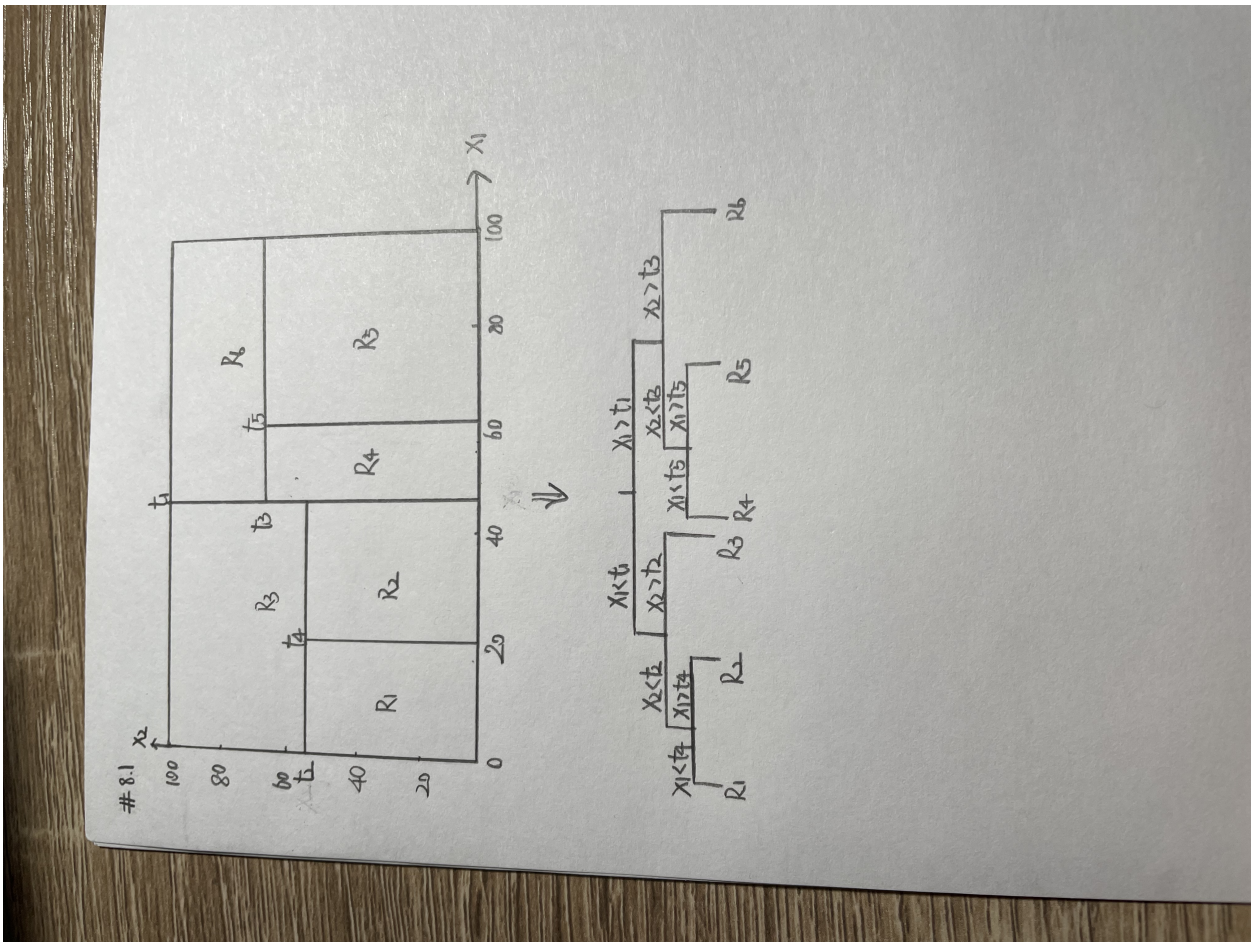
Chapter 8

Tong Sun

2/27/2022

8.1

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions R_1, R_2,..., the cutpoints t_1, t_2,..., and so forth. Hint: Your result should look something like Figures 8.1 and 8.2.



8.2 It is mentioned in Section 8.2.3 that boosting using depth-one trees (or stumps) leads to an additive model: that is, a model of the form $f(X) = \sum_{j=1}^{p} f_j(X_j)$. Explain why this is the case. You can begin with (8.12) in Algorithm 8.2.

