HW14 M072040019 梅瀚中

set.seed(1)  
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

library(ISLR)  
library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(leaps)  
library(ggplot2)  
library(gam)

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.16

library(randomForest)

## randomForest 4.6-14

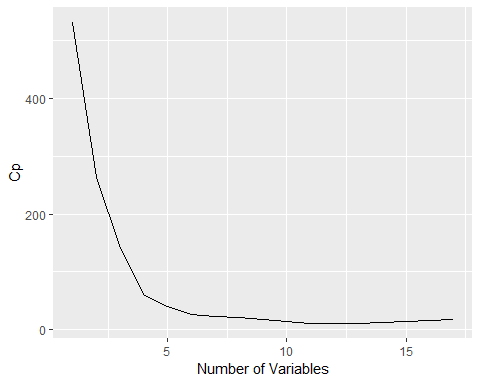
## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

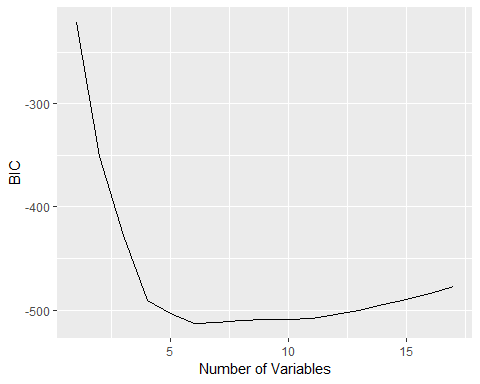
dt <- data.table(College)  
n <- length(dt$Outstate)  
train <- sample(n, n/2)  
dt.train <- College[train, ] %>% data.table()  
dt.test <- College[-train, ] %>% data.table()  
reg.fit <- regsubsets(Outstate ~ ., data = dt.train, nvmax = 17, method = "forward")  
reg.summary <- summary(reg.fit)  
  
ggplot(data.frame(cp =reg.summary$cp, nrVar=1:17), aes(x=nrVar, y=cp))+xlab("Number of Variables") + ylab("Cp") + geom\_line()



which.min(reg.summary$cp)

## [1] 13

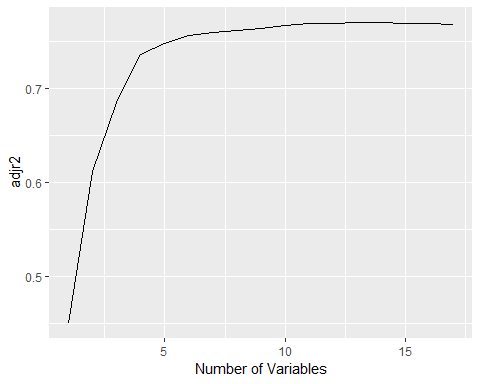
ggplot(data.frame(bic =reg.summary$bic, nrVar=1:17), aes(x=nrVar, y=bic))+xlab("Number of Variables") + ylab("BIC") + geom\_line()



which.min(reg.summary$bic)

## [1] 6

ggplot(data.frame(adjr2 =reg.summary$adjr2, nrVar=1:17), aes(x=nrVar, y=adjr2))+xlab("Number of Variables") + ylab("adjr2") + geom\_line()



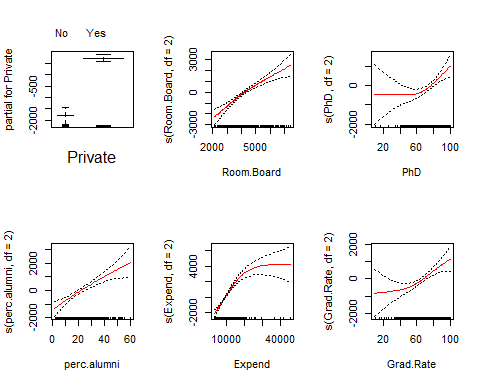
which.max(reg.summary$adjr2)

## [1] 13

co <- coef(reg.fit, id = 6)  
names(co)

