Predictive Modeling

Chapter 11: Measuring Performance in Classification Models

STA 6543

The University of Texas at San Antonio

Overview

- Part I: General Strategies
- Part II: Regression Models (response variable is continuous)
  - Chapter 5: Measuring Performance in Regression Models (RMSE, R^2, MAE)
  - Chapter 6: Linear Regression and Its Cousins
  - Chapter 7: Nonlinear Regression Models
  - Chapter 8: Regression Trees and Rule-Based Models
- Part III: Classification Models (response variable is qualitative, such as binary)
  - Chapter 11: Measuring Performance in Classification Models
  - Chapter 12: Discriminant Analysis and Other Linear Classification Models
  - Chapter 13: Nonlinear Classification Models
  - Chapter 14: Classification Trees and Rule-Based Models

An overview of classification

- Classification problems occur when the response variable is qualitative, which takes values in a discrete set *C*. For instance,
  - default ϵ {yes, no};
  - eyecolor ϵ {brown, blue, green}

Cancer { 0 if having cancer; 1 if no cancer

- Given a training data set $D_{training} = \{(x_i , y_i)\}$ for $i = 1,\ldots,n$, we want to build a classifier $C(x)$ that assigns a class label from C to a future unlabeled input vector x.
  - The logistic regression
  - The Bayes classifier
  - The linear discriminant analysis
  - K-nearest neighbors
  - etc

Confusion matrix

- The *confusion matrix* is a common method for describing the performance of a classification model. The confusion matrix for the two-class problem is

FP: The patient has no caner, but you conclude the patient has Cancer

FN: The patient has cancer, but you conclude the patient has no cancer.

Predicted	Observed	
	Event	Nonevent
Event	*TP*	*FP*
Nonevent	*FN*	*TN*

no errors

obs

predicted	Cancer	Noncancer
Cancer	TP	FP
noncancer	FN	TN

- The table cells indicate number of the true positives (*TP*), false positives (*FP*), true negatives (*TN*), and false negatives (*FN*)
- Diagonal cells denote cases where the classes are *correctly* predicted while the off-diagonals illustrate the number of errors for each possible case.

Evaluating predicted classes

- The *Kappa statistic* is

$$\text{Kappa} = \frac{O - E}{1 - E}$$

where *O* is the observed accuracy and *E* is the expected accuracy based on the marginal totals of the confusion matrix.

- The value of Kappa is between −1 and 1:
  - A Kappa of 0 means there is no difference between the observers and chance alone).
  - Kappa values of 0.4 to 0.75 are considered moderate to good and a kappa of >0.75 represents excellent agreement.
  - A kappa of 1.0 (-1.0) means that there is perfect agreement (disagreement) between all raters.

Two-class problems

- The *sensitivity* of the model (i.e., the *true positive rate)* is the rate that the event of interest is predicted correctly for all samples having the event, or

$$Sensitivity = \frac{\#\text{ samples with the event } and \text{ predicted to have the event}}{\#\text{ samples having the event}} = \frac{TP}{TP + FN}$$

- The *specificity (i.e, 1 - the false-positive rate)* is defined as the rate that nonevent samples are predicted as nonevents, or

$$Specificity = \frac{\#\text{ samples without the event } and \text{ predicted as nonevents}}{\#\text{ samples without the event}} = \frac{TN}{TN + FP}$$

