Regression Tree

3조 곽진주
박지은
임경룡

TABLE OF CONTENTS

1. 의사결정나무 간단 복습

Data

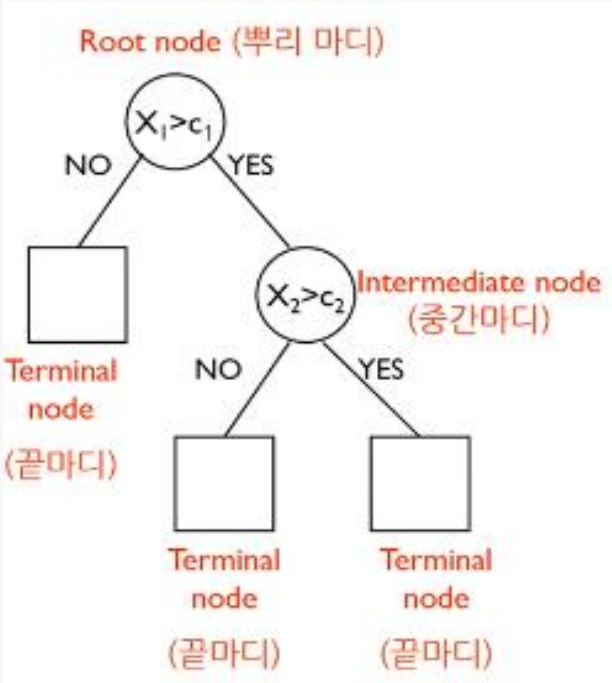
Input				Output
X_1	X_2	⋯	X_p	Y

Algorithm

데이터를 2개 혹은 그 이상의 부분집합으로 분할
→ 데이터가 균일해지도록 분할

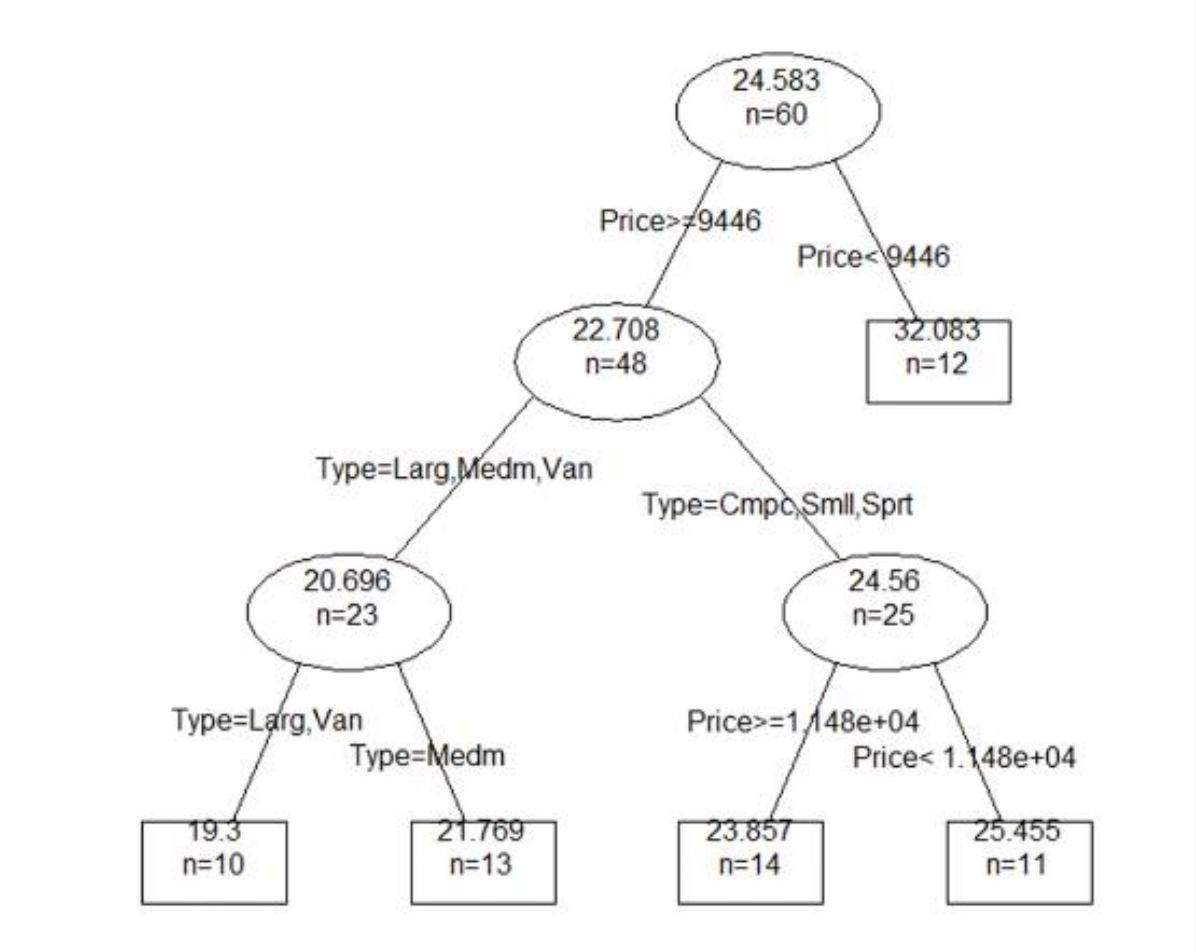
- 분류 : 비슷한 **범주**를 갖고 있는 관측치끼리 모음
- 예측 : 비슷한 수치를 갖고 있는 관측치끼리 모음

Model (Output)



