

STA401

Fall 2024

Dr. Ayman Alzaatreh

Homework 7

Halla, Rumisal, Eleonora

18 November 2024

**[4 Pts] Question One:** Section 8.4 Questions 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.

library(randomForest)

# Warning: package 'randomForest' was built under R version 4.4.2

# randomForest 4.7-1.2

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

library(MASS)

data("Boston")

set.seed(1234)

train=sample(1:nrow(Boston),nrow(Boston)/2)

test=Boston[-train,"medv"]

rf.boston1=randomForest(medv~.,data=Boston,subset=train,mtry=13,importance=TRUE, ntree = 400)

yhat.rf1=predict(rf.boston1,newdata=Boston[-train,])

rf.boston2=randomForest(medv~.,data=Boston,subset=train,mtry=13/2,importance=TRUE, ntree = 400)

yhat.rf2 = predict(rf.boston2,newdata=Boston[-train,])

rf.boston3=randomForest(medv~.,data=Boston,subset=train,mtry=sqrt(13) ,importance=TRUE, ntree = 400)

yhat.rf3 = predict(rf.boston3,newdata=Boston[-train,])

plot(1:400, rf.boston1$mse, col = "green", type = "l", xlab = "Number of Trees", ylab = "MSE", ylim = c(10,32))

lines(1:400, rf.boston2$mse, col = "blue", type = "l")

lines(1:400, rf.boston3$mse, col = "red", type = "l")

legend("topright", c("m = p","m = p/2","m = sqrt(p)"), col=c ("green", "blue", "red"), cex=1, lty=1) A graph of different colored lines

Description automatically generated

* Using less predictors (p/2, sqrt(p)) we get a lower MSE compared to using all predictors.
* MSE is high for single tree.
* As the number of tree increases. the test classification error decreases.

**[16 Pts] Question Two:** Section 8.4 Questions 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.

library(MASS)

library(installr)

# Warning: package 'installr' was built under R version 4.4.2

# Welcome to installr version 0.23.4

# More information is available on the installr project website: https://github.com/talgalili/installr/

# Contact: <tal.galili@gmail.com>

# Suggestions and bug-reports can be submitted at: https://github.com/talgalili/installr/issues

library(tree)

library(randomForest)

# Warning: package 'randomForest' was built under R version 4.4.2

# randomForest 4.7-1.2

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

library(ISLR)

data(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 ...

(a) Split the data set into a training set and a test set.

set.seed(123)

train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)

Carseats.train <- Carseats[train, ]

Carseats.test <- Carseats[-train, ]

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

tree.Carseats<-tree(Sales~

.,data=Carseats.train)

summary(tree.Carseats)

# Regression tree:

# tree(formula = Sales ~., data = Carseats.train)

# Variables actually used in tree construction:

# [1] "ShelveLoc" "Price" "Income" "Age" "Population"

# [6] "Education" "CompPrice" "Advertising"

# Number of terminal nodes: 18

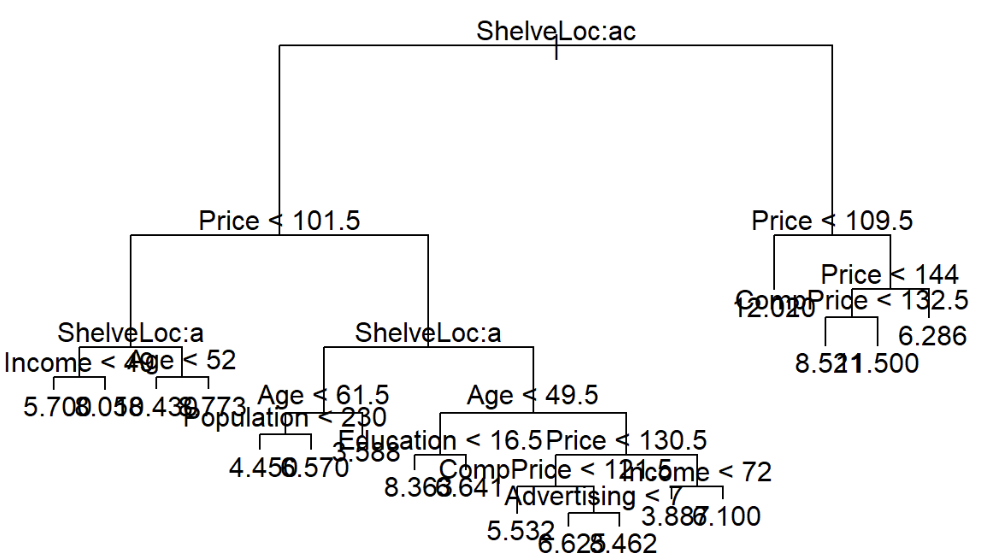
# Residual mean deviance: 2.132 = 388.1 / 182

# Distribution of residuals:

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# -4.08000 -0.92870 0.06244 0.00000 0.87020 3.71700

plot(tree.Carseats)

text(tree.Carseats) 

# Train MSE

yhat <- predict(tree.Carseats, newdata = Carseats.test)

mean((yhat - Carseats.test$Sales)^2)