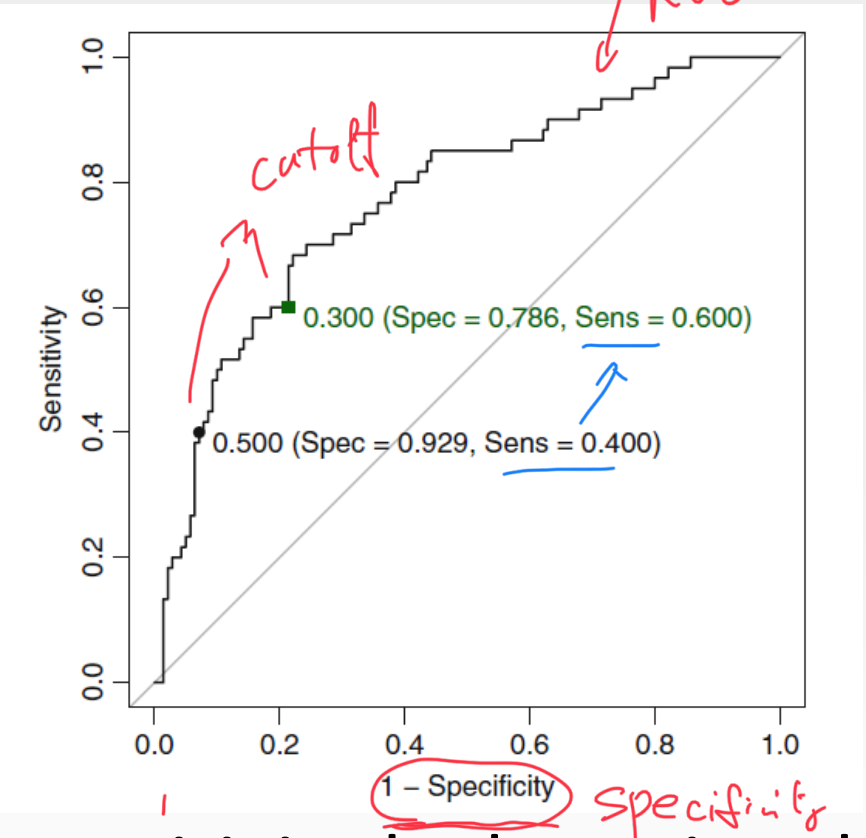
Two-class problems

- There is typically a trade-off to be made between the sensitivity and specificity. (seesaw)
- Increasing the sensitivity of a model is likely to incur a loss of specificity, since more samples are being predicted as events.
- Potential trade-offs between sensitivity and specificity may be appropriate when there are different penalties associated with each type of error.
  - In spam filtering, there is usually a focus on specificity; most people are willing to accept seeing some spam if emails from family members or coworkers are not deleted.

The receiver operating characteristic (ROC)

- The ROC is one technique for evaluating the trade-off between the sensitivity and specificity.
- The ROC curve can be used for determining alternate cutoffs for class probabilities.

The ROC curve for the logistic regression model results for the credit model



- Can we improve the sensitivity by lowering the threshold to capture more true positives (TP)? We can lower value of the threshold

Remarks of the ROC curve

- This plot is a helpful tool for choosing a threshold that appropriately maximizes the trade-off between sensitivity and specificity.
- We see from the plot that decreasing the threshold begins to capture more of the customers with bad credit but also begins to encroach on the bulk of the customers with good credit.
- *The model with the largest area under the ROC curve would be the most effective.*

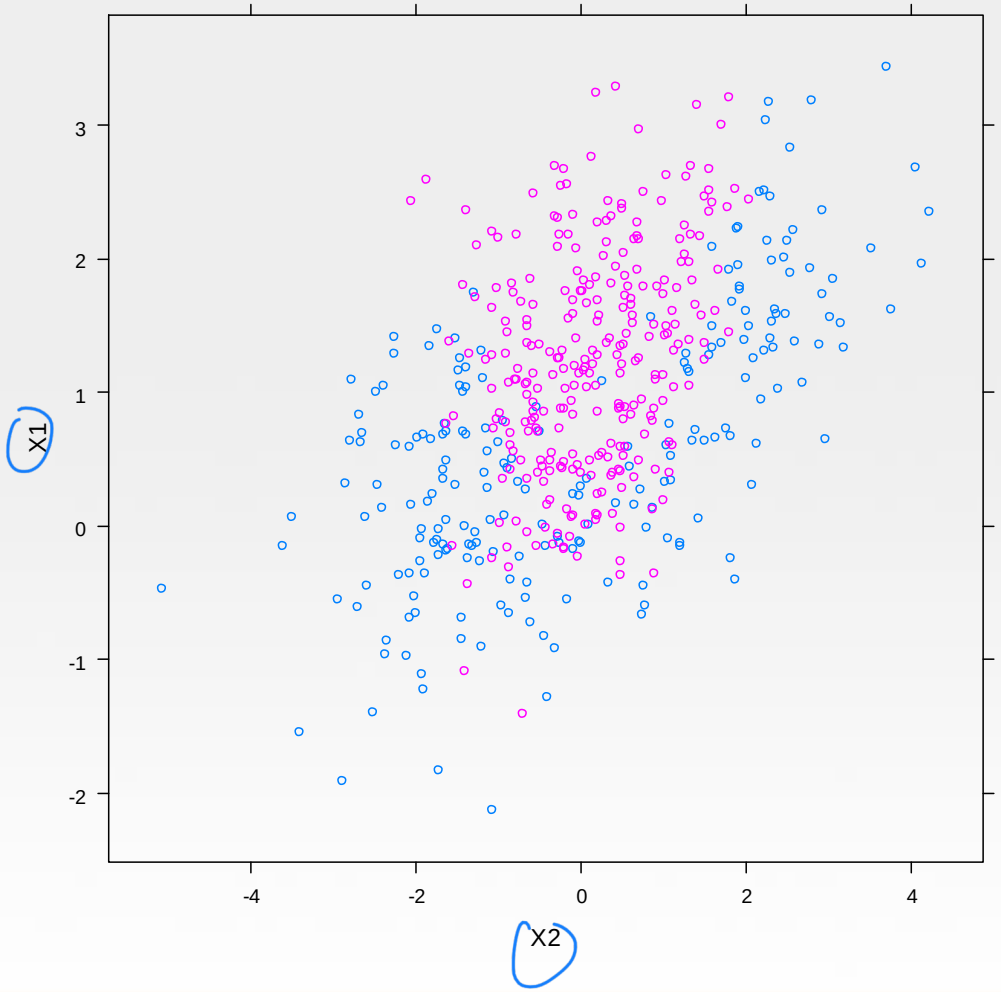
Example

```
library(AppliedPredictiveModeling)
library(randomForest)

### Simulate some two class data with two predictors
set.seed(975)
training <- quadBoundaryFunc(500)
testing <- quadBoundaryFunc(1000)
testing$class2 <- ifelse(testing$class == "Class1", 1, 0)
testing$ID <- 1:nrow(testing)

#Scatter plot
xyplot(X1~ X2, groups=class, data=training)
```

