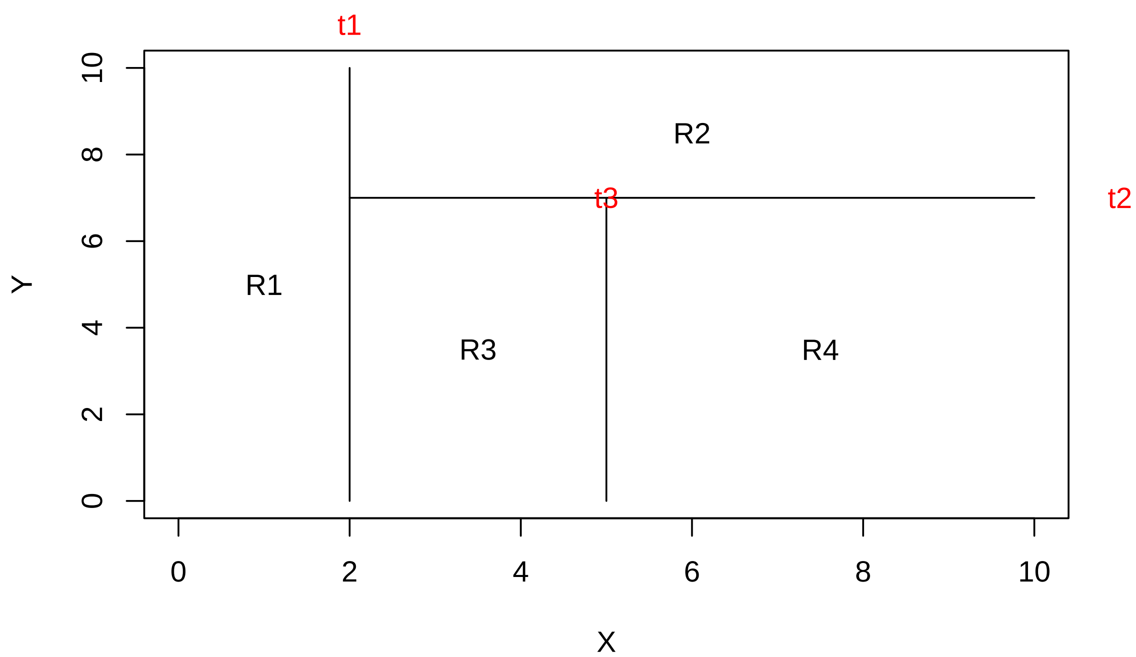
Tree Homework

Tao He

2/27/2022

8.1 ***sol.n:***

```r
par(xpd = NA)
plot(NA, NA, xlim = c(0,10), ylim = c(0,10), xlab = "X", ylab = "Y")

# t1: x = 2; (0, 2) (2, 10)
lines(x = c(2, 2), y = c(0, 10))
text(x = 2, y = 11, labels = c("t1"), col = "red")
# t2: y = 8; (2, 10) (7, 7)
lines(x = c(2, 10), y = c(7, 7))
text(x = 11, y = 7, labels = c("t2"), col = "red")
# t3: x = 5; (5, 5) (0, 7)
lines(x = c(5, 5), y = c(0, 7))
text(x = 5, y = 7, labels = c("t3"), col = "red")

text(x = (0 + 2)/2, y = 5, labels = c("R1"))
text(x = (2 + 10)/2, y = (7 + 10)/2, labels = c("R2"))
text(x = (2 + 5)/2, y = (0 + 7)/2, labels = c("R3"))
text(x = (5 + 10)/2, y = (0 + 7)/2, labels = c("R4"))
```



