Ensemble Techniques

David Park

2022-10-25

```
library(e1071)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(mltools)
```

```
##
## Attaching package: 'mltools'
##
## The following object is masked from 'package:tidyr':
##
##     replace_na
##
## The following object is masked from 'package:e1071':
##
##     skewness
```

```
set.seed(1234)
```

Extract 10k observations from data set with certain features

```
df <- read.csv("C:\\School\\CS 4375 Machine Learning\\Kernel Ensemble Methods\\Invistico_Airline.csv")
df <- sample_n(df, 10000, replace=FALSE)
df$satisfaction <- as.factor(df$satisfaction)
df <- df[, c(1,4,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21)]
str(df)
```

```
## 'data.frame':    10000 obs. of  17 variables:
##  $ satisfaction                     : Factor w/ 2 levels "dissatisfied",..: 2 2 1 2 1 2 2 2 1 2 ...
##  $ Age                              : int  32 43 66 69 10 36 42 46 41 26 ...
```

```
##  $ Flight.Distance                  : int  4906 334 1724 433 1566 1227 741 2389 1423 3195 ...
##  $ Seat.comfort                     : int  1 1 1 4 4 4 3 5 3 3 ...
##  $ Departure.Arrival.time.convenient: int  1 1 1 1 3 4 3 5 2 1 ...
##  $ Food.and.drink                   : int  1 1 1 1 4 4 3 5 2 3 ...
##  $ Gate.location                    : int  1 1 3 1 3 4 3 5 2 3 ...
##  $ Inflight.wifi.service            : int  4 2 5 5 3 4 2 2 3 3 ...
##  $ Inflight.entertainment           : int  4 4 1 5 4 4 5 4 3 3 ...
##  $ Online.support                   : int  4 5 4 3 2 4 5 5 3 3 ...
##  $ Ease.of.Online.booking           : int  4 4 5 4 3 4 5 5 3 3 ...
##  $ On.board.service                 : int  3 4 4 4 5 3 5 5 3 3 ...
##  $ Leg.room.service                 : int  2 4 2 4 1 2 5 5 2 5 ...
##  $ Baggage.handling                 : int  5 4 3 4 2 1 5 5 4 4 ...
##  $ Checkin.service                  : int  4 5 1 5 1 1 3 5 3 5 ...
##  $ Cleanliness                      : int  4 4 3 4 4 2 5 5 3 4 ...
##  $ Online.boarding                  : int  4 4 5 5 3 4 4 3 3 3 ...
```

```
head(df)
```

```
##   satisfaction Age Flight.Distance Seat.comfort
## 1    satisfied  32            4906            1
## 2    satisfied  43             334            1
## 3 dissatisfied  66            1724            1
## 4    satisfied  69             433            4
## 5 dissatisfied  10            1566            4
## 6    satisfied  36            1227            4
##   Departure.Arrival.time.convenient Food.and.drink Gate.location
## 1                                 1              1             1
## 2                                 1              1             1
## 3                                 1              1             3
## 4                                 1              1             1
## 5                                 3              4             3
## 6                                 4              4             4
##   Inflight.wifi.service Inflight.entertainment Online.support
## 1                     4                     4              4
## 2                     2                     4              5
## 3                     5                     1              4
## 4                     5                     5              3
## 5                     3                     4              2
## 6                     4                     4              4
##   Ease.of.Online.booking On.board.service Leg.room.service Baggage.handling
## 1                      4                3                2                 5
## 2                      4                4                4                 4
## 3                      5                4                2                 3
## 4                      4                4                4                 4
## 5                      3                5                1                 2
## 6                      4                3                2                 1
##   Checkin.service Cleanliness Online.boarding
## 1               4           4               4
## 2               5           4               4
## 3               1           3               5
## 4               5           4               5
## 5               1           4               3
## 6               1           2               4
```