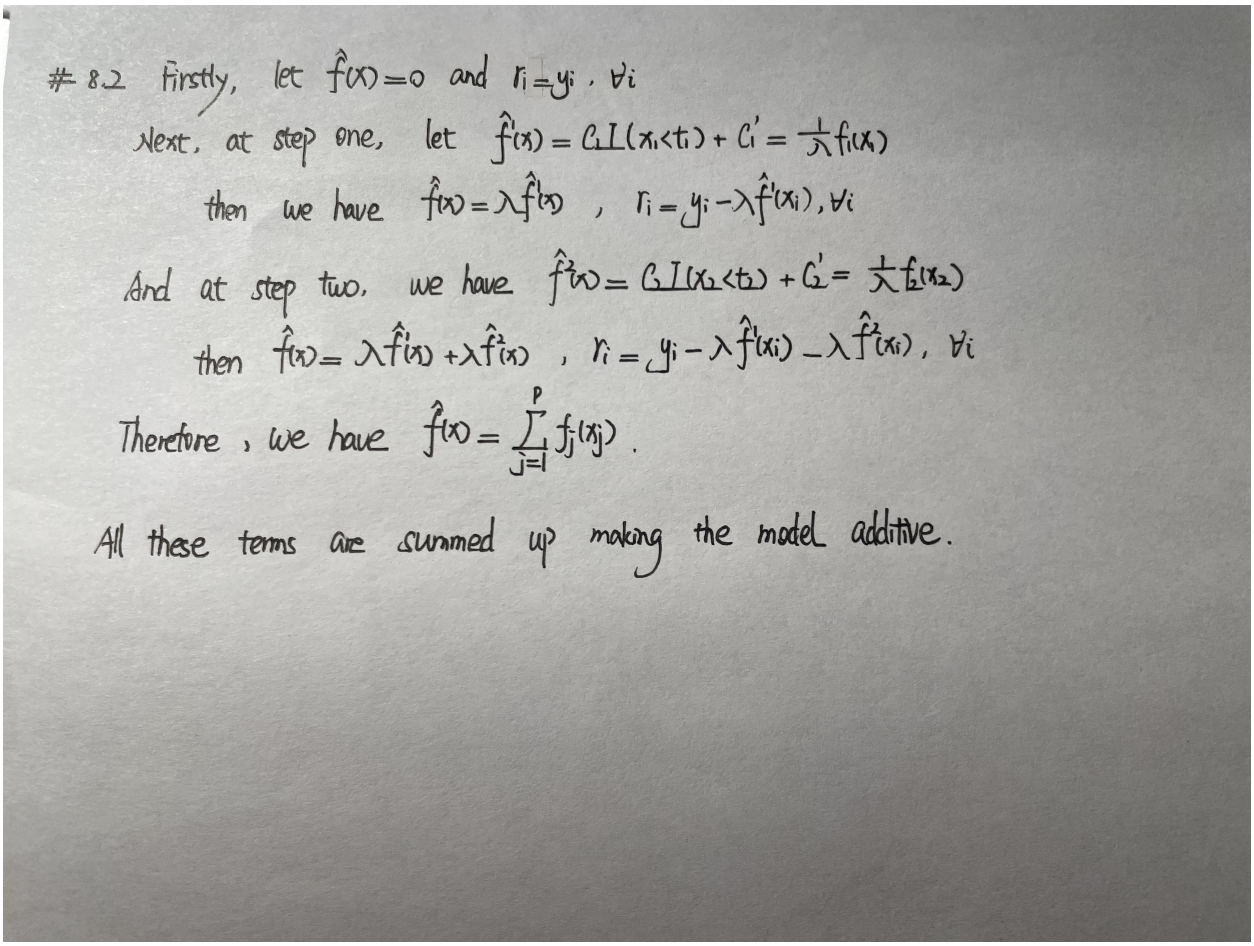
