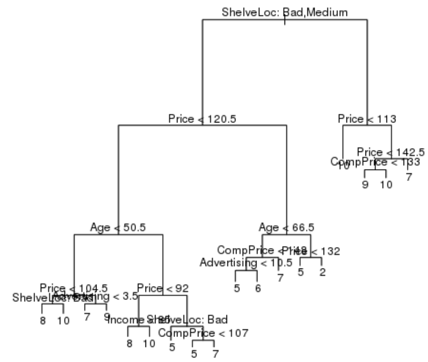
5)
Majority approach

```
> p = c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)>
sum(p >= 0.5) > sum(p < 0.5)
[1] TRUE
```

The number of red predictions is greater than the number of green predictions based on a 50% threshold, thus RED.

Average approach
```
> mean(p)
[1] 0.45
```

The average of the probabilities is less than the 50% threshold, thus GREEN.

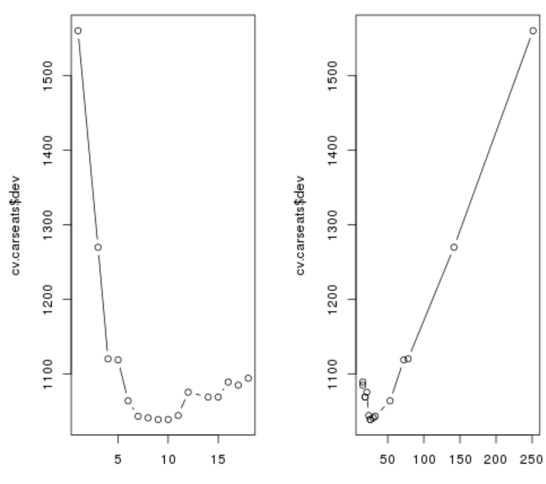


8)
a)
```
library(ISLR)
attach(Carseats)
set.seed(1)

train = sample(dim(Carseats)[1], dim(Carseats)[1]/2)
Carseats.train = Carseats[train, ]
Carseats.test = Carseats[-train, ]
```

b)
```
library(tree)
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 / 182
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -4.260  -1.040   0.102   0.000   0.930   3.910
plot(tree.carseats)
text(tree.carseats, pretty = 0)
pred.carseats = predict(tree.carseats, Carseats.test)
mean((Carseats.test$Sales – pred.carseats)^2)
## [1] 4.149
```
The test MSE is about 4.15



c)
```
cv.carseats = cv.tree(tree.carseats, FUN = prune.tree)
par(mfrow = c(1, 2))
plot(cv.carseats$size, cv.carseats$dev, type = "b")
plot(cv.carseats$k, cv.carseats$dev, type = "b")

# Best size = 9
pruned.carseats = prune.tree(tree.carseats, best = 9)
par(mfrow = c(1, 1))
plot(pruned.carseats)
text(pruned.carseats, pretty = 0)

pred.pruned = predict(pruned.carseats, Carseats.test)
mean((Carseats.test$Sales – pred.pruned)^2)
```

```
## [1] 4.993
```

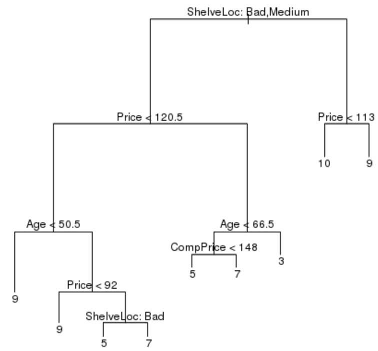
Pruning the tree in this case increases the test MSE to 4.99

d)

```
library(randomForest)
bag.carseats = randomForest(Sales ~ ., data = Carseats.train,
mtry = 10, ntree = 500, importance = T)
bag.pred = predict(bag.carseats, Carseats.test)
mean((Carseats.test$Sales – bag.pred)^2)

## [1] 2.586

importance(bag.carseats)
##             %IncMSE IncNodePurity
## CompPrice   13.8790       131.095
## Income       5.6042        77.033
## Advertising 14.1720       129.218
## Population   0.6071        65.196
## Price       53.6119       506.604
## ShelveLoc   44.0311       323.189
## Age         20.1751       189.269
## Education    1.4244        41.811
## Urban       -1.9640         8.124
## US           5.6989        14.307
```



Bagging improves the test MSE to 2.58. We also see that PricePrice, ShelveLocc and Age are three most important predictors of Sale.

e)

```
rf.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry = 5, ntree = 500,
    importance = T)
rf.pred = predict(rf.carseats, Carseats.test)
mean((Carseats.test$Sales – rf.pred)^2)

## [1] 2.87

importance(rf.carseats)
##             %IncMSE IncNodePurity
## CompPrice   11.2746        126.64
## Income       4.4397        101.63
## Advertising 12.9346        137.96
## Population   0.2725         78.78
## Price       49.2418        449.52
## ShelveLoc   38.8406        283.46
## Age         19.1329        195.14
## Education    1.9818         54.26
## Urban       -2.2083         11.35
## US           6.6487         26.71
```

In this case, random forest worsens the MSE on test set to 2.87. Changing mm varies test MSE between 2.6 to 3. We again see that Price, ShelveLoc and Age are three most important predictors of Sale.

