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

def read_csv_file(file_name):
    """
    Reads a csv file after second line and returns list of dictionaries with the dat
a. Initial values of predictions are -1
    """
    numbers = []
    with open(file_name, 'r') as csv_file:
        csv_reader = csv.reader(csv_file)
        next(csv_reader)
        for item in list(csv_reader):
            my_dict = { 'value': int(item[0]), 'class': int(item[1]) , 'prediction_z
ero_one' : -1, 'prediction_reject' : -1}
            numbers.append(my_dict)
    return numbers


def calculate_appearances(numbers, class_id):
    """
    Calculates how many times an item that belongs to a class appears in a list
    """
    appear = 0.0
    for dict in numbers:
        if dict['class'] == class_id:
            appear += 1.0
    return appear


def sum_numbers(numbers, class_id = None):
    """
    Sums all the numbers in a list
    """
    sum = 0.0
    if class_id == None:
        for number in numbers:
            sum += float(number)
    else:
        interested_class = [x for x in numbers if x['class'] == class_id]
        for number in interested_class:
            sum += float(number['value'])
    return sum


def my_mean(numbers, class_id = None):
    """
    Calculates the mean of a list of numbers
    """
    if class_id == None:
        return sum_numbers(numbers) / float(len(numbers))
    else:
        return sum_numbers(numbers, class_id) / calculate_appearances(numbers, class
_id)


def standard_deviation(numbers, class_id = None):
    """
    Calculates the standard deviation of a list of numbers
    """
    average = 0.0
    variance = 0.0
    if class_id == None:
        average = my_mean(numbers)
        variance = sum_numbers([float((x['value']) - average) ** 2 for x in numbers]
) / float(len(numbers))
    else:
        average = my_mean(numbers, class_id)
        variance = sum_numbers([(float(x['value']) - average) ** 2 for x in numbers
if x['class'] == class_id]) / calculate_appearances(numbers, class_id)
    return variance ** 0.5
```

```
def calculate_likelihood(number, mean, std_dev):
    """
    Calculates the likelihood
    """
    likelihood = 1.0 / (std_dev * (2.0 * math.pi) ** 0.5) * (math.exp(-((number - me
an) ** 2) / (2.0 * std_dev ** 2)))
    return likelihood


numbers = read_csv_file("training.csv")
numbers_one   = [x['value'] for x in numbers if x['class'] == 1]
numbers_two   = [x['value'] for x in numbers if x['class'] == 2]
numbers_three = [x['value'] for x in numbers if x['class'] == 3]

count = float(len(numbers))

prior_class_1 = calculate_appearances(numbers, 1) / count
prior_class_2 = calculate_appearances(numbers, 2) / count
prior_class_3 = calculate_appearances(numbers, 3) / count

print("Priors    : ",prior_class_1, prior_class_2, prior_class_3)

mean_class_1 = my_mean(numbers, 1)
mean_class_2 = my_mean(numbers, 2)
mean_class_3 = my_mean(numbers, 3)

print("Averages  : ",mean_class_1,mean_class_2,mean_class_3)

std_class_1 = standard_deviation(numbers, 1)
std_class_2 = standard_deviation(numbers, 2)
std_class_3 = standard_deviation(numbers, 3)

print("Std dev.  : ",std_class_1,std_class_2,std_class_3, end='\n\n')

likelihoods_class_1 = [calculate_likelihood(numbers[i]['value'], mean_class_1, std_c
lass_1) for i in range(len(numbers))]
likelihoods_class_2 = [calculate_likelihood(numbers[i]['value'], mean_class_2, std_c
lass_2) for i in range(len(numbers))]
likelihoods_class_3 = [calculate_likelihood(numbers[i]['value'], mean_class_3, std_c
lass_3) for i in range(len(numbers))]

print("Likelihoods [:5]",likelihoods_class_1[:5],likelihoods_class_2[:5],likelihoods
_class_3[:5], sep='\n',end='\n\n')

posterior_ones      = [prior_class_1 * likelihoods_class_1[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(numbers))]
posterior_twos      = [prior_class_2 * likelihoods_class_2[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(numbers))]
posterior_threes    = [prior_class_3 * likelihoods_class_3[i] / (likelihoods_class_1
[i] * prior_class_1 + likelihoods_class_2[i] * prior_class_2 + likelihoods_class_3[i
] * prior_class_3) for i in range(len(numbers))]

print("Posteriors [:5]", posterior_ones[:5], posterior_twos[:5], posterior_threes[:5
], sep='\n',end='\n\n')

for i in range(len(numbers)):

    if posterior_ones[i] > posterior_twos[i] and posterior_ones[i] > posterior_three
s[i]:
        numbers[i]['prediction_zero_one'] = 1
    elif posterior_twos[i] > posterior_ones[i] and posterior_twos[i] > posterior_thr
ees[i]:
        numbers[i]['prediction_zero_one'] = 2
    elif posterior_threes[i] > posterior_ones[i] and posterior_threes[i] > posterior
_twos[i]:
        numbers[i]['prediction_zero_one'] = 3

    if posterior_ones[i] > 0.75:
        numbers[i]['prediction_reject'] = 1
    elif posterior_twos[i] > 0.75:
```

