통계학과 진리장학금 빅데이터 분석 보고서

서울시 동별 지가와 관련된 지리적 요인 탐색

---

1. 분석 목표

교통편의성, 학군, 인구, 재개발 계획 등의 요인들은 (공시)지가에 직접 영향을 준다고 명시적으로 알려져 있다. 하지만 이 외에도 다양한 요인들이 영향을 미칠 수 있다고 알려져 있으며, 특히 한국 사회에서는 '집값'이 굉장히 민감한 이슈이기 때문에 관련 요인에 대한 관심은 항상 뜨거운 편이다. 예를 들어, 최근 강서구에서는 특수학교 건립을 반대하는 주민들이 지가 및 주택 가격을 떨어뜨린다는 주장을 제기하며 장애학생 부모와 갈등을 빚은 바 있다. 이에 본 분석에서는 특수학교, 대형병원, 유흥업소, 숙박업소, 대형마트 등 주요 시설·인프라의 분포를 지가에 영향을 미칠 수 있는 잠재적인 변수로 선정하고, 이들 변수가 실제로도 지가와 관련되어 있는지 탐색해보고자 한다. 또한, 다양한 머신러닝 알고리즘을 이용하여 지가를 예측해보고, 오차가 작은 예측 모델을 만들어보려고 한다.

2. 데이터 소개

1) 반응변수: 2017 년 서울특별시 법정동별 개별공시지가

- 출처: 서울 열린데이터 광장 - 서울시 개별공시지가 정보 (https://data.seoul.go.kr/openinf/fileview.jsp?infId=OA-1180)
- 정의

개별공시지가는 표준지공시지가를 이용하여 산정한 개별토지의 단위면적당 가격이다. 국토교통부장관은 매년 전국의 토지 중 대표성이 높은 표준지를 선정하고, 단위면적(m2)당 적정가격인 표준지공시지가를 결정하며 시장·군수·구청장은 표준지의 공시지가를 바탕으로 하여 개별토지의 단위면적당 적정가격인 개별공시지가를 산정한다. 개별공시지가는 개별토지의 용도(주거용, 상업용, 공업용 등), 도로·교통조건, 토지이용규제사항 등을 유사한 이용가치를 가진 표준지와 비교하여 토지가격비준표에 의해 산출된 가격배율에 표준지공시지가를 곱하여 산정된다.

- 전처리 과정

단위면적당 가격이기 때문에 같은 법정동 내에 여러 행의 토지 가격 정보가 포함되어 있었다. 따라서 행정구와 법정동을 기준으로 평균을 내어 해당 구 또는 동의 지가를 계산하였다.

2) 설명변수: 동별 대형마트, 특수학교, 대형병원, 유흥·단란주점, 숙박업소, 학교 개수

- 출처: 서울 열린데이터 광장

- 기본 전처리

업체 주소 자료로부터 행정구역 정보(자치구명, 법정동명)를 추출하였다. 주소 자료에 법정동명이 아닌 행정동명이 입력되어있는 경우 법정동명으로 변경하였다. 결측값의 경우 직접 네이버 주소검색을 이용하여 행정구역 정보를 추출하여 채워넣었다. 법정동별로 해당하는 자료의 개수에 대한 빈도표를 작성하였으며, 분석에는 이 빈도 데이터를 사용하였다.

(법정동: 옛 지명 등에서 유래된 이름을 사용하여 법으로 정한 동으로 모든 정부 기관의 공무나 재산권 및 권리행사 등의 법률행위에 사용되는 동. 변동이 거의 없다.

행정동: 행정 운영의 편의를 위하여 설정한 행정구역으로 도시의 확장, 인구의 이동 등 지역여건 변화와 주민 수의 증감에 따라 수시로 설치 또는 폐지할 수 있는 동)

현재 서울특별시에는 총 25 개 구, 467 개의 법정동이 존재하며, 이름이 같은 동이 서로 다른 구에 있는 경우가 두 가지 존재하므로 (신사동: 강남구, 은평구 / 신정동: 양천구, 마포구) 같은 값으로 처리되지 않도록 구분하였다.

- 데이터별 특이사항

유흥주점과 단란주점은 법적으로는 구분되어 있으나 이번 분석에서는 구별할 필요가 없으므로 '유흥업소'라는 상위 카테고리로 묶어서 고려하였다.

유흥업소, 대형마트, 숙박업소 등은 폐업한 업소도 데이터셋에 포함되어 있으므로 현재 영업 중인 곳만 추출하여 사용하였다.

대형마트 데이터는 법적으로 매장 면적이 3000m^2 이상인 대형마트와 아울렛, 백화점 목록이다. 이 중 목록 안에 있더라도 식료품이나 의류, 생활용품을 종합적으로 판매하지 않는 곳은 삭제하였다. (예: 세운상가 등 전자상가 삭제, GS 슈퍼 등 의류매장이 없는 슈퍼형 마트 삭제)

또한 각 대형마트 홈페이지에 명시되어 있는 점포 개수와 비교하여 목록에 없는 경우 추가하였다.

마트와 백화점이 붙어있는 경우 각각 따로 집계하였다.

아울렛 내에 있는 식료품매장이 따로 목록에 있는 경우 아울렛에 포함하여 하나로 집계하였다.

당초 계획에서는 녹지 및 공원도 설명변수로 사용하려 하였으나 녹지, 공원의 면적이 천차만별이기 때문에 법정동별 개수로 활용하는 것은 합리적이지 못하다고 판단하였다. 더불어 그 용도나 관리 실태 또한 다양하기 때문에 법정동별 면적으로 활용하는 것도 합리적이지 못하다고 판단하였다. 따라서 녹지, 공원 변수는 설명변수 목록에서 제외하였다.

- 변수 변환

숙박업소, 유흥업소, 학교의 경우 동별 개수 데이터가 동별 면적의 영향을 받을 것이므로 해당 법정동의 면적으로 나누어 1km^2 당 개수로 변환하였다.

```
newDat$yuheung <- newDat$yuheung / newDat$landsize
newDat$sukbak <- newDat$sukbak / newDat$landsize
newDat$school <- newDat$school / newDat$landsize
# 1m^2 당 개수이므로 10^6 을 곱하여 1km^2 당 개수로 변환
newDat$yuheung <- newDat$yuheung * 10^6
newDat$sukbak <- newDat$sukbak * 10^6
newDat$school <- newDat$school * 10^6
```

대형마트, 대형병원, 특수학교의 경우 기본적으로 전체 업체 수가 적고 좁은 구역에 집중되어 있는 경향이 있으므로 0 또는 1 의 바이너리 변수로 변환하였다.

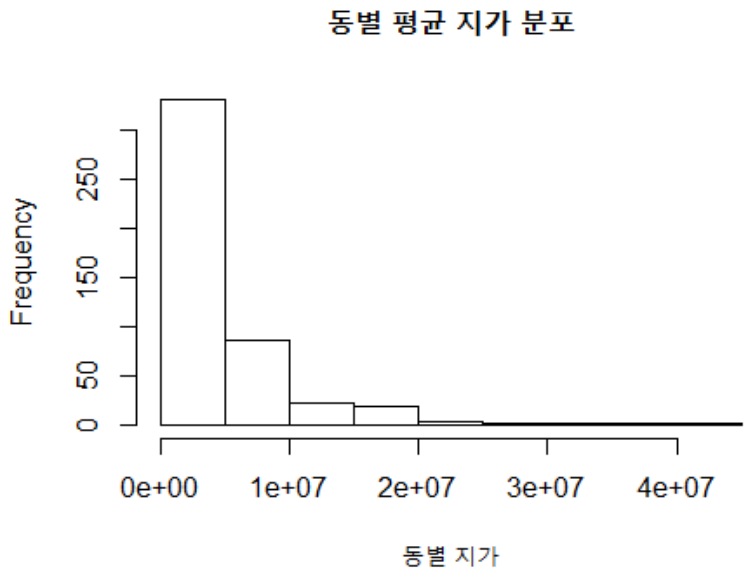
```
## [1] "특수학교 존재하는 동 비율: 0.0556745182012848"

## [1] "학교 존재하는 동 비율: 0.488222698072805"
```

```
## [1] "병원 존재하는 동 비율: 0.113490364025696"

## [1] "대형마트 존재하는 동 비율: 0.139186295503212"

## [1] "숙박업소 존재하는 동 비율: 0.644539614561028"

## [1] "유흥업소 존재하는 동 비율: 0.627408993576017"
```

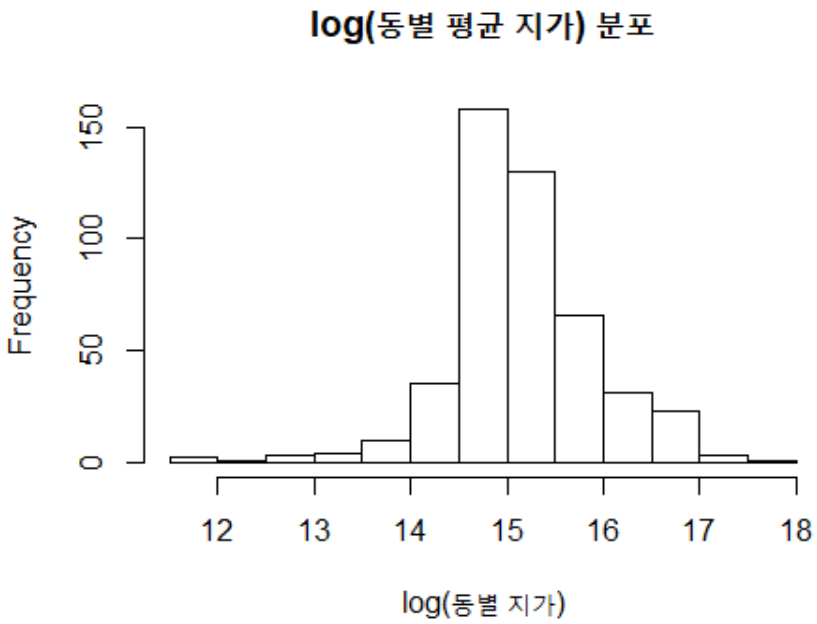
```r
# 1 이상 값의 비율이 매우 낮은 특수학교, 병원, 대형마트만 binary variable 로 변환
for (i in 1:nrow(newDat)) {
  if (newDat$specialsch[i] >= 1) {
    newDat$specialsch[i] <- 1
  } else newDat$specialsch[i] <- 0
}

for (i in 1:nrow(newDat)) {
  if (newDat$hospital[i] >= 1) {
    newDat$hospital[i] <- 1
  } else newDat$hospital[i] <- 0
}

for (i in 1:nrow(newDat)) {
  if (newDat$store[i] >= 1) {
    newDat$store[i] <- 1
  } else newDat$store[i] <- 0
}

dat <- newDat
```

3. 탐색적 데이터 분석

1) 반응변수 분포



데이터의 분포가 한 쪽으로 쏠려 있기 때문에 자연로그를 취하여 정규분포에 가깝게 변환하였다.



```
dat$landvalue <- log(dat$landvalue)
```

2) 설명변수 분포

- 숙박업소

진하게 표시된 지역은 단위면적당 숙박업소 개수가 많은 지역이다.

