데이터 사이언티스트를 위한 DMR-5
Ensemble: bagging, boosting, Random Forest

Jinseog Kim
Dongguk University
jinseog.kim@gmail.com

2016-11-08

Supervised learning

1. generalized linear models
    1. linear regression
    2. logistic regression
    3. poisson, gamma, ...
2. non-parametric models
    1. decision tree, knn,
3. neural network(1 hidden layer)
4. penalized regression models
    1. ridge - l_2 penalty
    2. lasso - l_1 penalty
    3. elastic net - $l_1 + l_2$ penalty
5. **ensembles**
    1. bagging
    2. various boosting
    3. random forest
6. deep learning : h2o

Predictive modeling workflow and R methods(functions)

1. fit model `fit <- knn(trainingData, outcome, k = 5)`
2. check or validation models `print`, `plot`, `summary`
3. predict using selected model & test data `predict(fit, newdata)`
4. performance evaluation `ROCR`, 'caret' packages

Ensemble model

- Combine many base models(or weak learners) $\rightarrow$ strong model

$$f(x) = \sum_{i=1}^{M} w_i f_i(x) \tag{1}$$

1. Bagging: generate weak models (decision trees) from random samples and agregate them simply
2. Boosting: sequentially generate weak models from previous residuals & combime them with diferent weights
3. Random Forest: inject more randomness compared with Bagging

Ensemble R 패키지

R패키지	function	설명
ipred	bagging()	Bagging with rpart tree
adabag	bagging()	Bagging (rpart)
	boosting()	AdaBoost.M1
ada	Culp et al.	AdaBoost + Friedman's mods
randomForest	Breiman et al.	Random Forest
gbm	Ridgeway et al.	Stochastic Gradient Boosting
party	Hothorn	RF with faster tree growing
mboost	Hothorn	Boosting appl to glm, gam

bagging: fitting

```
library(ipred); data(spam, package = "ElemStatLearn")
# sampling index of train data
tr.idx <- sample(nrow(spam), 0.7*nrow(spam))
# fit bagging model with stump(2-nodes trees)
m.bag <- bagging(spam~., data=spam[tr.idx, ], nbagg=50, control=list(maxdepth=1))
names(m.bag)
```

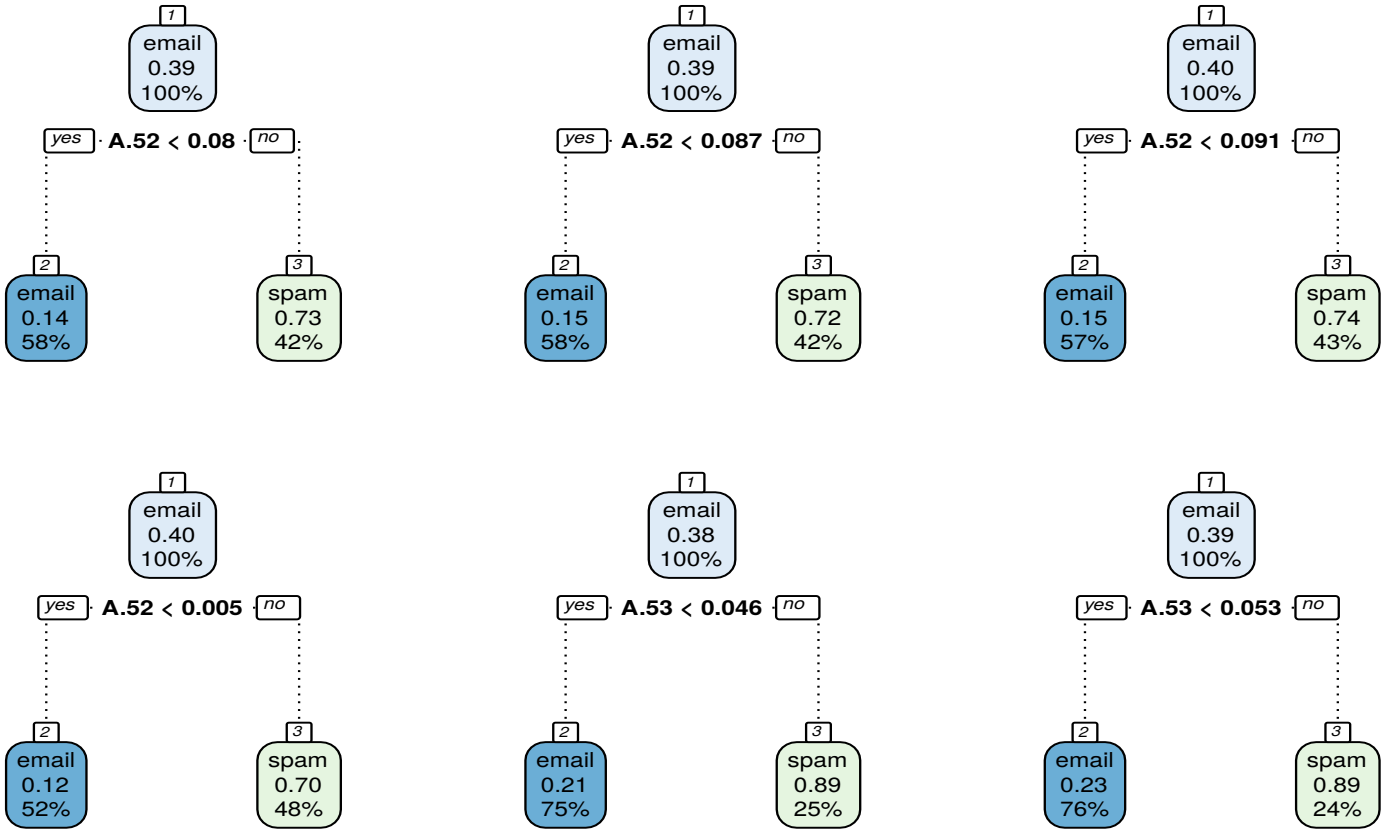
```
## [1] "y"      "X"      "mtrees" "OOB"    "comb"   "call"
```

```
m.bag$mtrees[[1]]$btree
```

```
## n= 3220
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3220 1245 email (0.6133540 0.3866460)
##   2) A.52< 0.0795 1876  260 email (0.8614072 0.1385928) *
##   3) A.52>=0.0795 1344  359 spam (0.2671131 0.7328869) *
```