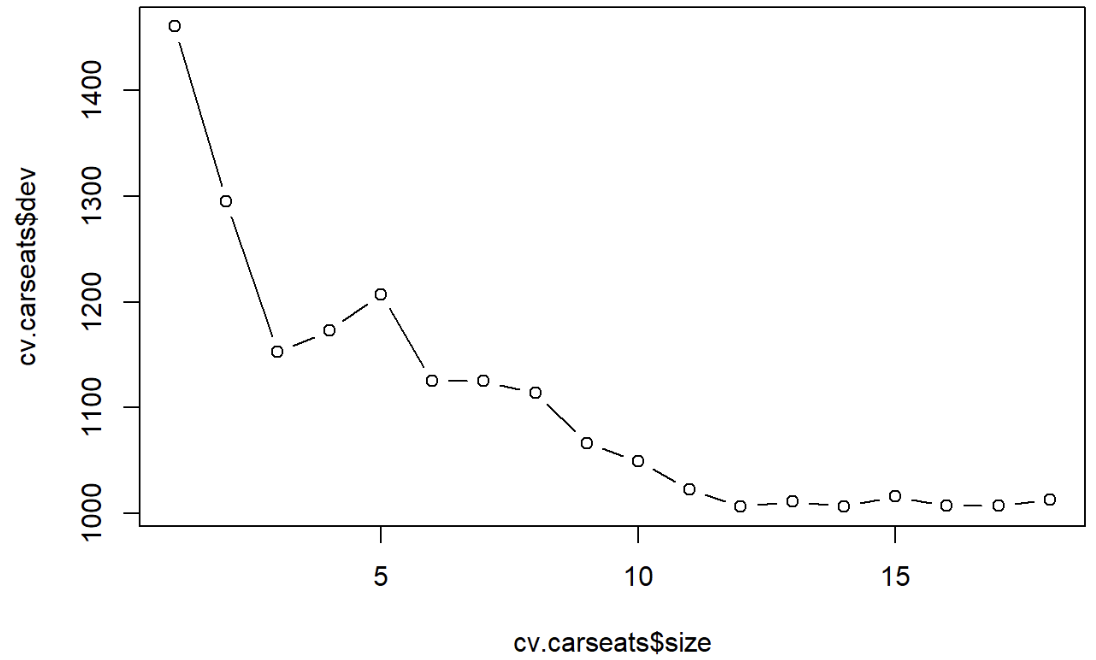
# [1] 4.395357

* The tree formatted has 18 nodes and has a MSE is 4.395357.
* Our tress has 18 nodes which is rather large her it is hard to interpret and may be a indicator of overfitting.

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

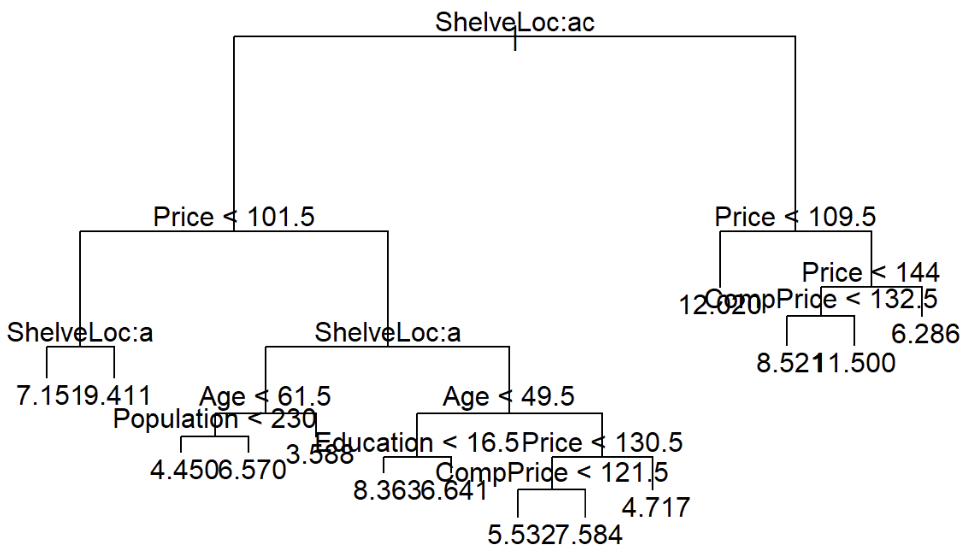
set.seed(123)

cv.carseats <- cv.tree(tree.Carseats)

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

prune.carseats <- prune.tree(tree.Carseats, best = 14)

plot(prune.carseats)

text(prune.carseats) 

summary(prune.carseats)

# Regression tree:

# snip.tree(tree = tree.Carseats, nodes = c(9L, 8L, 47L, 93L))

# Variables actually used in tree construction:

# [1] "ShelveLoc" "Price" "Age" "Population" "Education"

# [6] "CompPrice"

# Number of terminal nodes: 14

# Residual mean deviance: 2.472 = 459.9 / 186

# Distribution of residuals:

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# -4.55700 -0.91850 0.04014 0.00000 1.03800 3.99300

#Train MSE

yhat <- predict(prune.carseats, newdata = Carseats.test)

mean((yhat - Carseats.test$Sales)^2)

# [1] 4.658628

* Looking at the graph above we chose the size of the tree to be 12. Here the MSE increases slightly to 4.65828. Hence pruning in this case did not make many improvements.

