Final Model

Eduardo Yuki Yada

Imports

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

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

Minutes to run: 0

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
```

Minutes to run: 0.001

Filtering eligible pacients

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

df %>% dim
```

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

Minutes to run: 0.005

Eligible features

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

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

```r
correlated_columns = c('year_procedure_1', # com year_adm_t0
                       'age_surgery_1', # com age
                       'admission_t0', # com admission_pre_t0_count
                       '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))

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

gluedown::md_order(features, seq = TRUE, pad = TRUE)
```

```
## 01. sex
## 02. age
## 03. education_level
## 04. underlying_heart_disease
## 05. heart_disease
## 06. nyha_basal
## 07. hypertension
## 08. prior_mi
## 09. heart_failure
## 10. af
## 11. cardiac_arrest
## 12. valvopathy
## 13. diabetes
## 14. renal_failure
## 15. hemodialysis
## 16. stroke
## 17. copd
## 18. cancer
## 19. comorbidities_count
## 20. procedure_type_1
## 21. reop_type_1
## 22. procedure_type_new
## 23. cied_final_1
## 24. cied_final_group_1
## 25. admission_pre_t0_count
## 26. admission_pre_t0_180d
## 27. year_adm_t0
## 28. icu_t0
## 29. dialysis_t0
## 30. disch_outcomes_t0
## 31. admission_t0_emergency
## 32. aco
## 33. antiarritmico
## 34. ieca_bra
## 35. dva
## 36. digoxina
## 37. estatina
## 38. diuretico
## 39. vasodilatador
## 40. insuf_cardiaca
## 41. espironolactona
```

```
## 42. antiplaquetario_ev
## 43. insulina
## 44. psicofarmacos
## 45. antifungico
## 46. classe_meds_qtde
## 47. meds_cardiovasc_qtde
## 48. meds_antimicrobianos
## 49. vni
## 50. outros_proced_cirurgicos
## 51. icp
## 52. cateterismo
## 53. cateter_venoso_central
## 54. proced_invasivos_qtde
## 55. transfusao
## 56. interconsulta
## 57. equipe_multiprof
## 58. ecg
## 59. holter
## 60. teste_esforco
## 61. metodos_graficos_qtde
## 62. laboratorio
## 63. cultura
## 64. analises_clinicas_qtde
## 65. citologia
## 66. histopatologia_qtde
## 67. angiografia
## 68. aortografia
## 69. arteriografia
## 70. cintilografia
## 71. ecocardiograma
## 72. ultrassom
## 73. tomografia
## 74. radiografia
## 75. ressonancia
## 76. exames_imagem_qtde
## 77. bic
```

Minutes to run: 0

Train test split (70%/30%)

```
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
```

Minutes to run: 0.003

Global parameters

```
k <- 4 # Number of folds for cross validation
grid_size <- 50 # 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))

max_auc_loss <- 0.01
```

Minutes to run: 0

Functions

```
validation = function(model_fit, new_data, plot=TRUE) {
  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,
    direction = "<",
    levels = c(0, 1),
    smoothed = TRUE,
    ci = TRUE,
    ci.alpha = 0.9,
    stratified = FALSE,
    plot = plot,
    auc.polygon = TRUE,
    max.auc.polygon = TRUE,
    grid = TRUE,
    print.auc = TRUE,
    show.thres = TRUE
  )

  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]])

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

    confusionMatrix(conf_matrix) %>% print
  }

  return(pROC_obj)
}
```

Minutes to run: 0

Feature Selection

```
model_fit_wf <- function(features, outcome_column, hyperparameters){
  model_recipe <-
    recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
           data = df_train %>% select(all_of(c(features, outcome_column)))) %>%
    step_novel(all_nominal_predictors()) %>%
    step_unknown(all_nominal_predictors()) %>%
    step_other(all_nominal_predictors(), threshold = 0.05, other=".merged") %>%
    step_impute_mean(all_numeric_predictors()) %>%
    step_zv(all_predictors())

  model_spec <-
    do.call(boost_tree, hyperparameters) %>%
    set_engine("lightgbm") %>%
    set_mode("classification")

  model_workflow <-
    workflow() %>%
    add_recipe(model_recipe) %>%
    add_model(model_spec)

  model_fit_rs <- model_workflow %>%
    fit_resamples(df_folds)

  model_fit <- model_workflow %>%
    fit(df_train)

  model_auc <- validation(model_fit, df_test, plot=F)

  raw_model <- parsnip::extract_fit_engine(model_fit)

  feature_importance <- lgb.importance(raw_model, percentage = TRUE)

  return(list(cv_auc = collect_metrics(model_fit_rs) %>% filter(.metric == 'roc_auc') %>% .$mean,
              importance = feature_importance,
              auc = as.numeric(model_auc$auc),
              auc_lower = model_auc$ci[1],
              auc_upper = model_auc$ci[3]))
}
```

