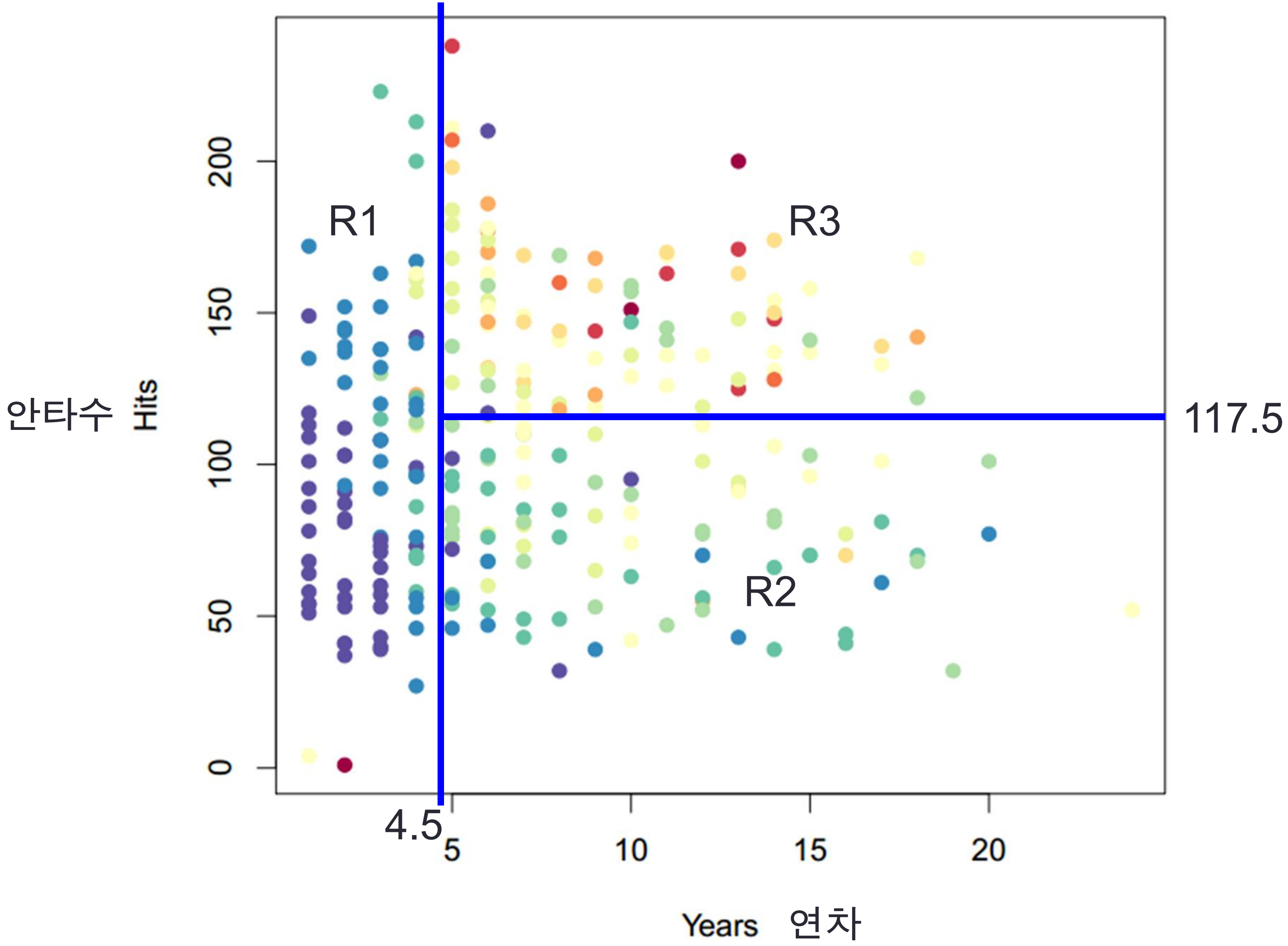
Ch.8
Tree based Methods

Decision Tree

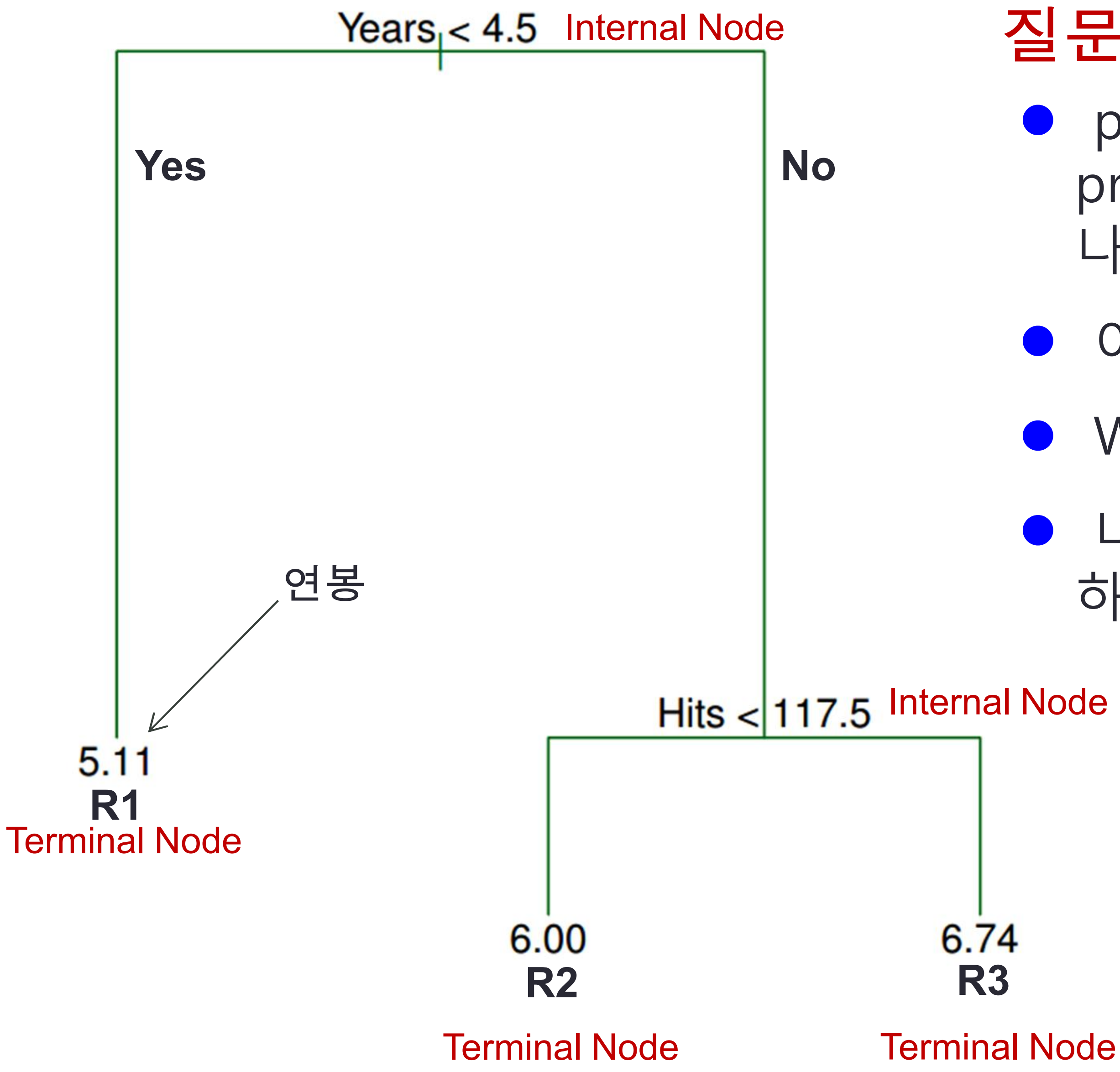
- Regression과 Classification에 모두 사용할 수 있음
- Predictor space를 predictor(feature)의 특성에 따라 나눔
- 이해하기 쉬움.
- predictor가 Numeric 또는 categoric 이어도 됨
- Exploratory Data Analysis 에도 유용 : Feature 선정 등
- 성능은 그리 만족스럽지 않을 수도 : Overfitting 등

