# 10장 앙상블모형


충북대학교 정보통계학과 나 종 화
(cherin@cbnu.ac.kr)

# CONTENTS

10.1 서론

10.2 배깅

10.3 부스팅

10.4 랜덤포리스트

10.5* {caret}를 이용한 랜덤포리스트

# 10.1 서론

- 앙상블(ensemble) 모형은 여러 개의 분류모형에 의한 결과를 종합하여 분류의 정확도를 높이는 방법이다.
- 이는 적절한 표본추출법으로 데이터에서 여러 개의 훈련용 데이터 집합을 만들어 각각의 데이터 집합에서 하나의 분류기를 만들어 앙상블 하는 방법이다. 즉, 새로운 자료에 대해 분류기 예측값들의 가중 투표(weighted vote)를 통해 분류를 수행한다.
- 데이터를 조절하는 가장 대표적인 방법에는 배깅(bagging)과 부스팅(boosting)이 있다. 랜덤 포리스트(random forest) 방법은 배깅의 개념과 속성(또는 변수)의 임의 선택(random selection)을 결합한 앙상블 기법이다.

# 10.1 서론

- 앙상블 방법은 개별 모형에 비해 다음의 장점을 가진다.
  - 평균을 취함으로써 편의를 제거해준다: 치우침이 있는 여러 모형의 평균을 취하면, 어느 쪽에도 치우치지 않는 결과(평균)를 얻게 된다.
  - 분산을 감소시킨다: 한 개 모형으로부터의 단일 의견보다 여러 모형의 의견을 결합하면 변동이 작아진다.
  - 과적합의 가능성을 줄여준다: 과적합이 없는 각 모형으로부터 예측을 결합(평균, 가중 평균, 로지스틱 회귀)하면 과적합의 여지가 줄어든다.

# 10.2 배깅

- 배깅(bagging)은 bootstrap aggregating의 준말로 원 데이터 집합으로부터 크기가 같은 표본을 여러 번 단순임의 복원 추출하여 각 표본(이를 붓스트랩 표본이라 함)에 대해 분류기(classifiers)를 생성한 후 그 결과를 앙상블 하는 방법이다.
- 반복추출 방법을 사용하기 때문에 같은 데이터가 한 표본에 여러 번 추출될 수도 있고, 어떤 데이터는 추출되지 않을 수도 있다.
- 데이터가 충분히 큰 경우, 각 데이터가 하나의 붓스트랩 표본에서 제외될 확률은 36.78%이다

  $(\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=e^{-1}=0.3678)$.

# 10.2 배깅

**예제 3** iris 자료에 대해 R 패키지 {adabag}의 bagging() 함수를 통해 분석을 수행한다.

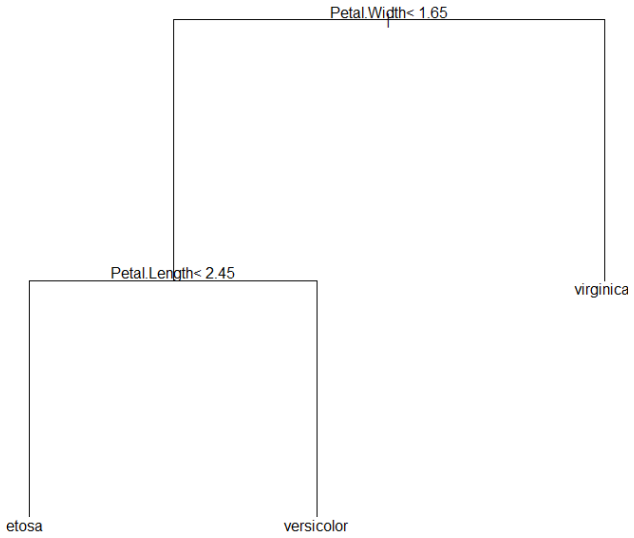
```
> library(adabag)
> data(iris)
> iris.bagging <- bagging(Species~., data=iris, mfinal=10)
> # mfinal= 반복수 또는 트리의 수(디폴트=100)
> iris.bagging$importance    # 변수의 상대적인 중요도
 Petal.Length Petal.Width Sepal.Length Sepal.Width
     76.70895    23.29105      0.00000     0.00000
```

변수의 중요도는 각 트리에서 변수에 의해 주어지는 지니지수의 이익(gain)(또는 불확실성의 감소량)을 고려한 측도이다.