- 유흥업소

진하게 표시된 지역은 단위면적당 유흥업소 개수가 많은 지역이다. 숙박업소 분포와 대체로 유사한 경향을 보인다.

- 학교

진하게 표시된 지역은 단위면적당 학교 개수가 많은 지역이다.

4. 모델 만들기

1) Train & Test data 나누기

```
dat <- dat[, -c(1, 9, 10)]
set.seed(321)
randomindex <- sample(1:nrow(dat), size=round(nrow(dat)*0.8), replace=F)
train_x <- dat[randomindex, -1]
test_x <- dat[-randomindex, -1]

train_y <- dat[randomindex, 1]
test_y <- dat[-randomindex, 1]

traindat <- data.frame(train_x, train_y)
testdat <- data.frame(test_x, test_y)
```

2) 선형회귀분석

- 모델 적합

```
lmFit <- lm(train_y ~ ., data = traindat)
summary(lmFit)

##
## Call:
## lm(formula = train_y ~ ., data = traindat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.14651 -0.35910 -0.06623  0.37806  2.24053
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.1098657  0.0513311 294.361  < 2e-16 ***
## yuheung      0.0013818  0.0006902   2.002  0.04601 *
## sukbak       0.0026022  0.0011512   2.260  0.02438 *
## school      -0.0108662  0.0101768  -1.068  0.28634
## specialsch  -0.4589152  0.1637956  -2.802  0.00535 **
## hospital    -0.1666691  0.1254293  -1.329  0.18474
## store        0.1278765  0.1105388   1.157  0.24809
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7349 on 367 degrees of freedom
## Multiple R-squared:  0.1383, Adjusted R-squared:  0.1242
## F-statistic: 9.815 on 6 and 367 DF,  p-value: 4.837e-10

lmFit_null <- lm(train_y ~ 1, data = traindat)
```

적합한 모델을 이용하여 변수선택 단계를 진행하였다.

- 변수선택 결과

```
summary(forward)

##
## Call:
## lm(formula = train_y ~ sukbak + specialsch + yuheung, data = traindat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.12563 -0.37782 -0.06703  0.34601  2.25535
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.0889819  0.0425764 354.398  < 2e-16 ***
```

```
## sukbak       0.0026885  0.0011505   2.337  0.01998 *
## specialsch  -0.4949437  0.1623085  -3.049  0.00246 **
## yuheung      0.0013949  0.0006898   2.022  0.04388 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7357 on 370 degrees of freedom
## Multiple R-squared:  0.1293, Adjusted R-squared:  0.1223
## F-statistic: 18.32 on 3 and 370 DF,  p-value: 4.187e-11
```

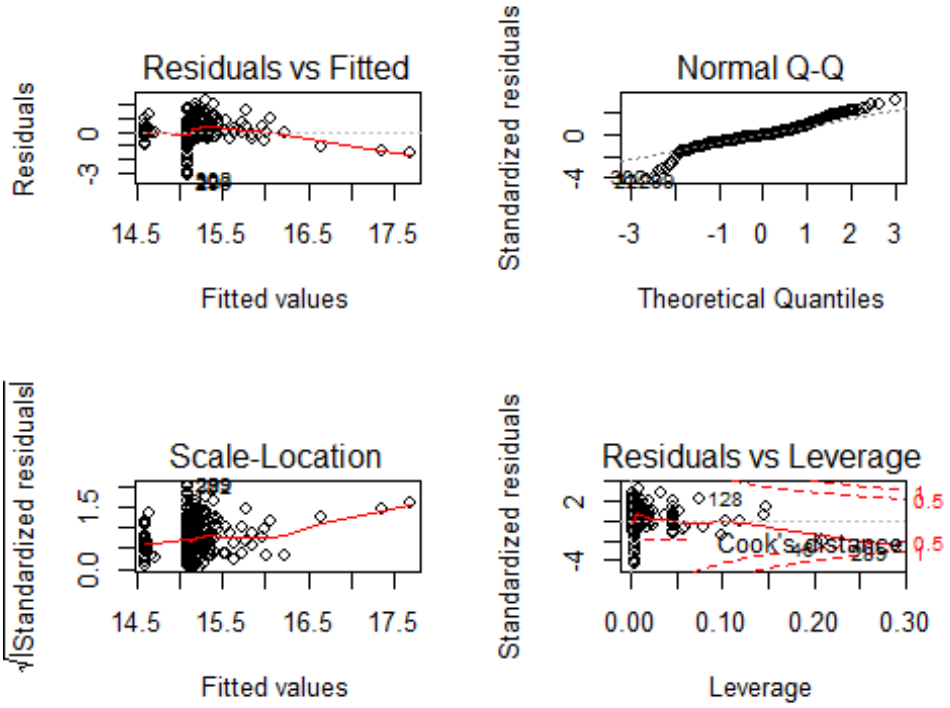
```
summary(backward)
```

```
##
## Call:
## lm(formula = train_y ~ yuheung + sukbak + specialsch, data = traindat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.12563 -0.37782 -0.06703  0.34601  2.25535
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.0889819  0.0425764 354.398  < 2e-16 ***
## yuheung      0.0013949  0.0006898   2.022  0.04388 *
## sukbak       0.0026885  0.0011505   2.337  0.01998 *
## specialsch  -0.4949437  0.1623085  -3.049  0.00246 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7357 on 370 degrees of freedom
## Multiple R-squared:  0.1293, Adjusted R-squared:  0.1223
## F-statistic: 18.32 on 3 and 370 DF,  p-value: 4.187e-11
```

```
summary(stepwise)
```

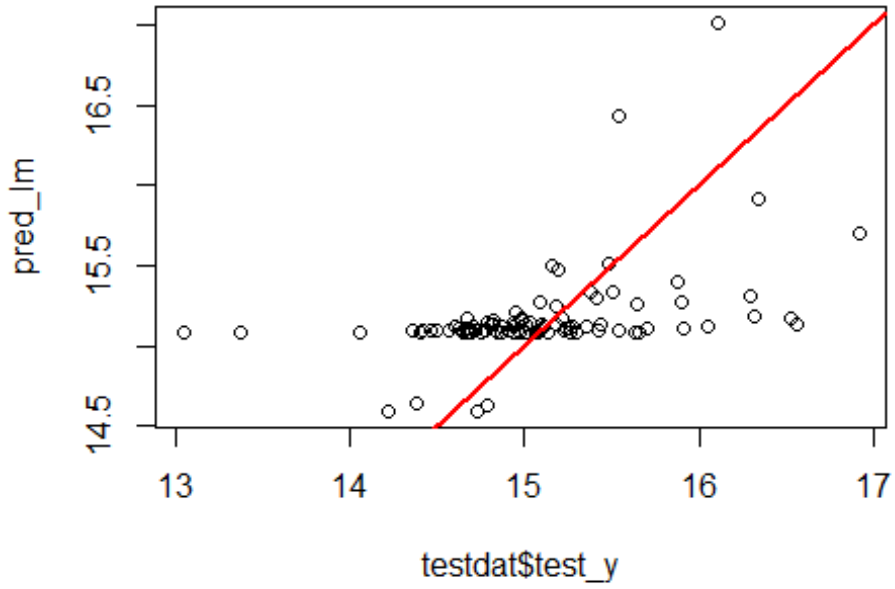
```
##
## Call:
## lm(formula = train_y ~ yuheung + sukbak + specialsch, data = traindat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.12563 -0.37782 -0.06703  0.34601  2.25535
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.0889819  0.0425764 354.398  < 2e-16 ***
## yuheung      0.0013949  0.0006898   2.022  0.04388 *
## sukbak       0.0026885  0.0011505   2.337  0.01998 *
## specialsch  -0.4949437  0.1623085  -3.049  0.00246 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7357 on 370 degrees of freedom
## Multiple R-squared:  0.1293, Adjusted R-squared:  0.1223
## F-statistic: 18.32 on 3 and 370 DF,  p-value: 4.187e-11

# 세 방법 모두 숙박업소, 유흥업소, 특수학교를 유의한 변수로 분류하였다.

par(mfrow=c(2, 2))
plot(forward)
```



- 예측

```
pred_lm <- predict(forward, testdat)
mean((pred_lm - testdat$test_y)^2)

## [1] 0.3101954

par(mfrow=c(1, 1))
plot(pred_lm ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```




