나무모형

이재용,임요한

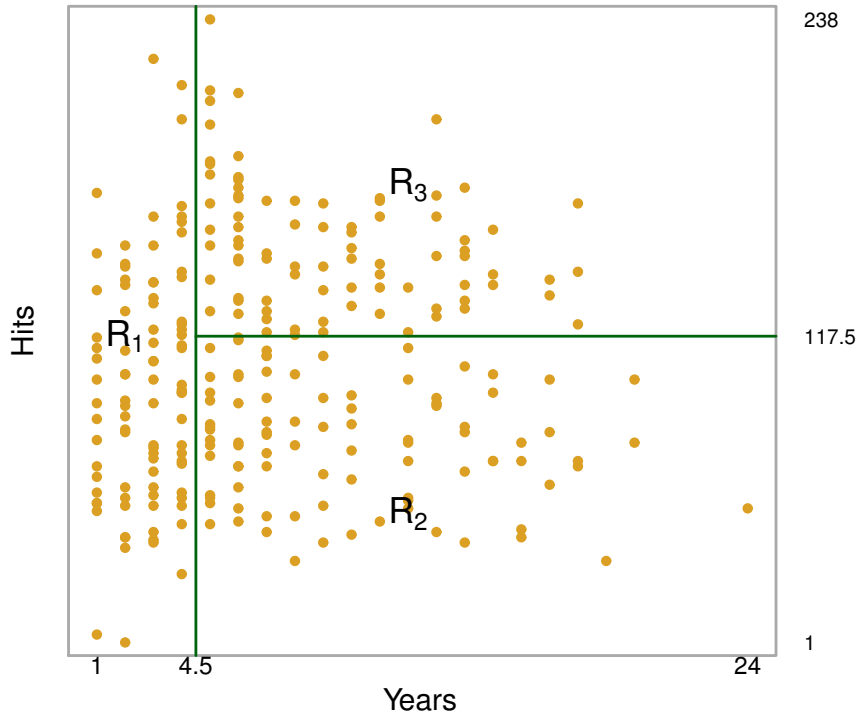
서 울 대 학 교
통 계 학 과

2017년 8월

노트. 이 장에서 다룰 내용 I

1. 의사결정나무의 소개
2. 의사결정나무의 적합
3. 분류나무
4. 배깅
5. 랜덤숲
6. 부스팅

의사결정나무 I



함수 추정의 문제
선형 분류자와 비교

의사결정나무 II

노트.

1. 첫 두 장은 의사결정나무가 어떤 것인지 보여주기 위해 넣었다.

Hitters 자료

1. ISLR 패키지에 있는 자료
2. 1986년 322명의 메이저리그 선수들에 대한 관측치. 20개의 변수
3. log(*Salary*)를 Hits(1986년 안타의 개수)와 Years(메이저리그에서 뛴 햇수) 등을 이용해 예측하는 것이 목적.

노트. 그림에 대한 설명

1. 전체가 R_1, R_2, R_3의 영역으로 나뉘고, 각 분할 영역에서 로그-연봉의 평균은 $5.11, 6.00, 6.14$이다.
2. 연봉을 결정하는 가장 중요한 변수는 Years이고 4.5년미만과 이상으로 나뉜다. 4.5년 이상인 선수들 중에는 안타의 갯수가 중요한 요소이다.
3. 해석이 쉽다.

의사결정나무 III

개요

1. 예측 변수의 공간을 분할하여 분할된 부분에서 반응변수의 값을 예측하는 회귀분석 혹은 분류 방법.
2. 의사결정나무(decision tree)방법이라고도 한다.
3. 예측변수의 공간을 분할하기 때문에 해석이 쉽다.
4. 예측성능은 보통 좋지 않다.
5. 배깅(bagging), 랜덤숲(random forest), 부스팅(boosting)과 같이 다수의 나무를 합치면 종종 매우 좋은 결과를 나타낸다.
6. 반응변수 Y가 범주형인가, 연속형인가에 따라 분류나무, 회귀나무로 나뉜다.

용어

1. 종점마디 혹은 종점노드(terminal nodes, or leaves) : R_1, R_2, R_3를 말한다.
2. 내부마디 혹은 내부노드(internal nodes) : 예측변수의 공간이 분할된 곳으로 종점노드가 아닌 노드를 말한다.
3. 가지(branch) : 노드에서 분리되는 나무의 일부분을 가지라 한다.

의사결정나무의 적합 I

의사결정나무의 적합의 요약

1. 예측변수 공간의 분할. 예측변수 $X_1, \ldots, X_p$가 취할 수 있는 값들의 공간을 $R_1, \ldots, R_J$로 분할한다.
2. 예측값 R_j에 속하는 예측변수의 값에는 동일한 $\hat{y}$의 값으로 예측한다. 예. R_j에 속한 관측치들의 평균을 쓴다.
3. 의사결정나무의 적합은 나무기르기(splitting)와 가지치기(prunning) 두가지 단계로 이루어진다.

의사결정나무의 적합 II

나무기르기 : 반복이진분할(recursive binary splitting)

1. $\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$ 를 최소로하는 j와 s를 찾고 예측변수의 공간을 $R_1(j,s)$와 $R_2(j,s)$로 나눈다.

$$\begin{aligned} R_1(j,s) &= \{X : X_j < s\} \\ R_2(j,s) &= \{X : X_j \geq s\} \\ \hat{y}_{R_1} &= R_1 \text{ 에 속한 } y \text{ 들의 평균} \\ \hat{y}_{R_2} &= R_2 \text{ 에 속한 } y \text{ 들의 평균.} \end{aligned}$$

2. 동일한 방식으로 R_1이나 R_2를 분할하여 R_1, R_2, R_3라고 한다. 즉, R_1내에서 분할하여 가장 RSS를 최소화하는 분할과 R_2내에서 분할하여 RSS를 최소화하는 분할을 비교하여 RSS를 더 작게 하는 분할을 하여 분할된 공간을 R_1, R_2, R_3라고 한다.
3. $R_1, \ldots, R_j$가 주어져 있을 때, 각 R_i를 두 개로 분할하는 방법들을 비교하여 RSS를 최소화하는 분할 방법을 찾고 새로 분할된 영역들을 $R_1, \ldots, R_{j+1}$이라 한다.
4. 분할을 계속하여 정해진 기준이 만족될 때까지 분할한다. 예. 모든 R_i가 5개 이하의 관측치를 포함한다.

의사결정나무의 적합 III

가지치기 : 비용 복잡도 가지치기(cost complexity pruning)

모든 부분나무를 교차검증으로 비교하는 것은 계산량이 많으므로 다음과 같은 방법을 쓴다.

1. 주어진 $\alpha \geq 0$에 대해서,

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

를 최소화하는 T_α를 구한다.

의사결정나무의 적합 IV

노트

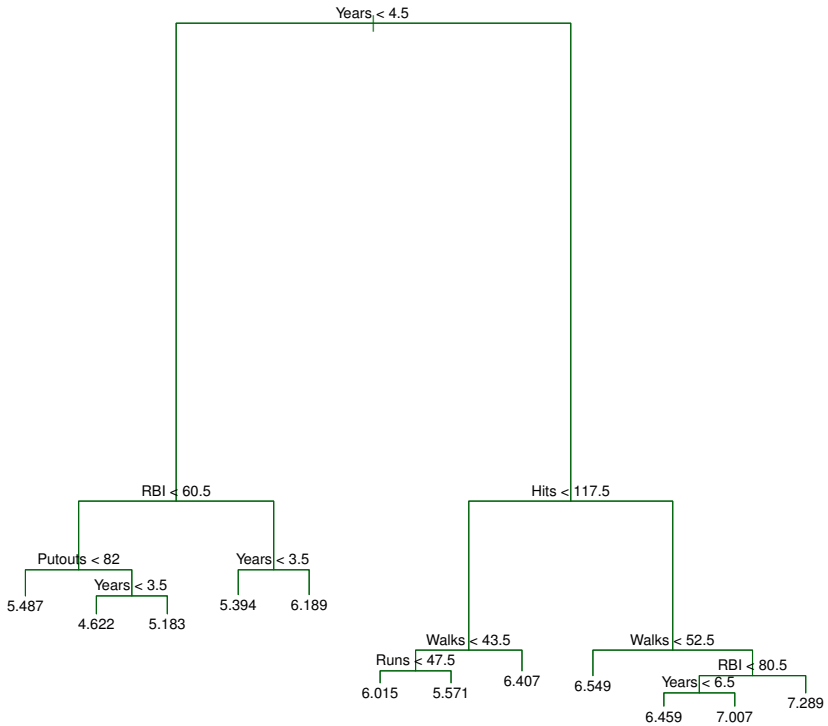
1.1 이렇게 구해지는 T_α들은 겹겹의 나무열(nested tree sequence)이 되는 것 같다.

