ML-2017-FALL-NOUL-PROJECT-RF

Noul Singla

November 14, 2017

```
load("Project_Trees.RData")
load("Dataset.RData")

#install.packages("caret")
library("caret")
```

```
## Warning: package 'caret' was built under R version 3.4.2
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#install.packages("e1071")
library("e1071")
```

```
## Warning: package 'e1071' was built under R version 3.4.2
```

```
#pop <- Train.Student.Data[,c(3,9,22)]
#pop  <-model.matrix(~.,pop)[,-1]
#pop.rf <- train(as.factor(internetyes)~., data=pop,type="rf")
#getTree(pop.rf$finalModel,2)
#prd.tst33 <- predict(pop.rf,newdata=pop)
#table(prd.tst33,pop[,6])

#install.packages("randomForest")
library("randomForest")
```

```
## Warning: package 'randomForest' was built under R version 3.4.2
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

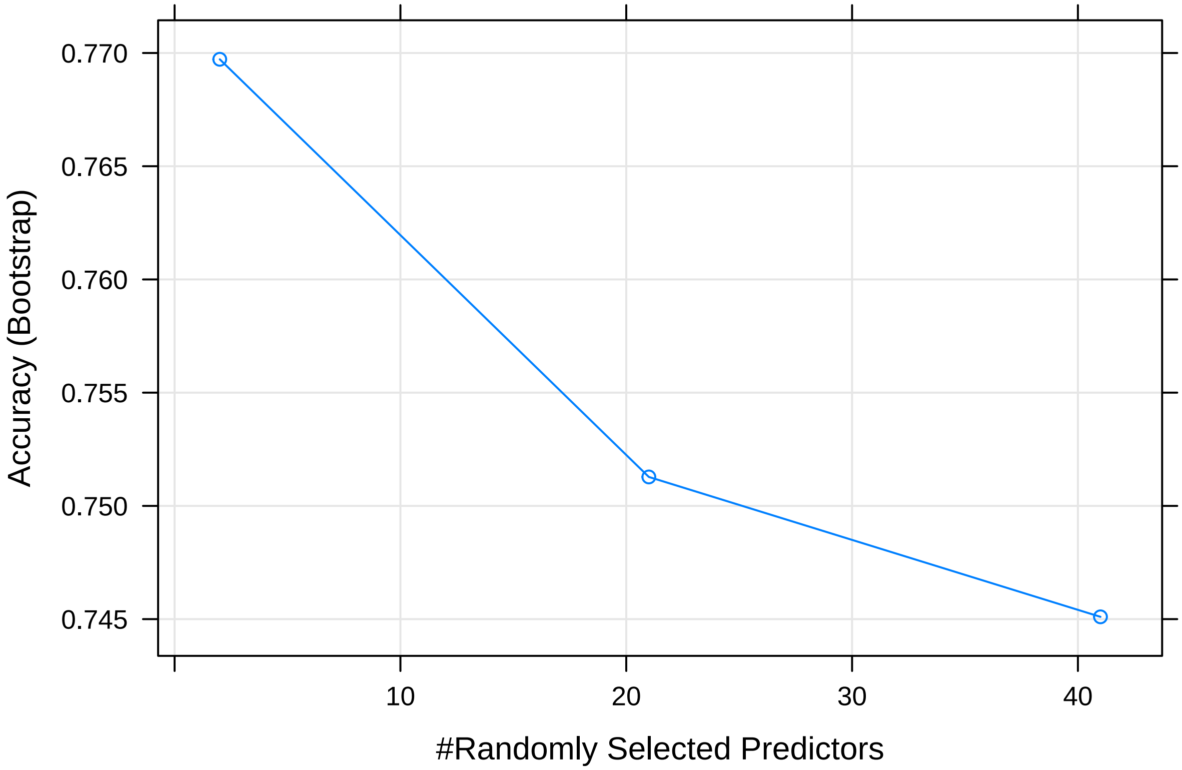
```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
internet.rf <- train(internet~., data=Train.Student.Data,type="rf")
internet.rf
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 519, 519, 519, 519, 519, 519, ...
## Resampling results across tuning parameters:
```

```
##
##   mtry  Accuracy   Kappa
##    2    0.7697212  0.01858643
##   21    0.7512779  0.10047253
##   41    0.7451011  0.10812143
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(internet.rf)
```



