ADNI Analysis: Multinomial Models

Justin M. Leach

1 Introduction

This file contains analyses for ADNI data using multinomial models. The data analyses utilizes the `R` package `ssnet` version 0.2.0, which is available at https://github.com/jmleach-bst/ssnet. The version used is 0.0.0.9000; the primary dependencies are `BhGLM` (version 1.1.0) and `rstan` (version 2.19.2). You may also need `sim2Dpredictr` (version 0.1.0), which is available at https://github.com/jmleach-bst/sim2Dpredictr.

You need to load the following packages:

```
library(tidyverse)
library(sim2Dpredictr)
library(BhGLM)
library(rstan)
rstan_options(auto_write = TRUE)
library(ssnet)
```

Here we load the data and remove observations with missing measures.

```
# tau data
tau_bl <- readRDS("C:/Users/cotto/Documents/Publications/multinomial models in R/ADNI-analyses/data/tau_

# obtain necessary data
rm.unknown <- c("CTX_LH_UNKNOWN_SUVR", "CTX_RH_UNKNOWN_SUVR")
tau_bl <- tau_bl[ , !(names(tau_bl) %in% rm.unknown)]

x_tau_bl <- 0.2 * scale(as.matrix(
  tau_bl[, grep("CTX.*",
                names(tau_bl),
                perl = TRUE)]))

y_tau_bl <- factor(tau_bl$DIAGNOSIS)

# remove missing observations
for (i in 1:nrow(x_tau_bl)) {
  if (any(is.na(x_tau_bl[i,]))) {
    cat("Observation(s) ", i, "has missing x measures. \n")
  }
}
```

```
## Observation(s)  265 has missing x measures.
```

```r
cat("Observation(s) ", which(is.na(y_tau_bl)), " are missing y measures. \n")
```

```
## Observation(s)  358  are missing y measures.
```

```r
x_tau_bl <- x_tau_bl[-c(265, 358), ]
y_tau_bl <- y_tau_bl[-c(265, 358)]

# cortical thickness data
thick_sc <- readRDS("C:/Users/cotto/Documents/Publications/multinomial models in R/ADNI-analyses/data/t
x_thick_sc <- 0.2 * scale(as.matrix(thick_sc[, grep("ST.*", names(thick_sc), perl = TRUE)]))
y_thick_sc <- factor(thick_sc$DIAGNOSIS)

# remove missing observations (none for CT)
# for (i in 1:nrow(x_thick_sc)) {
#   if (any(is.na(x_thick_sc[i,]))) {
#     cat("Observation ", i, "has missing measures. \n")
#   }
# }
#
# which(is.na(y_thick_sc))
```

2 Summary

	CN_Freq	CN_Perc	MCI_Freq	MCI_Perc	Dementia_Freq	Dementia_Perc
Cortical Thickness	261	0.6084	127	0.2960	41	0.0956
Tau PET	234	0.6015	116	0.2982	39	0.1003