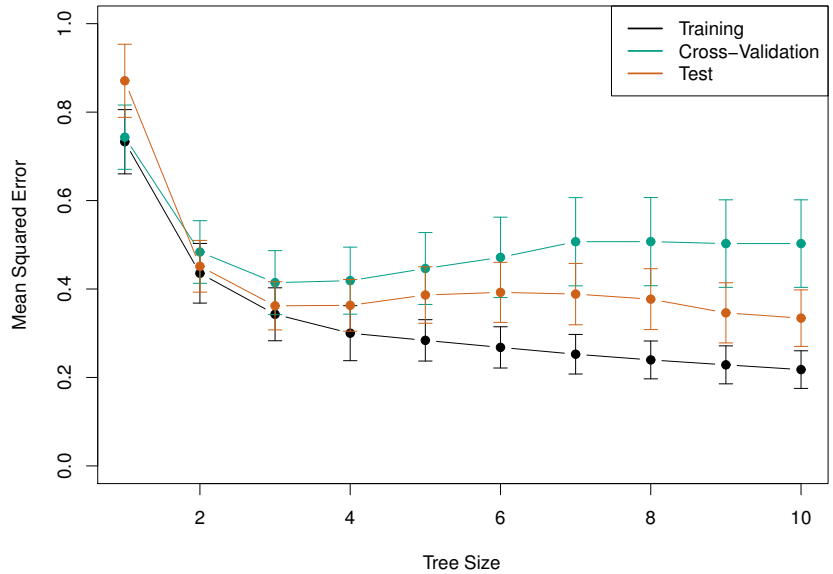
1.2 α가 주어졌을 때, T_α를 구하는 방법은 가장 큰 나무로부터 가지를 하나씩 쳐가면서 $\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$를 비교해보면 된다.

2. α의 추정. K겹 교차검증을 통해 $\hat{\alpha}$를 구한다. $T_{\hat{\alpha}}$이 추정량이 된다.

의사결정나무의 적합 V

의사결정나무의 적합





노트.

1. 왼쪽의 그림은 가지치기 전의 다 큰 나무이다.
2. 교차검증의 결과이다. $|T| = 3$인 나무가 선택되었다.

분류나무 I

분류나무의 적합 방식은 회귀나무와 동일하다. 다만, 잔차제곱합 대신 아래의 순수성 측도 중 하나를 사용한다.

순수성 측도

1. 오분류율

$$E = 1 - \max_k \hat{p}_{mk}$$

2. 기니 지수(Gini index)

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

3. 엔트로피(cross-entropy)

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

예측치

$\hat{y}_{R_m} = R_m$에서 가장 빈도가 높은 범주

분류나무 II

노트.

1. 기니지수. 각 범주마다 하나의 이항변수를 정의한다. 즉, 범주 k에 해당하는 이항변수 X_k는 범주 k일 때는 $X_k = 1$ 그렇지 않을 때는 $X_k = 0$으로 정의한다. 기니지수는

$$\sum_k \mathbb{V}ar(X_k)$$

   이다.
2. $D \geq 0$이다.
3. G나 D 모두 $p_{mk} \approx 0$ 이나 1이면 0 근처의 값을 갖는다.
4. 위의 값들을 최소로 하는 분할을 찾는다. 변동을 최소로 하는 분할을 찾는다.

분류나무 III

장점

1. 사람들에게 설명하기 쉽다.
2. 의사결정나무는 사람들의 의사 결정과 매우 흡사하다.
3. 나무는 그림으로 표현될 수 있고, 비전문가도 해석을 쉽게 할 수 있다.
4. 범주형 예측변수도 쉽게 이용이 가능하다. 가변수가 필요없다.

단점

1. 예측력이 떨어진다.
2. 분산이 커서 추정량이 안정적이지 않다.

의사결정나무 방법의 분산

의사결정나무 방법은 분산이 매우 크다. 주어진 자료를 반으로 나누어 의사결정나무를 적합하면 두 개의 매우 다른 나무가 나온다. 회귀모형은 그렇지 않다.

분류나무 R 코드 I

```
library(tree)
library(ISLR)
attach(Carseats)
High=ifelse(Sales<=8,"No","Yes")
Carseats=data.frame(Carseats,High)
```

노트.

1. tree 패키지를 이용한다.
2. Carseats 자료는 모의 자료로서, 크기는 400개 가게의 carseat 판매 자료이다. 11개의 변수가 있다. Sales는 각 가게에서 팔린 개수로 단위가 천이다.
3. Sales를 이용해서 이산형 반응변수 High를 작성했다.

분류나무 R 코드 II

```
tree.carseats=tree(High~.-Sales,Carseats)
summary(tree.carseats)

##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

tree(formula, data, weights, subset, method = "recursive.partition", split = c("deviance", "gini"))

training error

Sales를 제거하고 나무모형을 적합했다. 가지치기 이전의 나무이다.
이탈도(deviance)는

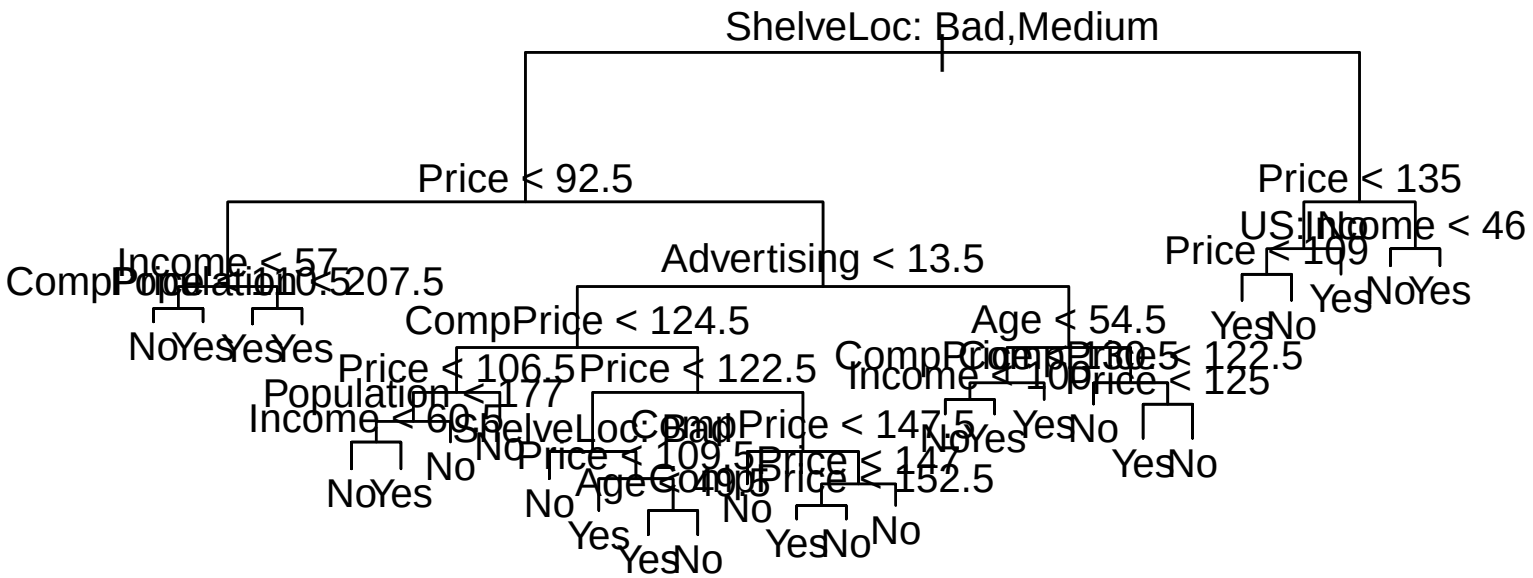
$$-2\sum\sum n_{mk}\log\hat{p}_{mk}$$

이다. 여기서, m의 마지막 노드의 인덱스이고, k는 카테고리 인덱스이다.

분류나무 R 코드 III

```
plot(tree.carseats)
text(tree.carseats,pretty=0)
```

"pch=7" 옵션을 사용 글자크기 조절



plot은 나무 그림만 그린다. text는 그림에 변수들 이름을 써넣는다.

분류나무 R 코드 IV