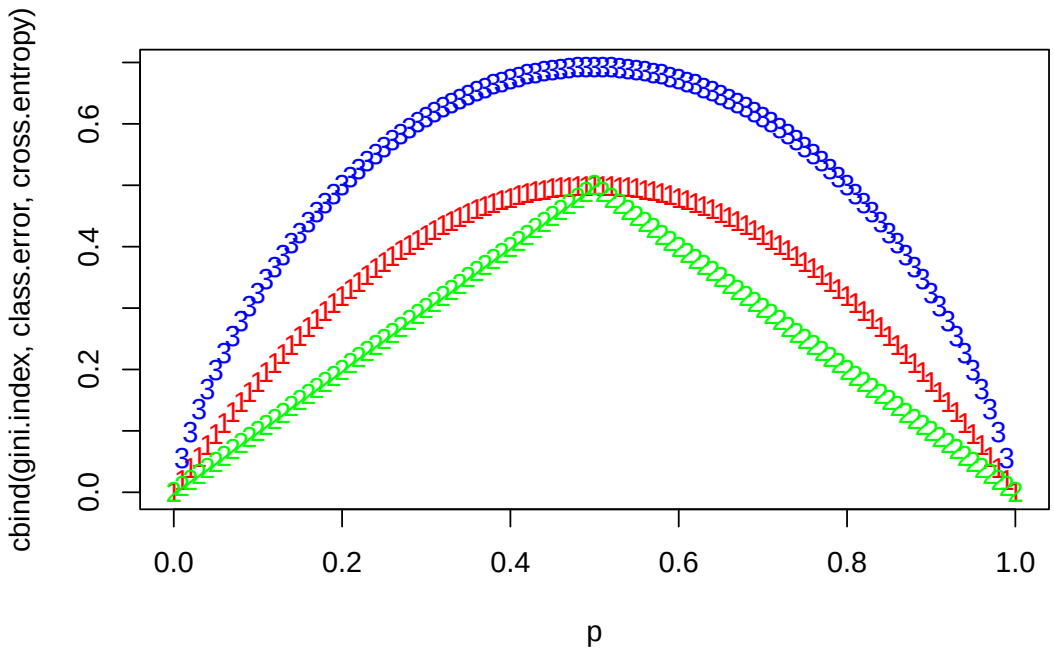


