Ensemble Techniques

Vladimir Sokolov

10/24/2022

This notebook explores Hotel Booking data from Kaggle.

Load the data set.

```
df <- read.csv("hotel_bookings.csv")
str(df)
```

```
## 'data.frame':    119390 obs. of  32 variables:
##  $ hotel                         : chr  "Resort Hotel" "Resort Hotel" "Resort Hotel" "Resort Hotel"
##  $ is_canceled                   : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ lead_time                     : int  342 737 7 13 14 14 0 9 85 75 ...
##  $ arrival_date_year             : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
##  $ arrival_date_month            : chr  "July" "July" "July" "July" ...
##  $ arrival_date_week_number      : int  27 27 27 27 27 27 27 27 27 27 ...
##  $ arrival_date_day_of_month     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ stays_in_weekend_nights       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ stays_in_week_nights          : int  0 0 1 1 2 2 2 2 3 3 ...
##  $ adults                        : int  2 2 1 1 2 2 2 2 2 2 ...
##  $ children                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ babies                        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ meal                          : chr  "BB" "BB" "BB" "BB" ...
##  $ country                       : chr  "PRT" "PRT" "GBR" "GBR" ...
##  $ market_segment                : chr  "Direct" "Direct" "Direct" "Corporate" ...
##  $ distribution_channel          : chr  "Direct" "Direct" "Direct" "Corporate" ...
##  $ is_repeated_guest             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ previous_cancellations        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ previous_bookings_not_canceled: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ reserved_room_type            : chr  "C" "C" "A" "A" ...
##  $ assigned_room_type            : chr  "C" "C" "C" "A" ...
##  $ booking_changes               : int  3 4 0 0 0 0 0 0 0 0 ...
##  $ deposit_type                  : chr  "No Deposit" "No Deposit" "No Deposit" "No Deposit" ...
##  $ agent                         : chr  "NULL" "NULL" "NULL" "304" ...
##  $ company                       : chr  "NULL" "NULL" "NULL" "NULL" ...
##  $ days_in_waiting_list          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ customer_type                 : chr  "Transient" "Transient" "Transient" "Transient" ...
##  $ adr                           : num  0 0 75 75 98 ...
##  $ required_car_parking_spaces   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_of_special_requests     : int  0 0 0 0 1 1 0 1 1 0 ...
##  $ reservation_status            : chr  "Check-Out" "Check-Out" "Check-Out" "Check-Out" ...
##  $ reservation_status_date       : chr  "2015-07-01" "2015-07-01" "2015-07-02" "2015-07-02" ...
```

Factor and simplify data. This time country is removed because tree does not like factors with over 32 levels.

```
df$hotel <- factor(df$hotel)
df$is_canceled <- factor(df$is_canceled)
df$country <- factor(df$country)
df$market_segment <- factor(df$market_segment)
df$deposit_type <- factor(df$deposit_type)
df$customer_type <- factor(df$customer_type)
df<-df[c(1,2,3,6:10,12,15,23,26,27)]
str(df)
```

```
## ’data.frame':    119390 obs. of  13 variables:
##  $ hotel                    : Factor w/ 2 levels "City Hotel","Resort Hotel": 2 2 2 2 2 2 2 2 2 2 ..
##  $ is_canceled              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 2 ...
##  $ lead_time                : int  342 737 7 13 14 14 0 9 85 75 ...
##  $ arrival_date_week_number : int  27 27 27 27 27 27 27 27 27 27 ...
##  $ arrival_date_day_of_month: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ stays_in_weekend_nights  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ stays_in_week_nights     : int  0 0 1 1 2 2 2 2 3 3 ...
##  $ adults                   : int  2 2 1 1 2 2 2 2 2 2 ...
##  $ babies                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ market_segment           : Factor w/ 8 levels "Aviation","Complementary",..: 4 4 4 3 7 7 4 4 7 6
##  $ deposit_type             : Factor w/ 3 levels "No Deposit","Non Refund",..: 1 1 1 1 1 1 1 1 1 1 .
##  $ days_in_waiting_list     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ customer_type            : Factor w/ 4 levels "Contract","Group",..: 3 3 3 3 3 3 3 3 3 3 ...
```

Check for null values

```
sapply(df, function(x) sum(is.na(x)))
```

```
##                     hotel               is_canceled                 lead_time
##                         0                         0                         0
##  arrival_date_week_number arrival_date_day_of_month   stays_in_weekend_nights
##                         0                         0                         0
##      stays_in_week_nights                    adults                    babies
##                         0                         0                         0
##            market_segment              deposit_type      days_in_waiting_list
##                         0                         0                         0
##             customer_type
##                         0
```