예) 야구 선수 연봉을 feature(predictor) space에서 분할한다면

연봉 : 낮음(**blue**, **green**) → 높음(yellow, red)



예) Regression Tree로 나타내면



질문 :

- predictor space를 나눌 때 predictor 중에 어떤 것을 써서 나누나?
- 어떻게 나누나? 어떤 근거로?
- Where to split?
- 나누는 것을 어느 레벨까지 해야 하나?

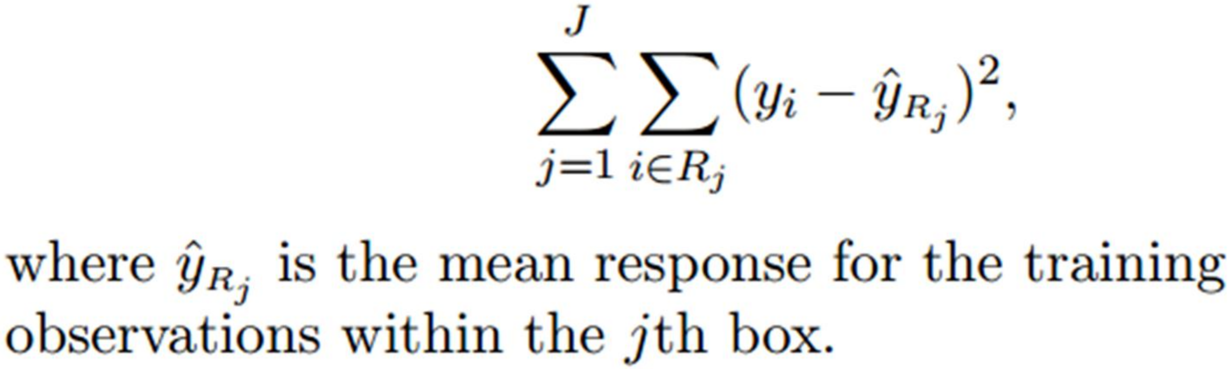
Regression Tree의 해석

- **Years** predictor가 연봉 결정에 가장 중요한 변수. 오래된 선수들의 연봉이 높다
- 년수가 낮은 선수들에게 **Hits**(전년)안타수는 연봉에 별 영향이 없는 모양
- 년수가 오래된 선수들에게는 안타수가 연봉 결정에 변수가 됨. 안타수가 높은 선수들의 연봉이 높음

Regression Tree 만들기

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.