(d) 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.

#m=p=11-1=10

bag.carseats <- randomForest(Sales ~., data = Carseats.train, mtry=10, ntree = 500, importance = TRUE)

yhat.bag <- predict(bag.carseats, newdata = Carseats.test)

mean((yhat.bag - Carseats.test$Sales)^2)

# [1] 2.745027

importance(bag.carseats)

# %IncMSE IncNodePurity

# CompPrice 20.4013451 159.095368

# Income 6.1612049 91.592151

# Advertising 8.7028129 76.445263

# Population -0.7324223 53.858972

# Price 43.3236360 379.318509

# ShelveLoc 49.1591661 385.039082

# Age 18.8396723 182.547300

# Education 2.7741792 59.907123

# Urban 0.4909844 8.497199

# US 0.4195280 5.892226

* Bagging technique(m=p) results to the MSE dropping to 2.72% which compared to the other models id a signification improvement, From the importance we can see that top 5 important variables in order are ShelveLoc, Price, ComPrice, Age & Advertising.

(e) 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. Describe the eﬀect of *m*, the number of variables considered at each split, on the error rate obtained.

#m=P=11-1=10

rf.carseats <- randomForest(Sales ~., data = Carseats.train, mtry = 10, ntree=500, importance = TRUE)

yhat.rf <- predict(rf.carseats, newdata = Carseats.test)

mean((yhat.rf - Carseats.test$Sales)^2)

# [1] 2.74408

importance(rf.carseats)

# %IncMSE IncNodePurity

# CompPrice 21.790942 157.947583

# Income 5.691801 93.565958

# Advertising 7.988997 75.420727

# Population -1.535281 51.881264

# Price 45.018559 386.860943

# ShelveLoc 47.095504 390.615288

# Age 20.302978 178.081085

# Education 1.214453 54.033763

# Urban -2.250965 8.171754

# US 1.115918 5.887787

#m=P/2= (approx.) 6

rf.carseats <- randomForest(Sales ~., data = Carseats.train, mtry=6, ntree=500, importance = TRUE)

yhat.rf <- predict(rf.carseats, newdata = Carseats.test)

mean((yhat.rf - Carseats.test$Sales)^2)

# [1] 2.935834

importance(rf.carseats)

# %IncMSE IncNodePurity

# CompPrice 17.5348313 154.815742

# Income 6.3074382 101.906997

# Advertising 6.4498653 83.259250

# Population -0.1513790 72.679727

# Price 37.4720003 339.576542

# ShelveLoc 45.3500763 357.128316

# Age 17.4860950 200.746783

# Education 3.1752553 64.368015

# Urban -0.9851129 9.518903

# US 0.6068700 9.146504

#m=sqrt(P)= (approx.) 3

rf.carseats <- randomForest(Sales ~., data = Carseats.train, mtry = 3, ntree=500, importance = TRUE)

yhat.rf <- predict(rf.carseats, newdata = Carseats.test)

mean((yhat.rf - Carseats.test$Sales)^2)

## [1] 3.580806

importance(rf.carseats)

# %IncMSE IncNodePurity

# CompPrice 12.5377980 148.69421

# Income 4.9266939 124.32997

# Advertising 7.2094469 102.28182

# Population 0.7075058 99.14692

# Price 31.3474320 291.74468

# ShelveLoc 32.3906554 275.63201

# Age 19.1275379 211.13580

# Education 0.9640362 74.65386

# Urban -0.6873275 16.07897

# US 0.9545223 15.34713

**[20 Pts] Question Three:**

Remark: This question is based on the dataset Organic available on iLearn. Support your answers with appropriate outputs from R.

A supermarket is offering a new line of organic products. The supermarket's management wants to determine which customers are likely to purchase these products. The supermarket has a customer loyalty program. As an initial buyer incentive plan, the supermarket provided coupons for the organic products to all of the loyalty program participants and collected data that includes whether these customers purchased any of the organic products.

The Organic Data set contains 13 variables and more than 22,000 observations. The variables in the data set are shown below with the appropriate roles and levels.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Model Role | Measurement Level | Description |
| ID | ID | Nominal | Customer loyalty identification number |
| DemAffl | Input | Interval | Affluence grade on a scale from 1 to 30 |
| DemAge | Input | Interval | Age, in years |
| DemCluster | Rejected | Nominal | Type of residential neighborhood |
| DemClusterGroup | Input | Nominal | Neighborhood group |
| DemGender | Input | Nominal | M = male, F = female, U = unknown |
| DemRegion | Input | Nominal | Geographic region |
| DemTVReg | Input | Nominal | Television region |
| PromClass | Input | Nominal | Loyalty status: tin, silver, gold, or platinum |
| PromSpend | Input | Nominal | Total amount spent |
| PromTime | Input | Interval | Time as loyalty card member |
| TargetBuy | Target | Binary | Organics purchased? 1 = Yes, 0 = No |
| TargetAmt | Rejected | Interval | Number of organic products purchased |

**Note:** Although two target variables are listed, the question concentrate on the binary variable

TargetBuy. **Therefore, you must not include the rejected variables in your analysis**.