Divide train/test

```
i <- sample(nrow(df), .75*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Decision tree for baseline

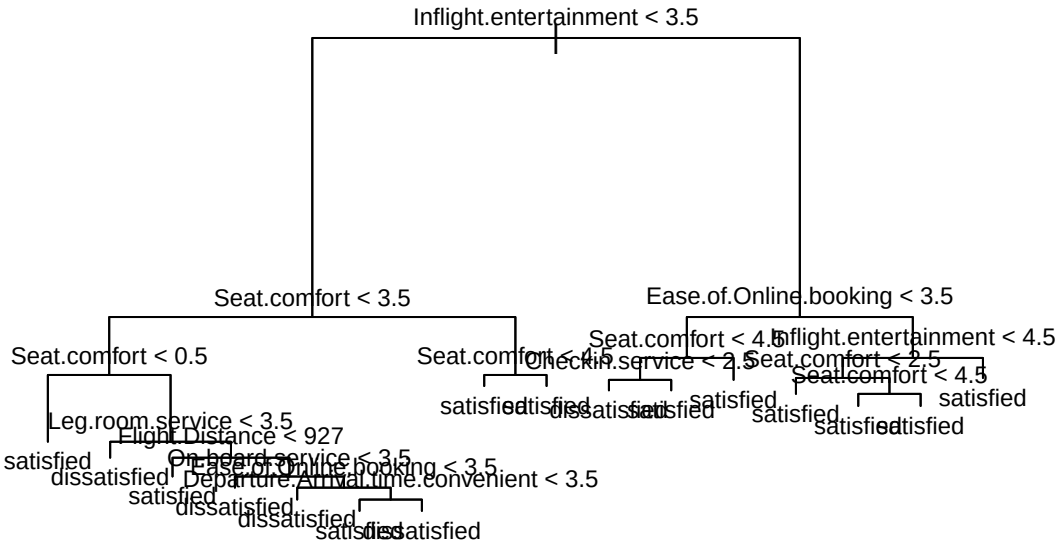
```
library(tree)
start_time_dt <- Sys.time()
tree <- tree(satisfaction~., data=train)
end_time_dt <- Sys.time()
tree
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 7500 10320.00 satisfied ( 0.45053 0.54947 )
##     2) Inflight.entertainment < 3.5 3339  3517.00 dissatisfied ( 0.78017 0.21983 )
##       4) Seat.comfort < 3.5 2922  2445.00 dissatisfied ( 0.85250 0.14750 )
##         8) Seat.comfort < 0.5 160     0.00 satisfied ( 0.00000 1.00000 ) *
##         9) Seat.comfort > 0.5 2762  1773.00 dissatisfied ( 0.90188 0.09812 )
##          18) Leg.room.service < 3.5 1592   504.60 dissatisfied ( 0.96294 0.03706 ) *
##          19) Leg.room.service > 3.5 1170  1107.00 dissatisfied ( 0.81880 0.18120 )
##            38) Flight.Distance < 927 52     0.00 satisfied ( 0.00000 1.00000 ) *
##            39) Flight.Distance > 927 1118   918.00 dissatisfied ( 0.85689 0.14311 )
##              78) On.board.service < 3.5 593   207.20 dissatisfied ( 0.95784 0.04216 ) *
##              79) On.board.service > 3.5 525   598.50 dissatisfied ( 0.74286 0.25714 )
##               158) Ease.of.Online.booking < 3.5 261   114.80 dissatisfied ( 0.94253 0.05747 ) *
##               159) Ease.of.Online.booking > 3.5 264   363.80 dissatisfied ( 0.54545 0.45455 )
##                 318) Departure.Arrival.time.convenient < 3.5 190   250.10 satisfied ( 0.36842 0.6315
##                 319) Departure.Arrival.time.convenient > 3.5 74     0.00 dissatisfied ( 1.00000 0.00
##       5) Seat.comfort > 3.5 417   489.20 satisfied ( 0.27338 0.72662 )
##        10) Seat.comfort < 4.5 261   356.60 satisfied ( 0.42912 0.57088 ) *
##        11) Seat.comfort > 4.5 156    21.40 satisfied ( 0.01282 0.98718 ) *
##     3) Inflight.entertainment > 3.5 4161  3998.00 satisfied ( 0.18601 0.81399 )
##       6) Ease.of.Online.booking < 3.5 1120  1506.00 satisfied ( 0.39821 0.60179 )
##        12) Seat.comfort < 4.5 928  1285.00 satisfied ( 0.48060 0.51940 )
##          24) Checkin.service < 2.5 179   171.10 dissatisfied ( 0.81564 0.18436 ) *
##          25) Checkin.service > 2.5 749  1008.00 satisfied ( 0.40053 0.59947 ) *
##        13) Seat.comfort > 4.5 192     0.00 satisfied ( 0.00000 1.00000 ) *
##       7) Ease.of.Online.booking > 3.5 3041  2080.00 satisfied ( 0.10786 0.89214 )
##        14) Inflight.entertainment < 4.5 1687  1561.00 satisfied ( 0.17427 0.82573 )
##          28) Seat.comfort < 2.5 485    81.54 satisfied ( 0.01649 0.98351 ) *
##          29) Seat.comfort > 2.5 1202  1319.00 satisfied ( 0.23794 0.76206 )
##            58) Seat.comfort < 4.5 933  1140.00 satisfied ( 0.30011 0.69989 ) *
##            59) Seat.comfort > 4.5 269    57.50 satisfied ( 0.02230 0.97770 ) *
##        15) Inflight.entertainment > 4.5 1354   317.70 satisfied ( 0.02511 0.97489 ) *
```

