Emin Sadikhov **S017026**

Introduction to Machine Learning and Artificial Neural Networks

Parametric Classification

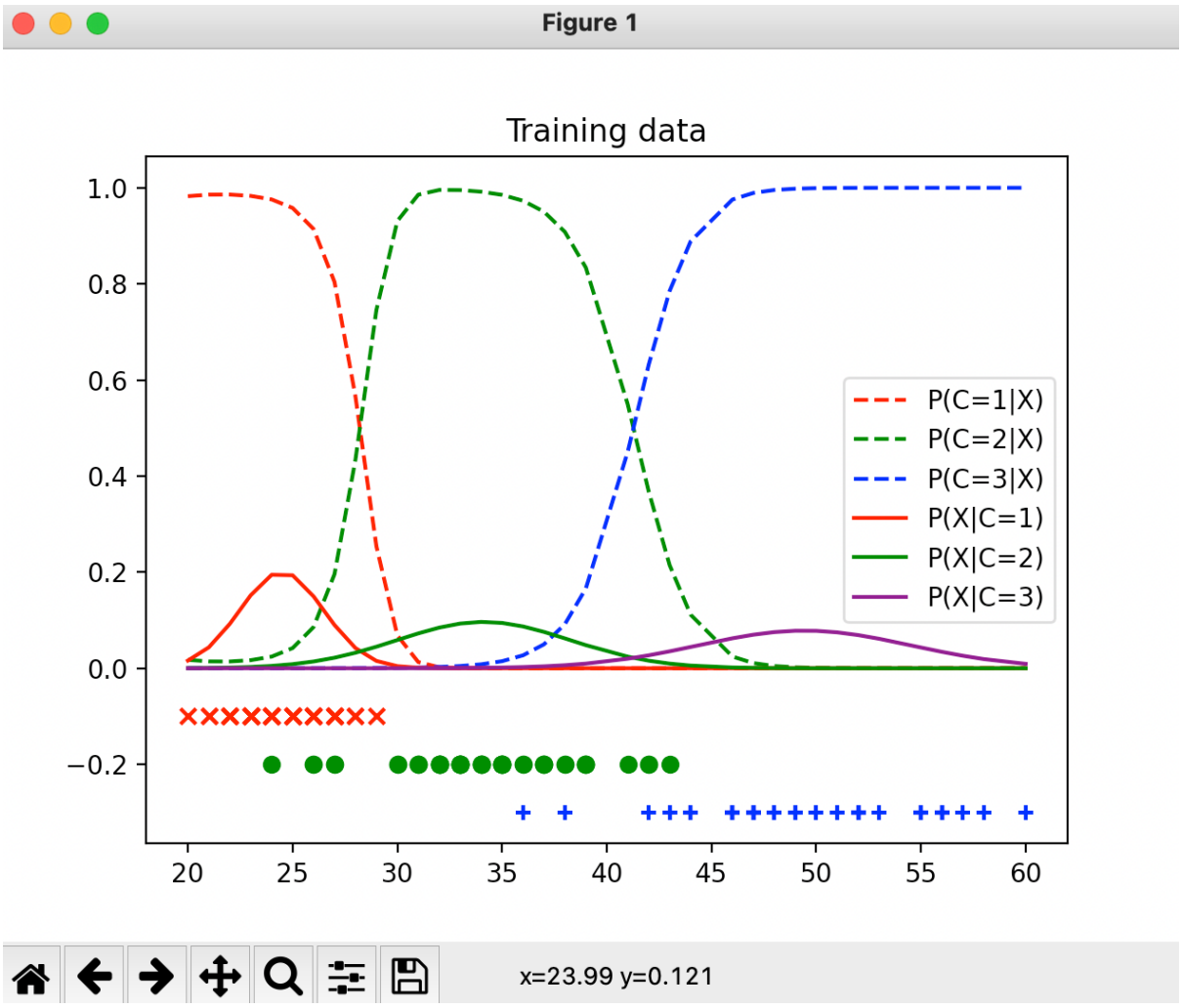
17 October 2021

Part 1

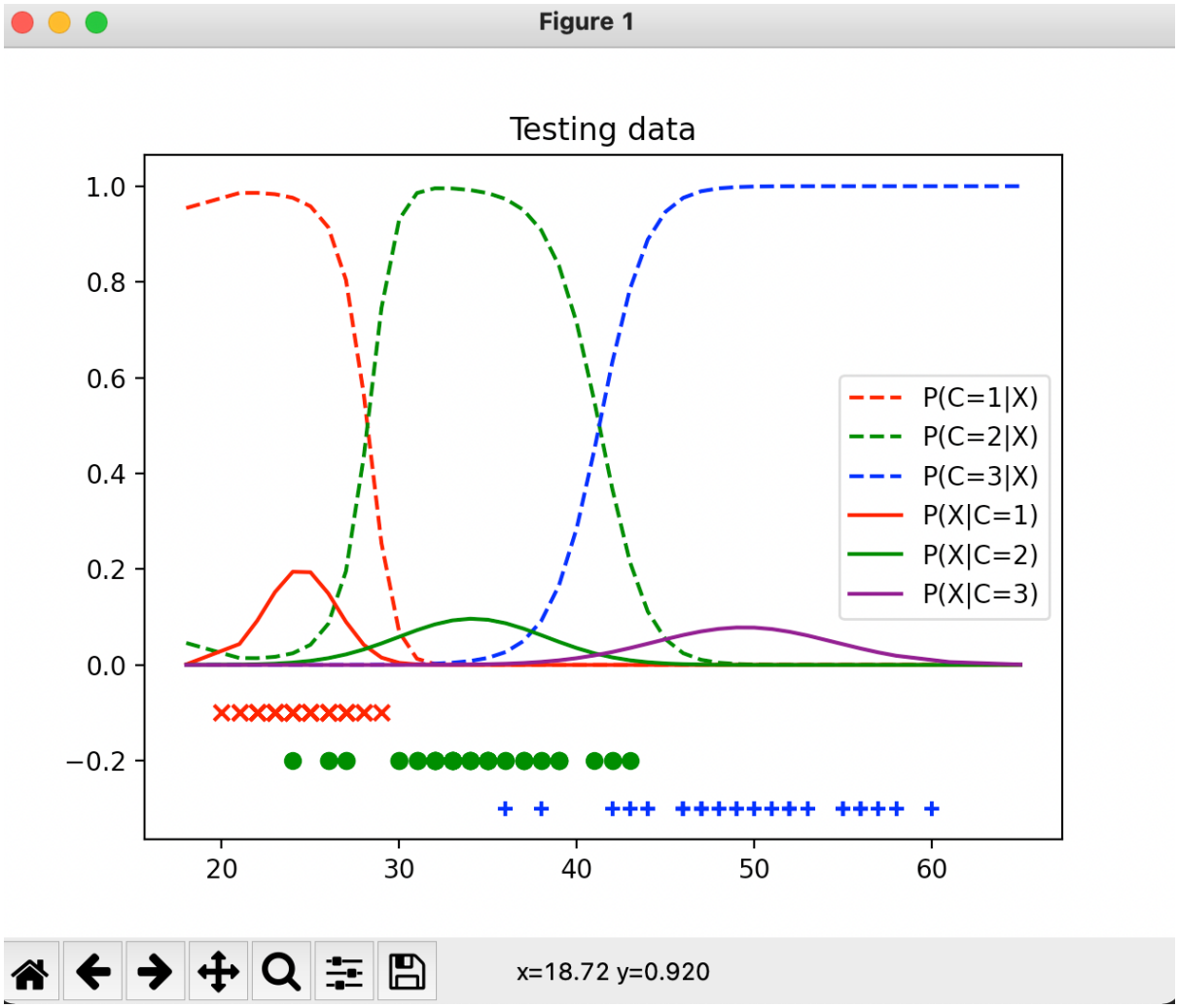
First, I started with the implementation of the homework. I read the given data to us and wrote a function to calculate mean, standard deviation, and priors. Below you can find my calculations for the training data.

```
mean  24.48 34.12 49.44
std  1.992385504865963 4.1358916813669095 5.091797325110258
priors  0.3333333333333333 0.3333333333333333 0.3333333333333333
```

After successfully calculation of likelihood and posteriors for the training data I started plotting the graph. I have used matplotlib library for plotting. Below is the graph of training data.