3 Analyses

3.1 I am not copying and pasting again. Functional programming.

Okay, so there's still some copying and pasting, mainly because I do not want to run all the models at once, nor am I prepared to try some kind of crazy parallel programming trick in this instance. Nevertheless, the following function will (should?) make things less confusing. While we reproduce the functions for fitting the data here, it may be easiest to toss the analyses on a cluster for reproducing the results (the models fit pretty quickly, but not immediately; it will take a few minutes, but not a few hours). You can find the required `R` code (which is essentially the same) in the folder "Rcode_analyses_cheaha" and the corresponding scripts to run the code on Cheaha in the folder "scripts_analyses_cheaha".

```r
fit_adni_mn <- function(x, y, alpha,
                        model, fold.seed,
                        s0 = seq(0.01, 0.2, 0.01),
                        s1 = seq(1, 5, 1),
                        nfolds = 5,
                        family = "multinomial",
                        type.multinomial = "grouped",
                        iar.data = NULL,
```

```
                      choose_lambda = FALSE # only for model = "glmnet"
                      ) {
if (model == "glmnet") {
  if (choose_lambda == TRUE) {
    cv1 <- cv.glmnet(
    x = x, y = y,
    alpha = alpha, nfolds = nfolds,
    family = family,
    type.multinomial = type.multinomial
    )

    cvs <- cv_ssnet(
    model = "glmnet",
    x = x, y = y,
    alpha = alpha, nfolds = nfolds,
    family = family,
    type.multinomial = type.multinomial,
    s0 = cv1$lambda,
    s1 = cv1$lambda,
    fold.seed = fold.seed)
  } else {
    cvs <- cv_ssnet(
    model = "glmnet",
    x = x, y = y,
    alpha = alpha, nfolds = nfolds,
    family = family,
    type.multinomial = type.multinomial,
    s0 = s0,
    s1 = s0,
    fold.seed = fold.seed)
  }

  cvs_gr <- cvs |>
    select(-ncv.id, -alpha, -model, -fold.id) |>
    group_by(s0) |>
    nest() |>
    mutate(
      mean = map(.x = data, .f = function(x) map_df(.x = x, .f = mean, na.rm = TRUE)),
      sd = map(.x = data, .f = function(x) map_df(.x = x, .f = sd, na.rm = TRUE))
  )
} else {
  if (model == "ss") {
    cvs <- cv_ssnet(
    model = model,
    x = x, y = y,
    alpha = alpha, nfolds = nfolds,
    family = family,
    type.multinomial = type.multinomial,
    s0 = s0,
    s1 = s1,
    fold.seed = fold.seed
  )
  }
```

```
    if (model == "ss_iar") {
      cvs <- cv_ssnet(
      model = model,
      x = x, y = y,
      alpha = alpha, nfolds = nfolds,
      family = family,
      type.multinomial = type.multinomial,
      s0 = s0,
      s1 = s1,
      fold.seed = fold.seed,
      iar.data = iar.data,
      iar.prior = TRUE,
      tau.prior = "none"
    )
    }

    cvs_gr <- cvs |>
      select(-ncv.id, -alpha, -model, -fold.id) |>
      group_by(s0, s1) |>
      nest() |>
      mutate(
        mean = map(.x = data, .f = function(x) map_df(.x = x, .f = mean, na.rm = TRUE)),
        sd = map(.x = data, .f = function(x) map_df(.x = x, .f = sd, na.rm = TRUE))
    )
  }

  cvs_gr_mean <- cvs_gr |>
    select(-data, -sd) |>
    unnest(mean)

  cvs_gr_sd <- cvs_gr |>
    select(-data, -mean) |>
    unnest(sd)

  cvs_gr_mean_opt <- cvs_gr_mean |>
    filter(deviance == min(cvs_gr_mean$deviance))

  out <- list(
    all_data = cvs_gr,
    all_means = cvs_gr_mean,
    all_sds = cvs_gr_sd,
    best_fit = cbind(model = model, alpha = alpha, cvs_gr_mean_opt)
  )
}
```

3.2 Cortical Thickness

```
fit_thick_lasso <- fit_adni_mn(
  model = "glmnet",
  x = x_thick_sc,
  y = y_thick_sc,
```

```
  alpha = 1,
  fold.seed = 7816631
)

saveRDS(fit_thick_lasso, "results/fit_mn_thick_lasso.rds")

fit_thick_en <- fit_adni_mn(
  model = "glmnet",
  x = x_thick_sc,
  y = y_thick_sc,
  alpha = 0.5,
  fold.seed = 7816631
)

saveRDS(fit_thick_en, "results/fit_mn_thick_en.rds")

fit_thick_ssl <- fit_adni_mn(
  model = "ss",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_thick_sc,
  y = y_thick_sc,
  alpha = 1,
  fold.seed = 7816631
)

saveRDS(fit_thick_ssl, "results/fit_mn_thick_ssl.rds")

fit_thick_ssen <- fit_adni_mn(
  model = "ss",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_thick_sc,
  y = y_thick_sc,
  alpha = 0.5,
  fold.seed = 7816631
)

saveRDS(fit_thick_ssen, "results/fit_mn_thick_ssen.rds")

dk_nb <- readRDS("data/dk_nb.rds")
fit_thick_ssliar <- fit_adni_mn(
  model = "ss_iar",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_thick_sc,
  y = y_thick_sc,
  alpha = 1,
  fold.seed = 7816631,
  iar.data = dk_nb
)

saveRDS(fit_thick_ssliar, "results/fit_mn_thick_ssliar.rds")
```

```
dk_nb <- readRDS("data/dk_nb.rds")
fit_thick_sseniar <- fit_adni_mn(
  model = "ss_iar",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_thick_sc,
  y = y_thick_sc,
  alpha = 0.5,
  fold.seed = 7816631,
  iar.data = dk_nb
)

saveRDS(fit_thick_sseniar, "results/fit_mn_thick_sseniar.rds")
```

