 2nd\ Case: where\ F.P.=2\ \&\ F.P.=1}$$

$$\boldsymbol{R\left(\frac{\partial i}{x}\right)=1-P\left(\frac{Ci}{x}\right)}$$

$$\boldsymbol{In\ this\ case: R\left(\frac{\partial i}{x}\right)=1-2*P\left(\frac{Ci}{x}\right)}$$

$$\boldsymbol{To\ determine\ the\ boundary, the\ risk\ of\ choosing\ one\ class\ on\ another\ must\ be\ less\ costly. So,}$$

$$\boldsymbol{R\left(\frac{i}{x}\right)<R\left(\frac{\partial 2}{x}\right)}$$

$$\boldsymbol{1-2*P\left(\frac{C1}{x}\right)>?\,1-\ P\left(\frac{C2}{x}\right)}$$

$$\boldsymbol{P\left(\frac{C1}{x}\right)<?\,P\left(\frac{C2}{x}\right)/2}$$

$$\boldsymbol{Above\ is\ the\ question\ determines\ which\ class\ to\ be\ chosen.}$$

This risk analysis is applied to all boundary issues to find the right point and the region of rejection.

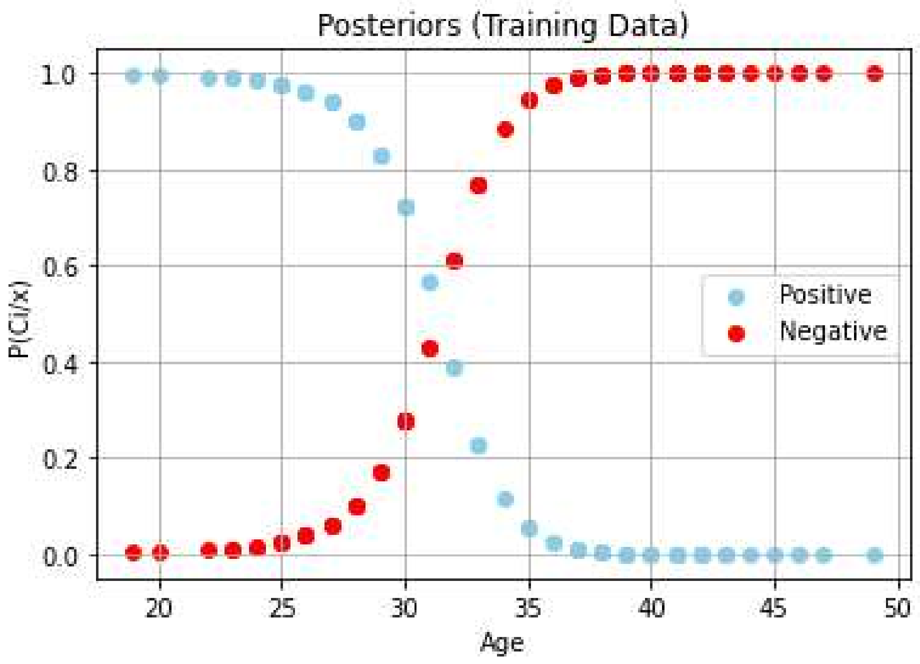

*Figure 1:Posteriors of 2 Classes: P(C=+) and P(C=-)*

It can also be observed posteriors intersect at only one point.

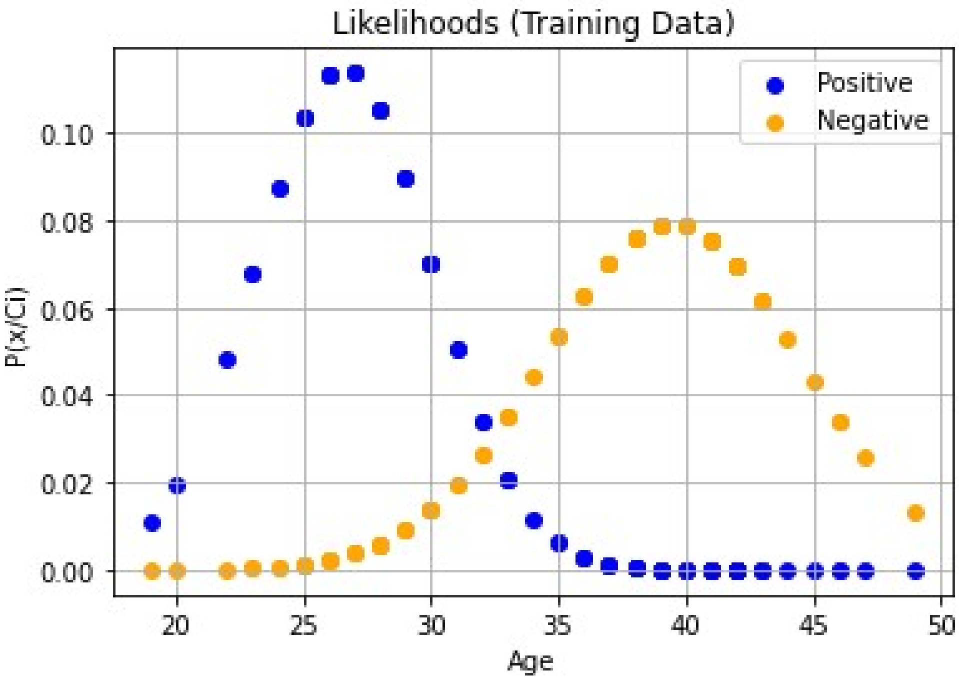

*Figure 2: Likelihoods of 2 Classes: P(C=+) and P(C=-)*

Based on likelihoods plot, it can be stated that variances are different for two classes.