Example



Example

```
### Fit three models: QDA, RM, Logistic

#QDA
library(MASS)
qdaFit <- qda(class ~ X1 + X2, data = training)

#RM
library(randomForest)
rfFit <- randomForest(class ~ X1 + X2, data = training, ntree = 2000)

#Logistic
glmFit <- train(class ~ X1 + X2, data = training, method = "glm",
                trControl = trainControl(method = "repeatedcv", repeats = 5))

### Predict the test set based on three models
testing$qda <- predict(qdaFit, testing)$posterior[,1]
testing$rf <- predict(rfFit, testing, type = "prob")[,1]
testing$glm <- predict(glmFit, testing, type = "prob")[,1]
```

train function will be discussed in the following chapters.

probability

Example

```
### Create the confusion matrix from the test set.

#Confusion Matrix of qda Fit
confusionMatrix(data = predict(qdaFit, testing, type = "prob")$class, reference =
testing$class)

#Confusion matrix of random forest
confusionMatrix(data = predict(rfFit, testing), reference = testing$class)

#Confusion matrix of logistic
confusionMatrix(data = predict(glmFit, testing), reference = testing$class)
```

Confusion Matrix of qdaFit

```
> confusionMatrix(data = predict(qdaFit, testing, type = "prob")$class, reference = testing$class)
Confusion Matrix and Statistics

          Reference
Prediction Class1 Class2
    Class1    360     30
    Class2     99    511

               Accuracy : 0.871
                 95% CI : (0.8486, 0.8912)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7373

 Mcnemar's Test P-Value : 2.137e-09

            Sensitivity : 0.7843
            Specificity : 0.9445
         Pos Pred Value : 0.9231
         Neg Pred Value : 0.8377
             Prevalence : 0.4590
         Detection Rate : 0.3600
   Detection Prevalence : 0.3900
      Balanced Accuracy : 0.8644

       'Positive' Class : Class1
```

TP: 360

FP = 30

FN = 99

TN = 511

Confusion matrix of random forest

```
> confusionMatrix(data = predict(rfFit, testing), reference = testing$class)
Confusion Matrix and Statistics

          Reference
Prediction Class1 Class2
    Class1    378     57
    Class2     81    484

               Accuracy : 0.862
                 95% CI : (0.8391, 0.8828)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.721

 Mcnemar's Test P-Value : 0.05024

            Sensitivity : 0.8235
            Specificity : 0.8946
         Pos Pred Value : 0.8690
         Neg Pred Value : 0.8566
             Prevalence : 0.4590
         Detection Rate : 0.3780
   Detection Prevalence : 0.4350
      Balanced Accuracy : 0.8591

       'Positive' Class : Class1
```

Random Forest is better than QDA. with respect to kappa. Acuuray.

Confusion matrix of logistic

```
> confusionMatrix(data = predict(glmFit, testing), reference = testing$class)
Confusion Matrix and Statistics

          Reference
Prediction Class1 Class2
    Class1    219    100
    Class2    240    441

               Accuracy : 0.66
                 95% CI : (0.6297, 0.6894)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 1.414e-14

                  Kappa : 0.2992

 Mcnemar's Test P-Value : 4.760e-14

            Sensitivity : 0.4771
            Specificity : 0.8152
         Pos Pred Value : 0.6865
         Neg Pred Value : 0.6476
             Prevalence : 0.4590
         Detection Rate : 0.2190
   Detection Prevalence : 0.3190
      Balanced Accuracy : 0.6461

       'Positive' Class : Class1
```

errors are large

ROC curves

```
### ROC curves:
### Like glm(), roc() treats the last level of the factor as the event
### of interest so we use relevel() to change the observed class data

library(pROC)
#ROC for QDA
qdaROC <- roc(testing$class, testing$qda)
plot(qdaROC, col=1, lty=1)

#ROC for Random forest
rfROC <- roc(testing$class,testing$rf)
lines(rfROC, col=2, lty=2)

#ROC for GLM
glmROC <- roc(testing$class, testing$glm)
lines(glmROC, col=3, lty=3)
legend('bottomright', c('QDA','RF','GLM'), col=1:3, lty=1:3,lwd=2)
```

predicted value of class from qda

The calibration plot