3.3 Tau PET

```
fit_tau_lasso <- fit_adni_mn(
  model = "glmnet",
  x = x_tau_bl,
  y = y_tau_bl,
  alpha = 1,
  fold.seed = 7816631,
  s0 = seq(0.001, 0.2, 0.001)
)

saveRDS(fit_tau_lasso, "results/fit_mn_tau_lasso.rds")

set.seed(7816631)
fit_tau_en <- fit_adni_mn(
  model = "glmnet",
  x = x_tau_bl,
  y = y_tau_bl,
  alpha = 0.5,
  fold.seed = 7816631,
  s0 = seq(0.001, 0.2, 0.001)
)

saveRDS(fit_tau_en, "results/fit_mn_tau_en.rds")

fit_tau_ssl <- fit_adni_mn(
  model = "ss",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_tau_bl,
  y = y_tau_bl,
  alpha = 1,
  fold.seed = 7816631
)

saveRDS(fit_tau_ssl, "results/fit_mn_tau_ssl.rds")
```

```
fit_tau_ssen <- fit_adni_mn(
  model = "ss",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_tau_bl,
  y = y_tau_bl,
  alpha = 0.5,
  fold.seed = 7816631
)

saveRDS(fit_tau_ssen, "results/fit_mn_tau_ssen.rds")

dk_nb <- readRDS("data/dk_nb.rds")
fit_tau_ssliar <- fit_adni_mn(
  model = "ss_iar",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_tau_bl,
  y = y_tau_bl,
  alpha = 1,
  fold.seed = 7816631,
  iar.data = dk_nb
)

saveRDS(fit_tau_ssliar, "results/fit_mn_tau_ssliar.rds")

dk_nb <- readRDS("data/dk_nb.rds")
fit_tau_sseniar <- fit_adni_mn(
  model = "ss_iar",
  s0 = seq(0.01, 0.2, 0.01),
  s1 = seq(1, 10, 1),
  x = x_tau_bl,
  y = y_tau_bl,
  alpha = 0.5,
  fold.seed = 7816631,
  iar.data = dk_nb
)

saveRDS(fit_tau_sseniar, "results/fit_mn_tau_sseniar.rds")
```

4 Results

4.1 Metric Definitions

Here we focus on classification ability. Model evaluation is trickier for multi-class classification, because standard measures such as sensitivity, specificity, positive and negative predictive value, F_1, and Matthews Correlation Coefficient cannot be applied without modification. We apply several metrics described in **?**.

The average accuracy (AA) and error rate (ER) are the average per-class effectiveness and classification error, respectively:

$$\text{Average Accuracy} = \frac{1}{V}\sum_{v=1}^{V}\frac{tp_v + tn_v}{tp_v + tn_v + fp_v + fn_v} \tag{1}$$

$$\text{Error Rate} = \frac{1}{V}\sum_{v=1}^{V}\frac{fp_v + fn_v}{tp_v + tn_v + fp_v + fn_v} \tag{2}$$

where tp_v, tn_v, fp_v, and fn_v are true positive, true negative, false positive, and false negative for class v.

Positive predictive value (PPV; alternatively, precision in the machine learning literature), sensitivity (SN; alternatively recall in the machine learning literature), and F_1 score can be averaged in one of two ways: micro- or macro-averaged. Micro-averaged values are given subscript μ and are calculated using cumulative counts of true positives, false positives, true negatives, and false negatives.

$$\underset{\mu}{\text{PPV}} = \frac{\sum_{v=1}^{V} tp_v}{\sum_{v=1}^{V}(tp_v + fp_v)} \tag{3}$$

$$\underset{\mu}{\text{SN}} = \frac{\sum_{v=1}^{V} tp_v}{\sum_{v=1}^{V}(tp_v + fn_v)} \tag{4}$$

$$\underset{1,\mu}{\text{F}} = \frac{2 \times PPV_\mu \times SN_\mu}{PPV_\mu \times + SN_\mu} \tag{5}$$

Macro averaged-values are given subscript M and are obtained as the averages of the aforementioned values over each class.

