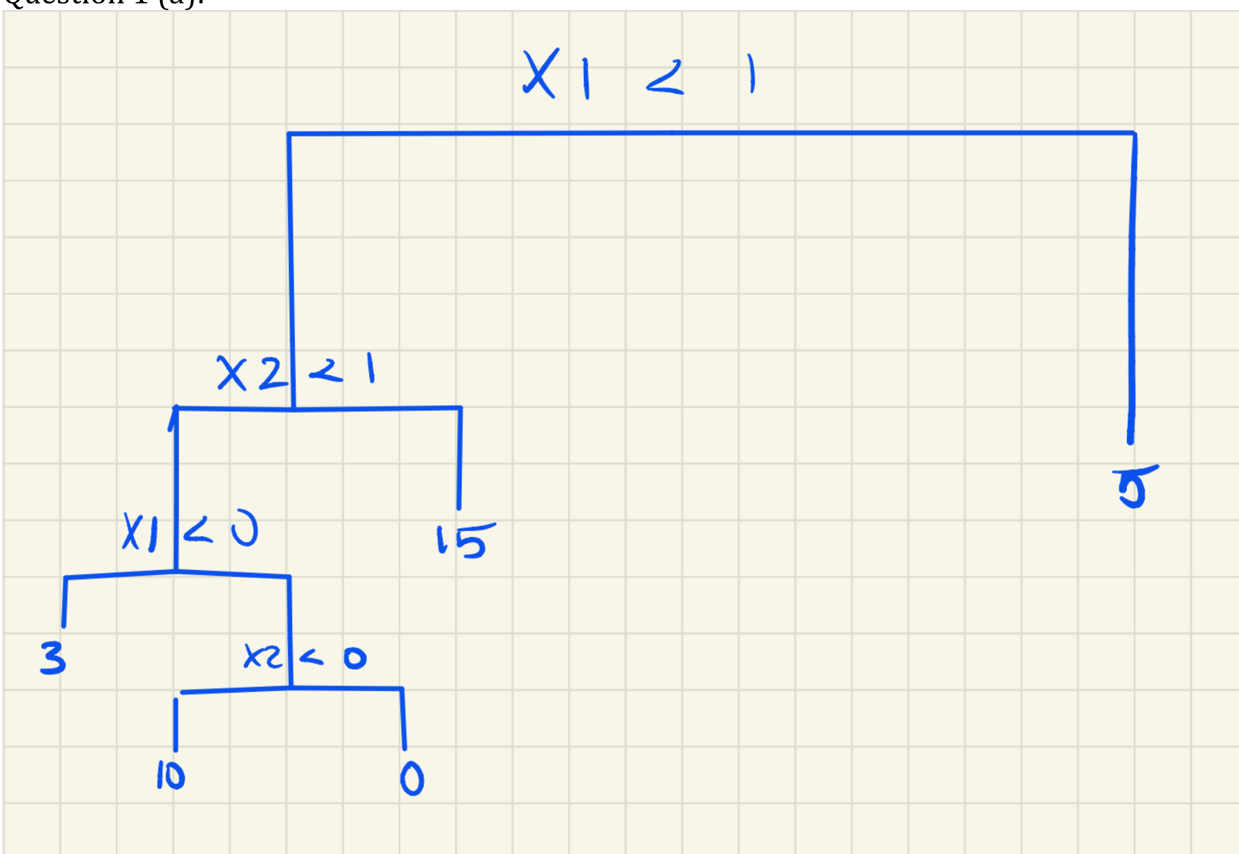
HW5-MATH4322

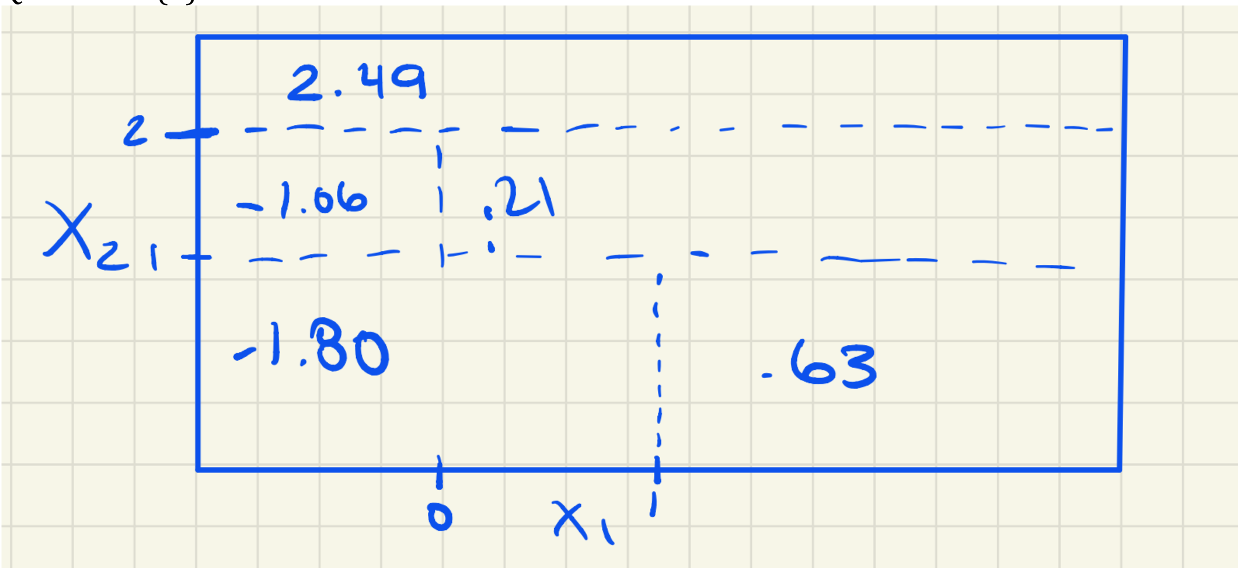
Anthony Castillo-ID:1670011

2022-11-04

Question 1 (a):



Question 1 (b):



Question 2:

```
x <- c(0.1,0.15,0.2,0.2,0.55,0.6,0.6,0.65,0.7,0.75)
mean(x)

## [1] 0.45
```

The majority vote approach will have the final classification be Red because there are six values greater than 0.5 making them red and only 4 values less than 0.5 making them green The average probability approach will have the final classification of Green since the average is less than 0.5

Question 3:

1. use recursive binary splitting to grow tree on the training data
2. apply pruning to the tree to obtain the best sequence of best subtrees
3. use k-fold cross validation to prune the tree
4. average the results for each value and choose the best value
5. return subtree from step 2 that corresponds to chosen value

Question 4 (a):

```
library(ISLR2)
n <- nrow(OJ)
Train <- sample(1:n,800)
oj.train <- OJ[Train,]
oj.test <- OJ[-Train,]
```

Question 4 (b):

```r
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"     "ListPriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7765 = 615.7 / 793
