Project II – Classification Methods
STAT 897D (Applied Data Mining) – Fall 2015 by XIANG WANG

Introduction

Gene Expression Data and Physical Interaction Network (Brem and Kruglyak, 2005): Time course data reflect the changes in expression levels in response to *Rapamycin*. The original data source forms a 231 × 95 × 6 array: 231 active genes; 95 segregants (individuals); 6 time points for each segregant at time 0, 10, 20, 30, 40, 50 mins (i.e., 95 × 6 = 570 total expression measurements for each gene). Proteins are involved in most cell functions and they typically bind together to form complex structures. Protein-protein interactions (PPI) are highly important for almost all biological processes. Gene expression data measure the mRNA levels that approximate states of a complex biological system. PPI data is widely used to predict interactions among gene expression data. The objective of this project is to identify interacted genes by using the provided gene expression data.

Analysis and Results

Data preparation

First, load the data file "Project2.RData" and we can see there are four files: Network.train is a 122 × 122 matrix showing the PPI network (0 if not interacted, 1 if interacted); Network.valid is a 53 × 53 matrix showing the PPI network for the validation data; DATA.train is a 122 × 570 gene expression matrix; DATA.valid is a 53 × 570 gene expression matrix for the validation data. Then, put the PPI networks into vectors with each element being a gene pair. Next, we'll use 5 predictors: mean of gene i (across the 570 measurements for that gene), variance of gene i; mean of gene j, variance of gene j; covariance between gene i and gene j, to predict the interaction between gene i and gene j.

Model building process

Fit the following five models (logistic regression, linear discriminant analysis, quadratic discriminant analysis, K-nearest neighbors, logistic regression generalized additive model) to the training data using the data frame, data.train, that contains Y as the response variable and {X1, X2, X3, X4, X5} as the predictors.

- For each model, obtain the posterior probabilities of interaction for the 7381 gene pairs in the training data, and classify them into class 1 (interacted) if they are in the top 200 most likely interacted pairs and into class 0 (not interacted) otherwise.
- Use these classification results and Y.train to calculate the training classification error rate, sensitivity, and specificity for each model.
- Use each of the five models fit to the training data to obtain posterior probabilities of interaction for the 1378 gene pairs in the *validation* data. Use the same "cut-off probability" values to classify the gene pairs into class 1 if their posterior probability is greater than this cut-off probability and into class 0 otherwise.
- Also, use these classification results and Y.valid to calculate the validation classification error rate, sensitivity, and specificity for each model.

Model 1: Logistic regression (LR) using glm function

R glm function was used to fit the logistic regression to the training data that contains Y as the response variable and {X1, X2, X3, X4, X5} as the predictors. The model's coefficients and their estimates are: Intercept (-14.12), X1(0.49), X2(0.40), X3(1.44), X4(0.002) and X5(-0.08). The training classification error rate, sensitivity, and specificity are 3.3%, 39.6% and 98.3%, respectively. For the validation data, the same cut-off probability values were used and the validation classification error rate, sensitivity, and specificity are 3.8%, 55.8% and 97.5%, respectively.

Model 2: Linear discriminant analysis (LDA) using qda function in MASS

R lda function was used to fit LDA model to the training data that contains Y as the response variable and {X1, X2, X3, X4, X5} as the predictors. The model's coefficients and their estimates are: X1(0.35), X2(0.29), X3(1.03), X4(0.08) and X5(0.04). The training classification error rate, sensitivity, and specificity are 3.6%, 34.0% and 98.1%, respectively. For the validation data, the same cut-off probability values were used and the validation classification error rate, sensitivity, and specificity are 3.3%, 51.2% and 98.2%, respectively.

Model 3: Quadratic discriminant analysis (QDA) using qda function in MASS

R qda function in the MASS package was used to fit QDA model to the training data that contains Y as the response variable and {X1, X2, X3, X4, X5} as the predictors. The training classification error rate, sensitivity, and specificity are 3.5%, 36.0% and 98.2%, respectively. For the validation data, the same cut-off probability values were used and the validation classification error rate, sensitivity, and specificity are 3.4%, 55.8% and 97.9%, respectively.

