Ensemble Techniques

Muhammad Shariq Azeem and Aarushi Pandey

10/16/2022

Ensemble Techniques:

Data:

For this assignment, I selected a data set that contains information about the room environment, such as, room temperature, humidity, CO2 levels, etc. I need to use that information to decide if the room is occupied or not.

Source for the data: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+#

Note: The data sets provided through the link were not meeting the "at least 10K rows" requirement, so I used Excel to combine two data sets into one.

Cleaning the data:

- Got rid of the date column because I don't need that for my model.
- Converted 'Occupancy' attribute to a factor.

```
df <- read.csv("data.csv", header=T)
df <- df[,c(2,3,4,5,6,7)]
df$Occupancy <- factor(df$Occupancy)
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.8, replace = FALSE)
train <- df[i,]
test <- df[-i,]
```

Perform Decision Tree as a baseline:

```
library(tree)
library(mccr)
time1 <- Sys.time()
tree <- tree(Occupancy~., data=train)
pred1 <- predict(tree, newdata=test, type="class")
time1 <- Sys.time() - time1

acc1 <- mean(pred1==test$Occupancy)
table(pred1, test$Occupancy)
```

```
##
## pred1    0    1
##     0 1871    1
##     1   13  599
```

```
mcc1 <- mccr(pred1, test$Occupancy)
print(paste("Decission Tree Accuracy =", acc1))
```

```
## [1] "Decission Tree Accuracy = 0.994363929146538"
```

```
print(paste("mcc =", mcc1))
```

```
## [1] "mcc = 0.98480640505193"
```

```
print(paste("time taken =", time1, "secs"))
```

```
## [1] "time taken = 0.0638461112976074 secs"
```

Peform Random Forest:

```
library(randomForest)
```

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

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

```
set.seed(1234)

time2 <- Sys.time()
rf <- randomForest(Occupancy~., data=train, importance=TRUE)
pred2 <- predict(rf, newdata=test, type="class")
time2 <- Sys.time() - time2

acc2 <- mean(pred2==test$Occupancy)
table(pred2, test$Occupancy)
```

```
##
## pred2    0    1
##     0 1876    5
##     1    8  595
```

```
mcc2 <- mccr(pred2, test$Occupancy)
print(paste("Decision Tree Accuracy =", acc2))
```

```
## [1] "Decision Tree Accuracy = 0.994766505636071"
```

```
print(paste("mcc =", mcc2))
```

```
## [1] "mcc = 0.985746192325351"
```

```
print(paste("time taken =", time2, "secs"))
```

```
## [1] "time taken = 2.65636992454529 secs"
```