bagging: plot bagging trees

```
library(rpart.plot); par(mfrow=c(2,3))
for(i in 1:6){
  rpart.plot(m.bag$mtrees[[i]]$btree, branch.lty=3, nn=TRUE)
}
```



bagging: prediction & performance

```
pred <- predict(m.bag, newdata=spam[-tr.idx, -58], type="class") #prob
kable(table(pred, spam$spam[-tr.idx]))
```

	email	spam
email	792	226
spam	47	316

boosting.M1: fitting

- Freund & Shapire (1996)

```
library(adabag); #require(rpart)
# fit boosting model with stump(2-nodes trees)
m.boo <- boosting(spam~., data=spam[tr.idx, ], mfinal=50, control=list(maxdepth=2))
names(m.boo)
```

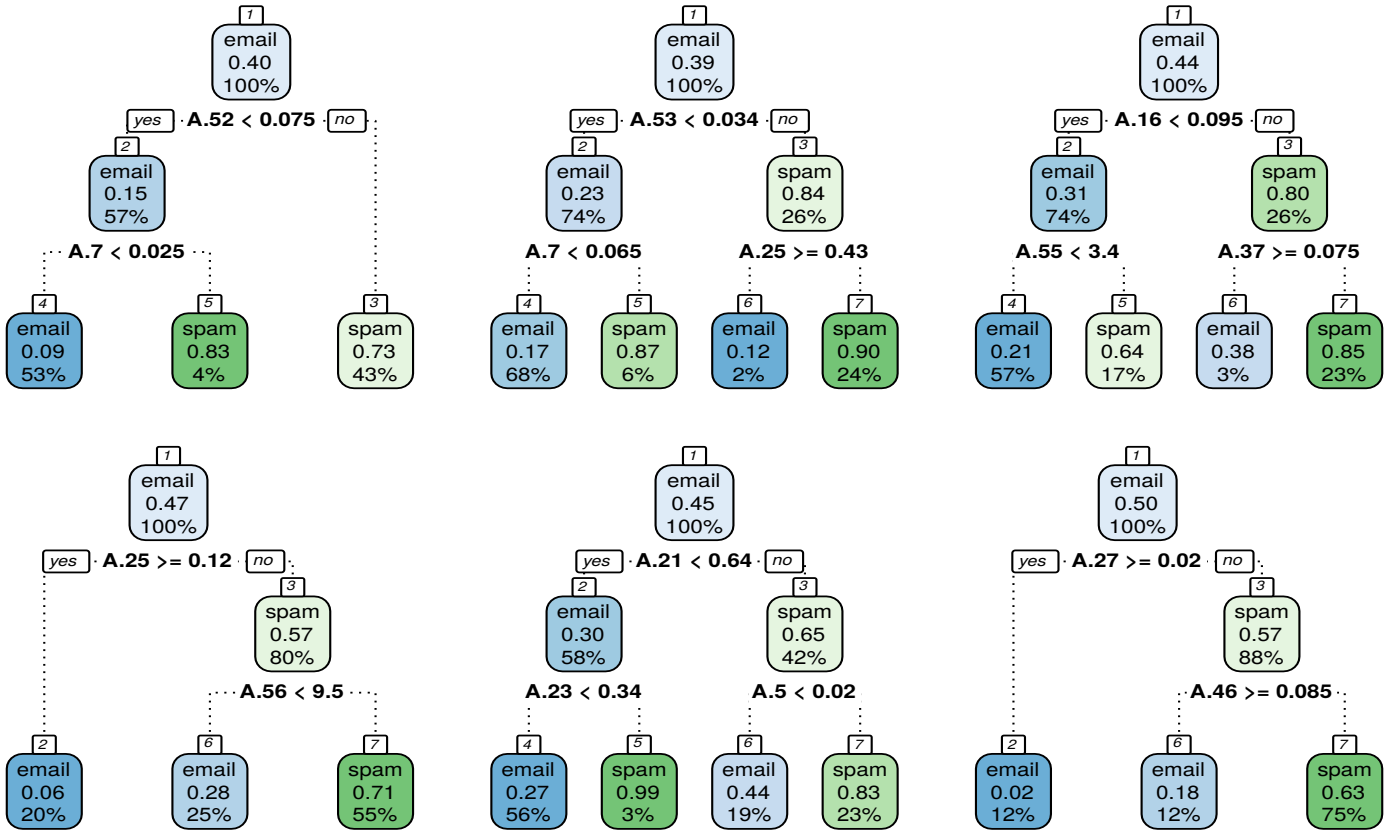
```
## [1] "formula"    "trees"      "weights"    "votes"      "prob"
## [6] "class"      "importance" "terms"      "call"
```

```
m.boo$trees[[1]]
```

```
## n= 3220
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 3220 1286 email (0.60062112 0.39937888)
##   2) A.52< 0.075 1823  265 email (0.85463522 0.14536478)
##     4) A.7< 0.025 1691  155 email (0.90833826 0.09166174) *
##     5) A.7>=0.025 132   22 spam (0.16666667 0.83333333) *
##   3) A.52>=0.075 1397  376 spam (0.26914817 0.73085183) *
```