# 10.2 배깅

- R 패키지 {adabag}의 bagging() 함수는 배깅을 이용하여 분류를 수행한다. plot() 함수를 통해 분류 결과를 트리 형태로 나타낼 수 있다.

```
> plot(iris.bagging$trees[[10]])
> text(iris.bagging$trees[[10]])
```



# 10.2 배깅

- redict() 함수를 통해 새로운 자료에 대한 분류(예측)를 수행 할 수 있다. 여기서는 모형 구축에 사용된 자료를 재사용하여 분류를 수행하였다. 그 결과 setosa는 50개 모두, versicolor는 50개 중 45개, virginica는 50개 중 49개가 제대로 분류되었다.

```
> pred <- predict(iris.bagging, newdata=iris)
> table(pred$class, iris[,5])
            setosa versicolor virginica
 setosa         50          0         0
 versicolor      0         45         1
 virginica       0          5        49
```

# 10.3 부스팅

- 부스팅(boosting)은 배깅의 과정과 유사하나 붓스트랩 표본을 구성하는 재표본(re-sampling) 과정에서 각 자료에 동일한 확률을 부여하는 것이 아니라, 분류가 잘못된 데이터에 더 큰 가중을 주어 표본을 추출한다.
- 부스팅에서는 붓스트랩 표본을 추출하여 분류기를 만든 후, 그 분류결과를 이용하여 각 데이터가 추출될 확률을 조정한 후, 다음 붓스트랩 표본을 추출하는 과정을 반복한다.
- 아다부스팅(AdaBoosting: adaptive boosting)은 가장 많이 사용되는 부스팅 알고리즘이다.

# 10.3 부스팅

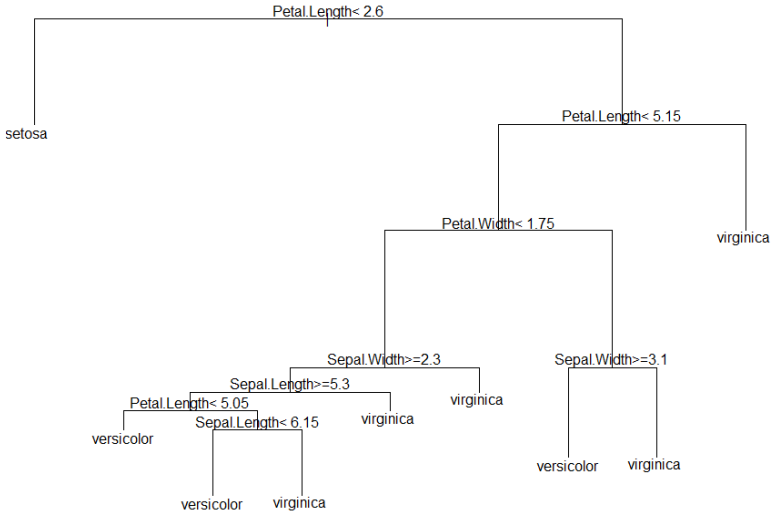
**예제 2** iris 자료에 대해 R 패키지 {adabag}의 boosting() 함수를 통해 분석을 수행한다.

```
> library(adabag)
> data(iris)
> boo.adabag <- boosting(Species~., data=iris, boos=TRUE, mfinal=10)
> boo.adabag$importance
 Petal.Length Petal.Width Sepal.Length Sepal.Width
    61.458478   19.826149     6.216103   12.499270
```

# 10.3 부스팅

- R 패키지 {adabag}의 boosting() 함수는 부스팅을 이용하여 분류를 수행한다. plot() 함수를 통해 분류 결과를 트리 형태로 나타낼 수 있다.

```
> plot(boo.adabag$trees[[10]])
> text(boo.adabag$trees[[10]])
```



# 10.3 부스팅

- predict() 함수를 통해 새로운 자료에 대한 분류(예측)를 수행 할 수 있다. 여기서는 모형 구축에 사용된 자료를 재사용하여 분류를 수행하였다. 그 결과 setosa, versicolor, virginica 모두 정확히 분류되었음을 알 수 있다.

```
> pred <- predict(boo.adabag, newdata=iris)
> tb <- table(pred$class, iris[,5])
> tb
            setosa versicolor virginica
 setosa         50          0         0
 versicolor      0         50         0
 virginica       0          0        50
```

# 10.3 부스팅

- 위의 결과로부터 오분류율을 계산하면 다음과 같이 그 값이 0임을 알 수 있다.

```
> error.rpart <- 1-(sum(diag(tb))/sum(tb))
> error.rpart
 [1] 0
```

- 다음의 [예제 3]은 R 패키지 {nnet}의 nnet() 함수를 이용하여 신경망 모형을 적합한다.

