Recitation 15: Ensemble Learning

Seamus Wagner

2021-12-07

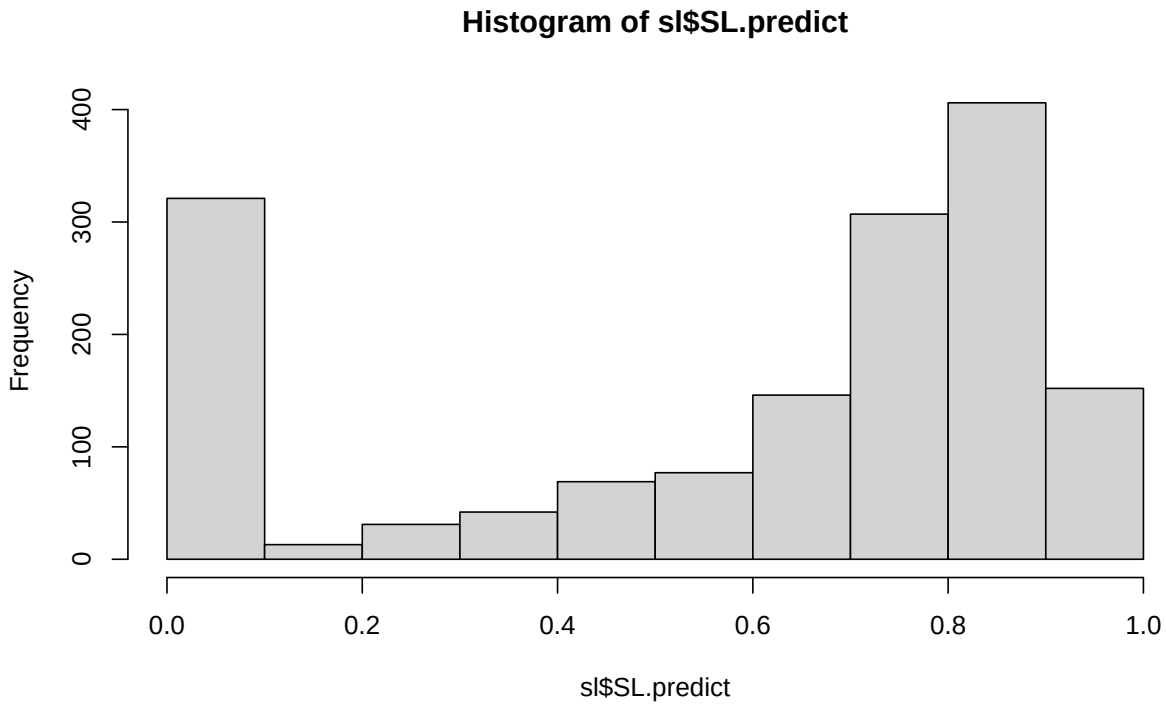
```
rm(list=ls())
library(SuperLearner)
library(kernlab)
library(data.table)
library(ROCR)
set.seed(877654)
load("C:/Users/spw51/Downloads/c2c-A21 (1).Rdata")

# useful for which models you can use. Some require extra packages to work (kernlab, arm, etc)
# ksvm is kernel support vector machine, which is why I have kernlan loaded. I don't end up using it
listWrappers()
```

```
##  [1] "SL.bartMachine"      "SL.bayesglm"         "SL.biglasso"
##  [4] "SL.caret"            "SL.caret.rpart"      "SL.cforest"
##  [7] "SL.earth"            "SL.extraTrees"       "SL.gam"
## [10] "SL.gbm"              "SL.glm"              "SL.glm.interaction"
## [13] "SL.glmnet"           "SL.ipredbagg"        "SL.kernelKnn"
## [16] "SL.knn"              "SL.ksvm"             "SL.lda"
## [19] "SL.leekasso"         "SL.lm"               "SL.loess"
## [22] "SL.logreg"           "SL.mean"             "SL.nnet"
## [25] "SL.nnls"             "SL.polymars"         "SL.qda"
## [28] "SL.randomForest"     "SL.ranger"           "SL.ridge"
## [31] "SL.rpart"            "SL.rpartPrune"       "SL.speedglm"
## [34] "SL.speedlm"          "SL.step"             "SL.step.forward"
## [37] "SL.step.interaction" "SL.stepAIC"          "SL.svm"
## [40] "SL.template"         "SL.xgboost"
## [1] "All"
## [1] "screen.corP"           "screen.corRank"        "screen.glmnet"
## [4] "screen.randomForest"   "screen.SIS"            "screen.template"
## [7] "screen.ttest"          "write.screen.template"
```

```
Y <- data$responder # SuperLearner wants a separate X and Y as input
X <- data[, -c("Y", "responder")]
?SuperLearner
sl <- SuperLearner(
Y, X, newX = X, family = binomial(), #binomial as we want a logit in this case since our outcome is res
SL.library = "SL.glm", #the type of model to use
method = "method.AUC", #the type of method to evaluate the model
verbose = FALSE,
cvControl = SuperLearner.CV.control( #cvcontrol is for control in cross valdiation, v is number of fold
V = 10L,
```

```
stratifyCV = FALSE,
shuffle = TRUE,
validRows = NULL))
?SuperLearner
hist(sl$SL.predict)
```




```
calc_AUROC <-
  function(predictions, labels) {
    performance(prediction(predictions, labels), measure = "auc")@y.values[[1]]
  }
calc_AUROC(sl$SL.predict[, 1], data[, responder])
```

```
## [1] 0.8673058
```

```
mean(sapply(1:10, function(i) {
  test_rows <- sl$validRows[[i]] # here's where the test set rows live
  calc_AUROC(sl$SL.predict[test_rows, 1], data[test_rows, responder])
}))
```

```
## [1] 0.8674496
```

```
# Inverse probability weights are 1 over predicted values, that way you get a weight for each value

# After that, you can apply those weights to your Y := formula for all responders.

# That will give you a weighted mean estimate to compare to the naive one.
```