Exercise 10

Nikolaus Czernin

```
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library("knitr")
library(rpart)
library(mgcv)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
library("ISLR")
# install.packages("randomForest")
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:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed(11721138)
```

Loading & Preprocessing

```r
data("Caravan")

Caravan %>% head()
```

```
##   MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE
## 1      33        1       3        2        8      0      5      1      3      7
## 2      37        1       2        2        8      1      4      1      4      6
## 3      37        1       2        2        8      0      4      2      4      3
## 4       9        1       3        3        3      2      3      2      4      5
## 5      40        1       4        2       10      1      4      1      4      7
## 6      23        1       2        1        5      0      5      0      5      0
##   MRELSA MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG MBERHOOG
## 1      0      2        1        2        6        1        2        7        1
## 2      2      2        0        4        5        0        5        4        0
## 3      2      4        4        4        2        0        5        4        0
## 4      2      2        2        3        4        3        4        2        4
## 5      1      2        2        4        4        5        4        0        0
## 6      6      3        3        5        2        0        5        4        2
##   MBERZELF MBERBOER MBERMIDD MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD
## 1        0        1        2        5        2    1     1     2    6    1
## 2        0        0        5        0        4    0     2     3    5    0
## 3        0        0        7        0        2    0     5     0    4    0
## 4        0        0        3        1        2    3     2     1    4    0
## 5        5        4        0        0        0    9     0     0    0    0
## 6        0        0        4        2        2    2     2     2    4    2
##   MHHUUR MHKOOP MAUT1 MAUT2 MAUT0 MZFONDS MZPART MINKM30 MINK3045 MINK4575
## 1      1      8     8     0     1       8      1       0        4        5
## 2      2      7     7     1     2       6      3       2        0        5
## 3      7      2     7     0     2       9      0       4        5        0
## 4      5      4     9     0     0       7      2       1        5        3
## 5      4      5     6     2     1       5      4       0        0        9
## 6      9      0     5     3     3       9      0       5        2        3
##   MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART PWABEDR PWALAND PPERSAUT PBESAUT
## 1        0        0       4        3       0       0       0        6       0
## 2        2        0       5        4       2       0       0        0       0
## 3        0        0       3        4       2       0       0        6       0
## 4        0        0       4        4       0       0       0        6       0
## 5        0        0       6        3       0       0       0        0       0
## 6        0        0       3        3       0       0       0        6       0
##   PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN PPERSONG PGEZONG
## 1       0       0        0        0      0     0      0        0       0
## 2       0       0        0        0      0     0      0        0       0
## 3       0       0        0        0      0     0      0        0       0
## 4       0       0        0        0      0     0      0        0       0
## 5       0       0        0        0      0     0      0        0       0
## 6       0       0        0        0      0     0      0        0       0
```

```
##   PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART AWABEDR
## 1       0      5       0        0      0       0        0       0       0
## 2       0      2       0        0      0       0        0       2       0
## 3       0      2       0        0      0       0        0       1       0
## 4       0      2       0        0      0       0        0       0       0
## 5       0      6       0        0      0       0        0       0       0
## 6       0      0       0        0      0       0        0       0       0
##   AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM
## 1       0        1       0       0       0        0        0      0     0
## 2       0        0       0       0       0        0        0      0     0
## 3       0        1       0       0       0        0        0      0     0
## 4       0        1       0       0       0        0        0      0     0
## 5       0        0       0       0       0        0        0      0     0
## 6       0        1       0       0       0        0        0      0     0
##   ALEVEN APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED
## 1      0        0       0       0      1       0        0      0       0
## 2      0        0       0       0      1       0        0      0       0
## 3      0        0       0       0      1       0        0      0       0
## 4      0        0       0       0      1       0        0      0       0
## 5      0        0       0       0      1       0        0      0       0
## 6      0        0       0       0      0       0        0      0       0
##   ABYSTAND Purchase
## 1        0       No
## 2        0       No
## 3        0       No
## 4        0       No
## 5        0       No
## 6        0       No
```

```
Caravan <- Caravan %>%
  mutate(Purchase = ifelse(grepl("Yes", Purchase), 1, 0))
  # mutate(Purchase = factor(Purchase))
# Caravan$Purchase



N <- nrow(Caravan)
train_idx <- sample(1:N, N%/%3*2)
train <- Caravan[train_idx,]
test <- Caravan[-train_idx,]
```

```
eval_ <- function(y, yhat){
  conf.mat <- table(y, yhat)
  TP <- conf.mat[2, 2]
  FP <- conf.mat[1, 2]
  FN <- conf.mat[2, 1]
  TN <- conf.mat[1, 1]
  return (list(TP=TP, FP=FP, FN=FN, TN=TN))
}

# RMSE
RMSE <- function(y, yhat){
  (y - yhat)^2 %>% mean() %>% sqrt()
}
```

```r
# balanced accuracy: (TPR+TNR)/2
BACC <- function(y, yhat){
  metrics <- eval_(y, yhat)
  TPR <-  metrics$TP / (metrics$TP + metrics$FN)
  TNR <-  metrics$TN / (metrics$TN + metrics$FP)
  (TPR + TNR) / 2
}
```

