# Tree-Based Methods HW

*Megha Pandit*

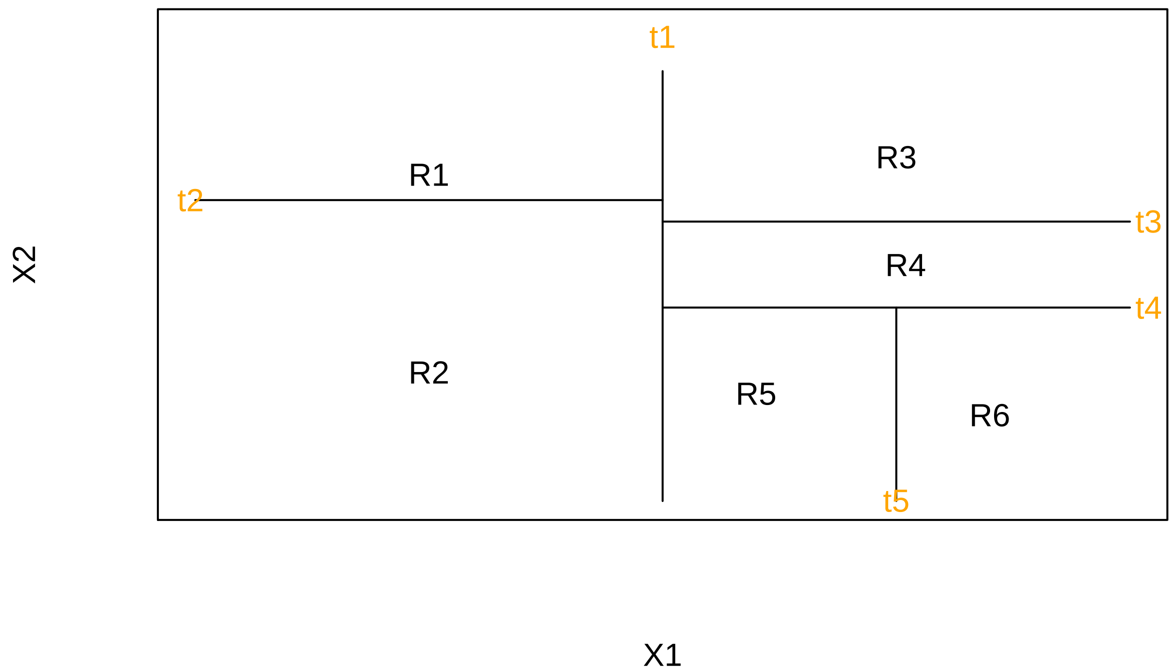
*March 3, 2019*

## Problem 1

```r
data <- data.frame(c(25,25,75,76,60,85), c(76,30,80,55,25,20))
plot(data, xlim = c(0,100), ylim = c(0,110), xlab = "X1", ylab = "X2", xaxt = "n", yaxt = "n", pch = ""
lines(x = c(50,50), y = c(0,100))
lines(x = c(0,50), y = c(70,70))
lines(x = c(50,100), y = c(65,65))
lines(x = c(50,100), y = c(45,45))
lines(x = c(75,75), y = c(0,45))

text(data, labels = paste("R", 1:6, sep = ""))

text(x = 50, y = 108, labels = c("t1"), col = "orange")
text(x = -0.5, y = 70, labels = c("t2"), col = "orange")
text(x = 102, y = 65, labels = c("t3"), col = "orange")
text(x = 102, y = 45, labels = c("t4"), col = "orange")
text(x = 75, y = 0, labels = c("t5"), col = "orange")
```



## Problem 2

```r
include_graphics("C:/Users/GP/Desktop/MEGHA/SemII/MA679 - Appl Stat Learning/Homework/MA679---Stat-Machi
```

2: Considering only stumps, at the first step,

$$\hat{f}_1^*(x) = c_1 I(x_1 < t_1) + c_1' = \frac{1}{\lambda} f_1(x_1),$$

$\therefore \hat{f}(x) = \lambda \hat{f}_1(x)$ and residuals $r_i = y_i - \lambda \hat{f}_1(x)$

In the second step,

$$\hat{f}_2(x) = c_2 I(x_2 < t_2) + c_2' = \frac{1}{\lambda} f_2(x_2)$$

To update the residuals after the second step:

$$\hat{f}(x) = \lambda \hat{f}_1(x) + \lambda \hat{f}_2(x)$$

and $r_i = y_i - \lambda \hat{f}_1(x_i) - \lambda \hat{f}_2(x_i)$ for $i$.

