STAT 621 Lecture Notes
ROC Curves

A Receiver Operator Characteristic (ROC) curve provides a means to compare classifiers in terms of overall classification accuracy. Before we discuss this graphical summary, there are a few more details to note about classification and classification errors.

Recall that generally speaking, classification methods work by using a set of explanatory variables to estimate class probabilities, and assigning the response to the class that has the highest estimated probability. In the case of a binary response, this amounts to

Here the value 0.5 is called the `threshold`, and it can be shown that this value minimizes the `overall` error rate for a given classifier. However the threshold value affects the classifier's ability to correctly predict one class over the other. Given a binary response Y, whose levels we can consider "positive" and "negative", there are two types of errors we need to be concerned about. Let's define these below, along with their respective rates, known as *sensitivity* and *specificity*.

Sometimes we may be more concerned with correctly classifying "positive" responses, and less concerned with classifying "negative" responses. The following example (adapted from Chapter 4 of AISL) illustrates this situation.

Example: Predicting credit card default. Let Y be a binary variable indicating whether an individual defaults on a loan. Suppose the "positive" level is `Default` and the "negative" level is `No Default`. Consider two predictors, X_1=`student status` and X_2=`credit card balance`. What do sensitivity and specificity indicate here? Do you think that these are equally important to the loan company, or are they likely more concerned about one over the other?

The threshold value affects sensitivity and specificity differently. Here are some hypothetical estimated probabilities.

$\widehat{p}$	0.23	0.67	0.45	0.88	0.36	0.51	0.19	0.41	0.44	0.39
Y	0	1	1	1	0	1	0	0	1	0

- Predict responses using a threshold of 0.5. Compute the specificity and sensitivity.

- Repeat using a threshold of 0.4

The figure below (from Chapter 4 of AISL) shows the trade-off between sensitivity, speficity and overall error rate for the credit card default example.

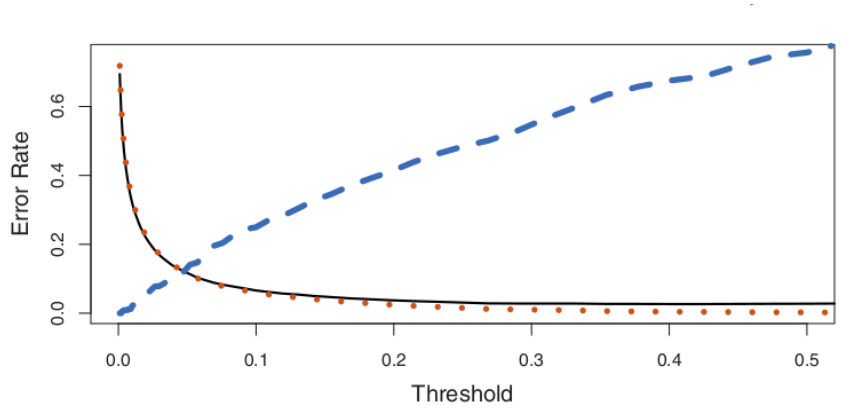


FIGURE 4.7. *For the* `Default` *data set, error rates are shown as a function of the threshold value for the posterior probability that is used to perform the assignment. The black solid line displays the overall error rate. The blue dashed line represents the fraction of defaulting customers that are incorrectly classified, and the orange dotted line indicates the fraction of errors among the non-defaulting customers.*

The **ROC** curve is a graphical summary of the overall performance of a classifier, for all possible threshold values. Typically, this is constructed as the false positive rate (sensitivity) on the y-axis, and the false negative rate (1-specificity) on the x-axis. These rates are plotted for thresholds from 1 to 0, moving from left to right along the x-axis. Below is the ROC curve from the credit card default example.

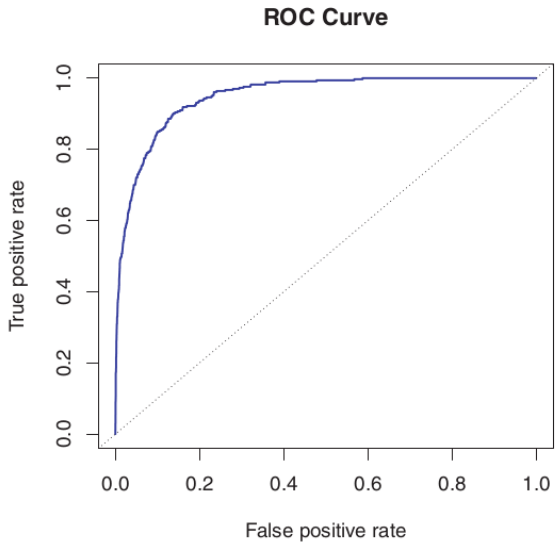


FIGURE 4.8. *A ROC curve for the LDA classifier on the* `Default` *data. It traces out two types of error as we vary the threshold value for the posterior probability of default. The actual thresholds are not shown. The true positive rate is the sensitivity: the fraction of defaulters that are correctly identified, using a given threshold value. The false positive rate is 1-specificity: the fraction of non-defaulters that we classify incorrectly as defaulters, using that same threshold value. The ideal ROC curve hugs the top left corner, indicating a high true positive rate and a low false positive rate. The dotted line represents the "no information" classifier; this is what we would expect if student status and credit card balance are not associated with probability of default.*

Some disucssion of this plot, including the *Area Under the Curve*, or AUC.

