Ensemble

Park Ju ho

2022 6 11

bagging

어군 탐지기 데이터

```
library(tree)
library(mlbench)
data(Sonar)
str(Sonar)
```

```
## 'data.frame':    208 obs. of  61 variables:
##  $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
##  $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173 ...
##  $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
##  $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
##  $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
##  $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
##  $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
##  $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
##  $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
##  $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
##  $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
##  $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
##  $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
##  $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
##  $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
##  $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
##  $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
##  $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
##  $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
##  $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
##  $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
##  $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
##  $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
##  $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
##  $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
##  $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
##  $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
##  $ V28  : num  0.808 0.277 0.853 0.252 1 ...
##  $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
##  $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
```

```
##  $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
##  $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
##  $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
##  $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
##  $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
##  $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
##  $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
##  $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
##  $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
##  $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
##  $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
##  $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
##  $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
##  $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
##  $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
##  $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
##  $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
##  $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
##  $ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092 ...
##  $ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.0198 ...
##  $ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.0118 ...
##  $ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.009 ...
##  $ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.0223 ...
##  $ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.0179 ...
##  $ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084 ...
##  $ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.0068 ...
##  $ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032 ...
##  $ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.0035 ...
##  $ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056 ...
##  $ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.004 ...
##  $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
```

데이터 전처리

```
#
clr = Sonar$Class; sonar = Sonar[,1:60]

#
snx = as.matrix(sonar)

#  0,1
sny = rep (1, 208); sny[which(clr == "R")] = 0
set.seed(120)

#test\validation
lst = sample(208)
tr = lst[1:145]
val = lst[146:208]
da = data.frame(y=clr, xx=snx)
```

tree 만들기

```
#
fgl.tr = tree(y ~ ., data=da[tr,], subset=tr)

#  k-fo
fgl.cv = cv.tree(fgl.tr, , prune.tree, K=10)
fgl.cv
```

```
## $size
## [1] 8 7 6 5 4 3 2 1
##
## $dev
## [1] 211.4754 211.8535 174.6241 164.6753 165.9568 160.7978 116.5989 140.4553
##
## $k
## [1]      -Inf  2.200898 11.822916 12.842449 13.239294 14.296887 15.837549
## [8] 32.301938
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

size = terminal nodes의 수 dev = cv_error k = cost-complexity parameter => 이 규제값을 활용하여 terminal nodes의 수를 결정

```
#dev   cv_err
opt = fgl.cv$k[which.min(fgl.cv$dev)]
opt
```

```
## [1] 15.83755
```

```
tt = prune.tree(fgl.tr, k=opt)
PP = predict(tt, da[val,], type="class")
mean(PP != clr[val])
```

```
## [1] 0.3968254
```

0.3 정도의 오분류율을 보여주고 있음

bagging 수행

```
library(adabag)
#mfinal: m =      =>
fit.bag = bagging(y ~., data=da[-val,], mfinal=50)
# bagging      (  )
predict.bagging(fit.bag, newdata=da[val,])$error
```

```
## [1] 0.2380952
```

총 50개의 나무가 생성됨(mfinal = 50)

오차율이 0.23으로 감소한 것을 알 수 있다

error이외에도 밑의 5가지 값들을 확인 가능

```
predict.bagging(fit.bag, newdata=da[val,])$class
```

```
##  [1] "M" "R" "R" "M" "M" "M" "M" "M" "M" "R" "M" "M" "R" "M" "M" "M" "R" "R" "R"
## [20] "M" "R" "R" "R" "R" "M" "M" "R" "M" "M" "R" "R" "R" "R" "M" "R" "M" "R" "R"
## [39] "M" "R" "M" "R" "R" "M" "R" "M" "M" "R" "M" "R" "M" "R" "M" "R" "M" "M" "M"
## [58] "R" "M" "M" "R" "R" "R"
```

votes => 50개의 나무가 각 data에 대하여 투표한 결과 probs => votes를 비율로 나타낸 것 class => 실제 입력 y sample => 145개의 표본을 복원추출한 결과들 importance => Feature의 중요도를 나타내는 것

boosting

데이터 전처리

```
wine = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",
                  sep=",", header=T)
colnames(wine) = c("Type","Alcohol","Malic","Ash","Alcalinity","Magnesium","Phenols","Flavanoids","Nonf

#test,train
lst = sample(nrow(wine)); tr = lst[1:100]; ts = lst[101:nrow(wine)]