```
tree.carseats
```

나무 자체를 출력한다.

노트.

결과는 너무 길어서 생략했다.

분류나무 R 코드 V

```
set.seed(2)
train=sample(1:nrow(Carseats), 200)
Carseats.test=Carseats[-train,]
High.test=High[-train]
tree.carseats=tree(High~.-Sales,Carseats,subset=train)
tree.pred=predict(tree.carseats,Carseats.test,type="class")
table(tree.pred,High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  86  27
##       Yes 30  57
```

```
(86+57)/200
```

```
## [1] 0.715
```

testing error=0.285

예측오차를 계산한다.

분류나무 R 코드 VI

```
set.seed(3)
cv.carseats=cv.tree(tree.carseats,FUN=prune.misclass)
names(cv.carseats)

## [1] "size"   "dev"    "k"      "method"

cv.carseats

## $size
## [1] 19 17 14 13  9  7  3  2  1
##
## $dev
## [1] 55 55 53 52 50 56 69 65 80
##
## $k
## [1]       -Inf  0.0000000  0.6666667  1.0000000  1.7500000  2.0000000
## [7]  4.2500000  5.0000000 23.0000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

분류나무 R 코드 VII

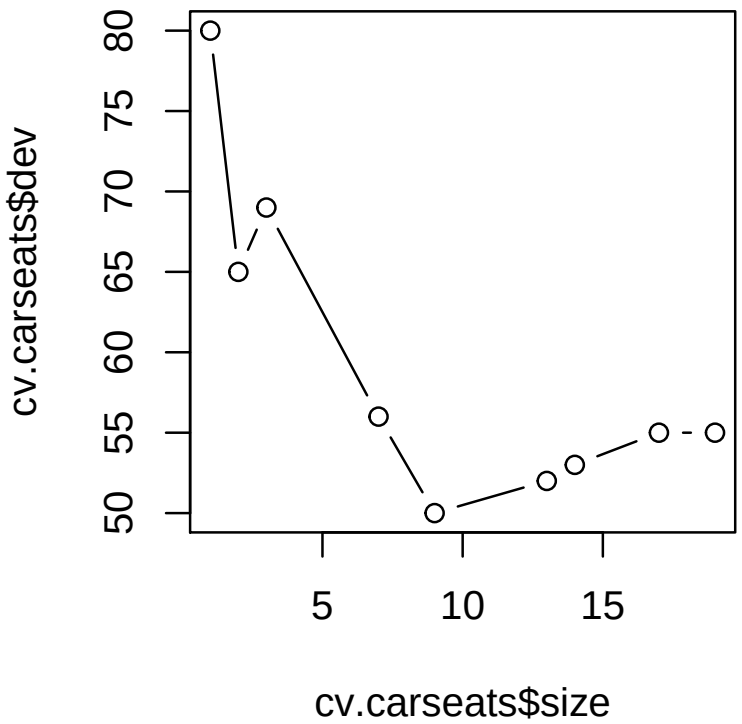
노트.

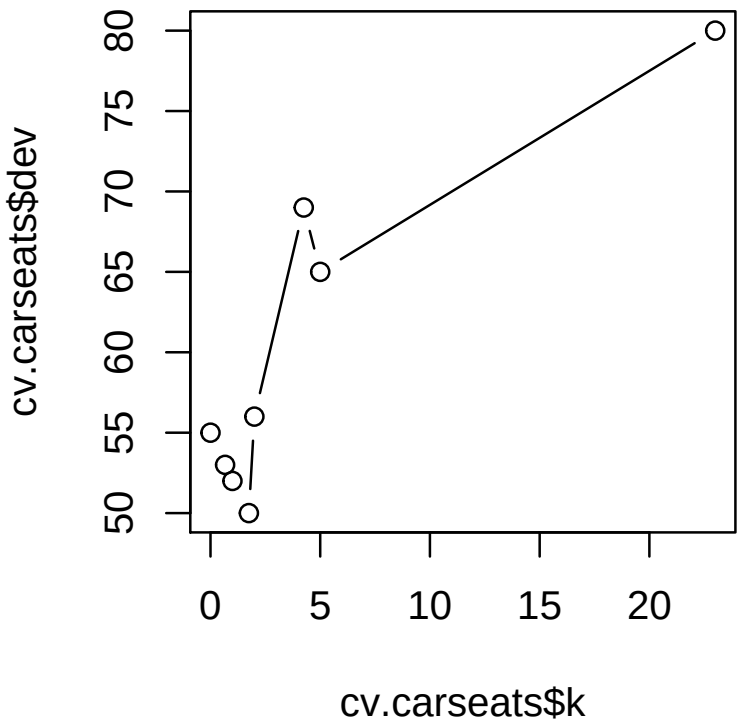
교차검증오차를 계산하는 코드이다.

1. cv.tree 함수에서 FUN=prune.misclass는 교차검증을 할 때, 오분류율을 기준으로 사용한다는 뜻이다. 이 옵션의 디폴트는 deviance이다.
2. cv.carseats의 size는 나무의 끝마디의 개수, k는 비용복잡도 식의 α에 대응하는 값이다. size는 작아지고 k는 커진다.
3. cv.carseats의 구성요서인 dev는 교차검증오차로 가장 작은 것이 좋은 것이다.

분류나무 R 코드 VIII

```
par(mfrow=c(1,2))
plot(cv.carseats$size,cv.carseats$dev,type="b")
plot(cv.carseats$k,cv.carseats$dev,type="b")
```





분류나무 R 코드 IX

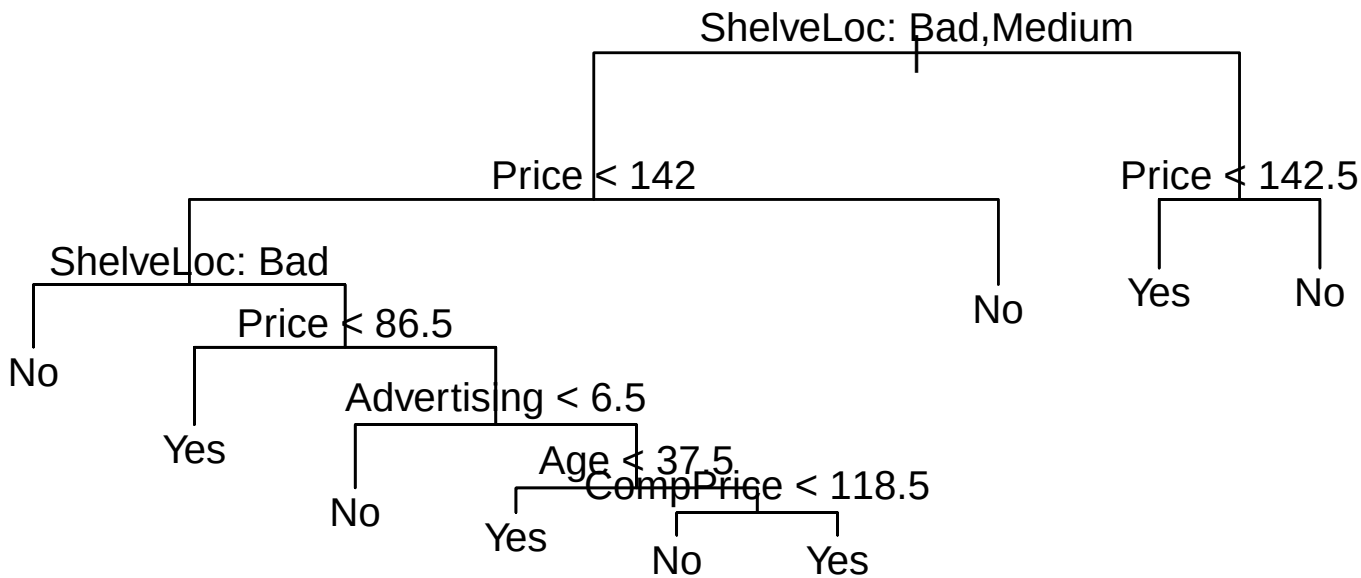
```
par(mfrow=c(1,1))
```

교차검증오차를 나무의 크기와 조율파라미터에 대해 그렸다.

분류나무 R 코드 X

```
prune.carseats=prune.misclass(tree.carseats,best=9)
plot(prune.carseats)
text(prune.carseats,pretty=0)
```

minlength=0L



가지치기하여 끝마디가 9개인 나무를 얻는다.