```
str(internet.rf)
```

```
## List of 24
##  $ method      : chr "rf"
##  $ modelInfo   :List of 15
##   ..$ label      : chr "Random Forest"
##   ..$ library    : chr "randomForest"
##   ..$ loop       : NULL
##   ..$ type       : chr [1:2] "Classification" "Regression"
##   ..$ parameters:'data.frame':   1 obs. of  3 variables:
##   .. ..$ parameter: Factor w/ 1 level "mtry": 1
##   .. ..$ class    : Factor w/ 1 level "numeric": 1
##   .. ..$ label    : Factor w/ 1 level "#Randomly Selected Predictors": 1
##   ..$ grid      :function (x, y, len = NULL, search = "grid")
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 8 26 17 19 26 19 8 17
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ fit       :function (x, y, wts, param, lev, last, classProbs, ...)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 18 25 19 76 25 76 18 19
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ predict   :function (modelFit, newdata, submodels = NULL)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 20 29 21 91 29 91 20 21
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ prob      :function (modelFit, newdata, submodels = NULL)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 22 26 23 121 26 121 22 23
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ predictors:function (x, ...)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 24 32 32 19 32 19 24 32
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ varImp    :function (object, ...)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 33 28 52 19 28 19 33 52
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ levels    :function (x)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 53 28 53 48 28 48 53 53
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ tags      : chr [1:4] "Random Forest" "Ensemble Model" "Bagging" "Implicit Feature Selection"
##   ..$ sort      :function (x)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 55 26 55 53 26 53 55 55
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##   ..$ oob       :function (x)
##   .. ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 56 25 63 19 25 19 56 63
##   .. .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x0000000022d02e88>
##  $ modelType   : chr "Classification"
##  $ results     :'data.frame':    3 obs. of  5 variables:
##   ..$ mtry      : num [1:3] 2 21 41
##   ..$ Accuracy  : num [1:3] 0.77 0.751 0.745
##   ..$ Kappa     : num [1:3] 0.0186 0.1005 0.1081
##   ..$ AccuracySD: num [1:3] 0.0247 0.0282 0.0265
##   ..$ KappaSD   : num [1:3] 0.0305 0.0635 0.0583
##  $ pred        : NULL
##  $ bestTune    :'data.frame':    1 obs. of  1 variable:
##   ..$ mtry: num 2
##  $ call        : language train.formula(form = internet ~ ., data = Train.Student.Data, type = "rf")
##  $ dots        :List of 1
##   ..$ type: chr "rf"
##  $ metric      : chr "Accuracy"
##  $ control     :List of 27
##   ..$ method           : chr "boot"
##   ..$ number           : num 25
##   ..$ repeats          : logi NA
##   ..$ search           : chr "grid"
##   ..$ p                : num 0.75
##   ..$ initialWindow    : NULL
##   ..$ horizon          : num 1
##   ..$ fixedWindow      : logi TRUE
##   ..$ skip             : num 0
##   ..$ verboseIter      : logi FALSE
##   ..$ returnData       : logi TRUE
##   ..$ returnResamp     : chr "final"
##   ..$ savePredictions  : chr "none"
##   ..$ classProbs       : logi FALSE
##   ..$ summaryFunction  :function (data, lev = NULL, model = NULL)
##   ..$ selectionFunction: chr "best"
##   ..$ preProcOptions   :List of 6
##   .. ..$ thresh   : num 0.95
##   .. ..$ ICAcomp  : num 3
##   .. ..$ k        : num 5
##   .. ..$ freqCut  : num 19
##   .. ..$ uniqueCut: num 10
##   .. ..$ cutoff   : num 0.9
##   ..$ sampling         : NULL
##   ..$ index            :List of 25
##   .. ..$ Resample01: int [1:519] 3 3 5 6 7 7 9 9 9 9 ...
##   .. ..$ Resample02: int [1:519] 1 2 3 3 9 10 11 15 17 18 ...
##   .. ..$ Resample03: int [1:519] 5 8 9 9 10 10 11 11 12 12 ...
##   .. ..$ Resample04: int [1:519] 1 2 3 3 5 7 8 10 10 10 ...
##   .. ..$ Resample05: int [1:519] 3 3 5 8 8 8 8 11 13 13 ...
##   .. ..$ Resample06: int [1:519] 1 2 2 5 6 7 8 8 16 22 ...
##   .. ..$ Resample07: int [1:519] 1 3 6 7 8 9 13 14 17 18 ...
##   .. ..$ Resample08: int [1:519] 1 4 4 5 6 7 8 9 10 10 ...
##   .. ..$ Resample09: int [1:519] 1 4 5 6 6 8 8 9 10 12 ...
##   .. ..$ Resample10: int [1:519] 2 2 4 4 4 5 5 5 5 6 ...
##   .. ..$ Resample11: int [1:519] 3 3 4 5 7 9 9 10 10 11 ...
##   .. ..$ Resample12: int [1:519] 2 2 3 5 6 6 6 11 14 18 ...
##   .. ..$ Resample13: int [1:519] 1 1 1 1 2 3 4 5 8 9 ...
##   .. ..$ Resample14: int [1:519] 1 2 3 3 4 5 6 8 9 9 ...
##   .. ..$ Resample15: int [1:519] 3 3 7 7 8 8 9 9 9 10 ...
##   .. ..$ Resample16: int [1:519] 2 4 5 6 6 7 9 10 11 12 ...
##   .. ..$ Resample17: int [1:519] 4 5 5 7 8 8 11 12 14 15 ...
##   .. ..$ Resample18: int [1:519] 2 4 5 6 6 6 7 7 12 14 ...
##   .. ..$ Resample19: int [1:519] 1 1 5 5 7 9 10 10 11 12 ...
##   .. ..$ Resample20: int [1:519] 1 2 2 2 3 5 5 7 10 10 ...
##   .. ..$ Resample21: int [1:519] 3 3 5 6 8 8 10 11 12 12 ...
##   .. ..$ Resample22: int [1:519] 2 2 6 6 7 7 7 8 10 11 ...
##   .. ..$ Resample23: int [1:519] 1 1 2 3 4 4 5 6 6 6 ...
##   .. ..$ Resample24: int [1:519] 2 2 2 5 6 7 7 7 8 8 ...
##   .. ..$ Resample25: int [1:519] 3 3 4 4 4 6 6 6 7 8 ...
##   ..$ indexOut         :List of 25
##   .. ..$ Resample01: int [1:183] 1 2 4 8 12 13 14 16 23 24 ...
##   .. ..$ Resample02: int [1:187] 4 5 6 7 8 12 13 14 16 20 ...
##   .. ..$ Resample03: int [1:201] 1 2 3 4 6 7 14 15 16 19 ...
##   .. ..$ Resample04: int [1:181] 4 6 9 11 22 26 30 31 33 40 ...
##   .. ..$ Resample05: int [1:196] 1 2 4 6 7 9 10 12 15 18 ...
##   .. ..$ Resample06: int [1:187] 3 4 9 10 11 12 13 14 15 17 ...
##   .. ..$ Resample07: int [1:205] 2 4 5 10 11 12 15 16 20 21 ...
##   .. ..$ Resample08: int [1:178] 2 3 16 18 19 25 28 30 32 33 ...
