CS422 - HW4

Jane Downer

9/28/2020

Part 2.1

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(magrittr)
library(knitr)
setwd("~/Desktop")
df_hotel <- read.csv("hotel_bookings.csv", sep=",", header=T)
```

Part 2.1-A

```
h1_count = sum(df_hotel$hotel == "Resort Hotel")
cat("Number of 'H1' hotels: ", h1_count, "\n")
```

```
## Number of 'H1' hotels:  40060
```

```
h2_count = sum(df_hotel$hotel == "City Hotel")
cat("Number of 'H2' hotels: ", h2_count, "\n")
```

```
## Number of 'H2' hotels:  79330
```

Part 2.1-B

```
cancel_count = sum(df_hotel$is_canceled == 1)
not_canceled_count = sum(df_hotel$is_canceled == 0)

cat("Number of guests who canceled reservation: ", cancel_count, "\n")
```

```
## Number of guests who canceled reservation:  44224
```

```
cat("Number of guests who did not cancel the resrvation: ", not_canceled_count, "\n")
```

```
## Number of guests who did not cancel the resrvation:  75166
```

Part 2.1-C

```
Contract_count = sum(df_hotel$customer_type == "Contract")
Group_count = sum(df_hotel$customer_type == "Group")
Transient_count = sum(df_hotel$customer_type == "Transient")
type_list_count = c(Contract_count, Group_count, Transient_count)
type_list = c("Contract", "Group", "Transient")
most_common = which.max(type_list_count)
cat(paste0("Customer type with the most reservations is ", type_list[most_common], ", with ", type_list_count[mos
t_common], " reservations."))
```

```
## Customer type with the most reservations is Transient, with 89613 reservations.
```

Part 2.1-D

```
max_index = which.max(df_hotel$required_car_parking_spaces)
max_parking = df_hotel$required_car_parking[max_index]
count_max_parking = sum(df_hotel$required_car_parking_spaces == max_parking)
cat(paste0(count_max_parking, " customers required the most number of parking spaces (", max_parking, ")."))
```

```
## 2 customers required the most number of parking spaces (8).
```

Part 2.1-E

```
min_index = which.min(df_hotel$required_car_parking_spaces)
min_parking = df_hotel$required_car_parking[min_index]
count_min_parking = sum(df_hotel$required_car_parking_spaces == min_parking)
cat(paste0(count_min_parking, " customers required the least number of parking spaces (", min_parking, ")."))
```

```
## 111974 customers required the least number of parking spaces (0).
```

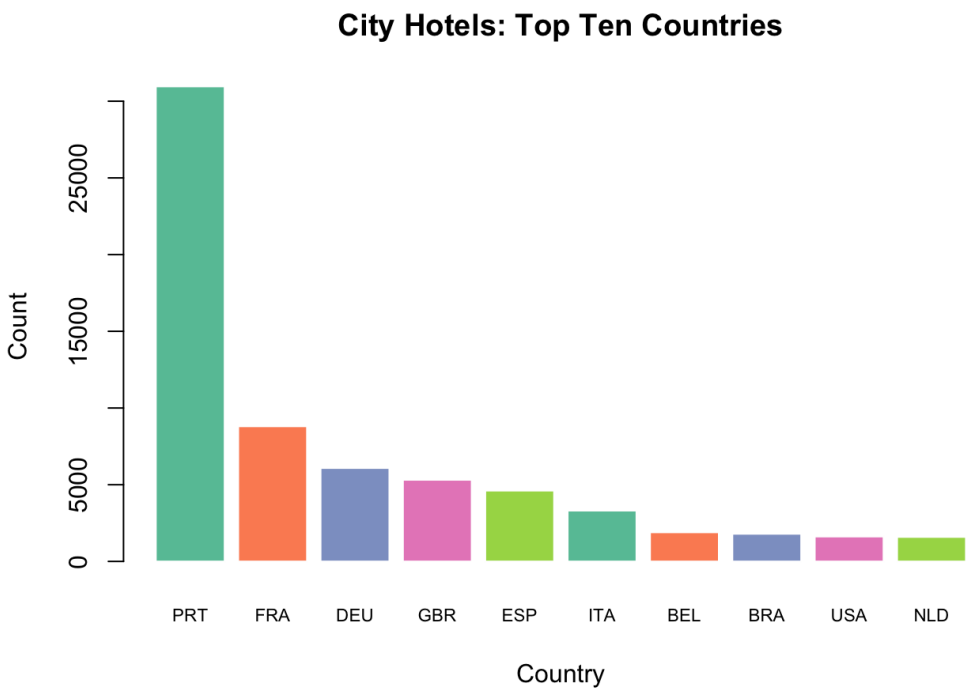
Part 2.1-F