Answer the following questions:

1. Create an appropriate graph to explore the target variable TargetBuy. What is the proportion of Yes in the sample data?

library(readxl)

library(ggplot2)

library(lattice)

setwd(r"(/Users/halla.d/Library/Mobile Documents/com~apple~CloudDocs/Desktop/STA401)")

df = read\_excel("Organic.xlsx")

df = df[-c(1, 4, 13)]

str(df)

# convert necessary variables to factors or numeric first

# There are invalid values for the DemAffl => valid range is from 1 to 30

df <- df[df$DemAffl>0 & df$DemAffl<=30, ]

# remove missing values first

sum(is.na(df))

df <- na.omit(df)

df$DemClusterGroup = as.factor(df$DemClusterGroup)

df$DemAffl = as.factor(df$DemAffl)

df$PromClass = as.factor(df$PromClass)

df$DemGender = as.factor(df$DemGender)

df$TargetBuy = as.factor(df$TargetBuy)

df$DemReg = as.factor(df$DemReg)

df$DemTVReg = as.factor(df$DemTVReg)

table(df$TargetBuy)/16402

A close-up of a number

Description automatically generated

plot(df$TargetBuy,

xlab="0: Not Organic | 1: Organic",

ylab="Number of Customers")

A graph of a graph

Description automatically generated

* The proportion of Yes (TargetBuy = 1) in the sample data is 26.35%.

**b)** Partition the data into 50% training and 50% test. Use stratified sampling to insure equally proportion of the Target in the portioned data sets.

library("caret")

set.seed(123)

s <- createDataPartition(y = df$TargetBuy

,p = 0.5,list = FALSE)

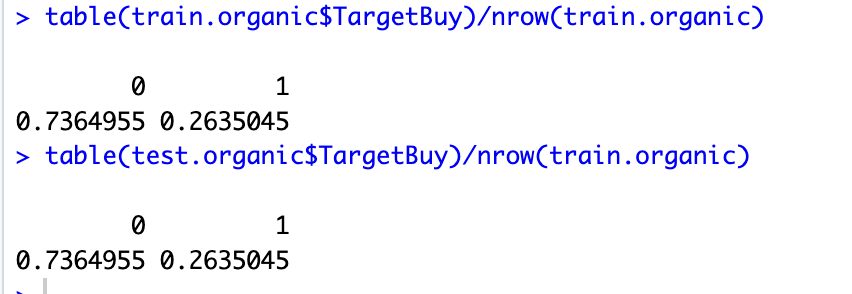
# ensures the prop. is maintained

train.organic <- df[s,]

test.organic <- df[-s,]

table(train.organic$TargetBuy)/nrow(train.organic)

table(test.organic$TargetBuy)/nrow(train.organic)



**c)** Use seed value of 12345. Fit a Decision Tree model to the training data. Plot the tree and interpret the results. What is the test misclassification rate?

set.seed(12345)

library(tree)

tree.organic=tree(TargetBuy~.,

data=train.organic) # classification tree

summary(tree.organic)

plot(tree.organic)

text(tree.organic)

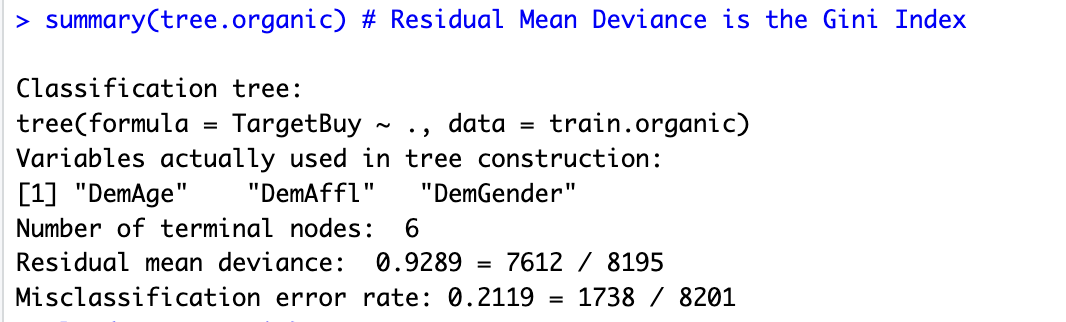
tree.pred=predict(tree.organic, test.organic,

type="class") # type="class"