2. 회귀트리 개요 - 이진분할

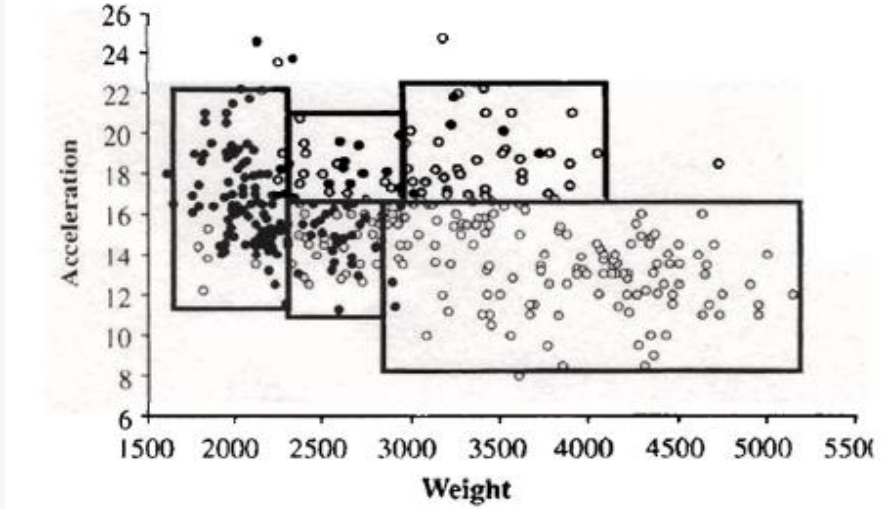
- 뿌리마디 = 맨 위
- 끝마디 = 5개
- 중간마디 = 3개

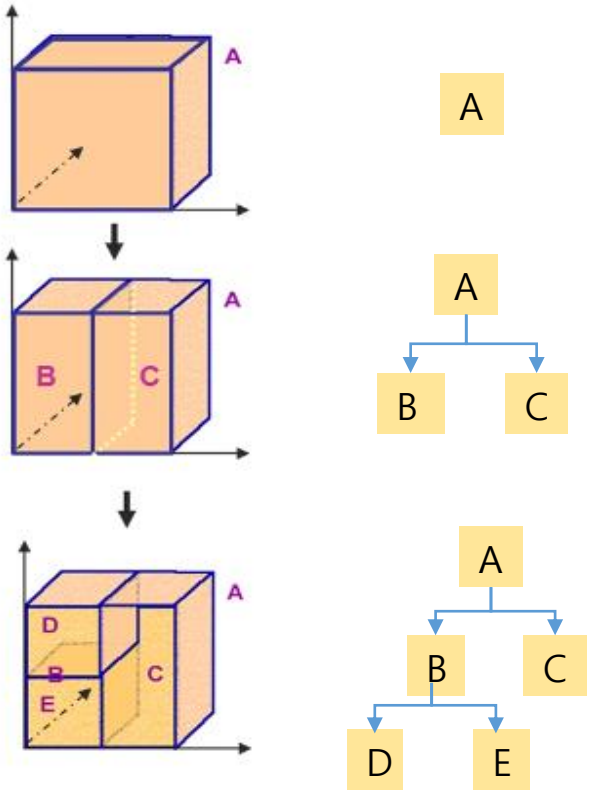


2. 회귀트리 개요 - 이진분할

< 2차원 >

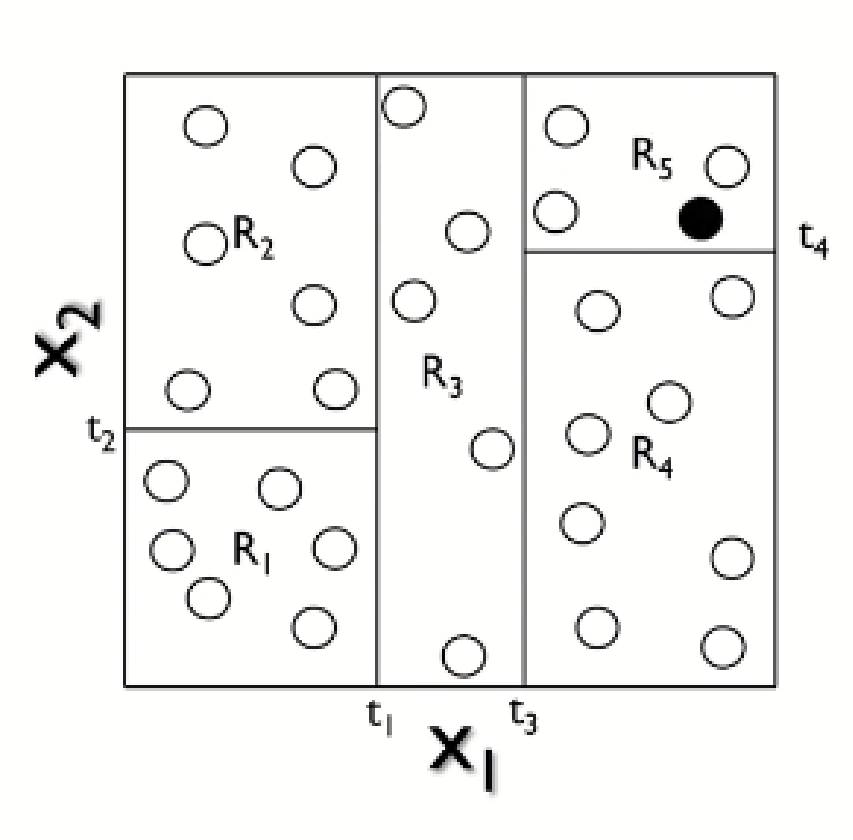
< 3차원 >

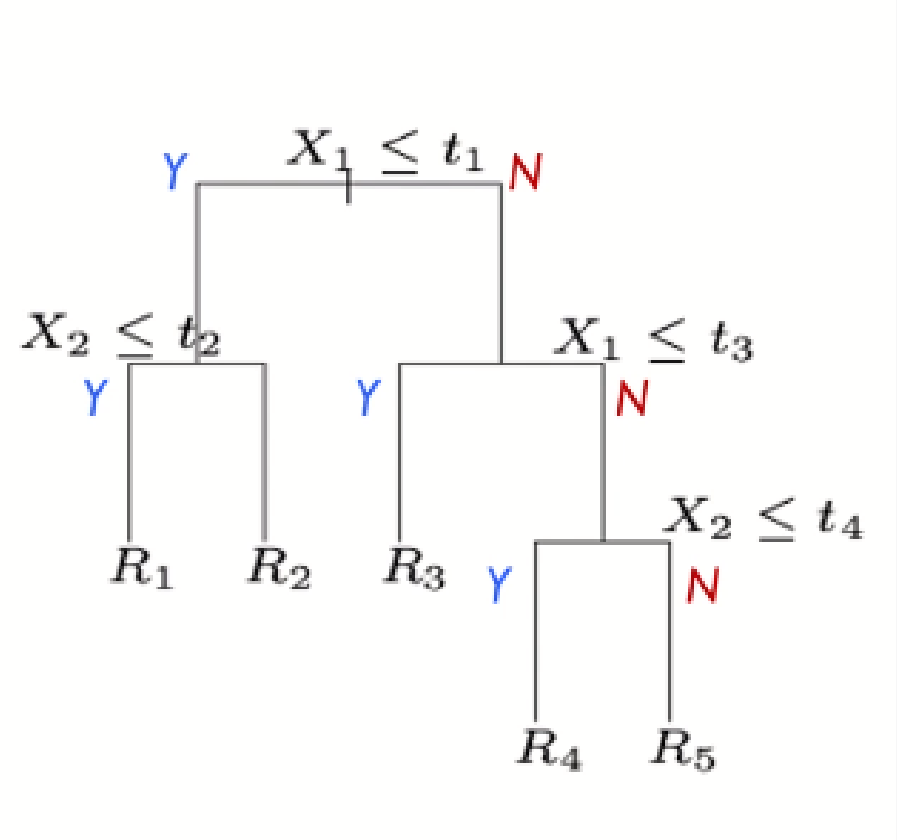




부분집합의 개수 = 끝마디의 개수

2. 회귀트리 개요 - 이진분할





부분집합의 개수 = 끝마디의 개수

함수의 형태로 알아보자!

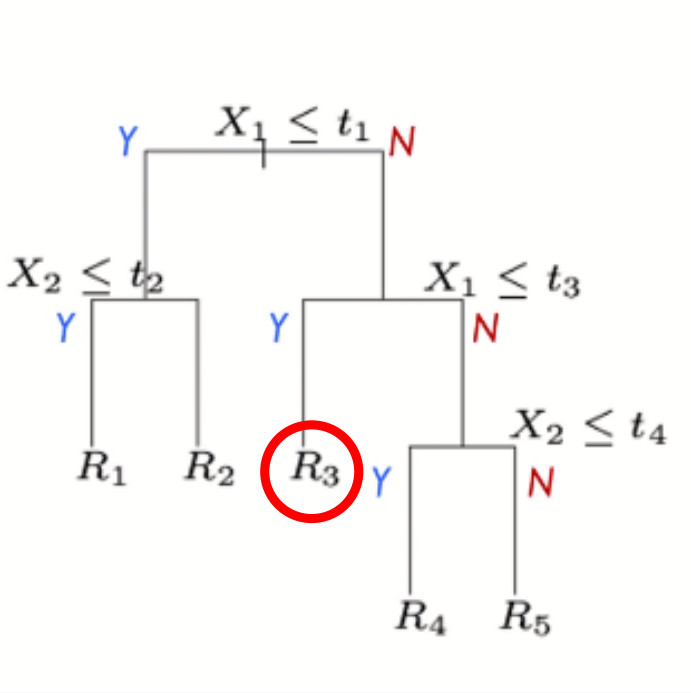
$$\hat{f}(x) = \sum_{m=1}^{5} c_m I\{(x_1, x_2) \in R_m\}$$

$$= c_1 I\{(x_1, x_2) \in R_1\} + c_2 I\{(x_1, x_2) \in R_2\} + c_3 I\{(x_1, x_2) \in R_3\} + c_4 I\{(x_1, x_2) \in R_4\} + c_5 I\{(x_1, x_2) \in R_5\}$$

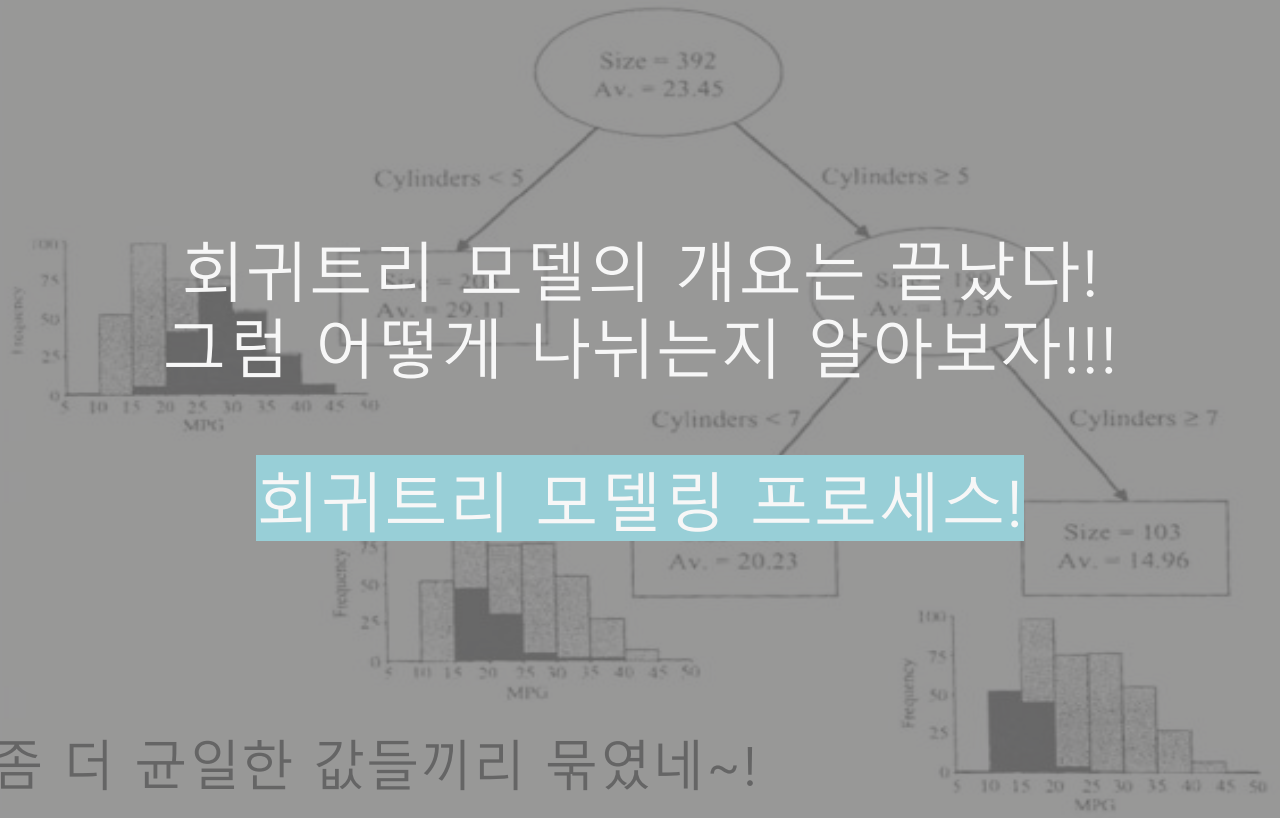
* c_m : 회귀나무모델로부터 얻어진 R_m 부분의 예측값
* $I\{\}$: Indicator function(지시 함수)

  특정 집합에 특정 값이 속하는지를 표시하는 함수로,

  특정 값이 집합에 속한다면 1, 속하지 않는다면 0의 값을 가짐



예시를 한 번 볼까?



회귀트리 모델의 개요는 끝났다!
그럼 어떻게 나뉘는지 알아보자!!!

회귀트리 모델링 프로세스!

좀 더 균일한 값들끼리 묶였네~!

3. 회귀트리 모델링 프로세스

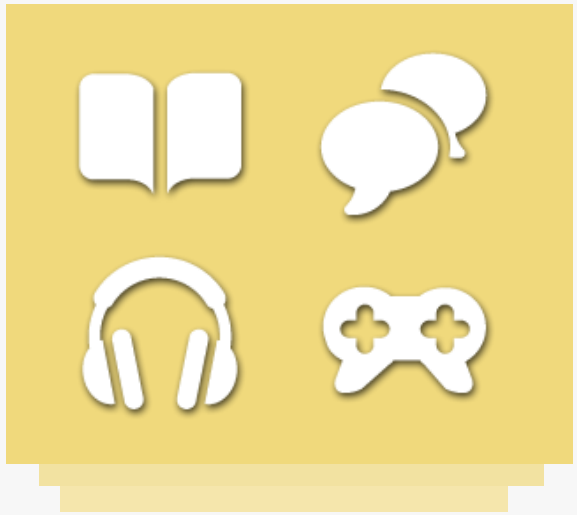
2. 분할변수와 분할점

분할변수와 분할점은 어떻게 결정되는 거지?