9)

a)

```
library(ISLR)
attach(OJ)
set.seed(1013)

train = sample(dim(OJ)[1], 800)
OJ.train = OJ[train, ]
OJ.test = OJ[-train, ]
```

b)

```
library(tree)
```

```
oj.tree = tree(Purchase ~ ., data = OJ.train)
summary(oj.tree)
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.752 = 596 / 793
## Misclassification error rate: 0.155 = 124 / 800
```

The tree only uses two variables: LoyalCH and PriceDiff. It has 7 terminal nodes. Training error rate (misclassification error) for the tree is 0.155

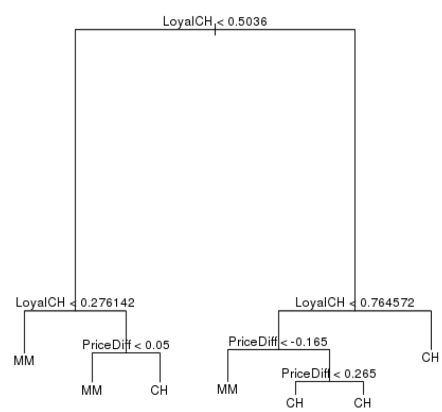
c)

```
oj.tree
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1000 CH ( 0.60 0.40 )
##    2) LoyalCH < 0.5036 359  400 MM ( 0.28 0.72 )
##      4) LoyalCH < 0.276142 170  100 MM ( 0.11 0.89 ) *
##      5) LoyalCH > 0.276142 189  300 MM ( 0.42 0.58 )
##       10) PriceDiff < 0.05 79   80 MM ( 0.19 0.81 ) *
##       11) PriceDiff > 0.05 110  100 CH ( 0.59 0.41 ) *
##    3) LoyalCH > 0.5036 441  300 CH ( 0.87 0.13 )
##      6) LoyalCH < 0.764572 186  200 CH ( 0.75 0.25 )
##       12) PriceDiff < -0.165 29   30 MM ( 0.28 0.72 ) *
##       13) PriceDiff > -0.165 157  100 CH ( 0.83 0.17 )
##         26) PriceDiff < 0.265 82  100 CH ( 0.73 0.27 ) *
##         27) PriceDiff > 0.265 75   30 CH ( 0.95 0.05 ) *
##      7) LoyalCH > 0.764572 255   90 CH ( 0.96 0.04 ) *
```

Let's pick terminal node labeled “10)”. The splitting variable at this node is PriceDiff. The splitting value of this node is 0.05. There are 79 points in the subtree below this node. The deviance for all points contained in region below this node is 80. A * in the line denotes that this is in fact a terminal node. The prediction at this node is Sales = MM. About 19% points in this node have CH as value of Sales. Remaining 81% points have MM as value of Sales.

d)

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



LoyalCH is the most important variable of the tree, in fact top 3 nodes contain LoyalCH. If LoyalCH<0.27, the tree predicts MM. If LoyalCH>0.76, the tree predicts CHCH. For intermediate values of LoyalCH, the decision also depends on the value of PriceDiff.

e)

```
oj.pred = predict(oj.tree, OJ.test, type = "class")
table(OJ.test$Purchase, oj.pred)

##     oj.pred
##       CH  MM
##   CH 152  19
##   MM  32  67
```

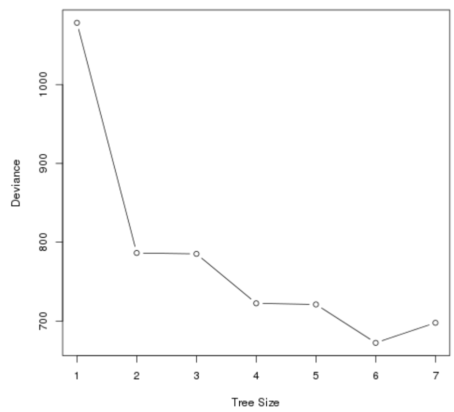
f)

```
cv.oj = cv.tree(oj.tree, FUN = prune.tree)
```

g)

```
plot(cv.oj$size, cv.oj$dev, type = "b", xlab = "Tree Size", ylab = "Deviance")
```

h)Size of 6 gives lowest cross-validation error.



