

MSD 2019 Final Project

An extension (in sample testing) 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 05:59:32

Contents

Reading Data	1
Helper Functions	1
Free variables	2
Opportunity Models	2
Grievance Models	4
Combined Model	6
Converting Results into DataFrame	8

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, "")
}

```

Free variables

```

all_tests <- matrix(data = 0, nrow = 15, ncol = 5)
colnames(all_tests) <- c("model", "sens", "spec", "auc",
  "accuracy")

thresholding_flag = TRUE
threshold_value = as.numeric(params$threshold)

sens_index <- 2
spec_index <- 3
auc_index <- 4
accuracy_index <- 5

```

Opportunity Models

Generating the various opportunity models

```

# Opportunity Models

filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,prevwara,mount,geogia,frac,lnpop",
  "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,frac,lnpop",
  "warsa,sxp,sxp2,coldwar,lngdp_,gy1,peace,mount,geogia,frac,lnpop",
  "warsa,sxp,sxp2,lngdp_,peace,lnpop,diaspeaa", "warsa,sxp,sxp2,lngdp_,peace,lnpop,difdpeaa,diahpeaa")

regression_formula_list <- list("warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + prevwara + mount +
  "warsa ~  sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + frac + lnpop",
  "warsa ~  sxp + sxp2 + coldwar + lngdp_ + gy1 + peace + mount + geogia + frac + lnpop",
  "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + diaspeaa",
  "warsa ~  sxp + sxp2 + lngdp_ + peace + lnpop + difdpeaa + diahpeaa")

for (i in c(1:5)) {

  filtering_columns <- strsplit(filtering_columns_list[[i]],
    ",")[[1]]

  opportunity.data <- data[, filtering_columns]

  testData <- opportunity.data
  trainData <- opportunity.data

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

```

```

# na.omit(testData)

opportunity_fit <- glm(as.formula(regression_formula_list[[i]]),
  family = binomial(link = "logit"), data = trainData)

opportunity_predict <- predict(opportunity_fit,
  newdata = testData, type = "response")

opportunity_y.hat <- as.matrix(opportunity_predict)

y <- as.matrix(testData$warsa)

all_tests[i, 1] <- paste(c("opportunity", i), collapse = ".")

if (thresholding_flag == TRUE) {

  opportunity_y.hat_normalized <- opportunity_y.hat

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

  opp_predict_normalized <- prediction(opportunity_y.hat_normalized,
    y)

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

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

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

  opp_predict <- prediction(opportunity_y.hat,
    y)
  opp_auc <- performance(opp_predict, measure = "auc")
  all_tests[i, auc_index] <- as.numeric(unlist(slot(opp_auc,
    "y.values"))))

} else {

  opp_predict <- prediction(opportunity_y.hat,
    y)

```

```

    opp_f <- performance(opp_predict, measure = "f")
    opp_where.F <- which.max(as.numeric(unlist(slot(opp_f,
      "y.values"))))
    opp_what.F <- performance(opp_predict, measure = "sens",
      x.measure = "spec")

    all_tests[i, sens_index] <- as.numeric(unlist(slot(opp_what.F,
      "y.values")))[opp_where.F]
    all_tests[i, spec_index] <- as.numeric(unlist(slot(opp_what.F,
      "x.values")))[opp_where.F]

    opp_auc <- performance(opp_predict, measure = "auc")
    all_tests[i, auc_index] <- as.numeric(unlist(slot(opp_auc,
      "y.values"))))

  }
}

```

Grievance Models

Generating the various grievance models

```
# Grievance Models
```

```

filtering_columns_list <- list("warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop",
  "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,ygini",
  "warsa,elfo,rf,pol16,etdo4590,dem,peace,mount,geogia,lnpop,lgini")

regression_formula_list <- list("warsa ~ elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia +
  "warsa ~ elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + ygini",
  "warsa ~ elfo + rf + pol16 + etdo4590 + dem + peace + mount + geogia + lnpop + lgini")

for (i in c(1:3)) {

  filtering_columns <- strsplit(filtering_columns_list[[i]],
    ",")[1]

  grievance.data <- data[, filtering_columns]

  testData <- grievance.data
  trainData <- grievance.data

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

  grievance_fit <- glm(as.formula(regression_formula_list[[i]]),
    family = binomial(link = "logit"), data = trainData)

  grievance_predict <- predict(grievance_fit, newdata = testData,
    type = "response")

```