```
        numbers[i]['prediction_reject'] = 2
    elif posterior_threes[i] > 0.75:
        numbers[i]['prediction_reject'] = 3
    else:
        numbers[i]['prediction_reject'] = 4


confusion_matrix = [[0,0,0],[0,0,0],[0,0,0]]
confusion_matrix_r = [[0,0,0],[0,0,0],[0,0,0],[0,0,0]]
for item in numbers:
    confusion_matrix[item['prediction_zero_one']-1][item['class']-1] += 1
    confusion_matrix_r[item['prediction_reject']-1][item['class']-1] += 1

print("Confusion Matrix Training 0-1",*confusion_matrix, sep='\n', end='\n\n')
print("Confusion Matrix Training w/ Rejection",*confusion_matrix_r, sep='\n', end='\
n\n')



test_list = read_csv_file("testing.csv")
test_list_one   = [x['value'] for x in test_list if x['class'] == 1]
test_list_two   = [x['value'] for x in test_list if x['class'] == 2]
test_list_three = [x['value'] for x in test_list if x['class'] == 3]

likelihoods_class_1_test = [calculate_likelihood(test_list[i]['value'], mean_class_1
, std_class_1) for i in range(len(test_list))]
likelihoods_class_2_test = [calculate_likelihood(test_list[i]['value'], mean_class_2
, std_class_2) for i in range(len(test_list))]
likelihoods_class_3_test = [calculate_likelihood(test_list[i]['value'], mean_class_3
, std_class_3) for i in range(len(test_list))]

posterior_ones_test      = [prior_class_1 * likelihoods_class_1_test[i] / (likelihoo
ds_class_1_test[i] * prior_class_1 + likelihoods_class_2_test[i] * prior_class_2 + l
ikelihoods_class_3_test[i] * prior_class_3) for i in range(len(test_list))]
posterior_twos_test      = [prior_class_2 * likelihoods_class_2_test[i] / (likelihoo
ds_class_1_test[i] * prior_class_1 + likelihoods_class_2_test[i] * prior_class_2 + l
ikelihoods_class_3_test[i] * prior_class_3) for i in range(len(test_list))]
posterior_threes_test    = [prior_class_3 * likelihoods_class_3_test[i] / (likelihoo
ds_class_1_test[i] * prior_class_1 + likelihoods_class_2_test[i] * prior_class_2 + l
ikelihoods_class_3_test[i] * prior_class_3) for i in range(len(test_list))]

for i in range(len(test_list)):
    if posterior_ones_test[i] > posterior_twos_test[i] and posterior_ones_test[i] >
posterior_threes_test[i]:
        test_list[i]['prediction_zero_one'] = 1
    elif posterior_twos_test[i] > posterior_ones_test[i] and posterior_twos_test[i]
> posterior_threes_test[i]:
        test_list[i]['prediction_zero_one'] = 2
    elif posterior_threes_test[i] > posterior_ones_test[i] and posterior_threes_test
[i] > posterior_twos_test[i]:
        test_list[i]['prediction_zero_one'] = 3

    if posterior_ones_test[i] > 0.75:
        test_list[i]['prediction_reject'] = 1
    elif posterior_twos_test[i] > 0.75:
        test_list[i]['prediction_reject'] = 2
    elif posterior_threes_test[i] > 0.75:
        test_list[i]['prediction_reject'] = 3
    else :
        test_list[i]['prediction_reject'] = 4

confusion_matrix_test = [[0,0,0],[0,0,0],[0,0,0]]
confusion_matrix_test_r = [[0,0,0],[0,0,0],[0,0,0],[0,0,0]]

for item in test_list:
    confusion_matrix_test[item['prediction_zero_one']-1][item['class']-1] += 1
    confusion_matrix_test_r[item['prediction_reject']-1][item['class']-1] += 1

print("Confusion Matrix Test 0-1",*confusion_matrix_test, sep='\n', end='\n\n')
print("Confusion Matrix Test w/ Rejection",*confusion_matrix_test_r, sep='\n', end='
\n\n')
```