Homework 3: Continuous Variables & Classification

In [5]:
```
pip install scipy
```

```
Requirement already satisfied: scipy in c:\users\danny nguyen\anaconda3\lib\site-packages (1.9.1)Note: you may need
to restart the kernel to use updated packages.

Requirement already satisfied: numpy<1.25.0,>=1.18.5 in c:\users\danny nguyen\anaconda3\lib\site-packages (from scip
y) (1.21.5)
```

In [6]:
```
import numpy as np
from scipy.stats import norm
```

Question 1 (15 points)

a) *What is the probability that a random program has a runtime greater than 22 seconds?*

In [26]:
```
ans = 1 - norm.cdf(22, loc=15, scale=3)

print(f"The probability that a random program has a runtime between 11 and 19 seconds is {ans}.")
```

```
The probability that a random program has a runtime between 11 and 19 seconds is 0.009815328628645315.
```

b) *What is the probability that a random program has a runtime between 11 and 19 seconds?*

In [27]:
```
ans = norm.cdf(19, loc=15, scale=3) - (norm.cdf(11, loc=15, scale=3))

print(f"The probability that a random program has a runtime between 11 and 19 seconds is {ans}.")
```

```
The probability that a random program has a runtime between 11 and 19 seconds is 0.8175775605482642.
```

c) *The TA's want to help the students complete their work faster. What would they have to lower the average runtime to so that only 1.0% of students have runtimes over 15 seconds? Assume the standard deviation remains fixed at σ = 3.0 seconds.*

In [28]:
```
X = norm(loc=15, scale=3)
Z = norm.ppf(0.99)
sigma = 3
mu = 15 - 3 * Z

print(f"They would have to lower the average runtime to {mu}.")
```

```
They would have to lower the average runtime to 8.020956377877479.
```

Question 2 (25 points)

a) *The total time that it takes all 5 students ahead of you to receive help from the TA is a random variable. What is the mean of this total time?*

$$(6 * 0.7 + 9 * 0.3) * 5 = 34.5$$

Therefore, the mean total time is 34.5 minutes.

b) *What is the standard deviation of the total time taken by the 5 students ahead of you?*

$$\sqrt{5 * (6 * 0.7 + 9 * 0.3)^2} \approx 15.43$$

Therefore, the standard deviation of the total time taken is 15.43 minutes.

c) *It takes the TA 10 minutes to help the first student. Given this observation, what is the posterior probability that the TA remembered to get coffee?*

Y is whether or not the TA remembered to get coffee.

$P(Y = 1|X = 10) \approx 0.668$

$$\frac{0.7\theta_1 e^{-\theta_1 t}}{0.7\theta_1 e^{-\theta_1 t} + 0.3\theta_0 e^{-\theta_0 t}} = 0.66756832913$$

$\theta_1 = \frac{1}{6}$ $\theta_1 = 0.166666666667$

$t = 10$

−10 10

$\theta_0 = \frac{1}{9}$ $\theta_0 = 0.111111111111$

Screenshot of input into calculator (Desmos):

d) *Suppose that the TA has finished helping the first 4 students, and by looking at their desk, you see that they did remember to get coffee. If they have already spent 12 minutes with the fifth student, what is the mean and standard deviation of the amount of additional time you need to wait for help?*

$$\mu = E[X] = \frac{1}{\theta} = \frac{1}{\frac{1}{6}} = 6$$

$$\sigma = \sqrt{Var[X]} = \sqrt{\frac{1}{\theta^2}} = \sqrt{\frac{1}{\frac{1}{6}^2}} = 6$$

Because exponential distributions are memoryless, our mean time to wait is still 6 minutes, and the standard deviation is also 6 minutes.

Question 3 (20 points)

a) *Suppose that a new fabrication process has just been deployed, and the probability that the factory manufactures correctly functioning processors is only P (Y = 0) = 0.5. What threshold c of the observed test suite time X = x maximizes the probability that your prediction is correct?*

Y = 0 if

$$0.5 * ce^{-x} \geq 0.5 * \frac{1}{40} * ce^{-\frac{c}{40}}$$
$$= e^{\frac{-39c}{40}} \geq \frac{1}{40}$$
$$= \frac{-39c}{40} \geq \ln \frac{1}{40}$$
$$= c \leq \ln \frac{1}{40} * 40 \ / \ -39$$
$$= c \leq 3.78$$

b) *Suppose that after some improvements to the new fabrication process, the probability that the factory manufactures correctly functioning processors increases to P (Y = 0) = 0.95. What threshold c of the observed test suite time X = x maximizes the probability that your prediction is correct?*