```

grievance_y.hat <- as.matrix(grievance_predict)

y <- as.matrix(testData$warsa)

all_tests[5 + i, 1] <- paste(c("grievance", i),
  collapse = ".")

if (thresholding_flag == TRUE) {
  grievance_y.hat_normalized <- grievance_y.hat

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

  griev_predict_normalized <- prediction(grievance_y.hat_normalized,
    y)

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

  all_tests[5 + i, sens_index] <- tp/(tp + fn)
  all_tests[5 + i, spec_index] <- tn/(tn + fp)

  all_tests[5 + i, accuracy_index] <- (tp + tn)/(tp +
    tn + fp + fn)

  griev_predict <- prediction(grievance_y.hat,
    y)
  griev_auc <- performance(griev_predict, measure = "auc")
  all_tests[5 + i, auc_index] <- as.numeric(unlist(slot(griev_auc,
    "y.values"))))
} else {

  griev_predict <- prediction(grievance_y.hat,
    y)

  griev_f <- performance(griev_predict, measure = "f")
  griev_where.F <- which.max(as.numeric(unlist(slot(griev_f,
    "y.values")))))
  griev_what.F <- performance(griev_predict,
    measure = "sens", x.measure = "spec")

  all_tests[5 + i, sens_index] <- as.numeric(unlist(slot(griev_what.F,
    "y.values")))[griev_where.F]

```

```

    all_tests[5 + i, spec_index] <- as.numeric(unlist(slot(griev_what.F,
      "x.values")))[griev_where.F]

    griev_auc <- performance(griev_predict, measure = "auc")
    all_tests[5 + i, auc_index] <- as.numeric(unlist(slot(griev_auc,
      "y.values")))

  }

}

```

Combined Model

Generating the combined opportunity and grievance models

Combined Models

```

filtering_columns_list <- list("warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,grievxb",
  "warsa,peace,mount,geogia,lnpop,elfo,rf,pol16,etdo4590,dem,greedxb",
  "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem,ygini",
  "warsa,sxp,sxp2,coldwar,secm,gy1,peace,mount,geogia,lnpop,frac,elfo,rf,pol16,etdo4590,dem",
  "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590",
  "warsa,sxp,sxp2,lngdp_,gy1,peace,geogia,lnpop,frac,etdo4590",
  "warsa,sxp,sxp2,secm,gy1,peace,geogia,lnpop,frac,etdo4590,oilsxp,oilsxp2")

regression_formula_list <- list("warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + elfo + rf + pol16 + etdo4590 + dem + greedxb",
  "warsa ~ peace + mount + geogia + lnpop + elfo + rf + pol16 + etdo4590 + dem + greedxb",
  "warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf + pol16 + etdo4590 + dem + ygini",
  "warsa ~ sxp + sxp2 + coldwar + secm + gy1 + peace + mount + geogia + lnpop + frac + elfo + rf + pol16 + etdo4590 + dem",
  "warsa ~ sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590",
  "warsa ~ sxp + sxp2 + lngdp_ + gy1 + peace + geogia + lnpop + frac + etdo4590",
  "warsa ~ sxp + sxp2 + secm + gy1 + peace + geogia + lnpop + frac + etdo4590 + oilsxp + oilsxp2")

for (i in c(1:7)) {

  filtering_columns <- strsplit(filtering_columns_list[[i]],
    ",")[1]

  combined.data <- data[, filtering_columns]

  testData <- combined.data
  trainData <- combined.data

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

  combined_fit <- glm(as.formula(regression_formula_list[[i]]),
    family = binomial(link = "logit"), data = trainData)

  combined_predict <- predict(combined_fit, newdata = testData,

```

```

    type = "response")

combined_y.hat <- as.matrix(combined_predict)

y <- as.matrix(testData$warsa)

all_tests[(5 + 3) + i, 1] <- paste(c("combined",
    i), collapse = ".")

