

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')
```

```

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. comorbidities_count
## 19. procedure_type_1
## 20. reop_type_1
## 21. procedure_type_new
## 22. cied_final_1
## 23. cied_final_group_1
## 24. admission_pre_t0_count
## 25. admission_pre_t0_180d
## 26. year_adm_t0
## 27. icu_t0
## 28. dialysis_t0
## 29. disch_outcomes_t0
## 30. admission_t0_emergency
## 31. aco
## 32. antiarritmico
## 33. ieca_bra
## 34. dva
## 35. digoxina
## 36. estatina
## 37. diuretico
## 38. vasodilatador
## 39. insuf_cardiaca
## 40. espironolactona
## 41. antiplaquetario_ev

```

```

## 42. insulina
## 43. psicofarmacos
## 44. antifungico
## 45. classe_meds_qtde
## 46. meds_cardiovasc_qtde
## 47. meds_antimicrobianos
## 48. outros_proced_cirurgicos
## 49. icp
## 50. angioplastia
## 51. cateterismo
## 52. cateter_venoso_central
## 53. proced_invasivos_qtde
## 54. transfusao
## 55. equipe_multiprof
## 56. ecg
## 57. holter
## 58. teste_esforco
## 59. tilt_teste
## 60. metodos_graficos_qtde
## 61. laboratorio
## 62. cultura
## 63. analises_clinicas_qtde
## 64. citologia
## 65. histopatologia_qtde
## 66. angio_tc
## 67. cintilografia
## 68. ecocardiograma
## 69. endoscopia
## 70. flebografia
## 71. ultrassom
## 72. tomografia
## 73. radiografia
## 74. ressonancia
## 75. exames_imagem_qtde
## 76. 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: 0.794"  
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)  
  
## [1] "Full Model Test AUC: 0.784"
```

Minutes to run: 0.102

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

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: 0.796"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.787"

Minutes to run: 0.1

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 <- 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_dropped <- 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 0.002807779
## [1] 73.000000000 -0.001388065
## [1] 72.000000000 -0.002084355
## [1] 7.10000e+01 1.92392e-04
## [1] 70.000000000 -0.002912257
## [1] 69.000000000 0.002957129
## [1] 6.800000e+01 1.183862e-04
## [1] 67.000000000 0.001448815
## [1] 66.000000000 -0.0006487411
## [1] 65.000000000 -0.0006539102
## [1] 64.000000000 0.001710431

```

```

## [1] 63.000000000 0.001534488
## [1] 62.000000000 -0.003217651
## [1] 61.000000000 -0.0007578386
## [1] 60.000000000 0.001104596
## [1] 59.000000000 -0.002249132
## [1] 58.000000000 0.003588709
## [1] 57.000000000 -0.0006350987
## [1] 56.000000000 -0.004655715
## [1] 55.000000000 -0.001774632
## [1] 54.000000000 -0.00397491
## [1] 53.000000000 -0.003203755
## [1] 5.200000e+01 2.984299e-05
## [1] 51.000000000 -0.0006200232
## [1] 50.000000000 -0.002613865
## [1] 49.000000000 0.001790862
## [1] 4.800000e+01 8.293989e-04
## [1] 47.000000000 -0.001111528
## [1] 46.000000000 -0.003992332
## [1] 45.000000000 -0.000353754
## [1] 44.000000000 -0.0006221621
## [1] 4.300000e+01 4.103839e-04
## [1] 42.000000000 -0.001237447
## [1] 41.000000000 -0.001157139
## [1] 40.000000000 -0.001100186
## [1] 39.000000000 0.003382816
## [1] 38.000000000 0.004925578
## [1] 37.000000000 -0.001268674
## [1] 36.000000000 0.002248402
## [1] 35.000000000 -0.0005656797
## [1] 34.000000000 0.001942292
## [1] 33.000000000 0.002343123
## [1] 32.0000000 0.00243962
## [1] 31.000000000 -0.0009422174
## [1] 30.000000000 0.002085274
## [1] 29.000000000 0.003802653
## [1] 28.000000000 0.004201858
## [1] 27.000000000 0.003124311
## [1] 26.000000000 0.001173798
## [1] 2.500000e+01 5.451354e-04
## [1] 24.000000000 0.003366458
## [1] 23.0000000 0.01303049

selection_results

## # A tibble: 54 x 3
##   `Number of Features` `AUC Loss` `Least Important Feature`
##   <int>        <dbl> <chr>
## 1 76          0     heart_disease
## 2 75      -0.00175 angioplastia
## 3 74      0.00281 histopatologia_qtde
## 4 73      -0.00139 tilt_teste
## 5 72      -0.00208 transfusao
## 6 71      0.000192 heart_disease
## 7 70      -0.00291 teste_esforco
## 8 69      0.00296 antiplaquetario_ev
## 9 68      0.000118 endoscopia
## 10 67      0.00145 dialysis_t0
## # ... with 44 more rows

Minutes to run: 4.567

selected_features <- c(current_features, last_feature_dropped)

