question3

oscar

2025-10-29

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(broom)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(here)
```

```
## here() starts at /Users/oscarodonnell/Desktop/PSTAT 197/module1
```

```
library(infer)
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.4.1 --
## v dials        1.4.2     v tailor       0.1.0
## v modeldata    1.5.1     v tune         2.0.1
## v parsnip      1.3.3     v workflows    1.3.0
## v recipes      1.3.1     v workflowsets 1.1.1
## v rsample      1.3.1     v yardstick    1.3.2
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x randomForest::combine() masks dplyr::combine()
## x scales::discard()       masks purrr::discard()
## x dplyr::filter()         masks stats::filter()
## x recipes::fixed()        masks stringr::fixed()
## x dplyr::lag()            masks stats::lag()
## x randomForest::margin()  masks ggplot2::margin()
## x yardstick::spec()       masks readr::spec()
## x recipes::step()         masks stats::step()
```

```
library(modelr)
```

```
##
## Attaching package: 'modelr'
##
## The following objects are masked from 'package:yardstick':
##
##     mae, mape, rmse
##
## The following object is masked from 'package:broom':
##
##     bootstrap
```

```
library(yardstick)
```

```
set.seed(10292025)
```

```
rdata_path <- here::here("data", "biomarker-clean.RData")
objs <- load(rdata_path)
```

```
biomarker_clean
```

```
## # A tibble: 154 x 1,319
##    group  ados    CHIP  CEBPB     NSE   PIAS4 `IL-10 Ra`   STAT3    IRF1 `c-Jun`
##    <chr> <dbl>   <dbl>  <dbl>   <dbl>   <dbl>      <dbl>   <dbl>   <dbl>   <dbl>
##  1 ASD       8  0.335   0.520 -0.554   0.650     -0.358   0.305  -0.484    0.309
##  2 ASD      21 -0.0715  1.01   3       1.28      -0.133   1.13    0.253    0.408
##  3 ASD      12 -0.406  -0.531 -0.0592  1.13       0.554  -0.334   0.287   -0.845
##  4 ASD      20 -0.102  -0.251  1.47    0.0773    -0.705   0.893   2.61    -0.372
##  5 ASD      22 -0.395  -0.536  0.0410 -0.299     -0.830   0.899   1.01    -0.843
##  6 ASD      17 -0.126   1.27  -0.892   0.239     -0.344   0.216   0.211    0.221
##  7 ASD      15  0.486   0.748 -1.09    0.462      0.570  -0.0682  1.01     1.21
##  8 ASD      10 -0.990  -1.10   0.231  -0.885     -0.151   0.0307 -0.0346  -0.891
##  9 ASD      22 -0.108   3      2.32    3          2.76    1.70    0.209    3
## 10 ASD      17  0.485  -0.234 -0.697  -0.286      0.0331  1.01   -0.248   -0.293
## # i 144 more rows
## # i 1,309 more variables: `Mcl-1` <dbl>, OAS1 <dbl>, `c-Myc` <dbl>,
## #   SMAD3 <dbl>, SMAD2 <dbl>, `IL-23` <dbl>, PDGFRA <dbl>, `IL-12` <dbl>,
```

```
## #   STAT1 <dbl>, STAT6 <dbl>, LRRK2 <dbl>, Osteocalcin <dbl>, `IL-5` <dbl>,
## #   GPDA <dbl>, IgA <dbl>, LPPL <dbl>, HEMK2 <dbl>, PDXK <dbl>, TLR4 <dbl>,
## #   REG4 <dbl>, `HSP 27` <dbl>, `YKL-40` <dbl>, `Alpha enolase` <dbl>,
## #   `Apo L1` <dbl>, CD38 <dbl>, CD59 <dbl>, FABPL <dbl>, `GDF-11` <dbl>, ...
```

Make sure group is factorized

```
bio <- biomarker_clean
```

3. Experiment with the following modifications:

- repeat the analysis but carry out the entire selection procedure on a training partition -- in othe

```
idx_td  <- which(bio$group == "TD")
idx_asd <- which(bio$group == "ASD")

split_idx <- function(ix, prop = 0.7) sample(ix, floor(length(ix)*prop))
tr_idx <- c(split_idx(idx_td), split_idx(idx_asd))
te_idx <- setdiff(seq_len(nrow(bio)), tr_idx)

bio_tr <- bio[tr_idx, , drop = FALSE]
bio_te <- bio[te_idx, , drop = FALSE]

table(Train = bio_tr$group)
```

```
## Train
## ASD  TD
##  53  54
```

```
table(Test  = bio_te$group)
```

```
## Test
## ASD  TD
##  23  24
```

- choose a larger number (more than ten) of top predictive proteins using each selection method

