Assignment 3 - Part 2 - Diagnosing Schizophrenia from Voice

Søren Orm Hansen

October XX, 2020

```r
# Loading packages
library(pacman)

p_load(lme4, tidyverse, caret, pROC, Metrics, tidymodels, kernlab, recipes, groupdata2, yardstick, knit

# Setting theme minimal
theme_set(theme_minimal())
```

Assignment 3 - Part 2 - Diagnosing schizophrenia from voice

In the previous part of the assignment you generated a bunch of "features", that is, of quantitative descriptors of voice in schizophrenia. We then looked at whether we could replicate results from the previous literature. We now want to know whether we can automatically diagnose schizophrenia from voice only, that is, relying on the set of features you produced last time, we will try to produce an automated classifier. Again, remember that the dataset containst 7 studies and 3 languages. Feel free to only include Danish (Study 1-4) if you feel that adds too much complexity.

Issues to be discussed your report: - Should you run the analysis on all languages/studies at the same time? - Choose your best acoustic feature from part 1. How well can you diagnose schizophrenia just using it? - Identify the best combination of acoustic features to diagnose schizophrenia using logistic regression. - Discuss the "classification" process: which methods are you using? Which confounds should you be aware of? What are the strength and limitation of the analysis? - Bonus question: Logistic regression is only one of many classification algorithms. Try using others and compare performance. Some examples: Discriminant Function, Random Forest, Support Vector Machine, etc. The package caret provides them. - Bonus Bonus question: It is possible combine the output of multiple classification models to improve classification accuracy. For inspiration see, https://machinelearningmastery.com/machine-learning-ensembles-with-r/ The interested reader might also want to look up 'The BigChaos Solution to the Netflix Grand Prize'

Learning objectives

- Learn the basics of classification in a machine learning framework
- Design, fit and report logistic regressions
- Apply feature selection techniques

Let's start

```r
# Load data
df <- read.csv("npause_fixed.csv")
```

```
# Fixing categories
df <- df %>%
  mutate(
    Diagnosis = as.factor(Diagnosis.y),
    Diagnosis.y = as.factor(Diagnosis.y),
    Diagnosis.x = as.factor(Diagnosis.x),
    pause_dur = as.numeric(pause_dur),
    Participant = as.factor(Participant),
    maybe = as.numeric(Participant), # The participant column is behaving odd later - 'maybe' was a sho
    ProportionSpokenTime = phonationtime..s./dur..s.
  )

variable.names(df)
```

```
##  [1] "X"                           "Diagnosis.x"
##  [3] "Diagnosis.y"                 "Participant"
##  [5] "SANS"                        "SAPS"
##  [7] "Gender"                      "Study.y"
##  [9] "Trial"                       "npause"
## [11] "dur..s."                     "phonationtime..s."
## [13] "IQR"                         "speechrate..nsyll.dur."
## [15] "Age"                         "Mean"
## [17] "pause_dur"                   "pause_dur_scaled"
## [19] "IQR_scaled"                  "SpeechRate_scaled"
## [21] "ProportionSpokenTime_scaled" "SAPS_scaled"
## [23] "Mean_scaled"                 "Diagnosis"
## [25] "maybe"                       "ProportionSpokenTime"
```

```
# df_par <- partition(df,p = 0.8, cat_col = "Diagnosis.y", id_col = "Participant", list_out = F)
#
# df_train <- subset(df_par, .partitions == 1)
# df_test <- subset(df_par, .partitions == 2)

# Is it stratified/balanced?
# df_train %>% group_by(Diagnosis.y) %>% count()
# df_test %>% group_by(Diagnosis.y) %>% count()
```

We first want to build a logistic regression to see whether you can diagnose schizophrenia from your best acoustic feature. Let's use the full dataset and calculate the different performance measures (accuracy, sensitivity, specificity, PPV, NPV, ROC curve). You need to think carefully as to how we should (or not) use study and subject ID.

Then cross-validate the logistic regression and re-calculate performance on the testing folds. N.B. The cross-validation functions you already have should be tweaked: you need to calculate these new performance measures. Alternatively, the groupdata2 and cvms package created by Ludvig are an easy solution.

