MATH 4322 Homework 5

Cathy Poliak

Spring 2023

Instructions

1. Due date: April 4, 2023
2. Answer the questions fully for full credit.
3. Scan or Type your answers and submit only one file. (If you submit several files only the recent one uploaded will be graded).
4. Preferably save your file as PDF before uploading.
5. Submit in Canvas.
6. These questions are from *An Introduction to Statistical Learning with Applications* in R by James, et. al., chapter 8.
7. The information in the gray boxes are R code that you can use to answer the questions.

Problem 1

The questions relate to the following plots:

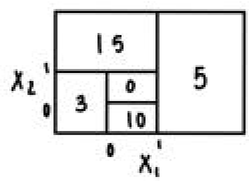
a) Sketch the tree corresponding to the partition of the predictor space illustrated on the left-hand plot. The numbers inside the boxes indicate the mean of Y within each region.

b) Create a diagram similar to the left-hand plot using the tree illustrated in the right-hand plot. You should divide up the predictor space inot the correct regions, and indicate the mean for each region.

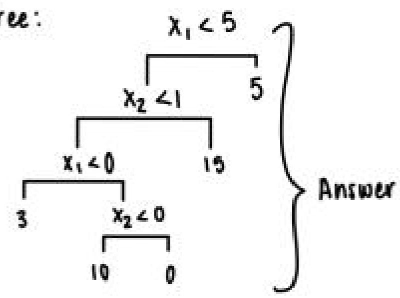
This is the tree from part a and the diagram for part b.

Problem #1

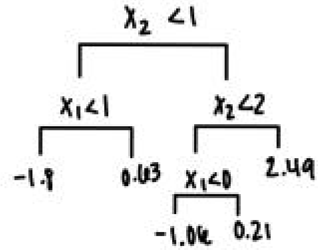
a) Sketch the tree corresponding to the partition of the predictor space illustrated on the left-hand plot



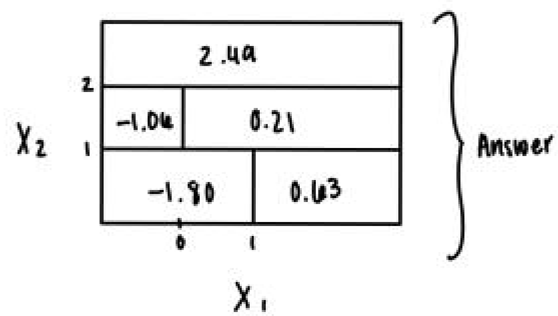
Tree:



b) Create a diagram similar to the left-hand plot using the tree illustrated in the right-hand plot.



Part B Diagram:



Problem 2

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of $P(\text{Class is Red}|X)$:

$$0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, \text{ and } 0.75.$$

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

There are two common ways to combine the results together into a single class prediction. One of the approaches could be the majority vote approach and the other could be the average probability appraoch.

The Majority Vote Approach

First we will take our values = {0.1,0.15, 0.2, 0.2,0.55, 0.6, 0.6, 0.65, 0.7, 0.75} Then the final classification of these values will be red because more cases are in favor of red rather than green. The values = {G,G,G,G,R,R,R,R,R,R}

The Average Probability Approach:

```
mean(c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75))
```

```
## [1] 0.45
```

The average or the mean of these values is 0.45 therefore it shows that it is in favor of green.

Problem 3

There are two main steps following the regression tree algorithm. First we take a set of possible values, predictor space, which we will divide in N distinct and non-overlapping rectangular regions. For each value we will create a prediction, test, which will equal to a mean of the response values which we are training. Then we will use a top-down appraoch to divide the values using a method known as recursive binary splitting. A decision tree can tehn be created from this when the decision nodes in the tree contain a test that corresponds to a terminal node, this was shown in the diagram from problem 1.

