Data
Science

R 강의: pruning of rpart tree

rpart 나무의 post pruning

- cost-complexity pruning
  - error 와 나무의 complexity 를 함께 고려 하여 가지치기
  - 방법: full tree 생성 후
    cost(or cost-complexity) 함수를 최소화 하도록
    rpart 나무를 가지 치기
  - cost = error + complexity
  - complexity = cp * 잎노드의 수
  - cp: 잎노드 하나에 부과하는 penalty

- 최적 cp
  - 실험적으로 정함

최적 cp 값 검색

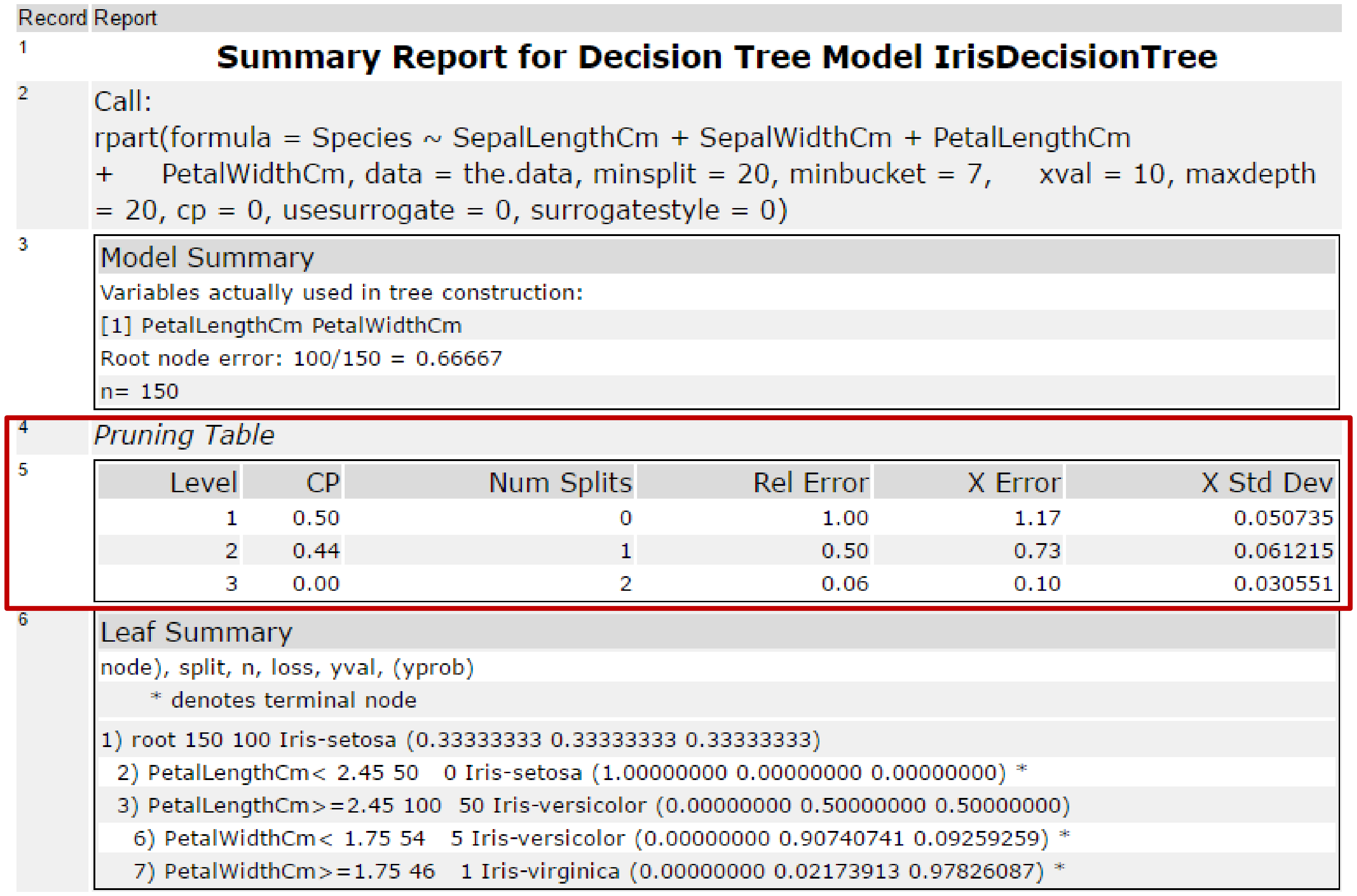
방법

- Full tree 생성하고 이 tree 에 대하여
- cp (complexity parameter) 를 0으로부터 시작하여 증가 시켜 가며 각 cp 값에 대하여
  - 최적 pruning 을 시도하고 pruning 된 나무에 대하여
  - xerror (cross validated relative error)를 측정합니다.
- xerror 가 최소인 경우의 cp 값이 최적 이므로
- 이 cp 값을 선택하여 full tree 를 pruning → final model
