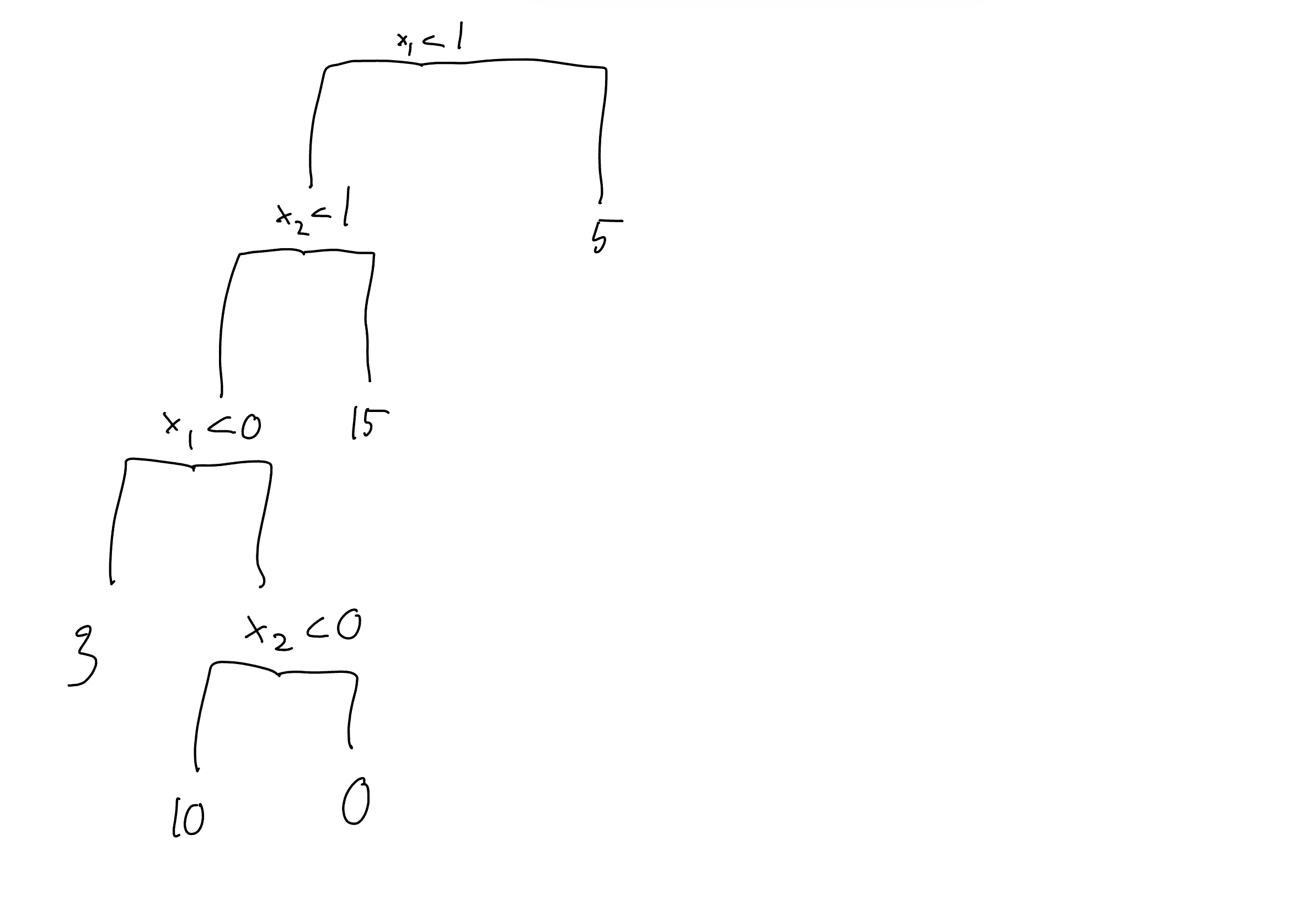
HW 8

Trey Davidson

2024-11-03

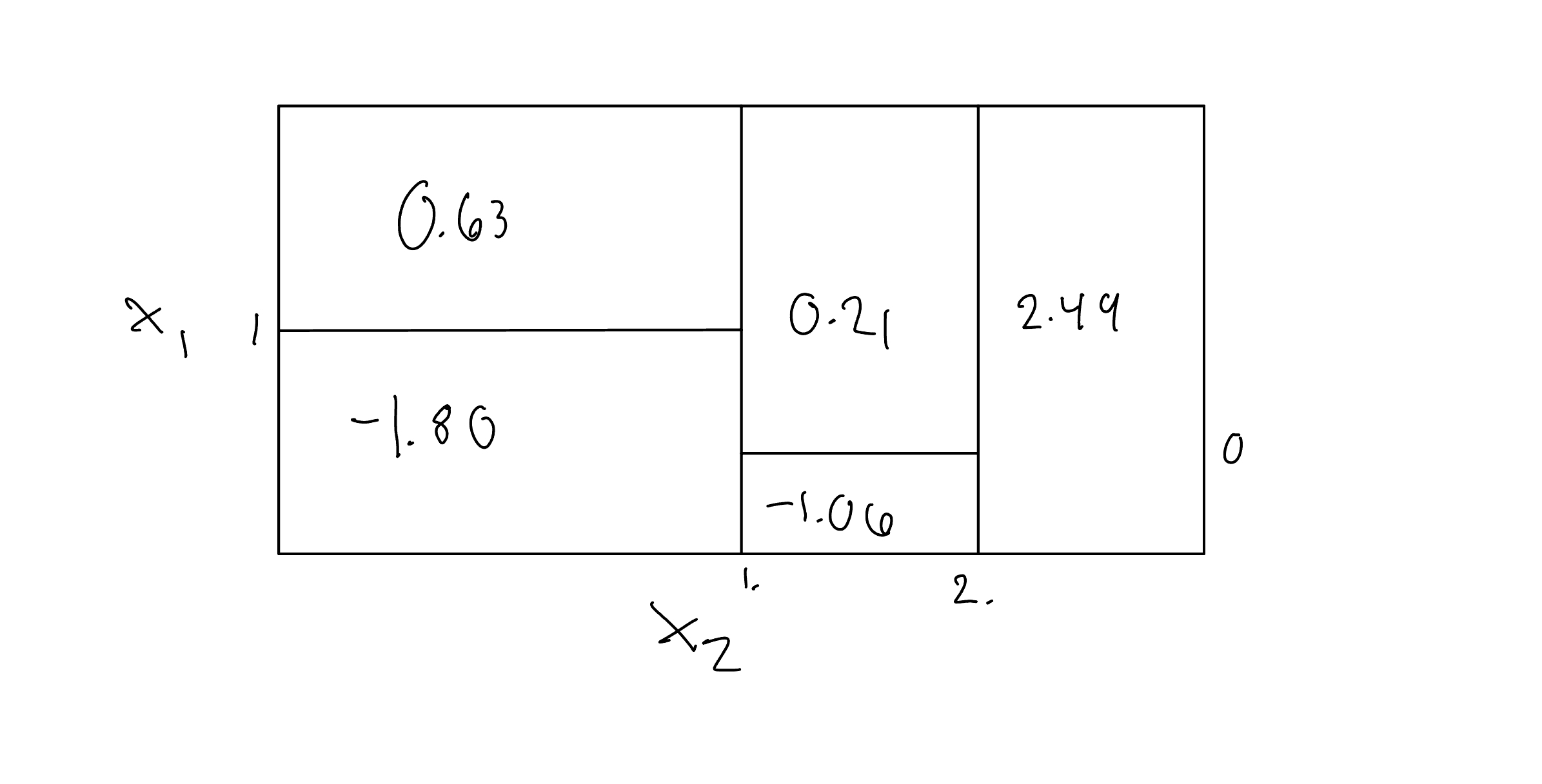
# Question 4

#### A.)



Alt text

#### B.)



Alt text

# Question 5

data <- c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)  
  
# The mean  
mean(data)

## [1] 0.45

# Median  
median(data)

## [1] 0.575

# Since the mean is < 0.5. The average would dicate green, but if you use siple majority you would pick red as the median is greater then 0.5

