COMP3354 ASSIGNMENT 2

JAIN RAJAT, 3035453684

Chapter 6 Exercise:

Question 8.

(a) Using rnorm to generate x and noise vector e.

```
#Chapyter 6 Exercise 8
#(a)
set.seed(3684)
x=rnorm(100)
e=rnorm(100)
```

(b) Generating y:

```
b0=1
b1=2
b2=-1
b3=0.3
y= b0 + b1*x + b2*(x^2)+b3*(x^3)+e
```

(c) Merging x and y to create the dataset and applying regsubsets to find models with best subset selection.

```
#Creating the dataset
data = data.frame(y,poly(x,10))

#Using regsubsets
regfit.full=regsubsets(y~.,data = data,nvmax=10)
regfit.summary=summary(regfit.full)
```

Finding the best model using Adjusted R-Square, Cp and AIC.

(i) Best model using Adjusted R-square: the model should have 3 predictor.

```
> which.max(regfit.summary$adjr2)
[1] 3
```

(ii) Best model using Cp: the model should have 3 predictors.

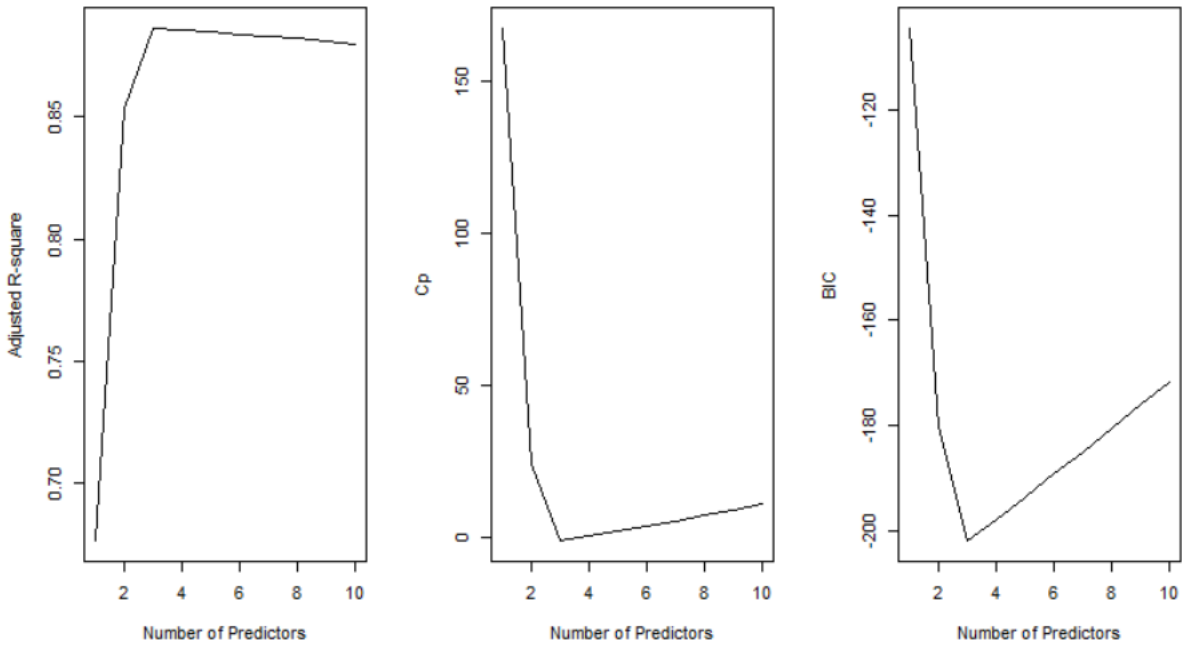
```
> which.min(regfit.summary$cp)
[1] 3
```

(iii) Best model using BIC: the model should have 3 predictors.