N.B. the predict() function generates log odds (the full scale between minus and plus infinity). Log odds > 0 indicates a choice of 1, below a choice of 0. N.N.B. you need to decide whether calculate performance on each single test fold or save all the prediction for test folds in one datase, so to calculate overall performance. N.N.N.B. Now you have two levels of structure: subject and study. Should this impact your cross-validation? N.N.N.N.B. A more advanced solution could rely on the tidy-models set of packages (warning: Time-consuming to learn as the documentation is sparse, but totally worth it)

```r
# Creating logit to probability function
logit2prob <- function(logit) {
  odds <- exp(logit)
  probs <- odds / (1 + odds)
  return(probs)
}
```

```r
# Creating function for folding and for creating and saving values we are interested in
folding <- function(syntax, random) {
 k = 10

# Splitting the data frame in the two categories to ensure that we have both categories in all folds
  control <- df %>% filter(Diagnosis == "Control") %>%
    mutate(maybe = as.factor(maybe))
  schizophrenia <- df %>% filter(Diagnosis == "Schizophrenia") %>%
    mutate(maybe = as.factor(maybe))


# Creating folds for the two categories
  foldsCon <-
    createFolds(
      unique(control$maybe),
      k = k,
      list = TRUE,
      returnTrain = FALSE)
  foldsSchiz <-
    createFolds(
      unique(schizophrenia$maybe),
      k = k,
      list = TRUE,
      returnTrain = FALSE)



# Creating various place holders
  accuracy_train = rep(NA, k)
  accuracy_test = rep(NA, k)

  train_auc = rep(NA, k)
  test_auc = rep(NA, k)

  train_sens = rep(NA, k)
  test_sens = rep(NA, k)

  train_spec = rep(NA, k)
  test_spec = rep(NA, k)

  train_PPV = rep(NA, k)
  test_PPV = rep(NA, k)

  train_NPV = rep(NA, k)
  test_NPV = rep(NA, k)
  train_nir = rep(NA, k)
  test_nir = rep(NA, k)

  log_acc = rep(NA, k)
  log_kap = rep(NA, k)

  svm_acc = rep(NA, k)
  svm_kap = rep(NA, k)

# Looping through the different folds
  for (i in 1:k) {
    i <- as.numeric(i)
# Creating test and train subset for the i'th fold. We add 88 because that's how many unique maybes the
    df_train <- rbind(subset(control,!(maybe %in% foldsCon[[i]])),
                     subset(schizophrenia,!(maybe %in% (foldsSchiz[[i]] + 88))))
    df_test <- rbind(subset(control, maybe %in% foldsCon[[i]]),
                    subset(schizophrenia, maybe %in% (foldsSchiz[[i]] + 88)))


    rec <- df_train %>%
      recipe(Diagnosis ~ ., data = df_train) #. chooses all predictors. WHAT IT DO?

    rec_steps <- rec %>%
      step_scale(all_numeric()) %>% # making standard dev of 1 + make numeric
      step_center(all_numeric()) # mean centering + make numeric

    prepped_recipe <- prep(rec_steps, training = df_train, retain = T) #retain = T important

# get training set ready
    df_train_n <- juice(prepped_recipe) %>%
      select(-c(Study.y, Participant, X,ProportionSpokenTime_scaled, Mean_scaled, SAPS_scaled,SpeechRate

# BAKE NOW
    df_test_n <- bake(prepped_recipe, new_data = df_test, all_predictors()) %>%
      select(-c(Study.y, Participant, X,ProportionSpokenTime_scaled, Mean_scaled, SAPS_scaled,SpeechRate

    log_fit <-
      logistic_reg() %>%
      set_mode("classification") %>%
      set_engine("glm") %>%
      fit(Diagnosis ~ .,data = df_train_n)

   svm_fit <-
     svm_rbf() %>%
     set_mode("classification") %>%
     set_engine("kernlab") %>%
     fit(
       Diagnosis ~ SANS + Trial + phonationtime..s. + Age + SAPS + npause + IQR +
          Mean + Gender + dur..s. + speechrate..nsyll.dur. + pause_dur,
        data = df_train_n
     )

  # Performance - within sample vs out of sample error
```

```
    test_results <-
      df_test %>%
      as.tibble() %>%
      mutate(
        log_class = predict(log_fit, new_data = df_test_n) %>%
          pull(.pred_class),
        log_prob = predict(log_fit, new_data = df_test_n, type = "prob") %>%
