Ensemble Methods

Spencer Gray

2022-10-23

Data Setup

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
set.seed(1234)

star <- read.csv("star_classification.csv")

star$class[which(star$class=="STAR")] <- 0
star$class[which(star$class=="GALAXY")] <- 1
star$class[which(star$class=="QSO")] <- 2
star$class <- as.factor(star$class)

train <- sample_n(star, 12000)
test <- sample_n(star, 3000)


dim(star)
```

```
## [1] 100000     18
```

```
head(star)
```

```
##         obj_ID    alpha      delta        u        g        r        i        z
## 1 1.237661e+18 135.6891 32.4946318 23.87882 22.27530 20.39501 19.16573 18.79371
## 2 1.237665e+18 144.8261 31.2741849 24.77759 22.83188 22.58444 21.16812 21.61427
## 3 1.237661e+18 142.1888 35.5824442 25.26307 22.66389 20.60976 19.34857 18.94827
## 4 1.237663e+18 338.7410 -0.4028276 22.13682 23.77656 21.61162 20.50454 19.25010
## 5 1.237680e+18 345.2826 21.1838656 19.43718 17.58028 16.49747 15.97711 15.54461
## 6 1.237680e+18 340.9951 20.5894763 23.48827 23.33776 21.32195 20.25615 19.54544
##   run_ID rerun_ID cam_col field_ID  spec_obj_ID class  redshift plate   MJD
## 1   3606      301       2       79 6.543777e+18     1 0.6347936  5812 56354
## 2   4518      301       5      119 1.176014e+19     1 0.7791360 10445 58158
## 3   3606      301       2      120 5.152200e+18     1 0.6441945  4576 55592
## 4   4192      301       3      214 1.030107e+19     1 0.9323456  9149 58039
## 5   8102      301       3      137 6.891865e+18     1 0.1161227  6121 56187
## 6   8102      301       3      110 5.658977e+18     2 1.4246590  5026 55855
##   fiber_ID
## 1      171
## 2      427
## 3      299
## 4      775
## 5      842
## 6      741
```

Random Forest

Random Forest has a relatively fast training time compared to our SVM model and actually performs 1% better at an accuracy of 88.3%.

```
library(mltools)
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
set.seed
```

```
## function (seed, kind = NULL, normal.kind = NULL, sample.kind = NULL)
## {
##     kinds <- c("Wichmann-Hill", "Marsaglia-Multicarry", "Super-Duper",
##         "Mersenne-Twister", "Knuth-TAOCP", "user-supplied", "Knuth-TAOCP-2002",
##         "L'Ecuyer-CMRG", "default")
##     n.kinds <- c("Buggy Kinderman-Ramage", "Ahrens-Dieter", "Box-Muller",
##         "user-supplied", "Inversion", "Kinderman-Ramage", "default")
##     s.kinds <- c("Rounding", "Rejection", "default")
##     if (length(kind)) {
##         if (!is.character(kind) || length(kind) > 1L)
##             stop("'kind' must be a character string of length 1 (RNG to be used).")
##         if (is.na(i.knd <- pmatch(kind, kinds) - 1L))
##             stop(gettextf("'%s' is not a valid abbreviation of an RNG",
##                 kind), domain = NA)
##         if (i.knd == length(kinds) - 1L)
##             i.knd <- -1L
##     }
##     else i.knd <- NULL
##     if (!is.null(normal.kind)) {
##         if (!is.character(normal.kind) || length(normal.kind) !=
##             1L)
##             stop("'normal.kind' must be a character string of length 1")
##         normal.kind <- pmatch(normal.kind, n.kinds) - 1L
##         if (is.na(normal.kind))
##             stop(gettextf("'%s' is not a valid choice", normal.kind),
##                 domain = NA)
##         if (normal.kind == 0L)
##             stop("buggy version of Kinderman-Ramage generator is not allowed",
##                 domain = NA)
##         if (normal.kind == length(n.kinds) - 1L)
##             normal.kind <- -1L
##     }
##     if (!is.null(sample.kind)) {
##         if (!is.character(sample.kind) || length(sample.kind) !=
##             1L)
##             stop("'sample.kind' must be a character string of length 1")
##         sample.kind <- pmatch(sample.kind, s.kinds) - 1L
##         if (is.na(sample.kind))
##             stop(gettextf("'%s' is not a valid choice", sample.kind),
##                 domain = NA)
##         if (sample.kind == 0L)
##             warning("non-uniform 'Rounding' sampler used", domain = NA)
##         if (sample.kind == length(s.kinds) - 1L)
##             sample.kind <- -1L
##     }
##     .Internal(set.seed(seed, i.knd, normal.kind, sample.kind))
## }
## <bytecode: 0x000001ec4d4063b0>
## <environment: namespace:base>
```

