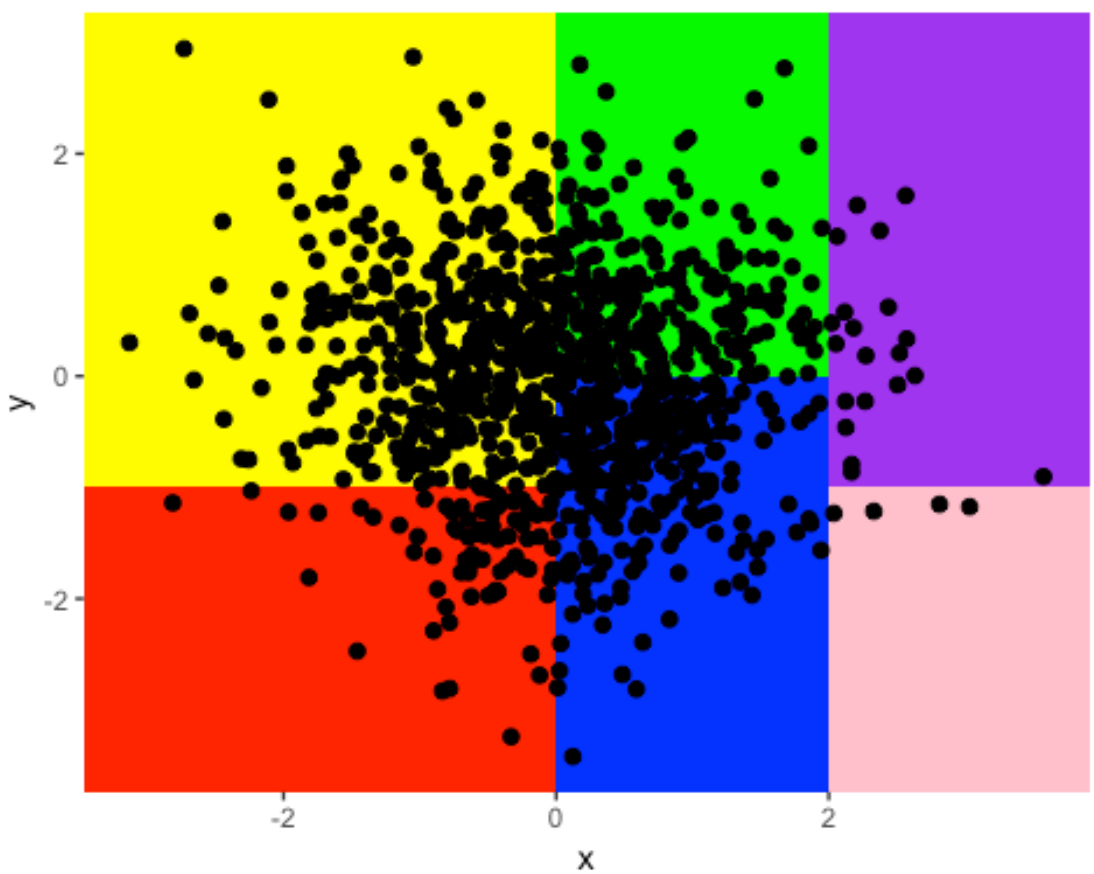
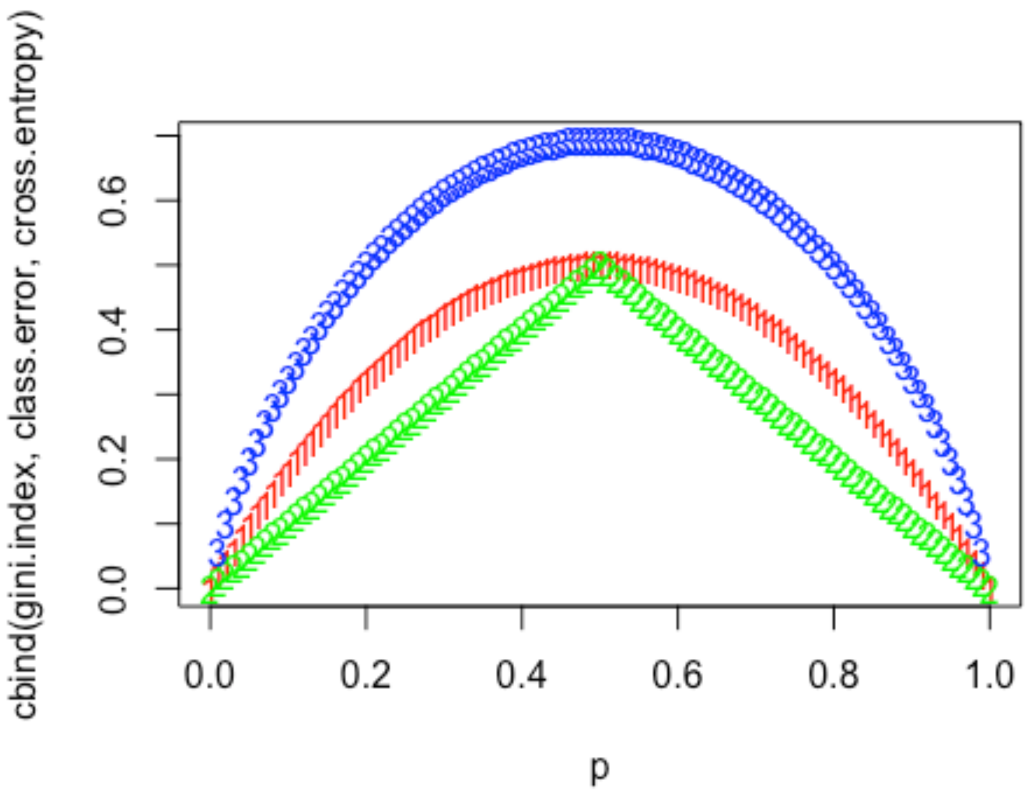
# Assignment 4 Patricia Maya

```
#Ex1 Page 332
library(ggplot2)
df=data.frame(x = rnorm(1000), y = rnorm(1000))
ggplot(df) +
  geom_rect(xmin = -Inf, xmax = 0,   ymin = -Inf, ymax = -1,   fill = "red")+
  geom_rect(xmin = 0,    xmax = 2,   ymin = -Inf, ymax = 0,   fill = "blue")
+
  geom_rect(xmin = 0,    xmax = 2,   ymin = 0,    ymax = Inf, fill = "green")
+
  geom_rect(xmin = -Inf, xmax = 0,   ymin = -1,    ymax = Inf, fill =
"yellow") +
  geom_rect(xmin = 2,    xmax = 4, ymin = -1,   ymax = Inf, fill = "purple")
+
  geom_rect(xmin = 2,    xmax = 4, ymin = -Inf, ymax = -1,  fill = "pink") +
  geom_point(aes(x, y), size = 2)
```

