BDA 551-Assignment 2

Feray Ece Topcu

July 14, 2018

- Load required libraries:

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

- Read Dataset & examine columns:

```
#attach(Carseats)


carseats <- Carseats


str(carseats)

## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2
3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

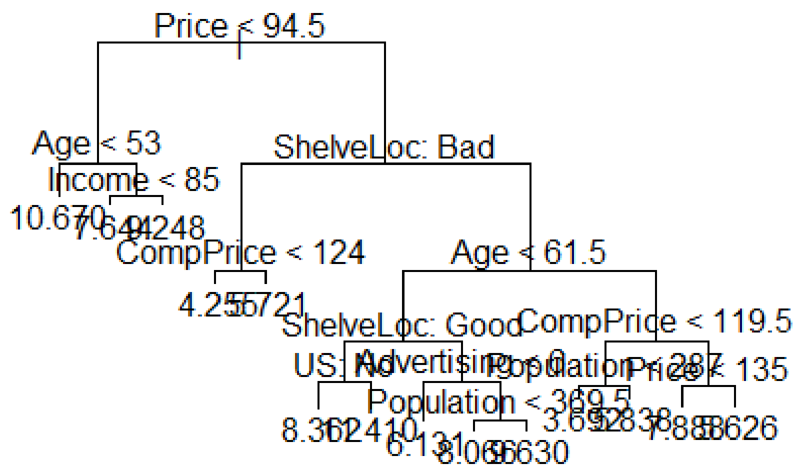
Question 1. Split the data set into a training set and a test set.

```r
set.seed(1)
train  =  sample(1:nrow(carseats),  nrow(carseats)/4)
```

Question 2.Fit a regression tree to the training set. Plot the tree, and interpret the results. What test error rate do you obtain ?

- Fit a tree with train dataset, plot the regression tree:

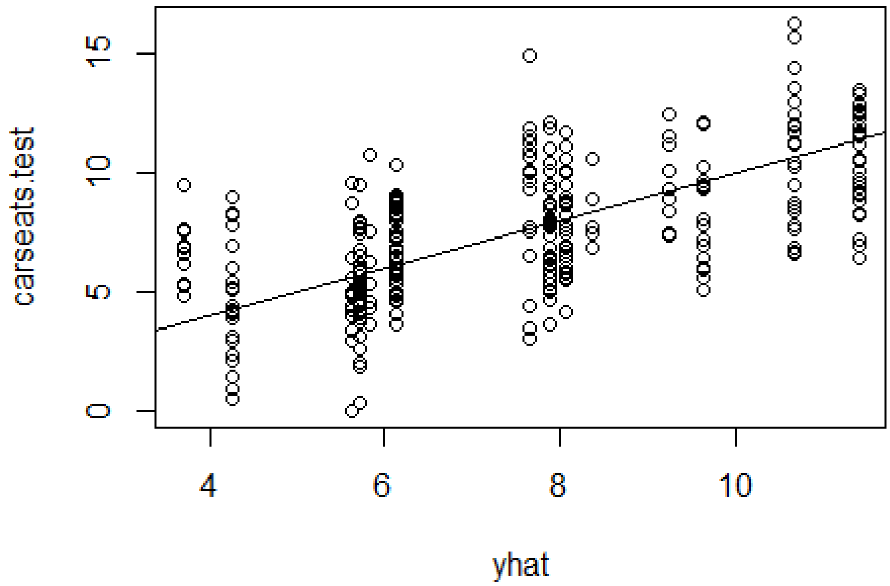
```r
tree.carseats = tree(Sales~., carseats, subset = train) # fit tree
summary(tree.carseats)

##
## Regression tree:
## tree(formula = Sales ~ ., data = carseats, subset = train)
## Variables actually used in tree construction:
## [1] "Price"       "Age"         "Income"      "ShelveLoc"   "CompPrice"
## [6] "US"          "Advertising" "Population"
## Number of terminal nodes:  14
## Residual mean deviance:  2.524 = 217.1 / 86
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.53200 -0.96410 -0.04806  0.00000  0.81960  4.41900