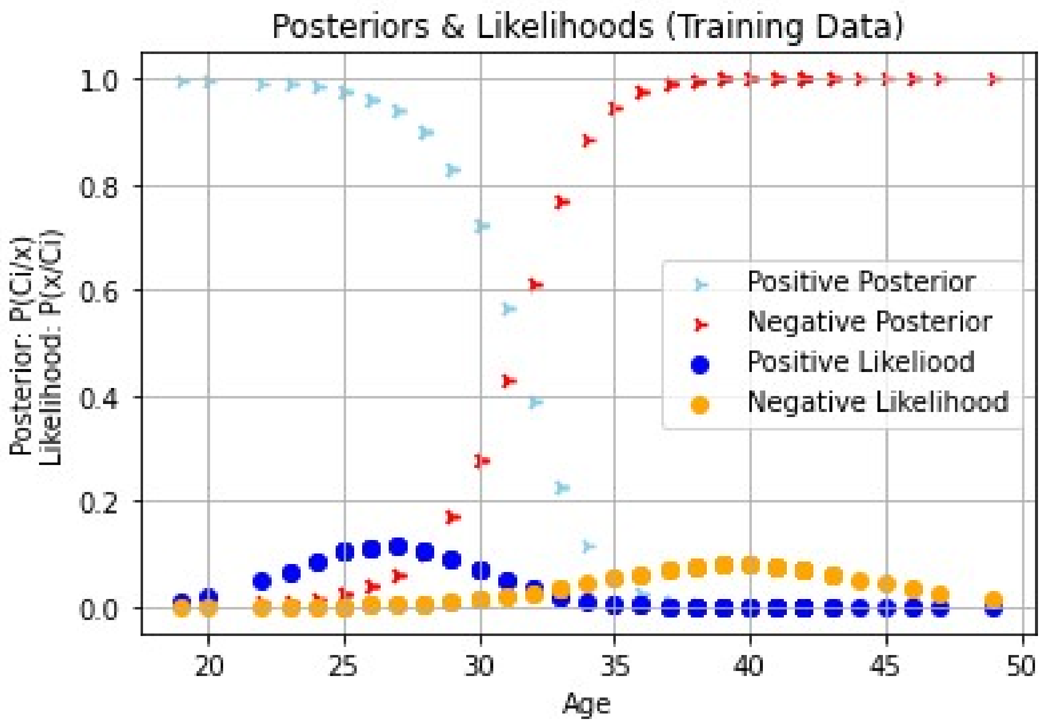

*Figure 3: Posteriors and Likelihoods Combined*

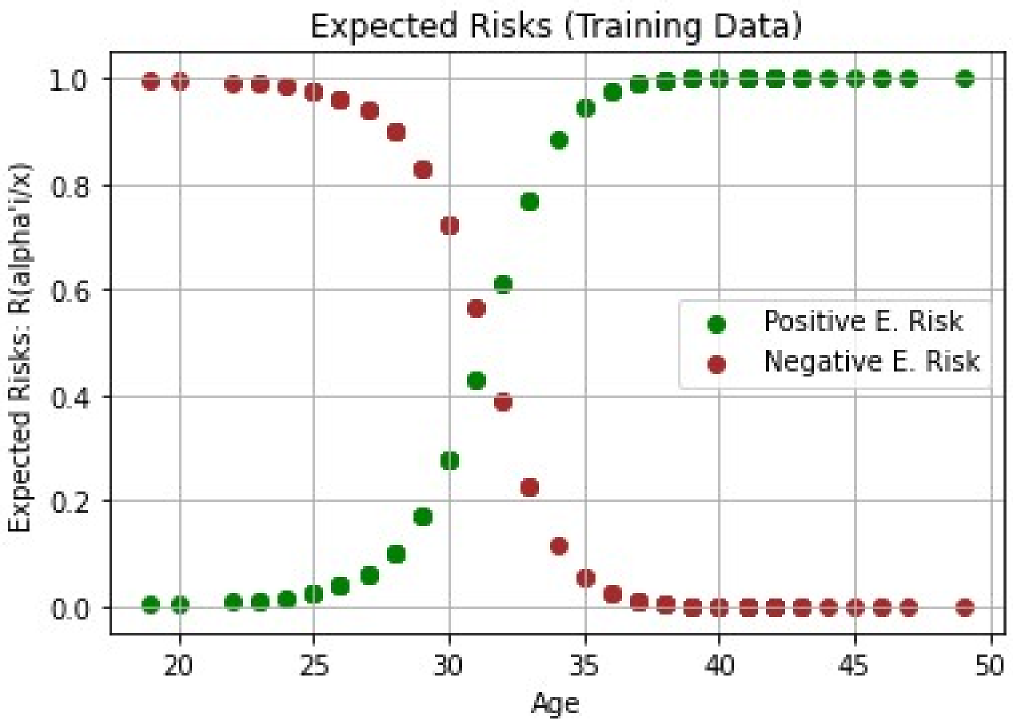

*Figure 4: Expected Risks of 2 Classes*

Although there hasn't been a reject system integrated into the model, if there were, one intersection can be described. It would have occurred as a single area based rejection domain.

**3 Different Cases are provided in the brief for this report's subject.**

The cases can be summarized as:

1 - The loss of a False Positive = 1 & The loss of a False Negative = 1

2 - The loss of a False Positive = 2 & The loss of a False Negative = 1

3 - The loss of a False Positive = 1 & The loss of a False Negative = 2

Below are provided threshold calculations and plots for the first case.

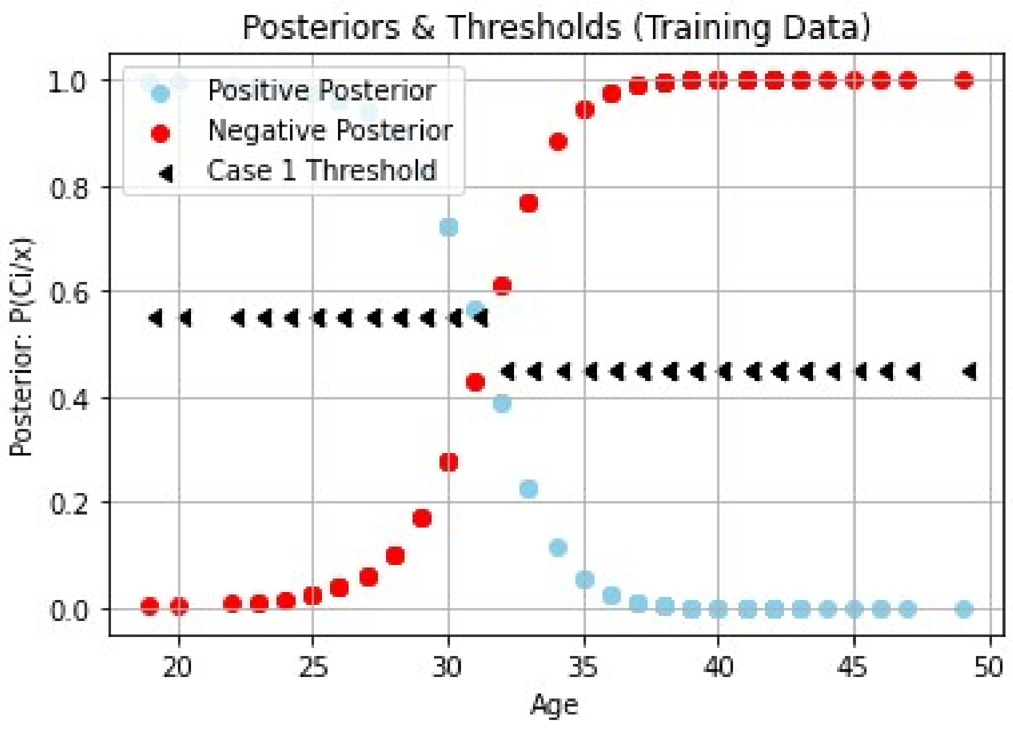

*Figure 5: Threshold & posterior: 1st Case*

For case 1, since there aren't any rejection system introduced to the model and both losses are in the same representation, the boundary is where both posteriors intersect. On the other hand, it can be stated that the region around the intersection can be stated as the reject region in order to minimize the cost of misclassification.

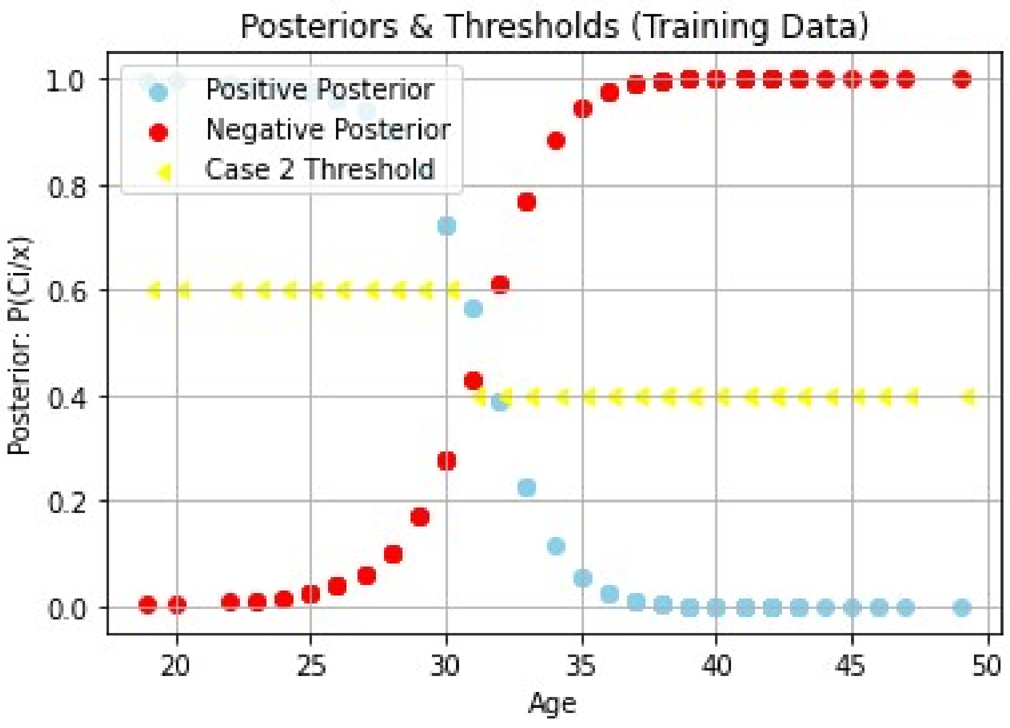

*Figure 6: Thresold & Posterior: Case 2*

In the 2nd case, it's significantly more costly to assign a negative item to the positive class. That results as the boundary of decision moving to the right hand side. When the losses are unsymmetrical, the boundary shifts towards the class that incurs higher risk when misclassified. In other words, since assigning negative class's elements to positive class is highly risky, more negative class assignments are made confidently.