##   .. ..$ Resample09: int [1:204] 2 3 7 11 13 14 16 17 18 19 ...
##   .. ..$ Resample10: int [1:184] 1 3 7 9 15 17 19 28 29 30 ...
##   .. ..$ Resample11: int [1:181] 1 2 6 8 14 15 17 22 23 25 ...
##   .. ..$ Resample12: int [1:186] 1 4 7 8 9 10 12 13 15 16 ...
##   .. ..$ Resample13: int [1:199] 6 7 11 13 14 22 25 28 32 34 ...
##   .. ..$ Resample14: int [1:186] 7 12 13 14 15 19 33 34 40 42 ...
##   .. ..$ Resample15: int [1:202] 1 2 4 5 6 12 15 18 19 20 ...
##   .. ..$ Resample16: int [1:201] 1 3 8 17 19 21 24 26 28 33 ...
##   .. ..$ Resample17: int [1:188] 1 2 3 6 9 10 13 16 17 19 ...
##   .. ..$ Resample18: int [1:197] 1 3 8 9 10 11 13 16 19 21 ...
##   .. ..$ Resample19: int [1:189] 2 3 4 6 8 15 16 21 23 24 ...
##   .. ..$ Resample20: int [1:190] 4 6 8 9 11 14 18 24 28 30 ...
##   .. ..$ Resample21: int [1:193] 1 2 4 7 9 17 18 22 33 38 ...
##   .. ..$ Resample22: int [1:189] 1 3 4 5 9 13 20 22 26 28 ...
##   .. ..$ Resample23: int [1:191] 7 8 9 11 15 16 18 19 21 22 ...
##   .. ..$ Resample24: int [1:197] 1 3 4 9 10 17 21 23 25 26 ...
##   .. ..$ Resample25: int [1:190] 1 2 5 11 15 18 19 22 23 31 ...
##   ..$ indexFinal       : NULL
##   ..$ timingSamps      : num 0
##   ..$ predictionBounds : logi [1:2] FALSE FALSE
##   ..$ seeds            :List of 26
##   .. ..$ : int [1:3] 402464 960593 932300
##   .. ..$ : int [1:3] 270080 971895 546573
##   .. ..$ : int [1:3] 238490 849587 145632
##   .. ..$ : int [1:3] 78130 589421 887544
##   .. ..$ : int [1:3] 792180 712956 498856
##   .. ..$ : int [1:3] 941305 430623 572307
##   .. ..$ : int [1:3] 384731 917016 702158
##   .. ..$ : int [1:3] 849485 139287 76792
##   .. ..$ : int [1:3] 406359 52228 530681
##   .. ..$ : int [1:3] 184047 12378 249226
##   .. ..$ : int [1:3] 89511 507503 635825
##   .. ..$ : int [1:3] 65737 120462 583921
##   .. ..$ : int [1:3] 736212 471540 368638
##   .. ..$ : int [1:3] 957302 723515 811408
##   .. ..$ : int [1:3] 562115 919288 972251
##   .. ..$ : int [1:3] 38975 813216 354082
##   .. ..$ : int [1:3] 918266 572751 257214
##   .. ..$ : int [1:3] 938783 884377 363526
##   .. ..$ : int [1:3] 744856 385535 427713
##   .. ..$ : int [1:3] 355918 131057 708356
##   .. ..$ : int [1:3] 606305 685591 530636
##   .. ..$ : int [1:3] 211603 902483 230123
##   .. ..$ : int [1:3] 837500 82301 90142
##   .. ..$ : int [1:3] 167312 141837 941365
##   .. ..$ : int [1:3] 429495 776896 89832
##   .. ..$ : int 112376
##   ..$ adaptive         :List of 4
##   .. ..$ min     : num 5
##   .. ..$ alpha   : num 0.05
##   .. ..$ method  : chr "gls"
##   .. ..$ complete: logi TRUE
##   ..$ trim             : logi FALSE
##   ..$ allowParallel    : logi TRUE
##  $ finalModel  :List of 23
##   ..$ call           : language randomForest(x = x, y = y, mtry = param$mtry, type = "rf")
##   ..$ type           : chr "classification"
##   ..$ predicted      : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##   .. ..- attr(*, "names")= chr [1:519] "X98" "X121" "X473" "X236" ...
##   ..$ err.rate       : num [1:500, 1:3] 0.323 0.319 0.349 0.345 0.336 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:3] "OOB" "no" "yes"
##   ..$ confusion      : num [1:2, 1:3] 3 1 116 399 0.975 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "no" "yes"
##   .. .. ..$ : chr [1:3] "no" "yes" "class.error"
##   ..$ votes          : matrix [1:519, 1:2] 0.161 0.237 0.314 0.124 0.146 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:519] "X98" "X121" "X473" "X236" ...
##   .. .. ..$ : chr [1:2] "no" "yes"
##   ..$ oob.times      : num [1:519] 199 186 188 161 185 168 204 176 196 178 ...
##   ..$ classes        : chr [1:2] "no" "yes"
##   ..$ importance     : num [1:41, 1] 4.8 2.46 5.29 3.11 2.21 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:41] "schoolMS" "sexM" "age" "addressU" ...
##   .. .. ..$ : chr "MeanDecreaseGini"
##   ..$ importanceSD   : NULL
##   ..$ localImportance: NULL
##   ..$ proximity      : NULL
##   ..$ ntree          : num 500
##   ..$ mtry           : num 2
##   ..$ forest         :List of 14
##   .. ..$ ndbigtree : int [1:500] 211 235 213 207 261 225 239 195 237 209 ...
##   .. ..$ nodestatus: int [1:285, 1:500] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ bestvar   : int [1:285, 1:500] 21 3 24 10 10 10 23 24 18 32 ...
##   .. ..$ treemap   : int [1:285, 1:2, 1:500] 2 4 6 8 10 12 14 16 18 20 ...
##   .. ..$ nodepred  : int [1:285, 1:500] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ xbestsplit: num [1:285, 1:500] 0.5 17.5 2.5 0.5 0.5 0.5 1.5 0.5 0.5 3.5 ...
##   .. ..$ pid       : num [1:2] 1 1
##   .. ..$ cutoff    : num [1:2] 0.5 0.5
##   .. ..$ ncat      : Named num [1:41] 1 1 1 1 1 1 1 1 1 1 ...
##   .. .. ..- attr(*, "names")= chr [1:41] "schoolMS" "sexM" "age" "addressU" ...
##   .. ..$ maxcat    : num 1
##   .. ..$ nrnodes   : int 285
##   .. ..$ ntree     : num 500
##   .. ..$ nclass    : int 2
##   .. ..$ xlevels   :List of 41
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   .. .. ..$ : num 0
##   ..$ y              : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 2 2 2 2 ...
##   .. ..- attr(*, "names")= chr [1:519] "98" "121" "473" "236" ...
##   ..$ test           : NULL
##   ..$ inbag          : NULL
##   ..$ xNames         : chr [1:41] "schoolMS" "sexM" "age" "addressU" ...
##   ..$ problemType    : chr "Classification"
##   ..$ tuneValue      :'data.frame':  1 obs. of  1 variable:
##   .. ..$ mtry: num 2
##   ..$ obsLevels      : atomic [1:2] no yes
##   .. ..- attr(*, "ordered")= logi FALSE
##   ..$ param          :List of 1
##   .. ..$ type: chr "rf"
##   ..- attr(*, "class")= chr "randomForest"
##  $ preProcess  : NULL
##  $ trainingData:'data.frame':    519 obs. of  33 variables:
##   ..$ .outcome  : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 2 2 2 2 ...
##   ..$ school    : Factor w/ 2 levels "GP","MS": 1 1 2 1 2 1 1 1 1 1 ...
##   ..