## [1] "(Intercept)" "PrivateYes" "Room.Board" "Terminal" "perc.alumni"  
## [6] "Expend" "Grad.Rate"

gam.fit <- gam(Outstate ~ Private + s(Room.Board, df = 2) + s(PhD, df = 2) +s(perc.alumni, df = 2) + s(Expend, df = 2) + s(Grad.Rate, df = 2), data = dt.train)  
par(mfrow = c(2, 3))  
plot(gam.fit, se = T, col = "red")

 (c)

gam.pred <- predict(gam.fit, dt.test)  
gam.err <- mean((dt.test$Outstate - gam.pred)^2)  
gam.err

## [1] 3917505

lm.pred <- predict(lm(Outstate~Private+Room.Board+PhD+perc.alumni+Expend+Grad.Rate, data = dt.train), dt.test)  
lm.err <- mean((dt.test$Outstate - lm.pred)^2)  
lm.err

## [1] 4366402

summary(gam.fit)

##   
## Call: gam(formula = Outstate ~ Private + s(Room.Board, df = 2) + s(PhD,   
## df = 2) + s(perc.alumni, df = 2) + s(Expend, df = 2) + s(Grad.Rate,   
## df = 2), data = dt.train)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -4998.13 -1270.38 -56.62 1144.30 8654.66   
##   
## (Dispersion Parameter for gaussian family taken to be 3415465)  
##   
## Null Deviance: 6221998532 on 387 degrees of freedom  
## Residual Deviance: 1284215010 on 376 degrees of freedom  
## AIC: 6951.911   
##   
## Number of Local Scoring Iterations: 2   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## Private 1 1814362238 1814362238 531.220 < 2.2e-16 \*\*\*  
## s(Room.Board, df = 2) 1 1282301901 1282301901 375.440 < 2.2e-16 \*\*\*  
## s(PhD, df = 2) 1 411533247 411533247 120.491 < 2.2e-16 \*\*\*  
## s(perc.alumni, df = 2) 1 351023025 351023025 102.775 < 2.2e-16 \*\*\*  
## s(Expend, df = 2) 1 354006775 354006775 103.648 < 2.2e-16 \*\*\*  
## s(Grad.Rate, df = 2) 1 52725971 52725971 15.437 0.0001015 \*\*\*  
## Residuals 376 1284215010 3415465   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## Private   
## s(Room.Board, df = 2) 1 3.409 0.065637 .   
## s(PhD, df = 2) 1 7.133 0.007897 \*\*   
## s(perc.alumni, df = 2) 1 0.735 0.391874   
## s(Expend, df = 2) 1 46.337 3.947e-11 \*\*\*  
## s(Grad.Rate, df = 2) 1 3.814 0.051569 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

expend and outstate ,phd and outstate. 都有非線性的相關 11.(a,b)

set.seed(666)  
y <- rnorm(100)  
x1 <- rnorm(100)  
x2 <- rnorm(100)  
beta1 <- 3.2

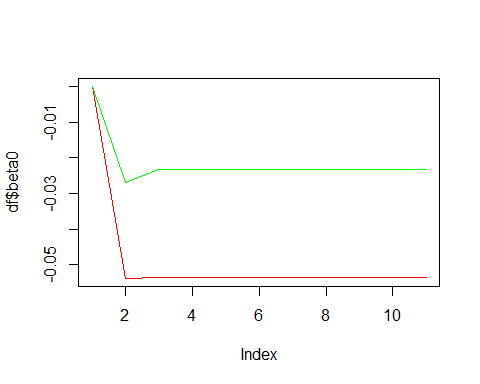
a <- y - beta1 \* x1  
beta2 <- lm(a ~ x2)$coef[2]

a <- y - beta2 \* x2  
beta1 <- lm(a ~ x1)$coef[2]