8.2 ***sol.n:*** Explain

$$f(X) = \sum_{i=1}^{p} f_i(X_j)$$

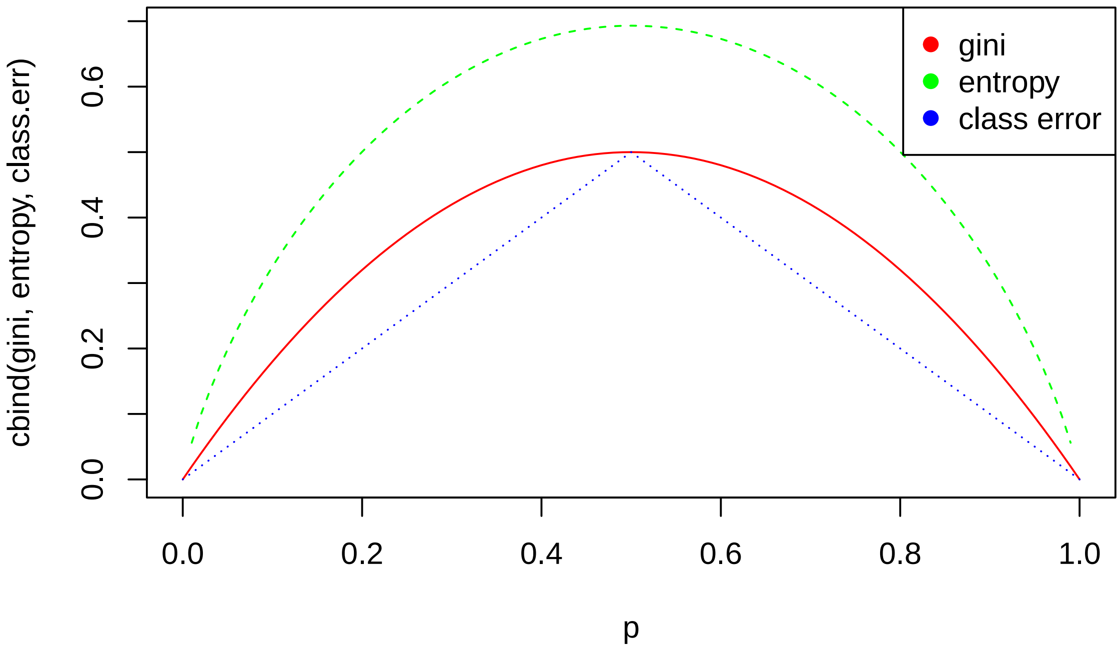
If d = 1 every term in

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

is based on a single predictor. All these terms are summed up making the model additive.

8.3 ***sol.n:***

```
p <- seq(0, 1, 0.01)
gini <- p * (1 - p) * 2
entropy <- -(p * log(p) + (1 - p) * log(1 - p))
class.err <- 1 - pmax(p, 1 - p)
matplot(p, cbind(gini, entropy, class.err), type = "l", col = c("red", "green", "blue"))
legend("topright",legend=c("gini","entropy", "class error"),pch=19,col=c("red", "green", "blue"))
```



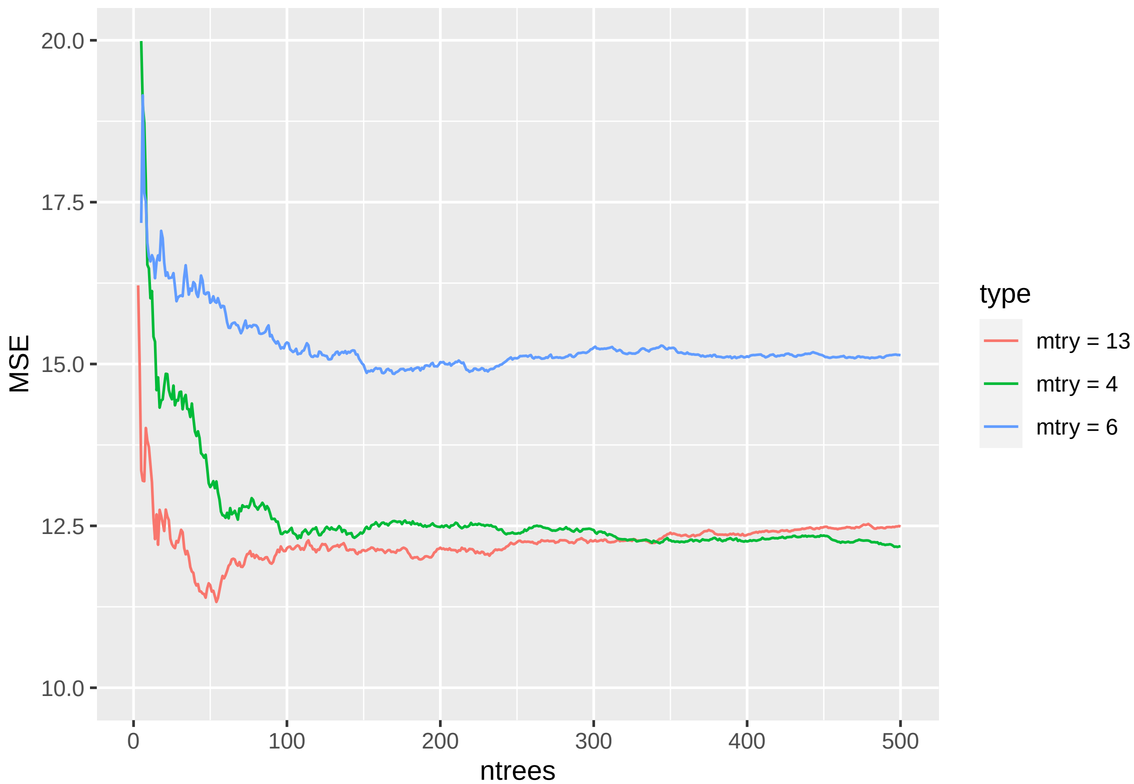
8.5 ***sol.n:*** majority vote: 6 vs 4 –> red avg. prob.: P(Class is Red|X) = 4.5/10 = 0.45 –> green