Task 1

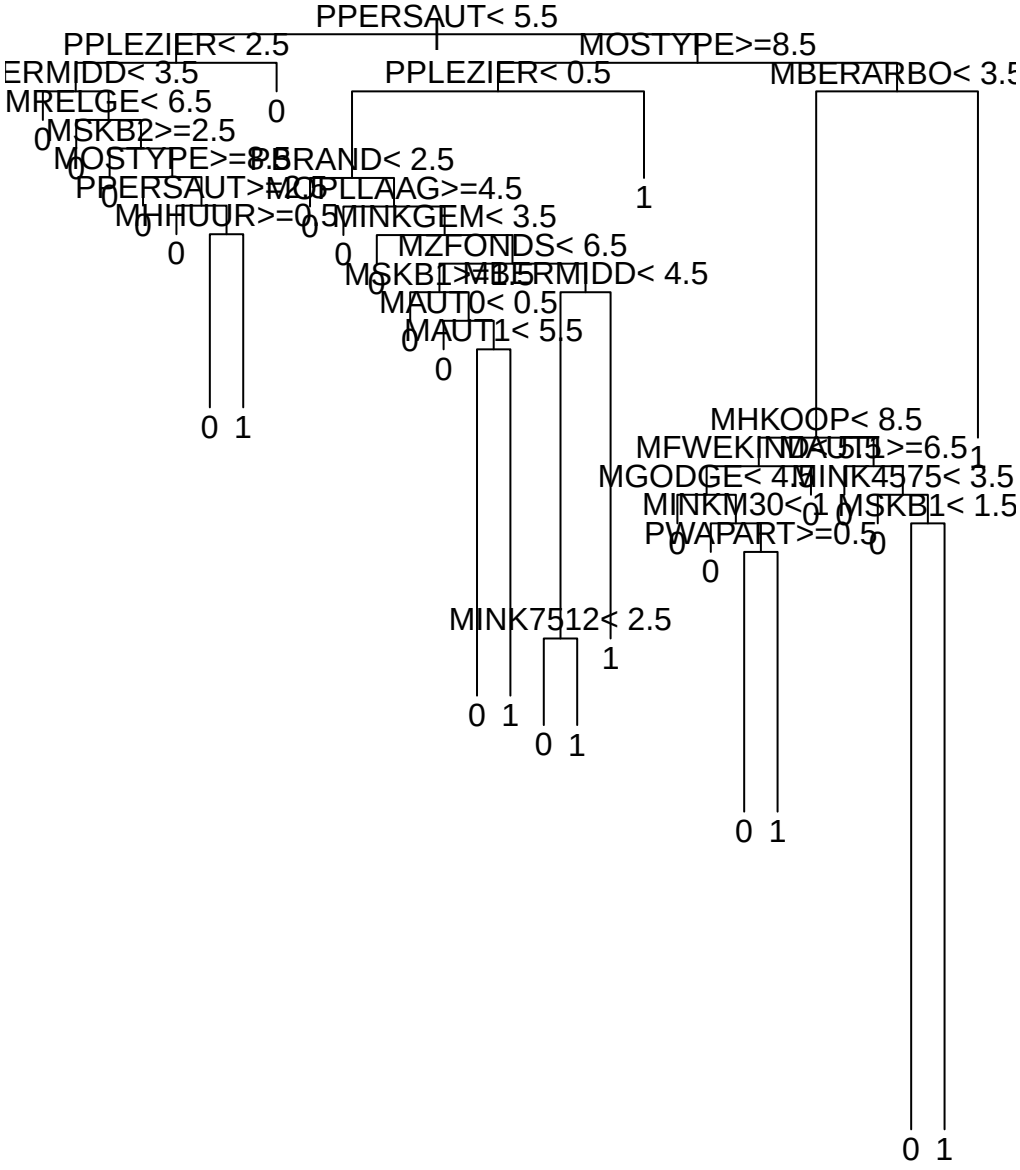
Initial Tree

```r
t0 <- rpart(Purchase~., data=train, cp=0.001, xval=20, method="class")
t0
```

```
## n= 3880
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##    1) root 3880 227 0 (0.94149485 0.05850515)
##      2) PPERSAUT< 5.5 2290  56 0 (0.97554585 0.02445415)
##        4) PPLEZIER< 2.5 2281  53 0 (0.97676458 0.02323542)
##          8) MBERMIDD< 3.5 1564  21 0 (0.98657289 0.01342711) *
##          9) MBERMIDD>=3.5 717  32 0 (0.95536960 0.04463040)
##           18) MRELGE< 6.5 383   7 0 (0.98172324 0.01827676) *
##           19) MRELGE>=6.5 334  25 0 (0.92514970 0.07485030)
##             38) MSKB2>=2.5 183   6 0 (0.96721311 0.03278689) *
##             39) MSKB2< 2.5 151  19 0 (0.87417219 0.12582781)
##               78) MOSTYPE>=8.5 116   9 0 (0.92241379 0.07758621) *
##               79) MOSTYPE< 8.5 35  10 0 (0.71428571 0.28571429)
##                158) PPERSAUT>=2.5 8   0 0 (1.00000000 0.00000000) *
##                159) PPERSAUT< 2.5 27  10 0 (0.62962963 0.37037037)
##                  318) MHHUUR>=0.5 17   4 0 (0.76470588 0.23529412) *
##                  319) MHHUUR< 0.5 10   4 1 (0.40000000 0.60000000) *
##        5) PPLEZIER>=2.5 9   3 0 (0.66666667 0.33333333) *
##      3) PPERSAUT>=5.5 1590 171 0 (0.89245283 0.10754717)
##        6) MOSTYPE>=8.5 1287 109 0 (0.91530692 0.08469308)
##         12) PPLEZIER< 0.5 1278 104 0 (0.91862285 0.08137715)
##           24) PBRAND< 2.5 654  33 0 (0.94954128 0.05045872) *
##           25) PBRAND>=2.5 624  71 0 (0.88621795 0.11378205)
##             50) MOPLLAAG>=4.5 367  27 0 (0.92643052 0.07356948) *
##             51) MOPLLAAG< 4.5 257  44 0 (0.82879377 0.17120623)
##              102) MINKGEM< 3.5 66   4 0 (0.93939394 0.06060606) *
##              103) MINKGEM>=3.5 191  40 0 (0.79057592 0.20942408)
##                206) MZFONDS< 6.5 139  22 0 (0.84172662 0.15827338)
##                  412) MSKB1>=1.5 99  11 0 (0.88888889 0.11111111) *
##                  413) MSKB1< 1.5 40  11 0 (0.72500000 0.27500000)
##                    826) MAUT0< 0.5 19   1 0 (0.94736842 0.05263158) *
##                    827) MAUT0>=0.5 21  10 0 (0.52380952 0.47619048)
```

```
##                     1654) MAUT1< 5.5 7   1 0 (0.85714286 0.14285714) *
##                     1655) MAUT1>=5.5 14   5 1 (0.35714286 0.64285714) *
##                 207) MZFONDS>=6.5 52  18 0 (0.65384615 0.34615385)
##                   414) MBERMIDD< 4.5 38   9 0 (0.76315789 0.23684211)
##                     828) MINK7512< 2.5 31   5 0 (0.83870968 0.16129032) *
##                     829) MINK7512>=2.5 7   3 1 (0.42857143 0.57142857) *
##                   415) MBERMIDD>=4.5 14   5 1 (0.35714286 0.64285714) *
##          13) PPLEZIER>=0.5 9   4 1 (0.44444444 0.55555556) *
##        7) MOSTYPE< 8.5 303  62 0 (0.79537954 0.20462046)
##          14) MBERARBO< 3.5 289  53 0 (0.81660900 0.18339100)
##            28) MHKOOP< 8.5 174  21 0 (0.87931034 0.12068966)
##              56) MFWEKIND< 5.5 143  13 0 (0.90909091 0.09090909)
##                112) MGODGE< 4.5 111   5 0 (0.95495495 0.04504505) *
##                113) MGODGE>=4.5 32   8 0 (0.75000000 0.25000000)
##                  226) MINKM30< 1 7   0 0 (1.00000000 0.00000000) *
##                  227) MINKM30>=1 25   8 0 (0.68000000 0.32000000)
##                    454) PWAPART>=0.5 18   3 0 (0.83333333 0.16666667) *
##                    455) PWAPART< 0.5 7   2 1 (0.28571429 0.71428571) *
##              57) MFWEKIND>=5.5 31   8 0 (0.74193548 0.25806452) *
##            29) MHKOOP>=8.5 115  32 0 (0.72173913 0.27826087)
##              58) MAUT1>=6.5 58  10 0 (0.82758621 0.17241379) *
##              59) MAUT1< 6.5 57  22 0 (0.61403509 0.38596491)
##                118) MINK4575< 3.5 20   4 0 (0.80000000 0.20000000) *
##                119) MINK4575>=3.5 37  18 0 (0.51351351 0.48648649)
##                  238) MSKB1< 1.5 16   4 0 (0.75000000 0.25000000) *
##                  239) MSKB1>=1.5 21   7 1 (0.33333333 0.66666667) *
##          15) MBERARBO>=3.5 14   5 1 (0.35714286 0.64285714) *
```

```
t0 %>% plot()
t0 %>% text()
```



The tree as a whole is too complex to interpret fully. In the first node, if `PPERSAUT<5.5`, the left branch will be evaluated further, otherwise the right branch. A 1 at a leaf node indicates predicting a purchase, otherwise no purchase.

