STAT 270 - Final - 02 Models

Jeff Kong, Noah Lee, Jacob Ventura

```
#| include: false
library(tidyverse)
library(tidymodels)
library(dplyr)
library(ggformula)
library(ggplot2)
library(GGally)
library(mosaic)
library(ranger)
library(vip)
library(caret)
library(ggfortify)
library(tibble)
library(poLCA)
library(kknn)
library(discrim)
library(klaR)
tidymodels_prefer(quiet = TRUE)
```

```
# Data preprocessing - copied from EDA section
df <- read.csv('conversion_predictors_of_clinically_isolated_syndrome_to_multiple_sclerosis.c
df$Gender <- as.factor(df$Gender)
df$Initial_Symptom <- as.factor(df$Initial_Symptom)
df$LLSSEP <- as.factor(df$LLSSEP)
df$ULSSEP <- as.factor(df$ULSSEP)
df$VEP <- as.factor(df$VEP)
df$BAEP <- as.factor(df$BAEP)
df$Periventricular_MRI <- as.factor(df$Periventricular_MRI)
df$Cortical_MRI <- as.factor(df$Cortical_MRI)
df$Infratentorial_MRI <- as.factor(df$Infratentorial_MRI)
df$Spinal_Cord_MRI <- as.factor(df$Spinal_Cord_MRI)
```

```
df$Initial_EDSS <- as.factor(df$Initial_EDSS)
df$Final_EDSS <- as.factor(df$Final_EDSS)
df$group <- as.factor(df$group)

df$X <- NULL
df$Initial_EDSS <- NULL
df$Final_EDSS <- NULL
df <- df |> filter(!if_any(Schooling, is.na))
df <- df |> filter(!if_any(Initial_Symptom, is.na))

replace_with_mode <- function(x) {
  mode_value <- names(sort(table(x), decreasing = TRUE))[1]
  x <- ifelse(x == 3, as.numeric(mode_value), x)
  return(x)
}

replace_with_mode2 <- function(x) {
  mode_value <- names(sort(table(x), decreasing = TRUE))[1]
  x <- ifelse(x == 2, as.numeric(mode_value), x)
  return(x)
}

df <- df |>
  mutate(
    Breastfeeding = replace_with_mode(Breastfeeding),
    Varicella = replace_with_mode(Varicella),
    Mono_or_Polysymptomatic = replace_with_mode(Mono_or_Polysymptomatic),
    Oligoclonal_Bands = replace_with_mode2(Oligoclonal_Bands)
  ) |>
  na.omit()

df$Breastfeeding <- as.factor(df$Breastfeeding)
df$Varicella <- as.factor(df$Varicella)
df$Mono_or_Polysymptomatic <- as.factor(df$Mono_or_Polysymptomatic)
df$Oligoclonal_Bands <- as.factor(df$Oligoclonal_Bands)
```

TRAIN, TEST, VALIDATION SPLIT - 70/15/15 SPLIT

```
set.seed(8080)
df_split <- df |> initial_split(prop = 0.7, strata = group)
df_train <- training(df_split)
df_testing <- testing(df_split)
```

```
df_testing_split <- df_testing |> initial_split(prop = 0.5, strata = group)
df_test <- testing(df_testing_split)
df_validation <- training(df_testing_split)
```

Logistic Regression:

```
set.seed(8080)

#initial model selection
model <- glm(group ~ Gender + Age + Schooling + Initial_Symptom + Mono_or_Polysymptomatic + C

summary(model)
```

```
# Breastfeeding, Varicella, LLSSEP, ULSSEP, VEP, Cortical - informed by PCA
logit_mod <- logistic_reg() |>
  set_mode("classification") |>
  set_engine("glm")

df_rec <- recipe(group ~ Gender + Age + Schooling + Initial_Symptom + Mono_or_Polysymptomatic
  step_naomit()

df_logit_wf <- workflow() |>
  add_recipe(df_rec) |>
  add_model(logit_mod)

df_fit <- df_logit_wf |> fit(data = df_train)

df_predictions <- augment(df_fit, new_data = df_validation)

conf_mat(df_predictions, truth = group, estimate = .pred_class)

accuracy(df_predictions, group, .pred_class)
```

78% Accuracy on the validation set.

KNN:

```
# Using grid-search 10 fold cross validation to find optimal k - params informed from PCA fac
conflicts_prefer(kknn::contr.dummy)
set.seed(8080)
```

```
knn_model <- nearest_neighbor(neighbors = tune()) |>
    set_engine("kknn") |>
    set_mode("classification")

my_rec <- recipe(group ~ Gender + Age + Schooling + Initial_Symptom + Mono_or_Polysymptomatic
  step_zv(all_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_dummy(all_nominal_predictors())

knn_wf <- workflow() |>
    add_recipe(my_rec) |>
    add_model(knn_model)

my_folds <- vfold_cv(df_train, v = 10)
k_grid <- grid_regular(neighbors(range = c(1, 21)), levels = 11)

cv_results <- tune_grid(object = knn_wf, resamples = my_folds, grid = k_grid)

collect_metrics(cv_results)
autoplot(cv_results,  metric = "accuracy")
```

Choosing 7 NN to reduce chance of overfitting.