Problem 4

This problem involves the `OJ` data set which is part of the `ISLR2` package.

a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
suppressMessages(library(ISLR))
suppressMessages(library(tree))

oj = OJ
names(oj) = tolower(names(oj))
```

```r
set.seed(1000)
index = sample(1:nrow(oj), 800)
train = oj[index,]
test = oj[-index,]
purchase.test = oj$purchase[-index]
```

b) Fit a tree to the training data, with `Purchase` as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```r
tree.oj = tree(purchase ~ ., train)
summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = purchase ~ ., data = train)
## Variables actually used in tree construction:
## [1] "loyalch"     "pricediff"   "salepricemm"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7486 = 592.9 / 792
## Misclassification error rate: 0.16 = 128 / 800
```

training error rate = 16% Number of Terminal Nodes = 8

c) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

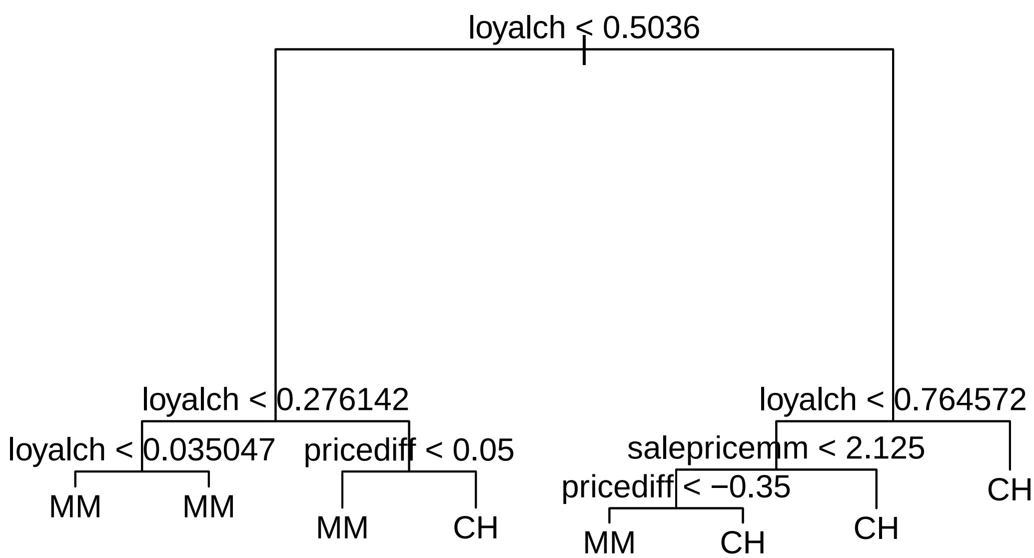
```r
tree.oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1066.00 CH ( 0.61500 0.38500 )
##    2) loyalch < 0.5036 353  422.60 MM ( 0.28612 0.71388 )
##      4) loyalch < 0.276142 170  131.00 MM ( 0.12941 0.87059 )
##        8) loyalch < 0.035047 57   10.07 MM ( 0.01754 0.98246 ) *
##        9) loyalch > 0.035047 113  108.50 MM ( 0.18584 0.81416 ) *
##      5) loyalch > 0.276142 183  250.30 MM ( 0.43169 0.56831 )
##       10) pricediff < 0.05 78   79.16 MM ( 0.20513 0.79487 ) *
##       11) pricediff > 0.05 105  141.30 CH ( 0.60000 0.40000 ) *
##    3) loyalch > 0.5036 447  337.30 CH ( 0.87472 0.12528 )
##      6) loyalch < 0.764572 187  206.40 CH ( 0.75936 0.24064 )
##       12) salepricemm < 2.125 120  156.60 CH ( 0.64167 0.35833 )
##         24) pricediff < -0.35 16   17.99 MM ( 0.25000 0.75000 ) *
##         25) pricediff > -0.35 104  126.70 CH ( 0.70192 0.29808 ) *
##       13) salepricemm > 2.125 67   17.99 CH ( 0.97015 0.02985 ) *
##      7) loyalch > 0.764572 260   91.11 CH ( 0.95769 0.04231 ) *
```

Interpretation of the terminal node 7) loyalch > 0.764572 Number of Observations in the branch = 260 Deviance = 91.11 Overall Prediction = "CH" Fraction of Observations in this branch taking on the values of "CH" and "MN" = (0.95769 0.04231)

d) Create a plot of the tree, and interpret the results.