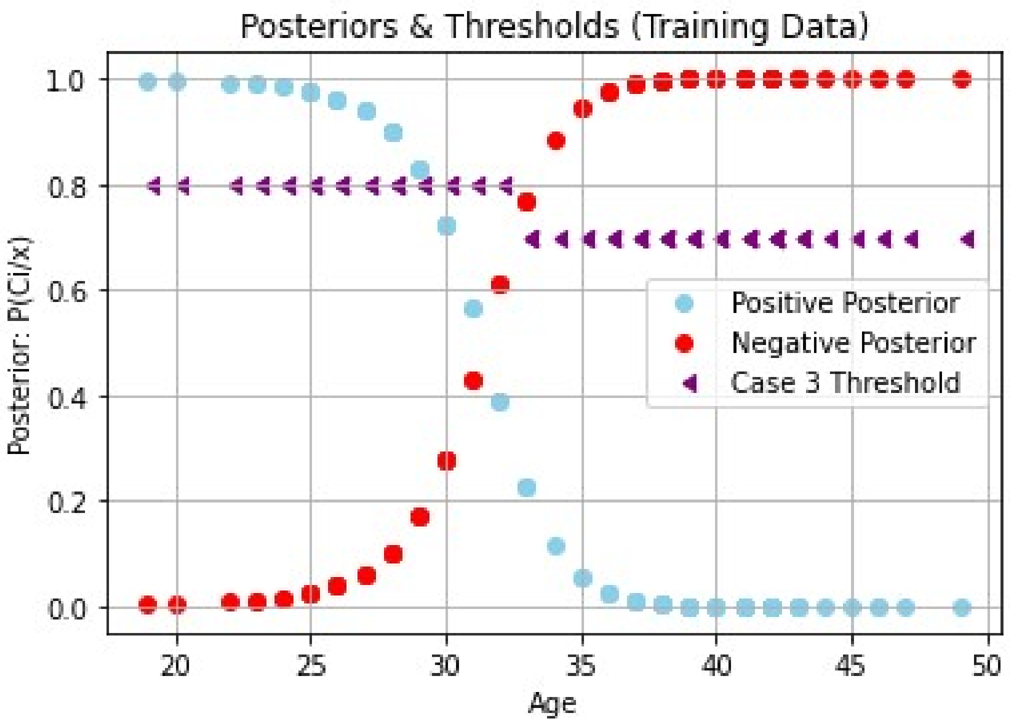

*Figure 7: Posteriors & Thresholds: 3rd Case*

In the above plot which is based on the third case, it can be observed that due to positive class carrying a higher risk when misclassified, as well as also result of risk and loss functions, the boundary is visible slightly on the right. Here is where it's more costly to assign to negative so we only assign when we need to.

Below plot covering all cases can be utilized for an overview of all boundaries and their comparisons.

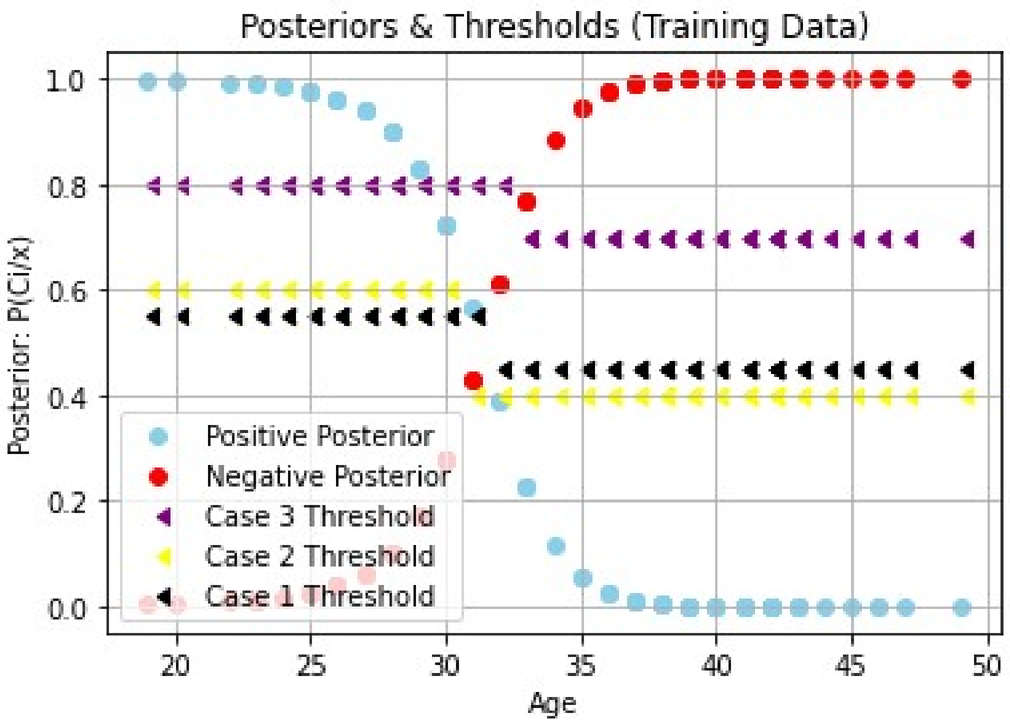

*Figure 8:All Thresholds, Training Dataset*

As it can be observed, all thresholds are dispersed based on their respective risk outcomes.

**b. Testing Dataset Results**

Through the application of the model on testing dataset, following values which are obtained from training dataset is utilized.

$$mi(for\ negative\ instances\ \ P(C=-)=39.45$$
$$mi\ (for\ positive\ instances, P(C=\ +)=26.56$$
$$P\ (C=\ +)(Prior)=0.3333$$
$$P\ (C=\ -)(Prior)=0.6666$$
$$Standard\ Deviation\ (for\ negative\ instances, P\ (C=\ -)=5.0318$$
$$Standard\ Deviation\ (for\ positive\ instances, P\ (C=\ +)=3.478$$

$$Total\ False\ Positive\ from\ Training\ Dataset: 0$$

$$Total\ False\ Negative\ from\ Training\ Dataset: 5$$

Although the above result is the product of further calculations as it'll be provided in next pages, it's also shared here. Based on the model, there aren't any false positives occurred and 5 false negative classifications have occurred.