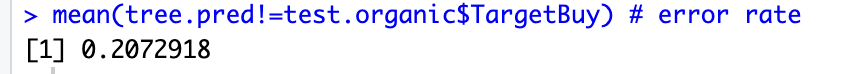
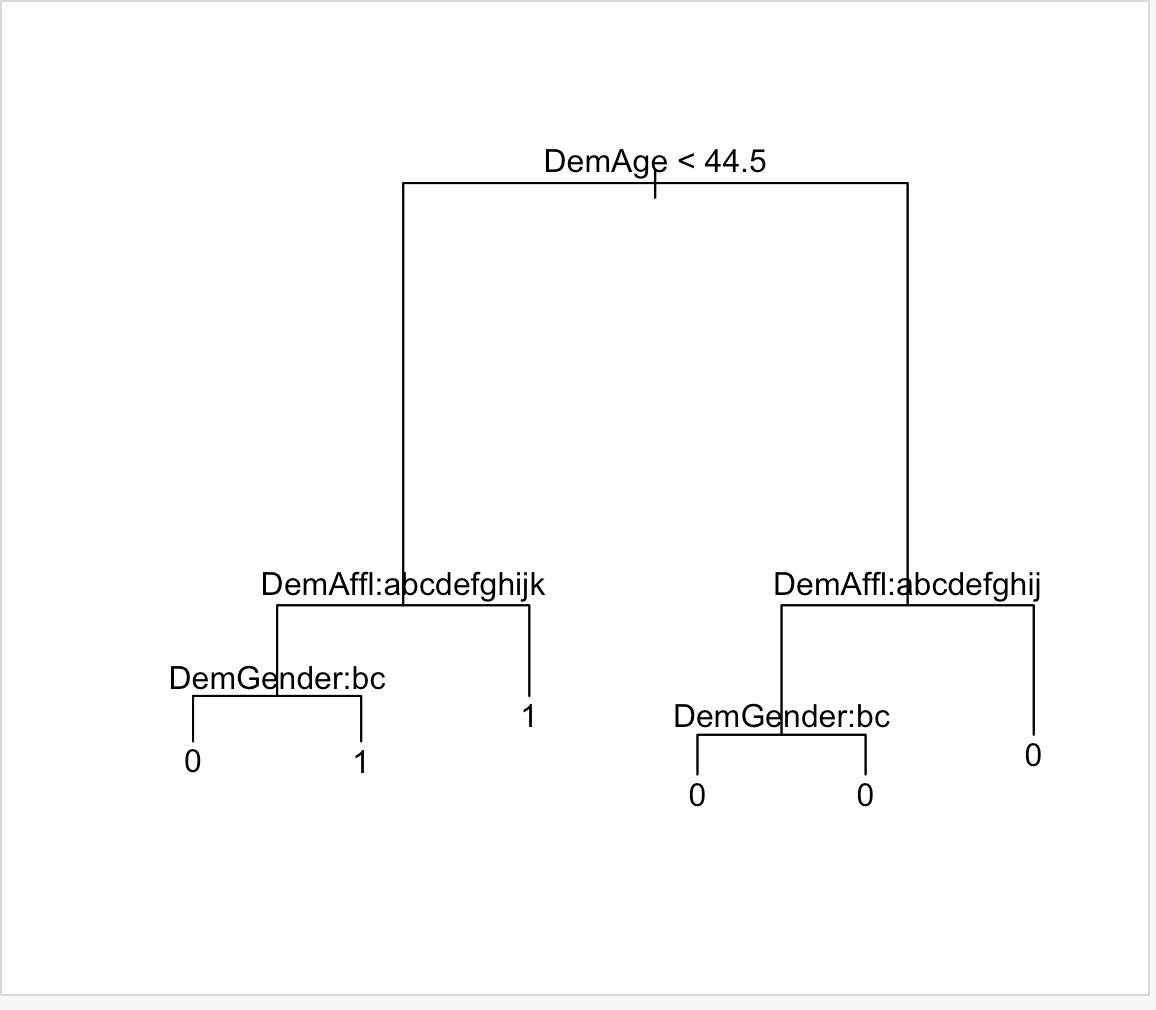
# is necessary for classification trees

summary(tree.pred)

mean(tree.pred!=test.organic$TargetBuy) # error rate



**The 3 variables used in this classification tree are the age of the customer, the affluence grade, and the gender**



* The test misclassification rate is 20.73%.

**d)** Use cross-validation to determine the best pruned tree. What is the test misclassification rate for the pruned tree? Does pruning improve the test error rate?

# lowest error rate

cv.organic=cv.tree(tree.organic)

cbind(cv.organic$size,cv.organic$dev)

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

prune.organic=prune.misclass(tree.organic,best=6)