```
cor(pred_lm, testdat$test_y)
```

```
## [1] 0.4375469
```

- 결과 해석: 숙박업소, 유흥업소, 특수학교가 유의한 변수로 분류되었다.

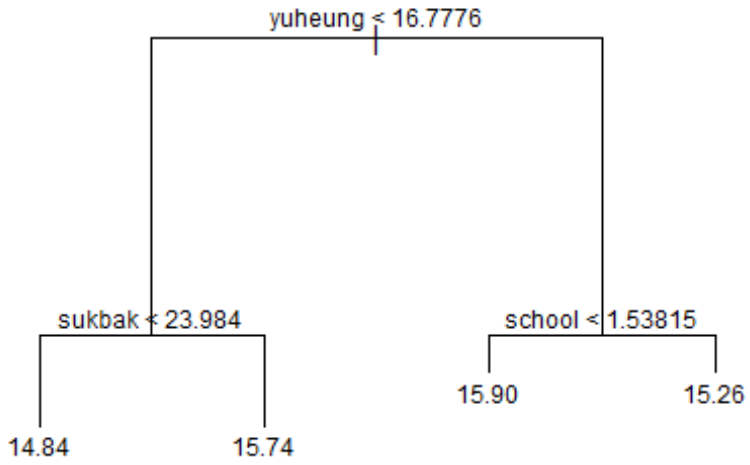
숙박업소와 유흥업소가 많이 분포한 지역은 업무지구보다는 유동인구가 적지만 주거지구보다는 유동인구가 많은 각 행정구의 중심가에 해당한다. 따라서 숙박업소나 유흥업소가 많은 곳이라면 그렇지 않은 곳보다 지가가 조금 더 높을 것이라고 예상할 수 있다.

특수학교의 경우 특수학교가 위치한 동은 그렇지 않은 동보다 지가가 상당히 낮다는 결과가 도출되었다. 하지만 이는 상관관계일 뿐 인과관계가 아니므로 선후관계를 함부로 단정지을 수 없다. 장애인은 비장애인보다 저소득층 비율이 높은 집단이기 때문에, 소득 및 생활 수준이 상대적으로 낮은 구 및 동에 장애인 거주 비율이 높을 수밖에 없다.
특수학교가 장애인 거주 비율이 높은 지역에 설립된 것이지, 특수학교 설립에 영향을 받아 지가가 하락했다는 추론에는 타당성이 부족하다.

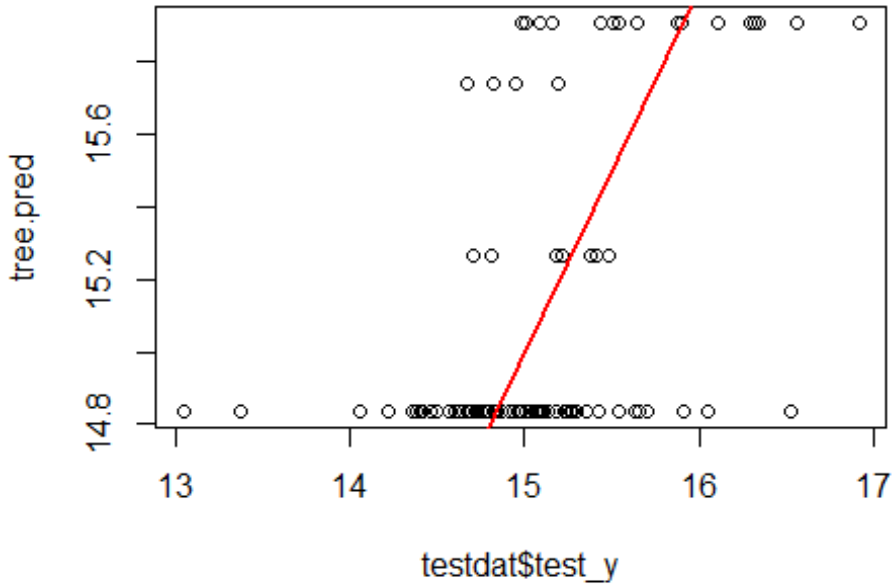
3) 회귀나무 (Regression Tree)

- tree library 를 이용한 회귀나무 모델

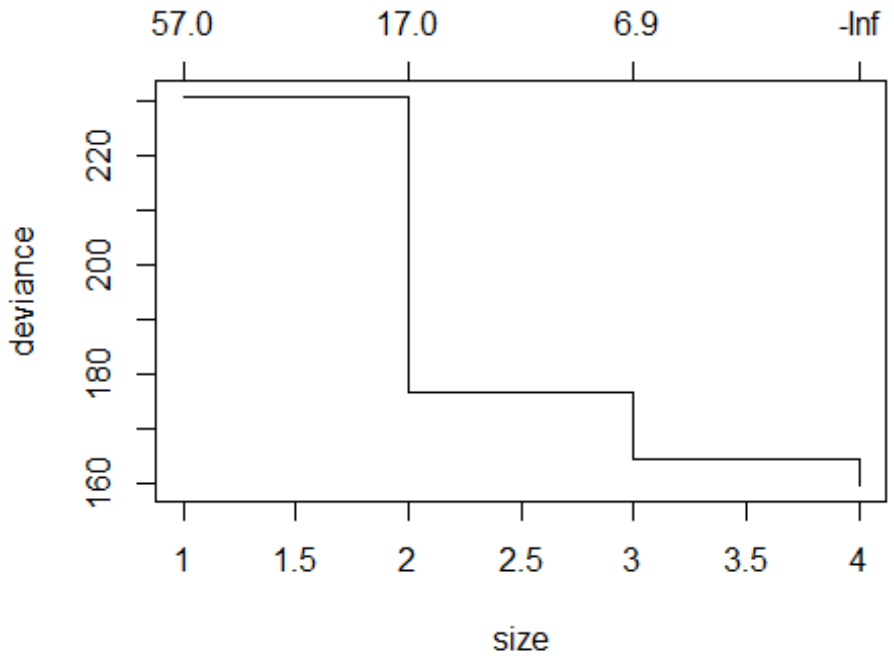
```
library(tree)

tree.fit <- tree(train_y ~ ., data=traindat)
tree.fit

## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 374 230.000 15.16
##   2) yuheung < 16.7776 268 129.800 14.91
##     4) sukbak < 23.984 245 105.500 14.84 *
##     5) sukbak > 23.984 23   7.212 15.74 *
##   3) yuheung > 16.7776 106  43.510 15.78
##     6) school < 1.53815 85  32.450 15.90 *
##     7) school > 1.53815 21   4.120 15.26 *

summary(tree.fit)

##
## Regression tree:
## tree(formula = train_y ~ ., data = traindat)
## Variables actually used in tree construction:
## [1] "yuheung" "sukbak"  "school"
## Number of terminal nodes:  4
## Residual mean deviance:  0.4036 = 149.3 / 370
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.8730 -0.2938  0.0318  0.0000  0.3143  1.9330
```




```
tree.pred <- predict(tree.fit, testdat)
mean((tree.pred - testdat$test_y)^2)

## [1] 0.3016692

plot(tree.pred ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



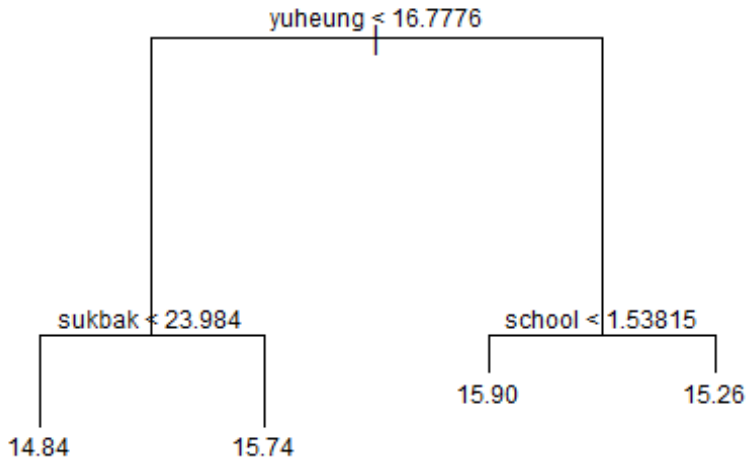

```
cor(tree.pred, testdat$test_y)

## [1] 0.4859298

tree.CV <- cv.tree(tree.fit, K=10, FUN=prune.tree)

# Deviance 에 대한 plot
plot(tree.CV)
```



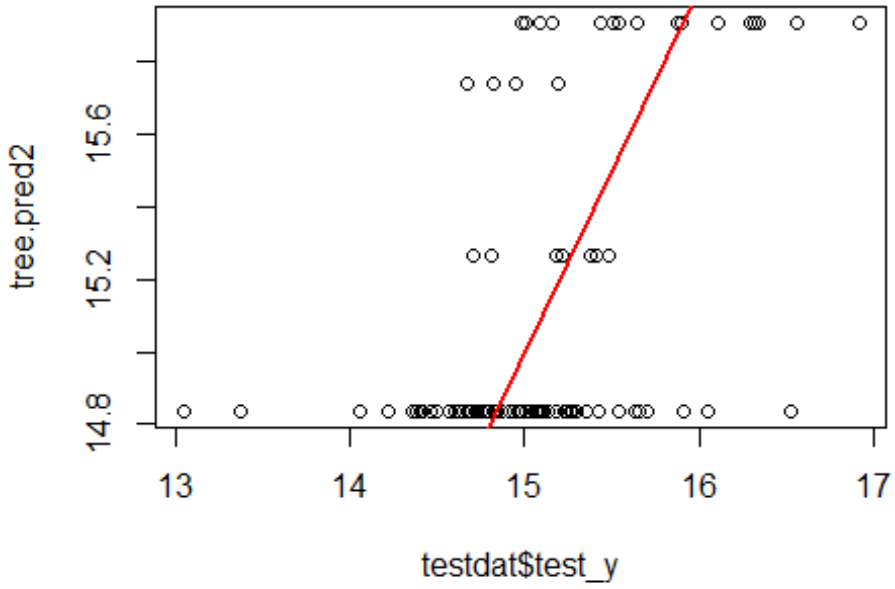

```
tree.CV$size[tree.CV$dev==min(tree.CV$dev)] # deviance 를 최소화하는 가지의 개수
```

```
## [1] 4
```

```
tree.prune <- prune.tree(tree.fit, best=tree.CV$size[tree.CV$dev==min(tree.CV$dev)])
plot(tree.prune)
text(tree.prune, cex=.75)
```



