ML Exam

Sagar Kalauni

2023-12-15

set.seed(100)  
library(datasets)  
library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.2

library(tree)

## Warning: package 'tree' was built under R version 4.3.2

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.3.2

## randomForest 4.7-1.1

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

library(boot)

## Warning: package 'boot' was built under R version 4.3.2

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.3.1

##   
## Attaching package: 'ggplot2'

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

library(e1071)

## Warning: package 'e1071' was built under R version 4.3.2

library(gbm)

## Warning: package 'gbm' was built under R version 4.3.2

## Loaded gbm 2.1.8.1

library(nnet)

## Warning: package 'nnet' was built under R version 4.3.2

library(ISLR)

## Warning: package 'ISLR' was built under R version 4.3.2

##   
## Attaching package: 'ISLR'

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

library(factoextra)

## Warning: package 'factoextra' was built under R version 4.3.2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

This problem involves the Caravan data set which is part of the ISLR2 package. Perform the following analysis. This is an imbalanced data set.

library(ISLR2)  
dim(Caravan)

## [1] 5822 86

nrow(Caravan)

## [1] 5822

#  
standardize=function(x) {(x-min(x))/(max(x)-min(x))}  
Caravan$MOSTYPE=standardize(Caravan$MOSTYPE)  
#Caravan$MOSHOOFD=standardize(Caravan$MOSHOOFD)

colnames(Caravan)

## [1] "MOSTYPE" "MAANTHUI" "MGEMOMV" "MGEMLEEF" "MOSHOOFD" "MGODRK"   
## [7] "MGODPR" "MGODOV" "MGODGE" "MRELGE" "MRELSA" "MRELOV"   
## [13] "MFALLEEN" "MFGEKIND" "MFWEKIND" "MOPLHOOG" "MOPLMIDD" "MOPLLAAG"  
## [19] "MBERHOOG" "MBERZELF" "MBERBOER" "MBERMIDD" "MBERARBG" "MBERARBO"  
## [25] "MSKA" "MSKB1" "MSKB2" "MSKC" "MSKD" "MHHUUR"   
## [31] "MHKOOP" "MAUT1" "MAUT2" "MAUT0" "MZFONDS" "MZPART"   
## [37] "MINKM30" "MINK3045" "MINK4575" "MINK7512" "MINK123M" "MINKGEM"   
## [43] "MKOOPKLA" "PWAPART" "PWABEDR" "PWALAND" "PPERSAUT" "PBESAUT"   
## [49] "PMOTSCO" "PVRAAUT" "PAANHANG" "PTRACTOR" "PWERKT" "PBROM"   
## [55] "PLEVEN" "PPERSONG" "PGEZONG" "PWAOREG" "PBRAND" "PZEILPL"   
## [61] "PPLEZIER" "PFIETS" "PINBOED" "PBYSTAND" "AWAPART" "AWABEDR"   
## [67] "AWALAND" "APERSAUT" "ABESAUT" "AMOTSCO" "AVRAAUT" "AAANHANG"  
## [73] "ATRACTOR" "AWERKT" "ABROM" "ALEVEN" "APERSONG" "AGEZONG"   
## [79] "AWAOREG" "ABRAND" "AZEILPL" "APLEZIER" "AFIETS" "AINBOED"   
## [85] "ABYSTAND" "Purchase"

1. It’s a large data. To reduce computational time (and just for illustration), let’s first create a training set containing the first 1000 observations, and a test set containing the next 300 observations.

# Step 1: Split the data into training and testing sets  
train\_data=Caravan[1:1000, ]  
test\_data=Caravan[1001:1300, ]  
  
train\_index=1:1000

1. Fit a tree to the training data, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

set.seed(100)  
library(tree)  
tree.d=tree(Purchase~., Caravan, split = 'gini', subset =train\_index ) # except Purchase all other variables in the data set are be considered as predictors.

summary(tree.d)

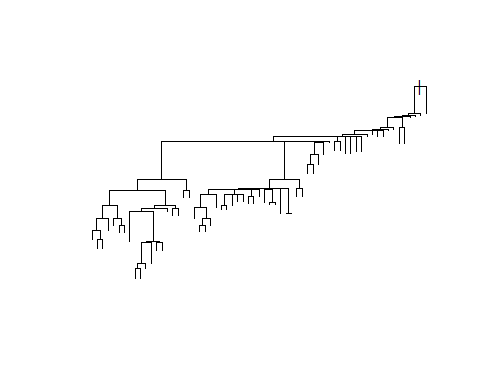
##   
## Classification tree:  
## tree(formula = Purchase ~ ., data = Caravan, subset = train\_index,   
## split = "gini")  
## Variables actually used in tree construction:  
## [1] "PPLEZIER" "PINBOED" "PWABEDR" "PGEZONG" "PBYSTAND" "PPERSONG"  
## [7] "PAANHANG" "PFIETS" "PWALAND" "PTRACTOR" "PLEVEN" "PMOTSCO"   
## [13] "PBROM" "MAANTHUI" "MKOOPKLA" "MBERARBG" "PPERSAUT" "MAUT2"   
## [19] "MSKB1" "MOPLLAAG" "MBERMIDD" "MHKOOP" "MBERARBO" "APERSAUT"  
## [25] "MINK123M" "PBRAND" "MBERZELF" "MGODPR" "MINK7512" "MGODRK"   
## [31] "MOSTYPE" "MBERBOER" "MSKD" "MBERHOOG" "MINKM30" "MSKA"   
## [37] "MRELSA" "MAUT0" "MOSHOOFD" "MFALLEEN" "MGODGE" "MGODOV"   
## [43] "MFWEKIND" "MZPART" "MSKC"   
## Number of terminal nodes: 63   
## Residual mean deviance: 0.2391 = 224 / 937   
## Misclassification error rate: 0.058 = 58 / 1000

This is a classification tree, we have a total number of terminal node of 63, so it’s a big tree. we have mean deviance: 0.2391 , which is calculated deviance divided by total number of training observation minus the number of terminal nodes. We also have Misclassification error rate: 0.058, which is calculated as Number of Misclassification divided by total training set.

we see that the training error rate is 5.8%. The residual mean deviance reported is simply the deviance divided by , which in this case is 1000-63= 937.

