```
# -------------------------------------------------------------------------------
#
# Generative Models
#
# -------------------------------------------------------------------------------

# Setup
library(tidyverse)
library(dslabs)
library(dplyr)
library(ggplot2)
library(Lahman)
library(HistData)
library(caret)
library(e1071)
library(matrixStats)

# We have described how when using square loss the conditional expectation or
# probabilities provide the best approach to developing a decision rule. In a
# binary case, the best we can do is called Bayes' rule which is a decision rule
# based on the true conditional probability, probably y equals one given the
# predictors x.

#    p(x)=Pr(Y=1|X=x)

# We have described several approaches to estimating this conditional
# probability. Note that in all these approaches, we estimate the conditional
# probability directly and do not consider the distribution of the predictors.
# In machine learning, these are referred to as discriminative approaches.
# However, Bayes' theorem tells us that knowing the distribution of the
# predictors x may be useful. Methods that model the joint distribution of y and
# the predictors x are referred to as generative models.

# We start by describing the most general generative model naive Bayes and then
# proceed to describe some more specific cases, quadratic discriminant analysis
# QDA and linear discriminant analysis LDA. Recall that Bayes' theorem tells us
# that we can rewrite the conditional probability like this with the f's
# representing the distribution functions of the predictors x for the two
# classes y equals 1 and when y equals 0. The formula implies that if we can
# estimate these conditional distributions, the predictors, we can develop a
# powerful decision realm. However, this is a big if.

# As we go froward, we will encounter examples in which the predictors x have
# many dimensions and we do not have much information about their distribution.
# So it will be very hard to estimate those conditional distributions. In these
# cases, naive Bayes would be practically impossible to implement. However,
# there are instances in which we have a small number of predictors, not much
# more than two, and many categories in which generated models can be quite
# powerful. We describe two specific examples and use our previously described
# case study to illustrate them.

# Let's start with a very simple and uninteresting, yet illustrative, example.
# Predict sex from the height example. We can get the data and generate training
# and test set using this code.

library(caret)
data("heights")
y <- heights$height
set.seed(2)
test_index <- createDataPartition(y, times = 1, p = 0.5, list = FALSE)
train_set <- heights %>% slice(-test_index)
test_set <- heights %>% slice(test_index)

# In this example, the naive Bayes approach is particularly appropriate. Because
# we know that the normal distribution is a very good approximation of the
# conditional distributions of height given sex for both classes, females and
# males. This implies that we can approximate the conditional distributions by
# simply estimating averages and standard deviations from the data with this
# very simple piece of code, like this.

params <- train_set %>%
     group_by(sex) %>%
     summarize(avg = mean(height), sd = sd(height))
params
#> # A tibble: 2 x 3
#>   sex      avg    sd
#>   <fct>  <dbl> <dbl>
#> 1 Female  64.5  4.02
#> 2 Male    69.3  3.52

# The prevalence, which we will denote with pi, which is equal to the
# probability of y equals 1, can be estimated from the data as well like this.

pi <- train_set %>%
     summarize(pi = mean(sex=="Female")) %>%
     .$pi
pi
#> [1] 0.2290076

# We basically compute the proportion of females. Now we can use our estimates
# of average and standard deviations to get the actual rule. We get the
# conditional distributions, f0 and f1, and then we use Bayes theorem to compute
# the naive Bayes estimate of the conditional probability.

x <- test_set$height

f0 <- dnorm(x, params$avg[2], params$sd[2])
f1 <- dnorm(x, params$avg[1], params$sd[1])

p_hat_bayes <- f1*pi / (f1*pi + f0*(1 - pi))

# This estimate of the conditional probably looks a lot like a logistic
# regression estimate, as we can see in this graph.

qplot(x, p_hat_bayes)

# In fact, we can show mathematically that the naive Bayes approach is similar
# to the logistic regression approach in this particular case. But we're not
# going to show that derivation here.

# One nice feature of the Naive Bayes approach is that it includes a parameter
# to account for differences in prevalence. Using our sample, we estimated the
# conditional probabilities and the prevalence pi. If we use hats to denote the
# estimates, we can rewrite the estimate of the conditional probability with
# this formula. As we discussed, our sample has much lower prevalence than the
# general population. We only have 23% women. So if we use our rule that the
# conditional probability has to be bigger than 0.5 to predict females, our
# accuracy will be affected due to the low sensitivity, which we can see by
# typing this code.

y_hat_bayes <- ifelse(p_hat_bayes > 0.5, "Female", "Male")
sensitivity(data = factor(y_hat_bayes), reference = factor(test_set$sex))
#> [1] 0.263

# Again, this is because the algorithm gives more weight to specificity to
# account for the low prevalence. You can see that we have very high specificity
# by typing this code.

specificity(data = factor(y_hat_bayes), reference = factor(test_set$sex))
#> [1] 0.953

# This is due mainly to the fact that pi hat is substantially less than 0.5, so
# we tend to predict male more often than female. It makes sense for a machine
# learning algorithm to do this in our sample because we do have a higher
# percentage of males. But if we were to extrapolate this to the general
# population, our overall accuracy would be affected by the low sensitivity. The
# Naive Bayes approach gives us a direct way to correct this, since we can
# simply force our estimate of pi to be different. So to balance specificity and
# sensitivity, instead of changing the cutoff in the decision rule, we could
# simply change pi hat.

p_hat_bayes_unbiased <- f1*0.5 / (f1*0.5 + f0*(1-0.5))
y_hat_bayes_unbiased <- ifelse(p_hat_bayes_unbiased> 0.5, "Female", "Male")

# Here in this code, we changed it to 0.5. Now note the difference in
# sensitivity and the better balance. We can see it using this code.

sensitivity(data = factor(y_hat_bayes_unbiased), reference = factor(test_set$sex))
#> [1] 0.712
specificity(data = factor(y_hat_bayes_unbiased), reference = factor(test_set$sex))
#> [1] 0.821

# This plot shows us that the new rule also gives us a very intuitive cutoff
# between 66 and 67, which is about the middle of the female and male average
# heights.

qplot(x, p_hat_bayes_unbiased, geom = "line") +
     geom_hline(yintercept = 0.5, lty = 2) +
     geom_vline(xintercept = 67, lty = 2)
```