Classification: Beyond a Majority Vote

36-600

Fall 2021

Dataset

Our example dataset contains 16 predictor measurements for each of 10,000 stars: 5000 so-called contact binary stars (or CBs) and 5000 stars that are not contact binaries (dubbed NON-CBs).

```
response = factor(response.new,levels=c("NON-CB","CB"))
set.seed(101)
s = sample(length(response),round(0.7*length(response)))
pred.train = predictors[s,]
pred.test  = predictors[-s,]
resp.train = response[s]
resp.test  = response[-s]
```

Note the first line above. When one defines a factor variable, by default the levels are defined via alphabetical order. But here, we envision that NON-CB is a "negative" response (and thus naturally to be associated with value 0) and CB is a "positive" response, so we assign the levels to be in this order. (You don't have to do this, so long as you are careful when interpreting your confusion matrix. You should just know that you have the option of moving your factor levels around!)

Logistic Regression Analysis

```
log.out = suppressWarnings(glm(resp.train~.,data=pred.train,family=binomial))
resp.prob = predict(log.out,newdata=pred.test,type="response")
resp.pred = factor(ifelse(resp.prob>0.5,"CB","NON-CB"),levels=c("NON-CB","CB"))
(log.mcr = round(mean(resp.pred!=resp.test),3))
```

```
## [1] 0.278
```

```
table(resp.pred,resp.test)
```

```
##          resp.test
## resp.pred NON-CB   CB
##    NON-CB   1030  405
##    CB        430 1135
```

The Classification Threshold

Let's look at the estimated probabilities for class `CB`:

```
predict(log.out,newdata=pred.test,type="response")[1:3]
```

```
##     30601     21545     37506
## 0.8905524 0.9001895 0.8420999
```

Here, the first row represents the first test-set object: the decision tree estimates that there is a 89.1% chance that it is a contact binary. For objects 2 and 3, the probabilities are 90.0% and 84.2%. Etc. Because we assumed that the default threshold for choosing one class over the other is 50%, these three objects will be classified as contact binaries.

Altering the Classification Threshold

We can change the threshold by hand. For instance, if we want to map all probabilities of being a contact binary of 0.65 and above to the class `CB`, rather than 0.5 and above, we just change the number:

```
resp.pred = factor(ifelse(resp.prob>0.65,"CB","NON-CB"),levels=c("NON-CB","CB"))
(rpart.mcr = mean(resp.pred!=resp.test))
```

```
## [1] 0.331
```

```
table(resp.pred,resp.test)
```

```
##          resp.test
## resp.pred NON-CB   CB
##    NON-CB   1255  788
##    CB        205  752
```

We find that changing the threshold to 0.65 may not actually be a good thing to do: the MCR went up. Besides that, we see that our ability to identify contact binaries is diminished.

Why then do I say "may not actually be a good thing to do"? Well, there is an upside: our ability to identify the alternative class (`NON-CB`) is enhanced. If our goal is to produce a "pure" catalog of variables that are not contact binaries, then increasing the threshold from, e.g., 0.5 to 0.65 is actually a good thing to do. (In fact, maybe you want to set the threshold even higher!) (Another way of stating this: if constructing a pure catalog representative of one of the classes is our goal, then the MCR is not going to be the correct model assessment metric to use.)

Receiver Operating Characteristics (ROC) Curve

One can examine how the threshold matters by repeatedly calling `predict()` with different cutoff arguments. But a far more efficient approach is to generate a so-called *receiver operating characteristics* curve, or *ROC* curve. Let's simply run one and look at the output:

```
library(pROC)
resp.prob = predict(log.out,newdata=pred.test,type="response")
(roc.log = roc(resp.test,resp.prob))
```

```
## Setting levels: control = NON-CB, case = CB

## Setting direction: controls < cases

##
## Call:
## roc.default(response = resp.test, predictor = resp.prob)
##
## Data: resp.prob in 1460 controls (resp.test NON-CB) < 1540 cases (resp.test CB).
## Area under the curve: 0.789
```