1. 비용함수

최상의 분할을 위한 비용함수 (cost function)의 최소화

각 분할에 속해 있는 y값들의 평균으로 예측했을 때 오류가 최소



3. Pruning

규제항을 통한 Pruning!

데이터를 M개로 분할 : $R_1, R_2, \ldots, R_M$

$$\hat{f}(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

$$RSS = \sum_{i=1}^{N} \left(y_i - \hat{f}(x_i)\right)^2$$

최상의 분할은 다음 비용함수(cost function)를 최소로 해야함

$$\min_{C_m} \sum_{i=1}^{N} \left(y_i - \hat{f}(x_i)\right)^2$$

$$= \min_{C_m} \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I(x \in R_m) \right)^2$$

$$\widehat{C_m} = \text{ave}(y_i | x_i \in R_m)$$

평균일 때 오류가 최소구나!
대표값으로 평균을 씀!

그렇다면 분할변수(j)와 분할점(s)은 어떻게 결정하지?

$$\boldsymbol{R_1(j,s) = \{x | x_j \leq s\}}$$
$$\boldsymbol{R_2(j,s) = \{x | x_j > s\}}$$

$$\underset{j,s}{argmin}\left[\min_{c_1}\sum_{\boldsymbol{x_i \epsilon R_1(j,s)}}\boldsymbol{(y_i - c_1)^2} + \min_{c_2}\sum_{\boldsymbol{x_i \epsilon R_2(j,s)}}\boldsymbol{(y_i - c_2)^2}\right]$$

$$= \underset{j,s}{argmin}\left[\sum_{\boldsymbol{x_i \epsilon R_1(j,s)}}\boldsymbol{(y_i - \widehat{c_1})^2} + \sum_{\boldsymbol{x_i \epsilon R_2(j,s)}}\boldsymbol{(y_i - \widehat{c_2})^2}\right]$$

* $\widehat{c_1} = \text{ave}(y_i | x_i \in \boldsymbol{R_1(j,s)})$ and $\widehat{c_1} = \text{ave}(y_i | x_i \in \boldsymbol{R_2(j,s)})$

Greedy 알고리즘

⇒ 실행되는 단계만 생각해서 RSS를 최소화

⇒ 미래의 어떠한 최적의 조합이 나오는지에 대한 고려는 X

분할변수(j)와 분할점(s) 풀어보자!

x_1	x_2	x_3
2	5	1
3	6	7

$$R_1(j,s) = \{x | x_j \leq s\}$$
$$R_2(j,s) = \{x | x_j > s\}$$

$$\underset{j,s}{argmin} \left[\min_{c_1} \sum_{x_i \epsilon R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \epsilon R_2(j,s)} (y_i - c_2)^2 \right]$$

$$= \underset{j,s}{argmin} \left[\sum_{x_i \epsilon R_1(j,s)} (y_i - \widehat{c_1})^2 + \sum_{x_i \epsilon R_2(j,s)} (y_i - \widehat{c_2})^2 \right]$$

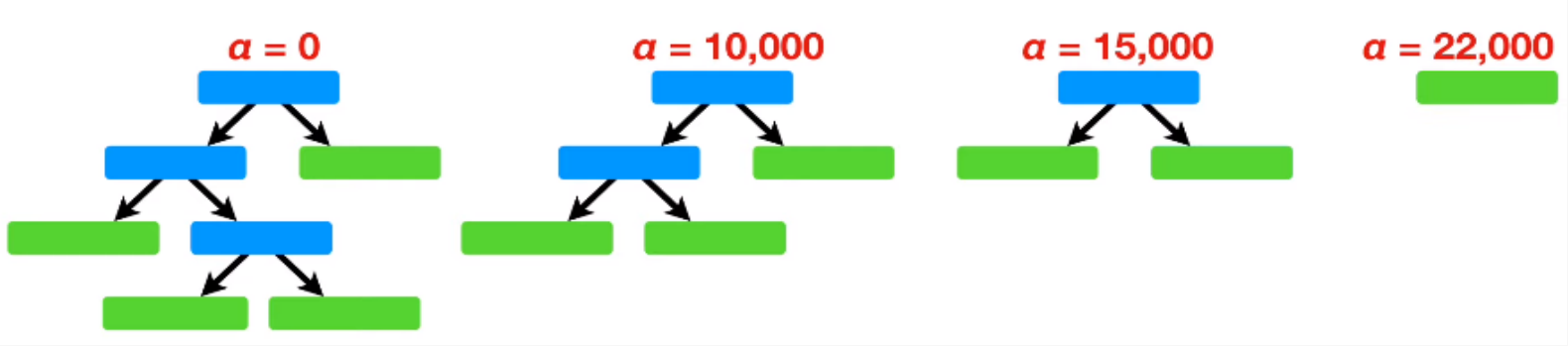
Pruning이 무엇이고 왜 해주는 거지?

⇒ 과적합을 방지하기 위해, 트리를 가지치기 하는 과정!

⇒ 본래의 cost function인 RSS에 *패널티를 주는 것! = Tree Complexity Penalty

$$C_\alpha(T) = RSS + \alpha|T|$$

Cost Complexity Pruning



* 패널티 : 모델의 복잡성을 T트리(끝노드 개수)의 크기 텀을 넣어서 그 값을 줄이는 것! (이 영향은 α 가 결정)

1. 계층적 구조로 인해 중간에 에러가 발생하면 다음 단계로 에러가 계속 전파

2. 학습 데이터의 미세한 변동에도 최종 결과에 크게 영향

3. 적은 개수의 노이즈에도 크게 영향

개별 트리 모델의 단점을 보완한 방법?

→ Random Forest

4. 나무의 최종노드 개수를 늘리면 과적합 위험

예제 설명

- **목표 : 회귀 트리와** 모델 **트리를 사용한 대구 지역 아파트 가격 추정**

- **데이터 : 매매 가격에 영향을 미치는 건축 연도, 면적 등 아파트 기본 정보, 내부 및 근린생활시설에 대한 29가지 정보와**
  매매 가격을 담은 총 5,891개의 아파트 표본
  (SalePrice : 매매 가격 / YearBuilt : 건축 연도 / YrSold : 매매일(연도) / MonthSold : 매매일(월)
  Size.sqf : 면적(square foot, 평방 피트) / Floor : 층 / … / N_SchoolNearBy.Total. : 근처 학교 총
- 계)

- 목표변수 : **"SalePrice" (단위: 만 원(추측))**

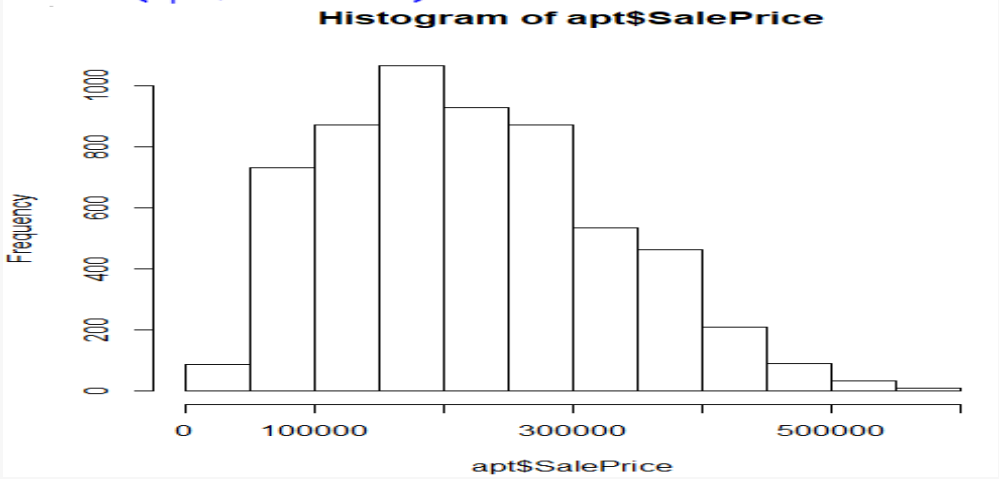
모델링 과정 :
1. 데이터 탐색 및 training/test data 분할
2. 회귀 트리 모델 생성 (CART algorithm)
3. 가지치기(pruning)
4. 모델 성능 평가 (using MAE)
5. 모델 성능 향상 – 모델 트리 (M5P algorithm)

4. R 예제 실습 – (1) 데이터 탐색 및 분할

```
## Example : Predicting Price of Real Estate(using regression and model tree)
## Step1. Exploring and Splitting the data
apt <- read.csv("Daegu_Real_Estate_data.csv")
str(apt)
summary(apt)

options("scipen" = 100, "digits" = 10)   # change exponential notation to fixed notation
# format(apt$SalePrice, scientific = F) : limited local var
hist(apt$SalePrice)

# Splitting the data into train and test set
library(caret)

set.seed(408)

train_idx <- createDataPartition(apt$SalePrice, p = .7, list = F)
apt_train <- apt[train_idx,]
apt_test <- apt[-train_idx,]

hist(apt_train$SalePrice)
hist(apt_test$SalePrice)
```

```
> str(apt)
'data.frame':   5891 obs. of  30 variables:
 $ SalePrice                    : int  141592 51327 48672 380530 221238 35840 78318 61946 84070 83185 ...
 $ YearBuilt                    : int  2006 1985 1985 2006 1993 1992 1992 1993 1993 1992 ...
 $ YrSold                       : int  2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
 $ MonthSold                    : int  8 8 8 8 8 8 8 8 8 8 ...
 $ Size.sqf.                    : int  814 587 587 2056 1761 355 644 644 644 644 ...
 $ Floor                        : int  3 8 6 8 3 5 2 10 3 13 ...

 $ N_SchoolNearBy.High.         : int  2 1 1 1 5 5 4 5 5 4 ...
 $ N_SchoolNearBy.University.   : int  2 0 0 2 5 5 4 5 5 4 ...
 $ N_FacilitiesInApt            : int  5 3 3 5 4 3 3 4 4 3 ...
 $ N_FacilitiesNearBy.Total.    : int  6 12 12 3 14 16 9 14 14 9 ...
 $ N_SchoolNearBy.Total.        : int  9 4 4 7 17 17 14 17 17 14 ...
```

```
> hist(apt$SalePrice)
```




```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 32743  144247  207964  221218  291150  585840
```

목표 변수의 분포

- 트리 모델은 전처리가 필요 없으나 보다 더 나은 모델 성능을
  위해 목표 변수의 분포를 살펴봄
  (예를 들어 아파트 간의 가격에서 별 차이가 없거나 양봉 분포일 경우 모델에 대해 문제가 됨)
- 아파트 가격의 분포는 왼쪽으로 다소 치우쳐져 있으나 대체적으로 종 모양의 분포를 보여 모델링하기 괜찮은 데이터로 보임

의사결정나무의 특징

기계 학습 모델의 다른 종류와 비교해, 의사결정나무의 장점 중 하나는 전처리(Pre-processing) 없이 많은 데이터 타입을 다룰 수 있는데, 이는 속성의 정규화나 표준화가 필요 없음을 의미함

정규화/표준화 ?