1. Create a plot of the tree. Pick one of the terminal nodes, and interpret the information displayed.

plot(tree.d) # for Plotting the decision tree



#text(tree.d, pretty= 0) #if you want to see labels also

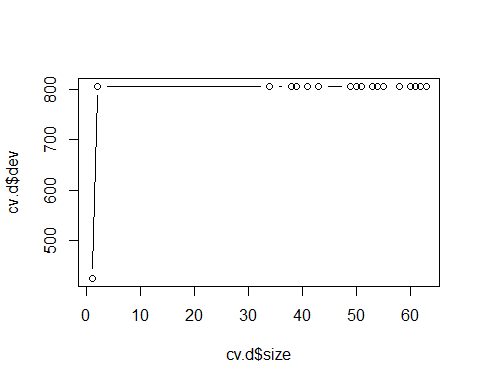
set.seed(100)  
tree.d

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 1000 448.400 No ( 0.94100 0.05900 )   
## 2) PPLEZIER < 0.5 992 425.100 No ( 0.94456 0.05544 )   
## 4) PINBOED < 0.5 987 424.500 No ( 0.94428 0.05572 )   
## 8) PWABEDR < 0.5 978 417.800 No ( 0.94479 0.05521 )   
## 16) PGEZONG < 1 969 411.100 No ( 0.94530 0.05470 )   
## 32) PBYSTAND < 1 956 392.400 No ( 0.94770 0.05230 )   
## 64) PPERSONG < 1 947 391.400 No ( 0.94720 0.05280 )   
## 128) PAANHANG < 0.5 937 384.600 No ( 0.94771 0.05229 )   
## 256) PFIETS < 0.5 920 377.000 No ( 0.94783 0.05217 )   
## 512) PWALAND < 1.5 904 375.200 No ( 0.94690 0.05310 )   
## 1024) PTRACTOR < 1.5 894 374.100 No ( 0.94631 0.05369 )   
## 2048) PLEVEN < 0.5 863 359.200 No ( 0.94670 0.05330 )   
## 4096) PMOTSCO < 2 830 344.100 No ( 0.94699 0.05301 )   
## 8192) PBROM < 1 775 332.300 No ( 0.94452 0.05548 )   
## 16384) MAANTHUI < 1.5 700 301.000 No ( 0.94429 0.05571 )   
## 32768) MKOOPKLA < 5.5 488 134.000 No ( 0.96926 0.03074 )   
## 65536) MBERARBG < 6.5 478 119.400 No ( 0.97280 0.02720 )   
## 131072) PPERSAUT < 5.5 315 42.880 No ( 0.98730 0.01270 )   
## 262144) MAUT2 < 2.5 295 23.960 No ( 0.99322 0.00678 )   
## 524288) MSKB1 < 4.5 289 13.330 No ( 0.99654 0.00346 )   
## 1048576) MOPLLAAG < 7.5 249 0.000 No ( 1.00000 0.00000 ) \*  
## 1048577) MOPLLAAG > 7.5 40 9.353 No ( 0.97500 0.02500 )   
## 2097154) MBERMIDD < 3.5 34 0.000 No ( 1.00000 0.00000 ) \*  
## 2097155) MBERMIDD > 3.5 6 5.407 No ( 0.83333 0.16667 ) \*  
## 524289) MSKB1 > 4.5 6 5.407 No ( 0.83333 0.16667 ) \*  
## 262145) MAUT2 > 2.5 20 13.000 No ( 0.90000 0.10000 )   
## 524290) MHKOOP < 6.5 10 0.000 No ( 1.00000 0.00000 ) \*  
## 524291) MHKOOP > 6.5 10 10.010 No ( 0.80000 0.20000 )   
## 1048582) MBERARBO < 1.5 5 0.000 No ( 1.00000 0.00000 ) \*  
## 1048583) MBERARBO > 1.5 5 6.730 No ( 0.60000 0.40000 ) \*  
## 131073) PPERSAUT > 5.5 163 69.630 No ( 0.94479 0.05521 )   
## 262146) APERSAUT < 1.5 147 56.280 No ( 0.95238 0.04762 )   
## 524292) MINK123M < 0.5 135 55.060 No ( 0.94815 0.05185 )   
## 1048584) PBRAND < 2.5 82 0.000 No ( 1.00000 0.00000 ) \*  
## 1048585) PBRAND > 2.5 53 41.370 No ( 0.86792 0.13208 )   
## 2097170) MBERZELF < 0.5 43 30.910 No ( 0.88372 0.11628 )   
## 4194340) MGODPR < 4.5 19 21.900 No ( 0.73684 0.26316 )   
## 8388680) MINK7512 < 0.5 13 11.160 No ( 0.84615 0.15385 )   
## 16777360) MGODRK < 1.5 8 0.000 No ( 1.00000 0.00000 ) \*  
## 16777361) MGODRK > 1.5 5 6.730 No ( 0.60000 0.40000 ) \*  
## 8388681) MINK7512 > 0.5 6 8.318 No ( 0.50000 0.50000 ) \*  
## 4194341) MGODPR > 4.5 24 0.000 No ( 1.00000 0.00000 ) \*  
## 2097171) MBERZELF > 0.5 10 10.010 No ( 0.80000 0.20000 )   
## 4194342) MGODRK < 0.5 5 0.000 No ( 1.00000 0.00000 ) \*  
## 4194343) MGODRK > 0.5 5 6.730 No ( 0.60000 0.40000 ) \*  
## 524293) MINK123M > 0.5 12 0.000 No ( 1.00000 0.00000 ) \*  
## 262147) APERSAUT > 1.5 16 12.060 No ( 0.87500 0.12500 )   
## 524294) MBERARBG < 3.5 8 0.000 No ( 1.00000 0.00000 ) \*  
## 524295) MBERARBG > 3.5 8 8.997 No ( 0.75000 0.25000 ) \*  
## 65537) MBERARBG > 6.5 10 10.010 No ( 0.80000 0.20000 )   
## 131074) MOSTYPE < 0.825 5 0.000 No ( 1.00000 0.00000 ) \*  
## 131075) MOSTYPE > 0.825 5 6.730 No ( 0.60000 0.40000 ) \*  
## 32769) MKOOPKLA > 5.5 212 149.700 No ( 0.88679 0.11321 )   
## 65538) MBERBOER < 0.5 178 137.000 No ( 0.87079 0.12921 )   
## 131076) MINK123M < 0.5 141 111.500 No ( 0.86525 0.13475 )   
## 262152) MBERZELF < 0.5 113 92.170 No ( 0.85841 0.14159 )   
## 524304) MSKD < 0.5 86 73.050 No ( 0.84884 0.15116 )   
## 1048608) MINK7512 < 0.5 55 37.910 No ( 0.89091 0.10909 )   
## 2097216) MBERHOOG < 6.5 48 22.440 No ( 0.93750 0.06250 )   
## 4194432) MINKM30 < 1.5 26 0.000 No ( 1.00000 0.00000 ) \*  
## 4194433) MINKM30 > 1.5 22 17.530 No ( 0.86364 0.13636 )   
## 8388866) MSKA < 2.5 17 7.606 No ( 0.94118 0.05882 )   
## 16777732) MRELSA < 1.5 12 0.000 No ( 1.00000 0.00000 ) \*  
## 16777733) MRELSA > 1.5 5 5.004 No ( 0.80000 0.20000 ) \*  
## 8388867) MSKA > 2.5 5 6.730 No ( 0.60000 0.40000 ) \*  
## 2097217) MBERHOOG > 6.5 7 9.561 No ( 0.57143 0.42857 ) \*  
## 1048609) MINK7512 > 0.5 31 33.120 No ( 0.77419 0.22581 )   
## 2097218) MAUT0 < 0.5 21 23.050 No ( 0.76190 0.23810 )   
## 4194436) MOSHOOFD < 1.5 13 7.051 No ( 0.92308 0.07692 )   
## 8388872) PBRAND < 3.5 8 0.000 No ( 1.00000 0.00000 ) \*  
## 8388873) PBRAND > 3.5 5 5.004 No ( 0.80000 0.20000 ) \*  
## 4194437) MOSHOOFD > 1.5 8 11.090 No ( 0.50000 0.50000 ) \*  
## 2097219) MAUT0 > 0.5 10 10.010 No ( 0.80000 0.20000 )   
## 4194438) MFALLEEN < 1.5 5 0.000 No ( 1.00000 0.00000 ) \*  
## 4194439) MFALLEEN > 1.5 5 6.730 No ( 0.60000 0.40000 ) \*  
## 524305) MSKD > 0.5 27 18.840 No ( 0.88889 0.11111 )   
## 1048610) MBERMIDD < 4.5 21 8.041 No ( 0.95238 0.04762 )   
## 2097220) MGODGE < 4.5 16 0.000 No ( 1.00000 0.00000 ) \*  
## 2097221) MGODGE > 4.5 5 5.004 No ( 0.80000 0.20000 ) \*  
## 1048611) MBERMIDD > 4.5 6 7.638 No ( 0.66667 0.33333 ) \*  
## 262153) MBERZELF > 0.5 28 19.070 No ( 0.89286 0.10714 )   
## 524306) MGODPR < 5.5 17 0.000 No ( 1.00000 0.00000 ) \*  
## 524307) MGODPR > 5.5 11 12.890 No ( 0.72727 0.27273 )   
## 1048614) MGODOV < 0.5 6 5.407 No ( 0.83333 0.16667 ) \*  
## 1048615) MGODOV > 0.5 5 6.730 No ( 0.60000 0.40000 ) \*  
## 131077) MINK123M > 0.5 37 25.350 No ( 0.89189 0.10811 )   
## 262154) MSKB1 < 2.5 26 0.000 No ( 1.00000 0.00000 ) \*  
## 262155) MSKB1 > 2.5 11 14.420 No ( 0.63636 0.36364 )   
## 524310) PPERSAUT < 3 6 7.638 No ( 0.66667 0.33333 ) \*  
## 524311) PPERSAUT > 3 5 6.730 No ( 0.60000 0.40000 ) \*  
## 65539) MBERBOER > 0.5 34 9.023 No ( 0.97059 0.02941 )   
## 131078) MHKOOP < 8.5 28 0.000 No ( 1.00000 0.00000 ) \*  
## 131079) MHKOOP > 8.5 6 5.407 No ( 0.83333 0.16667 ) \*  
## 16385) MAANTHUI > 1.5 75 31.230 No ( 0.94667 0.05333 )   
## 32770) MAANTHUI < 2.5 69 30.550 No ( 0.94203 0.05797 )   
## 65540) MFWEKIND < 8.5 63 17.740 No ( 0.96825 0.03175 )   
## 131080) PBRAND < 1.5 21 13.210 No ( 0.90476 0.09524 )   
## 262160) MZPART < 2.5 13 0.000 No ( 1.00000 0.00000 ) \*  
## 262161) MZPART > 2.5 8 8.997 No ( 0.75000 0.25000 ) \*  
## 131081) PBRAND > 1.5 42 0.000 No ( 1.00000 0.00000 ) \*  
## 65541) MFWEKIND > 8.5 6 7.638 No ( 0.66667 0.33333 ) \*  
## 32771) MAANTHUI > 2.5 6 0.000 No ( 1.00000 0.00000 ) \*  
## 8193) PBROM > 1 55 9.996 No ( 0.98182 0.01818 )   
## 16386) PPERSAUT < 5.5 48 0.000 No ( 1.00000 0.00000 ) \*  
## 16387) PPERSAUT > 5.5 7 5.742 No ( 0.85714 0.14286 ) \*  
## 4097) PMOTSCO > 2 33 15.090 No ( 0.93939 0.06061 )   
## 8194) MOSTYPE < 0.9125 27 0.000 No ( 1.00000 0.00000 ) \*  
## 8195) MOSTYPE > 0.9125 6 7.638 No ( 0.66667 0.33333 ) \*  
## 2049) PLEVEN > 0.5 31 14.830 No ( 0.93548 0.06452 )   
## 4098) MSKC < 4.5 24 0.000 No ( 1.00000 0.00000 ) \*  
## 4099) MSKC > 4.5 7 8.376 No ( 0.71429 0.28571 ) \*  
## 1025) PTRACTOR > 1.5 10 0.000 No ( 1.00000 0.00000 ) \*  
## 513) PWALAND > 1.5 16 0.000 No ( 1.00000 0.00000 ) \*  
## 257) PFIETS > 0.5 17 7.606 No ( 0.94118 0.05882 )   
## 514) MGODRK < 1.5 12 0.000 No ( 1.00000 0.00000 ) \*  
## 515) MGODRK > 1.5 5 5.004 No ( 0.80000 0.20000 ) \*  
## 129) PAANHANG > 0.5 10 6.502 No ( 0.90000 0.10000 ) \*  
## 65) PPERSONG > 1 9 0.000 No ( 1.00000 0.00000 ) \*  
## 33) PBYSTAND > 1 13 14.050 No ( 0.76923 0.23077 )   
## 66) MBERMIDD < 3.5 8 0.000 No ( 1.00000 0.00000 ) \*  
## 67) MBERMIDD > 3.5 5 6.730 Yes ( 0.40000 0.60000 ) \*  
## 17) PGEZONG > 1 9 6.279 No ( 0.88889 0.11111 ) \*  
## 9) PWABEDR > 0.5 9 6.279 No ( 0.88889 0.11111 ) \*  
## 5) PINBOED > 0.5 5 0.000 No ( 1.00000 0.00000 ) \*  
## 3) PPLEZIER > 0.5 8 11.090 No ( 0.50000 0.50000 ) \*