Model 4: K-nearest neighbors (KNN) with k chosen using leave-one-out cross validation

The mean and standard deviation of predictor variables {X1, X2, X3, X4, X5} are not zero and one. Thus, I standardized them using scale function in R. Then, R knn function was used to fit KNN model to the training data that contains Y as the response variable and standardized {X1, X2, X3, X4, X5} as the predictors. I used a random number seed of 2014 immediately before running the cross validation loop and also immediately before fitting the model with the selected value of k. The training classification error rate, sensitivity, and specificity are 2.7%, 47.7% and 98.7%, respectively. For the validation data, the same cut-off probability values were used and the validation classification error rate, sensitivity, and specificity are 5.2%, 30.2% and 96.7%, respectively.

Model 5: Logistic regression generalized additive model (GAM)

R gam function was used to fit GAM model in which a smoothing spline with 5 degrees of freedom for each predictor was applied to the training data that contains Y as the response variable and {X1, X2, X3, X4, X5} as the predictors. The training classification error rate, sensitivity, and specificity are 3.0%, 44.2% and 98.4%, respectively. For the validation data, the same cut-off probability values were used and the validation classification error rate, sensitivity, and specificity are 7.3%, 55.8% and 93.9%, respectively.

Summary and Conclusion

The key results (%) of the above five models are summarized in the following table.

Model	Classification error rate		Sensitivity		Specificity	
	Train	Valid	Train	Valid	Train	Valid
LR	3.3	3.8	39.6	55.8	98.3	97.5
LDA	3.6	3.3	34.0	51.2	98.1	98.2
QDA	3.5	3.4	36.0	55.8	98.2	97.9
KNN	2.7	5.2	47.7	30.2	98.7	96.9
GAM	3.0	7.3	44.2	55.8	98.4	93.9

For training data, the classification error rate ranges from 2.7% to 3.6%; sensitivity ranges from 34.0% to 47.7%; specificity all are above 98%. To choose better models, we need the models that have less classification error rate, higher sensitivity and higher specificity for the validation data. Here, I would choose the LDA model to identify the interacted genes since it gives us the smallest classification error rate (3.3%) and highest specificity (98.2%) although its sensitivity is not the highest but still not far away from the highest sensitivity. LR and QDA are also good models to make the prediction of interacted genes.

Appendix: R Codes and Results

```
load("Project2.RData")
ls()
```

```
## [1] "DATA.train"    "DATA.valid"    "Network.train" "Network.valid"
```

- Network.train: 122 × 122 matrix showing the PPI network (0 if not interacted, 1 if interacted);
- Network.valid: 53 × 53 matrix showing the PPI network for the validation data;
- DATA.train: 122 × 570 gene expression matrix;
- DATA.valid: 53 × 570 gene expression matrix for the validation data. Next, put the PPI networks into vectors with each element being a gene pair:

```
Y.train <- Network.train[lower.tri(Network.train)]
n.train <- length(Y.train) # 7381 pairs in training # data
Y.train.mean <- mean(Y.train) # 0.02669, proportion of PPI interactions in
training data
Y.valid <- Network.valid[lower.tri(Network.valid)]
n.valid <- length(Y.valid) # 1378 pairs in validation # data
mean(Y.valid) # 0.0312 proportion of PPI interactions in validation data
```

```
## [1] 0.03120464
```

Next, to predict the interaction between gene i and gene j, we'll use 5 predictors: mean of gene i (across the 570 measurements for that gene), variance of gene i; mean of gene j, variance of gene j; covariance between gene i and gene j:

```
X.train = NULL # 7381 by 5 matrix with 5 predictors for each gene pair
for (i in 1:(dim(DATA.train)[1]-1))
  for (j in (i+1):dim(DATA.train)[1])
    X.train = rbind(X.train,
                    c(mean(DATA.train[i,]), mean(DATA.train[j,]),
                      cov(DATA.train[i,],DATA.train[j,]),
                      var(DATA.train[i,]), var(DATA.train[j,])))
data.train <- as.data.frame(cbind(Y.train, X.train))
names(data.train) <- c("Y", "X1", "X2", "X3", "X4", "X5")

X.valid = NULL # 1378 by 5 matrix with 5 predictors for each gene pair
for (i in 1:(dim(DATA.valid)[1]-1))
  for (j in (i+1):dim(DATA.valid)[1])
    X.valid = rbind(X.valid,
                    c(mean(DATA.valid[i,]), mean(DATA.valid[j,]),
                      cov(DATA.valid[i,],DATA.valid[j,]), var(DATA.valid[i,
]),
                      var(DATA.valid[j,])))
data.valid <- as.data.frame(X.valid)
names(data.valid) <- c("X1", "X2", "X3", "X4", "X5")
```