Final KNN Model:

```
conflicts_prefer(kknn::contr.dummy) # packages were conflicting with step_dummy
set.seed(8080)

knn_model <- nearest_neighbor(neighbors = 7) |>
    set_engine("kknn") |>
    set_mode("classification")

my_rec <- recipe(group ~ Gender + Age + Schooling + Initial_Symptom + Mono_or_Polysymptomatic
  step_zv(all_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_dummy(all_nominal_predictors())

knn_wf <- workflow() |>
    add_recipe(my_rec) |>
    add_model(knn_model)

# Fit model to training set
```

```
knn_model <- fit(knn_wf, df_train)

knn_preds <- augment(knn_model, new_data = df_validation)
conf_mat(knn_preds, truth = group, estimate = .pred_class)
accuracy(knn_preds, group, .pred_class)
```

Bad accuracy - 68%

LDA:

```
lda_model <- discrim_linear() |>
  set_engine("MASS") |>
  set_mode("classification")

lda_rec <- recipe(group ~ ., data = df_train) |> # LDA and NB performed better on validation
  step_impute_median(all_numeric_predictors())

lda_wf <- workflow() |>
  add_model(lda_model) |>
  add_recipe(lda_rec)

lda_fit <- lda_wf |> fit(data = df_train)
```

```
# Confusion matrix
lda_pred <- lda_fit |> augment(new_data = df_validation)
conf_mat(lda_pred, truth = group, estimate = .pred_class)
```

```
# Model metrics
accuracy(lda_pred, truth = group, estimate = .pred_class)
yardstick::specificity(lda_pred, truth = group, estimate = .pred_class) # good specificity
yardstick::sensitivity(lda_pred, truth = group, estimate = .pred_class) # worse sensitivity
```

Naive Bayes:

```
nb_model <- naive_Bayes() |>
  set_engine("klaR") |>
  set_mode("classification")

nb_wf <- workflow() |>
  add_model(nb_model) |>
  add_recipe(lda_rec)
```

```
nb_fit <- nb_wf |> fit(data = df_train)
```

```
# Confusion matrix
nb_pred <- nb_fit |> augment(new_data = df_validation)
conf_mat(nb_pred, truth = group, estimate = .pred_class)
```

```
# Model metrics
accuracy(nb_pred, truth = group, estimate = .pred_class)
yardstick::specificity(nb_pred, truth = group, estimate = .pred_class)
yardstick::sensitivity(nb_pred, truth = group, estimate = .pred_class)
```

Better accuracy, specificity and sensitivity than LDA.

```
nb_details <- nb_fit$fit # Run later for interpretability of naive bayes
nb_details
```

Random Forest:

```
# Grid-search 10 fold cross validation to select number of trees and m
set.seed(8080)

ranger_recipe <- recipe(formula = group ~ ., data = df_train)|>
  step_dummy(all_nominal_predictors())

ranger_folds <- vfold_cv(df_train, v = 10, strata = group)

rf_spec_tune <- rand_forest(trees = tune(), mtry = tune()) |>
  set_mode("classification") |>
  set_engine("ranger", importance = "impurity")

ranger_workflow_tune <- workflow() |>
  add_recipe(ranger_recipe) |>
  add_model(rf_spec_tune)

ranger_grid <- grid_regular(
  trees(range = c(20, 200)),
  mtry(range = c(3, 5)), # sqrt(17) about 4
  levels = 5
)
```

```
ranger_tune <- tune_grid(
  ranger_workflow_tune,
  resamples = ranger_folds,
  grid = ranger_grid
)

best_cc <- select_by_one_std_err(
  ranger_tune,
  metric = "accuracy",
  trees,
  mtry
)

autoplot(ranger_tune,  metric = "accuracy")
best_cc
```

Moving forward with m=4 and tress=160.

Final RF model:

```
set.seed(8080)

rf_spec_best <- rand_forest(trees = 160, mtry = 4) |>
  set_mode("classification") |>
  set_engine("ranger", importance = "impurity")

ranger_recipe <- recipe(formula = group ~ ., data = df_train)|>
  step_dummy(all_nominal_predictors())

ranger_workflow <- workflow() |>
  add_recipe(ranger_recipe) |>
  add_model(rf_spec_best)

(rf_fit_best <- fit(ranger_workflow, data = df_train))
```

```
#RF Confusion matrix
rf_pred_best <- augment(rf_fit_best, new_data = df_validation)
conf_mat(rf_pred_best, truth = group, estimate = .pred_class)
```

```
# Model metrics
accuracy(rf_pred_best, truth = group, estimate = .pred_class)
```

```
yardstick::specificity(rf_pred_best, truth = group, estimate = .pred_class)
yardstick::sensitivity(rf_pred_best, truth = group, estimate = .pred_class)
```

metrics: worse than nb slightly

```
# Most important variables in determining CDMS in Random Forest
vip::vip(rf_fit_best)
```

We have chosen random forest - time to evaluate it on the test set. TEST SET:

```
rf_pred_best <- augment(rf_fit_best, new_data = df_test)
conf_mat(rf_pred_best, truth = group, estimate = .pred_class)
```

```
# Model metrics
accuracy(rf_pred_best, truth = group, estimate = .pred_class)
yardstick::specificity(rf_pred_best, truth = group, estimate = .pred_class)
yardstick::sensitivity(rf_pred_best, truth = group, estimate = .pred_class)
```