MSD 2019 Final Project

An extension (regularized model) of Greed and Grievance in Civil War by Paul Collier and Anke Hoeffler, 2000

Kiran Ramesh (kr2789), Sai Srujan Chinta (sc4401), Bhavya Shahi (bs3118)

2019-05-13 14:11:07

Contents

Reading Data 1

Helper Functions 1

Free variables 2

Regularized Model 2

Converting Results into DataFrame 4

Reading Data

```
setwd(".")
options(scipen = 100, digits = 4)

data <- read.dta("data/G&G.dta")
data <- data[!is.na(data$warsa), ]
```

Helper Functions

```
summarize_into_table <- function(summary_obj) {
    # takes summary object as input, and returns a DF
    # res.  to print res, use print(summ(obj), quote =
    # FALSE) is you don't want quotes

    options(digits = 4)
    res <- t(round(summary_obj$coefficients, digits = 4))
    z.values <- res[4, ]
    Signif <- symnum(z.values, corr = FALSE, na = FALSE,
        cutpoints = c(0, 0.01, 0.05, 0.1, 1), symbols = c("***",
            "**", "*", ""))
    res <- rbind(res, Signif)
    res <- t(res)
    res <- res[-1, ]
    res
}

comma_sep = function(x) {
```

```
    x = strsplit(x, "")
}

convert2dArrayToDf = function(all_tests) {
    model_names <- all_tests[, 1]

    invisible(apply(all_tests, 2, as.numeric))
    invisible(sapply(all_tests, as.numeric))
    class(all_tests) <- "numeric"
    storage.mode(all_tests) <- "numeric"

    all_tests <- as.data.frame(all_tests)

    all_tests[, 1] <- model_names

    return(all_tests)
}
```

Free variables

```
k <- 5

all_tests <- matrix(data = 0, nrow = k, ncol = 6)
colnames(all_tests) <- c("model", "test_index", "sens",
    "spec", "auc", "accuracy")

set.seed(42)
shuffled_data <- data[sample(nrow(data)), ]
folds <- cut(seq(1, nrow(shuffled_data)), breaks = k,
    labels = FALSE)

thresholding_flag = TRUE

threshold_value = as.numeric(params$threshold)

sens_index <- 3
spec_index <- 4
auc_index <- 5
accuracy_index <- 6
```

Regularized Model

```
for (testIndex in 1:k) {

    regularized.data <- shuffled_data %>% select(warsa,
        coldwar, prevwara, peace, elfo, rf, frac, geogia,
        mount, lnpop, sxp, sxp2, dem, ygini, lgini,
        lngdp_, gy1, secm, diaspeaa, difdpeaa, diahpeaa,
        pol16, oilsxp, oilsxp2, etdo4590)
```

```
# Segement your data by fold using the which()
# function
testIndexes <- which(folds == testIndex, arr.ind = TRUE)
testData <- regularized.data[testIndexes, ]
trainData <- regularized.data[-testIndexes, ]

trainData <- na.omit(trainData)
testData <- na.omit(testData)

train_x <- as.matrix(trainData[, 2:ncol(trainData)])
train_y <- as.matrix(trainData$warsa)

test_x <- as.matrix(testData[, 2:ncol(testData)])
test_y <- as.matrix(testData$warsa)

regularized_fit <- cv.glmnet(x = train_x, y = train_y,
    family = "binomial", type.measure = "auc")
regularized_predict <- predict(regularized_fit,
    test_x, type = "response", s = "lambda.min")

regularized_y.hat <- as.matrix(regularized_predict)

all_tests[testIndex, 1] <- paste(c("regularized"),
    collapse = ".")
all_tests[testIndex, 2] <- as.numeric(testIndex)

regularized_y.hat_normalized <- regularized_y.hat

regularized_y.hat_normalized[regularized_y.hat_normalized >=
    threshold_value] <- 1
regularized_y.hat_normalized[regularized_y.hat_normalized <
    threshold_value] <- 0

regularized_predict_normalized <- prediction(regularized_y.hat_normalized,
    test_y)

len <- length(regularized_predict_normalized@fp[[1]])
fp <- as.numeric(regularized_predict_normalized@fp[[1]][[len -
    1]])
tp <- as.numeric(regularized_predict_normalized@tp[[1]][[len -
    1]])
fn <- as.numeric(regularized_predict_normalized@fn[[1]][[len -
    1]])
tn <- as.numeric(regularized_predict_normalized@tn[[1]][[len -
    1]])

all_tests[testIndex, sens_index] <- tp/(tp + fn)
all_tests[testIndex, spec_index] <- tn/(tn + fp)

all_tests[testIndex, accuracy_index] <- (tp + tn)/(tp +
    tn + fp + fn)

regularized_predict <- prediction(regularized_y.hat,
```

```
        test_y)
    regular_auc <- performance(regularized_predict,
        measure = "auc")
    all_tests[testIndex, auc_index] <- as.numeric(unlist(slot(regular_auc,
        "y.values")))
}

lower_lim = 1
upper_lim = k
print(convert2dArrayToDf(all_tests[lower_lim:upper_lim,
    1:6]))
```

```
##         model test_index   sens   spec    auc accuracy
## 1 regularized          1 0.3333 0.9661 0.7966   0.9355
## 2 regularized          2 0.0000 1.0000 0.9153   0.9394
## 3 regularized          3 0.0000 1.0000 0.8776   0.9608
## 4 regularized          4 0.1667 0.9808 0.7788   0.8966
## 5 regularized          5 0.0000 1.0000 0.8480   0.9273
```