**예제 3** iris 자료 중 setosa를 제외한 versicolor와 virginica 자료만으로 분석을 수행한다.

```
> library(ada)
> data(iris)
> iris[iris$Species!="setosa", ] -> iris    # setosa 50개 자료 제외
> n <- dim(iris)[1]
```

# 10.3 부스팅

- 총 100개의 자료를 60개의 훈련용 자료(training data)와 검증용 자료(testing data)로 나누었다.

```
> trind <- sample(1:n, floor(.6*n), FALSE)
> teind <- setdiff(1:n, trind)    # set difference(차집합)
> iris[,5] <- as.factor((levels(iris[, 5])[2:3])[as.numeric(iris[,
5])-1])    # 밑줄 부분: 0,1,2가 차례대로 50개(총 150개)
```

- 훈련용 데이터를 이용하여 부스팅 방법으로 모형을 구축하고, 검증용 자료에 대해 분류(예측)를 실시하였다. 그 결과 검증용 자료에 대한 정분류율이 100%로 나타났다.

```
> gdis<-ada(Species~., data=iris[trind,], iter=20, nu=1, type="discrete")
> # nu=1(디폴트)은 부스팅을 위한 축소(shrinkage) 모수
> # type="discrete"(디폴트)은 부스팅 알고리즘 지정. "real", "gentle" 부스팅
> gdis<-addtest(gdis, iris[teind, -5], iris[teind, 5])
```

# 10.3 부스팅

```
> gdis
Call:
 ada(Species ~ ., data = iris[trind, ], iter = 20, nu = 1, type =
"discrete")
Loss: exponential Method: discrete Iteration: 20

 Final Confusion Matrix for Data:
           Final Prediction
True value  versicolor virginica
 versicolor         31         0
 virginica           0        29

Train Error: 0
Out-Of-Bag Error: 0 iteration= 20

Additional Estimates of number of iterations:
train.err1 train.kap1 test.errs2 test.kaps2
         9          9          1          1
```

# 10.3 부스팅

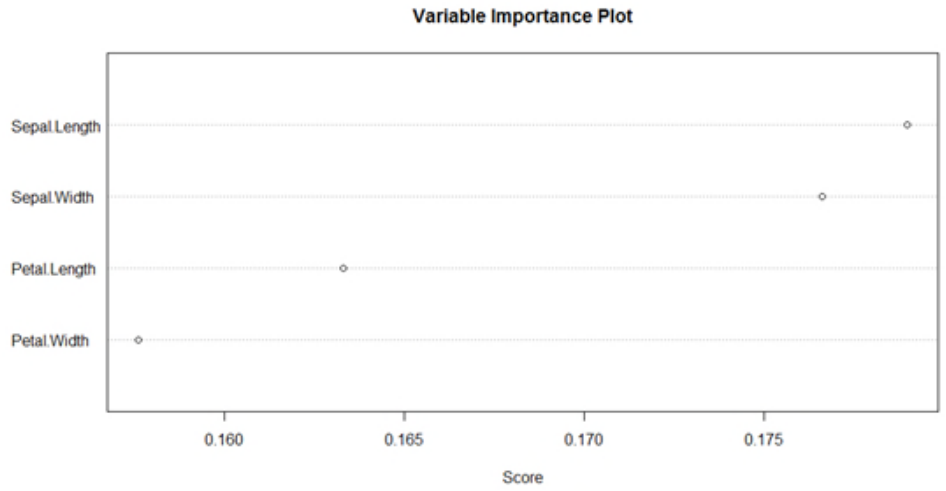
- plot(), varplot(), pairs() 함수를 이용하여 부스팅 결과를 시각화 한 결과는 다음과 같다. 아래의 plot() 함수는 오차와 일치도를 나타내는 카파(kappa) 계수를 그려준다. 두 개의 TRUE 옵션은 훈련용, 검증용 자료 모두에 대해 그림을 그려준다.

```
> plot(gdis, TRUE, TRUE)
```

# 10.3 부스팅

- varplot() 함수는 변수의 중요도(importance)를 나타내는 그림을 제공한다. Sepal.Length 변수가 분류에 가장 중요한 변수로 사용되었음을 보여준다.

