Tree-Based Methods Ch.8 Exercises

Lucy L.

2024-02-01

```
library(tree)
library(pls)
library(BART)
library(ISLR2)
library(gbm)
library(glmnet)
library(randomForest)
library(class)
set.seed(1)

carseats = Carseats
attach(Carseats)
```

Exercise 8

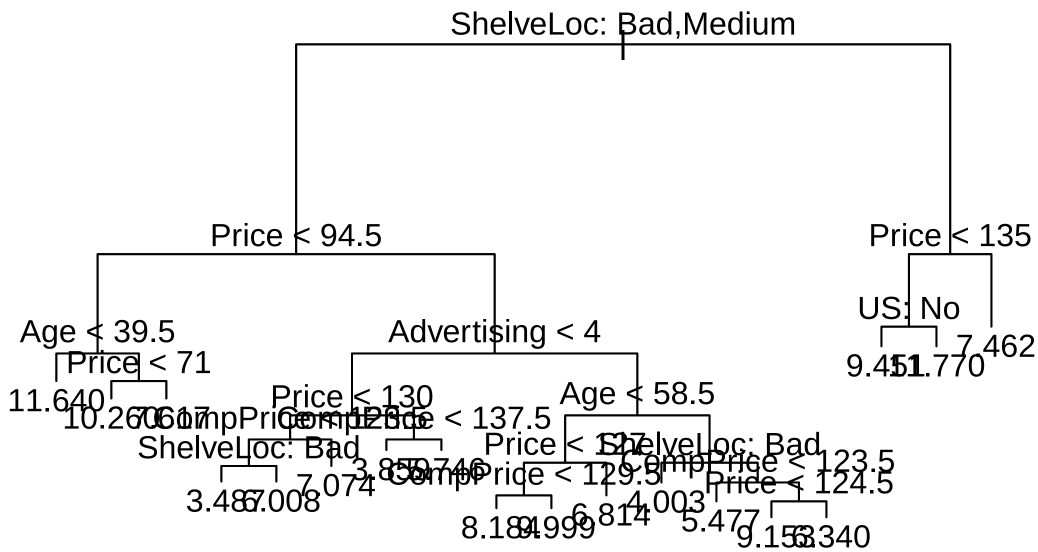
In the lab, a classification tree was applied to the `Carseats` data set after converting `Sales` into a qualitative response variable. Now we will seek to predict `Sales` using regression trees and related approaches, treating the response as a quantitative variable.

(a) Split the data set into a training set and a test set.

```
train = sample(1:nrow(Carseats), 200)
Carseats.test = Carseats[-train, ]
```

(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
# fitting tree
tree.8 = tree(Sales ~ ., data = Carseats, subset = train)

# plotting tree
plot(tree.8)
text(tree.8, pretty = 0)
```




```r
# generating predictions and calculating MSE
tree.8.yhat = predict(tree.8, newdata = Carseats.test)
mean((tree.8.yhat - Carseats.test$Sales)^2)
```