```
start_time <- Sys.time()
rf <- randomForest(class~alpha + delta + u + g + r + i + z, data=train, importance=TRUE)
end_time <- Sys.time()

pred <- predict(rf, newdata=test, type="response")
acc <- mean(pred==test$class)
print(paste("accuracy=", acc))
```

```
## [1] "accuracy= 0.880333333333333"
```

```
print(paste("time=",end_time - start_time," seconds"))
```

```
## [1] "time= 10.5865848064423  seconds"
```

XGBoost

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
set.seed(1234)

train_lb <- ifelse(train$class==1, 1, 0)
train_mt <- data.matrix(train[, c(2, 3, 4, 5, 6, 7, 8)])

start_time <- Sys.time()
model <- xgboost(data=train_mt, label=train_lb, nrounds=100, objective='binary:logistic')
```

```
## [1]  train-logloss:0.546834
## [2]  train-logloss:0.468197
## [3]  train-logloss:0.415330
## [4]  train-logloss:0.379439
## [5]  train-logloss:0.351279
## [6]  train-logloss:0.330897
## [7]  train-logloss:0.316031
## [8]  train-logloss:0.302782
## [9]  train-logloss:0.294957
## [10] train-logloss:0.285391
## [11] train-logloss:0.275389
## [12] train-logloss:0.270761
## [13] train-logloss:0.268544
## [14] train-logloss:0.261245
## [15] train-logloss:0.254703
## [16] train-logloss:0.249504
## [17] train-logloss:0.247830
## [18] train-logloss:0.245217
## [19] train-logloss:0.237363
## [20] train-logloss:0.232807
## [21] train-logloss:0.229949
## [22] train-logloss:0.228659
## [23] train-logloss:0.225886
## [24] train-logloss:0.224838
## [25] train-logloss:0.219499
## [26] train-logloss:0.215163
## [27] train-logloss:0.214428
## [28] train-logloss:0.209905
## [29] train-logloss:0.206595
## [30] train-logloss:0.203462
## [31] train-logloss:0.202299
## [32] train-logloss:0.201753
## [33] train-logloss:0.199526
## [34] train-logloss:0.198438
## [35] train-logloss:0.196627
## [36] train-logloss:0.195811
## [37] train-logloss:0.194673
## [38] train-logloss:0.193742
## [39] train-logloss:0.190773
## [40] train-logloss:0.187936
## [41] train-logloss:0.186399
## [42] train-logloss:0.185725
## [43] train-logloss:0.184594
## [44] train-logloss:0.181410
## [45] train-logloss:0.180937
## [46] train-logloss:0.179543
## [47] train-logloss:0.177713
## [48] train-logloss:0.177202
## [49] train-logloss:0.175383
## [50] train-logloss:0.172456
## [51] train-logloss:0.170163
## [52] train-logloss:0.168858
```

```
## [53] train-logloss:0.167616
## [54] train-logloss:0.165653
## [55] train-logloss:0.163890
## [56] train-logloss:0.162207
## [57] train-logloss:0.161305
## [58] train-logloss:0.159764
## [59] train-logloss:0.157641
## [60] train-logloss:0.157367
## [61] train-logloss:0.155700
## [62] train-logloss:0.153659
## [63] train-logloss:0.152602
## [64] train-logloss:0.150626
## [65] train-logloss:0.149524
## [66] train-logloss:0.148800
## [67] train-logloss:0.148070
## [68] train-logloss:0.146765
## [69] train-logloss:0.144736
## [70] train-logloss:0.143560
## [71] train-logloss:0.142620
## [72] train-logloss:0.141574
## [73] train-logloss:0.140225
## [74] train-logloss:0.139757
## [75] train-logloss:0.139115
## [76] train-logloss:0.137675
## [77] train-logloss:0.135841
## [78] train-logloss:0.134621
## [79] train-logloss:0.133644
## [80] train-logloss:0.133070
## [81] train-logloss:0.132047
## [82] train-logloss:0.131717
## [83] train-logloss:0.130385
## [84] train-logloss:0.129211
## [85] train-logloss:0.128959
## [86] train-logloss:0.128009
## [87] train-logloss:0.127007
## [88] train-logloss:0.126746
## [89] train-logloss:0.126464
## [90] train-logloss:0.126238
## [91] train-logloss:0.126035
## [92] train-logloss:0.124431
## [93] train-logloss:0.122428
## [94] train-logloss:0.121814
## [95] train-logloss:0.121585
## [96] train-logloss:0.121314
## [97] train-logloss:0.120783
## [98] train-logloss:0.119151
## [99] train-logloss:0.118053
## [100]    train-logloss:0.117411
```

```
end_time <- Sys.time()

test_lb <- ifelse(test$class==1, 1, 0)
test_mt <- data.matrix(test[, c(2, 3, 4, 5, 6, 7, 8)])

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

acc <- mean(pred==test_lb)
print(paste("accuracy=", acc))
```

```
## [1] "accuracy= 0.912"
```

```
print(paste("time=",end_time-start_time," seconds"))
```

```
## [1] "time= 0.777476787567139  seconds"
```