8.7

```
set.seed(679)
train <- sample(1 : nrow(Boston), nrow(Boston) / 2)
test <- (-train)
boston.train <- Boston[train, -14]
boston.test <- Boston[-train, -14]

y.train <- Boston[train, "medv"]
y.test <- Boston[test, "medv"]
```

```
ntrees <- 500
p <- ncol(boston.train)
test.mse <- c()
type <- c()
```

```
mtry <- c(round(p/2), round(sqrt(p)), p)
for (i in mtry) {
  rf.boston <- randomForest(x = boston.train, y = y.train,
                            xtest = boston.test, ytest = y.test,
                            ntree = ntrees, mtry = i)
  test.mse <- c(rf.boston$test$mse, test.mse)
}
for (i in mtry) {
  type <- c(type, paste("mtry = ", rep(i, ntrees), sep = ""))
}
plot.data <- data.frame(
  ntrees = rep(c(1:ntrees), length(mtry)),
  MSE = test.mse,
  type = factor(type))
ggplot(data = plot.data) +
  geom_line(aes(x = ntrees, y = MSE, group = type, color = type)) + ylim(10, 20)
```



In this plot, we can see that for a single tree, the test MSE is very high. The test MSE decreases as the number of trees increases.

8.8

(a)

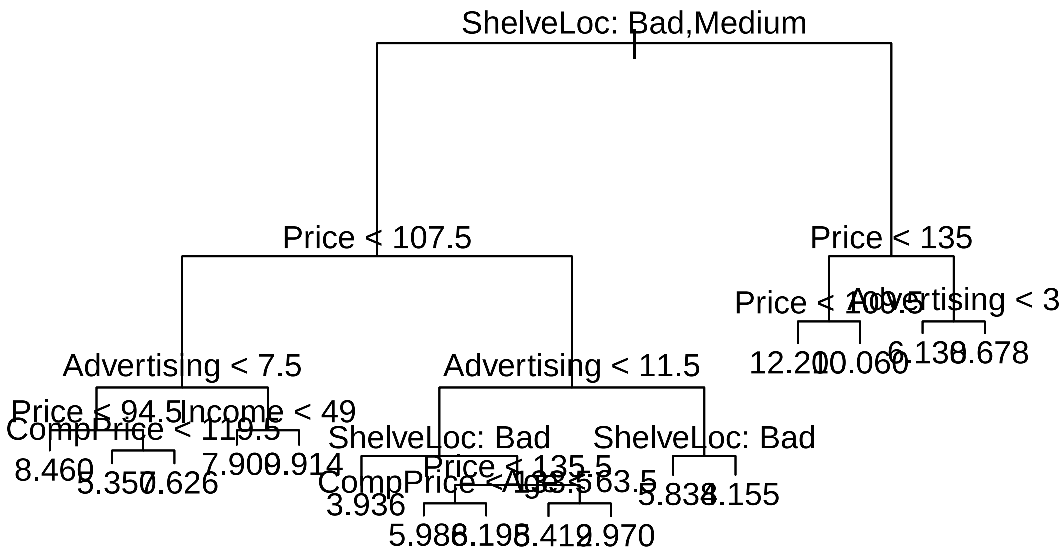
```
# Split the data set into a training set and a test set.
set.seed(679)

train <- sample(1 : nrow(Carseats), nrow(Carseats) /2 )
car.train <- Carseats[train,]
car.test <- Carseats[-train,]
```