```
#A B C D E F G H L P
#A B C D E F G H I K L P
#room_type = c("A","B","C","D","E","F","G","H","I","K","L","P")
#df_hotel$reserved_room_type
#df_hotel$reserved_room_type
equal_df <- df_hotel[as.character(df_hotel$reserved_room_type)==as.character(df_hotel$assigned_room_type), ]
percent_assigned = round(100*(nrow(equal_df)/nrow(df_hotel)), digits = 2)

cat(paste0(percent_assigned, "% of the people who expressed a room preference during reservation got the room dur
ing check-in."))
```

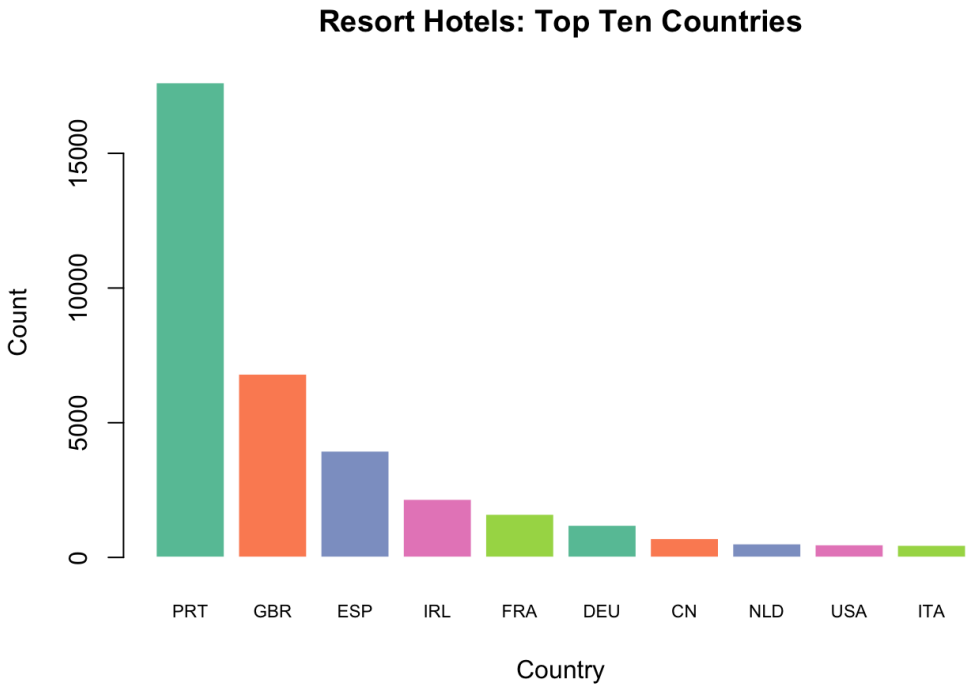
```
## 87.51% of the people who expressed a room preference during reservation got the room during check-in.
```

Part 2.1-G

```
library(dplyr)

# City Hotel data
city_df <- filter(df_hotel, hotel == "City Hotel")
country_cities <- sort(summary(as.factor(city_df$country)), decreasing = TRUE)[1:10]

# Resort Hotel data
resort_df <- filter(df_hotel, hotel == "Resort Hotel")
country_resorts <- sort(summary(as.factor(resort_df$country)), decreasing = TRUE)
resort_names = union(names(country_resorts)[1:9], names(country_resorts)[11])
resort_vals = union(as.vector(country_resorts)[1:9], as.vector(country_resorts)[11])

# Bar charts
library(RColorBrewer)
coul <- brewer.pal(5, "Set2")

barplot(as.vector(country_cities),
        main = "City Hotels: Top Ten Countries",
        xlab = "Country", ylab = "Count",
        names.arg = names(country_cities), cex.names=.7,
        col=coul, border="white")
```




```
barplot(resort_vals,
        main = "Resort Hotels: Top Ten Countries",
        xlab = "Country", ylab = "Count",
        names.arg = resort_names, cex.names=.7,
        col=coul, border="white")
```



Part 2.1-H-i