```
summary(tree)
```

```
##
## Classification tree:
## tree(formula = satisfaction ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Inflight.entertainment"            "Seat.comfort"
## [3] "Leg.room.service"                  "Flight.Distance"
## [5] "On.board.service"                  "Ease.of.Online.booking"
## [7] "Departure.Arrival.time.convenient" "Checkin.service"
## Number of terminal nodes:  16
## Residual mean deviance:  0.5654 = 4231 / 7484
## Misclassification error rate: 0.1259 = 944 / 7500
```

```
plot(tree)
text(tree, cex=0.75, pretty = 0)
```




```
end_time_dt - start_time_dt
```

```
## Time difference of 0.05322599 secs
```

Decision tree evaluation

```
pred <- predict(tree, newdata=test, type="class")
table(pred, test$satisfaction)
```

```
##
## pred           dissatisfied satisfied
##   dissatisfied          897        44
##   satisfied             260      1299
```

```
mean(pred==test$satisfaction)
```

```
## [1] 0.8784
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
start_time_rf <- Sys.time()
rf <- randomForest(satisfaction~., data=train, importance=TRUE)
end_time_rf <- Sys.time()
rf
```

```
##
## Call:
##  randomForest(formula = satisfaction ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 7.97%
## Confusion matrix:
##              dissatisfied satisfied class.error
## dissatisfied         3064       315  0.09322285
## satisfied             283      3838  0.06867265
```

```
pred <- predict(rf, newdata=test, type="response")
acc_rf <- mean(pred==test$satisfaction)
acc_rf
```