boosting: plot boosting trees

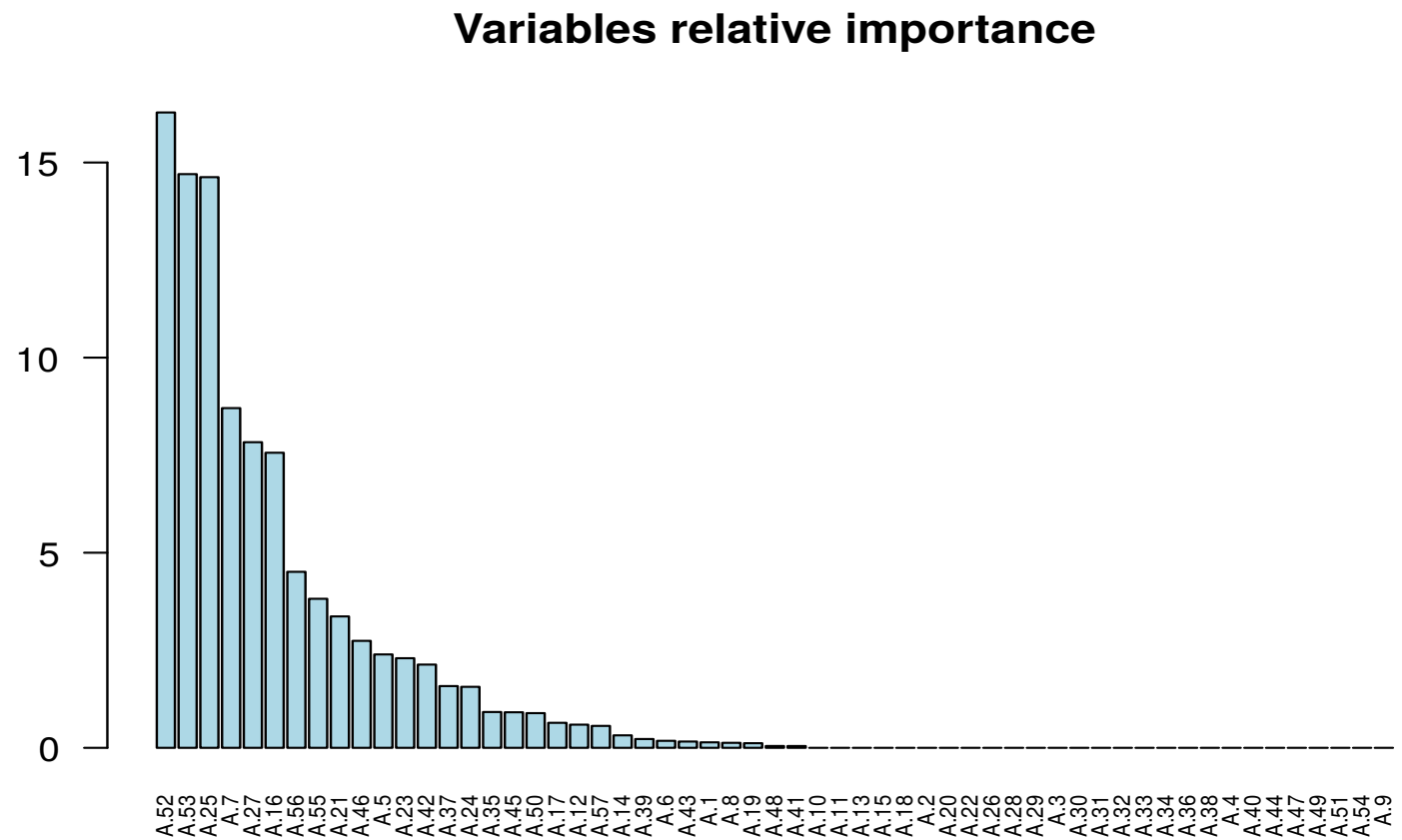
```
library(rpart.plot); par(mfrow=c(2,3))
for(i in 1:6){
  rpart.plot(m.boo$trees[[i]], branch.lty=3, nn=TRUE)
}
```



boosting: Variables Importance plot

- Gain of the Gini index of the variables

```
importanceplot(m.boo, las=2, cex.names=.6)
```



boosting: prediction & performance

```
pred.boo <- predict(m.boo, newdata=spam[-tr.idx, -58], type="class")$class #prob
kable(table(pred, spam$spam[-tr.idx]))
```

	email	spam
email	792	226
spam	47	316

```
kable(table(pred.boo, spam$spam[-tr.idx]))
```

	email	spam
email	806	47
spam	33	495

Random Forest (Breiman, 2001)

- Node size in tree: 2 (stump)
- Number of trees: 500
- No. of candidate split variables: 2-5