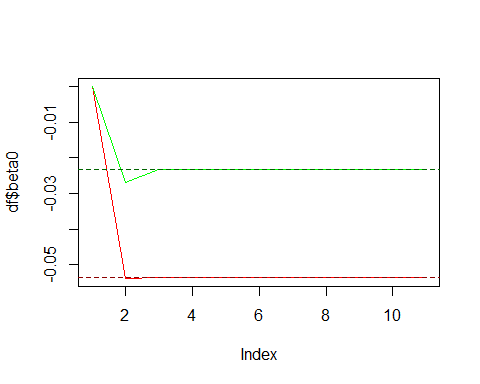
iter <- 10  
df <- data.frame(0.0, 0.27, 0.0)  
names(df) <- c('beta0', 'beta1', 'beta2')  
for (i in 1:iter) {  
 beta1 <- df[nrow(df), 2]  
 a <- y - beta1 \* x1  
 beta2 <- lm(a ~ x2)$coef[2]  
 a <- y - beta2 \* x2  
 beta1 <- lm(a ~ x1)$coef[2]  
 beta0 <- lm(a ~ x1)$coef[1]  
 print(beta0)  
 print(beta1)  
 print(beta2)  
 df[nrow(df) + 1,] <- list(beta0, beta1, beta2)  
}

## (Intercept)   
## -0.05377207   
## x1   
## 0.1648746   
## x2   
## -0.02701404   
## (Intercept)   
## -0.05345771   
## x1   
## 0.1647774   
## x2   
## -0.02337703   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367   
## (Intercept)   
## -0.05345742   
## x1   
## 0.1647774   
## x2   
## -0.02337367

plot(df$beta0, col = 'red', type = 'l')  
lines(df$beta1, col = 'blue')  
lines(df$beta2, col = 'green')

 (f)

plot(df$beta0, col = 'red', type = 'l')  
lines(df$beta1, col = 'blue')  
  
lines(df$beta2, col = 'green')  
res <- coef(lm(y ~ x1 + x2))  
abline(h = res[1], col = 'darkred', lty = 2)  
  
abline(h = res[2], col = 'darkblue', lty = 2)  
abline(h = res[3], col = 'darkgreen', lty = 2)



8.(a)

library(ISLR)  
set.seed(1)  
train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)  
Carseats.train <- Carseats[train, ]  
Carseats.test <- Carseats[-train, ]

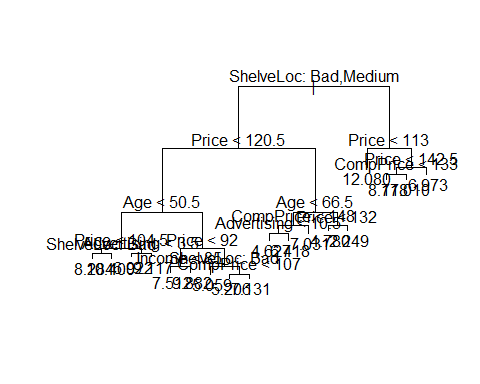
library(tree)

## Warning: package 'tree' was built under R version 3.4.4

tree.carseats <- tree(Sales ~ ., data = Carseats.train)  
summary(tree.carseats)

##   
## Regression tree:  
## tree(formula = Sales ~ ., data = Carseats.train)  
## Variables actually used in tree construction:  
## [1] "ShelveLoc" "Price" "Age" "Advertising" "Income"   
## [6] "CompPrice"   
## Number of terminal nodes: 18   
## Residual mean deviance: 2.36 = 429.5 / 182   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -4.2570 -1.0360 0.1024 0.0000 0.9301 3.9130

plot(tree.carseats)  
text(tree.carseats, pretty = 0)



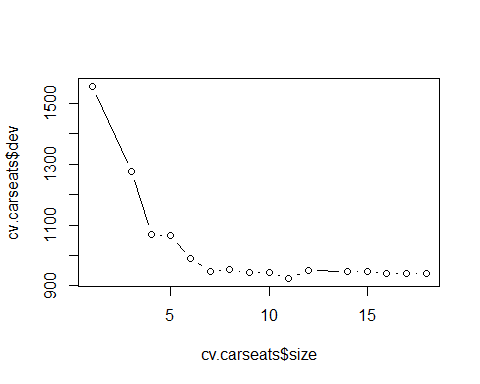
yhat <- predict(tree.carseats, newdata = Carseats.test)  
mean((yhat - Carseats.test$Sales)^2)