- The goal is to find boxes $R_1,\ldots,R_J$ that minimize the RSS, given by

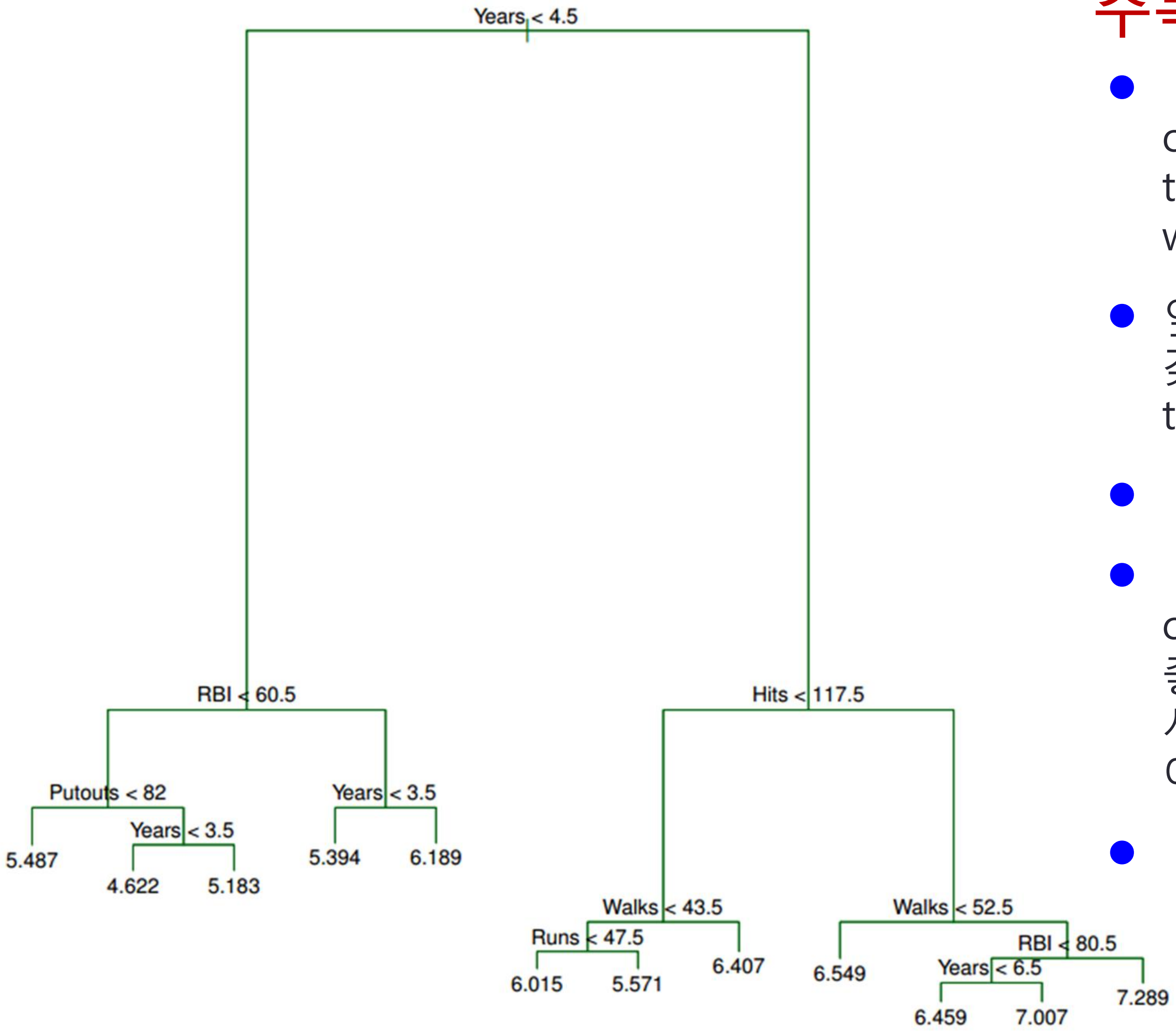


$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i-\hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

- We take a **top-down, greedy** approach that is known as recursive binary splitting.

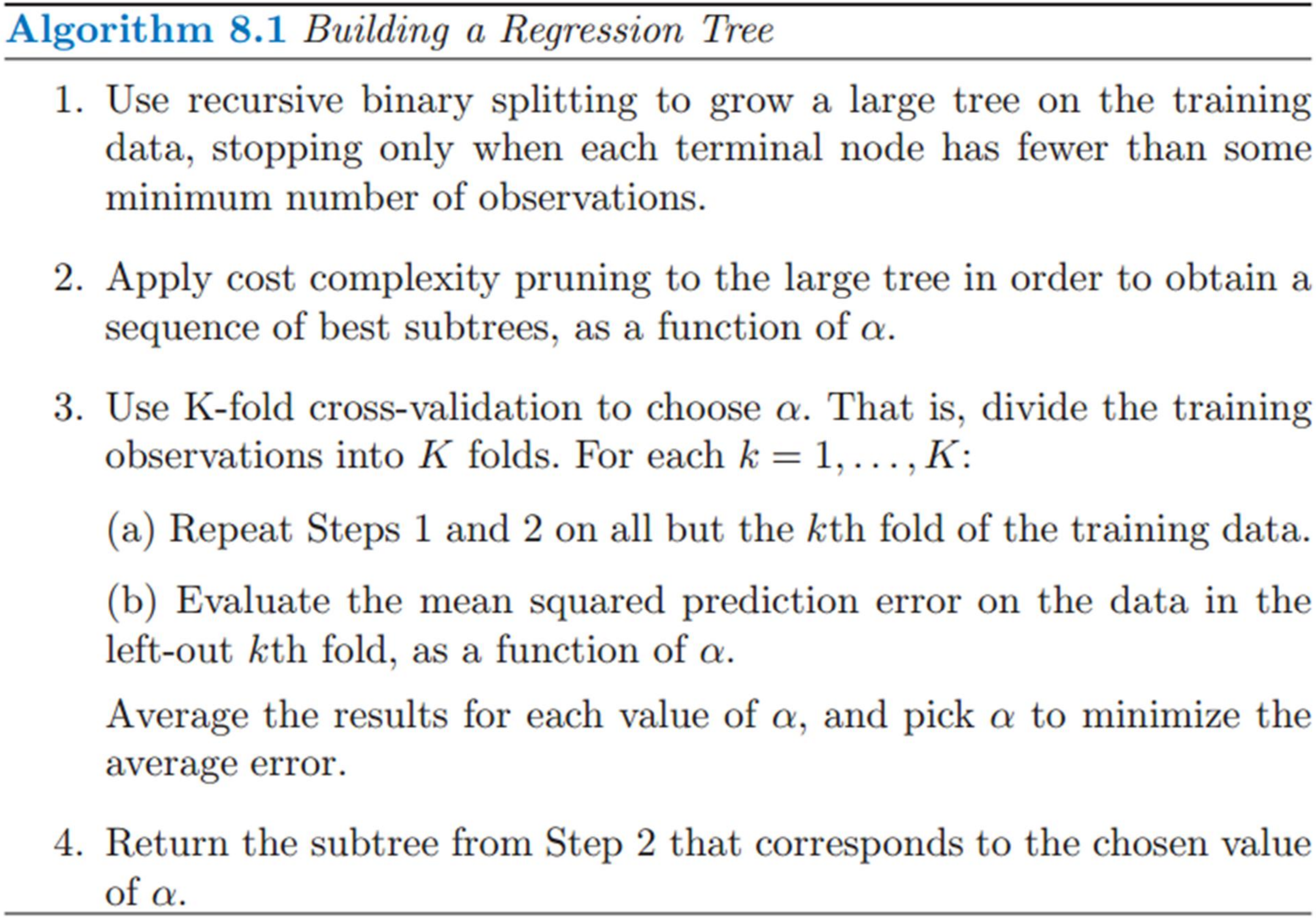
Prediction, Overfit & Pruning



주목 :

