HW17

107070008

```
insurance <- read.csv("insurance.csv")
insur <- na.omit(insurance)
```

Question 1) Create some explanatory models to learn more about charges:

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

```
ols_lm <- lm(charges ~ age + factor(sex) + bmi + children +
               factor(smoker) + factor(region), data = insur)
summary(ols_lm)
```

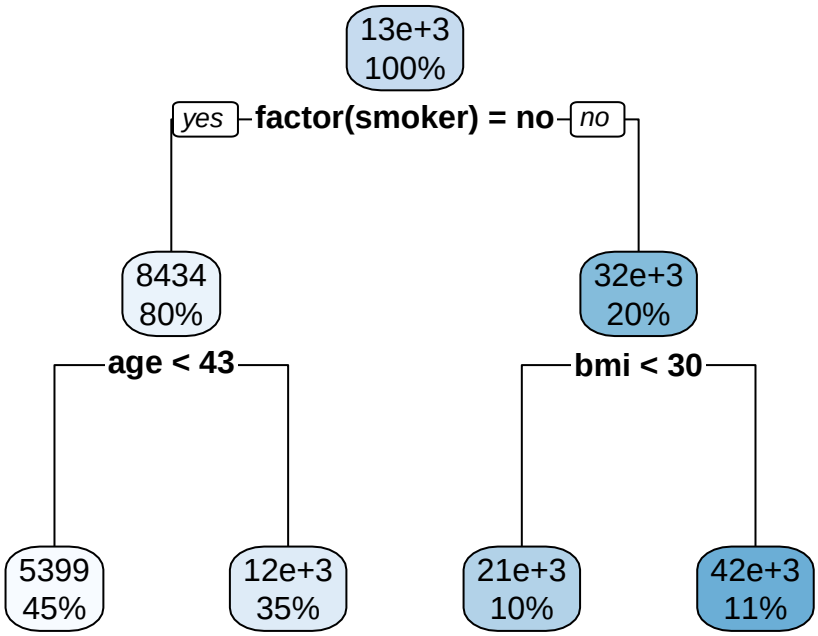
```
##
## Call:
## lm(formula = charges ~ age + factor(sex) + bmi + children + factor(smoker) +
##     factor(region), data = insur)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -11938.5      987.8 -12.086  < 2e-16 ***
## age                        256.9       11.9  21.587  < 2e-16 ***
## factor(sex)male           -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 ***
## factor(smoker)yes        23848.5      413.1  57.723  < 2e-16 ***
## factor(region)northwest   -353.0      476.3  -0.741 0.458769
## factor(region)southeast  -1035.0      478.7  -2.162 0.030782 *
## factor(region)southwest   -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
```

b. Create a decision (regression) tree with default parameters

i. Plot a visual representation of the tree

```
library(rpart)
library(rpart.plot)
tree_lm <- rpart(charges ~ age + factor(sex) + bmi +
                   children + factor(smoker) + factor(region), data = insur)

rpart.plot(tree_lm)
```



ii. How deep is the tree (see nodes with "decisions" – ignore the leaves at the bottom) `2`

iii. How many leaf groups does it suggest to bin the data into? `4`

iv. What is the average charges of each leaf group? `2705.2182, 8.4364`

v. What conditions (decisions) describe each group? `age < 43, bmi < 30`

Question 2) Let's use LOOCV to se how how our models perform predictively

```
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
}

#library(Metrics)
k_fold_mse <- function(model, dataset, outcome, k){
    shuffled_indicies <- sample(1:nrow(dataset))
    dataset <- dataset[shuffled_indicies,]
    fold_pred_errors <- sapply(1:k, function(kth) {
        fold_i_pe(kth, k, model, dataset, outcome)
      })
    pred_errors <- unlist(fold_pred_errors)

    mse <- function(errs){mean(errs^2)}
    c(is = mse(residuals(model)), oos = mse(pred_errors))
}
```

a. What is the RMSEoos for the OLS regression model?

```
ols_kfm <- k_fold_mse(ols_lm, insur, "charges", 10)
sqrt(ols_kfm[2])
```

```
##      oos
## 6089.327
```

b. What is the RMSEoos for the decision tree model?

```
tree_kfm <- k_fold_mse(tree_lm, insur, "charges", 10)
sqrt(tree_kfm[2])
```

```
##      oos
## 5110.174
```