```
## [1] 4.922039
```

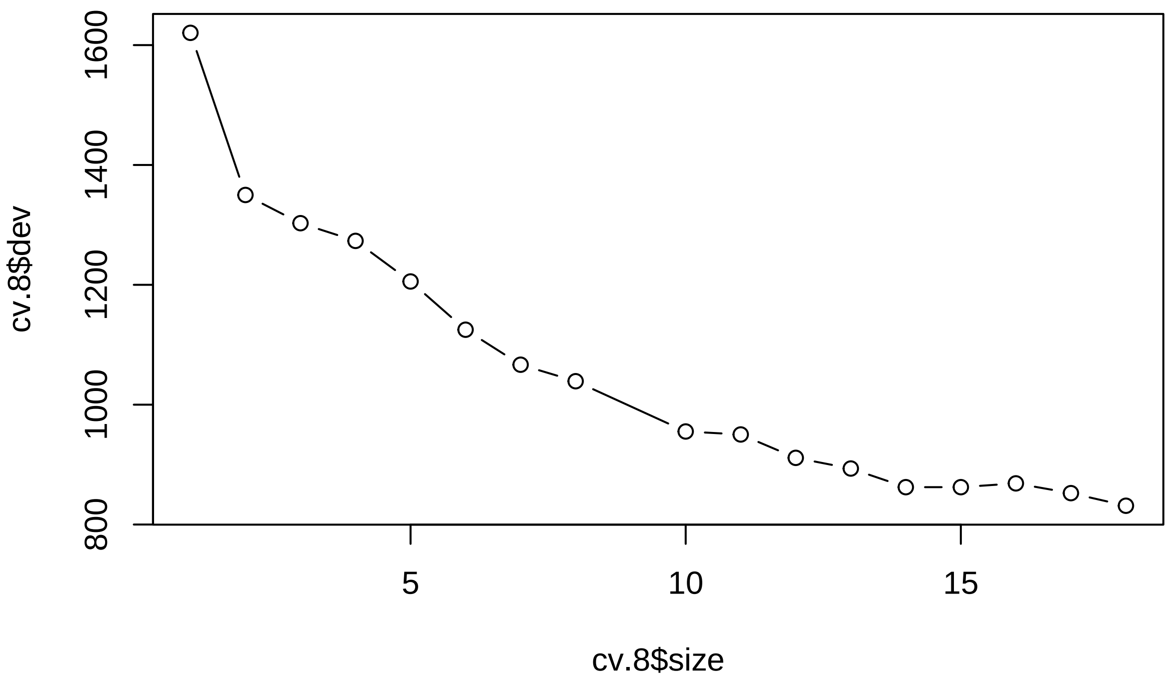
The test MSE is about 4.922.

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```r
cv.8 = cv.tree(tree.8)
names(cv.8)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
# plot error rate as function of predictors
plot(cv.8$size, cv.8$dev, type = "b")
```




```r
# doing a small prune and calculating test MSE
prune.8 =  prune.tree(tree.8, best = 17)
tree.pred = predict(prune.8, newdata = Carseats.test)
mean((tree.pred - Carseats.test$Sales)^2)
```

```
## [1] 4.827162
```

The size of the tree that results in the lowest error is size 17. Pruning the tree slightly has slightly reduced the test MSE. The test MSE is now about 4.827.

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the `importance()` function to determine which variables are most important.

```r
set.seed(1)
bag.8 = randomForest(Sales ~ ., data = Carseats, subset = train,
                     mtry = 10, importance = TRUE)

importance(bag.8)
```

```
##                %IncMSE IncNodePurity
## CompPrice   24.8888481    170.182937
## Income       4.7121131     91.264880
## Advertising 12.7692401     97.164338
## Population  -1.8074075     58.244596
```

```
## Price       56.3326252    502.903407
## ShelveLoc   48.8886689    380.032715
## Age         17.7275460    157.846774
## Education    0.5962186     44.598731
## Urban        0.1728373      9.822082
## US           4.2172102     18.073863
```

```r
yhat.bag = predict(bag.8, newdata = Carseats.test)
mean((yhat.bag - Carseats.test$Sales)^2)
```

```
## [1] 2.605253
```

The most important variables by order of decreasing purity are `Price`, `ShelveLoc`, `CompPrice`, `Age`, and `Advertising`. The test MSE is about 2.605.

