1. Considering the following two-dimensional measurements:

	y_1	y_2
x_1	-2	2
x_2	-1	3
x_3	0	1
x_4	-2	1

(a) What are the maximum likelihood parameters of a multivariate Gaussian distribution for this data set?

(b) What is the Gaussian's shape? Draw its contour plot.

(a) Regarding the motivation for finding the "maximum likelihood parameters", this and other articles are helpful to delineate the MLE method, which aims to find the parameters of a distribution that maximize the probability of observing a given data set. Considering a data set normally distributed (as referenced in the question's statement), the MLE method is equivalent to finding the parameters of the distribution that minimize the sum of squared errors between the data set and the distribution's probability density function.

As we know, a multivariate Gaussian distribution is defined by its mean vector $\boldsymbol{\mu}$ and its covariance matrix $\boldsymbol{\Sigma}$:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

The mean vector is simply the average of the data points (considering each coordinate), while the covariance matrix, on the other hand, requires a bit more work, requiring us to calculate the deviations of each data point from the mean:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i, \quad \boldsymbol{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^T$$

Note, of course, that $\boldsymbol{\Sigma}$ is a symmetric matrix. We can, then, start calculating the maximum likelihood parameters:

$$\boldsymbol{\mu} = \frac{1}{4}\begin{bmatrix}-2-1+0-2\\2+3+1+1\end{bmatrix} = \begin{bmatrix}-1.25\\1.75\end{bmatrix}$$

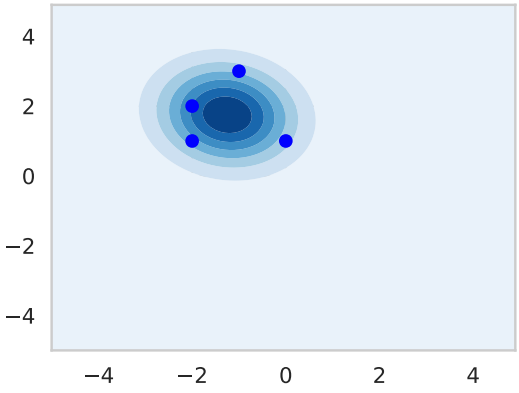
$$\boldsymbol{\Sigma} = \frac{1}{3}\begin{bmatrix}(-2+1.25)^2+\cdots+(-2+1.25)^2 & (-2+1.25)(2-1.75)+\cdots\\(-2+1.25)(2-1.75)+\cdots & (2-1.75)^2+\cdots+(1-1.75)^2\end{bmatrix} = \begin{bmatrix}0.916667 & -0.0833333\\-0.0833333 & 0.916667\end{bmatrix}$$

As such, we can calculate both the determinant and the inverse of the covariance matrix, effectively gathering all the needed parameters for the multivariate Gaussian distribution:

$$|\boldsymbol{\Sigma}| = 0.916667^2 - 0.083333^2 = 0.8(3)$$

$$X = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \implies X^{-1} = \frac{1}{ac - bd} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}; \quad \mathbf{\Sigma}^{-1} = \frac{1}{0.8(3)} \begin{bmatrix} 0.916667 & 0.083333 \\ 0.083333 & 0.916667 \end{bmatrix} = \begin{bmatrix} 1.1 & 0.1 \\ 0.1 & 1.1 \end{bmatrix}$$

(b) The following contour plot was computed utilizing Python (the code snippets are available in this sheet's respective notebook):



Contour plots are a great way to visualize the shape of a multivariate Gaussian distribution in two dimensions. The plot above shows the distribution's probability density function, which is proportional to the probability of observing a data point in a given region - the darker the region, the higher the probability of observing a data point in that region!

Essentially, each line in the plot above represents a constant probability density value. The ellipsoid is always centered at the mean vector, and its shape is determined by the covariance matrix. The ellipsoid's axes' directions are determined by the eigenvectors of the covariance matrix, and its axes' lengths are determined by the square root of the eigenvalues of the covariance matrix. The ellipsoid's axes' lengths are, then, proportional to the standard deviations of the distribution along each axis.

2. Consider the following data-set, paired with a query vector $x_{new} = \begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}^T$:

	y_1	y_2	y_3	y_4	y_5	z
x_1	1	1	0	1	0	1
x_2	1	1	1	0	0	0
x_3	0	1	1	1	0	0
x_4	0	0	0	1	1	0
x_5	1	0	1	1	1	1
x_6	0	0	1	0	0	1
x_7	0	0	0	0	1	1

(a) Using Bayes' rule, without making any assumptions, compute the posterior probabilities for the query vector. How is it classified?