#boosting    Type  factor
ds = wine[tr,]; ds$Type = as.numeric(ds$Type)
ds$Type[ds$Type>1] = 0;
```

boosting 수행

adaboost

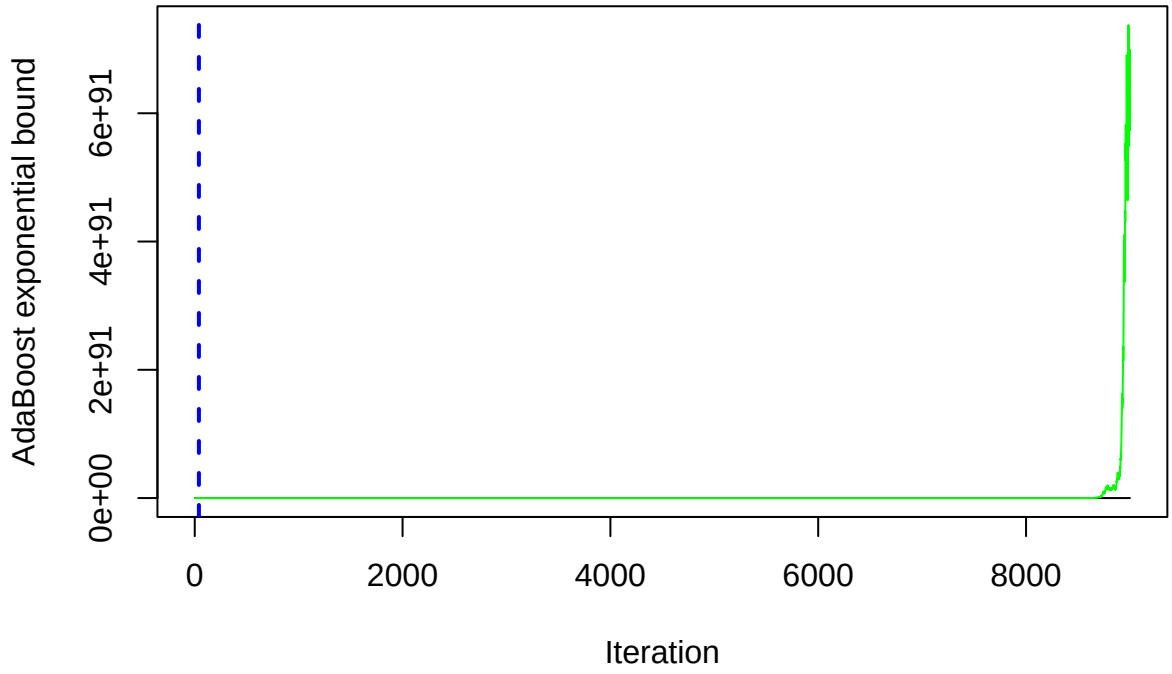
adaboost = boosting 방법 n.trees = 반복 횟수 cv.folds = k-fold validation

```
library(gbm)

ds1.gbm = gbm(Type ~ Alcohol + Malic + Ash + Alcalinity + Magnesium +
                Phenols + Flavanoids + Nonflavanoids +
                Proanthocyanins +
                Color + Hue + Dilution + Proline,
              data=ds, distribution="adaboost",
```

```
              n.trees=9000, cv.folds=5)

best1.iter = gbm.perf(ds1.gbm,method="cv")
```



여기서 iteration이 n.trees를 의미함 8000정도부터 exp가 급증하다가 9000이후로 조금 씩 내려옴 => 변화 이후에 안정되는 상태가 n.trees의 가장 적절한 숫자라고 할 수 있기에 조금 더 큰 n.tree를 사용할 필요가 있음