Minutes to run: 0

```
hyperparameters <- readRDS(
  sprintf(
    "../EDA/auxiliar/hyperparameters/model_selection/lightgbm_parameters_%s.rds",
    outcome_column
  )
)

full_model <- model_fit_wf(features, outcome_column, hyperparameters)

sprintf('Full Model CV Train AUC: %.3f' ,full_model$cv_auc)
```

```
## [1] "Full Model CV Train AUC: 0.797"
```

```
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)
```

```
## [1] "Full Model Test AUC: 0.819"
```

Minutes to run: 0.123

Features with zero importance on the initial model:

```
unimportant_features <- setdiff(features, full_model$importance$Feature)

unimportant_features %>%
  gluedown::md_order()
```

```
## 1. disch_outcomes_t0
## 2. vni
```

Minutes to run: 0

```
trimmed_features <- full_model$importance$Feature
hyperparameters$mtry = min(hyperparameters$mtry, length(trimmed_features))
trimmed_model <- model_fit_wf(trimmed_features,
                              outcome_column, hyperparameters)

sprintf('Trimmed Model CV Train AUC: %.3f' ,trimmed_model$cv_auc)
```

```
## [1] "Trimmed Model CV Train AUC: 0.794"
```

```
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)
```

```
## [1] "Trimmed Model Test AUC: 0.813"
```

Minutes to run: 0.121

```
current_features <- trimmed_features
current_model <- trimmed_model
current_least_important <- tail(trimmed_model$importance$Feature, 1)
current_auc_loss <- full_model$cv_auc - trimmed_model$cv_auc

selection_results <- tibble::tribble(
  ~`Number of Features`, ~`AUC Loss`, ~`Least Important Feature`,
  length(features), 0, tail(full_model$importance$Feature, 1),
  length(trimmed_features), current_auc_loss, tail(trimmed_model$importance$Feature, 1)
)

while (current_auc_loss < max_auc_loss){
  last_feature_droped <- current_least_important

  current_features <- setdiff(current_features, current_least_important)
  hyperparameters$mtry = min(hyperparameters$mtry, length(current_features))
  current_model <- model_fit_wf(current_features, outcome_column, hyperparameters)
  current_least_important <- tail(current_model$importance$Feature, 1)

  current_auc_loss <- full_model$cv_auc - current_model$cv_auc

  selection_results <- selection_results %>%
    add_row(`Number of Features` = length(current_features),
            `AUC Loss` = current_auc_loss,
            `Least Important Feature` = current_least_important)

  print(c(length(current_features), current_auc_loss))
}
```

```
## [1] 74.000000000  0.005325269
## [1] 73.000000000  0.002390804
## [1] 72.000000000  0.008393768
## [1] 71.000000000  0.003818706
## [1] 70.000000000  0.003077512
## [1] 69.000000000  0.005684823
## [1] 68.00000000  0.00884357
## [1] 67.000000000  0.005456663
## [1] 66.000000000  0.007096342
## [1] 65.000000000  0.004619324
```

```
## [1] 64.000000000  0.009085761
## [1] 63.000000000  0.004759424
## [1] 62.000000000  0.006370864
## [1] 61.0000000000 -0.0009050901
## [1] 60.000000000  0.008295779
## [1] 59.000000000  0.001240467
## [1] 58.000000000  0.004330314
## [1] 5.700000e+01 1.926912e-04
## [1] 56.000000000  0.002910817
## [1] 55.000000000  0.006196997
## [1] 54.00000000  0.00378463
## [1] 5.300000e+01 7.340023e-04
## [1] 52.000000000  0.006483005
## [1] 51.000000000  0.004408446
## [1] 50.000000000  0.003502054
## [1] 49.0000000000 -0.0008204999
## [1] 48.000000000  0.004843395
## [1] 47.000000000  0.002439928
## [1] 4.600000e+01 4.950577e-04
## [1] 45.000000000 -0.002585943
## [1] 44.00000000  0.00518865
## [1] 43.000000000  0.003712589
## [1] 42.000000000  0.001739978
## [1] 4.100000e+01 8.945304e-04
## [1] 40.000000000  0.002959902
## [1] 39.000000000  0.002277934
## [1] 38.000000000  0.002776259
## [1] 37.000000000  0.007813333
## [1] 36.00000000  0.01008627
```

```
selection_results
```

```
## # A tibble: 41 x 3
##    `Number of Features` `AUC Loss` `Least Important Feature`
##                   <int>      <dbl> <chr>
##  1                   77    0       teste_esforco
##  2                   75    0.00254 teste_esforco
##  3                   74    0.00533 aortografia
##  4                   73    0.00239 arteriografia
##  5                   72    0.00839 heart_disease
##  6                   71    0.00382 transfusao
##  7                   70    0.00308 histopatologia_qtde
##  8                   69    0.00568 hemodialysis
##  9                   68    0.00884 cintilografia
## 10                   67    0.00546 cateter_venoso_central
## # ... with 31 more rows
```