$ sex       : Factor w/ 2 levels "F","M": 1 1 1 1 2 1 1 1 2 2 ...
##   ..$ age       : int [1:519] 16 15 16 17 18 17 15 19 17 16 ...
##   ..$ address   : Factor w/ 2 levels "R","U": 2 2 1 2 1 2 2 1 1 2 ...
##   ..$ famsize   : Factor w/ 2 levels "GT3","LE3": 1 1 2 1 1 1 1 1 1 1 ...
##   ..$ Pstatus   : Factor w/ 2 levels "A","T": 2 2 2 2 2 2 2 2 2 2 ...
##   ..$ Medu      : int [1:519] 2 1 2 1 4 3 1 3 3 1 ...
##   ..$ Fedu      : int [1:519] 1 2 2 1 4 2 2 2 2 0 ...
##   ..$ Mjob      : Factor w/ 5 levels "at_home","health",..: 3 1 3 1 1 3 1 4 3 3 ...
##   ..$ Fjob      : Factor w/ 5 levels "at_home","health",..: 3 4 3 3 4 3 3 4 3 3 ...
##   ..$ reason    : Factor w/ 4 levels "course","home",..: 1 1 2 4 3 1 1 4 1 4 ...
##   ..$ guardian  : Factor w/ 3 levels "father","mother",..: 2 2 1 2 2 1 2 1 2 2 ...
##   ..$ traveltime: int [1:519] 1 1 3 1 3 1 1 1 2 2 ...
##   ..$ studytime : int [1:519] 2 2 1 3 1 2 2 2 2 2 ...
##   ..$ failures  : int [1:519] 0 0 0 0 0 0 0 1 2 0 ...
##   ..$ schoolsup : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 2 1 ...
##   ..$ famsup    : Factor w/ 2 levels "no","yes": 2 1 2 2 2 1 2 2 2 2 ...
##   ..$ paid      : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 1 1 ...
##   ..$ activities: Factor w/ 2 levels "no","yes": 1 1 2 2 2 2 1 1 1 2 ...
##   ..$ nursery   : Factor w/ 2 levels "no","yes": 2 1 2 2 2 2 1 2 2 2 ...
##   ..$ higher    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 1 2 2 ...
##   ..$ romantic  : Factor w/ 2 levels "no","yes": 2 1 2 2 2 1 1 1 2 2 ...
##   ..$ famrel    : int [1:519] 4 3 4 4 2 5 4 3 4 4 ...
##   ..$ freetime  : int [1:519] 3 2 3 3 5 4 3 3 4 3 ...
##   ..$ goout     : int [1:519] 5 3 2 4 5 2 2 3 4 2 ...
##   ..$ Dalc      : int [1:519] 1 1 1 1 1 1 1 4 1 1 ...
##   ..$ Walc      : int [1:519] 1 2 1 1 1 1 1 3 4 1 ...
##   ..$ health    : int [1:519] 5 1 4 5 1 3 5 3 3 3 ...
##   ..$ absences  : int [1:519] 0 0 0 12 5 4 6 0 4 0 ...
##   ..$ G1        : int [1:519] 13 14 14 12 12 14 13 9 7 16 ...
##   ..$ G2        : int [1:519] 12 14 14 12 13 14 12 8 6 17 ...
##   ..$ G3        : int [1:519] 12 14 16 12 14 15 13 10 8 18 ...
##  $ resample    :'data.frame':    25 obs. of  3 variables:
##   ..$ Accuracy: num [1:25] 0.779 0.781 0.76 0.766 0.786 ...
##   ..$ Kappa   : num [1:25] 0.0556 0 -0.0108 0.0242 -0.0105 ...
##   ..$ Resample: chr [1:25] "Resample09" "Resample05" "Resample01" "Resample10" ...
##  $ resampledCM :'data.frame':    75 obs. of  6 variables:
##   ..$ cell1   : num [1:75] 0 2 3 0 6 5 1 11 11 0 ...
##   ..$ cell2   : num [1:75] 43 41 40 44 38 39 54 44 44 47 ...
##   ..$ cell3   : num [1:75] 1 10 11 0 7 10 1 12 13 0 ...
##   ..$ cell4   : num [1:75] 139 130 129 143 136 133 145 134 133 134 ...
##   ..$ mtry    : num [1:75] 2 21 41 2 21 41 2 21 41 2 ...
##   ..$ Resample: chr [1:75] "Resample01" "Resample01" "Resample01" "Resample02" ...
##  $ perfNames   : chr [1:2] "Accuracy" "Kappa"
##  $ maximize    : logi TRUE
##  $ yLimits     : NULL
##  $ times       :List of 3
##   ..$ everything:Class 'proc_time'  Named num [1:5] 210.71 3.08 215.47 NA NA
##   .. .. ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...
##   ..$ final     :Class 'proc_time'  Named num [1:5] 2.31 0.05 2.38 NA NA
##   .. .. ..- attr(*, "names")= chr [1:5] "user.self" "sys.self" "elapsed" "user.child" ...
##   ..$ prediction: logi [1:3] NA NA NA
##  $ levels      : atomic [1:2] no yes
##   ..- attr(*, "ordered")= logi FALSE
##  $ terms       :Classes 'terms', 'formula'  language internet ~ school + sex + age + address + famsi
##   .. ..- attr(*, "variables")= language list(internet, school, sex, age, address, famsize, Pstatus,
##   .. ..- attr(*, "factors")= int [1:33, 1:32] 0 1 0 0 0 0 0 0 0 0 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:33] "internet" "school" "sex" "age" ...
##   .. .. .. ..$ : chr [1:32] "school" "sex" "age" "address" ...
##   .. ..- attr(*, "term.labels")= chr [1:32] "school" "sex" "age" "address" ...
##   .. ..- attr(*, "order")= int [1:32] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(internet, school, sex, age, address, famsize, Pstatus, M
##   .. ..- attr(*, "dataClasses")= Named chr [1:33] "factor" "factor" "factor" "numeric" ...
##   .. .. ..- attr(*, "names")= chr [1:33] "internet" "school" "sex" "age" ...
##  $ coefnames   : chr [1:41] "schoolMS" "sexM" "age" "addressU" ...
##  $ contrasts   :List of 16
##   ..$ school    : chr "contr.treatment"
##   ..$ sex       : chr "contr.treatment"
##   ..$ address   : chr "contr.treatment"
##   ..$ famsize   : chr "contr.treatment"
##   ..$ Pstatus   : chr "contr.treatment"
##   ..$ Mjob      : chr "contr.treatment"
##   ..$ Fjob      : chr "contr.treatment"
##   ..$ reason    : chr "contr.treatment"
##   ..$ guardian  : chr "contr.treatment"
##   ..$ schoolsup : chr "contr.treatment"
##   ..$ famsup    : chr "contr.treatment"
##   ..$ paid      : chr "contr.treatment"
##   ..$ activities: chr "contr.treatment"
##   ..$ nursery   : chr "contr.treatment"
##   ..$ higher    : chr "contr.treatment"
##   ..$ romantic  : chr "contr.treatment"
##  $ xlevels     :List of 16
##   ..$ school    : chr [1:2] "GP" "MS"
##   ..$ sex       : chr [1:2] "F" "M"
##   ..$ address   : chr [1:2] "R" "U"
##   ..$ famsize   : chr [1:2] "GT3" "LE3"
##   ..$ Pstatus   : chr [1:2] "A" "T"
##   ..$ Mjob      : chr [1:5] "at_home" "health" "other" "services" ...
##   ..$ Fjob      : chr [1:5] "at_home" "health" "other" "services" ...
##   ..$ reason    : chr [1:4] "course" "home" "other" "reputation"
##   ..$ guardian  : chr [1:3] "father" "mother" "other"
##   ..$ schoolsup : chr [1:2] "no" "yes"
##   ..$ famsup    : chr [1:2] "no" "yes"
##   ..$ paid      : chr [1:2] "no" "yes"
##   ..$ activities: chr [1:2] "no" "yes"
##   ..$ nursery   : chr [1:2] "no" "yes"
##   ..$ higher    : chr [1:2] "no" "yes"
##   ..$ romantic  : chr [1:2] "no" "yes"
##  - attr(*, "class")= chr [1:2] "train" "train.formula"
```