- 데이터 스케일링(; 크기 조정)(Data Scaling)
  - 속성별 단위(scale)가 다를 경우 속성의 영향이 제대로 반영되기 어렵기 때문에 속성의 크기를 변환하여 상대적 크기를 같게 함

- 정규화(Normalization)
  - 최소-최대 정규화(min-max normalization)
  - 모든 속성의 값을 0과 1사이 범위의 값으로 변환

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

(min-max normalization)

- 표준화(Standardization)
  - z 점수 표준화(z score standardization)
  - 정규 분포의 특성을 기반으로 하여 평균에서 얼마나 떨어져 있는지의 관점에서 각 속성의 값을 조정함

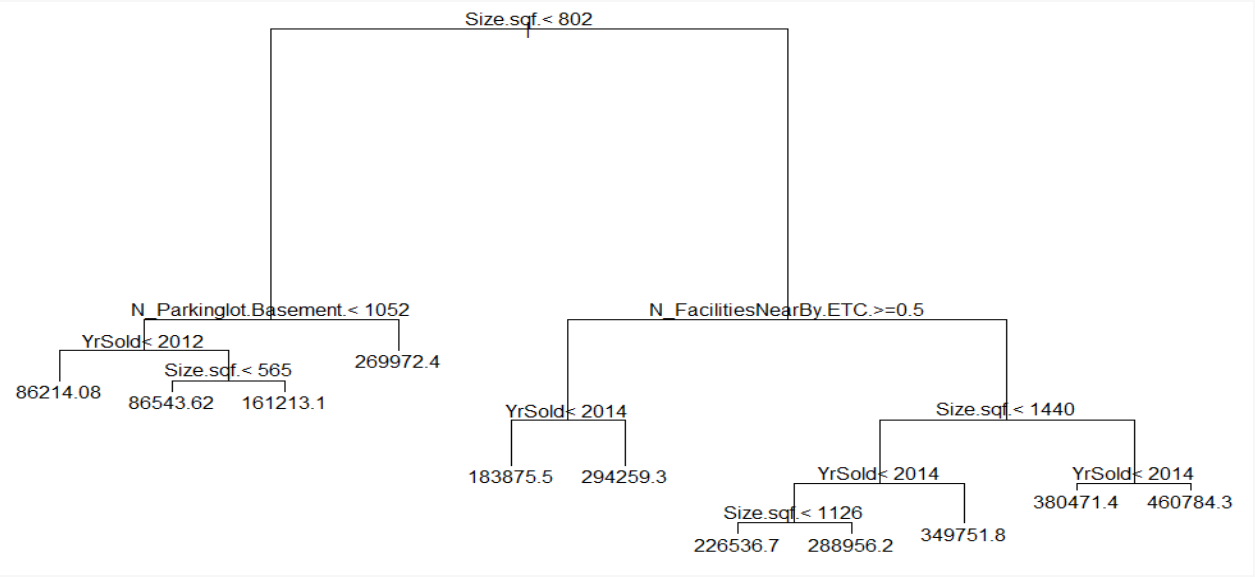
$$x_{new} = \frac{x - \mu}{\sigma}$$

(z score standardization)

4. R 예제 실습 – (2) 회귀 트리 모델 생성

rpart(dv ~ iv, data = , subset = , na.action = , *control = rpart.control(), ...)

```
##Step2. Training a model on the data
# Regression tree using rpart
library(rpart)

set.seed(408)

apt_reg.tr <- rpart(SalePrice ~ ., data = apt_train)
apt_reg.tr
summary(apt_reg.tr)

plot(apt_reg.tr); text(apt_reg.tr)
```



control = rpart.control() : 모델의 파라미터 조정

- minsplit : 분할을 위해 splitting node에 있어야 하는 관측치 개수의 최솟값
- minbucket : leaf node에 있어야 하는 관측치 개수의 최솟값
- cp : complexity parameter.
  각 노드에 필요한 모델의 최소한의 향상도
- maxcompete : 결과에 남아있는 경쟁 분할의 수
  (어떤 분할 기준이 선택되었는지 알 수 있음)
- maxsurrogate : 결과에 남아있는 대리 분할의 수
- usesurrogate : 분할 과정에서 대리(분할)을 사용하는 방법
  (분할 규칙에서 누락된 관찰값에 대한 처리)
- Xval : 교차검증의 수
- Surrogatestyle : 최선의 대리(분할) 선택을 통제
- maxdepth : root node에서 leaf node까지의 최대 깊이
- ...
- 추가 설명은 ? rpart.control 참고

4. R 예제 실습 – (2) 회귀 트리 모델 생성

```
> apt_reg.tr
n= 4125

node), split, n, deviance, yval
      * denotes terminal node

 1) root 4125 47482650000000 221411.90
   2) Size.sqf.< 802 1329  4703525000000 122626.00
     4) N_Parkinglot.Basement.< 1052 1257  3017543000000 114186.10
       8) YrSold< 2012.5 603   602097200000  86214.08 *
       9) YrSold>=2012.5 654  1508622000000 139976.90
        18) Size.sqf.< 565 186    83311550000  86543.62 *
        19) Size.sqf.>=565 468   683201300000 161213.10 *
     5) N_Parkinglot.Basement.>=1052 72    33253870000 269972.40 *
   3) Size.sqf.>=802 2796 23645300000000 268367.00
     6) N_FacilitiesNearBy.ETC.>=0.5 1242  5497780000000 214093.30
      12) YrSold< 2014.5 902  1458807000000 183875.50 *
      13) YrSold>=2014.5 340  1030302000000 294259.30 *
     7) N_FacilitiesNearBy.ETC.< 0.5 1554 11565070000000 311744.10
      14) Size.sqf.< 1440.5 1167  5915196000000 282002.00
        28) YrSold< 2014.5 782  1920277000000 248646.90
          56) Size.sqf.< 1126 505   745861400000 226536.70 *
          57) Size.sqf.>=1126 277   477460100000 288956.20 *
        29) YrSold>=2014.5 385  1357731000000 349751.80 *
      15) Size.sqf.>=1440.5 387  1504588000000 401431.60
        30) YrSold< 2014.5 286   764020600000 380471.40 *
        31) YrSold>=2014.5 101   259121800000 460784.30 *
```

결과 해석

- 총 4125개의 표본이 root node에서 시작해서 1329개가 2) Size.sqf. < 802, 나머지 2796개가 3) Size.sqf. >= 802에 해당됨
- **Size.sqf.는 트리의 첫 분할 기준으로 사용되었기 때문에 이 속성은 아파트 가격의 가장 중요한 요소**(=예측기)라고 할 수 있음
- yval은 node의 예측값을 의미함. 예를 들어 node 4는 Size.sqf. < 802이고 N_Parkinglot.Basement. < 1,052인 아파트 표본의 가격을114,186.10으로 예측함
- *로 표시된 node는 terminal(leaf) node를 의미함
  >> 이 회귀 트리 모델은 11개의 terminal node를 가짐