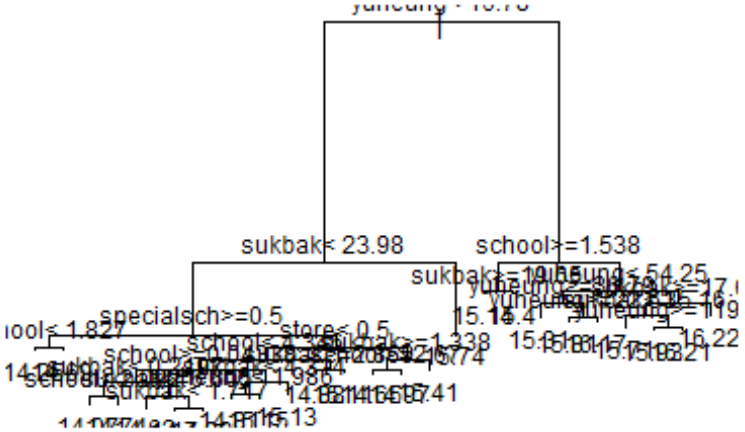

```
tree.pred2 <- predict(tree.prune, testdat)
mean((tree.pred2 - testdat$test_y)^2)

## [1] 0.3016692

plot(tree.pred2 ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



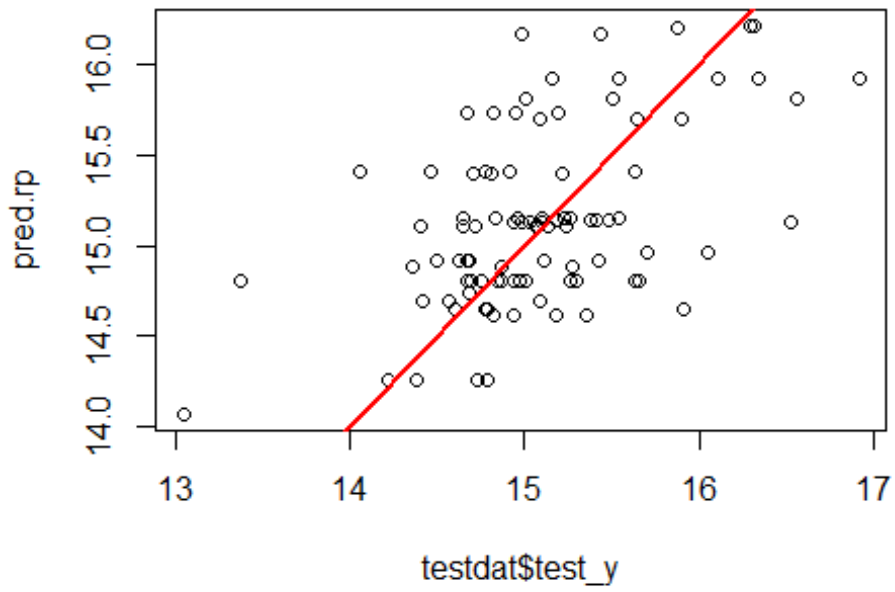

```
cor(tree.pred2, testdat$test_y)

## [1] 0.4859298
```

- rpart library 를 이용한 회귀나무 모델

```
library(rpart)
rpart.fit <- rpart(train_y~., data=traindat, control = rpart.control(cp=0.00
1))

plot(rpart.fit)
text(rpart.fit, cex=.75, pretty=0)
```



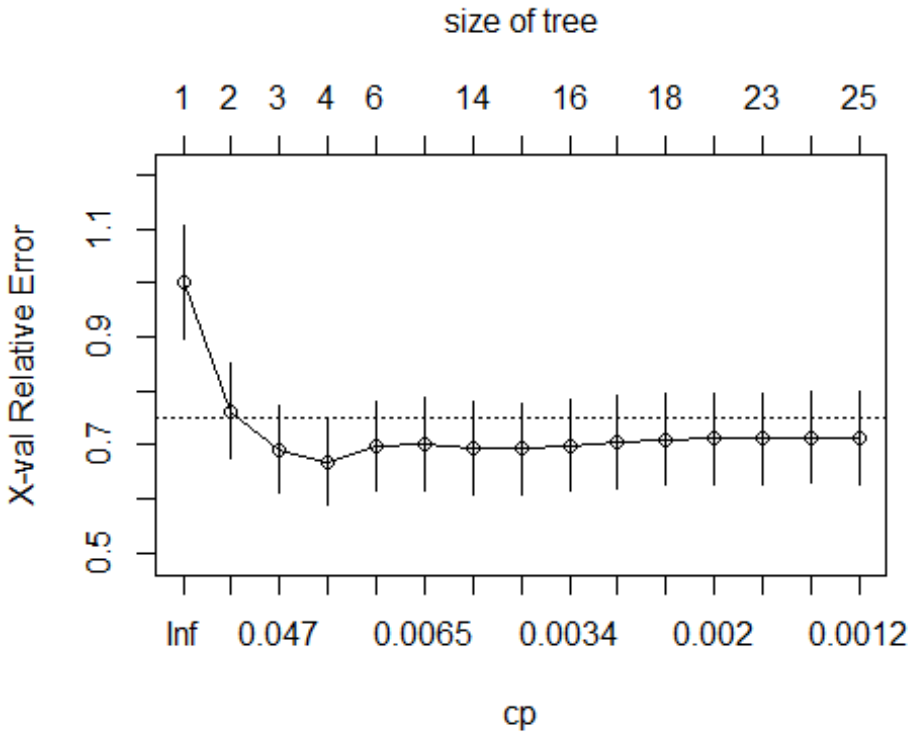

```
pred.rp <- predict(rpart.fit, testdat)
mean((pred.rp - testdat$test_y)^2)

## [1] 0.3024767

plot(pred.rp ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



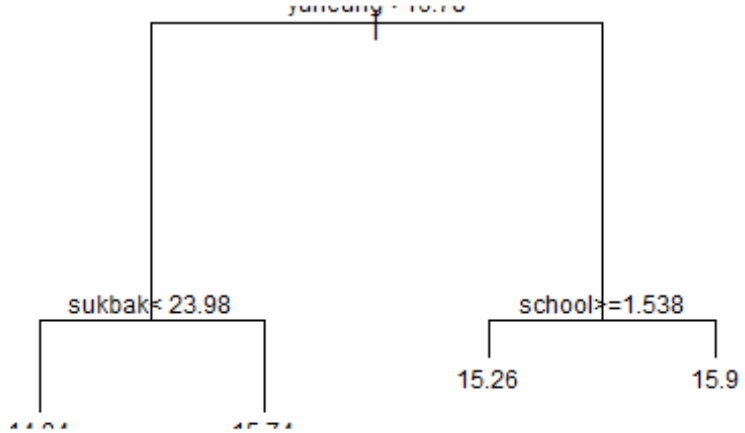

```
cor(pred.rp, testdat$test_y)

## [1] 0.5232065

printcp(rpart.fit)

##
## Regression tree:
## rpart(formula = train_y ~ ., data = traindat, control = rpart.control(cp =
 0.001))
##
## Variables actually used in tree construction:
## [1] school     specialsch store      sukbak     yuheung
##
## Root node error: 230/374 = 0.61498
##
## n= 374
##
##           CP nsplit rel error  xerror     xstd
## 1  0.2463082      0   1.00000 1.00156 0.105597
## 2  0.0743309      1   0.75369 0.76127 0.088075
## 3  0.0301669      2   0.67936 0.69153 0.081191
## 4  0.0108485      3   0.64919 0.66833 0.080554
## 5  0.0086001      5   0.62750 0.69777 0.083091
## 6  0.0049363     12   0.56664 0.70170 0.085980
## 7  0.0037659     13   0.56170 0.69383 0.084676
```

```
## 8  0.0036719     14   0.55794 0.69291 0.084481
## 9  0.0031871     15   0.55427 0.69882 0.084669
## 10 0.0024424     16   0.55108 0.70468 0.085023
## 11 0.0022625     17   0.54864 0.70979 0.085094
## 12 0.0018400     20   0.54157 0.71072 0.085121
## 13 0.0015432     22   0.53789 0.71101 0.085157
## 14 0.0014148     23   0.53635 0.71383 0.085518
## 15 0.0010000     24   0.53493 0.71131 0.085408
```

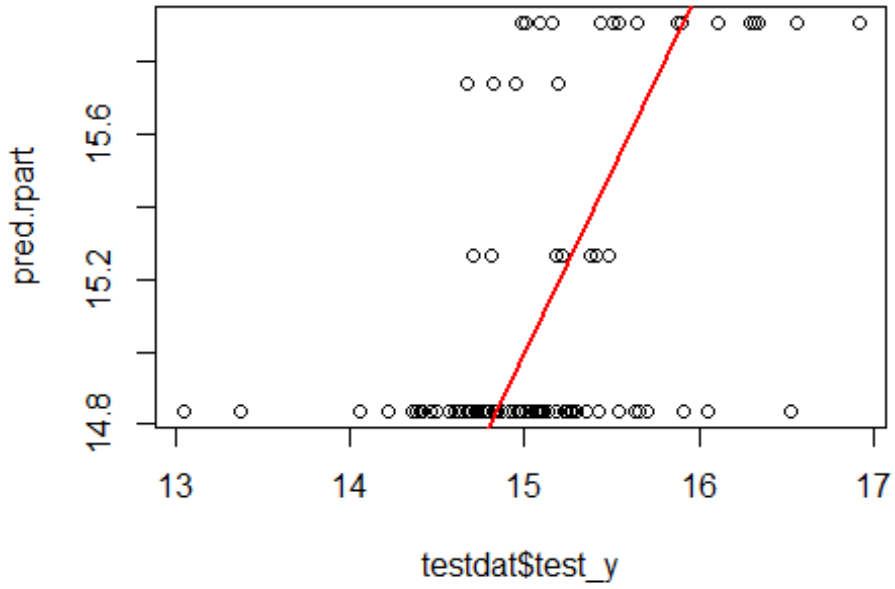
```
plotcp(rpart.fit)
```




```
prune.rpart <- prune(rpart.fit, cp=rpart.fit$cptable[which.min(rpart.fit$cpta
ble[,"xerror"]), "CP"])
plot(prune.rpart)
text(prune.rpart, cex=.75, pretty=0)
```