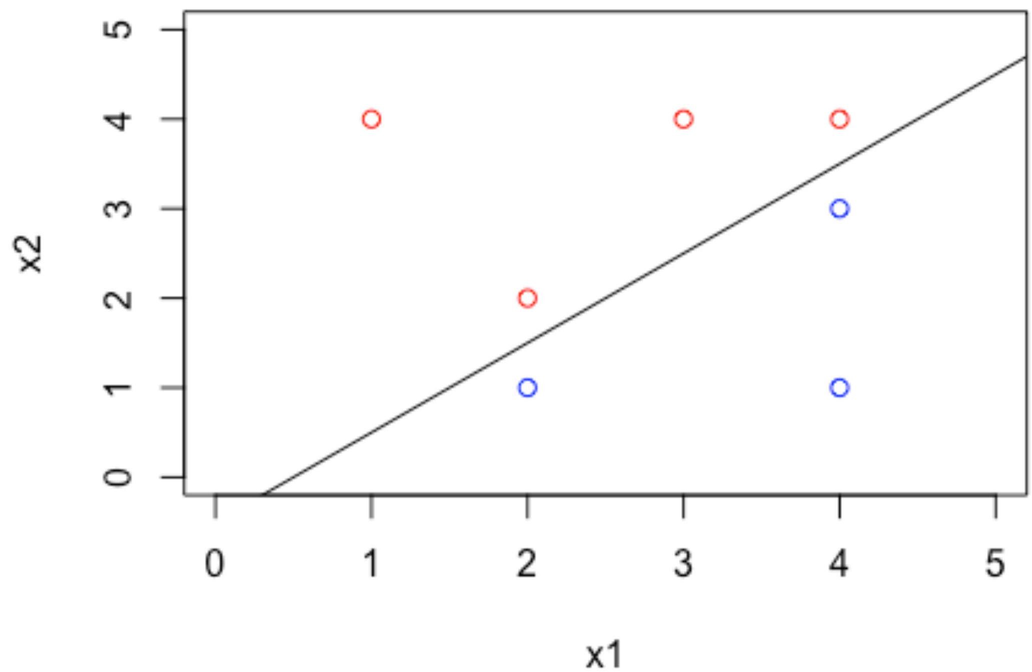



```
#Ex3 Page 332
p <- seq(0, 1, 0.01)
gini.index <- 2 * p * (1 - p)
```