Predictions

```
predicted <- predict(t0, newdata=test, type = "vector")
table(test$Purchase, predicted)
```

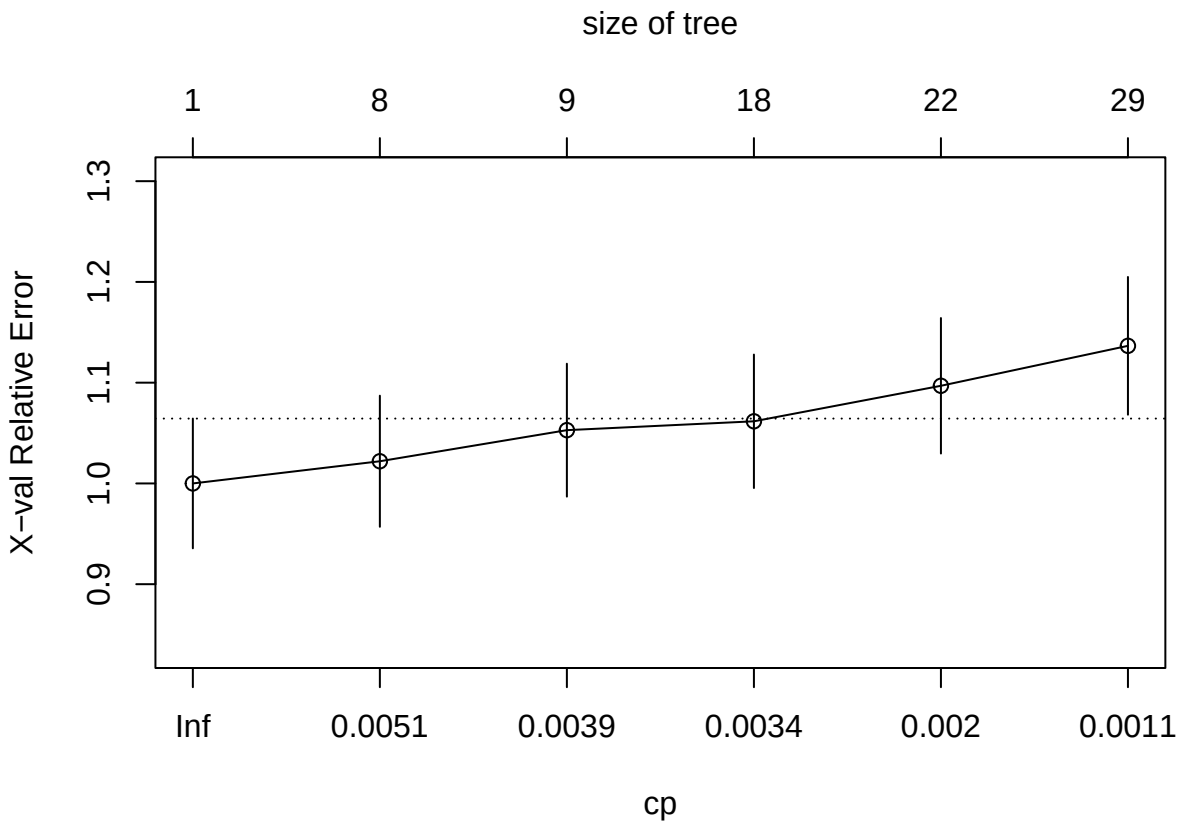
```
##    predicted
##        1    2
##   0 1779   42
##   1  115    6
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.5133"
```

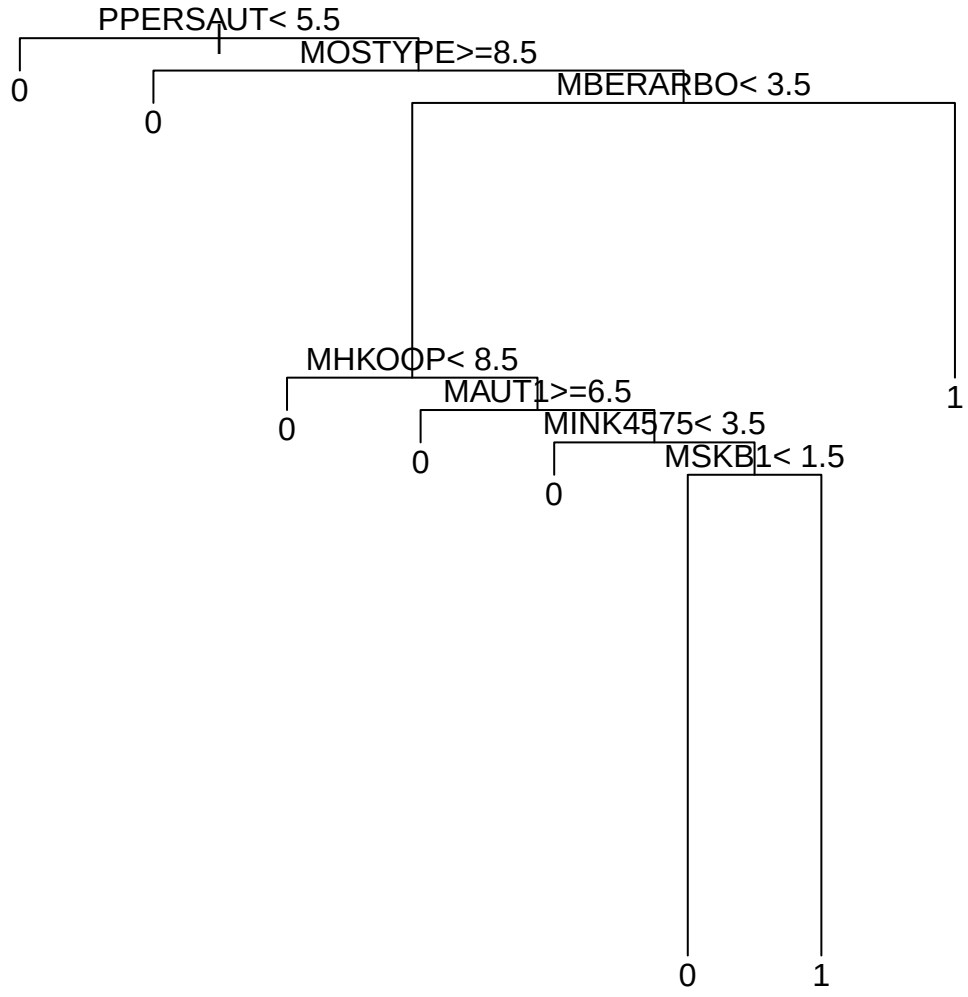
Pruning the tree

```
plotcp(t0)
```



The plot suggests the optimal tree complexity to be ~0.006, judging by the 1se-rule.

```
t1 <- prune(t0, cp=0.0051)
t1 %>% plot()
t1 %>% text()
```




```
predicted <- predict(t1, newdata=test, type = "vector")
table(test$Purchase, predicted)
```

```
##    predicted
##        1    2
##   0 1808   13
```

```
##   1  121    0
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.4964"
```

The tree was pruned nicely, now is more leggible.

Weighting the observations

I apply the ratio of the two classes and use them as weights.