Receiver Operating Characteristics (ROC) Curve

```
names(roc.log)
```

```
##  [1] "percent"            "sensitivities"      "specificities"      "thresholds"         "direction"
##  [6] "cases"              "controls"           "fun.sesp"           "auc"                "call"
## [11] "original.predictor" "original.response"  "predictor"          "response"           "levels"
```

The first part of the output to highlight is `thresholds`: this is a vector of different cutoff values...for our example, 2999 of them!

```
roc.log$thresholds[1:10]
```

```
##  [1]         -Inf 9.276178e-14 1.511849e-12 9.007357e-07 2.238489e-06 3.442148e-06 4.329776e-06 7.355695e-06 1.357
## [10] 2.778007e-04
```

ROC Curve

We also notice `sensitivities` and `specificities`. These make most sense if I lay out the following matrix first:

True Negative (TN)	False Negative (FN)
False Positive (FP)	True Positive (TP)

(Note: some sources, like wikipedia, switch the rows and columns; as long as you know what is where, it ultimately doesn't make a difference which convention you choose.) A "true positive," or TP, is classifying a `CB` as a `CB`, while a "false positive," or FP, is classifying a `NON-CB` as a `CB`, etc.

- sensitivity or recall or true positive rate or completeness: TP/(TP+FN), evaluated on right column
- specificity or true negative rate: TN/(TN+FP), evaluated on left column
- purity or positive predictive value: TP/(TP+FP), evaluated on bottom row

Ultimately, how you combine the elements of a confusion matrix into a model assessment metric is up to you. Keep in mind that while the misclassification rate is a classic metric, it does implicitly give the same weight to both types of misclassification. What if you are predicting whether someone has cancer? What's worse:

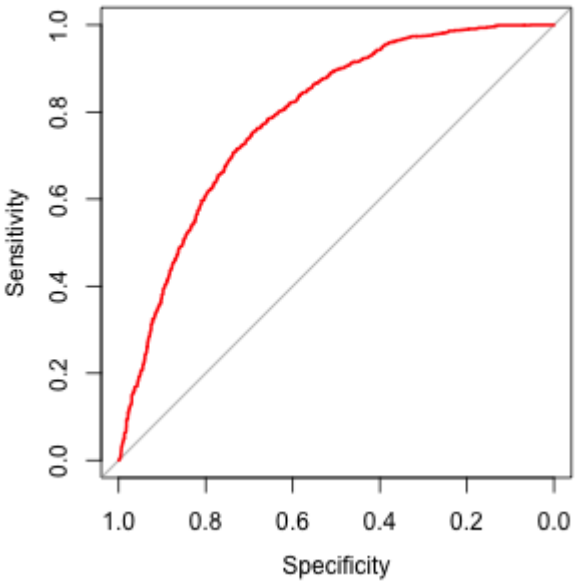
- telling a person with cancer that they don't have it; or
- telling a person without cancer that they do have it?

(There's no clear right answer, but you may feel one is "worse" than the other, and so you would be compelled to weight the misclassifications differently.)

ROC Curve

A ROC curve shows the sensitivity (how well we identify the positive class) versus the specificity (how well we identify the negative class) as a function of classification threshold:

```
plot(roc.log,col="red",xlim=c(1,0),ylim=c(0,1))         # Another non-ggplot production
```



The diagonal line represents the baseline of "random guessing." The better the overall performance of a classifier, the more the curve moves towards the upper left.

Area Under Curve (AUC)

One means to compare classifiers that goes beyond misclassification rates based on majority vote is to compute the area under the ROC curve. For random guessing, the area is 0.5. For perfect classification at all thresholds, the AUC is 1.

```
cat("AUC for logistic regression: ",round(roc.log$auc,3),"\n")
```

```
## AUC for logistic regression:  0.789
```

The advantage of AUC over simple MCR via majority vote is that it attempts to take into account all possible threshold values at once, i.e., it allows us to identify which classifier performs the best over a range of conditions.