(e) Use random forests to analyze this data. What test MSE do you obtain? Use the `importance()` function to determine which variables are most important. Describe the effect of `m`, the number of variables considered at each split, on the error rate obtained.

```r
set.seed(1)

# building a random forest
rf.8 = randomForest(Sales ~ ., data = Carseats, subset = train,
                    mtry = 5, importance = TRUE)

importance(rf.8)
```

```
##                %IncMSE IncNodePurity
## CompPrice   17.4126238     157.53631
## Income       2.9969399     110.40731
## Advertising 11.0485672     105.75049
## Population  -1.5321044      80.73318
## Price       43.3572135     452.02367
## ShelveLoc   44.4474163     331.64508
## Age         14.5322339     176.64252
## Education    0.8237454      55.91141
## Urban       -2.7805788      11.07321
## US           3.7773881      23.75322
```

```r
yhat.rf = predict(rf.8, newdata = Carseats.test)
mean((yhat.rf - Carseats.test$Sales)^2)
```

```
## [1] 2.714168
```

The test MSE obtained is about 2.714. The variables considered most important now are the same, but now with `Income` included. Reducing `m` to be the value of the important variables with purity over 100 has slightly increased the test MSE.

(f) Now analyze the data using BART, and report your results.

```r
set.seed(1)

x = Carseats[, 2:11]
y = Boston[, 1]
xtrain = x[train, ]
ytrain = y[train]
xtest = x[-train, ]
ytest = y[-train]

bart.fit = gbart(xtrain, ytrain, x.test = xtest)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 200, 14, 200
## y1,yn: -1.469566, -1.141946
## x1,x[n*p]: 107.000000, 1.000000
## xp1,xp[np*p]: 111.000000, 1.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 63 ... 1
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,1.57278,3,10.5381,1.75349
## *****sigma: 7.355236
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,14,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 2s
## trcnt,tecnt: 1000,1000
```

```r
# compute test error
yhat.bart = bart.fit$yhat.test.mean
mean((ytest - yhat.bart)^2)
```

```
## [1] 101.5311
```

The test MSE is about 101.531.

Exercise 10

We now use boosting to predict `Salary` in the `Hitters` data set.

(a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

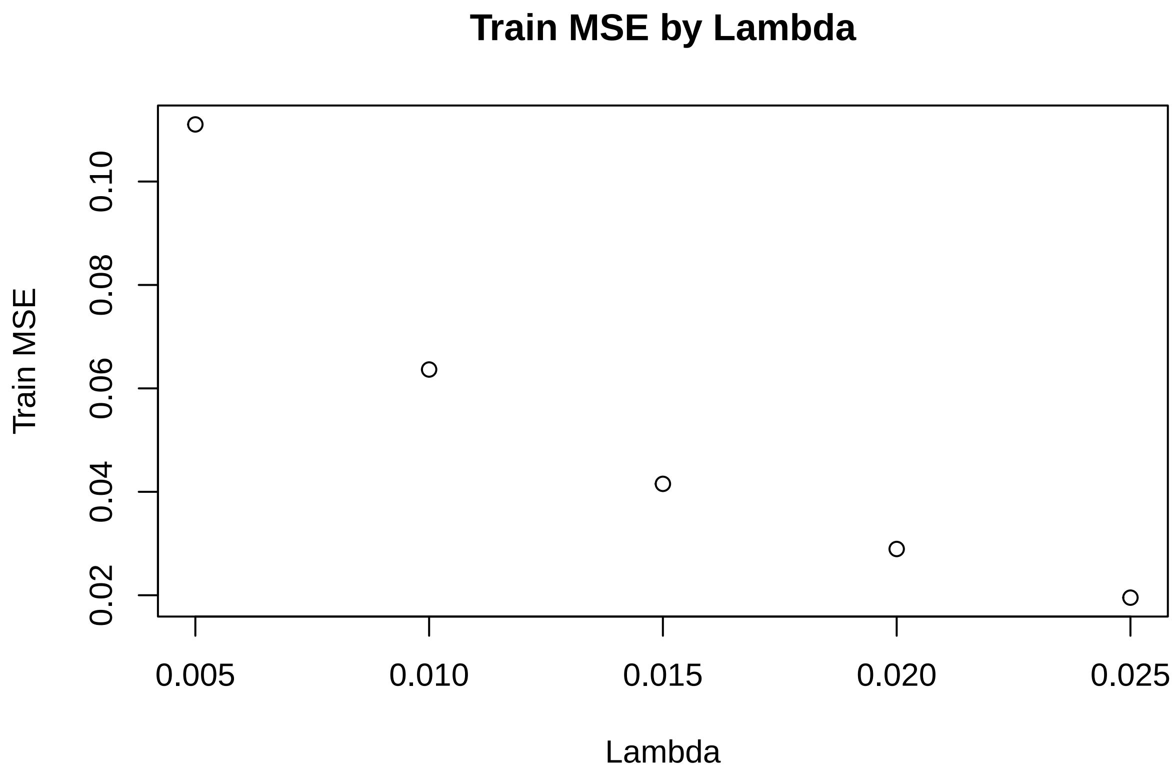
```
Hitters = Hitters[!is.na(Hitters$Salary),]
Hitters$log_salary = log(Hitters$Salary)
```