(b)

```
reg.tree <- tree(Sales ~. , data = Carseats, subset = train)
summary(reg.tree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats, subset = train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Advertising" "CompPrice"   "Income"
## [6] "Age"
## Number of terminal nodes:  16
## Residual mean deviance:  2.559 = 470.9 / 184
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.4140 -1.1860 -0.0050  0.0000  0.9284  4.9000
```

```
plot(reg.tree)
text(reg.tree ,pretty =0)
```

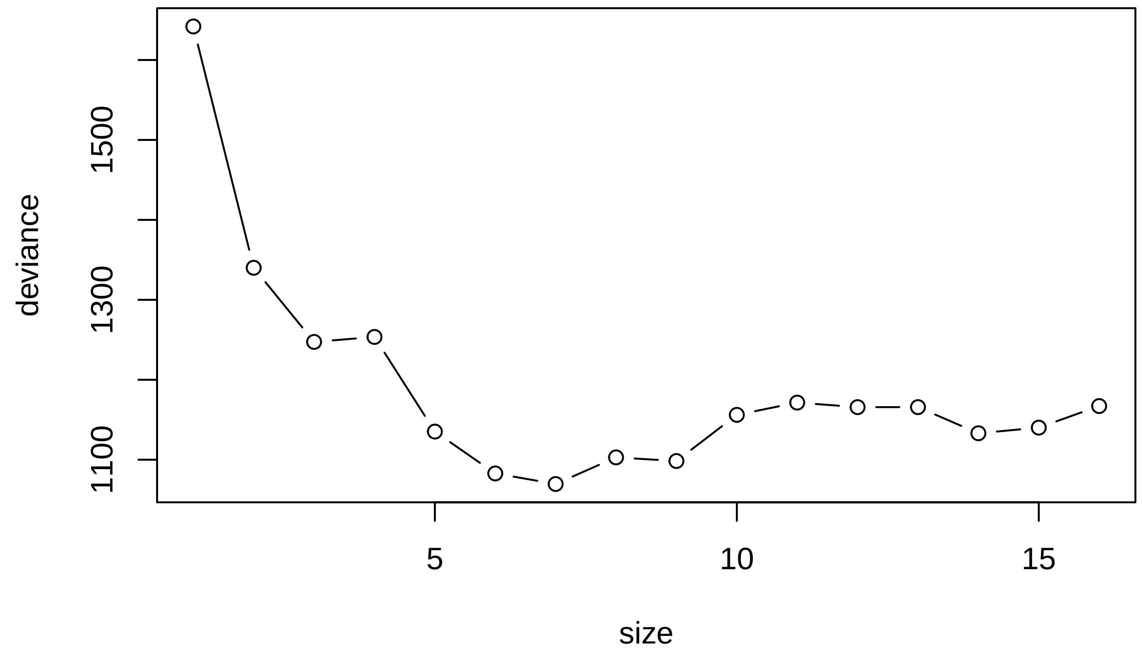



```
yhat <- predict(reg.tree, newdata = car.test)
mean((yhat - car.test$Sales)^2)
```

```
## [1] 4.801487
```