## [1] 4.148897

set.seed(66)  
cv.carseats <- cv.tree(tree.carseats)  
cv.carseats

## $size  
## [1] 18 17 16 15 14 12 11 10 9 8 7 6 5 4 3 1  
##   
## $dev  
## [1] 939.9094 940.8531 939.3613 947.0518 947.0518 950.7363 924.3627  
## [8] 943.2623 943.2483 952.6955 948.6485 989.8776 1064.0029 1070.0242  
## [15] 1277.8642 1556.5289  
##   
## $k  
## [1] -Inf 15.48181 15.53599 18.69038 18.74886 21.05038 23.79480  
## [8] 25.78579 26.01210 30.10435 32.74801 53.28569 72.33061 78.19599  
## [15] 141.73781 251.22901  
##   
## $method  
## [1] "deviance"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

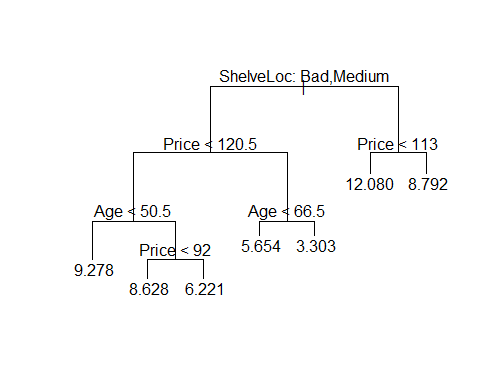
plot(cv.carseats$size, cv.carseats$dev, type = "b")



tree.min <- which.min(cv.carseats$dev)  
tree.min

## [1] 7

prune.carseats <- prune.tree(tree.carseats, best = 7)  
plot(prune.carseats)  
text(prune.carseats, pretty = 0)



yhat <- predict(prune.carseats, newdata = Carseats.test)  
mean((yhat - Carseats.test$Sales)^2)

## [1] 5.340397

set.seed(2)  
bag.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500, importance = TRUE)  
yhat.bag <- predict(bag.carseats, newdata = Carseats.test)  
mean((yhat.bag - Carseats.test$Sales)^2)

## [1] 2.589997

importance(bag.carseats)

## %IncMSE IncNodePurity  
## CompPrice 13.2344536 131.849636  
## Income 4.6367511 75.526457  
## Advertising 16.6245040 121.816698  
## Population 1.6032835 64.313699  
## Price 52.7977583 514.114830  
## ShelveLoc 44.2943454 322.568738  
## Age 21.5475296 189.109365  
## Education 0.1292647 41.497630  
## Urban -1.5963483 8.910302  
## US 7.0788293 14.977665

“Price” and “ShelveLoc” are the two most important variables.

set.seed(2)  
rf.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree = 500, importance = TRUE)  
yhat.rf <- predict(rf.carseats, newdata = Carseats.test)  
mean((yhat.rf - Carseats.test$Sales)^2)

## [1] 3.330217

importance(rf.carseats)

## %IncMSE IncNodePurity  
## CompPrice 8.1072721 129.32783  
## Income 3.7431236 130.88067  
## Advertising 13.9075613 142.85316  
## Population 0.3403256 98.64407  
## Price 38.3612684 381.39357  
## ShelveLoc 31.7251761 231.59902  
## Age 16.9960584 202.01140  
## Education 1.5026488 68.56184  
## Urban -2.6623358 14.19184  
## US 7.0944213 32.56292

m^2=p 這裡取3 “Price” and “ShelveLoc” are the two most important variables.