- Tree 생성및 성능 측정시 Cross valiation 방법 사용.
- xerror 는 순수한 cross validated error, 복잡도 포함 안되어 있음.
- cost 계산시 error 는 재대입 error = rel error 사용

http://mlwiki.org/index.php/Cost-Complexity_Pruning

printcp(rpart_model)

- printcp(rpart_model): rpart 모델 출력



Record	Report
1	**Summary Report for Decision Tree Model IrisDecisionTree**

2
```
Call:
rpart(formula = Species ~ SepalLengthCm + SepalWidthCm + PetalLengthCm
+     PetalWidthCm, data = the.data, minsplit = 20, minbucket = 7,     xval = 10, maxdepth
= 20, cp = 0, usesurrogate = 0, surrogatestyle = 0)
```

3 Model Summary
```
Variables actually used in tree construction:
[1] PetalLengthCm PetalWidthCm
Root node error: 100/150 = 0.66667
n= 150
```

4 *Pruning Table*

5

Level	CP	Num Splits	Rel Error	X Error	X Std Dev
1	0.50	0	1.00	1.17	0.050735
2	0.44	1	0.50	0.73	0.061215
3	0.00	2	0.06	0.10	0.030551

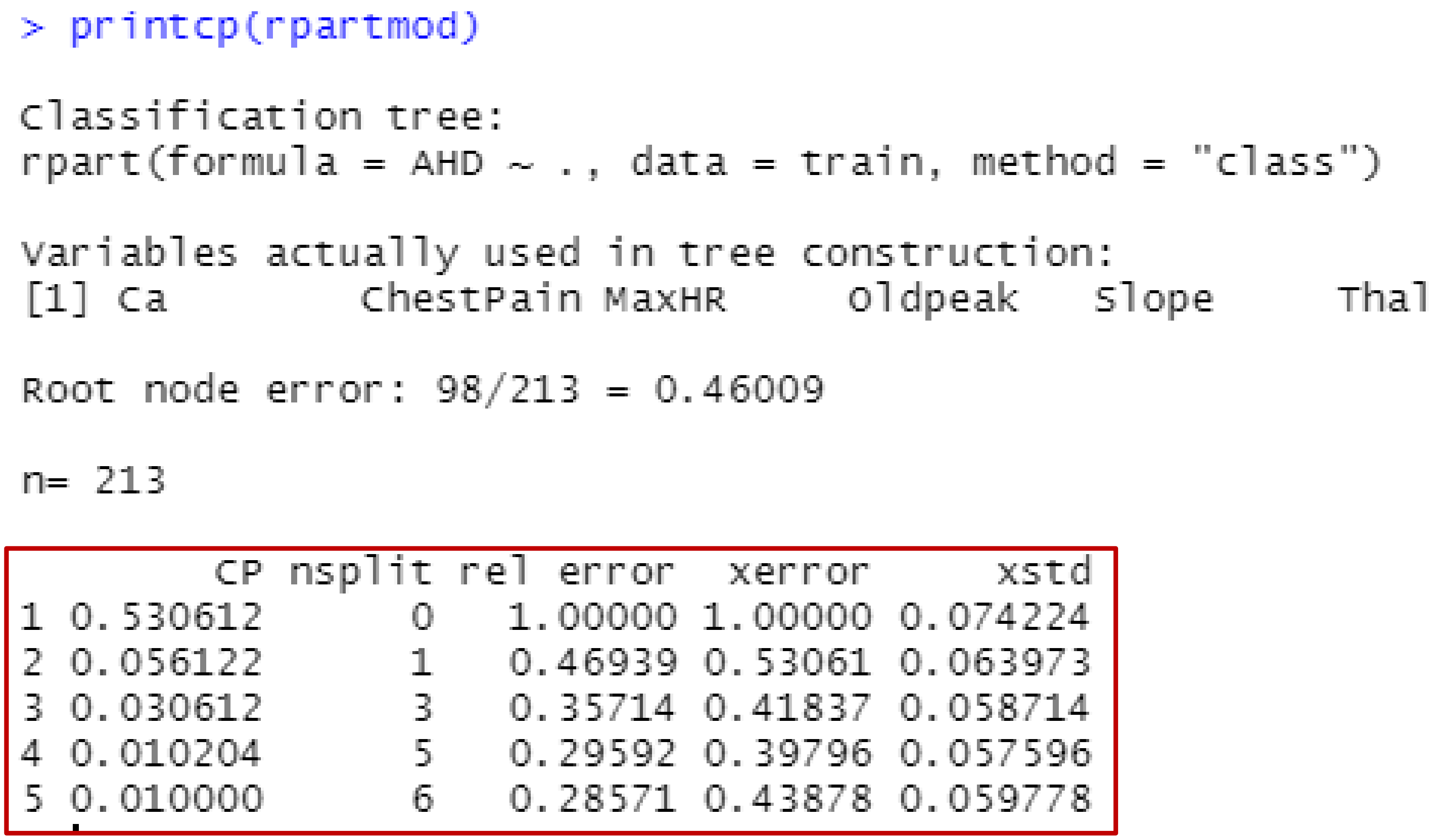
6 Leaf Summary
```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 Iris-setosa (0.33333333 0.33333333 0.33333333)
  2) PetalLengthCm< 2.45 50   0 Iris-setosa (1.00000000 0.00000000 0.00000000) *
  3) PetalLengthCm>=2.45 100  50 Iris-versicolor (0.00000000 0.50000000 0.50000000)
    6) PetalWidthCm< 1.75 54   5 Iris-versicolor (0.00000000 0.90740741 0.09259259) *
    7) PetalWidthCm>=1.75 46   1 Iris-virginica (0.00000000 0.02173913 0.97826087) *
```