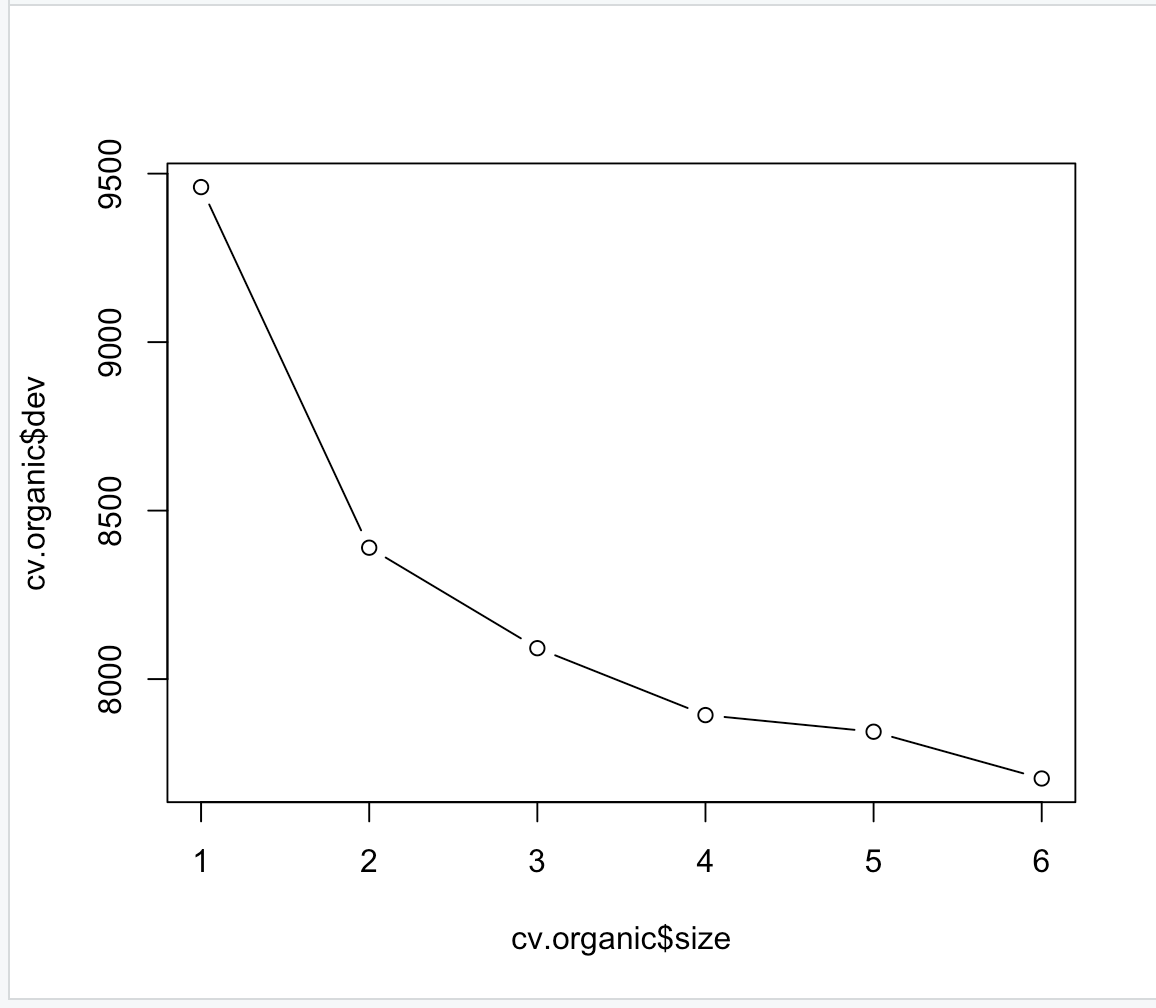
plot(prune.organic)

text(prune.organic,pretty=0)

tree.pred=predict(prune.organic,test.organic,type="class")

table(predicted=tree.pred,actual=test.organic$TargetBuy)

mean(tree.pred!=test.organic$TargetBuy)



6-node tree has the lowest cv error

A screenshot of a computer screen

Description automatically generated

A close-up of a website

Description automatically generated

test error = 20.73% which has no improvement from the previous tree (they’re the same tree)

**e)** Fit the data using Bagging method with 1000 trees. Compare the test misclassification error rate with part (d). Which one is better?

# bagging (random forest with number of tree=number of predictors)

set.seed(12345)

dim(df) # p=10-1=9

str(df)

# number of predictors for Organic data is p=9

bag.organic=randomForest(TargetBuy~.

,data=train.organic,mtry=9,

importance=TRUE, ntree = 1000)

# p=9,that is a bagging approach

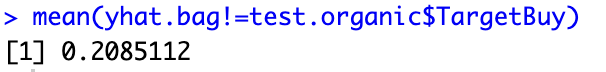
yhat.bag = predict(bag.organic,newdata=test.organic, type="class")

mean(yhat.bag!=test.organic$TargetBuy)