```
## MULTIPLE TESTING
####################

# function to compute tests
test_fn <- function(.df){
  t_test(.df,
         formula = level ~ group,
         order = c('ASD', 'TD'),
         alternative = 'two-sided',
         var.equal = F)
}

ttests_out <- biomarker_clean %>%
  # drop ADOS score
  select(-ados) %>%
  # arrange in long format
  pivot_longer(-group,
               names_to = 'protein',
               values_to = 'level') %>%
  # nest by protein
```

```
  nest(data = c(level, group)) %>%
  # compute t tests
  mutate(ttest = map(data, test_fn)) %>%
  unnest(ttest) %>%
  # sort by p-value
  arrange(p_value) %>%
  # multiple testing correction
  mutate(m = n(),
         hm = log(m) + 1/(2*m) - digamma(1),
         rank = row_number(),
         p.adj = m*hm*p_value/rank)

# select significant proteins
proteins_s1 <- ttests_out %>%
  slice_min(p.adj, n = 10) %>%
  pull(protein)

proteins_s1_large <- ttests_out %>%
  slice_min(p.adj, n = 15) %>%
  pull(protein)
```

```
predictors <- biomarker_clean %>%
  select(-c(group, ados))

response <- biomarker_clean %>% pull(group) %>% factor()

# fit RF
set.seed(101422)
rf_out <- randomForest(x = predictors,
                       y = response,
                       ntree = 1000,
                       importance = T)

# check errors
rf_out$confusion
```

```
##     ASD TD class.error
## ASD  48 28   0.3684211
## TD   17 61   0.2179487
```

```
# compute importance scores
proteins_s2 <- rf_out$importance %>%
  as_tibble() %>%
  mutate(protein = rownames(rf_out$importance)) %>%
  slice_max(MeanDecreaseGini, n = 10) %>%
  pull(protein)

proteins_s2_large <- rf_out$importance %>%
  as_tibble() %>%
  mutate(protein = rownames(rf_out$importance)) %>%
  slice_max(MeanDecreaseGini, n = 15) %>%
  pull(protein)
```

```
## LOGISTIC REGRESSION
########################
```

```r
# select subset of interest
proteins_sstar <- intersect(proteins_s1, proteins_s2)

proteins_sstar_large <- intersect(proteins_s1_large, proteins_s2_large)

biomarker_sstar <- biomarker_clean %>%
  select(group, any_of(proteins_sstar)) %>%
  mutate(class = factor(if_else(group == "ASD", "ASD", "TD"),
                        levels = c("TD","ASD"))) %>% #factorized group
  select(-group)

biomarker_sstar_large <- biomarker_clean %>%
  select(group, any_of(proteins_sstar_large)) %>%
  mutate(class = factor(if_else(group == "ASD", "ASD", "TD"),
                        levels = c("TD","ASD"))) %>% #factorized group
  select(-group)

# partition into training and test set
set.seed(101422)
biomarker_split <- biomarker_sstar %>%
  initial_split(prop = 0.8)

biomarker_split_large <- biomarker_sstar_large %>%
  initial_split(prop = 0.8)

#added formula to backtick column names because of column name errors when getting results
mk_formula <- function(df, resp = "class") {
  preds <- setdiff(names(df), resp)
  as.formula(paste(resp, "~", paste(sprintf("`%s`", preds), collapse = " + ")))
}

# fit logistic regression model to training set
fit <- glm(
  formula = mk_formula(training(biomarker_split), resp = "class"),
  data = training(biomarker_split),
  family = 'binomial')

fit_large <- glm(
  formula = mk_formula(training(biomarker_split_large), resp = "class"),
  data = training(biomarker_split_large),
  family = 'binomial')

# evaluate errors on test set
class_metrics <- metric_set(sensitivity,
                            specificity,
                            accuracy,
                            roc_auc)

normal_results <- testing(biomarker_split) %>%
  add_predictions(fit, type = "response") %>%
  mutate(
    pred_prob  = pred,
    pred_class = factor(if_else(pred_prob > 0.5, "ASD", "TD"),
```

```
                        levels = c("TD","ASD"))
  ) # adjusted to allow for factorized class

large_n_results <- testing(biomarker_split_large) %>%
  add_predictions(fit_large, type = 'response') %>%
  mutate(
    pred_prob  = pred,
    pred_class = factor(if_else(pred_prob > 0.5, "ASD", "TD"),
                        levels = c("TD","ASD"))
  ) # adjusted to allow for factorized class
```

