Assignment 2

Hao Wang

1/29/2017

Homework 2

Due February 2.

(a) Fit trees of various sized to the simple x=mileage, y =price problem using the susedcars.csv data.

What looks like a reasonable tree size?

(b) Still just using the x=mileage, y =price problem, use cross-validation to choose the tree size. How does the tree chosen with CV compare with the one you chose in (a)?

(c) Use cross validation to fit a tree using y = price and x = all the other variables.

How "good" is the fit?

Is the tree you fit interpretable?

Loading Dataset: used cars

```
mydata <-read.csv('https://raw.githubusercontent.com/ChicagoBoothML/DATA___UsedCars/master/UsedCars_sma
```

Fit tree regressions with x=mileage, y=price

```
library(MASS)
library(tree)
#-----------------------------------------------------
#fit a tree to car data just using mileage.
#first get a big tree using a small value of mindev
temp = tree(price~mileage,data=mydata,mindev=.0001)
cat("first big tree size: \n")
```

```
## first big tree size:
```

```
print(length(unique(temp$where)))
```

```
## [1] 127
```

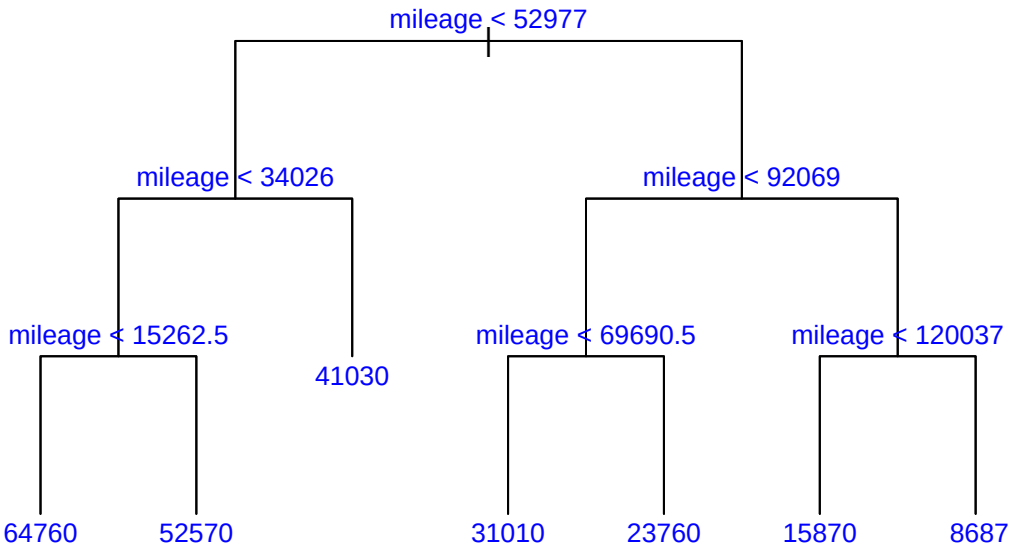
```
#if the tree is too small, make mindev smaller!!
#-----------------------------------------------------
#then prune it down to one with 7 leaves
car.tree.7=prune.tree(temp,best=7)
cat("pruned tree size: \n")
```

```
## pruned tree size:
```

```
print(length(unique(car.tree.7$where)))
```

```
## [1] 7
```