- **Predict** the response for a given test observation using the **mean** of the training observations in the region to which that test observation belongs.
- 옆과 같이 많은 terminal node들을 갖는 tree는 training error는 작지만, test error는 크다
- 어떻게 하면 좋을까?
- 모든 가능한 tree의 형태에 대해 cross-validation 평가를 하여 가장 좋은 tree를 구할 수 있으나 너무 시간이 많이 들 수 있다. 이 경우에는 어떻게?
- Cost Complexity Pruning을 사용

Terminal Node의 수를 변화시키면서 최적 Tree 구하기



Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

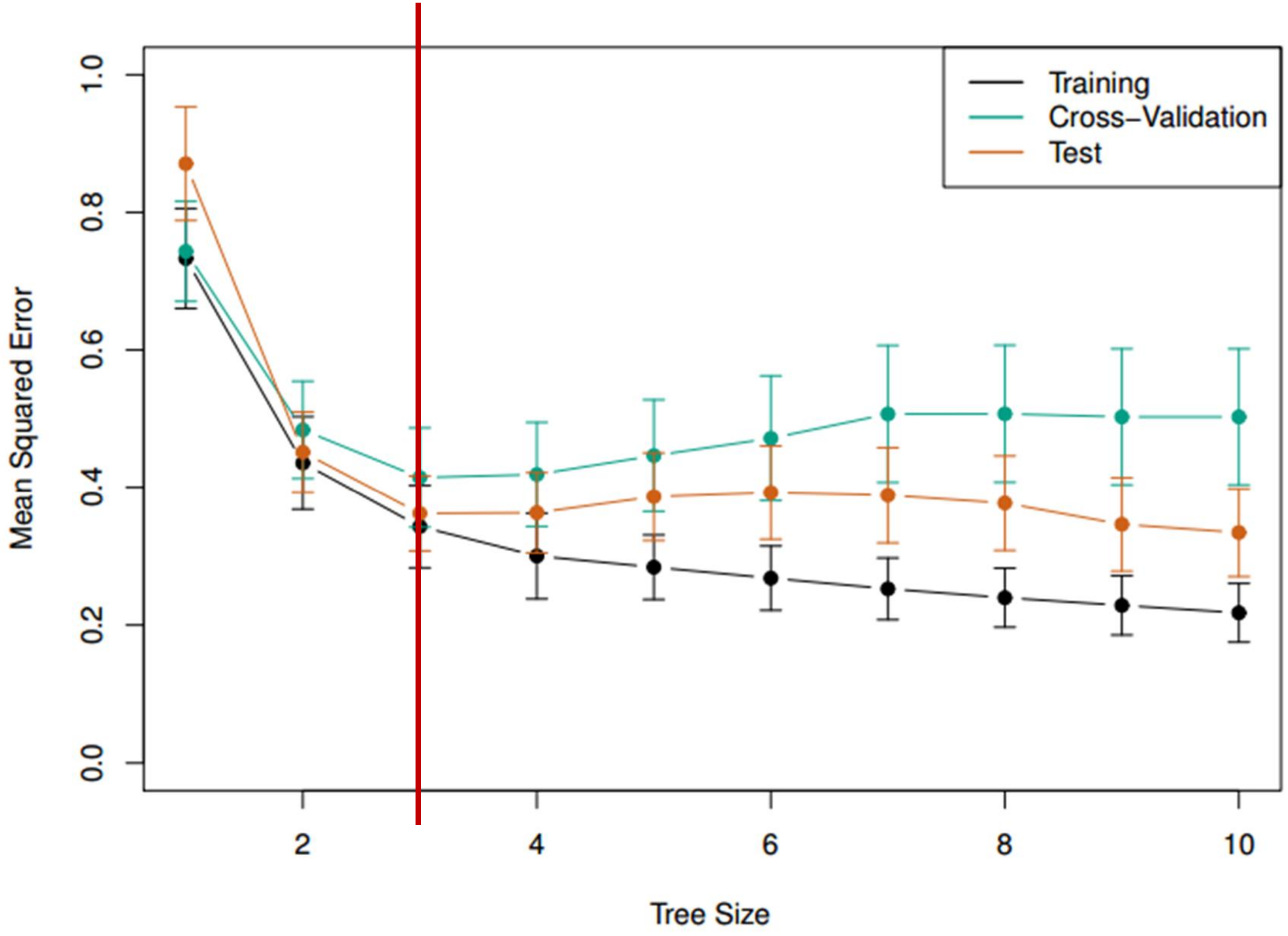
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Baseball example :



- Cross Validation shows that the minimum MSE is when the tree size is three (i.e. the number of leaf nodes is 3)