```
#prd <- predict(internet.rf,newdata=Train.Student.Data)

table(prd,Train.Student.Data$internet)
```

```
##
## prd    no yes
##   no   85   1
##   yes  34 399
```

```
505/519
```

```
## [1] 0.973025
```

```
tcontrol = trainControl(method="repeatedcv",number=5,repeats=5)
#internet.rf1 <- train(internet~., data=Train.Student.Data,metric="Accuracy",type="rf",ntree=10, trCont
internet.rf1
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 415, 415, 415, 415, 416, 415, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7641598  0.1039512
##   21    0.7352838  0.1097832
##   41    0.7352763  0.1366174
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
prd1 <- predict(internet.rf1,newdata =  Train.Student.Data)
table(prd1,Train.Student.Data$internet)
```

```
##
## prd1   no yes
##   no   80   1
##   yes  39 399
```

```
tcontrol = trainControl(method="repeatedcv",number=5,repeats=5,search="random")
#internet.rf2 <- train(internet~., data=Train.Student.Data,metric="Accuracy",type="rf",ntree=10, trCont
internet.rf2
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 415, 415, 416, 415, 415, 415, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    9    0.7402577  0.11672234
##   16    0.7391113  0.11563659
##   28    0.7298506  0.08117639
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 9.
```

```
tcontrol = trainControl(method="repeatedcv",number=5,repeats=5,search="grid")
#internet.rf3 <- train(internet~., data=Train.Student.Data,metric="Accuracy",type="rf",ntree=10, trCont
internet.rf3
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 415, 415, 416, 415, 415, 415, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7626400  0.0820876
##   21    0.7356423  0.1115321
##   41    0.7283495  0.1104003
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
tcontrol = trainControl(method="repeatedcv",number=10,repeats=4)
#internet.rf11 <- train(internet~., data=Train.Student.Data,metric="Accuracy",type="rf",ntree=20, trCon
internet.rf11
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 467, 468, 467, 467, 467, 467, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7615573  0.03154507
##   21    0.7456542  0.11096609
##   41    0.7485388  0.11892772
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
tcontrol = trainControl(method="repeatedcv",number=20,repeats=10,search="random")
#internet.rf22 <- train(internet~., data=Train.Student.Data,metric="Accuracy",type="rf",ntree=20, trCon
internet.rf22
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (20 fold, repeated 10 times)
## Summary of sample sizes: 493, 494, 493, 493, 493, 493, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   22    0.7478615  0.1055476
##   23    0.7478462  0.1074644
##   34    0.7418692  0.1071709
##
```

```
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 22.
```

```
tcontrol = trainControl(method="repeatedcv",number=20,repeats=10,search="grid")
#internet.rf33 <- train(internet~., data=Train.Student.Data,metric="Accuracy",type="rf",ntree=20, trCon
internet.rf33
```

```
## Random Forest
##
## 519 samples
##  32 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 4 times)
## Summary of sample sizes: 467, 467, 468, 467, 467, 467, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7644796  0.06712569
##   21    0.7461633  0.10683527
##   41    0.7457108  0.13594565
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
prd.tst <- predict(internet.rf,Test.Student.Data)
table(prd.tst,Test.Student.Data$internet)
```

```
##
## prd.tst no yes
##     no   1   0
##     yes 31  98
```

```
prd.tst1 <- predict(internet.rf1,Test.Student.Data)
table(prd.tst1,Test.Student.Data$internet)
```

```
##
## prd.tst1 no yes
##      no   0   3
##      yes 32  95
```

```
prd.tst2 <- predict(internet.rf2,Test.Student.Data)
table(prd.tst2,Test.Student.Data$internet)
```

```
##
## prd.tst2 no yes
##      no   8  10
##      yes 24  88
```

```
prd.tst3 <- predict(internet.rf3,Test.Student.Data)
table(prd.tst3,Test.Student.Data$internet)
```

```
##
## prd.tst3 no yes
##      no   4   6
##      yes 28  92
```

```
prd.tst11 <- predict(internet.rf11,Test.Student.Data)
table(prd.tst11,Test.Student.Data$internet)
```

```
##
## prd.tst11 no yes
##       no   1   3
##       yes 31  95
```

```
prd.tst22 <- predict(internet.rf22,Test.Student.Data)
table(prd.tst22,Test.Student.Data$internet)
```