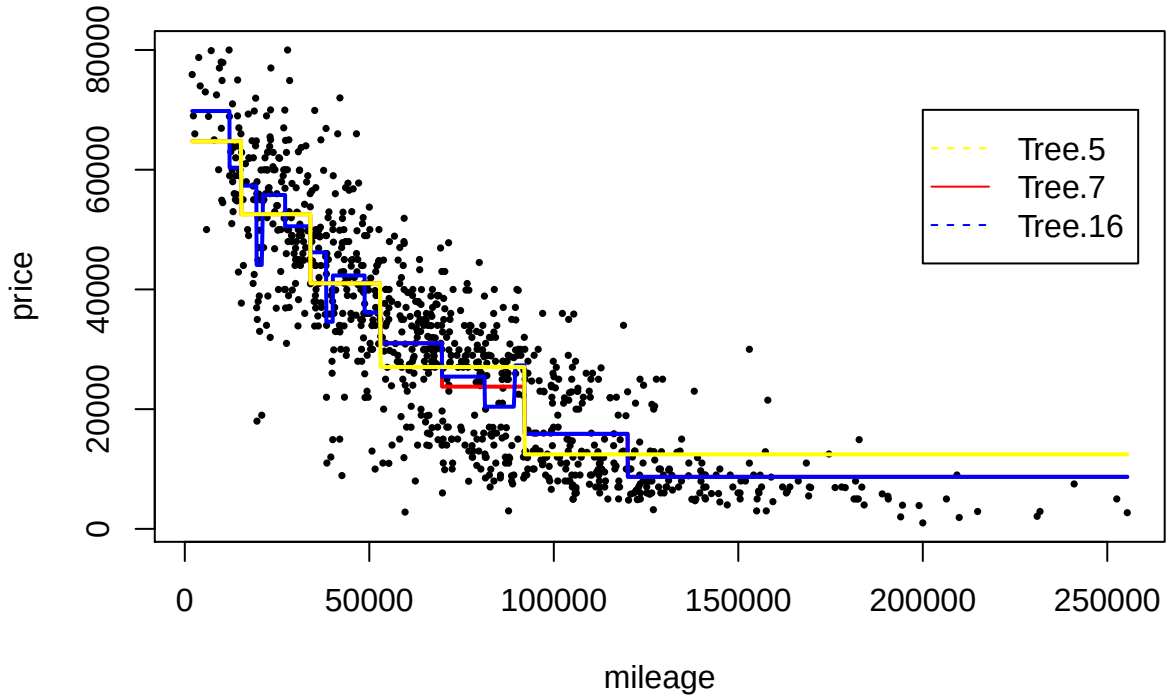
```
#-----------------------------------------------
#plot the tree
plot(car.tree.7,type="uniform")
text(car.tree.7,col="blue",label=c("yval"),cex=.8)
```




```
#-----------------------------------------------
#plot data with fit
#get fit
car.fit.7 = predict(car.tree.7) #get training fitted values
#plot fit
attach(mydata)
plot(mileage,price,cex=.5,pch=16) #plot data
oo=order(mileage)
#lines(mileage[oo],car.fit.7[oo],col="red",lwd=3) #step function fit
#-----------------------------------------------
#get tree at different size
car.tree.16 = prune.tree(temp,best=16)
car.fit.16 = predict(car.tree.16) #get training fitted values

car.tree.5 = prune.tree(temp,best=5)
car.fit.5 = predict(car.tree.5) #get training fitted values

lines(mileage[oo],car.fit.7[oo],col="red",lwd=2) #step function fit
lines(mileage[oo],car.fit.16[oo],col="blue",lwd=2) #step function fit
lines(mileage[oo],car.fit.5[oo],col="yellow",lwd=2) #step function fit
leg.txt <- c("Tree.5", "Tree.7","Tree.16") # Text for legend
legend(list(x = 200000,y = 70000),           # Set location of the legend
       legend = leg.txt,                # Specify text
       col = c("yellow","red","blue"),     # Set colors for legend
       lty = c(2,1),                    # Set type of lines in legend
       merge = TRUE)                    # merge points and lines
```




```
detach(mydata)
```

