Final Model

Eduardo Yuki Yada

Imports

```
library(tidyverse)
library(yaml)
library(tidymodels)
library(usemodels)
library(vip)
```

Loading data

```
load('../dataset/processed_data.RData')
load('../dataset/processed_dictionary.RData')

columns_list <- yaml.load_file("./auxiliar/columns_list.yaml")

outcome_column <- params$outcome_column
features_list <- params$features_list
```

Filtering eligible pacients

```
df = df %>%
  filter(disch_outcomes_t0 == 0)

df %>% dim
```

```
## [1] 15766   239
```

Eligible features

```
eligible_columns = df_names %>%
  filter(momento.aquisicao == 'Admissão t0') %>%
  .$variable.name

exception_columns = c('death_intraop', 'death_intraop_1')

correlated_columns = c('year_procedure_1', # com year_adm_t0
                       'age_surgery_1', # com age
                       'admission_pre_t0_count', # com admission_t0
                       'atb', # com meds_antimicrobianos
                       'classe_meds_cardio_qtde', # com classe_meds_qtde
                       'suporte_hemod' # com proced_invasivos_qtde
                       )

eligible_features = eligible_columns %>%
  base::intersect(c(columns_list$categorical_columns, columns_list$numerical_columns)) %>%
  setdiff(c(exception_columns, correlated_columns))
```

```r
if (is.null(features_list)) {
  features = eligible_features
} else {
  features = base::intersect(eligible_features, features_list)
}

length(features)
```

```
## [1] 118
```

```r
print(features)
```

```
##   [1] "sex"                      "age"                      "race"
##   [4] "education_level"          "patient_state"            "underlying_heart_disease"
##   [7] "heart_disease"            "nyha_basal"               "hypertension"
##  [10] "prior_mi"                 "heart_failure"            "af"
##  [13] "cardiac_arrest"           "transplant"               "valvopathy"
##  [16] "endocardites"             "diabetes"                 "renal_failure"
##  [19] "hemodialysis"             "stroke"                   "copd"
##  [22] "cancer"                   "comorbidities_count"      "procedure_type_1"
##  [25] "reop_type_1"              "procedure_type_new"       "cied_final_1"
##  [28] "cied_final_group_1"       "admission_t0"             "admission_pre_t0_180d"
##  [31] "year_adm_t0"              "icu_t0"                   "dialysis_t0"
##  [34] "disch_outcomes_t0"        "n_procedure_t0"           "admission_t0_emergency"
##  [37] "aco"                      "antiarritmico"            "antihipertensivo"
##  [40] "betabloqueador"           "ieca_bra"                 "dva"
##  [43] "digoxina"                 "estatina"                 "diuretico"
##  [46] "vasodilatador"            "insuf_cardiaca"           "espironolactona"
##  [49] "bloq_calcio"              "trombolitico"             "antiplaquetario_vo"
##  [52] "antiplaquetario_ev"       "insulina"                 "hipoglicemiante"
##  [55] "hormonio_tireoidiano"     "broncodilatador"          "anticonvulsivante"
##  [58] "psicofarmacos"            "antifungico"              "antiviral"
##  [61] "antiretroviral"           "classe_meds_qtde"         "meds_cardiovasc_qtde"
##  [64] "meds_antimicrobianos"     "vni"                      "ventilacao_mecanica"
##  [67] "cec"                      "cir_cardiovascular"       "transplante_cardiaco"
##  [70] "cir_toracica"             "outros_proced_cirurgicos" "traqueostomia"
##  [73] "icp"                      "intervencao_cv"           "stent"
##  [76] "angioplastia"             "cateterismo"              "eletrofisiologia"
##  [79] "cateter_venoso_central"   "drenagem_torax"           "proced_invasivos_qtde"
##  [82] "cve_desf"                 "transfusao"               "interconsulta"
##  [85] "equipe_multiprof"         "ecg"                      "holter"
##  [88] "teste_esforco"            "espiro_ergoespiro"        "tilt_teste"
##  [91] "polissonografia"          "metodos_graficos_qtde"    "laboratorio"
##  [94] "cultura"                  "analises_clinicas_qtde"   "citologia"
##  [97] "biopsia"                  "histopatologia_qtde"      "angio_rm"
## [100] "angio_tc"                 "angiografia"              "aortografia"
## [103] "arteriografia"            "cavografia"               "cintilografia"
## [106] "ecocardiograma"           "endoscopia"               "flebografia"
## [109] "pet_ct"                   "ultrassom"                "tomografia"
## [112] "radiografia"              "ressonancia"              "exames_imagem_qtde"
## [115] "dieta_enteral"            "dieta_parenteral"         "bic"
## [118] "mpp"
```