Super Learner

```
library(SuperLearner)
```

```
## Loading required package: nnls
```

```
## Loading required package: gam
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20.2
```

```
## Super Learner
```

```
## Version: 2.0-28
```

```
## Package created on 2021-05-04
```

```
set.seed(1234)

start_time <- Sys.time()
model <- SuperLearner(train_lb, train[, c(2, 3, 4, 5, 6, 7, 8)], family=binomial(), SL.library=l
ist("SL.ranger", "SL.ksvm", "SL.ipredbagg"))
```

```
## Loading required namespace: ipred
```

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

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

```
model
```

```
##
## Call:
## SuperLearner(Y = train_lb, X = train[, c(2, 3, 4, 5, 6, 7, 8)], family = binomial(),
##     SL.library = list("SL.ranger", "SL.ksvm", "SL.ipredbagg"))
##
##
##                         Risk      Coef
## SL.ranger_All    0.08128012 0.9509731
## SL.ksvm_All      0.09308238 0.0490269
## SL.ipredbagg_All 0.13428241 0.0000000
```

```
end_time <- Sys.time()

probs <- predict.SuperLearner(model, newdata=test[, c(2, 3, 4, 5, 6, 7, 8)])
pred <- ifelse(probs$pred>0, 1, 0)
acc <- mean(pred==test_lb)
print(paste("accuracy=",acc))
```

```
## [1] "accuracy= 0.599333333333333"
```

```
print(paste("time=",end_time-start_time," minutes"))
```

```
## [1] "time= 3.80151248375575  minutes"
```

Conclusion

Random Forest algorithm was quite fast in its training and performed relatively well. Especially considering the time it took to train the SVM model and how it performs worse. XGBoost took a fair while longer but ended on a new highscore for accuracy at 91.2%. A prime example of a model that both takes a long time to run and performs poorly is the super learner, taking over 3 minutes with an accuracy of 60%.