분류나무 R 코드 XI

```
tree.pred=predict(prune.carseats,Carseats.test,type="class")
table(tree.pred,High.test)

##          High.test
## tree.pred No Yes
##       No  94  24
##       Yes 22  60
```

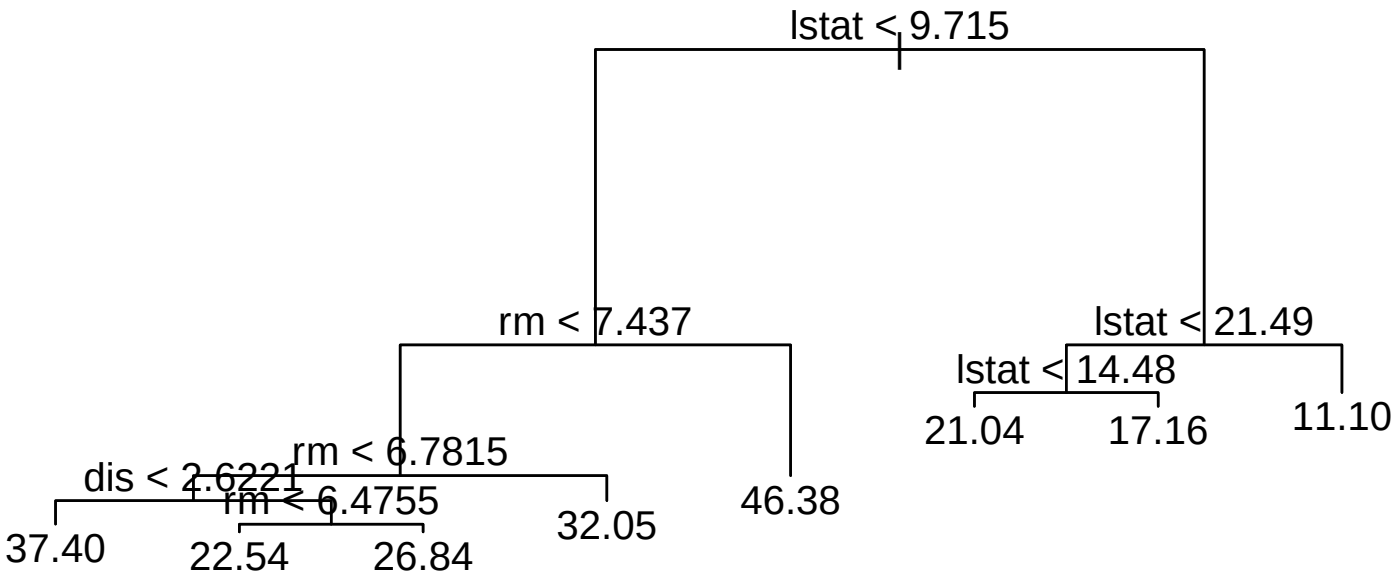
오차율을 계산하였다.

회귀나무 R 코드 I

MASS 패키지에 있는 Boston 자료를 이용한다. 보스톤 내의 506개 동네에 관한 자료이다. 14개의 변수가 있다. medv는 자가 소유 주택 가격의 중앙값으로 단위는 천불이다.

```
library(MASS)
set.seed(1)
train = sample(1:nrow(Boston), nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)

##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "lstat" "rm"    "dis"
## Number of terminal nodes:  8
## Residual mean deviance:  12.65 = 3099 / 245
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -14.10000  -2.04200  -0.05357   0.00000   1.96000  12.60000

plot(tree.boston)
text(tree.boston,pretty=0)
```

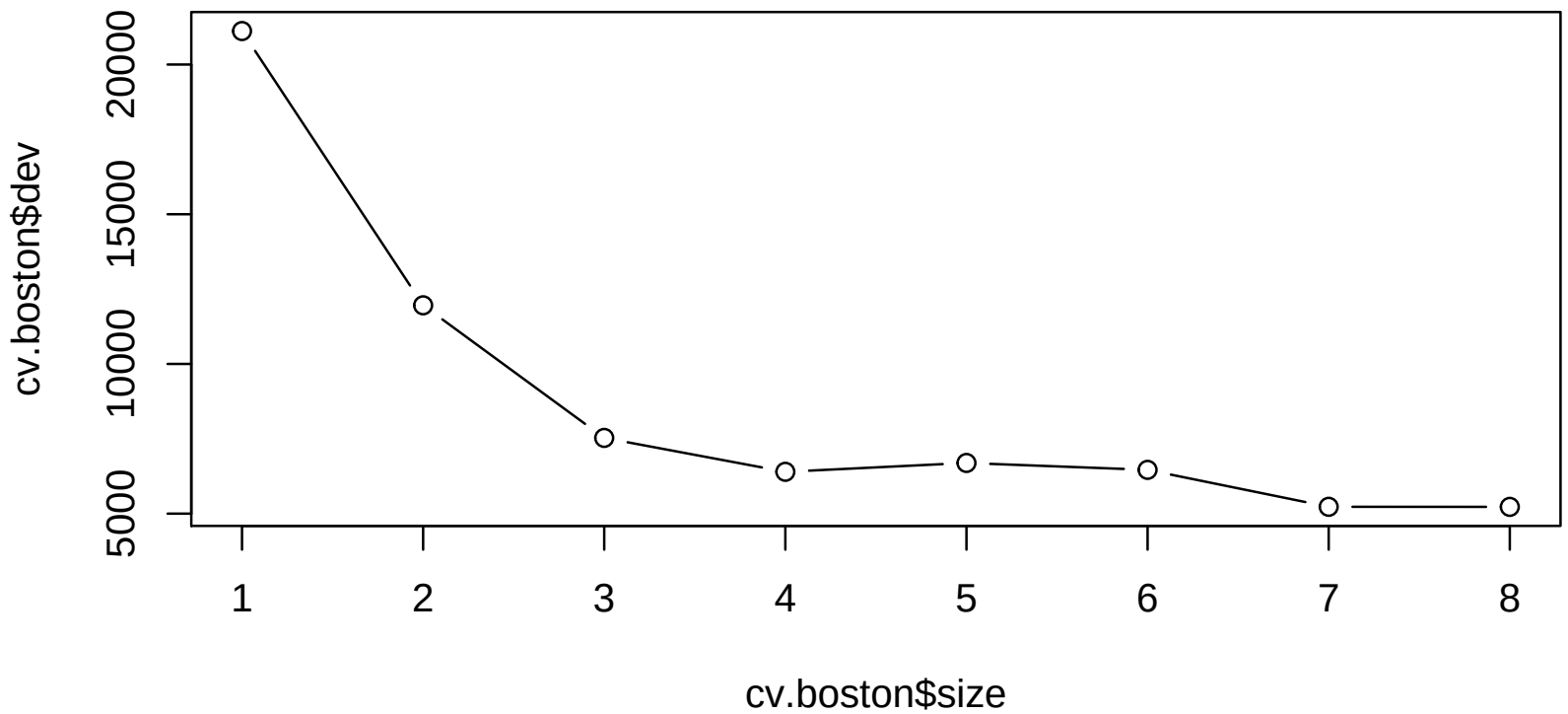
회귀나무 R 코드 II



나무를 적합하였다.

회귀나무 R 코드 III

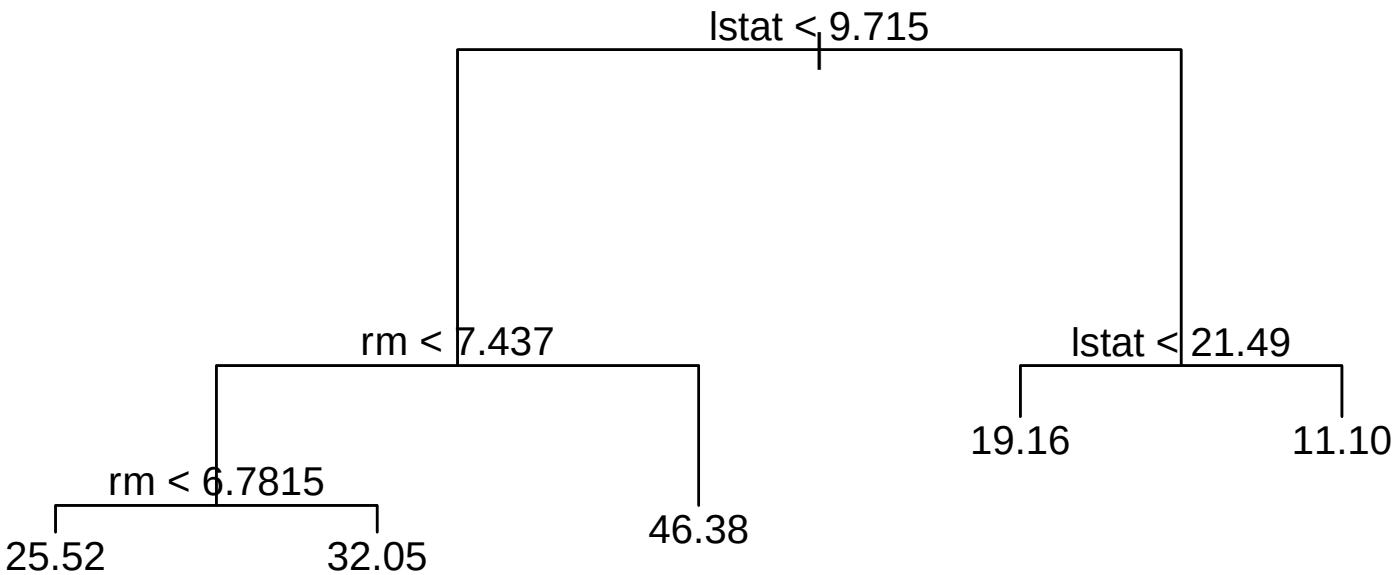
```
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type='b')
```



교차검증한 결과이다.