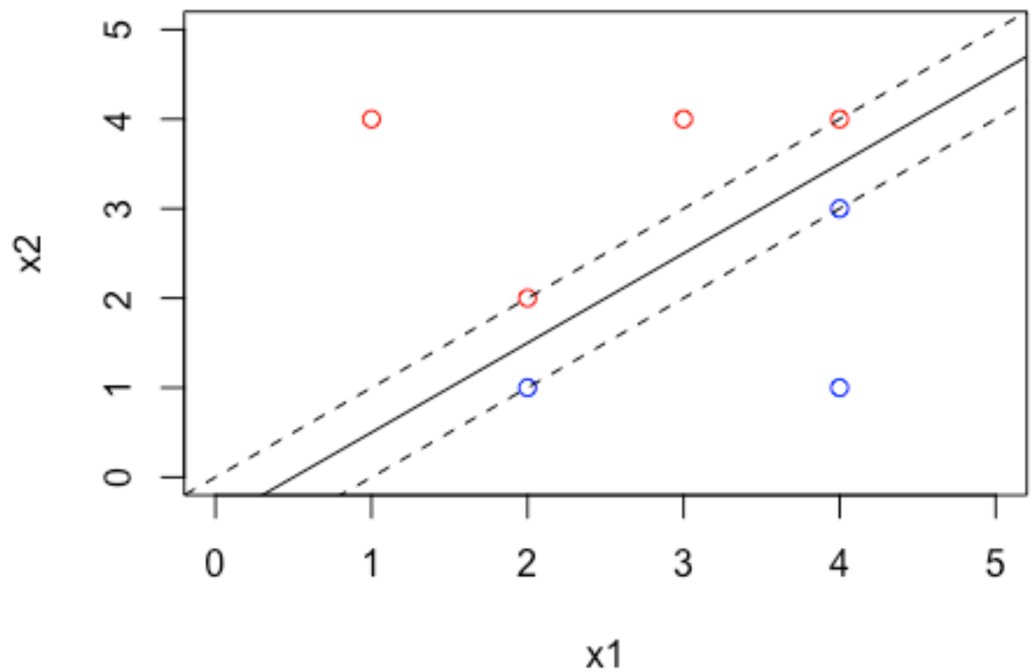
```
class.error <- 1 - pmax(p, 1 - p)
cross.entropy <- - (p * log(p) + (1 - p) * log(1 - p))
matplot(p, cbind(gini.index, class.error, cross.entropy), col = c("red",
"green", "blue"))
```



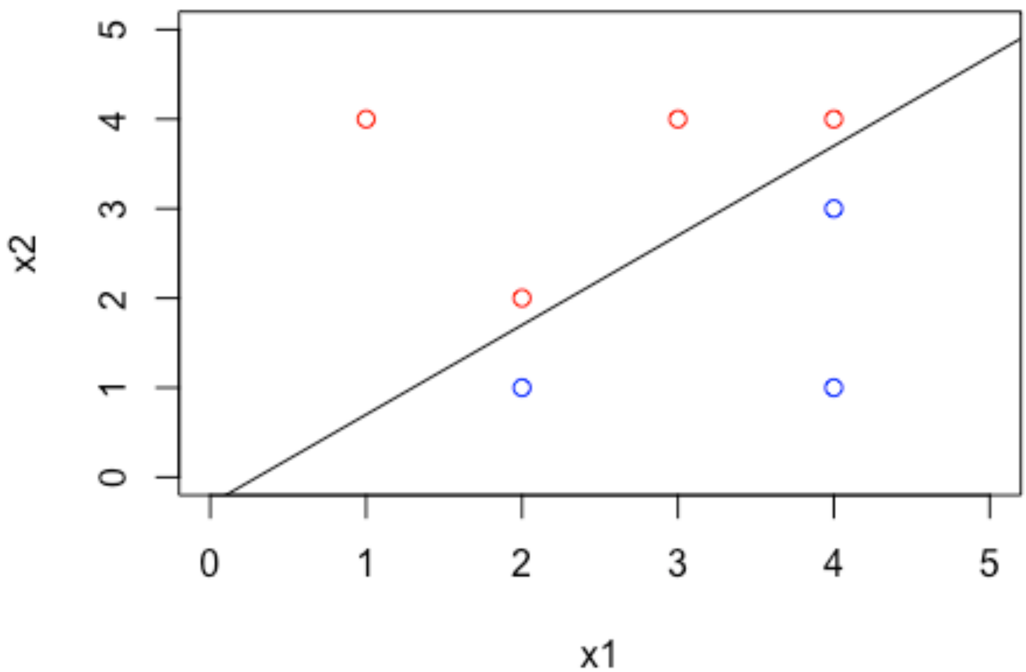

```
#Ex3 page 368
#a
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
colors = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))

#b
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
```



