Solution 1:

a) When using the naive Bayes classifier, the features $\mathbf{x} := (x_{\text{Color}}, x_{\text{Form}}, x_{\text{Origin}})$ are assumed to be conditionally independent of each other, given the category $y = k \in \{\text{yes}, \text{no}\}$, s.t.

$$\mathbb{P}(\mathbf{x} \mid y = k) = \mathbb{P}((x_{\text{Color}}, x_{\text{Form}}, x_{\text{Origin}}) \mid y = k) = \mathbb{P}(x_{\text{Color}} \mid y = k) \cdot \mathbb{P}(x_{\text{Form}} \mid y = k) \cdot \mathbb{P}(x_{\text{Origin}} \mid y = k).$$

Recall Bayes' theorem:

$$\pi_k(\mathbf{x}) = \mathbb{P}(y = k \mid \mathbf{x}) = \frac{\mathbb{P}(\mathbf{x}|y = k)\mathbb{P}(y = k)}{\mathbb{P}(\mathbf{x})}.$$

As the denominator is constant across all classes, the following holds for the posterior probabilities:

$$\begin{aligned}\pi_k(\mathbf{x}) &\propto \underbrace{\pi_k \cdot \mathbb{P}(x_{\text{Color}} \mid y = k) \cdot \mathbb{P}(x_{\text{Form}} \mid y = k) \cdot \mathbb{P}(x_{\text{Origin}} \mid y = k)}_{=:\alpha_k(x)} \\ &\iff \exists c \in \mathbb{R} : \pi_k(\mathbf{x}) = c \cdot \alpha_k(\mathbf{x}),\end{aligned}$$

where $\pi_k = \mathbb{P}(y = k)$ is the prior probability of class k and c is the normalizing constant.

From this and since the posterior probabilities need to sum up to 1, we know that

$$1 = \; c \cdot \alpha_{\text{yes}}(\mathbf{x}) + c \cdot \alpha_{\text{no}}(\mathbf{x}) \iff c = \frac{1}{\alpha_{\text{yes}}(\mathbf{x}) + \alpha_{\text{no}}(\mathbf{x})}.$$

This means that, in order to compute $\pi_{\text{yes}}(\mathbf{x})$, the scores $\alpha_{\text{yes}}(\mathbf{x})$ and $\alpha_{\text{no}}(\mathbf{x})$ are needed.

Now we want to estimate for a new fruit the posterior probability $\hat{\pi}_{yes}((\text{yellow}, \text{round}, \text{imported}))$.

Obviously, we do not know the *true* prior probability and the *true* conditional densities. Here – since the target and the features are categorical – we use a categorical distribution, i.e., the simplest distribution over a g-way event that is fully specified by the individual probabilities for each class (which must of course sum to 1). This is a generalization of the Bernoulli distribution to the multi-class case. We can estimate the distribution parameters via the relative frequencies encountered in the data:

$$\begin{aligned}\hat{\alpha}_{\text{yes}}(\mathbf{x}_*) &= \hat{\pi}_{yes} \cdot \hat{\mathbb{P}}(x_{\text{Color}} = \text{yellow} \mid y = \text{yes}) \cdot \hat{\mathbb{P}}(x_{\text{Form}} = \text{round} \mid y = \text{yes}) \cdot \hat{\mathbb{P}}(x_{\text{Origin}} = \text{imported} \mid y = \text{yes}) \\ &= \frac{3}{8} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot 1 = \frac{1}{24} \approx 0.042, \\ \hat{\alpha}_{\text{no}}(\mathbf{x}_*) &= \hat{\pi}_{no} \cdot \hat{\mathbb{P}}(x_{\text{Color}} = \text{yellow} \mid y = \text{no}) \cdot \hat{\mathbb{P}}(x_{\text{Form}} = \text{round} \mid y = \text{no}) \cdot \hat{\mathbb{P}}(x_{\text{Origin}} = \text{imported} \mid y = \text{no}) \\ &= \frac{5}{8} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} = \frac{3}{50} = 0.060.\end{aligned}$$

At this stage we can already see that the predicted label is "no", since $\hat{\alpha}_{\text{no}}(\mathbf{x}_*) = 0.060 > \frac{1}{24} = \hat{\alpha}_{\text{yes}}(\mathbf{x}_*)$ – that is, if we threshold at 0.5 for predicting "yes".

With the above we can compute the posterior probability

$$\hat{\pi}_{\text{yes}}(\mathbf{x}_*) = \frac{\hat{\alpha}_{\text{yes}}(\mathbf{x}_*)}{\hat{\alpha}_{\text{yes}}(\mathbf{x}_*) + \hat{\alpha}_{\text{no}}(\mathbf{x}_*)} \approx 0.410 < 0.5,$$

and check our calculations against the corresponding `R` results:

```r
df_banana <- data.frame(
  color = as.factor(
    c("yellow", "yellow", "yellow", "brown", "brown", "green", "green", "red")),
  form = as.factor(
    c("oblong", "round", "oblong", "oblong", "round", "round", "oblong", "round")),
  origin = as.factor(
    c("imported", "domestic", "imported", "imported", "domestic", "imported",
    "domestic", "imported")),
  banana = as.factor(c("yes", "no", "no", "yes", "no", "yes", "no", "no")))

new_fruit <- data.frame(color = "yellow", form = "round", origin = "imported")

library(mlr3)
library(mlr3learners)

nb_learner <- lrn("classif.naive_bayes", predict_type = "prob")

banana_task <- TaskClassif$new(
  id = "banana",
  backend = df_banana,
  target = "banana")

nb_learner$train(banana_task)
nb_learner$predict_newdata(new_fruit)

## <PredictionClassif> for 1 observations:
##  row_ids truth response   prob.no  prob.yes
##        1  <NA>       no 0.5901639 0.4098361
```

b) Before, we only had categorical features and could use the empirical frequencies as our parameters in a categorical distribution. For the distribution of a numerical feature, given the the category, we need to define a probability distribution with continuous support. A popular choice is to use Gaussian distributions. For example, for the information x_{Length} we could assume that

$$\mathbb{P}(x_{\text{Length}} \mid y = \text{yes}) \sim \mathcal{N}(\mu_{\text{yes}}, \sigma^2_{\text{yes}})$$

and

$$\mathbb{P}(x_{\text{Length}} \mid y = \text{no}) \sim \mathcal{N}(\mu_{\text{no}}, \sigma^2_{\text{no}}).$$