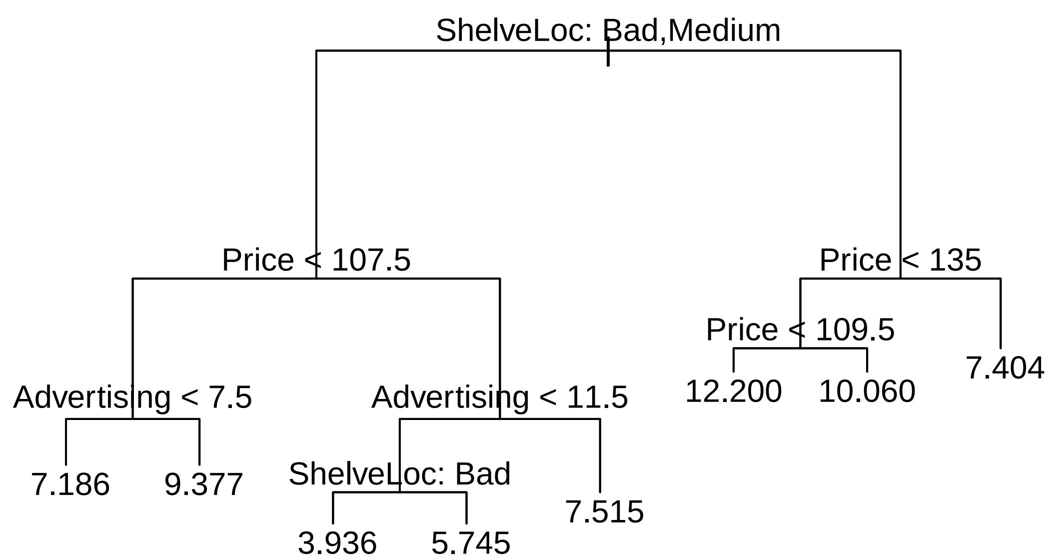
(c)

```
# Use cross-validation in order to determine the optimal level of tree complexity.
set.seed(679)

cv.car <- cv.tree(reg.tree)
plot(cv.car$size, cv.car$dev, xlab = "size", ylab = "deviance", type = "b")
```



From this plot, we can see the tree of size 8 is selected by cross-validation. We can prune the tree to obtain the 8-node tree.

```
prune.car <- prune.tree(reg.tree, best = 8)
plot(prune.car)
text(prune.car,pretty=0)
```




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

```
## [1] 5.003553
```

After we pruned the tree, the test error increases to 5.0.

(d)

```
# Use the bagging approach in order to analyze this data.
bag.car <- randomForest(Sales ~ . ,data = car.train, mtry = 10, importance = TRUE)
```

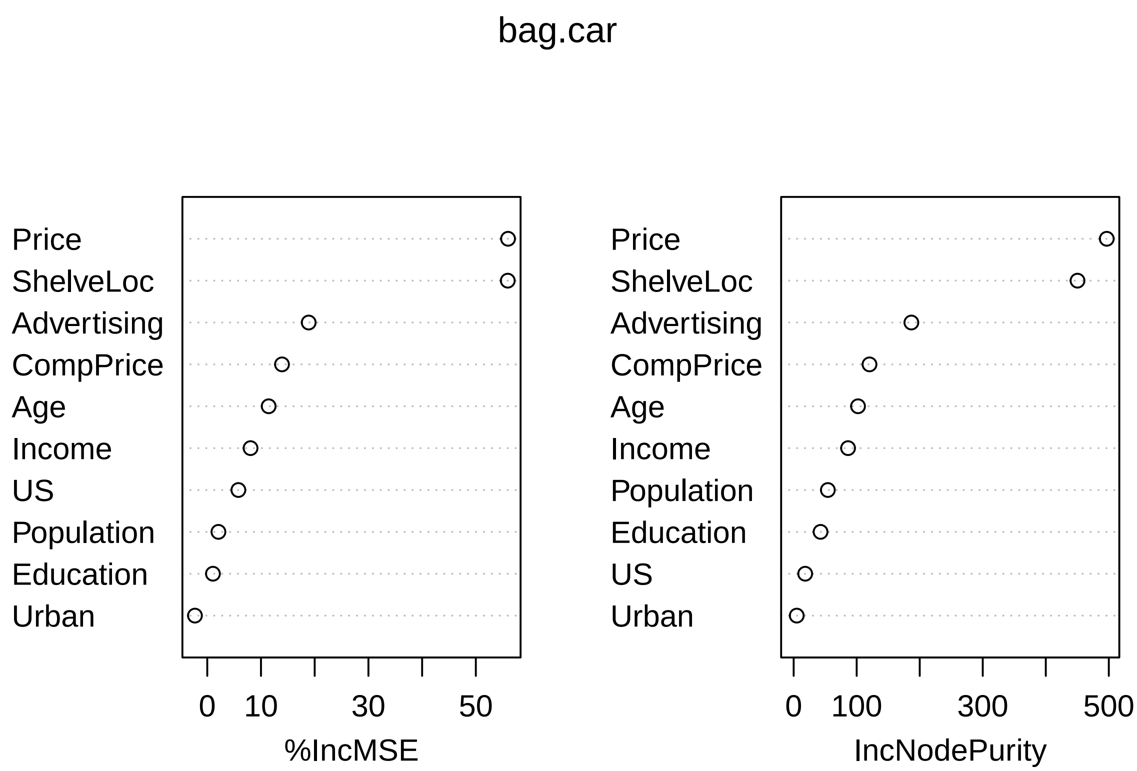
```r
yhat.bag <- predict(bag.car, newdata=car.test)
mean((yhat.bag - car.test$Sales)^2)
```

```
## [1] 2.708648
```

```r
importance(bag.car)
```

```
##               %IncMSE IncNodePurity
## CompPrice   13.913590    120.338483
## Income       8.056700     86.323205
## Advertising 18.886442    186.747986
## Population   2.081533     54.260132
## Price       55.997739    496.766529
## ShelveLoc   55.944138    450.332837
## Age         11.431590    102.122139
## Education    1.054648     42.692046
## Urban       -2.294864      4.901185
## US           5.792602     18.282633
```

```r
varImpPlot(bag.car)
```