$$\underset{M}{\text{PPV}} = \frac{1}{V}\sum_{v=1}^{V}\frac{tp_v}{tp_v + fp_v} \tag{6}$$

$$\underset{M}{\text{SN}} = \frac{1}{V}\sum_{v=1}^{V}\frac{tp_v}{tp_v + fn_v} \tag{7}$$

$$\underset{1,M}{\text{F}} = \frac{2 \times PPV_M \times SN_M}{PPV_M \times + SN_M} \tag{8}$$

Micro-averaging favors classes/categories with relatively larger proportions of subjects, while macro-averaging treats classes/categories equally. Note that micro-averaging will give the same values for PPV, F_1, and SN - it turns out this is not an error, e.g., see https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/ or perhaps "A review on multi-label learning algorithms'' (doi: 10.1109/TKDE.2013.39).

4.2 Cortical Thickness

```
fit_mn_thick_lasso <- readRDS("results/fit_mn_thick_lasso.rds")
fit_mn_thick_en <- readRDS("results/fit_mn_thick_en.rds")
fit_mn_thick_ssl <- readRDS("results/fit_mn_thick_ssl.rds")
fit_mn_thick_ssen <- readRDS("results/fit_mn_thick_ssen.rds")
fit_mn_thick_ssliar <- readRDS("results/fit_mn_thick_ssliar.rds")
fit_mn_thick_sseniar <- readRDS("results/fit_mn_thick_sseniar.rds")
```

```
fit_mn_thick <- rbind(
    fit_mn_thick_lasso$best_fit,
    fit_mn_thick_en$best_fit,
    fit_mn_thick_ssl$best_fit,
    fit_mn_thick_ssen$best_fit,
    fit_mn_thick_ssliar$best_fit,
    fit_mn_thick_sseniar$best_fit
  ) |>
  select(-model, -alpha)

fit_mn_thick <- cbind(
  Model = c("Lasso",
            "EN",
            "SSL",
            "SSEN",
            "SSL-IAR",
            "SSEN-IAR"),
  fit_mn_thick
)

knitr::kable(
  fit_mn_thick,
  digits = 3,
  caption = "Model Fitness Comparison: Cortical Thickness",
  col.names = c("Model", "s0", "s1", "Dev.",
                "AC_avg", "PCE", "PPV_M", "SN_M", "F1_M",
                "PPV_mu", "SN_mu", "F1_mu")
)
```

Table 2: Model Fitness Comparison: Cortical Thickness

Model	s0	s1	Dev.	AC_avg	PCE	PPV_M	SN_M	F1_M	PPV_mu	SN_mu	F1_mu
Lasso	0.02	0.02	115.381	0.776	0.224	0.620	0.536	0.574	0.664	0.664	0.664
EN	0.04	0.04	114.574	0.781	0.219	0.632	0.539	0.581	0.671	0.671	0.671
SSL	0.09	2.00	119.652	0.777	0.223	0.640	0.489	0.551	0.665	0.665	0.665
SSEN	0.19	1.00	117.714	0.777	0.223	0.667	0.493	0.564	0.666	0.666	0.666
SSL-IAR	0.20	1.00	118.995	0.780	0.220	0.637	0.510	0.561	0.671	0.671	0.671
SSEN-IAR	0.20	2.00	116.544	0.781	0.219	0.612	0.513	0.556	0.671	0.671	0.671

```
# xtable::xtable(
#   fit_mn_thick |>
#     select(-ppv_micro, -sn_micro, -f1_micro),
#   digits = 4
# )
#
# xtable::xtable(
#   fit_mn_thick |>
#     select(-avg_acc, -pce, -ppv_macro, -sn_macro, -f1_macro),
#   digits = 4
# )
```

```
# xtable::xtable(
#   fit_mn_thick |>
#     select(-ppv_macro, -sn_macro, -f1_macro,
#            -ppv_micro, -sn_micro, -f1_micro),
#   digits = 4
# )
#
# xtable::xtable(
#   fit_mn_thick |>
#     select(-deviance, -avg_acc, -pce),
#   digits = 4
# )

# fit_mn_thick_t <- data.frame(
#   metrics = colnames(fit_mn_thick)[-1],
#   Lasso = t(fit_mn_thick[1, -1]),
#   EN = t(fit_mn_thick[2, -1]),
#   SSL = t(fit_mn_thick[3, -1]),
#   SSEN = t(fit_mn_thick[4, -1]),
#   SSLIAR = t(fit_mn_thick[5, -1]),
#   SSENIAR = t(fit_mn_thick[6, -1])
# )
# rownames(fit_mn_thick_t) <- NULL
# colnames(fit_mn_thick_t) <- c("Metrics", "Lasso", "EN", "SSL", "SSEN", "SSLIAR", "SSENIAR")
#
# knitr::kable(
#   fit_mn_thick_t,
#   digits = 4
# )

# xtable::xtable(
#   fit_mn_thick_t,
#   digits = 4
# )
```