Determining an Optimal Classification Threshold Value

By introducing ROC curves and the concept of AUC, we now have a new metric for model selection. But ultimately, we do need to choose *one* threshold value so as to generate predictions. We can use the ROC curve to do this, while acknowledging that there is no unique means to choose that one value.

An often-used choice is *Youden's J* statistic:

$$\mathrm{J} \;=\; \text{sensitivity} \;+\; \text{specificity} \;-\; 1\,.$$

Basically, we want to determine the threshold value that allows us to best identify `NON-CBs` and `CBs` at the same time. For our example:

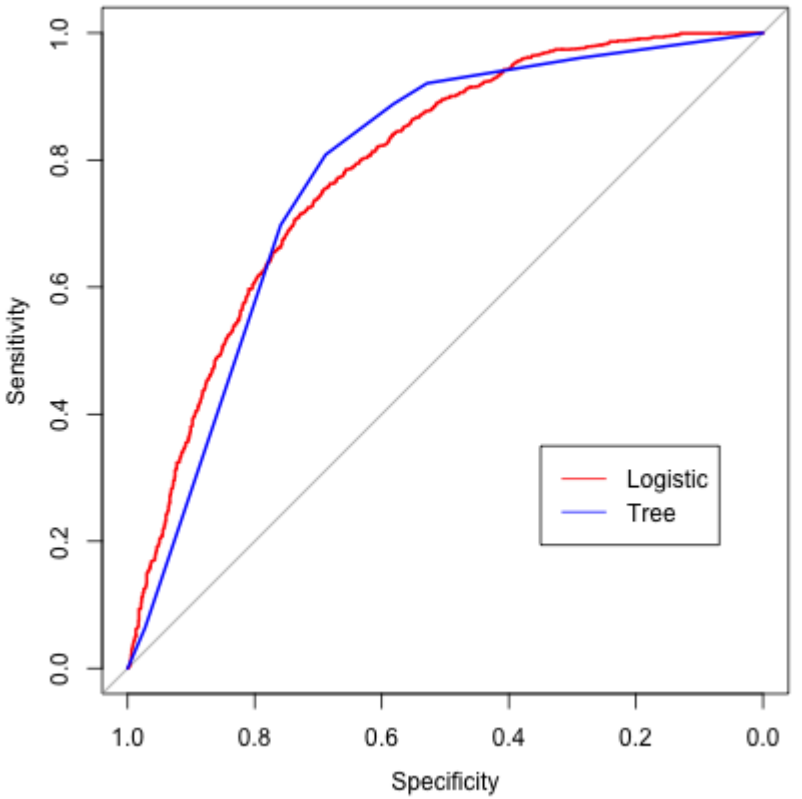
```
J = roc.log$sensitivities + roc.log$specificities - 1
w = which.max(J)
cat("Optimum threshold for logistic regression: ",round(roc.log$thresholds[w],3),"\n")
```

```
## Optimum threshold for logistic regression:  0.488
```

The value is close to 0.5, which is not surprising because our data had balanced classes (i.e., just as many `CBs` as `NON-CBs`). When the data feature unbalanced classes, optimal thresholds may differ greatly from 0.5.

Does a Decision Tree Do Better?

```
## AUC for decision tree:  0.783
```




```
## Optimum threshold for decision tree:  0.497
```

Logistic regression and a classification tree get similar results!

A Classification Analysis Workflow

We will wrap this up by stating what you might do when analyzing binary categorical response data:

1. Learn a model as before and generate response class probabilities for each test-set datum using `predict()`.
2. Using these probabilities as input, generate a ROC curve using the `roc()` function.
3. Use the output from `roc()` to determine the optimal threshold for your chosen metric (e.g., Youden's J statistic). (Report the metric and the threshold.)
4. Rerun `predict()` with the optimal threshold encoded in the `cutoff` parameter and generate *class predictions* for each test-set datum.
5. Use the output to create a final confusion matrix (for visualization) and, e.g., a final misclassification rate.
6. Ultimately pick the model with, e.g., the highest AUC value, or the lowest MCR value, or...

Done.