```
normal_results
```

```
## # A tibble: 31 x 8
##       DERM    RELT    IgD   FSTL1 class  pred pred_prob pred_class
##      <dbl>   <dbl>  <dbl>   <dbl> <fct> <dbl>     <dbl> <fct>
##  1 -1.90   -1.18   -1.29   1.16   ASD   0.893     0.893 ASD
##  2  0.0607 -1.06   -1.13  -0.133  ASD   0.717     0.717 ASD
##  3  0.140  -0.586  -1.53  -1.96   ASD   0.822     0.822 ASD
##  4  0.174  -0.410  -1.42  -0.727  ASD   0.734     0.734 ASD
##  5  1.56    0.0406  0.697 -1.55   ASD   0.225     0.225 TD
##  6 -0.738  -0.835  -1.56  -1.78   ASD   0.902     0.902 ASD
##  7  0.437  -0.122  -0.726 -0.558  ASD   0.561     0.561 ASD
##  8  0.740   0.0281 -1.31   0.174  ASD   0.532     0.532 ASD
##  9  0.0195 -1.22   -0.247  1.45   ASD   0.493     0.493 TD
## 10  0.300   0.707  -0.911  0.0363 ASD   0.511     0.511 ASD
## # i 21 more rows
```

```
large_n_results
```

```
## # A tibble: 31 x 11
##        DERM    RELT Calcineurin    IgD  FSTL1   MAPK2 `TGF-b R III` class  pred
##       <dbl>   <dbl>       <dbl>  <dbl>  <dbl>   <dbl>         <dbl> <fct> <dbl>
##  1 -0.726    0.136        0.106 -1.33  -0.891  0.131        0.00555 ASD   0.786
##  2  0.740    0.0281      -0.160 -1.31   0.174 -0.172        0.675   ASD   0.522
##  3  0.513    1.01        -0.387 -1.19  -0.112 -0.383        0.988   ASD   0.568
##  4 -1.90     0.218       -1.45  -1.15  -1.50   0.685       -1.36    ASD   0.921
##  5 -0.276    0.410       -0.660  0.454 -1.41  -0.881        0.149   ASD   0.679
##  6  0.927   -1.12        -1.20   1.15   0.190 -0.653        0.0694  ASD   0.300
##  7 -0.00743 -0.319       -0.214 -0.762 -1.05   0.0194       0.766   ASD   0.599
##  8 -1.62    -1.49        -0.539 -1.04  -0.501 -1.02        -1.64    ASD   0.963
##  9 -0.596   -0.813        0.229 -0.951 -1.04   0.861        0.0398  ASD   0.620
## 10 -0.484   -0.612        0.188 -0.625 -1.16  -0.751       -0.763   ASD   0.838
## # i 21 more rows
## # i 2 more variables: pred_prob <dbl>, pred_class <fct>
```

Selecting the top 15 predictive proteins using each selection method has a moderate effect on the final selection results. The original process selected 4 proteins, while the larger (n=15) process yielded 7. The additional proteins that were included are Calcineurin, MAPK2, and TGF-b R III.

- `use a fuzzy intersection instead of a hard intersection to combine the sets of top predictive protei`

To create a fuzzy intersection, we need to include proteins that match show up in at least two of the selection methods. To do this, we must first individually find the top proteins most strongly correlated with ADOS severity.

```r
asd_df <- biomarker_clean %>%
  filter(group == "ASD") %>%
  select(group, ados, everything())

prot_cols <- setdiff(names(asd_df), c("group","ados"))

cors <- map_dbl(prot_cols, ~ suppressWarnings(
  cor(asd_df[[.x]], asd_df$ados, use = "pairwise.complete.obs")
))

proteins_s3 <- tibble(protein = prot_cols, score = abs(cors)) %>%
  arrange(desc(score)) %>%
  slice_head(n = 10) %>%
  pull(protein)

proteins_s3
```

```
##  [1] "CO8A1"            "C5b, 6 Complex"   "ILT-2"            "GM-CSF"
##  [5] "Thrombospondin-1" "Angiogenin"       "HCE004331"        "PDGF Rb"
##  [9] "C5a"              "IL-1F8"
```

Now that we have our top proteins, let's find the "fuzzy" intersection.

```r
proteins_fuzzy <- union(
  union(intersect(proteins_s1, proteins_s2), intersect(proteins_s1, proteins_s3)),
  intersect(proteins_s2, proteins_s3)
)

proteins_fuzzy
```

```
## [1] "DERM"  "RELT"  "IgD"   "FSTL1"
```

Interestingly enough, our results are the same as with the hard selection. This is because our proteins_s3 variable is independent of the two pre-processing steps, so it selected 10 proteins that weren't present in the other two steps, thus making no difference, and leaving only the proteins that were common between proteins_s1 and proteins_s2.