By default, the most "optimal'' parameters are selected using deviance. However, we care more about say, macro F_1 score (F1_M), then perhaps we should see what happens if we choose parameter values with that score? We see below it doesn't make much difference. In addition, I think it is hard to justify which classification assessment to use, e.g., what's wrong with F1_mu or why not maximize SN_M? In contrast, deviance is calculated before classification and is probably more appropriate as it gives a more general assessment of model fitness before imposing any kind of classification rule.

```
# cvs_gr_mean_opt <- cvs_gr_mean |>
#     filter(deviance == min(cvs_gr_mean$deviance))

fit_mn_thick_f1 <- rbind(
  cbind(model = "Lasso", fit_mn_thick_lasso$all_means |>
    filter(f1_macro == max(fit_mn_thick_lasso$all_means$f1_macro, na.rm = TRUE))),
  cbind(model = "EN",fit_mn_thick_en$all_means |>
    filter(f1_macro == max(fit_mn_thick_en$all_means$f1_macro, na.rm = TRUE))),
  cbind(model = "SSL",fit_mn_thick_ssl$all_means |>
    filter(f1_macro == max(fit_mn_thick_ssl$all_means$f1_macro, na.rm = TRUE))),
  cbind(model = "SSEN",fit_mn_thick_ssen$all_means |>
```

```
    filter(f1_macro == max(fit_mn_thick_ssen$all_means$f1_macro, na.rm = TRUE))),
  cbind(model = "SSL-IAR",fit_mn_thick_ssliar$all_means |>
    filter(f1_macro == max(fit_mn_thick_ssliar$all_means$f1_macro, na.rm = TRUE))),
  cbind(model = "SSEN-IAR",fit_mn_thick_sseniar$all_means |>
    filter(f1_macro == max(fit_mn_thick_sseniar$all_means$f1_macro, na.rm = TRUE)))
)

# fit_mn_thick_f1 <- cbind(
#   Model = c("Lasso",
#                "EN",
#                "SSL",
#                "SSEN",
#                "SSL-IAR",
#                "SSEN-IAR"),
#   fit_mn_thick_f1
# )

knitr::kable(
  fit_mn_thick_f1,
  digits = 3,
  caption = "Model Fitness Comparison (Selected by F1_M): Cortical Thickness",
  col.names = c("Model", "s0", "s1", "Dev.",
                "AC_avg", "PCE", "PPV_M", "SN_M", "F1_M",
                "PPV_mu", "SN_mu", "F1_mu")
)
```

Table 3: Model Fitness Comparison (Selected by F1_M): Cortical Thickness

Model	s0	s1	Dev.	AC_avg	PCE	PPV_M	SN_M	F1_M	PPV_mu	SN_mu	F1_mu
Lasso	0.05	0.05	116.786	0.777	0.223	0.669	0.503	0.595	0.666	0.666	0.666
EN	0.04	0.04	114.574	0.781	0.219	0.632	0.539	0.581	0.671	0.671	0.671
SSL	0.17	9.00	119.852	0.780	0.220	0.653	0.499	0.562	0.670	0.670	0.670
SSL	0.17	10.00	119.855	0.780	0.220	0.653	0.499	0.562	0.670	0.670	0.670
SSEN	0.18	1.00	117.989	0.777	0.223	0.665	0.490	0.580	0.666	0.666	0.666
SSL-IAR	0.19	1.00	119.529	0.784	0.216	0.674	0.507	0.575	0.676	0.676	0.676
SSEN-IAR	0.14	7.00	119.638	0.780	0.220	0.678	0.506	0.576	0.671	0.671	0.671
SSEN-IAR	0.14	8.00	119.672	0.780	0.220	0.678	0.506	0.576	0.671	0.671	0.671
SSEN-IAR	0.14	9.00	119.701	0.780	0.220	0.678	0.506	0.576	0.671	0.671	0.671

