HW17

109006206

Set Working Directories & Reading Files

```
setwd("/Users/olivia/Documents/Documents/Study/Semester 6/BACS/HW17")
library(rpart)
library(rpart.plot)

insurance <- read.csv("insurance.csv", header = TRUE)
insurance <- na.omit(insurance)
insurance$charges <- as.numeric(insurance$charges)
```

QUESTION 1

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

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

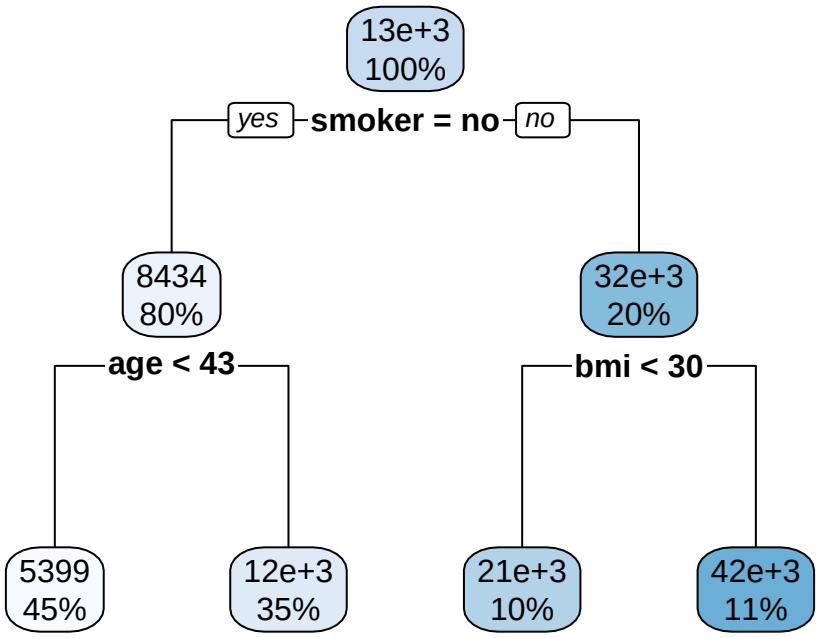
```
##
## Call:
## lm(formula = charges ~ age + factor(sex) + bmi + children + factor(smoker) +
##     factor(region), data = insurance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##                           Estimate 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 tree (specifically, a regression tree) with default parameters to rpart().

I) Plot a visual representation of the tree structure

```
tree_model <- rpart(charges ~ age + sex + bmi + children + smoker + region, data = insurance)
rpart.plot(tree_model)
```



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

```
min_cp_row <- which.min(tree_model$cptable[, "CP"])
max_depth <- tree_model$cptable[min_cp_row, "nsplit"]
print(max_depth-1)
```

```
## [1] 2
```

III) How many leaf groups does it suggest to bin the data into?

```
min_xerror_row <- which.min(tree_model$cptable[, "xerror"])
num_leaf_groups <- tree_model$cptable[min_xerror_row, "nsplit"] + 1
```

```
print(num_leaf_groups)
```

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

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

```
rules <- rpart.rules(tree_model,"tall")
print(rules)
```

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

QUESTION 2

A) What is the RMSEout for the OLS regression model?

```
models <- list(
  "ols_regr" = lm(charges ~ age + sex + bmi + children + smoker + region, data = insurance),
  "decision_tree" = rpart(charges ~ age + sex + bmi + children + smoker + region, data = insurance)
)



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))
}

# Calculate RMSEout for each model using LOOCV
models_loocv <-sapply(models, \(m) k_fold_mse(m, insurance, "charges", nrow(insurance)))
print(models_loocv[1])
```

```
## ols_regr
## 6087.388
```

B) What is the RMSEout for the decision tree model?

```
print(models_loocv[2])
```

```
## decision_tree
##      5135.175
```

QUESTION 3

A) Implement the 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 on Teams to get feedback, or ask others for help.

```
set.seed(123)

bagged_learn <- function(model, dataset, b = 100) {
  bagged_models <- lapply(1:b, function(i) {
    resampled_data <- dataset[sample(nrow(dataset), replace = TRUE), ]

    # 2. Train the model on the bootstrapped dataset
    trained_model <- update(model, data = resampled_data)

    # Return the retrained model
    trained_model
  })

  # Return the list of bagged models
  bagged_models
}

bagged_predict <- function(bagged_models, new_data) {
  predictions <- lapply(bagged_models, function(model) {
    # Get predictions of new_data using each bagged model
    predict(model, newdata = new_data)
  })

  # Apply mean over the columns of predictions
  predictions <- as.data.frame(predictions)
  mean_predictions <- apply(predictions, 1, mean)

  # Return the mean predictions
  mean_predictions
}
train_indices <- sample(1:nrow(insurance), size = 0.8 * nrow(insurance))
train_set <- insurance[train_indices, ]
test_set <- insurance[-train_indices, ]
```

B) What is the RMSEout for the bagged OLS regression?

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

```
}
bagged_models <- bagged_learn(model, train_set, b = 100)

bagged_predictions <- bagged_predict(bagged_models, test_set)
rmse_out <- sqrt(mse_oos(test_set$charges, bagged_predictions))
rmse_out
```

```
## [1] 5768.367
```

C) What is the RMSEout for the bagged decision tree?

