Demography Predictive Modeling Working Group - Cross-validation of models

Corey Sparks, Ph.D.

11/19/2019

Classification models

I would suggest you read section 5.1 of Introduction to Statistical Learning to get a full treatment of this topic

In classification methods, we are typically interested in using some observed characteristics of a case to predict a binary categorical outcome. This can be extended to a multi-category outcome, but the largest number of applications involve a 1/0 outcome.

In these examples, we will use the Demographic and Health Survey Model Data. These are based on the DHS survey, but are publicly available and are used to practice using the DHS data sets, but don't represent a real country.

In this example, we will use the outcome of contraceptive choice (modern vs other/none) as our outcome.

```r
library(haven)
dat<-url("https://github.com/coreysparks/data/blob/master/ZZIR62FL.DTA?raw=true")
model.dat<-read_dta(dat)
```

Here we recode some of our variables and limit our data to those women who are not currently pregnant and who are sexually active.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
model.dat2<-model.dat%>%
  mutate(region = v024,
         modcontra= as.factor(ifelse(v364 ==1,"Modcontra", "NoModContra")),
         age = cut(v012, breaks = 5),
         livchildren=v218,
         educ = v106,
         currpreg=v213,
         knowmodern=ifelse(v301==3, 1, 0),
         age2=v012^2,
         rural = ifelse(v025==2, 1,0),
         wantmore = ifelse(v605%in%c(1,2), 1, 0))%>%
  filter(currpreg==0, v536>0)%>% #notpreg, sex active
  dplyr::select(caseid, region, modcontra,age, age2,livchildren, educ, knowmodern, rural, wantmore)
```

```
knitr::kable(head(model.dat2))
```

caseid	region	modcontra	age	age2	livchildren	educ	knowmodern	rural	wantmore
1 1 2	2	NoModContra	(28.6,35.4]	900	4	0	1	1	1
1 4 2	2	NoModContra	(35.4,42.2]	1764	2	0	1	1	0
1 4 3	2	NoModContra	(21.8,28.6]	625	3	1	1	1	0
1 5 1	2	NoModContra	(21.8,28.6]	625	2	2	1	1	1
1 6 2	2	NoModContra	(35.4,42.2]	1369	2	0	1	1	1
1 6 3	2	NoModContra	(15,21.8]	289	0	2	0	1	1

Cross-validation of predictive models

The term cross-validation refers to fitting a model on a subset of data and then testing it on another subset of the data. Typically this process is repeated several times.

The simplest way of doing this is to leave out a single observation, refit the model without it in the data, then predict its value using the rest of the data. This is called **hold out** cross-validation.

K-fold cross-validation is a process where you leave out a "group" of observations, it is as follows:

1. Randomize the data
2. Split the data into k groups, where k is an integer
3. For each of the k groups,
    - Take one of the groups as a hold out test set
    - Use the other k-1 groups as training data
    - Fit a model using the data on the k-1 groups, and test it on the hold out group
    - Measure predictive accuracy of that model, and **throw the model away!**
4. Summarize the model accuracy over the measured model accuracy metrics

A further method is called **leave one out, or LOO** cross-validation. This combines hold out and k-fold cross-validation.

Why?

By doing this, we can see how model accuracy is affected by particular individuals, and overall allows for model accuracy to be measured repeatedly so we can assess things such as model **tuning parameters**.

If you remember from last time, the regression partition (rpart) analysis depended upon us choosing a good value for the **complexity parameter, CP**. In a cross-validation analysis, we can use the various resamplings of the data to examine the model's accuracy sensitivity to alternative values of this parameter.

This evaluation can either be done systematically, along a grid, or using a random search.

Alternative accuracy measures

We talked last time about using model accuracy as a measure of overall fit. This was calculated using the observed and predicted values of our outcome. For classification model, another commonly used metric of model predictive power is the Receiver Operating Characteristics (**ROC**) curve. This is a probability curve, and is often accompanied by the area under the curve (**AUC**) measure, which summarizes the separability of the classes. Together they tell you how capable the model is of determining difference between the classes in the data. The higher the values of these, the better, and they are both bound on (0,1).

A nice description of these are found here.

Regression partition tree

As we saw in the first working group example, the regression tree is another common technique used in classification problems. Regression or classification trees attempt to find optimal splits in the data so that the best classification of observations can be found.

Create design matrix

If we have a mixture of factor variables and continuous predictors in our analysis, it is best to set up the design matrix for our models before we run them. Many methods within `caret` won't use factor variables correctly unless we set up the dummy variable representations first.

