Ensemble SVM

Jonathan Ho

October 20, 2022

Load packages and data

```
df <- read.csv("ill_dataset.csv", header=TRUE)
df <- subset(df, select=-c(1))
df$City <- factor(df$City)
df$Gender <- factor(df$Gender)
df$Illness <- factor(df$Illness)
```

Decision tree

```
library(tictoc)
library(tree)
library(mltools)
set.seed(420)

train <- df[sample(nrow(df), 8000),]
test <- df[sample(nrow(df), 2000),]

tic("runtime")

tree_gender <- tree(Gender~., data=train)

time_1 <- toc()
```

```
## runtime: 0.03 sec elapsed
```

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

```
##
## pred     Female Male
##   Female    288  131
##   Male      564 1017
```

```
acc_dt <- mean(pred==test$Gender)
mcc_dt <- mcc(pred, test$Gender)
```

Random Forest

```
library(randomForest)
```

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

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

```
set.seed(420)

tic("runtime")

rf <- randomForest(Gender~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Gender ~ ., 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: 33.65%
## Confusion matrix:
##        Female Male class.error
## Female   2031 1540   0.4312518
## Male     1152 3277   0.2601039
```

```
time_2 <- toc()
```

```
## runtime: 6.25 sec elapsed
```

Prediction of Random Forest.