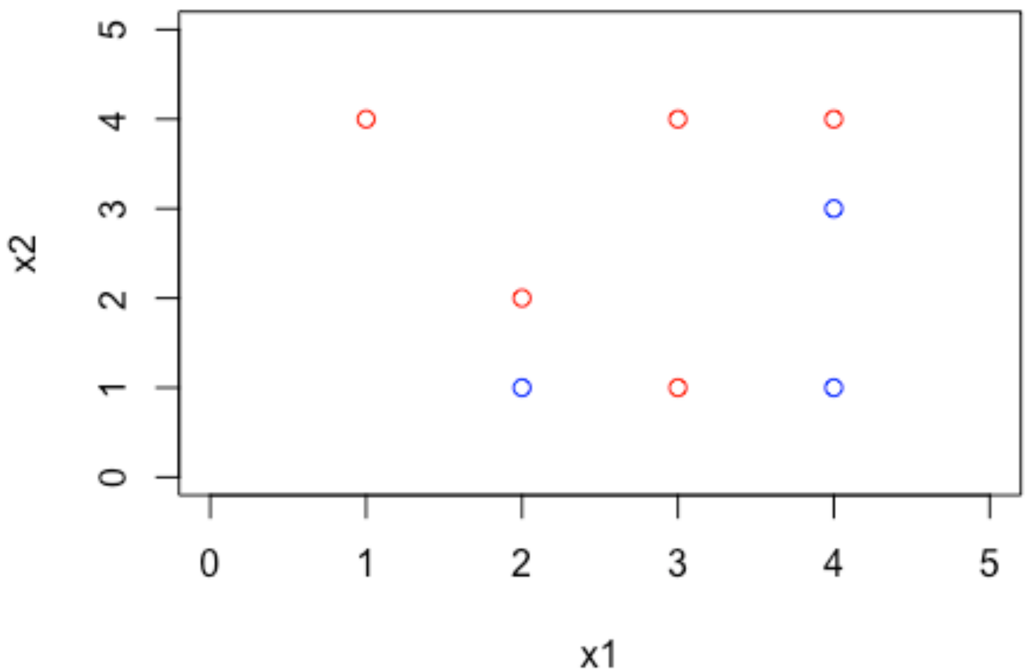

```r
#As shown in the plot, the optimal separating hyperplane has to be between
the observations (2,1) and (2,2), and between the observations (4,3) and
(4,4). So it is a line that passes through the points (2,1.5) and (4,3.5)
which equation is X1 - X2 - 0.5 = 0



#c
#The classification rule is classifies to Red if X1-X2-0.5<0, and classify to
Blue otherwise

#d
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
abline(-1, 1, lty = 2)
abline(0, 1, lty = 2)
```




```
#The margin is here equal to 1/4.

#e
#The support vectors are the points (2,1), (2,2), (4,3) and (4,4).

#f
#By examining the plot, it is clear that if we moved the observation (4,1),
we would not change the maximal margin hyperplane since it is not a support
vector.

#g
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.3, 1)
```

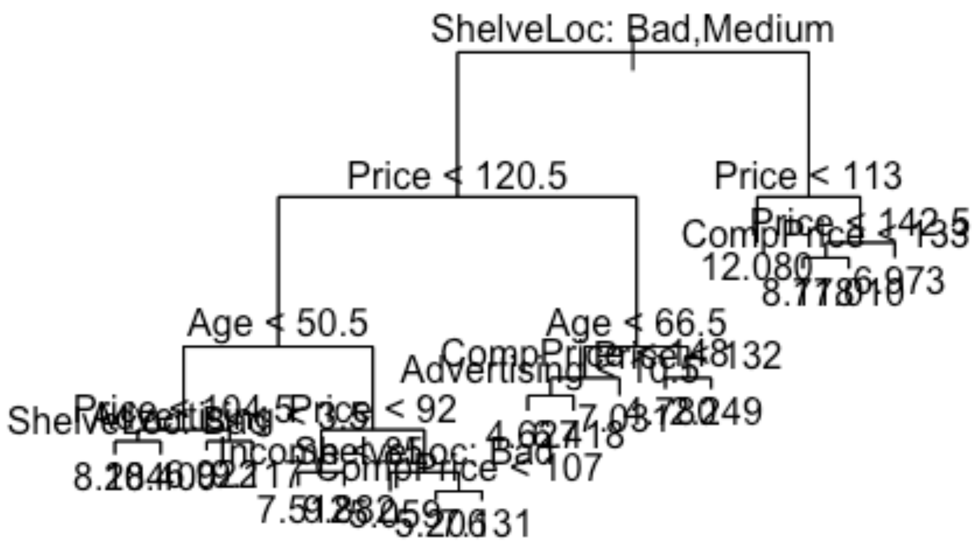



```r
#h
plot(x1, x2, col = colors, xlim = c(0, 5), ylim = c(0, 5))
points(c(3), c(1), col = c("red"))
```