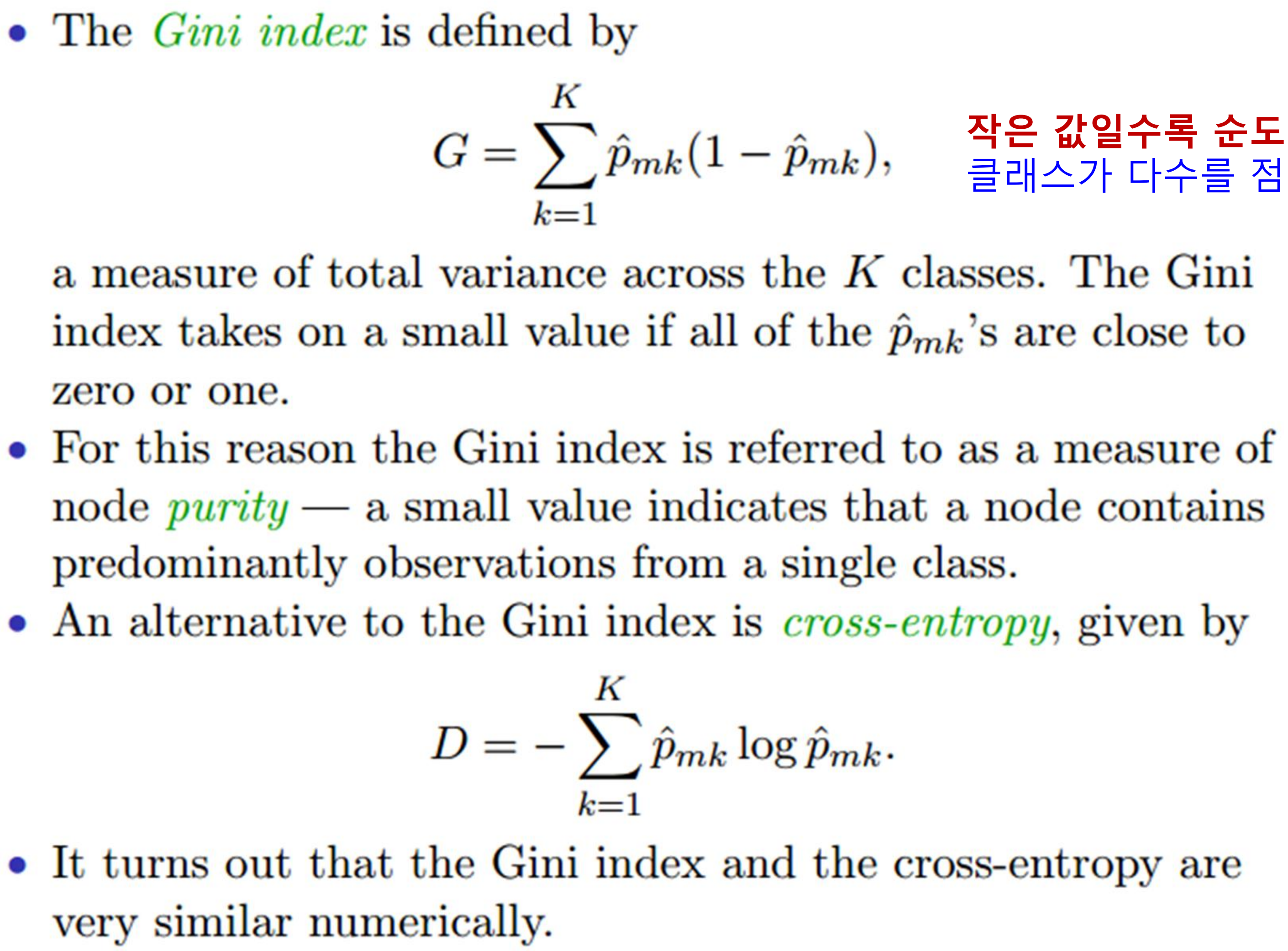
Classification Tree

- A classification tree is very similar to a regression tree except that we try to make a prediction for a categorical rather than continuous Y.
- For each region (or node) we predict the most common category among the training data within that region.
- The tree is grown (i.e. the splits are chosen) in exactly the same way as with a regression tree except that minimizing MSE no longer makes sense.
- The natural alternative to MSE is the classification error rate. The classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_{k}(\hat{p}_{mk})$$

m 노드(region)에 3개의 클래스에 속한 training observation들의 분포가 각각 5/10, 3/10, 2/10 비율로 있다면 5/10 비율을 차지한 클래스가 그 node의 대표 클래스이다. 따라서 이 경우 E = 1-0.5 = 0.5

- Classification error is not sufficiently sensitive when tree-growing. There are several possible different criteria to use such as the “gini index” and “cross-entropy”. 하지만 최종적으로 pruned tree들을 평가할 때는 classification error rate가 더 적당한 지표(metric) 일수 있다



- The *Gini index* is defined by

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

작은 값일수록 순도가 높다. 즉, 하나의 클래스가 다수를 점하고 있다

a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one.

- For this reason the Gini index is referred to as a measure of node *purity* — a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is *cross-entropy*, given by

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

- It turns out that the Gini index and the cross-entropy are very similar numerically.

예)

- m 노드(region)에 3개의 클래스에 속한 training observation들의 분포가 각각 5/10, 3/10, 2/10 비율로 있다면 G= 0.5*0.5 + 0.3*0.7 + 0.2*0.8 = 0.62

-training observation들의 분포가 각각 7/10, 1/10, 1/10, 1/10 이라면

G = 0.7*0.3 + 0.1*0.9 + 0.1*0.9 + 0.1*0.9 = 0.48 (순도가 높음, Higher Purity(Homogeneity)

Decision Tree의 장단점

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression! (A+)
- ▲ Regression과 Classification에 모두 사용
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches (C+). Tree 크기(깊이)에 제한을 주지 않으면 매우 overfit.

Bagging & Random Forest

Decision Tree를 이용한 개선책은?

- Decision Tree의 가장 큰 문제는 High Variance. 즉, Training set에 민감해 정작 실제 환경에서는 썩 잘 맞추지 못할 수도
- Test Case에서도 Tree 모델이 low variance를 보이게 할 수 없을까?
- **Bagging** (**b**oostrap **agg**regat**ing**)방법을 쓰자
- **Bagging** : Training Set의 구성을 달리해 여러 개의 Decision Tree를 만들어 그 것들의 통합된 판단에 따름. 일종의 집단 지성(판단) 효과를 이용한 것. 단, 이때 개별 tree들은 어떤 특정 분야의 전문가라는 말이 아니라, 제각기 조금씩 다른 데이터를 보고 판단을 내렸음

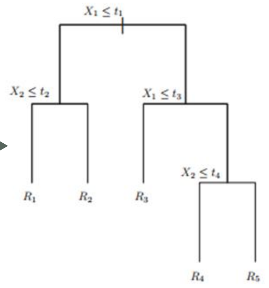
Decision Tree를 활용한 Bagging 예

Observation	X	Y
1	5.0	2.8
2	3.4	3.3
3	4.2	6.1

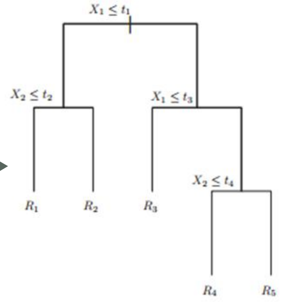
오리지널 데이터

Random sampling with **replacement**

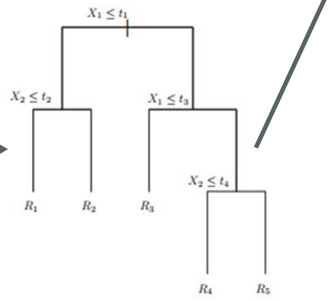
Observation	X	Y
3	4.2	6.1
2	3.4	3.3
3	4.2	6.1