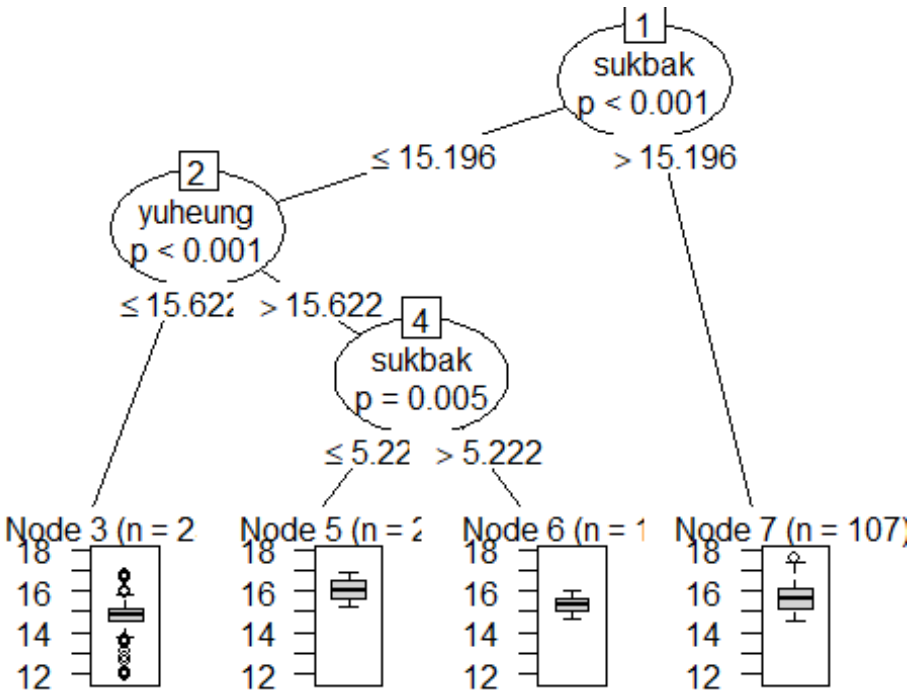

```
pred.rpart <- predict(prune.rpart, testdat)
mean((pred.rpart - testdat$test_y)^2)

## [1] 0.3016692

plot(pred.rpart ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



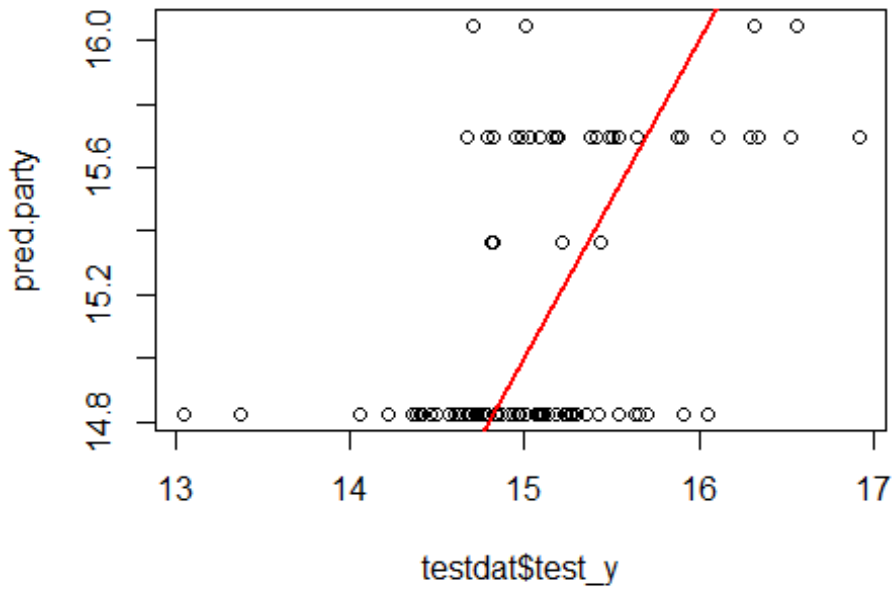

```
cor(pred.rpart, testdat$test_y)

## [1] 0.4859298
```

- party library 를 이용한 회귀나무 모델

```
party.fit <- ctree(train_y~., data=traindat)
plot(party.fit)
```




```
pred.party <- predict(party.fit, testdat)
mean((pred.party-testdat$test_y)^2)

## [1] 0.3162056

plot(pred.party ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```




```
cor(pred.party, testdat$test_y)

##               [,1]
## train_y 0.4623888
```

- 결과 해석: 가지가 4 개인 regression tree 가 가장 예측력이 좋았다.

우선 유흥업소가 많은 곳이 지가가 높은 경향을 보였고, 다음으로는 숙박업소가 많거나 학교가 적은 지역이 지가가 높았다. 학교가 많이 분포하는 지역은 주택이 밀집된 지역인 경우가 많기 때문에 외부에서 유입되는 유동인구가 상대적으로 적고, 업무용·상업용 건물이 밀집된 지역보다 지가가 낮을 것이다.

4) KNN (unweighted)

- kknn library 를 이용, unweighted 이므로 kernel 을 rectangular 로 지정

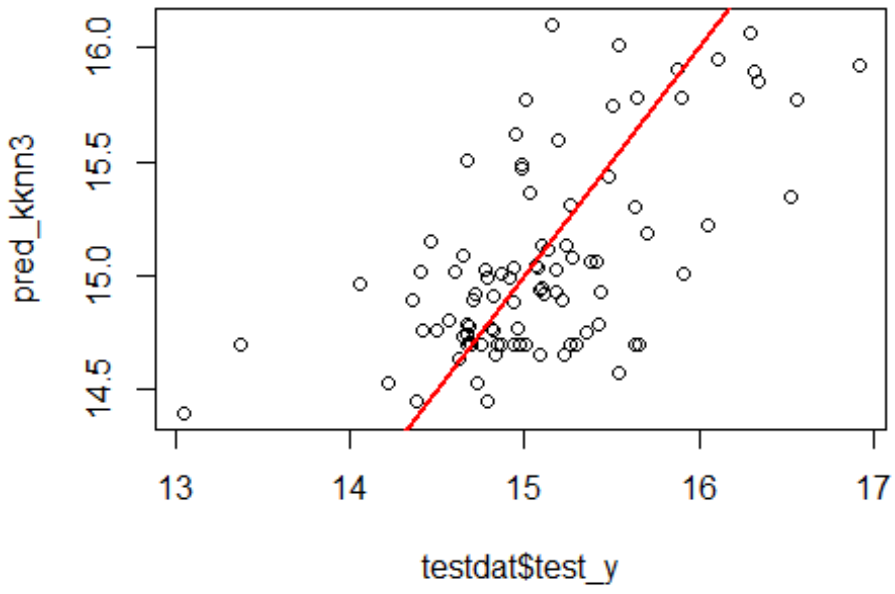
- LOOCV(Leave One Out Cross Validation)를 이용해 가장 적합한 k 찾기

```
library(kknn)
LOOCV_kknn2 <- train.kknn(train_y~., traindat, kmax=29, distance=2, kernel="r
ectangular")
LOOCV_kknn2
```

```
##
## Call:
## train.kknn(formula = train_y ~ ., data = traindat, kmax = 29,     distance
 = 2, kernel = "rectangular")
##
## Type of response variable: continuous
## minimal mean absolute error: 0.4613457
## Minimal mean squared error: 0.4262473
## Best kernel: rectangular
## Best k: 16
```

가장 적합한 k 는 16 인 것으로 나왔다. 이를 이용하여 모델을 만든다.

```
kknn_fit3 <- kknn(train_y~., train=traindat, test=testdat,
                  k=16, distance=2, kernel="rectangular")
pred_kknn3 <- fitted(kknn_fit3)

mse4 <- mean((pred_kknn3 - testdat$test_y)^2)
mse4

## [1] 0.234068

plot(pred_kknn3 ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



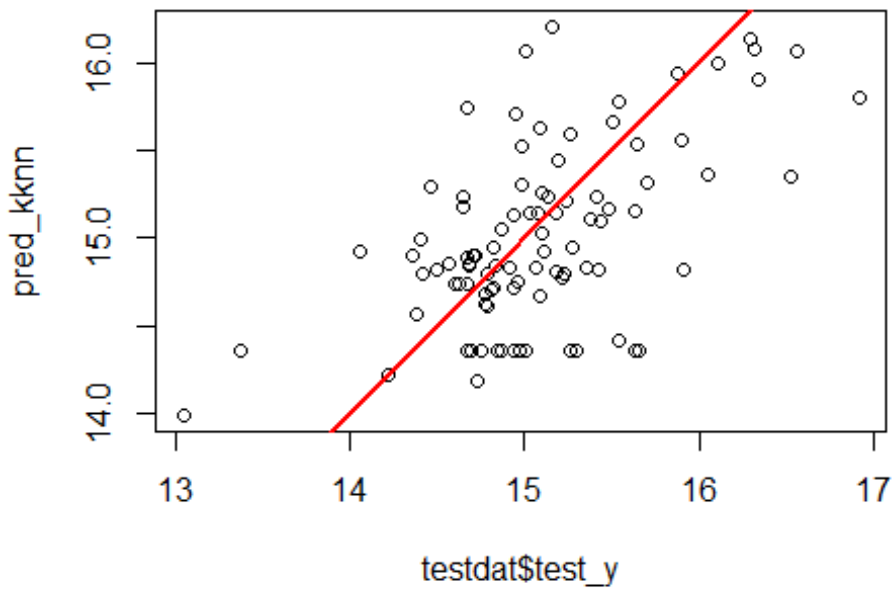