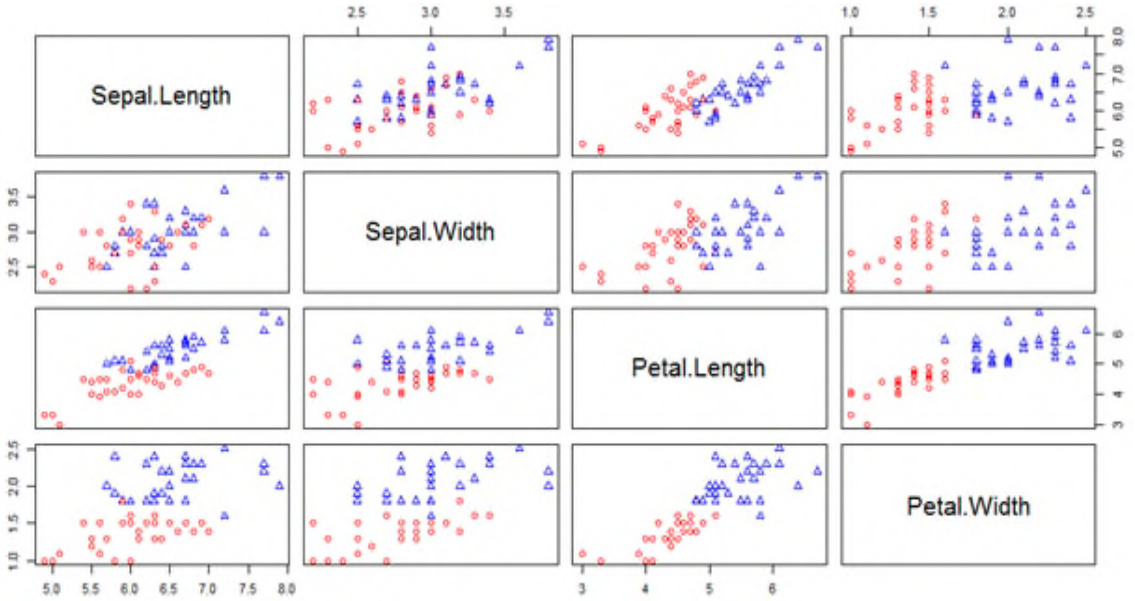
```
> varplot(gdis)
```



# 10.3 부스팅

- pairs() 함수는 두 예측변수의 조합별로 분류된 결과를 그려준다. maxvar= 옵션을 통해 변수의 수(중요도가 높은 상위 변수의 수)를 지정할 수 있다.

```
> pairs(gdis, iris[trind,-5], maxvar=4)
```



# 10.4 랜덤포리스트

- 랜덤포리스트(random forest)는 배깅에 랜덤 과정을 추가한 방법이다.
- 원 자료로부터 붓스트랩 샘플을 추출하고, 각 붓스트랩 샘플에 대해 트리를 형성해 나가는 과정은 배깅과 유사하나, 각 노드마다 모든 예측변수 안에서 최적의 분할(split)을 선택하는 방법 대신 예측변수들을 임의로 추출하고, 추출된 변수 내에서 최적의 분할을 만들어 나가는 방법을 사용한다.
- 새로운 자료에 대한 예측은 분류(classification)의 경우는 다수결(majority votes)로, 회귀(regression)의 경우에는 평균을 취하는 방법을 사용하며, 이는 다른 앙상블 모형과 동일하다.

