Student ID: 112077423

```r
library(dplyr)
library(tidyr)
library(rpart)
```

```
## Warning:     'rpart'           R     4.3.3
```

```r
library(rpart.plot)
```

```
## Warning:     'rpart.plot'           R     4.3.3
```

```r
df <- read.csv('insurance.csv', header=TRUE)
df <- na.omit(df)
head(df)
```

```
##   age    sex    bmi children smoker    region   charges
## 1  19 female 27.900        0    yes southwest 16884.924
## 2  18   male 33.770        1     no southeast  1725.552
## 3  28   male 33.000        3     no southeast  4449.462
## 4  33   male 22.705        0     no northwest 21984.471
## 5  32   male 28.880        0     no northwest  3866.855
## 6  31 female 25.740        0     no southeast  3756.622
```

```r
str(df)
```

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : chr  "female" "male" "male" "male" ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : chr  "yes" "no" "no" "no" ...
##  $ region  : chr  "southwest" "southeast" "southeast" "northwest" ...
##  $ charges : num  16885 1726 4449 21984 3867 ...
```

```r
df <- df %>% mutate(across(where(is.character), as.factor))
```

```r
cor(df[, sapply(df, is.numeric)])
```

```
##                age       bmi   children    charges
## age      1.0000000 0.1092719 0.04246900 0.29900819
## bmi      0.1092719 1.0000000 0.01275890 0.19834097
## children 0.0424690 0.0127589 1.00000000 0.06799823
## charges  0.2990082 0.1983410 0.06799823 1.00000000
```

Question 1(a)

Create an OLS regression model and report which factors are significantly related to charges

```r
ols <- lm(charges ~ ., data=df)
summary(ols)
```

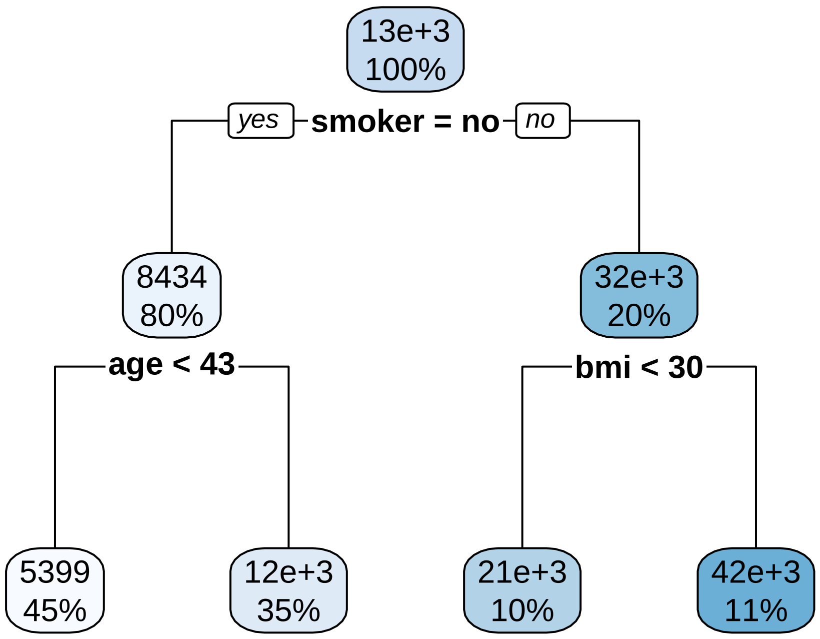
```
##
## Call:
## lm(formula = charges ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -11938.5      987.8 -12.086  < 2e-16 ***
## age                256.9       11.9  21.587  < 2e-16 ***
## sexmale           -131.3      332.9  -0.394 0.693348
## bmi                339.2       28.6  11.860  < 2e-16 ***
## children           475.5      137.8   3.451 0.000577 ***
## smokeryes        23848.5      413.1  57.723  < 2e-16 ***
## regionnorthwest   -353.0      476.3  -0.741 0.458769
## regionsoutheast  -1035.0      478.7  -2.162 0.030782 *
## regionsouthwest   -960.0      477.9  -2.009 0.044765 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6062 on 1329 degrees of freedom
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7494
## F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 2.2e-16
```

As it can be seen from the model summary, variables, such as children, age, bmi, smokeryes, regionsoutheast and regionsouthwest, are significant at alpha=0.05

Question 1(b)

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