```
print(best1.iter)
```

```
## [1] 40
```

```
ds1.gbm
```

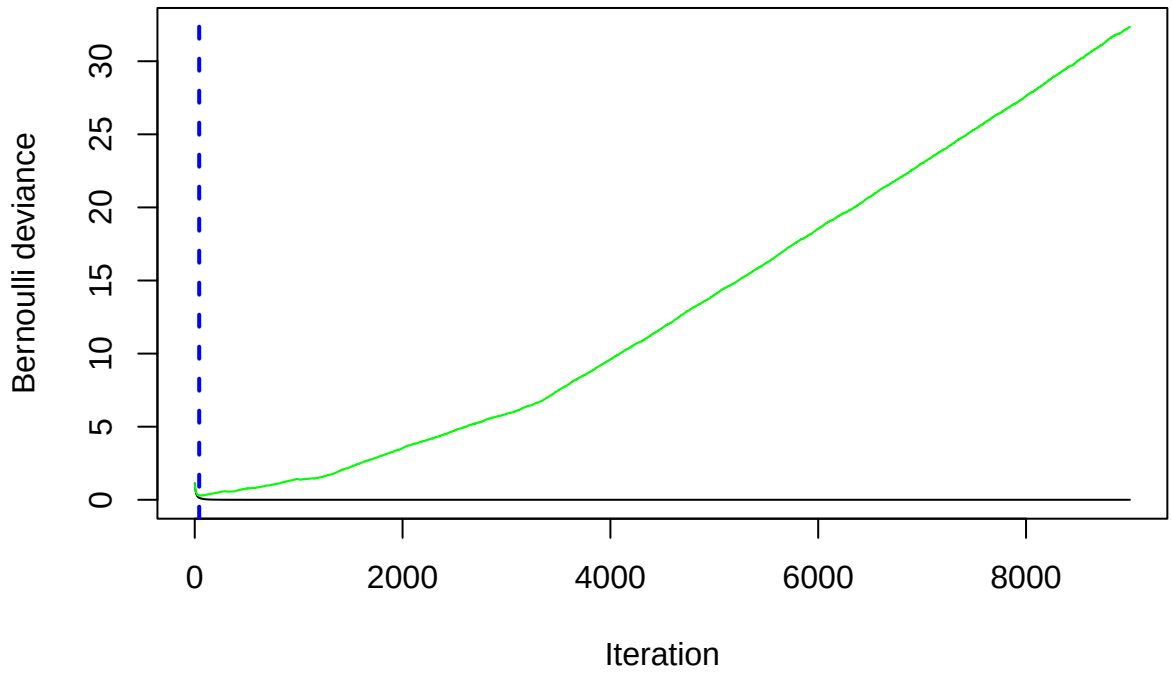
```
## gbm(formula = Type ~ Alcohol + Malic + Ash + Alcalinity + Magnesium +
##     Phenols + Flavanoids + Nonflavanoids + Proanthocyanins +
##     Color + Hue + Dilution + Proline, distribution = "adaboost",
##     data = ds, n.trees = 9000, cv.folds = 5)
## A gradient boosted model with adaboost loss function.
## 9000 iterations were performed.
## The best cross-validation iteration was 40.
## There were 13 predictors of which 7 had non-zero influence.
```

There were 13 predictors of which 5 had non-zero influence.= 즉 13개의 변수 중 5개만 영향을 준다고 할 수 있음

bernoulli

bernoulli 즉 일반적인 의사결정 나무를 사용 => 9000개의 나무이므로 Random Forest라고 볼 수 있음

```
ds2.gbm = gbm(Type ~ Alcohol + Malic + Ash + Alcalinity + Magnesium +
                Phenols + Flavanoids + Nonflavanoids +
                Proanthocyanins +
                Color + Hue + Dilution + Proline,
              data=ds, distribution="bernoulli", n.trees=9000, cv.folds=5)
best2.iter = gbm.perf(ds2.gbm,method="cv")
```




```
#deviance
print(best2.iter)
```

```
## [1] 43
```

오차율 비교

```
pp = predict(ds1.gbm,wine[ts,-1],type="response",n.trees=best1.iter)
#y  -1  1
pyy = ifelse(wine$Type[ts]>1, -1, 1)
```

```
# gbm        0.5     1    -1
#
mean(sign(pp-0.5) != pyy)
```

```
## [1] 0.03896104
```

```
pp = predict(ds2.gbm,wine[ts,-1],type="response",n.trees=best2.iter)
pyy = ifelse(wine$Type[ts]>1, -1, 1)
mean(sign(pp-0.5) != pyy)
```

```
## [1] 0.02597403
```

Random Forest

데이터 전처리

```
rm(list = ls())
setwd('D:/ /4-1 /  /R')
library(randomForest)
library(MASS)
library(gbm)

XY_tr = read.csv("LC_sample_tr.csv")
XY_ts = read.csv("LC_sample_ts.csv")
XY_tr = XY_tr[,-1]; XY_ts = XY_ts[,-1]
XY_tr[,4] = as.factor(XY_tr[,4]); XY_ts[,4] = as.factor(XY_ts[,4])
```

RF 수행

ntree = 나무 개수