```
top_country = resort_names[1]
cat(paste0("The code for the country with the most visitors is ", top_country, "."))
```

```
## The code for the country with the most visitors is PRT.
```

Part 2.1-H-ii

```
cat("Based on this country code, and assuming the other frequent countries are nearby, I am guessing that this da
ta is from Portugal.")
```

```
## Based on this country code, and assuming the other frequent countries are nearby, I am guessing that this data
is from Portugal.
```

Part 2.2-A-i

```
# Modifying the dataset -- grouping variables into single columns
library(rpart)
library("data.table")
```

```
##
## Attaching package: 'data.table'
```

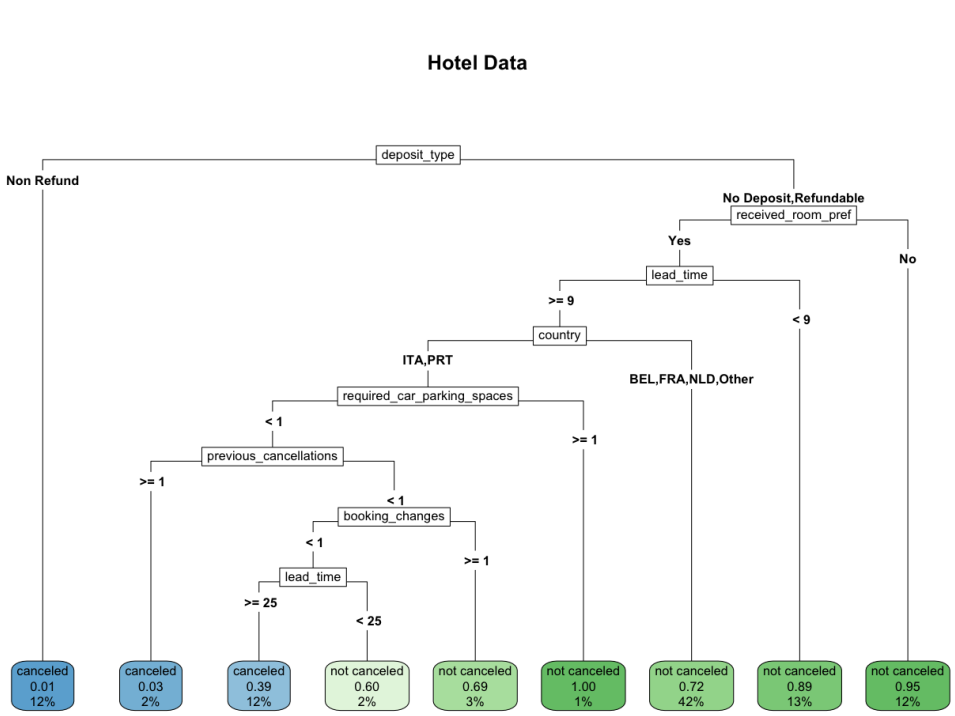
```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library("caret")
```

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

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

```
library(rpart.plot)
library(ROCR)
set.seed(1122)

df_hotel[, "arrival_date_month"] <- sapply(df_hotel[, "arrival_date_month"], as.character)
df_hotel$arrival_date_month[df_hotel$arrival_date_month %in% c("March","April","May")] <- "spring"
df_hotel$arrival_date_month[df_hotel$arrival_date_month %in% c("December","January","February")] <- "winter"
df_hotel$arrival_date_month[df_hotel$arrival_date_month %in% c("June","July","August")] <- "summer"
df_hotel$arrival_date_month[df_hotel$arrival_date_month %in% c("September","October","November")] <- "fall"
df_hotel <- df_hotel %>% rename(season = arrival_date_month)

df_hotel[, "country"] <- sapply(df_hotel[, "country"], as.character)
df_hotel$country[df_hotel$country %in% c("GBR","IRL","USA")] <- "English-speaking"
df_hotel$country[df_hotel$country %in% c("BRA","ESP")] <- "Spanish-speaking"
df_hotel$country[df_hotel$country %in% c("PRT")] <- "PRT"
df_hotel$country[df_hotel$country %in% c("Other Popular European")] <- "Other Popular European"
`%nin%` = Negate(`%in%`)
df_hotel$country[df_hotel$country %nin% c("BEL","FRA","ITA","NLD","PRT","BRA","ESP","GBR","IRL","USA")] <- "Othe
r"