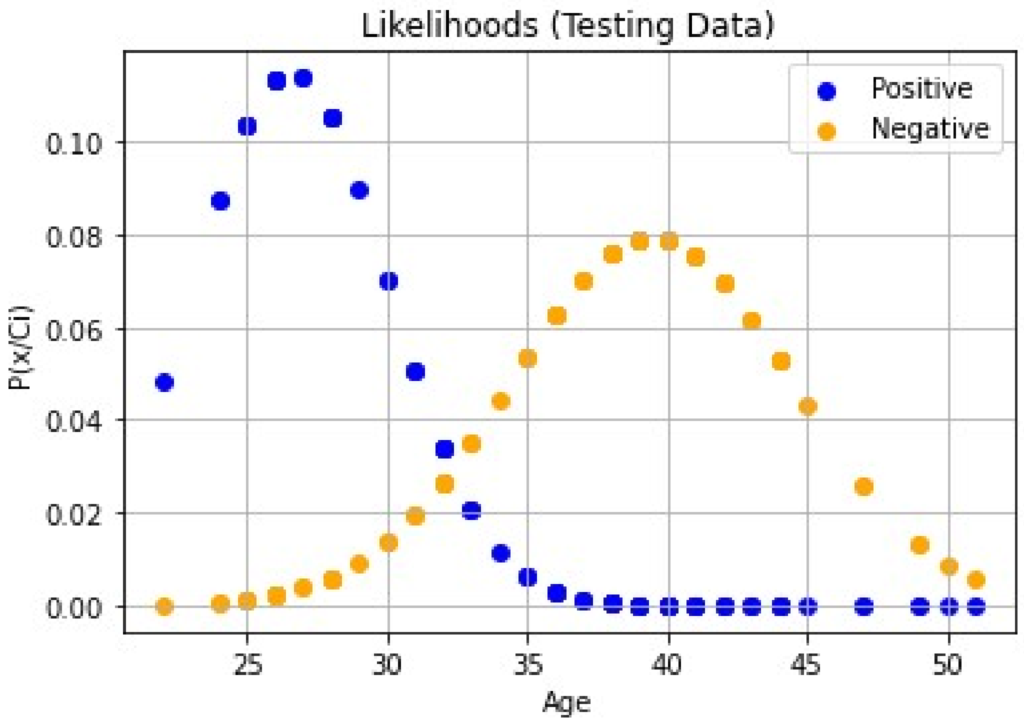

*Figure 9: Likelihood Test Outcome, Testing Dataset*

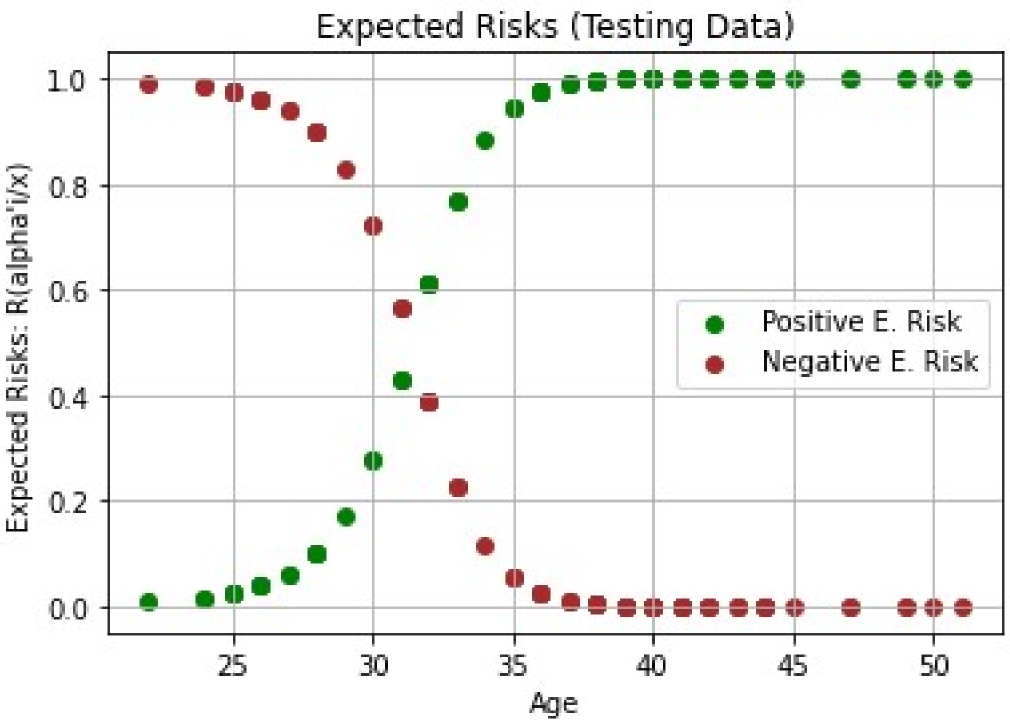

*Figure 10: Expected Risks on Testing Dataset*

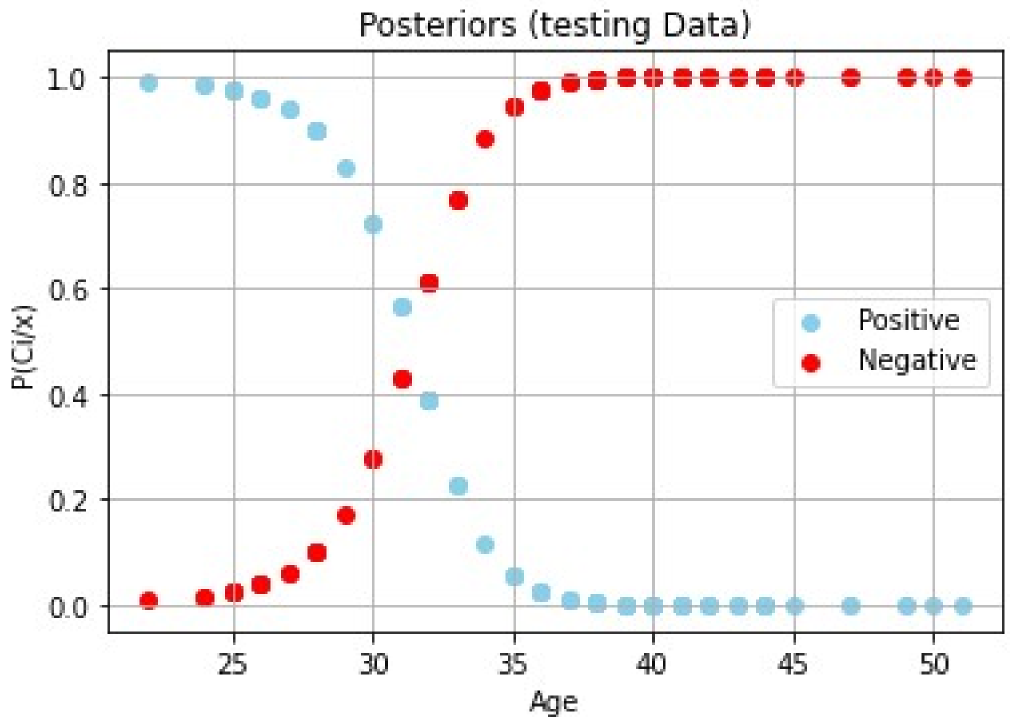

*Figure 11:Posteriors of Testing Dataset*

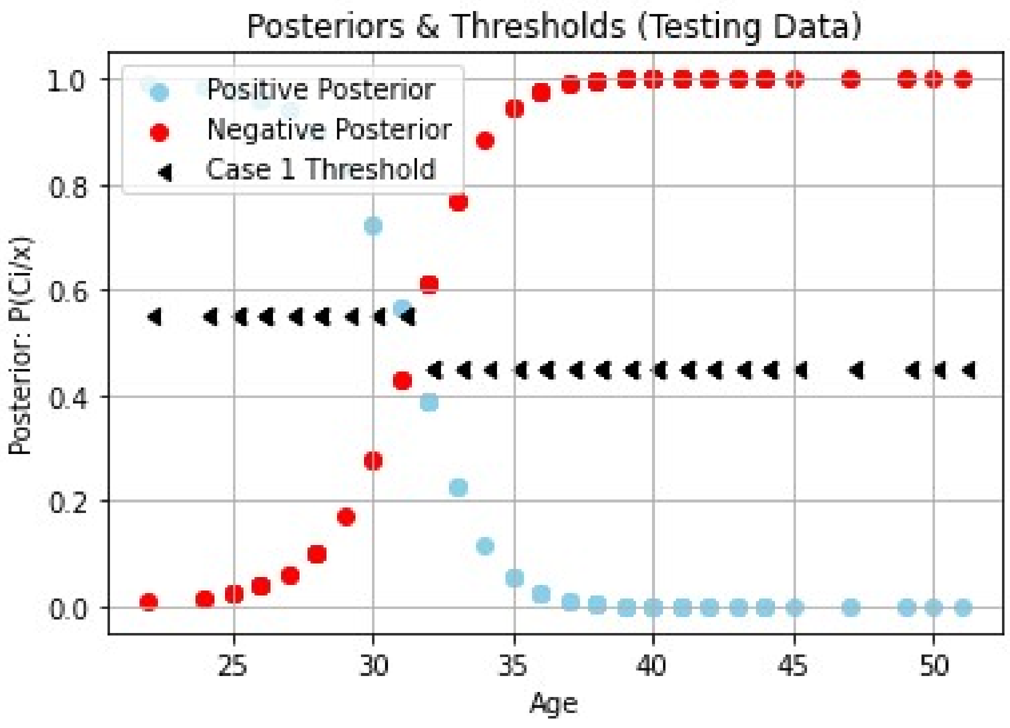

*Figure 12:Case 1: Threshold and Posteriors*

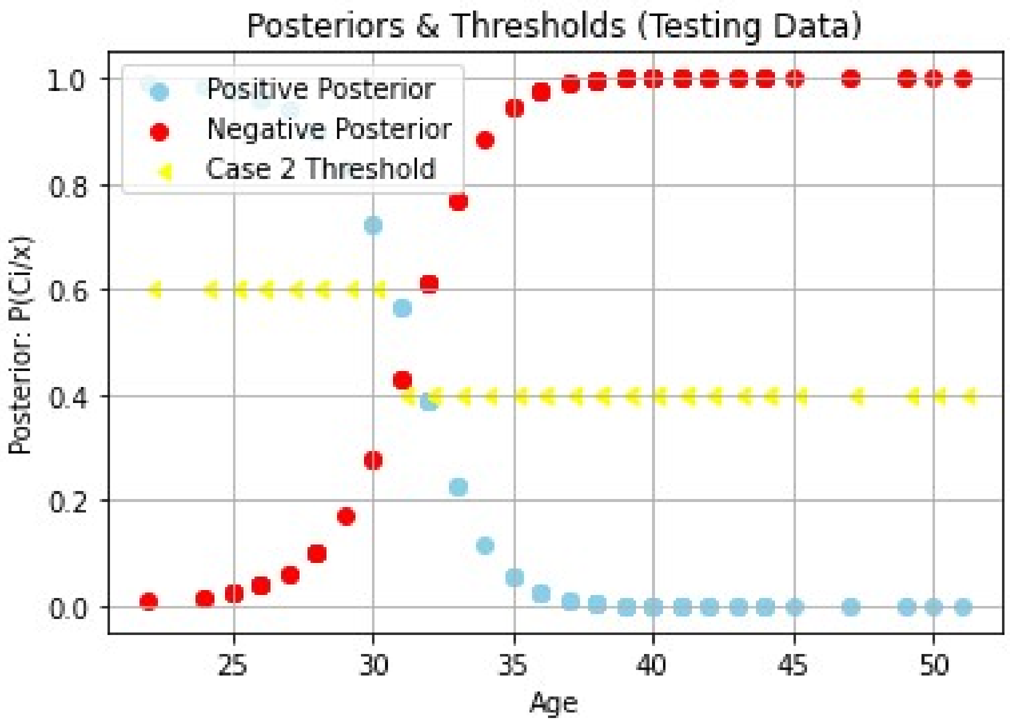

*Figure 13:Case 2: Posteriors and Threshold Boundary*