Perform XGBoost:

```
set.seed(1234)
library(xgboost)

# data must be converted to numeric matrix
# labels must be 0/1 integers
train_label <- ifelse(train$Occupancy==1, 1, 0)
train_matrix <- data.matrix(train[,-6])

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

# running xgboost and predictions
time3 <- Sys.time()
xgb <- xgboost(data=train_matrix, label=train_label, nrounds=100, objective="binary:logistic")
```

```
## [1]  train-logloss:0.445204
## [2]  train-logloss:0.307853
## [3]  train-logloss:0.221570
## [4]  train-logloss:0.164214
## [5]  train-logloss:0.124702
## [6]  train-logloss:0.096408
## [7]  train-logloss:0.076404
## [8]  train-logloss:0.061740
## [9]  train-logloss:0.050548
## [10] train-logloss:0.042458
## [11] train-logloss:0.036731
## [12] train-logloss:0.032274
## [13] train-logloss:0.028885
## [14] train-logloss:0.026440
## [15] train-logloss:0.024485
## [16] train-logloss:0.022803
## [17] train-logloss:0.021820
## [18] train-logloss:0.020592
## [19] train-logloss:0.019478
## [20] train-logloss:0.018892
## [21] train-logloss:0.018190
## [22] train-logloss:0.017444
## [23] train-logloss:0.017089
## [24] train-logloss:0.016634
## [25] train-logloss:0.016235
```

```
## [26] train-logloss:0.015736
## [27] train-logloss:0.015329
## [28] train-logloss:0.014863
## [29] train-logloss:0.014184
## [30] train-logloss:0.013658
## [31] train-logloss:0.013300
## [32] train-logloss:0.013071
## [33] train-logloss:0.012873
## [34] train-logloss:0.012265
## [35] train-logloss:0.011635
## [36] train-logloss:0.011436
## [37] train-logloss:0.010945
## [38] train-logloss:0.010307
## [39] train-logloss:0.010059
## [40] train-logloss:0.009906
## [41] train-logloss:0.009391
## [42] train-logloss:0.009025
## [43] train-logloss:0.008907
## [44] train-logloss:0.008715
## [45] train-logloss:0.008566
## [46] train-logloss:0.008174
## [47] train-logloss:0.008023
## [48] train-logloss:0.007645
## [49] train-logloss:0.007541
## [50] train-logloss:0.007447
## [51] train-logloss:0.007372
## [52] train-logloss:0.007228
## [53] train-logloss:0.007167
## [54] train-logloss:0.007114
## [55] train-logloss:0.007024
## [56] train-logloss:0.006935
## [57] train-logloss:0.006831
## [58] train-logloss:0.006741
## [59] train-logloss:0.006670
## [60] train-logloss:0.006465
## [61] train-logloss:0.006346
## [62] train-logloss:0.006205
## [63] train-logloss:0.006109
## [64] train-logloss:0.006015
## [65] train-logloss:0.005882
## [66] train-logloss:0.005678
## [67] train-logloss:0.005583
## [68] train-logloss:0.005535
## [69] train-logloss:0.005375
## [70] train-logloss:0.005331
## [71] train-logloss:0.005292
## [72] train-logloss:0.005097
## [73] train-logloss:0.005027
## [74] train-logloss:0.004887
## [75] train-logloss:0.004841
## [76] train-logloss:0.004795
## [77] train-logloss:0.004736
## [78] train-logloss:0.004669
## [79] train-logloss:0.004634
```

```
## [80] train-logloss:0.004608
## [81] train-logloss:0.004558
## [82] train-logloss:0.004511
## [83] train-logloss:0.004399
## [84] train-logloss:0.004348
## [85] train-logloss:0.004304
## [86] train-logloss:0.004196
## [87] train-logloss:0.004141
## [88] train-logloss:0.004096
## [89] train-logloss:0.004044
## [90] train-logloss:0.003932
## [91] train-logloss:0.003865
## [92] train-logloss:0.003828
## [93] train-logloss:0.003790
## [94] train-logloss:0.003771
## [95] train-logloss:0.003754
## [96] train-logloss:0.003665
## [97] train-logloss:0.003627
## [98] train-logloss:0.003610
## [99] train-logloss:0.003565
## [100]    train-logloss:0.003545
```

```
pred3 <- predict(xgb, newdata=test_matrix)
pred3 <- ifelse(pred3>0.5, 1, 0)
time3 <- Sys.time() - time3

acc3 <- mean(pred3==test_label)
mcc3 <- mccr(pred3, test_label)
table(pred3, test_label)
```

```
##      test_label
## pred3    0    1
##     0 1877    7
##     1    7  593
```

```
print(paste("accuracy =", acc3))
```

```
## [1] "accuracy = 0.994363929146538"
```

```
print(paste("mcc =", mcc3))
```

```
## [1] "mcc = 0.984617834394904"
```

```
print(paste("time taken =", time3, "secs"))
```

```
## [1] "time taken = 0.529742956161499 secs"
```

Perform 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
```

```
set.seed(6758)

time4 <- Sys.time()
adab <- boosting(Occupancy~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
pred4 <- predict(adab, newdata=test, type="response")
time4 <- Sys.time() - time4

acc4 <- mean(pred4$class==test$Occupancy)
mcc4 <- mccr(pred4$class, test$Occupancy)
table(pred4$class, test$Occupancy)
```

```
##
##        0    1
##   0 1876    5
##   1    8  595
```

```
print(paste("AdaBoost accuracy =", acc4))
```

```
## [1] "AdaBoost accuracy = 0.994766505636071"
```

```
print(paste("mcc =", mcc4))
```

```
## [1] "mcc = 0.985746192325351"
```

```
print(paste("time taken =", time4, "secs"))
```

```
## [1] "time taken = 3.60913014411926 secs"
```

Compare the results:

For this dataset, the accuracies are nearly the same, with a difference of about 0.0004 between the greatest and least accuracy. Ensemble package AdaBoost and Random Forest have the same accuracy (~0.9948), and so do XGBoost and Decision Tree (~0.9944). Surprisingly, AdaBoost and Random Forest also have the same mcc (~0.9857) while XGBoost has an mcc of ~0.9846 and Decision Tree has one of ~0.9848. The mcc values are not much different from the accuracies (with a difference of about 1%). Their runtimes vary vastly though. Decision tree took the least time with only 0.090 seconds needed for model creation and prediction. XGBoost (which needs the data to be converted to a numeric matrix and labels to be 0/1 integers) takes 1.922 seconds while Random forest (which is more complex than Decision tree) takes 6.108 seconds. Lastly, AdaBoost takes the most time which is 18.885 seconds.