```
RF_res = randomForest(y ~ ., data=XY_tr, ntree=1000, Importance=TRUE)
summary(RF_res)
```

```
##                 Length Class  Mode
## call               5   -none- call
## type               1   -none- character
## predicted        100   factor numeric
## err.rate        3000   -none- numeric
## confusion          6   -none- numeric
## votes            200   matrix numeric
## oob.times        100   -none- numeric
## classes            2   -none- character
## importance         3   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
```

```
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                100   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

importanceSD = GINI 지수에 대한 표준 편차

```
RF_res$importance
```

```
##           MeanDecreaseGini
## A1AT              16.15018
## CYFRA21.1         20.33849
## RANTES            12.03281
```

MeanDecreaseGini = GINI지수의 평균값을 말해줌(작을 수록 좋음)

```
RF_res$confusion
```

```
##    0  1 class.error
## 0 47  3        0.06
## 1  3 47        0.06
```

class.error = 범주 0과 1에서 각각의 error율

```
PP = predict(RF_res, XY_ts[,1:3])
mean(PP != XY_ts[,4])
```

```
## [1] 0.075
```

7.5%정도의 오분류율을 가짐(Random이기 때문에 결과가 달라짐 하지만 크게 다르진 않음)

Spam mail 데이터를 활용한 ensemble

tree

데이터 전처리

```
spamD <- read.table('https://raw.github.com/WinVector/zmPDSwR/master/Spambase/spamD.tsv',header=T,sep='
head(spamD)
```

```
##   word.freq.make word.freq.address word.freq.all word.freq.3d word.freq.our
## 1           0.00              0.64          0.64            0          0.32
## 2           0.21              0.28          0.50            0          0.14
## 3           0.06              0.00          0.71            0          1.23
```

```
## 4           0.00              0.00          0.00            0          0.63
## 5           0.00              0.00          0.00            0          0.63
## 6           0.00              0.00          0.00            0          1.85
##   word.freq.over word.freq.remove word.freq.internet word.freq.order
## 1           0.00             0.00              0.00           0.00
## 2           0.28             0.21              0.07           0.00
## 3           0.19             0.19              0.12           0.64
## 4           0.00             0.31              0.63           0.31
## 5           0.00             0.31              0.63           0.31
## 6           0.00             0.00              1.85           0.00
##   word.freq.mail word.freq.receive word.freq.will word.freq.people
## 1           0.00              0.00           0.64            0.00
## 2           0.94              0.21           0.79            0.65
## 3           0.25              0.38           0.45            0.12
## 4           0.63              0.31           0.31            0.31
## 5           0.63              0.31           0.31            0.31
## 6           0.00              0.00           0.00            0.00
##   word.freq.report word.freq.addresses word.freq.free word.freq.business
## 1             0.00                0.00           0.32               0.00
## 2             0.21                0.14           0.14               0.07
## 3             0.00                1.75           0.06               0.06
## 4             0.00                0.00           0.31               0.00
## 5             0.00                0.00           0.31               0.00
## 6             0.00                0.00           0.00               0.00
##   word.freq.email word.freq.you word.freq.credit word.freq.your word.freq.font
## 1            1.29          1.93             0.00           0.96              0
## 2            0.28          3.47             0.00           1.59              0
## 3            1.03          1.36             0.32           0.51              0
## 4            0.00          3.18             0.00           0.31              0
## 5            0.00          3.18             0.00           0.31              0
## 6            0.00          0.00             0.00           0.00              0
##   word.freq.000 word.freq.money word.freq.hp word.freq.hpl word.freq.george
## 1          0.00             0.00           0             0                0
## 2          0.43             0.43           0             0                0
## 3          1.16             0.06           0             0                0
## 4          0.00             0.00           0             0                0
## 5          0.00             0.00           0             0                0
## 6          0.00             0.00           0             0                0
##   word.freq.650 word.freq.lab word.freq.labs word.freq.telnet word.freq.857
## 1             0             0              0                0             0
## 2             0             0              0                0             0
## 3             0             0              0                0             0
## 4             0             0              0                0             0
## 5             0             0              0                0             0
## 6             0             0              0                0             0
##   word.freq.data word.freq.415 word.freq.85 word.freq.technology word.freq.1999
## 1              0             0            0                    0          0.00
## 2              0             0            0                    0          0.07
## 3              0             0            0                    0          0.00
## 4              0             0            0                    0          0.00
## 5              0             0            0                    0          0.00
## 6              0             0            0                    0          0.00
##   word.freq.parts word.freq.pm word.freq.direct word.freq.cs word.freq.meeting
## 1               0            0             0.00            0                0
```