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



The most important indicator of purcahse seems to be loaylch, its value is less than 0.5036, it seems that its going to be classified as CH since values >= .50 get classified as CH.

e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```
pred = predict(tree.oj, test, type = "class")
table(pred, purchase.test)
```

```
##     purchase.test
## pred  CH  MM
##   CH 150  38
##   MM  11  71
```

```
test.error = round(mean(pred != purchase.test)*100, 2)
test.error
```

```
## [1] 18.15
```

Test Error Rate = 18.15

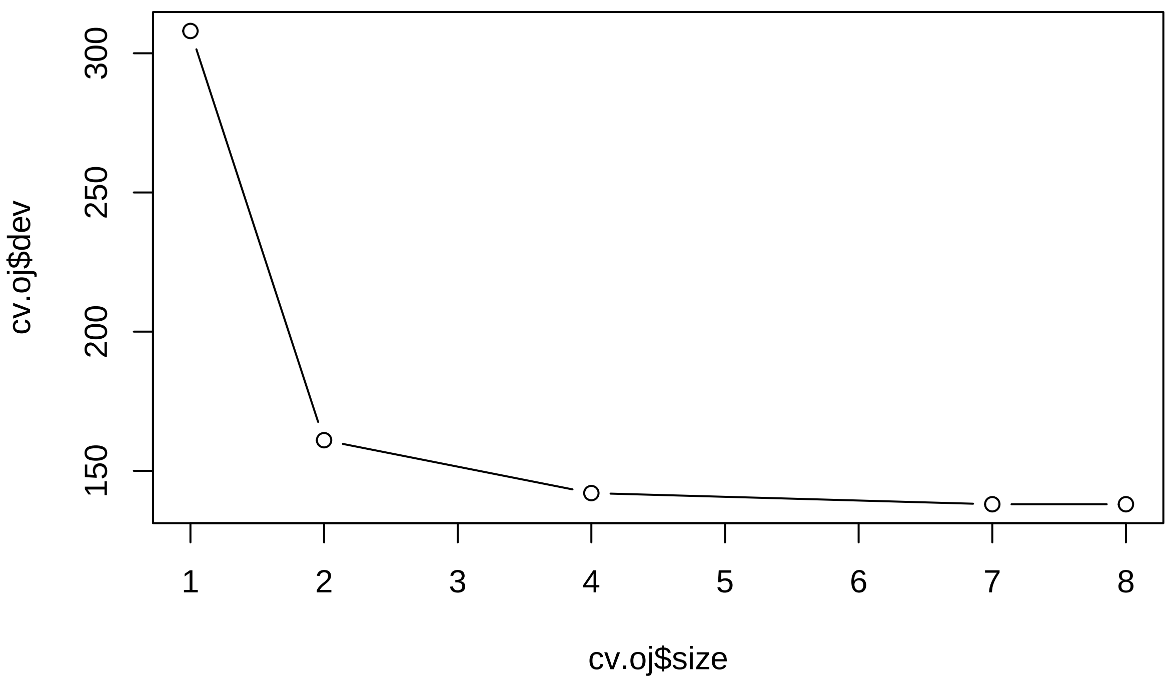
f) Apply the `cv.tree()` function to the training set in order to determine the optimal tree size.

```
set.seed(1000)
cv.oj = cv.tree(tree.oj, FUN = prune.misclass)
cv.oj
```

```
## $size
## [1] 8 7 4 2 1
##
## $dev
## [1] 138 138 142 161 308
##
## $k
## [1]       -Inf   0.000000   2.666667  10.500000 151.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

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



h) Which tree size corresponds to the lowest cross-validated classification error rate?