```
## [1] 0.93
```

```
end_time_rf - start_time_rf
```

```
## Time difference of 9.740083 secs
```

xgboost

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
train_label <- ifelse(train$satisfaction=='satisfied', 1, 0)
train_matrix <- data.matrix(train[,-1])
start_time_xgboost <- Sys.time()
model <- xgboost(data=train_matrix, label=train_label, nrounds=100, eval_metric='error', objective='bina
```

```
## [1]  train-error:0.111333
## [2]  train-error:0.107600
## [3]  train-error:0.104800
## [4]  train-error:0.095333
## [5]  train-error:0.094933
## [6]  train-error:0.085067
## [7]  train-error:0.079600
## [8]  train-error:0.075733
## [9]  train-error:0.075467
## [10] train-error:0.072400
## [11] train-error:0.068533
## [12] train-error:0.068400
## [13] train-error:0.066933
## [14] train-error:0.066400
## [15] train-error:0.062533
## [16] train-error:0.061867
## [17] train-error:0.058800
## [18] train-error:0.059333
## [19] train-error:0.055733
## [20] train-error:0.055333
## [21] train-error:0.054133
## [22] train-error:0.053200
## [23] train-error:0.049200
```

```
## [24] train-error:0.049067
## [25] train-error:0.047467
## [26] train-error:0.046267
## [27] train-error:0.043600
## [28] train-error:0.042667
## [29] train-error:0.042933
## [30] train-error:0.041067
## [31] train-error:0.038933
## [32] train-error:0.036667
## [33] train-error:0.035467
## [34] train-error:0.034667
## [35] train-error:0.034667
## [36] train-error:0.033467
## [37] train-error:0.032800
## [38] train-error:0.033067
## [39] train-error:0.032267
## [40] train-error:0.031333
## [41] train-error:0.029333
## [42] train-error:0.028800
## [43] train-error:0.026667
## [44] train-error:0.024667
## [45] train-error:0.023600
## [46] train-error:0.024400
## [47] train-error:0.023600
## [48] train-error:0.022933
## [49] train-error:0.020133
## [50] train-error:0.019867
## [51] train-error:0.019200
## [52] train-error:0.018267
## [53] train-error:0.018000
## [54] train-error:0.016800
## [55] train-error:0.016133
## [56] train-error:0.016133
## [57] train-error:0.015467
## [58] train-error:0.015467
## [59] train-error:0.014933
## [60] train-error:0.014800
## [61] train-error:0.014400
## [62] train-error:0.014000
## [63] train-error:0.014000
## [64] train-error:0.013733
## [65] train-error:0.013467
## [66] train-error:0.012933
## [67] train-error:0.012400
## [68] train-error:0.012000
## [69] train-error:0.011200
## [70] train-error:0.010133
## [71] train-error:0.009733
## [72] train-error:0.008267
## [73] train-error:0.008267
## [74] train-error:0.007867
## [75] train-error:0.007067
## [76] train-error:0.006667
## [77] train-error:0.006400
```

```
## [78] train-error:0.006267
## [79] train-error:0.006267
## [80] train-error:0.006000
## [81] train-error:0.005733
## [82] train-error:0.005333
## [83] train-error:0.004667
## [84] train-error:0.004667
## [85] train-error:0.004667
## [86] train-error:0.004133
## [87] train-error:0.003867
## [88] train-error:0.003600
## [89] train-error:0.003600
## [90] train-error:0.003467
## [91] train-error:0.003200
## [92] train-error:0.003200
## [93] train-error:0.003067
## [94] train-error:0.003067
## [95] train-error:0.002933
## [96] train-error:0.002800
## [97] train-error:0.002800
## [98] train-error:0.002267
## [99] train-error:0.002133
## [100]    train-error:0.002267
```

```
end_time_xgboost <- Sys.time()
end_time_xgboost - start_time_xgboost
```

```
## Time difference of 0.438355 secs
```

xgboost evaluation

```
test_label <- ifelse(train$satisfaction=='satisfied', 1, 0)
test_matrix <- data.matrix(train[,-1])
probs <- predict(model, test_matrix)
pred <- ifelse(probs>0.5, 1, 0)

acc_xg <- mean(pred==test_label)
mcc_xg <- mcc(pred, test_label)
acc_xg
```

```
## [1] 0.9977333
```

```
mcc_xg
```

```
## [1] 0.9954234
```

adaboost

```
library(adabag)
```

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

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

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

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

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
start_time_adaboost <- Sys.time()
adab <- boosting(satisfaction~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
end_time_adaboost <- Sys.time()
summary(adab)
```

```
##            Length Class   Mode
## formula         3 formula call
## trees          20 -none-  list
## weights        20 -none-  numeric
## votes       15000 -none-  numeric
## prob        15000 -none-  numeric
## class        7500 -none-  character
## importance     16 -none-  numeric
## terms           3 terms   call
## call            6 -none-  call
```

```
end_time_adaboost - start_time_adaboost
```

```
## Time difference of 5.567061 secs
```

adaboost evaluation

```
pred <- predict(adab, newdata=test, type="response")
acc_adabag <- mean(pred$class==test$satisfaction)
mcc_adabag <- mcc(factor(pred$class), test$satisfaction)
acc_adabag
```

```
## [1] 0.9176
```

```
mcc_adabag
```

```
## [1] 0.8346755
```

Comparing Results in Terms of Metrics and Run Times

Decision tree ran notably faster compared to the other 3 techniques used with a run time of only 0.068 seconds. This is most likely due to a decision tree not being an ensemble method. Comparing this run time to the random forest which ran for 9.3 seconds makes perfect sense because a random forest is created from multiple decision trees. Xgboost runs much faster than adaboost possibly because adaboost spends more time learning extreme cases, which can lead to increased times in noisy data. Since my data has multiple targets that possibly overlap it makes sense that xgboost would perform better than adaboost.