회귀나무 R 코드 IV

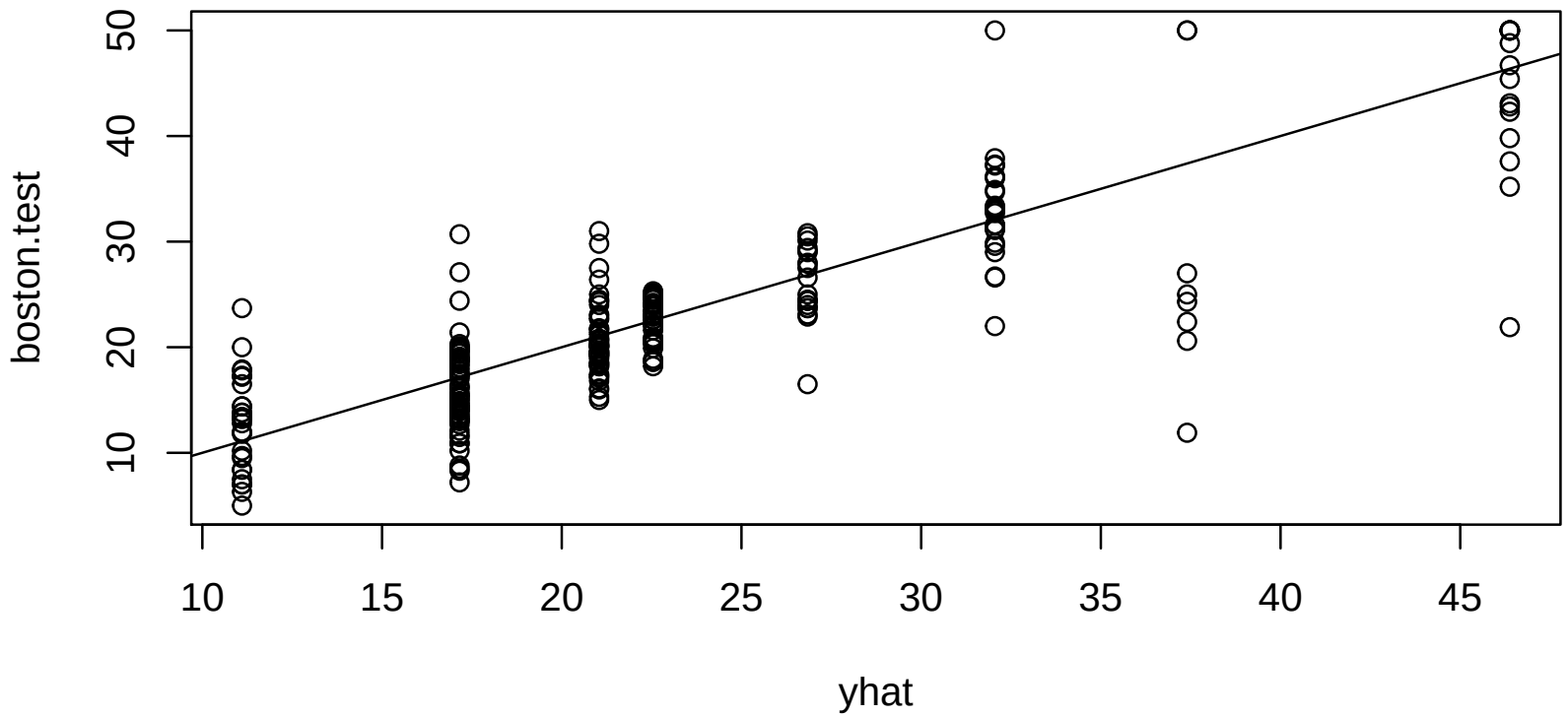
```
prune.boston=prune.tree(tree.boston,best=5)
plot(prune.boston)
text(prune.boston,pretty=0)
```



가지치기한 결과이다.

회귀나무 R 코드 V

```
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test)
abline(0,1)
```



회귀나무 R 코드 VI

```
mean((yhat-boston.test)^2)

## [1] 25.04559
```

시험자료에 예측한 결과이다.

배깅(bagging) I

붓스트랩 샘플 $X_1^*, X_2^*, \ldots, X_B^*$를 X에서 추출한다.
각 붓스트랩 샘플 X_b^*에 의사결정나무를 적합해서 $\hat{f}_b^*$라 부른다.
이를 평균 낸 것이 배깅 추정량이다. 즉,

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b^*(x)$$

이다.

y가 범주형일때는 다수결(majority vote) 방법을 쓴다. 즉, $\hat{f}_{bag}(x)$는 $\hat{f}_b^*(x), b = 1, 2, \ldots, B$ 중 가장 많은 값을 갖는 범주를 예측값으로 한다.

배깅(bagging) II

근거 : Brieman의 설명

D를 주어진 자료라 하고 X, Y를 미래의 관측치라 하자. $f(X, D)$를 D로 구축한 추정량의 X에서의 값, 즉 $\hat{Y}$이라 하자. 또한, 평균추정량을

$$f_A(X) = \mathbb{E}_D f(X, D)$$

와 같이 표기하자.
젠센의 부등식을 이용하면

$$\mathbb{E}_{(X,Y)}\mathbb{E}_D(Y - f(X, D))^2 \geq \mathbb{E}_{(X,Y)}(Y - f_A(X))^2$$

가 성립하는 것을 알 수 있다.
자료 D의 분포를 붓스트랩분포로 근사를 한다고 생각하면 f_A는 배깅추정량과 비슷하다. 따라서, 우변은 배깅추정량의 예측오차와 비슷하다.
좌변은 f라는 추정방법이 평균적으로 갖는 예측오차이다. 여기서 평균은 주어진 자료 D와 미래의 자료 (X, Y)에 대해 이루어 졌다.
이를 해석하면 배깅추정량의 예측오차는 주어진 자료로 한 번 구축한 추정량 $f(X, D)$ 보다 평균적으로 좋다고 해석할 수 있다.

배깅(bagging) III

배깅으로 얻는 것

예측오차의 차이는

$$\mathbb{E}_{(X,Y)}\mathbb{E}_D(Y - f(X,D))^2 - \mathbb{E}_{(X,Y)}(Y - f_A(X))^2 = \mathbb{E}_{(X,Y)}\mathbb{V}ar_D f(X,D)$$

라는 것이 알려져있다.
이는 배깅으로 줄일 수 있는 것은 추정량의 분산이라는 것을 알 수 있다.
분산이 작으면 배깅으로 얻는 것이 별로 없다.
이는 배깅을 할 때, 그루터기(stump)를 이용하는 이유이다.

랜덤숲(random forest) I

랜덤숲도 의사결정나무의 배깅과 매우 흡사하다. 다른점은 분할이 필요할 때마다 모든 p개의 변수를 다 고려하는 것이 아니라 $m \approx \sqrt{p}$개의 변수를 랜덤하게 골라 이 변수들에서만 분할을 한다. $m = p$이면 배깅과 같다.

직관적 근거

주어진 변수 중 한 개의 변수가 매우 중요하고 나머지 변수들은 중간 정도의 중요성을 가진다고 하자. 배깅을 하면 모든 나무들이 매우 중요한 변수부터 분할을 하게될 것이다. 그러면 나무들이 서로 비슷해지고 나무를 이용한 추정치 사이에 상관계수들이 커질 것이다. 그런데 m개의 변수만 고려한다면 매우 중요한 변수로 분할을 시작하지 않을 나무가 $\frac{p-m}{p}$의 확률이나 될 것이다. 이는 붓스트랩 나무들 사이의 상관성을 줄여서 평균의 분산을 줄여준다.