Interpertation: For interpertaion purpose I took the terminal node at the 1048576 position in the tree(internal node), Clearly it is a terminal node because it has \* sign with it. the node is going to assign data examples that reached this node to label No when MOPLLAAG < 7.5 249 and to label Yes otherwise.

1. Produce a pruned tree corresponding to the optimal tree size using cross-validation. Which tree size corresponds to the lowest cross-validated classification error rate? Call this your model #1.

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



cv.d

## $size  
## [1] 63 62 61 60 58 55 54 53 51 50 49 43 41 39 38 34 2 1  
##   
## $dev  
## [1] 806.6980 806.6980 806.6980 806.6980 806.6980 806.6980 806.6980 806.6980  
## [9] 806.6980 806.6980 806.6980 806.6980 806.6980 806.6980 806.6980 806.6980  
## [17] 806.6980 424.1166  
##   
## $k  
## [1] -Inf 0.05231234 0.75410643 2.04689946 2.60239035 2.74882849  
## [7] 3.03662721 3.05928285 3.13660113 3.15811171 3.18898175 3.25807916  
## [13] 3.27793180 3.42640701 3.61628211 3.64919682 3.93616155 12.26827243  
##   
## $method  
## [1] "deviance"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

model\_1=prune.d=prune.tree(tree.d,best=2)   
summary(prune.d)

##   
## Classification tree:  
## snip.tree(tree = tree.d, nodes = 2L)  
## Variables actually used in tree construction:  
## [1] "PPLEZIER"  
## Number of terminal nodes: 2   
## Residual mean deviance: 0.437 = 436.1 / 998   
## Misclassification error rate: 0.059 = 59 / 1000

pred.prune.d=predict(prune.d,test\_data,type="class")  
table(pred.prune.d,test\_data$Purchase)

##   
## pred.prune.d No Yes  
## No 282 18  
## Yes 0 0

tree size=2

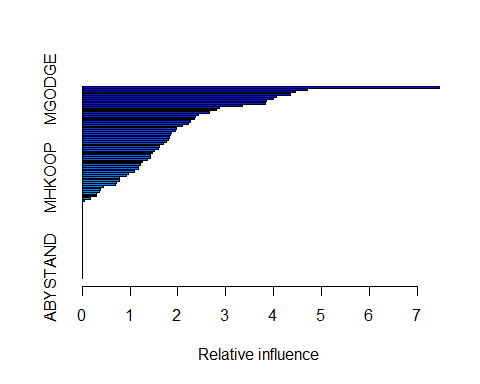
1. Fit a boosting model to the training set with Purchase as the response label and the other variables as features. Use 1,000 trees, and a shrinkage value of 0.01. Call this your model #2. Which predictor appear to be the most important?