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

XGBoost

XGBoost did up to 200 rounds. It could technically keep going, but putting it at 4000 rounds it still did not converge.

```
library(xgboost)

tic("runtime")

train_label <- ifelse(train$Gender=="Male", 1, 0)

train_matrix <- data.matrix(train[, -c(2)])
model <- xgboost(data=train_matrix, label=train_label,
                 nrounds=200, objective='binary:logistic')
```

```
## [1]  train-logloss:0.636604
## [2]  train-logloss:0.605219
## [3]  train-logloss:0.586835
## [4]  train-logloss:0.574645
## [5]  train-logloss:0.567190
## [6]  train-logloss:0.561643
## [7]  train-logloss:0.557770
## [8]  train-logloss:0.555150
## [9]  train-logloss:0.553181
## [10] train-logloss:0.550255
## [11] train-logloss:0.547967
## [12] train-logloss:0.545224
## [13] train-logloss:0.544475
## [14] train-logloss:0.543223
## [15] train-logloss:0.541198
## [16] train-logloss:0.540736
## [17] train-logloss:0.539032
## [18] train-logloss:0.538640
## [19] train-logloss:0.537102
## [20] train-logloss:0.535365
## [21] train-logloss:0.533875
## [22] train-logloss:0.532026
## [23] train-logloss:0.531768
## [24] train-logloss:0.530665
## [25] train-logloss:0.528717
## [26] train-logloss:0.528527
## [27] train-logloss:0.526832
## [28] train-logloss:0.523917
## [29] train-logloss:0.522403
## [30] train-logloss:0.521699
## [31] train-logloss:0.520796
## [32] train-logloss:0.519132
## [33] train-logloss:0.518677
## [34] train-logloss:0.517524
## [35] train-logloss:0.515680
## [36] train-logloss:0.515552
## [37] train-logloss:0.515252
## [38] train-logloss:0.514948
## [39] train-logloss:0.513019
## [40] train-logloss:0.512089
## [41] train-logloss:0.510303
## [42] train-logloss:0.509225
## [43] train-logloss:0.508518
## [44] train-logloss:0.507590
## [45] train-logloss:0.505795
## [46] train-logloss:0.505290
## [47] train-logloss:0.503422
## [48] train-logloss:0.502681
## [49] train-logloss:0.502434
## [50] train-logloss:0.501311
## [51] train-logloss:0.499992
## [52] train-logloss:0.498931
```

```
## [53] train-logloss:0.497452
## [54] train-logloss:0.497148
## [55] train-logloss:0.496235
## [56] train-logloss:0.495704
## [57] train-logloss:0.495568
## [58] train-logloss:0.495306
## [59] train-logloss:0.495131
## [60] train-logloss:0.495030
## [61] train-logloss:0.493719
## [62] train-logloss:0.492411
## [63] train-logloss:0.491750
## [64] train-logloss:0.490734
## [65] train-logloss:0.489287
## [66] train-logloss:0.488371
## [67] train-logloss:0.487678
## [68] train-logloss:0.486999
## [69] train-logloss:0.486599
## [70] train-logloss:0.485305
## [71] train-logloss:0.484594
## [72] train-logloss:0.483027
## [73] train-logloss:0.482963
## [74] train-logloss:0.481879
## [75] train-logloss:0.480694
## [76] train-logloss:0.478730
## [77] train-logloss:0.478410
## [78] train-logloss:0.478273
## [79] train-logloss:0.476447
## [80] train-logloss:0.476334
## [81] train-logloss:0.475621
## [82] train-logloss:0.473650
## [83] train-logloss:0.473153
## [84] train-logloss:0.472609
## [85] train-logloss:0.472190
## [86] train-logloss:0.472052
## [87] train-logloss:0.470837
## [88] train-logloss:0.468110
## [89] train-logloss:0.467013
## [90] train-logloss:0.466070
## [91] train-logloss:0.465727
## [92] train-logloss:0.462700
## [93] train-logloss:0.462069
## [94] train-logloss:0.461781
## [95] train-logloss:0.461734
## [96] train-logloss:0.461346
## [97] train-logloss:0.459584
## [98] train-logloss:0.457973
## [99] train-logloss:0.455716
## [100]    train-logloss:0.454506
## [101]    train-logloss:0.453449
## [102]    train-logloss:0.453397
## [103]    train-logloss:0.452954
## [104]    train-logloss:0.452303
```

```
## [105]	train-logloss:0.452105
## [106]	train-logloss:0.452063
## [107]	train-logloss:0.451840
## [108]	train-logloss:0.451783
## [109]	train-logloss:0.451524
## [110]	train-logloss:0.450315
## [111]	train-logloss:0.449184
## [112]	train-logloss:0.448859
## [113]	train-logloss:0.447804
## [114]	train-logloss:0.446025
## [115]	train-logloss:0.445117
## [116]	train-logloss:0.443864
## [117]	train-logloss:0.442713
## [118]	train-logloss:0.442124
## [119]	train-logloss:0.440801
## [120]	train-logloss:0.439794
## [121]	train-logloss:0.438709
## [122]	train-logloss:0.438393
## [123]	train-logloss:0.438338
## [124]	train-logloss:0.438294
## [125]	train-logloss:0.438140
## [126]	train-logloss:0.438104
## [127]	train-logloss:0.436859
## [128]	train-logloss:0.436656
## [129]	train-logloss:0.436593
## [130]	train-logloss:0.435874
## [131]	train-logloss:0.435814
## [132]	train-logloss:0.435770
## [133]	train-logloss:0.435381
## [134]	train-logloss:0.434665
## [135]	train-logloss:0.434598
## [136]	train-logloss:0.434489
## [137]	train-logloss:0.433858
## [138]	train-logloss:0.433762
## [139]	train-logloss:0.433721
## [140]	train-logloss:0.433136
## [141]	train-logloss:0.433029
## [142]	train-logloss:0.432873
## [143]	train-logloss:0.432839
## [144]	train-logloss:0.432704
## [145]	train-logloss:0.431754
## [146]	train-logloss:0.429564
## [147]	train-logloss:0.428689
## [148]	train-logloss:0.427746
## [149]	train-logloss:0.426153
## [150]	train-logloss:0.425718
## [151]	train-logloss:0.424465
## [152]	train-logloss:0.424187
## [153]	train-logloss:0.424028
## [154]	train-logloss:0.422918
## [155]	train-logloss:0.422783
## [156]	train-logloss:0.422013
```

```
## [157]	train-logloss:0.420731
## [158]	train-logloss:0.420271
## [159]	train-logloss:0.419954
## [160]	train-logloss:0.419249
## [161]	train-logloss:0.418809
## [162]	train-logloss:0.418181
## [163]	train-logloss:0.417702
## [164]	train-logloss:0.417305
## [165]	train-logloss:0.417216
## [166]	train-logloss:0.416992
## [167]	train-logloss:0.415885
## [168]	train-logloss:0.415814
## [169]	train-logloss:0.415781
## [170]	train-logloss:0.415744
## [171]	train-logloss:0.415443
## [172]	train-logloss:0.413577
## [173]	train-logloss:0.412087
## [174]	train-logloss:0.410981
## [175]	train-logloss:0.408940
## [176]	train-logloss:0.408194
## [177]	train-logloss:0.408057
## [178]	train-logloss:0.407512
## [179]	train-logloss:0.407362
## [180]	train-logloss:0.407327
## [181]	train-logloss:0.407242
## [182]	train-logloss:0.406215
## [183]	train-logloss:0.405783
## [184]	train-logloss:0.405326
## [185]	train-logloss:0.404241
## [186]	train-logloss:0.404098
## [187]	train-logloss:0.403837
## [188]	train-logloss:0.403786
## [189]	train-logloss:0.403725
## [190]	train-logloss:0.403667
## [191]	train-logloss:0.403027
## [192]	train-logloss:0.402421
## [193]	train-logloss:0.401424
## [194]	train-logloss:0.400722
## [195]	train-logloss:0.400347
## [196]	train-logloss:0.399884
## [197]	train-logloss:0.399850
## [198]	train-logloss:0.399723
## [199]	train-logloss:0.399691
## [200]	train-logloss:0.399620
```

```
time_3 <- toc()
```

```
## runtime: 0.97 sec elapsed
```