(b) What is the problem of working without assumptions?

(c) Compute the class for the same query vector under the naive Bayes assumption?

(d) Consider the presence of missings. Under the same naive Bayes assumption, how would you classify the query vector $x_{new} = \begin{bmatrix}1 & ? & 1 & ? & 1\end{bmatrix}^T$?

(a) Bayes' rule tells us that the posterior probability of a class c given a query vector x_{new} is:

$$P(z = c|x_{new}) = \frac{P(x_{new}|z = c)P(z = c)}{P(x_{new})}$$

We call **priors** the probabilities $P(z = c)$, and **likelihoods** the "probabilities" $P(x_{new}|z = c)$. To translate the product between the likelihood and the prior into an actual probability, we need to normalize such product by the probability $P(x_{new})$, which is the probability of observing the query vector x_{new} "in the wild", regardless of the class.

Here, we have two possible classes (i.e, z values) possible for any given query vector: $z = 0$ and $z = 1$. Note that we're not allowed to make any assumptions - that is, we mustn't assume any conditional independence or distribution for the data's features.

By looking at the data set, we can deduce the **priors** for each class:

$$P(z = 0) = \frac{3}{7} \quad \text{and} \quad P(z = 1) = \frac{4}{7}$$

Without any assumptions, however, $P(x_{new})$ will essentially be the probability of observing the query vector x_{new} in the "training" data set. Since we never observed the query vector x_{new} in the training data set, $P(x_{new})$ is zero (as are the likelihoods). As such, we can't use Bayes' rule to compute the posterior probabilities for the query vector x_{new}, and such a sample would be **unclassified**.

(b) Without assumptions, we're essentially strapped to the data set we have at hand. In cases like the one at hand, for example, we're unable to compute the posterior probabilities for the query vector x_{new} (and for a myriad of other query vectors). As such, data sets with a large number of features and/or a small number of samples are problematic to work with without making any assumptions, since we're more likely than not going to run into this problem often.

(c) The Naive Bayes assumption essentially states that the features are conditionally independent given the class. In other words, it assumes that observing a given feature in the presence of a class is completely independent of observing any other feature. Such a strong assumption, although often misleading, is often a good starting point for classification problems, letting us compute more easily the posterior probabilities for new query vectors. Regarding x_{new}, we can compute the posterior probabilities as follows (and considering the previously computed priors):

$$P(z = c|x_{new}) = \frac{P(x_{new}|z = c)P(z = c)}{P(x_{new})} = \frac{P(y_1 = 1 \mid z = c) \cdot ... \cdot P(y_5 = 1 \mid z = c) \cdot P(z = c)}{P(y_1 = 1) \cdot ... \cdot P(y_5 = 1)}$$

The denominator will always be the same for both classes, so we can ignore it, effectively needing only to compute the numerator for each class. The numerators for each class are given by:

$$\begin{aligned} z = 0: &\quad (1/3)^2 \cdot (2/3)^3 \cdot (3/7) \approx 0.014109 \\ z = 1: &\quad (2/4)^4 \cdot (1/4)^1 \cdot (4/7) \approx 0.0089 \end{aligned}$$

With class $z = 0$ having a higher posterior probability, the query vector x_{new} is classified as belonging to class $z = 0$.

(d) Here, we're essentially asking for the probability of observing a given sample with features y_1, y_3 and y_5 valued at 1, considering that the feature values for both y_2 and y_4 are missing (i.e., unknown). We can compute the posterior probabilities for the query vector x_{new} as follows:

$$P(z = c|x_{new}) = \frac{P(y_1 = 1 \mid z = c) \cdot P(y_3 = 1 \mid z = c) \cdot P(y_5 = 1 \mid z = c) \cdot P(z = c)}{P(y_1 = 1) \cdot P(y_3 = 1) \cdot P(y_5 = 1)}$$

The denominator will always be the same for both classes, so we can ignore it, once again. The numerators for each class are given by:

$$\begin{aligned} z = 0: &\quad (1/3)^2 \cdot (2/3) \cdot (3/7) \approx 0.031746 \\ z = 1: &\quad (2/4)^3 \cdot (4/7) \approx 0.07143 \end{aligned}$$

With class $z = 1$ having a higher posterior probability, the query vector x_{new}, with missing values for y_2 and y_4, is classified as belonging to class $z = 1$.