~~Therefore~~

Therefore, we have:

$$\hat{f}(x) = \sum_{j=1}^{p} f_j(x_j)$$

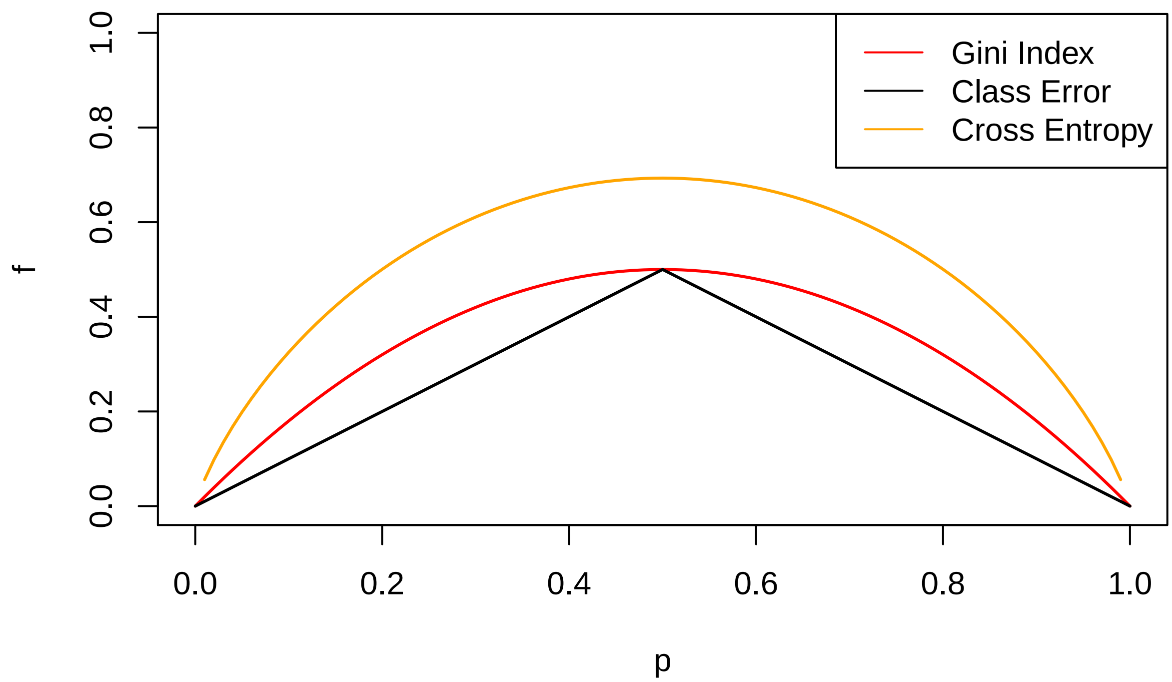
## Problem 3

```r
p <- seq(0,1,0.01)

#For two classes
gini <- 2*p*(1-p)
error <- 1 - pmax(p, 1-p)
entropy <- -(p*log(p) + (1-p)*log(1-p))

plot(NA, xlim = c(0,1), ylim = c(0,1), xlab = "p", ylab = "f")
lines(p, gini, type = "l", col = "red", lwd = 1.5)
lines(p, error, type = "l", col = "black", lwd = 1.5)
lines(p, entropy, type = "l", col = "orange", lwd = 1.5)

legend(x = "topright", legend = c("Gini Index", "Class Error", "Cross Entropy"),
       col = c("red", "black", "orange"), lty = 1)
```