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




```r
yhat = predict(tree.carseats, newdata = carseats[-train, ]) # test set predic
tions
table(yhat) # test set predictions
```

```
## yhat
##            3.692 4.25454545454545            5.626          5.72125
##               13               23               13               34
##            5.838 6.13111111111111 7.64428571428572 7.88833333333333
##                8               43               17               32
## 8.06555555555556            8.362            9.248             9.63
##               30                5                9               18
## 10.6677777777778           11.414
##               24               31
```

```r
carseats.test = carseats[-train, "Sales"] # test set actual values
plot(yhat, carseats.test) # note as we average at each node predictions are b
unched
abline(0, 1)
```




```
mean((yhat - carseats.test)^2) # calc MSE

## [1] 5.179786
```

- MSE is *5.179* when seed=1.

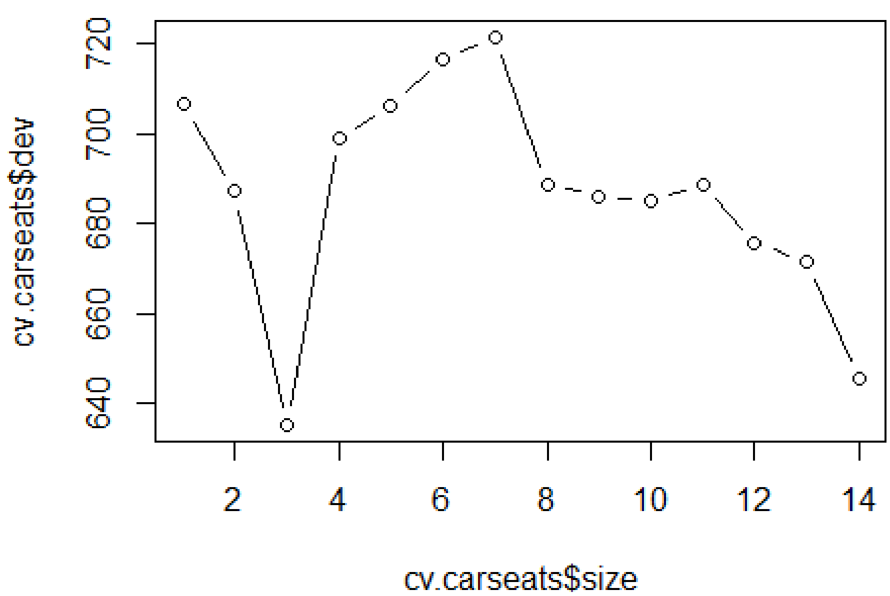
Question 3.Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test error rate ?

- Apply cross validation:

```
cv.carseats=cv.tree(tree.carseats)


cv.carseats

## $size
##  [1] 14 13 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 645.2752 671.5432 675.3820 688.7686 684.7958 686.0333 688.7464
```

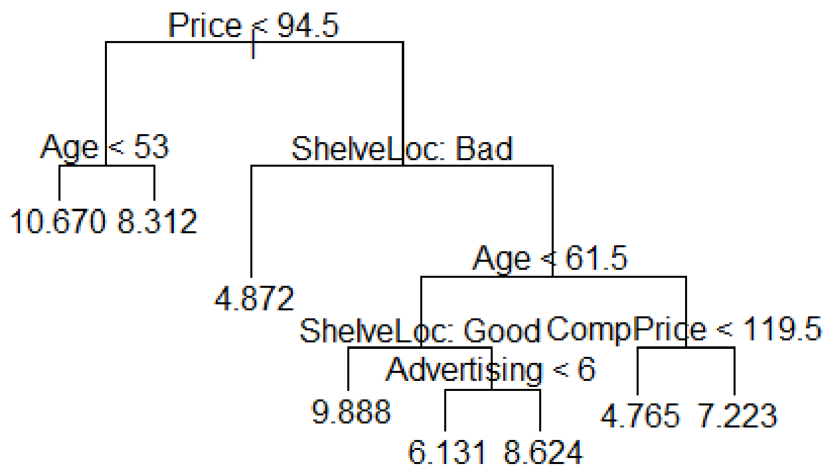
```
##  [8] 721.4186 716.5108 706.1968 698.9934 635.1100 687.1329 706.3378
##
## $k
##  [1]       -Inf   7.501374   7.866921   9.963556  11.513290  18.064066
##  [7]  23.286760  28.529143  34.052429  34.949434  38.038916  60.275200
## [13]  93.727701 104.867337
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv.carseats$size,cv.carseats$dev,type='b')
```