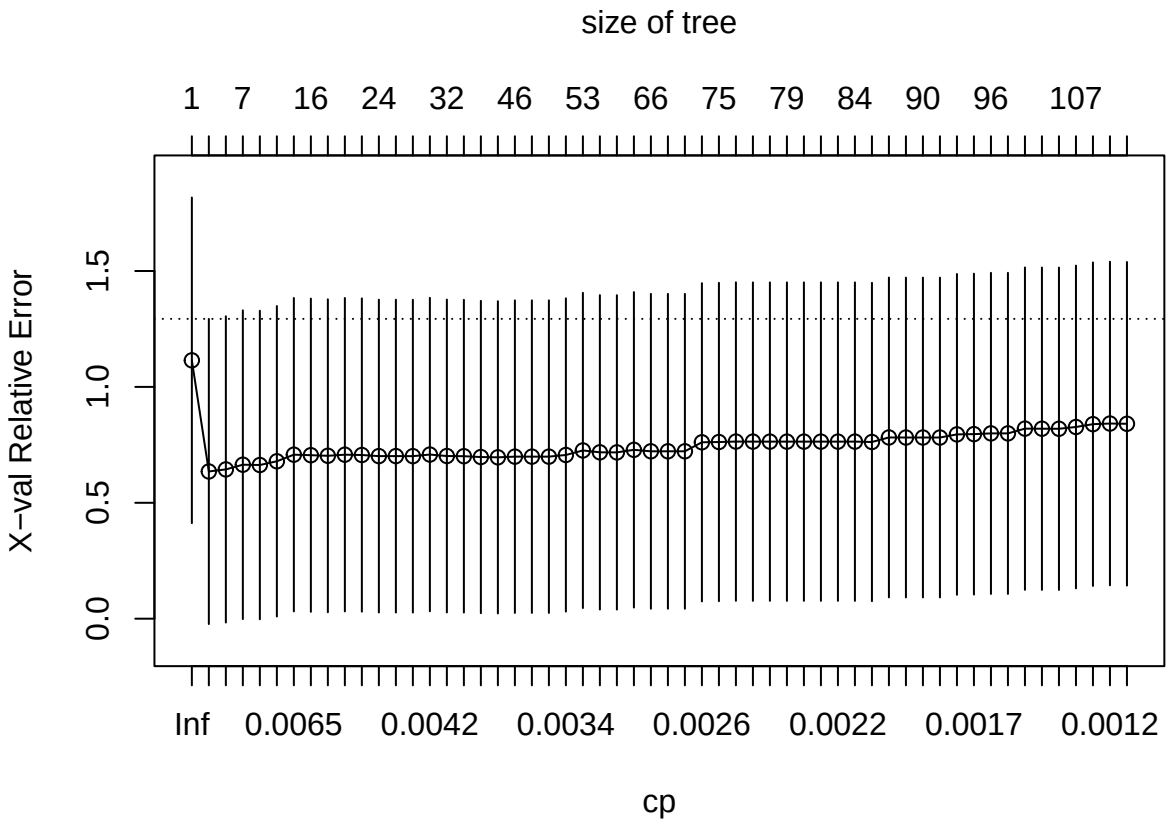
```
weights <- ifelse(train$Purchase == 1, 1 / sum(train$Purchase == 1), 1 / sum(train$Purchase == 0))

t2 <- rpart(Purchase~., data=train, cp=0.001, xval=20, weights = weights, method="class")

predicted <- predict(t2, newdata=test, type = "vector")
table(test$Purchase, predicted)
```

```
##    predicted
##        1    2
##   0 1478  343
##   1   79   42
```

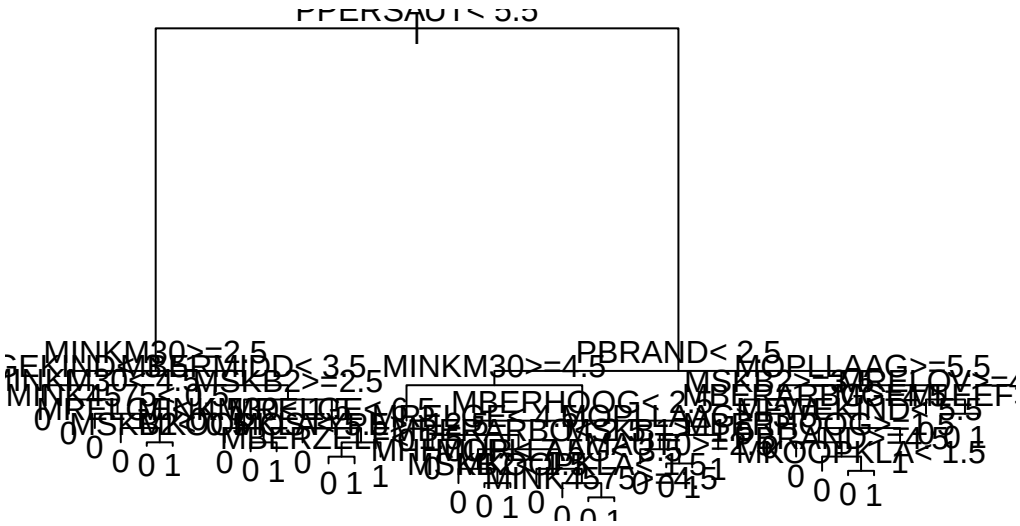
```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.5794"
```

```
plotcp(t2)
```




```
t3 <- prune(t2, cp=0.0036)
t3 %>% plot()
t3 %>% text()
```




```
predicted <- predict(t3, newdata=test, type = "vector")
table(test$Purchase, predicted)
```

```
##    predicted
##        1    2
##   0 1384  437
##   1   58   63
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.6403"
```

Applying the weights disappointingly did not improve the performance of the tree, even after pruning to the 1se-rule.

Task 2: Random forests

a

```
train <- train %>% mutate(Purchase=Purchase %>% as.factor())
rf <- randomForest(Purchase~., data=train, importance=T)
```

```
test <- test %>% mutate(Purchase=Purchase %>% as.factor)
predicted <- predict(rf, newdata=test)
ct <- table(test$Purchase, predicted)
ct
```

```
##    predicted
##        0    1
##   0 1801   20
##   1  119    2
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

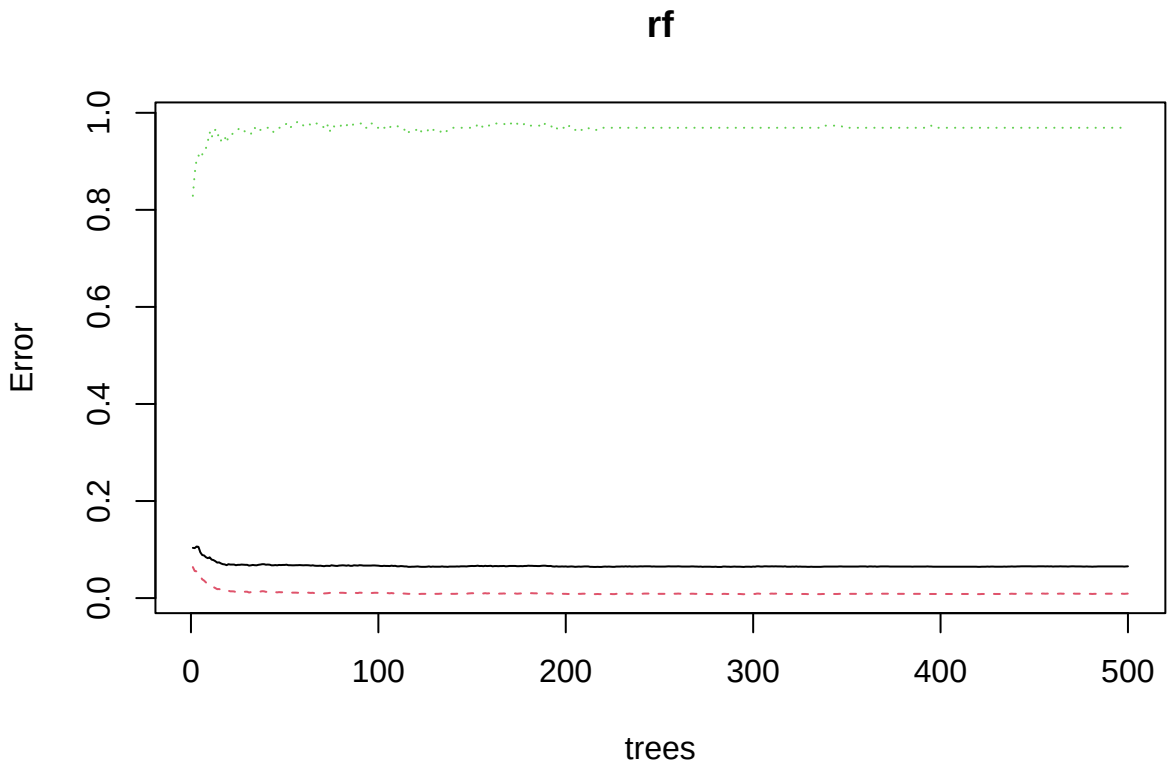
```
## [1] "Balanced accuracy: 0.5028"
```

b

