Decision Tree

Park Ju ho

2022 4 16

Rpart를 활용한 Tree

```
library(rpart)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#setosa,versicolor,virginica
c = rpart(Species ~., data = iris)

c
```

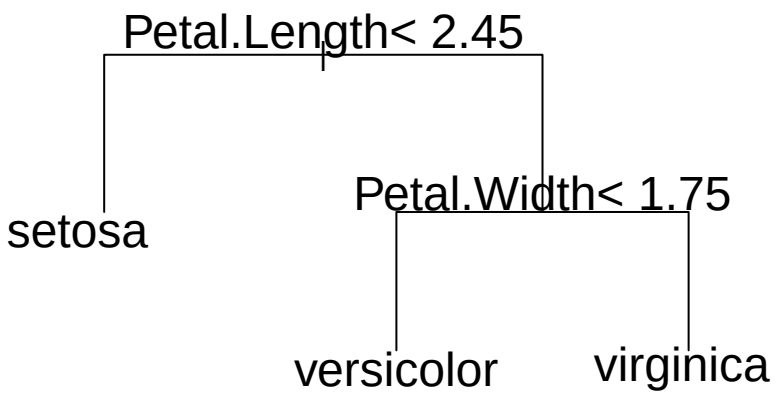
```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

총 150개의 iris data를 분할
왼쪽부터 노드(만약 Full Tree일 경우에 왼쪽부터 노드의 번호가 부여됨), 분할 기준, 노드 안의 원소 수, 손실율(yval를 기준으로 해당 범주가 아닌 것의 수), 현재 노드의 최빈 라벨, 각 범주별 비율(비율의 label 순서는 factor의 level순서와 같음)

1번 노드는 루트 노드이므로 생략
2번 노드는 Petal.Length를 기준으로 나뉘어졌으며 왼쪽 노드에는 100% Setosa가 들어감
3번 노드는 같은 기준으로 나뉘어진 노드 중 오른쪽 노드로 versicolor와 virginica가 반반씩 들어감
6번 노드는 Petal.width를 기준으로 나뉘어졌으며 54개 중 49의 versicolor가 있음
7번 노드는 같은 기준으로 나뉘어진 오른쪽 노드로 46개 중 45개의 virginica가 있음

그래프

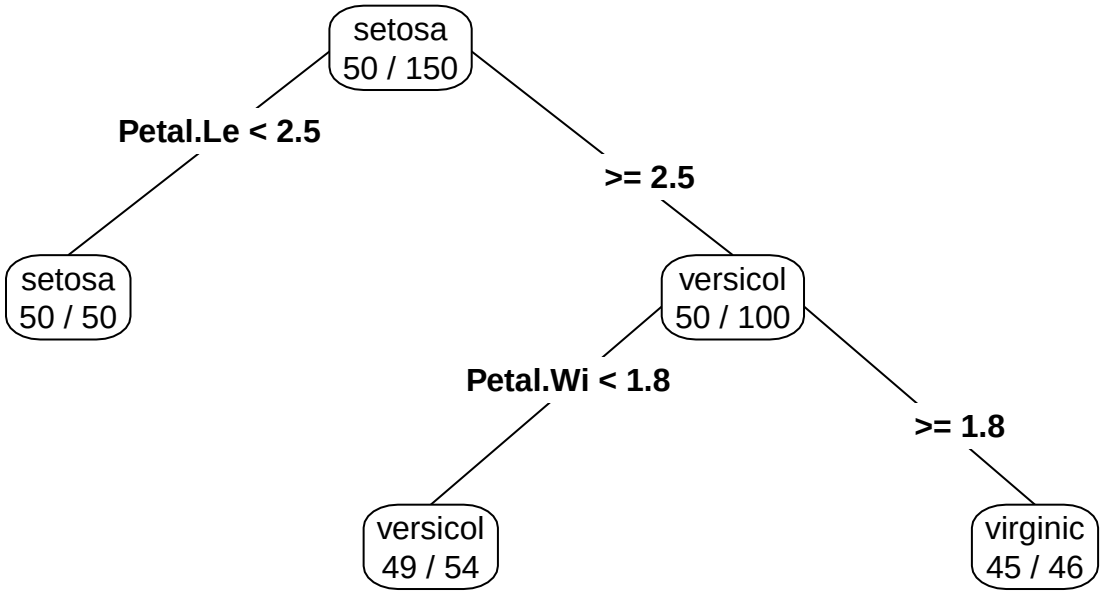
```
plot(c,compress=T,margin=0.3)
text(c,cex=1.5)
```



위에서부터 차례대로 1,2,3,6,7노드