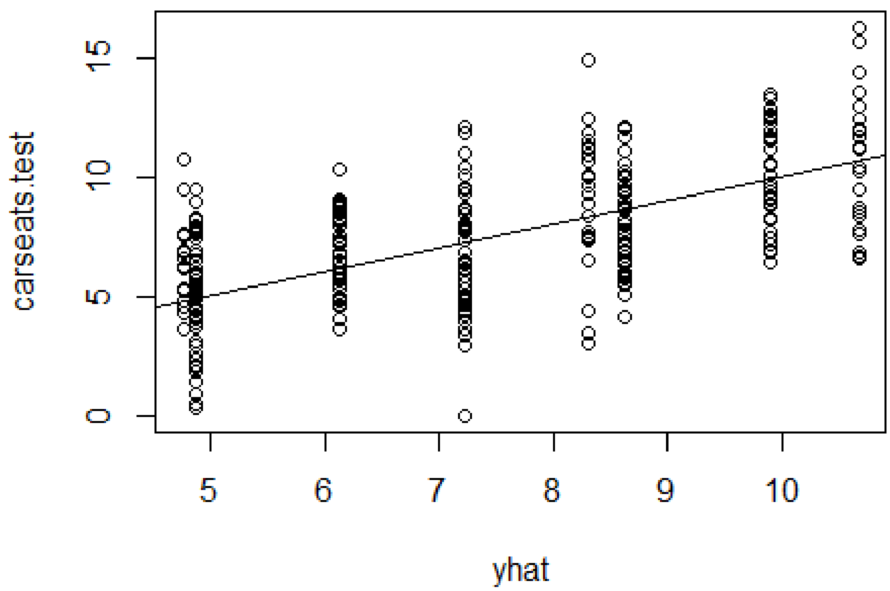
- Fitting Regression Trees using 8 terminal nodes:

```
prune.carseats=prune.tree(tree.carseats,best=8)
plot(prune.carseats)
text(prune.carseats,pretty=0)
```



- Calculate MSE for pruned tree:

```
yhat=predict(prune.carseats,newdata=carseats[-train,])
carseats.test=carseats[-train,"Sales"]
plot(yhat,carseats.test)
abline(0,1)
```




```
mean((yhat-carseats.test)^2)

## [1] 5.152601
```

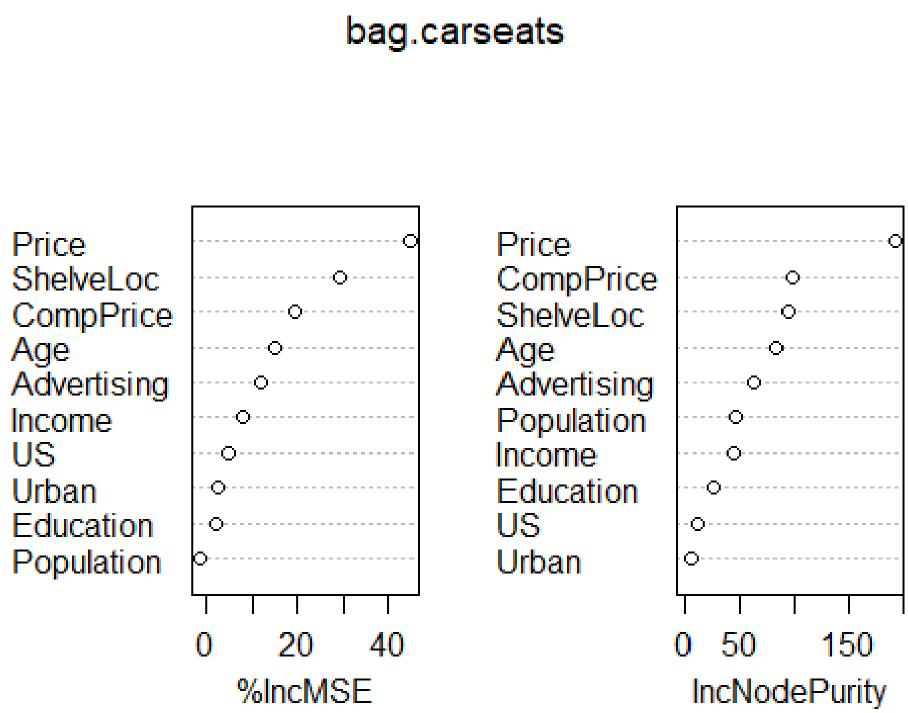
MSE is *4.780099* for pruned tree. So, pruned tree gives better results.