```
datmat<-model.matrix(~factor(region)-1+factor(age)+livchildren+rural+wantmore+factor(educ)-1, data=model
datmat<-data.frame(datmat)
datmat$modcontra<- model.dat2$modcontra

head(datmat)
```

```
##   factor.region.1 factor.region.2 factor.region.3 factor.region.4
## 1               0               1               0               0
## 2               0               1               0               0
## 3               0               1               0               0
## 4               0               1               0               0
## 5               0               1               0               0
## 6               0               1               0               0
##   factor.age..21.8.28.6. factor.age..28.6.35.4. factor.age..35.4.42.2.
## 1                      0                      1                      0
## 2                      0                      0                      1
## 3                      1                      0                      0
## 4                      1                      0                      0
## 5                      0                      0                      1
## 6                      0                      0                      0
##   factor.age..42.2.49. livchildren rural wantmore factor.educ.1
## 1                    0           4     1        1             0
## 2                    0           2     1        0             0
## 3                    0           3     1        0             1
## 4                    0           2     1        1             0
## 5                    0           2     1        1             0
## 6                    0           0     1        1             0
##   factor.educ.2 factor.educ.3   modcontra
## 1             0             0 NoModContra
## 2             0             0 NoModContra
## 3             0             0 NoModContra
## 4             1             0 NoModContra
## 5             0             0 NoModContra
## 6             1             0 NoModContra
```