Train test split (70%/30%)

```r
set.seed(42)

df[columns_list$outcome_columns] <- lapply(df[columns_list$outcome_columns], factor)
df <- mutate(df, across(where(is.character), as.factor))

df_split <- initial_split(df %>% dplyr::select(all_of(c(features, outcome_column))),
```

```
                          prop = .7, strata = all_of(outcome_column))
df_train <- training(df_split)
df_test <- testing(df_split)

dim(df_train)[1] / dim(df)[1]
```

```
## [1] 0.6999873
```

```
dim(df_test)[1] / dim(df)[1]
```

```
## [1] 0.3000127
```

Global parameters

```
k = 4 # Number of folds for cross validation
grid_size = 20 # Number of parameter combination to tune on each model

set.seed(234)
df_folds <- vfold_cv(df_train, v = k,
                     strata = all_of(outcome_column))
```

Functions

```
validation = function(model_fit, new_data) {
  library(pROC)
  library(caret)

  test_predictions_prob <-
    predict(model_fit, new_data = new_data, type = "prob") %>%
    rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
    .$`1`

  pROC_obj <- roc(
    new_data[[outcome_column]],
    test_predictions_prob,
    smoothed = TRUE,
    # arguments for ci
    ci = TRUE,
    ci.alpha = 0.9,
    stratified = FALSE,
    # arguments for plot
    plot = TRUE,
    auc.polygon = TRUE,
    max.auc.polygon = TRUE,
    grid = TRUE,
    print.auc = TRUE,
    show.thres = TRUE
  )

  sens.ci <- ci.se(pROC_obj)
  plot(sens.ci, type = "shape", col = "lightblue")
  plot(sens.ci, type = "bars")

  test_predictions_class <-
    predict(model_fit, new_data = new_data, type = "class") %>%
    rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
    .$class

  conf_matrix = table(test_predictions_class, new_data[[outcome_column]])
```