         pull(.pred_Schizophrenia),
       svm_class = predict(svm_fit, new_data = df_test_n) %>%
         pull(.pred_class),
        svm_prob = predict(svm_fit, new_data = df_test_n, type = "prob") %>%
          pull(.pred_Schizophrenia)
       )
# adding diag.y
#test_results$Diagnosis <- as.factor(df_test$Diagnosis.y)

#df_test$Diagnosis.y <- as.factor(df_test$Diagnosis.y)
# log model
  log_out <- yardstick::metrics(test_results, truth = Diagnosis, estimate = log_class) %>%
   knitr::kable()
  log_acc[i] <- as.numeric(str_extract((log_out)[3], '\\d'))
  log_kap[i] <- as.numeric(str_extract((log_out)[4], '\\d'))

# svm model
  svm_out<- yardstick::metrics(test_results, truth = Diagnosis, estimate = svm_class) %>%
    knitr::kable()

  svm_acc[i] <- as.numeric(str_extract((svm_out)[3], '\\d'))
  svm_kap[i] <- as.numeric(str_extract((svm_out)[4], '\\d'))

# Building models on the training set. The function do either glm or glmer models
    df_train_n$Participant <- as.factor(df_train$Participant)
    df_test_n$Participant <- as.factor(df_test$Participant)

  if(random == F) {
      model <- glm(syntax, data = df_train_n, family = "binomial")
    } else {
      model <- glmer(syntax, data = df_train_n, family = "binomial")
    }

  # Creating predictions
    test <- df_test_n %>%
      mutate(prediction = logit2prob(predict(model, newdata = df_test_n, allow.new.levels = TRUE, re.fo
             pred = as.factor(ifelse(prediction > .5, "Schizophrenia", "Control")))
    train <- df_train_n %>%
      mutate(prediction = logit2prob(predict(model, re.form = NA)),
             pred = as.factor(ifelse(prediction > .5, "Schizophrenia", "Control")))

# Checking the quality of our predictions - if there is only one category, it will mess stuff up later
    qualiCheck <- ifelse(nlevels(test$pred) == 1, 0, 1)

# ROC for train
```

```
    train_rocCurve = roc(response = train$Diagnosis,
                        predictor = train$prediction)

# Using the quality check to avoid errors in ROC etc. for test
    test$Diagnosis <- df_test$Diagnosis

    if(qualiCheck == 1){
      test_rocCurve = roc(response = test$Diagnosis, predictor = test$prediction)
      test_auc[i] = as.numeric(pROC::auc(test_rocCurve))
    } else {
      test_auc[i] = 0
    }

    cm_test <- confusionMatrix (
      data = test$pred,
      reference = test$Diagnosis,
      positive = "Schizophrenia"
      )
    cm_train <- confusionMatrix (
      data = train$pred,
      reference = train$Diagnosis,
      positive = "Schizophrenia"
      )
    accuracy_train[i] <- cm_train[["overall"]][["Accuracy"]]
    accuracy_test[i]<- cm_test[["overall"]][["Accuracy"]]
# AUC for training set
    train_auc[i] = as.numeric(pROC::auc(train_rocCurve))

# Sensitivity for test and train
    train_sens[i] = cm_train[["byClass"]][["Sensitivity"]]
    test_sens[i] = cm_test[["byClass"]][["Sensitivity"]]

# Specificity for test and train
    train_spec[i] = cm_train[["byClass"]][["Specificity"]]
    test_spec[i] = cm_test[["byClass"]][["Specificity"]]

# PPV for test and train using
    train_PPV[i] = cm_train[["byClass"]][["Pos Pred Value"]]
    test_PPV[i] = cm_test[["byClass"]][["Pos Pred Value"]]

# NPV for test and train using
    train_NPV[i] = cm_train[["byClass"]][["Neg Pred Value"]]
    test_NPV[i] = cm_test[["byClass"]][["Neg Pred Value"]]

    train_nir[i] = cm_train[["overall"]][["AccuracyNull"]]
    test_nir[i] = cm_test[["overall"]][["AccuracyNull"]]
  }

# Finding the mean for the different values
  train_auc <- mean(train_auc)
  test_auc <- mean(test_auc)
  train_sens <- mean(train_sens)
  train_sens <- mean(train_sens)
```

```r
  test_sens <- mean(test_sens)
  train_spec <- mean(train_spec)
  test_spec <- mean(test_spec)
  train_PPV <- mean(train_PPV)
  test_PPV <- mean(test_PPV)
  train_NPV <- mean(train_NPV)
  test_NPV <- mean(test_NPV)
  accuracy_train <- mean(accuracy_train)
  accuracy_test <- mean(accuracy_test)
  train_nir <- mean(train_nir)