Question 3) Let's see if bagging helps our models

a. Write bagged_learn(...) and bagged_predict(...) functions using the hints in the class notes and help from your classmates on Teams. Feel free to share your code generously on Teams to get feedback, or ask others for help.

```
set.seed(27935752)
train_indices <- sample(1:nrow(insur), size = 0.80*nrow(insur))
train_set <- insur[train_indices,]
test_set <- insur[-train_indices,]

bagged_learn <- function(model, dataset, b=100) {
    lapply(1:b, function(i) {
       boot_index <- sample(nrow(dataset), replace = TRUE)
       boot_dataset <- dataset[boot_index,]
       boot_model <- update(model, data=boot_dataset)
       return(boot_model)
    })
}
bagged_predict <- function(bagged_models, new_data, b=100) {
    predictions <- lapply(1:b, function(i){
      predict(bagged_models[[i]], new_data)
    })
    apply(as.data.frame(predictions), 1, mean)
}

mse_oos <- function(actual, pred){
  sqrt(mean((actual-pred)^2))
}
```

b. What is the RMSEoos for the bagged OLS regression?

```
bagged_model <- bagged_learn(ols_lm, train_set, 100)
bagged_prediction <- bagged_predict(bagged_model, test_set, 100)

mse_oos(test_set$charges, bagged_prediction)
```

```
## [1] 6022.205
```

c. What is the RMSEoos for the bagged decision tree?

```
bagged_model <- bagged_learn(tree_lm, train_set, 100)
bagged_prediction <- bagged_predict(bagged_model, test_set, 100)

mse_oos(test_set$charges, bagged_prediction)
```

```
## [1] 4907.184
```

Question 3) Let's see if boosting helps our models. You can use a learning rate of 0.1 and adjust it if you find a better rate.

a. Write boosted_learn(...) and boosted_predict(...) functions using the hints in the class notes and help from your classmates on Teams. Feel free to share your code generously on Teams to get feedback, or ask others for help.

```
boost_learn <- function(model, dataset, n=100, rate=0.1) {
    predictors <- dataset[, 1:6]

    res <-dataset[, 7]
    models <- list()

    for(i in 1:n) {
      this_model <- update(model, data = cbind(charges=res, predictors))
      res <- res - rate*predict(this_model)
      models[[i]] <-this_model
    }
    list(models=models, rate=rate)
}

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

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

    pred_frame <- unname(as.data.frame(predictions))
    apply(pred_frame, 1, sum)
}
```

b. What is the RMSEoos for the boosted OLS regression?

```
boosted_model <- boost_learn(ols_lm, train_set, 100, 0.1)
boost_prediction <- boost_predict(boosted_model, test_set)

mse_oos(test_set$charges, unlist(boost_prediction))
```

```
## [1] 6020.292
```

c. What is the RMSEoos for the boosted decision tree?

```
boosted_model <- boost_learn(tree_lm, train_set, 100, 0.1)
boost_prediction <- boost_predict(boosted_model, test_set)

mse_oos(test_set$charges, unlist(boost_prediction))
```

```
## [1] 4494.743
```

Question 4) Let's engineer the best predictive decision trees. Let's repeat the bagging and boosting decision tree several times to see what kind of base tree helps us learn the fastest. Report the RMSEoos at each step.

a. Repeat the bagging of the decision tree, using a base tree of maximum depth 1, 2, ... n while the RMSEoos keeps dropping; stop when the RMSEoos has started increasing again.

```
pre <- 1000000
for(i in 1:30){
  tree_stump <- rpart(charges ~ age + factor(sex) + bmi +
                        children + factor(smoker) + factor(region),
                      data = insur, cp=0, maxdepth=i)

  bagged_model <- bagged_learn(tree_lm, train_set, 30)
  bagged_prediction <- bagged_predict(bagged_model, test_set, 30)
  rmse_tree <- mse_oos(test_set$charges, bagged_prediction)
  cat(rmse_tree , sep = "\n")
  if(rmse_tree > pre) break

  pre <- rmse_tree
}
```

```
## 4904.416
## 4924.96
```

b. Repeat the boosting of the decision tree, using a base tree of maximum depth 1, 2, ... n while the RMSEoos keeps dropping; stop when the RMSEoos has started increasing again.

```
pre <- 1000000
for(i in 1:30){
  tree_stump <- rpart(charges ~ age + factor(sex) + bmi +
                          children + factor(smoker) + factor(region),
                        data = insur, cp=0, maxdepth=i)

  boosted_model <- boost_learn(tree_stump, train_set, 30, 0.1)
```

```
  boost_prediction <- boost_predict(boosted_model, test_set)
  rmse_tree <- mse_oos(test_set$charges, unlist(boost_prediction))
  cat(rmse_tree , sep = "\n")
  if(rmse_tree > pre) break

  pre <- rmse_tree

}
```

```
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
## 4743.061
```