Figure 1: my answer

8.3

Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of $\hat{p}_{m1}$ The x axis should display $\hat{p}_{m1}$, ranging from 0 to 1, and the y axis should display the value of the Gini index, classification error, and entropy. Hint: In a setting with two classes, $\hat{p}_{m1} = 1 - \hat{p}_{m2}$. You could make this plot by hand, but it will be much easier to make in R.



8.5

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red): 0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75. There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

With the majority vote approach, the conclusion of this question is X is red because there are 6 for red and 4 for green among these 10 predictions. And with the average probability approach, the conclusion is that X is green, because the average of these 10 probabilities is $\frac{0.1+0.15+0.2+0.2+0.55+0.6+0.6+0.65+0.7+0.75}{10} = 0.45$, which is lower than 0.5.

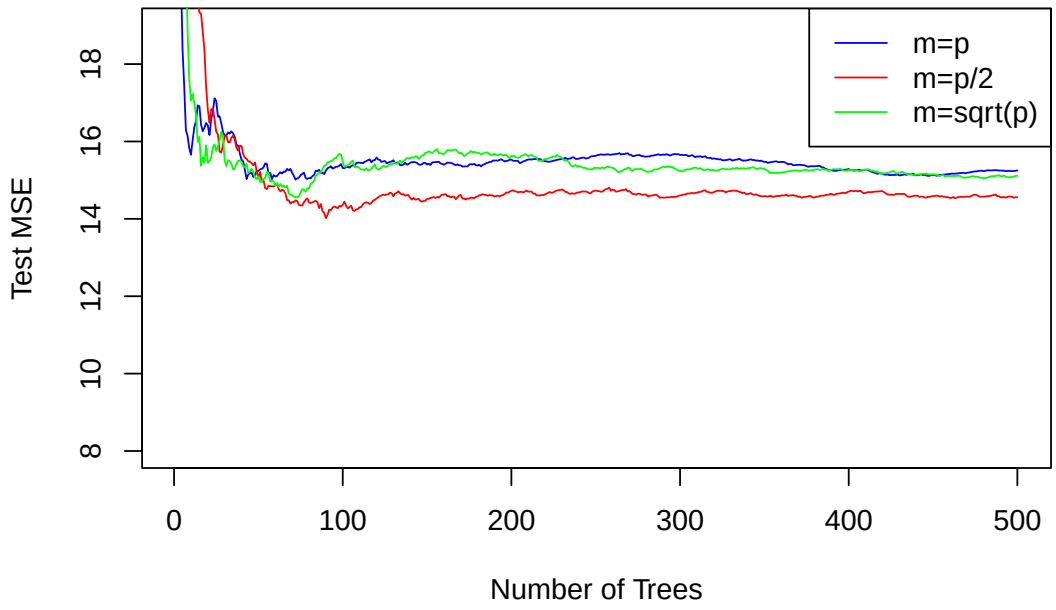
8.7

In the lab, we applied random forests to the Boston data using "mtry = 6" and using "ntree = 25" and "ntree = 500". Create a plot displaying the test error resulting from random forests on this data set for a more comprehensive range of values for "mtry" and "ntree". You can model your plot after Figure 8.10. Describe the results obtained.

```
## Warning: package 'randomForest' was built under R version 4.1.2
```

```
## randomForest 4.7-1
```

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



Results from random forests for the data set with p = 500 predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node. Random forests ($m = p/2$) lead to a slight improvement over bagging (m=p). We may also see that the Test MSE is very high for a single tree, it decreases as the number of trees increases. But from this plot, the difference between these three Test MSEs are not so significant although the Test MSE for all predictors is higher than for half the predictors or the square root of the number of predictors.

8.8

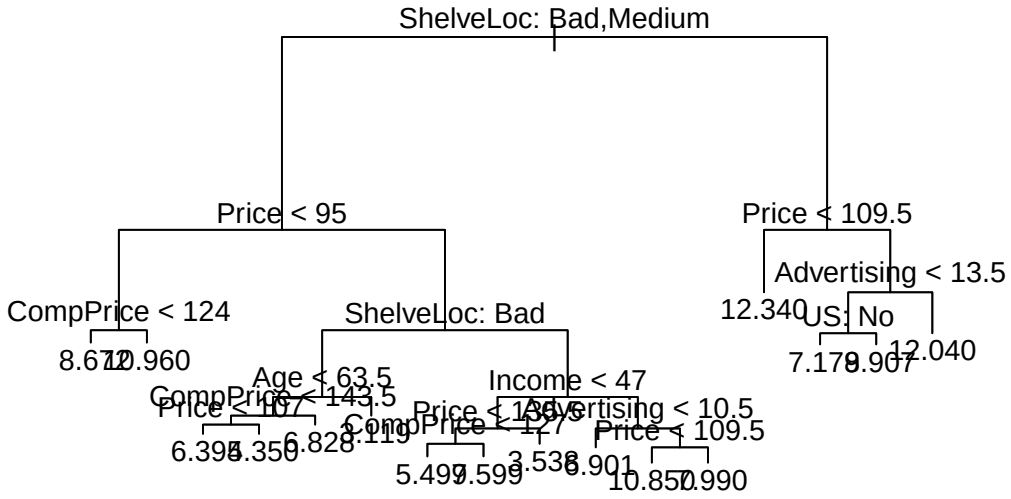
In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable. ##(a) Split the data set into a training set and a test set.