# 10.4 랜덤포리스트

**예제 4** 의사결정나무분석에 사용되었던 ploidy자료에 대해 randomForest() 함수를 이용하여 분석을 수행한다(n=146인 전립선 암 환자 자료) .

```
> library(randomForest)
> rf <- randomForest(ploidy ~ ., data=trainData, ntree=100,
proximity=TRUE)    # 반응변수는 상동염색체수(예측변수는 7개임)
> # proximity=TRUE는 개체들 간의 근접도 행렬을 제공: 동일한 최종노드에
포함되는 빈도에 기초함
> table(predict(rf), trainData$ploidy)
           diploid tetraploid aneuploid
diploid         47          0         3
tetraploid       0         51         0
aneuploid        1          0         0
```

```
> print(rf)

                                (…)
```

# 10.4 랜덤포리스트

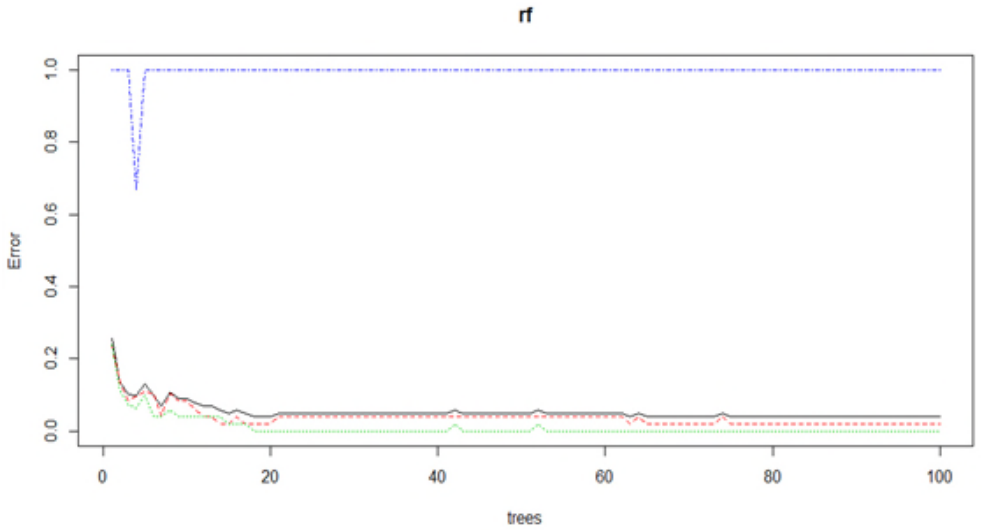
```
Call:
 randomForest(formula = ploidy ~ ., data = trainData, ntree = 100,
proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 100
 No. of variables tried at each split: 2

          OOB estimate of error rate: 3.92%
Confusion matrix:
           diploid tetraploid aneuploid class.error
diploid         47          0         1  0.02083333
tetraploid       0         51         0  0.00000000
aneuploidy       3          0         0  1.00000000
```

- 위 결과는 정오분류표(confusion matrix)와 함께, 오류율에 대한 OOB(out-of-bag) 추정치를 제공한다. 랜덤포리스트에서는 별도의 검증용 데이터를 사용하지 않더라도 붓스트랩 샘플과정에서 제외된(out-of-bag) 자료를 사용하여 검증을 실시할 수 있다.

# 10.4 랜덤포리스트

- 아래의 plot() 함수는 트리 수에 따른 종속변수의 범주별 오분류율 나타낸다. 검은색은 전체 오분류율을 나타낸다. 오분류율이 1로 나타난 범주는 aneuploid 범주로 개체수가 매우 작은($n = 3$) 범주에서 발생된 결과이다.

```
> plot(rf)
```