4.3 Tau PET

```
fit_mn_tau_lasso <- readRDS("results/fit_mn_tau_lasso.rds")
fit_mn_tau_en <- readRDS("results/fit_mn_tau_en.rds")
fit_mn_tau_ssl <- readRDS("results/fit_mn_tau_ssl.rds")
fit_mn_tau_ssen <- readRDS("results/fit_mn_tau_ssen.rds")
fit_mn_tau_ssliar <- readRDS("results/fit_mn_tau_ssliar.rds")
fit_mn_tau_sseniar <- readRDS("results/fit_mn_tau_sseniar.rds")

fit_mn_tau <- rbind(
```

```
    fit_mn_tau_lasso$best_fit,
    fit_mn_tau_en$best_fit,
    fit_mn_tau_ssl$best_fit,
    fit_mn_tau_ssen$best_fit,
    fit_mn_tau_ssliar$best_fit,
    fit_mn_tau_sseniar$best_fit
  ) |>
  select(-model, -alpha)

fit_mn_tau <- cbind(
  Model = c("Lasso",
            "EN",
            "SSL",
            "SSEN",
            "SSL-IAR",
            "SSEN-IAR"),
  fit_mn_tau
)

knitr::kable(
  fit_mn_tau,
  digits = 3,
  caption = "Model Fitness Comparison: Tau PET",
  col.names = c("Model", "s0", "s1", "Dev.",
                "AC_avg", "PCE", "PPV_M", "SN_M", "F1_M",
                "PPV_mu", "SN_mu", "F1_mu")
)
```

Table 4: Model Fitness Comparison: Tau PET

Model	s0	s1	Dev.	AC_avg	PCE	PPV_M	SN_M	F1_M	PPV_mu	SN_mu	F1_mu
Lasso	0.019	0.019	131.383	0.767	0.233	0.533	0.448	0.486	0.650	0.650	0.650
EN	0.028	0.028	130.996	0.770	0.230	0.536	0.457	0.493	0.655	0.655	0.655
SSL	0.200	9.000	135.118	0.767	0.233	0.596	0.419	0.512	0.651	0.651	0.651
SSEN	0.180	1.000	133.007	0.776	0.224	0.609	0.452	0.516	0.665	0.665	0.665
SSL-IAR	0.070	1.000	136.004	0.767	0.233	0.589	0.418	0.497	0.651	0.651	0.651
SSEN-IAR	0.200	5.000	132.547	0.780	0.220	0.610	0.458	0.521	0.669	0.669	0.669

```
# xtable::xtable(
#   fit_mn_tau,
#   digits = 3
# )

# xtable::xtable(
#   fit_mn_tau |>
#     select(-ppv_micro, -sn_micro, -f1_micro),
#   digits = 4
# )
#
# xtable::xtable(
#   fit_mn_tau |>
#     select(-avg_acc, -pce, -ppv_macro, -sn_macro, -f1_macro),
```

```
#   digits = 4
# )

# xtable::xtable(
#   fit_mn_tau |>
#     select(-ppv_macro, -sn_macro, -f1_macro,
#            -ppv_micro, -sn_micro, -f1_micro),
#   digits = 4
# )
#
# xtable::xtable(
#   fit_mn_tau |>
#     select(-deviance, -avg_acc, -pce),
#   digits = 4
# )

# fit_mn_tau_t <- data.frame(
#   metrics = colnames(fit_mn_tau)[-1],
#   Lasso = t(fit_mn_tau[1, -1]),
#   EN = t(fit_mn_tau[2, -1]),
#   SSL = t(fit_mn_tau[3, -1]),
#   SSEN = t(fit_mn_tau[4, -1]),
#   SSLIAR = t(fit_mn_tau[5, -1]),
#   SSENIAR = t(fit_mn_tau[6, -1])
# )
# rownames(fit_mn_tau_t) <- NULL
# colnames(fit_mn_tau_t) <- c("Metrics", "Lasso", "EN", "SSL", "SSEN", "SSLIAR", "SSENIAR")
#
# knitr::kable(
#   fit_mn_tau_t,
#   digits = 4
# )

# xtable::xtable(
#   fit_mn_tau_t,
#   digits = 4
# )
```