train\_data$Purchase = factor(train\_data$Purchase, levels=c("No","Yes"), labels=c(0,1))  
train\_data$Purchase = as.integer(train\_data$Purchase)-1  
  
model\_2= gbm(Purchase ~ ., data=train\_data, distribution="bernoulli",  
n.trees=1000, interaction.depth=4, shrinkage=0.01)

## 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.

summary(model\_2)



## var rel.inf  
## PPERSAUT PPERSAUT 7.470630643  
## MGODGE MGODGE 4.711731856  
## MKOOPKLA MKOOPKLA 4.465713856  
## MOPLHOOG MOPLHOOG 4.349066063  
## MOSTYPE MOSTYPE 4.055714391  
## PBRAND PBRAND 4.004838117  
## MINK3045 MINK3045 3.847254554  
## MBERMIDD MBERMIDD 3.826022011  
## MGODPR MGODPR 3.346766550  
## MBERARBG MBERARBG 2.876262003  
## MSKC MSKC 2.807521364  
## MSKA MSKA 2.650297607  
## MAUT2 MAUT2 2.430456970  
## MSKB1 MSKB1 2.367871627  
## MINK7512 MINK7512 2.352940169  
## MGODOV MGODOV 2.263662272  
## MFGEKIND MFGEKIND 2.217892681  
## MFWEKIND MFWEKIND 2.100219570  
## MRELOV MRELOV 1.974073598  
## MBERARBO MBERARBO 1.958784153  
## MAUT0 MAUT0 1.867785685  
## MRELGE MRELGE 1.854221157  
## MSKB2 MSKB2 1.814082871  
## MINKGEM MINKGEM 1.805489234  
## PWAPART PWAPART 1.769056217  
## MINKM30 MINKM30 1.699308117  
## MFALLEEN MFALLEEN 1.612947930  
## MBERHOOG MBERHOOG 1.591361700  
## MHHUUR MHHUUR 1.507613792  
## MAUT1 MAUT1 1.471996800  
## MZPART MZPART 1.421476110  
## MZFONDS MZFONDS 1.415684057  
## MOPLMIDD MOPLMIDD 1.360295568  
## MRELSA MRELSA 1.249358252  
## MINK4575 MINK4575 1.211076895  
## MGEMLEEF MGEMLEEF 1.180946915  
## ABRAND ABRAND 1.168346161  
## MSKD MSKD 1.091191732  
## MBERZELF MBERZELF 0.960556069  
## MGODRK MGODRK 0.922845190  
## MHKOOP MHKOOP 0.769834623  
## MGEMOMV MGEMOMV 0.767686248  
## APERSAUT APERSAUT 0.722108857  
## MOPLLAAG MOPLLAAG 0.694323775  
## MOSHOOFD MOSHOOFD 0.438663438  
## PMOTSCO PMOTSCO 0.387884766  
## PLEVEN PLEVEN 0.349707934  
## MINK123M MINK123M 0.291411872  
## MBERBOER MBERBOER 0.291238432  
## PBYSTAND PBYSTAND 0.167509803  
## MAANTHUI MAANTHUI 0.046184933  
## ALEVEN ALEVEN 0.009043742  
## PFIETS PFIETS 0.005860660  
## PAANHANG PAANHANG 0.005180414  
## PWABEDR PWABEDR 0.000000000  
## PWALAND PWALAND 0.000000000  
## PBESAUT PBESAUT 0.000000000  
## PVRAAUT PVRAAUT 0.000000000  
## PTRACTOR PTRACTOR 0.000000000  
## PWERKT PWERKT 0.000000000  
## PBROM PBROM 0.000000000  
## PPERSONG PPERSONG 0.000000000  
## PGEZONG PGEZONG 0.000000000  
## PWAOREG PWAOREG 0.000000000  
## PZEILPL PZEILPL 0.000000000  
## PPLEZIER PPLEZIER 0.000000000  
## PINBOED PINBOED 0.000000000  
## AWAPART AWAPART 0.000000000  
## AWABEDR AWABEDR 0.000000000  
## AWALAND AWALAND 0.000000000  
## ABESAUT ABESAUT 0.000000000  
## AMOTSCO AMOTSCO 0.000000000  
## AVRAAUT AVRAAUT 0.000000000  
## AAANHANG AAANHANG 0.000000000  
## ATRACTOR ATRACTOR 0.000000000  
## AWERKT AWERKT 0.000000000  
## ABROM ABROM 0.000000000  
## APERSONG APERSONG 0.000000000  
## AGEZONG AGEZONG 0.000000000  
## AWAOREG AWAOREG 0.000000000  
## AZEILPL AZEILPL 0.000000000  
## APLEZIER APLEZIER 0.000000000  
## AFIETS AFIETS 0.000000000  
## AINBOED AINBOED 0.000000000  
## ABYSTAND ABYSTAND 0.000000000

predictor PPERSAUT appear to be the most important

1. 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%.

set.seed(100)  
glm.probs=predict(model\_2 , test\_data, type = "response")

## Using 1000 trees...

glm.pred <- rep("No", 300)  
glm.pred[glm.probs > .2] = "Yes"

glm.pred[1:10][1:10] # first 10 prediction

## [1] "No" "No" "No" "No" "Yes" "No" "No" "No" "No" "No"

(cm <- table(test\_data$Purchase, glm.pred))

## glm.pred  
## No Yes  
## No 274 8  
## Yes 18 0

1. Fit a radial kernel SVM to the training data with Purchase as the label and the other variables as features. Use a cost value of 0.01, and a gamma value of 0.5 . Call this your model #3.