Converting Results into DataFrame

```
all_tests <- convert2dArrayToDf(all_tests)

result <- aggregate(all_tests[, 3:6], list(all_tests$model),
    mean)

names(result)[1] <- "model"

print(result)
```

```
##         model sens   spec    auc accuracy
## 1 regularized  0.1 0.9894 0.8433   0.9319
```

```
write.csv(result, file = paste0("Project_Extension_3_Threshold_",
    params$threshold, ".csv"))
```

The following is a list of all packages used to generate these results. (Leave at very end of file.)

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin17.7.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS/LAPACK: /usr/local/Cellar/openblas/0.3.5/lib/libopenblasp-r0.3.5.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
```

```
##  [1] glmnet_2.0-16     foreach_1.4.4     ROCR_1.0-7
##  [4] gplots_3.0.1.1    lme4_1.1-21       Matrix_1.2-15
##  [7] DescTools_0.99.28 foreign_0.8-71    forcats_0.3.0
## [10] stringr_1.4.0     dplyr_0.8.0       purrr_0.3.0
## [13] readr_1.3.1       tidyr_0.8.2       tibble_2.0.1
## [16] ggplot2_3.1.0     tidyverse_1.2.1   scales_1.0.0
## [19] here_0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0          lubridate_1.7.4    mvtnorm_1.0-10
##  [4] lattice_0.20-38     gtools_3.8.1       assertthat_0.2.0
##  [7] rprojroot_1.3-2     digest_0.6.18      R6_2.4.0
## [10] cellranger_1.1.0    plyr_1.8.4         backports_1.1.3
## [13] evaluate_0.13       httr_1.4.0         pillar_1.3.1
## [16] rlang_0.3.1         lazyeval_0.2.1     readxl_1.3.0
## [19] rstudioapi_0.9.0    minqa_1.2.4        gdata_2.18.0
## [22] nloptr_1.2.1        rmarkdown_1.11     splines_3.5.2
## [25] munsell_0.5.0       broom_0.5.1        compiler_3.5.2
## [28] modelr_0.1.3        xfun_0.4           pkgconfig_2.0.2
## [31] manipulate_1.0.1    htmltools_0.3.6    tidyselect_0.2.5
## [34] expm_0.999-4        codetools_0.2-15   crayon_1.3.4
## [37] withr_2.1.2         MASS_7.3-51.1      bitops_1.0-6
## [40] grid_3.5.2          nlme_3.1-137       jsonlite_1.6
## [43] gtable_0.2.0        formatR_1.6        magrittr_1.5
## [46] KernSmooth_2.23-15 cli_1.0.1          stringi_1.3.1
## [49] xml2_1.2.0          generics_0.0.2     boot_1.3-20
## [52] iterators_1.0.10    tools_3.5.2        glue_1.3.0
## [55] hms_0.4.2           yaml_2.2.0         colorspace_1.4-0
## [58] caTools_1.17.1.2    rvest_0.3.2        knitr_1.21
## [61] haven_2.0.0
```