CS 422 Homework 5

Jane Downer

10/16/2020

Part 2.1-A

```
setwd("/Users/user/Desktop/CS_422/Jane-Downer-HW5")

# Quick pre-processing
df_hotel <- read.csv("hotel_bookings.csv", sep=",", header=T)
df_hotel = df_hotel %>% mutate(adr = replace(adr, adr>1000, mean(adr)))
df_hotel <- df_hotel %>% mutate(stays_nights = stays_in_weekend_nights + stays_in_week_nights,
                                total_cost = adr * stays_nights)
df_hotel[, "reserved_room_type"] <- sapply(df_hotel[, "reserved_room_type"], as.character)
df_hotel[, "assigned_room_type"] <- sapply(df_hotel[, "assigned_room_type"], as.character)
df_hotel[df_hotel$reserved_room_type == df_hotel$assigned_room_type,"received_room_pref"] <- "Yes"
df_hotel[df_hotel$reserved_room_type != df_hotel$assigned_room_type,"received_room_pref"] <- "No"
df_hotel <- filter(df_hotel, !(adults == 0 & children == 0 & babies == 0))
df_hotel <- filter(df_hotel, !(stays_in_week_nights == 0 & stays_in_weekend_nights== 0))
df_hotel <- df_hotel[!apply(is.na(df_hotel) | df_hotel == "", 1, all),]

# Training the data & building a tree
index <- sample(1:nrow(df_hotel), size=0.1*nrow(df_hotel))
test <- df_hotel[index, ]
train <- df_hotel[-index, ]
train2 <- train[c('hotel', 'is_canceled', 'lead_time', 'adults', 'children', 'babies', 'meal',
                  'market_segment', 'distribution_channel', 'is_repeated_guest', 'previous_cancellations',
                  'previous_bookings_not_canceled', 'received_room_pref', 'deposit_type',
                  'days_in_waiting_list', 'customer_type', 'adr', 'required_car_parking_spaces',
                  'stays_nights','total_cost')]

test2 <- test[c('hotel', 'is_canceled', 'lead_time', 'adults', 'children', 'babies', 'meal',
                  'market_segment', 'distribution_channel', 'is_repeated_guest', 'previous_cancellations',
                  'previous_bookings_not_canceled', 'received_room_pref', 'deposit_type',
                  'days_in_waiting_list', 'customer_type', 'adr', 'required_car_parking_spaces',
                  'stays_nights','total_cost')]

mytree <- rpart(is_canceled ~., control=rpart.control(cp = 0.0), method = "class", data = train2)
```

Part 2.1-B

```
## Number of splits in the unpruned tree: 2636
```

Part 2.1-C

```
pred <- predict(mytree, test, type="class")
pred.prob <- predict(mytree, test, type="prob")
conmat<- confusionMatrix(pred, as.factor(test$is_canceled))

acc <- round(conmat$overall[1], digits=4)
bal_acc <- round(conmat$byClass[11], digits=4)
spec <- round(conmat$byClass[2], digits=4)
sens <- round(conmat$byClass[1], digits=4)
prec <- round(conmat$byClass[5], digits=4)

cat(paste0("Before pruning:\nAccuracy: ", acc,
           "\nError: ", 1-acc,
           "\nBalanced Accuracy: ", bal_acc,
           "\nSpecificity: ", spec,
           "\nSensitivity: ", sens,
           "\nPrecision: ", prec,"\n"))
```

```
## Before pruning:
## Accuracy: 0.8068
## Error: 0.1932
## Balanced Accuracy: 0.7865
## Specificity: 0.7042
## Sensitivity: 0.8688
## Precision: 0.8295
```