Fit the following models to the training data using the data frame, data.train, that contains Y as the response variable and {X1, X2, X3, X4, X5} as the predictors. For each model obtain the posterior probabilities of interaction for the 7381 gene pairs in the training data, and classify them into class 1 (interacted) if they are in the top 200 most likely interacted pairs and into class 0 (not interacted) otherwise (see hint 2 below). Use these classification results and Y.train to calculate the training classification error rate, sensitivity, and specificity for each model.

Model a: Logistic regression analysis using glm function

```
model.logistic <- glm(Y ~ X1+X2+X3+X4+X5, data.train, family=binomial("logi
t"))
summary(model.logistic)

##
## Call:
## glm(formula = Y ~ X1 + X2 + X3 + X4 + X5, family = binomial("logit"),
##     data = data.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2821  -0.2354  -0.1453  -0.0863   3.7638
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.123839   0.903543 -15.632  < 2e-16 ***
## X1            0.494346   0.051149   9.665  < 2e-16 ***
## X2            0.403708   0.048423   8.337  < 2e-16 ***
## X3            1.435135   0.197449   7.268 3.64e-13 ***
## X4            0.001906   0.094916   0.020    0.984
## X5           -0.083625   0.096594  -0.866    0.387
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1816.3  on 7380  degrees of freedom
## Residual deviance: 1507.3  on 7375  degrees of freedom
## AIC: 1519.3
##
## Number of Fisher Scoring iterations: 7

# Coefficients:
#             Estimate Std. Error z value Pr(>|z|)
#(Intercept) -14.123839   0.903543 -15.632  < 2e-16 ***
#X1            0.494346   0.051149   9.665  < 2e-16 ***
#X2            0.403708   0.048423   8.337  < 2e-16 ***
#X3            1.435135   0.197449   7.268 3.64e-13 ***
#X4            0.001906   0.094916   0.020    0.984
#X5           -0.083625   0.096594  -0.866    0.387
```

```
#---
#Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#(Dispersion parameter for binomial family taken to be 1)
#
#    Null deviance: 1816.3  on 7380  degrees of freedom
#Residual deviance: 1507.3  on 7375  degrees of freedom
#AIC: 1519.3
#Number of Fisher Scoring iterations: 7

post.train.logistic <- model.logistic$fitted.values # n.train posterior pro
babilities of Y=1
cutoff.logistic <- sort(post.train.logistic, decreasing=T)[201] # probabili
ty cutoff for predicting classes
Ghat.train.logistic <- ifelse(post.train.logistic > cutoff.logistic,1,0) #
classification rule
table(Ghat.train.logistic,Y.train) # classification table

##                    Y.train
## Ghat.train.logistic    0    1
##                   0 7062  119
##                   1  122   78

#                    Y.train
#Ghat.train.logistic    0    1
#                   0 7062  119
#                   1  122   78
sum(abs(Ghat.train.logistic-Y.train))/n.train # training data classificatio
n error rate = (122+119)/7381 = 0.033

## [1] 0.0326514

sum(Ghat.train.logistic==1&Y.train==1)/sum(Y.train==1) # sensitivity = 78/(
119+78) = 0.396

## [1] 0.3959391

sum(Ghat.train.logistic==0&Y.train==0)/sum(Y.train==0) # specificity = 7062
/(7062+122) = 0.983

## [1] 0.9830178

post.valid.logistic <- predict(model.logistic, data.valid, type="response")
 # n.valid post probs
Ghat.valid.logistic <- ifelse(post.valid.logistic > cutoff.logistic,1,0) #
use same probability cutoff
table(Ghat.valid.logistic,Y.valid) # classification table

##                    Y.valid
## Ghat.valid.logistic    0    1
##                   0 1301   19
##                   1   34   24
```

```
#                   Y.valid
#Ghat.valid.logistic    0    1
#                   0 1301   18
#                   1   34   24
sum(abs(Ghat.valid.logistic-Y.valid))/n.valid # classification error rate =
 (34+18)/1378 = 0.038
```

```
## [1] 0.03846154
```

```
sum(Ghat.valid.logistic==1&Y.valid==1)/sum(Y.valid==1) # sensitivity = 24/(
24+18) = 0.558
```

```
## [1] 0.5581395
```

```
sum(Ghat.valid.logistic==0&Y.valid==0)/sum(Y.valid==0) # specificity = 1301
/(34+1301) = 0.869
```

```
## [1] 0.9745318
```