```
rf
```

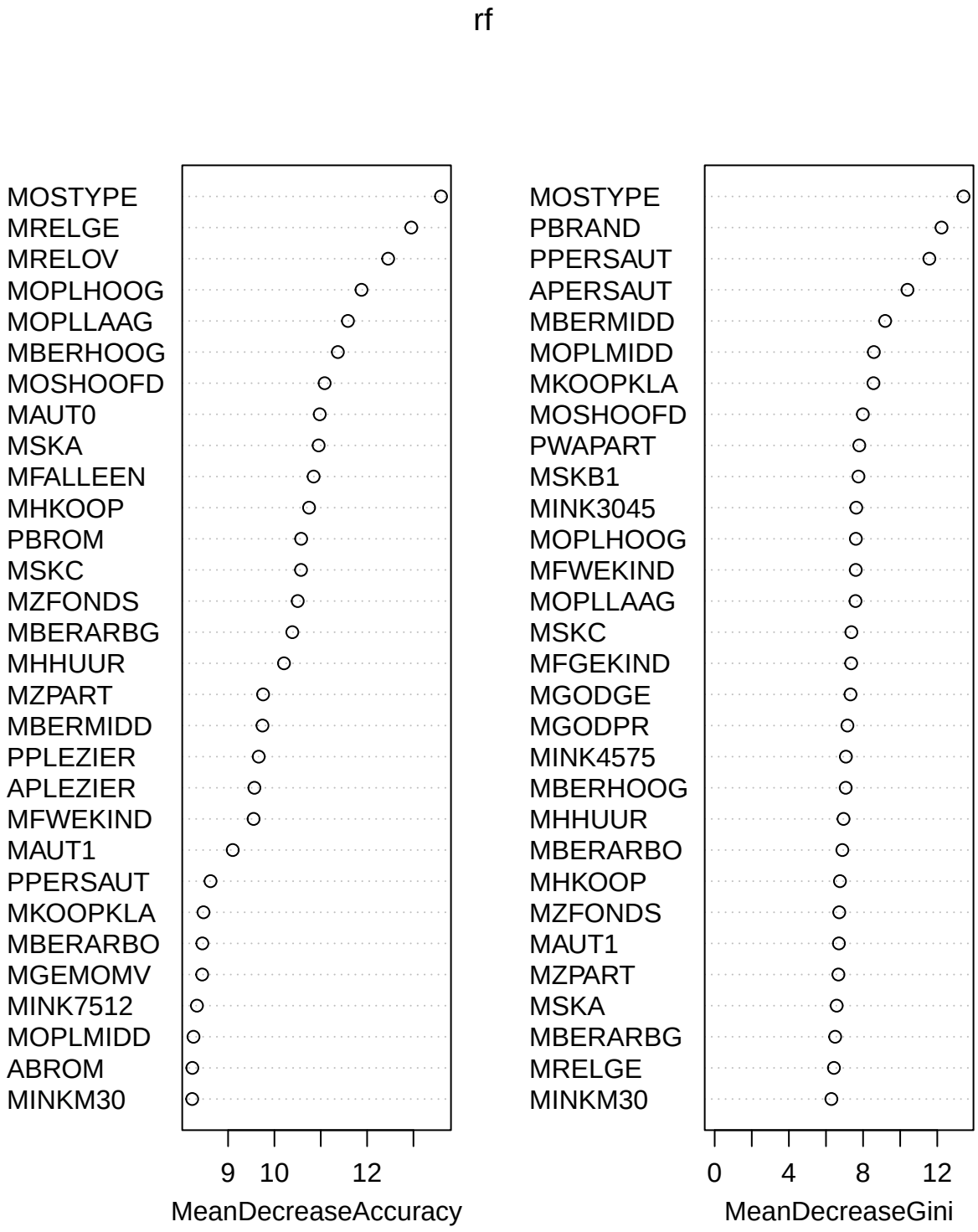
```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.55%
## Confusion matrix:
##      0  1 class.error
## 0 3619 34 0.009307419
## 1  220  7 0.969162996
```

```
plot(rf)
```



The plot shows 3 lines. Across the 500 trees the algorithm created, after about ~75 trees, the relative out-of-bag error seems to be kind of stabilized. The red and green line are for the two classes, Purchase being either 1 or 0, the black line is the average of that. The green line shows, that for one of the classes, presumably the successes, nearly 100% of the classifications are false negatives, which is quite bad.

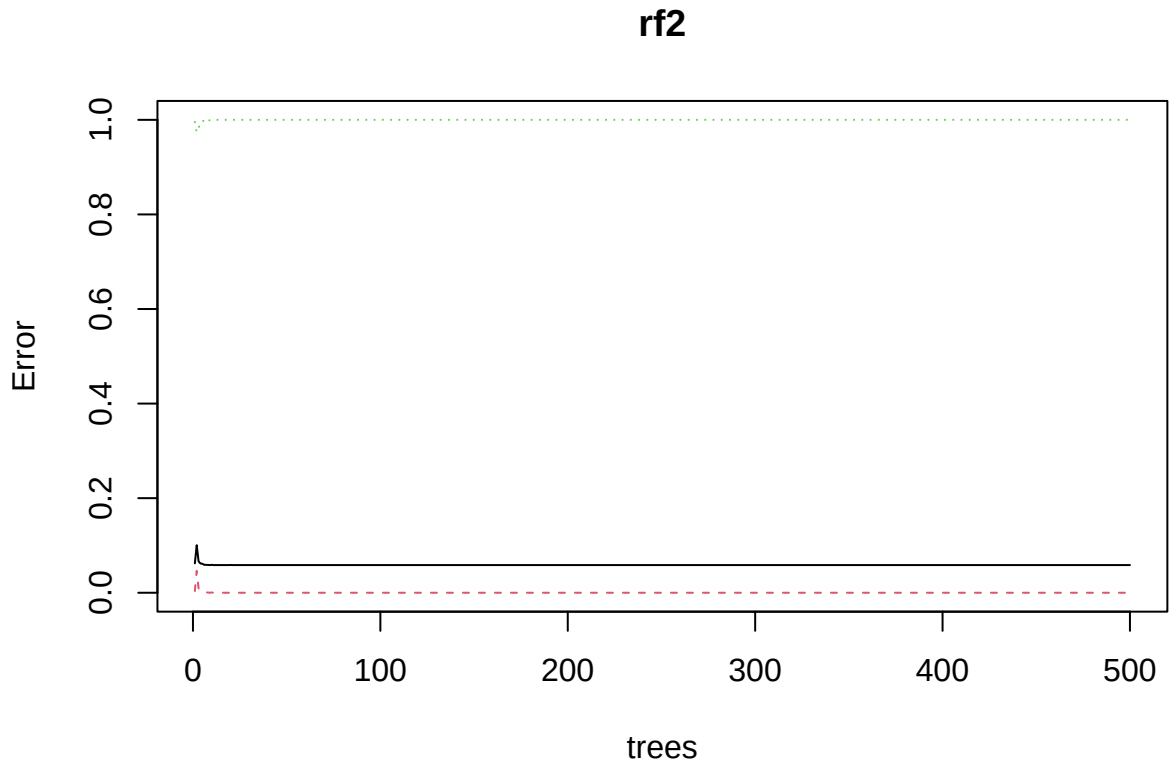
```
varImpPlot(rf)
```



c

samplsize is by default 63% of the number of observations, which is roughly 2400. It is the number of samples drawn from the training dataset to build a single tree, with replacement that is.