4. R 예제 실습 – (2) 회귀 트리 모델 생성

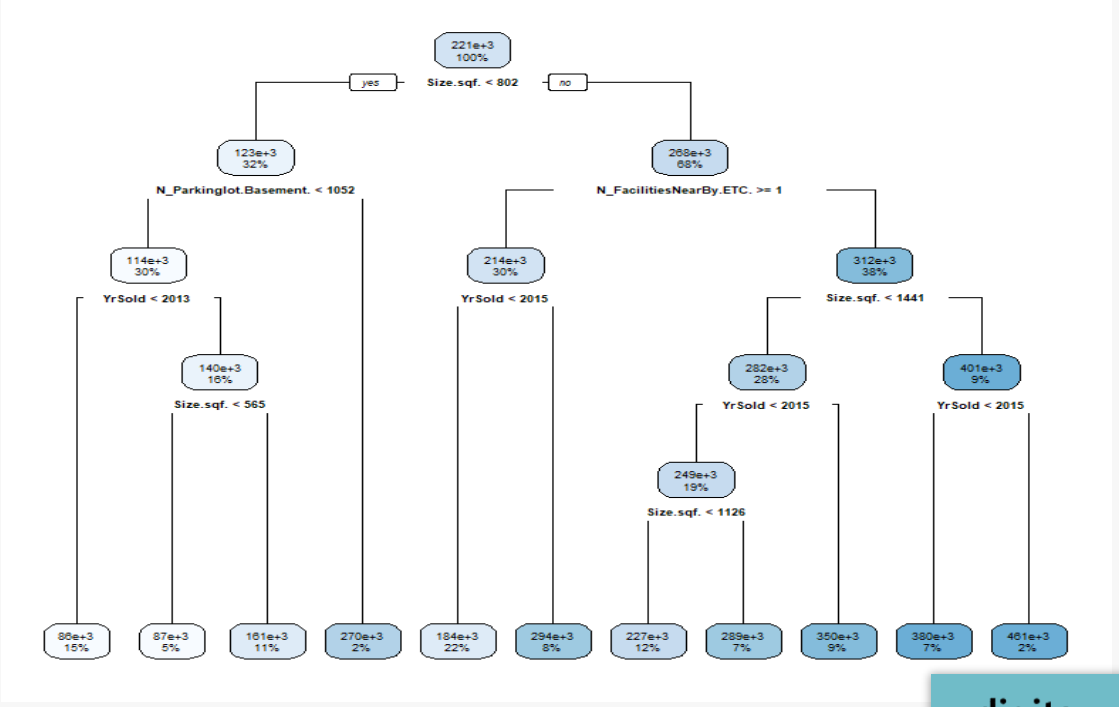
```
> summary(apt_reg.tr)
Call:
rpart(formula = SalePrice ~ ., data = apt_train)
  n= 4125

           CP nsplit rel error    xerror        xstd
1  0.40296452      0 1.0000000 1.0008755 0.019924226
2  0.13862847      1 0.5970355 0.5979061 0.013472310
3  0.08730116      2 0.4584070 0.4594409 0.010602563
4  0.06336358      3 0.3711059 0.3724621 0.008750269
5  0.05554005      4 0.3077423 0.3107987 0.007617929
6  0.03480699      5 0.2522022 0.2569220 0.006521248
7  0.01909801      6 0.2173952 0.2220653 0.005545032
8  0.01562906      7 0.1982972 0.2029585 0.005281590
9  0.01467811      8 0.1826682 0.1912042 0.005165915
10 0.01013940      9 0.1679901 0.1730566 0.004877430
11 0.01000000     10 0.1578507 0.1679653 0.004763389

Variable importance
                      Size.sqf.                 N_FacilitiesInApt                           YrSold
                             23                                 7                                7
        N_FacilitiesNearBy.ETC.                       HallwayType                        YearBuilt
                              7                                 6                                6
     N_SchoolNearBy.University.                      TimeToSubway       N_FacilitiesNearBy.Total.
                              5                                 5                                5
                 SubwayStation            N_SchoolNearBy.Total. N_FacilitiesNearBy.PublicOffice.
                              5                                 5                                5
                         N_APT              N_SchoolNearBy.High.             N_Parkinglot.Basement.
                              4                                 3                                2
                   N_elevators            N_SchoolNearBy.Middle.
                              2                                 1
Node number 2: 1329 observations,    complexity param=0.03480699
  mean=122626, MSE=3.539146e+09
  left son=4 (1257 obs) right son=5 (72 obs)
  Primary splits:
      N_Parkinglot.Basement. < 1052   to the left,  improve=0.3513807, (0 missing)
      N_FacilitiesInApt      < 8      to the left,  improve=0.3513807, (0 missing)
      N_APT                  < 9      to the left,  improve=0.3513807, (0 missing)
      YrSold                 < 2012.5 to the left,  improve=0.3111498, (0 missing)
      Size.sqf.              < 611.5  to the left,  improve=0.2808787, (0 missing)
  Surrogate splits:
      N_APT                              < 9      to the left,  agree=1.000, adj=1.000, (0 split)
      N_FacilitiesInApt                  < 8      to the left,  agree=1.000, adj=1.000, (0 split)
      N_elevators                        < 22     to the left,  agree=0.964, adj=0.333, (0 split)
      N_FacilitiesNearBy.PublicOffice. < 1.5    to the right, agree=0.964, adj=0.333, (0 split)
      YearBuilt                          < 2014.5 to the left,  agree=0.958, adj=0.222, (0 split)
```

결과 해석

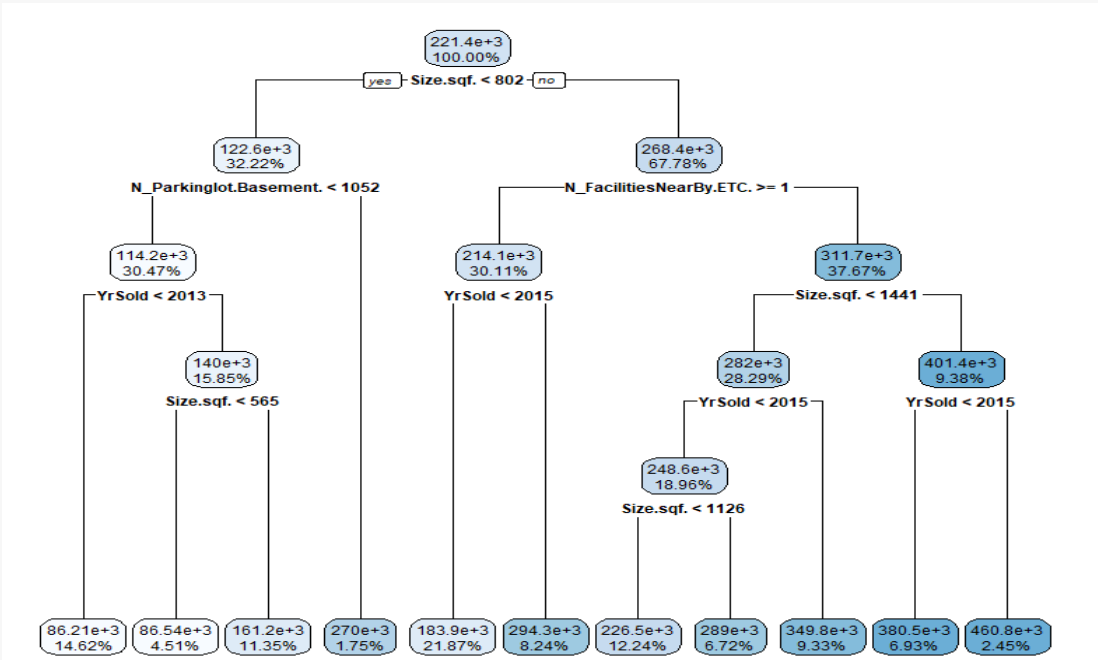
- Variable importance는 각 속성(변수) 중요도를 보여주며 앞의 결과와 마찬가지로 Size.sqf.가 가장 중요한 속성임을 알 수 있음
- Node 2에서의 표본은 1,329개, CP는 0.03, 아파트 가격을 122,626으로 예측(=node 2에 해당하는 표본 1,329개의 평균)
- node 2에서 다시 node 4(left)와 node 5(right)로 분할되며 분할 기준이 되는 변수는 N_Parkinglot.Basement.로, N_Parkinglot.Basement. < 1052이면 node 4(나머지는 node 5)에 해당됨

4. R 예제 실습 – (2) 회귀 트리 모델 생성

```
# Visualization for decision tree
library(rpart.plot)   # ref) "http://www.milbo.org/rpart-plot/"

rpart.plot(apt_reg.tr, digits = 2)
rpart.plot(apt_reg.tr, digits = 3, fallen.leaves = F, type = 4, extra = 101)
```

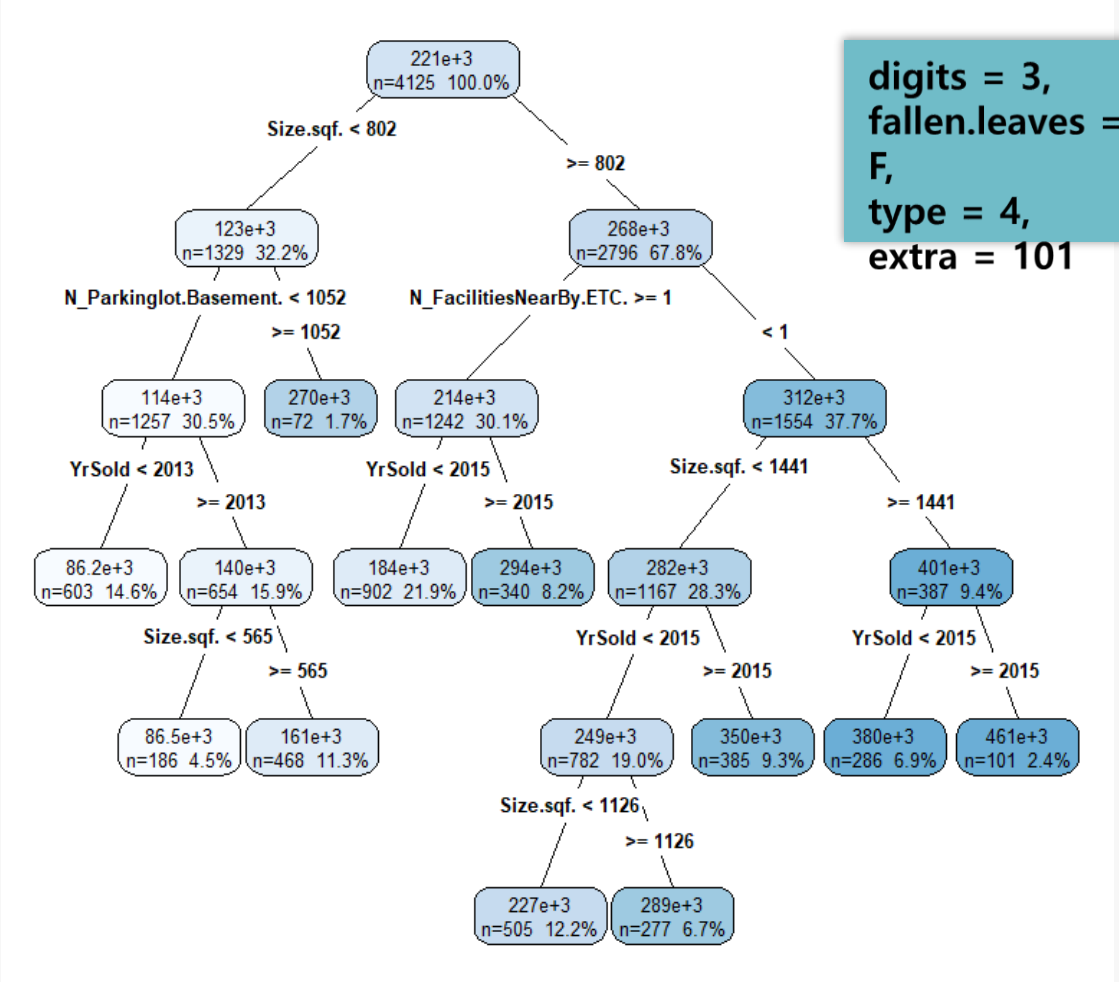


digits = 1



digits = 4

4. R 예제 실습 – (2) 회귀 트리 모델 생성



digits = 3,
fallen.leaves = F,
type = 4,
extra = 101

- digits : 값의 소수점 자릿수 조정 (-22 ~ 22)
- fallen.leaves : leaf node의 도식 하단 표기 여부 (T/F)
- type : 라벨된 node의 표기 방식 (0 ~ 5)
- extra : 추가 정보 제공 여부 (0 ~ 10(+100))