Part 2.1-D

```
min_xerr <- mytree$cptable[which.min(mytree$cptable[,"xerror"]), "xerror"]
cpx <- mytree$cptable[which.min(mytree$cptable[,"xerror"]), "CP"]
pruned<-prune(mytree, cpx)
cat(paste0("Prune point occurs at a complexity of ", round(cpx, digits=4), ".\n",
           "At this complexity, xerror is ", round(min_xerr, digits=4), "."))
```

```
## Prune point occurs at a complexity of 1e-04.
## At this complexity, xerror is 0.5104.
```

Part 2.1-E

```
pred <- predict(pruned, test, type="class")
pred.prob <- predict(pruned, test, type="prob")
conmat<- confusionMatrix(pred, as.factor(test$is_canceled))

acc <- round(conmat$overall[1], digits=4)
bal_acc <- round(conmat$byClass[11], digits=4)
spec <- round(conmat$byClass[2], digits=4)
sens <- round(conmat$byClass[1], digits=4)
prec <- round(conmat$byClass[5], digits=4)

cat(paste0("After pruning:",
           "\n\nAccuracy: ", acc,
           "\nBalanced Accuracy: ", bal_acc,
           "\nSpecificity: ", spec,
           "\nSensitivity: ", sens,
           "\nPrecision: ", prec, "\n"))
```

```
## After pruning:
##
## Accuracy: 0.8136
## Balanced Accuracy: 0.788
## Specificity: 0.6843
## Sensitivity: 0.8917
## Precision: 0.8239
```