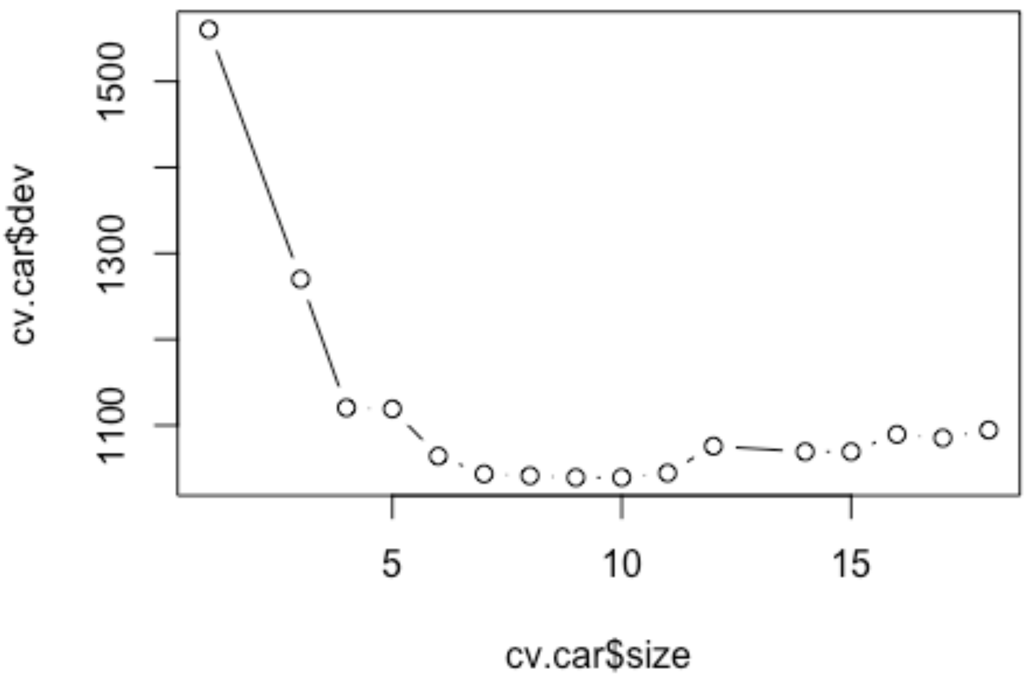



```
#Ex 8 Page 333
#a
library(ISLR)
set.seed(1)
train = sample(1:nrow(Carseats), nrow(Carseats) / 2)
Car.train = Carseats[train, ]
Car.test = Carseats[-train,]

#b
library(tree)
reg.tree = tree(Sales~.,data = Car.train)
summary(reg.tree)

##
## Regression tree:
## tree(formula = Sales ~ ., data = Car.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:
```

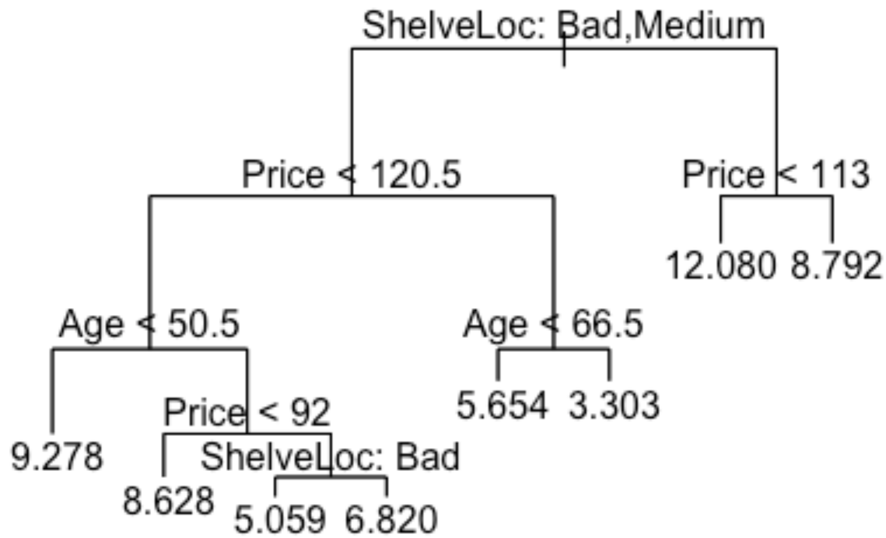
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -4.2570 -1.0360  0.1024  0.0000  0.9301  3.9130