## Misclassification error rate: 0.1662 = 133 / 800
```

The training error rate is 0.1962 and it has 8 terminal nodes

Question 4 (c):

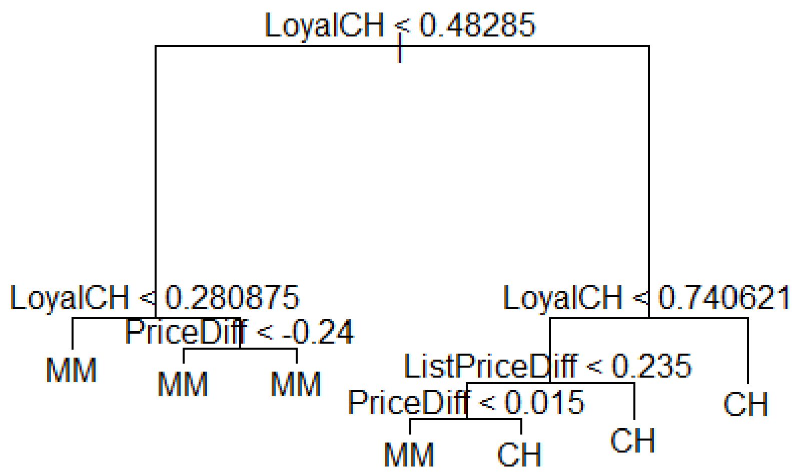
```r
oj.tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.00 CH ( 0.60625 0.39375 )
##    2) LoyalCH < 0.48285 301  328.90 MM ( 0.23588 0.76412 )
##      4) LoyalCH < 0.280875 163  113.30 MM ( 0.11043 0.88957 ) *
##      5) LoyalCH > 0.280875 138  183.80 MM ( 0.38406 0.61594 )
##       10) PriceDiff < -0.24 15    0.00 MM ( 0.00000 1.00000 ) *
##       11) PriceDiff > -0.24 123  168.20 MM ( 0.43089 0.56911 ) *
##    3) LoyalCH > 0.48285 499  455.50 CH ( 0.82966 0.17034 )
##      6) LoyalCH < 0.740621 220  276.70 CH ( 0.67727 0.32273 )
##       12) ListPriceDiff < 0.235 92  126.40 MM ( 0.44565 0.55435 )
##         24) PriceDiff < 0.015 49   56.70 MM ( 0.26531 0.73469 ) *
##         25) PriceDiff > 0.015 43   55.62 CH ( 0.65116 0.34884 ) *
##       13) ListPriceDiff > 0.235 128  111.00 CH ( 0.84375 0.15625 ) *
##      7) LoyalCH > 0.740621 279  111.10 CH ( 0.94982 0.05018 ) *
```