```
summary(datmat)
```

```
##  factor.region.1  factor.region.2  factor.region.3  factor.region.4
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.000
##  Mean   :0.3855   Mean   :0.2586   Mean   :0.1669   Mean   :0.189
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000  
##  factor.age..21.8.28.6. factor.age..28.6.35.4. factor.age..35.4.42.2.
##  Min.   :0.0000         Min.   :0.0000         Min.   :0.0000        
##  1st Qu.:0.0000         1st Qu.:0.0000         1st Qu.:0.0000        
##  Median :0.0000         Median :0.0000         Median :0.0000        
##  Mean   :0.2448         Mean   :0.2168         Mean   :0.1605        
##  3rd Qu.:0.0000         3rd Qu.:0.0000         3rd Qu.:0.0000        
##  Max.   :1.0000         Max.   :1.0000         Max.   :1.0000        
##  factor.age..42.2.49.  livchildren         rural           wantmore     
##  Min.   :0.0000       Min.   : 0.000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000       1st Qu.: 1.000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :0.0000       Median : 2.000   Median :1.0000   Median :1.0000  
##  Mean   :0.1255       Mean   : 2.555   Mean   :0.5934   Mean   :0.5209  
##  3rd Qu.:0.0000       3rd Qu.: 4.000   3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000       Max.   :11.000   Max.   :1.0000   Max.   :1.0000  
##  factor.educ.1    factor.educ.2    factor.educ.3           modcontra   
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Modcontra  :1761  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   NoModContra:5044  
##  Median :0.0000   Median :0.0000   Median :0.00000                     
##  Mean   :0.1187   Mean   :0.2575   Mean   :0.03542                     
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.00000                     
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000                     
```

using caret to create training and test sets.

We use an 80% training fraction

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
set.seed(1115)
train<- createDataPartition(y = datmat$modcontra , p = .80, list=F)

dtrain<-datmat[train,c(-1,-5, -9)]
dtest<-datmat[-train,c(-1,-5, -9)]

table(dtrain$modcontra)
```

```
## 
##   Modcontra NoModContra 
##        1409        4036
```

```r
prop.table(table(dtrain$modcontra))
```

```
## 
##   Modcontra NoModContra 
##   0.2587695   0.7412305
```

```r
summary(dtrain)
```

```
##  factor.region.2  factor.region.3 factor.region.4  factor.age..28.6.35.4.
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000        
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000        
##  Median :0.0000   Median :0.000   Median :0.0000   Median :0.0000        
##  Mean   :0.2573   Mean   :0.168   Mean   :0.1914   Mean   :0.2101
##  3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000
##  factor.age..35.4.42.2. factor.age..42.2.49.     rural
##  Min.   :0.0000         Min.   :0.0000       Min.   :0.0000
##  1st Qu.:0.0000         1st Qu.:0.0000       1st Qu.:0.0000
##  Median :0.0000         Median :0.0000       Median :1.0000
##  Mean   :0.1638         Mean   :0.1225       Mean   :0.5927
##  3rd Qu.:0.0000         3rd Qu.:0.0000       3rd Qu.:1.0000
##  Max.   :1.0000         Max.   :1.0000       Max.   :1.0000
##     wantmore     factor.educ.1    factor.educ.2    factor.educ.3
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :1.000   Median :0.0000   Median :0.0000   Median :0.00000
##  Mean   :0.521   Mean   :0.1188   Mean   :0.2597   Mean   :0.03581
##  3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##        modcontra
##  Modcontra  :1409
##  NoModContra:4036
##
##
##
##
```

```
summary(dtest)
```

```
##  factor.region.2 factor.region.3  factor.region.4  factor.age..28.6.35.4.
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.264   Mean   :0.1625   Mean   :0.1794   Mean   :0.2434
##  3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  factor.age..35.4.42.2. factor.age..42.2.49.     rural
##  Min.   :0.0000         Min.   :0.0000       Min.   :0.0000
##  1st Qu.:0.0000         1st Qu.:0.0000       1st Qu.:0.0000
##  Median :0.0000         Median :0.0000       Median :1.0000
##  Mean   :0.1471         Mean   :0.1375       Mean   :0.5963
##  3rd Qu.:0.0000         3rd Qu.:0.0000       3rd Qu.:1.0000
##  Max.   :1.0000         Max.   :1.0000       Max.   :1.0000
##     wantmore      factor.educ.1    factor.educ.2    factor.educ.3
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :0.00000
##  Mean   :0.5206   Mean   :0.1184   Mean   :0.2485   Mean   :0.03382
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##        modcontra
##  Modcontra  : 352
##  NoModContra:1008
##
##
##
##
```

Set up caret for 10 fold cross-validation

To set up the training controls for a caret model, we typically have to specify the type of re-sampling method, the number of resamplings, the number of repeats (if you're doing repeated sampling). Here we will do a 10 fold cross-validation, 10 is often recommended as a choice for k based on experimental sensitivity analysis.

The other things we specify are:

- repeats - These are the number of times we wish to repeat the cross-validation, typically 3 or more is used
- classProbs = TRUE - this is necessary to assess accuracy in the confusion matrix
- search = "random" is used if you want to randomly search along the values of the tuning parameter
- sampling - Here we can specify alternative sampling methods to account for unbalanced outcomes
- SummaryFunction=twoClassSummary - keeps information on the two classes of the outcome
- savePredictions = T - have the process save all the predicted values throughout the process, we need this for the ROC curves

```
fitctrl <- trainControl(method="repeatedcv",
                        number=10,
                        repeats=3,
                        classProbs = TRUE,
                     search="random",
                     sampling = "down",
                     summaryFunction=twoClassSummary,
                     savePredictions = "all")
```