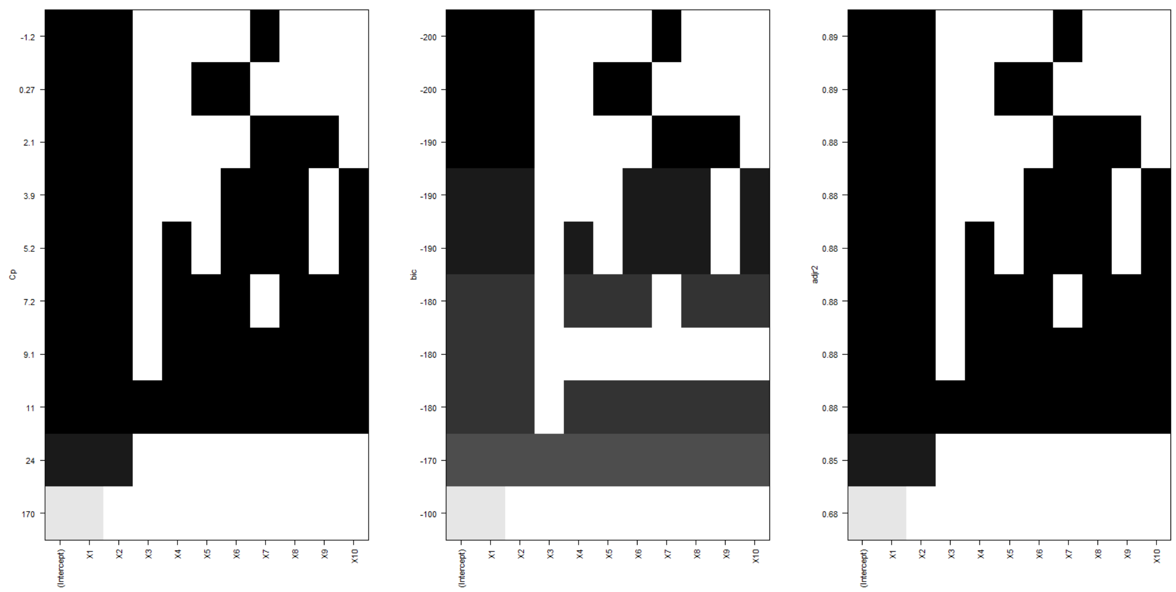
```
> which.min(regfit.summary$bic)
[1] 3
```

```
> coefficients(regfit.full,3)
(Intercept)          X1          X2          X7
 1.07627412  2.35623596 -1.16514887  0.01046843
```

- The methods selects 3 predictor variables but selects x^7 over x^3.

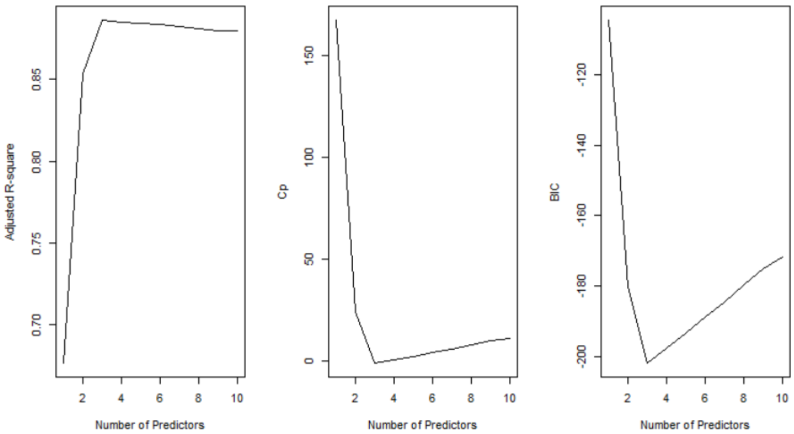



```
par(mfrow=c(1,3))
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
plot(regfit.full,scale="adjr2")
```