Later, I have plotted the graph for the test data. For it, we need to recalculate the likelihoods and posteriors of test data set. However, we will be using the mean, standard deviation, and priors that we calculated before for the training data set. Below is the plot for the testing data.



Question:

We assumed that the distributions of the samples(instances, observations) are Gaussian. When you plot the instances together with the likelihood functions does it seem to fit that distribution?

Answer:

To be a gaussian we should have a symmetric distribution around mean. As we can see from the graphs we obtained above, the probability density looks like a bell curve. Also, gaussian distribution is known also as a bell distribution. This means that it fits the Gaussian Distribution.

Part 2

For the first part of part 2, I calculated 3x3 confusion matrix. Below is the result of it for the training set.

Training	C1	C2	C3
α1 choose C1	49	5	0
α2 choose C2	1	42	2
α3 choose C3	0	3	48

Below is the result for a confusion matrix for the testing data set.

Testing	C1	C2	C3
α1 choose C1	48	7	0
α2 choose C2	2	41	0
α3 choose C3	0	2	50

Question:

You may have observed that the model made misclassifications during prediction. What can be the cause of that?

Answer:

From the obtained results, we can see the differences between the training and testing confusion matrix. The model made misclassifications. Since some of the instances are clashing, the model might make a misprediction. From the graphs obtained in the first part we see that there are some regions where red and green and green and blue clash. Main reason for the misclassification of model is that it is unable to correctly classify to which distribution the instance belongs to.

In the second part (b) of part 2, we needed to calculate 4x3 confusion matrix which included rejecting cases. Wrong classification had cost of 4 and cost of rejecting an instance was 1. Below is the loss table.

	C1	C2	C3
α1 choose C1	0	4	4
α2 choose C2	4	0	4
α3 choose C3	4	4	0
Reject	1	1	1

After creation of the loss table, I needed to calculate threshold for minimum expected risk. First, I calculated the risks.

$R(\alpha1|X) = 0 * P(C1|X) + 4 * P(C2|X) + 4 * P(C3|X) = 4 * P(C2|X) + 4 * P(C3|X)$
$R(\alpha2|X) = 4 * P(C1|X) + 0 * P(C2|X) + 4 * P(C3|X) = 4 * P(C1|X) + 4 * P(C3|X)$
$R(\alpha3|X) = 4 * P(C1|X) + 4 * P(C2|X) + 0 * P(C3|X) = 4 * P(C1|X) + 4 * P(C2|X)$
$R(reject|X) = 1 * P(C1|X) + 1 * P(C2|X) + 1 * P(C3|X) = P(C1|X) + P(C2|X) + P(C3|X) = 1$

One and four

$4*P(C2|X) + 4*P(C3|X) < 1$
$4P(C2|X) + 4P(C3|X) < 1$
$4(P(C2|X) + P(C3|X) - 1 < 0$
$4(1 - P(C1|X)) - 1 < 0$
$4 - 4P(C1|X) - 1 < 0$
P(C1\|X) > ¾

One and three

$4*P(C2|X) + 4*P(C3|X) < 4*P(C1|X) + 4*P(C2|X)$
$4*P(C3|X) < 4*P(C1|X)$
$P(C3|X) < P(C1|X)$

One and Two

$4*P(C2|X) + 4*P(C3|X) < 4* P(C1|X) + 4* P(C3|X)$
$P(C2|X) < P(C1|X)$

After finding the threshold I wrote a program and below are the results of confusion matrix for training and testing data sets.

Training	C1	C2	C3
α1 choose C1	47	5	0
α2 choose C2	0	41	2
α3 choose C3	0	1	47
Reject	3	3	1

Testing	C1	C2	C3
α1 choose C1	47	5	0
α2 choose C2	0	35	0
α3 choose C3	0	2	48
Reject	3	8	2

Question:

You may have observed some rejected instances during classification in Part 2.b of the homework. Why are those instances rejected?

Answer:

It maybe because of lack of enough data to put it to some class. The model is unsure about which class the instance belongs and instead of making a misclassification which would eventually give a loss, it decided to reject it with less loss.