Model b: Linear discriminant analysis using lda function in MASS

```
library(MASS)
model.lda <- lda(Y ~ X1+X2+X3+X4+X5, data.train); model.lda
```

```
## Call:
## lda(Y ~ X1 + X2 + X3 + X4 + X5, data = data.train)
##
## Prior probabilities of groups:
##          0          1
## 0.97330985 0.02669015
##
## Group means:
##         X1       X2         X3        X4        X5
## 0 10.67300 10.80075 0.01331963 0.8531054 0.9114245
## 1 12.26554 12.22904 0.27908714 0.7315978 0.6442998
##
## Coefficients of linear discriminants:
##           LD1
## X1 0.35181460
## X2 0.29353120
## X3 1.03316558
## X4 0.08062700
## X5 0.04478497
```

```
#Prior probabilities of groups:
#         0          1
#0.97330985 0.02669015
#
#Group means:
#        X1       X2         X3        X4        X5
#0 10.67300 10.80075 0.01331963 0.8531054 0.9114245
#1 12.26554 12.22904 0.27908714 0.7315978 0.6442998
```

```
#
#Coefficients of linear discriminants:
#          LD1
#X1 0.35181460
#X2 0.29353120
#X3 1.03316558
#X4 0.08062700
#X5 0.04478497

post.train.lda <- predict(model.lda)$posterior[,2] # n.train posterior prob
abilities of Y=1
cutoff.lda <- sort(post.train.lda, decreasing=T)[201] # probability cutoff
for predicting classes
Ghat.train.lda <- ifelse(post.train.lda > cutoff.lda,1,0) # classification
rule
table(Ghat.train.lda,Y.train) # classification table

##                Y.train
## Ghat.train.lda    0    1
##              0 7051  130
##              1  133   67

#              Y.train
#Ghat.train.lda    0    1
#             0 7051  130
#             1  133   67
sum(abs(Ghat.train.lda-Y.train))/n.train # training data classification err
or rate = (133+130)/7381 = 0.036

## [1] 0.03563203

sum(Ghat.train.lda==1&Y.train==1)/sum(Y.train==1) # sensitivity = 67/(130+6
7) = 0.340

## [1] 0.3401015

sum(Ghat.train.lda==0&Y.train==0)/sum(Y.train==0) # specificity = 7051/(705
1+133) = 0.981

## [1] 0.9814866

post.valid.lda <- predict(model.lda, data.valid)$posterior[,2] # n.valid po
sterior probabilities of Y=1
Ghat.valid.lda <- ifelse(post.valid.lda > cutoff.lda,1,0) # use same probab
ility cutoff
table(Ghat.valid.lda,Y.valid) # classification table

##                Y.valid
## Ghat.valid.lda    0    1
##              0 1311   21
##              1   24   22
```

```
#              Y.valid
#Ghat.valid.lda    0    1
#             0 1311   21
#             1   24   22
sum(abs(Ghat.valid.lda-Y.valid))/n.valid # classification error rate = (24+21)/1378 = 0.033
```

```
## [1] 0.03265602
```

```
sum(Ghat.valid.lda==1&Y.valid==1)/sum(Y.valid==1) # sensitivity = 22/(22+21) = 0.512
```

```
## [1] 0.5116279
```

```
sum(Ghat.valid.lda==0&Y.valid==0)/sum(Y.valid==0) # specificity = 1311/(1311+24) = 0.982
```

```
## [1] 0.9820225
```