```
##
## prd.tst22 no yes
##       no   5  10
##       yes 27  88
```

```
prd.tst33 <- predict(internet.rf33,Test.Student.Data)
table(prd.tst33,Test.Student.Data$internet)
```

```
##
## prd.tst33 no yes
##       no   3   1
##       yes 29  97
```

```
#install.packages("adabag")
library(adabag)
```

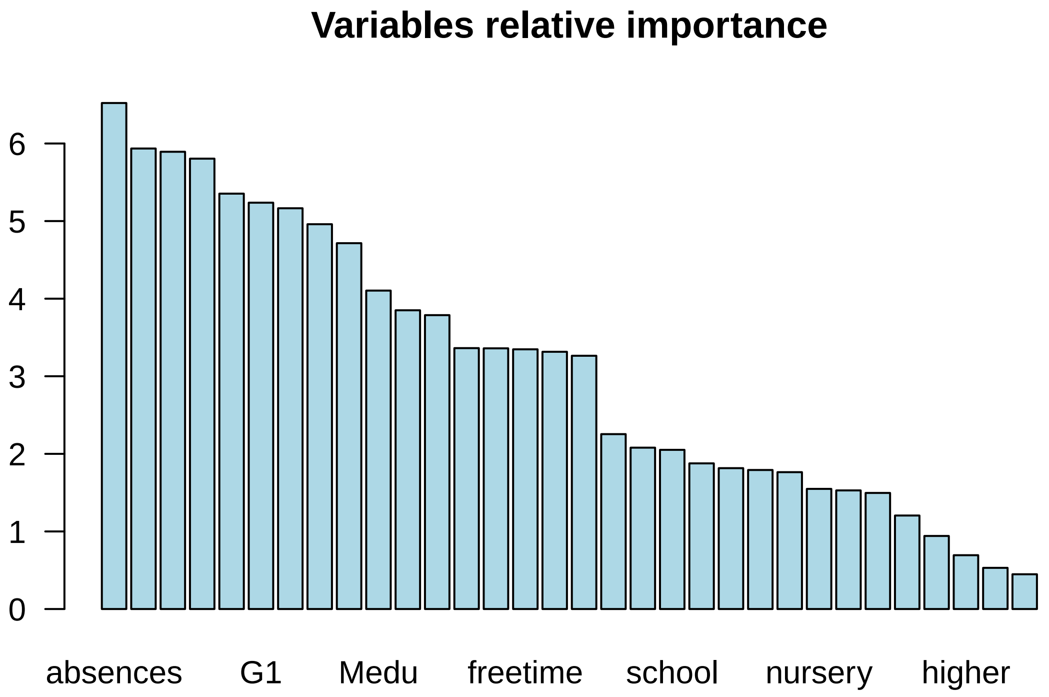
```
## Warning: package 'adabag' was built under R version 3.4.2
```

```
## Loading required package: rpart
```

```
## Loading required package: mlbench
```

```
## Warning: package 'mlbench' was built under R version 3.4.2
```

```
#internet.boost1 <- boosting(internet~.,data=Train.Student.Data,boos=TRUE,mfinal=100)
predict.train.boost1 <- predict(internet.boost1,newdata=Train.Student.Data)
importanceplot(internet.boost1)
```



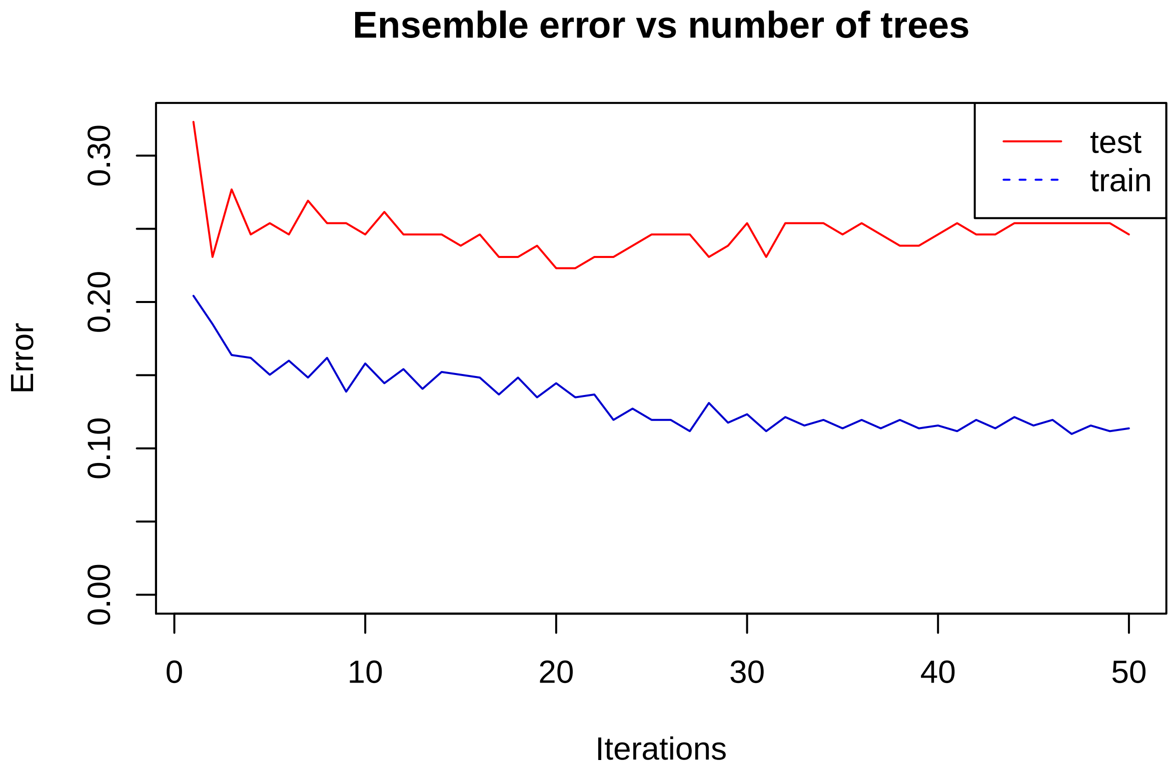
```
predict.train.boost1$confusion
```