```
bagged_modelsrt <- bagged_learn(tree_model, train_set, b = 100)
bagged_predictionsrt <- bagged_predict(bagged_modelsrt, test_set)
rmse_out2 <- sqrt(mse_oos(test_set$charges, bagged_predictionsrt))
rmse_out2
```

```
## [1] 4912.698
```

QUESTION 4

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, outcome, n = 100, rate = 0.1) {
  predictors <- dataset[, !names(dataset) %in% c(outcome)] # get data frame of only predictor variables

  # Initialize residuals and models
  res <- dataset[, names(dataset) %in% c(outcome)] # set res to vector of actuals (y) to start
  models <- c()

  for (i in 1:n) {
    t_data<-cbind(res, predictors)
    colnames(t_data)<- c("charges" ,colnames(predictors))
    this_model <- update(model, data = t_data)
    res <- res - rate*predict(this_model)
    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
  predict_boost<-function(x){
  predict(x, new_data)
  }
  sum_rate<-function(x){
  sum(x)*rate
  }
  n <- nrow(new_data)
  predictions <- lapply(boosted_models,predict_boost ) # get predictions of new_data from each model
  pred_frame <- as.data.frame(predictions) |> unname()
  apply(pred_frame,1,sum_rate)
}
```

B) What is the RMSEout for the boosted OLS regression?

```
boosted_model3 <- boost_learn(model, train_set,"charges", n = 100, rate = 0.1)
# Step 2: Predict using the boosted model
boosted_predictions3 <- boost_predict(boosted_model3, test_set)
```

```r
# Step 3: Calculate RMSEout
rmse_out <- sqrt(mean((test_set$charges - boosted_predictions3)^2))

# Print the RMSEout
rmse_out
```

```
## [1] 5763.366
```

C) What is the RMSEout for the boosted decision tree?

```r
boosted_modelrt <- boost_learn(tree_model, train_set,"charges", n = 100, rate = 0.1)
# Step 2: Predict using the boosted model
boosted_predictionsrt <- boost_predict(boosted_modelrt, test_set)

# Step 3: Calculate RMSEout
rmse_outrt <- sqrt(mean((test_set$charges - boosted_predictionsrt)^2))

# Print the RMSEout
rmse_outrt
```

```
## [1] 4435.965
```

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 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.

```
n <- nrow(test_set)
train_indices <- sample(1:nrow(test_set), size = 0.7 * nrow(test_set))
remaining_indices <- setdiff(1:n, train_indices)
fine_tune_indices <- sample(remaining_indices, floor(0.2 * n))
test_indices <- setdiff(remaining_indices, fine_tune_indices)

train_set <- test_set[train_indices, ]
fine_tune_set <- test_set[fine_tune_indices,]
test_set <- test_set[test_indices, ]

# Create a function to calculate RMSE_out
rmse_out <- function(predictions, actual) {
  sqrt(mean((actual - predictions)^2))
}

# Initialize variables
max_depth <- 1
prev_rmse_out <- Inf
rmse_out_list <- c()
models <- list()

# Perform bagging with increasing maximum depth
while (TRUE) {
  # Train the decision tree model
  model <- rpart(charges ~ age + sex + bmi + children + smoker + region,
                 data = train_set, maxdepth = max_depth)

  # Bagged learning
  bagged_models <- bagged_learn(model, train_set)

  models[[max_depth]] <- bagged_models
  # Make predictions on the fine-tune set
  predictions <- bagged_predict(bagged_models, fine_tune_set)

  # Calculate RMSE_out on the fine-tune set
  rmse <- rmse_out(predictions, fine_tune_set$charges)
```

```
  # Store RMSE_out
  rmse_out_list[max_depth] <- rmse

  # Print prediction error at each depth
  cat("Prediction error at depth", max_depth, ":", rmse, "\n")

  # Check if RMSE_out starts increasing
  if (rmse > prev_rmse_out) {
    break
  }

  # Update variables for the next iteration
  prev_rmse_out <- rmse
  max_depth <- max_depth + 1
}
```

```
## Prediction error at depth 1 : 7485.539
## Prediction error at depth 2 : 4696.465
## Prediction error at depth 3 : 4584.179
## Prediction error at depth 4 : 4596.434
```

```
# Report the final RMSE_out using the test set
final_predictions <- bagged_predict(models[[max_depth - 1]], test_set)
final_rmse_out <- rmse_out(final_predictions, test_set$charges)
cat("Final RMSE_out using the last 10% of the data as the test set:", final_rmse_out, "\n")
```

```
## Final RMSE_out using the last 10% of the data as the test set: 5571.31
```

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.

```
max_depth <- 1
prev_rmse_out <- Inf
rmse_out_list <- c()
models <- list()

# Perform bagging with increasing maximum depth
while (TRUE) {
  # Train the decision tree model
  model <- rpart(charges ~ age + sex + bmi + children + smoker + region,
                 data = train_set, maxdepth = max_depth)

  # Bagged learning
```

```
  boost_models <- boost_learn(model, train_set,"charges")

  models[[max_depth]] <- boost_models
  # Make predictions on the fine-tune set
  predictions <- boost_predict(boost_models, fine_tune_set)

  # Calculate RMSE_out on the fine-tune set
  rmse <- rmse_out(predictions, fine_tune_set$charges)

  # Store RMSE_out
  rmse_out_list[max_depth] <- rmse

  # Print prediction error at each depth
  cat("Prediction error at depth", max_depth, ":", rmse, "\n")

  # Check if RMSE_out starts increasing
  if (rmse > prev_rmse_out) {
    break
  }

  # Update variables for the next iteration
  prev_rmse_out <- rmse
  max_depth <- max_depth + 1
}
```

```
## Prediction error at depth 1 : 5674.267
## Prediction error at depth 2 : 4528.086
## Prediction error at depth 3 : 4825.55
```

```
# Report the final RMSE_out using the test set
final_predictions <- boost_predict(models[[max_depth - 1]], test_set)
final_rmse_out <- rmse_out(final_predictions, test_set$charges)
cat("Final RMSE_out using the last 10% of the data as the test set:", final_rmse_out, "\n")
```

```
## Final RMSE_out using the last 10% of the data as the test set: 5504.635
```