Hitters <- na.omit(Hitters)  
Hitters$Salary <- log(Hitters$Salary)

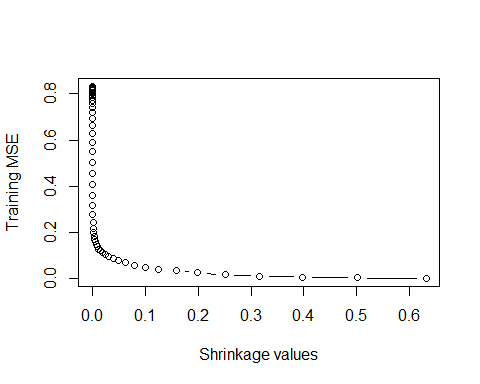
Hitters.train <- Hitters[1:200,]  
Hitters.test <- Hitters[201:263, ]

library(gbm)

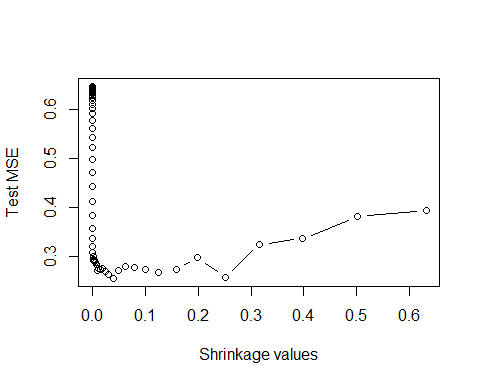
## Warning: package 'gbm' was built under R version 3.4.4

## Loaded gbm 2.1.4

set.seed(3)  
pows <- seq(-10, -0.2, by = 0.1)  
lambdas <- 10^pows  
train.err <- rep(NA, length(lambdas))  
for (i in 1:length(lambdas)) {  
 boost.hitters <- gbm(Salary ~ ., data = Hitters.train, distribution = "gaussian", n.trees = 1000, shrinkage = lambdas[i])  
 pred.train <- predict(boost.hitters, Hitters.train, n.trees = 1000)  
 train.err[i] <- mean((pred.train - Hitters.train$Salary)^2)  
}  
plot(lambdas, train.err, type = "b", xlab = "Shrinkage values", ylab = "Training MSE")

 (d)

set.seed(3)  
test.err <- rep(NA, length(lambdas))  
for (i in 1:length(lambdas)) {  
 boost.hitters <- gbm(Salary ~ ., data = Hitters.train, distribution = "gaussian", n.trees = 1000, shrinkage = lambdas[i])  
 yhat <- predict(boost.hitters, Hitters.test, n.trees = 1000)  
 test.err[i] <- mean((yhat - Hitters.test$Salary)^2)  
}  
plot(lambdas, test.err, type = "b", xlab = "Shrinkage values", ylab = "Test MSE")



min(test.err)

## [1] 0.2556347

lambdas[which.min(test.err)]

## [1] 0.03981072

minimum test MSE is 0.2556347, and is obtained for λ=0.0398.

set.seed(3)  
library(glmnet)

## Warning: package 'glmnet' was built under R version 3.4.4

## Loading required package: Matrix

## Loaded glmnet 2.0-16

x <- model.matrix(Salary ~ ., data = Hitters.train)  
x.test <- model.matrix(Salary ~ ., data = Hitters.test)  
y <- Hitters.train$Salary  
fit1 <- glmnet(x, y, alpha = 0)  
pred1 <- predict(fit1, s = 0.01, newx = x.test)  
mean((pred1 - Hitters.test$Salary)^2)

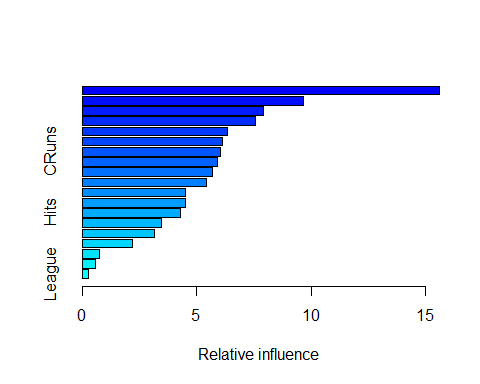
## [1] 0.4570283

fit2 <- glmnet(x, y, alpha = 1)  
pred2 <- predict(fit2, s = 0.01, newx = x.test)  
mean((pred2 - Hitters.test$Salary)^2)

## [1] 0.4700537

The test MSE for boosting is lower than for lasso regression and ridge regression.