##(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
```

```
## [1] "ShelveLoc"   "Price"       "CompPrice"   "Age"         "Income"
## [6] "Advertising" "US"
## Number of terminal nodes:  16
## Residual mean deviance:  2.32 = 426.8 / 184
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.5990 -1.0440  0.1321  0.0000  0.9857  3.6230
```

Here I fit a regression tree to the Carseats data set. First, I create a training set and fit the tree to the training data. The output of "summary()" indicates that only seven variables have been used in constructing the tree. The deviance is simply the sum of squared errors for the tree.

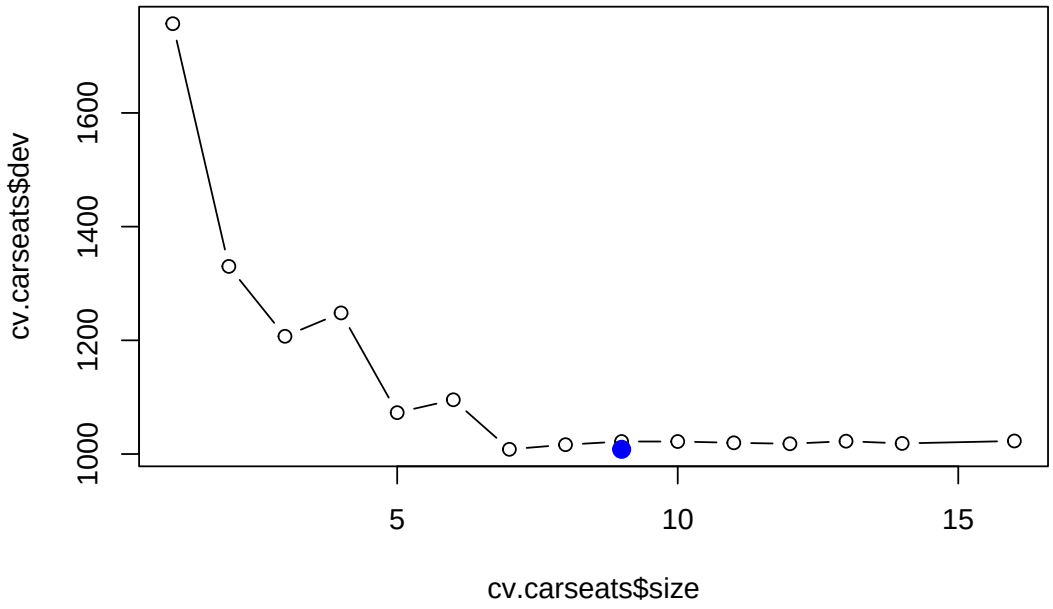


From the regression tree plot, the split at the top of the tree results in two large branches. The left-hand branch corresponds to "ShelveLoc" in "Good" level, and the right-hand branch corresponds to "ShelveLoc" in "Bad and Medium" level. And also the predictor "ShelveLoc" is the most important factor in determining "Sales". The number in each leaf is the mean of the response for the observations that fall there.

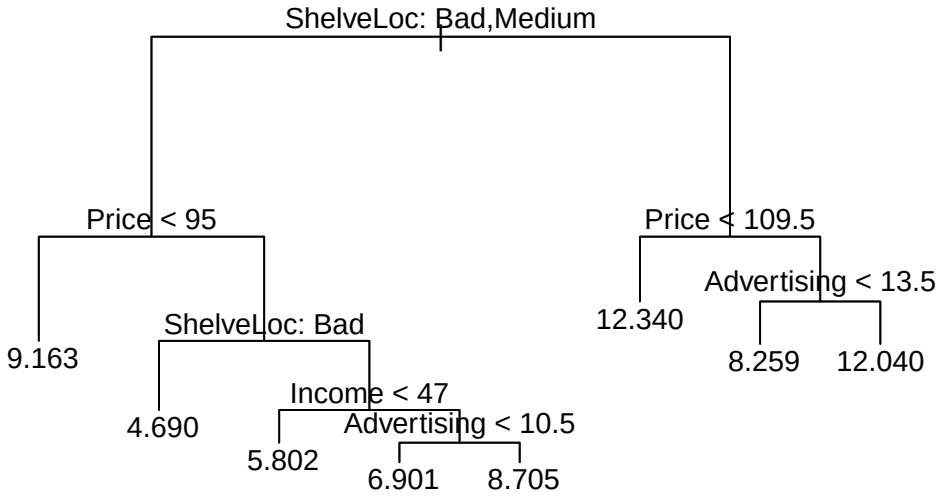
```
## [1] 4.347222
```

The conclusion is that the Test MSE is about 4.35.

##(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?



The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance. This is because the resulting tree might be too complex. So here I use the "cv.tree()" function to see whether pruning the tree will improve performance.In this case, the most complex tree under consideration is selected by cross-validation. In this case, the tree of size 8 is selected by cross-validation. We now prune the tree to obtain the 8-node tree.




```
## [1] 4.681248
```

I find that pruning the tree increases the Test MSE to 4.68.

##(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

```
## [1] 2.485188
```

I find that bagging decreases the Test MSE to 2.48.