#compare test misclassification error rate with the previous

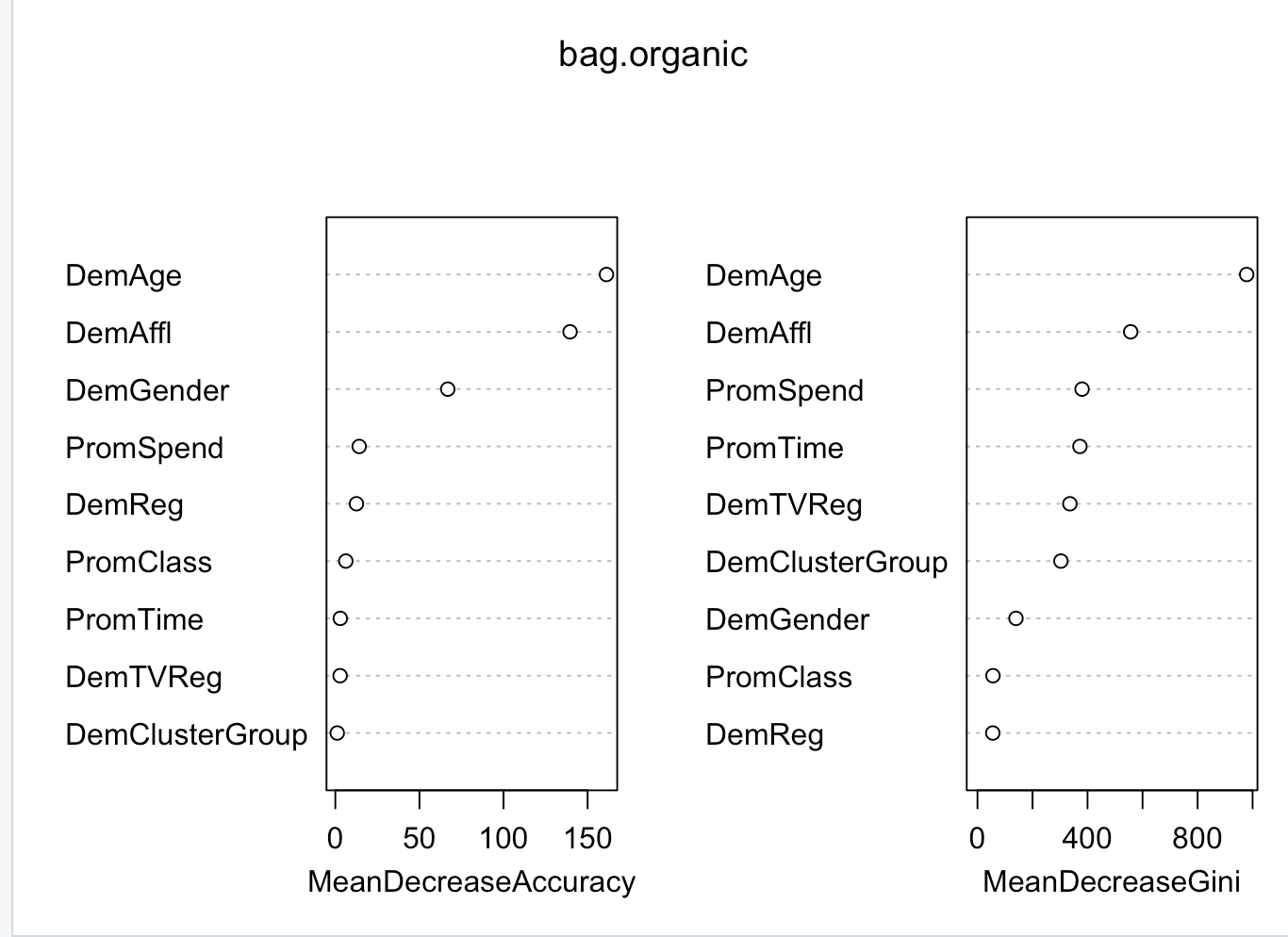
# Decision Tree

varImpPlot(bag.organic)



The test misclassification error rate is 20.85% hence no improvement over the pruned tree in part d

**f)** Plot the variable importance curve. Indicate the top 3 variables.



DemAge, DemAffl, and DemGender are the 3 most important variables for the accuracy of prediction

**g)** Repeat parts (e) and (f) using Random forest with m=3 and 1000 trees.

# random forest (number of variables used < number of predictors,

# default=p/3 for regression tree and sqrt(p) for classification tree)

set.seed(12345)

#p=sqrt(9) approx 3

rf.organic\_2=randomForest(TargetBuy~.,data=train.organic,mtry=3,

importance=TRUE, ntree=1000)

rf.organic\_2

yhat.rf\_2 = predict(rf.organic\_2,newdata=test.organic, type="class")