Using CV choosing the size of tree

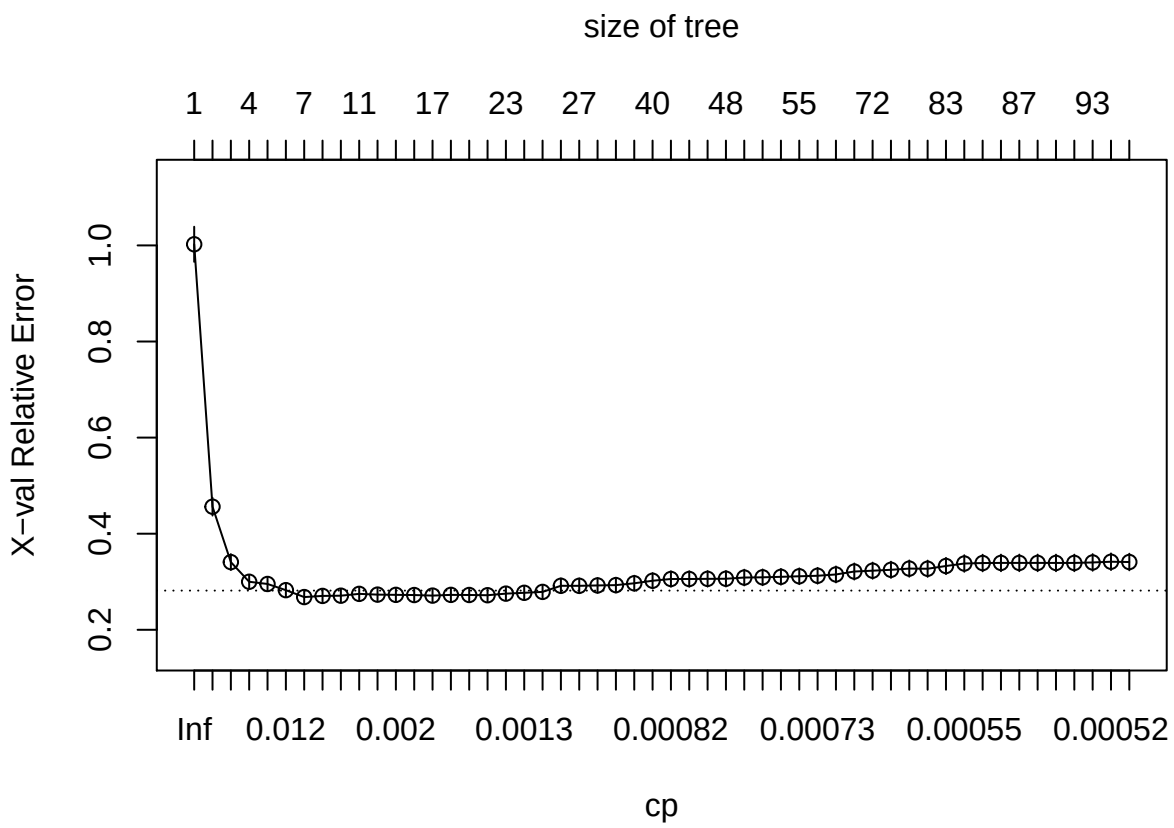
```
library(rpart)
attach(mydata)
set.seed(99)
#-----------------------------------------------
#Pick plotdata
ddf <- mydata[, c('price', 'mileage')]
attach(ddf)
```

```
## The following objects are masked from mydata:
##
##     mileage, price
```

```
#-------------------------------------------------
#fit a single tree and plot variable importance
#fit a big tree using rpart.control
big.tree = rpart(price~.,method="anova",data=ddf,
control=rpart.control(minsplit=5,cp=.0005))
nbig = length(unique(big.tree$where))
cat("size of big tree: ",nbig,"\n")
```

```
## size of big tree:  95
```

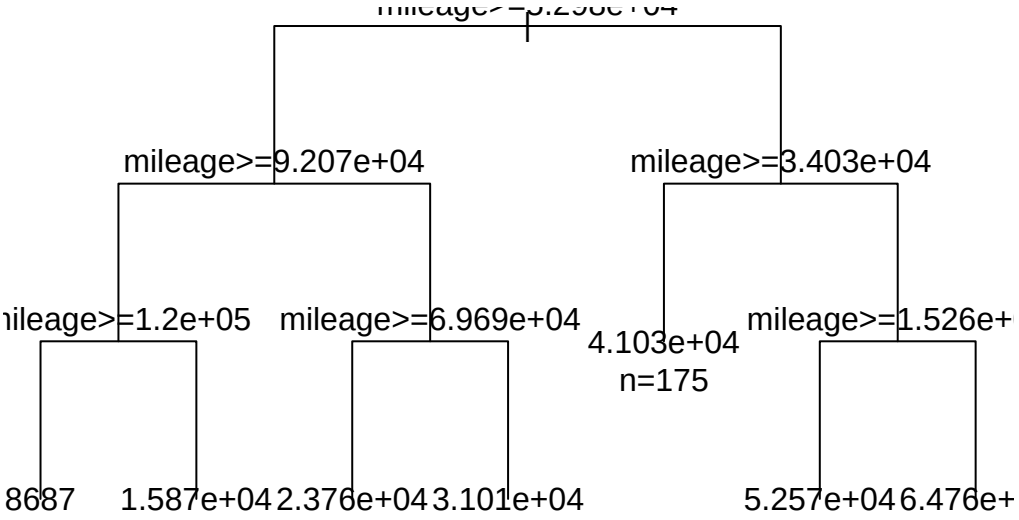
```
#-------------------------------------------------
#look at CV results
plotcp(big.tree)
```



