Ensemble Methods

Karen Mazidi

Using a phishing data set with binary target type: +1 or -1. Convert from Weka format to a data frame.

```
library(RWeka)
df <- read.arff("phishing/Training Dataset.arff")
str(df)
```

```
## 'data.frame':    11055 obs. of  31 variables:
##  $ having_IP_Address          : Factor w/ 2 levels "-1","1": 1 2 2 2 2 1 2 2 2 2 ...
##  $ URL_Length                 : Factor w/ 3 levels "1","0","-1": 1 1 2 2 2 2 2 2 2 1 ...
##  $ Shortining_Service         : Factor w/ 2 levels "1","-1": 1 1 1 1 2 2 2 1 2 2 ...
##  $ having_At_Symbol           : Factor w/ 2 levels "1","-1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ double_slash_redirecting   : Factor w/ 2 levels "-1","1": 1 2 2 2 2 1 2 2 2 2 ...
##  $ Prefix_Suffix              : Factor w/ 2 levels "-1","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ having_Sub_Domain          : Factor w/ 3 levels "-1","0","1": 1 2 1 1 3 3 1 1 3 1 ...
##  $ SSLfinal_State             : Factor w/ 3 levels "-1","1","0": 1 2 1 1 2 2 1 1 2 2 ...
##  $ Domain_registeration_length: Factor w/ 2 levels "-1","1": 1 1 1 2 1 1 2 2 1 1 ...
##  $ Favicon                    : Factor w/ 2 levels "1","-1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ port                       : Factor w/ 2 levels "1","-1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ HTTPS_token                : Factor w/ 2 levels "-1","1": 1 1 1 1 2 1 2 1 1 2 ...
##  $ Request_URL                : Factor w/ 2 levels "1","-1": 1 1 1 2 1 1 2 2 1 1 ...
##  $ URL_of_Anchor              : Factor w/ 3 levels "-1","0","1": 1 2 2 2 2 2 1 2 2 2 ...
##  $ Links_in_tags              : Factor w/ 3 levels "1","-1","0": 1 2 2 3 3 3 3 2 1 1 ...
##  $ SFH                        : Factor w/ 3 levels "-1","1","0": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Submitting_to_email        : Factor w/ 2 levels "-1","1": 1 2 1 2 2 1 1 2 2 2 ...
##  $ Abnormal_URL               : Factor w/ 2 levels "-1","1": 1 2 1 2 2 1 1 2 2 2 ...
##  $ Redirect                   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ on_mouseover               : Factor w/ 2 levels "1","-1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ RightClick                 : Factor w/ 2 levels "1","-1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ popUpWidnow                : Factor w/ 2 levels "1","-1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ Iframe                     : Factor w/ 2 levels "1","-1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ age_of_domain              : Factor w/ 2 levels "-1","1": 1 1 2 1 1 2 2 1 2 2 ...
##  $ DNSRecord                  : Factor w/ 2 levels "-1","1": 1 1 1 1 1 2 1 1 1 1 ...
##  $ web_traffic                : Factor w/ 3 levels "-1","0","1": 1 2 3 3 2 3 1 2 3 2 ...
##  $ Page_Rank                  : Factor w/ 2 levels "-1","1": 1 1 1 1 1 1 1 1 2 1 ...
##  $ Google_Index               : Factor w/ 2 levels "1","-1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Links_pointing_to_page     : Factor w/ 3 levels "1","0","-1": 1 1 2 3 1 3 2 2 2 2 ...
##  $ Statistical_report         : Factor w/ 2 levels "-1","1": 1 2 1 2 2 1 1 2 2 2 ...
##  $ Result                     : Factor w/ 2 levels "-1","1": 1 1 1 1 2 2 1 1 2 1 ...
```

Train Test Split

```
set.seed(1234)
i <- sample(nrow(df), .75*nrow(df), replace=FALSE)
train <- df[i,]
```

```
test <- df[-i,]
```

Logistic regression on all predictors

```
library(mltools)
glm1 <- glm(Result~., data=train, family=binomial)
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>0.5, 2, 1)
acc_logreg <- mean(pred==as.integer(test$Result))
mcc_logreg <- mcc(pred, as.integer(test$Result))
print(paste("accuracy=", acc_logreg))
```

```
## [1] "accuracy= 0.937771345875543"
```

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

```
## [1] "mcc= 0.873866225379213"
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
set.seed(1234)
rf <- randomForest(Result~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Result ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 3.35%
## Confusion matrix:
##      -1    1 class.error
## -1 3508  166  0.04518236
## 1   112 4505  0.02425818
```

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

```
## [1] "accuracy= 0.964182344428365"
```

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

```
## [1] "mcc= 0.927467703448562"
```

boosting from adabag library

```
library(adabag)
```

```
## Loading required package: rpart
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
## Loading required package: foreach
## Loading required package: doParallel
## Loading required package: iterators
## Loading required package: parallel
```

```
adab1 <- boosting(Result~., 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       16582  -none-  numeric
## prob        16582  -none-  numeric
## class        8291  -none-  character
## importance     30  -none-  numeric
## terms           3  terms   call
## call            6  -none-  call
```

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

```
## [1] "accuracy= 0.94862518089725"
```

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

```
## [1] "mcc= 0.895811377251421"
```

fastAdaboost

```
library(fastAdaboost)
set.seed(1234)
fadab <- adaboost(Result~., train, 10)
summary(fadab)
```

```
##                    Length Class   Mode
## formula             3     formula call
```

```
## trees              10     -none-  list
## weights            10     -none-  numeric
## classnames          2     -none-  character
## dependent_variable  1     -none-  character
## call                4     -none-  call
```

```
pred <- predict(fadab, newdata=test, type="response")
# pred$class holds the classification
acc_fadab <- mean(pred$class==test$Result)
mcc_fadab <- mcc(pred$class, test$Result)
print(paste("accuracy=", acc_fadab))
```

```
## [1] "accuracy= 0.963820549927641"
```