```
## 2               0           0             0.00           0                0
## 3               0           0             0.06           0                0
## 4               0           0             0.00           0                0
## 5               0           0             0.00           0                0
## 6               0           0             0.00           0                0
##   word.freq.original word.freq.project word.freq.re word.freq.edu
## 1               0.00                 0         0.00          0.00
## 2               0.00                 0         0.00          0.00
## 3               0.12                 0         0.06          0.06
## 4               0.00                 0         0.00          0.00
## 5               0.00                 0         0.00          0.00
## 6               0.00                 0         0.00          0.00
##   word.freq.table word.freq.conference char.freq.semi char.freq.lparen
## 1               0                    0           0.00            0.000
## 2               0                    0           0.00            0.132
## 3               0                    0           0.01            0.143
## 4               0                    0           0.00            0.137
## 5               0                    0           0.00            0.135
## 6               0                    0           0.00            0.223
##   char.freq.lbrack char.freq.bang char.freq.dollar char.freq.hash
## 1                0          0.778            0.000          0.000
## 2                0          0.372            0.180          0.048
## 3                0          0.276            0.184          0.010
## 4                0          0.137            0.000          0.000
## 5                0          0.135            0.000          0.000
## 6                0          0.000            0.000          0.000
##   capital.run.length.average capital.run.length.longest
## 1                      3.756                         61
## 2                      5.114                        101
## 3                      9.821                        485
## 4                      3.537                         40
## 5                      3.537                         40
## 6                      3.000                         15
##   capital.run.length.total spam rgroup
## 1                      278 spam     52
## 2                     1028 spam     91
## 3                     2259 spam     49
## 4                      191 spam     88
## 5                      191 spam     73
## 6                       54 spam     45
```

```
#rgoup     test  train
spamTrain <- subset(spamD,spamD$rgroup>=10)
spamTest <- subset(spamD,spamD$rgroup<10)

#setdiff =         x  y
spamVars <- setdiff(colnames(spamD),list('rgroup','spam'))
spamFormula <- as.formula(paste('spam=="spam"',
                                paste(spamVars,collapse=' + '),sep=' ~ '))

#
loglikelihood <- function(y, py) {
  pysmooth <- ifelse(py==0, 1e-12,
                     ifelse(py==1, 1-1e-12, py))
```

```
  sum(y * log(pysmooth) + (1-y)*log(1 - pysmooth))
}

#
accuracyMeasures <- function(pred, truth, name="model") {
  dev.norm <- -2*loglikelihood(as.numeric(truth), pred)/length(pred)#Deviance
  ctable <- table(truth=truth,
                  pred=(pred>0.5))
  accuracy <- sum(diag(ctable))/sum(ctable)
  precision <- ctable[2,2]/sum(ctable[,2])
  recall <- ctable[2,2]/sum(ctable[2,])
  f1 <- 2*precision*recall/(precision+recall)
  data.frame(model=name, accuracy=accuracy, f1=f1, dev.norm)
}
```

tree 수행

```
library(rpart)
treemodel <- rpart(spamFormula , spamTrain)
```

평가

```
accuracyMeasures(predict(treemodel, newdata=spamTrain),
                 spamTrain$spam=="spam",
                 name="tree, training")
```

```
##            model  accuracy      f1  dev.norm
## 1 tree, training 0.9104514 0.88337 0.5618654
```

```
accuracyMeasures(predict(treemodel, newdata=spamTest),
                 spamTest$spam=="spam",
                 name="tree, test")
```

```
##        model  accuracy        f1  dev.norm
## 1 tree, test 0.8799127 0.8414986 0.6702857
```

train의 결과를 보았을 때 상당히 학습 잘 된 것을 알 수 있음. test의 결과를 보았을 때 대략 87의 정확도를 보여주고 있음

bagging

bootstrap

```
#
ntrain <- dim(spamTrain)[1]
n <- ntrain
ntree <- 100

#
#ntree = 100        100             100
samples <- sapply(1:ntree,
                  FUN = function(iter)
                  {sample(1:ntrain, size=n, replace=T)})#     replace=T

#bagging
#            (   )
predict.bag <- function(treelist, newdata) {
  preds <- sapply(1:length(treelist),
                  FUN=function(iter) {
                    predict(treelist[[iter]], newdata=newdata)})
  predsums <- rowSums(preds)
  predsums/length(treelist)
}
```

tree 생성

```
treelist <-lapply(1:ntree,
                  FUN=function(iter)
                  {samp <- samples[,iter];
                  rpart(spamFormula, spamTrain[samp,])})
```

