Decision Tree and Random Forest

Candy Li

10/27/2020

Load data

```
library(readr)
library(tidyverse)
library(bnstruct)
library(tree)
library(randomForest)
data = read_csv("../data/NFWBS_PUF_2016_data.csv")
head(data)
```

```
## # A tibble: 6 x 217
##   PUF_ID sample   fpl SWB_1 SWB_2 SWB_3 FWBscore FWB1_1 FWB1_2 FWB1_3 FWB1_4
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1  10350      2     3     5     5     6       55      3      3      3      3
## 2   7740      1     3     6     6     6       51      2      2      3      3
## 3  13699      1     3     4     3     4       49      3      3      3      3
## 4   7267      1     3     6     6     6       49      3      3      3      3
## 5   7375      1     3     4     4     4       49      3      3      3      3
## 6  10910      1     3     5     7     5       67      5      1      1      1
## # ... with 206 more variables: FWB1_5 <dbl>, FWB1_6 <dbl>, FWB2_1 <dbl>,
## #   FWB2_2 <dbl>, FWB2_3 <dbl>, FWB2_4 <dbl>, FSscore <dbl>, FS1_1 <dbl>,
## #   FS1_2 <dbl>, FS1_3 <dbl>, FS1_4 <dbl>, FS1_5 <dbl>, FS1_6 <dbl>,
## #   FS1_7 <dbl>, FS2_1 <dbl>, FS2_2 <dbl>, FS2_3 <dbl>, SUBKNOWL1 <dbl>,
## #   ACT1_1 <dbl>, ACT1_2 <dbl>, FINGOALS <dbl>, PROPPLAN_1 <dbl>,
## #   PROPPLAN_2 <dbl>, PROPPLAN_3 <dbl>, PROPPLAN_4 <dbl>, MANAGE1_1 <dbl>,
## #   MANAGE1_2 <dbl>, MANAGE1_3 <dbl>, MANAGE1_4 <dbl>, SAVEHABIT <dbl>,
## #   FRUGALITY <dbl>, AUTOMATED_1 <dbl>, AUTOMATED_2 <dbl>, ASK1_1 <dbl>,
## #   ASK1_2 <dbl>, SUBNUMERACY2 <dbl>, SUBNUMERACY1 <dbl>, CHANGEABLE <dbl>,
## #   GOALCONF <dbl>, LMscore <dbl>, FINKNOWL1 <dbl>, FINKNOWL2 <dbl>,
## #   FINKNOWL3 <dbl>, FK1correct <dbl>, FK2correct <dbl>, FK3correct <dbl>,
## #   KHscore <dbl>, KHKNOWL1 <dbl>, KHKNOWL2 <dbl>, KHKNOWL3 <dbl>,
## #   KHKNOWL4 <dbl>, KHKNOWL5 <dbl>, KHKNOWL6 <dbl>, KHKNOWL7 <dbl>,
## #   KHKNOWL8 <dbl>, KHKNOWL9 <dbl>, KH1correct <dbl>, KH2correct <dbl>,
## #   KH3correct <dbl>, KH4correct <dbl>, KH5correct <dbl>, KH6correct <dbl>,
## #   KH7correct <dbl>, KH8correct <dbl>, KH9correct <dbl>, ENDSMEET <dbl>,
## #   HOUSING <dbl>, LIVINGARRANGEMENT <dbl>, HOUSERANGES <dbl>,
## #   IMPUTATION_FLAG <dbl>, VALUERANGES <dbl>, MORTGAGE <dbl>,
## #   SAVINGSRANGES <dbl>, PRODHAVE_1 <dbl>, PRODHAVE_2 <dbl>, PRODHAVE_3 <dbl>,
## #   PRODHAVE_4 <dbl>, PRODHAVE_5 <dbl>, PRODHAVE_6 <dbl>, PRODHAVE_7 <dbl>,
## #   PRODHAVE_8 <dbl>, PRODHAVE_9 <dbl>, PRODUSE_1 <dbl>, PRODUSE_2 <dbl>,
## #   PRODUSE_3 <dbl>, PRODUSE_4 <dbl>, PRODUSE_5 <dbl>, PRODUSE_6 <dbl>,
## #   CONSPROTECT1 <dbl>, CONSPROTECT2 <dbl>, CONSPROTECT3 <dbl>, EARNERS <dbl>,
## #   VOLATILITY <dbl>, SNAP <dbl>, MATHARDSHIP_1 <dbl>, MATHARDSHIP_2 <dbl>,
## #   MATHARDSHIP_3 <dbl>, MATHARDSHIP_4 <dbl>, MATHARDSHIP_5 <dbl>,
## #   MATHARDSHIP_6 <dbl>, ...
```

```
data <- data %>%
  remove_rownames %>%
  column_to_rownames(var="PUF_ID")

# notice that negative values are invalid entries,
# so replacing them with NA
for (i in 1:nrow(data)){
  for (j in 1:ncol(data)){
    if (data[i,j] < 0){
      data[i,j] = NA
    }
  }
}
```

```
# use knn impute to resolve NA problem
cleandata = knn.impute(as.matrix(data)) %>%
  as.data.frame()
rownames(cleandata) = rownames(data)
colnames(cleandata) = colnames(data)
colSums(is.na(cleandata)) %>% mean
```

```
## [1] 0
```