Y = 0 if

$$0.95e^{-c} \geq 0.05 * \frac{1}{40} * e^{\frac{-c}{40}}$$
$$= 760e^{\frac{-39c}{40}} \geq 1$$
$$= \frac{-39c}{40} \geq \ln \frac{1}{760}$$
$$= c \leq \ln \frac{1}{760} * 40 \, / \, -39$$
$$= c \leq 6.80$$

c) *Market research suggests that the loss (or cost) of a missed detection (predicting Y = 0 when the processor is actually defective) is 500 times greater than the loss of a false alarm (predicting Y = 1 when the processor was correctly manufactured). Assuming again that P (Y = 0) = 0.95, what threshold c of the observed test suite time X = x minimizes the expected loss?*

$$L(1,0) = 500 * L(0,1)$$
$$\lambda_{10} = 500\lambda_{01}$$
$$= \lambda_{01} * 0.05 * \frac{1}{40} * e^{\frac{-c}{40}} \leq \lambda_{10} * 0.95e^{-c}$$
$$= \lambda_{01} * 0.05 * \frac{1}{40} * e^{\frac{-c}{40}} \leq 500\lambda_{01} * 0.95e^{-c}$$
$$= \frac{25}{40} * e^{\frac{-c}{40}} \leq 0.95e^{-c}$$
$$= \frac{25}{40} \leq 0.95e^{\frac{-39c}{40}}$$
$$= 0.657894737 \leq e^{\frac{-39c}{40}}$$
$$= -0.418710335 \leq \frac{-39c}{40}$$
$$= c \leq 0.43$$

Question 4: (40 points)

a) *Derive equations for* $\ln f_{X|Y}(x_i|1)$ *and* $\ln f_{X|Y}(x_i|0)$*, the (natural) logarithms of the conditional probability density functions in Equations (2,3). For numerical robustness, simplify your answer so that it does not involve the exponential function.*

$$\begin{aligned}
&\ln f_{X|Y}(x_i|1) \\
&= \ln \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{1j}^2}} exp\left\{ -\frac{(x_{ij}-\mu_{1j})^2}{2\sigma_{1j}^2} \right\} \\
&= \sum_{j=1}^{M} \ln \frac{1}{\sqrt{2\pi\sigma_{1j}^2}} - \frac{(x_{ij}-\mu_{1j})^2}{2\sigma_{1j}^2\sqrt{2\pi\sigma_{1j}^2}}
\end{aligned}$$

$$\begin{aligned}
&\ln f_{X|Y}(x_i|0) \\
&= \ln \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_{0j}^2}} exp\left\{ -\frac{(x_{ij}-\mu_{0j})^2}{2\sigma_{0j}^2} \right\} \\
&= \sum_{j=1}^{M} \ln \frac{1}{\sqrt{2\pi\sigma_{0j}^2}} - \frac{(x_{ij}-\mu_{0j})^2}{2\sigma_{0j}^2\sqrt{2\pi\sigma_{0j}^2}}
\end{aligned}$$

b) *Start by assuming the classes are equally probable* $(p_Y(1) = p_Y(0) = 1/2)$*, and have unit variance* $(\sigma_{1j}^2 = \sigma_{0j}^2 = 1)$*. Write code to compute the log conditional densities from part (a). Then using Equation (1), classify each test example. Report your classification accuracy, and the numbers of false alarms and missed detections. Hint: Your classifer should have fewer than 10 missed detections.*

In [79]:
```
# Load data
ozone = np.load('ozone.npz')
trainFeat, trainLabels = ozone['trainFeat'], ozone['trainLabels']
testFeat, testLabels = ozone['testFeat'], ozone['testLabels']
numTrain = trainFeat.shape[0]
numTest  = testFeat.shape[0]
M = trainFeat.shape[1]

# Split training data into two separate classes
trainFeat0 = trainFeat[trainLabels == 0]
trainFeat1 = trainFeat[trainLabels == 1]

# Estimate mean of Gaussian f_X|Y(x_ij | y_i)
# muhat[0,j] equals the mean of X_ij given Y_i=0
# muhat[1,j] equals the mean of X_ij given Y_i=1
muhat = np.zeros((2,M))
muhat[0] = np.mean(trainFeat0, axis=0)
muhat[1] = np.mean(trainFeat1, axis=0)
```

In [134…

```
# # Naive baseline classifier:  Predict all test examples are class 0
# yHat = np.zeros(numTest)
equation_2 = []
for i in range(numTest):
    curr_day = 0
    for j in range(M):
        curr = np.log(1/(np.sqrt(2 * np.pi))) - ((testFeat[i, j] - muhat[1, j]) ** 2) / 2
        curr_day += curr

    equation_2.append(curr_day)

equation_3 = []
for i in range(numTest):
    curr_day = 0
    for j in range(M):
        curr = np.log(1/(np.sqrt(2 * np.pi))) - ((testFeat[i, j] - muhat[0, j]) ** 2) / 2
        curr_day += curr
    equation_3.append(curr_day)

py_1 = 0.5
py_0 = 0.5

equation_2 = np.array(equation_2)
equation_3 = np.array(equation_3)

py_1_2nd_cond = equation_2 + np.log(py_1)
py_0_2nd_cond = equation_3 + np.log(py_0)

py_1_1st_cond = (py_1 * equation_2)
py_0_1st_cond = py_0 * equation_3

yHat = np.zeros(numTest)
yHat[np.logical_and(py_1_1st_cond > py_0_1st_cond, py_1_2nd_cond > py_0_2nd_cond)] = 1

# Accuracy (fraction of test days classified correctly)
accuracy = np.sum(yHat==testLabels)/numTest
print('Test accuracy: %f' % accuracy)
falseAlarms = np.sum(np.logical_and(yHat==1, testLabels==0))
print('Test false alarms: %d' % falseAlarms)
misses = np.sum(np.logical_and(yHat==0, testLabels==1))
print('Test missed detections: %d' % misses)
```

```
Test accuracy: 0.583333
Test false alarms: 267
Test missed detections: 3
```

