|  |
| --- |
| SR\_CHLD |
| SR\_AVHV |
| SR\_INCM |
| SR\_INCA |
| SR\_PLOW |
| SR\_NPRO |
| SR\_TGIF |
| SR\_LGIF |
| SR\_RGIF |
| SR\_TDON |
| SR\_TLAG |
| SR\_AGIF |

# Logistic Regression using GAM with Local Regression Smoothing

library(gam)

# M1c: Trimmed subset of 12 variables

model.tr1 = gam(donr ~ reg1 + reg2 + home + sr\_chld + hinc + hinc2 + wrat2 + wrat3 +

lo(ln\_incm,span=0.5) + lo(ln\_tgif,span=0.5) + lo(sr\_tdon,span=0.5) + lo(sr\_tlag,span=0.5),

data.train.std.c, family=binomial("logit"))

# M1b: Best 20 variable model from regsubsets

model.tr1 = gam(donr ~ reg1 + reg2 + home + chld + hinc + lo(inca,span=0.5) + lo(tgif,span=0.5) +

lo(tlag,span=0.5) + wrat2 + wrat3 + hinc2 + lo(ln\_incm,span=0.5) + lo(ln\_tgif,span=0.5) +

lo(ln\_lgif,span=0.5) + lo(ln\_tdon,span=0.5) + lo(ln\_tlag,span=0.5) + lo(ln\_agif,span=0.5) +

sr\_chld + lo(sr\_tdon,span=0.5) + lo(sr\_tlag,span=0.5),

data.train.std.c, family=binomial("logit"))

# M1a: Default Original subset

model.tr1 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +

lo(ln\_avhv,span=0.5) + lo(incm,span=0.5) + lo(inca,span=0.5) + lo(plow,span=0.5) +

lo(npro,span=0.5) + lo(tgif,span=0.5) + lo(lgif,span=0.5) + lo(rgif,span=0.5) +

lo(tdon,span=0.5) + lo(tlag,span=0.5) + lo(agif,span=0.5),

data.train.std.c, family=binomial("logit"))

post.valid.tr1 <- predict(model.tr1, data.valid.std.c, type="response") # n.valid post probs

profit.tr1 <- cumsum(14.5\*c.valid[order(post.valid.tr1, decreasing=T)]-2)

plot(profit.tr1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.tr1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.tr1)) # report number of mailings and maximum profit

cutoff.tr1 <- sort(post.valid.tr1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.tr1 <- ifelse(post.valid.tr1>cutoff.tr1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.tr1, c.valid) # classification table

# Least Squares Regression using GAM with Local Regression Smoothing (span=0.5)

library(gam)

# M7c: Trimmed subset of 12 variables

model.lo1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + lo(plow,span=0.5) + wrat + wrat2 +

hinc3 + lo(ln\_incm,span=0.5) + lo(ln\_lgif,span=0.5) + lo(ln\_tgif,span=0.5) +

lo(ln\_rgif,span=0.5) + lo(ln\_agif,span=0.5),

data.train.std.y, family=gaussian("identity"))

# M5b: Best 20 variable model from regsubsets

model.lo1 = gam(damt ~ reg3 + reg4 + reg5 + home + chld + hinc + lo(plow,span=0.5) + wrat + wrat2 +

wrat3 + hinc3 + lo(ln\_incm,span=0.5) + lo(ln\_plow,span=0.5) + lo(ln\_tgif,span=0.5) +

lo(ln\_lgif,span=0.5) + lo(ln\_rgif,span=0.5) + lo(ln\_agif,span=0.5) + lo(sr\_incm,span=0.5) +

lo(sr\_plow,span=0.5) + lo(sr\_lgif,span=0.5),

data.train.std.y, family=gaussian("identity"))

# M5a: Default Original subset

model.lo1 <- gam(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + genf + wrat +

lo(ln\_avhv,span=0.5) + lo(incm,span=0.5) + lo(inca,span=0.5) +

lo(plow,span=0.5) + lo(npro,span=0.5) + lo(tgif,span=0.5) +

lo(lgif,span=0.5) + lo(rgif,span=0.5) + lo(tdon,span=0.5) +

lo(tlag,span=0.5) + lo(agif,span=0.5),

data.train.std.y, family=gaussian("identity"))

pred.valid.lo1 <- predict(model.lo1, newdata = data.valid.std.y) # validation predictions

mean((y.valid - pred.valid.lo1)^2) # mean prediction error

sd((y.valid - pred.valid.lo1)^2)/sqrt(n.valid.y) # std error

AIC(model.lo1)

AIC(model.lo1,k=8.290042) # for BIC, k=ln(#obs in training set)

post.valid.bag1 <- predict(model.bag1, data.valid.std.c, type="response") # n.valid post probs

profit.bag1 <- cumsum(14.5\*c.valid[order(post.valid.bag1, decreasing=T)]-2)

plot(profit.bag1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.bag1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.bag1)) # report number of mailings and maximum profit

cutoff.bag1 <- sort(post.valid.bag1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.bag1 <- ifelse(post.valid.bag1>cutoff.bag1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.bag1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.bag1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

post.valid.rf1 <- predict(model.rf1, data.valid.std.c) # n.valid post probs

profit.rf1 <- cumsum(14.5\*c.valid[order(post.valid.rf1, decreasing=T)]-2)

plot(profit.rf1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.rf1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.rf1)) # report number of mailings and maximum profit

cutoff.rf1 <- sort(post.valid.rf1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.rf1 <- ifelse(post.valid.rf1>cutoff.rf1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.rf1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.rf1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)

post.valid.boo1 <- predict(model.boo1, data.valid.std.c, n.trees=5000) # n.valid post probs

profit.boo1 <- cumsum(14.5\*c.valid[order(post.valid.boo1, decreasing=T)]-2)

plot(profit.boo1) # see how profits change as more mailings are made

n.mail.valid <- which.max(profit.boo1) # number of mailings that maximizes profits

c(n.mail.valid, max(profit.boo1)) # report number of mailings and maximum profit

cutoff.boo1 <- sort(post.valid.boo1, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid

chat.valid.boo1 <- ifelse(post.valid.boo1>cutoff.boo1, 1, 0) # mail to everyone above the cutoff

table(chat.valid.boo1, c.valid) # classification table

## computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

library(ROCR)

pred <- prediction(chat.valid.boo1, c.valid)

perf <- performance(pred,"tpr","fpr")

auc.perf = performance(pred, measure = "auc")

auc.perf@y.values

plot(perf)