```
rf2 <- randomForest(Purchase~., data=train, importance=T, sampsize=50)
plot(rf2)
```




```
rf2
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, importance = T,      sampsize = 50)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 5.85%
## Confusion matrix:
##      0 1 class.error
## 0 3653 0           0
## 1  227 0           1
```

```
predicted <- predict(rf2, newdata=test)
ct <- table(test$Purchase, predicted)
ct
```

```
##    predicted
##        0    1
```

```
##   0 1821    0
##   1  121    0
```

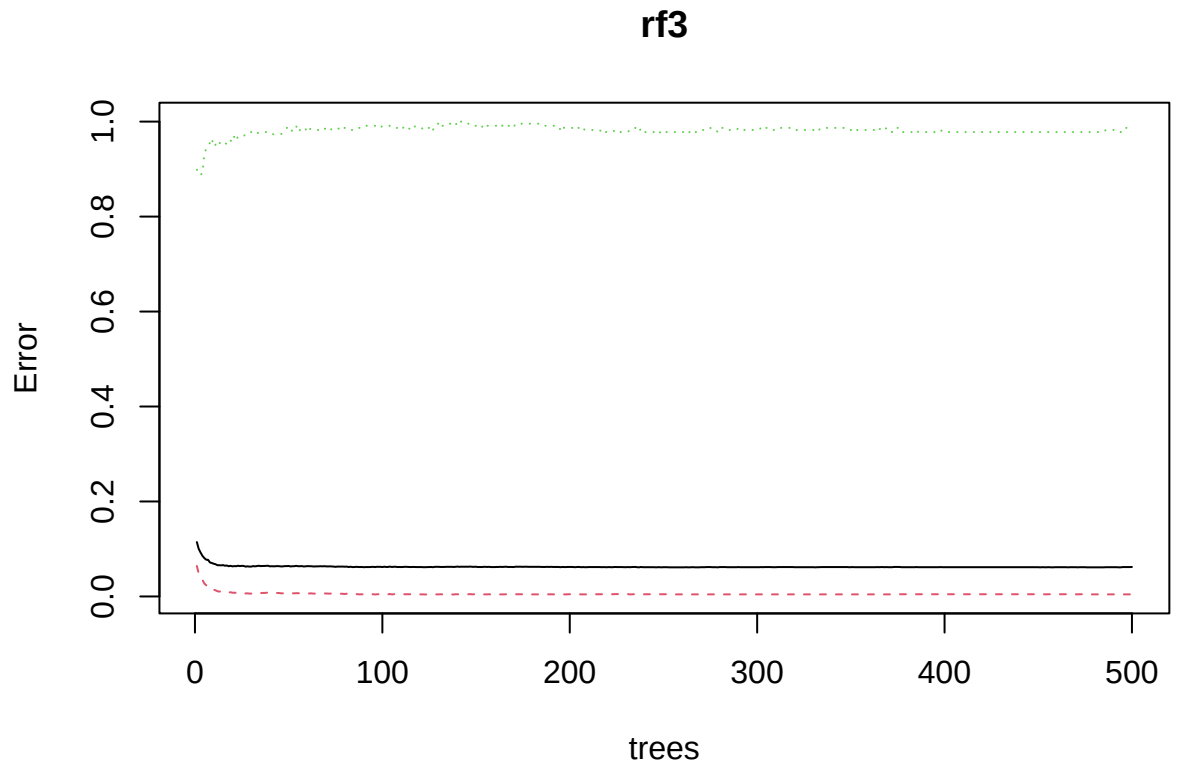
```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.5"
```

When picking an in comparison low sampsize, in this case 50, we get a majority class classifier.

You can also pass a vector of strats to the parameter to balance the classes. So instead of passig 2400 random samples, lets instead pass 140 successes and 2200 failures, which reflects the ratio the 2 classes have.

```
# 3653 vs 227 from total of 3880

rf3 <- randomForest(Purchase~., data=train, importance=T, sampsize=c(2200, 140))
plot(rf3)
```




```
rf3
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, importance = T,      sampsize = c(2200, 140))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
```

```
##
##         OOB estimate of  error rate: 6.21%
## Confusion matrix:
##      0  1 class.error
## 0 3636 17 0.004653709
## 1  224  3 0.986784141
```

```
predicted <- predict(rf3, newdata=test)
ct <- table(test$Purchase, predicted)
ct
```

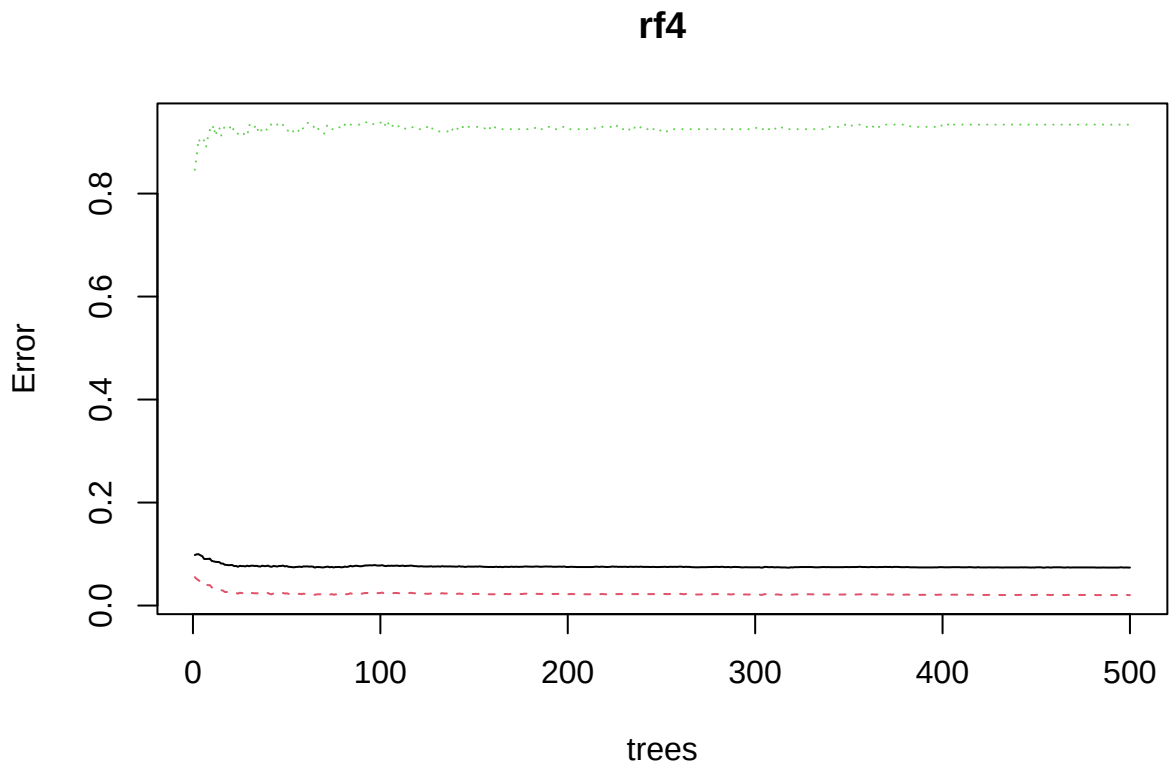
```
##    predicted
##        0    1
##   0 1811   10
##   1  119    2
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.5055"
```

The real way to adjust the sample picking to class weights is the classwt parameter tough.