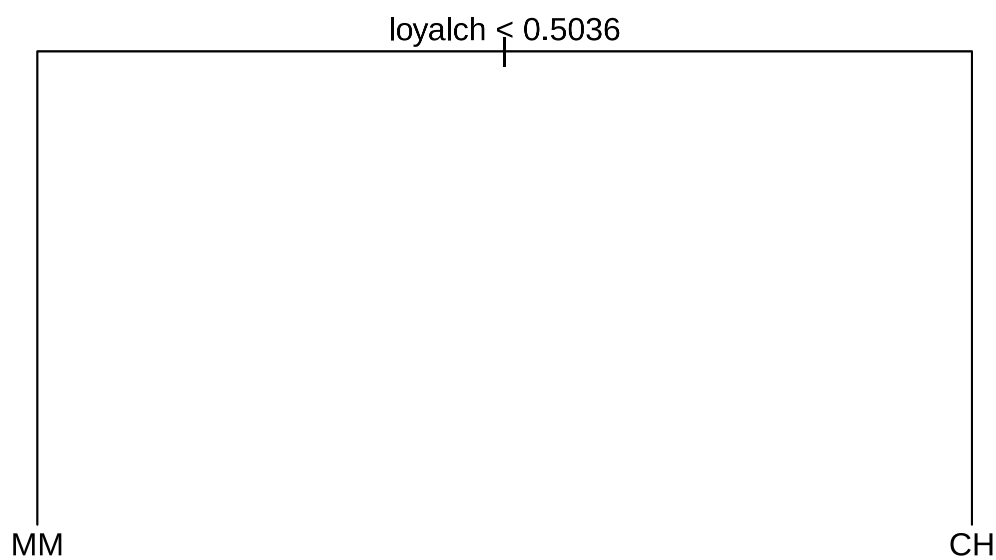
The tree size that corresponds to the lowest cross-validation rate is the tree size of 4.

i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

```
prune.oj = prune.misclass(tree.oj, best=2)
summary(prune.oj)
```

```
##
## Classification tree:
## snip.tree(tree = tree.oj, nodes = 3:2)
## Variables actually used in tree construction:
## [1] "loyalch"
## Number of terminal nodes:  2
## Residual mean deviance:  0.9523 = 760 / 798
## Misclassification error rate: 0.1962 = 157 / 800
```

```
plot(prune.oj)
text(prune.oj, pretty=0)
```




```
prune.train.error =  19.62
prune.train.accuracy = 100-prune.train.error
```

The training error rate = 19.62

j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

The training error rate of the pruned tree is 19.62% and the unpruned tree is 18.15%. The training error rate for the pruned tree is higher because by pruning we reduce the flexibility in the model. Since the training error rate has risen due to a rise in the bias.

k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

(incomplete)

Problem 5

We will use the `Carseats` data set that is in the `ISLR` package to see to predict `Sales` using regression trees and related approaches.

