# BACS HW17 - 109006234

Credit: 109006278

June 11th 2023

## Setup

```
dataset <- read.csv("G:/My Drive/111_2_BACS/HW17/insurance.csv")
dataset <- na.omit(dataset)

set.seed(987654321)
train_indices<- sample(1:nrow(dataset), size = 0.8*nrow(dataset))
train_set <- dataset[train_indices,]
test_set <- dataset[-train_indices,]
```

## Problem 1

### (a) Create an OLS regression model and report which factors are significantly related to charges

```
insurance_lm <- lm(charges ~ age + factor(sex) + bmi + factor(smoker) +
               factor(region), data=dataset)
summary(insurance_lm)
```

```
##
## Call:
## lm(formula = charges ~ age + factor(sex) + bmi + factor(smoker) +
##     factor(region), data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11852.6  -3010.9   -987.8   1515.8  29467.1
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -11556.96     985.63 -11.725   <2e-16 ***
## age                        258.54      11.94  21.658   <2e-16 ***
## factor(sex)male           -111.57     334.26  -0.334   0.7386
## bmi                        340.46      28.71  11.857   <2e-16 ***
## factor(smoker)yes        23862.91     414.82  57.526   <2e-16 ***
## factor(region)northwest   -304.10     478.01  -0.636   0.5248
## factor(region)southeast  -1039.20     480.65  -2.162   0.0308 *
## factor(region)southwest   -916.44     479.72  -1.910   0.0563 .
```

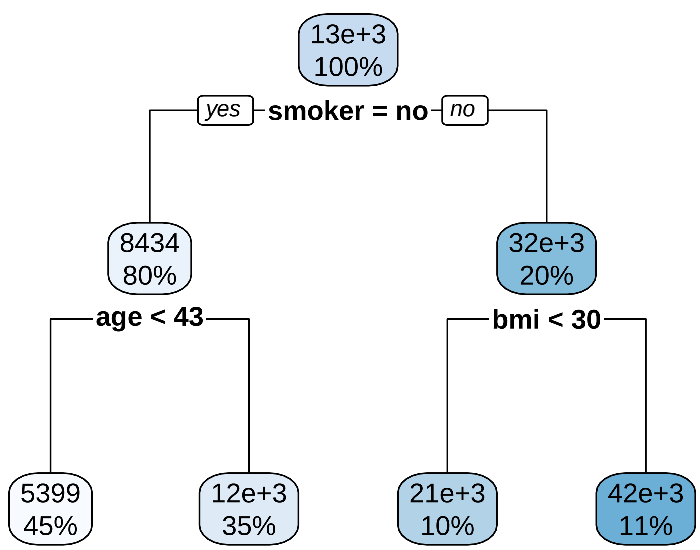
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6087 on 1330 degrees of freedom
## Multiple R-squared:  0.7487, Adjusted R-squared:  0.7474
## F-statistic:   566 on 7 and 1330 DF,  p-value: < 2.2e-16
```

### (b) Create a decision tree (specifically, a regression tree) with default parameters to rpart().

```
formula <- charges ~ bmi + age + sex + children + smoker + region
insurance_tree <- rpart(formula, data=dataset)
```

#### (i) Plot a visual representation of the tree structure

```
rpart.plot(insurance_tree)
```




#### (ii) How deep is the tree (see nodes with "decisions" – ignore the leaves at the bottom)

```
nodes <- as.numeric(rownames(insurance_tree$frame))
depth <- max(rpart:::tree.depth(nodes))
```

```
## [1] "Depth of the tree:  2"
```

### (iii) How many leaf groups does it suggest to bin the data into?

```
total_leaves <- sum(insurance_tree$frame$var == "<leaf>")
leaf_group <- total_leaves/depth
```

```
## [1] "Leaf groups:  2"
```

### (iv) What conditions (combination of decisions) describe each leaf group?

