HW_Week17_108020033

Che-Wei, Chang

2023-06-10 helped by 108020024

This week, we will look at a dataset of US health insurance premium charges. We will build models that can predict what someone's insurance charges might be, given several factors about them. You download the dataset, and find more information about it, at the Kaggle platform where machine learning people like to host challenges and share datasets: https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset

Setup: Download the data, load it in your script, and omit any rows with missing values (NAs)

Note: The values of charges are large, so MSE values will be very large. This week let's use RMSE, or the Root-Mean-Square Error (the square-root of MSE), so we have smaller numbers.

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

```
# import the library
library("rpart")
library("rpart.plot")

# Load the dataset
insurance <- read.csv("insurance.csv")

# Drop rows with missing values
insurance <- na.omit(insurance)
```

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

```
# Fit the OLS regression model
model_ols <- lm(charges ~ age + factor(sex) + bmi + children +
                factor(smoker) + factor(region), data =  insurance)

# Show the summary of the model
summary(model_ols)
```

```
##
## 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
```

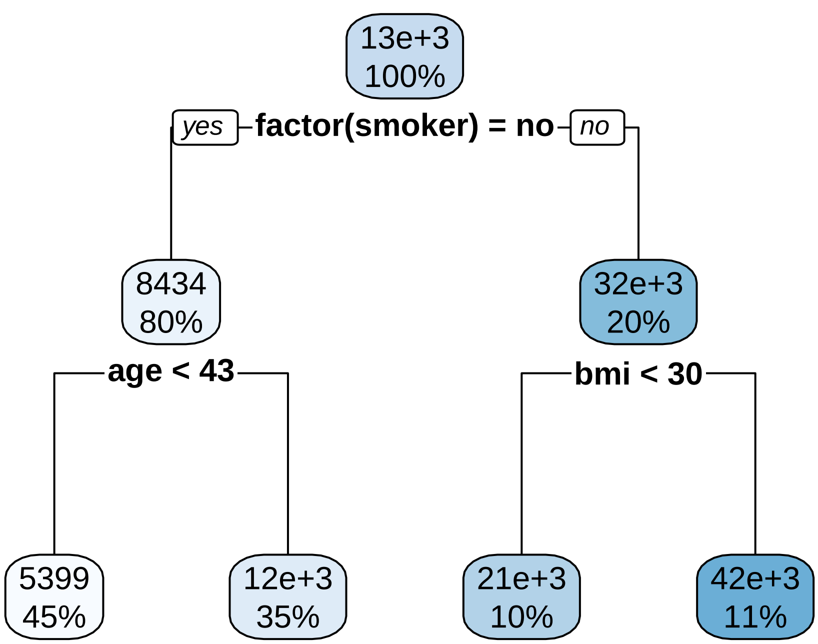
```
# By the table, we can find that age, bmi, children, smoker, (region)southeast
# (region)southwest are significantly related to charges
```

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

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

```
# Create and fit decision tree model
model_tree <- rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                    factor(region), data = insurance)

#Show the plot
rpart.plot(model_tree)
```



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

```
# Assume the level of root is 0, then the deepest node is at level 2.
```

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

```
# By the plot above, there are 4 leaf groups.
```

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

```
# Use the following function to show the conditions describe each leaf group
rpart.rules(model_tree)
```

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

```
# By the table, we can conclude that conditions describe:
# Smoker: yes or no
# age: < 43 or >= 43
# bmi: < 30 or >= 30)
```

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

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

```r
# import the library
library("caret")
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
# Perform LOOCV for OLS regression
ols_loocv <- train(charges ~ age + factor(sex) + bmi + children + factor(smoker) + factor(region),
                   data = insurance, method = "lm", trControl = trainControl(method = "LOOCV"),
                   metric = "RMSE")

# Show the RMSEout
ols_loocv$results$RMSE
```

```
## [1] 6087.388
```

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

```r
# Perform LOOCV for decision tree
tree_loocv <- train(charges ~ age + factor(sex) + bmi + children + factor(smoker) + factor(region),
                    data = insurance, method = "rpart", tuneGrid = expand.grid(cp = seq(0.01)),
                    trControl = trainControl(method = "LOOCV"), metric = "RMSE")

# Calculate RMSEout
tree_loocv$results$RMSE
```

```
## [1] 12114.54
```

For bagging and boosting, we will only use split-sample testing to save time: partition the data to create training and test sets using an 80:20 split. Use the regression model and decision tree you created in Question 1 for bagging and boosting.