```
cor(pred_kknn3, testdat$test_y)
```

```
## [1] 0.6175182
```

5) KNN (weighted)

- 거리에 따라 더 가까운 요소에 더 큰 weight 를 주는 weighted KNN
- kernel 을 gaussian 으로 지정

```
kknn_fit <- kknn(train_y~., train=traindat, test=testdat, distance=2, kernel=
"gaussian")
pred_kknn <- fitted(kknn_fit)

mse2 <- mean((pred_kknn - testdat$test_y)^2)
mse2

## [1] 0.2819292

plot(pred_kknn ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



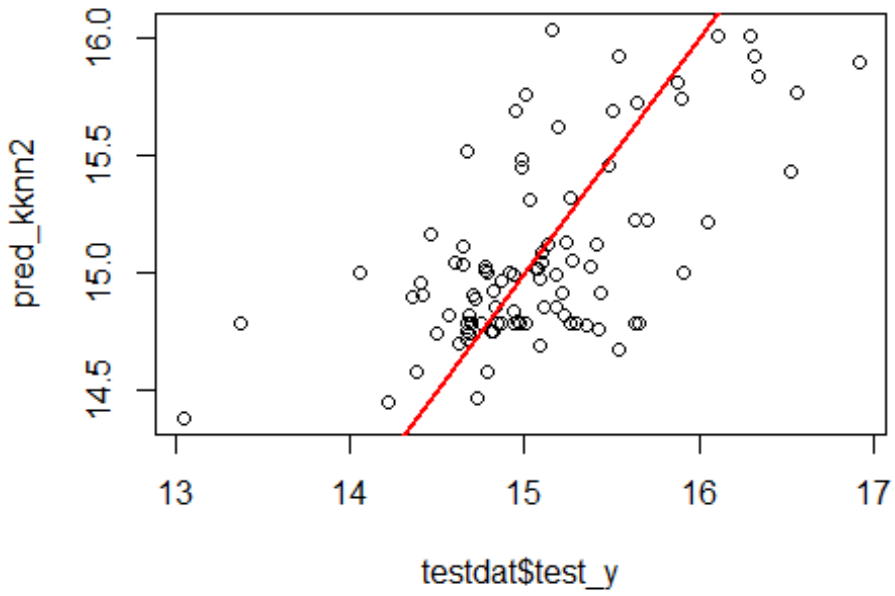

```
cor(pred_kknn, testdat$test_y)

## [1] 0.5737355
```

- LOOCV 를 이용해 가장 적합한 k 찾기

```
LOOCV_kknn <- train.kknn(train_y~., traindat, kmax=29, distance=2, kernel="ga
ussian")
LOOCV_kknn

##
## Call:
## train.kknn(formula = train_y ~ ., data = traindat, kmax = 29,     distance
 = 2, kernel = "gaussian")
##
## Type of response variable: continuous
## minimal mean absolute error: 0.4599847
## Minimal mean squared error: 0.4271206
## Best kernel: gaussian
## Best k: 29
```

가장 적합한 k 는 29 인 것으로 나왔다. 이를 이용하여 모델을 만든다.

```
kknn_fit2 <- kknn(train_y ~., train=traindat, test=testdat,
                  k=29, distance=2, kernel="gaussian")
pred_kknn2 <- fitted(kknn_fit2)

mse3 <- mean((pred_kknn2 - testdat$test_y)^2)
mse3

## [1] 0.2258529

plot(pred_kknn2 ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```




```
cor(pred_kknn2, testdat$test_y)
```

```
## [1] 0.6294451
```

- 결과 해석: Cross validation 을 통해 모델을 개선하니 이전보다 MSE 가 많이 줄어들었다. 또한 kernel 이 rectangular 일 때보다 MSE 가 줄어들었다. (모델 향상)

5. 모델 향상

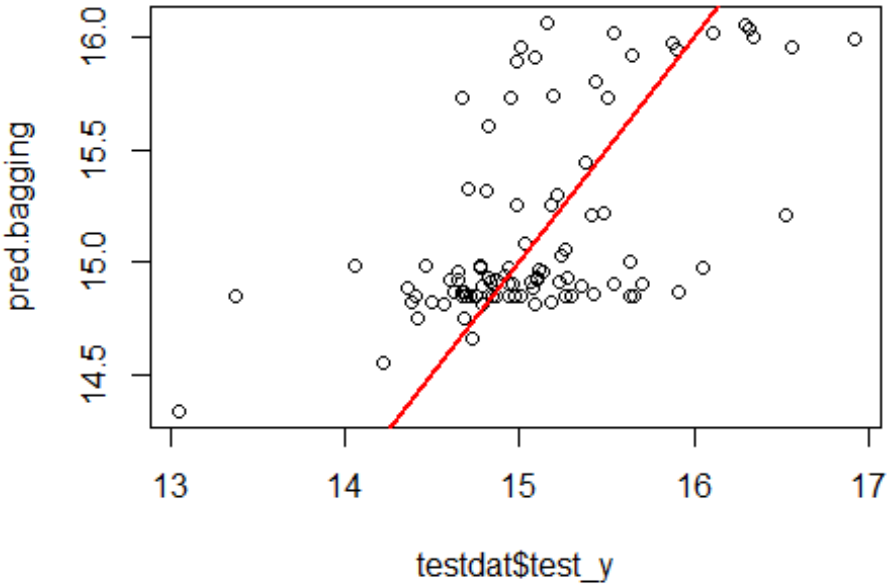
1) Bagging

- Bagging 을 이용하여 rpart 라이브러리를 이용한 회귀나무 모델을 향상

```
library(ipred)

bagging.fit <- ipred::bagging(train_y ~ ., data=traindat, nbagg=1000)

pred.bagging <- predict(bagging.fit, testdat)
mean((pred.bagging - testdat$test_y)^2)
```

```
## [1] 0.2569188
```

```
plot(pred.bagging ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```




```
cor(pred.bagging, testdat$test_y)
```

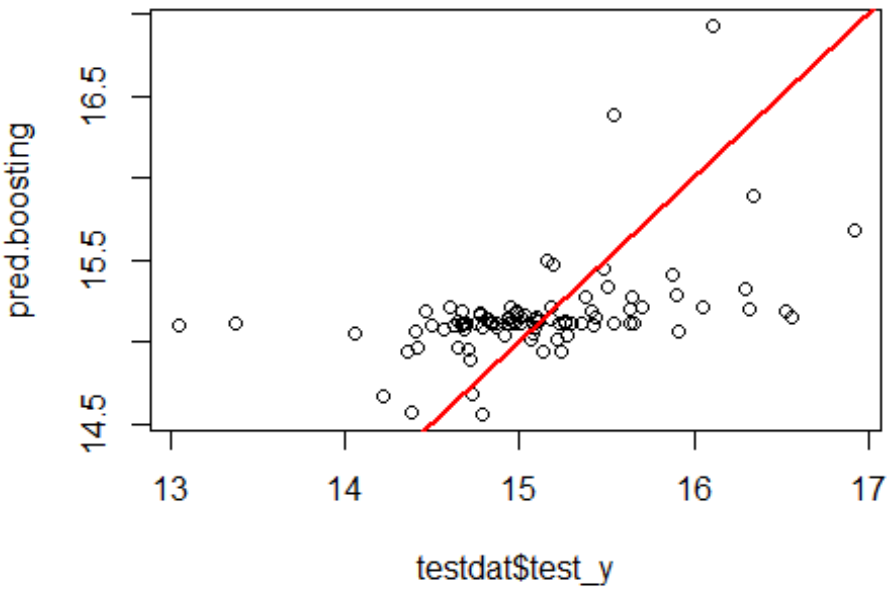
```
## [1] 0.5801231
```

- 결과 해석: MSE 가 약 0.05 정도로 크게 감소하여 모델이 상당히 개선되었다.

2) Boosting

- Boosting 을 이용하여 linear regression 모델을 향상

```
library(mboost)

## Warning: package 'mboost' was built under R version 3.4.2

## Loading required package: parallel

## Loading required package: stabs

## Warning: package 'stabs' was built under R version 3.4.2

##
## Attaching package: 'stabs'

## The following object is masked from 'package:modeltools':
##
##     parameters

## This is mboost 2.8-1. See 'package?mboost' and 'news(package  = "mboost")'
## for a complete list of changes.

##
## Attaching package: 'mboost'

## The following object is masked from 'package:ipred':
##
##     cv

## The following object is masked from 'package:party':
##
##     varimp

boosting.fit <- glmboost(train_y ~ ., data = traindat)

pred.boosting <- predict(boosting.fit, testdat)
mean((pred.boosting - testdat$test_y)^2)

## [1] 0.3018675

plot(pred.boosting ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```



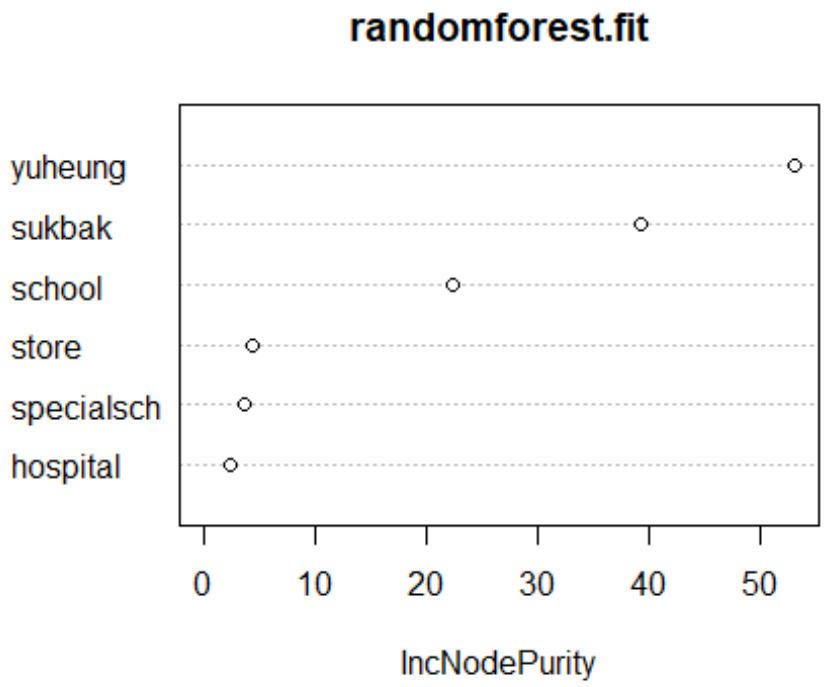