```
##  charges
##     5399 when smoker is  no & age <  43
##    12300 when smoker is  no & age >= 43
##    21369 when smoker is yes             & bmi <  30
##    41693 when smoker is yes             & bmi >= 30
```

# Problem 2

```
mse_oos <- function(actuals, preds) {
  sqrt(mean( (actuals - preds)^2 ))
}
```

## (a) What is the RMSEout for the OLS regression model?

```
mse_oos(dataset$charges, predict(insurance_lm, dataset))
```

```
## [1] 6068.683
```

## (b) What is the RMSEout for the decision tree model?

```
mse_oos(dataset$charges, predict(insurance_tree, dataset))
```

```
## [1] 5029.781
```

# Problem 3

## (a) Implement the bagged_learn(…) and bagged_predict(…) functions

```
bagged_retrain <- function(model, dataset, b){
  resample <- unique(sample(1:nrow(dataset), replace = TRUE))
  train_data <- dataset[resample,]
  train_model <- update(model, data = train_data)
  train_model
}
```

```r
bagged_learn <- function(model, dataset, b=100){
  lapply(1:b, bagged_retrain, model = model, dataset = dataset)
}

pred <- function(model, dataset, b){
  model = model[[b]]
  predict(model, dataset)
}

bagged_predict <- function(bagged_model, dataset, b){
  prediction <- lapply(1:b, pred, model = bagged_model, dataset = dataset)
  mse_oos(unlist(prediction), rep(unlist(dataset[7]), times = b))
}
```

### (b) What is the RMSEout for the bagged OLS regression?

```r
set.seed(987654321)
lm_bagged_models <- bagged_learn(insurance_lm, train_set, 100)
lm_bagged_mse <- bagged_predict(lm_bagged_models, test_set, 100)
```

```
## [1] "Bagged MSE of lm model:  6181.79899252622"
```

### (c) What is the RMSEout for the bagged decision tree?

```r
ins_tree_models <- bagged_learn(insurance_tree, train_set, 100)
ins_tree_bagged_mse <- bagged_predict(ins_tree_models, test_set, 100)
```

```
## [1] "Bagged MSE of rt model:  5164.90577720516"
```

## Problem 4

### (a) Write boosted_learn(...) and boosted_predict(...) functions

```r
boosted_learn <- function(model, dataset, outcome, n=100, rate=0.1, type) {
  predictors <- dataset[, -which(names(dataset) %in% outcome)]

  res <- dataset[,outcome]
  models <- list()
  for (i in 1:n) {
    this_model <- update(model, data = cbind(charges = res, predictors))
    if (type == "l") {
      res <- res - rate * this_model$fitted.values
    }
    else{
      res <- res - rate * this_model$y
    }
    models[[i]] <- this_model
```

```
  }
  list(models=models, rate=rate)
}

boost_predict <- function(boosted_learning, new_data) {
  boosted_models <- boosted_learning$models
  rate <- boosted_learning$rate
  n <- length(boosted_models)

  predictions = lapply(1:n, function(i){
    rate*predict(boosted_models[[i]], new_data)
  })

  pred_frame = as.data.frame(predictions)
  pred <- apply(pred_frame, 1, sum)
  mse_oos(pred, new_data[,7])
}
```

### (b) What is the RMSEout for the boosted OLS regression?

```
ins_lm_boosted_model <- boosted_learn(insurance_lm, train_set,
        outcome = "charges", type = "l")
ins_lm_boosted_mse <- boost_predict(ins_lm_boosted_model, test_set)
```

```
## [1] "Boosted MSE of rt model:  6168.08458226006"
```

### (c) What is the RMSEout for the boosted decision tree?