printcp(rpart_model)

- Root node error
  - error on root node
- Pruning Table
  - CP
    - 잎 노드 한 개당 penalty
  - 각 CP 값에 대하여
    - Num Splits: number of splits in the tree
    - Rel(ative) error: resubstitution error rate scaled by Root node error, scaled by root_node_error(= resubstitution error / root_node_error)
    - X error: Cross validated error rate also, scaled by root_node_error
    - X std dev: Standard error of CV

printcp(rpart_model)

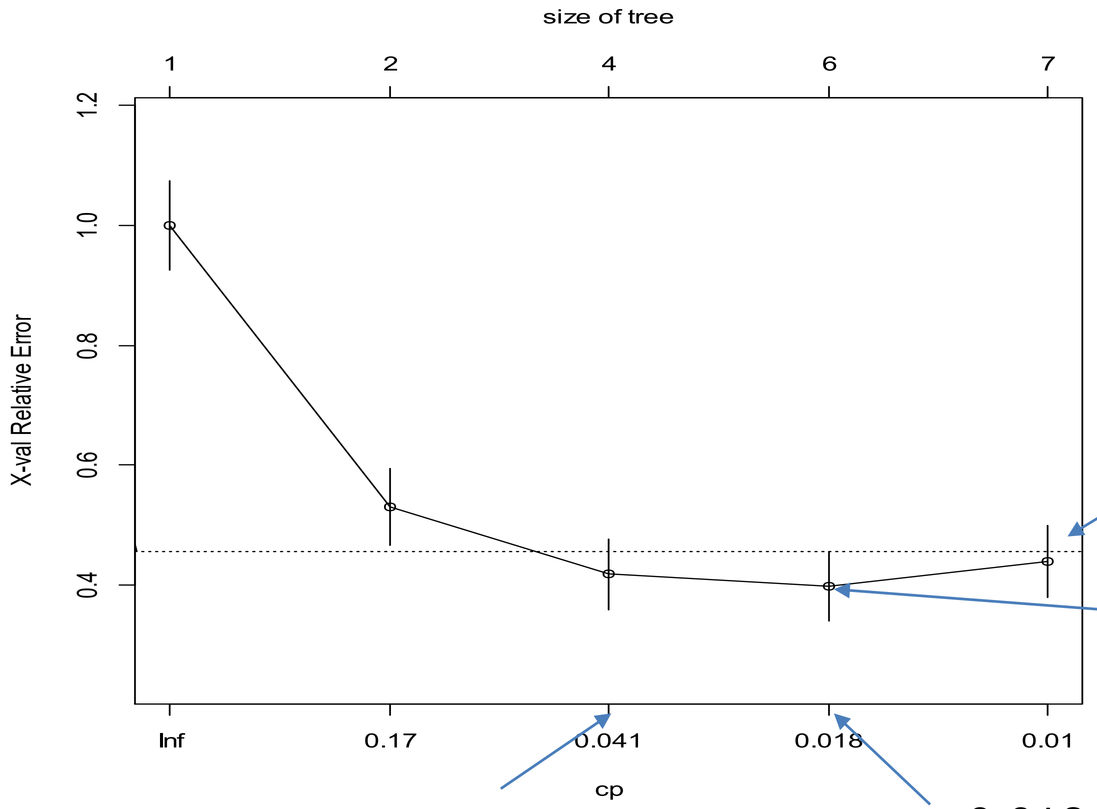
- Example



```
> printcp(rpartmod)

Classification tree:
rpart(formula = AHD ~ ., data = train, method = "class")

Variables actually used in tree construction:
[1] Ca        ChestPain MaxHR     Oldpeak   Slope     Thal

Root node error: 98/213 = 0.46009

n= 213

        CP nsplit rel error  xerror     xstd
1 0.530612      0   1.00000 1.00000 0.074224
2 0.056122      1   0.46939 0.53061 0.063973
3 0.030612      3   0.35714 0.41837 0.058714
4 0.010204      5   0.29592 0.39796 0.057596
5 0.010000      6   0.28571 0.43878 0.059778
```

plotcp(rpart_model)

- optimal cp value
  - Choose the leftmost one below the horizontal line



Horizontal line:
min + standard error(sd)

min xerror

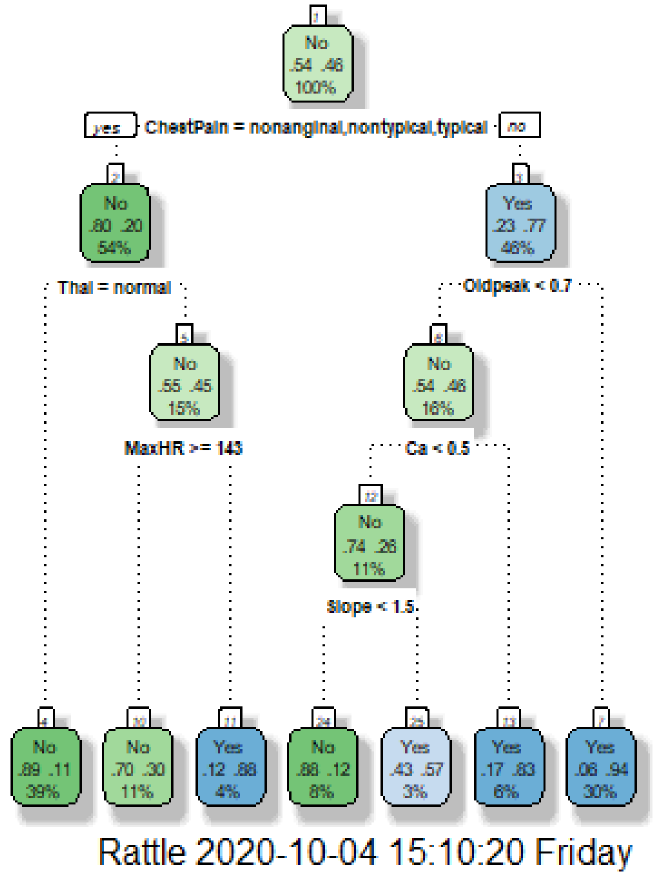
SE 범위 내에서 가장 작은 (간단한) 모델(cp = 0.041)을 최적모델(최적값)으로 선택