```r
tree <- rpart(charges ~ ., data=df)
rpart.plot(tree)
```



- The depth of the tree = 2
- The total number of leaves = 4
  - The first group described **smoker=yes** and **age<43**
  - The second group described by **smoker=yes** and **age>=43**
  - The third group described by **smoker=no** and **bmi<30**
  - The fourth group described by **smoker=no** and **age>=30**

Question 2

Let's use LOOCV to see how how our models perform predictively overall

```
fold_i_pe <- function(i, k, model, dataset, outcome) {
  folds <- cut(1:nrow(dataset), breaks=k, labels=FALSE)
  test_indices <- which(folds==i)
  test_set <- dataset[test_indices, ]
  train_set <- dataset[-test_indices, ]
  trained_model <- update(model, data = train_set)
  predictions <- predict(trained_model, test_set)
  dataset[test_indices, outcome] - predictions
}

k_fold_mse <- function(model, dataset, outcome, k=nrow(dataset)) {
  shuffled_indicies <- sample(1:nrow(dataset))
  dataset <- dataset[shuffled_indicies,]
  fold_pred_errors <- sapply(1:k, \(kth) {
    fold_i_pe(kth, k, model, dataset, outcome)
  })
  pred_errors <- unlist(fold_pred_errors)
  sqrt(mean(pred_errors^2))
}

out1 <- paste('RMSEout of the OLS regression model =', k_fold_mse(ols, df, "charges", k=nrow(df)))
out2 <- paste('RMSEout of the decision tree model =', k_fold_mse(tree, df, "charges", k=nrow(df)))
cat(out1, out2, sep='\n')
```

```
## RMSEout of the OLS regression model = 6087.38800655031
## RMSEout of the decision tree model = 5135.1747343426
```

Question 3

Let's see if bagging helps our models

```
train_indices <- sample(1:nrow(df), size=0.80*nrow(df))
train_set <- df[train_indices,]
test_set <- df[-train_indices,]

bagged_learn <- function(model, dataset, b=100) {
  lapply(1:b, \(i) {
    # Get a bootstrapped (resampled w/ replacement) dataset
    bootstrapped_df <- dataset[sample(1:nrow(dataset), replace = TRUE),]
    # Return a retrained (updated) model
    update(model, data=bootstrapped_df)
  })
}

bagged_predict <- function(bagged_models, new_data) {
  # get b predictions of new_data
  predictions <- lapply(bagged_models, \(i) {predict(i, new_data)})
  # apply a mean over the rows of predictions
  as.data.frame(predictions, col.names = c(1:100)) %>%
    apply(1, mean)
}
```

```
trained_models <- bagged_learn(ols, train_set)
b_final <- bagged_predict(trained_models, test_set)

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

out1 <- paste('RMSEout of the bagged OLS regression =', rmse(test_set$charges, b_final))

trained_models <- bagged_learn(tree, train_set)
b_final <- bagged_predict(trained_models, test_set)
```

```r
out2 <- paste('RMSEout of the bagged decision tree =', rmse(test_set$charges, b_final))
cat(out1, out2, sep='\n')
```

```
## RMSEout of the bagged OLS regression = 5570.22093895996
## RMSEout of the bagged decision tree = 4263.4125047398
```

Question 4

Let's see if boosting helps our models.

```r
boost_learn <- function(model, dataset, outcome, n=100, rate=0.1) {
  # get data frame of only predictor variables
  predictors <- dataset[,!names(dataset) %in% c(outcome)]
  # Initialize residuals and models
  res <- dataset[, outcome] # set res to vector of actuals (y) to start
  models <- list()
  for (i in 1:n) {
    this_model <- update(model, data = cbind(charges=res, predictors))
    # update residuals with learning rate
    res <- res - rate * predict(this_model, predictors)
    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 <- nrow(new_data)
  predictions <- lapply(boosted_models, \(i) {predict(i, new_data)})
  pred_frame <- as.data.frame(predictions) |> unname()
  # apply a sum over the rows of predictions, weighted by learning rate
  apply(pred_frame, 1, \(row) {sum(rate*row)})
}
```