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

```
## [1] "mcc= 0.927126186350532"
```

XGBoost

```
library(xgboost)
train_label <- ifelse(train$Result==1, 1, 0)
train_matrix <- data.matrix(train[, -31])
model <- xgboost(data=train_matrix, label=train_label,
                 nrounds=100, objective='binary:logistic')
```

```
## [1]  train-error:0.069594
## [2]  train-error:0.067061
## [3]  train-error:0.063080
## [4]  train-error:0.061633
## [5]  train-error:0.060427
## [6]  train-error:0.061392
## [7]  train-error:0.059341
## [8]  train-error:0.055241
## [9]  train-error:0.051863
## [10] train-error:0.051140
## [11] train-error:0.049089
## [12] train-error:0.048848
## [13] train-error:0.044747
## [14] train-error:0.043179
## [15] train-error:0.042214
## [16] train-error:0.041853
## [17] train-error:0.038958
## [18] train-error:0.037028
## [19] train-error:0.036666
## [20] train-error:0.035943
## [21] train-error:0.036666
## [22] train-error:0.034616
## [23] train-error:0.033772
## [24] train-error:0.032807
## [25] train-error:0.032807
## [26] train-error:0.033048
## [27] train-error:0.032083
## [28] train-error:0.030877
## [29] train-error:0.031239
```

```
## [30] train-error:0.030756
## [31] train-error:0.028344
## [32] train-error:0.028585
## [33] train-error:0.029309
## [34] train-error:0.028826
## [35] train-error:0.028947
## [36] train-error:0.028465
## [37] train-error:0.027982
## [38] train-error:0.027258
## [39] train-error:0.026776
## [40] train-error:0.026655
## [41] train-error:0.026655
## [42] train-error:0.025208
## [43] train-error:0.024123
## [44] train-error:0.024123
## [45] train-error:0.022072
## [46] train-error:0.021952
## [47] train-error:0.022313
## [48] train-error:0.022072
## [49] train-error:0.021469
## [50] train-error:0.021831
## [51] train-error:0.021831
## [52] train-error:0.021710
## [53] train-error:0.021590
## [54] train-error:0.021710
## [55] train-error:0.021590
## [56] train-error:0.021228
## [57] train-error:0.021228
## [58] train-error:0.020745
## [59] train-error:0.020022
## [60] train-error:0.019780
## [61] train-error:0.019780
## [62] train-error:0.019419
## [63] train-error:0.019660
## [64] train-error:0.018936
## [65] train-error:0.018816
## [66] train-error:0.018695
## [67] train-error:0.018574
## [68] train-error:0.018092
## [69] train-error:0.018454
## [70] train-error:0.018454
## [71] train-error:0.018213
## [72] train-error:0.017730
## [73] train-error:0.017730
## [74] train-error:0.017730
## [75] train-error:0.018333
## [76] train-error:0.018092
## [77] train-error:0.018454
## [78] train-error:0.017971
## [79] train-error:0.016886
## [80] train-error:0.016765
## [81] train-error:0.016524
## [82] train-error:0.016403
## [83] train-error:0.016645
```

```
## [84] train-error:0.016524
## [85] train-error:0.016524
## [86] train-error:0.016041
## [87] train-error:0.015318
## [88] train-error:0.014956
## [89] train-error:0.014474
## [90] train-error:0.014835
## [91] train-error:0.014956
## [92] train-error:0.014594
## [93] train-error:0.014353
## [94] train-error:0.014353
## [95] train-error:0.014232
## [96] train-error:0.013991
## [97] train-error:0.013991
## [98] train-error:0.013870
## [99] train-error:0.013629
## [100]    train-error:0.013629
```

```
test_label <- ifelse(test$Result==1, 1, 0)
test_matrix <- data.matrix(test[, -31])

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

acc_xg <- mean(pred==test_label)
mcc_xg <- mcc(pred, test_label)
print(paste("accuracy=", acc_xg))
```

```
## [1] "accuracy= 0.964905933429812"
```

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

```
## [1] "mcc= 0.928846940485277"
```

SuperLearner

Had to install packages: ranger kernlab

Super is not super. Can get better results with a lot of parameter tuning, but why? There are better methods.

```
library(SuperLearner)
```

```
## Loading required package: nnls
## Super Learner
## Version: 2.0-26
## Package created on 2019-10-27
```

```
set.seed(1234)
model <- SuperLearner(train_label,
                      train[, -31],
                      family=binomial(),
                      SL.library=list("SL.ranger",
                                      "SL.ksvm",
                                      "SL.ipredbagg"))
```

```
## Loading required namespace: ranger
```

```
## Loading required namespace: kernlab
```

```
model
```

```
##
## Call:
## SuperLearner(Y = train_label, X = train[, -31], family = binomial(), SL.library = list("SL.ranger",
##     "SL.ksvm", "SL.ipredbagg"))
##
##
##                          Risk       Coef
## SL.ranger_All    0.02661880 0.94307843
## SL.ksvm_All      0.03393682 0.05692157
## SL.ipredbagg_All 0.07378788 0.00000000
```

```
probs <- predict.SuperLearner(model, newdata=test[,-31])
pred <- ifelse(probs$pred>0, 1, 0)
acc_sl <- mean(pred==test_label)
mcc_sl <- mcc(as.integer(pred), as.integer(test_label))
print(paste("accuracy=", acc_sl))
```

```
## [1] "accuracy= 0.557163531114327"
```

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

```
## [1] "mcc= 0"
```