You can find the source code from this link: https://github.com/SadikhovEmin/CS454

```
from csv import reader
import math
import matplotlib.pyplot as plt


def mean(age):
    values = 0
    count = 0
    for row in age:
        values += int(row)
        count += 1

    return values / count


def square(x):
    return float(x) * float(x)


def std(age):
    average = mean(age)
    result = 0.0  # The result
    count = 0.0  # Count number of rows for particular class index
    for row in age:
        result += square(float(row) - average)
        count += 1.0

    return math.sqrt(result / count)


def priors(class_age, total_count):
    return class_age / total_count


def likelihoods(age, class_mean, class_std):
    list_of_likelihoods = []

    for i in age:
        likelihoods_class_1 = 1.0 / (class_std * math.sqrt(2.0 * math.pi)) * math.ex
p(
            -1.0 * ((square(float(i) - class_mean)) / (2.0 * square(class_std))))
        list_of_likelihoods.append(likelihoods_class_1)  # Appending tuple

    return list_of_likelihoods


def posteriors(age, class_1_likelihood, class_2_likelihood, class_3_likelihood, clas
s1_prior, class2_prior,
               class3_prior):
    list_of_posteriors = []
    count = 0

    for i in age:
        posterior_class_1 = (class_1_likelihood[count] * class1_prior) / (
                (class_1_likelihood[count] * class1_prior) + (class_2_likelihood[cou
nt] * class2_prior) + (
                class_3_likelihood[count] * class3_prior))
        list_of_posteriors.append(posterior_class_1)
        count += 1

    return list_of_posteriors


with open('training.csv') as training_file:
    csv_reader = reader(training_file)
    list_of_rows = list(csv_reader)
    print(list_of_rows)
    list_of_rows.pop(0)
    print(list_of_rows)  # This is the data without including the 0 row (age and cla
ss)

    age_class_1 = []
    age_class_2 = []
```

```
    age_class_3 = []

    age = []

    # Stores the number of ages in class1 2 and 3 and total
    class1_count_age, class2_count_age, class3_count_age, total_count_age = 0, 0, 0,
 0

    for i in list_of_rows:
        if i[1] == "1":
            age_class_1.append(int(i[0]))
            age.append(int(i[0]))
            class1_count_age += 1
            total_count_age += 1
        elif i[1] == "2":
            age_class_2.append(int(i[0]))
            age.append(int(i[0]))
            class2_count_age += 1
            total_count_age += 1
        elif i[1] == "3":
            age_class_3.append(int(i[0]))
            age.append(int(i[0]))
            class3_count_age += 1
            total_count_age += 1

    # Means
    mean_class_1, mean_class_2, mean_class_3 = mean(age_class_1), mean(age_class_2),
 mean(age_class_3)

    # STD
    std_class_1, std_class_2, std_class_3 = std(age_class_1), std(age_class_2), std(
age_class_3)

    # Priors
    class_1_priors, class_2_priors, class_3_priors = priors(class1_count_age, total_
count_age), priors(class2_count_age,

                    total_count_age), priors(
        class3_count_age, total_count_age)

    # Likelihoods
    class_1_likelihood, class_2_likelihood, class_3_likelihood = likelihoods(age, me
an_class_1,
                                                                             std_cla
ss_1), likelihoods(age,

                    mean_class_2,

                    std_class_2), likelihoods(
        age, mean_class_3, std_class_3)

    # Posteriors
    class_1_posteriors, class_2_posteriors, class_3_posteriors = posteriors(age, cla
ss_1_likelihood, class_2_likelihood,
                                                                            class_3_
likelihood, class_1_priors,
                                                                            class_2_
priors,
                                                                            class_3_
priors), posteriors(age,

                     class_2_likelihood,

                     class_1_likelihood,

                     class_3_likelihood,

                     class_1_priors,

                     class_2_priors,