```r
boosted <- boost_learn(ols, train_set, 'charges', rate=0.3)
pred <- boost_predict(boosted, test_set)

out1 <- paste('RMSEout of the boosted OLS regression =', rmse(test_set$charges, pred))

boosted <- boost_learn(tree, train_set, 'charges', rate=0.3)
pred <- boost_predict(boosted, test_set)

out2 <- paste('RMSEout of the boosted decision tree =', rmse(test_set$charges, pred))
cat(out1, out2, sep='\n')
```

```
## RMSEout of the boosted OLS regression = 5575.13035086171
## RMSEout of the boosted decision tree = 3859.45315905424
```

Question 5(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 15% validation set keeps dropping; stop when the RMSEout has started increasing again (show prediction error at each depth). When you have identified the best maximum depth from the validation set, report the final RMSEout using the final 15% test set data.

```r
train_indices <- sample(1:nrow(df), size=0.70*nrow(df))
train_set <- df[train_indices,]
tmp <- df[-train_indices,]
val_indices <- sample(1:nrow(tmp), size=0.50*nrow(tmp))
val_set <- tmp[val_indices,]
test_set <- tmp[-val_indices,]

lowest_rmse <- 1000000000
best_max_d <- 1
models <- list()

for (i in 1:30) {
  tree_model <- rpart(charges ~ ., data=train_set, control=list(maxdepth=i))
  trained_models <- bagged_learn(tree_model, train_set)
  b_final <- bagged_predict(trained_models, val_set)
  tmp <- rmse(val_set$charges, b_final)
  cat('RMSEout at depth', i, '=', tmp, '\n', sep=' ')
  if (tmp > lowest_rmse) {
    best_max_d <- i - 1
    break
  }
  lowest_rmse <- tmp
  models[[i]] <- trained_models
}
```

```
## RMSEout at depth 1 = 4961.96
## RMSEout at depth 2 = 4930.604
## RMSEout at depth 3 = 4950.122
```

```r
b_final <- bagged_predict(models[[best_max_d]], test_set)
final_rmse <- rmse(test_set$charges, b_final)
cat('Final RMSEout', '=', final_rmse, '\n', sep=' ')
```

```
## Final RMSEout = 4287.161
```

Question 5(b)

Let's find the best set of max tree depth and learning rate for boosting the decision tree: Use tree stumps of differing maximum depth (e.g., try intervals between 1 – 5) and differing learning rates (e.g., try regular intervals from 0.01 to 0.20). For each combination of maximum depth and learning rate, train on the 70% training set while and use the 15% validation set to compute RMSEout. When you have tried all your combinations, identify the best combination of maximum depth and learning rate from the validation set, but report the final RMSEout using the final 15% test set data.

```r
results <- expand.grid(max_depth=1:5, learning_rate=seq(0.01, 0.20, by = 0.01), RMSE = NA)

for (i in 1:nrow(results)) {
  depth <- results$max_depth[i]
  rate <- results$learning_rate[i]

  tree_model <- rpart(charges ~ ., data=train_set, control=list(maxdepth=depth))
  boosted <- boost_learn(tree_model, train_set, 'charges', rate=rate)
  pred <- boost_predict(boosted, val_set)

  results$RMSE[i] <- rmse(val_set$charges, pred)
}

best_params <- results[which.min(results$RMSE), ]
print(best_params)
```

```
##    max_depth learning_rate     RMSE
## 89         4          0.18 4721.325
```

```r
best_depth <- best_params$max_depth
best_rate <- best_params$learning_rate

final_tree <- rpart(charges ~ ., data=train_set, control=list(maxdepth=best_depth))
boosted <- boost_learn(final_tree, train_set, 'charges', rate=best_rate)
pred <- boost_predict(boosted, test_set)

final_rmse <- rmse(test_set$charges, pred)
cat('Final RMSEout', '=', final_rmse, '\n', sep=' ')
```

```
## Final RMSEout = 3979.429
```