It can be observed that as in the training dataset, the boundary moves slightly left since it's too costly to express a positive when it's negative.

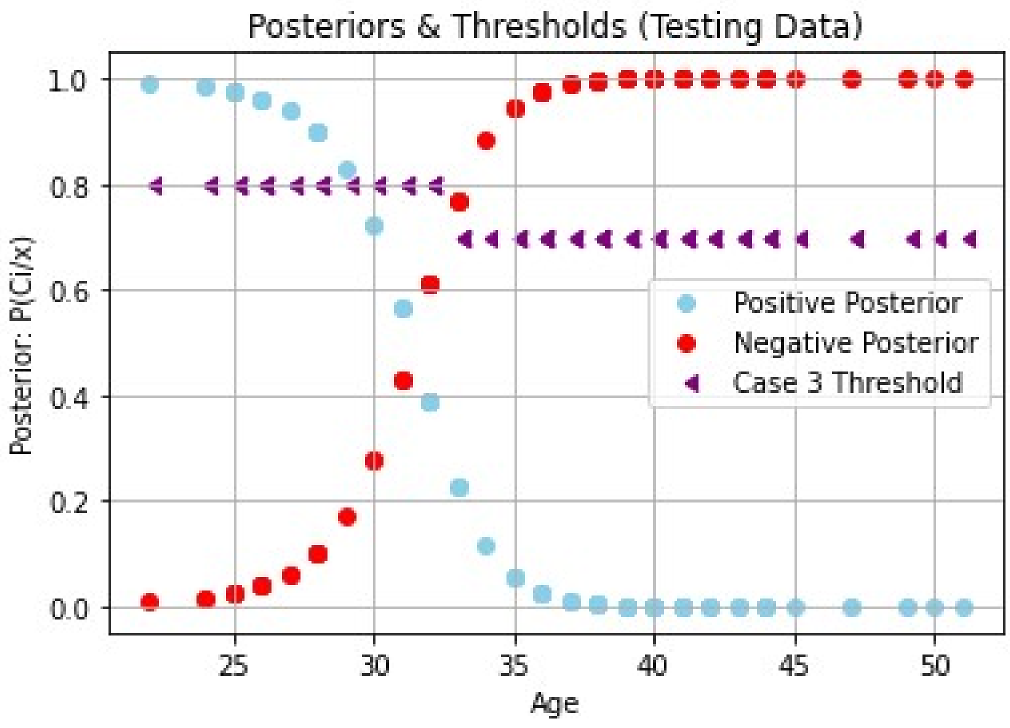

*Figure 14: Case 3: Testing Set*

As in the training set, it's observable that it's too costly to assign negative when it's positive, so we go a step further and assign even some negative ones to the positive although they may be misclassified, for the sake of less cost.