```
cor(pred.boosting, testdat$test_y)

##           [,1]
## [1,] 0.4558159
```

- 결과 해석: 기존 linear regression 모델보다 MSE 가 약간 줄어들었다.

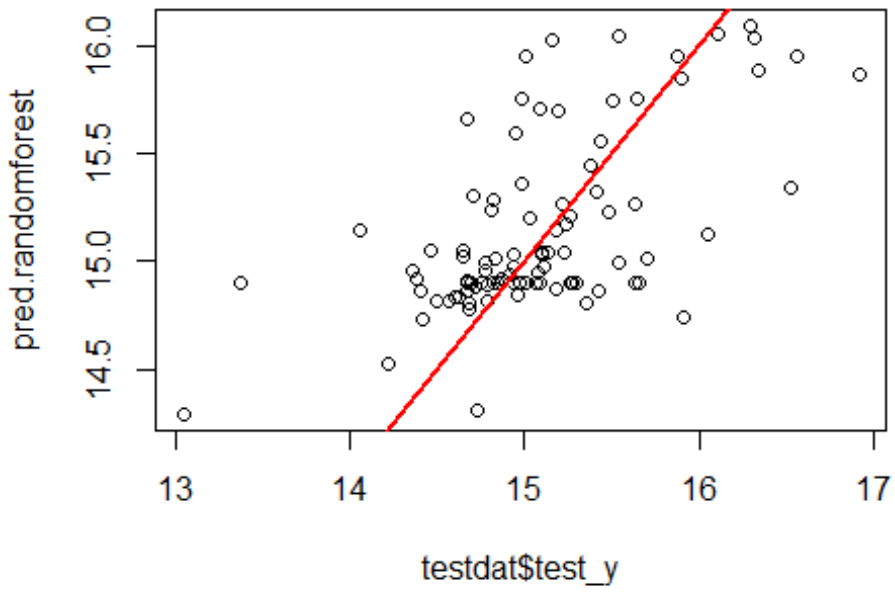
3) Random Forest

- Random Forest 를 사용하여 회귀나무 모델을 향상

```
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

randomforest.fit <- randomForest(train_y ~ ., data = traindat)
varImpPlot(randomforest.fit) # Accuracy 를 높이는데 중요한 변수와 gini 계수를 줄이는데 중요한 변수
```




```
pred.randomforest <- predict(randomforest.fit, testdat)
mean((pred.randomforest - testdat$test_y)^2)

## [1] 0.2382136

plot(pred.randomforest ~ testdat$test_y)
abline(a=0, b=1, col="red", lwd=2)
```




```
cor(pred.randomforest, testdat$test_y)
```

```
## [1] 0.6125328
```

- 결과 해석: MSE 가 0.23 으로 크게 줄어들었다. Weighted KNN 다음으로 MSE 가 작은 모델이다.

4) Ridge

- Ridge 를 사용하여 linear regression 모델을 향상

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.2
```

```
## Loading required package: Matrix
```

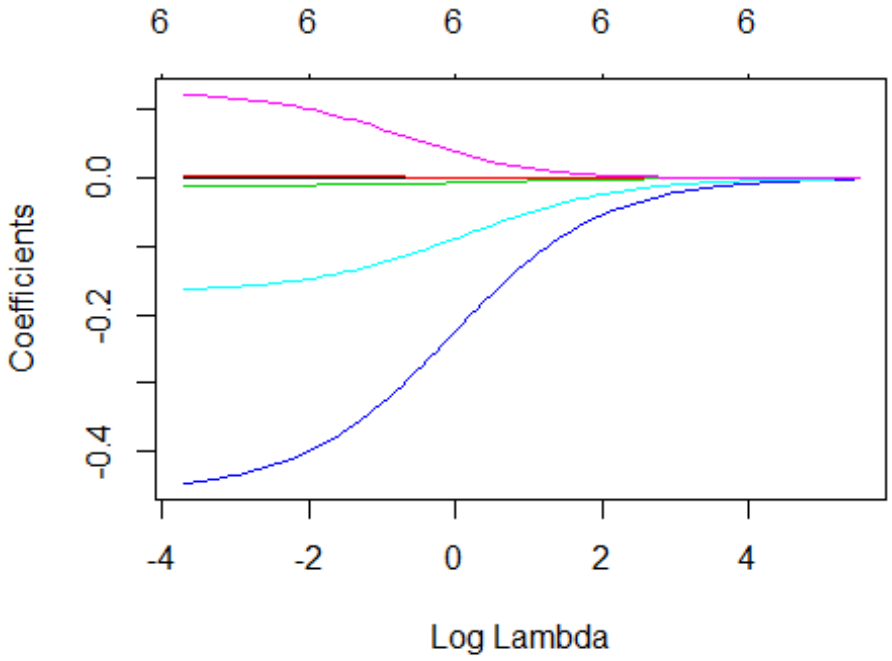
```
## Warning: package 'Matrix' was built under R version 3.4.2
```

```
## Loading required package: foreach
```

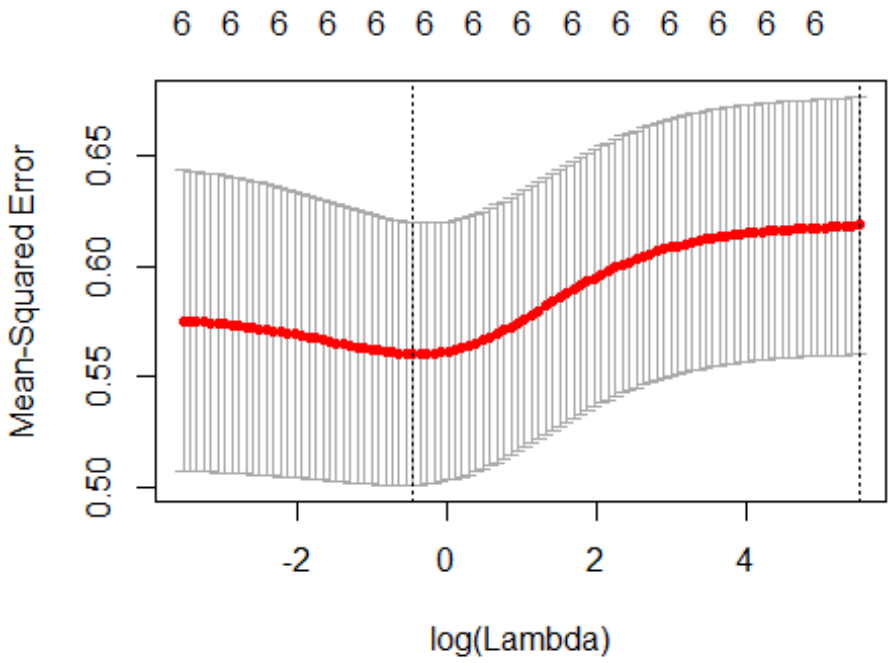
```
## Loaded glmnet 2.0-13
```

```
RidgeFit <- glmnet(x = as.matrix(train_x), y = as.matrix(train_y),
                   family = "gaussian", alpha = 0)   # alpha=0 이면 Ridge, alpha=1
```

```r
이번 Rasso
plot(RidgeFit, xvar = "lambda")
```




```r
RidgeCV <- cv.glmnet(x = as.matrix(train_x), y = as.matrix(train_y),
                     family = "gaussian", alpha = 0, nfold = 10)

RidgeCV$lambda.min

## [1] 0.6361991

plot(RidgeCV)
```



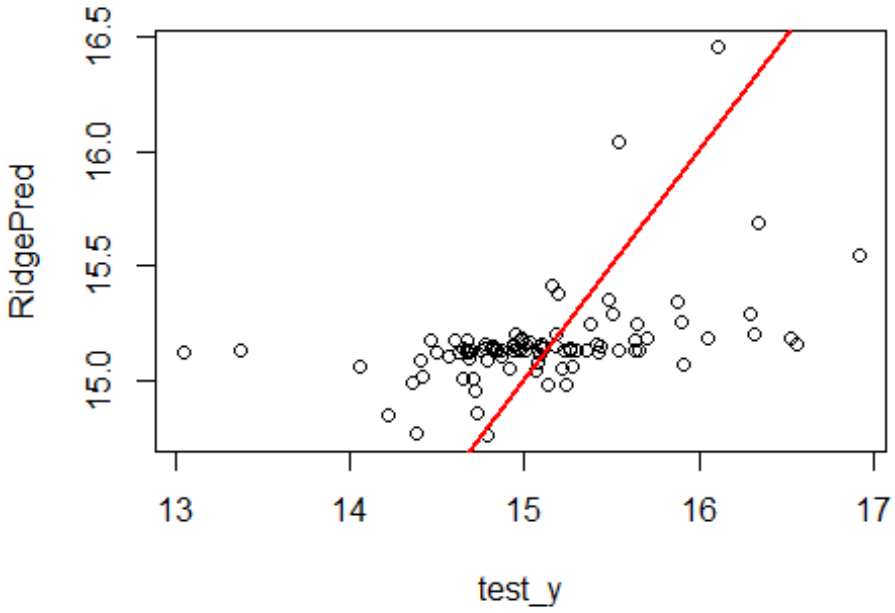

```
coef(RidgeCV, s = "lambda.min")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept) 15.126566124
## yuheung      0.001003639
## sukbak       0.001757917
## school      -0.007855151
## specialsch  -0.274356188
## hospital    -0.105817143
## store        0.053358195
```

```
RidgePred <- predict(RidgeCV, newx = as.matrix(test_x),
                     s = "lambda.min")
mean((RidgePred - test_y)^2)
```

```
## [1] 0.3068844
```

```
plot(RidgePred ~ test_y)  # 더 일반화된 모형이라 일반 모형보다 결과가 잘 나옴
abline(a=0, b=1, col="red", lwd=2)
```