```
rf4 <- randomForest(Purchase~., data=train, importance=T, classwt = 1 / table(train$Purchase))
plot(rf4)
```



```
rf4
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, importance = T,      classwt = 1/table(train$Pur
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 7.37%
## Confusion matrix:
##      0  1 class.error
## 0 3579 74  0.02025732
## 1  212 15  0.93392070
```

```
predicted <- predict(rf4, newdata=test)
ct <- table(test$Purchase, predicted)
ct
```

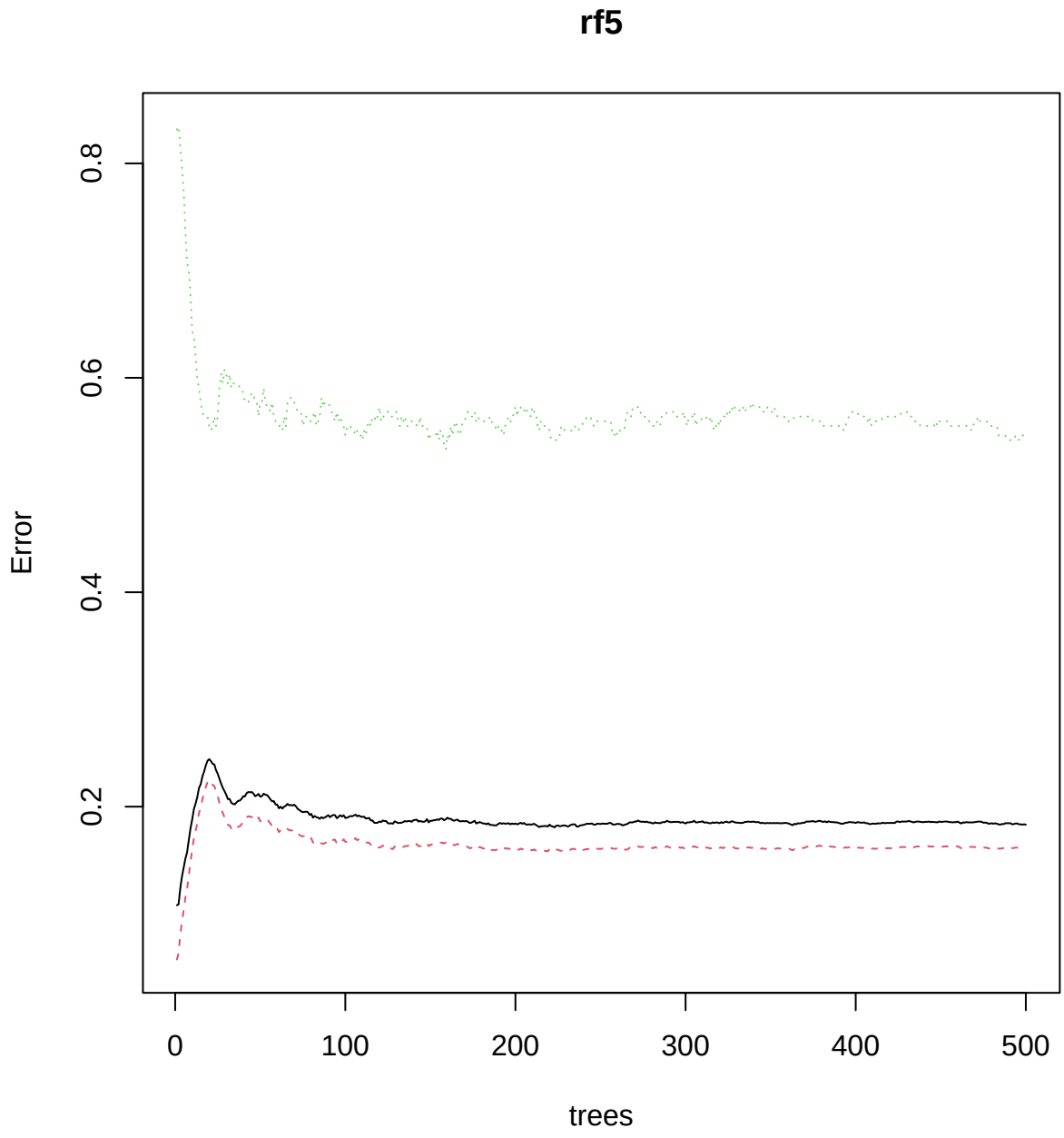
```
##    predicted
##        0    1
##   0 1788   33
##   1  117    4
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.5075"
```

If this does not help enough, there is the third parameter, cutoff, which is a ratio vector that ajusts the model's sensitivity to each class. Here we lower the threshold and thus increase the sensitivity for/to the success class:

```
rf5 <- randomForest(Purchase~., data=train, importance=T, cutoff=c(9/10, 1/10))

plot(rf5)
```




```
rf5
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, importance = T,      cutoff = c(9/10, 1/10))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 18.32%
## Confusion matrix:
```

```
##      0   1 class.error
## 0 3066 587   0.1606898
## 1  124 103   0.5462555
```