평가

```
accuracyMeasures(predict.bag(treelist, newdata=spamTrain),
                 spamTrain$spam=="spam",
                 name="bagging, training")
```

```
##               model  accuracy        f1  dev.norm
## 1 bagging, training 0.9227613 0.8992443 0.4693575
```

```
accuracyMeasures(predict.bag(treelist, newdata=spamTest),
                 spamTest$spam=="spam",
                 name="bagging, test")
```

```
##           model  accuracy        f1  dev.norm
## 1 bagging, test 0.9104803 0.8797654 0.5313188
```

test 데이터의 결과를 보았을 때 그냥 트리보다 정확도가 향상한 것을 확인 할 수 있음

RandomForest

```
set.seed(12345)

fmodel <- randomForest(x=spamTrain[,spamVars],
                       y=factor(spamTrain$spam),
                       ntree=100,
                       nodesize=7,#Tree  node   (   2  Tree     2  Tree   )
                       importance=T)

summary(fmodel)
```

```
##                 Length Class  Mode
## call               6   -none- call
## type               1   -none- character
## predicted       4143   factor numeric
## err.rate         300   -none- numeric
## confusion          6   -none- numeric
## votes           8286   matrix numeric
## oob.times       4143   -none- numeric
## classes            2   -none- character
## importance       228   -none- numeric
## importanceSD     171   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y               4143   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
```

```
fmodel$err.rate
```

```
##                OOB   non-spam       spam
##   [1,] 0.11722331 0.09200438 0.15472313
##   [2,] 0.12853678 0.10180844 0.16921509
##   [3,] 0.11258492 0.08847185 0.14926591
##   [4,] 0.10071942 0.06932574 0.14901388
##   [5,] 0.10051199 0.07034728 0.14675768
##   [6,] 0.09741602 0.06817213 0.14248194
##   [7,] 0.09285895 0.06320166 0.13863928
##   [8,] 0.08775409 0.06076672 0.12958281
##   [9,] 0.08582639 0.05611627 0.13179263
##  [10,] 0.08028306 0.05225080 0.12360248
##  [11,] 0.07632474 0.04807692 0.11990111
##  [12,] 0.07296970 0.04558177 0.11514778
##  [13,] 0.07263923 0.04394726 0.11677935
##  [14,] 0.07232704 0.04668795 0.11179361
##  [15,] 0.06648936 0.04148385 0.10497238
##  [16,] 0.06618357 0.04103586 0.10490798
##  [17,] 0.06520164 0.04143426 0.10177805
```

```
##  [18,] 0.06615162 0.04143426 0.10416667
##  [19,] 0.06542733 0.04023904 0.10416667
##  [20,] 0.06323920 0.03745020 0.10287814
##  [21,] 0.06348057 0.03824701 0.10226577
##  [22,] 0.06179097 0.03824701 0.09797918
##  [23,] 0.05937726 0.03705179 0.09369259
##  [24,] 0.05865315 0.03505976 0.09491733
##  [25,] 0.05841178 0.03466135 0.09491733
##  [26,] 0.05768767 0.03426295 0.09369259
##  [27,] 0.05792904 0.03545817 0.09246785
##  [28,] 0.05913589 0.03665339 0.09369259
##  [29,] 0.05672218 0.03505976 0.09001837
##  [30,] 0.05913589 0.03824701 0.09124311
##  [31,] 0.06058412 0.03824701 0.09491733
##  [32,] 0.05792904 0.03585657 0.09185548
##  [33,] 0.05696355 0.03545817 0.09001837
##  [34,] 0.05744629 0.03426295 0.09308022
##  [35,] 0.05768767 0.03665339 0.09001837
##  [36,] 0.05672218 0.03426295 0.09124311
##  [37,] 0.05744629 0.03505976 0.09185548
##  [38,] 0.05792904 0.03505976 0.09308022
##  [39,] 0.05744629 0.03585657 0.09063074
##  [40,] 0.05599807 