Minutes to run: 4.246

```
selected_features <- c(current_features, last_feature_droped)

feature_selected_model <- model_fit_wf(selected_features,
                                       outcome_column, hyperparameters)

sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)
```

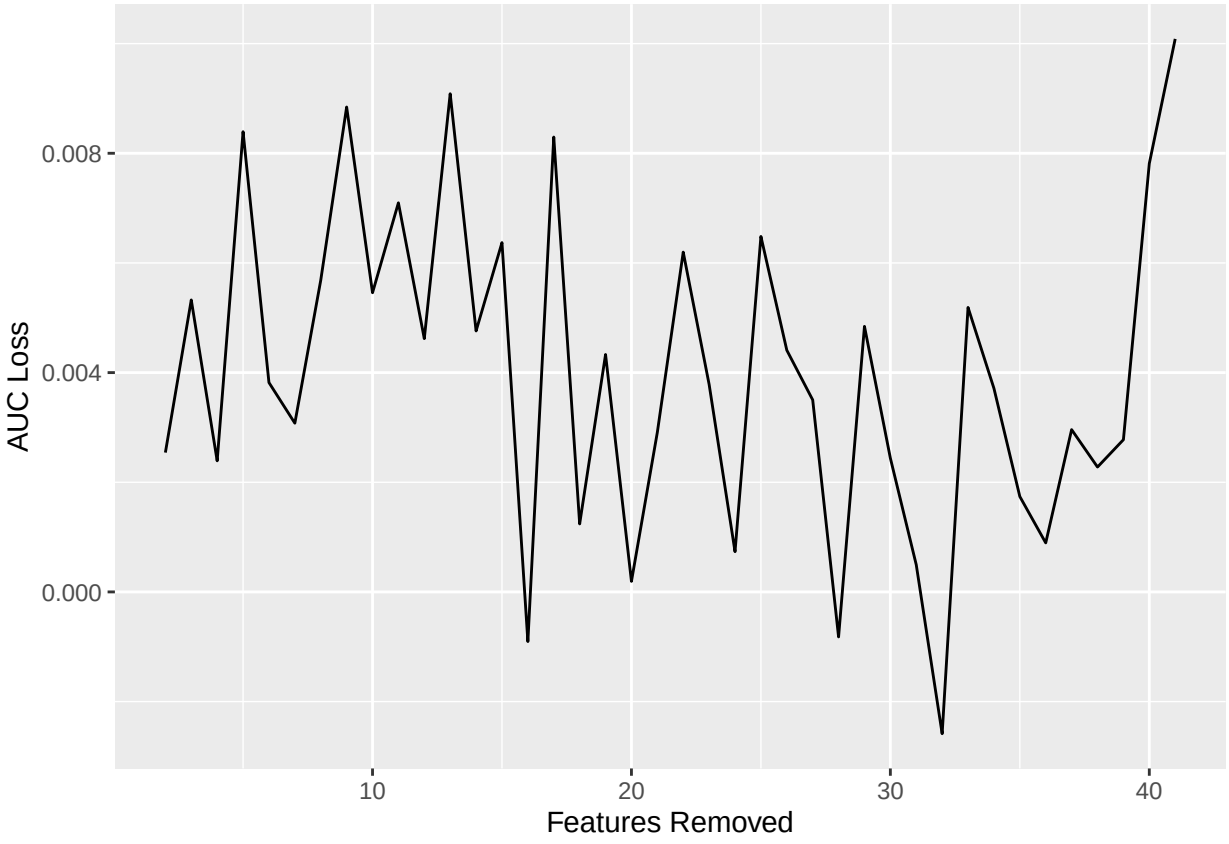
```
## [1] "Trimmed Model CV Train AUC: 0.790"
```

```
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)
```

```
## [1] "Trimmed Model Test AUC: 0.806"
```

```
selection_results %>%
  filter(`Number of Features` < length(features)) %>%
```

```
  mutate(`Features Removed` = length(features) - `Number of Features`) %>%
  ggplot(aes(x = `Features Removed`, y = `AUC Loss`)) +
  geom_line()
```