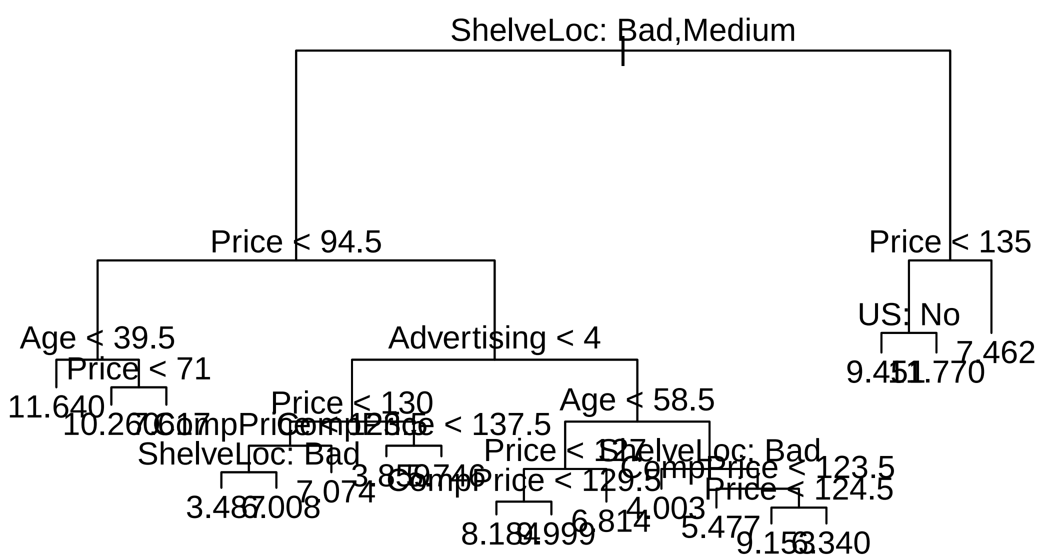
a) Split the data set into a training set and a test set.

```
library(tree)
library(ISLR)
attach(Carseats)

set.seed(1)
train <- sample(1:nrow(Carseats), nrow(Carseats)/2)
car.train <- Carseats[train, ]
car.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?

```
car.tree <- tree(Sales ~ ., data = car.train)
plot(car.tree)
text(car.tree, pretty=0)
```



```
summary(car.tree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = car.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "CompPrice"
## [6] "US"
## Number of terminal nodes:  18
## Residual mean deviance:  2.167 = 394.3 / 182
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.88200 -0.88200 -0.08712  0.00000  0.89590  4.09900