The most important variables are the price and shelving location. And compare to the regression tree, bagging approach has less test MSE.

(e)

```r
# Use random forests to analyze this data.
set.seed(679)

rf.car <- randomForest(Sales ~ . ,data = car.train, mtry = 3, importance = TRUE)
yhat.rf <- predict(rf.car, newdata=car.test)
```

```
mean((yhat.rf - car.test$Sales)^2)
```

```
## [1] 3.212319
```

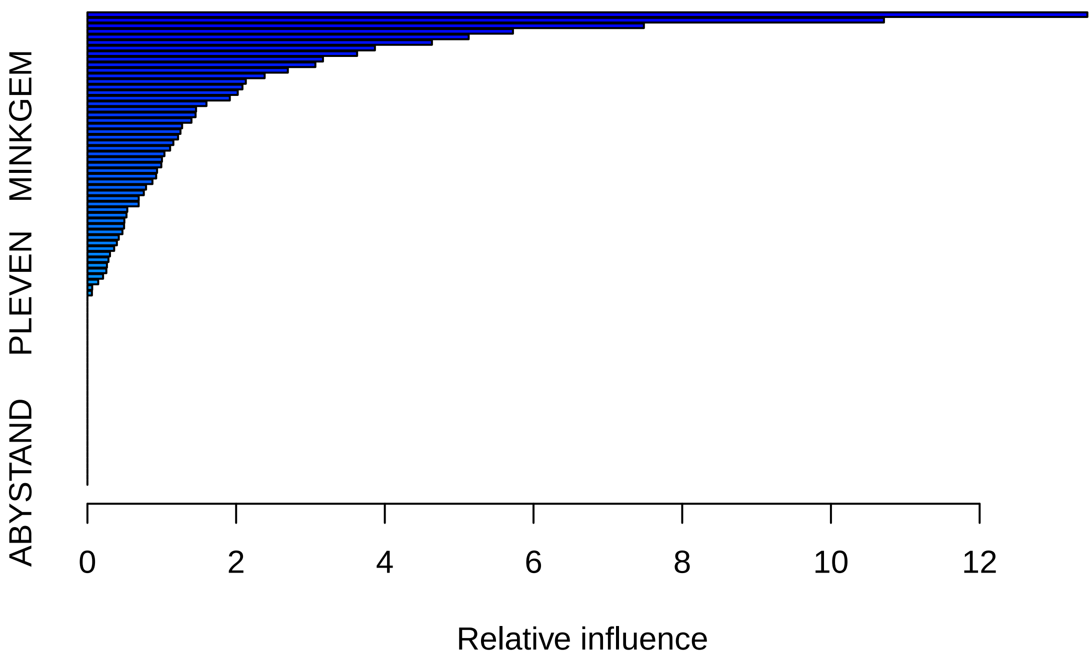
8.11

(a)

```
# Create a training set consisting of the first 1,000 observations,
# and a test set consisting of the remaining observations.
train <- 1:1000
Caravan$Purchase <- ifelse(Caravan$Purchase == "Yes", 1, 0)

Caravan.train <- Caravan[train,]
Caravan.test <- Caravan[-train,]
```

(b)

```
# Fit a boosting model to the training set with Purchase as the response and the other variables as pre
set.seed(679)
boost.caravan <- gbm(Purchase ~ ., data = Caravan.train, distribution = "gaussian",
                     n.trees = 1000, shrinkage = 0.01)
summary(boost.caravan)
```




```
##               var    rel.inf
## PPERSAUT PPERSAUT 13.44914850
## MKOOPKLA MKOOPKLA 10.71313660
## MOPLHOOG MOPLHOOG  7.48430634
## MBERMIDD MBERMIDD  5.72327771
## PBRAND     PBRAND  5.12783676
## ABRAND     ABRAND  4.63368650
## MGODGE     MGODGE  3.86667116
## MINK3045 MINK3045  3.62725681
## PWAPART   PWAPART  3.16812571
## MOSTYPE   MOSTYPE  3.06577435
```

```