```
##                Observed Class
## Predicted Class  no yes
##             no  119   0
##             yes   0 400
```

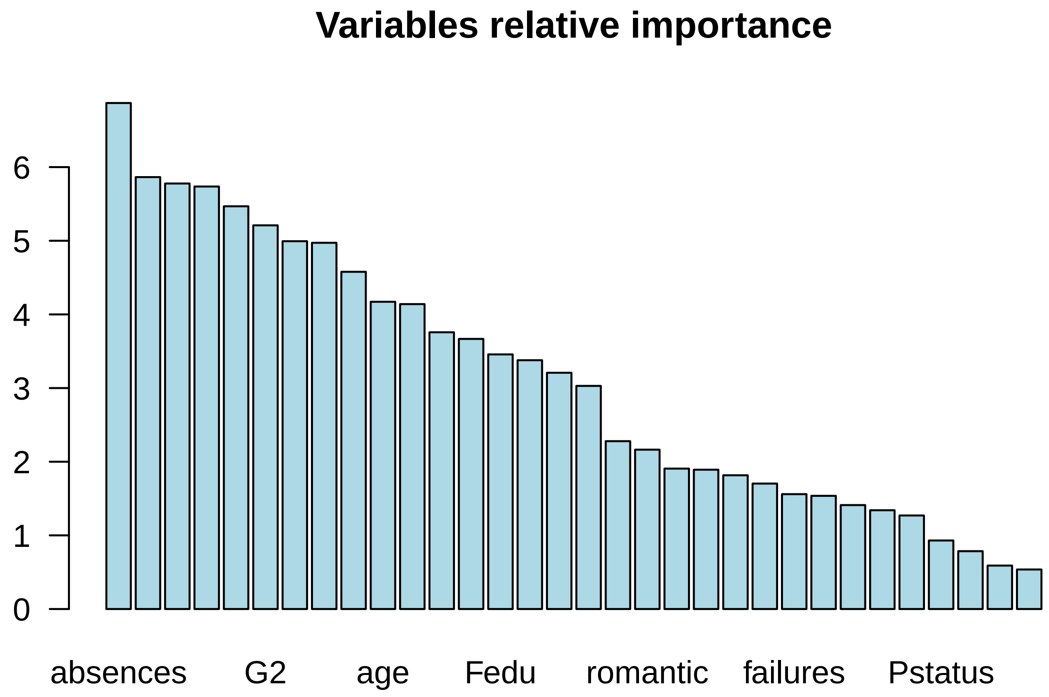
```
predict.test.boost1 <- predict(internet.boost1,newdata=Test.Student.Data)
predict.test.boost1$confusion
```

```
##                Observed Class
## Predicted Class no yes
##             no  10   8
##             yes 22  90
```

```
error11 <- errorevol(internet.boost1,newdata=Train.Student.Data)
error12 <- errorevol(internet.boost1,newdata=Test.Student.Data)
plot.errorevol(error2,error1)
```



```
#internet.boost2 <- boosting(internet~.,data=Train.Student.Data,boos=TRUE,mfinal=500)
predict.train.boost2 <- predict(internet.boost2,newdata=Train.Student.Data)
importanceplot(internet.boost2)
```



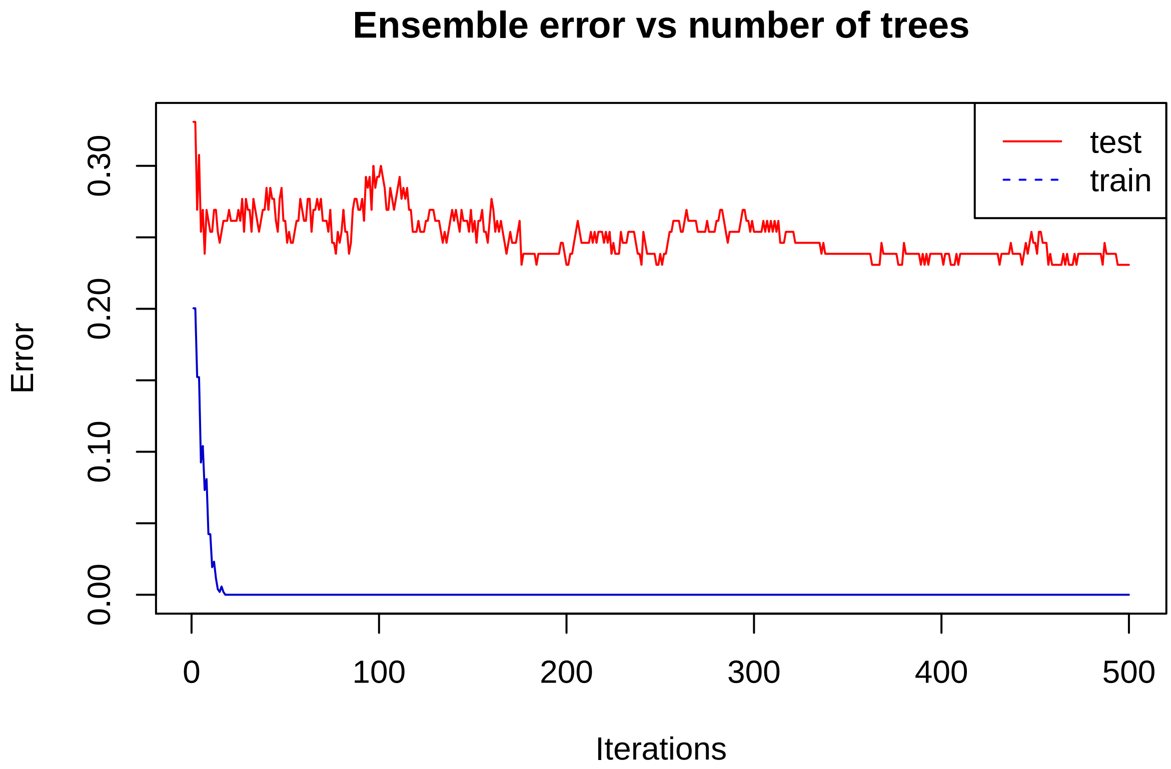
```
predict.train.boost2$confusion
```