노트.

$$\mathbb{V}ar(\frac{X_1 + X_2}{2}) = \frac{1}{4}\left[\mathbb{V}ar(X_1) + \mathbb{V}ar(X_2) + \mathbb{C}ov(X_1, X_2)\right]$$

랜덤숲(random forest) II

$\mathbb{C}ov(X_1, X_2)$가 작아지면 $\mathbb{V}ar(\frac{X_1 + X_2}{2})$가 작아진다.

배깅과 랜덤숲 R 코드 I

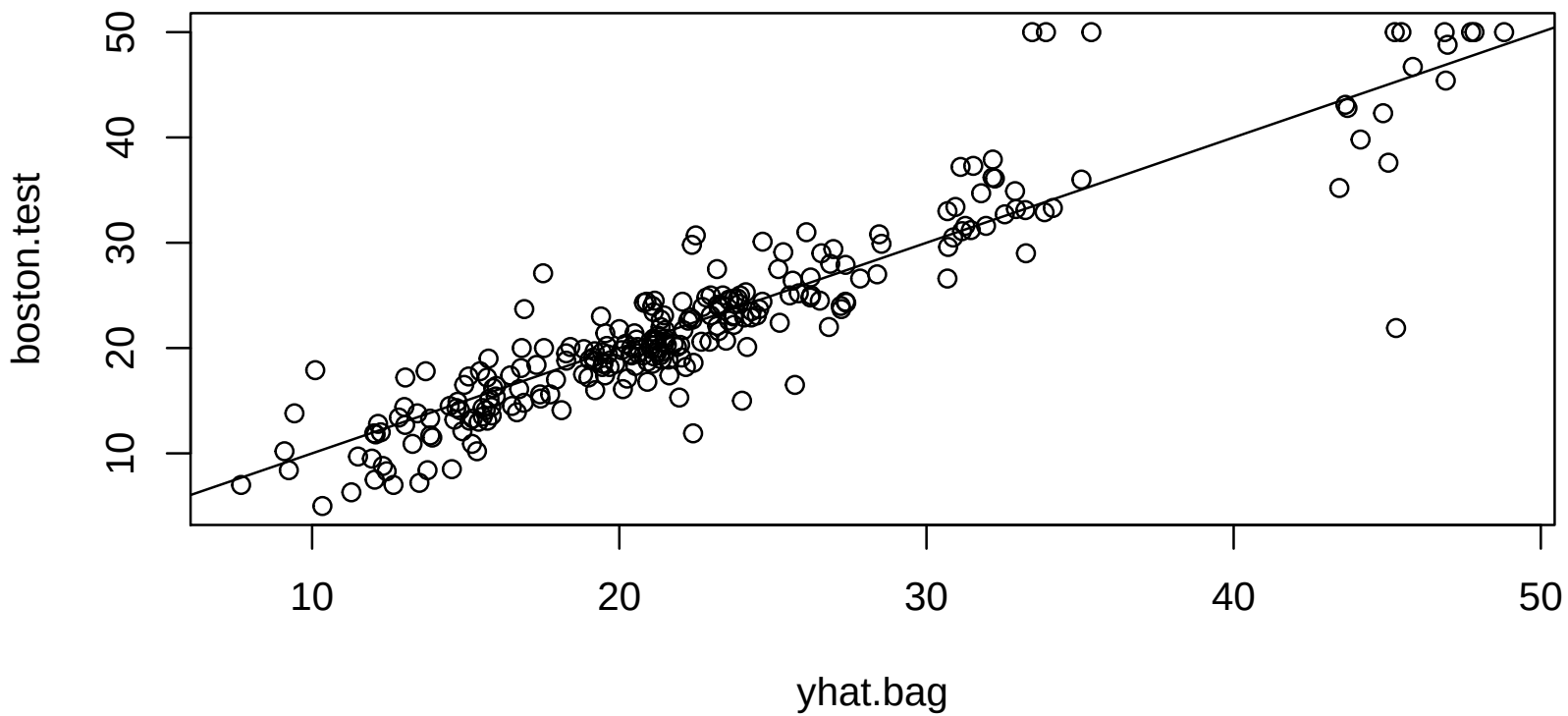
배깅은 랜덤숲의 일종이다. randomForest 패키지를 이용해서 적합할 수 있다.

```
library(randomForest)

## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.

set.seed(1)
bag.boston=randomForest(medv~.,data=Boston, subset=train, mtry=13, importance=TRUE)
bag.boston

##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance = TRUE,      subs
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 11.02509
##                     % Var explained: 86.65
```

mtry 옵션은 매 반복에서 13개의 설명변수가 고려되어야 한다는 뜻이다. 다시 말하면 배깅을 하라는 뜻이다. importance는 변수의 중요성을 평가하라는 뜻이다.

배깅과 랜덤숲 R 코드 II

```
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test)
abline(0,1)
```



배깅과 랜덤숲 R 코드 III

```
mean((yhat.bag-boston.test)^2)

## [1] 13.47349
```

시험오차를 계산하였다.

```
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
mean((yhat.bag-boston.test)^2)

## [1] 13.43068
```

ntree 옵션은 계산하는 나무의 개수를 정한 것이다.
노트.

붓스트랩 샘플의 개수 아닌가?

```
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=TRUE)
yhat.rf = predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)

## [1] 11.48022
```

배깅과 랜덤숲 R 코드 IV

나무숲을 구했다. mtry=6를 썼다. 디폴트는 회귀나무의 경우는 $p/3$, 분류나무의 경우는 $\sqrt{p}$이다.

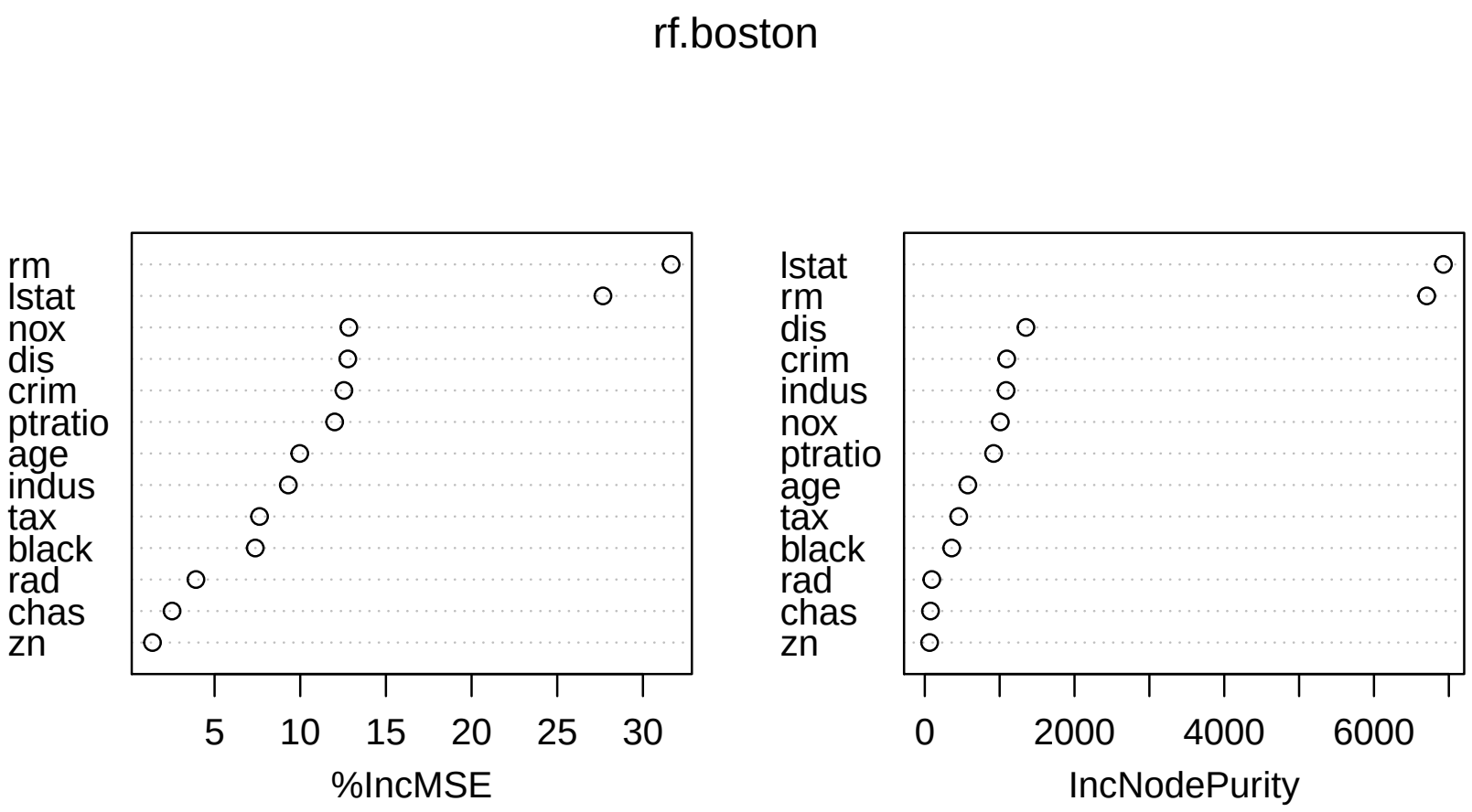
```
importance(rf.boston)
```

```
##           %IncMSE IncNodePurity
## crim    12.547772    1094.65382
## zn       1.375489      64.40060
## indus    9.304258    1086.09103
## chas     2.518766      76.36804
## nox     12.835614    1008.73703
## rm      31.646147    6705.02638
## age      9.970243     575.13702
## dis     12.774430    1351.01978
## rad      3.911852      93.78200
## tax      7.624043     453.19472
## ptratio 12.008194     919.06760
## black    7.376024     358.96935
## lstat   27.666896    6927.98475
```