(d) Using the forward subset selection method:

```
#(d)
#Using regsubsets
regfit.full=regsubsets(y~.,data = data,nvmax=10, method="forward")
regfit.summary=summary(regfit.full)

par(mfrow=c(1,3))
plot(regfit.summary$adjr2, xlab="Number of Predictors", ylab = "Adjusted R-square", type
plot(regfit.summary$cp, xlab="Number of Predictors", ylab="Cp", type="l")
plot(regfit.summary$bic, xlab="Number of Predictors", ylab="BIC", type="l")


which.min(regfit.summary$cp)

which.min(regfit.summary$bic)

which.max(regfit.summary$adjr2)
```



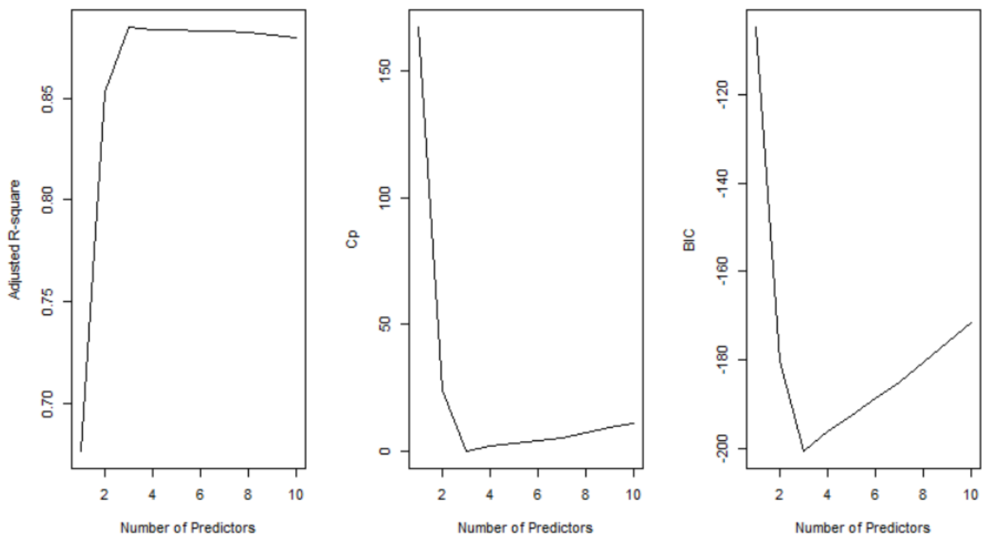

```
> which.min(regfit.summary$cp)
[1] 3
>
> which.min(regfit.summary$bic)
[1] 3
>
> which.max(regfit.summary$adjr2)
[1] 3
```

```
> coefficients(regfit.full, 3)
(Intercept)          X1          X2          X7
 1.07627412  2.35623596 -1.16514887  0.01046843
```

- The best model from forward selection involves x, x^2 and x^7.

Using the backward subset selection method:

```
#(d)
#Using regsubsets
regfit.full=regsubsets(y~.,data = data,nvmax=10, method="backward")
regfit.summary=summary(regfit.full)

par(mfrow=c(1,3))
plot(regfit.summary$adjr2, xlab="Number of Predictors", ylab = "Adjusted R-square", type
plot(regfit.summary$cp, xlab="Number of Predictors", ylab="Cp", type="l")
plot(regfit.summary$bic, xlab="Number of Predictors", ylab="BIC", type="l")


which.min(regfit.summary$cp)

which.min(regfit.summary$bic)

which.max(regfit.summary$adjr2)
```




```
> which.min(regfit.summary$cp)
[1] 3
>
> which.min(regfit.summary$bic)
[1] 3
>
> which.max(regfit.summary$adjr2)
[1] 3
```

```
> coefficients(regfit.full,3)
 (Intercept)           X1           X2           X9
 1.078881355  2.419817953 -1.177235617  0.001870457
```

- The best model from backward selection involves 3 variables, x, x^2 and x^9. However the coefficient of x^9 is not significant.