# Fitting a linear model with cost=0.01  
Model\_3 = svm(Purchase ~ ., data =train\_data, kernel = "radial", gamma=0.5, cost =0.01, scale = FALSE)  
summary(Model\_3)

##   
## Call:  
## svm(formula = Purchase ~ ., data = train\_data, kernel = "radial",   
## gamma = 0.5, cost = 0.01, scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 0.01   
## gamma: 0.5   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 936

svm.pred.test=predict(Model\_3, test\_data, type="response")  
table(svm.pred.test, test\_data$Purchase)

##   
## svm.pred.test No Yes  
## 0.100007718607004 1 0  
## 0.100007795068601 1 0  
## 0.10000779561317 1 0  
## 0.100007795732229 1 0  
## 0.100007795732749 1 0  
## 0.10000795959768 1 0  
## 0.100043674566473 1 0  
## 0.100225270064136 1 0  
## 0.100242963190728 1 0  
## 0.100249621772715 0 1  
## 0.100253687743874 1 0  
## 0.100253695696042 0 1  
## 0.100253704140392 1 0  
## 0.100253710976673 1 0  
## 0.100253714065467 0 1  
## 0.100253714068729 1 0  
## 0.100253714069552 1 0  
## 0.100253714070011 1 0  
## 0.100253714070025 2 0  
## 0.100253714070245 1 0  
## 0.100253715761634 1 0  
## 0.100253831979099 1 0  
## 0.100255604748218 1 0  
## 0.100402871081577 1 0  
## 0.100402871082148 1 0  
## 0.100402940614945 0 1  
## 0.100456544370106 1 0  
## 0.100493370848941 1 0  
## 0.100548493718552 0 1  
## 0.100581150398841 1 0  
## 0.100581486260735 1 0  
## 0.100581488947752 1 0  
## 0.100581491153999 1 0  
## 0.10058149260817 1 0  
## 0.100581508637973 1 0  
## 0.100581935967413 1 0  
## 0.100584489889871 1 0  
## 0.100601672407637 1 0  
## 0.100601678814453 0 1  
## 0.100601678814631 1 0  
## 0.10060190035732 1 0  
## 0.100602151581051 1 0  
## 0.100613774688032 1 0  
## 0.100613922347626 1 0  
## 0.100625791775197 1 0  
## 0.100625824234297 1 0  
## 0.100625852609483 1 0  
## 0.100626167793517 1 0  
## 0.100628576632808 1 0  
## 0.100628578137693 1 0  
## 0.100628584407058 1 0  
## 0.100628584507725 1 0  
## 0.10062858451385 1 0  
## 0.100628584515284 1 1  
## 0.100628584515433 1 0  
## 0.100628584515441 0 1  
## 0.100628586021049 2 0  
## 0.100628590979775 1 0  
## 0.100628627594488 1 0  
## 0.100628855960195 1 0  
## 0.100630240082183 1 0  
## 0.100630241387206 1 0  
## 0.100631246677567 1 0  
## 0.100631248444509 1 0  
## 0.100631463585217 1 0  
## 0.100631463643182 1 0  
## 0.100632112813965 1 0  
## 0.100632226509697 1 0  
## 0.10063222740627 1 0  
## 0.10063222823826 1 0  
## 0.100632449978744 1 0  
## 0.100632450048759 1 0  
## 0.10063245005457 1 0  
## 0.100632586067299 1 0  
## 0.100632586068165 2 0  
## 0.10063259143345 1 0  
## 0.10063265236542 1 0  
## 0.10063265812607 1 0  
## 0.100632668545703 1 0  
## 0.100632668921733 1 0  
## 0.100632718554502 1 0  
## 0.100632718599891 1 0  
## 0.100632718601102 1 0  
## 0.100632718601179 1 0  
## 0.100632718604212 1 0  
## 0.100632738672532 1 0  
## 0.100632748036184 1 0  
## 0.100632748643599 1 0  
## 0.100632749007495 1 0  
## 0.100632766845154 1 0  
## 0.100632767224445 1 0  
## 0.100632767354045 1 0  