Terminal Node 22 has a split of LoyalCH < 0.277977 and the deviance is 58.63 and is predicted for value MM which has a probability of 71% of being value MM and 29% of being value CH

Question 4 (d):

```r
plot(oj.tree)
text(oj.tree)
```



The value CH has a higher chance of being predicted the value MM

Question 4 (e):

```
oj.pred <- predict(oj.tree,newdata = oj.test, type = "class")
table(pred=oj.pred,true=oj.test$Purchase)

##     true
## pred  CH  MM
##   CH 139  17
##   MM  29  85

#test error rate
mean(oj.pred != oj.test$Purchase)

## [1] 0.1703704
```

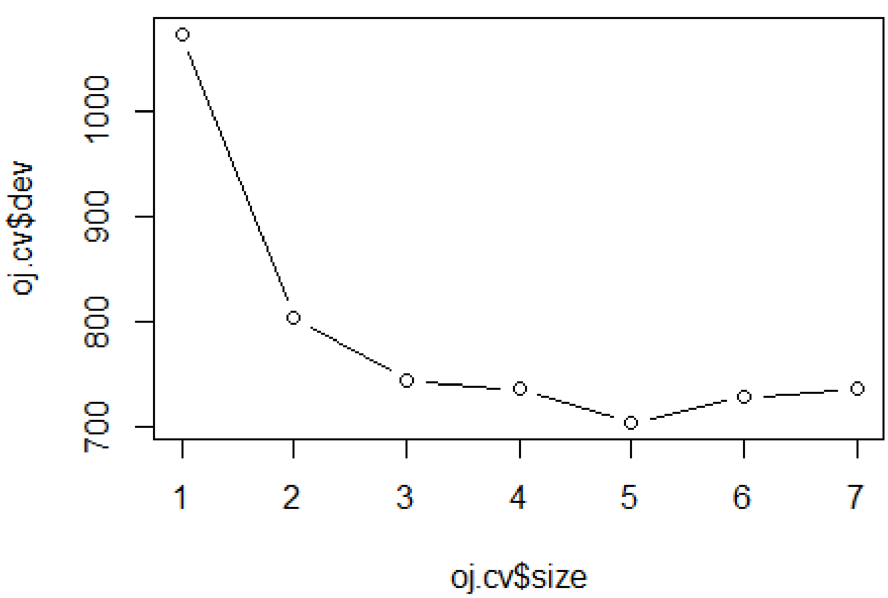
Question 4 (f):

```
set.seed(10)
oj.cv <- cv.tree(oj.tree)
oj.cv

## $size
## [1] 7 6 5 4 3 2 1
##
## $dev
```

```
## [1]  735.6377  728.0531  703.7670  735.5013  744.5177  803.3631 1074.0156
##
## $k
## [1]      -Inf  14.13538  15.66329  31.78660  39.31949  67.72780 288.25708
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

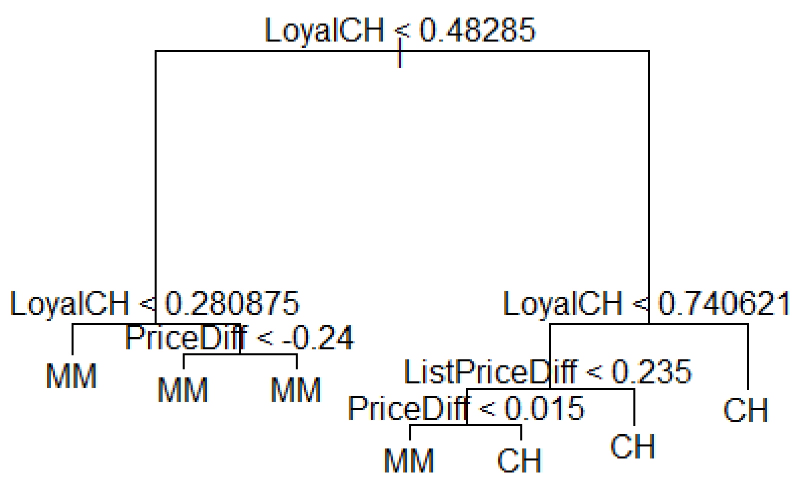
Question 4 (g):

