Ensemble Techniques

Jeffrey Li, Valari Graham

10/23/2022

Source for the data: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+# The provided data sets above were combined into one data set.

Clean the data

Remove date column, turn "Occupancy" into a factor.

```
data <- read.csv("data.csv")
data <- data[,c(2:7)]
data$Occupancy <- factor(data$Occupancy)
```

Divide into train, test

```
set.seed(1234)
i <- sample(1:nrow(data), nrow(data)*0.75, replace=FALSE)
train <- data[i,]
test <- data[-i,]
```

Decision tree baseline

```
library(tree)
library(mccr)
tree <- tree(Occupancy~., data=train)

startTime <- Sys.time()
pred <- predict(tree, newdata=test, type="class")
endTime <- Sys.time()
print(paste("Runtime: ", endTime - startTime, "seconds"))
```

```
## [1] "Runtime:  0.0796349048614502 seconds"
```

```
acc <- mean(pred==test$Occupancy)
mcc <- mccr(pred, test$Occupancy)

print(paste("Accuracy: ", acc))
```

```
## [1] "Accuracy:  0.993236714975845"
```

```
print(paste("MCC: ", mcc))
```

```
## [1] "MCC:  0.981570753114818"
```

Random forest

```
library(randomForest)
```

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

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

```
set.seed(1234)
startTime <- Sys.time()
rf <- randomForest(Occupancy~., data=train, importance=TRUE)
endTime <- Sys.time()
rf
```

```
##
## Call:
##  randomForest(formula = Occupancy ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.81%
## Confusion matrix:
##      0    1 class.error
## 0 6977   48  0.00683274
## 1   27 2260  0.01180586
```

```
print(paste("Runtime: ", endTime - startTime, "seconds"))
```

```
## [1] "Runtime:  2.93621015548706 seconds"
```

```
pred <- predict(rf, newdata=test, type="response")
acc_rf <- mean(pred==test$Occupancy)
mcc_rf <- mccr(factor(pred), test$Occupancy)

print(paste("Accuracy: ", acc_rf))
```

```
## [1] "Accuracy:  0.994524959742351"
```

```
print(paste("MCC: ", mcc_rf))
```

```
## [1] "MCC:  0.984930033495458"
```