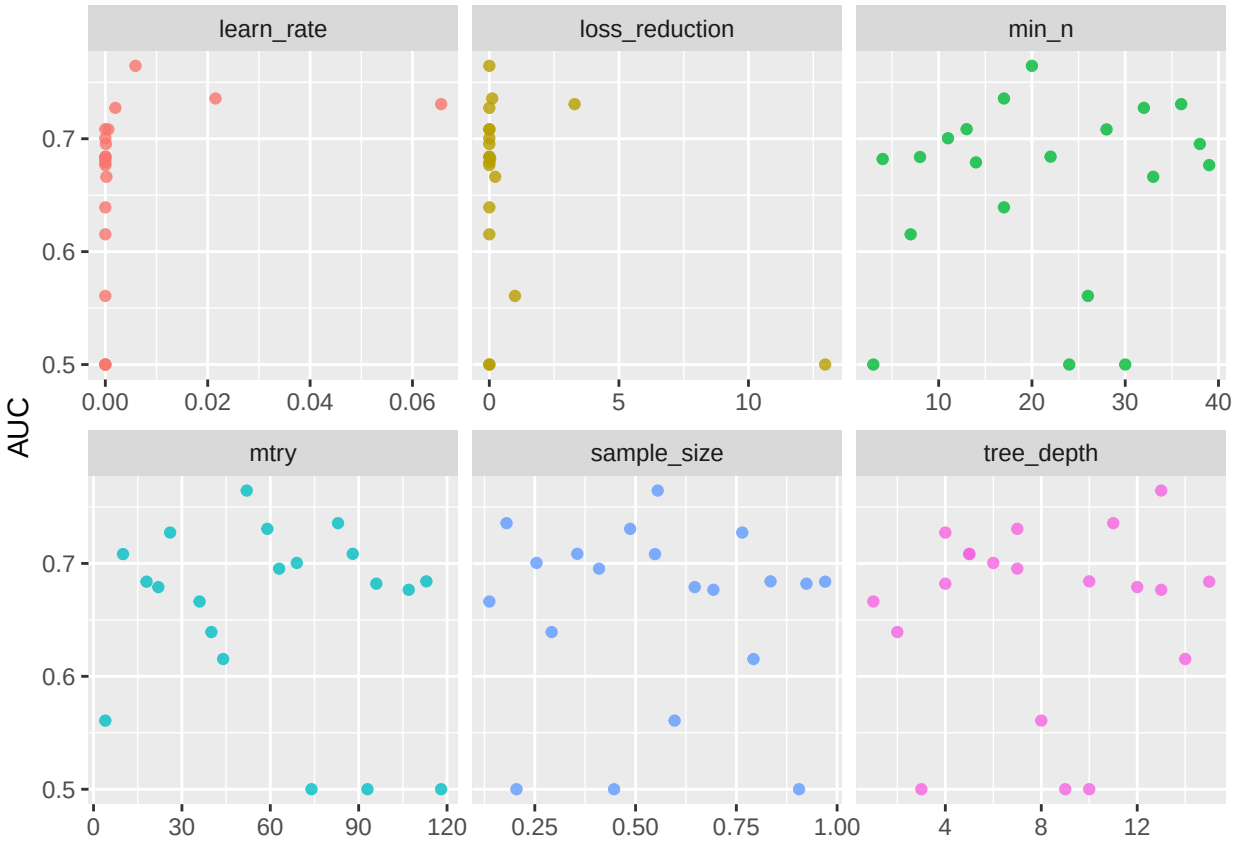
```
  confusionMatrix(conf_matrix) %>% print

  return(pROC_obj)
}
```

Boosted Tree (XGBoost)