## Problem 5

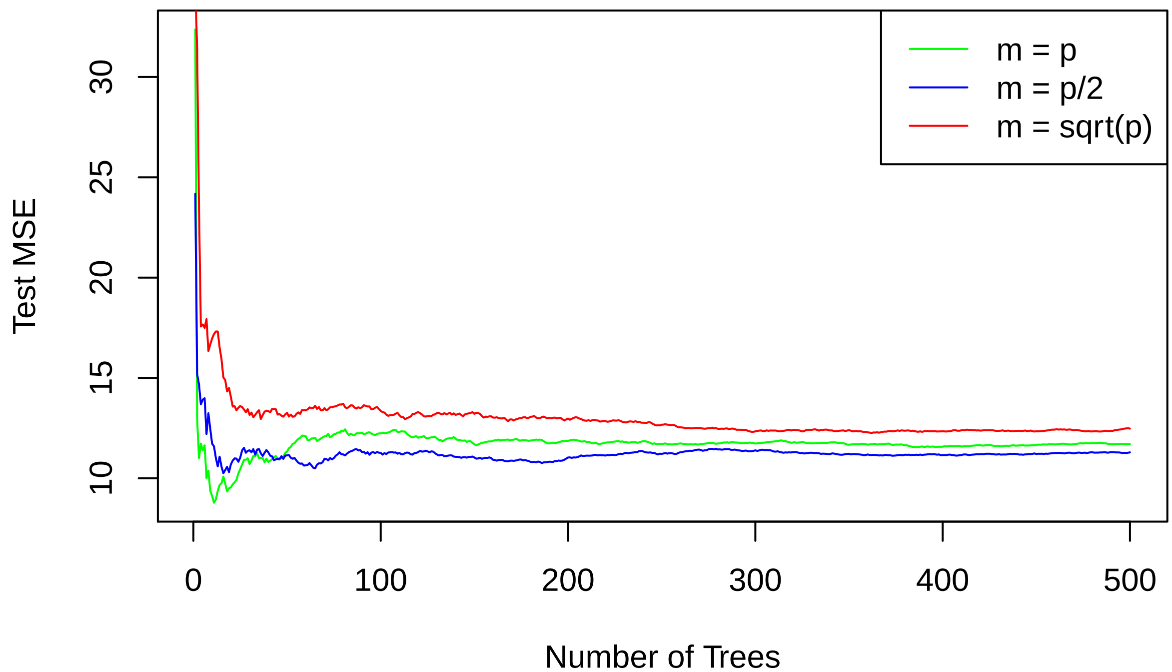
P(Class is Red | X) is greater than 0.5 in 6 of the 10 times. Therefore, according to the majority vote way, the final classification is Red. According to the approach based on average probability, the average probability for the 10 estimates is 0.45, i.e., P(Class is Red | X) $< 0.5$, implying that the final classification is Green.

## Problem 7

```
data("Boston")
set.seed(9)
train <- sample(1:nrow(Boston), nrow(Boston)/2)
Boston_train <- Boston[train, -14]
Boston_test <- Boston[-train, -14]
y_train <- Boston[train, 14]
y_test <- Boston[-train, 14]

rf1 <- randomForest(Boston_train, y = y_train, xtest = Boston_test, ytest = y_test, mtry = ncol(Boston)
rf2 <- randomForest(Boston_train, y = y_train, xtest = Boston_test, ytest = y_test, mtry = (ncol(Boston)
rf3 <- randomForest(Boston_train, y = y_train, xtest = Boston_test, ytest = y_test, mtry = sqrt(ncol(Bo
```

```r
plot(1:500, rf1$test$mse, type = "l", col = "green", xlab = "Number of Trees", ylab = "Test MSE")
lines(1:500, rf2$test$mse, type = "l", col = "blue")
lines(1:500, rf3$test$mse, type = "l", col = "red")
legend(x = "topright", c("m = p", "m = p/2", "m = sqrt(p)"), col = c("green", "blue", "red"), lty = 1)
```




## Problem 8

### (a)

```r
data("Carseats")
set.seed(9)
subs <- sample(1:nrow(Carseats), nrow(Carseats)*0.7)
car_train <- Carseats[subs, ]
car_test <- Carseats[-subs, ]
```