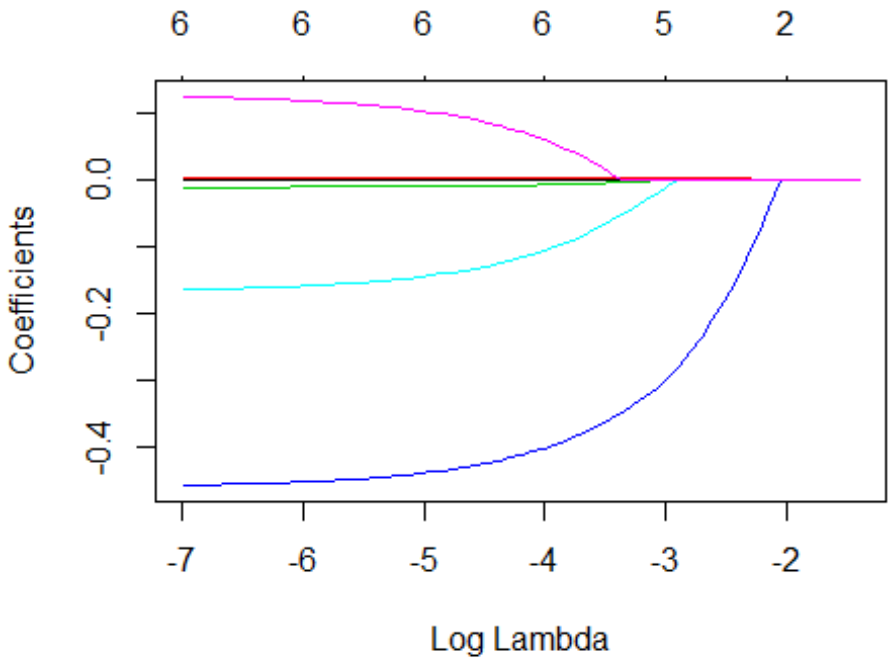



```
cor(RidgePred, test_y)

##         [,1]
## 1 0.4551832
```

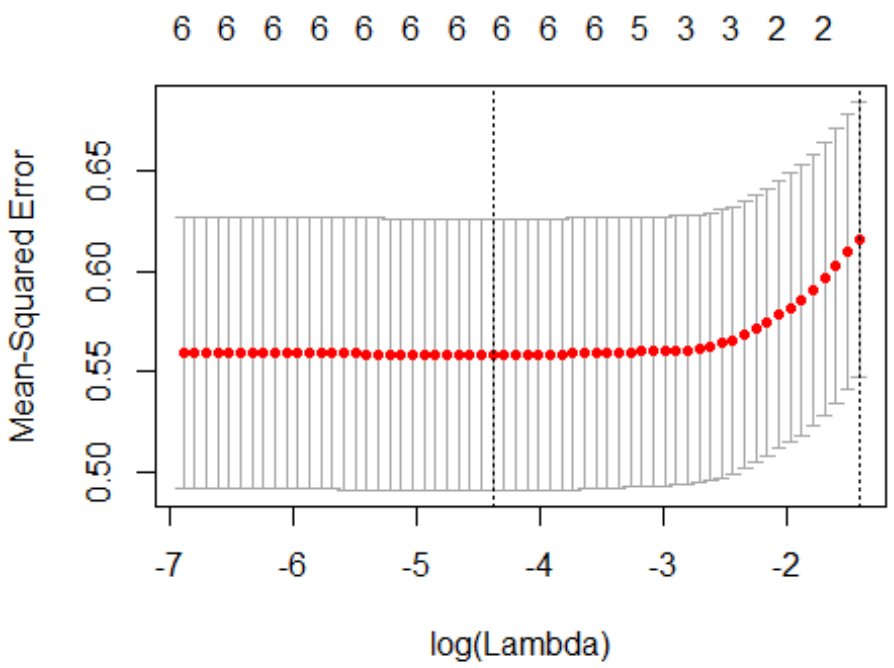
5) Lasso

- Lasso 를 사용하여 linear regression 모델을 향상

```
LassoFit <- glmnet(x = as.matrix(train_x), y = as.matrix(train_y),
                   family = "gaussian", alpha = 1)   # alpha=0 이면 Ridge, alpha=1
이면 Rasso
plot(LassoFit, xvar = "lambda")
```




```
LassoCV <- cv.glmnet(x = as.matrix(train_x), y = as.matrix(train_y),
                     family = "gaussian", alpha = 1, nfold = 10)

LassoCV$lambda.min

## [1] 0.01248885

plot(LassoCV)
```



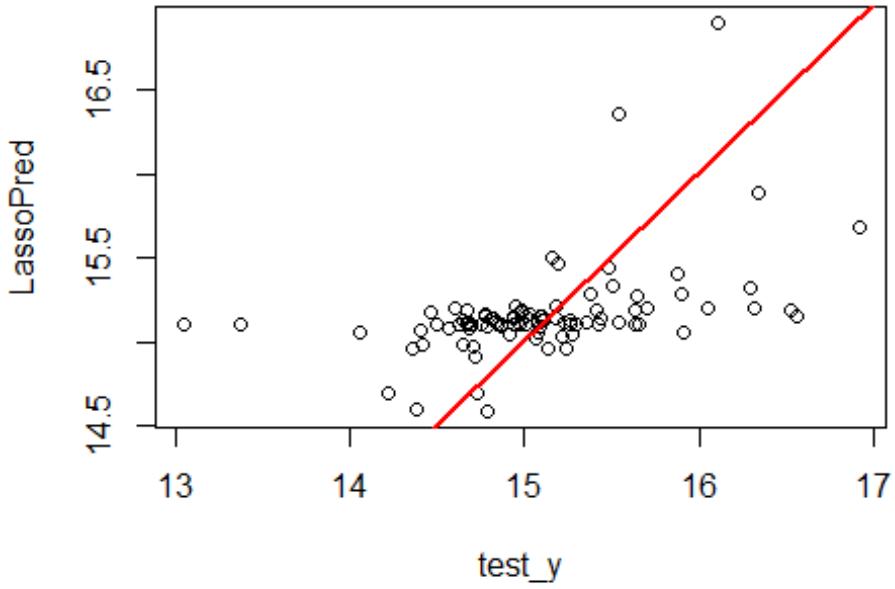

```
coef(LassoCV, s = "lambda.min")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) 15.108904799
## yuheung      0.001302628
## sukbak       0.002503382
## school      -0.008209807
## specialsch  -0.418792572
## hospital    -0.124585513
## store        0.081589662
```

```
LassoPred <- predict(LassoCV, newx = as.matrix(test_x),
                     s = "lambda.min")
mean((LassoPred - test_y)^2)
```

```
## [1] 0.3021418
```

```
plot(LassoPred ~ test_y)
abline(a=0, b=1, col="red", lwd=2)
```

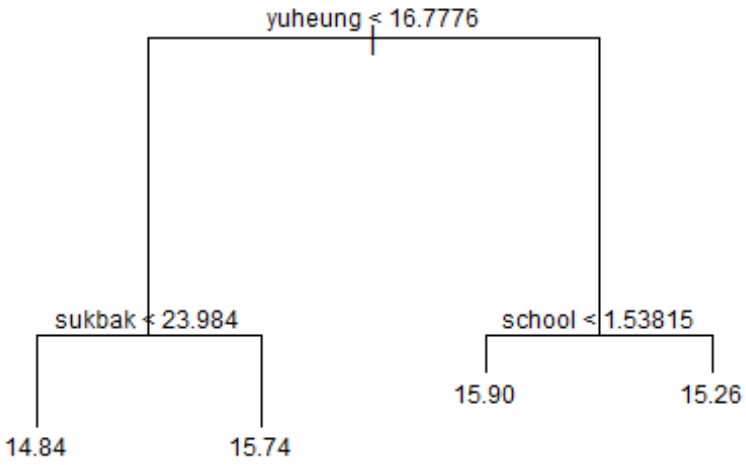



```
cor(LassoPred, test_y)

##         [,1]
## 1 0.4551175
```

- 결과 해석: Ridge 와 Lasso 모두 기존 linear regression 모델보다 MSE 가 줄어들었다. 도출된 회귀 계수에 따르면 유흥업소, 숙박업소, 대형마트는 지가와 양의 관계, 학교, 특수학교, 병원은 지가와 음의 관계에 있다.

6. 최종 모델 선택

- 예측력이 우수한 모델(= MSE(Mean Squared Error)가 작은 모델): Weighted KNN(Gaussian), Random Forest, Regression Tree(가지 4 개)
- 결과 해석이 용이한 모델: Linear Regression, Regression Tree
- 결과 해석이 용이하면서 예측도 잘하는 모델: Regression Tree(가지 4 개)



7. 분석 의의 및 한계

- 의의: 공시지가와 관련성이 높은 변수들을 탐색하고, 해당 변수들을 이용하여 지가를 예측하는 다양한 모델들을 만들었다. 또한, cross validation 을 통해 찾은 최적의 parameter 로 모델 예측력을 향상시켰을 뿐 아니라 bagging, boosting, random forest, ridge, lasso 등 다양한 모델 향상 알고리즘을 적용하여 예측력을 더욱 개선하였다.

- 한계: 반응변수 및 설명변수가 (전년 대비) 변동률이 아닌 현재 값으로 이루어져 있기 때문에 각 시설의 분포 변화가 지가 변동에 어떤 영향을 미치는지에 대해서는 알기 어려웠다. 즉, 변수로 선정한 6 종류의 시설 분포와 지가가 갖는 인과관계를 알 수는 없었고, 상관관계를 파악하는 데에 그쳤다. 또한, 467 개의 데이터로는 원하는 성능만큼 모델을 학습시키기에 어려웠다는 본질적인 한계도 존재했다.