Minutes to run: 0.096

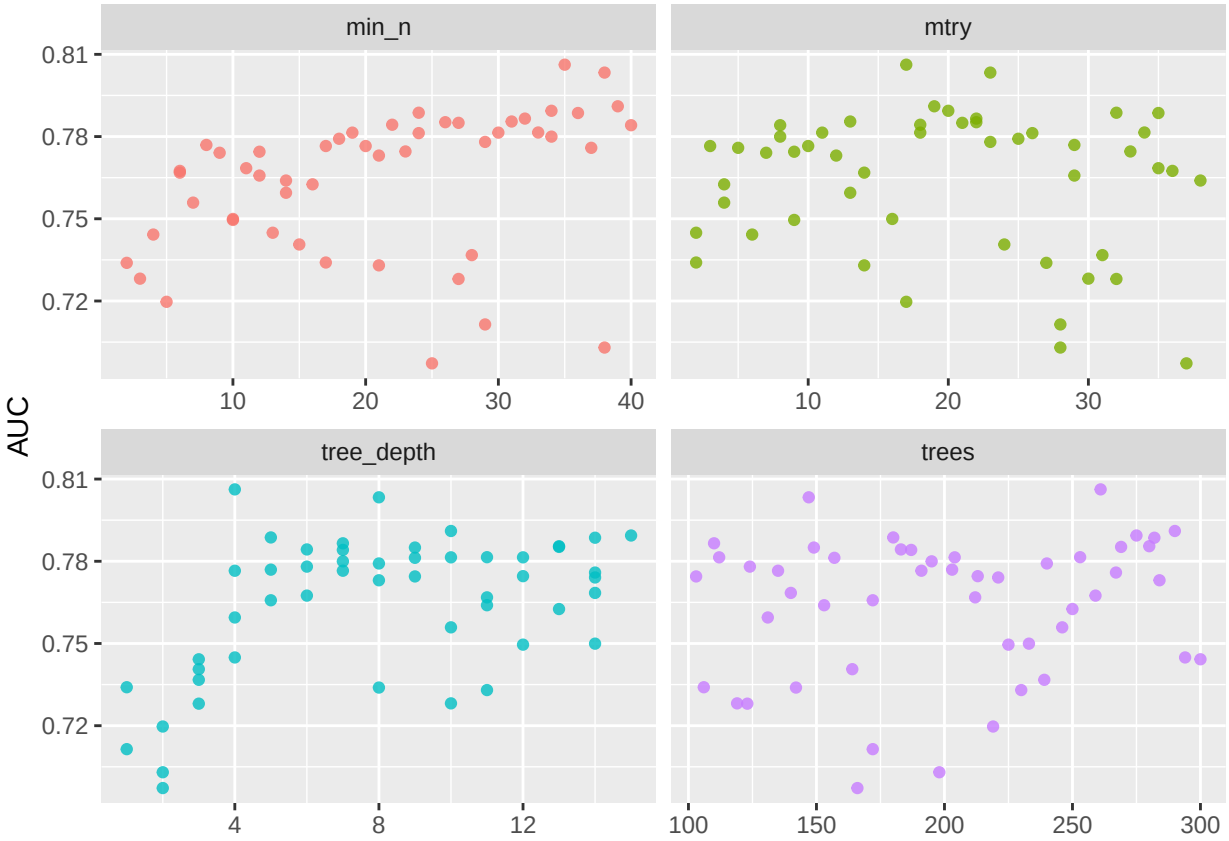
Hyperparameter tuning

```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% dplyr::select(all_of(c(selected_features, outcome_column)))) %>%
  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_impute_mean(all_numeric_predictors()) %>%
  step_zv(all_predictors())

lightgbm_spec <- boost_tree(
  mtry = tune(),
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  loss_reduction = tune()
) %>%
  set_engine("lightgbm") %>%
  set_mode("classification")

lightgbm_grid <- grid_latin_hypercube(
  finalize(mtry(),
           df_train %>% dplyr::select(all_of(c(selected_features, outcome_column)))),
  dials::trees(range = c(100L, 300L)),
  min_n(),
  tree_depth(),
```