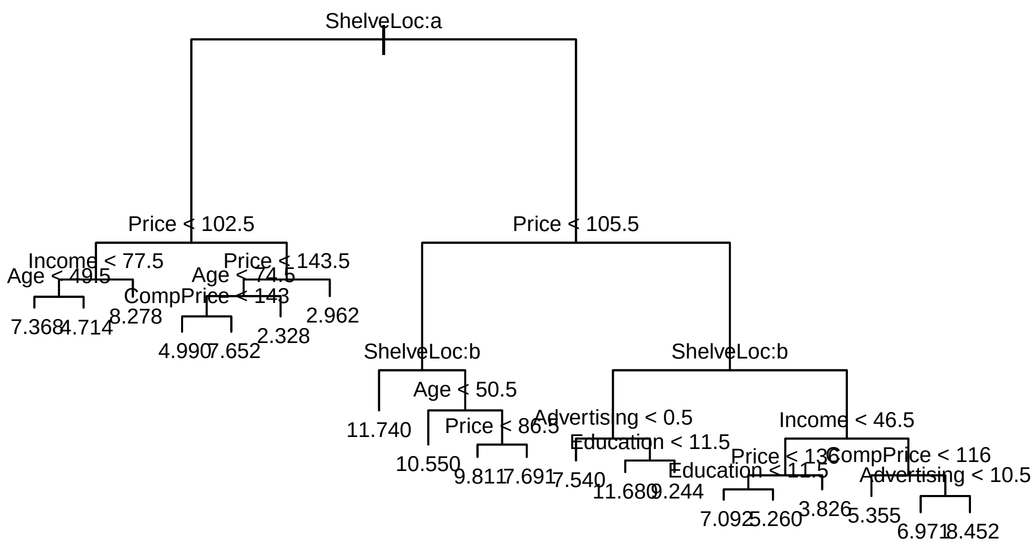
### (b)

```r
#Regression Tree

rtree <- tree(Sales ~ ., data = car_train)
summary(rtree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = car_train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"        "Income"       "Age"          "CompPrice"
## [6] "Advertising" "Education"
## Number of terminal nodes:  20
## Residual mean deviance:  2.317 = 602.5 / 260
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.9600 -0.9205 -0.1062  0.0000  1.0170  3.4400
```

```r
plot(rtree)
text(rtree, cex = 0.65)
```



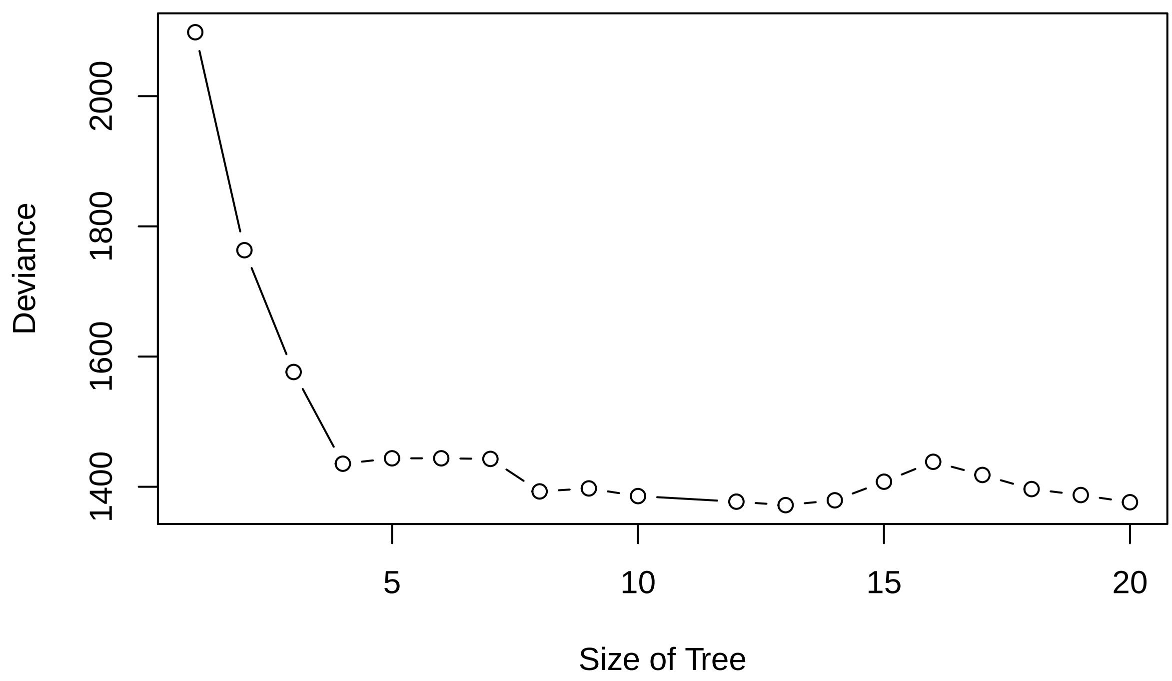

```r
#MSE
pred_rtree <- predict(rtree, car_test)
mse_rtree <- mean((car_test$Sales - pred_rtree)^2)
print(paste0("The test MSE for the regression tree is: ", mse_rtree))
```

```
## [1] "The test MSE for the regression tree is: 4.86831249805261"
```