Shrink so adaboost can run on my computer. Divide into train and test.

```
set.seed(12345)
i <- sample(1:nrow(df), nrow(df)*.3, replace=FALSE)
shrink <- df[i,]
j <- sample(1:nrow(shrink), nrow(shrink)*.8, replace=FALSE)
train <- df[j,]
test <- df[-j,]
```

Decision tree to set baseline using tree()

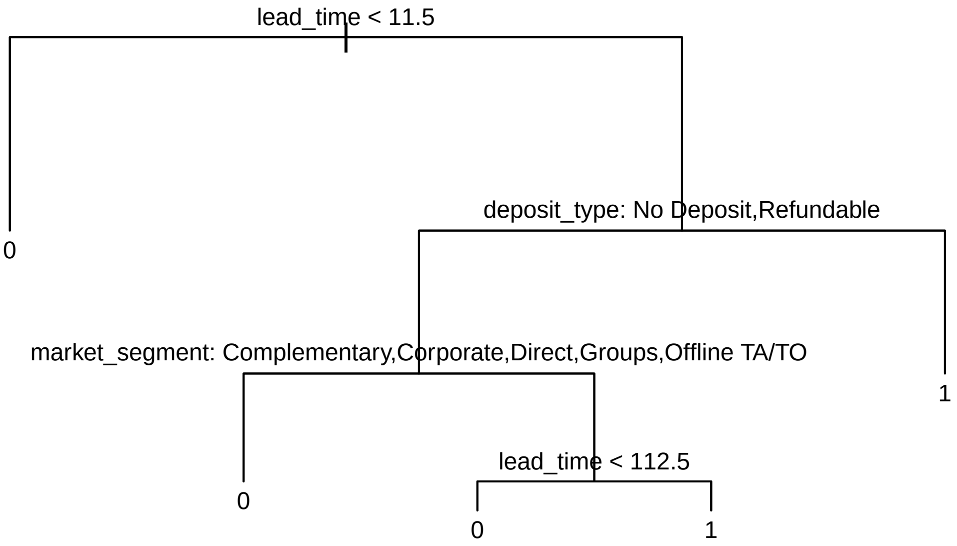
```
library(tree)
tree1 <- tree(is_canceled~., data=train)
pred1 <- predict(tree1, newdata=test, type="class")
table(pred1, test$is_canceled)
```

```
##
## pred1     0     1
##     0 48137 16321
##     1  7302 18977
```

```
mean(pred1==test$is_canceled)
```

```
## [1] 0.7396542
```

```
plot(tree1)
text(tree1, cex=.75, pretty=0)
```


lead_time < 11.5
deposit_type: No Deposit,Refundable
0
market_segment: Complementary,Corporate,Direct,Groups,Offline TA/TO
1
lead_time < 112.5
0
0
1

Try Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(12345)
rf <- randomForest(is_canceled~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = is_canceled ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.53%
## Confusion matrix:
##       0    1 class.error
## 0 18028 1699  0.08612561
## 1  3610 5316  0.40443648
```

```
pred2 <- predict(rf, newdata=test, type="response")
mean(pred2==test$is_canceled)
```

```
## [1] 0.7326559
```

Try XGBoost

```
library(xgboost)
train_label <-ifelse(train$is_canceled==1, 1,0)
train_matrix <- data.matrix(train[,-2])
model <-xgboost(data=train_matrix, label=train_label, nround=100, objective='binary:logistic')
```