```
library(rpart.plot)

prp(c,type=4,extra=2)
```



마찬가지, 조금 더 보기 친절한 그래프 ## Predict Classifier에서 predict는 Vote를 통하여 진행됨

```
head(predict(c,newdata = iris, type = "class"))
```

```
##      1      2      3      4      5      6
## setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

```
tail(predict(c,newdata = iris, type = "class"))
```

```
##       145       146       147       148       149       150
## virginica virginica virginica virginica virginica virginica
## Levels: setosa versicolor virginica
```

Cptable

cost-complexity parameter의 약자
Pruning과 트리의 최대 크기를 조절하는 옵션으로 사용됨

```
c$cptable
```

```
##     CP nsplit rel error xerror       xstd
## 1 0.50      0      1.00   1.17 0.05073460
## 2 0.44      1      0.50   0.71 0.06115009
## 3 0.01      2      0.06   0.11 0.03192700
```

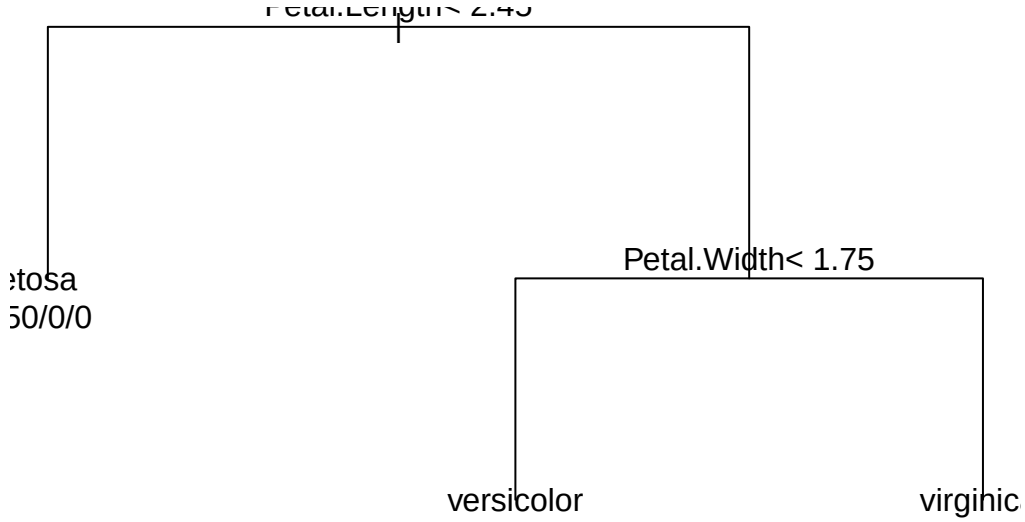
```
opt=which.min(c$cptable[,"xerror"])
opt
```

```
## 3
## 3
```

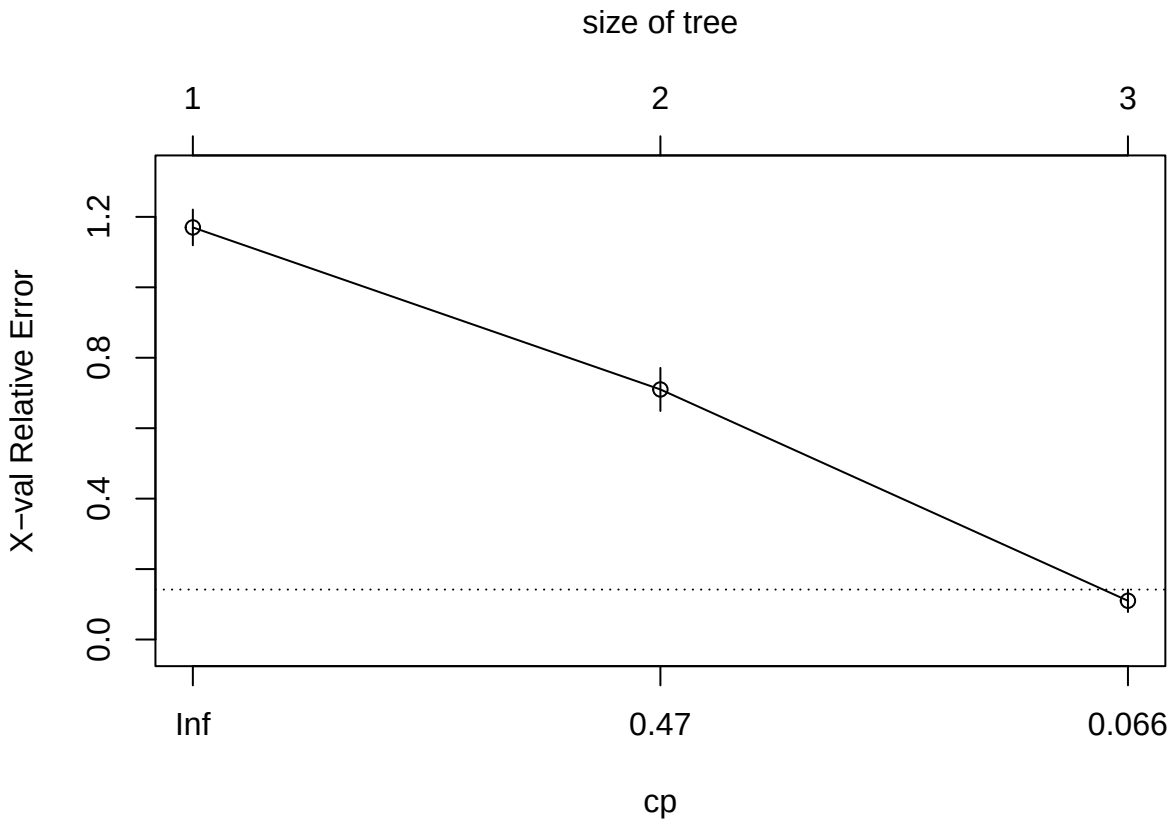
```
cp = c$cptable[opt,"CP"]
cp
```

```
## [1] 0.01
```

```
prune.c = prune(c,cp=cp)#pruning with opt
plot(prune.c)
text(prune.c,use.n=T)
```




```
plotcp(c)
```