```
predicted <- predict(rf5, newdata=test)
ct <- table(test$Purchase, predicted)
ct
```

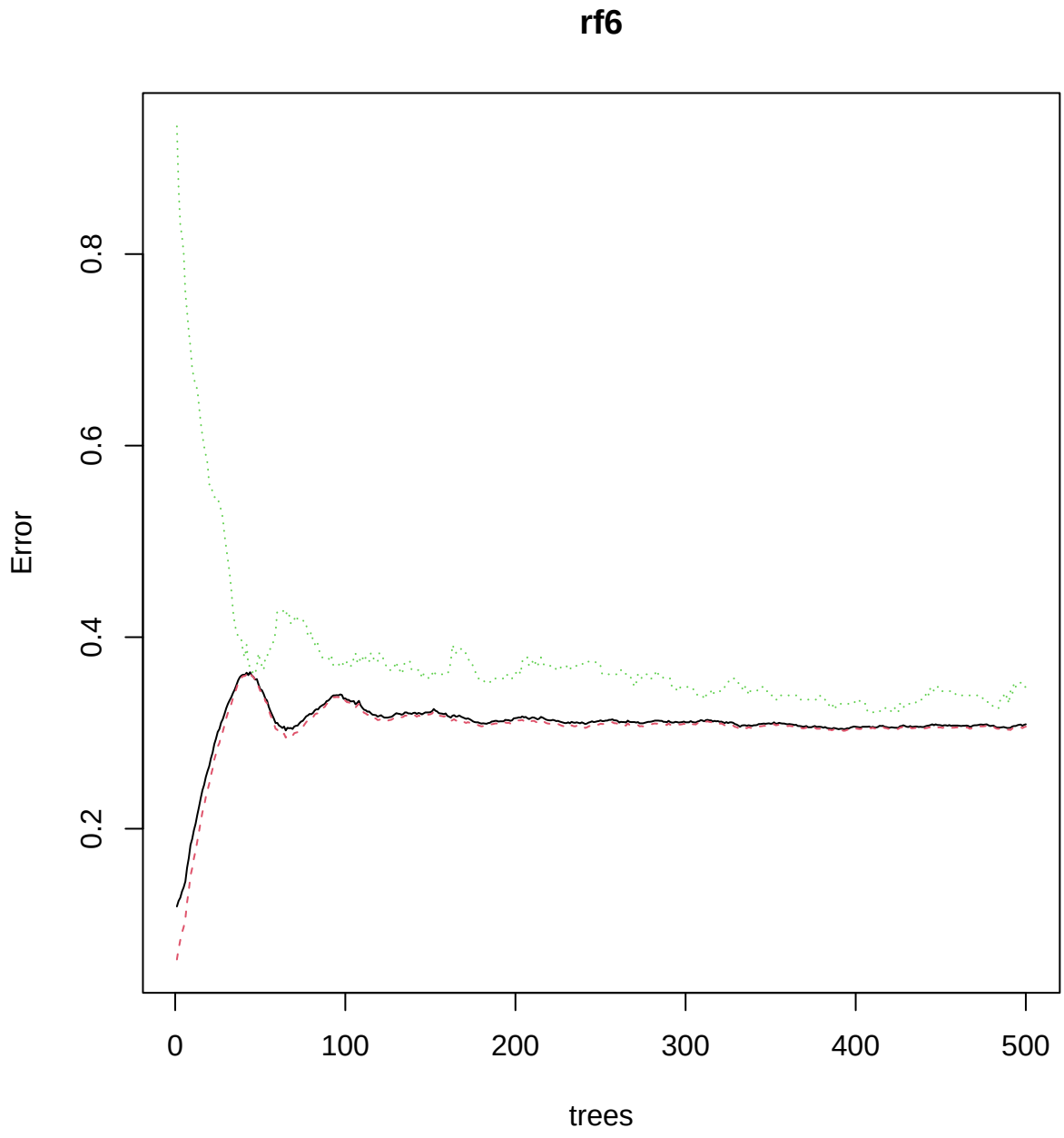
```
##    predicted
##        0    1
##   0 1514  307
##   1   80   41
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.5851"
```

This is the first adjustment that actually led to a noticeable increase in balanced accuracy. The green line in the plot finally gets lowered down, which reflects the model picking the success class more often and lowering the number of false negatives. We end up with a strong increase of false positives too tough.

```
rf6 <- randomForest(Purchase~., data=train, importance=T, cutoff=c(19/20, 1/20))

plot(rf6)
```




```
rf6
```

```
##
## Call:
##  randomForest(formula = Purchase ~ ., data = train, importance = T,      cutoff = c(19/20, 1/20))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 30.9%
## Confusion matrix:
```

```
##      0    1 class.error
## 0 2533 1120   0.3065973
## 1   79  148   0.3480176
```

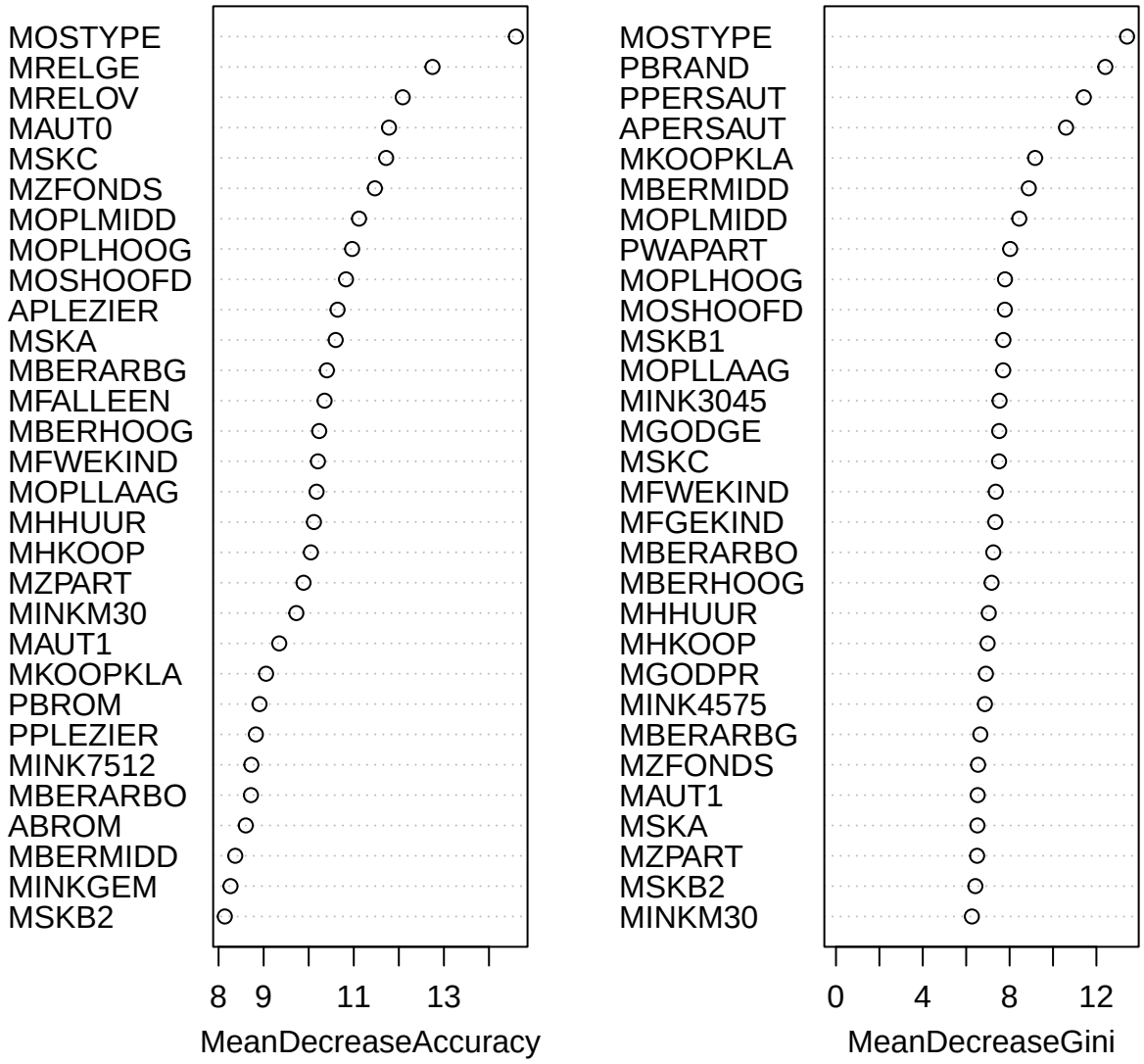
```
predicted <- predict(rf6, newdata=test)
ct <- table(test$Purchase, predicted)
ct
```

```
##    predicted
##        0    1
##   0 1249  572
##   1   53   68
```

```
print("Balanced accuracy:" %>% paste(BACC(test$Purchase, predicted) %>% round(4)))
```

```
## [1] "Balanced accuracy: 0.6239"
```

```
varImpPlot(rf6)
```



Here I took it even further and lowered the threshold even more than the disparity between the class ratios, which led to even more false positives, but an overall higher balanced accuracy.

The varimportance plot shows the variables which, when they have been picked, led to a higher predicting accuracy and a stronger decrease of the Gini impurity. The most important variable appears to be MOSTYPE. The left plot actually picked different top predicting variables compared to the random forest model before adjusting the cutoff thresholds.