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




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

## [1] 4.148897

#mean squared error is 4.148897

#c
cv.car = cv.tree(reg.tree)
plot(cv.car$size, cv.car$dev, type = "b")
```




```r
#8 is the optimal size
prune.car = prune.tree(reg.tree, best = 8)
plot(prune.car)
text(prune.car,pretty=0)
```



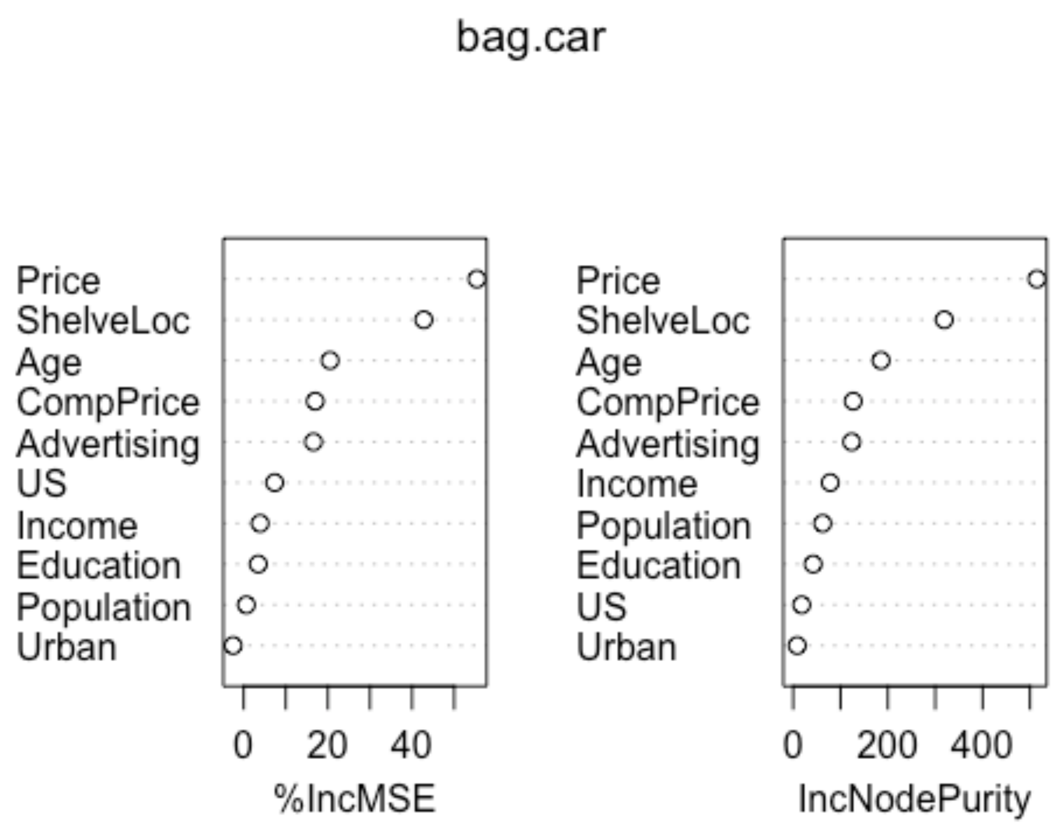

```
yhat=predict(prune.car, newdata= Car.test)
mean((yhat-Car.test$Sales)^2)

## [1] 5.09085

#the MSE increases from previous result

#d
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

bag.car = randomForest(Sales~.,data=Car.train,mtry = 10, importance = TRUE)
yhat.bag = predict(bag.car,newdata=Car.test)
mean((yhat.bag-Car.test$Sales)^2)
```