Example: The data set `care` is from a study of the quality of care received by diabetes patients at a certain hospital. The response is `PoorCare` which is coded as 1 if a patient received poor care and 0 if a patient recieved good care (or at least not poor). There are several predictors, as seen below. For this example we'll focus on trying to classify care as a function of two of these, `OfficeVisits` and `Narcotics`. First import all the libraries we will need. ROC curves are constructed with functions from the `ROCR` library.

```
library(caret)
library(e1071)
library(MASS)
library(ROCR)
library(klaR)
```

Note that the response `PoorCare` is binary, coded 0 and 1. First I designate this as a factor in R (by default R reads this in as a numeric variable). Then I divide the data set into training and test sets.

```
care$PoorCare=factor(care$PoorCare)
set.seed(492)
temp=sample(nrow(care), 99, replace=F)
care.train=care[temp,]
care.test=care[-temp,]
```

Logistic Regression Classifier: The logistic regression model is fit first below.

```
> care.glm=glm(PoorCare ~ OfficeVisits + Narcotics, data=care.train, family=binomial)
> summary(care.glm)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.48409    0.50728  -4.897 9.74e-07 ***
OfficeVisits  0.06453    0.02717   2.375  0.01757 *
Narcotics     0.10149    0.03307   3.068  0.00215 **
---
```

Use the fitted model to make predictions with a threshhold of 0.5. Note that for the moment, I'm predicting the response on the `training set`. This is what is done to plot the ROC curve. Here's the confusion matrix. Note that the option `positive = 1` indicates that the model predicts the probability of `PoorCare`. This designation affects the meaning of the sensitivity, specificity, and other statistics printed with the matrix.

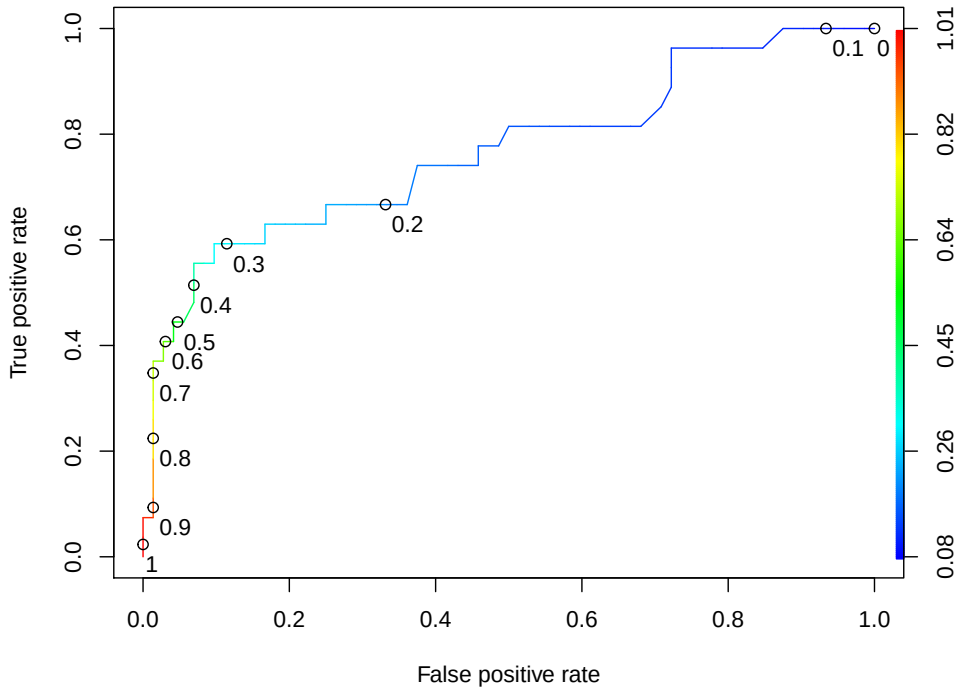
```
> phat.glm = predict(care.glm, type="response")
> predict.glm=factor(ifelse(phat.glm > .5,"1","0"))  #label the preds; make factor
> confusionMatrix(predict.glm, care.train$PoorCare, positive="1")

     Confusion Matrix and Statistics
         Reference
Prediction  0  1                          Accuracy : 0.8182
         0 69 15                          Sensitivity : 0.4444
         1  3 12                          Specificity : 0.9583
                                          Balanced Accuracy : 0.7014

          'Positive' Class : 1
```