Model c: Quadratic discriminant analysis using qda function in MASS

```
model.qda <- qda(Y ~ X1+X2+X3+X4+X5, data.train); model.qda
```

```
## Call:
## qda(Y ~ X1 + X2 + X3 + X4 + X5, data = data.train)
##
## Prior probabilities of groups:
##          0          1
## 0.97330985 0.02669015
##
## Group means:
##         X1       X2         X3        X4        X5
## 0 10.67300 10.80075 0.01331963 0.8531054 0.9114245
## 1 12.26554 12.22904 0.27908714 0.7315978 0.6442998
```

```
#Prior probabilities of groups:
#         0          1
#0.97330985 0.02669015
#
#Group means:
#        X1       X2         X3        X4        X5
#0 10.67300 10.80075 0.01331963 0.8531054 0.9114245
#1 12.26554 12.22904 0.27908714 0.7315978 0.6442998
```

```
post.train.qda <- predict(model.qda)$posterior[,2] # n.train posterior probabilities of Y=1
cutoff.qda <- sort(post.train.qda, decreasing=T)[201] # probability cutoff for predicting classes
Ghat.train.qda <- ifelse(post.train.qda > cutoff.qda,1,0) # classification rule
table(Ghat.train.qda,Y.train) # classification table
```

```
##                Y.train
## Ghat.train.qda    0    1
##              0 7055  126
##              1  129   71
```

```
#              Y.train
#Ghat.train.qda    0    1
#             0 7055  126
#             1  129   71
sum(abs(Ghat.train.qda-Y.train))/n.train # training data classification err
or rate = (129+126)/7381 = 0.035
```

```
## [1] 0.03454816
```

```
sum(Ghat.train.qda==1&Y.train==1)/sum(Y.train==1) # sensitivity = 71/(126+7
1) = 0.360
```

```
## [1] 0.3604061
```

```
sum(Ghat.train.qda==0&Y.train==0)/sum(Y.train==0) # specificity = 7055/(705
5+129) = 0.982
```

```
## [1] 0.9820434
```

```
post.valid.qda <- predict(model.qda, data.valid)$posterior[,2] # n.valid po
sterior probabilities of Y=1
Ghat.valid.qda <- ifelse(post.valid.qda > cutoff.qda,1,0) # use same probab
ility cutoff
table(Ghat.valid.qda,Y.valid) # classification table
```

```
##                Y.valid
## Ghat.valid.qda    0    1
##              0 1307   19
##              1   28   24
```

```
#              Y.valid
#Ghat.valid.qda    0    1
#             0 1307   19
#             1   28   24
sum(abs(Ghat.valid.qda-Y.valid))/n.valid # classification error rate = (28+
19)/1378 = 0.034
```

```
## [1] 0.0341074
```

```
sum(Ghat.valid.qda==1&Y.valid==1)/sum(Y.valid==1) # sensitivity = 24/(24+19
) = 0.558
```

```
## [1] 0.5581395
```

```
sum(Ghat.valid.qda==0&Y.valid==0)/sum(Y.valid==0) # specificity = 1307/(130
7+28) = 0.979
```

```
## [1] 0.9790262
```