## MAUT2       MAUT2  2.69569139
## MBERARBG MBERARBG  2.38390511
## MSKA         MSKA  2.13063614
## MAUT1       MAUT1  2.08495103
## MGODOV     MGODOV  2.02192090
## MSKC         MSKC  1.91535935
## MGODPR     MGODPR  1.60116140
## MSKB1       MSKB1  1.46117135
## MFGEKIND MFGEKIND  1.45481846
## MGODRK     MGODRK  1.39901590
## MINKGEM   MINKGEM  1.27362510
## APERSAUT APERSAUT  1.25134333
## PBYSTAND PBYSTAND  1.21918784
## MFWEKIND MFWEKIND  1.15597477
## MHHUUR     MHHUUR  1.11237104
## MOPLMIDD MOPLMIDD  1.03593195
## MINK4575 MINK4575  1.00266411
## MRELGE     MRELGE  0.99405233
## MINK7512 MINK7512  0.93684277
## MBERBOER MBERBOER  0.92449145
## MAUT0       MAUT0  0.87457840
## MOSHOOFD MOSHOOFD  0.78871528
## MSKD         MSKD  0.75868569
## MBERHOOG MBERHOOG  0.69018259
## MINKM30   MINKM30  0.68943589
## MRELOV     MRELOV  0.53607303
## MZFONDS   MZFONDS  0.52538696
## MGEMOMV   MGEMOMV  0.49673150
## MHKOOP     MHKOOP  0.49322035
## MZPART     MZPART  0.47110958
## MSKB2       MSKB2  0.42196134
## MINK123M MINK123M  0.39615582
## MGEMLEEF MGEMLEEF  0.35971388
## MFALLEEN MFALLEEN  0.30338019
## PMOTSCO   PMOTSCO  0.28284031
## MBERARBO MBERARBO  0.26247095
## MOPLLAAG MOPLLAAG  0.25452815
## MRELSA     MRELSA  0.21040861
## MBERZELF MBERZELF  0.14532534
## MAANTHUI MAANTHUI  0.06414630
## PLEVEN     PLEVEN  0.06061368
## 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
```

(c)

```
# Use the boosting model to predict the response on the test data.
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
##        0    1
##   0 4495   38
##   1  277   12
```

```
logit.caravan <- glm(Purchase ~ ., data = Caravan.train, family = "binomial")
probs.test2 <- predict(logit.caravan, Caravan.test, type = "response")

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

```
##    pred.test2
##        0    1
##   0 4495   38
##   1  277   12
```