```
plot(oj.cv$size,oj.cv$dev,type="b")
```



Question 4 (h):
The lowest cross-validated classification error rate is for tree size = 7
Question 4 (i):

```
oj.prune <- prune.tree(oj.tree, best = 7)
plot(oj.prune)
text(oj.prune)
```



Question 4 (j):

```
summary(oj.prune)

##
## Classification tree:
## tree(formula = Purchase ~ ., data = oj.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7765 = 615.7 / 793
## Misclassification error rate: 0.1662 = 133 / 800

summary(oj.tree)

##
## Classification tree:
## tree(formula = Purchase ~ ., data = oj.train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "ListPriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7765 = 615.7 / 793
## Misclassification error rate: 0.1662 = 133 / 800
```

The pruned tree has a higher training error rate
Question 4 (k):

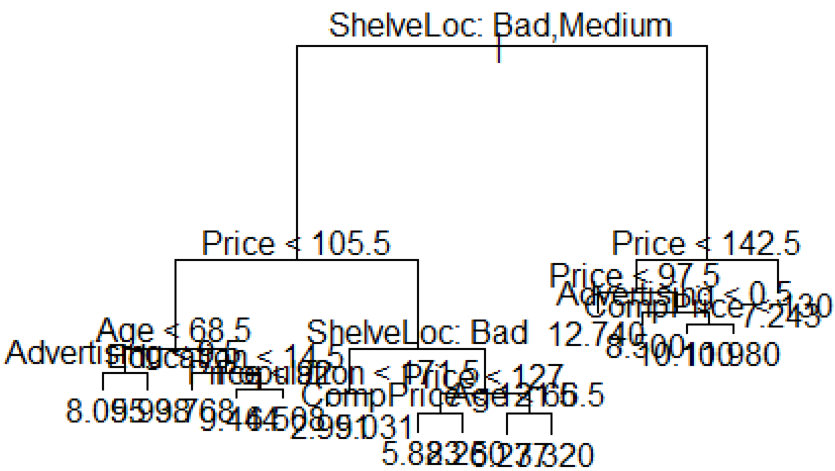
```
oj.prune.pred <- predict(oj.prune, newdata = oj.test, type="class")
table(pred=oj.prune.pred,true=oj.test$Purchase)

##      true
## pred  CH  MM
##   CH 139  17
##   MM  29  85

#pruned tree test error rate
mean(oj.prune.pred != oj.test$Purchase)

## [1] 0.1703704
```

The pruned tree has a higher test error rate
Question 5 (a):

```
library(ISLR)

##
## Attaching package: 'ISLR'

## The following objects are masked from 'package:ISLR2':
##
##     Auto, Credit

n <- nrow(Carseats)
Train <- sample(1:n,0.70*n)
carseats.train <- Carseats[Train,]
carseats.test <- Carseats[-Train,]
```

Question 5 (b):

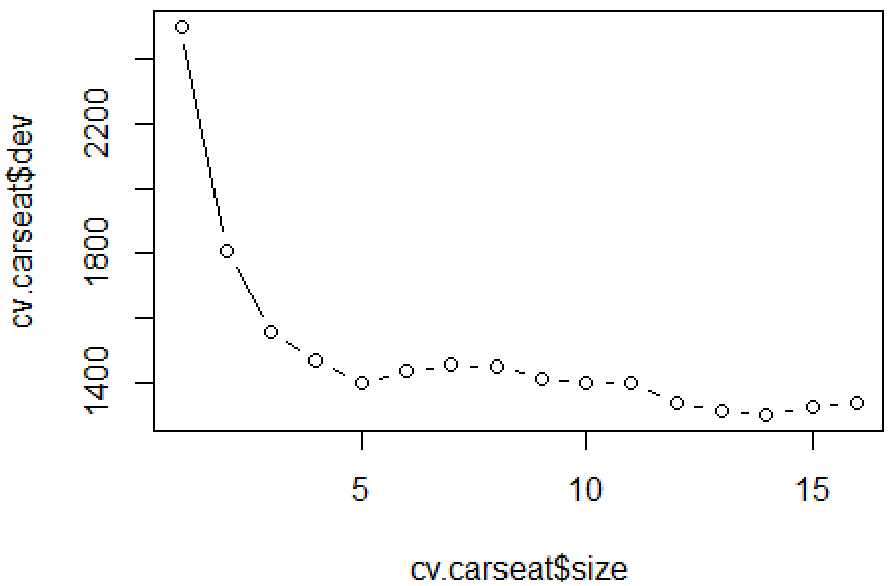
```
library(tree)
carseats.tree <- tree(Sales~., data = carseats.train)
plot(carseats.tree)
text(carseats.tree, pretty = 0)
```