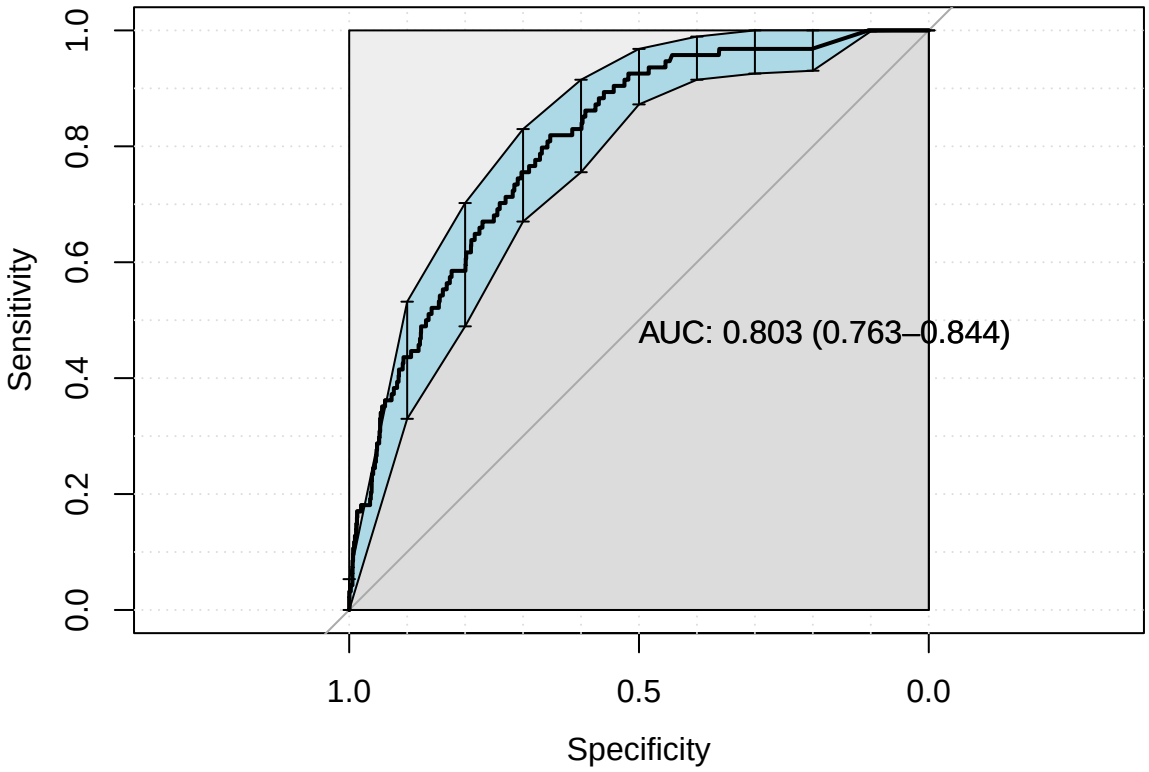
```
xgboost_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula, data = df_train) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other=".merged") %>%
  step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
  step_zv(all_predictors())

xgboost_spec <- boost_tree(
  trees = 200,
  tree_depth = tune(),
  min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  mtry = tune(),
  learn_rate = tune()
) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

xgboost_grid <- grid_latin_hypercube(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(), df_train),
  learn_rate(),
  size = grid_size
)

xgboost_workflow <-
  workflow() %>%
  add_recipe(xgboost_recipe) %>%
  add_model(xgboost_spec)

xgboost_tune <-
  xgboost_workflow %>%
  tune_grid(resamples = df_folds,
            grid = xgboost_grid)

xgboost_tune %>%
  show_best("roc_auc")
```

```
## # A tibble: 5 x 12
##    mtry min_n tree_depth learn_rate loss_~1 sampl~2 .metric .esti~3  mean     n std_err .config
##   <int> <int>      <int>      <dbl>   <dbl>   <dbl> <chr>   <chr>   <dbl> <int>   <dbl> <chr>
## 1    52    20         13 0.00591    1.76e-7   0.555 roc_auc binary  0.765     4  0.0108 Prepro~
## 2    83    17         11 0.0215     1.09e-1   0.180 roc_auc binary  0.736     4  0.0250 Prepro~
## 3    59    36          7 0.0656     3.29e+0   0.487 roc_auc binary  0.731     4  0.0335 Prepro~
## 4    26    32          4 0.00196    6.72e-8   0.765 roc_auc binary  0.727     4  0.0143 Prepro~
## 5    88    13          5 0.00000213 1.88e-8   0.355 roc_auc binary  0.709     4  0.0180 Prepro~
## # ... with abbreviated variable names 1: loss_reduction, 2: sample_size, 3: .estimator
```

```
best_xgboost <- xgboost_tune %>%
  select_best("roc_auc")

xgboost_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, mtry:sample_size) %>%
  pivot_longer(mtry:sample_size,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")
```