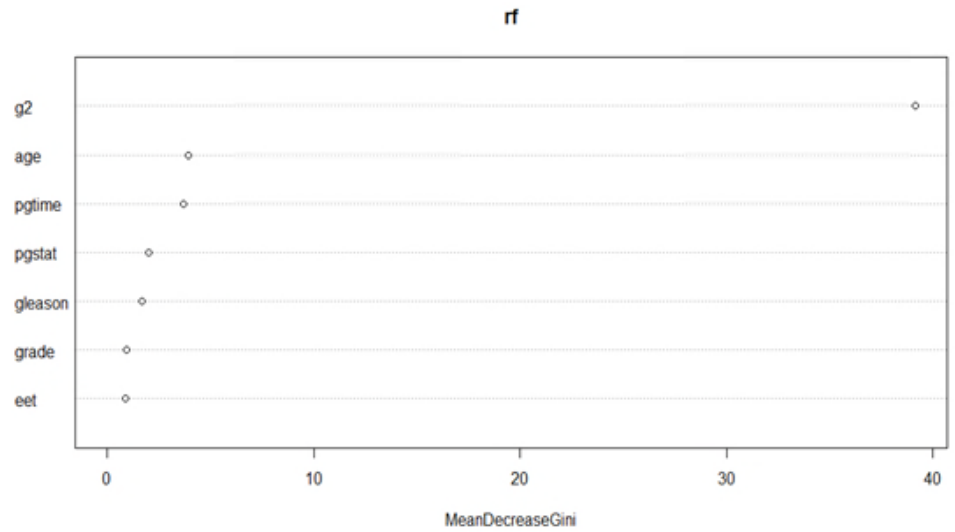
# 10.4 랜덤포리스트

- importance()와 varImpPlot()로 변수의 중요성을 알 수 있다.

```
> importance(rf)
         MeanDecreaseGini
pgtime         3.6620387
pgstat         1.9713041
age            3.9007535
eet            0.8592785
g2            39.1422888
grade          0.9361285
gleason        1.6824158
```

```
> varImpPlot(rf)
```

# 10.4 랜덤포리스트




- 위의 그림은 각 변수의 중요도를 나타내는 그림으로, 해당 변수로부터 분할이 일어날 때 불순도(impurity)의 감소가 얼마나 일어나는지를 나타내는 값이다(불순도의 감소가 클수록 순수도가 증가함). 지니 지수(Gini index)는 노드의 불순도를 나타내는 값이다.
- 회귀의 경우에는 잔차제곱합(residual sum of square)을 통해 측정된다.

# 10.4 랜덤포리스트

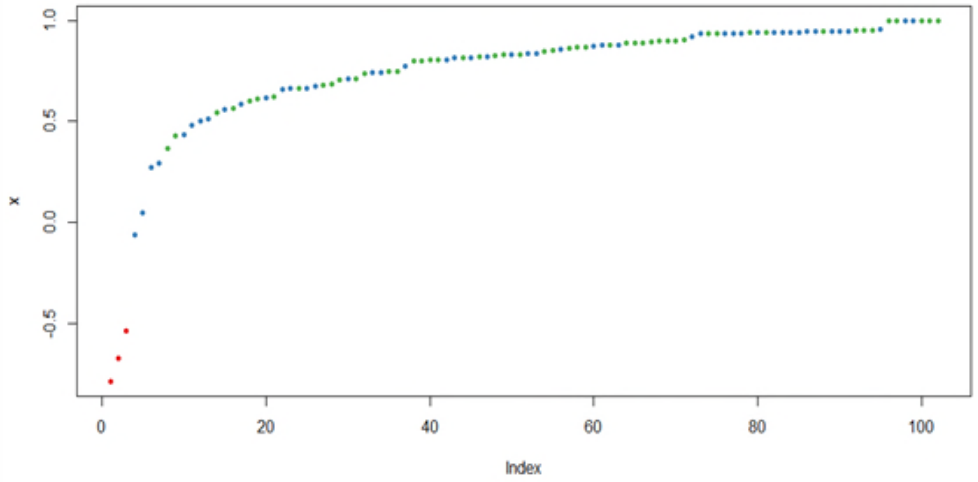
- 다음은 테스트 자료에 대해 예측을 수행한 결과이다.

```
> rf.pred <- predict(rf, newdata=testData)
> table(rf.pred, testData$ploidy)
 rf.pred    diploid tetraploid aneuploidy
 diploid         17          0          1
 tetraploid       0         13          1
 aneuploid        0          0          0
```

# 10.4 랜덤포리스트

- 아래의 그림은 훈련용 자료 값(총 102개: 전체의 70%)의 마진을 나타낸다. 마진(margin)은 랜덤포리스트의 분류기(classifiers) 가운데 정분류를 수행한 비율에서 다른 클래스로 분류한 비율의 최대치를 뺀 값을 나타낸다.
- 즉, 양(positive)의 마진은 정확한 분류를 의미하며, 음(negative)은 그 반대이다.

```
> plot(margin(rf))
```