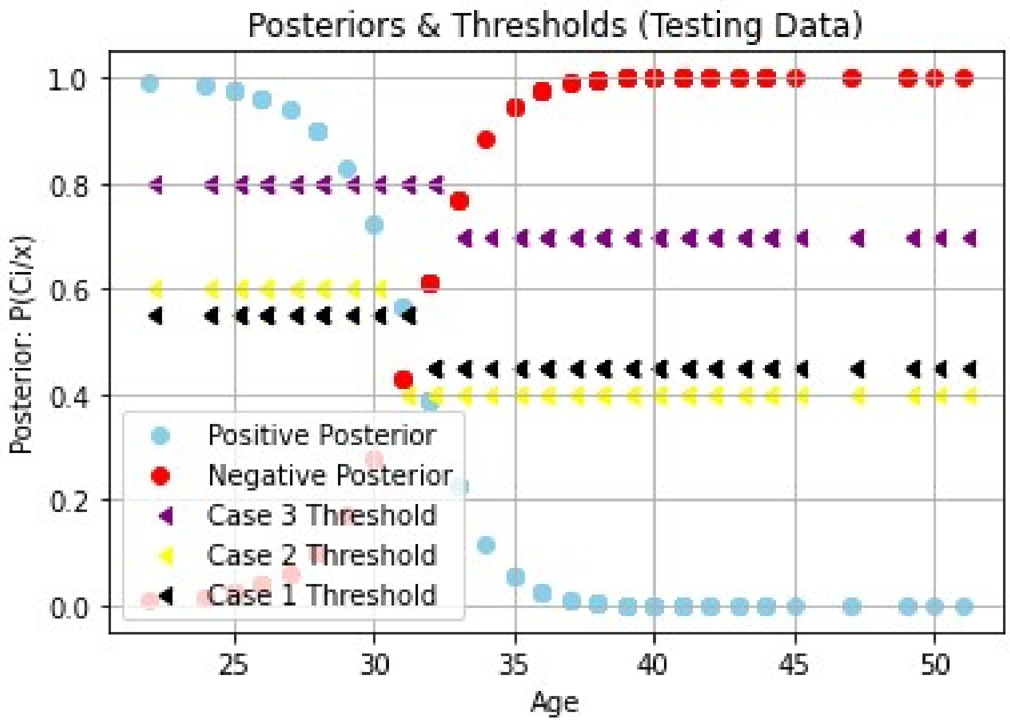

*Figure 15: All Thresholds of Testing Dataset*

## 3.Source Code

```
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 18 01:10:28 2020

@author: User
"""


import pandas
import math

###Read CSV-Console Working Directory is the file.

training_set = pandas.read_csv("training.csv", sep=",")

#total_list_counter

total_rows = 0
for row in training_set["result"]:
    total_rows += 1



###Calculate Priors

#positive_counter

positive_list = training_set.loc[training_set["result"] == "Positive", :]
```

```
row_count_positive = 0
for row in positive_list["result"]:
    row_count_positive = row_count_positive + 1

prior_positive = row_count_positive / total_rows

#negative_counter

negative_list = training_set.loc[training_set["result"] == "Negative", :]

row_count_negative = 0
for row in negative_list["result"]:
    row_count_negative = row_count_negative + 1

prior_negative = row_count_negative / total_rows


###Calculate Mean & Variance

#means

positive_sum = training_set.loc[training_set["result"] == "Positive", "age"].sum()
negative_sum = training_set.loc[training_set["result"] == "Negative", "age"].sum()

mean_positive = positive_sum / row_count_positive
mean_negative = negative_sum / row_count_negative

#variances
```

```
variance_positive_holder = 0
for row in positive_list["age"]:
    variance_positive_holder += (row - mean_positive)**2

variance_positive = (variance_positive_holder / mean_positive)**(0.5)

variance_negative_holder = 0
for row in negative_list["age"]:
    variance_negative_holder += (row - mean_negative)**2

variance_negative = (variance_negative_holder / mean_negative)**(0.5)



###Calculate Likelihoods

i = 0
likelihood_positive = {}
likelihood_negative = {}

for row in training_set["age"]:
    likelihood_positive[i] = ((1)/((variance_positive)*((2*(math.pi))**(0.5))))*(math.exp((-
0.5)*(((row-mean_positive)/variance_positive))**2))
    likelihood_negative[i] = ((1)/((variance_negative)*((2*(math.pi))**(0.5))))*(math.exp((-
0.5)*(((row-mean_negative)/variance_negative))**2))
    i = i + 1



###Calculate_Posteriors

k=0
```

```
posterior_positive = {}

posterior_negative = {}


for row in training_set["age"]:

    posterior_positive[k] = (likelihood_positive[k] * prior_positive)/((likelihood_positive[k]*prior_positive)+(likelihood_negative[k]*prior_negative))

    posterior_negative[k] = (likelihood_negative[k] * prior_negative)/((likelihood_positive[k]*prior_positive)+(likelihood_negative[k]*prior_negative))

    k = k + 1


###Minimum Expected Risk Calculation


t=0

risk_positive_case_1 = {}

risk_negative_case_1 = {}


for t in range (0,90):

    risk_positive_case_1[t] = 1 - posterior_positive[t]

    risk_negative_case_1[t] = 1 - posterior_negative[t]

    t = t + 1


###Introducing the Loss Function


#Case 1: False Positive:1 & False Negative:1

#For this case: Choose which one ever has the more probability since correct decisions have no loss and all errors are equally costly.

#This would minimize the risk int the best way.


loss_function = {}
```

```
loss_false_positive_counter_1 = 0
loss_false_negative_counter_1 = 0

#false positive firstly
m=0

for row in training_set["result"]:
    if (posterior_positive[m] > 0.5) & (row == "Negative"):
        loss_false_positive_counter_1 = loss_false_positive_counter_1 + 1
        m = m + 1
    else:
        m = m + 1

print ("Total False Positive for Training:")
print (loss_false_positive_counter_1)

#false negative secondly

b=0

for row in training_set["result"]:
    if (posterior_positive[b] < 0.5) & (row == "Positive"):
        loss_false_negative_counter_1 = loss_false_negative_counter_1 + 1
        b = b + 1
    else:
        b = b + 1
print ("Total False Negative for Training:")
print (loss_false_negative_counter_1)
```

```
#threshold:case_1

r = 0
threshold_case_1 = {}
threshold_case_1_check = {}
for r in range (0,90):
    if posterior_negative[r] > (posterior_positive[r]):
        threshold_case_1[r] = "Negative"
        threshold_case_1_check[r] = 0.45
    elif posterior_negative[r] < (posterior_positive[r]):
        threshold_case_1[r] = "Positive"
        threshold_case_1_check[r] = 0.55

#Case 2: False Positive:2 & False Negative:1
#Need to recalculate the expected risk since both losses are not equally costly. This will help
to explore the threshold.

risk_positive_case_2 = {}
risk_negative_case_2 = {}
t=0
for t in range (0,90):
    risk_positive_case_2[t] = 1 - (2*posterior_positive[t])
    risk_negative_case_2[t] = 1 - posterior_negative[t]
    t = t + 1

#for threshold of case 2: according to loss&expected risk calculation provided in the report

p = 0
threshold_case_2 = {}
threshold_case_2_check = {}
```

```
for p in range (0,90):
    if posterior_positive[p] > (posterior_negative[p]/2):
        threshold_case_2[p] = "Positive"
        threshold_case_2_check[p] = 0.8
    elif posterior_positive[p] < (posterior_negative[p]/2):
        threshold_case_2[p] = "Negative"
        threshold_case_2_check[p] = 0.7


#false positive & false negative numbers do not change based on loss function, so same as the
first case




#Case 3: False Positive:1 & False Negative:2


#Need to recalculate the expected risk since both losses are not equally costly. This will help
to explore the threshold.


risk_positive_case_3 = {}
risk_negative_case_3 = {}
y=0
for y in range (0,90):
    risk_negative_case_3[y] = 1 - (2*posterior_negative[y])
    risk_positive_case_3[y] = 1 - posterior_positive[y]
    y = y + 1


#for threshold of case 2: according to loss&expected risk calculation provided in the report


q = 0
threshold_case_3 = {}
threshold_case_3_check = {}
```

```
for q in range (0,90):
    if posterior_negative[q] > (posterior_positive[q]/2):
        threshold_case_3[q] = "Negative"
        threshold_case_3_check[q] = 0.4
    elif posterior_negative[q] < (posterior_positive[q]/2):
        threshold_case_3[q] = "Positive"
        threshold_case_3_check[q] = 0.6

###Plot Training Results
#Dictionary to Tuple Change

import numpy as np
from matplotlib.pylab import plt



list_posterior_positive = sorted(posterior_positive.items())
xpp,ypp=zip(*list_posterior_positive)

list_likelihood_positive = sorted(likelihood_positive.items())
xlp,ylp=zip(*list_likelihood_positive)

list_posterior_negative = sorted(posterior_negative.items())
xpn,ypn=zip(*list_posterior_negative)

list_likelihood_negative = sorted(likelihood_negative.items())
xln,yln=zip(*list_likelihood_negative)

x_age = training_set["age"].values.tolist()
```

```
#multiple_plot

#posteriors


plt.scatter(x_age, ypp, color='skyblue', label='Positive')