Observation	X	Y
1	5.0	2.8
2	3.4	3.3
3	4.2	6.1



Observation	X	Y
2	3.4	3.3
2	3.4	3.3
1	5.0	2.8



Bootstrap 데이터

Ave

Variance가 낮아짐

Bagging 알고리즘

- 오리지널 데이터에서 B개의 bootstap training data set을 만든다
- B개의 training data set 각각을 이용해 B개의 모델을 만들고, test set에 대해 B개의 prediction을 얻는다
- 통합 Prediction

  - Regression : B개의 Prediction의 평균

  - Classification : B개 Prediction의 다수결

■ Regression Tree를 이용한 Bagging

- B개의 bootstap training data set를 생성해 B개의 regression tree를 만든다 (Pruning을 안함)
- B개의 regression tree를 test set에 적용해 B개의 결과를 얻은 후 평균을 구한다

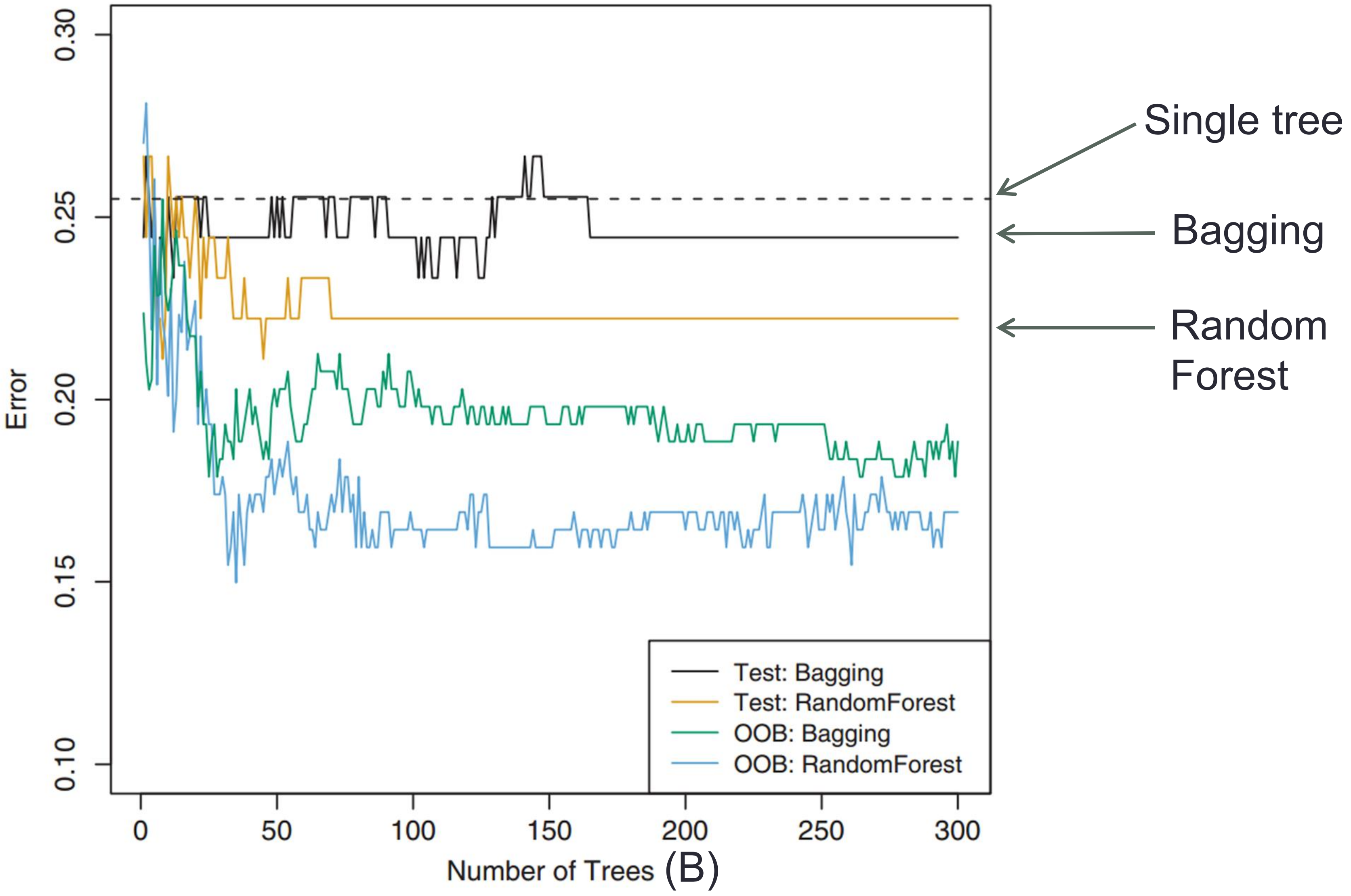
■ Classification Tree를 이용한 Bagging

- B개의 bootstap training data set를 생성해 B개의 classification tree를 만든다 (Pruning을 안함)
- 예측 결과를 통합한다

  - 방법 1 : B개의 개별 classification tree결과를 다수결 처리

  - 방법 2 : 개별 classifier가 classification을 확률로도 제공하면 확률 평균을 구해 가장 큰 확률평균을 보인 클래스로 할당