if (thresholding_flag == TRUE) {
    combined_y.hat_normalized <- combined_y.hat

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

    comb_predict_normalized <- prediction(combined_y.hat_normalized,
        y)

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

    all_tests[(5 + 3) + i, sens_index] <- tp/(tp +
        fn)
    all_tests[(5 + 3) + i, spec_index] <- tn/(tn +
        fp)

    all_tests[(5 + 3) + i, accuracy_index] <- (tp +
        tn)/(tp + tn + fp + fn)

    comb_predict <- prediction(combined_y.hat,
        y)
    comb_auc <- performance(comb_predict, measure = "auc")
    all_tests[(5 + 3) + i, auc_index] <- as.numeric(unlist(slot(comb_auc,
        "y.values"))))

} else {

    comb_predict <- prediction(combined_y.hat,
        y)

    comb_f <- performance(comb_predict, measure = "f")
    comb_where.F <- which.max(as.numeric(unlist(slot(comb_f,
        "y.values"))))
    comb_what.F <- performance(comb_predict, measure = "sens",

```

```

        x.measure = "spec")

    all_tests[(5 + 3) + i, sens_index] <- as.numeric(unlist(slot(comb_what.F,
        "y.values")))[comb_where.F]
    all_tests[(5 + 3) + i, spec_index] <- as.numeric(unlist(slot(comb_what.F,
        "x.values")))[comb_where.F]

    comb_auc <- performance(comb_predict, measure = "auc")
    all_tests[(5 + 3) + i, auc_index] <- as.numeric(unlist(slot(comb_auc,
        "y.values")))

}

}

```

Converting Results into DataFrame

```

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

result <- all_tests

print(result)

```

	model	sens	spec	auc	accuracy
## 1	opportunity.1	0.06522	0.9953	0.8540	0.9331
## 2	opportunity.2	0.06522	0.9953	0.8558	0.9331
## 3	opportunity.3	0.07692	0.9957	0.8360	0.9320
## 4	opportunity.4	0.06250	0.9982	0.8610	0.9479
## 5	opportunity.5	0.09375	0.9982	0.8610	0.9496
## 6	grievance.1	0.00000	1.0000	0.7774	0.9306
## 7	grievance.2	0.00000	1.0000	0.7593	0.9321
## 8	grievance.3	0.00000	1.0000	0.8065	0.9370
## 9	combined.1	0.08696	0.9952	0.8573	0.9323
## 10	combined.2	0.08696	0.9935	0.8602	0.9308
## 11	combined.3	0.03125	1.0000	0.8498	0.9353
## 12	combined.4	0.08696	0.9919	0.8614	0.9293
## 13	combined.5	0.06522	0.9922	0.8598	0.9302
## 14	combined.6	0.07692	0.9957	0.8358	0.9320
## 15	combined.7	0.08889	0.9885	0.8943	0.9266

```

write.csv(result, file = paste0("Project_Extension_1_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/libopenblas-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] ROCR_1.0-7      gplots_3.0.1.1  lme4_1.1-21
## [4] Matrix_1.2-15  DescTools_0.99.28 foreign_0.8-71
## [7] forcats_0.3.0  stringr_1.4.0   dplyr_0.8.0
## [10] purrr_0.3.0    readr_1.3.1     tidyr_0.8.2
## [13] tibble_2.0.1   ggplot2_3.1.0   tidyverse_1.2.1
## [16] scales_1.0.0   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    crayon_1.3.4     withr_2.1.2
## [37] MASS_7.3-51.1   bitops_1.0-6     grid_3.5.2
## [40] nlme_3.1-137    jsonlite_1.6     gtable_0.2.0
## [43] formatR_1.6     magrittr_1.5     KernSmooth_2.23-15
## [46] cli_1.0.1       stringi_1.3.1    xml2_1.2.0
## [49] generics_0.0.2  boot_1.3-20      tools_3.5.2
## [52] glue_1.3.0      hms_0.4.2        yaml_2.2.0
## [55] colorspace_1.4-0 caTools_1.17.1.2 rvest_0.3.2
## [58] knitr_1.21      haven_2.0.0
```