plt.scatter(x_age,ypn, color='red', label='Negative')

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Posteriors (Training Data)")

plt.xlabel("Age")

plt.ylabel("P(Ci/x)")

plt.legend()

plt.grid(True)


#likelihoods

plt.scatter(x_age, ylp, color='blue', label='Positive')

plt.scatter(x_age,yln, color='orange', label='Negative')

plt.title("Likelihoods (Training Data)")

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.legend()

plt.xlabel("Age")

plt.ylabel("P(x/Ci)")

plt.grid(True)



#posterior&threshold_training


#Training: for case 2 + case 3 threshold analysis
```

```
list_threshold_case_2 = sorted(threshold_case_2_check.items())

xt2,yt2=zip(*list_threshold_case_2)

list_threshold_case_3 = sorted(threshold_case_3_check.items())

xt3,yt3=zip(*list_threshold_case_3)

list_threshold_case_1 = sorted(threshold_case_1_check.items())

xt1,yt1=zip(*list_threshold_case_1)


#plot threshold+posterior


plt.scatter(x_age, ypp, color='skyblue', label='Positive Posterior')

plt.scatter(x_age,ypn, color='red', label='Negative Posterior')


#plt.scatter(x_age, yt2, color='purple', label='Case 3 Threshold', marker = 4)

plt.scatter(x_age, yt3, color='yellow', label='Case 2 Threshold', marker = 4)

#plt.scatter(x_age, yt1, color='black', label='Case 1 Threshold', marker = 4)


#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Posteriors & Thresholds (Training Data)")

plt.xlabel("Age")

plt.ylabel("Posterior: P(Ci/x)")

plt.legend()

plt.grid(True)

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.xlabel("Age")


#Expected Risks


list_risk_positive_1 = sorted(risk_positive_case_1.items())
```

```
xrp1,yrp1=zip(*list_risk_positive_1)


list_risk_negative_1 = sorted(risk_negative_case_1.items())

xrn1,yrn1=zip(*list_risk_negative_1)


plt.scatter(x_age, yrp1, color='green', label='Positive E. Risk')

plt.scatter(x_age,yrn1, color='brown', label='Negative E. Risk')

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Expected Risks (Training Data)")

plt.xlabel("Age")

plt.ylabel("Expected Risks: R(alpha'i/x)")

plt.legend()

plt.grid(True)




#Solo Expected Risk


plt.scatter(x_age, yrp1, color='green')

plt.scatter(x_age,yrn1, color='brown')

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.legend("Age", "Probability")

plt.grid(True)


###


plt.scatter(x_age, yt2, color='purple', label='Case 2 Threshold')

plt.scatter(x_age, yt3, color='black', label='Case 3 Threshold')

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")
```

```
plt.title("Threshold of Decision for Minimum Expected Risk")

plt.xlabel("Age")

plt.ylabel("2=Positive for Case 2 & 1.8=Positive for Case 3\n1=Negative for Case 2 & 1.4 = Negative for Case 3")

plt.legend()

plt.grid(True)


###Test the Testing Data


testing_set = pandas.read_csv("testing.csv", sep=",")


###Calculate Likelihoods


i = 0

likelihood_positive_t = {}

likelihood_negative_t = {}


for row in testing_set["age"]:

    likelihood_positive_t[i] = ((1)/((variance_positive)*((2*(math.pi))**(0.5))))*(math.exp((-0.5)*(((row-mean_positive)/variance_positive))**2))

    likelihood_negative_t[i] = ((1)/((variance_negative)*((2*(math.pi))**(0.5))))*(math.exp((-0.5)*(((row-mean_negative)/variance_negative))**2))

    i = i + 1



###Calculate_Posteriors


k=0

posterior_positive_t = {}

posterior_negative_t = {}
```

```
for row in testing_set["age"]:

    posterior_positive_t[k] = (likelihood_positive_t[k] * prior_positive)/((likelihood_positive_t[k]*prior_positive)+(likelihood_negative_t[k]*prior_negative))

    posterior_negative_t[k] = (likelihood_negative_t[k] * prior_negative)/((likelihood_positive_t[k]*prior_positive)+(likelihood_negative_t[k]*prior_negative))

    k = k + 1


###Minimum Expected Risk Calculation


t=0

risk_positive_case_1_t = {}

risk_negative_case_1_t = {}


for t in range (0,90):

    risk_positive_case_1_t[t] = 1 - posterior_positive_t[t]

    risk_negative_case_1_t[t] = 1 - posterior_negative_t[t]

    t = t + 1


###Introducing the Loss Function


#Case 1: False Positive:1 & False Negative:1

#For this case: Choose which one ever has the more probability since correct decisions have no loss and all errors are equally costly.

#This would minimize the risk int the best way.


loss_function = {}

loss_false_positive_counter_1_t = 0

loss_false_negative_counter_1_t = 0
```

```
#false positive firstly
m=0

for row in testing_set["result"]:
    if (posterior_positive_t[m] > 0.5) & (row == "Negative"):
        loss_false_positive_counter_1_t = loss_false_positive_counter_1_t + 1
        m = m + 1
    else:
        m = m + 1

print ("Total False Positive for testing:")
print (loss_false_positive_counter_1_t)

#false negative secondly

b=0

for row in testing_set["result"]:
    if (posterior_positive_t[b] < 0.5) & (row == "Positive"):
        loss_false_negative_counter_1_t = loss_false_negative_counter_1_t + 1
        b = b + 1
    else:
        b = b + 1
print ("Total False Negative for testing:")
print (loss_false_negative_counter_1_t)

#threshold:case_1

r = 0
```

```
threshold_case_1_t = {}

threshold_case_1_check_t = {}

for r in range (0,90):

    if posterior_negative_t[r] > (posterior_positive_t[r]):

        threshold_case_1_t[r] = "Negative"

        threshold_case_1_check_t[r] = 0.45

    elif posterior_negative_t[r] < (posterior_positive_t[r]):

        threshold_case_1_t[r] = "Positive"

        threshold_case_1_check_t[r] = 0.55


#Case 2: False Positive:2 & False Negative:1

#Need to recalculate the expected risk since both losses are not equally costly. This will help
to explore the threshold.


risk_positive_case_2_t = {}

risk_negative_case_2_t = {}

t=0

for t in range (0,90):

    risk_positive_case_2_t[t] = 1 - (2*posterior_positive_t[t])

    risk_negative_case_2_t[t] = 1 - posterior_negative_t[t]

    t = t + 1


#for threshold of case 2: according to loss&expected risk calculation provided in the report


p = 0

threshold_case_2_t = {}

threshold_case_2_check_t = {}

for p in range (0,90):

    if posterior_positive_t[p] > (posterior_negative_t[p]/2):

        threshold_case_2_t[p] = "Positive"
        threshold_case_2_check_t[p] = 0.8
    elif posterior_positive_t[p] < (posterior_negative_t[p]/2):
        threshold_case_2_t[p] = "Negative"
        threshold_case_2_check_t[p] = 0.7



#false positive & false negative numbers do not change based on loss function, so same as the first case



#Case 3: False Positive:1 & False Negative:2


#Need to recalculate the expected risk since both losses are not equally costly. This will help to explore the threshold.


risk_positive_case_3_t = {}