y축 = X에 대한 error(Xerror)
X축(아래) = cp X축(위) = depth of the tree

ctree를 활용

카이제곱을 이용한 split을 진행함

```
library(party)
library(rpart)

data(stagec)
str(stagec)
```

```
## 'data.frame':    146 obs. of  8 variables:
##  $ pgtime : num  6.1 9.4 5.2 3.2 1.9 4.8 5.8 7.3 3.7 15.9 ...
##  $ pgstat : int  0 0 1 1 1 0 0 0 1 0 ...
##  $ age    : int  64 62 59 62 64 69 75 71 73 64 ...
##  $ eet    : int  2 1 2 2 2 1 2 2 2 2 ...
##  $ g2     : num  10.26 NA 9.99 3.57 22.56 ...
##  $ grade  : int  2 3 3 2 4 3 2 3 3 3 ...
##  $ gleason: int  4 8 7 4 8 7 NA 7 6 7 ...
##  $ ploidy : Factor w/ 3 levels "diploid","tetraploid",..: 1 3 1 1 2 1 2 3 1 2 ...
```

```r
#remove na
stagec1 = subset(stagec,!is.na(g2))
stagec2 = subset(stagec1,!is.na(gleason))
stagec3 = subset(stagec2,!is.na(eet))
str(stagec3)
```

```
## 'data.frame':    134 obs. of  8 variables:
##  $ pgtime : num  6.1 5.2 3.2 1.9 4.8 3.7 15.9 6.3 2.9 1.5 ...
##  $ pgstat : int  0 1 1 1 0 1 0 0 1 1 ...
##  $ age    : int  64 59 62 64 69 73 64 65 58 70 ...
##  $ eet    : int  2 2 2 2 1 2 2 2 2 2 ...
##  $ g2     : num  10.26 9.99 3.57 22.56 6.14 ...
##  $ grade  : int  2 3 2 4 3 3 3 3 4 3 ...
##  $ gleason: int  4 7 4 8 7 6 7 7 8 8 ...
##  $ ploidy : Factor w/ 3 levels "diploid","tetraploid",..: 1 1 1 2 1 1 2 2 2 1 ...
```