```
## [1] 2.633915
```

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

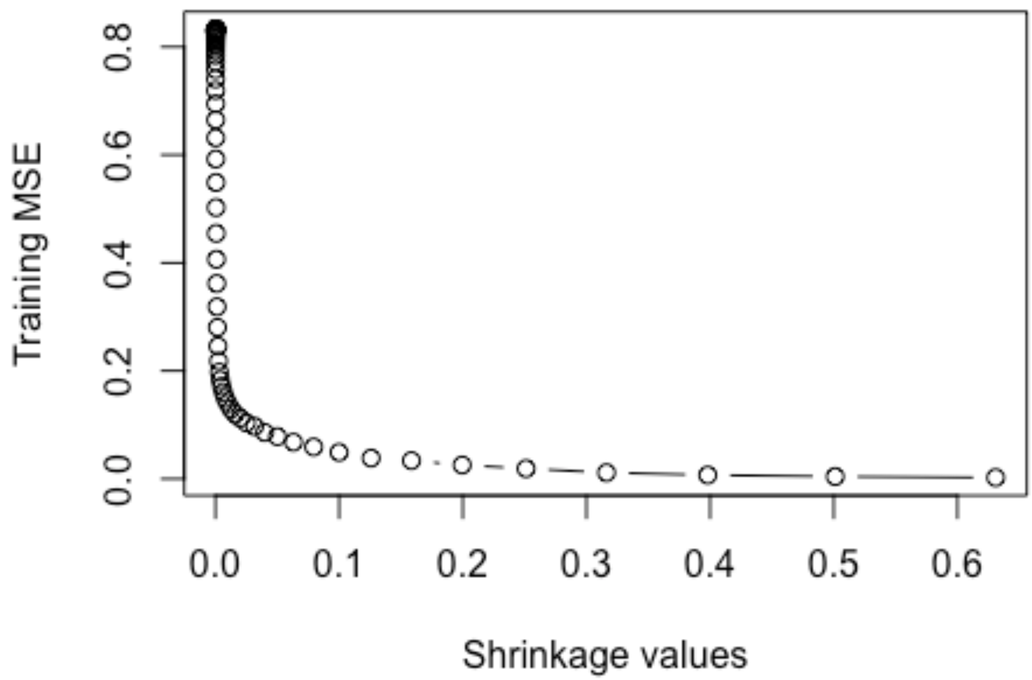
```
##                %IncMSE IncNodePurity
## CompPrice   16.9874366    126.852848
## Income       3.8985402     78.314126
## Advertising 16.5698586    123.702901
## Population   0.6487058     62.328851
## Price       55.3976775    514.654890
## ShelveLoc   42.7849818    319.133777
## Age         20.5135255    185.582077
## Education    3.4615211     42.253410
## Urban       -2.5125087      8.700009
## US           7.3586645     18.180651
```

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

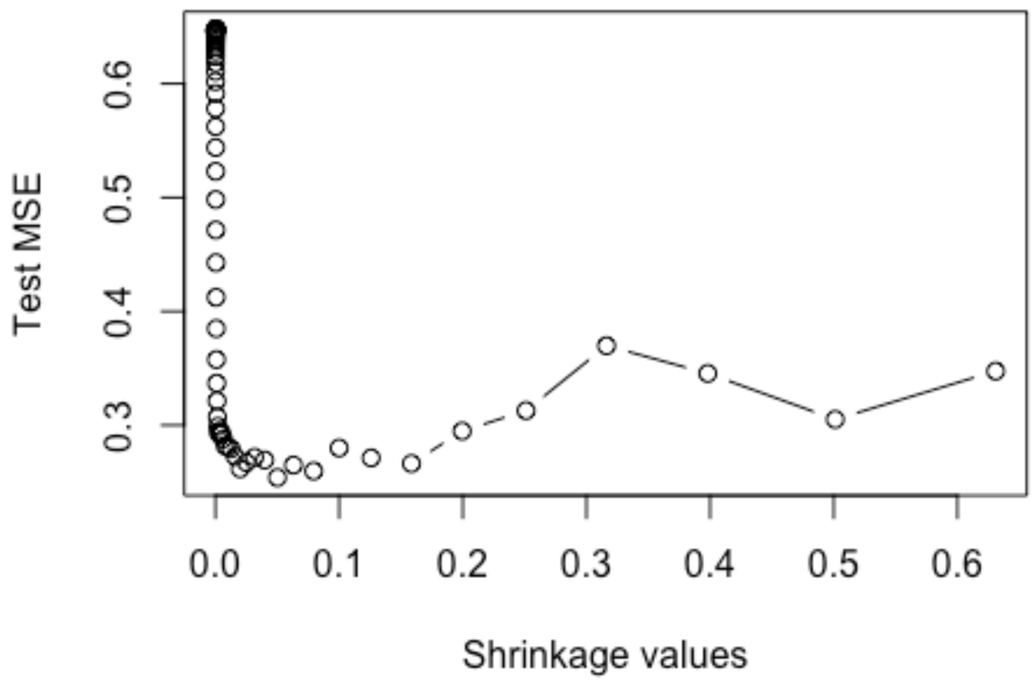



```r
#the MSE is the lowest we have obtain so far.
#the most important variables are the price and the quality of shelving
location

#e
rf.car = randomForest(Sales~.,data=Car.train,mtry = 3, importance = TRUE)
```

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

## [1] 3.321154

#mse is higher than using bagging, but lower than previous approaches

#Ex 10 Page 334
#a
attach(Hitters)
Hitters = na.omit(Hitters)
Hitters$Salary = log(Hitters$Salary)

#b
train = 1:200
hitters.train = Hitters[train,]
hitters.test = Hitters[-train,]

#c
library(gbm)

## Warning: package 'gbm' was built under R version 3.5.2

## Loaded gbm 2.1.5

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")
```