0.03545817 0.08756889
##  [41,] 0.05744629 0.03665339 0.08940600
##  [42,] 0.05841178 0.03824701 0.08940600
##  [43,] 0.05792904 0.03705179 0.09001837
##  [44,] 0.05696355 0.03784861 0.08634415
##  [45,] 0.05672218 0.03625498 0.08818126
##  [46,] 0.05623944 0.03625498 0.08695652
##  [47,] 0.05648081 0.03665339 0.08695652
##  [48,] 0.05527396 0.03466135 0.08695652
##  [49,] 0.05575670 0.03545817 0.08695652
##  [50,] 0.05406710 0.03386454 0.08511941
##  [51,] 0.05527396 0.03505976 0.08634415
##  [52,] 0.05623944 0.03585657 0.08756889
##  [53,] 0.05454984 0.03545817 0.08389467
##  [54,] 0.05479121 0.03625498 0.08328230
##  [55,] 0.05406710 0.03545817 0.08266993
##  [56,] 0.05454984 0.03585657 0.08328230
##  [57,] 0.05382573 0.03466135 0.08328230
##  [58,] 0.05527396 0.03585657 0.08511941
##  [59,] 0.05430847 0.03505976 0.08389467
##  [60,] 0.05406710 0.03466135 0.08389467
##  [61,] 0.05382573 0.03386454 0.08450704
##  [62,] 0.05454984 0.03426295 0.08573178
##  [63,] 0.05334299 0.03386454 0.08328230
##  [64,] 0.05430847 0.03426295 0.08511941
##  [65,] 0.05382573 0.03466135 0.08328230
##  [66,] 0.05382573 0.03426295 0.08389467
##  [67,] 0.05382573 0.03386454 0.08450704
##  [68,] 0.05358436 0.03346614 0.08450704
##  [69,] 0.05503259 0.03545817 0.08511941
##  [70,] 0.05527396 0.03545817 0.08573178
##  [71,] 0.05454984 0.03346614 0.08695652
```

```
##  [72,] 0.05310162 0.03266932 0.08450704
##  [73,] 0.05310162 0.03306773 0.08389467
##  [74,] 0.05213613 0.03187251 0.08328230
##  [75,] 0.05310162 0.03266932 0.08450704
##  [76,] 0.05310162 0.03426295 0.08205756
##  [77,] 0.05334299 0.03346614 0.08389467
##  [78,] 0.05286025 0.03386454 0.08205756
##  [79,] 0.05358436 0.03306773 0.08511941
##  [80,] 0.05261888 0.03227092 0.08389467
##  [81,] 0.05334299 0.03227092 0.08573178
##  [82,] 0.05406710 0.03346614 0.08573178
##  [83,] 0.05286025 0.03266932 0.08389467
##  [84,] 0.05358436 0.03266932 0.08573178
##  [85,] 0.05358436 0.03266932 0.08573178
##  [86,] 0.05310162 0.03147410 0.08634415
##  [87,] 0.05334299 0.03147410 0.08695652
##  [88,] 0.05237750 0.03067729 0.08573178
##  [89,] 0.05189476 0.03107570 0.08389467
##  [90,] 0.05261888 0.03107570 0.08573178
##  [91,] 0.05237750 0.03107570 0.08511941
##  [92,] 0.05286025 0.03107570 0.08634415
##  [93,] 0.05310162 0.03147410 0.08634415
##  [94,] 0.05213613 0.03027888 0.08573178
##  [95,] 0.05261888 0.03107570 0.08573178
##  [96,] 0.05237750 0.03107570 0.08511941
##  [97,] 0.05358436 0.03227092 0.08634415
##  [98,] 0.05358436 0.03306773 0.08511941
##  [99,] 0.05358436 0.03227092 0.08634415
## [100,] 0.05334299 0.03147410 0.08695652
```

```
accuracyMeasures(predict(fmodel,
                         newdata=spamTrain[,spamVars], type='prob')[,'spam'],
                 spamTrain$spam=="spam",name="random forest, train")
```

```
##                   model  accuracy        f1 dev.norm
## 1 random forest, train 0.9879315 0.9845679 0.144498
```

```
accuracyMeasures(predict(fmodel,
                         newdata=spamTest[,spamVars],type='prob')[,'spam'],
                 spamTest$spam=="spam",name="random forest, test")
```

```
##                  model  accuracy        f1 dev.norm
## 1 random forest, test 0.9497817 0.9340974 0.412967
```