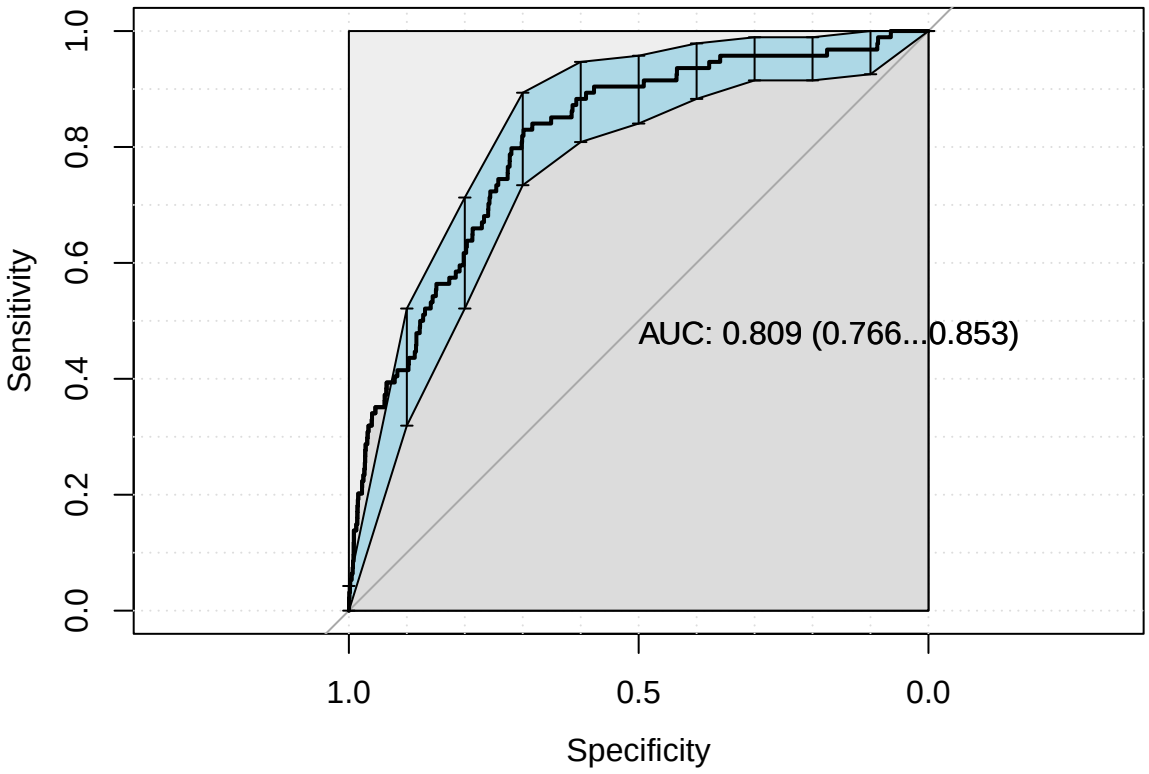
```
  learn_rate(),
  loss_reduction(),
  size = grid_size
)

lightgbm_workflow <-
  workflow() %>%
  add_recipe(lightgbm_recipe) %>%
  add_model(lightgbm_spec)

lightgbm_tune <-
  lightgbm_workflow %>%
  tune_grid(resamples = df_folds,
            grid = lightgbm_grid)

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

```
## # A tibble: 5 x 12
##    mtry trees min_n tree_depth  learn_rate loss_reduction .metric .estimator  mean     n std_err .config
##   <int> <int> <int>      <int>       <dbl>          <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    17   261    35          4 0.0163           7.40e- 5 roc_auc binary     0.806     4 0.0106  Preprocessor
## 2    23   147    38          8 0.00737          4.74e- 9 roc_auc binary     0.803     4 0.00842 Preprocessor
## 3    19   290    39         10 0.000000412      2.30e-10 roc_auc binary     0.791     4 0.0107  Preprocessor
## 4    20   275    34         15 0.000000110      2.11e- 5 roc_auc binary     0.789     4 0.0110  Preprocessor
## 5    32   180    24          5 0.00454          6.79e-10 roc_auc binary     0.789     4 0.00788 Preprocessor
```

```
best_lightgbm <- lightgbm_tune %>%
  select_best("roc_auc")

lightgbm_tune %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  select(mean, mtry:tree_depth) %>%
  pivot_longer(mtry:tree_depth,
               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_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_auc <- validation(final_lightgbm_fit, df_test)
```



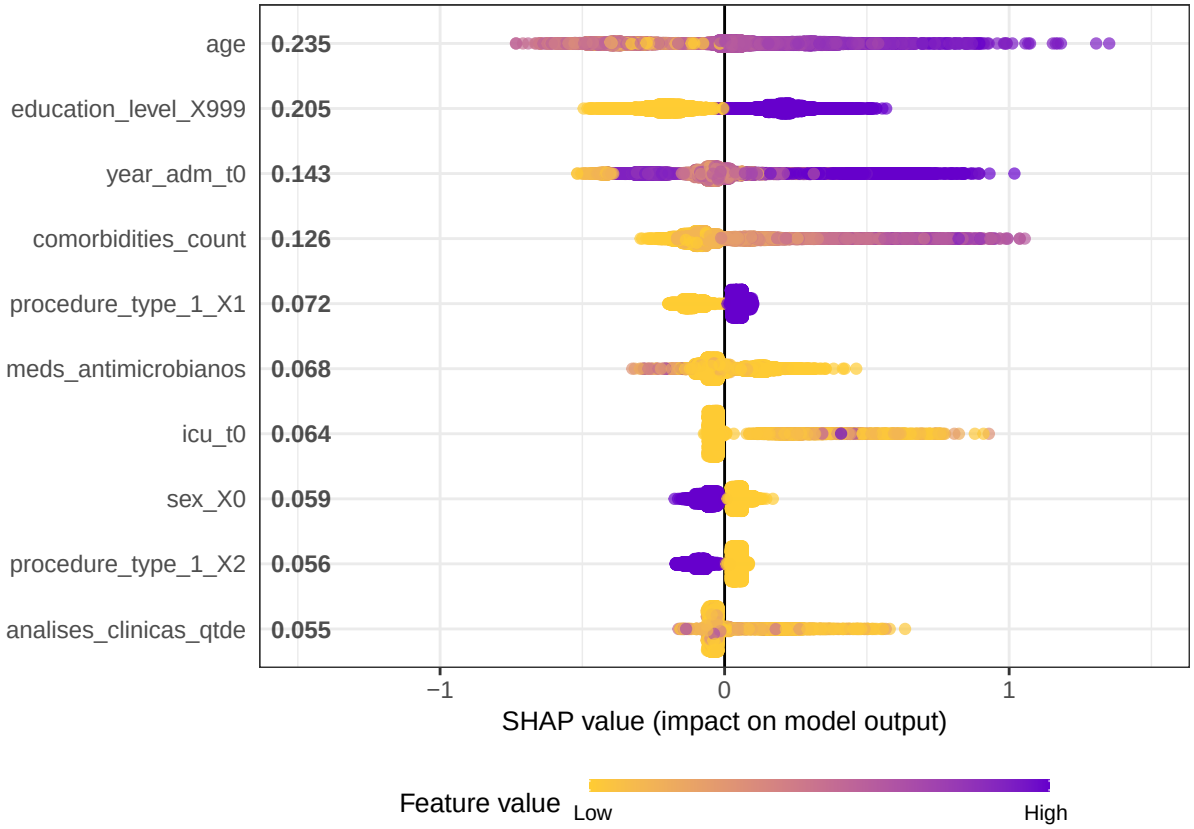