library(gbm)  
boost.hitters <- gbm(Salary ~ ., data = Hitters.train, distribution = "gaussian", n.trees = 5000, shrinkage = lambdas[which.min(test.err)])  
summary(boost.hitters)



## var rel.inf  
## CAtBat CAtBat 15.6183486  
## PutOuts PutOuts 9.6836489  
## Walks Walks 7.9265626  
## CRBI CRBI 7.5584225  
## CWalks CWalks 6.3285703  
## Assists Assists 6.1232738  
## CRuns CRuns 6.0587244  
## CHmRun CHmRun 5.9038047  
## CHits CHits 5.6982699  
## Years Years 5.4087280  
## RBI RBI 4.5121143  
## AtBat AtBat 4.5030386  
## Hits Hits 4.2766237  
## HmRun HmRun 3.4823999  
## Runs Runs 3.1408200  
## Errors Errors 2.1889159  
## Division Division 0.7611512  
## NewLeague NewLeague 0.5721485  
## League League 0.2544340

“CAtBat” is the most important variable. (g)

set.seed(3)  
bag.hitters <- randomForest(Salary ~ ., data = Hitters.train, mtry = 19, ntree = 500)  
yhat.bag <- predict(bag.hitters, newdata = Hitters.test)  
mean((yhat.bag - Hitters.test$Salary)^2)

## [1] 0.2316231

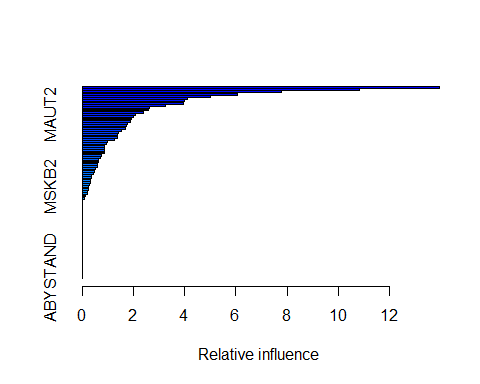
set.seed(66)  
train <- 1:1000  
Caravan.train <- Caravan[train, ]  
Caravan.test <- Caravan[-train, ]

set.seed(66)  
Caravan$Purchase <- ifelse(Caravan$Purchase == "Yes", 1, 0)  
boost.caravan <- gbm(Purchase ~ ., data = Caravan.train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.01)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 50: PVRAAUT has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =  
## distribution, : variable 71: AVRAAUT has no variation.

summary(boost.caravan)