Train models using caret

Here we use caret to fit the rpart model

```
library(rpart)
rp1<-caret::train(modcontra~.,
           data=dtrain,
           metric="ROC",
           method ="rpart",
          tuneLength=20, #try 20 random values of the tuning parameters
           trControl=fitctrl)

rp1
```

```
## CART
##
## 5445 samples
##   11 predictor
##    2 classes: 'Modcontra', 'NoModContra'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4901, 4900, 4900, 4901, 4900, 4901, ...
## Addtional sampling using down-sampling
##
## Resampling results across tuning parameters:
##
##   cp            ROC        Sens       Spec
##   0.0000000000  0.7118201  0.6656991  0.6573267
##   0.0001182872  0.7118201  0.6656991  0.6573267
##   0.0001419446  0.7118201  0.6656991  0.6573267
##   0.0002365744  0.7115241  0.6654627  0.6573277
##   0.0007097232  0.7078988  0.6671310  0.6574947
##   0.0012775018  0.7042976  0.6732945  0.6570073
##   0.0033120416  0.6940002  0.6654779  0.6620476
##   0.0035486160  0.6919881  0.6586221  0.6662556
##   0.0106458481  0.6818322  0.6709591  0.6462660
##   0.0124201561  0.6809661  0.6844343  0.6344422
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0001419446.
```

```
gl1<-caret::train(modcontra~.,
           data=dtrain,
           metric="ROC",
           method ="glm",
           #family=binomial,
          #tuneLength=20, #try 20 random values of the tuning parameters
           trControl=fitctrl)

gl1
```

```
## Generalized Linear Model
##
## 5445 samples
##   11 predictor
##    2 classes: 'Modcontra', 'NoModContra'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4901, 4900, 4900, 4900, 4900, 4901, ...
## Addtional sampling using down-sampling
##
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7241083  0.6394596  0.705647
```

```
##Accuracy on training set
predrp1<-predict(rp1, newdata=dtrain)
confusionMatrix(data = predrp1,dtrain$modcontra, positive = "Modcontra" )
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Modcontra NoModContra
##   Modcontra        1002        1389
##   NoModContra       407        2647
##
##                Accuracy : 0.6702
##                  95% CI : (0.6575, 0.6826)
##     No Information Rate : 0.7412
##     P-Value [Acc > NIR] : 1
##
```

```
##                   Kappa : 0.2991
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7111
##             Specificity : 0.6558
##          Pos Pred Value : 0.4191
##          Neg Pred Value : 0.8667
##              Prevalence : 0.2588
##          Detection Rate : 0.1840
##    Detection Prevalence : 0.4391
##       Balanced Accuracy : 0.6835
##
##        'Positive' Class : Modcontra
##
```

```
predgl1<-predict(gl1, newdata=dtrain)
confusionMatrix(data = predgl1,dtrain$modcontra, positive = "Modcontra" )
```