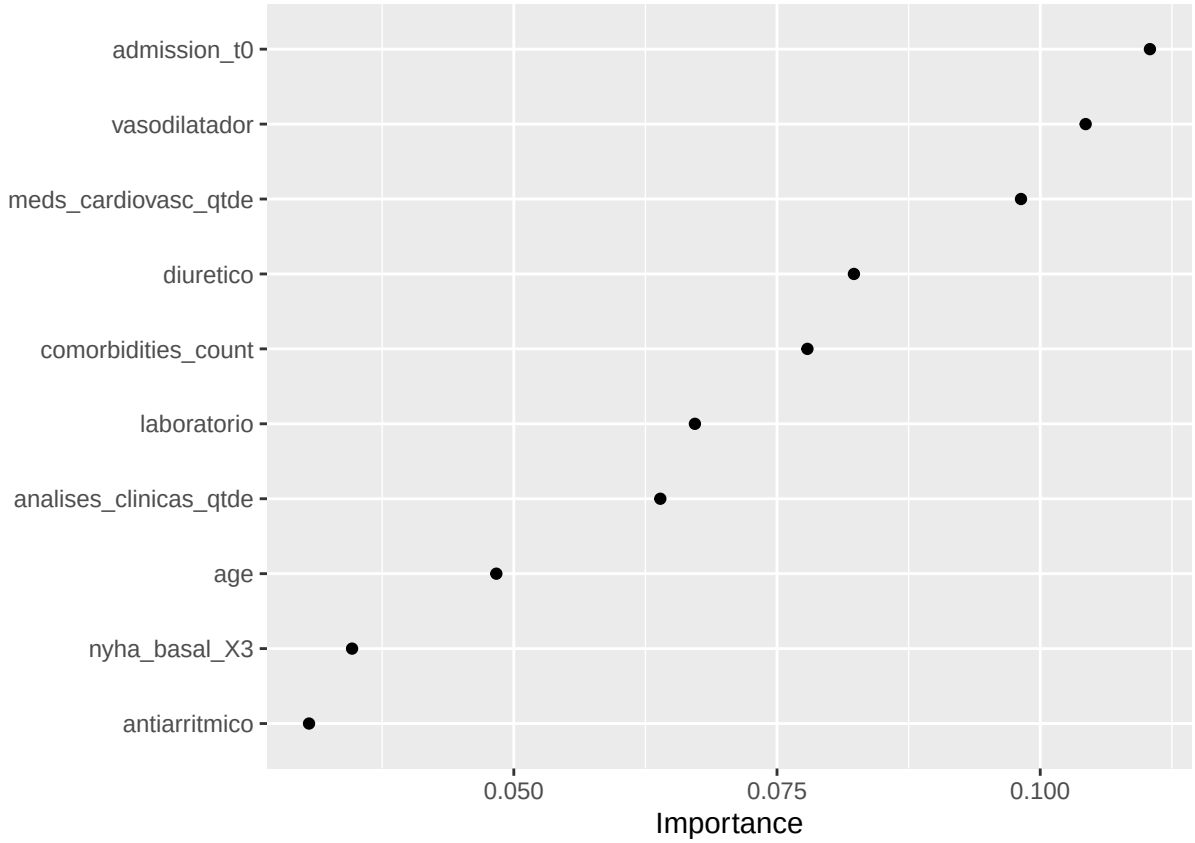



```
final_xgboost_workflow <-
  xgboost_workflow %>%
  finalize_workflow(best_xgboost)

last_xgboost_fit <-
  final_xgboost_workflow %>%
  last_fit(df_split)

final_xgboost_fit <- extract_workflow(last_xgboost_fit)

xgboost_auc = validation(final_xgboost_fit, df_test)
```