0 No extra information (only yval)
1 Number of observations in the node
2 Class models: Classification rate
(# of correct / # of observations)
Poisson and exp models: number of events
3 Class models: Misclassification rate
4 Class models: Probability per class
5 Class models: Like 4 but don't display the fitted class
6 Class models: Probability of second class only
7 Class models: Like 6 but don't display the fitted class
8 Class models: Probability of the fitted class
9 Class models: Probability relative to all observations
10 Class models: like 9 but display the probability of the
second class only

add 100 means to also display the percentage of observations in the node

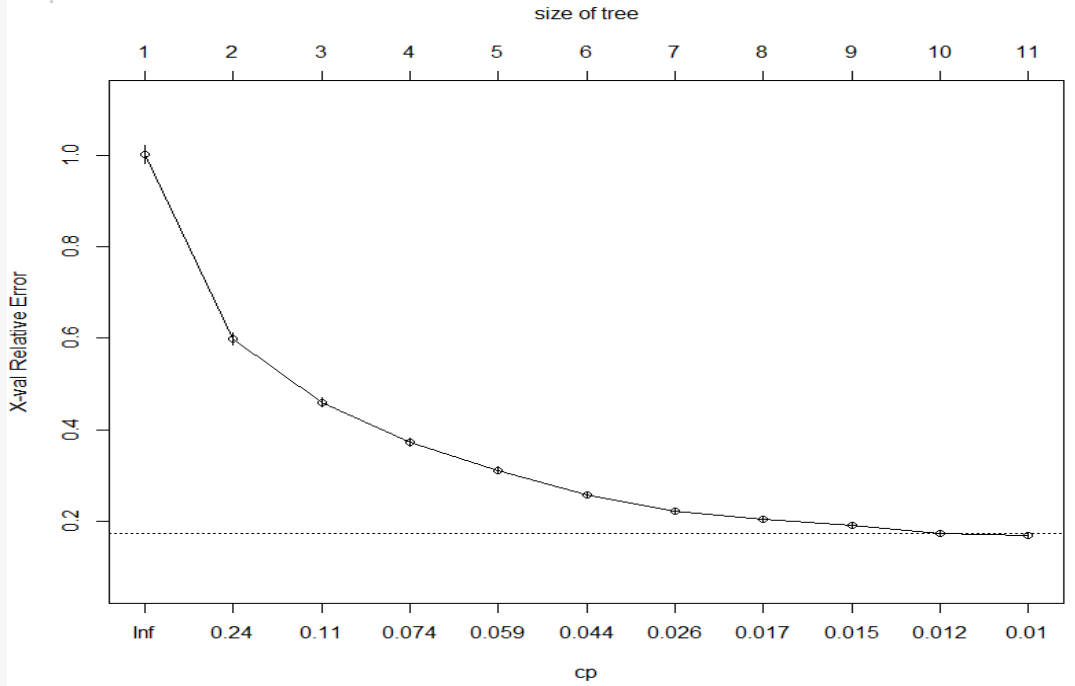
4. R 예제 실습 – (3) 가지치기

```
## Step3. Pruning the tree (for avoid overfitting)
printcp(apt_reg.tr)   # or apt_reg.tr$cptable
plotcp(apt_reg.tr)
apt_reg_prune <- prune.rpart(apt_reg.tr, cp = apt_reg.tr$cptable[8, "CP"])

rpart.plot(apt_reg_prune, digits = 3, fallen.leaves = F, type = 4, extra = 101)
```

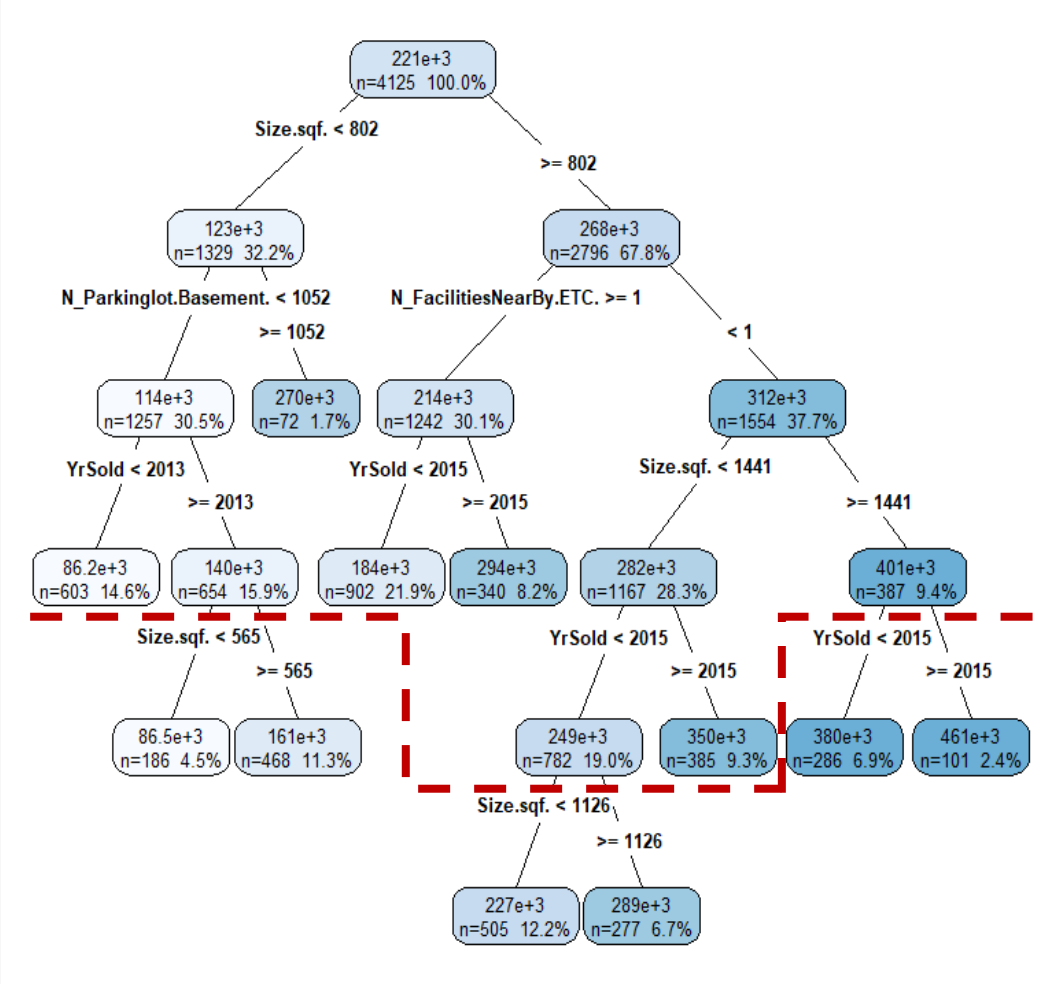
```
> printcp(apt_reg.tr)    # or apt_reg.tr$cptable

Regression tree:
rpart(formula = SalePrice ~ ., data = apt_train)

Variables actually used in tree construction:
[1] N_FacilitiesNearBy.ETC. N_Parkinglot.Basement.  Size.sqf.
[4] YrSold

Root node error: 47482650486588/4125 = 11510945573

n= 4125

          CP nsplit rel error  xerror      xstd
1   0.402965      0   1.00000 1.00088 0.0199242
2   0.138628      1   0.59704 0.59791 0.0134723
3   0.087301      2   0.45841 0.45944 0.0106026
4   0.063364      3   0.37111 0.37246 0.0087503
5   0.055540      4   0.30774 0.31080 0.0076179
6   0.034807      5   0.25220 0.25692 0.0065212
7   0.019098      6   0.21740 0.22207 0.0055450
8   0.015629      7   0.19830 0.20296 0.0052816
9   0.014678      8   0.18267 0.19120 0.0051659
10  0.010139      9   0.16799 0.17306 0.0048774
11  0.010000     10   0.15785 0.16797 0.0047634
```

- CP(Complexity Parameter)
  : 결정 트리의 크기를 조정하고 최적의 트리 크기를 선택하는데 쓰임.
  값이 작을수록 모델의 복잡도 ↑
- nsplit : 가지의 분기 수 (nsplit+1만큼의 leaf node 생성)
- rel error : 오류율 (=1-R^2)
- xerror : 교차검증 오류율
- xstd : 교차검증 오류의 표준오차