변수의 중요성을 계산했다. 첫번째 열은 주어진 변수를 제거했을 때, 나무모형의 평균제곱오차의 변화율이고, 두번째 열은 주어진 변수를 제거했을 때 노드의 순수도(분류나무의 경우는 이탈도, 회귀나무의 경우는 잔차제곱합)의 변화율이다.

배깅과 랜덤숲 R 코드 V

```
varImpPlot(rf.boston)
```



변수의 중요도를 그림으로 그렸다.

부스팅(boosting) I

알고리듬

1. $\hat{f}(x) \equiv 0$, $r_i = y_i, i = 1, 2, \ldots, n$이라 놓는다.
2. $b = 1, 2, \ldots, B$
   - 2.1 잔차 r을 반응변수로 하여 (X, r)에 종료노드가 $d + 1$개인 나무를 적합하여 $\hat{f}^b$라 한다.
   - 2.2 $\hat{f} \longleftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
   - 2.3 $r_i \longleftarrow r_i - \lambda \hat{f}^b(x_i)$
3. 최종 추정량

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

부스팅(boosting) II

근거

부스팅 방법은 느린 예측모형(slow learner)라고 하는데 한꺼번에 $\hat{f}$을 발견하는 것이 아니라 조금씩 잔차를 줄여가며 $\hat{f}$을 개선시켜나가는 것이다.

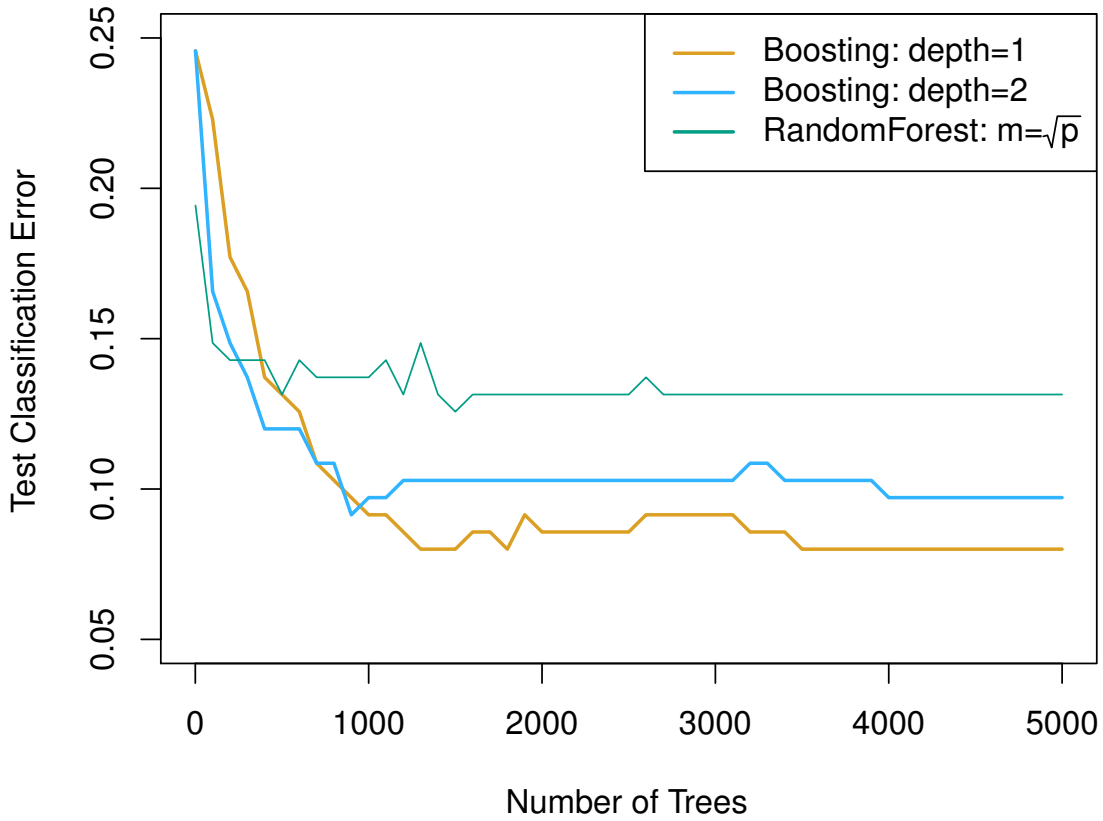
조율파라미터 λ의 선택

교차검증을 통해 결정한다.

참고

1. B가 커지면 배깅과 다르게 과적합 할 수 있다. 과적합이 매우 느리게 나타난다. 교차검증을 통해 B를 결정한다. B가 커질수록 모형 공간이 커진다. 즉 유연해진다.
2. λ를 보통 0.01이나 0.001을 쓴다. λ가 매우 작으면 B가 커져야 한다.
3. $d = 1$인 나무, 즉 1번 분할한 나무를 많이 쓴다. 1번 분할한 나무를 그루터기(stump)라고 한다.

부스팅(boosting) III



부스팅(boosting) IV

15개 범주 cancer gene expression 자료의 예. boosting with $d = 1$이 가장 좋다.

아다부스트(adaboost) : 분류문제에서 부스팅

여기서 반응변수의 값은 $-1, 1$를 갖는다고 하자.

1. 가중치 $w_i = \frac{1}{n}, i = 1, 2, \ldots$라 초기화 한다.
2. $b = 1, 2, \ldots, B$.
   - 2.1 가중치 (w_i)를 이용하여 분류기 f_b를 적합한다.
   - 2.2 오차를 다음과 같이 계산한다.

$$err_b := \frac{\sum_{i=1}^{n} w_i I(y_i \neq f_b(x_i))}{\sum_{i=1}^{n} w_i}$$

   - 2.3 $c_b := \log \frac{1 - err_b}{err_b}$로 놓는다.
   - 2.4 가중치를

$$w_i = w_i e^{c_b I(y_i \neq f_b(x_i))}$$

로 업데이트 한다.

부스팅(boosting) V

노트.

1. 아다부스트가 가장 먼저 제안된 알고리듬이다.
2. 아다부스트의 본래 목적은 훈련오차를 빠르고 쉽게 줄이는 것이었다. f_b의 오분류율이 $\frac{1}{2}$보다 작다면 훈련오차는 지수적으로 줄어든다는 것이 알려져 있다. 그런데 경험적으로 예측오차가 유의하게 향상되었다.
3. 아다부스트의 원리는 단계별전진선택법과 같다는 것이 밝혀졌고, 축소추정으로 과대적합을 피할 수 있다는 것을 경험적으로 보였다.
4. 아다부스트의 원리는 가파른 강하(steepest descent) 알고리듬으로 해석할 수 있다는 것이 알려졌다. 즉, 손실함수를

$$L(y, f) = e^{-yf}$$

라 할 때, 경험위험최소추정량

$$argmin_{f \in L(\mathcal{F})} \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(x_i)}$$