Question 4. Use the bagging approach in order to analyze this data. What test error rate do you obtain? Use the "importance()" function to determine which variables are most important. Use ntree=1000.

- Apply bagging:

```
library(randomForest)

## randomForest 4.6-14

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

set.seed(1)
```

```r
bag.carseats=randomForest(Sales~.,data=carseats,subset=train,mtry=13,ntree=1000,importance=TRUE)


yhat.bag = predict(bag.carseats,newdata=carseats[-train,])
mean((yhat.bag-carseats.test)^2)

## [1] 3.57208

importance(bag.carseats)

##                %IncMSE IncNodePurity
## CompPrice   19.568050     97.703899
## Income       8.046965     43.960119
## Advertising 11.982097     62.940157
## Population  -1.348149     45.458754
## Price       44.934798    193.698574
## ShelveLoc   29.040872     94.287851
## Age         15.024811     82.350640
## Education    2.151313     25.327554
## Urban        2.355167      4.435798
## US           4.654712      9.741332

varImpPlot(bag.carseats)
```

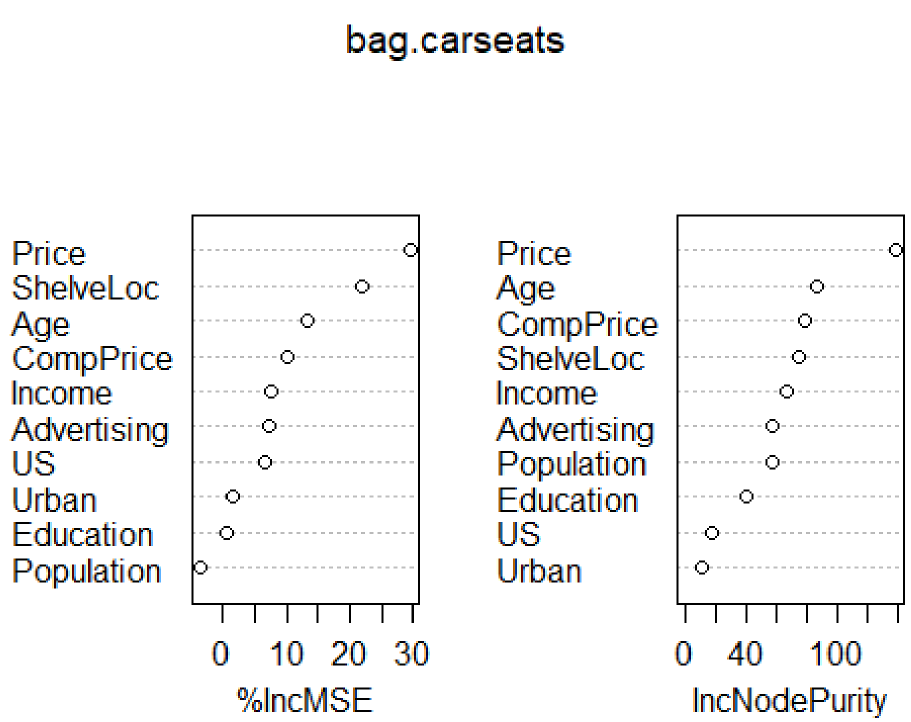


MSE is *3.51809* for bagging. So, Bagging is a better algorithm for this dataset to forecast Sales. Due to output of importance() function, Price is the most important predictor in this dataset.