Model d: K Nearest Neighbors using knn function

```
# By running var and mean functions, we can see the predictors are not stan
darized to a mean of zero and standard deviation of one. Thus, standardizat
ion step does need.
X <- rbind(X.train,X.valid)
X.std <- scale(X)
X.train.std <- X.std[1:n.train,]
X.valid.std <- X.std[(n.train+1):(n.train+n.valid),]

library(class)
mer <- rep(NA, 30) # misclassification error rates based on leave-one-out c
ross-validation
set.seed(2014) # seed must be set because R randomly breaks ties
for (i in 1:30) mer[i] <- sum((Y.train-(c(knn.cv(train=X.train.std, cl=Y.tr
ain, k=i))-1))^2)/n.train
which.min(mer) # minimum occurs at k=13
```

```
## [1] 13
```

```
set.seed(2014)
model.knn <- knn(train=X.train.std, test=X.train.std, cl=Y.train, k=13, pro
b=T)
predclass.knn <- c(model.knn)-1 # convert factor to numeric classes
predprob.knn <- attr(model.knn, "prob") # proportion of votes for winning c
lass
post.train.knn <- predclass.knn*predprob.knn+(1-predclass.knn)*(1-predprob.
knn) # n.train post probs of Y=1
cutoff.knn <- sort(post.train.knn, decreasing=T)[201] # probability cutoff
for predicting classes
Ghat.train.knn <- ifelse(post.train.knn > cutoff.knn,1,0) # classification
rule
table(Ghat.train.knn,Y.train) # classification table
```

```
##                Y.train
## Ghat.train.knn    0    1
##              0 7091  103
##              1   93   94
```

```
#              Y.train
#Ghat.train.knn    0    1
#             0 7091  103
#             1   93   94
sum(abs(Ghat.train.knn-Y.train))/n.train # training data classification err
or rate = (93+103)/7381 = 0.027
```

```
## [1] 0.02655467
```

```
sum(Ghat.train.knn==1&Y.train==1)/sum(Y.train==1) # sensitivity = 94/(94+10
3) = 0.477
```

```
## [1] 0.4771574
```

```
sum(Ghat.train.knn==0&Y.train==0)/sum(Y.train==0) # specificity = 7091/(709
1+93)
```

```
## [1] 0.9870546
```

```
# 0.987

set.seed(2014)
model.knn <- knn(train=X.train.std, test=X.valid.std, cl=Y.train, k=13, pro
b=T)
predclass.knn <- c(model.knn)-1 # convert factor to numeric classes
predprob.knn <- attr(model.knn, "prob") # proportion of votes for winning c
lass
post.valid.knn <- predclass.knn*predprob.knn+(1-predclass.knn)*(1-predprob.
knn) # n.valid post probs of Y=1
Ghat.valid.knn <- ifelse(post.valid.knn > cutoff.knn,1,0) # use same probab
ility cutoff
table(Ghat.valid.knn,Y.valid) # classification table
```

```
##               Y.valid
## Ghat.valid.knn    0    1
##              0 1294   30
##              1   41   13
```

```
#              Y.valid
#Ghat.valid.knn    0    1
#             0 1294   30
#             1   41   13
sum(abs(Ghat.valid.knn-Y.valid))/n.valid # classification error rate = (21+
64)/1378 = 0.052
```

```
## [1] 0.05152395
```

```
sum(Ghat.valid.knn==1&Y.valid==1)/sum(Y.valid==1) # sensitivity = 13/(13+30
) = 0.302
```

```
## [1] 0.3023256
```

```
sum(Ghat.valid.knn==0&Y.valid==0)/sum(Y.valid==0) # specificity = 1294/(129
4+41) = 0.969
```

```
## [1] 0.9692884
```