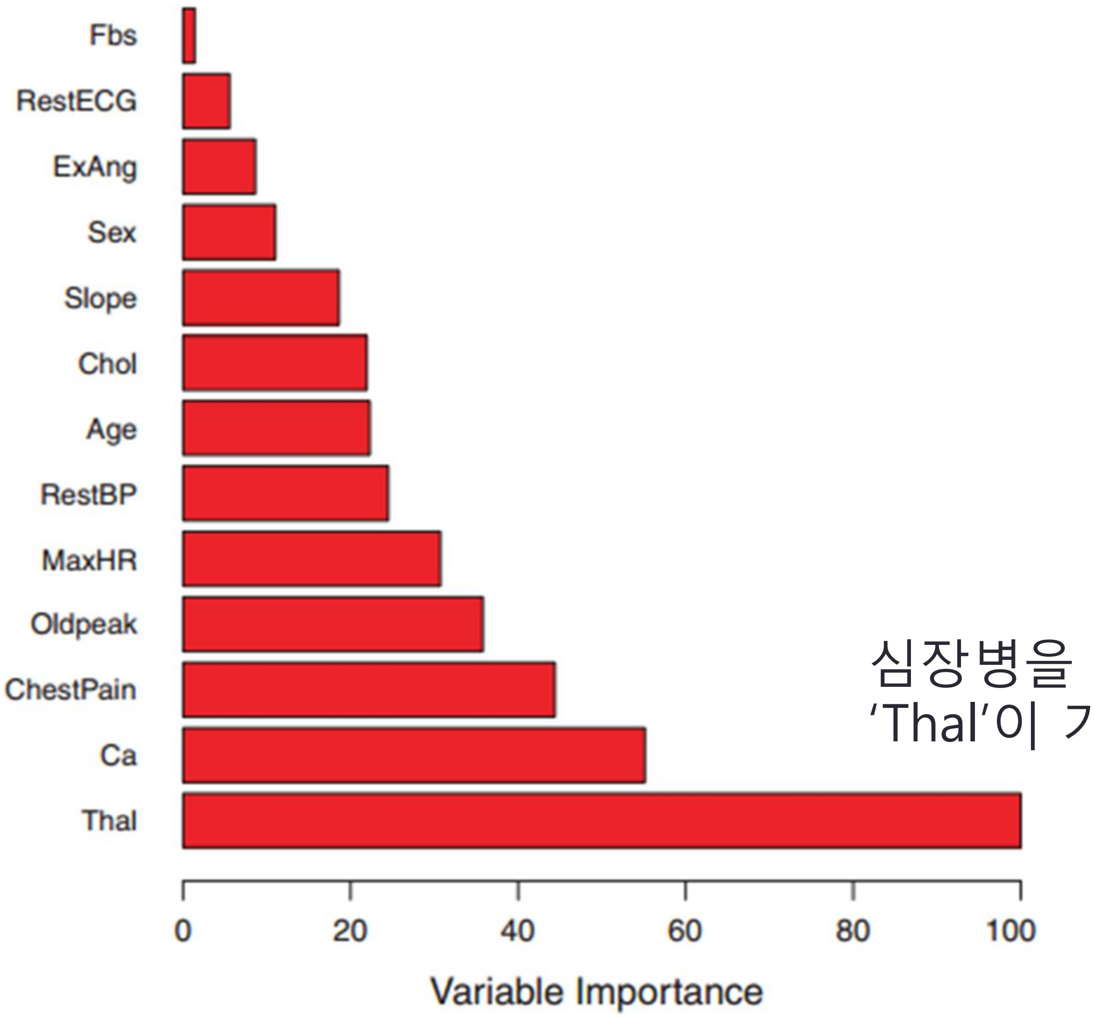
Bagging and Random Forest on Heart data



Bagging을 통한 변수(Predictor, Feature)들의 중요성 판단

* 앞서 우리는 decision tree가 어떤 변수를 사용해 tree를 확장해 나갈까 판단할 때, 확장 후 gini index를 가장 작게 만드는 것을 택함을 알았다. 마찬가지로, bagging에서 변수 중 어떤 것을 split하면 B개의 tree에 걸쳐 가장 평균적으로 gini index를 낮출 수 있는가를 구하면 변수의 중요성을 가늠할 수 있다

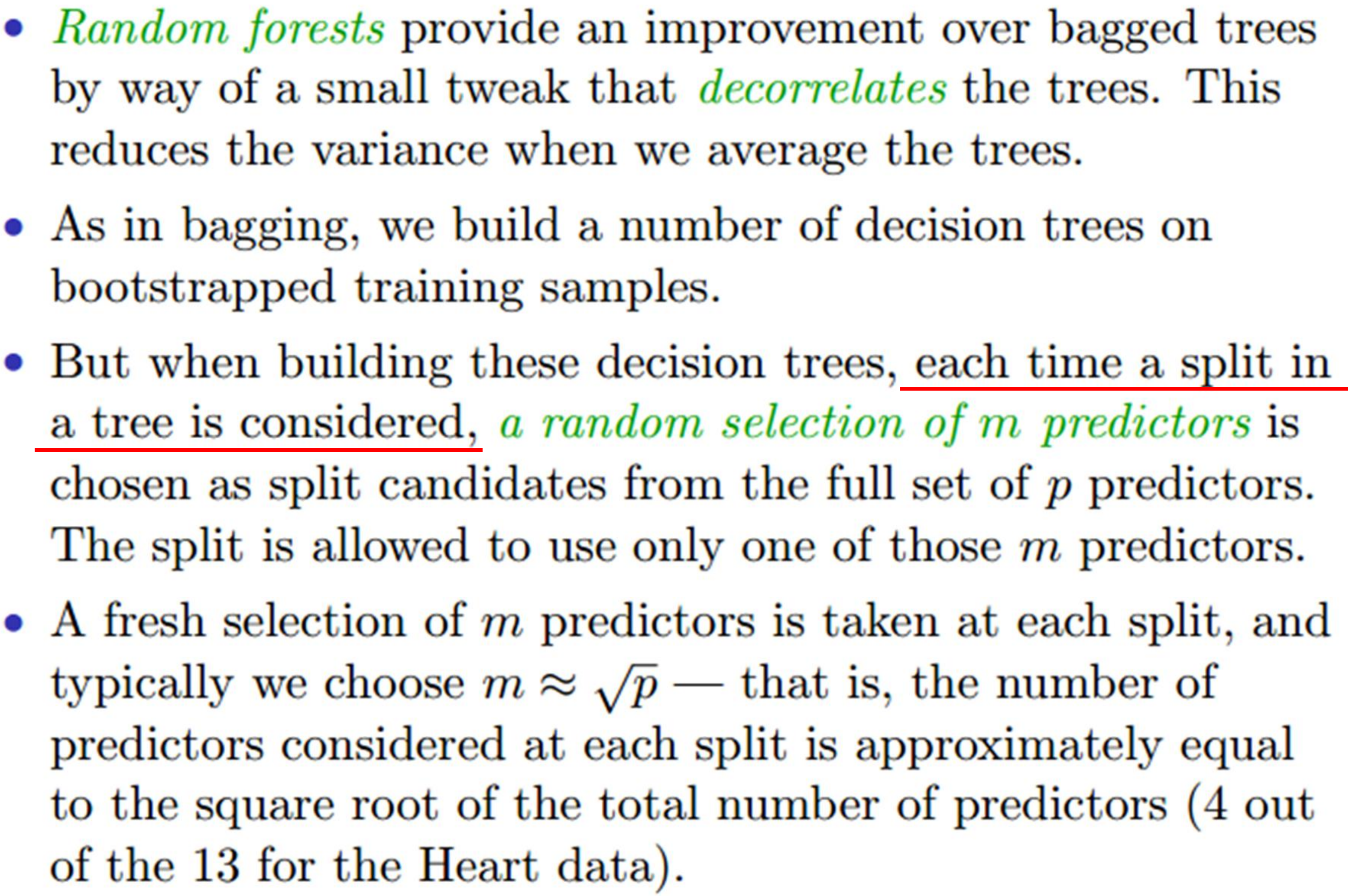
- Which variables are most useful in predicting the response?
  - We can compute something called relative influence plots.
  - These plots give a score for each variable.
  - These scores represents the decrease in MSE(gini index) when splitting on a particular variable
  - A number close to zero indicates the variable is not important and could be dropped (MSE나 gini index를 낮추지 못했음)
  - The larger the score the more influence the variable has.