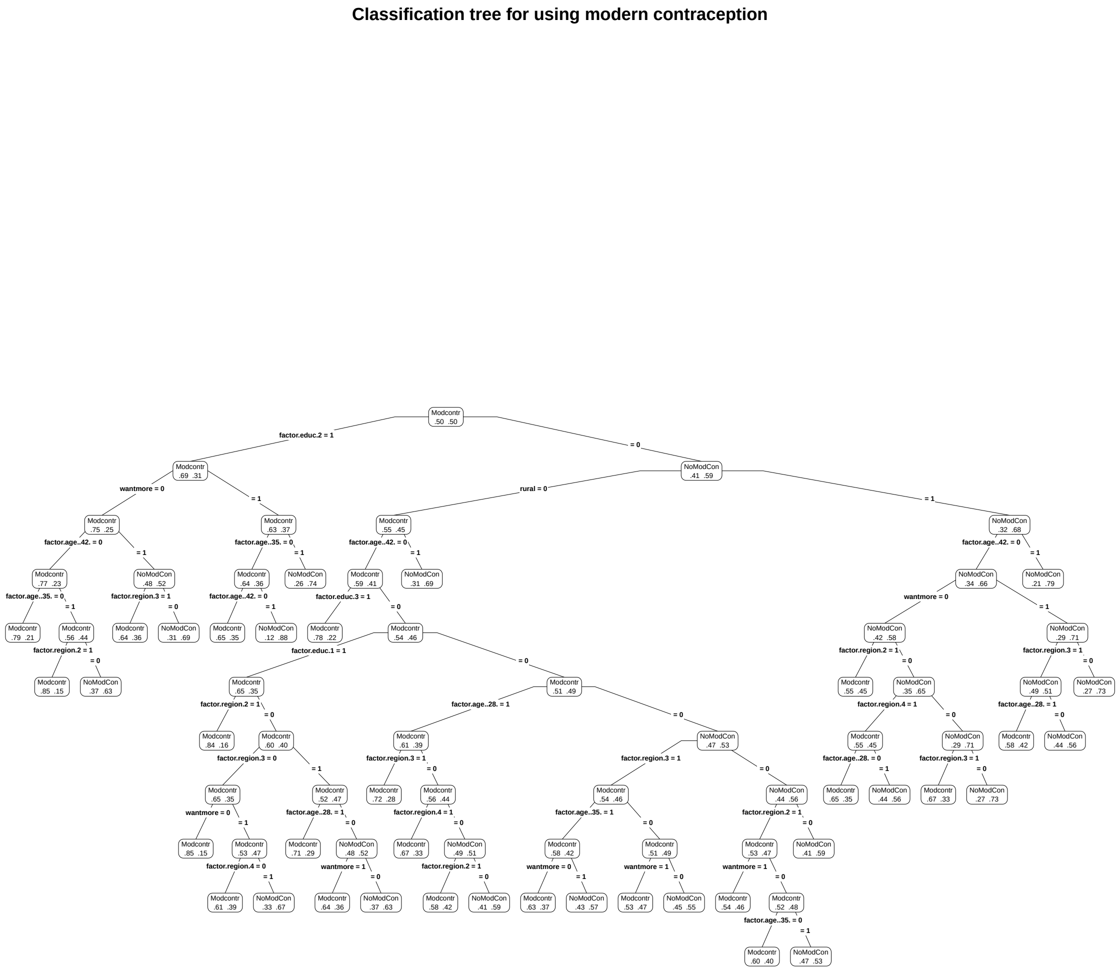
```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Modcontra NoModContra
##   Modcontra         902        1179
##   NoModContra       507        2857
##
##                Accuracy : 0.6904
##                  95% CI : (0.6779, 0.7026)
##     No Information Rate : 0.7412
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3013
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6402
##             Specificity : 0.7079
##          Pos Pred Value : 0.4334
##          Neg Pred Value : 0.8493
##              Prevalence : 0.2588
##          Detection Rate : 0.1657
##    Detection Prevalence : 0.3822
##       Balanced Accuracy : 0.6740
##
##        'Positive' Class : Modcontra
##
```

We see that by down sampling the more common level of the outcome, we end up with much more balanced accuracy in terms of specificity and sensitivity.

We can visualize the resulting “best fitting” model like we did before:

```
library(rpart.plot)
# rpart.plot(rp1$finalModel,
# box.palette="GnBu",
# shadow.col="gray",
# nn=TRUE, main="Classification tree for using modern contraception")

prp(rp1$finalModel,type=4, extra = 4,
    main="Classification tree for using modern contraception")
```



Classification tree for using modern contraception

You see that the best fitting model is much more complicated than the previous one. Each node box displays the classification, the probability of each class at that node (i.e. the probability of the class conditioned on the node) and the percentage of observations used at that node. From here.