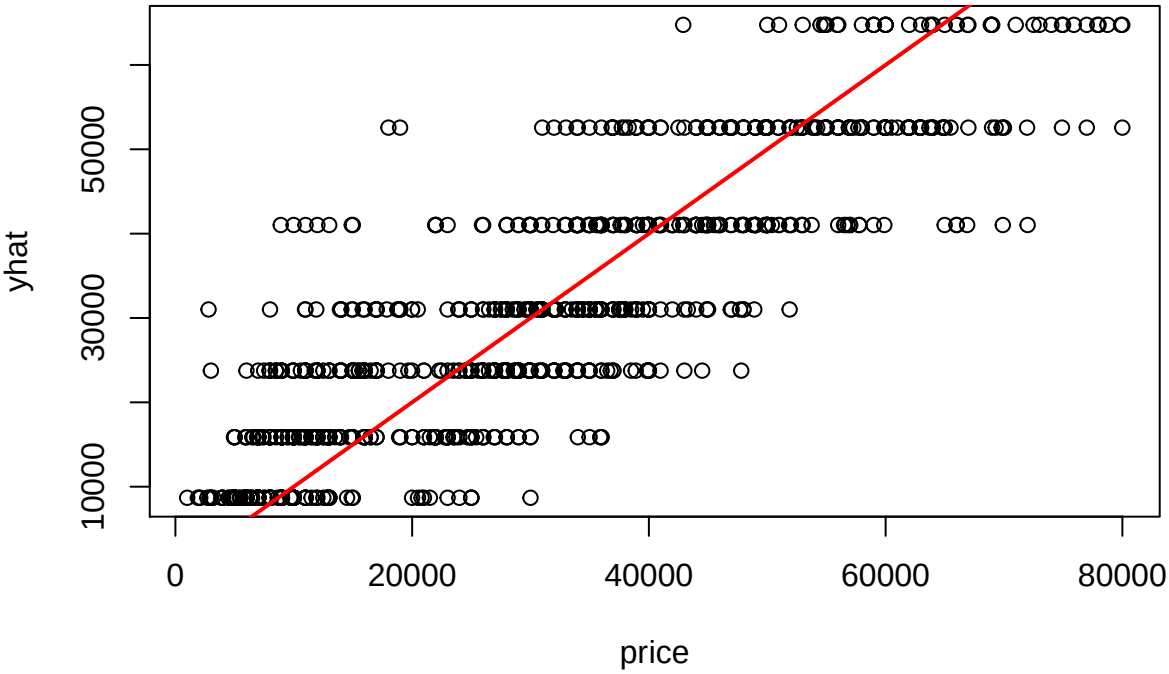

```
iibest = which.min(big.tree$cptable[,"xerror"]) #which has the lowest error
bestcp=big.tree$cptable[iibest,"CP"]
bestsize = big.tree$cptable[iibest,"nsplit"]+1
cat("Best size is ", bestsize, "\n")
```

```
## Best size is  7
```

```
#-------------------------------------------------
#prune to good tree
best.tree = prune(big.tree,cp=bestcp)
#-------------------------------------------------
#plot tree
#plot(best.tree,uniform=TRUE,branch=.5,margin=.5)
#text(best.tree,digits=4,use.n=TRUE,fancy=TRUE,bg="lightblue")
plot(best.tree,uniform=TRUE)
text(best.tree,digits=4,use.n=TRUE)
```




```
#-------------------------------------------------
#get fits
yhat = predict(best.tree)
yhat1 =yhat
plot(price,yhat)
abline(0,1,col="red",lwd=2)
```




```
detach(ddf)
```

Best size is 7.

Tree regression using all variables.