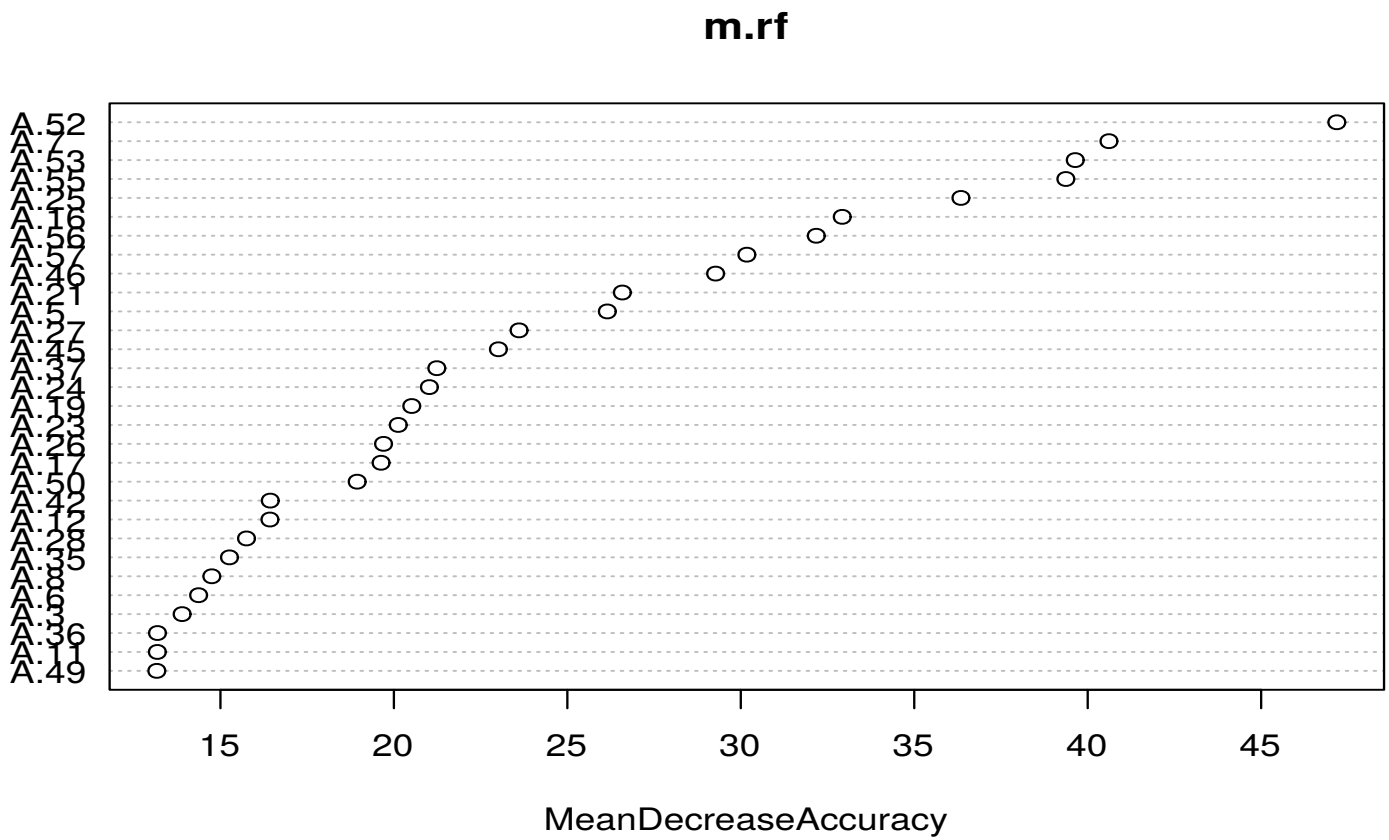
RF: fitting

```
library(randomForest); #require(rpart)
# fit RF model with stump(2-nodes trees)
m.rf <- randomForest(x=spam[tr.idx, -58], y=spam[tr.idx, 58],
                     ntree=500, mtry=7,
                     importance=TRUE)
names(m.rf)
```

```
##  [1] "call"            "type"            "predicted"
##  [4] "err.rate"        "confusion"       "votes"
##  [7] "oob.times"       "classes"         "importance"
## [10] "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"
## [16] "y"               "test"            "inbag"
```

RF: Variables Importance plot

```
# type=1: mean decrease in accuracy, 2:mean decrease in node impurity
varImpPlot(m.rf, type=1)
```