## 0.100632767356547 1 0  
## 0.100632767357402 1 0  
## 0.100632767404053 1 0  
## 0.100632778212257 1 0  
## 0.100632778518036 1 0  
## 0.100632781756668 1 0  
## 0.100632785218711 1 0  
## 0.10063278529374 1 0  
## 0.100632786100734 1 0  
## 0.100632787947656 1 0  
## 0.100632787978536 1 0  
## 0.100632789433008 1 0  
## 0.100632791892037 1 0  
## 0.100632793416244 0 1  
## 0.100632793993375 1 0  
## 0.100632794119593 1 0  
## 0.100632794204934 1 0  
## 0.10063279428976 1 0  
## 0.100632794319556 1 0  
## 0.100632794319565 1 0  
## 0.10063279432277 1 0  
## 0.100632794324359 1 0  
## 0.100632794333924 1 0  
## 0.100632794533765 1 0  
## 0.100632794835287 1 0  
## 0.10063279483968 2 0  
## 0.100632794868753 1 0  
## 0.100632794875457 1 0  
## 0.100632795212601 1 0  
## 0.100632795212764 1 0  
## 0.100632795410987 1 0  
## 0.10063279541709 1 0  
## 0.100632795422046 1 0  
## 0.100632795464136 1 0  
## 0.100632795573211 1 0  
## 0.100632795630507 1 0  
## 0.100632795650531 1 0  
## 0.100632795652696 1 0  
## 0.100632795661972 1 0  
## 0.100632795678556 1 0  
## 0.100632795689636 1 0  
## 0.100632795706415 1 0  
## 0.100632795709878 1 0  
## 0.100632795710699 1 0  
## 0.100632795716613 1 0  
## 0.10063279571669 1 0  
## 0.100632795720601 1 0  
## 0.100632795722788 1 0  
## 0.100632795722836 1 0  
## 0.10063279572527 1 0  
## 0.100632795726534 1 0  
## 0.100632795727496 1 0  
## 0.100632795727678 1 0  
## 0.100632795728606 1 0  
## 0.100632795728805 1 0  
## 0.100632795728831 1 0  
## 0.100632795729005 1 0  
## 0.100632795729472 1 0  
## 0.100632795729607 1 0  
## 0.100632795730034 1 0  
## 0.100632795730149 1 0  
## 0.100632795730186 1 0  
## 0.100632795730545 1 0  
## 0.100632795730604 1 0  
## 0.100632795731018 1 0  
## 0.100632795731019 1 0  
## 0.100632795731072 1 0  
## 0.100632795731193 1 0  
## 0.100632795731238 2 0  
## 0.100632795731407 0 1  
## 0.100632795731486 1 0  
## 0.100632795731526 1 0  
## 0.100632795731827 1 0  
## 0.100632795731833 1 0  
## 0.100632795731838 1 0  
## 0.100632795731932 0 1  
## 0.100632795731937 1 0  
## 0.100632795732011 1 0  
## 0.10063279573202 1 0  
## 0.100632795732133 1 0  
## 0.1006327957322 1 0  
## 0.100632795732201 2 0  
## 0.10063279573223 1 0  
## 0.100632795732236 1 0  
## 0.100632795732241 1 0  
## 0.100632795732242 1 0  
## 0.100632795732243 3 0  
## 0.100632795732268 1 1  
## 0.100632795732281 1 0  
## 0.100632795732293 3 0  
## 0.100632795732298 4 0  
## 0.100632795732302 2 0  
## 0.100632795732303 1 0  
## 0.100632795732304 1 0  
## 0.100632795732305 1 0  
## 0.100632795732306 6 0  
## 0.100632795732307 70 5  
## 0.100632795732308 1 0  
## 0.100632795732311 1 0  
## 0.100632795732929 1 0  
## 0.100632795742613 1 0  
## 0.100632795779475 1 0  
## 0.100632795809677 1 0  
## 0.100632796826951 1 0  
## 0.10063279756393 1 0  
## 0.100632832687438 1 0  
## 0.100632846948738 0 1  
## 0.100632869471341 1 0  
## 0.100632897017538 1 0  
## 0.102738751506 1 0  
## 0.110512956873465 1 0  
## 0.110632795710225 1 0  
## 0.110632795718918 1 0