```
##               %IncMSE IncNodePurity
## CompPrice   25.442967    158.707625
## Income       5.569262     86.013566
## Advertising 16.398206    112.148189
## Population  -1.604064     56.711805
## Price       56.438246    494.090382
## ShelveLoc   67.026774    504.428854
## Age         18.956823    150.990810
## Education    1.144853     46.853734
## Urban        1.740876      5.303027
## US           2.871592      6.775507
```

We may conclude that “ShelveLoc” and “Price” are the two most important variables.

##(e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```
## [1] 2.937091
```

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, here I build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.The split is allowed to use only one of those m predictors.In this case, with m = $\sqrt{p}$, we have a Test MSE of 2.94, which leads to a reduction in test error.

```
##                %IncMSE IncNodePurity
## CompPrice   12.7464190     143.77900
## Income       3.4460212     140.24039
## Advertising 13.0069046     131.91187
## Population  -1.7518578     110.33644
## Price       35.3671612     370.20432
## ShelveLoc   40.1076881     364.96470
## Age         12.4105107     186.17048
## Education    4.0757034      78.82914
## Urban       -0.4882708      15.74092
## US           3.1487631      24.21342
```

In this case, the two most important variables are also the "Price" and "ShelveLoc".

##(f) Now analyze the data using BART, and report your results.

```
## Loading required package: nlme
```

```
## Loading required package: nnet
```

```
## Loading required package: survival
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 200, 14, 200
## y1,yn: 0.203600, -3.706400
## x1,x[n*p]: 121.000000, 1.000000
## xp1,xp[np*p]: 138.000000, 1.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 58 ... 1
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.276302,3,0.186883,7.5964
## *****sigma: 0.979491
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,14,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
```

```
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 3s
## trcnt,tecnt: 1000,1000
```

```
## [1] 1.616522
```

On this data set, the test error of BART is 1.62, lower than the test error of random forests and boosting. Next I check how many times each variable appeared in the collection of trees.

```
##       Price   CompPrice  ShelveLoc1  ShelveLoc2         Age         US2
##      23.663      19.743      18.492      18.201      17.650      16.419
##         US1  Population      Urban1  ShelveLoc3      Urban2 Advertising
##      15.941      15.619      15.546      15.482      15.458      15.285
##   Education      Income
##      15.007      13.714
```

8.11

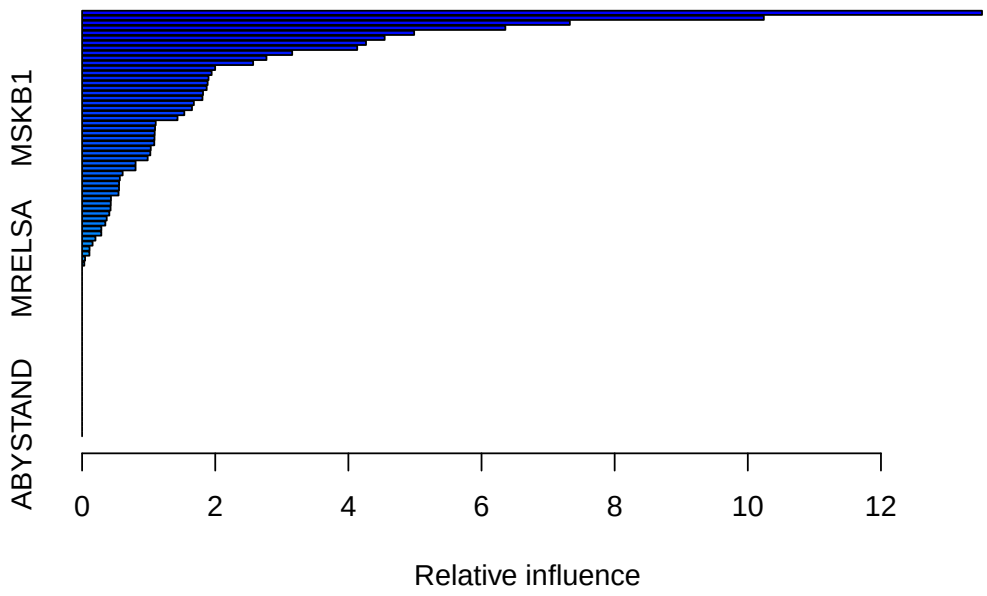
This question uses the Caravan data set. ##(a) Create a training set consisting of the first 1,000 observations,and a test set consisting of the remaining observations.