Question 5. Use random forests to analyze this data. What test error rate do you obtain? Use the "importance()" function to determine which variables are most important. Use ntree=1000, mtry=3.

```
library(randomForest)
set.seed(1)


bag.carseats=randomForest(Sales~.,data=carseats,subset=train,mtry=3,ntree=1000,importance=TRUE)
```

```
yhat.bag = predict(bag.carseats,newdata=carseats[-train,])
mean((yhat.bag-carseats.test)^2)

## [1] 4.417369

importance(bag.carseats)

##                %IncMSE IncNodePurity
## CompPrice    9.9937219      79.21195
## Income       7.5513798      67.11762
## Advertising  7.2126094      57.66589
## Population  -3.4882346      57.20082
## Price       29.6031283     139.44304
## ShelveLoc   21.8266814      75.19197
## Age         13.4348826      86.63961
## Education    0.4021779      39.98516
## Urban        1.5420490      10.29001
## US           6.4859709      16.54111

varImpPlot(bag.carseats)
```



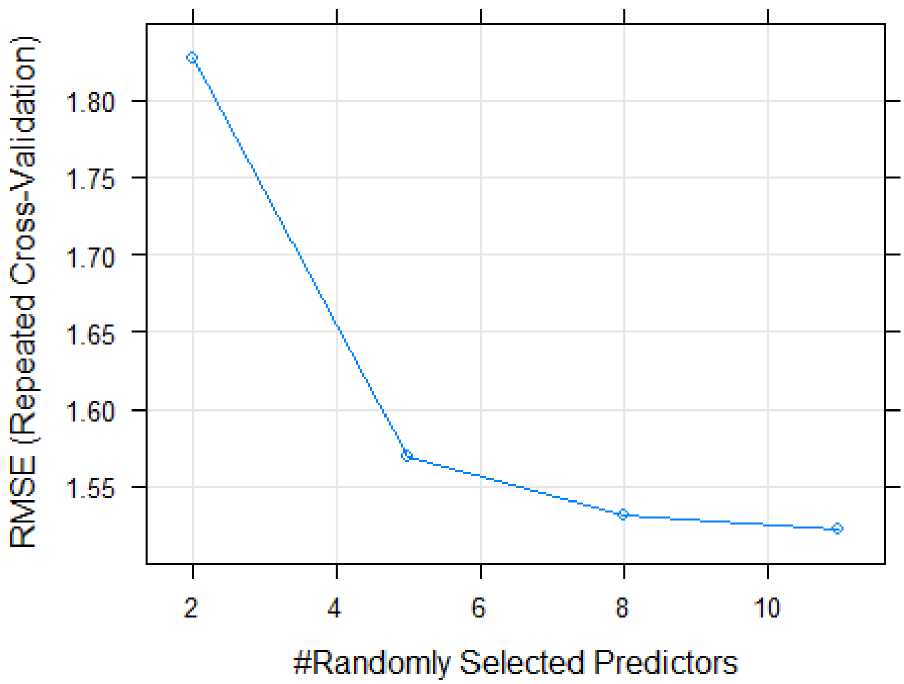
MSE is *4.417369* for RandomForest while mtry=3 and ntree=1000. So, Bagging is a better method than RandomForest to forecast Sales.

Question 6. By using 10 fold cross validation and grid search detect best parameters of ntree and mtry for random forests. What test error rate do you obtain by using best parameters.