```
## [1]  train-logloss:0.593535
## [2]  train-logloss:0.538704
## [3]  train-logloss:0.506464
## [4]  train-logloss:0.486628
## [5]  train-logloss:0.472722
## [6]  train-logloss:0.463024
## [7]  train-logloss:0.456580
## [8]  train-logloss:0.450538
## [9]  train-logloss:0.446915
## [10] train-logloss:0.444315
## [11] train-logloss:0.441858
## [12] train-logloss:0.439443
## [13] train-logloss:0.435750
## [14] train-logloss:0.434531
## [15] train-logloss:0.433079
## [16] train-logloss:0.428435
## [17] train-logloss:0.425180
## [18] train-logloss:0.424087
## [19] train-logloss:0.422670
## [20] train-logloss:0.420975
## [21] train-logloss:0.420129
## [22] train-logloss:0.417513
## [23] train-logloss:0.416674
## [24] train-logloss:0.416121
## [25] train-logloss:0.415121
## [26] train-logloss:0.413410
## [27] train-logloss:0.413109
## [28] train-logloss:0.409838
## [29] train-logloss:0.407423
## [30] train-logloss:0.405939
## [31] train-logloss:0.404550
## [32] train-logloss:0.401483
## [33] train-logloss:0.400493
## [34] train-logloss:0.399315
## [35] train-logloss:0.399169
## [36] train-logloss:0.397255
## [37] train-logloss:0.396929
## [38] train-logloss:0.395730
## [39] train-logloss:0.395015
## [40] train-logloss:0.394493
## [41] train-logloss:0.392528
## [42] train-logloss:0.391166
## [43] train-logloss:0.390710
## [44] train-logloss:0.390021
## [45] train-logloss:0.389244
## [46] train-logloss:0.387921
## [47] train-logloss:0.387238
## [48] train-logloss:0.385895
## [49] train-logloss:0.384486
## [50] train-logloss:0.380859
## [51] train-logloss:0.379778
## [52] train-logloss:0.379215
## [53] train-logloss:0.377505
## [54] train-logloss:0.375510
## [55] train-logloss:0.374762
## [56] train-logloss:0.373070
## [57] train-logloss:0.371796
## [58] train-logloss:0.369736
## [59] train-logloss:0.368999
## [60] train-logloss:0.367905
## [61] train-logloss:0.366642
## [62] train-logloss:0.364699
## [63] train-logloss:0.364094
## [64] train-logloss:0.361972
## [65] train-logloss:0.361296
## [66] train-logloss:0.360432
## [67] train-logloss:0.360315
## [68] train-logloss:0.359578
## [69] train-logloss:0.358471
## [70] train-logloss:0.356791
## [71] train-logloss:0.355384
## [72] train-logloss:0.354758
## [73] train-logloss:0.353909
## [74] train-logloss:0.353676
## [75] train-logloss:0.352233
## [76] train-logloss:0.350685
## [77] train-logloss:0.350532
## [78] train-logloss:0.349310
## [79] train-logloss:0.348177
## [80] train-logloss:0.346709
## [81] train-logloss:0.346053
## [82] train-logloss:0.345944
## [83] train-logloss:0.344905
## [84] train-logloss:0.344200
## [85] train-logloss:0.342164
## [86] train-logloss:0.341230
## [87] train-logloss:0.341105
## [88] train-logloss:0.340334
## [89] train-logloss:0.339342
## [90] train-logloss:0.338826
## [91] train-logloss:0.338265
## [92] train-logloss:0.336823
## [93] train-logloss:0.336690
## [94] train-logloss:0.335975
## [95] train-logloss:0.335158
## [96] train-logloss:0.334738
## [97] train-logloss:0.334368
## [98] train-logloss:0.333725
## [99] train-logloss:0.333546
## [100]    train-logloss:0.333054
```

```
test_label <- ifelse(test$is_canceled==1, 1, 0)
test_matrix <- data.matrix(test[,-2])
probs <- predict(model, test_matrix)
pred3 <- ifelse(probs>0.5, 1,0)
mean(pred3==test_label)
```

```
## [1] 0.7037151
```

Try Adaboost

```
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

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

```
##
## Attaching package: ’ggplot2’
```

```
## The following object is masked from ’package:randomForest’:
##
##     margin
```

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

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel
```

```
adab1 <-boosting(is_canceled~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
summary(adab1)
```

```
##            Length Class   Mode
## formula        3  formula call
## trees         20  -none-  list
## weights       20  -none-  numeric
## votes      57306  -none-  numeric
## prob       57306  -none-  numeric
## class      28653  -none-  character
## importance    12  -none-  numeric
## terms          3  terms   call
## call           6  -none-  call
```

```
pred4 <- predict(adab1, newdata=test, type="response")
mean(pred4$class==test$is_canceled)
```

```
## [1] 0.7339454
```

Analysis

```
mean(pred1==test$is_canceled)
```

```
## [1] 0.7396542
```

```
mean(pred2==test$is_canceled)
```

```
## [1] 0.7326559
```

```
mean(pred3==test_label)
```

```
## [1] 0.7037151
```

```
mean(pred4$class==test$is_canceled)
```

```
## [1] 0.7339454
```

The accuracy of DT, random forest, and adaboost were very similar while xgboost had a lower accuracy than the rest. I am not completely sure I modified the data correctly to give xgboost the best chance. The run time of DT and xgboost were very fast, less than 5 seconds, while the run time of random forest and adaboost was 30 or more seconds. Fastadaboost was archived and I couldn't get the download to work so I used adaboost. Adaboost was very slow and I shrank the data to make sure it didn't freeze my computer. Shrinking the data had a significant effect on the accuracy which was unfortunate.