Mushroom_Classification

Carol Hardy

2022-07-14

```
#import data
mush <- read.csv("https://raw.githubusercontent.com/call-me-carol/Personal_Classification_Mushrooms/mai
head(mush)
```

```
##   class cap.shape cap.surface cap.color bruises odor gill.attachment
## 1     p         x           s         n       t    p               f
## 2     e         x           s         y       t    a               f
## 3     e         b           s         w       t    l               f
## 4     p         x           y         w       t    p               f
## 5     e         x           s         g       f    n               f
## 6     e         x           y         y       t    a               f
##   gill.spacing gill.size gill.color stalk.shape stalk.root
## 1            c         n          k           e          e
## 2            c         b          k           e          c
## 3            c         b          n           e          c
## 4            c         n          n           e          e
## 5            w         b          k           t          e
## 6            c         b          n           e          c
##   stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
## 1                        s                        s                      w
## 2                        s                        s                      w
## 3                        s                        s                      w
## 4                        s                        s                      w
## 5                        s                        s                      w
## 6                        s                        s                      w
##   stalk.color.below.ring veil.type veil.color ring.number ring.type
## 1                      w         p          w           o         p
## 2                      w         p          w           o         p
## 3                      w         p          w           o         p
## 4                      w         p          w           o         p
## 5                      w         p          w           o         e
## 6                      w         p          w           o         p
##   spore.print.color population habitat
## 1                 k          s       u
## 2                 n          n       g
## 3                 n          n       m
## 4                 k          s       u
## 5                 n          a       g
## 6                 k          n       g
```

```
mush <- mush %>% mutate_all(., list(~factor(.))) # turn all vars into factors
levels(mush$class) # meaning that edible = 0 and poisonous = 1
```

```
## [1] "e" "p"
```

Goal: Identify which mushrooms are poisonous and which are edible Looking at the comments in Kaggle, it seems that users are able to predict poisonous vs edible mushrooms with 100% accuracy. However, rather than accuracy of prediction, I am looking to deeper understand logistic regression. A regression tree or some other machine learning method will certainly be able to predict with 100% accuracy. I am looking to understand the relationships between the predictors and the outcome. I think this is more of a challenge because the coefficients are especially difficult to interpret in this framework.

Based off an unscientific Google search, spore print and smell are good indicators of whether a mushroom is edible or not. Thus, I begin my modeling with those two predictors. Both of these predictors have nine levels.

Model:

$$y_i \sim Bernoulli(\pi_i)$$

Where $y_i = 1$ is the probability that a mushroom is poisonous

$$Pr(y_i = 1) = \pi_i = logit^{-1}(X\underline{\beta})$$

Where $X\underline{\beta}$ is the linear combination. In this case the x's will be spore print color and odor.

- Where the link is the inverse logistic function. It maps a continuous variable to the range $(0, 1)$

$$logit(x)^{-1} = \frac{exp(x)}{1 + exp(x)}$$

- Note: there is no error term because the randomness comes from the Bernoulli distribution
- Note: There is no linearity assumption in this model, this seems obvious, but I often forget!

Simple Model Fit:

I wanted to use odor and spore print color as predictors. However when we dig into the data, it becomes clear that there will be identifiability issues with those predictors. Let us take a look:

```
mosaic::tally(class ~ odor, data = mush, margins = T) #There is perfect separation here
```

```
##        odor
## class      a    c    f    l    m    n    p    s    y
##   e      400    0    0  400    0 3408    0    0    0
##   p        0  192 2160    0   36  120  256  576  576
##   Total  400  192 2160  400   36 3528  256  576  576
```

```
mosaic::tally(class ~ spore.print.color, data = mush, margins = T) #There is perfect separation here
```

```
##        spore.print.color
## class      b    h    k    n    o    r    u    w    y
##   e       48   48 1648 1744   48    0   48  576   48
##   p        0 1584  224  224    0   72    0 1812    0
##   Total   48 1632 1872 1968   48   72   48 2388   48
```

Notice: Odor, unless neutral, perfectly predicts whether a mushroom will be poisonous or not.

These tables bring up the issue of separation, when a predictor is perfectly aligned with an outcome. You probably do not need a fancy model to be doing prediction in this case.

In a frequentist setting this model would be:

```
freq_simple <- glm(class ~ odor, family = binomial(link = "logit"),
                     data = mush)
summary(freq_simple)
```

```
##
## Call:
## glm(formula = class ~ odor, family = binomial(link = "logit"),
##     data = mush)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.26308  -0.26308  -0.00003   0.00003   2.60038
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.157e+01  1.462e+03  -0.015    0.988
## odorc        4.313e+01  2.567e+03   0.017    0.987
## odorf        4.313e+01  1.591e+03   0.027    0.978
## odorl        5.586e-11  2.067e+03   0.000    1.000
## odorm        4.313e+01  5.087e+03   0.008    0.993
## odorn        1.822e+01  1.462e+03   0.012    0.990
## odorp        4.313e+01  2.340e+03   0.018    0.985
## odors        4.313e+01  1.903e+03   0.023    0.982
## odory        4.313e+01  1.903e+03   0.023    0.982
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11251.8  on 8123  degrees of freedom
## Residual deviance:  1047.3  on 8115  degrees of freedom
## AIC: 1065.3
##
## Number of Fisher Scoring iterations: 20
```

Example of Interpretation: A fishy smelling mushroom has a 100% or invlogit(-2.157e+01 + 4.313e+01*1) chance of being poisonous.