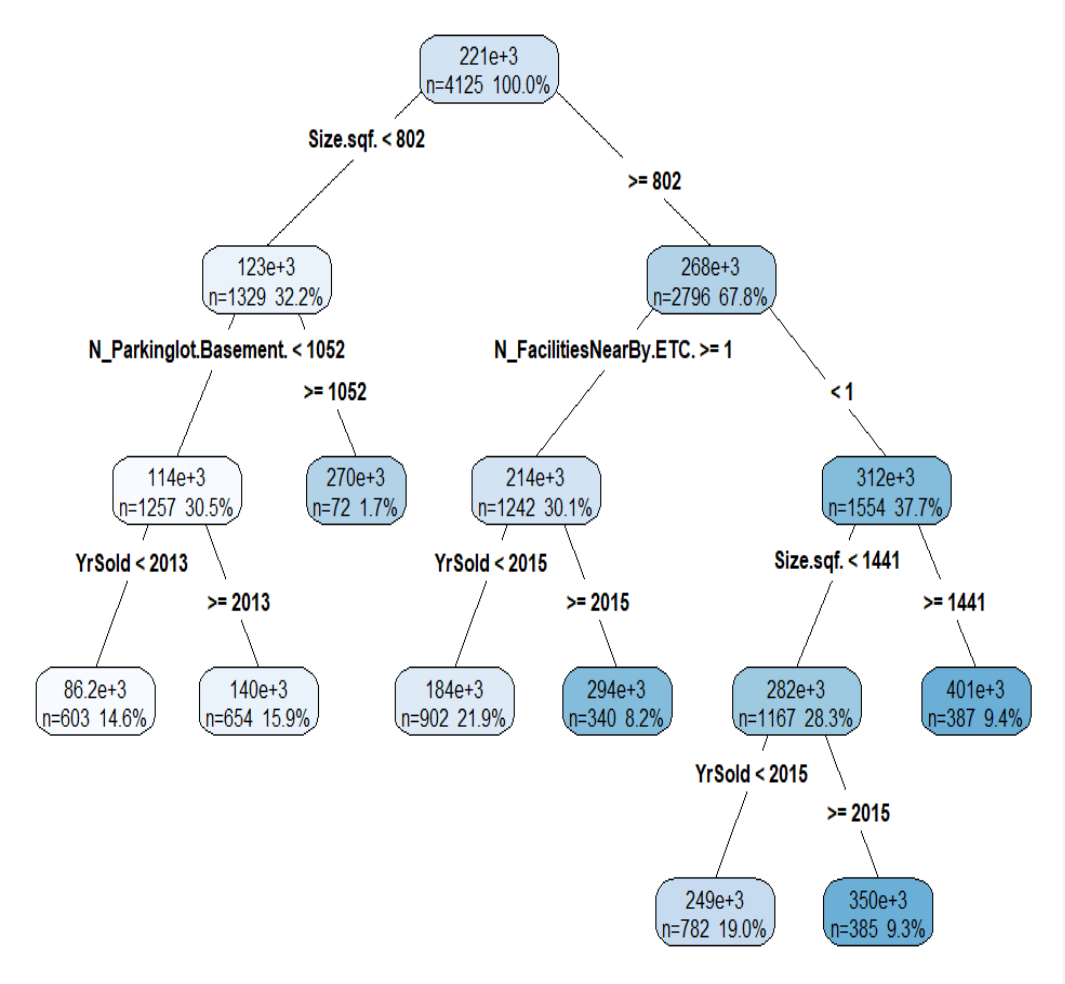
```
> plotcp(apt_reg.tr)
```



4. R 예제 실습 – (3) 가지치기



pruning



4. R 예제 실습 – (4) 모델 성능 평가

```
## Step4. Evaluating model performance (using MAE)
apt_reg_pred <- predict(apt_reg.tr, apt_test)
summary(apt_reg_pred)
summary(apt_test$SalePrice)

cor(apt_reg_pred, apt_test$SalePrice)
```

```
> summary(apt_reg_pred)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  86214  161213  183876  221560  294259  460784
> summary(apt_test$SalePrice)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  34513  144373  207964  220765  291039  570796
> cor(apt_reg_pred, apt_test$SalePrice)
[1] 0.916737
```

```
# Function to calculate the mean absolute error(MAE)
MAE <- function(actual, predicted){
  mean(abs(actual - predicted))
}

MAE(apt_reg_pred, apt_test$SalePrice)

# Comparing orig.regression tree with pruned tree & training data
apt_reg_prune_pred <- predict(apt_reg_prune, apt_test)

cor(apt_reg_prune_pred, apt_test$SalePrice)   # not pruned : 0.917
MAE(apt_reg_prune_pred, apt_test$SalePrice)   #            : 32,035.81

mean(apt_train$SalePrice)   # result = 221280
MAE(221280, apt_test$SalePrice)
```

```
> MAE(apt_reg_pred, apt_test$SalePrice)
[1] 32035.81
> cor(apt_reg_prune_pred, apt_test$SalePrice)
[1] 0.8924752
> MAE(apt_reg_prune_pred, apt_test$SalePrice)
[1] 35798.62
> mean(apt_train$SalePrice)   # result = 221280
[1] 221411.9
> MAE(221280, apt_test$SalePrice)
[1] 85228.96
```

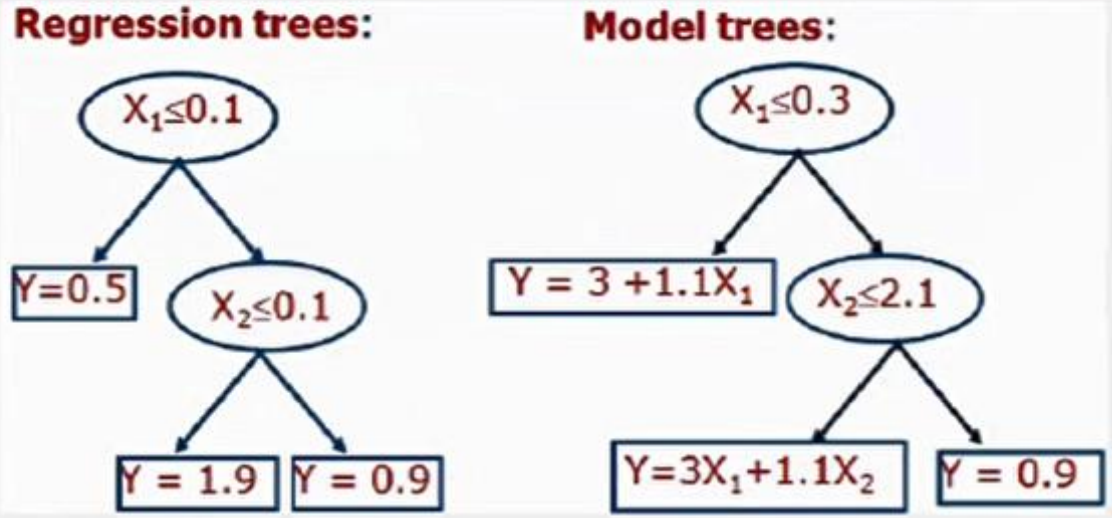
결과 해석

- summary함수를 통해 모델의 예측값은 실제값보다 훨씬 범위가 좁음을 확인, 제1사분위수와 제3사분위수 사이는 잘 적용되지만 극단값(최하/최상)은
  정확하게 식별하지 못할 것으로 판단됨
- 예측값과 실제값의 상관도를 살펴본 결과 상관관계가 매우 높음을 확인. 하지만 오차에 대한 측정은 아님 >> 평균 절대 오차(Mean Absolute Error)
- 실제 아파트 가격에 대한 회귀 트리 모델(not pruned)의 MAE는 32,035.81,
  임의로 가지치기한 회귀 트리 모델의 MAE는 35,798.62로 가지치기 하지 않은 회귀 트리 모델의 MAE보다 높게 나타나 가지치기를 하지 않는 것이 좋다고 할 수 있음
- Training data의 평균의 MAE 또한 85,228.96으로 회귀 트리 모델보다 매우 높게 나타나는데, 이는 모델이 평균적으로 실제 아파트 가격에 더 가깝다고 볼 수 있음
- BUT, 아파트 가격의 범위를 고려해보면 MAE의 수치가 높으며 극단값에 대한
  정확도 문제가 있음 >> 모델 성능 향상이 필요함

4. R 예제 실습 - (5) 모델 성능 향상

Model tree

- 모델 트리는 leaf node를 회귀 모델로 변환하여 예측에 회귀식을 사용함
- Leaf node에서 예측에 대해 특정 값(평균)을 사용하는 회귀 트리 모델보다 더 정확함
- 현재의 최신 모델 트리는 M5'(M5-prime) algorithm으로, 본래의 M5 algorithm을 보완한 것



M5P(dv ~ iv, data = , subset = , na.action = , ···)
* 모델에 대한 조정은 Weka_control 옵션 사용

```
## Step5. Improving model performance
# Model tree using M5' algorithm
library(RWeka)

set.seed(408)

apt_mod.tr <- M5P(SalePrice ~ ., data = apt_train)
apt_mod.tr
# capture.output(apt_mod.tr, file = "filename.txt") : txt 파일 형태로 전체 출력결과 보기
summary(apt_mod.tr)

plot(apt_mod.tr); text(apt_mod.tr)
# Error : argument is of length zero (plot이 안 그려지는 이유를 모르겠음ㅠ_ㅠ)
```

4. R 예제 실습 – (5) 모델 성능 향상

```
M5 pruned model tree:
(using smoothed linear models)

Size.sqf. <= 802 :
|   YrSold <= 2012.5 :
|   |   N_FacilitiesNearBy.Park. <= 0.5 :
|   |   |   YrSold <= 2010.5 :
|   |   |   |   Size.sqf. <= 604 : LM1 (117/7.759%)
|   |   |   |   Size.sqf. >  604 :
|   |   |   |   |   YearBuilt <= 1999 :
|   |   |   |   |   |   Floor <= 6.5 : LM2 (25/4.979%)
|   |   |   |   |   |   Floor >  6.5 : LM3 (72/4.883%)
|   |   |   |   |   YearBuilt >  1999 :
|   |   |   |   |   |   Floor <= 8.5 : LM4 (15/7.491%)
|   |   |   |   |   |   Floor >  8.5 :
|   |   |   |   |   |   |   YrSold <= 2008.5 : LM5 (8/4.784%)
|   |   |   |   |   |   |   YrSold >  2008.5 : LM6 (25/3.263%)
|   |   |   YrSold >  2010.5 : LM7 (114/7.13%)
|   |   N_FacilitiesNearBy.Park. >  0.5 : LM8 (227/5.8%)
:
:
LM num: 80
SalePrice =
        10216.1438 * YearBuilt
        + 17775.3776 * YrSold
        + 2905.152 * MonthSold
        + 293.6222 * Size.sqf.
        + 615.1985 * Floor
        + 3567.7398 * HallwayType=terraced
        + 68.0376 * AptManageType=management_in_trust
        + 12.2197 * N_Parkinglot.Ground.
        + 1.8535 * N_Parkinglot.Basement.
        + 1522.0805 * TimeToSubway=5min~10min,no_bus_stop_nearby,0-5min
        - 22.4544 * N_APT
        - 91.9369 * N_FacilitiesNearBy.Hospital.
        + 440.0458 * N_FacilitiesNearBy.Dpartmentstore.
        + 160.5039 * N_FacilitiesNearBy.ETC.
        - 143.7843 * N_FacilitiesNearBy.Park.
        + 61.428 * N_FacilitiesInApt
        - 67.8971 * N_FacilitiesNearBy.Total.
        - 56379084.2351

Number of Rules : 80
```

```
> summary(apt_mod.tr)

=== Summary ===

Correlation coefficient                  0.9897
Mean absolute error                  10449.7708
Root mean squared error              15401.9906
Relative absolute error                 11.8776 %
Root relative squared error             14.3556 %
Total Number of Instances             4125
```

```
> apt_reg.tr
n= 4125

node), split, n, deviance, yval
      * denotes terminal node

 1) root 4125 47482650000000 221411.90
   2) Size.sqf.< 802 1329  4703525000000 122626.00
     4) N_Parkinglot.Basement.< 1052 1257  3017543000000 114186.10
       8) YrSold< 2012.5 603   602097200000  86214.08 *
       9) YrSold>=2012.5 654  1508622000000 139976.90
        18) Size.sqf.< 565 186    83311550000  86543.62 *
        19) Size.sqf.>=565 468   683201300000 161213.10 *
     5) N_Parkinglot.Basement.>=1052 72    33253870000 269972.40 *
```