ROC curve

The ROC curve can be shown for the model:

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

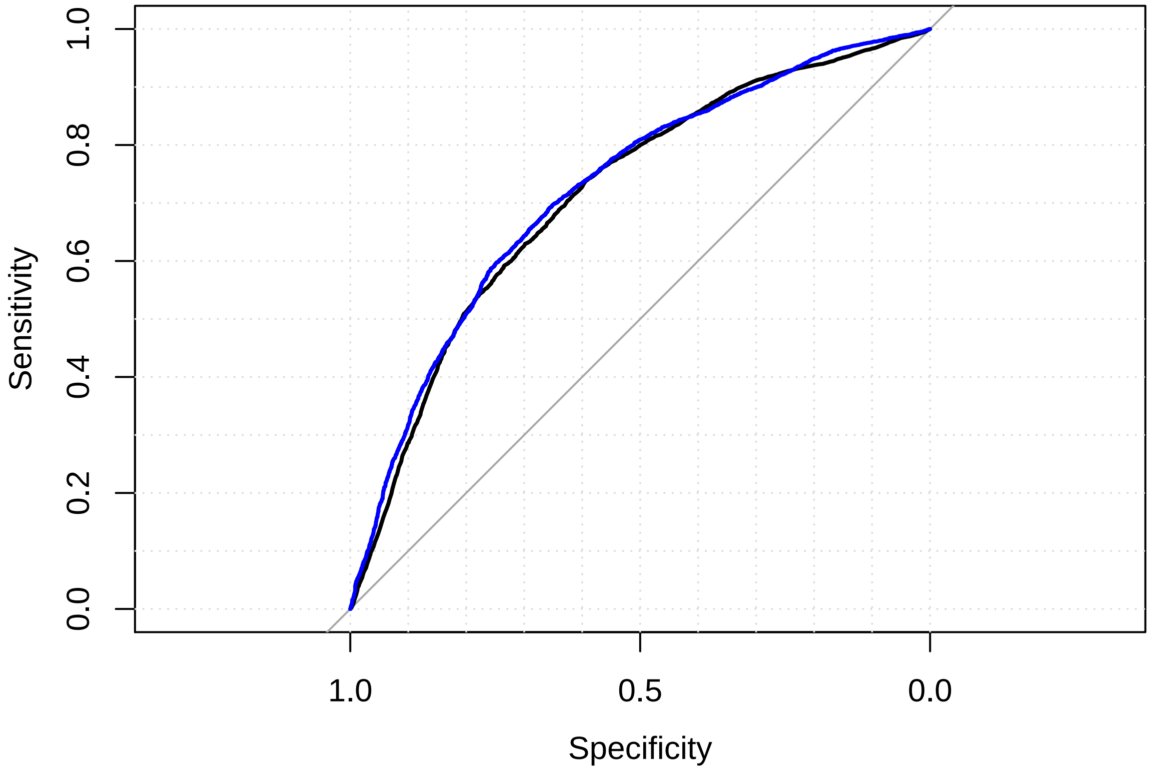
```
# Select a parameter setting
mycp<-rp1$pred$cp==rp1$bestTune$cp
selectedIndices <- rp1$pred$cp==mycp
# Plot:
plot.roc(rp1$pred$obs[selectedIndices], rp1$pred$Modcontra[selectedIndices], grid=T)
```

```
## Setting levels: control = Modcontra, case = NoModContra
```

```
## Setting direction: controls > cases
```

```
plot.roc(gl1$pred$obs, gl1$pred$Modcontra, add=T, col="blue")
```

```
## Setting levels: control = Modcontra, case = NoModContra
## Setting direction: controls > cases
```



```r
#Value of ROC and AUC
roc(rp1$pred$obs[selectedIndices],  rp1$pred$Modcontra[selectedIndices])
```

```
## Setting levels: control = Modcontra, case = NoModContra
## Setting direction: controls > cases

##
## Call:
## roc.default(response = rp1$pred$obs[selectedIndices], predictor = rp1$pred$Modcontra[selectedIndices]
##
## Data: rp1$pred$Modcontra[selectedIndices] in 4227 controls (rp1$pred$obs[selectedIndices] Modcontra)
## Area under the curve: 0.7137
```

```r
roc(gl1$pred$obs, gl1$pred$Modcontra)
```

```
## Setting levels: control = Modcontra, case = NoModContra
## Setting direction: controls > cases

##
## Call:
## roc.default(response = gl1$pred$obs, predictor = gl1$pred$Modcontra)
##
## Data: gl1$pred$Modcontra in 4227 controls (gl1$pred$obs Modcontra) > 12108 cases (gl1$pred$obs NoMod
## Area under the curve: 0.7232
```

```r
auc(rp1$pred$obs[selectedIndices],  rp1$pred$Modcontra[selectedIndices])
```

```
## Setting levels: control = Modcontra, case = NoModContra
## Setting direction: controls > cases

## Area under the curve: 0.7137
```

```r
auc(gl1$pred$obs, gl1$pred$Modcontra)
```

```
## Setting levels: control = Modcontra, case = NoModContra
## Setting direction: controls > cases

## Area under the curve: 0.7232
```

```r
roc.test(roc(gl1$pred$obs, gl1$pred$Modcontra),roc(rp1$pred$obs[selectedIndices],  rp1$pred$Modcontra[s
```

```
## Setting levels: control = Modcontra, case = NoModContra
## Setting direction: controls > cases

## Setting levels: control = Modcontra, case = NoModContra

## Setting direction: controls > cases

##
##  DeLong's test for two ROC curves
##
## data:  roc(gl1$pred$obs, gl1$pred$Modcontra) and roc(rp1$pred$obs[selectedIndices], rp1$pred$Modcont
## D = -1.4705, df = 32651, p-value = 0.1414
## alternative hypothesis: true difference in AUC is not equal to 0
## sample estimates:
## AUC of roc1 AUC of roc2
##   0.7231722   0.7136719
```