Comment: This is a perfect example of where the prior in a Bayesian setting helps stabilize estimates. Rather than having a 0 or 1 probability, we have left room for uncertainty.

```
simple_m <- stan_glm(class ~ odor, family = binomial(link = "logit"),
                     data = mush, refresh = 0)
print(simple_m)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      class ~ odor
##  observations: 8124
##  predictors:   9
## ------
##             Median MAD_SD
## (Intercept) -6.2    1.0
## odorc       19.3    6.8
## odorf       15.2    1.8
## odorl       -8.9    7.2
## odorm       31.1   19.3
## odorn        2.9    1.0
## odorp       18.5    5.9
## odors       16.5    3.4
## odory       16.4    3.5
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Now we can ask questions. For example, we want to know whether a mushroom with an almondy smell (In our case, this is the intercept) is poisonous. The point prediction can be calculated:

```
invlogit(-6.5)
```

```
## [1] 0.001501182
```

Thus, based off this model and data, there is a 0.15% chance that a mushroom with an almondy smell is poisonous.

Another Example: A fishy smelling mushroom has a 99.9958922% or invlogit(-6.2 + 16.3*1) chance of being poisonous.

Using a Simulation to Understand the Priors The prior is the reason the probability of a poisonous mushroom is not 100% when the mushroom smells fishy. So this got me thinking about priors and the code below (from Gelman and Vehtari) allows a nifty comparison of the model output between the traditional and Bayesian models.

```
log_sim <- function(n, b_0, b_1){
x <- runif(n, -11, 11) # generate x-values that can range from -11 to 11
z <- rlogis(n, b_0 + b_1*x, 1) # need the logistic distribution as the link function to generate the y'
# now generate a binary y using the probability from the link function/distribution
y <- ifelse(z>0, 1, 0) # Think about the latent variable formulation where there are cutoff points
sim_data <- data.frame(x, y, z)
glm_m <- glm(y ~ x, family = binomial(link = "logit"), data = sim_data) # fit glm model
stan_m <- stan_glm(y ~ x, family = binomial(link = "logit"), data = sim_data, refresh = 0)
display(glm_m, digits=1)
print(stan_m, digits=1)
}

log_sim(10, -2, .2) # 10 observations from -2 + 0.2*x model
```

```
## glm(formula = y ~ x, family = binomial(link = "logit"), data = sim_data)
##             coef.est coef.se
## (Intercept) -2.0      1.9
## x            0.6      0.4
## ---
##   n = 10, k = 2
##   residual deviance = 5.3, null deviance = 13.5 (difference = 8.2)
## stan_glm
##  family:       binomial [logit]
##  formula:      y ~ x
##  observations: 10
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) -1.2    1.0
## x            0.4    0.2
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
log_sim(1000, -2, .2) # 1000 observations from -2 + 0.2*x model
```

```
## glm(formula = y ~ x, family = binomial(link = "logit"), data = sim_data)
##             coef.est coef.se
## (Intercept) -2.0      0.1
## x            0.2      0.0
## ---
##   n = 1000, k = 2
##   residual deviance = 769.6, null deviance = 933.6 (difference = 164.0)
## stan_glm
##  family:       binomial [logit]
##  formula:      y ~ x
##  observations: 1000
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) -2.0    0.1
## x            0.2    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

When $n = 10$ both of the models are noisy, but the stan_model is less noisy. By the time, we have 1000 "observations" both models recover the simulated coefficients.

Next:

I am curious about predictors that do not perfectly predict the outcome.

- Gill Size: broad=b,narrow=n
- Gill Attachment: attached=a, free=f

```
mosaic::tally(class ~ gill.size, data = mush, margins = T)
```

```
##        gill.size
## class      b    n
##   e     3920  288
##   p     1692 2224
##   Total 5612 2512
```

```
mosaic::tally(class ~ gill.attachment, data = mush, margins = T)
```

```
##        gill.attachment
## class      a    f
##   e      192 4016
##   p       18 3898
##   Total  210 7914
```

- The table tells us that approximately 49% of the mushrooms with a free gill attachment type are poisonous. So, gill attachment style should not be a highly informative predictor.

```
gill_m <- stan_glm(class ~ gill.size + gill.attachment, family = binomial(link = "logit"),
                     data = mush, refresh = 0)
print(gill_m)
```

```
## stan_glm
##  family:       binomial [logit]
##  formula:      class ~ gill.size + gill.attachment
##  observations: 8124
##  predictors:   3
## ------
##                   Median MAD_SD
## (Intercept)      -2.4    0.2
## gill.sizen        2.8    0.1
## gill.attachmentf  1.6    0.2
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Credible interval interpretations

ROS (Regression and Other Stories) does not describe the logistic regression credible intervals in detail. This is probably because the interpretations are generally the same. It is useful to look at the standard errors of the estimate to understand the uncertainty of the estimate, but Gelman, Hill and Vehtari do not recommend using “statistical significance” as a criteria for including predictors in a model.

Example: In the above model, the smallest effect size estimate is the gill attachment style. So, a broad size gill with a free attachment style has a 31.0025519% (invlogit(-2.4 + 1.6)) chance of being poisonous. The standard error of 0.2 indicates that a β in the range of [1.6 ± 2*0.2] = [1.2, 2] would be consistent with the data.

Further Questions

In the future, it would be valuable to look at interactions in a logistic regression model. I would like to know more about how they are interpreted (or how uninterpretable they may be).