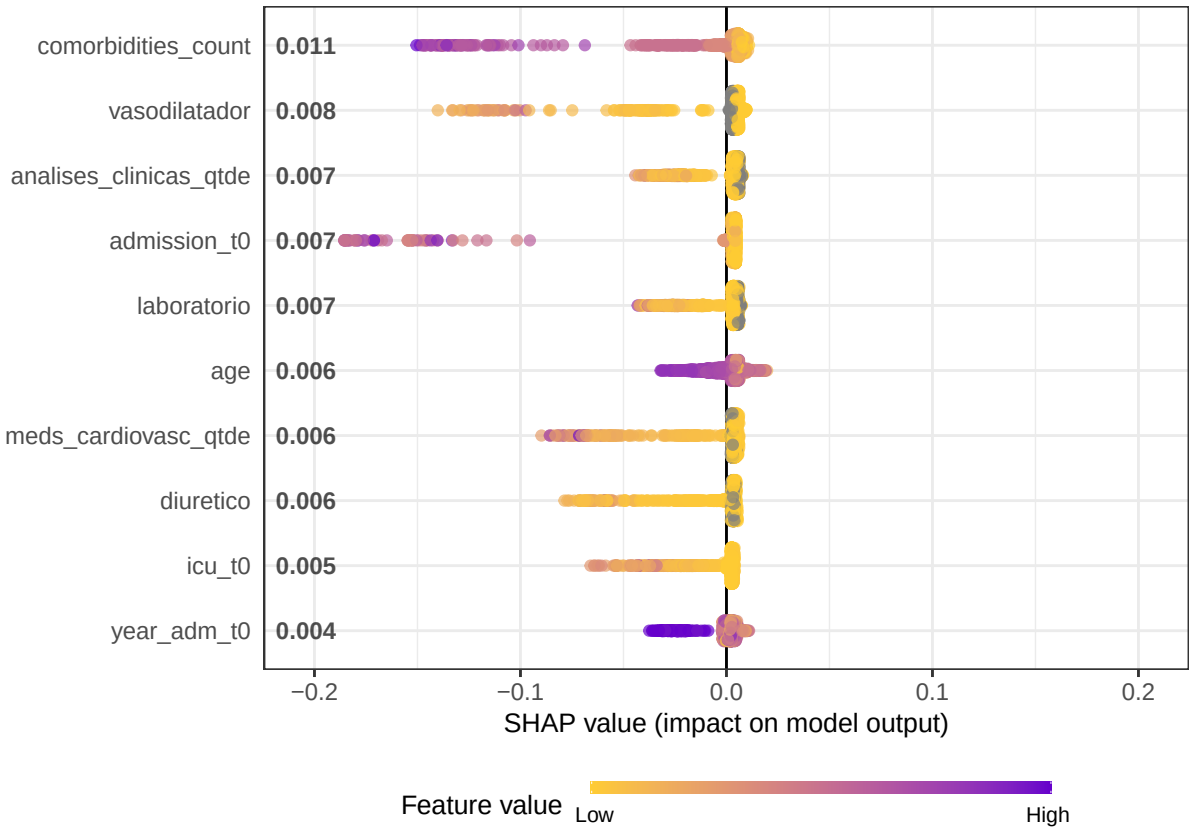



```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                       0 4636   94
##                       1    0    0
##
##                Accuracy : 0.9801
##                  95% CI : (0.9757, 0.9839)
##     No Information Rate : 0.9801
##     P-Value [Acc > NIR] : 0.5274
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9801
##          Neg Pred Value :    NaN
##              Prevalence : 0.9801
##          Detection Rate : 0.9801
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
final_xgboost_fit %>%
  fit(data = df_train) %>%
  extract_fit_parsnip() %>%
  vip(geom = "point")
```




```
library(SHAPforxgboost)
library(xgboost)
library(Matrix)
library(mltools)

xgb_model <- parsnip::extract_fit_engine(last_xgboost_fit)

trained_rec <- prep(xgboost_recipe, training = df_train)

df_test_baked <- bake(trained_rec, new_data = df_test)

matrix_test <- as.matrix(df_test_baked %>% select(-all_of(outcome_column)))

shap.plot.summary.wrap1(model = xgb_model, X = matrix_test, top_n = 10, dilute = T)
```


comorbidities_count
0.011
vasodilatador
0.008
analises_clinicas_qtde
0.007
admission_t0
0.007
laboratorio
0.007
age
0.006
meds_cardiovasc_qtde
0.006
diuretico
0.006
icu_t0
0.005
year_adm_t0
0.004
−0.2
−0.1
0.0
0.1
0.2
SHAP value (impact on model output)
Feature value
Low
High