```
ins_tree_stump <- rpart(formula, data=dataset, cp=0, maxdepth=1)
ins_tree_boosted_model <- boosted_learn(ins_tree_stump, train_set,
                                        outcome="charges", type='t')
ins_tree_boost_mse <- boost_predict(ins_tree_boosted_model, test_set)
```

```
## [1] "Boosted MSE of rt model:  7340.3753136244"
```

# Problem 5

```
set.seed(987654321)
train_indices1<- sample(1:nrow(dataset), size = 0.7*nrow(dataset))
train_set1 <- dataset[train_indices1,]

remaining_indices <- setdiff(1:nrow(dataset), train_indices1)
test_indices1 <- sample(remaining_indices, size = 0.2*nrow(dataset))
test_set1 <- dataset[test_indices1,]

test_indices2 <- setdiff(remaining_indices, test_indices1)
test_set2 <- dataset[test_indices2,]

nrow(train_set1)
```

```
## [1] 936
```

```r
nrow(test_set1)
```

```
## [1] 267
```

```r
nrow(test_set2)
```

```
## [1] 135
```

### (a) Repeat the bagging of the decision tree, using a base tree of maximum depth 1, 2, ... n, keep training on the 70% training set while the RMSEout of your 20% set keeps dropping; stop when the RMSEout has started increasing again (show prediction error at each depth). Report the final RMSEout using the final 10% of the data as your test set.

```r
flag <- 0
maxdepth_ins <- 1
maxdepth_vector <- c()
bagged_mse <- c()

while(flag == 0){
  control <- rpart.control(maxdepth = maxdepth_ins, cp = 0)

  ins_tree <- rpart(formula, data = train_set1, control = control)
  ins_tree_models <- bagged_learn(ins_tree, train_set1, 100)
  ins_tree_bagged_mse <- bagged_predict(ins_tree_models, test_set2, 100)

  maxdepth_vector <- c(maxdepth_ins)
  bagged_mse <- c(bagged_mse, ins_tree_bagged_mse)

  if(length(bagged_mse) >= 2){
    if(bagged_mse[maxdepth_ins-1] < bagged_mse[maxdepth_ins]){
      flag <- 1
    }
  }
  maxdepth_ins <- maxdepth_ins + 1
}
```

```
##      maxdepth_vector bagged_mse
## [1,]               5    7086.36
## [2,]               5    4678.59
## [3,]               5    4349.96
## [4,]               5    4253.88
## [5,]               5    4274.36
## attr(,"names")
##  [1] "max_depth"  "bagged_mse" NA           NA           NA
##  [6] NA           NA           NA           NA           NA
```

**(b) Repeat the boosting of the decision tree, using a base tree of maximum depth 1, 2, ... n, keep training on the 70% training set while the RMSEout of your 20% set keeps dropping; stop when the RMSEout has started increasing again (show prediction error at each depth). Report the final RMSEout using the final 10% of the data as your test set.**

```r
flag <- 0
maxdepth_ins <- 1
maxdepth_vector <- c()
boosted_mse <- c()

while(flag == 0){
  control <- rpart.control(maxdepth = maxdepth_ins, cp = 0)

  ins_tree_stump <- rpart(formula, data=dataset, control = control)
  ins_tree_boosted_model <- boosted_learn(ins_tree_stump, train_set1,
                                          outcome="charges", type='t')
  ins_tree_boost_mse <- boost_predict(ins_tree_boosted_model, test_set2)

  maxdepth_vector <- c(maxdepth_ins)
  boosted_mse <- c(boosted_mse, ins_tree_boost_mse)

  if(length(boosted_mse) >= 2){
    if(boosted_mse[maxdepth_ins-1] < boosted_mse[maxdepth_ins]){
      flag <- 1 # break
    }
  }
  maxdepth_ins <- maxdepth_ins + 1
}
```

```
##      maxdepth_vector boosted_mse
## [1,]               6     7078.91
## [2,]               6     4668.05
## [3,]               6     4378.34
## [4,]               6     4086.00
## [5,]               6     4044.32
## [6,]               6     4148.12
## attr(,"names")
##  [1] "max_depth"   "boosted_mse" NA            NA            NA
##  [6] NA            NA            NA            NA            NA
## [11] NA            NA
```