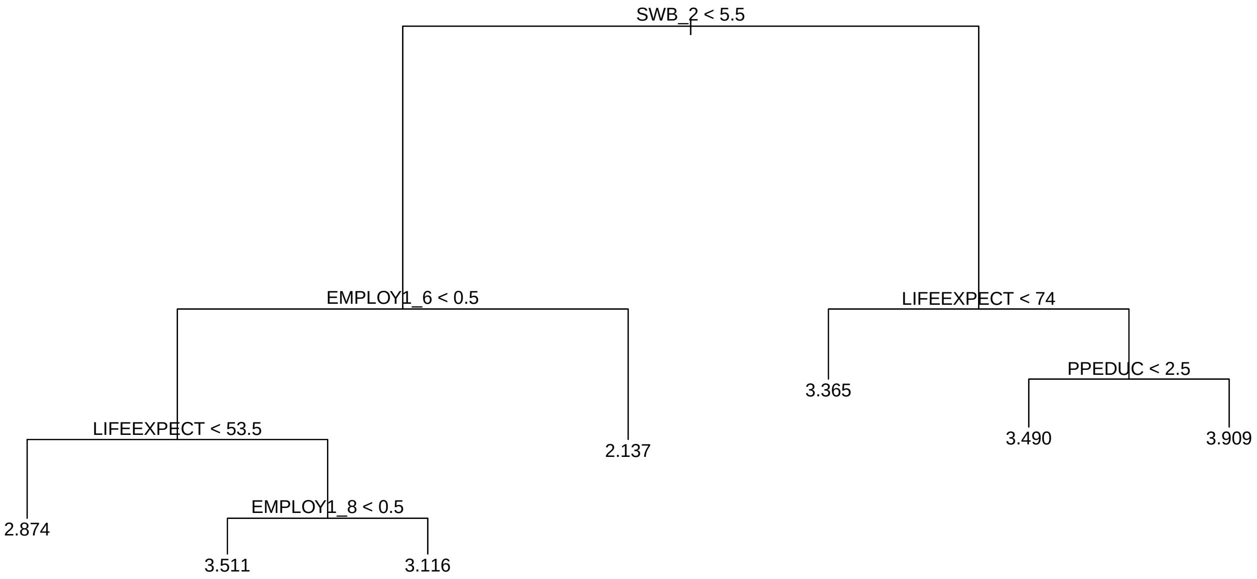
```
inpt = cleandata[, -which(colnames(cleandata) == "HEALTH")]
resp = cleandata$HEALTH
# separation of train and test data
testind = sample(1:nrow(cleandata), round(nrow(cleandata) * 0.2), replace = F)
train = cleandata[-testind, ]
test.x = inpt[testind, ]
test.y = resp[testind]
```

Decision Tree

```
fit.tree <- tree(HEALTH ~ ., data = train)
summary(fit.tree)
```

```
##
## Regression tree:
## tree(formula = HEALTH ~ ., data = train)
## Variables actually used in tree construction:
## [1] "SWB_2"      "EMPLOY1_6"  "LIFEEXPECT" "EMPLOY1_8"  "PPEDUC"
## Number of terminal nodes:  7
## Residual mean deviance:  0.6785 = 3466 / 5108
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.90900 -0.51120  0.09136  0.00000  0.50960  2.86300
```

```
plot(fit.tree)
text(fit.tree, pretty = 1)
```



True Error

```r
pred.tree <- predict(fit.tree, newdata = test.x)
mean((test.y - pred.tree)^2)
```

```
## [1] 0.6644506
```

Random Forest

```r
fit.rf <- randomForest(HEALTH ~ ., data = train, mtry = round(sqrt(ncol(train)-1)),
                       importance = TRUE, ntree = 10)
summary(fit.rf)
```

```
##                 Length Class  Mode
## call               6   -none- call
## type               1   -none- character
## predicted       5115   -none- numeric
## mse               10   -none- numeric
## rsq               10   -none- numeric
## oob.times       5115   -none- numeric
## importance       430   -none- numeric
## importanceSD     215   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               5115   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

True Error

```r
pred.rf <- predict(fit.rf, newdata = test.x)
mean((test.y - pred.rf)^2)
```

```
## [1] 0.6452461
```