```
summary(carseats.tree)

##
## Regression tree:
## tree(formula = Sales ~ ., data = carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Advertising" "Education"
## [6] "Population"  "CompPrice"
## Number of terminal nodes:  16
## Residual mean deviance:  2.369 = 625.4 / 264
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.6630 -0.9307 -0.0675  0.0000  0.9643  5.1090

#MSE
carseats.pred <- predict(carseats.tree, newdata = carseats.test)
mean((carseats.pred-carseats.test$Sales)^2)

## [1] 4.756139
```

Question 5 (c):

```
set.seed(11)
cv.carseat <- cv.tree(carseats.tree)
plot(cv.carseat$size,cv.carseat$dev, type="b")
```




```
prune.carseat <- prune.tree(carseats.tree,best = 15)
carseat.pred.pruned <- predict(prune.carseat, newdata = carseats.test)
mean((carseat.pred.pruned-carseats.test$Sales)^2)

## [1] 4.793973
```

Yes pruning the tree improved the test MSE
Question 5 (d):

```
library(randomForest)

## randomForest 4.7-1.1

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

set.seed(10)
car.bag <- randomForest(Sales~., data = carseats.train,mtry = ncol(Carseats)-
1, importance = TRUE)
car.bag

##
## Call:
##  randomForest(formula = Sales ~ ., data = carseats.train, mtry = ncol(Cars
eats) -      1, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
```

```
##
##           Mean of squared residuals: 2.483044
##                     % Var explained: 72.01
```

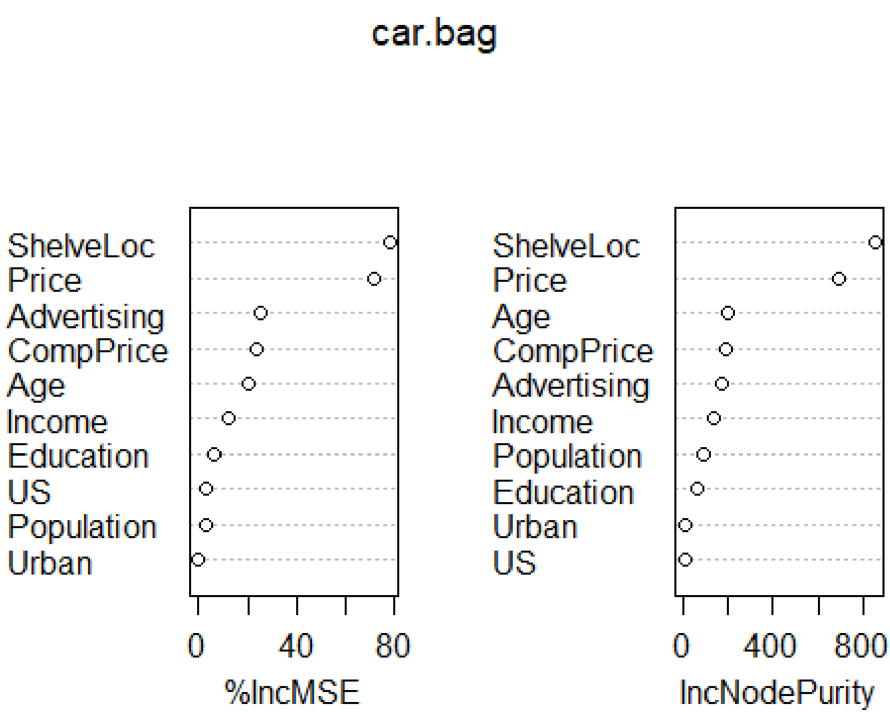
```r
#test MSE
car.bagpred <- predict(car.bag ,newdata = carseats.test)
mean((car.bagpred-carseats.test$Sales)^2)
```