i)

```
oj.pruned = prune.tree(oj.tree, best = 6)
```

j)

```
summary(oj.pruned)

##
## Classification tree:
## snip.tree(tree = oj.tree, nodes = 13L)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  6
## Residual mean deviance:  0.769 = 610 / 794
## Misclassification error rate: 0.155 = 124 / 800
```

Misclassification error of pruned tree is exactly same as that of original tree — 0.155.

k)

```
pred.unpruned = predict(oj.tree, OJ.test, type = "class")
misclass.unpruned = sum(OJ.test$Purchase != pred.unpruned)
misclass.unpruned/length(pred.unpruned)

## [1] 0.1889
pred.pruned = predict(oj.pruned, OJ.test, type = "class")
misclass.pruned = sum(OJ.test$Purchase != pred.pruned)
misclass.pruned/length(pred.pruned)

## [1] 0.1889
```

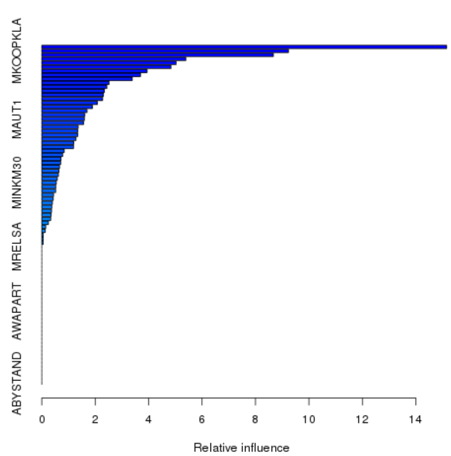
Pruned and unpruned trees have same test error rate of 0.189.

11)
a)

```
library(ISLR)
train = 1:1000
Caravan$Purchase = ifelse(Caravan$Purchase == "Yes", 1, 0)
Caravan.train = Caravan[train, ]

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

b)




```
library(gbm)
## Loading required package: survival
## Loading required package: splines
## Loading required package: lattice
## Loading required package: parallel
## Loaded gbm 2.1
set.seed(342)
boost.caravan = gbm(Purchase ~ ., data =
Caravan.train, n.trees = 1000, shrinkage = 0.01,
    distribution = "bernoulli")
## Warning: variable 50: PVRAAUT has no
variation.
## Warning: variable 71: AVRAAUT has no
variation.
summary(boost.caravan)

##              var  rel.inf
## PPERSAUT PPERSAUT 15.15534
## MKOOPKLA MKOOPKLA  9.23500
## MOPLHOOG MOPLHOOG  8.67017
## MBERMIDD MBERMIDD  5.39404
```

```
## MGODGE     MGODGE  5.03048
....
## AVRAAUT   AVRAAUT  0.00000
## AAANHANG AAANHANG  0.00000
## ATRACTOR ATRACTOR  0.00000
## AWERKT     AWERKT  0.00000
## ABROM       ABROM  0.00000
## ALEVEN     ALEVEN  0.00000
## APERSONG APERSONG  0.00000
## AGEZONG   AGEZONG  0.00000
## AWAOREG   AWAOREG  0.00000
## AZEILPL   AZEILPL  0.00000
## APLEZIER APLEZIER  0.00000
## AFIETS     AFIETS  0.00000
## AINBOED   AINBOED  0.00000
## ABYSTAND ABYSTAND  0.00000
```

PPERSAUT, MKOOPKLA and MOPLHOOG are three most important variables in that order.

c)

```
boost.prob = predict(boost.caravan, Caravan.test, n.trees = 1000, type = "response")
boost.pred = ifelse(boost.prob > 0.2, 1, 0)
table(Caravan.test$Purchase, boost.pred)
##    boost.pred
##        0    1
##   0 4396  137
##   1  255   34
34/(137 + 34)
## [1] 0.1988
```

About 20% of people predicted to make purchase actually end up making one.

```
lm.caravan = glm(Purchase ~ ., data = Caravan.train, family = binomial)
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
lm.prob = predict(lm.caravan, Caravan.test, type = "response")
## Warning: prediction from a rank-deficient fit may be misleading
lm.pred = ifelse(lm.prob > 0.2, 1, 0)
table(Caravan.test$Purchase, lm.pred)
##    lm.pred
##        0    1
##   0 4183  350
##   1  231   58
58/(350 + 58)
## [1] 0.1422
```

About 14% of people predicted to make purchase using logistic regression actually end up making one. This is lower than boosting.