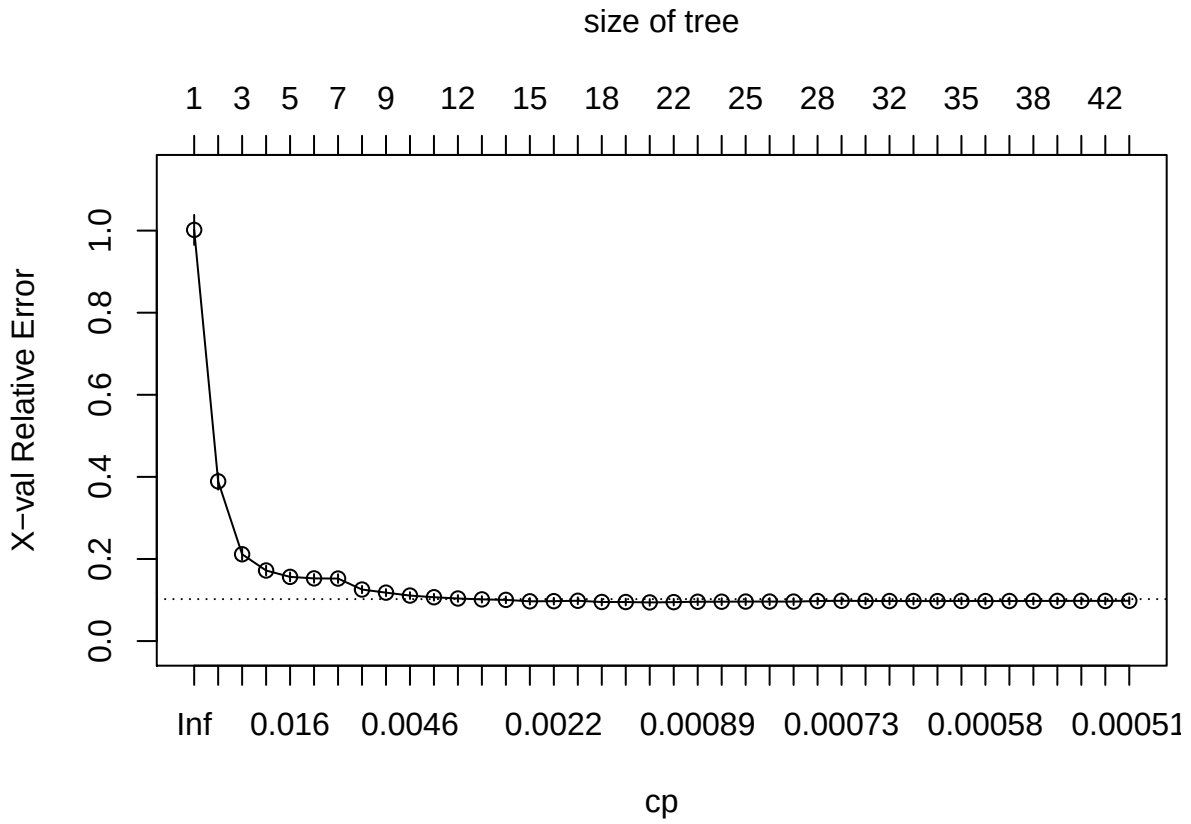
```
attach(mydata)
```

```
## The following objects are masked from mydata (pos = 3):
##
##     color, displacement, isOneOwner, mileage, price, trim, year
```

```
#-----------------------------------------------------
#fit a single tree and plot variable importance
#fit a big tree using rpart.control
big.tree = rpart(price~.,method="anova",data=mydata,
control=rpart.control(minsplit=5,cp=.0005))
nbig = length(unique(big.tree$where))
cat("size of big tree: ",nbig,"\n")
```

```
## size of big tree:  43
```

```
#-----------------------------------------------------
#look at CV results
plotcp(big.tree)
```

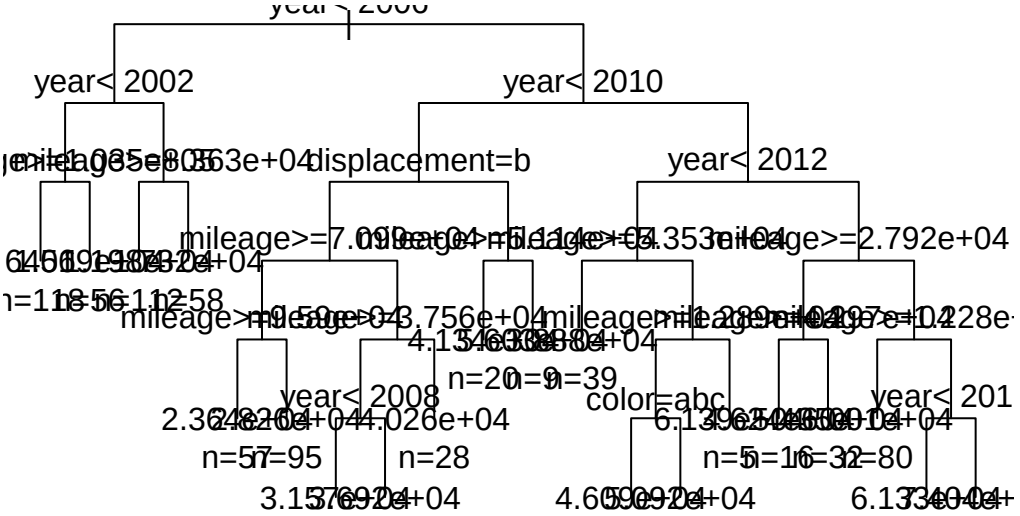



```
iibest = which.min(big.tree$cptable[,"xerror"]) #which has the lowest error
bestcp=big.tree$cptable[iibest,"CP"]
bestsize = big.tree$cptable[iibest,"nsplit"]+1
cat("Best size is ", bestsize, "\n")
```