  test_nir <- mean(test_nir)
  log_acc <- mean(log_acc)
  log_kap <- mean(log_kap)
  svm_kap <- mean(svm_kap)
  svm_acc <- mean(svm_acc)

# Returning the values
  return(
    tibble(
      train_auc,
      test_auc,
      train_sens,
      test_sens,
      train_spec,
      test_spec,
      train_PPV,
      test_PPV,
      train_NPV,
      test_NPV,
      accuracy_train,
      accuracy_test,
      train_nir,
      test_nir,
      log_acc,
      log_kap,
      svm_kap,
      svm_acc
    )
  )
}
```

```r
# Syntax of the models we want to build
syntax <- c('Diagnosis ~ IQR',
            'Diagnosis ~ IQR + (1|Participant)',
            'Diagnosis ~ IQR + speechrate..nsyll.dur.',
            'Diagnosis ~ IQR + speechrate..nsyll.dur. + (1 | Participant)',
            'Diagnosis ~ IQR + speechrate..nsyll.dur. + ProportionSpokenTime',
            'Diagnosis ~ IQR + speechrate..nsyll.dur. + ProportionSpokenTime + (1 | Participant)',
            'Diagnosis ~ IQR + speechrate..nsyll.dur. + ProportionSpokenTime + pause_dur',
            'Diagnosis ~ IQR + speechrate..nsyll.dur. + ProportionSpokenTime + pause_dur + (1 | Partici
            )

# Specifying whether the models have random effects
```

```r
random <- c(F, T, F, T, F, T, F, T)

# Collecting in tibble
models <- tibble(syntax, random)

# M-applying the function to the models
output <- mapply(folding, models$syntax, models$random)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: `as.tibble()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls > cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0859724 (tol = 0.002, component 1)
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unide
##  - Rescale variables?

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0871932 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unide
##  - Rescale variables?

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0837681 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unide
##  - Rescale variables?

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.086612 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly uniden
##  - Rescale variables?

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0858267 (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unider
##  - Rescale variables?
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls > cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Setting levels: control = Control, case = Schizophrenia
```

```
## Setting direction: controls < cases
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases
## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls > cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Setting levels: control = Control, case = Schizophrenia

## Setting direction: controls < cases

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0863603 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unide
##  - Rescale variables?

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Setting levels: control = Control, case = Schizophrenia
## Setting direction: controls < cases

## Warning in confusionMatrix.default(data = test$pred, reference =
## test$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Warning in confusionMatrix.default(data = train$pred, reference =
## train$Diagnosis, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
# Correcting the model names
colnames(output) <- c('I_glm', 'I_glmer', 'IS_glm', 'IS_glmer', 'ISP_glm', 'ISP_glmer', 'ISPP_glm', 'IS

# Fixing the df's for tables
df_o <- as.data.frame(t(output))
df_o$test_dif_nir_acc <- as.numeric(df_o$accuracy_test) - as.numeric(df_o$test_nir)

# Fixing the df's for tables
df_o %>% summarise(
  log_acc = log_acc %>% as.numeric() %>% mean(),
  log_kap = log_kap %>% as.numeric() %>% mean(),
  svm_acc = svm_acc %>% as.numeric() %>% mean(),
  svm_kap = svm_kap %>% as.numeric() %>% mean()
)
```

```
##   log_acc log_kap svm_acc svm_kap
## 1  0.2125  0.2125  0.2875  0.2875
```