```

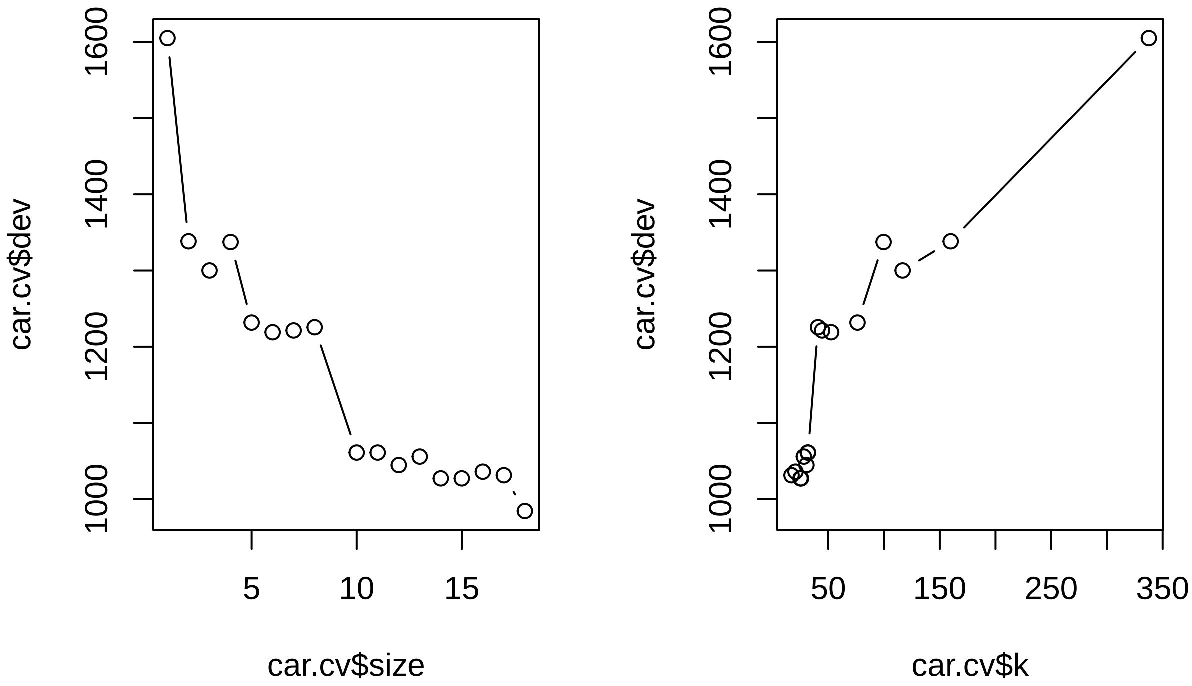
```
car.pred <- predict(car.tree, newdata = car.test)
mean((car.pred - car.test$Sales)^2)
```

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

The MSE is 4.922039.

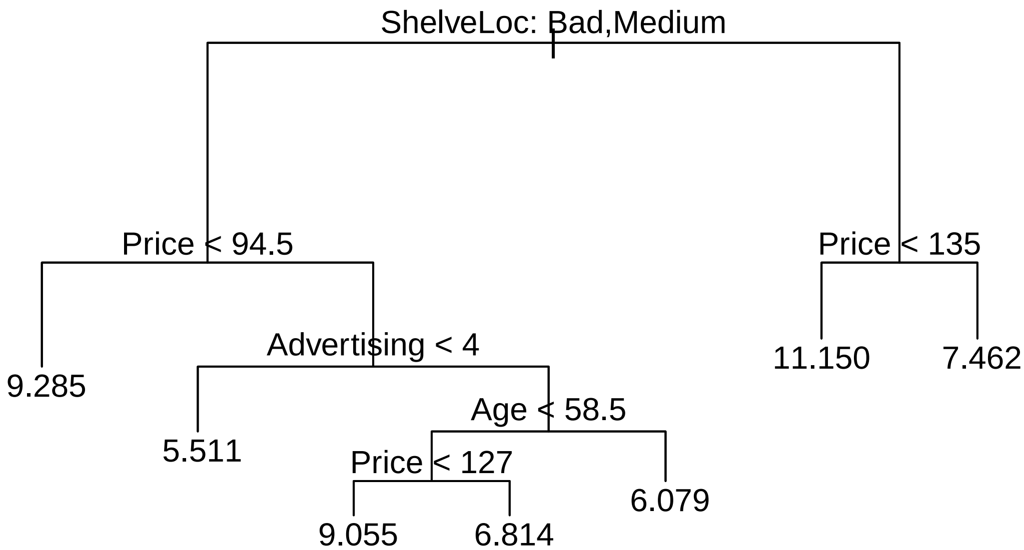
c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
set.seed(1)
car.cv <- cv.tree(car.tree)
par(mfrow = c(1, 2))
plot(car.cv$size, car.cv$dev, type = "b")
plot(car.cv$k, car.cv$dev, type = "b")
```




```
par(mfrow = c(1,1))

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




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