```
##   |
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                      0 4635   92
##                      1    1    2
##
##                Accuracy : 0.9803
##                  95% CI : (0.976, 0.9841)
##     No Information Rate : 0.9801
##     P-Value [Acc > NIR] : 0.4859
##
##                   Kappa : 0.0401
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99978
##             Specificity : 0.02128
##          Pos Pred Value : 0.98054
##          Neg Pred Value : 0.66667
##              Prevalence : 0.98013
##          Detection Rate : 0.97992
##    Detection Prevalence : 0.99937
##       Balanced Accuracy : 0.51053
##
##        'Positive' Class : 0
##
```

```
lightgbm_parameters <- lightgbm_tune %>%
  show_best("roc_auc", n=1) %>%
  select(trees, mtry, min_n, tree_depth, learn_rate, loss_reduction) %>%
  as.list
```

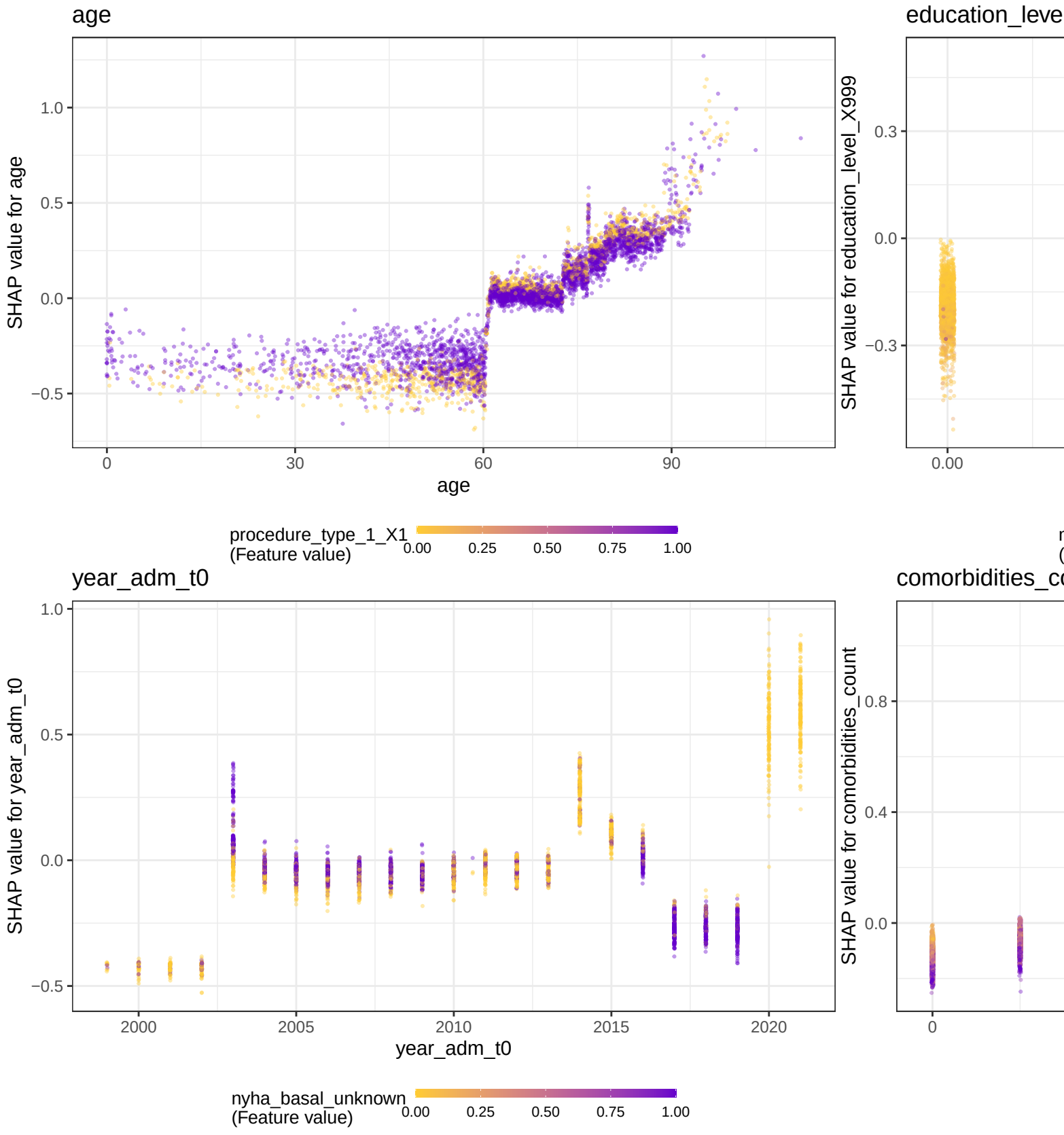
Minutes to run: 1.867

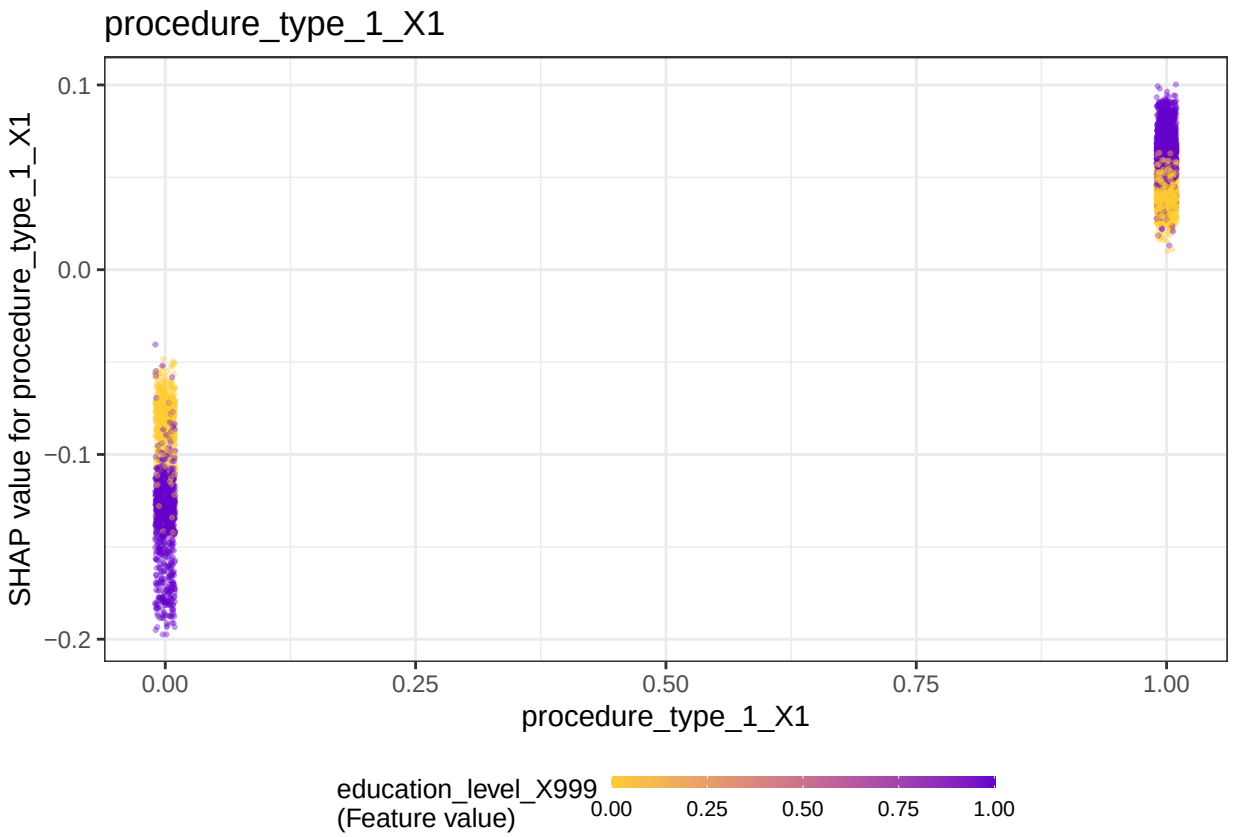
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)
```

```r
trained_rec <- prep(lightgbm_recipe, training = df_train)

df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

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

shap.plot.summary.wrap1(model = lightgbm_model, X = matrix_train, top_n = 10, dilute = F)
```




```r
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

for (x in shap.importance(shap, names_only = TRUE)[1:5]) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = FALSE,
    jitter_width = 0.01,
    alpha = 0.4
    ) +
  ggtitle(x)
  print(p)
}
```


age
SHAP value for age
1.0
0.5
0.0
−0.5
0
30
60
90
age
education_leve
SHAP value for education_level_X999
0.3
0.0
−0.3
0.00
procedure_type_1_X1
(Feature value)
0.00
0.25
0.50
0.75
1.00
year_adm_t0
SHAP value for year_adm_t0
1.0
0.5
0.0
−0.5
2000
2005
2010
2015
2020
year_adm_t0
comorbidities_co
SHAP value for comorbidities_count
0.8
0.4
0.0
0
nyha_basal_unknown
(Feature value)
0.00
0.25
0.50
0.75
1.00