risk_negative_case_3_t = {}
y=0
for y in range (0,90):
    risk_negative_case_3_t[y] = 1 - (2*posterior_negative_t[y])
    risk_positive_case_3_t[y] = 1 - posterior_positive_t[y]
    y = y + 1

#for threshold of case 2: according to loss&expected risk calculation provided in the report


q = 0
threshold_case_3_t = {}
threshold_case_3_check_t = {}
for q in range (0,90):
    if posterior_negative_t[q] > (posterior_positive_t[q]/2):
        threshold_case_3_t[q] = "Negative"
        threshold_case_3_check_t[q] = 0.4
    elif posterior_negative_t[q] < (posterior_positive_t[q]/2):
        threshold_case_3_t[q] = "Positive"
        threshold_case_3_check_t[q] = 0.6



###Plot testing Results
#Dictionary to Tuple Change

import numpy as np
from matplotlib.pylab import plt



list_posterior_positive_t = sorted(posterior_positive_t.items())
xpp_t,ypp_t=zip(*list_posterior_positive_t)

list_likelihood_positive_t = sorted(likelihood_positive_t.items())
xlp_t,ylp_t=zip(*list_likelihood_positive_t)

list_posterior_negative_t = sorted(posterior_negative_t.items())
xpn_t,ypn_t=zip(*list_posterior_negative_t)

list_likelihood_negative_t = sorted(likelihood_negative_t.items())
xln_t,yln_t=zip(*list_likelihood_negative_t)

x_age_t = testing_set["age"].values.tolist()
```

```
#multiple_plot

#posteriors


plt.scatter(x_age_t, ypp_t, color='skyblue', label='Posterior Positive')

plt.scatter(x_age_t, ypn_t, color='red', label='Posterior Negative')

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Posteriors (Testing Data)")

plt.xlabel("Age")

plt.ylabel("P(Ci/x)")

plt.legend()

plt.grid(True)


#likelihoods

plt.scatter(x_age_t, ylp_t, color='blue', label='Likelihood Positive', marker = 4)

plt.scatter(x_age_t, yln_t, color='orange', label='Likelihood Negative', marker = 4)

plt.title("Posterior & Likelihoods (Testing Data)")

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.legend()

plt.xlabel("Age")

plt.ylabel("Posterior: P(Ci/x) \n Likelihood: P(x/Ci)")

plt.grid(True)



#posterior&threshold_testing


#testing: for case 2 + case 3 threshold analysis
```

```
list_threshold_case_2_t = sorted(threshold_case_2_check_t.items())

xt2_t,yt2_t=zip(*list_threshold_case_2_t)

list_threshold_case_3_t = sorted(threshold_case_3_check_t.items())

xt3_t,yt3_t=zip(*list_threshold_case_3_t)

list_threshold_case_1_t = sorted(threshold_case_1_check_t.items())

xt1_t,yt1_t=zip(*list_threshold_case_1_t)


#plot threshold+posterior


plt.scatter(x_age_t, ypp_t, color='skyblue', label='Positive Posterior')

plt.scatter(x_age_t,ypn_t, color='red', label='Negative Posterior')


plt.scatter(x_age_t, yt2_t, color='purple', label='Case 3 Threshold', marker = 4)

plt.scatter(x_age_t, yt3_t, color='yellow', label='Case 2 Threshold', marker = 4)

plt.scatter(x_age_t, yt1_t, color='black', label='Case 1 Threshold', marker = 4)


#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Posteriors & Thresholds (Testing Data)")

plt.xlabel("Age")

plt.ylabel("Posterior: P(Ci/x)")

plt.legend()

plt.grid(True)

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.xlabel("Age")


#Expected Risks


list_risk_positive_1_t = sorted(risk_positive_case_1_t.items())
```

```
xrp1_t,yrp1_t=zip(*list_risk_positive_1_t)


list_risk_negative_1_t = sorted(risk_negative_case_1_t.items())

xrn1_t,yrn1_t=zip(*list_risk_negative_1_t)


plt.scatter(x_age_t, yrp1_t, color='green', label='Positive E. Risk')

plt.scatter(x_age_t,yrn1_t, color='brown', label='Negative E. Risk')

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Expected Risks (Testing Data)")

plt.xlabel("Age")

plt.ylabel("Expected Risks: R(alpha'i/x)")

plt.legend()

plt.grid(True)



#Solo Expected Risk


plt.scatter(x_age_t, yrp1_t, color='green')

plt.scatter(x_age_t,yrn1_t, color='brown')

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.legend("Age", "Probability")

plt.grid(True)

###


plt.scatter(x_age_t, yt2_t, color='purple', label='Case 2 Threshold')

plt.scatter(x_age_t, yt3_t, color='black', label='Case 3 Threshold')

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Threshold of Decision for Minimum Expected Risk")
plt.xlabel("Age")

plt.ylabel("2=Positive for Case 2 & 1.8=Positive for Case 3\n1=Negative for Case 2 & 1.4 = Negative for Case 3")

plt.legend()

plt.grid(True)


#Plot Training + Test Posterior & Likelihoods Together

#training_posterior


plt.scatter(x_age, ypp, color='skyblue', label='Positive')

plt.scatter(x_age,ypn, color='red', label='Negative')

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2, linestyle='dashed', label="toto")

plt.title("Posteriors (Training Data)")

plt.xlabel("Age")

plt.ylabel("P(Ci/x)")

plt.legend()

plt.grid(True)


#training_likelihoods


plt.scatter(x_age, ylp, color='blue', label='Positive')

plt.scatter(x_age,yln, color='orange', label='Negative')

plt.title("Likelihoods (Training Data)")

#plt.plot( training_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2, linestyle='dashed', label="toto")

plt.legend()

plt.xlabel("Age")

plt.ylabel("P(x/Ci)")

plt.grid(True)
```

```
#testing_posteriors


plt.scatter(x_age_t, ypp_t, color='skyblue', label='Posterior Positive')

plt.scatter(x_age_t, ypn_t, color='red', label='Posterior Negative')

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.title("Posteriors (Testing Data)")

plt.xlabel("Age")

plt.ylabel("P(Ci/x)")

plt.legend()

plt.grid(True)


#testing_likelihoods


plt.scatter(x_age_t, ylp_t, color='blue', label='Likelihood Positive', marker = 4)

plt.scatter(x_age_t, yln_t, color='orange', label='Likelihood Negative', marker = 4)

plt.title("Posterior & Likelihoods (Testing Data)")

#plt.plot( testing_set["age"], 'y3', data=df, marker='', color='olive', linewidth=2,
linestyle='dashed', label="toto")

plt.legend()

plt.xlabel("Age")

plt.ylabel("Posterior: P(Ci/x) \n Likelihood: P(x/Ci)")

plt.grid(True)


#MIX


plt.plot(x_age, ypp, color='brown', label='Training Posterior Positive')

plt.plot(x_age,ypn, color='orange', label='Training Posterior Negative')


#plt.scatter(x_age, ylp, color='black', label='Training Likelihood Positive')
```

```
#plt.scatter(x_age,yln, color='grey', label='Training Likelihood Negative')


plt.scatter(x_age_t, ypp_t, color='skyblue', label='Testing Posterior Positive')

plt.scatter(x_age_t, ypn_t, color='red', label='Testing Posterior Negative')


#plt.scatter(x_age_t, ylp_t, color='green', label='Testing Likelihood Positive', marker = 4)

#plt.scatter(x_age_t, yln_t, color='purple', label='Testing Likelihood Negative', marker = 4)


plt.legend()

plt.xlabel("Age")

plt.ylabel("Posterior: P(Ci/x) \n Likelihood: P(x/Ci)")

plt.grid(True)
```