Bruce Campbell ST-617 Homework 4

Wed Jul 20 13:12:33 2016

```
rm(list = ls())
set.seed(7)
```

Chapter 8

Problem 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.

```
library(ISLR)
train = sample(nrow(Caravan), 1000)

DF <- Caravan
DF$PZEILPL <- NULL
DF$AZEILPL <- NULL
# -----------We remove two variables causing the warning seen below Warning
# messages: 1: In gbm.fit(x, y, offset = offset, distribution =
# distribution, w = w, : variable 60: PZEILPL has no variation. 2: In
# gbm.fit(x, y, offset = offset, distribution = distribution, w = w, :
# variable 81: AZEILPL has no variation.


# gbm requires the classification response to be in {0,1} so we have to
# convert the factor {'No','Yes'} to {0,1}

# We were careful to ensure that the 'No' level was mapped to 0.
f <- DF$Purchase
levels(f) <- c("0", "1")
g <- as.numeric(levels(f)[f])
DF$Purchase <- g

# ----------This is NOT the reccomended way to convert a factor!
# DF$Purchase <-as.numeric(DF$Purchase)-1

DFTrain <- DF[train, ]
DFTest <- DF[-train, ]
```

b)

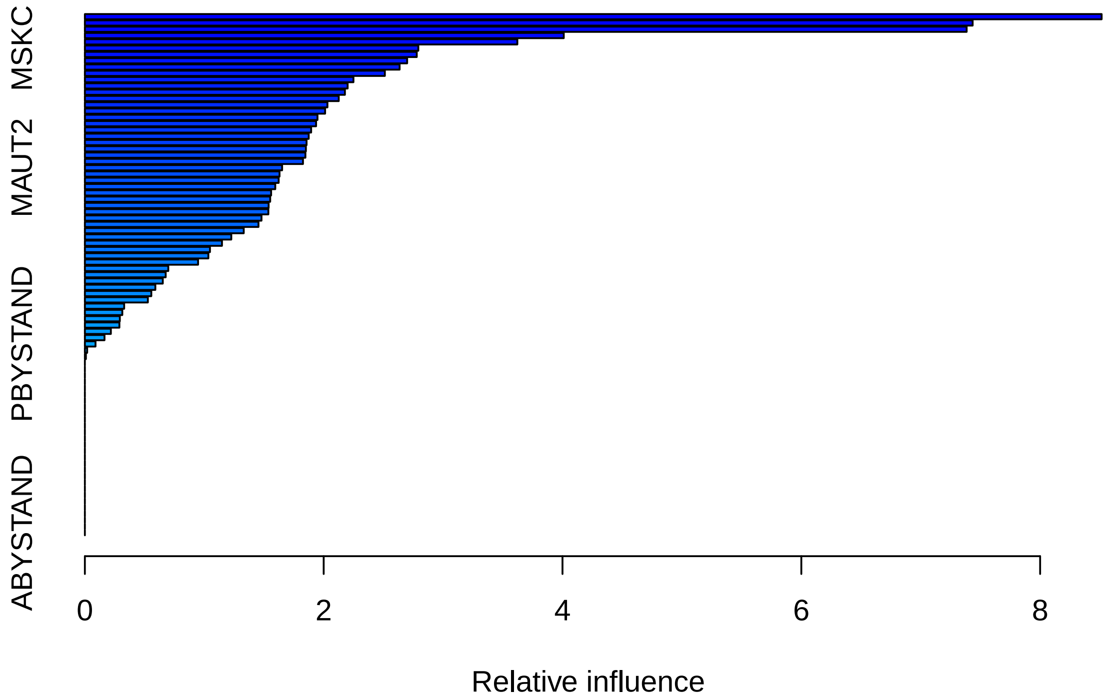
Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

```r
library(gbm)

boost.Caravan = gbm(Purchase ~ ., data = DFTrain, distribution = "bernoulli",
    n.trees = 1000, interaction.depth = 4, shrinkage = 0.01)

summary(boost.Caravan)
```

```
##               var    rel.inf
## PPERSAUT PPERSAUT 8.515086304
## PBRAND     PBRAND 7.434812645
## MOSTYPE   MOSTYPE 7.384544006
## APERSAUT APERSAUT 4.010615821
## MBERMIDD MBERMIDD 3.621893359
## MSKC         MSKC 2.792559830
## MOPLHOOG MOPLHOOG 2.779458022
## PWAPART   PWAPART 2.698927466
## MRELGE     MRELGE 2.635922765
## MBERARBG MBERARBG 2.512422774
## MGODPR     MGODPR 2.249122143
## MINK4575 MINK4575 2.200679917
## MFALLEEN MFALLEEN 2.177916290
## MINKGEM   MINKGEM 2.126173902
## MFWEKIND MFWEKIND 2.031398188
## MOPLMIDD MOPLMIDD 2.011891905
## MAUT1       MAUT1 1.948179274
## MINK7512 MINK7512 1.936422386
## MOPLLAAG MOPLLAAG 1.895075464
## MINKM30   MINKM30 1.874659400
## MKOOPKLA MKOOPKLA 1.856635075
## MZPART     MZPART 1.851024070
## MBERARBO MBERARBO 1.846856906
## MAUT0       MAUT0 1.826453790
## MAUT2       MAUT2 1.652015908
## MBERHOOG MBERHOOG 1.630184344
## MHKOOP     MHKOOP 1.623158838
## MINK3045 MINK3045 1.594416829
## MFGEKIND MFGEKIND 1.559793217
## MZFONDS   MZFONDS 1.551983323
## PMOTSCO   PMOTSCO 1.539013050
## MRELOV     MRELOV 1.536736350
## MGODRK     MGODRK 1.479249367
## MSKB1       MSKB1 1.453893896
## MGODGE     MGODGE 1.330516531
## MSKA         MSKA 1.226121798
## MHHUUR     MHHUUR 1.148179288
## MGODOV     MGODOV 1.047710954
## MSKB2       MSKB2 1.034107681
## ALEVEN     ALEVEN 0.948055193
## MRELSA     MRELSA 0.698661184
## MGEMOMV   MGEMOMV 0.676911985
## MGEMLEEF MGEMLEEF 0.652508573
## MINK123M MINK123M 0.590105543
```