## var rel.inf  
## PPERSAUT PPERSAUT 13.95296765  
## MKOOPKLA MKOOPKLA 10.83831350  
## MOPLHOOG MOPLHOOG 7.76549598  
## MBERMIDD MBERMIDD 6.05741299  
## PBRAND PBRAND 5.00733872  
## MGODGE MGODGE 4.12113823  
## ABRAND ABRAND 3.99513883  
## MINK3045 MINK3045 3.95740866  
## PWAPART PWAPART 3.24932769  
## MOSTYPE MOSTYPE 2.62094769  
## MAUT1 MAUT1 2.57840012  
## MSKC MSKC 2.39579479  
## MAUT2 MAUT2 2.07151938  
## MGODPR MGODPR 2.00075807  
## MBERARBG MBERARBG 1.90579453  
## MBERHOOG MBERHOOG 1.86562082  
## MFWEKIND MFWEKIND 1.77056258  
## MSKA MSKA 1.73559519  
## PBYSTAND PBYSTAND 1.70273449  
## MRELGE MRELGE 1.52521671  
## MFGEKIND MFGEKIND 1.39816091  
## MINKGEM MINKGEM 1.36763961  
## MSKB1 MSKB1 1.35557971  
## MGODOV MGODOV 1.26912366  
## MAUT0 MAUT0 0.99204495  
## MGODRK MGODRK 0.94965603  
## MINK7512 MINK7512 0.88036195  
## MHKOOP MHKOOP 0.87563109  
## MOPLMIDD MOPLMIDD 0.86994603  
## MINKM30 MINKM30 0.86755742  
## MINK4575 MINK4575 0.73476838  
## MZFONDS MZFONDS 0.70893373  
## MRELOV MRELOV 0.63245337  
## MBERBOER MBERBOER 0.63181952  
## MHHUUR MHHUUR 0.61101915  
## MGEMOMV MGEMOMV 0.60442559  
## MBERARBO MBERARBO 0.50065137  
## MOSHOOFD MOSHOOFD 0.46697934  
## MFALLEEN MFALLEEN 0.42089811  
## MSKD MSKD 0.36000330  
## MRELSA MRELSA 0.34739723  
## MZPART MZPART 0.33560963  
## APERSAUT APERSAUT 0.32535124  
## PLEVEN PLEVEN 0.26300564  
## MSKB2 MSKB2 0.25874648  
## MGEMLEEF MGEMLEEF 0.22417065  
## PMOTSCO PMOTSCO 0.20114921  
## MBERZELF MBERZELF 0.19938501  
## MOPLLAAG MOPLLAAG 0.13071190  
## MINK123M MINK123M 0.09933319  
## MAANTHUI MAANTHUI 0.00000000  
## PWABEDR PWABEDR 0.00000000  
## PWALAND PWALAND 0.00000000  
## PBESAUT PBESAUT 0.00000000  
## PVRAAUT PVRAAUT 0.00000000  
## PAANHANG PAANHANG 0.00000000  
## PTRACTOR PTRACTOR 0.00000000  
## PWERKT PWERKT 0.00000000  
## PBROM PBROM 0.00000000  
## PPERSONG PPERSONG 0.00000000  
## PGEZONG PGEZONG 0.00000000  
## PWAOREG PWAOREG 0.00000000  
## PZEILPL PZEILPL 0.00000000  
## PPLEZIER PPLEZIER 0.00000000  
## PFIETS PFIETS 0.00000000  
## PINBOED PINBOED 0.00000000  
## AWAPART AWAPART 0.00000000  
## AWABEDR AWABEDR 0.00000000  
## AWALAND AWALAND 0.00000000  
## ABESAUT ABESAUT 0.00000000  
## AMOTSCO AMOTSCO 0.00000000  
## AVRAAUT AVRAAUT 0.00000000  
## AAANHANG AAANHANG 0.00000000  
## ATRACTOR ATRACTOR 0.00000000  
## AWERKT AWERKT 0.00000000  
## ABROM ABROM 0.00000000  
## ALEVEN ALEVEN 0.00000000  
## APERSONG APERSONG 0.00000000  
## AGEZONG AGEZONG 0.00000000  
## AWAOREG AWAOREG 0.00000000  
## AZEILPL AZEILPL 0.00000000  
## APLEZIER APLEZIER 0.00000000  
## AFIETS AFIETS 0.00000000  
## AINBOED AINBOED 0.00000000  
## ABYSTAND ABYSTAND 0.00000000

“PPERSAUT” is the most important variables. (c)

probs.test <- predict(boost.caravan, Caravan.test, n.trees = 1000, type = "response")  
pred.test <- ifelse(probs.test > 0.2, 1, 0)  
table(Caravan.test$Purchase, pred.test)

## pred.test  
## 1  
## No 4533  
## Yes 289

14/(38+14)

## [1] 0.2692308

logit.caravan <- glm(Purchase ~., data = Caravan.train, family = "binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

probs.test2 <- predict(logit.caravan, Caravan.test, type = "response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

pred.test2 <- ifelse(probs.test > 0.2, 1, 0)  
table(Caravan.test$Purchase, pred.test2)

## pred.test2  
## 1  
## No 4533  
## Yes 289

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

summary(cars)

## speed dist   
## Min. : 4.0 Min. : 2.00   
## 1st Qu.:12.0 1st Qu.: 26.00   
## Median :15.0 Median : 36.00   
## Mean :15.4 Mean : 42.98   
## 3rd Qu.:19.0 3rd Qu.: 56.00   
## Max. :25.0 Max. :120.00

## Including Plots

You can also embed plots, for example:



Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.