                     class_3_priors), posteriors(
        age, class_3_likelihood, class_2_likelihood, class_1_likelihood, class_1_pri
```

```
ors, class_2_priors,
        class_3_priors)

    print('mean ', mean_class_1, mean_class_2, mean_class_3)
    print('std ', std(age_class_1), std(age_class_2), std(age_class_3))
    print('priors ', class_1_priors, class_2_priors, class_3_priors)
    print('age_class_1 ', age_class_1)
    print('age_class_1 len ', len(age_class_1))
    print('age_class_2 ', age_class_2)
    print('age_class_2 len ', len(age_class_2))
    print('age_class_3 ', age_class_3)
    print('age_class_3 len ', len(age_class_3))
    print('likelihoods class 1', class_1_likelihood)
    print('likelihoods class 2', class_2_likelihood)
    print('likelihoods class 3', class_3_likelihood)
    print('posteriors class 1', class_1_posteriors)
    print('posteriors class 2', class_2_posteriors)
    print('posteriors class 3', class_3_posteriors)

    ''' PLOTTING '''
    age_set = set()
    for i in list_of_rows:
        age_set.add(int(i[0]))

    # LIKELIHOODS
    plot_likelihood_class_1 = likelihoods(list(age_set), mean(age_class_1), std(age_
class_1))
    plot_likelihood_class_2 = likelihoods(list(age_set), mean(age_class_2), std(age_
class_2))
    plot_likelihood_class_3 = likelihoods(list(age_set), mean(age_class_3), std(age_
class_3))

    # POSTERIORS
    plot_posterior_class_1 = posteriors(list(age_set), plot_likelihood_class_1, plot
_likelihood_class_2,
                                        plot_likelihood_class_3, class_1_priors, cla
ss_2_priors, class_3_priors)
    plot_posterior_class_2 = posteriors(list(age_set), plot_likelihood_class_2, plot
_likelihood_class_1,
                                        plot_likelihood_class_3, class_1_priors, cla
ss_2_priors, class_3_priors)
    plot_posterior_class_3 = posteriors(list(age_set), plot_likelihood_class_3, plot
_likelihood_class_2,
                                        plot_likelihood_class_1, class_1_priors, cla
ss_2_priors, class_3_priors)

    plt.plot(list(age_set), plot_posterior_class_1, color='red', linestyle='dashed')
    plt.plot(list(age_set), plot_posterior_class_2, color='green', linestyle='dashed
')
    plt.plot(list(age_set), plot_posterior_class_3, color='blue', linestyle='dashed'
)

    plt.plot(list(age_set), plot_likelihood_class_1, color='red')
    plt.plot(list(age_set), plot_likelihood_class_2, color='green')
    plt.plot(list(age_set), plot_likelihood_class_3, color='purple')
    #
    plt.title("Training data")
    plt.legend(["P(C=1|X)", "P(C=2|X)", "P(C=3|X)", "P(X|C=1)", "P(X|C=2)", "P(X|C=3
)"])

    plt.scatter(age_class_1, [-0.1] * class1_count_age, color='r', marker="x")
    plt.scatter(age_class_2, [-0.2] * class2_count_age, color='g', marker="o")
    plt.scatter(age_class_3, [-0.3] * class3_count_age, color='b', marker="+")

    plt.show()

    ''' PART 2 '''
    print("Part 2 a training")
    matrix_a = [
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]
    ]
```

```
    count = 0
    for i in list_of_rows:
        max_prediction = max(class_1_posteriors[count], class_2_posteriors[count], c
lass_3_posteriors[count])

        if max_prediction == class_1_posteriors[count]:
            if int(i[1]) == 1:
                matrix_a[0][0] += 1
            elif int(i[1]) == 2:
                matrix_a[0][1] += 1
            elif int(i[1]) == 3:
                matrix_a[0][2] += 1
        elif max_prediction == class_2_posteriors[count]:
            if int(i[1]) == 1:
                matrix_a[1][0] += 1
            elif int(i[1]) == 2:
                matrix_a[1][1] += 1
            elif int(i[1]) == 3:
                matrix_a[1][2] += 1
        elif max_prediction == class_3_posteriors[count]:
            if int(i[1]) == 1:
                matrix_a[2][0] += 1
            elif int(i[1]) == 2:
                matrix_a[2][1] += 1
            elif int(i[1]) == 3:
                matrix_a[2][2] += 1
        count += 1

    print(*matrix_a, sep='\n')

    print("PART 2 b training")

    matrix_b = [