### (c)

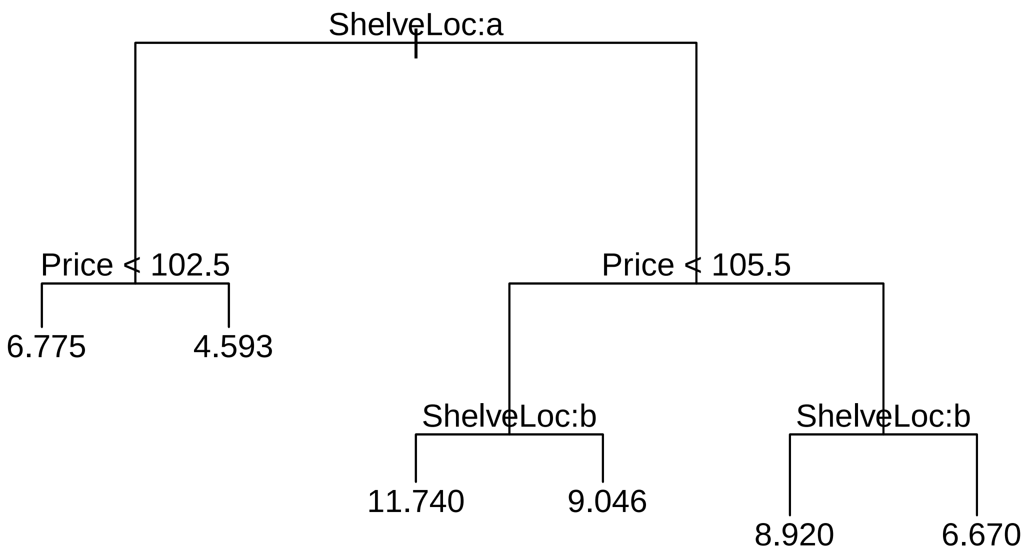
```r
#Cross-Validation for tree complexity

cv_rtree <- cv.tree(rtree)
plot(cv_rtree$size, cv_rtree$dev, xlab = "Size of Tree", ylab = "Deviance", type = "b")
```




```r
#Tree Pruning

prune_rtree <- prune.tree(rtree, best = 6)
plot(prune_rtree)
text(prune_rtree)
```




```r
#Test MSE for pruned tree

prune_pred <- predict(prune_rtree, car_test)
prune_mse <- mean((prune_pred - car_test$Sales)^2)
print(paste0("The test MSE for the pruned tree is: ", prune_mse))
```

```
## [1] "The test MSE for the pruned tree is: 4.67886020938024"
```

**The pruned tree gives a slightly lower MSE than the unpruned tree.**

## (d)

```r
#Bagging

car_bag <- randomForest(Sales ~ ., data = car_train, mtry = 10, importance = TRUE, ntree = 500)
pred_bag <- predict(car_bag, car_test)
bag_mse <- mean((pred_bag - car_test$Sales)^2)

print(paste0("The test MSE for bagging method is: ", bag_mse))
```

```
## [1] "The test MSE for bagging method is: 2.91140179244083"
```

**Bagging reduces the test MSE to 2.936**

```r
#Importance
importance(car_bag)
```

```
##               %IncMSE IncNodePurity
## CompPrice   24.872266    208.178505
## Income      11.746221    157.559249
## Advertising 19.434843    152.510222
## Population   2.081086     90.758372
## Price       57.326842    564.408982
## ShelveLoc   61.044251    571.411663
## Age         17.922860    188.529126
## Education    3.554631     63.675882
## Urban       -1.494946      9.849759
## US           1.859172      9.312038
```

**Price and ShelveLoc seem to be the two most important variables.**

# (e)

```r
#Random Forest

rf_mse <- c()
for (i in 1:10) {
  car_rf <- randomForest(Sales ~ ., data = car_train, mtry = i, importance = TRUE, ntree = 500)
  pred_rf <- predict(car_rf, car_test)
  rf_mse[i] <- mean((pred_rf - car_test$Sales)^2)
}

#Best model
which.min(rf_mse)
```

```
## [1] 10
```

```r
#Minimum MSE
rf_mse[which.min(rf_mse)]
```

```
## [1] 2.935711
```