(b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
train_hitters = sample(1:nrow(Hitters), 200)
train = Hitters[train_hitters, ]
test = Hitters[-train_hitters, ]
```

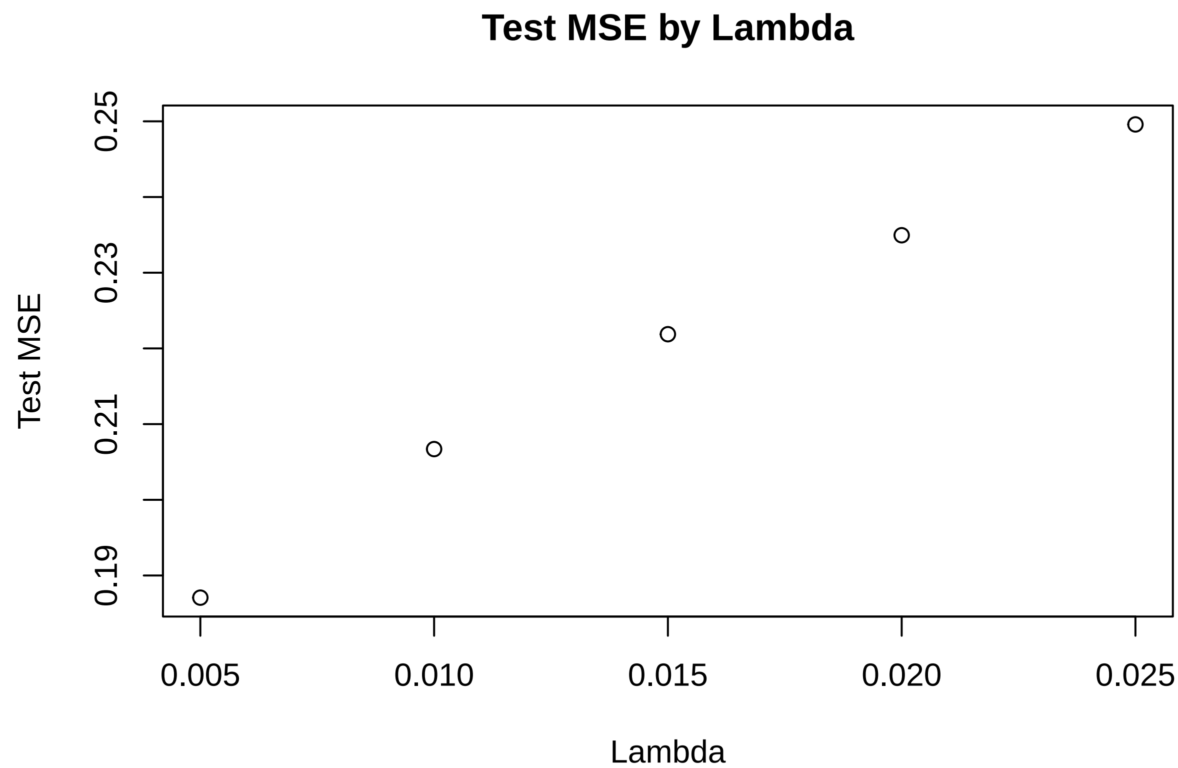
(c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