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]
    ]

    count = 0
    for i in list_of_rows:
        max_prediction = max(class_1_posteriors[count], class_2_posteriors[count], c
lass_3_posteriors[count])

        if class_1_posteriors[count] > 0.75:
            if int(i[1]) == 1:
                matrix_b[0][0] += 1
            elif int(i[1]) == 2:
                matrix_b[0][1] += 1
            elif int(i[1]) == 3:
                matrix_b[0][2] += 1
        elif class_2_posteriors[count] > 0.75:
            if int(i[1]) == 1:
                matrix_b[1][0] += 1
            elif int(i[1]) == 2:
                matrix_b[1][1] += 1
            elif int(i[1]) == 3:
                matrix_b[1][2] += 1
        elif class_3_posteriors[count] > 0.75:
            if int(i[1]) == 1:
                matrix_b[2][0] += 1
            elif int(i[1]) == 2:
                matrix_b[2][1] += 1
            elif int(i[1]) == 3:
                matrix_b[2][2] += 1
        else:
            if int(i[1]) == 1:
                matrix_b[3][0] += 1
            elif int(i[1]) == 2:
                matrix_b[3][1] += 1
            elif int(i[1]) == 3:
                matrix_b[3][2] += 1
```

```
        count += 1

    print(*matrix_b, sep='\n')

''' TESTING PLOT '''
with open('testing.csv') as testing_file:
    csv_reader = reader(testing_file)
    list_of_rows_test = list(csv_reader)
    print(list_of_rows_test)
    list_of_rows_test.pop(0)
    print(list_of_rows_test)  # This is the data without including the 0 row (age an
d class)

    age_testing = []

    for i in list_of_rows_test:
        if i[1] == "1":
            age_testing.append(int(i[0]))
        elif i[1] == "2":
            age_testing.append(int(i[0]))
        elif i[1] == "3":
            age_testing.append(int(i[0]))

        age_set_testing = set()
        for i in list_of_rows_test:
            age_set_testing.add(int(i[0]))

    # LIKELIHOODS
    plot_likelihood_class_1_testing = likelihoods(list(age_set_testing), mean(age_cl
ass_1), std(age_class_1))
    plot_likelihood_class_2_testing = likelihoods(list(age_set_testing), mean(age_cl
ass_2), std(age_class_2))
    plot_likelihood_class_3_testing = likelihoods(list(age_set_testing), mean(age_cl
ass_3), std(age_class_3))

    # POSTERIORS
    plot_posterior_class_1_testing = posteriors(list(age_set_testing), plot_likeliho
od_class_1_testing,
                                                plot_likelihood_class_2_testing,
                                                plot_likelihood_class_3_testing, cla
ss_1_priors, class_2_priors,
                                                class_3_priors)
    plot_posterior_class_2_testing = posteriors(list(age_set_testing), plot_likeliho
od_class_2_testing,
                                                plot_likelihood_class_1_testing,
                                                plot_likelihood_class_3_testing, cla
ss_1_priors, class_2_priors,
                                                class_3_priors)
    plot_posterior_class_3_testing = posteriors(list(age_set_testing), plot_likeliho
od_class_3_testing,
                                                plot_likelihood_class_2_testing,
                                                plot_likelihood_class_1_testing, cla
ss_1_priors, class_2_priors,
                                                class_3_priors)

    plt.plot(list(age_set_testing), plot_posterior_class_1_testing, color='red', lin
estyle='dashed')
    plt.plot(list(age_set_testing), plot_posterior_class_2_testing, color='green', l
inestyle='dashed')
    plt.plot(list(age_set_testing), plot_posterior_class_3_testing, color='blue', li
nestyle='dashed')

    plt.plot(list(age_set_testing), plot_likelihood_class_1_testing, color='red')
    plt.plot(list(age_set_testing), plot_likelihood_class_2_testing, color='green')
    plt.plot(list(age_set_testing), plot_likelihood_class_3_testing, color='purple')

    plt.title("Testing data")
    plt.legend(["P(C=1|X)", "P(C=2|X)", "P(C=3|X)", "P(X|C=1)", "P(X|C=2)", "P(X|C=3
)"])

    plt.scatter(age_class_1, [-0.1] * class1_count_age, color='r', marker="x")