```
## PBROM       PBROM 0.556404874
## MSKD         MSKD 0.527649936
## MBERZELF MBERZELF 0.329071127
## PLEVEN     PLEVEN 0.313524496
## MBERBOER MBERBOER 0.292742603
## PFIETS     PFIETS 0.289100525
## MOSHOOFD MOSHOOFD 0.217688196
## MAANTHUI MAANTHUI 0.164503075
## PBYSTAND PBYSTAND 0.089268027
## PTRACTOR PTRACTOR 0.019213628
## PAANHANG PAANHANG 0.008777961
## PWABEDR   PWABEDR 0.000000000
## PWALAND   PWALAND 0.000000000
## PBESAUT   PBESAUT 0.000000000
## PVRAAUT   PVRAAUT 0.000000000
## PWERKT     PWERKT 0.000000000
## PPERSONG PPERSONG 0.000000000
## PGEZONG   PGEZONG 0.000000000
## PWAOREG   PWAOREG 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
## ABRAND     ABRAND 0.000000000
## APLEZIER APLEZIER 0.000000000
## AFIETS     AFIETS 0.000000000
## AINBOED   AINBOED 0.000000000
## ABYSTAND ABYSTAND 0.000000000
```

```r
summ_gbm <- summary(boost.Caravan)
```




```r
library(pander)
pander(summ_gbm[1:10, ], caption = "Top 10 Features")
```

Table 1: Top 10 Features

	var	rel.inf
PPERSAUT	PPERSAUT	8.515
PBRAND	PBRAND	7.435
MOSTYPE	MOSTYPE	7.385
APERSAUT	APERSAUT	4.011
MBERMIDD	MBERMIDD	3.622
MSKC	MSKC	2.793
MOPLHOOG	MOPLHOOG	2.779
PWAPART	PWAPART	2.699
MRELGE	MRELGE	2.636
MBERARBG	MBERARBG	2.512

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?

```
# predictions are on the scale of f(x). For example, for the Bernoulli loss
# the returned value is on the log odds scale If type='response' then gbm
# converts back to the same scale as the outcome
caravan.probs = predict(boost.Caravan, newdata = DFTrain, n.trees = 1000, type = "response")

caravan.pred = rep(0, nrow(DFTest))
caravan.pred[caravan.probs > 0.2] = 1
TB <- table(caravan.pred, DFTest$Purchase)
library(pander)
pander(TB)
```

	0	1
0	4161	258
1	378	25

```
ACC_Tree = (TB[1] + TB[4])/length(DFTest$Purchase)

Specificity = TB[1]/sum(DFTest$Purchase == 0)
Sensitivity = TB[4]/sum(DFTest$Purchase == 1)
```

The specificity is quite good at `0.9167217` while the sensitivity is quite low at `0.0883392`. If we were looking to predict who *would* purchase a Caravan we need to revisit the data set or try other methods.