```
## [1] 2.580592
```

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

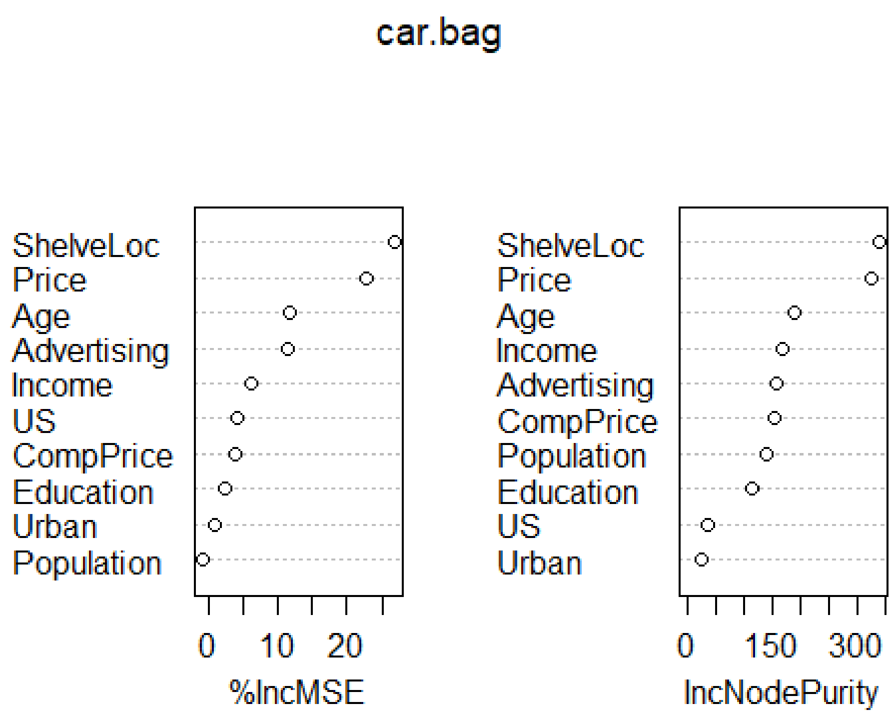
```
##               %IncMSE IncNodePurity
## CompPrice   23.974062    195.587895
## Income      11.942030    133.716750
## Advertising 25.535646    173.320606
## Population   2.711433     88.995413
## Price       71.662258    689.126045
## ShelveLoc   78.376524    854.486203
## Age         20.279028    202.421144
## Education    6.803589     69.914656
## Urban       -0.561290     10.643511
## US           2.778464      9.561316
```

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

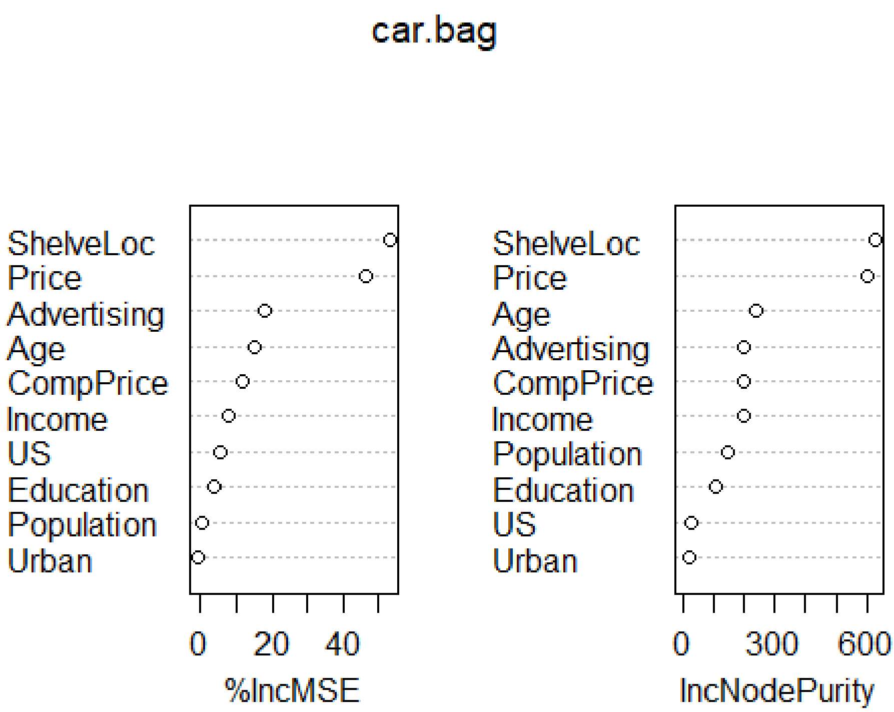


Price and ShelveLoc are the most important variables
Question 5 (e):

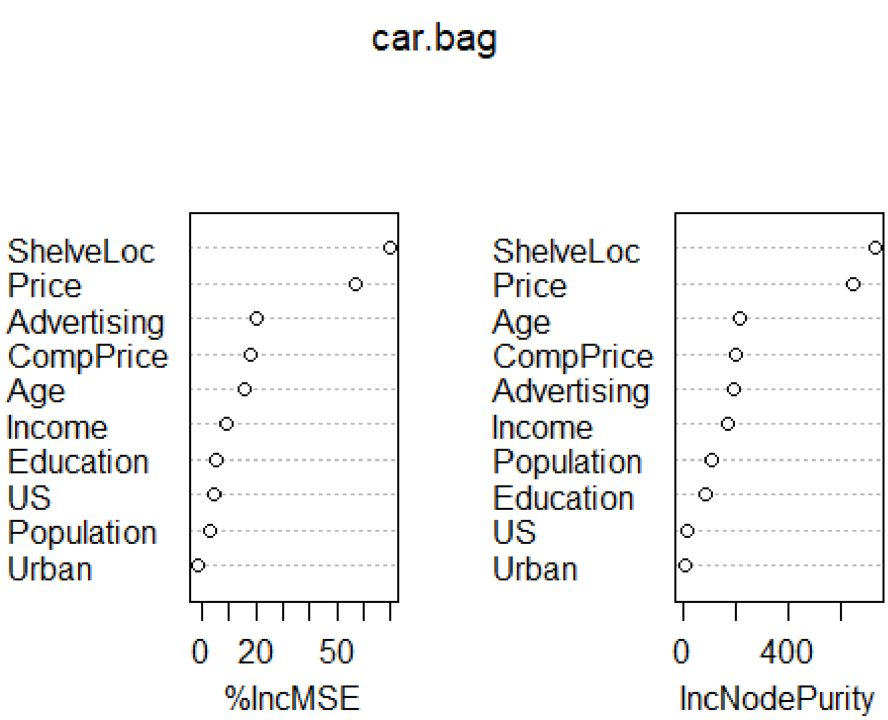
```
for(i in c(1,3,5)){
  set.seed(10)
  car.bag <- randomForest(Sales~., data = carseats.train,
                          mtry = i,
                          importance = TRUE)
  #print(importance(car.bag))
  varImpPlot(car.bag)