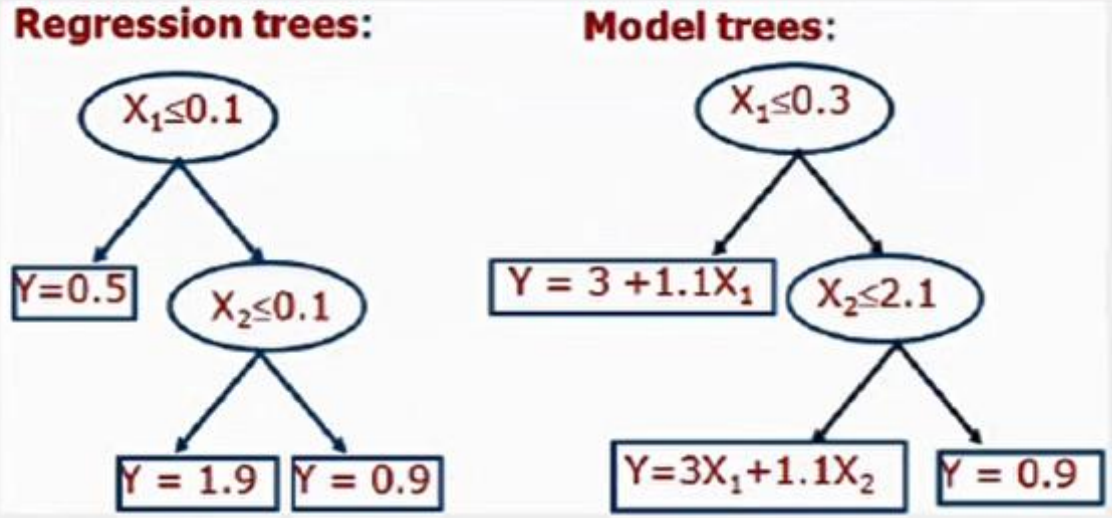
결과 해석

- 모델링 결과, leaf node에서 선형 회귀 모델(LM)이 80개가 생성됨. 회귀 트리와 첫 번째 분할 기준(Size.sqf. <= 802)은 같으나 두 번째
  분할에서 차이를 보임(N_Parkinglot.Basement. → YrSold)
- summary 함수를 통해 모델의 요약 결과를 살펴보면, 상관계수가 0.9897로 매우 높게 나타나며 MAE는 10,449.77로 회귀 트리 모델의 MAE(32,035.81)보다 작게 나타남

4. R 예제 실습 - (5) 모델 성능 향상

Model tree

- 모델 트리는 leaf node를 회귀 모델로 변환하여 예측에 회귀식을 사용함
- Leaf node에서 예측에 대해 특정 값(평균)을 사용하는 회귀 트리 모델보다 더 정확함
- 현재의 최신 모델 트리는 M5'(M5-prime) algorithm으로, 본래의 M5 algorithm을 보완한 것



M5P(dv ~ iv, data = , subset = , na.action = , ···)
* 모델에 대한 조정은 Weka_control 옵션 사용

```
## Step5. Improving model performance
# Model tree using M5' algorithm
library(RWeka)

set.seed(408)

apt_mod.tr <- M5P(SalePrice ~ ., data = apt_train)
apt_mod.tr
# capture.output(apt_mod.tr, file = "filename.txt") : txt 파일 형태로 전체 출력결과 보기
summary(apt_mod.tr)

plot(apt_mod.tr); text(apt_mod.tr)
# Error : argument is of length zero (plot이 안 그려지는 이유를 모르겠음ㅠ_ㅠ)
```

4. R 예제 실습 – (5) 모델 성능 향상

```
# Evaluation model tree
apt_mod_pred <- predict(apt_mod.tr, apt_test)
summary(apt_mod_pred)
summary(apt_test$SalePrice)

cor(apt_mod_pred, apt_test$SalePrice)
MAE(apt_mod_pred, apt_test$SalePrice)
```

```
> summary(apt_mod_pred)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  30937  145494  208926  219945  284875  552249
> summary(apt_test$SalePrice)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  34513  144373  207964  220765  291039  570796
>
> cor(apt_mod_pred, apt_test$SalePrice)
[1] 0.9874535
>
> MAE(apt_mod_pred, apt_test$SalePrice)
[1] 11341.4
```

결과 해석

- M5P algorithm을 사용하여 생성한 모델 트리의 예측값의 분포가 실제값의 분포와 유사함
- 모델 트리, 회귀 트리 각각의 실제값에 대한 상관계수와 MAE를
  비교해보면 상관계수는 0.917에서 0.987로 증가, MAE는 32,053.81에서 11,341.4로 감소함

  >> 모델의 성능이 향상되었음을 알 수 있음

```
> cor(apt_reg_pred, apt_test$SalePrice)
[1] 0.916737
> MAE(apt_reg_pred, apt_test$SalePrice)
[1] 32035.81
```

4. R 예제 실습 – rpart in caret package

```
## + Rpart in caret package
set.seed(408)

apt_rpart <- train(SalePrice ~., method = "rpart", data = apt_train)
apt_rpart

plot(apt_rpart)

plot(apt_rpart$finalModel, uniform = T, main = "Regression Tree")
text(apt_rpart$finalModel, use.n = T, all = T, cex = 0.8)
```

```
> apt_rpart
CART

4125 samples
  29 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 4125, 4125, 4125, 4125, 4125, 4125, ...
Resampling results across tuning parameters:

  cp             RMSE         Rsquared      MAE
  0.08730115935  68589.26871  0.5893751114  55063.53145
  0.13862846891  76597.59826  0.4877025636  61358.05163
  0.40296451641  93803.35285  0.3966083147  76202.91009

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.08730115935.
```

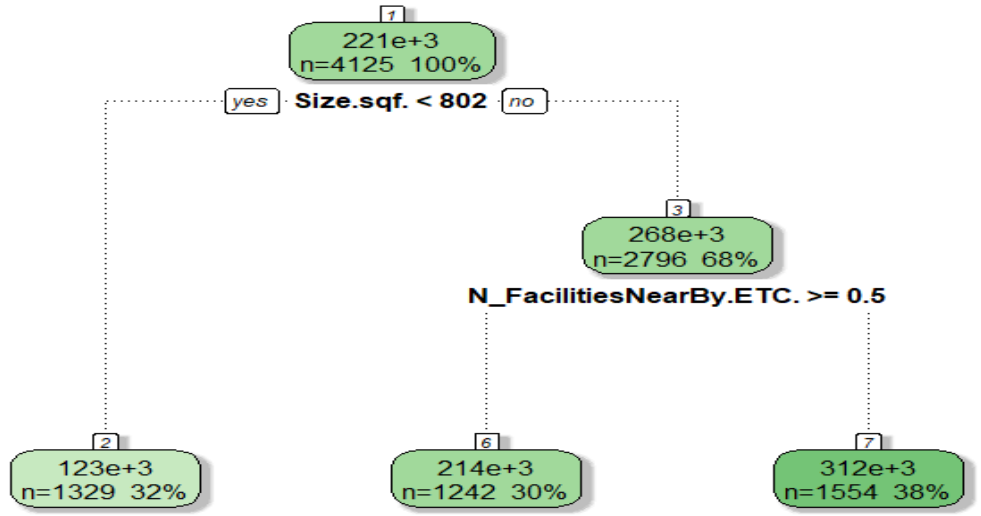
```
n= 4125

            CP nsplit    rel error
1 0.40296451641      0 1.0000000000
2 0.13862846891      1 0.5970354836
3 0.08730115935      2 0.4584070147

Variable importance
                          Size.sqf.       N_SchoolNearBy.University.
                                 28                               12
            N_SchoolNearBy.Total.              HallwayTypeterraced
                                 10                               10
          N_FacilitiesNearBy.ETC.       N_FacilitiesNearBy.Total.
                                 10                                7
                N_FacilitiesInApt                       YearBuilt
                                 7                                6
N_FacilitiesNearBy.PublicOffice.            N_SchoolNearBy.High.
                                 5                                5

Node number 1: 4125 observations,    complexity param=0.4029645164
  mean=221411.9193, MSE=1.151094557e+10
  left son=2 (1329 obs) right son=3 (2796 obs)
  Primary splits:
      Size.sqf.                     < 802    to the left,  improve=0.4029645164, (0 missing)
      HallwayTypeterraced           < 0.5    to the left,  improve=0.3940830690, (0 missing)
      YearBuilt                     < 2005.5 to the left,  improve=0.3403171914, (0 missing)
      N_FacilitiesNearBy.ETC.       < 0.5    to the right, improve=0.3022231161, (0 missing)
      N_FacilitiesNearBy.PublicOffice. < 4.5    to the right, improve=0.2775939043, (0 missing)
```

결과 해석

- 총 25개의 모델이 생성되었으며 그 중 RMSE가 가장 작은 모델이 최종 모델로 선택됨. 최종 모델의 cp는 약 0.087 (최종 모델 : apt_rpart$finalModel)
- 변수 중요도를 보면 앞선 회귀 트리 및 모델 트리와 첫 번째 분할 기준은 Size.sqf. < 802로 동일함

4. R 예제 실습 – rpart in caret package

```
# Visualization for decision tree
library(rattle)

fancyRpartPlot(apt_rpart$finalModel)

# Evaluation model performance
apt_rpart_pred <- predict(apt_rpart, apt_test)
cor(apt_rpart_pred, apt_test$SalePrice)
MAE(apt_rpart_pred, apt_test$SalePrice)
```




```
> cor(apt_rpart_pred, apt_test$SalePrice)
[1] 0.7270466531
> MAE(apt_rpart_pred, apt_test$SalePrice)
[1] 57553.58851
```

```
> cor(apt_mod_pred, apt_test$SalePrice)
[1] 0.9874534893
> MAE(apt_mod_pred, apt_test$SalePrice)
[1] 11341.39948
```

결과 해석

- caret 패키지의 rpart method로 생성한 회귀 트리 모델의 예측값과 실제값의 상관계수는 0.73, MAE는 57,553.59로, 이전의 모델
  트리와 비교해보면 MAE가 매우 크게 나타남
- Apt data로 실습해본 결과, rpart 회귀 트리, m5p 모델 트리, train(method="rpart") 회귀트리 중 m5p 모델 트리의 성능이 가장 좋음

감사합니다^0^