yhat.rf\_2

test.organic$TargetBuy

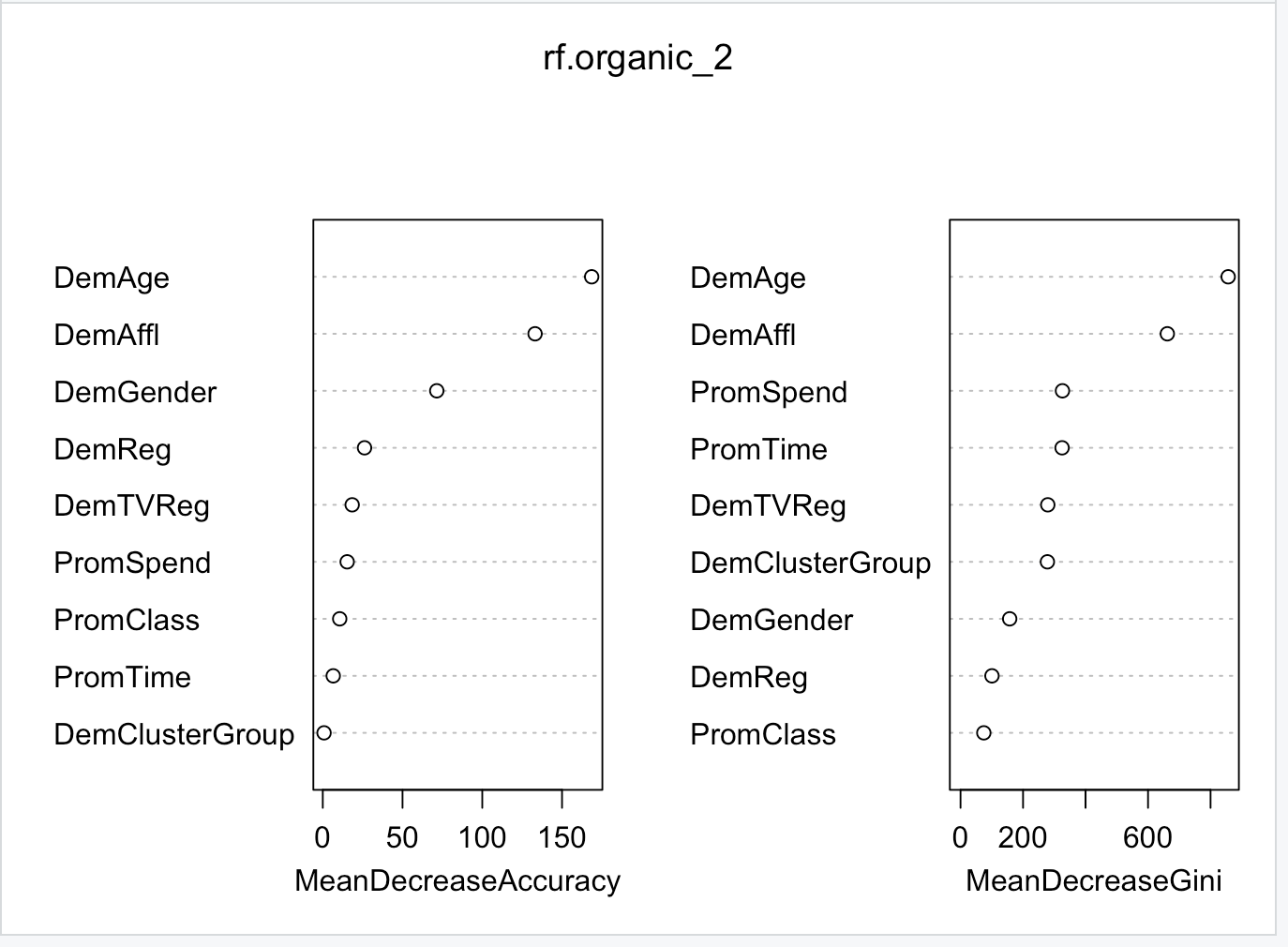
mean((yhat.rf\_2!=test.organic$TargetBuy))

#par(mfrow=c(1,2))

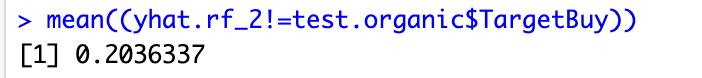
#importance(rf.organic\_2)

varImpPlot(rf.organic\_2)

#plot(rf.organic\_2)



DemAge, DemAffl, and DemGender are the 3 most important variables for the accuracy of prediction



The test misclassification error rate is 20.36%. It is an improvement over the misclassification error rate of the trees in part (e)

**h)** Use Boosting method with 5000 trees and compare the test error rate it with Bagging and

Random Forest results. Which method gives the best test misclassification error rate?

# Need to convert to numeric

df$TargetBuy = as.numeric(df$TargetBuy) -1

train.organic$TargetBuy = as.numeric(train.organic$TargetBuy) -1

test.organic$TargetBuy = as.numeric(test.organic$TargetBuy) -1

str(df)

install.packages("gbm")

library(gbm)

set.seed(12345)

str(test.organic)

boost.organic=gbm(TargetBuy~.,data=train.organic,

distribution="bernoulli",

n.trees=5000,interaction.depth=1)

summary(boost.organic)

yhat.boost=predict(boost.organic,newdata=test.organic,

n.trees=5000,type="response")

yhat.boost.class<-ifelse((yhat.boost<=0.5),0,1)

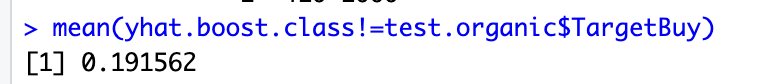
table(yhat.boost.class,test.organic$TargetBuy)

mean(yhat.boost.class!=test.organic$TargetBuy)

A blue bar graph with numbers

Description automatically generated

Based on this plot, the important variables seem to be DemAffl, DemTVReg, and DemGender as they have the highest relative influence as compared to the other variables



Based on all the methods, the gradient boosting gives the lowest test misclassification error rate of 19.16%