XGBoost

```
library(xgboost)
train_label <- ifelse(train$Occupancy==1, 1, 0)
train_matrix <- data.matrix(train[, -6])
startTime <- Sys.time()
xg <- xgboost(data=train_matrix, label=train_label,
                nrounds=100, objective='binary:logistic')
```

```
## [1]  train-logloss:0.444618
## [2]  train-logloss:0.307313
## [3]  train-logloss:0.220974
## [4]  train-logloss:0.163619
## [5]  train-logloss:0.123752
## [6]  train-logloss:0.095692
## [7]  train-logloss:0.075620
## [8]  train-logloss:0.061236
## [9]  train-logloss:0.050369
## [10] train-logloss:0.042114
## [11] train-logloss:0.036131
## [12] train-logloss:0.031769
## [13] train-logloss:0.028479
## [14] train-logloss:0.025758
## [15] train-logloss:0.023949
## [16] train-logloss:0.022474
## [17] train-logloss:0.021034
## [18] train-logloss:0.019942
## [19] train-logloss:0.019196
## [20] train-logloss:0.018602
## [21] train-logloss:0.018030
## [22] train-logloss:0.017337
## [23] train-logloss:0.016988
## [24] train-logloss:0.016706
## [25] train-logloss:0.016480
## [26] train-logloss:0.016075
## [27] train-logloss:0.015714
## [28] train-logloss:0.014839
## [29] train-logloss:0.014334
## [30] train-logloss:0.013883
## [31] train-logloss:0.012987
## [32] train-logloss:0.012650
## [33] train-logloss:0.012350
## [34] train-logloss:0.012051
## [35] train-logloss:0.011898
## [36] train-logloss:0.011729
## [37] train-logloss:0.011389
## [38] train-logloss:0.011055
## [39] train-logloss:0.010734
## [40] train-logloss:0.010520
## [41] train-logloss:0.010051
## [42] train-logloss:0.009873
## [43] train-logloss:0.009472
## [44] train-logloss:0.009264
```

```
## [45] train-logloss:0.009064
## [46] train-logloss:0.008938
## [47] train-logloss:0.008768
## [48] train-logloss:0.008557
## [49] train-logloss:0.008110
## [50] train-logloss:0.007903
## [51] train-logloss:0.007713
## [52] train-logloss:0.007525
## [53] train-logloss:0.007299
## [54] train-logloss:0.007232
## [55] train-logloss:0.007166
## [56] train-logloss:0.007022
## [57] train-logloss:0.006939
## [58] train-logloss:0.006587
## [59] train-logloss:0.006400
## [60] train-logloss:0.006290
## [61] train-logloss:0.006161
## [62] train-logloss:0.005939
## [63] train-logloss:0.005868
## [64] train-logloss:0.005812
## [65] train-logloss:0.005729
## [66] train-logloss:0.005677
## [67] train-logloss:0.005518
## [68] train-logloss:0.005441
## [69] train-logloss:0.005384
## [70] train-logloss:0.005324
## [71] train-logloss:0.005245
## [72] train-logloss:0.005112
## [73] train-logloss:0.005042
## [74] train-logloss:0.004986
## [75] train-logloss:0.004957
## [76] train-logloss:0.004908
## [77] train-logloss:0.004827
## [78] train-logloss:0.004705
## [79] train-logloss:0.004589
## [80] train-logloss:0.004508
## [81] train-logloss:0.004401
## [82] train-logloss:0.004368
## [83] train-logloss:0.004332
## [84] train-logloss:0.004207
## [85] train-logloss:0.004167
## [86] train-logloss:0.004094
## [87] train-logloss:0.004038
## [88] train-logloss:0.004007
## [89] train-logloss:0.003980
## [90] train-logloss:0.003955
## [91] train-logloss:0.003929
## [92] train-logloss:0.003891
## [93] train-logloss:0.003852
## [94] train-logloss:0.003824
## [95] train-logloss:0.003771
## [96] train-logloss:0.003737
## [97] train-logloss:0.003647
## [98] train-logloss:0.003579
```

```
## [99] train-logloss:0.003557
## [100]    train-logloss:0.003525
```

```
endTime <- Sys.time()
```

```
test_label <- ifelse(test$Occupancy==1, 1, 0)
test_matrix <- data.matrix(test[, -6])

probs <- predict(xg, test_matrix)
pred <- ifelse(probs>0.5, 1, 0)

acc_xg <- mean(pred==test_label)
mcc_xg <- mccr(pred, test_label)

print(paste("Runtime: ", endTime - startTime, "seconds"))
```

```
## [1] "Runtime:  0.765635967254639 seconds"
```

```
print(paste("Accuracy: ", acc_xg))
```

```
## [1] "Accuracy:  0.994202898550725"
```

```
print(paste("MCC: ", mcc_xg))
```

```
## [1] "MCC:  0.984027615093953"
```

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
```

```
startTime <- Sys.time()
ada <- boosting(Occupancy~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
endTime <- Sys.time()
summary(ada)
```

```
##            Length Class   Mode
## formula         3  formula call
## trees          20  -none-  list
## weights        20  -none-  numeric
## votes       18624  -none-  numeric
## prob        18624  -none-  numeric
## class        9312  -none-  character
## importance      5  -none-  numeric
## terms           3  terms   call
## call            6  -none-  call
```

```
pred <- predict(ada, newdata=test, type="response")
acc_ada <- mean(pred$class==test$Occupancy)
mcc_ada <- mccr(factor(pred$class), test$Occupancy)

print(paste("Runtime: ", endTime - startTime, "seconds"))
```

```
## [1] "Runtime:  3.88622283935547 seconds"
```

```
print(paste("Accuracy: ", acc_ada))
```

```
## [1] "Accuracy:  0.992592592592593"
```

```
print(paste("MCC: ", mcc_ada))
```

```
## [1] "MCC:  0.979539985681342"
```

Analysis

The following results were observed from a sample run.

Using decision tree as a baseline, the runtime is 0.07 seconds with an accuracy of 0.993237 and an mcc of 0.981571.

Using random forest, the runtime is 3.56 seconds with an accuracy of 0.994525 and an mcc of 0.984930.

Using XGBoost, the runtime is 1.19 seconds with an accuracy of 0.994203 and an mcc of 0.984028.

Using AdaBoost, the runtime is 4.09 seconds with an accuracy of 0.992593 and an mcc of 0.979540.

From these results, we can tell that the decision tree is the fastest, followed by XGBoost, then random forest, and finally AdaBoost. The random forest has the highest accuracy, then XGBoost, then the decision tree, and finally Adaboost. The Matthews correlation coefficient follows the same pattern. It should be noted that all 4 methods produce very similar results and the differences in accuracy and mcc are minimal, whereas the runtimes have more variance.