```
##                Observed Class
## Predicted Class  no yes
##             no  119   0
##             yes   0 400
```

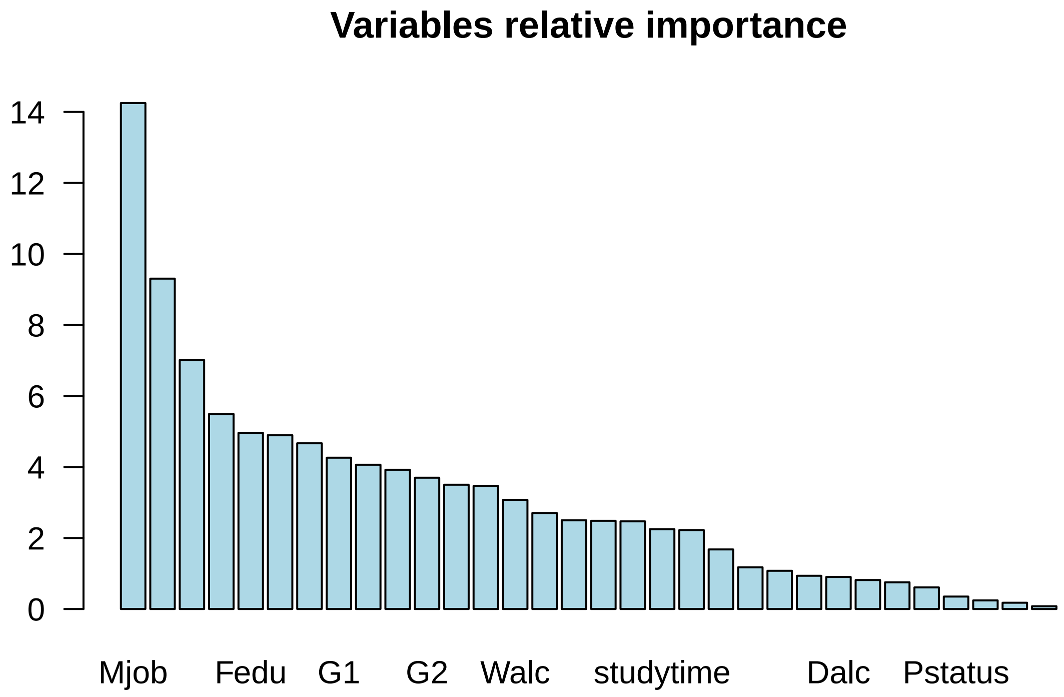
```
predict.test.boost2 <- predict(internet.boost2,newdata=Test.Student.Data)
predict.test.boost2$confusion
```

```
##                Observed Class
## Predicted Class no yes
##             no   8   9
##             yes 24  89
```

```
error21 <- errorevol(internet.boost1,newdata=Train.Student.Data)
error22 <- errorevol(internet.boost1,newdata=Test.Student.Data)
plot.errorevol(error22,error21)
```



```
#internet.bagging1 <- bagging(internet~.,data=Train.Student.Data,mfinal=50)
predict.train.bagging1 <- predict(internet.bagging1,newdata=Train.Student.Data)
importanceplot(internet.bagging1)
```



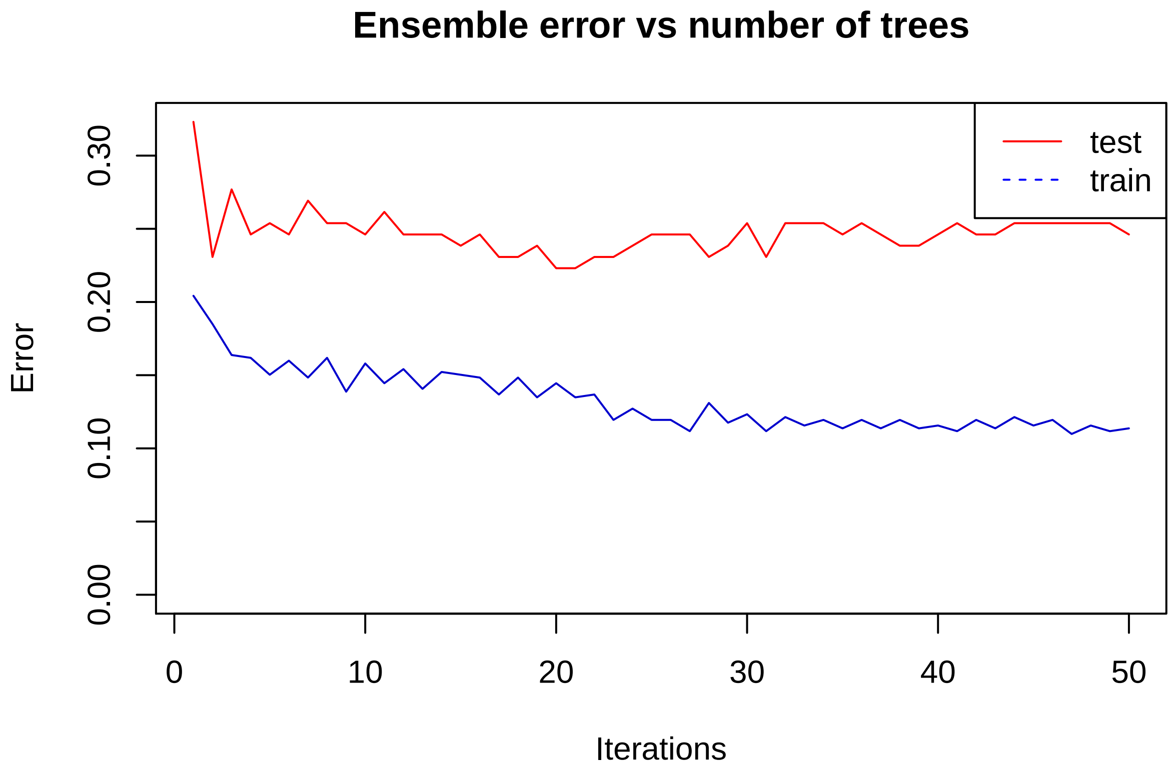
```
predict.train.bagging1$confusion
```

```
##                Observed Class
## Predicted Class  no yes
##             no   61   1
##             yes  58 399
```

```
predict.test.bagging1 <- predict(internet.bagging1,newdata=Test.Student.Data)
predict.test.bagging1$confusion
```

```
##                Observed Class
## Predicted Class no yes
##             no   3   3
##             yes 29  95
```

```
error1 <- errorevol(internet.bagging1,newdata=Train.Student.Data)
error2 <- errorevol(internet.bagging1,newdata=Test.Student.Data)
plot.errorevol(error2,error1)
```


Ensemble error vs number of trees
test
train
Error
0.00
0.10
0.20
0.30
0
10
20
30
40
50
Iterations