# Question 8

library(ISLR2)  
library(rpart)  
library(rpart.plot)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(randomForest)

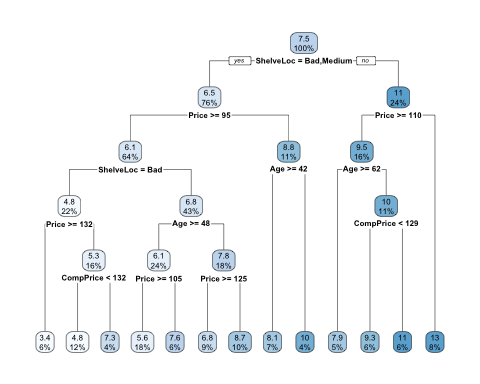
## randomForest 4.7-1.2

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(dbarts)  
set.seed(10)  
  
# Function gets train and test data index's  
getTrainTestIndex <- function(amount)   
 {  
 total\_numbers <- amount / 2  
  
 half <- amount/2  
   
 all\_indices <- sample(1:amount, size = amount, replace = FALSE)  
   
 train\_index <- all\_indices[1:half]  
 test\_index <- all\_indices[half:amount]  
   
 return(list(train\_index = train\_index, test\_index = test\_index))  
}  
  
getMSE <- function(actual, predictions)   
{  
 # This part throws an error  
 if (length(actual) != length(predictions)) {  
 stop("Length of actual and predictions must be the same.")  
 }  
   
 mse <- mean((actual - predictions)^2)  
   
 return(mse)  
}  
  
# A.   
  
data <- Carseats  
n <- nrow(data)  
  
  
indexes <- getTrainTestIndex(n)  
  
traindata <- data[indexes$train\_index, ]  
testdata <- data[indexes$test\_index, ]  
  
  
# B.)   
  
m1 <- rpart(Sales ~ ., data <- traindata)  
  
rpart.plot(m1, roundint=FALSE)



# The plot shows that shelf location is the best predictor of sales, which makes sense when thinking about how people shop.  
  
p1 <- predict(m1, data <- testdata)  
  
m1MSE <- getMSE(data$Sales, p1)  
  
m1MSE

## [1] 4.805864

# C.)   
  
# 5 Fold Cross validation is used   
ctrl <- trainControl(method = "cv", number = 5)   
  
  
m2 <- train(Sales ~ ., data <- traindata, method = "rpart", trControl = ctrl)

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,  
## : There were missing values in resampled performance measures.

p2 <- predict(m2, data <- testdata)  
  
m2MSE <- getMSE(testdata$Sales, p2)  
  
m2MSE

## [1] 5.648047

# Cross vlaidation tree test error is higher then regualr, which seems quite weird to me.   
  
  
# D.)   
  
ctrl <- trainControl(method = "cv", number = 5)  
  
  
m3 <- train(Sales ~ ., data = traindata, method = "rf", trControl = ctrl, importance = TRUE)  
  
p3 <- predict(m3, newdata <- testdata)  
  
  
m3MSE <- getMSE(testdata$Sales, p3)  
  
m3MSE

## [1] 2.974038

# m3MSE error = 2.974038  
  
  
importance\_var <- importance(m3$finalModel)  
  
  
importance\_var

## %IncMSE IncNodePurity  
## CompPrice 15.9129418 142.569881  
## Income 3.4136793 92.919818  
## Advertising 6.6554525 74.187720  
## Population 0.6579445 77.359285  
## Price 42.0831632 394.709779  
## ShelveLocGood 59.8928255 550.741251  
## ShelveLocMedium 21.8453838 80.241935  
## Age 20.0932700 188.373458  
## Education 1.9271461 50.356846  
## UrbanYes -0.7931541 8.574258  
## USYes 3.5380617 12.818894

# E.)   
  
ctrl <- trainControl(method = "cv", number = 5)  
  
  
m4 <- train(Sales ~ ., data = traindata, method = "rf", trControl = ctrl, importance = TRUE)  
  
  
p4 <- predict(m4, newdata = testdata)  
  