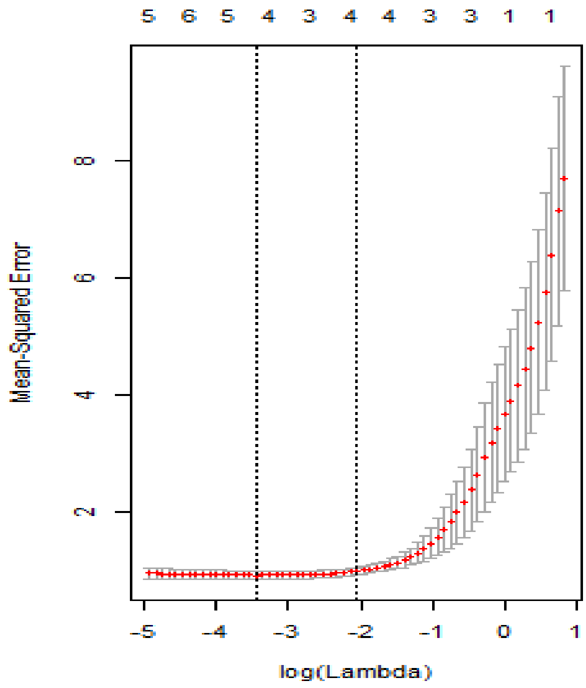
The results from forward and backward selection also involve 3 predictor variables. However, in case of backward selection x^9 is selected instead of x^7.

(e)

```
#(e)
library(glmnet)
X_matrix = model.matrix(y~.,data)[,-1]
lasso.model = cv.glmnet(X_matrix,y,alpha=1)
best.lambda = lasso.model$lambda.min
plot(lasso.model)
```

The best value of lambda obtained from cross-validation approa
ch is 0.03181702.

```
> best.lambda = lasso.model$lambda.min
> best.lambda
[1] 0.03181702
```



Plot of cross validation error as a function of lambda.

Now, we will fit the model on the training data on the basis of the value of lambda obtained from cross validation.

```
best.model = glmnet(X_matrix, y, alpha=1)
preds = predict(best.model, s=best.lambda, type="coefficients")
```

```
> preds
11 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  1.046971153
x1           2.247005459
x2          -1.114711339
x3           .
x4           .
x5           0.044136947
x6           .
x7           0.001976258
x8           .
x9           .
x10          .
```

- The best model from lasso regression selects only x, x^2,x^5 and x^7. However the coeffecients if x^5 and x^7 are not significant.

(f)

<u>Applying Best Subset Selection:</u>

```
b7=2
Y = b0 + b7*(x^7) + e
fulldata = data.frame(Y,poly(x,10,raw=T))
fullmodel=regsubsets(Y~.,data=fulldata, nvmax=10)
model.summary=summary(fullmodel)
which.min(model.summary$cp)
which.min(model.summary$bic)
which.max(model.summary$adjr2)
```

```
> which.min(model.summary$cp)
[1] 2
> which.min(model.summary$bic)
[1] 1
> which.max(model.summary$adjr2)
[1] 4
```

```
> coefficients(fullmodel,2)
(Intercept)          x2          x7
  1.0704904  -0.1417084   2.0015552
> coefficients(fullmodel,1)
(Intercept)          x7
  0.9589402   2.0007705
> coefficients(fullmodel,4)
(Intercept)          x1          x2          x3          x7
  1.0762524   0.2914016  -0.1617671  -0.2526527   2.0091338
>
```

- It can be observed that BIC picks the most accurate 1 variable model which is very close to the real model as the coefficient estimate is very close to the real one as well as the intercept. Adjusted R-square and Cp methods pick additional unnecessary variables.