Below we try a logistic regression and LDA on the top predictors

```
glm.fit = glm(Purchase ~ PPERSAUT + PBRAND + MOSTYPE + APERSAUT + MBERMIDD +
    PWAPART + MSKC, data = DFTrain, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Purchase ~ PPERSAUT + PBRAND + MOSTYPE + APERSAUT +
##     MBERMIDD + PWAPART + MSKC, family = binomial, data = DFTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0023  -0.3938  -0.2332  -0.1465   3.1898
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.49621    0.57170  -7.865  3.7e-15 ***
## PPERSAUT     0.19534    0.10113   1.932  0.05342 .
## PBRAND       0.23259    0.08640   2.692  0.00711 **
## MOSTYPE     -0.03282    0.01167  -2.811  0.00493 **
```

```
## APERSAUT     0.61853    0.37373   1.655  0.09792 .
## MBERMIDD     0.06000    0.07190   0.834  0.40400
## PWAPART      0.26275    0.16314   1.611  0.10726
## MSKC         0.06025    0.07588   0.794  0.42719
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 481.02  on 999  degrees of freedom
## Residual deviance: 404.98  on 992  degrees of freedom
## AIC: 420.98
##
## Number of Fisher Scoring iterations: 7
```

```r
glm.probs = predict(glm.fit, DFTest, type = "response")

glm.pred = rep(0, nrow(DFTest))
glm.pred[glm.probs > 0.5] = 1
TB <- table(glm.pred, DFTest$Purchase)
pander(TB, caption = "Logistic Regression")
```

Table 3: Logistic Regression

	0	1
0	4534	283
1	5	0

```r
ACC_Tree = (TB[1] + TB[4])/length(DFTest$Purchase)

Specificity = TB[1]/sum(DFTest$Purchase == 0)
Sensitivity = TB[4]/sum(DFTest$Purchase == 1)
```

The sensitivity using logistic with the regression is 0.

e) Repeat (d) using LDA.

```r
library(MASS)
lda.fit = lda(Purchase ~ PPERSAUT + PBRAND + MOSTYPE + APERSAUT + MBERMIDD +
    PWAPART + MSKC, data = DFTrain)
lda.fit
```

```
## Call:
## lda(Purchase ~ PPERSAUT + PBRAND + MOSTYPE + APERSAUT + MBERMIDD +
##     PWAPART + MSKC, data = DFTrain)
##
## Prior probabilities of groups:
##     0     1
## 0.935 0.065
```

```
##
## Group means:
##   PPERSAUT   PBRAND  MOSTYPE  APERSAUT MBERMIDD   PWAPART     MSKC
## 0 2.980749 1.762567 24.89305 0.5614973 2.746524 0.7561497 3.791444
## 1 5.215385 2.923077 19.16923 1.0461538 3.061538 1.3692308 3.507692
##
## Coefficients of linear discriminants:
##                  LD1
## PPERSAUT  0.01019238
## PBRAND    0.21498877
## MOSTYPE  -0.03144078
## APERSAUT  1.11311576
## MBERMIDD  0.03629933
## PWAPART   0.26577271
## MSKC      0.03147300
```

```r
lda.pred = predict(lda.fit, DFTest)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```r
lda.class = lda.pred$class
TB <- table(lda.class, DFTest$Purchase)
pander(TB, caption = "LDA")
```

Table 4: LDA

	0	1
0	4501	275
1	38	8

```r
ACC_Tree = (TB[1] + TB[4])/length(DFTest$Purchase)

Specificity = TB[1]/sum(DFTest$Purchase == 0)
Sensitivity = TB[4]/sum(DFTest$Purchase == 1)
```

LDA does a little better than Logistic regression, but we see the boosted tree classifier performs the best of the models we've considered.