```
set.seed(1)

start = 1
train_mses = seq(0, 4)
test_mses = seq(0, 4)
lambdas = c(0.005, 0.01, 0.015, 0.02, 0.025)

for (i in lambdas) {
  boost.10 = gbm(log_salary ~ . - Salary, data = train,
                 distribution = "gaussian",
                 n.trees = 1000,
                 interaction.depth = 4,
                 shrinkage = i,
                 verbose = F)

  yhat.boost.train = predict(boost.10, newdata = train, n.trees = 1000)
  yhat.boost.test = predict(boost.10, newdata = test, n.trees = 1000)
  train_mses[start] = mean((yhat.boost.train - train$log_salary)^2)
  test_mses[start] = mean((yhat.boost.test - test$log_salary)^2)

  start = start + 1
}

plot(lambdas, train_mses, main = "Train MSE by Lambda", ylab = "Train MSE",
     xlab = "Lambda")
```



(d) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```
plot(lambdas, test_mses, main = "Test MSE by Lambda", ylab = "Test MSE",
     xlab = "Lambda")
```



(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

```
set.seed(1)
Hitters = Hitters[, -19]

# performing Lasso
x =  model.matrix(log_salary ~ ., Hitters)
y = Hitters$log_salary
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
y.test = y[test]
grid = 10^seq(10, -2, length = 100)
hitters_test = Hitters[test, ]

lasso.mod = glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
cv.out = cv.glmnet(x[train, ], y[train], alpha = 1)
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test, ])
mean((lasso.pred - y.test)^2)
```

```
## [1] 0.5024675
```

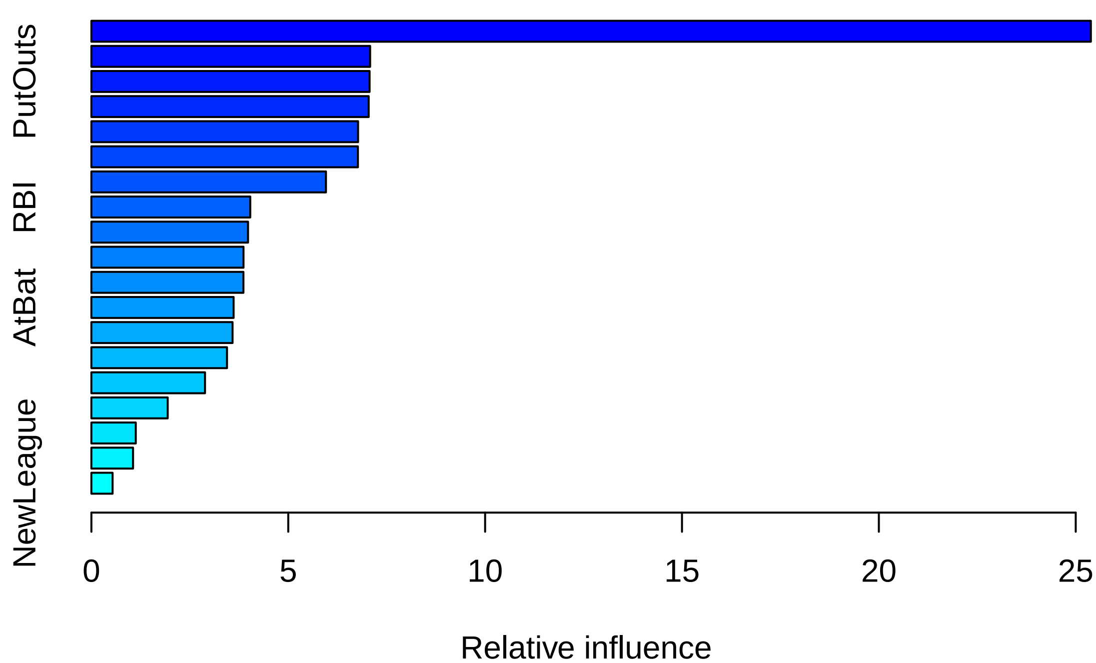
```r
# performing principal components regression
pcr.fit = pcr(log_salary ~ ., data = Hitters, subset = train,
              scale = TRUE,
              validation = "CV")
# summary(pcr.fit)
# validationplot(pcr.fit, val.type = "MSEP")
pcr.pred = predict(pcr.fit, hitters_test, ncomp = 5)
mean((pcr.pred - y.test)^2)
```

```
## [1] 0.5441653
```