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$reserved_room_type <- NULL
df_hotel$assigned_room_type <- NULL

df_hotel[, "is_canceled"] <- sapply(df_hotel[, "is_canceled"], as.character)
df_hotel$is_canceled[df_hotel$is_canceled %in% c(1)] <- "canceled"
df_hotel$is_canceled[df_hotel$is_canceled %in% c(0)] <- "not canceled"

df_hotel$nights_stayed <- rowSums(df_hotel[,c("stays_in_weekend_nights", "stays_in_week_nights")], na.rm=TRUE)
df_hotel$stays_in_week_nights <- NULL
df_hotel$stays_in_weekend_nights <- NULL


# Plotting the tree
index <- sample(1:nrow(df_hotel), size=0.1*nrow(df_hotel))
test <- df_hotel[index, ]
train <- df_hotel[-index, ]

mytree <- rpart(is_canceled ~ lead_time+
                  nights_stayed+
                  arrival_date_week_number+
                  is_repeated_guest+
                  country +
                  previous_cancellations+
                  booking_changes+
                  deposit_type+
                  received_room_pref+
                  required_car_parking_spaces,
                control=rpart.control(minsplit=5, maxdepth =8,cp =0.005),
                method = "class", parms = list(split = "gini"), data = df_hotel)


rpart.plot(mytree, main = "Hotel Data",
           type = 5, clip.right.labs=T, tweak = 1.2, extra = 106, fallen.leaves = T)
```



Part 2.2-A-i

```
agent_count = sum(df_hotel$customer_type == "Contract")
Group_count = sum(df_hotel$customer_type == "Group")
Transient_count = sum(df_hotel$customer_type == "Transient")
type_list_count = c(Contract_count, Group_count, Transient_count)
type_list = c("Contract", "Group", "Transient")
most_common = which.max(type_list_count)
cat(paste0("Customer type with the most reservations is ", type_list[most_common], ", with ", most_common))
```

```
## Customer type with the most reservations is Transient, with 3
```

Part 2.2-A-ii

```
var1 <- names(mytree$variable.importance)[1]
var2 <- names(mytree$variable.importance)[2]
var3 <- names(mytree$variable.importance)[3]
var4 <- names(mytree$variable.importance)[4]
var5 <- names(mytree$variable.importance)[5]

score1 <- round(as.vector(mytree$variable.importance)[1], digits = 2)
score2 <- round(as.vector(mytree$variable.importance)[2], digits = 2)
score3 <- round(as.vector(mytree$variable.importance)[3], digits = 2)
score4 <- round(as.vector(mytree$variable.importance)[4], digits = 2)
score5 <- round(as.vector(mytree$variable.importance)[5], digits = 2)


cat(paste0("\n\nThe five most important variables and their 'scores' (as defined by rpart's variable.imporance fu
nction):\n\n"))
```

```
##
##
## The five most important variables and their 'scores' (as defined by rpart's variable.imporance function):
```

```
cat(paste0("1. ", var1, ", ", score1, "\n"))
```

```
## 1. deposit_type, 12907.93
```

```
cat(paste0("2. ", var2, ", ", score2, "\n"))
```

```
## 2. lead_time, 2459.75
```

```
cat(paste0("3. ", var3, ", ", score3, "\n"))
```

```
## 3. country, 2223.43
```

```
cat(paste0("4. ", var4, ", ", score4, "\n"))
```

```
## 4. received_room_pref, 1904.31
```

```
cat(paste0("5. ", var5, ", ", score5, "\n"))
```

```
## 5. previous_cancellations, 1617.34
```

```
pred <- predict(mytree, test, type="class")
pred.prob <- predict(mytree, test, type="prob")

conmat<- confusionMatrix(pred, as.factor(test$is_canceled))
conmat
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     canceled not canceled
##   canceled         2570          557
##   not canceled     1954         6858
##
##                Accuracy : 0.7897
##                  95% CI : (0.7823, 0.797)
##     No Information Rate : 0.6211
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5245
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5681
##             Specificity : 0.9249
##          Pos Pred Value : 0.8219
##          Neg Pred Value : 0.7783
##              Prevalence : 0.3789
##          Detection Rate : 0.2153
##    Detection Prevalence : 0.2619
##       Balanced Accuracy : 0.7465
##
##        'Positive' Class : canceled
##
```