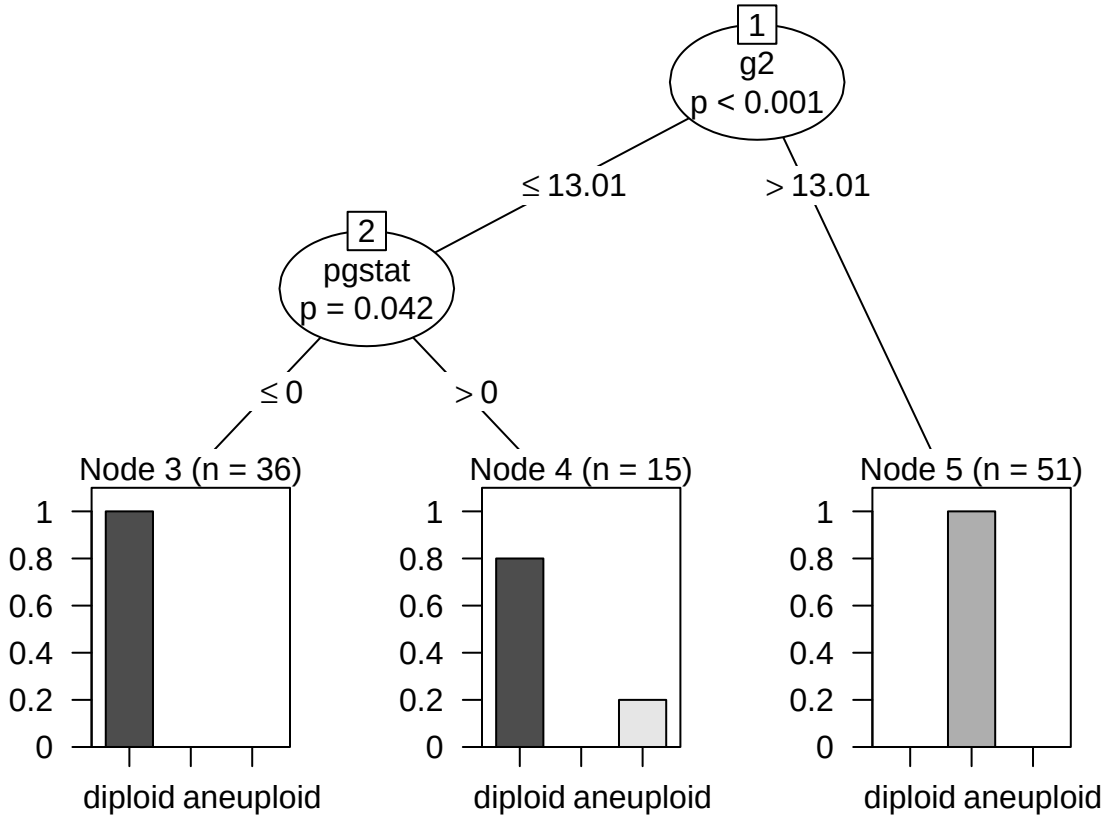
```r
#train_test split
set.seed(1234)
ind = sample(2,nrow(stagec3),replace=T,prob=c(0.7,0.3))#sample(number of group,total number...)

trainData = stagec3[ind==1,]
testData = stagec3[ind==2,]

tree = ctree(ploidy~.,data=trainData)
tree
```

```
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  ploidy
## Inputs:  pgtime, pgstat, age, eet, g2, grade, gleason
## Number of observations:  102
##
## 1) g2 <= 13.01; criterion = 1, statistic = 49.684
##   2) pgstat <= 0; criterion = 0.958, statistic = 7.5
##     3)*  weights = 36
##   2) pgstat > 0
##     4)*  weights = 15
## 1) g2 > 13.01
##   5)*  weights = 51
```

```r
plot(tree)
```



트리에 대한 해석은 같음 Weight = 노드 안의 표본 수 ## Predict

```
testPred = predict(tree,newdata = testData)
table(testPred,testData$ploidy)#Confusion Matrix
```

```
##
## testPred     diploid tetraploid aneuploid
##   diploid         17          0         1
##   tetraploid       0         13         1
##   aneuploid        0          0         0
```

대각선이 정확하게 분류한 것들이며 y축이 예측, x축이 실제값
=> 실제론 aneuploid이지만 모델은 diploid로 예측한 것이 한 개
실제론 aneuploid이지만 모델은 testraploid로 예측한 것이 한 개
즉 두 개밖에 틀리지 않음, 그렇다고 좋은 모델인가? => aneuploid는 총 표본 중 2개 밖에 차지하지 않기에 이 범주에 대한 분류는 다 틀렸음 => 단순히 정확도가 높다고 좋은 모델이 아님, 데이터의 상태에 따라 그 기준은 달라짐