```
## [1] 4.861001
```

The best size for the three seems to be 7, but in this

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.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.3
```

```
## randomForest 4.7-1.1
```

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

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

```
##
## Call:
##  randomForest(formula = Sales ~ ., data = Carseats, mtry = 10,      importance = TRUE, subset = trai
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.889221
##                     % Var explained: 63.26
```

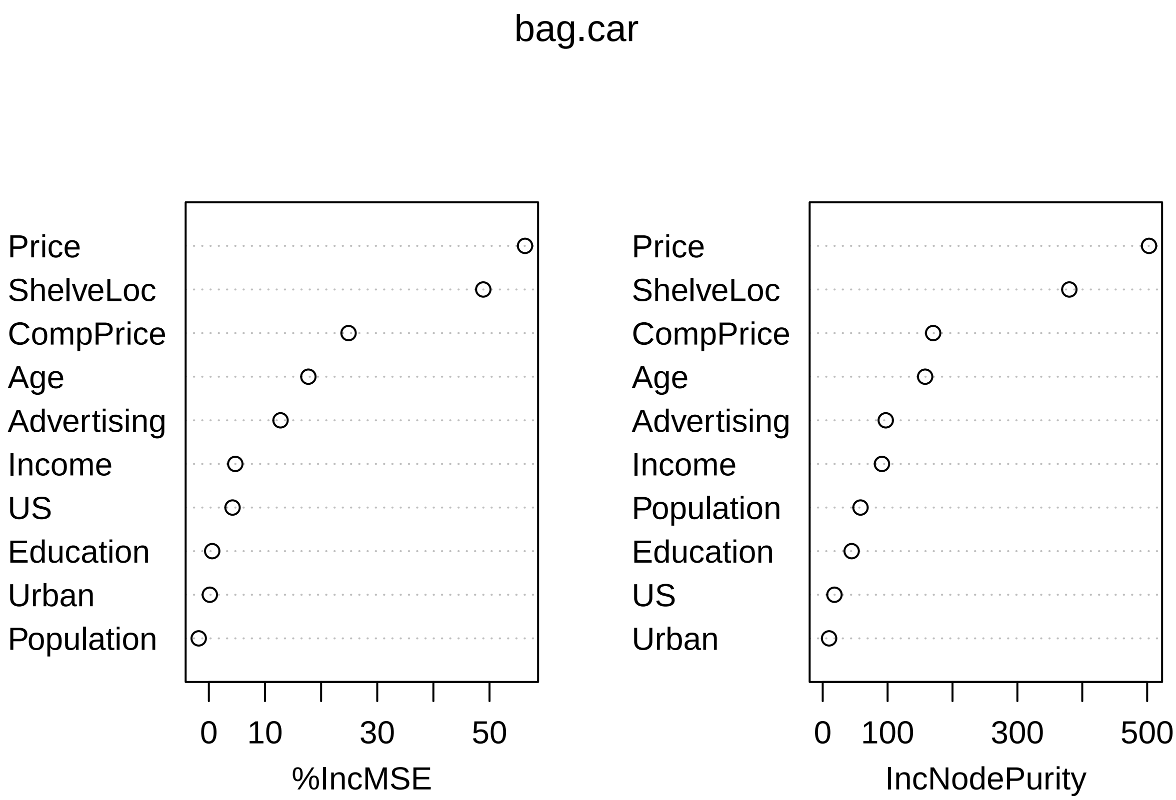
```
predict.bag <- predict(bag.car, newdata = car.test)
mean((predict.bag - car.test$Sales)^2)
```

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

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

```
##                %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
```

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



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

(incomplete) ## Problem 6

This question uses the `Caravan` data set in the `ISLR2` package.

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

```
library(ISLR)
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 predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.2.3
```