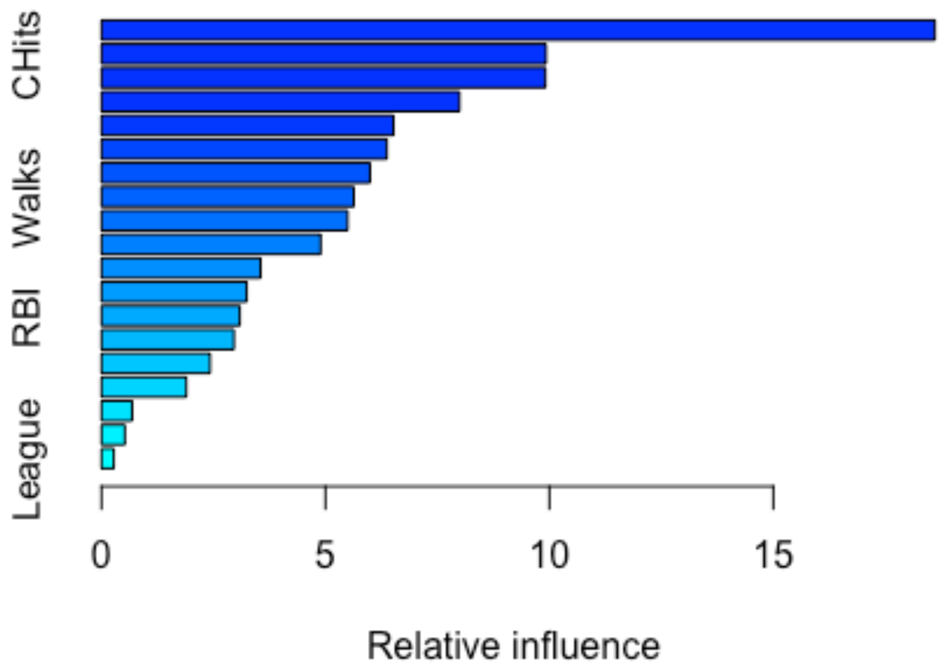



```
#d
test.err <- rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {
  boost.hitters = gbm(Salary ~ ., data = hitters.train, distribution =
"gaussian", n.trees = 1000, shrinkage = lambdas[i])
  yhat = predict(boost.hitters, hitters.test, n.trees = 1000)
  test.err[i] = mean((yhat - hitters.test$Salary)^2)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




```
min(test.err)

## [1] 0.2539576

lambdas[which.min(test.err)]

## [1] 0.05011872

#e
library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

fit1 = lm(Salary ~ ., data = hitters.train)
pred1 = predict(fit1, hitters.test)
mean((pred1 - hitters.test$Salary)^2)

## [1] 0.4917959

x = model.matrix(Salary ~ ., data = hitters.train)
x.test = model.matrix(Salary ~ ., data = hitters.test)
```

```r
y = hitters.train$Salary
fit2 = glmnet(x, y, alpha = 0)
pred2 = predict(fit2, s = 0.01, newx = x.test)
mean((pred2 - hitters.test$Salary)^2)

## [1] 0.4570283

#test for boosting is lower than for linear regression and ridge regression

#f
boost.hitters <- gbm(Salary ~ ., data = hitters.train, distribution =
"gaussian", n.trees = 1000, shrinkage = lambdas[which.min(test.err)])
summary(boost.hitters)
```




```
##                   var    rel.inf
## CAtBat         CAtBat 18.6088071
## CHits           CHits  9.9255599
## CRBI             CRBI  9.9127208
## CWalks         CWalks  7.9937659
## PutOuts       PutOuts  6.5213867
## CHmRun         CHmRun  6.3717198
## Years           Years  5.9956430
## Walks           Walks  5.6344320
## CRuns           CRuns  5.4956908
```

```
## Hits           Hits  4.8939430
## Assists     Assists  3.5530609
## HmRun         HmRun  3.2430160
## RBI             RBI  3.0879822
## AtBat         AtBat  2.9584142
## Errors       Errors  2.4256790
## Runs           Runs  1.8949909
## Division   Division  0.6851062
## NewLeague NewLeague  0.5221752
## League       League  0.2759064
```

```r
#from our summary we see that CAtBat is the most important variable.
#The next most important variables after CAtBat are CRuns and CRBI

#g
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] 0.2319799
```

```r
#The MSE for baaging is slightly better than for boosting
```