3. Considering the following data set, paired with the query vector $x_{new} = \begin{bmatrix}100 & 225\end{bmatrix}^T$:

	y_1	y_2	z
x_1	170	160	0
x_2	80	220	1
x_3	90	200	1
x_4	60	160	0
x_5	50	150	0
x_6	70	190	1

(a) Compute the most probable class for the query vector, assuming that the likelihoods are 2-dimensional Gaussians.

(b) Compute the most probable class for the query vector, under the Naive Bayes assumption, using 1-dimensional Gaussians to model the likelihoods.

(a) Just as had been illustrated in the previous question, Bayes' rule tells us that the probability of a class given a query vector is proportional to the product of the likelihood and the prior. As such, we can compute the posterior probabilities for the query vector x_{new} as follows:

$$P(z = c|x_{new}) = \frac{P(x_{new}|z = c)P(z = c)}{P(x_{new})}$$

The priors are calculated in the same way as before:

$$P(z = 0) = P(z = 1) = 3/6 = 1/2$$

Here, however, our likelihoods are 2-dimensional Gaussians:

$$P(x \mid z = c) \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_c}, \boldsymbol{\Sigma_c}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma_c}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_c})^T \boldsymbol{\Sigma_c}^{-1}(\boldsymbol{x} - \boldsymbol{\mu_c})\right)$$

Both the mean and covariance matrix are relatively straight forward to compute; having done so in the past, we'll simply plug the results here (the Python code used to compute the results is available in the notebook):

$$\boldsymbol{\mu_0} = \begin{bmatrix}93.3333\\156.667\end{bmatrix} \qquad \boldsymbol{\Sigma_0} = \begin{bmatrix}4433.33 & 216.667\\216.667 & 33.3333\end{bmatrix}$$

$$\boldsymbol{\mu_1} = \begin{bmatrix}80\\203.333\end{bmatrix} \qquad \boldsymbol{\Sigma_1} = \begin{bmatrix}100 & 50\\50 & 233.333\end{bmatrix}$$

Considering that, for both classes, both the denominator and the prior values will be the same, it'll be the likelihoods defining whether we assign the query vector to class $z = 0$ or $z = 1$. The likelihoods for each class are calculated as follows:

$$P(x_{new} \mid z=0) \sim \mathcal{N}(x_{new} \mid \mu_0, \Sigma_0) = \frac{1}{(2\pi)^{2/2}|\Sigma_0|^{1/2}} \exp\left(-\frac{1}{2}(x_{new}-\mu_0)^T\Sigma_0^{-1}(x_{new}-\mu_0)\right)$$
$$= \cdots = 3.47826 \cdot 10^{-48}$$
$$P(x_{new} \mid z=1) \sim \mathcal{N}(x_{new} \mid \mu_1, \Sigma_1) = \frac{1}{(2\pi)^{2/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x_{new}-\mu_1)^T\Sigma_1^{-1}(x_{new}-\mu_1)\right)$$
$$= \cdots = 0.000107642$$

With $z = 1$ having a higher likelihood (and thus a higher posterior probability), the query vector x_{new} is most likely to belong to class $z = 1$.

(b) Modelling the likelihoods as 1-dimensional Gaussians (instead of the 2-dimensional ones utilized in the previous exercise) ends up being a simplification of the problem; we can reutilize the previously computed mean vectors to compute the likelihoods for each class, as follows:

$$\mu_0 = \begin{bmatrix}\mu_0^{(1)}\\ \mu_0^{(2)}\end{bmatrix} = \begin{bmatrix}93.3333\\ 156.667\end{bmatrix}, \quad \mu_1 = \begin{bmatrix}\mu_1^{(1)}\\ \mu_1^{(2)}\end{bmatrix} = \begin{bmatrix}80\\ 203.333\end{bmatrix}$$
$$\sigma_0 = \begin{bmatrix}\sqrt{\Sigma_0^{(1,1)}}\\ \sqrt{\Sigma_0^{(2,2)}}\end{bmatrix} = \begin{bmatrix}66.5833\\ 5.7735\end{bmatrix}, \quad \sigma_1 = \begin{bmatrix}\sqrt{\Sigma_1^{(1,1)}}\\ \sqrt{\Sigma_1^{(2,2)}}\end{bmatrix} = \begin{bmatrix}10\\ 15.2753\end{bmatrix}$$