Let's change the threshold. What if we predict poor care whenever the estimated probability exceeds 0.3?

```
> a = predict(care.glm, type="response")
> predict.glm=factor(ifelse(phat.glm > .3,"1","0"))  #label the preds; make factor
> confusionMatrix(predict.glm, care.train$PoorCare, positive="1")

  Confusion Matrix and Statistics

          Reference
Prediction  0  1                 Accuracy : 0.8081
         0 64 11                 Sensitivity : 0.5926
         1  8 16                 Specificity : 0.8889

                                 Balanced Accuracy : 0.7407
      'Positive' Class : 1
```

Now to make the ROC plot. First we use the `prediction` function to format the data. The arguments are the predicted probabilities (`phat.glm`) from the fitted model, and the observed responses. In addition, the `label.ordering` option specifies the order of the labels (negative, positive). The `performance` function computes the actual curve. The options `"tpr"` and `"fpr"` are typical; see the Help files for others. Finally, the curve is plotted.

```
ROCpred = prediction(phat.glm, care.train$PoorCare, label.ordering=c("0","1"))
perf=performance(ROCpred, "tpr", "fpr")
plot(perf,colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj=c(-0.2,1.7))
```



Ultimately the threshold value you pick depends on the data set and the objectives. For this example we may want to choose a value that maximizes the true positive rate while keeping the false positive rate really low. Let's try a threshold of 0.3 on the test data.

```
> phat.test = predict(care.glm, care.test, type="response")
> predict.test=factor(ifelse(phat.test > .3,"1","0"))  #label the preds; make factor
> confusionMatrix(predict.test, care.test$PoorCare, positive="1")

Confusion Matrix and Statistics
          Reference
Prediction  0  1                          Accuracy : 0.8125
         0 24  4                          Sensitivity : 0.3333
         1  2  2                          Specificity : 0.9231
                                           Balanced Accuracy : 0.6282
       'Positive' Class : 1
```

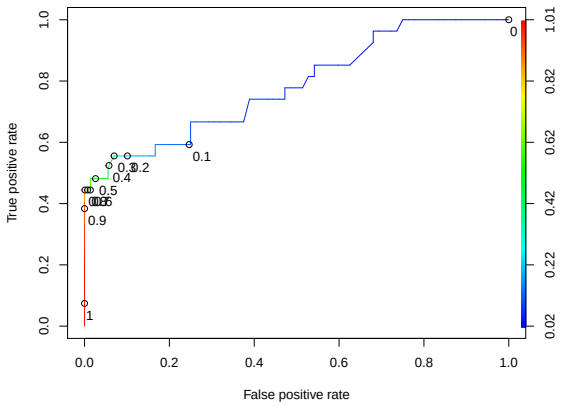
Interesting. Finally, the Area Under the Curve of a ROC curve is found with the `auc.perf` function. This returns a *performance object*; use the 'at' symbol to report just the area.

```
auc.perf = performance(ROCpred, measure = "auc")
auc.perf@y.values
[[1]]
[1] 0.7667181
```

Navie Bayes Classifier: Let's find the ROC curve for this classifier. First fit the classifier. Note that here `predict` returns a list. To make the ROC curve, we need the posterior probability of a "1" response. Below I plot the curve and compute the AUC.

```
nb.train=NaiveBayes(PoorCare ~ OfficeVisits + Narcotics, data=care.train, usekernel=T)
nb.pred=predict(nb.train, care.train)
names(nb.pred)
    [1] "class"     "posterior"
    head(nb.pred$posterior)
                0          1
     94 0.9422177 0.05778235
     85 0.9712957 0.02870434
     69 0.9786916 0.02130839
ROCpred.nb = prediction(nb.pred$posterior[,2], care.train$PoorCare, label.ordering=c("0","1"))
perf.nb=performance(ROCpred.nb, "tpr", "fpr")
plot(perf.nb,colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj=c(-0.2,1.7))
```

```
> auc.perf.nb = performance(ROCpred, measure = "auc")
> auc.perf.nb@y.values
[[1]]
[1] 0.7842078
```