  car.bagpred <- predict(car.bag ,newdata = carseats.test)
  print(mean((car.bagpred-carseats.test$Sales)^2))
}
```




```
## [1] 3.505526
```




```
## [1] 2.471869
```



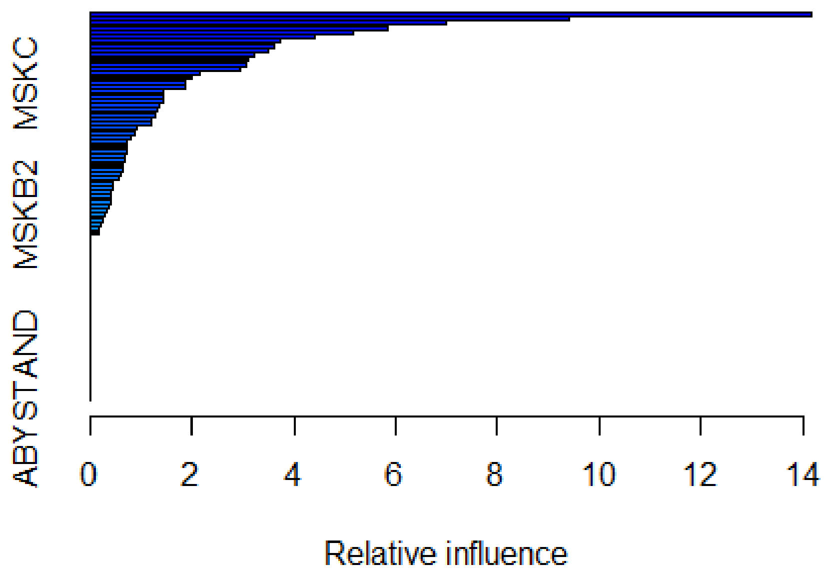

```
## [1] 2.476211
```

Price and ShelveLoc stay the most important variable throughout all the m's
The effect of m as it is increased the MSE test error decreases with each m
Question 6 (a):

```
library(ISLR2)
Train <- sample(1:1000)
Caravan$Purchase = ifelse(Caravan$Purchase == "Yes", 1, 0)
caravan.train <- Caravan[Train,]
caravan.test <- Caravan[-Train,]
```

Question 6 (b):

```
library(gbm)

## Loaded gbm 2.1.8.1

set.seed(1)
caravan.boost <- gbm(Purchase~., data = caravan.train,
                     distribution ="gaussian",
                     n.tree = 1000,
                     shrinkage = 0.01,)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribut
ion, :
## variable 50: PVRAAUT has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribut
ion, :
## variable 71: AVRAAUT has no variation.