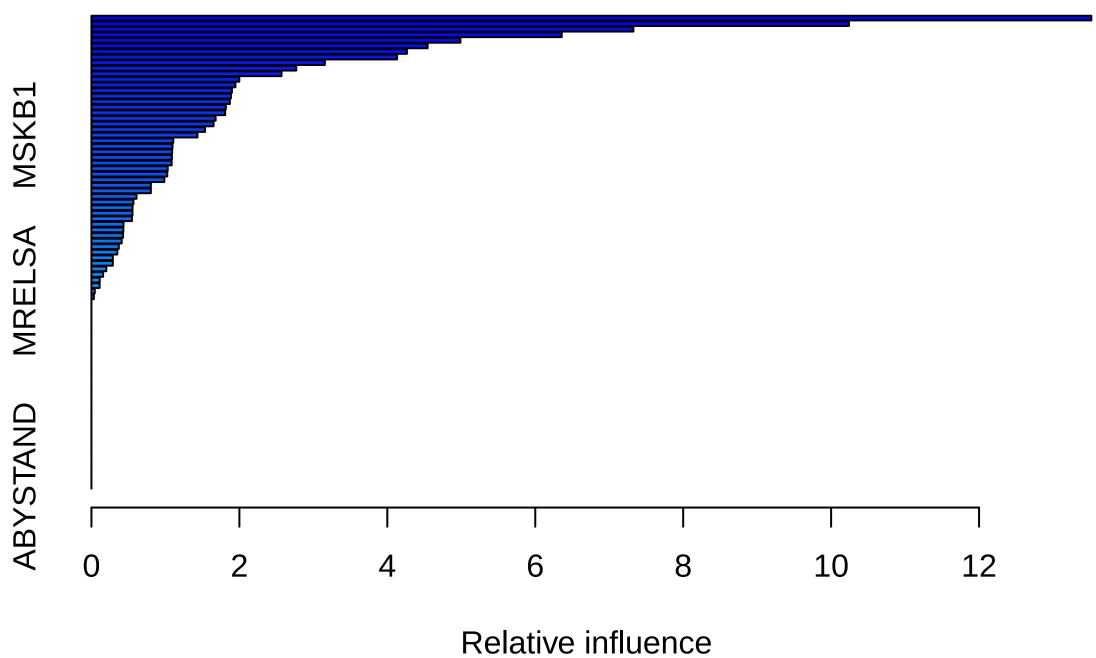
```
## Loaded gbm 2.1.8.1
```

```r
set.seed(1)
boost.caravan <- gbm(Purchase ~ ., data = Caravan.train, distribution = "gaussian", n.trees = 1000, shri
```

```
## 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.caravan)
```




```
##               var     rel.inf
## PPERSAUT PPERSAUT 13.51824557
## MKOOPKLA MKOOPKLA 10.24062778
## MOPLHOOG MOPLHOOG  7.32689780
## MBERMIDD MBERMIDD  6.35820558
## PBRAND     PBRAND  4.98826360
## ABRAND     ABRAND  4.54504653
## MGODGE     MGODGE  4.26496875
## MINK3045 MINK3045  4.13253907
## PWAPART   PWAPART  3.15612877
## MAUT1       MAUT1  2.76929763
## MOSTYPE   MOSTYPE  2.56937935
## MAUT2       MAUT2  1.99879666
```

```
## MSKA         MSKA  1.94618539
## MBERARBG MBERARBG  1.89917331
## PBYSTAND PBYSTAND  1.88591514
## MINKGEM   MINKGEM  1.87131472
## MGODOV     MGODOV  1.81673309
## MGODPR     MGODPR  1.80814745
## MFWEKIND MFWEKIND  1.67884570
## MSKC         MSKC  1.65075962
## MBERHOOG MBERHOOG  1.53559951
## MSKB1       MSKB1  1.43339514
## MOPLMIDD MOPLMIDD  1.10617074
## MHHUUR     MHHUUR  1.09608784
## MRELGE     MRELGE  1.09039794
## MINK7512 MINK7512  1.08772012
## MZFONDS   MZFONDS  1.08427551
## MGODRK     MGODRK  1.03126657
## MINK4575 MINK4575  1.02492795
## MZPART     MZPART  0.98536712
## MRELOV     MRELOV  0.80356854
## MFGEKIND MFGEKIND  0.80335689
## MBERARBO MBERARBO  0.60909852
## APERSAUT APERSAUT  0.56707821
## MGEMOMV   MGEMOMV  0.55589456
## MOSHOOFD MOSHOOFD  0.55498375
## MAUT0       MAUT0  0.54748481
## PMOTSCO   PMOTSCO  0.43362597
## MSKB2       MSKB2  0.43075446
## MSKD         MSKD  0.42751490
## MINK123M MINK123M  0.40920707
## MINKM30   MINKM30  0.36996576
## MHKOOP     MHKOOP  0.34941518
## MBERBOER MBERBOER  0.28967068
## MFALLEEN MFALLEEN  0.28877552
## MGEMLEEF MGEMLEEF  0.20084195
## MOPLLAAG MOPLLAAG  0.15750616
## MBERZELF MBERZELF  0.11203381
## PLEVEN     PLEVEN  0.11030994
## MRELSA     MRELSA  0.04500507
## MAANTHUI MAANTHUI  0.03322830
## 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
```

"PPERSUAT" and "MKOOPKLA" are the two most important variables.

(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 %. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one?

The fraction of predicted people to make a purchase and will make one again is 0.2156863.

```r
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 4493   40
##   1  278   11
```