```
mytree$variable.importance
```

```
##                deposit_type                   lead_time
##                   12907.9271                   2459.7484
##                     country          received_room_pref
##                    2223.4332                   1904.3072
##      previous_cancellations required_car_parking_spaces
##                    1617.3407                    804.2539
##             booking_changes           is_repeated_guest
##                     391.1062                    179.7804
##               nights_stayed
##                     22.8397
```

Part 2.2-A-iii

```
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("\nAccuracy: ", acc))
```

```
##
## Accuracy: 0.7897
```

```
cat(paste0("\nError: ", 1-acc))
```

```
##
## Error: 0.2103
```

```
cat(paste0("\nBalanced Accuracy: ", bal_acc))
```

```
##
## Balanced Accuracy: 0.7465
```

```
cat(paste0("\nSpecificity: ", spec))
```

```
##
## Specificity: 0.9249
```

```
cat(paste0("\nSensitivity: ", sens))
```

```
##
## Sensitivity: 0.5681
```

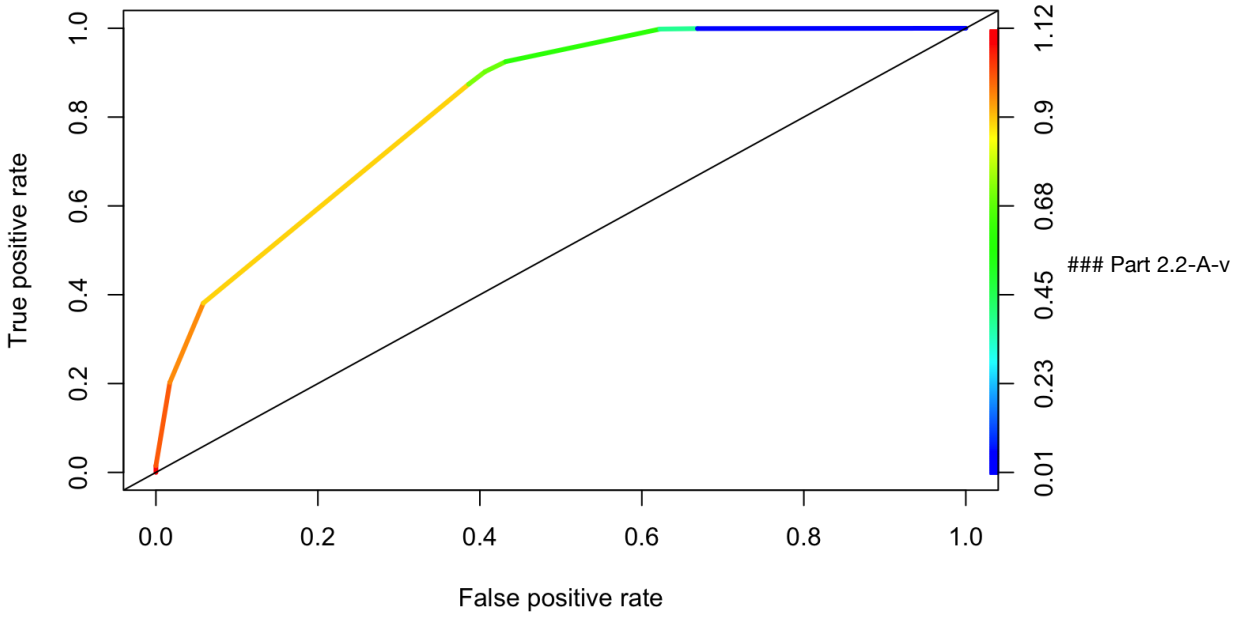
```
cat(paste0("\nPrecision: ", prec))
```

```
##
## Precision: 0.8219
```

```
cat(paste0("\n"))
```

Part 2.2-A-iv

```
# Calculate AUC and plot ROC
rocr <- predict(mytree, newdata=test, type="prob")[,2]
f.pred <- prediction(rocr, test$is_canceled)
plot(performance(f.pred, "tpr", "fpr"), colorize=T, lwd=3)
abline(0,1)
```



Part 2.2-A-v

```
#Print AUC
auc.pruned <- performance(f.pred, measure = "auc")
cat(paste0("The area under curve (AUC) for the full tree is ",
        round(auc.pruned@y.values[[1]], 3)))
```

```
## The area under curve (AUC) for the full tree is 0.821
```