##(b) Fit a boosting model to the training set with "Purchase" as the response and the other variables as predictors. Use 1,000 trees,and a shrinkage value of 0.01. Which predictors appear to be the most important?

```
## Loaded gbm 2.1.8
```

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

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

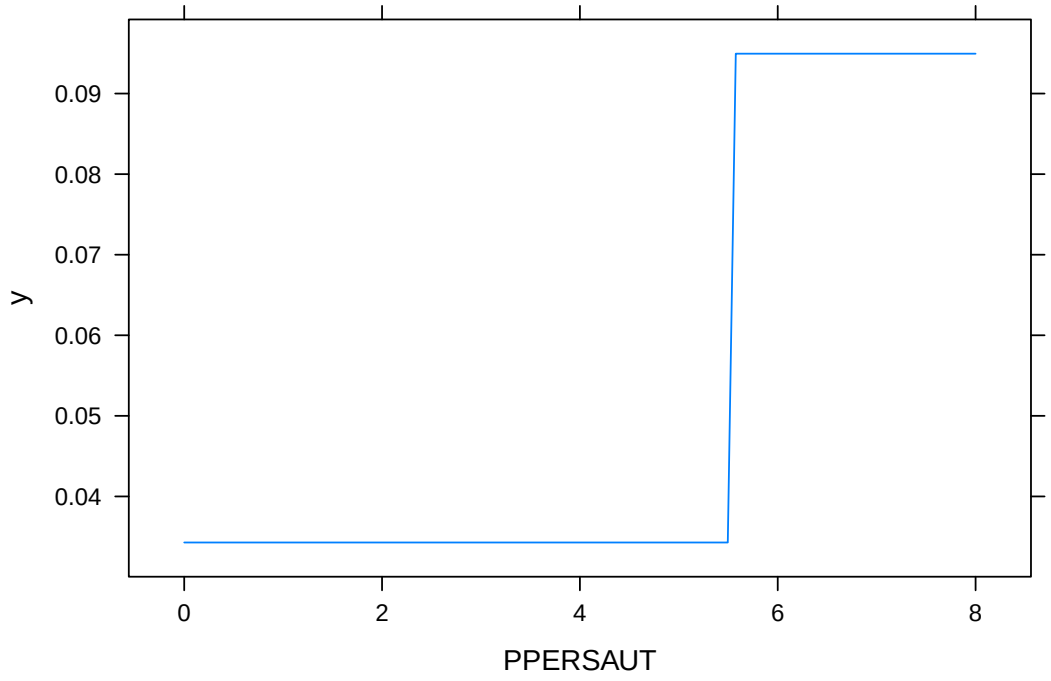


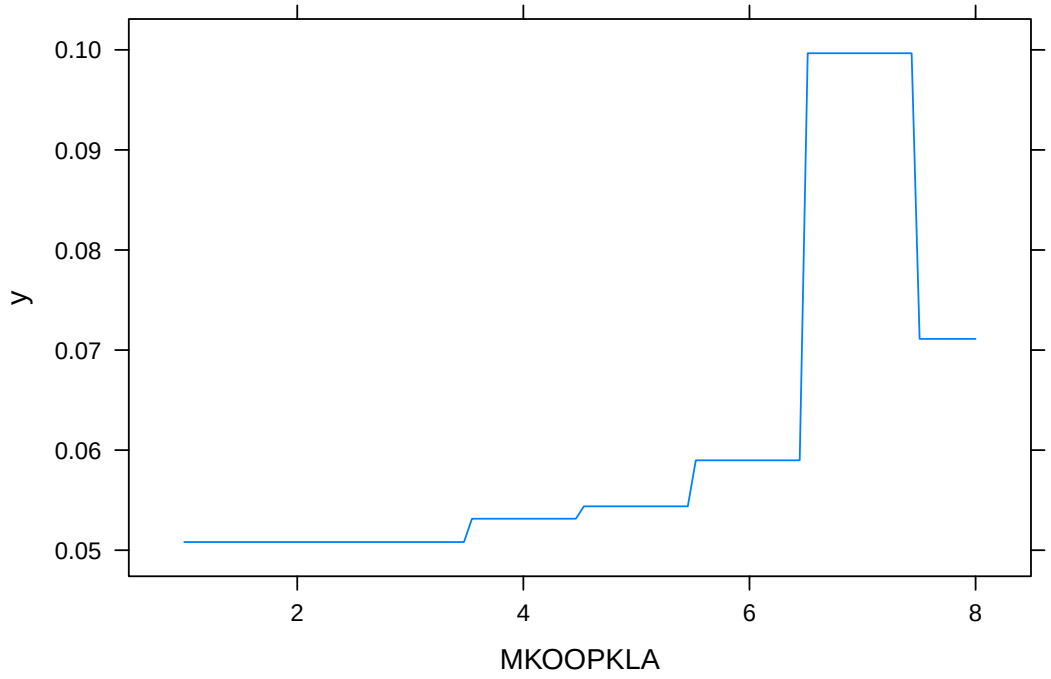

```
##                var     rel.inf
## PPERSAUT PPERSAUT 13.51824557
## MKOOPKLA MKOOPKLA 10.24062778
## MOPLHOOG MOPLHOOG  7.32689780
## MBERMIDD MBERMIDD  6.35820558
## PBRAND     PBRAND  4.98826360
## ABRAND     ABRAND  4.54504653
## MGODGE     MGODGE  4.26496875
## MINK3045 MINK3045  4.13253907
## PWAPART   PWAPART  3.15612877
## MAUT1       MAUT1  2.76929763
## MOSTYPE   MOSTYPE  2.56937935
## MAUT2       MAUT2  1.99879666
## MSKA         MSKA  1.94618539
## MBERARBG MBERARBG  1.89917331
## PBYSTAND PBYSTAND  1.88591514
## MINKGEM   MINKGEM  1.87131472
## MGODOV     MGODOV  1.81673309
## MGODPR     MGODPR  1.80814745
## MFWEKIND MFWEKIND  1.67884570
## MSKC         MSKC  1.65075962
## MBERHOOG MBERHOOG  1.53559951
## MSKB1       MSKB1  1.43339514
## MOPLMIDD MOPLMIDD  1.10617074
## MHHUUR     MHHUUR  1.09608784
## MRELGE     MRELGE  1.09039794
## MINK7512 MINK7512  1.08772012
## MZFONDS   MZFONDS  1.08427551
```

```
## MGODRK     MGODRK  1.03126657
## MINK4575 MINK4575  1.02492795
## MZPART     MZPART  0.98536712
## MRELOV     MRELOV  0.80356854
## MFGEKIND MFGEKIND  0.80335689
## MBERARBO MBERARBO  0.60909852
## APERSAUT APERSAUT  0.56707821
## MGEMOMV   MGEMOMV  0.55589456
## MOSHOOFD MOSHOOFD  0.55498375
## MAUT0       MAUT0  0.54748481
## PMOTSCO   PMOTSCO  0.43362597
## MSKB2       MSKB2  0.43075446
## MSKD         MSKD  0.42751490
## MINK123M MINK123M  0.40920707
## MINKM30   MINKM30  0.36996576