In order to fully specify these normal distributions we need to estimate their parameters $\mu_{\text{yes}}, \mu_{\text{no}}, \sigma^2_{\text{yes}}, \sigma^2_{\text{no}}$ from the data via the usual estimators (empirical mean and empirical variance with bias correction).

Solution 2:

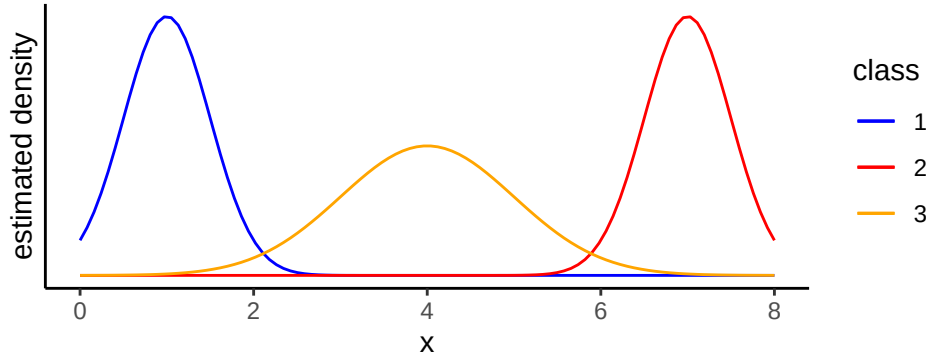
a) i) As the data seem to be pretty symmetric conditional on the respective class, we estimate the class means to lie roughly in the middle of the data clusters: $\hat{\mu}_1 = 1$, $\hat{\mu}_2 = 7$, $\hat{\mu}_3 = 4$. @Ludwig: die Datenlage ist tatsächlich etwas suboptimal, weil die aus einer Unif gezogen sind statt aus einer NV - genau genommen passen halt QDA und LDA hier nicht gut

ii) We see that the variances in classes 1 and 2 are similar and also much smaller than in class 3. Therefore, the densities could look roughly like this:

```r
library(ggplot2)

colors <- c("1" = "blue", "2" = "red", "3" = "orange")
```

```r
ggplot(data.frame(x = c(0, 8)), aes(x = x)) +
  stat_function(fun = dnorm, n = 100, args = list(mean = 1, sd = 0.5), aes(col = "1")) +
  stat_function(fun = dnorm, n = 100, args = list(mean = 7, sd = 0.5), aes(col = "2")) +
  stat_function(fun = dnorm, n = 100, args = list(mean = 4, sd = 1), aes(col = "3")) +
  scale_color_manual("class", values = colors) +
  theme_classic() +
  ylab("estimated density") +
  scale_y_continuous(breaks = NULL)
```



iii) Since LDA assumes constant variances across all classes (also if this does not reflect the data situation), all densities would have the same shape and only differ in location.

iv) As we have already noted, the assumption of equal class variances is not justified here, but LDA is confined to equivariant distributions. Therefore, the more flexible QDA is preferable in this case.

b) The prediction for $\mathbf{x}_{*1}$ will probably be $\hat{z}_{*1} = 3$ because the density of class 3 has much larger variance and will therefore overshoot the density of class 1. For $\mathbf{x}_{*2}$ the case is clear and we have $\hat{z}_{*2} = 2$.

c) In order to arrive at the equation for the decision boundary, we first need to understand that on the boundary of classes 1 and 2, both discriminant functions $\delta_1(\mathbf{x})$ and $\delta_2(\mathbf{x})$ will be exactly equal – it is the one location where we would not be able to unambiguously assign points to either one of the classes, whereas for $\delta_1(\mathbf{x}) > \delta_2(\mathbf{x})$ we would choose class 1 and vice versa. Therefore, we compute the equation as follows:

$$
\begin{aligned}
\delta_1(\mathbf{x}) &= \delta_2(\mathbf{x}) \\
\Leftrightarrow \log \pi_1 - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_1 &= \log \pi_2 - \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_2 \\
\Leftrightarrow \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_1 - \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_2 &= \log \frac{\pi_2}{\pi_1} + \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2\right) \\
\Leftrightarrow \mathbf{x}^\top \underbrace{\left(\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right)}_{=:\boldsymbol{\nu} \in \mathbb{R}^{2\times 1}} &= \log \frac{\pi_2}{\pi_1} + \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2\right) \\
\Leftrightarrow \nu_1 x_1 + \nu_2 x_2 &= \underbrace{\log \frac{\pi_2}{\pi_1} + \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2\right)}_{=:a \in \mathbb{R}}.
\end{aligned}
$$

The right hand side might look somewhat complicated but simply evaluates to a scalar and we obtain the hyperplane equation $\mathbf{x}^\top \nu = a$, in this case defining a line in $\mathbb{R}^2$.

Again, we see that LDA is indeed a linear classifier.

Solution 3:

See R code