  
  
  
m4MSE <- getMSE(testdata$Sales, p4)  
  
m4MSE

## [1] 2.922013

important\_var\_m4 <- importance(m4$finalModel)  
  
important\_var\_m4

## %IncMSE IncNodePurity  
## CompPrice 17.1122471 148.57476  
## Income 3.8375811 86.89012  
## Advertising 8.1837709 77.92788  
## Population 0.4538502 75.54795  
## Price 41.6783667 399.37937  
## ShelveLocGood 56.5898546 554.05667  
## ShelveLocMedium 20.9648880 79.25323  
## Age 20.2424348 192.09721  
## Education -0.1720823 48.75296  
## UrbanYes -1.9802406 7.89007  
## USYes 4.0967975 11.43385

# Random Forest had the lowest MSE so far   
  
  
# F.)   
  
# Bart Model  
  
m5 <- bart2(Sales ~ ., data = traindata, n.samples = 50, keepTrees = TRUE)

##   
## Running BART with numeric y  
##   
## number of trees: 75  
## number of chains: 4, number of threads 4  
## tree thinning rate: 1  
## Prior:  
## k prior fixed to 2.000000  
## degrees of freedom in sigma prior: 3.000000  
## quantile in sigma prior: 0.900000  
## scale in sigma prior: 0.000813  
## power and base for tree prior: 2.000000 0.950000  
## use quantiles for rule cut points: false  
## proposal probabilities: birth/death 0.50, swap 0.10, change 0.40; birth 0.50  
## data:  
## number of training observations: 200  
## number of test observations: 0  
## number of explanatory variables: 12  
## init sigma: 1.016678, curr sigma: 1.016678  
##   
## Cutoff rules c in x<=c vs x>c  
## Number of cutoffs: (var: number of possible c):  
## (1: 100) (2: 100) (3: 100) (4: 100) (5: 100)   
## (6: 100) (7: 100) (8: 100) (9: 100) (10: 100)   
## (11: 100) (12: 100)   
## Running mcmc loop:  
## total seconds in loop: 0.078850  
##   
## Tree sizes, last iteration:  
## [1] 2 3 2 2 2 4 3 2 3 5 4 2 2 4 2 2 2 3   
## 2 3 4 2 4 2 2 3 2 3 4 2 2 2 1 3 3 2 2 2   
## 4 2 2 3 3 3 2 4 2 3 3 2 2 2 2 3 3 2 3 2   
## 4 2 2 3 3 2 2 2 6 2 3 3 4 3 2 2 2   
## [2] 4 3 2 2 2 4 3 2 3 3 3 4 2 1 2 3 2 4   
## 3 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2   
## 2 5 3 2 3 2 2 2 1 3 2 2 4 4 2 3 2 2 2 4   
## 2 2 2 1 2 3 3 2 2 2 2 4 5 2 2 2 3   
## [3] 1 2 1 1 2 2 2 3 3 2 2 3 2 2 2 2 3 3   
## 1 2 3 2 2 2 2 2 3 2 3 2 2 2 5 3 2 2 2 2   
## 2 2 2 2 4 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2   
## 3 4 3 2 2 2 3 4 2 1 4 2 2 3 2 2 2   
## [4] 2 3 3 2 3 2 3 3 3 2 2 2 1 3 5 4 3 2   
## 2 3 4 1 3 2 2 3 2 2 4 3 2 2 4 2 2 2 2 2   
## 2 2 3 4 2 3 2 1 2 2 2 2 3 2 2 2 2 3 2 1   
## 2 2 2 2 2 1 1 2 2 2 2 2 2 3 2 2 4   
##   
## Variable Usage, last iteration (var:count):  
## (1: 55) (2: 32) (3: 38) (4: 30) (5: 65)   
## (6: 23) (7: 37) (8: 35) (9: 28) (10: 23)   
## (11: 31) (12: 33)   
## DONE BART

p5 <- predict(m5, newdata = testdata)  
  
  
m5MSE <- mean((testdata$Sales - p5)^2)  
  
m5MSE

## [1] 14.76847

# Error is higest in BART model

# Question 11

library(gbm)