    plt.scatter(age_class_2, [-0.2] * class2_count_age, color='g', marker="o")
    plt.scatter(age_class_3, [-0.3] * class3_count_age, color='b', marker="+")
```

```
    plt.show()

    class_1_likelihood_testing, class_2_likelihood_testing, class_3_likelihood_testi
ng = likelihoods(age_testing,

                 mean_class_1,

                 std_class_1), likelihoods(
        age_testing,
        mean_class_2,
        std_class_2), likelihoods(
        age_testing, mean_class_3, std_class_3)

    # Posteriors
    class_1_posteriors_testing, class_2_posteriors_testing, class_3_posteriors_testi
ng = posteriors(age_testing,

                class_1_likelihood_testing,

                class_2_likelihood_testing,

                class_3_likelihood_testing,

                class_1_priors,

                class_2_priors,

                class_3_priors), posteriors(
        age_testing,
        class_2_likelihood_testing,
        class_1_likelihood_testing,
        class_3_likelihood_testing,
        class_1_priors,
        class_2_priors,
        class_3_priors), posteriors(
        age_testing, class_3_likelihood_testing, class_2_likelihood_testing, class_1
_likelihood_testing, class_1_priors,
        class_2_priors,
        class_3_priors)

    ''' PART 2 '''
    print("Part 2 a testing")
    matrix_a_testing = [
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]
    ]

    count = 0
    for i in list_of_rows_test:
        max_prediction = max(class_1_posteriors_testing[count], class_2_posteriors_t
esting[count],
                             class_3_posteriors_testing[count])

        if max_prediction == class_1_posteriors_testing[count]:
            if int(i[1]) == 1:
                matrix_a_testing[0][0] += 1
            elif int(i[1]) == 2:
                matrix_a_testing[0][1] += 1
            elif int(i[1]) == 3:
                matrix_a_testing[0][2] += 1
        elif max_prediction == class_2_posteriors_testing[count]:
            if int(i[1]) == 1:
                matrix_a_testing[1][0] += 1
            elif int(i[1]) == 2:
                matrix_a_testing[1][1] += 1
            elif int(i[1]) == 3:
                matrix_a_testing[1][2] += 1
        elif max_prediction == class_3_posteriors_testing[count]:
            if int(i[1]) == 1:
                matrix_a_testing[2][0] += 1
            elif int(i[1]) == 2:
```

```
                matrix_a_testing[2][1] += 1
            elif int(i[1]) == 3:
                matrix_a_testing[2][2] += 1
        count += 1

    print(*matrix_a_testing, sep='\n')

    print("PART 2 b testing")

    matrix_b_testing = [
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]
    ]

    count = 0
    for i in list_of_rows_test:
        max_prediction = max(class_1_posteriors_testing[count], class_2_posteriors_t
esting[count],
                             class_3_posteriors_testing[count])

        if class_1_posteriors_testing[count] > 0.75:
            if int(i[1]) == 1:
                matrix_b_testing[0][0] += 1
            elif int(i[1]) == 2:
                matrix_b_testing[0][1] += 1
            elif int(i[1]) == 3:
                matrix_b_testing[0][2] += 1
        elif class_2_posteriors_testing[count] > 0.75:
            if int(i[1]) == 1:
                matrix_b_testing[1][0] += 1
            elif int(i[1]) == 2:
                matrix_b_testing[1][1] += 1
            elif int(i[1]) == 3:
                matrix_b_testing[1][2] += 1
        elif class_3_posteriors_testing[count] > 0.75:
            if int(i[1]) == 1:
                matrix_b_testing[2][0] += 1
            elif int(i[1]) == 2:
                matrix_b_testing[2][1] += 1
            elif int(i[1]) == 3:
                matrix_b_testing[2][2] += 1
        else:
            if int(i[1]) == 1:
                matrix_b_testing[3][0] += 1
            elif int(i[1]) == 2:
                matrix_b_testing[3][1] += 1
            elif int(i[1]) == 3:
                matrix_b_testing[3][2] += 1
        count += 1

    print(*matrix_b_testing, sep='\n')
```