The test MSE from performing Lasso on the data is about 0.502 while the test MSE for PCR is about 0.542.

(f) Which variables appear to be the most important predictors in the boosted model?

```r
set.seed(1)
boost.10 = gbm(log_salary ~ ., data = Hitters[train, ],
                   distribution = "gaussian",
                   n.trees = 1000,
                   interaction.depth = 4,
                   verbose = F)
summary(boost.10)
```




```
##                  var    rel.inf
```

```
## CRBI           CRBI 25.3847622
## CHits         CHits  7.0794260
## PutOuts     PutOuts  7.0656021
## CHmRun       CHmRun  7.0424226
## Assists     Assists  6.7728933
## CWalks       CWalks  6.7691790
## CRuns         CRuns  5.9576639
## RBI             RBI  4.0351263
## HmRun         HmRun  3.9786602
## Years         Years  3.8634198
## Walks         Walks  3.8616285
## AtBat         AtBat  3.6120332
## CAtBat       CAtBat  3.5853150
## Runs           Runs  3.4443673
## Hits           Hits  2.8861276
## Errors       Errors  1.9380793
## Division   Division  1.1270121
## League       League  1.0581172
## NewLeague NewLeague  0.5381646
```

The top 4 variables that appear to be the most important are `CRBI`, `CHits`, `PutOuts`, and `CHmRun`.

(g) Now apply bagging to the training set. What is the test set MSE for this approach?

```
set.seed(1)
bag.10 = randomForest(log_salary ~ ., data = Hitters, subset = train,
                      mtry = 12, importance = TRUE)
yhat.bag = predict(bag.10, newdata = Hitters[-train, ])
mean((yhat.bag - y.test)^2)
```

```
## [1] 0.1776044
```

The test MSE for bagging is about 0.178.

Exercise 11

This question uses the `Caravan` data set.