1. Generate confusion matrix for the test set predictions for Model 1,2,3. Discuss your findings. Which model did better in terms of overall test error? How about in terms of Recall?

table(pred.prune.d,test\_data$Purchase)

##   
## pred.prune.d No Yes  
## No 282 18  
## Yes 0 0

(cm <- table(test\_data$Purchase, glm.pred))

## glm.pred  
## No Yes  
## No 274 8  
## Yes 18 0

table(svm.pred.test, test\_data$Purchase)

##   
## svm.pred.test No Yes  
## 0.100007718607004 1 0  
## 0.100007795068601 1 0  
## 0.10000779561317 1 0  
## 0.100007795732229 1 0  
## 0.100007795732749 1 0  
## 0.10000795959768 1 0  
## 0.100043674566473 1 0  
## 0.100225270064136 1 0  
## 0.100242963190728 1 0  
## 0.100249621772715 0 1  
## 0.100253687743874 1 0  
## 0.100253695696042 0 1  
## 0.100253704140392 1 0  
## 0.100253710976673 1 0  
## 0.100253714065467 0 1  
## 0.100253714068729 1 0  
## 0.100253714069552 1 0  
## 0.100253714070011 1 0  
## 0.100253714070025 2 0  
## 0.100253714070245 1 0  
## 0.100253715761634 1 0  
## 0.100253831979099 1 0  
## 0.100255604748218 1 0  
## 0.100402871081577 1 0  
## 0.100402871082148 1 0  
## 0.100402940614945 0 1  
## 0.100456544370106 1 0  
## 0.100493370848941 1 0  
## 0.100548493718552 0 1  
## 0.100581150398841 1 0  
## 0.100581486260735 1 0  
## 0.100581488947752 1 0  
## 0.100581491153999 1 0  
## 0.10058149260817 1 0  
## 0.100581508637973 1 0  
## 0.100581935967413 1 0  
## 0.100584489889871 1 0  
## 0.100601672407637 1 0  
## 0.100601678814453 0 1  
## 0.100601678814631 1 0  
## 0.10060190035732 1 0  
## 0.100602151581051 1 0  
## 0.100613774688032 1 0  
## 0.100613922347626 1 0  
## 0.100625791775197 1 0  
## 0.100625824234297 1 0  
## 0.100625852609483 1 0  
## 0.100626167793517 1 0  
## 0.100628576632808 1 0  
## 0.100628578137693 1 0  
## 0.100628584407058 1 0  
## 0.100628584507725 1 0  
## 0.10062858451385 1 0  
## 0.100628584515284 1 1  
## 0.100628584515433 1 0  
## 0.100628584515441 0 1  
## 0.100628586021049 2 0  
## 0.100628590979775 1 0  
## 0.100628627594488 1 0  
## 0.100628855960195 1 0  
## 0.100630240082183 1 0  
## 0.100630241387206 1 0  
## 0.100631246677567 1 0  
## 0.100631248444509 1 0  
## 0.100631463585217 1 0  
## 0.100631463643182 1 0  
## 0.100632112813965 1 0  
## 0.100632226509697 1 0  
## 0.10063222740627 1 0  
## 0.10063222823826 1 0  
## 0.100632449978744 1 0  
## 0.100632450048759 1 0  
## 0.10063245005457 1 0  
## 0.100632586067299 1 0  
## 0.100632586068165 2 0  
## 0.10063259143345 1 0  
## 0.10063265236542 1 0  
## 0.10063265812607 1 0  
## 0.100632668545703 1 0  
## 0.100632668921733 1 0  
## 0.100632718554502 1 0  
## 0.100632718599891 1 0  
## 0.100632718601102 1 0  
## 0.100632718601179 1 0  
## 0.100632718604212 1 0  
## 0.100632738672532 1 0  
## 0.100632748036184 1 0  
## 0.100632748643599 1 0  
## 0.100632749007495 1 0  
## 0.100632766845154 1 0  
## 0.100632767224445 1 0  
## 0.100632767354045 1 0  
## 0.100632767356547 1 0  
## 0.100632767357402 1 0  
## 0.100632767404053 1 0  
## 0.100632778212257 1 0  
## 0.100632778518036 1 0  
## 0.100632781756668 1 0  
## 0.100632785218711 1 0  
## 0.10063278529374 1 0  
## 0.100632786100734 1 0  
## 0.100632787947656 1 0  
## 0.100632787978536 1 0  
## 0.100632789433008 1 0  
## 0.100632791892037 1 0  
## 0.100632793416244 0 1  
## 0.100632793993375 1 0  
## 0.100632794119593 1 0  
## 0.100632794204934 1 0  
## 0.10063279428976 1 0  
## 0.100632794319556 1 0  
## 0.100632794319565 1 0  
## 0.10063279432277 1 0  
## 0.100632794324359 1 0  
## 0.100632794333924 1 0  
## 0.100632794533765 1 0  
## 0.100632794835287 1 0  
## 0.10063279483968 2 0  
## 0.100632794868753 1 0  
## 0.100632794875457 1 0  
## 0.100632795212601 1 0  
## 0.100632795212764 1 0  
## 0.100632795410987 1 0  
## 0.10063279541709 1 0  
## 0.100632795422046 1 0  
## 0.100632795464136 1 0  
## 0.100632795573211 1 0  
## 0.100632795630507 1 0  
## 0.100632795650531 1 0  
## 0.100632795652696 1 0  
## 0.100632795661972 1 0  
## 0.100632795678556 1 0  
## 0.100632795689636 1 0  
## 0.100632795706415 1 0  
## 0.100632795709878 1 0  
## 0.100632795710699 1 0  
## 0.100632795716613 1 0  
## 0.10063279571669 1 0  
## 0.100632795720601 1 0  
## 0.100632795722788 1 0  
## 0.100632795722836 1 0  
## 0.10063279572527 1 0  
## 0.100632795726534 1 0  
## 0.100632795727496 1 0  
## 0.100632795727678 1 0  
## 0.100632795728606 1 0  
## 0.100632795728805 1 0  
## 0.100632795728831 1 0  
## 0.100632795729005 1 0  
## 0.100632795729472 1 0  
## 0.100632795729607 1 0  
## 0.100632795730034 1 0  
## 0.100632795730149 1 0  
## 0.100632795730186 1 0  
## 0.100632795730545 1 0  
## 0.100632795730604 1 0  
## 0.100632795731018 1 0  
## 0.100632795731019 1 0  
## 0.100632795731072 1 0  
## 0.100632795731193 1 0  
## 0.100632795731238 2 0  
## 0.100632795731407 0 1  
## 0.100632795731486 1 0  
## 0.100632795731526 1 0  
## 0.100632795731827 1 0  
## 0.100632795731833 1 0  
## 0.100632795731838 1 0  
## 0.100632795731932 0 1  
## 0.100632795731937 1 0  
## 0.100632795732011 1 0  
## 0.10063279573202 1 0  
## 0.100632795732133 1 0  
## 0.1006327957322 1 0  
## 0.100632795732201 2 0  
## 0.10063279573223 1 0  
## 0.100632795732236 1 0  
## 0.100632795732241 1 0  
## 0.100632795732242 1 0  
## 0.100632795732243 3 0  
## 0.100632795732268 1 1  
## 0.100632795732281 1 0  
## 0.100632795732293 3 0  
## 0.100632795732298 4 0  
## 0.100632795732302 2 0  
## 0.100632795732303 1 0  
## 0.100632795732304 1 0  
## 0.100632795732305 1 0  
## 0.100632795732306 6 0  
## 0.100632795732307 70 5  
## 0.100632795732308 1 0  
## 0.100632795732311 1 0  
## 0.100632795732929 1 0  
## 0.100632795742613 1 0  
## 0.100632795779475 1 0  
## 0.100632795809677 1 0  
## 0.100632796826951 1 0  
## 0.10063279756393 1 0  
## 0.100632832687438 1 0  
## 0.100632846948738 0 1  
## 0.100632869471341 1 0  
## 0.100632897017538 1 0  
## 0.102738751506 1 0  
## 0.110512956873465 1 0  
## 0.110632795710225 1 0  
## 0.110632795718918 1 0