**The best model uses 10 variables at each split. It does not quite reduce the test MSE compared to Bagging.**

```r
importance(car_rf)
```

```
##               %IncMSE IncNodePurity
## CompPrice   24.609575    205.110685
## Income      10.801320    166.771105
## Advertising 21.503330    154.491530
## Population   1.535275     91.653320
## Price       54.085754    565.725634
## ShelveLoc   60.944360    564.692752
## Age         19.764836    188.763535
## Education    3.063798     64.841276
## Urban       -2.619948      9.746756
## US           2.213438     10.712297
```

**ShelveLoc seems to be the most important variable, followed by Price.**

## Problem 11

### (a)

```r
data("Caravan")
Caravan$Purchase <- ifelse(Caravan$Purchase == "No", 0, 1)
crv_train <- Caravan[1:1000, ]
crv_test <- Caravan[1001:5822, ]
```

### (b)

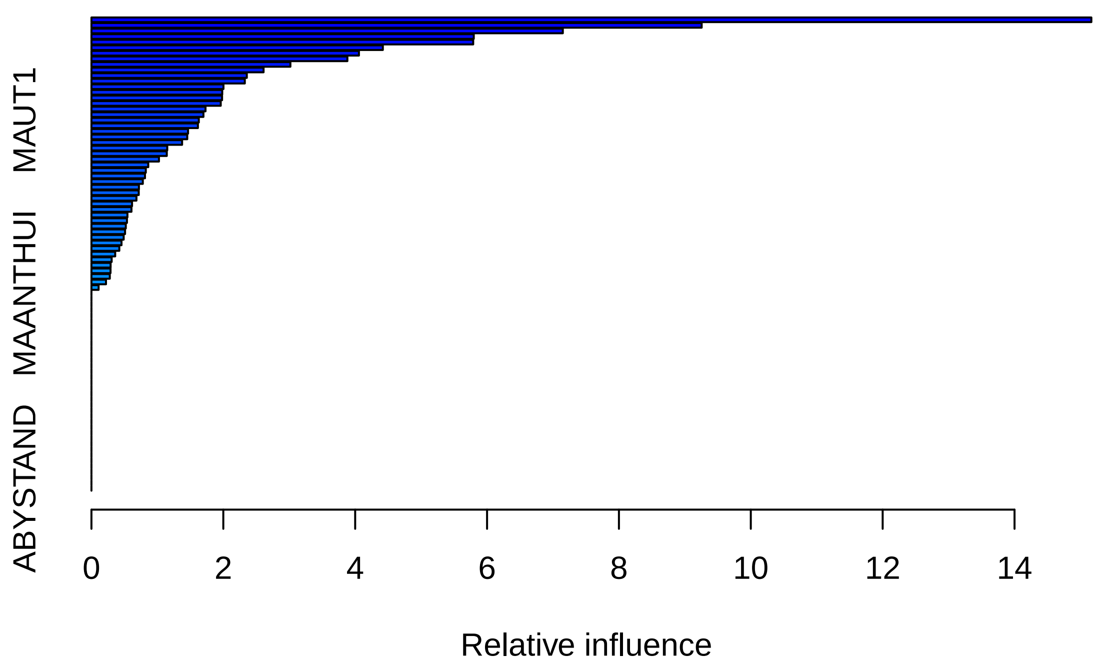
```r
#Boosting

set.seed(9)
boost <- gbm(Purchase ~ ., data = crv_train, shrinkage = 0.01, n.trees = 1000, distribution = "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.
```

```r
kable(summary(boost), row.names = F)
```