cp = 0.018 인 경우의 모델의 xerror 가 최소 이지만

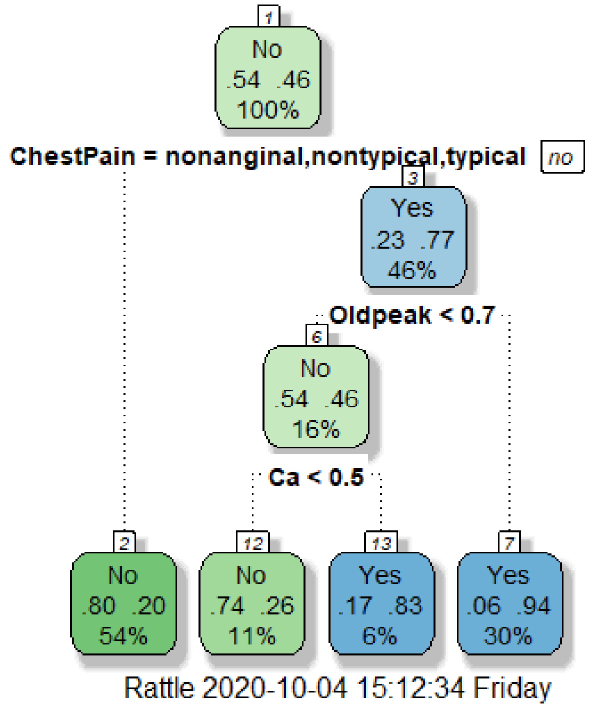
Full tree vs pruned tree

prune(rpart_model, cp = 0.041)

Full tree(acc = 0.71)



Pruned tree(acc = 0.73)



예제 코드 – pruning 예제

```
# rpart tree pruning

# Source
# https://rstudio-pubs-
static.s3.amazonaws.com/332255_d2b0ffd26ff64704a65573ba82a90ad6.html

# Data
# heart.csv
# https://www.kaggle.com/zhaoyingzhu/Heartcsv
# Heart desease data
# Target var : AHD with two values Yes and No

library(rpart)
library(caret)
library(e1071)
library(rattle)
library(tidyverse)

df<-read.csv('Heart.csv')
str(df)
head(df)
df$AHD <- as.factor(df$AHD)

# data partition

set.seed(10) # reproducability setting
intrain<-createDataPartition(y=df$AHD, p=0.7, list=FALSE)
train<-df[intrain, ]
test<-df[-intrain, ]

# Full tree
rpartmod<-rpart(AHD~. , data=train, method="class")
fancyRpartPlot(rpartmod)
rpartpred<-predict(rpartmod, test, type='class')
confusionMatrix(rpartpred, test$AHD)

printcp(rpartmod)
plotcp(rpartmod)

# choose the model (cp= 041) leftmost under the horizontal line
# which xerror is within the range of standard error
# from min xerror of the model with cp = 0.18

ptree<-prune(rpartmod, cp = 0.041)
fancyRpartPlot(ptree)

rpartpred<-predict(ptree, test, type='class')
confusionMatrix(rpartpred, test$AHD)
```

예제

- rpart pruning
  - https://dzone.com/articles/decision-trees-and-pruning-in-r
  - https://rstudio-pubs-static.s3.amazonaws.com/332255_d2b0ffd26ff64704a65573ba82a90ad6.html

Note:

Q: Why are the cp values in plotcp() chart modified from the original table?

A: printcp() gives the minimal cp for which the pruning happens. plotcp() plots against the geometric mean

https://stackoverflow.com/questions/31546895/why-are-the-cp-values-in-plotcp-chart-modified-from-the-original-table

Note: How to select the optimal cp value for pruning

1. Use the first level (i.e. least nsplit) with minimum xerror.
   The first level only kicks in when there are multiple level having same, minimum xerror. This is the most common used method.
2. Use the first level where xerror falls into the ±1 xstd range of min(xerror), i.e., xerror < min(xerror) + xstd, the level whose xerror is at or below horizontal line. This method takes into account the variability of xerror resulting from cross-validation.
   Note: rel_error should NOT be used in pruning.
3. (A rarely used method) Use the first level where xerror ± xstd overlaps with min(xerror) ± xstd. i.e., the level whose lower limit is at or below horizontal line.

https://stackoverflow.com/questions/29197213/what-is-the-difference-between-rel-error-and-x-error-in-a-rpart-decision-tree