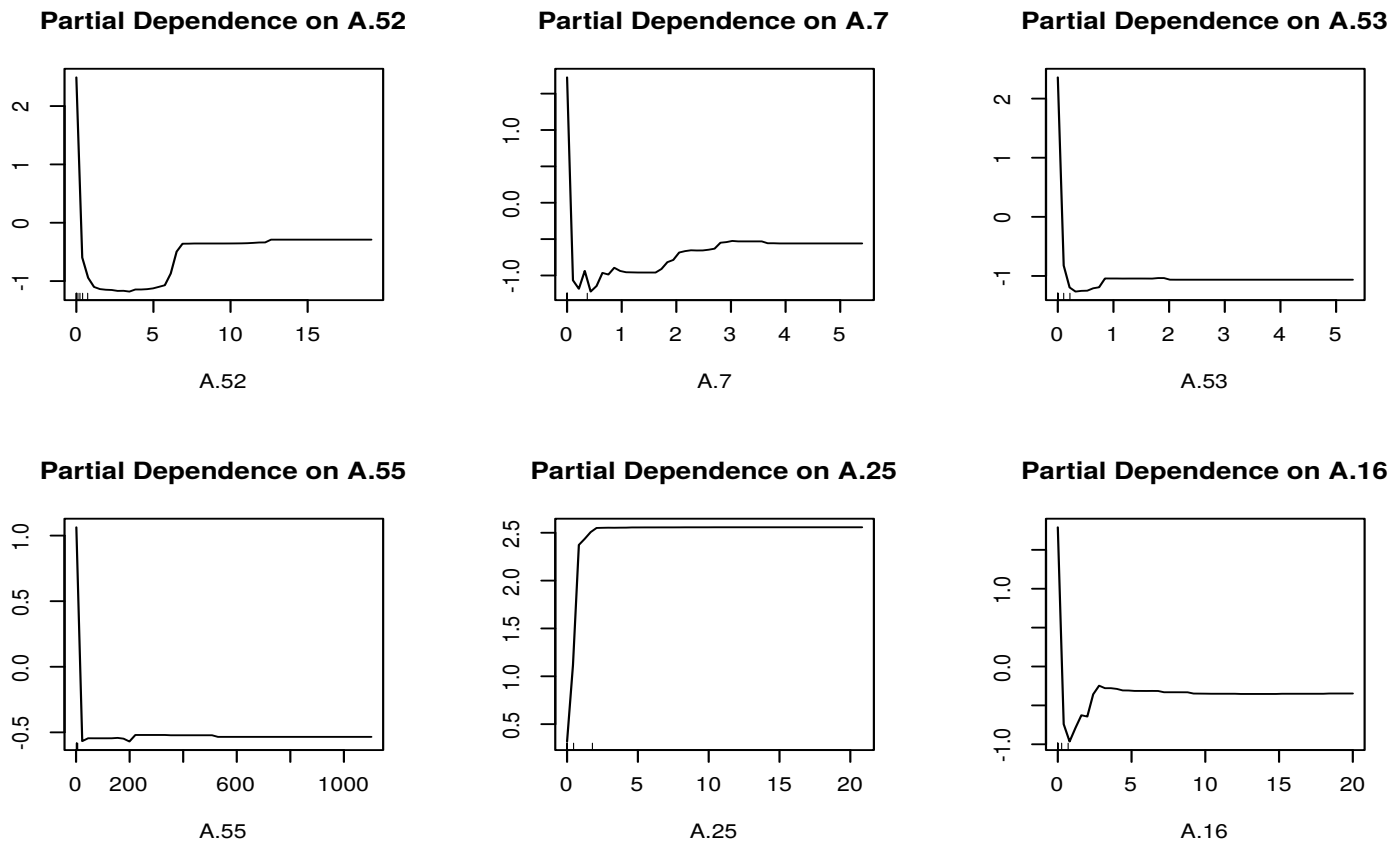


RF: Partial dependence plot

- partialPlot(x, pred.data, x.var)
  - x: rf model
  - pred.data : training data
  - x.var : variable name to be examined

```
imp <- importance(m.rf, type=1) # Mean Decrease Accuracy
impvar <- rownames(imp)[order(imp[, 1], decreasing=TRUE)]
par(mfrow=c(2, 3))
for (i in 1:6){
    partialPlot(m.rf, pred.data=spam[tr.idx,], impvar[i], xlab=impvar[i],
                main=paste("Partial Dependence on", impvar[i]))
}
```

RF: Partial dependence plot



RF: prediction & performance

```
pred.rf <- predict(m.rf, newdata=spam[-tr.idx, -58], type="response") #response/prob
table(pred.rf, spam$spam[-tr.idx])
```

```
##
## pred.rf email spam
##   email   814   40
##   spam     25  502
```

Comparison: bagging, boosting, RF

```
# bagging
mean(pred != spam$spam[-tr.idx])
```

```
## [1] 0.1976828
```

```
# boosting
mean(pred.boo != spam$spam[-tr.idx])
```

```
## [1] 0.05792904
```

```
# RF
mean(pred.rf != spam$spam[-tr.idx])
```

```
## [1] 0.04706734
```