## MHKOOP     MHKOOP  0.34941518
## MBERBOER MBERBOER  0.28967068
## MFALLEEN MFALLEEN  0.28877552
## MGEMLEEF MGEMLEEF  0.20084195
## MOPLLAAG MOPLLAAG  0.15750616
## MBERZELF MBERZELF  0.11203381
## PLEVEN     PLEVEN  0.11030994
## MRELSA     MRELSA  0.04500507
## MAANTHUI MAANTHUI  0.03322830
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PFIETS     PFIETS  0.00000000
## PINBOED   PINBOED  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
## ABROM       ABROM  0.00000000
## ALEVEN     ALEVEN  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## AZEILPL   AZEILPL  0.00000000
```

```
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

Here I use the "gbm" package, and within it the "gbm()" function, to fit boosted regression trees to the Caravan data set. I run "gbm()" with the option "distribution = 'gaussian' " since this is a regression problem. The argument "n.tree = 1000" indicates that here we want 1000 trees. The "summary()" function produces a relative plot and also outputs the relative influence statistics.The variables "PPERSAUT" and "MKOOPKLA" are the two most important variables.





Here I also produce partial dependence plots for these two variables. These plots illustrate the marginal effect of the selected variables on the response after integrating out the other variables.

##(c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20%. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?

```
##    pred.test
##        0    1
##   0 4493   40
##   1  278   11
```

For boosting, the fraction of people predicted to make a purchase that in fact make one is 0.2156863.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
##    pred.test2
##        0    1
##   0 4493   40
##   1  278   11
```

For logistic regression, the fraction of people predicted to make a purchase that in fact make one is again 0.2156863.