summary(caravan.boost)
```




```
##                  var    rel.inf
## PPERSAUT PPERSAUT 14.1789945
## MKOOPKLA MKOOPKLA  9.4143251
## MOPLHOOG MOPLHOOG  6.9794184
## MBERMIDD MBERMIDD  5.8329731
## PBRAND     PBRAND  5.1588061
## MGODGE     MGODGE  4.4264998
## MOSTYPE   MOSTYPE  3.7376459
## MINK3045 MINK3045  3.6212597
## ABRAND     ABRAND  3.4895489
## MAUT1       MAUT1  3.2085799
## MAUT2       MAUT2  3.1092005
## MSKA         MSKA  3.0582519
## PWAPART   PWAPART  2.9361318
## MGODPR     MGODPR  2.1424605
## MBERARBG MBERARBG  1.9768421
## MSKC         MSKC  1.8684321
## MBERHOOG MBERHOOG  1.8637551
## MINK7512 MINK7512  1.4403762
## PBYSTAND PBYSTAND  1.4359120
## MINKGEM   MINKGEM  1.4300349
## MGODOV     MGODOV  1.3682377
## MFWEKIND MFWEKIND  1.3047131
## MRELOV     MRELOV  1.2748401
## MFGEKIND MFGEKIND  1.1939343
## MHHUUR     MHHUUR  1.1805029
## MSKB1       MSKB1  0.9297678
```

```
## MRELGE      MRELGE  0.8651113
## MGODRK      MGODRK  0.7833458
## MZPART      MZPART  0.7385279
## MSKD          MSKD  0.7201284
## MBERARBO MBERARBO  0.7187453
## APERSAUT APERSAUT  0.6886313
## MOPLMIDD MOPLMIDD  0.6739193
## MGEMOMV   MGEMOMV  0.6377615
## MINKM30   MINKM30  0.6265648
## MZFONDS   MZFONDS  0.6196066
## MAUT0       MAUT0  0.5649730
## MHKOOP     MHKOOP  0.4374713
## MOSHOOFD MOSHOOFD  0.4315327
## MINK4575 MINK4575  0.4136190
## MGEMLEEF MGEMLEEF  0.4132185
## MINK123M MINK123M  0.3967414
## MRELSA     MRELSA  0.3455787
## MBERBOER MBERBOER  0.3095742
## MSKB2       MSKB2  0.2661275
## PMOTSCO   PMOTSCO  0.2355198
## MBERZELF MBERZELF  0.2142473
## MFALLEEN MFALLEEN  0.1690279
## MOPLLAAG MOPLLAAG  0.1685822
## MAANTHUI MAANTHUI  0.0000000
## PWABEDR   PWABEDR  0.0000000
## PWALAND    PWALAND  0.0000000
## PBESAUT   PBESAUT  0.0000000
## PVRAAUT   PVRAAUT  0.0000000
## PAANHANG PAANHANG  0.0000000
## PTRACTOR PTRACTOR  0.0000000
## PWERKT     PWERKT  0.0000000
## PBROM       PBROM  0.0000000
## PLEVEN     PLEVEN  0.0000000
## PPERSONG PPERSONG  0.0000000
## PGEZONG   PGEZONG  0.0000000
## PWAOREG   PWAOREG  0.0000000
## PZEILPL   PZEILPL  0.0000000
## PPLEZIER PPLEZIER  0.0000000
## PFIETS     PFIETS  0.0000000
## PINBOED   PINBOED  0.0000000
## AWAPART   AWAPART  0.0000000
## AWABEDR   AWABEDR  0.0000000
## AWALAND   AWALAND  0.0000000
## ABESAUT   ABESAUT  0.0000000
## AMOTSCO   AMOTSCO  0.0000000
## AVRAAUT   AVRAAUT  0.0000000
## AAANHANG AAANHANG  0.0000000
## ATRACTOR ATRACTOR  0.0000000
## AWERKT     AWERKT  0.0000000
## ABROM       ABROM  0.0000000
```

```
## ALEVEN      ALEVEN  0.0000000
## APERSONG APERSONG  0.0000000
## AGEZONG   AGEZONG  0.0000000
## AWAOREG   AWAOREG  0.0000000
## AZEILPL   AZEILPL  0.0000000
## APLEZIER APLEZIER  0.0000000
## AFIETS     AFIETS  0.0000000
## AINBOED   AINBOED  0.0000000
## ABYSTAND ABYSTAND  0.0000000
```

Question 6 (c):

```
set.seed(10)
pred.carboost <- predict(caravan.boost,
                         newdata = caravan.test,
                         n.trees = 1000)
greater20 <- ifelse(pred.carboost > .20,1,0)
table(pred = greater20,true= caravan.test$Purchase)

##     true
## pred    0    1
##    0 4501  279
##    1   32   10

#predicted to make purchase and in fact make one
print(10/(10+279))

## [1] 0.03460208
```