# 10.4 랜덤포리스트

- 랜덤포리스트는 다음과 같이 R 패키지 {party}의 cforest() 함수를 이용할 수도 있다.

```
> set.seed(1234)
> cf <- cforest(ploidy ~ ., data=trainData)
> cf.pred <- predict(cf, newdata=testData, OOB=TRUE,
                     type="response")
```

# 10.5* {caret}를 이용한 랜덤포리스트

- 대부분이 결측(NAs)인 열들을 제거한다. 이 변수들이 모형에서 유용할 수도 있으나, 편의상 이들을 제거 후 데이터프레임을 구축한 뒤, 먼저 분석을 실시한 후 정확도를 살펴보기로 한다.

```
> mostly_data <- apply(!is.na(training), 2, sum)>19621
> training <- training[,mostly_data]
> test <- test[,mostly_data]
> dim(training)
 [1] 19622 54
```

- 모형수립의 수행 속도를 높이기 위해 (편의상) 훈련용 자료를 더 작게 나눈다. 원 자료의 30%를 임의로 추출하여 새로운 훈련용 자료(training1)를 만든다.

```
> InTrain<-createDataPartition(y=training$classe, p=0.3, list=FALSE)
> training1<-training[InTrain,]
```

# 10.5* {caret}를 이용한 랜덤포리스트

- R 패키지 {caret}을 이용하여 Random Forest를 수행하되, 5-fold 교차타당도 방법을 적용한다.

```
> rf_model<-train(classe~., data=training1, method="rf",
        trControl=trainControl(method="cv", number=5),
        prox=TRUE, allowParallel=TRUE)
```

```
> print(rf_model)
 Random Forest
 5889 samples
 53 predictor
 5 classes: 'A', 'B', 'C', 'D', 'E'

 No pre-processing
 Resampling: Cross-Validated (5 fold)
                                    (…)
```

# 10.5* {caret}를 이용한 랜덤포리스트

```
Summary of sample sizes: 4711, 4711, 4712, 4710, 4712

Resampling results across tuning parameters:

  mtry  Accuracy  Kappa     Accuracy SD Kappa SD
   2    0.9836964 0.9793699 0.003965249 0.005019661
  27    0.9891302 0.9862454 0.004710279 0.005962570
  53    0.9862437 0.9825978 0.004556468 0.005762613

 Accuracy was used to select the optimal model using the largest
value.
 The final value used for the model was mtry = 27.
```

```
> print(rf_model$finalModel)
Call:
 randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE,
allowParallel = TRUE)
                                    (…)
```

# 10.5* {caret}를 이용한 랜덤포리스트

```
              Type of random forest: classification
                    Number of trees: 500
No. of variables tried at each split: 27

       OOB estimate of error rate: 0.85%
Confusion matrix:
     A    B    C   D    E class.error
A 1674    0    0   0    0 0.000000000
B   13 1120    6   1    0 0.017543860
C    0    7 1018   2    0 0.008763389
D    0    0   13 950    2 0.015544041
E    0    1    0   5 1077 0.005540166
```

- 위의 정오분류표로부터 정확도는 99.15%이며, 이는 매우 놀라운 결과이다.

# 10.5* {caret}를 이용한 랜덤포리스트

참 고 OOB(Out-of-Bag) 오차추정

랜덤포리스트에서는 검증 자료의 오차에 대한 불편추정치를 얻기 위해 교차타당법을 사용하거나 별도의 검증용 자료를 만들 필요가 없다. 이 방법은 오차에 대한 추정을 내부 알고리즘에서 자동으로 제공해 준다. 그 방법은 다음과 같다.

각 트리는 원 자료로부터 서로 다른 붓스트랩 표본을 사용하여 구축된다. 자료의 약 1/3은 붓스트랩 표본에서 제외되고, k-번째 트리의 형성에 사용되지 않는다. k-번째 트리의 형성에 제외된 (out-of-bag, 이하 oob) 자료를 구축된 k-번째 트리에 적용하여 분류를 수행한다.

이러한 방식을 각 트리에 대해 적용하면, n 번째 자료가 oob인 모든 트리에서, n 번째 자료를 가장 많은 표를 획득한(majority votes) 클래스로 분류한다. n 개의 모든 자료에 대해 오분류된 비율의 평균이 oob 오차 추정치이다. 이 값은 많은 검정에서 불편성을 만족하는 것으로 증명되었다.