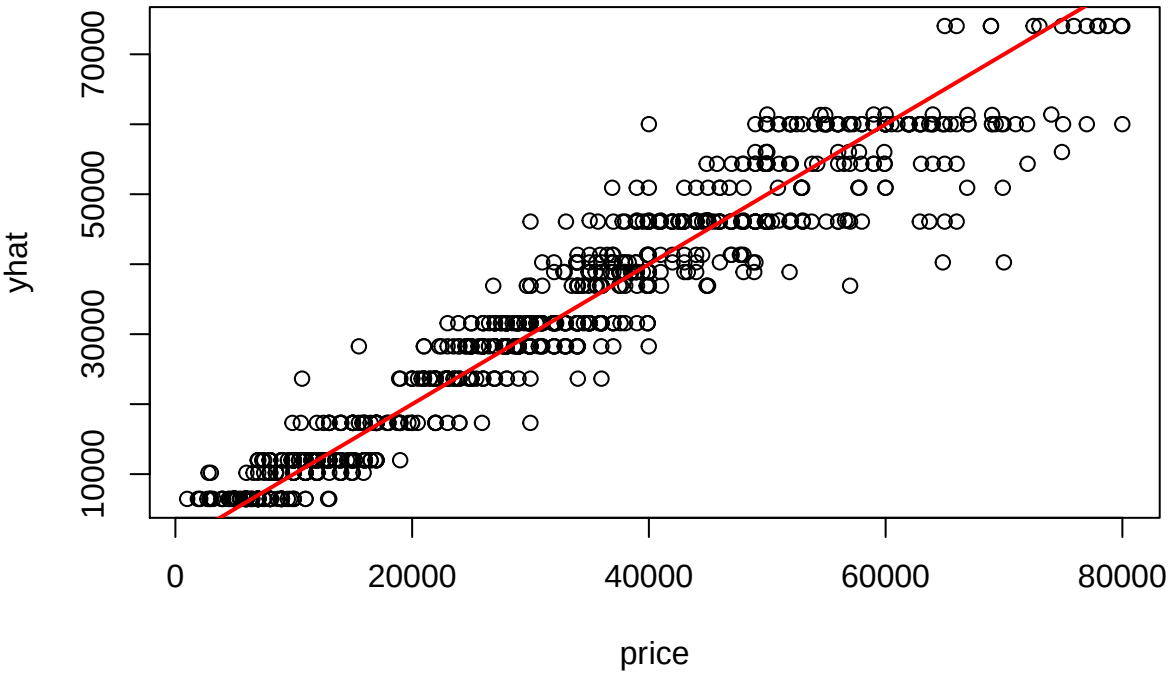
```
## Best size is  20
```

```
#-----------------------------------------------------
#prune to good tree
best.tree = prune(big.tree,cp=bestcp)
#-----------------------------------------------------
#plot tree
```

```
#plot(best.tree,uniform=TRUE,branch=.5,margin=.5)
#text(best.tree,digits=4,use.n=TRUE,fancy=TRUE,bg="lightblue")
plot(best.tree,uniform=TRUE)
text(best.tree,digits=4,use.n=TRUE)
```




```
#------------------------------------------------
#get fits
yhat = predict(best.tree)
yhat2 = yhat
plot(price,yhat)
abline(0,1,col="red",lwd=2)
```




```
detach(mydata)
#------------------------------------------------
#Comparing fit
yhat1.cor <- cor(yhat1, price)
```

```r
yhat2.cor <- cor(yhat2, price)
table.cor <- data.frame(yhat1.cor, yhat2.cor)
table(table.cor)
```

```
##                     yhat2.cor
## yhat1.cor            0.965964157772155
##   0.866485906863507                  1
```