RF_practice

Yujui Chang

2021/9/24

homeprice

```
H = read.csv(url("http://fs2.american.edu/~baron/627/R/HOME_SALES.csv"))
H$ID <- NULL
head(H)
```

```
##   SALES_PRICE FINISHED_AREA BEDROOMS BATHROOMS GARAGE_SIZE YEAR_BUILT STYLE
## 1      360000          3032        4         4           2       1972     1
## 2      340000          2058        4         2           2       1976     1
## 3      250000          1780        4         3           2       1980     1
## 4      205500          1638        4         2           2       1963     1
## 5      275500          2196        4         3           2       1968     7
## 6      248000          1966        4         3           5       1972     1
##   LOT_SIZE AIR_CONDITIONER POOL QUALITY HIGHWAY
## 1    22221             YES   NO  MEDIUM      NO
## 2    22912             YES   NO  MEDIUM      NO
## 3    21345             YES   NO  MEDIUM      NO
## 4    17342             YES   NO  MEDIUM      NO
## 5    21786             YES   NO  MEDIUM      NO
## 6    18902             YES  YES  MEDIUM      NO
```

```
set.seed(1)
n = nrow(H)
z = sample(n, n*0.8) # 0.8 for train, 0.2 for test
```

train RF model

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

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

```
train_RF = randomForest(SALES_PRICE~., data= H[z,])
train_RF
```

```
##
## Call:
##  randomForest(formula = SALES_PRICE ~ ., data = H[z, ])
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
```

```
##
##           Mean of squared residuals: 3265195110
##                     % Var explained: 82.79
```

Why 3 ? Rule of thumb: usually m ~= root(p)

```
p = ncol(H)-1 # deduct response
p
```

```
## [1] 11
```

```
sqrt(p) # No. of variables tried at each split: 3
```

```
## [1] 3.316625
```

test model

```
yhat = predict(train_RF, newdata= H[-z,])

sqrt(mean((yhat - H[-z,]$SALES_PRICE)^2)) # RMSE
```

```
## [1] 55914.49
```

tuning and test model again

```
train_RF = randomForest(SALES_PRICE~., data= H[z,],  mtry= 11) # 3 -> 11 variables
train_RF
```

```
##
## Call:
##  randomForest(formula = SALES_PRICE ~ ., data = H[z, ], mtry = 11)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##           Mean of squared residuals: 3379074480
##                     % Var explained: 82.19
```

```
yhat = predict(train_RF, newdata= H[-z,])
sqrt(mean((yhat - H[-z,]$SALES_PRICE)^2)) # RMSE: 3 is better than 11
```

```
## [1] 57911.54
```

try different trees

```
train_RF = randomForest(SALES_PRICE~. , data= H[z, ], mtry= 11, ntree= 100) # 3 -> 11 variables, 500 ->
yhat = predict(train_RF, newdata= H[-z, ])
sqrt(mean((yhat - H[-z, ]$SALES_PRICE)^2)) # RMSE is going up
```

```
## [1] 58345.46
```

find the best trees and variables

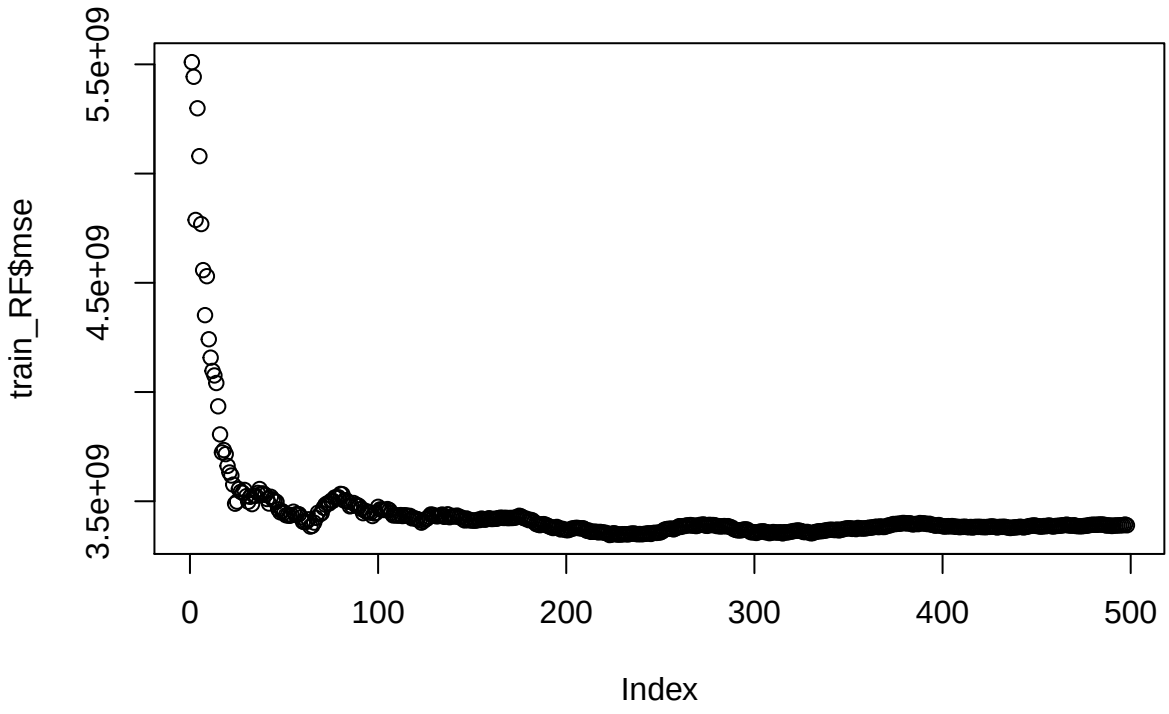
```
set.seed(1)
n = nrow(H)
z = sample(n, n*0.8)

RMSEP = rep(0,p) # p = 11 predictors. try every predictors to find the best random forest
optimaltrees = rep(0,p)

for(k in 1:p){
  train_RF = randomForest(SALES_PRICE~., data= H[z,], mtry= k)
  optimaltrees[k] = which.min(train_RF$mse) # In each variables selection, number with lowest mse = opt

  train_RF = randomForest(SALES_PRICE~., data= H[z,], mtry= k, ntree= optimaltrees[k])
  yhat = predict(train_RF, newdata = H[-z,])
  RMSEP[k] = sqrt(mean((yhat- H$SALES_PRICE[-z])^2))
}

plot(train_RF$mse)
```




```
which.min(RMSEP)
```