- GridSearch & CV:

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

#library("FactoMineR")
#library("e1071")
#library(dplyr)


set.seed(1)


fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  repeats = 1)
#


rf_gridsearch <- train(Sales~., data=carseats, method = "rf",
                trControl = fitControl, verbose = FALSE,
                tuneLength = 4)
```

```r
plot(rf_gridsearch)
```

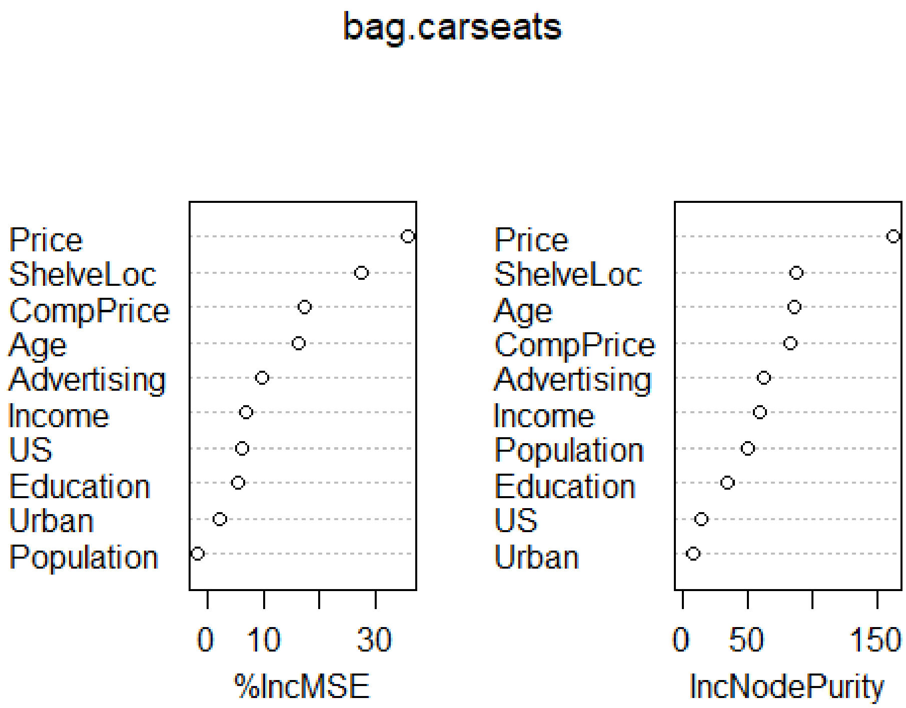



```r
#plot(rf_gridsearch, plotType = "level")
rf_gridsearch$results

##   mtry     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1    2 1.827595 0.6803037 1.474191 0.2203589 0.09071373 0.1609274
## 2    5 1.569929 0.7223584 1.260126 0.2174363 0.07556059 0.1707563
## 3    8 1.531404 0.7215676 1.221723 0.2129464 0.07542307 0.1723837
## 4   11 1.521658 0.7198204 1.218203 0.2038990 0.07049754 0.1700340

best(rf_gridsearch$results, metric="Rsquared", maximize=T)

## [1] 2

tolerance(rf_gridsearch$results, metric="Rsquared", maximize=T, tol=2)

## [1] 2

rf_gridsearch$results[2,]
```

```
##   mtry     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 2    5 1.569929 0.7223584 1.260126 0.2174363 0.07556059 0.1707563
```

The best value is *5* for mtry parameter. So, apply RandomForest with the best:

```
library(randomForest)
set.seed(1)



bag.carseats=randomForest(Sales~.,data=carseats,subset=train,mtry=5,ntree=100
0,importance=TRUE)



yhat.bag = predict(bag.carseats,newdata=carseats[-train,])
mean((yhat.bag-carseats.test)^2)

## [1] 3.864939

importance(bag.carseats)

##                %IncMSE IncNodePurity
## CompPrice   17.515685     83.072362
## Income       6.715830     58.983759
## Advertising  9.722781     61.356736
## Population  -1.734053     48.737245
## Price       35.898562    162.573253
## ShelveLoc   27.544971     87.295701
## Age         16.239920     86.265179
## Education    5.290071     34.353653
## Urban        2.095135      7.503634
## US           5.961277     13.292080

varImpPlot(bag.carseats)
```



MSE is *3.864939* when mtry=5 and ntree=1000 for RandomForest which is much more better before tuning. Before tuning, RandomForest's MSE is 4.4173 and MSE of bagging is 3.51809. Due to the results; tuned RandomForest is better than first RandomForest while Bagging approach is a bit improved than tuned RandomForest.