c) *Rather than assuming features have variance one, set the variance parameters σ21j , σ20j equal to the variance of the empirical distribution of the training data. Classify each test example using Equation (1) with these variance estimates. Report your classification accuracy, and the numbers of false alarms and missed detections.*

In [135…

```
var_1 = np.var(trainFeat1, axis=0)
var_0 = np.var(trainFeat0, axis=0)

equation_2 = []
for i in range(numTest):
    curr_day = 0
    for j in range(M):
        curr = np.log(1/(np.sqrt(2 * np.pi * var_1[j]))) - ((testFeat[i, j] - muhat[1, j]) ** 2) / (2 * var_1[j])
        curr_day += curr

    equation_2.append(curr_day)

equation_3 = []
for i in range(numTest):
    curr_day = 0
    for j in range(M):
        curr = np.log(1/(np.sqrt(2 * np.pi * var_0[j]))) - ((testFeat[i, j] - muhat[0, j]) ** 2) / (2 * var_0[j])
        curr_day += curr
    equation_3.append(curr_day)

py_1 = 0.5
py_0 = 0.5

equation_2 = np.array(equation_2)
equation_3 = np.array(equation_3)

py_1_2nd_cond = equation_2 + np.log(py_1)
py_0_2nd_cond = equation_3 + np.log(py_0)

py_1_1st_cond = (py_1 * equation_2)
py_0_1st_cond = py_0 * equation_3

yHat = np.zeros(numTest)
yHat[np.logical_and(py_1_1st_cond > py_0_1st_cond, py_1_2nd_cond > py_0_2nd_cond)] = 1

# Accuracy (fraction of test days classified correctly)
accuracy = np.sum(yHat==testLabels)/numTest
print('Test accuracy: %f' % accuracy)
falseAlarms = np.sum(np.logical_and(yHat==1, testLabels==0))
print('Test false alarms: %d' % falseAlarms)
misses = np.sum(np.logical_and(yHat==0, testLabels==1))
print('Test missed detections: %d' % misses)
```

```
Test accuracy: 0.739198
Test false alarms: 167
Test missed detections: 2
```

d) *Rather than assuming the classes are equally probable, estimate pY (1) as the fraction of training examples that are ozone days. Classify each test example using Equation (1) with this informative class prior, and the variances from part (c). Report your classification accuracy, and the numbers of false alarms and missed detections.*

In [137…

```
var_1 = np.var(trainFeat1, axis=0)
var_0 = np.var(trainFeat0, axis=0)

equation_2 = []
for i in range(numTest):
    curr_day = 0
    for j in range(M):
        curr = np.log(1/(np.sqrt(2 * np.pi * var_1[j]))) - ((testFeat[i, j] - muhat[1, j]) ** 2) / (2 * var_1[j])
        curr_day += curr

    equation_2.append(curr_day)

equation_3 = []
for i in range(numTest):
    curr_day = 0
    for j in range(M):
        curr = np.log(1/(np.sqrt(2 * np.pi * var_0[j]))) - ((testFeat[i, j] - muhat[0, j]) ** 2) / (2 * var_0[j])
        curr_day += curr
    equation_3.append(curr_day)

py_1 = trainFeat1.shape[0] / trainFeat.shape[0]
py_0 = 1 - py_1

equation_2 = np.array(equation_2)
equation_3 = np.array(equation_3)

py_1_2nd_cond = equation_2 + np.log(py_1)
py_0_2nd_cond = equation_3 + np.log(py_0)

py_1_1st_cond = (py_1 * equation_2)
py_0_1st_cond = py_0 * equation_3

yHat = np.zeros(numTest)
yHat[np.logical_and(py_1_1st_cond > py_0_1st_cond, py_1_2nd_cond > py_0_2nd_cond)] = 1

# Accuracy (fraction of test days classified correctly)
accuracy = np.sum(yHat==testLabels)/numTest
print('Test accuracy: %f' % accuracy)
falseAlarms = np.sum(np.logical_and(yHat==1, testLabels==0))
print('Test false alarms: %d' % falseAlarms)
misses = np.sum(np.logical_and(yHat==0, testLabels==1))
print('Test missed detections: %d' % misses)
```

```
Test accuracy: 0.750000
Test false alarms: 158
Test missed detections: 4
```