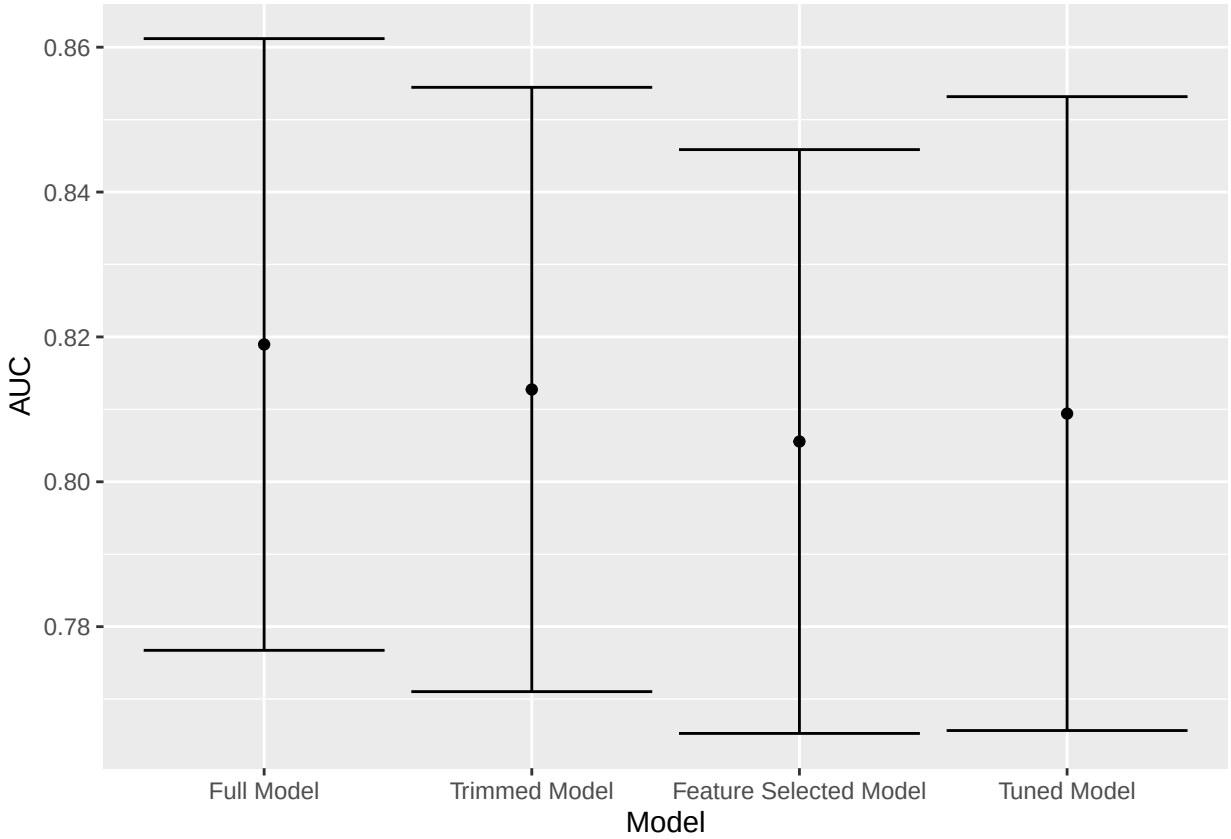


Minutes to run: 0.119

Minutes to run: 0

Models Comparison

```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper,
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper,
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower, feature_selected_model
  'Tuned Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3]
) %>%
  mutate(Target = outcome_column,
         Model = factor(Model,
                        levels = c('Full Model', 'Trimmed Model',
                                   'Feature Selected Model', 'Tuned Model')))

df_auc %>%
  ggplot(aes(x = Model, y = AUC, ymin = `Lower Limit`, ymax = `Upper Limit`)) +
    geom_point() +
    geom_errorbar()
```




```
saveRDS(df_auc, sprintf("../EDA/auxiliar/performance/tuning/%s_auc_result.RData", outcome_column))

# Save the final model
saveRDS(final_lightgbm_fit, sprintf("../EDA/results/%s/final_model.RData", outcome_column))
```

Minutes to run: 0.013