Model e: Logistic regression generalized additive model (GAM)

```
library(gam)
```

```
## Loading required package: splines
## Loading required package: foreach
## Loaded gam 1.12
```

```
model.gam <- gam(Y ~ s(X1,df=5)+ s(X2,df=5)+s(X3,df=5)+s(X4,df=5)+s(X5,df=5
), data.train, family=binomial)
summary(model.gam)
```

```
##
## Call: gam(formula = Y ~ s(X1, df = 5) + s(X2, df = 5) + s(X3, df = 5) +
##     s(X4, df = 5) + s(X5, df = 5), family = binomial, data = data.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98164 -0.21565 -0.13202 -0.09386  3.73839
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 1816.338 on 7380 degrees of freedom
## Residual Deviance: 1408.102 on 7355 degrees of freedom
## AIC: 1460.102
##
## Number of Local Scoring Iterations: 12
##
## Anova for Parametric Effects
##                  Df  Sum Sq Mean Sq F value    Pr(>F)
## s(X1, df = 5)     1    92.2  92.211 55.7125 9.352e-14 ***
## s(X2, df = 5)     1    60.5  60.484 36.5432 1.566e-09 ***
## s(X3, df = 5)     1    73.7  73.737 44.5508 2.659e-11 ***
## s(X4, df = 5)     1     0.0   0.014  0.0085    0.9264
## s(X5, df = 5)     1     3.3   3.260  1.9695    0.1605
## Residuals      7355 12173.5   1.655
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##               Npar Df Npar Chisq    P(Chi)
## (Intercept)
## s(X1, df = 5)       4     18.628 0.0009296 ***
## s(X2, df = 5)       4     25.319 4.339e-05 ***
## s(X3, df = 5)       4     36.109 2.749e-07 ***
## s(X4, df = 5)       4     11.332 0.0230774 *
## s(X5, df = 5)       4     10.367 0.0346830 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#Call: gam(formula = Y ~ s(X1, df = 5) + s(X2, df = 5) + s(X3, df = 5) +
#    s(X4, df = 5) + s(X5, df = 5), family = binomial, data = data.train)
#Deviance Residuals:
#     Min       1Q   Median       3Q      Max
#-0.98164 -0.21565 -0.13202 -0.09386  3.73839
#
#(Dispersion Parameter for binomial family taken to be 1)
#
#    Null Deviance: 1816.338 on 7380 degrees of freedom
#Residual Deviance: 1408.102 on 7355 degrees of freedom
#AIC: 1460.102
#
#Number of Local Scoring Iterations: 12
```

```
#Anova for Parametric Effects
#                Df  Sum Sq Mean Sq F value    Pr(>F)
#s(X1, df = 5)    1    92.2  92.211 55.7125 9.352e-14 ***
#s(X2, df = 5)    1    60.5  60.484 36.5432 1.566e-09 ***
#s(X3, df = 5)    1    73.7  73.737 44.5508 2.659e-11 ***
#s(X4, df = 5)    1     0.0   0.014  0.0085    0.9264
#s(X5, df = 5)    1     3.3   3.260  1.9695    0.1605
#Residuals     7355 12173.5   1.655
#---
#Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
#Anova for Nonparametric Effects
#              Npar Df Npar Chisq    P(Chi)
#(Intercept)
#s(X1, df = 5)       4     18.628 0.0009296 ***
#s(X2, df = 5)       4     25.319 4.339e-05 ***
#s(X3, df = 5)       4     36.109 2.749e-07 ***
#s(X4, df = 5)       4     11.332 0.0230774 *
#s(X5, df = 5)       4     10.367 0.0346830 *
#---
#Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

post.train.gam <- model.gam$fitted.values # n.train posterior probabilities
 of Y=1
cutoff.gam <- sort(post.train.gam, decreasing=T)[201] # probability cutoff
for predicting classes
Ghat.train.gam <- ifelse(post.train.gam > cutoff.gam,1,0) # classification
rule
table(Ghat.train.gam,Y.train) # classification table

##                Y.train
## Ghat.train.gam    0    1
##              0 7071  110
##              1  113   87

#              Y.train
#Ghat.train.gam    0    1
#             0 7071  110
#             1  113   87
sum(abs(Ghat.train.gam-Y.train))/n.train # training data classification err
or rate = (110+113)/7381 = 0.030

## [1] 0.03021271

sum(Ghat.train.gam==1&Y.train==1)/sum(Y.train==1) # sensitivity = 87/(110+8
7) = 0.442

## [1] 0.4416244

sum(Ghat.train.gam==0&Y.train==0)/sum(Y.train==0) # specificity = 7071/(707
1+113) = 0.984
```

```
## [1] 0.9842706
```

```
post.valid.gam <- predict(model.gam, data.valid, type="response") # n.valid
 post probs
Ghat.valid.gam <- ifelse(post.valid.gam > cutoff.gam,1,0) # use same probab
ility cutoff
table(Ghat.valid.gam,Y.valid) # classification table
```

```
##                Y.valid
## Ghat.valid.gam    0    1
##              0 1254   19
##              1   81   24
```

```
#               Y.valid
#Ghat.valid.gam    0    1
#             0 1254   19
#             1   81   24
sum(abs(Ghat.valid.gam-Y.valid))/n.valid # classification error rate = (81+
19)/1378 = 0.073
```

```
## [1] 0.07256894
```

```
sum(Ghat.valid.gam==1&Y.valid==1)/sum(Y.valid==1) # sensitivity = 24/(24+19
) = 0.558
```

```
## [1] 0.5581395
```

```
sum(Ghat.valid.gam==0&Y.valid==0)/sum(Y.valid==0) # specificity = 1254/(125
4+81) = 0.939
```

```
## [1] 0.9393258
```