## Loaded gbm 2.2.2

## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3

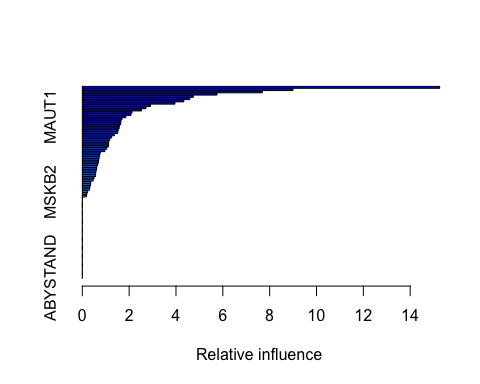
# A.)  
data <- Caravan  
data$Purchase <- as.numeric(data$Purchase) - 1  
traindata <- data[1:1000, ]  
testdata <- data[1000:2000, ]  
  
  
#B.)  
  
  
m1 <- gbm(Purchase ~ .,   
 data = traindata,   
 shrinkage = 0.01,   
 n.trees = 1000,   
 verbose = FALSE)

## Distribution not specified, assuming bernoulli ...

## 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.

importance <- summary(m1)



importance

## var rel.inf  
## PPERSAUT PPERSAUT 15.2599360  
## MKOOPKLA MKOOPKLA 8.9965703  
## MOPLHOOG MOPLHOOG 7.6839168  
## MBERMIDD MBERMIDD 5.7509151  
## PBRAND PBRAND 4.7378035  
## ABRAND ABRAND 4.5905195  
## MGODGE MGODGE 4.3331970  
## MINK3045 MINK3045 3.9550601  
## MOSTYPE MOSTYPE 2.9126810  
## PWAPART PWAPART 2.7208325  
## MAUT2 MAUT2 2.5213548  
## MSKC MSKC 2.1234955  
## MSKA MSKA 2.0745007  
## MAUT1 MAUT1 1.8557720  
## MBERARBG MBERARBG 1.6944525  
## MFWEKIND MFWEKIND 1.6503680  
## MGODOV MGODOV 1.6406509  
## PBYSTAND PBYSTAND 1.5957596  
## MGODPR MGODPR 1.5712622  
## MSKB1 MSKB1 1.5330946  
## MHHUUR MHHUUR 1.4987879  
## MGODRK MGODRK 1.3637335  
## MAUT0 MAUT0 1.2409838  
## MBERHOOG MBERHOOG 1.1683765  
## MINK7512 MINK7512 1.1308773  
## MOSHOOFD MOSHOOFD 1.1229088  
## MINKGEM MINKGEM 1.1107054  
## MRELGE MRELGE 1.0303012  
## MBERBOER MBERBOER 0.9616403  
## MGEMLEEF MGEMLEEF 0.7644460  
## MFGEKIND MFGEKIND 0.7405169  
## MINKM30 MINKM30 0.7222423  
## MRELOV MRELOV 0.7028762  
## MGEMOMV MGEMOMV 0.6926424  
## PLEVEN PLEVEN 0.6599423  
## MBERARBO MBERARBO 0.6256093  
## MOPLMIDD MOPLMIDD 0.6135263  
## PMOTSCO PMOTSCO 0.5884630  
## APERSAUT APERSAUT 0.5715136  
## MZPART MZPART 0.5656801  
## MZFONDS MZFONDS 0.5045545  
## MHKOOP MHKOOP 0.4610491  
## MINK4575 MINK4575 0.3620714  
## MFALLEEN MFALLEEN 0.3525245  
## MINK123M MINK123M 0.3277921  
## MSKD MSKD 0.3128017  
## MSKB2 MSKB2 0.2297329  
## MOPLLAAG MOPLLAAG 0.1995451  
## MRELSA MRELSA 0.1720130  
## MAANTHUI MAANTHUI 0.0000000  
## MBERZELF MBERZELF 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  
## 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

# MSKC clealry seems to be important. The rest seems to drop off.