Applying Lasso:

```
#Applying Lasso
mat_X = model.matrix(Y~.,data=fulldata)[,-1]
lasso.mod = cv.glmnet(mat_X,Y, alpha=1)
best_lambda = lasso.mod$lambda.min
best_mod=glmnet(mat_X,Y, alpha=1)
preds = predict(best_mod, s=best_lambda, type="coefficients")
```

Results:

```
> best_lambda
[1] 3.879577
```

The best value of lambda chosen by the lasso model is 3.879577.

```
> preds
11 x 1 sparse Matrix of class "dgCMatrix"
                   1
(Intercept) 1.229085
x1          .
x2          .
x3          .
x4          .
x5          .
x6          .
x7          1.936760
x8          .
x9          .
x10         .
```

The lasso method selects only one predictor variable which is the one which exists in the real model, furthermore the estimated coeffecients are close to the real coeffecients. Hence the model which results from lasso regression is a very close estimation of the real model.

Chapter 8 Exercise:

Question 8.

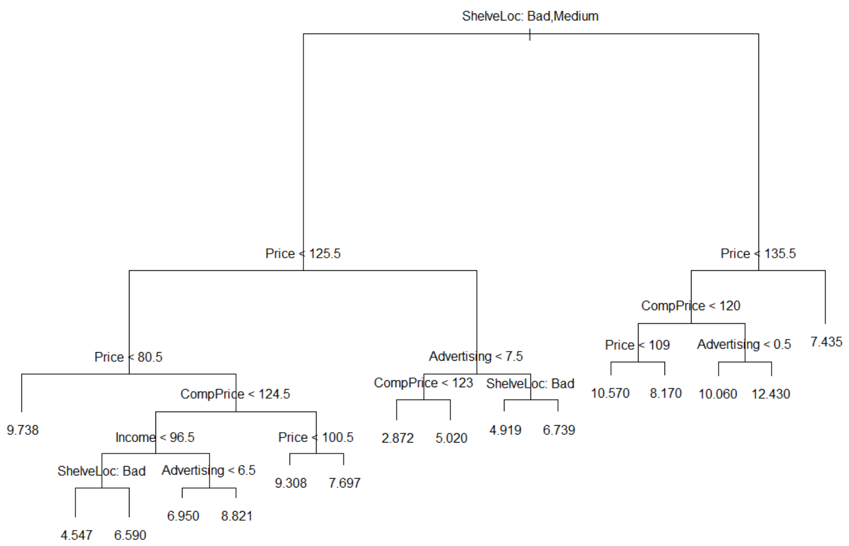
(a) Splitting the data into training and testing data.

```
library(tree)
set.seed(3684)
nrow(Carseats)

#(a) Splitting into training and testing data
train = sample(1:nrow(Carseats), 200)
carseats.train = Carseats[train,]
carseats.test = Carseats[-train,-1]
test.y = Carseats[-train, "Sales"]
```

(b) Fitting a regression tree to the training set and plotting the tree.

```
#(b) Fitting the regression tree to the training set
tree.carseats=tree(Sales~.,data = Carseats,subset=train)

#Plotting the tree
plot(tree.carseats)
text(tree.carseats, pretty=0)
summary(tree.carseats)
```

<u>Resulting regression tree:</u>



Summary of the regression tree:

```
Regression tree:
tree(formula = Sales ~ ., data = Carseats, subset = train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"        "CompPrice"   "Income"       "Advertising"
Number of terminal nodes:  16
Residual mean deviance:  1.92 = 353.3 / 184
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.20000 -0.90000  0.05773  0.00000  0.95170  3.48800
```

The regression tree has 16 terminal nodes.

Finding the test error:

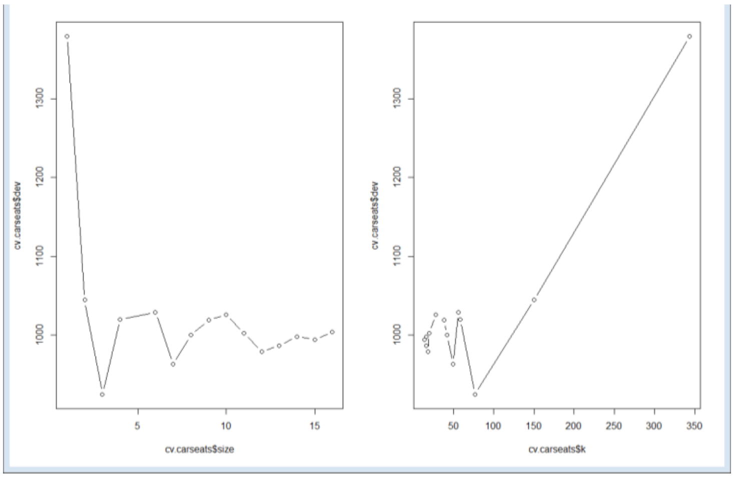
```
#predictions
preds=predict(tree.carseats,carseats.test)
#TEST MSE
test.mse = mean((preds-test.y)^2)
```

```
> test.mse = mean((preds-test.y)^2)
> test.mse
[1] 4.592864
```

- The test error is 4.592864.

(c) Using cross-validation to obtain optimal level of tree complexity.

```
#(c) using cross-validation to obtain optimal level of
cv.carseats = cv.tree(tree.carseats, FUN = prune.tree)
plot(cv.carseats$size, cv.carseats$dev, type="b")
plot(cv.carseats$k, cv.carseats$dev, type="b")
```

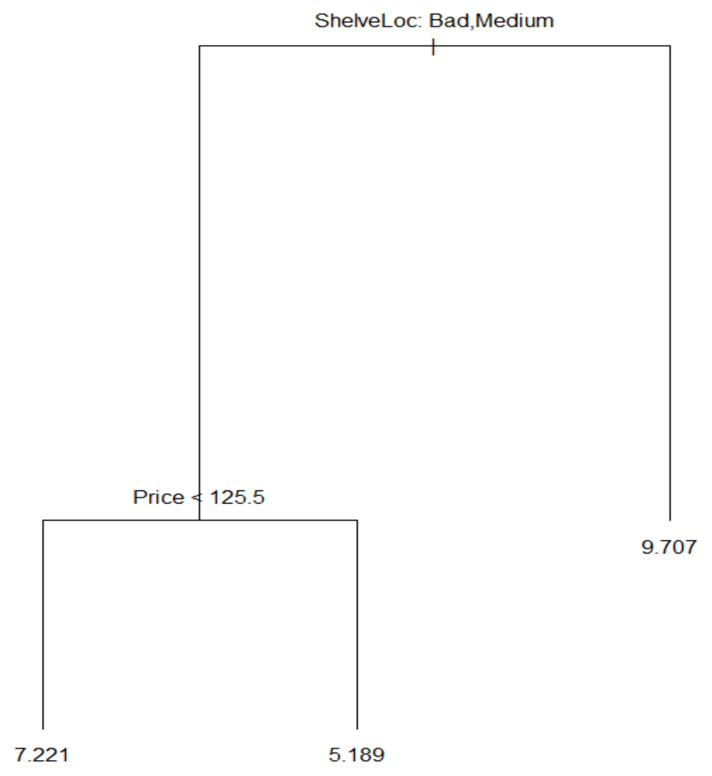


```
> best.tree = which.min(cv.carseats$dev)
> best.tree.size = cv.carseats$size[best.tree]
> best.tree.size
[1] 3
```

- Cross validation results in the tree of size 3.

Let us prune the tree according to the best tree size given by cross validation and calculate the test MSE.

```
#Pruning the tree
carseats.pruned = prune.tree(tree.carseats, best = 3)
plot(carseats.pruned)
text(carseats.pruned,pretty=0)
```



Calculating test MSE on the pruned tree:

```
pruned.preds = predict(carseats.pruned, carseats.test)
pruned.mse = mean((pruned.preds - test.y)^2)
```

```
> pruned.mse
[1] 6.451347
```

The test MSE on the pruned tree is 6.451347 which is higher than the test MSE of the non-pruned tree.