(a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

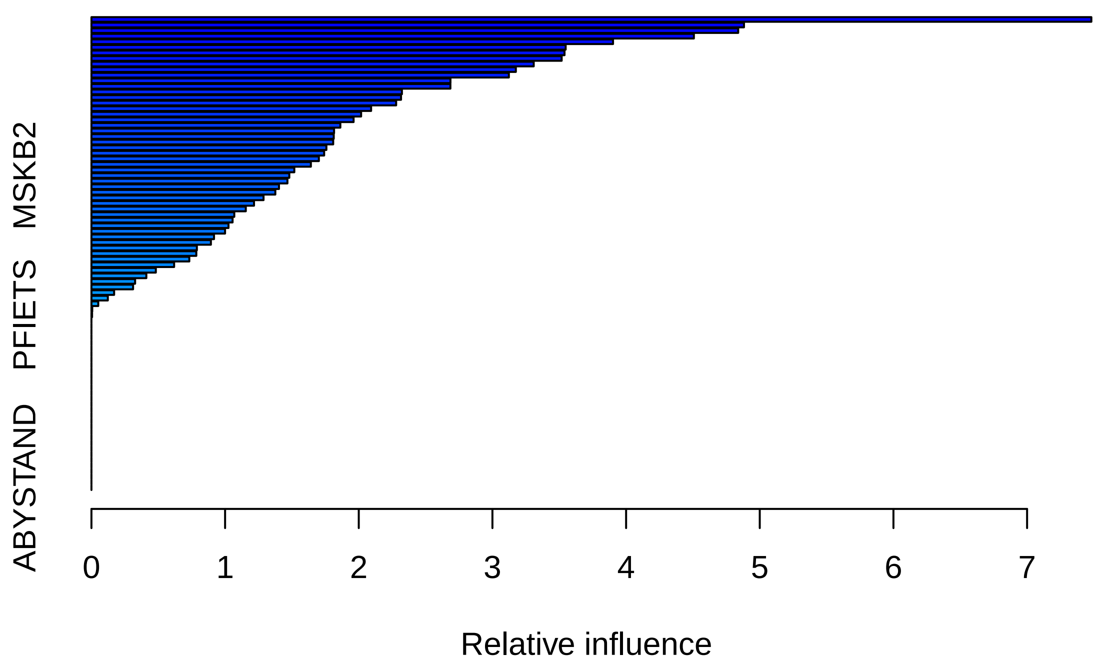
```
caravan = Caravan
train = 1:1000
train.caravan = caravan[train, ]
test.caravan = caravan[-train, ]
```

(b) Fit a boosting model to the training set with `Purchase` as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

```r
set.seed(1)

train.caravan$Purchase <- ifelse(train.caravan$Purchase == "Yes", 1, 0)
test.caravan$Purchase <- ifelse(test.caravan$Purchase == "Yes", 1, 0)

boost.11 = gbm(Purchase ~ ., data = train.caravan, distribution = "bernoulli",
                   n.trees = 1000, shrinkage = 0.01, interaction.depth = 4)
```

```
## 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.
```

```r
summary(boost.11)
```




```
##                 var     rel.inf
## PPERSAUT PPERSAUT 7.480819014
## MOPLHOOG MOPLHOOG 4.882054338
## MGODGE     MGODGE 4.838869962
## MKOOPKLA MKOOPKLA 4.507280400
## MOSTYPE   MOSTYPE 3.902338079
## MGODPR     MGODPR 3.547892360
## PBRAND     PBRAND 3.539487907
```

```
## MBERMIDD MBERMIDD 3.518082698
## MBERARBG MBERARBG 3.309004843
## MINK3045 MINK3045 3.175313873
## MSKC         MSKC 3.123008472
## MSKA         MSKA 2.685844523
## MAUT2       MAUT2 2.685548007
## MAUT1       MAUT1 2.322786246
## PWAPART   PWAPART 2.316252267
## MSKB1       MSKB1 2.279820190
## MRELOV     MRELOV 2.092410309
## MFWEKIND MFWEKIND 2.017651081
## MBERHOOG MBERHOOG 1.961378700
## MBERARBO MBERARBO 1.862074416
## MRELGE     MRELGE 1.815276446
## MINK7512 MINK7512 1.812894054
## MINKM30   MINKM30 1.808781053
## MOPLMIDD MOPLMIDD 1.757784665
## MFGEKIND MFGEKIND 1.741172971
## MGODOV     MGODOV 1.701539077
## MZFONDS   MZFONDS 1.641658796
## MFALLEEN MFALLEEN 1.517763739
## MSKB2       MSKB2 1.480397941
## MINK4575 MINK4575 1.466410983
## MAUT0       MAUT0 1.403097259
## ABRAND     ABRAND 1.375696683
## MHHUUR     MHHUUR 1.287672857
## MINKGEM   MINKGEM 1.216351643
## MHKOOP     MHKOOP 1.154970948
## MGEMLEEF MGEMLEEF 1.068800262
## MGODRK     MGODRK 1.056066524
## MRELSA     MRELSA 1.025383382
## MZPART     MZPART 0.999705745
## MSKD         MSKD 0.917077921
## MGEMOMV   MGEMOMV 0.893757812
## MBERZELF MBERZELF 0.788935429
## APERSAUT APERSAUT 0.784652995
## MOPLLAAG MOPLLAAG 0.732210597
## MOSHOOFD MOSHOOFD 0.618703929
## PMOTSCO   PMOTSCO 0.481824116
## PLEVEN     PLEVEN 0.410808274
## PBYSTAND PBYSTAND 0.326851643
## MBERBOER MBERBOER 0.311571820
## MINK123M MINK123M 0.169710044
## MAANTHUI MAANTHUI 0.122660387
## ALEVEN     ALEVEN 0.051158218
## PAANHANG PAANHANG 0.006040057
## PFIETS     PFIETS 0.004694048
## PWABEDR   PWABEDR 0.000000000
## PWALAND   PWALAND 0.000000000
## PBESAUT   PBESAUT 0.000000000
## PVRAAUT   PVRAAUT 0.000000000
## PTRACTOR PTRACTOR 0.000000000
## PWERKT     PWERKT 0.000000000
## PBROM       PBROM 0.000000000
```

```
## PPERSONG PPERSONG 0.000000000
## PGEZONG   PGEZONG 0.000000000
## PWAOREG   PWAOREG 0.000000000
## PZEILPL   PZEILPL 0.000000000
## PPLEZIER PPLEZIER 0.000000000
## PINBOED   PINBOED 0.000000000
## AWAPART   AWAPART 0.000000000
## AWABEDR   AWABEDR 0.000000000
## AWALAND   AWALAND 0.000000000
## ABESAUT   ABESAUT 0.000000000
## AMOTSCO   AMOTSCO 0.000000000
## AVRAAUT   AVRAAUT 0.000000000
## AAANHANG AAANHANG 0.000000000
## ATRACTOR ATRACTOR 0.000000000
## AWERKT     AWERKT 0.000000000
## ABROM       ABROM 0.000000000
## APERSONG APERSONG 0.000000000
## AGEZONG   AGEZONG 0.000000000
## AWAOREG   AWAOREG 0.000000000
## AZEILPL   AZEILPL 0.000000000
## APLEZIER APLEZIER 0.000000000
## AFIETS     AFIETS 0.000000000
## AINBOED   AINBOED 0.000000000
## ABYSTAND ABYSTAND 0.000000000
```