Assess fit on test data

```r
predrp1<-predict(rp1, newdata=dtest)
confusionMatrix(data = predrp1,dtest$modcontra, positive = "Modcontra" )
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Modcontra NoModContra
##   Modcontra         238         351
##   NoModContra       114         657
##
##                Accuracy : 0.6581
##                  95% CI : (0.6322, 0.6833)
##     No Information Rate : 0.7412
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.269
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6761
##             Specificity : 0.6518
##          Pos Pred Value : 0.4041
##          Neg Pred Value : 0.8521
##              Prevalence : 0.2588
##          Detection Rate : 0.1750
##    Detection Prevalence : 0.4331
##       Balanced Accuracy : 0.6640
##
##        'Positive' Class : Modcontra
##
```

```r
predgl1<-predict(gl1, newdata=dtest)
confusionMatrix(data = predgl1,dtest$modcontra, positive = "Modcontra" )
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Modcontra NoModContra
##   Modcontra         214         288
##   NoModContra       138         720
##
##                Accuracy : 0.6868
##                  95% CI : (0.6614, 0.7114)
##     No Information Rate : 0.7412
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.283
##
##  Mcnemar's Test P-Value : 5.234e-13
##
##             Sensitivity : 0.6080
##             Specificity : 0.7143
##          Pos Pred Value : 0.4263
##          Neg Pred Value : 0.8392
##              Prevalence : 0.2588
##          Detection Rate : 0.1574
##    Detection Prevalence : 0.3691
##       Balanced Accuracy : 0.6611
##
##        'Positive' Class : Modcontra
##
```

So what?

By doing cross-validation, we can do a few things:

1. Assess model fit better by using iterated subsets of data
2. Tune model parameters by allowing them to vary across cross-validation samples