(d) Using the bagging approach to analyse the data and calculating test MSE.

```
#Applying bagging
library(randomForest)
set.seed(3684)
bagged.carseats = randomForest(Sales~., data = Carseats, subset = train, mtry = 10, importance=TRUE)

#Calculating the test error from bagging
bagged.preds = predict(bagged.carseats, carseats.test)
bagged.mse = mean((bagged.preds-test.y)^2)
```

Result:

```
> bagged.mse
[1] 2.664685
```

- The resulting test error from bagging is 2.664685, which is significantly lower as compared to the test error from the original and pruned tree.

```
> importance(bagged.carseats)
              %IncMSE IncNodePurity
CompPrice   25.542665    155.372899
Income       7.630294     74.167675
Advertising  9.251380     73.814568
Population   4.026090     68.051862
Price       49.777716    356.246502
ShelveLoc   58.390677    405.199211
Age         14.158019    105.015983
Education   -2.804762     49.441201
Urban       -1.173205      5.001425
US           2.940843      4.935860
```

- The most important predictors of sale are ShelveLoc, Price and CompPrice.

(e) Using random forest to analyse the data and calculating test MSE.

```
rf.carseats = randomForest(Sales~., data = Carseats, mtry = 3,subset = train, importance = TRUE)
rf.preds = predict (rf.carseats, carseats.test)
rf.mse = mean((rf.preds-test.y)^2)
```

Resulting test MSE:

```
> rf.mse
[1] 2.943595
```

- The resulting test error from random forest method is 2.943595.
- The cross validation error resulting from random forest is higher than the cross validation error resulting from bagging.
- When we used the bagging approach, the number of variables tried, 'm' is 10 and the error rate is 2.664685 and when we use the random forest technique, where the value of 'm' is set to 3 (as $\sqrt{10} \approx 3$), the error rate is 2.943595. Hence the error rate increases as the value of m is decreased from 10 to 3.
  Running random forest with m=3 and 10 5 times. In each iteration, the error for m=3 is higher than the error for m=10.

```
> for (i in 1:5)
+ {
+   print(c("Iteration: ", i))
+   rf.carseats3 = randomForest(Sales~., data = Carseats, mtry = 3,subset = train, importance = TRUE)
+   rf.preds3 = predict (rf.carseats3, carseats.test)
+   rf.mse3 = mean((rf.preds3-test.y)^2)
+   print(c("3: ",rf.mse3))
+   rf.carseats10 = randomForest(Sales~., data = Carseats, mtry = 10,subset = train, importance = TRUE)
+   rf.preds10 = predict (rf.carseats10, carseats.test)
+   rf.mse10 = mean((rf.preds10-test.y)^2)
+   print(c("10: ",rf.mse10))
+   print("#########")
+
+
+ }
[1] "Iteration: " "1"
[1] "3: "                "3.0273266081782"
[1] "10: "               "2.66311586699701"
[1] "#########"
[1] "Iteration: " "2"
[1] "3: "                "3.01466102597222"
[1] "10: "               "2.64034624503806"
[1] "#########"
[1] "Iteration: " "3"
[1] "3: "                "2.96265090006855"
[1] "10: "               "2.68092986649158"
[1] "#########"
[1] "Iteration: " "4"
[1] "3: "                "2.95005527599708"
[1] "10: "               "2.69042306766772"
[1] "#########"
[1] "Iteration: " "5"
[1] "3: "                "2.96434434135938"
[1] "10: "               "2.72522113606118"
[1] "#########"
```

```
> importance(rf.carseats)
              %IncMSE IncNodePurity
CompPrice   12.9960360     137.38725
Income       5.6501137     106.98400
Advertising  4.1343059      87.55082
Population   0.9700867      90.72569
Price       31.4956729     292.03481
ShelveLoc   37.7140375     302.35280
Age          9.1710954     139.87162
Education    0.8813956      70.19944
Urban       -0.6698732      11.33560
US           2.2479087      14.21044
```

- The importance variables in the prediction of Sales are <u>ShelveLoc</u> and <u>Price.</u>