Part 2.1-F

```
## The pruned tree is (mildly) better.
```

Part 2.2-i

```
train2$children[is.na(train2$children)] <- 0
train2$is_canceled <- as.factor(train2$is_canceled)
train2$hotel <- as.factor(train2$hotel)
train2$meal <- as.factor(train2$meal)
train2$distribution_channel <- as.factor(train2$distribution_channel)
train2$market_segment <- as.factor(train2$market_segment)
train2$is_repeated_guest <- as.factor(train2$is_repeated_guest)
train2$deposit_type <- as.factor(train2$deposit_type)
train2$customer_type <- as.factor(train2$customer_type)
train2$received_room_pref <- as.factor(train2$received_room_pref)
```

```
### DO NOT RUN MORE THAN ONCE
n <- length(colnames(train2))
i <- 1
OOB_list <- list()
conmat_list <- list()
rf_list <- list()
for (nt in c(250, 500, 750)) {
  for (m in c(sqrt(n), sqrt(n+1), sqrt(n+2))) {
    rf <-randomForest(is_canceled~.,
                      data=train2,
                      ntree=nt,
                      mtry=m,
                      na.option = na.pass)
    OOB_est <- mean((rf$err.rate)[,])
    conmat <- rf$confusion
    rf_name <- paste0("rf_model_", as.character(i))
    OOB_name <- paste0("OOB_est_", as.character(i))
    conmat_name <- paste0("conmat_", as.character(i))
    saveRDS(conmat, conmat_name)
    saveRDS(rf, rf_name)
    saveRDS(OOB_est, OOB_name)
    rf_list[i] <- rf
    i <- i + 1
  }
}
```

```
OOB_1 <- readRDS("OOB_est_1")
OOB_2 <- readRDS("OOB_est_2")
OOB_3 <- readRDS("OOB_est_3")
OOB_4 <- readRDS("OOB_est_4")
OOB_5 <- readRDS("OOB_est_5")
OOB_7 <- readRDS("OOB_est_7")
OOB_6 <- readRDS("OOB_est_6")
OOB_8 <- readRDS("OOB_est_8")
OOB_9 <- readRDS("OOB_est_9")
rf_1 <- readRDS("rf_model_1")
rf_2 <- readRDS("rf_model_2")
rf_3 <- readRDS("rf_model_3")
rf_4 <- readRDS("rf_model_4")
rf_5 <- readRDS("rf_model_5")
rf_6 <- readRDS("rf_model_6")
rf_7 <- readRDS("rf_model_7")
rf_8 <- readRDS("rf_model_8")
rf_9 <- readRDS("rf_model_9")
conmat_1 <- readRDS("conmat_1")
conmat_2 <- readRDS("conmat_2")
conmat_3 <- readRDS("conmat_3")
conmat_4 <- readRDS("conmat_4")
conmat_5 <- readRDS("conmat_5")
conmat_6 <- readRDS("conmat_6")
conmat_7 <- readRDS("conmat_7")
conmat_8 <- readRDS("conmat_8")
conmat_9 <- readRDS("conmat_9")
```

```
get_stats(rf_1)
```

```
## mtry: 4
## ntree: 250
## Accuracy: 0.8176
## Balanced accuracy: 0.7836
## Specificity: 0.9164
## Sensitivity: 0.6508
## Precision: 0.822
##
## OOB: 0.2073
```

```
get_stats(rf_2)
```

```
## mtry: 5
## ntree: 250
## Accuracy: 0.8249
## Balanced accuracy: 0.7958
## Specificity: 0.9094
## Sensitivity: 0.6822
## Precision: 0.817
##
## OOB: 0.1972
```

```
get_stats(rf_3)
```

```
## mtry: 5
## ntree: 250
## Accuracy: 0.8259
## Balanced accuracy: 0.797
## Specificity: 0.9101
## Sensitivity: 0.6839
## Precision: 0.8184
##
## OOB: 0.1961
```

```
get_stats(rf_4)
```

```
## mtry: 4
## ntree: 500
## Accuracy: 0.8175
## Balanced accuracy: 0.7829
## Specificity: 0.9182
## Sensitivity: 0.6476
## Precision: 0.8243
##
## OOB: 0.2064
```

```
get_stats(rf_5)
```

```
## mtry: 5
## ntree: 500
## Accuracy: 0.8252
## Balanced accuracy: 0.7958
## Specificity: 0.9108
## Sensitivity: 0.6807
## Precision: 0.819
##
## OOB: 0.1953
```

```
get_stats(rf_6)
```

```
## mtry: 5
## ntree: 500
## Accuracy: 0.8249
## Balanced accuracy: 0.7963
## Specificity: 0.9082
## Sensitivity: 0.6844
## Precision: 0.8155
##
## OOB: 0.1956
```

```
get_stats(rf_7)
```

```
## mtry: 4
## ntree: 750
## Accuracy: 0.8179
## Balanced accuracy: 0.7839
## Specificity: 0.9167
## Sensitivity: 0.651
## Precision: 0.8225
##
## OOB: 0.2057
```

```
get_stats(rf_8)
```

```
## mtry: 5
## ntree: 750
## Accuracy: 0.8251
## Balanced accuracy: 0.7962
## Specificity: 0.9094
## Sensitivity: 0.683
## Precision: 0.8172
##
## OOB: 0.1953
```

```
get_stats(rf_9)
```

```
## mtry: 5
## ntree: 750
## Accuracy: 0.8254
## Balanced accuracy: 0.796
## Specificity: 0.911
## Sensitivity: 0.681
## Precision: 0.8194
##
## OOB: 0.1953
```

```
n <- length(colnames(train2))
n
```

```
## [1] 20
```

Part 2.2-i

```
## Balanced Accuracy, Specificity, and Sensitivity are maximized when mtry equals 5 (corresponding to root(n+2))
and ntree equals 250.
```

Part 2.2-ii

```
## OOB error is minimized in three places -- for combinations (mtry, nsplit) of (root(n+1), 500), (root(n+1), 75
0), and (root(n+2), 750). At each of these three places, the OOB error is 0.1953.
##
## (Note: n, the number of features in the model, is equal to 20. In these calculations, root(n+1) and root(n+2)
both evaluate to 5 -- possibly due to rounding.)
```

Part 2.2-iii

```
## The best model determined by (i) is a different model than the one determined by (ii). It makes sense that the
y would not be identical, because the OOB error is calculated using data outside of the training set.
```