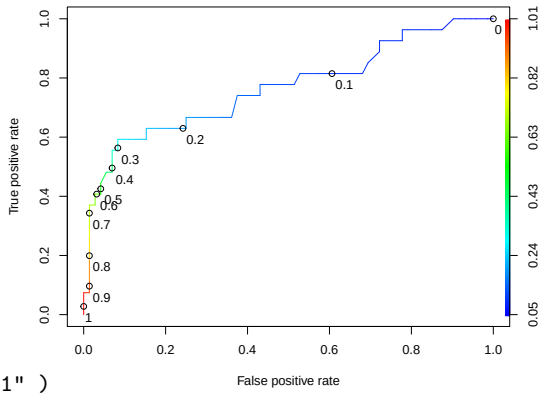
It looks like 0.3 would make a pretty good threshold again. Let's see how it does on the test data. From the previous page, note that `nb.pred` contains a matrix with the posterior probabilities. We want the 2nd column of this matrix, corresponding to the estimated probatilities that $Y = 1$. Below I use these for classification with a threshold of 0.3, and create the confusion matrix.

```
> nb.phat=nb.pred$posterior[,2]
> nb.3=factor(ifelse(nb.phat>.2, "1","0"))
> confusionMatrix(nb.3, care.test$PoorCare, positive="1" )

Confusion Matrix and Statistics

          Reference
Prediction  0  1                       Accuracy : 0.8125
         0 24  4                       Sensitivity : 0.3333
         1  2  2                       Specificity : 0.9231
                                       Balanced Accuracy : 0.6282
     'Positive' Class : 1
```

Well, we end up with the same table as the logistic model.

Linear Discriminant Analysis: Here is the fit of the LDA model, and the ROC curve.

```
lda.fit=lda(PoorCare!OfficeVisits + Narcotics, data=care.train)
lda.pred=predict(lda.fit, care.train)
lda.phat=lda.pred$
ROCpred.lda=prediction(lda.phat, care.train$PoorCare, label.ordering=c("0","1"))
perf.lda=performance(ROCpred.lda, "tpr", "fpr")
plot(perf.lda, colorize=T, print.cutoffs.at=seq(0.1,.1), text.adj=c(-.2,1.7))

auc.perf.lda = performance(ROCpred.lda, measure = "auc")
auc.perf.lda@y.values
[[1]]
[1] 0.7656893
```



Looks like 0.3 is still the best choice.

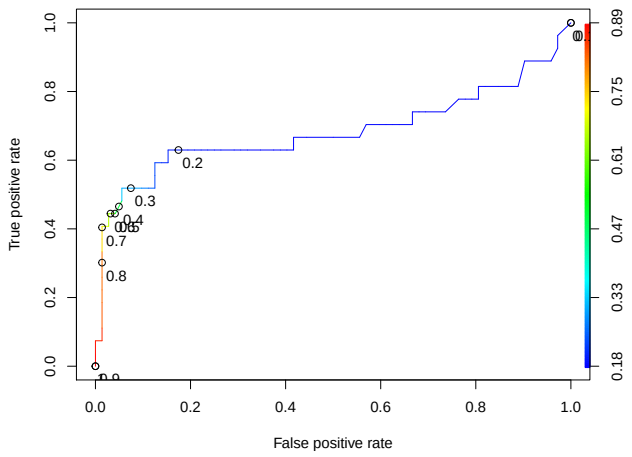
```
lda.pred=predict(lda.fit, care.test)
lda.phat=lda.pred$posterior[,2]
lda.3=factor(ifelse(lda.phat>.3, "1","0"))
confusionMatrix(lda.3, care.test$PoorCare, positive="1" )

Confusion Matrix and Statistics

          Reference
Prediction  0  1                       Accuracy : 0.8438
         0 25  4                       Sensitivity : 0.33333
         1  1  2                       Specificity : 0.96154
                                       Balanced Accuracy : 0.64744
    'Positive' Class : 1
```

Support Vector Machine: Finally let's fit the SVM. I'll use the radial kernel again. First estimate the parameter for this kernel and the cost, using the `tune` funciton. I don't show the output, but it indicated values of `cost`=0.6 and `gamma`=0.2. I fit the classifier with those values and construct the ROC curve.

It's a little tricky to get the ROC curve to plot. See below.

```
svm.fit=svm(PoorCare~OfficeVisits + Narcotics, data=care.train, kernel="radial", gamma=.2, cost=.6, probability=T)
svm.prob <- predict(svm.fit, type="prob", newdata=care.train, probability = TRUE)
svm.prob.rocr <- prediction(attr(svm.prob, "probabilities")[,2], care.train$PoorCare)
svm.perf <- performance(svm.prob.rocr, "tpr","fpr")
plot(svm.perf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1),text.adj=c(-0.2,1.7) )
```

```
> auc.perf.svm=performance(svm.prob.rocr, measure="auc")
> auc.perf.svm@y.values
[[1]]
[1] 0.682356
```




```
> confusionMatrix(svm.4, care.test$PoorCare, positive="1" )
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 25  5
         1  1  1

               Accuracy : 0.8125
             Sensitivity : 0.16667
             Specificity : 0.96154
      Balanced Accuracy : 0.56410
```