심장병을 예측할 때
‘Thal’이 가장 중요한 변수

FIGURE 8.9. *A variable importance plot for the* Heart *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

Random Forest



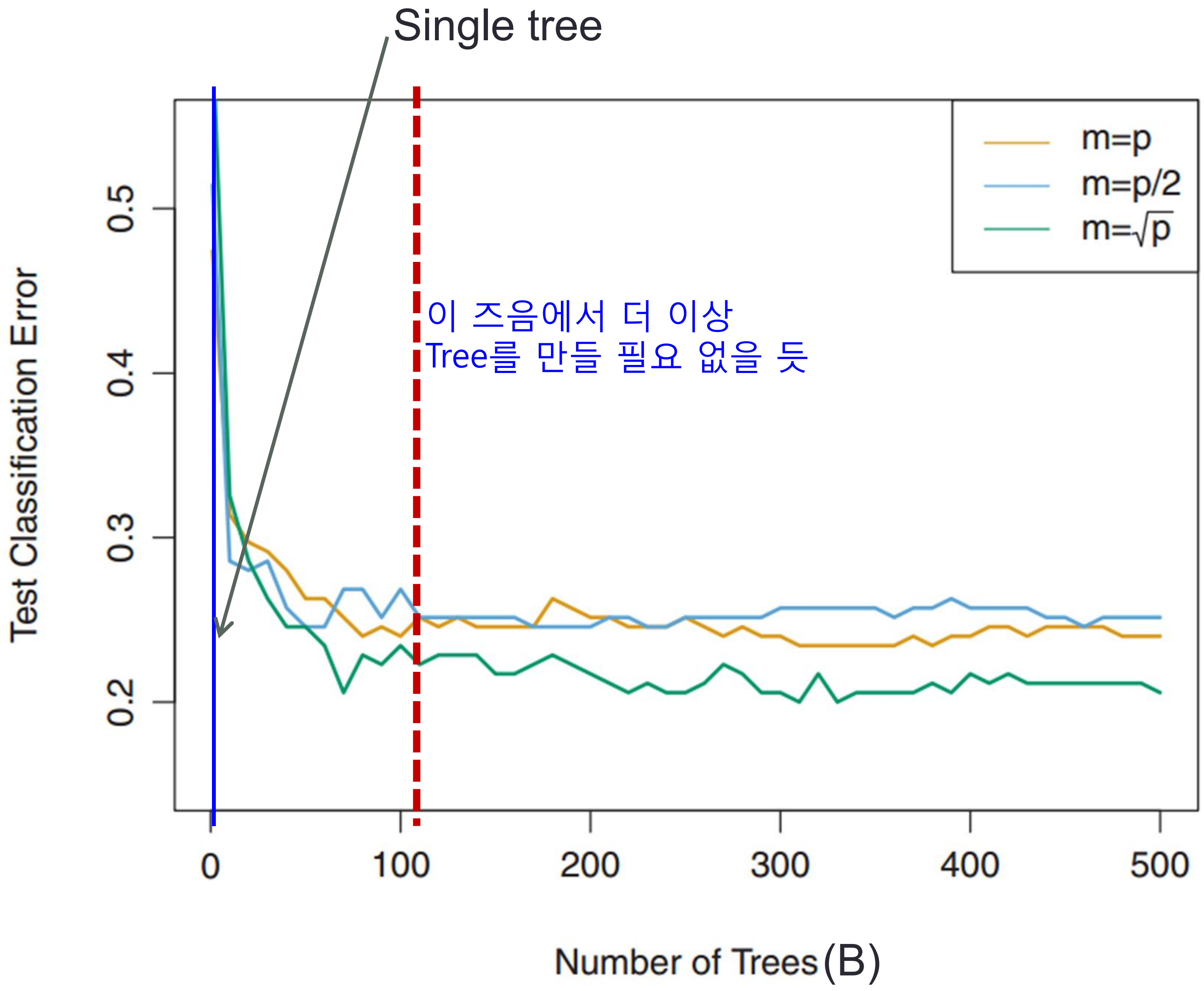
- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, *a random selection of m predictors* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Node를 나눌 때, 모든 변수중에서 고르지 말고, 변수 중 일부 m개 변수를 random 하게 골라 그 내에서 split할 때 쓸 변수를 고른다 (보통 $m = p^{1/2}$)

노드를 나눌 때 모든 predictor 중에서 가장 중요한 것을 써야지 random하게 뽑은 일부 predictor 중에서 고르라니 이런 해괴한. 그런데,

- Suppose that we have a very strong predictor in the data set along with a number of other moderately strong predictor, then in the collection of bagged trees, most or all of them will use the very strong predictor for the top split! (모든 tree들이 가장 중요한 영향가 있는 변수부터 쓰려고 할 것임)
- All bagged trees will look similar. Hence all the predictions from the bagged trees will be highly correlated (그럼, 모든 tree들이 비슷해 짐)
- Averaging many highly correlated quantities does not lead to a large variance reduction, and thus random forests "de-correlates" the bagged trees leading to more reduction in variance (비슷한 녀석들을 통합해 보아야 variance 절감 효과가 별로 없음)

‘m’ 값에 따른 Random Forest 성능



이 즈음에서 더 이상
Tree를 만들 필요 없을 듯

p=500 predictors

m=p : 단순 bagging

m= $500^{1/2} \approx 22$