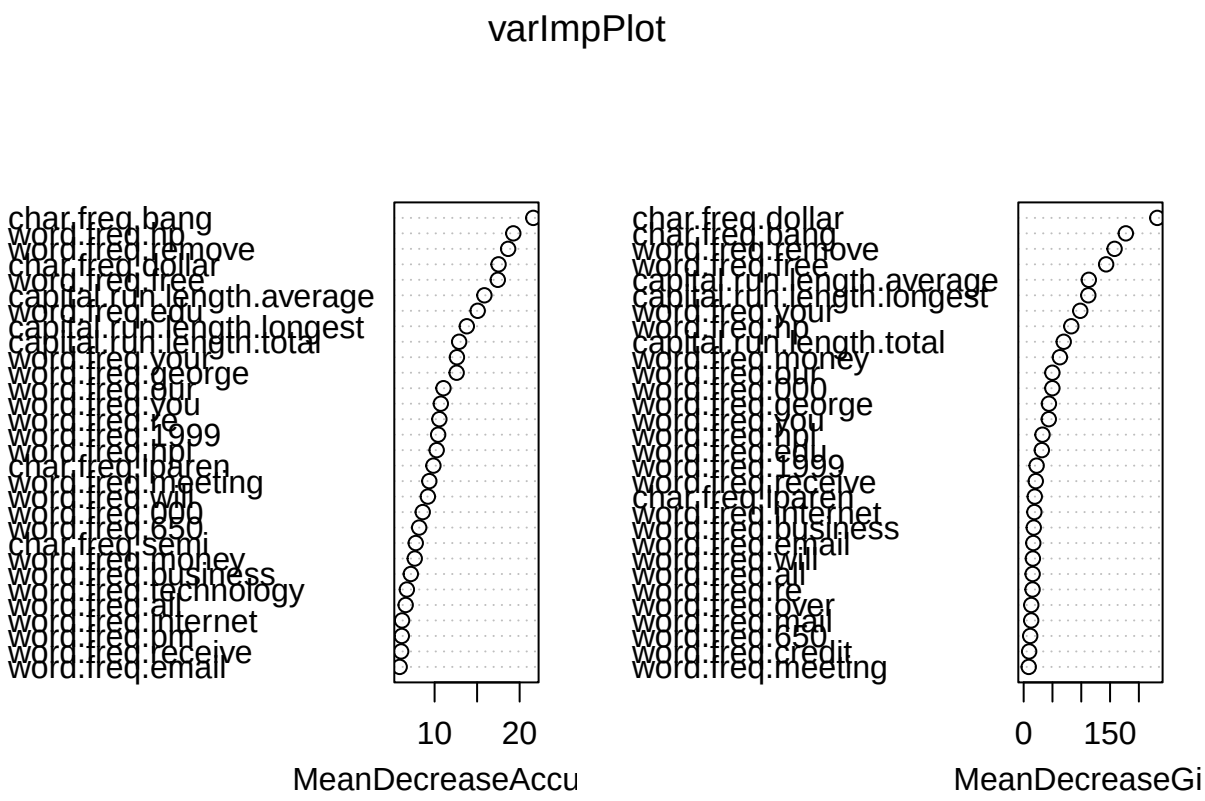
정확도가 더 개선된 것을 확인 할 수 있다

변수별 중요도

```
varImp <- importance(fmodel)
varImp[1:10, ]
```

```
##                      non-spam      spam MeanDecreaseAccuracy MeanDecreaseGini
## word.freq.make       1.645196  4.083521             3.713649         6.072775
## word.freq.address    3.436308  3.123265             4.066207         7.116630
## word.freq.all        2.191401  6.275111             6.598632        15.554456
## word.freq.3d         2.652428  1.801960             2.745202         1.700460
## word.freq.our        8.824627  9.490017            11.042159        49.830336
## word.freq.over       5.191639  4.203951             5.853567        13.218367
## word.freq.remove    16.921797 14.758570            18.648347       157.822115
## word.freq.internet   5.809705  3.575922             6.176436        18.584118
## word.freq.order      4.211616  3.056288             4.773744         7.122843
## word.freq.mail       4.375893  4.369267             5.474268        13.069926
```

```r
varImpPlot(fmodel, main="varImpPlot")
```



변수별 중요도를 파악 할 수 있음

MeanDecreaseAccuracy = 분류 정확도를 개선하는데 기여한 정도 => 높을 수록 좋음 MeanDecreaseGini = 노드 불순도 개선에 기여한 정도 => 높을 수록 좋음