Prediction of XGBoost.

```
test_label<- ifelse(test$Gender=="Male", 1, 0)
test_matrix <- data.matrix(test[, -c(2)])

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

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

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(420)

tic("runtime")

adab1 <- boosting(Gender~., 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      16000  -none-  numeric
## prob       16000  -none-  numeric
## class       8000  -none-  character
## importance     4  -none-  numeric
## terms          3  terms   call
## call           6  -none-  call
```

```
time_4 <- toc()
```

```
## runtime: 5.34 sec elapsed
```

Prediction of AdaBoost.

```
pred <- predict(adab1, newdata=test, type="response")
acc_adabag <- mean(pred$class==test$Gender)
mcc_adabag <- mcc(factor(pred$class), test$Gender)
```

Summary of Results

```
cat("Decision Tree:\n")
```

```
## Decision Tree:
```

```
print(paste("accuracy: ", acc_dt))
```

```
## [1] "accuracy:  0.6525"
```

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

```
## [1] "mcc:  0.272084764075715"
```

```
print(paste(time_1$callback_msg))
```

```
## [1] "runtime: 0.03 sec elapsed"
```

```
cat("\nRandom Forest:\n")
```

```
##
## Random Forest:
```

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

```
## [1] "accuracy:  0.6715"
```

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

```
## [1] "mcc:  0.319485402417008"
```

```
print(paste(time_2$callback_msg))
```

```
## [1] "runtime: 6.25 sec elapsed"
```

```
cat("\nXGBoost:\n")
```

```
##
## XGBoost:
```

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

```
## [1] "accuracy:  0.657"
```

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

```
## [1] "mcc:  0.291893606317686"
```

```
print(paste(time_3$callback_msg))
```

```
## [1] "runtime: 0.97 sec elapsed"
```

```
cat("\nAdaBoost\n")
```

```
##
## AdaBoost
```

```
print(paste("accuracy: ", acc_adabag))
```

```
## [1] "accuracy:  0.674"
```

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

```
## [1] "mcc:  0.326201279872501"
```

```
print(paste(time_4$callback_msg))
```

```
## [1] "runtime: 5.34 sec elapsed"
```

Results Discussion

The base accuracy of a decision tree on the data is about 0.6535 with a mcc of 0.2721. Runtime is fast, but most likely because the tree function is probably optimized. Of all the metrics across each ensemble method, AdaBoost resulted in the highest accuracy and mcc. Technically with low rounds of ~40, XGBoost resulted in a higher accuracy, but the method itself should have led to higher accuracy with more rounds. Due to this, XGBoost ended up having the lowest accuracy and mcc of the ensemble methods, but still higher than decision tree for 200 rounds. Runtimes of the ensemble methods showed XGBoost to be the fastest, which was expected. Unexpectedly however, Random Forest had a longer runtime than AdaBoost. My guess is maybe due to how R handles Random Forest, but not sure. I expected AdaBoost to be the slowest ensemble method.