| var | rel.inf |
|---|---|
| PPERSAUT | 15.1650119 |
| MKOOPKLA | 9.2549208 |
| MOPLHOOG | 7.1491523 |
| MBERMIDD | 5.7988033 |
| PBRAND | 5.7897473 |
| MGODGE | 4.4197200 |
| MINK3045 | 4.0569774 |
| ABRAND | 3.8816596 |
| MOSTYPE | 3.0173120 |
| MSKA | 2.6104471 |
| MSKC | 2.3567316 |
| MAUT2 | 2.3263968 |
| PWAPART | 2.0023871 |
| MINKGEM | 1.9838691 |
| MBERARBG | 1.9814157 |
| MGODPR | 1.9612263 |
| MGODOV | 1.7300166 |
| MFWEKIND | 1.6986371 |
| MAUT1 | 1.6287004 |
| PBYSTAND | 1.6148436 |
| MSKB1 | 1.4654283 |
| MRELGE | 1.4532182 |
| MBERHOOG | 1.3751342 |
| MHHUUR | 1.1499571 |
| MRELOV | 1.1429171 |
| APERSAUT | 1.0241970 |
| MOSHOOFD | 0.8617721 |
| MINK7512 | 0.8244418 |
| MFGEKIND | 0.8122327 |
| MSKD | 0.7794502 |
| MGODRK | 0.7204269 |
| MAUT0 | 0.7176358 |
| MINKM30 | 0.6825040 |
| MHKOOP | 0.6160159 |
| MOPLMIDD | 0.6069681 |
| MBERARBO | 0.5460070 |
| MINK123M | 0.5388234 |
| MBERBOER | 0.5211953 |
| MGEMOMV | 0.5101719 |
| MGEMLEEF | 0.4889504 |
| MINK4575 | 0.4563592 |
| MFALLEEN | 0.4226122 |
| PMOTSCO | 0.3601726 |
| MSKB2 | 0.3067847 |
| MZFONDS | 0.2906486 |
| MZPART | 0.2897102 |
| MOPLLAAG | 0.2787625 |
| PLEVEN | 0.2207685 |
| MRELSA | 0.1087580 |
| MAANTHUI | 0.0000000 |
| MBERZELF | 0.0000000 |
| PWABEDR | 0.0000000 |

| var | rel.inf |
|---|---|
| PWALAND | 0.0000000 |
| PBESAUT | 0.0000000 |
| PVRAAUT | 0.0000000 |
| PAANHANG | 0.0000000 |
| PTRACTOR | 0.0000000 |
| PWERKT | 0.0000000 |
| PBROM | 0.0000000 |
| PPERSONG | 0.0000000 |
| PGEZONG | 0.0000000 |
| PWAOREG | 0.0000000 |
| PZEILPL | 0.0000000 |
| PPLEZIER | 0.0000000 |
| PFIETS | 0.0000000 |
| PINBOED | 0.0000000 |
| AWAPART | 0.0000000 |
| AWABEDR | 0.0000000 |
| AWALAND | 0.0000000 |
| ABESAUT | 0.0000000 |
| AMOTSCO | 0.0000000 |
| AVRAAUT | 0.0000000 |
| AAANHANG | 0.0000000 |
| ATRACTOR | 0.0000000 |
| AWERKT | 0.0000000 |
| ABROM | 0.0000000 |
| ALEVEN | 0.0000000 |
| APERSONG | 0.0000000 |
| AGEZONG | 0.0000000 |
| AWAOREG | 0.0000000 |
| AZEILPL | 0.0000000 |
| APLEZIER | 0.0000000 |
| AFIETS | 0.0000000 |
| AINBOED | 0.0000000 |
| ABYSTAND | 0.0000000 |

**PPERSAUT, MKOOPKLA and MOPLHOOG are the three most important variables.**

## (c)

```r
pred_boost <- predict(boost, crv_test, n.trees = 1000, type = "response")
boost_pred <- ifelse(pred_boost > 0.2, 1, 0)
table(crv_test$Purchase, boost_pred)
```

```
##    boost_pred
##        0    1
##   0 4415  118
##   1  253   36
```

**The fraction of people who were predicted to make a purchase and who actually made a purchase is 36/(36 + 118), which is 0.2337 or 23.37%.**

```r
#Logistic Regression

crv_glm <- glm(Purchase ~ ., data = crv_train, family = binomial)
```

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

```r
pred_glm <- predict(crv_glm, crv_test, type = "response")
```

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

```r
glm_pred <- ifelse(pred_glm > 0.2, 1, 0)
table(crv_test$Purchase, glm_pred)
```

```
##    glm_pred
##        0    1
##   0 4183  350
##   1  231   58
```

**From Logistic Regression, the fraction of people predicted to make a purchase and who actually made a purchase is 58/(58 + 350), which is 0.1421 or 14.21%. Logistic regression performs worse than Boosting in this scenario.**