Note how, since we're now working with conditionally independent variables, we abandon the notion of a covariance matrix (and, as such, the covariance between all features is zero), and instead work with the variances of each feature. Here, we're going to split the likelihoods, utilizing the Naive Bayes assumption, into two 1-dimensional Gaussians, one for each feature:

$$P(x \mid z=c) \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu_c}, \boldsymbol{\sigma_c}) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{c,i}^2}} \exp\left(-\frac{1}{2\sigma_{c,i}^2}(x_i - \mu_{c,i})^2\right)$$

Once again, we only need to know the likelihoods in order to decide which class the query vector x_{new} belongs to, since the priors and the denominator are the same for both classes. The likelihoods for each class are the following (once again, intermediate steps aren't shown for brevity):

$$P(x_{new} \mid z=0) \sim \mathcal{N}(x_{new} \mid \mu_0, \sigma_0) = \prod_{i=1}^{2} \frac{1}{\sqrt{2\pi\sigma_{0,i}^2}} \exp\left(-\frac{1}{2\sigma_{0,i}^2}(x_{new,i} - \mu_{0,i})^2\right) = \cdots = 1.57083 \cdot 10^{-34}$$
$$P(x_{new} \mid z=1) \sim \mathcal{N}(x_{new} \mid \mu_1, \sigma_1) = \prod_{i=1}^{2} \frac{1}{\sqrt{2\pi\sigma_{1,i}^2}} \exp\left(-\frac{1}{2\sigma_{1,i}^2}(x_{new,i} - \mu_{1,i})^2\right) = \cdots = 5.1566 \cdot 10^{-5}$$

We opt to assign the query vector to class $z = 1$, since it has a higher likelihood.

4. Assuming training examples with m features and a binary class:

(a) How many parameters are needed to estimate, considering boolean features and:

i. No assumptions regarding the data's distribution.

ii. Naive Bayes assumption.

(b) How many parameters are needed to estimate, considering numeric-valued features and:

i. Multivariate Gaussian assumption.

ii. Naive Bayes with a Gaussian assumption.

For starters, it's worth noting that we'll always need to estimate at least one prior, independently of the features' type and the assumptions we make regarding the data. Considering a strictly binary class, we'll only need to estimate one prior, be it π_k, since the other one is trivially $1 - \pi_k$. As a side note, I'll also be adding the **generalized** formula for each case below, considering c classes, m features, and n values for each feature.

(a) i. Working with the classic Bayesian model, we'll need to estimate the likelihoods for each class; since we're working with boolean features, we'll need to estimate the probability of each feature being true/false for each class. There are, of course, 2^m possible combinations of boolean features for each class (since we can't assume any type of conditional independence between features); however, since the sum of the probabilities of all possible combinations must be equal to 1, we can reduce the number of parameters to be estimated to $2^m - 1$. As such, we'll need to estimate $2(2^m - 1) + 1$ parameters in total, since we'll also need to estimate the prior for each class.

Generalized formula: $(c - 1) + c(n^m - 1)$

ii. Working with the Naive Bayes assumption - i.e, assuming that the features are conditionally independent given the class - we'll need to estimate the probability of each feature being true/-false for each class; this way, there'll be $2m$ possible features combinations, and as such, we'll need to estimate $2m + 1$ parameters in total!

Generalized formula: $(c - 1) + cm(n - 1)$

(b) i. Working with a multivariate Gaussian distribution, we'll need to estimate the mean vector and the covariance matrix for each class; while we need to estimate all m parameters for the mean vector (per class), we can utilize the fact that the covariance matrix is symmetric to reduce the number of parameters to be estimated to $\frac{m(m+1)}{2}$ per class, the amount of entries in the upper triangular part of an $m \times m$ matrix. As such, we'll need to estimate $2(m + \frac{m(m+1)}{2}) + 1$ parameters in total.

Generalized formula: $(c - 1) + c\left(m + \dfrac{m(m+1)}{2}\right)$

ii. Working with the Naive Bayes gaussian assumption, we'll need to estimate the mean vector and the variance vector for each class; here, since there are no covariance parameters to be

estimated, we'll need to estimate m parameters for the mean vector and m parameters for the variance vector, per class. As such, we'll need to estimate $2(2m) + 1$ parameters in total.

Generalized formula: $(c - 1) + 2cm$