연속형의 Tree

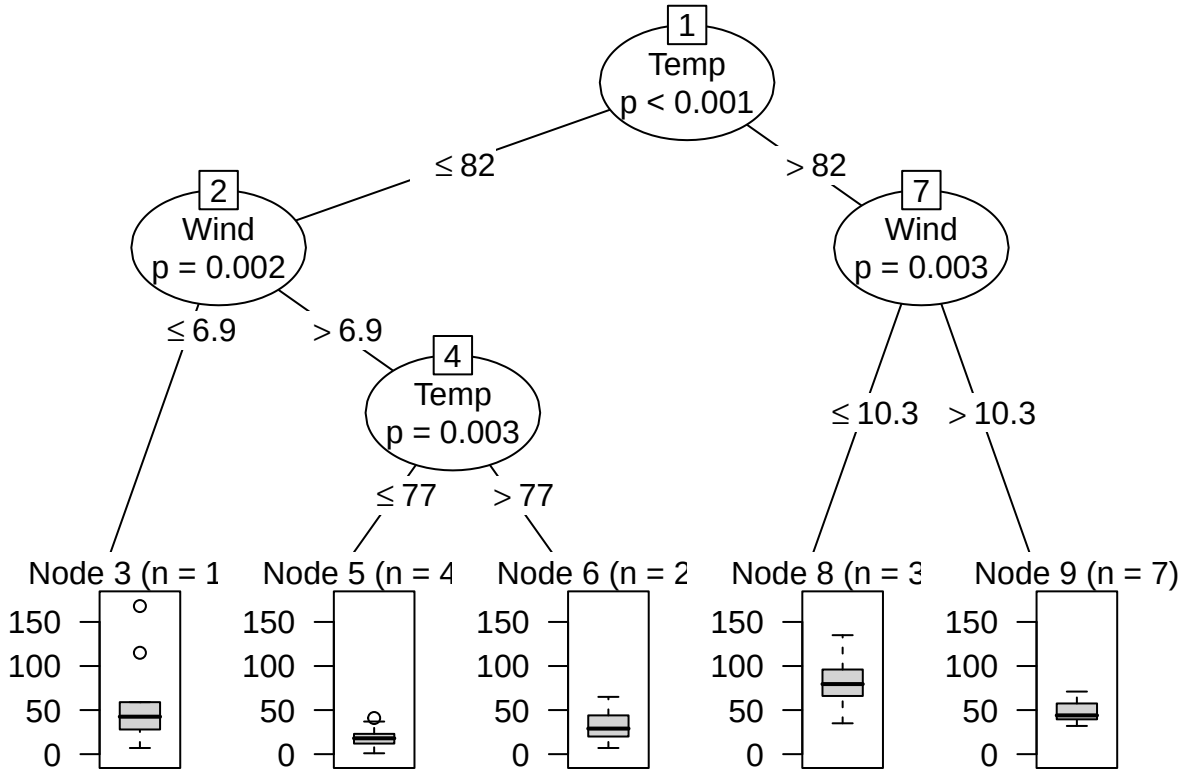
```
airq = subset(airquality,!is.na(Ozone))
head(airq)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 6    28      NA 14.9   66     5   6
## 7    23     299  8.6   65     5   7
```

```
airct = ctree(Ozone~.,data=airq)
airct
```

```
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  Ozone
## Inputs:  Solar.R, Wind, Temp, Month, Day
## Number of observations:  116
##
## 1) Temp <= 82; criterion = 1, statistic = 56.086
##   2) Wind <= 6.9; criterion = 0.998, statistic = 12.969
##     3)*  weights = 10
##   2) Wind > 6.9
##     4) Temp <= 77; criterion = 0.997, statistic = 11.599
##       5)*  weights = 48
##     4) Temp > 77
##       6)*  weights = 21
## 1) Temp > 82
##   7) Wind <= 10.3; criterion = 0.997, statistic = 11.712
##     8)*  weights = 30
##   7) Wind > 10.3
##     9)*  weights = 7
```

```
plot(airct)
```



predict 예측의 평균값이 반영
=> 같은 노드로 예측된 데이터는 같은 예측값을 가짐

```
head(predict(airct,data=airq))
```

```
##         Ozone
## [1,] 18.47917
## [2,] 18.47917
## [3,] 18.47917
## [4,] 18.47917
## [5,] 18.47917
## [6,] 18.47917
```

```
#We can show each data is in which node
predict(airct,data=airq,type="node")
```

```
##   [1] 5 5 5 5 5 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 6 3 5 6 9 9 6 5 5 5 5 5 8 8
##  [38] 6 8 9 8 8 8 8 5 6 6 3 6 8 8 9 3 8 8 6 9 8 8 8 6 3 6 6 8 8 8 8 8 8 9 6 6 5
##  [75] 3 5 6 6 5 5 6 3 8 8 8 8 8 8 8 8 8 8 9 6 6 5 5 6 5 3 5 5 3 5 5 5 6 5 5 6 5
## [112] 5 3 5 5 5
```

```
#mean Square Error
mean((airq$Ozone - predict(airct,data=airq))^2)
```

```
## [1] 403.6668
```