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

```
# Preprocessing the data
set.seed(123)
sample <- sample(c(TRUE, FALSE), nrow(insurance), replace = TRUE, prob = c(0.8, 0.2))
training_set  <- insurance[sample, ]
testing_set   <- insurance[!sample, ]
```

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.

```
bagged_learn=function(model,dataset, b = 100){
  lapply(1 : b,\(i){
    boot_index = sample(1 : nrow(dataset), nrow(dataset), replace=T)
    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,\(i){
    predict(bagged_models[[i]], new_data)
  })
  apply(as.data.frame(predictions), 1, mean)
}
```

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

```
bagged_model = bagged_learn(model_ols, training_set, b = 100)
bagged_predictions = bagged_predict(bagged_model, testing_set)

rmse_oos=function(actuals,preds){
  sqrt(mean((actuals - preds) ^ 2))
}

rmse_oos(testing_set$charges,unlist(bagged_predictions))
```

```
## [1] 5816.421
```

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

```
bagged_learn(model_tree,insurance,b=100) |>
  bagged_predict(testing_set) |>
  rmse_oos(testing_set$charges)
```

```
## [1] 4806.968
```

Question 4) 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, outcome, n = 100,rate = 0.1){
  predictors = dataset[, !names(dataset) %in% c(outcome)]
  res = dataset[, outcome]
  models = list()
  for(i in 1 : n) {
    this_model = update(model, data = cbind(charges = res,predictors))
    res = res - (rate * predict(this_model, dataset))
    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_learning$models)
  predictions = lapply(1:n, \(i){
    rate * predict(boosted_models[[i]], new_data)
  })
  pred_frame = as.data.frame(predictions) |> unname()
  apply(pred_frame, 1, sum)
}
```

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

```
boosted_model = boost_learn(model_ols, training_set, "charges", n = 100, rate = 0.1)
boosted_predictions = boost_predict(boosted_model, testing_set)
rmse_oos(testing_set$charges, unlist(boosted_predictions))
```

```
## [1] 5816.328
```

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

```
boost_learn(model_tree, insurance, "charges", n = 100) |>
  boost_predict(testing_set) |>
  rmse_oos(testing_set$charges)
```

```
## [1] 4302.502
```

Question 5) 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. But this time, split the data 70:20:10 — use 70% for training, 20% for fine-tuning, and use the last 10% to report the final RMSEout.

```
set.seed(123)
sample <- sample(c(TRUE, FALSE), nrow(insurance), replace = TRUE, prob = c(0.7, 0.3))
training_set  <- insurance[sample, ]
temp   <- insurance[!sample, ]
sample <- sample(c(TRUE, FALSE), nrow(temp), replace = TRUE, prob = c(2/3, 1/3))
tuneing_set  <- temp[sample, ]
testing_set   <- temp[!sample, ]
```

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.

```
ctrl = rpart.control(maxdepth = 1)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
bagged_learn(model_tree, insurance, b = 100) |>
  bagged_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 6632.417
```

```
ctrl = rpart.control(maxdepth = 2)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control=ctrl, data = insurance)
bagged_learn(model_tree, insurance, b = 100) |>
  bagged_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 4453.761
```

```
ctrl = rpart.control(maxdepth = 3)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
bagged_learn(model_tree, insurance, b = 100) |>
  bagged_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 4298.156
```

```
ctrl = rpart.control(maxdepth = 4)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
bagged_learn(model_tree, insurance, b = 100) |>
  bagged_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 4307.454
```

```r
ctrl = rpart.control(maxdepth = 4)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
bagged_learn(model_tree, insurance, b = 100) |>
  bagged_predict(testing_set) |>
  rmse_oos(testing_set$charges)
```

```
## [1] 4547.703
```

```r
# By the result above, we can find that max depth is 4 and final RMSEout is 4547.703
```

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
ctrl = rpart.control(maxdepth = 1)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
boost_learn(model_tree, insurance, "charges", n = 100) |>
  boost_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 5180.95
```

```r
ctrl = rpart.control(maxdepth = 2)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
boost_learn(model_tree, insurance, "charges", n = 100) |>
  boost_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 3875.175
```

```r
ctrl = rpart.control(maxdepth = 3)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
boost_learn(model_tree, insurance, "charges", n = 100) |>
  boost_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 3836.412
```

```r
ctrl = rpart.control(maxdepth = 4)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
boost_learn(model_tree, insurance, "charges", n = 100) |>
  boost_predict(tuneing_set) |>
  rmse_oos(tuneing_set$charges)
```

```
## [1] 3840.179
```

```r
ctrl = rpart.control(maxdepth = 4)
model_tree = rpart(charges ~ age + factor(sex) + bmi + children + factor(smoker) +
                   factor(region), control = ctrl, data = insurance)
boost_learn(model_tree, insurance, "charges",n = 100) |>
  boost_predict(testing_set) |>
  rmse_oos(testing_set$charges)
```

```
## [1] 4052.252
```

```r
# By the result above, we can find that max depth is 4 and final RMSEout is 4052.252
```