The most important variables appear to be `PPERSAUT`, `MGODGE`, `MKOOPKLA`, and `MOPLHOOG`.

(c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 percent. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?

```r
set.seed(1)

# creating boosting model
pred.11 = predict(boost.11, newdata = test.caravan, trees = 1000)
```

```
## Using 1000 trees...
```

```r
# person will make a purchase if probability of purchase is over 20 percent
pred.11 <- ifelse(pred.11 > 0.2, 1, 0)

# confusion matrix
table(pred.11, test.caravan$Purchase)
```

```
##
## pred.11    0    1
##       0 4509  279
##       1   24   10
```

```r
(4508 + 9) / length(pred.11)
```

```
## [1] 0.9367482
```

```r
# Applying logistic regression
glm.fits = glm(Purchase ~ ., data = train.caravan, family = binomial)
glm.probs = predict(glm.fits, test.caravan, type = "response")
glm.probs <- ifelse(glm.probs > 0.2, 1, 0)
table(glm.probs, test.caravan$Purchase)
```

```
##
## glm.probs    0    1
##         0 4183  231
##         1  350   58
```

```r
(4183 + 58) / length(glm.probs)
```

```
## [1] 0.8795106
```

Based on the boosting model, about 93.67 percent of people who are predicted to make a purchase actually do make one. This is higher than the accuracy rate from applying logistic regression, which accurately predicts about 87.95 percent of intended purchases.