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

```
optimaltrees[4]
```

```
## [1] 288
```

```
# optimize with lowest rmse. tuned random forest: m= 4 variables, n= 288 trees
```

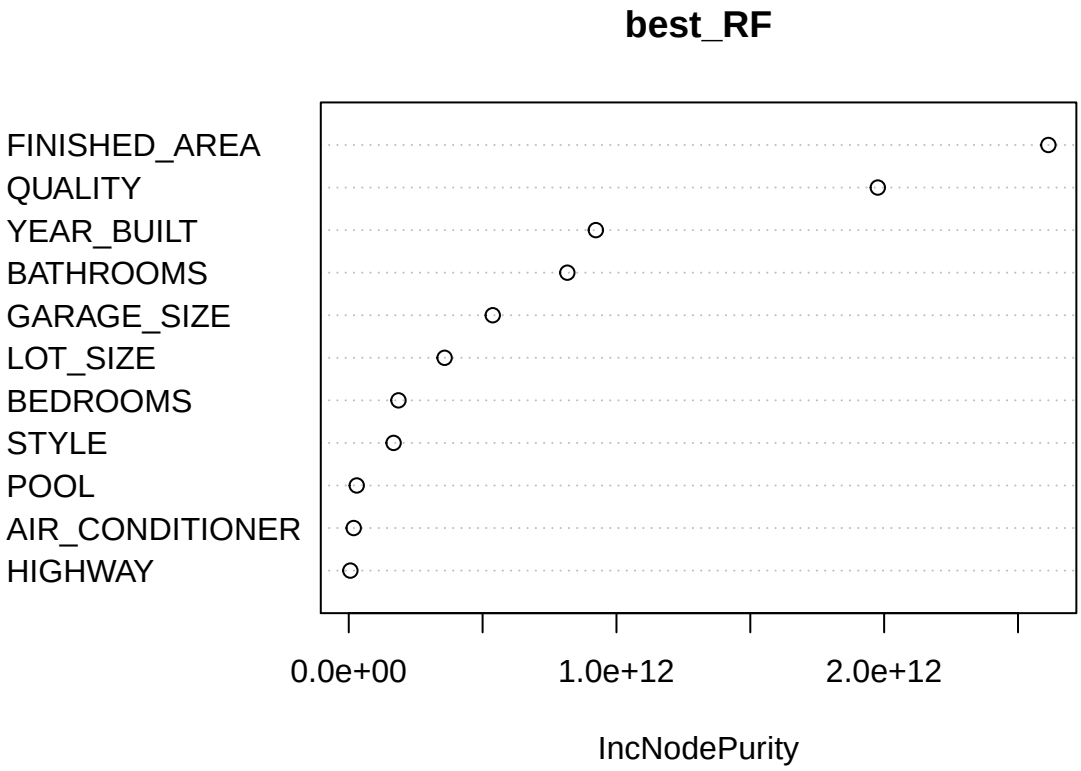
```
# fit best random forest trees
best_RF = randomForest(SALES_PRICE~., data= H[z, ], mtry= 4, ntree= 288)
best_RF
```

```
##
## Call:
##  randomForest(formula = SALES_PRICE ~ ., data = H[z, ], mtry = 4,      ntree = 288)
##                Type of random forest: regression
##                      Number of trees: 288
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 3311949216
##                     % Var explained: 82.55
```

```
importance(best_RF)
```

```
##                 IncNodePurity
## FINISHED_AREA    2.613451e+12
## BEDROOMS         1.857048e+11
## BATHROOMS        8.164726e+11
## GARAGE_SIZE      5.378127e+11
## YEAR_BUILT       9.229973e+11
## STYLE            1.674178e+11
## LOT_SIZE         3.580300e+11
## AIR_CONDITIONER  1.842449e+10
## POOL             2.982326e+10
## QUALITY          1.976288e+12
## HIGHWAY          5.983148e+09
```

```
varImpPlot(best_RF)
```


best_RF
FINISHED_AREA
QUALITY
YEAR_BUILT
BATHROOMS
GARAGE_SIZE
LOT_SIZE
BEDROOMS
STYLE
POOL
AIR_CONDITIONER
HIGHWAY
0.0e+00
1.0e+12
2.0e+12
IncNodePurity