부스팅(boosting) VI

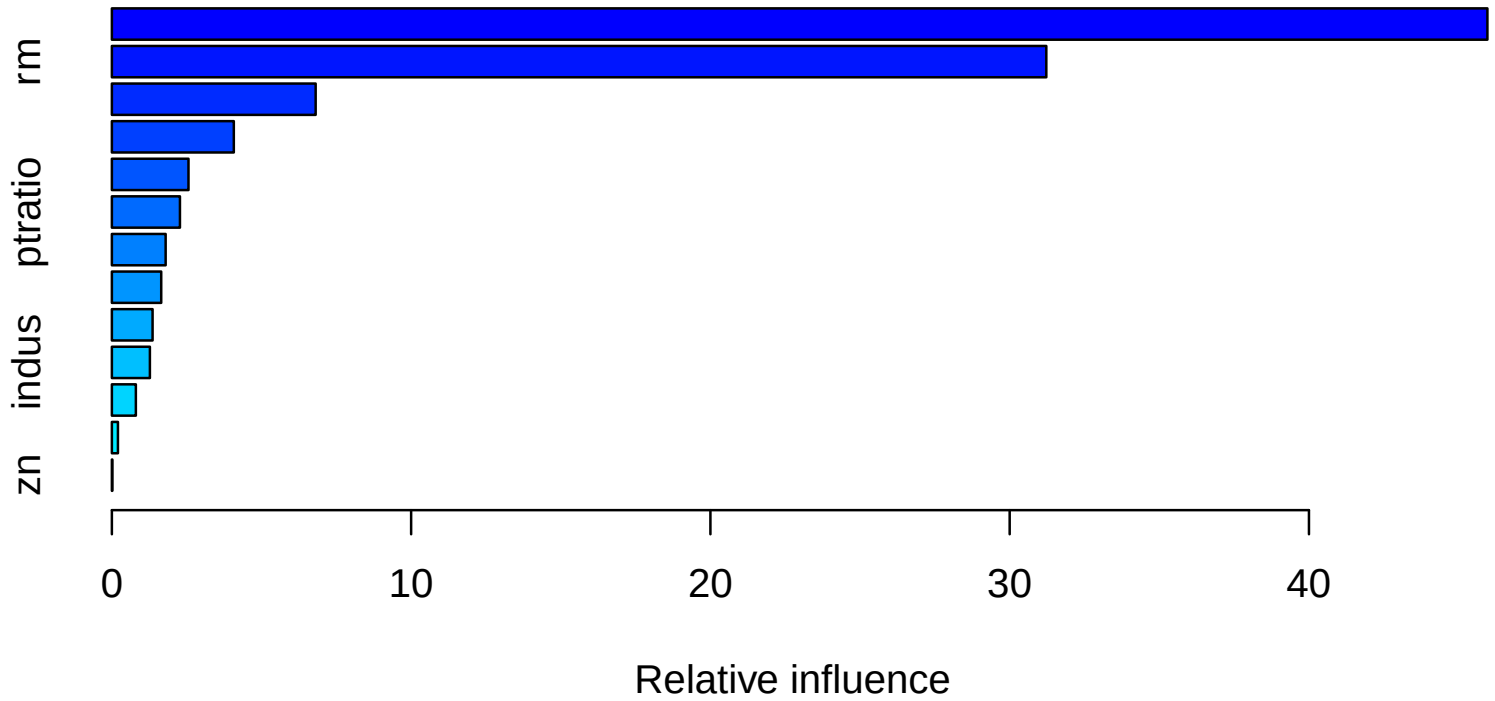
를 구하는 가파른 강하 알고리듬이다.

5. 이를 이용하여 다른 손실함수에도 적용할 수 있다.

부스팅 R 코드 I

```
library(gbm)

## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1

set.seed(1)
boost.boston=gbm(medv~.,data=Boston[train,], distribution="gaussian", n.trees=5000, intera
summary(boost.boston)
```

부스팅 R 코드 II



부스팅 R 코드 III

```
##             var    rel.inf
## lstat     lstat 45.9627334
## rm           rm 31.2238187
## dis         dis  6.8087398
## crim       crim  4.0743784
## nox         nox  2.5605001
## ptratio ptratio  2.2748652
## black     black  1.7971159
## age         age  1.6488532
## tax         tax  1.3595005
## indus     indus  1.2705924
## chas       chas  0.8014323
## rad         rad  0.2026619
## zn           zn  0.0148083
```

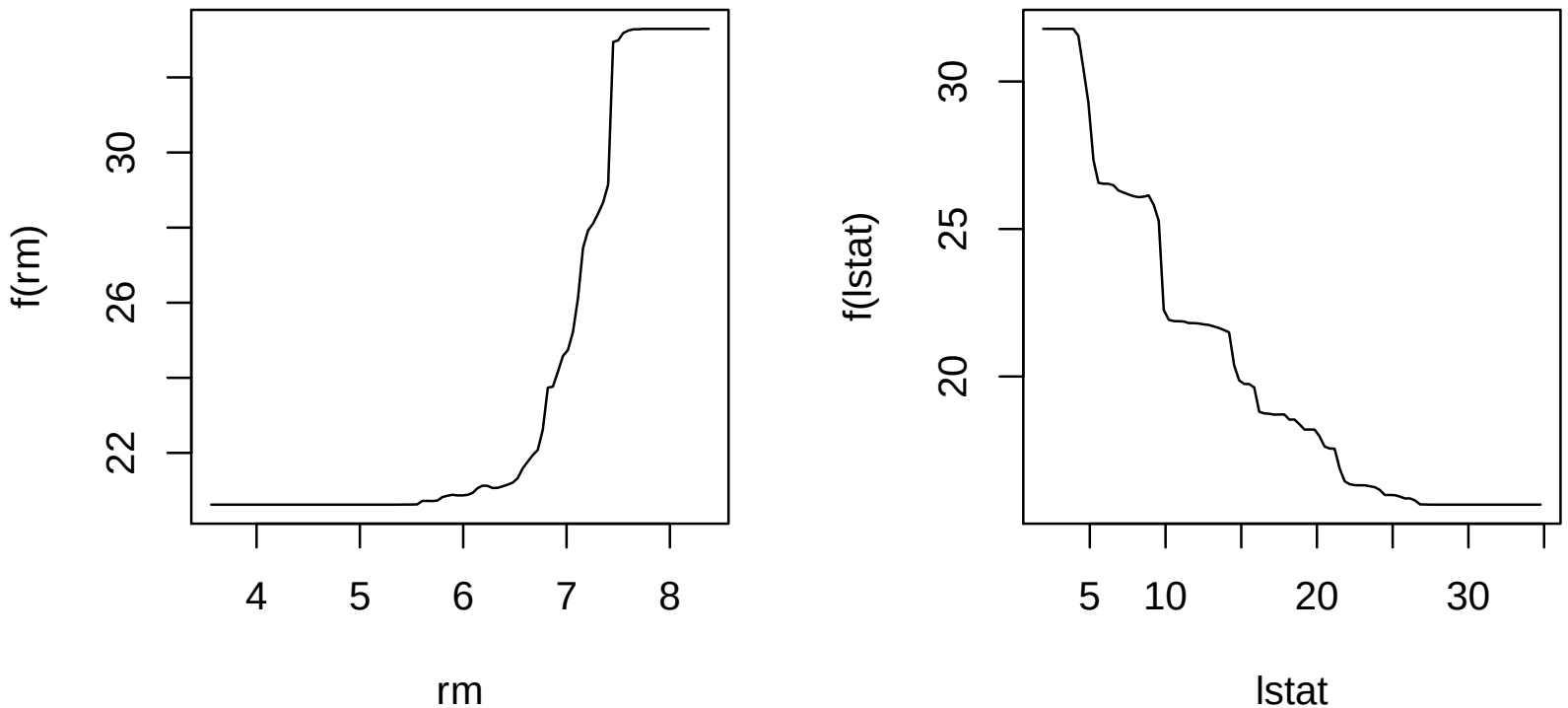
회귀나무이어서 distribution="gaussian"를 썼다. 분류나무인 경우는 "bernoulli"를 쓴다. n.tree=5000은 $B = 5000$개의 반복을 한다는 뜻이다. interaction.depth=4, 는 각 반복에서 더해지는 나무의 깊이를 4로 제한한다는 뜻이다.
summary 결과의 rel.inf는 상대적 영향력(relative influence)를 말한다.

노트.

상대적 영향력은 어떻게 정의되는가?

부스팅 R 코드 IV

```
par(mfrow=c(1,2))
plot(boost.boston,i="rm")
plot(boost.boston,i="lstat")
```



변수들의 주변효과(marginal effect)를 그린다.

부스팅 R 코드 V

```
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)

## [1] 11.84434
```

시험오차를 계산한다.

```
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",n.trees=5000,interacti
```

shrinkage 옵션은 부스팅 알고리듬에서 조율파라미터 λ의 값을 정한다. 디폴트는 0.001이다. 여기서는 $\lambda = 0.2$를 썼다.

```
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)

## [1] 11.51109
```

참고문헌

1. 아래의 책에서 제공되는 그림을 써서 슬라이드를 만들었다.
   James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning. New York: springer.
2. 배깅에 관한 설명은 김진석 교수(동국대)의 배깅에 관한 강의노트를 참고하였다.