```

```

feature_selected_model <- model_fit_wf(selected_features,
                                         outcome_column, hyperparameters)

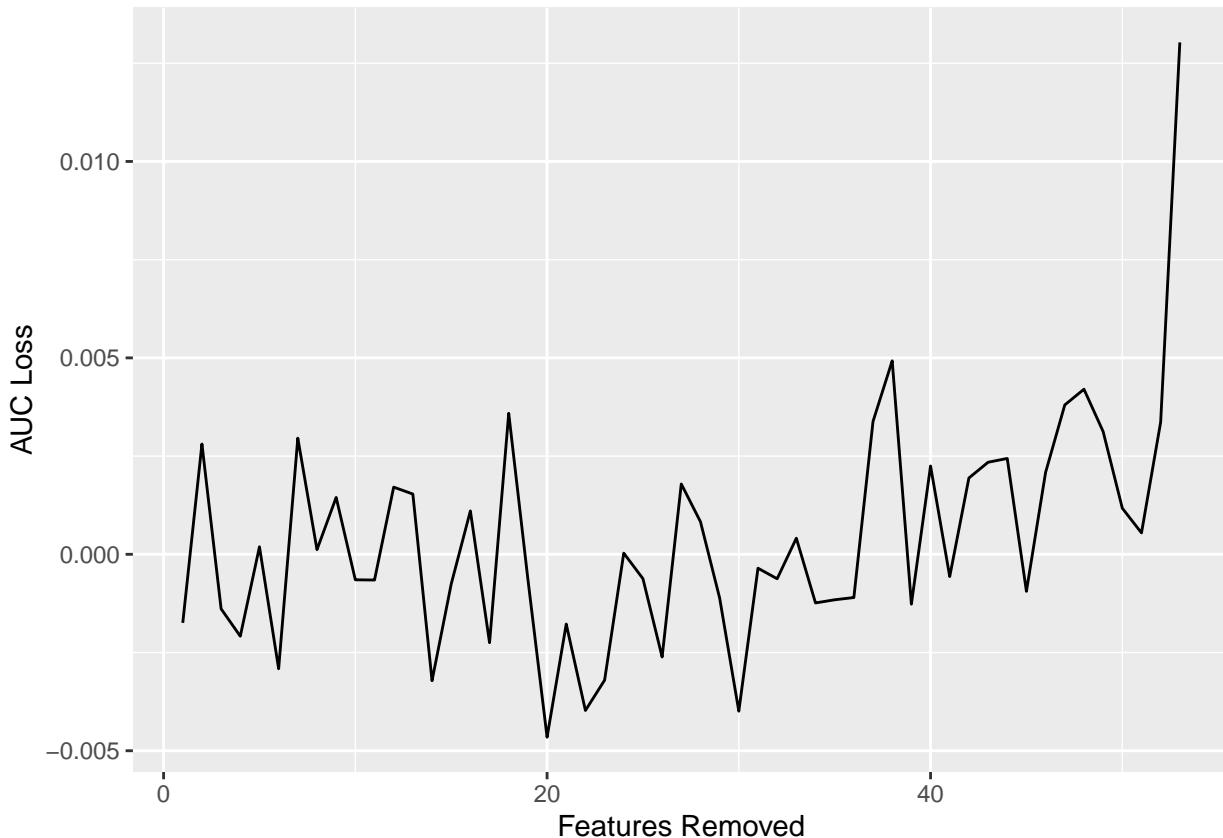
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.789"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.786"

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()

```



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)

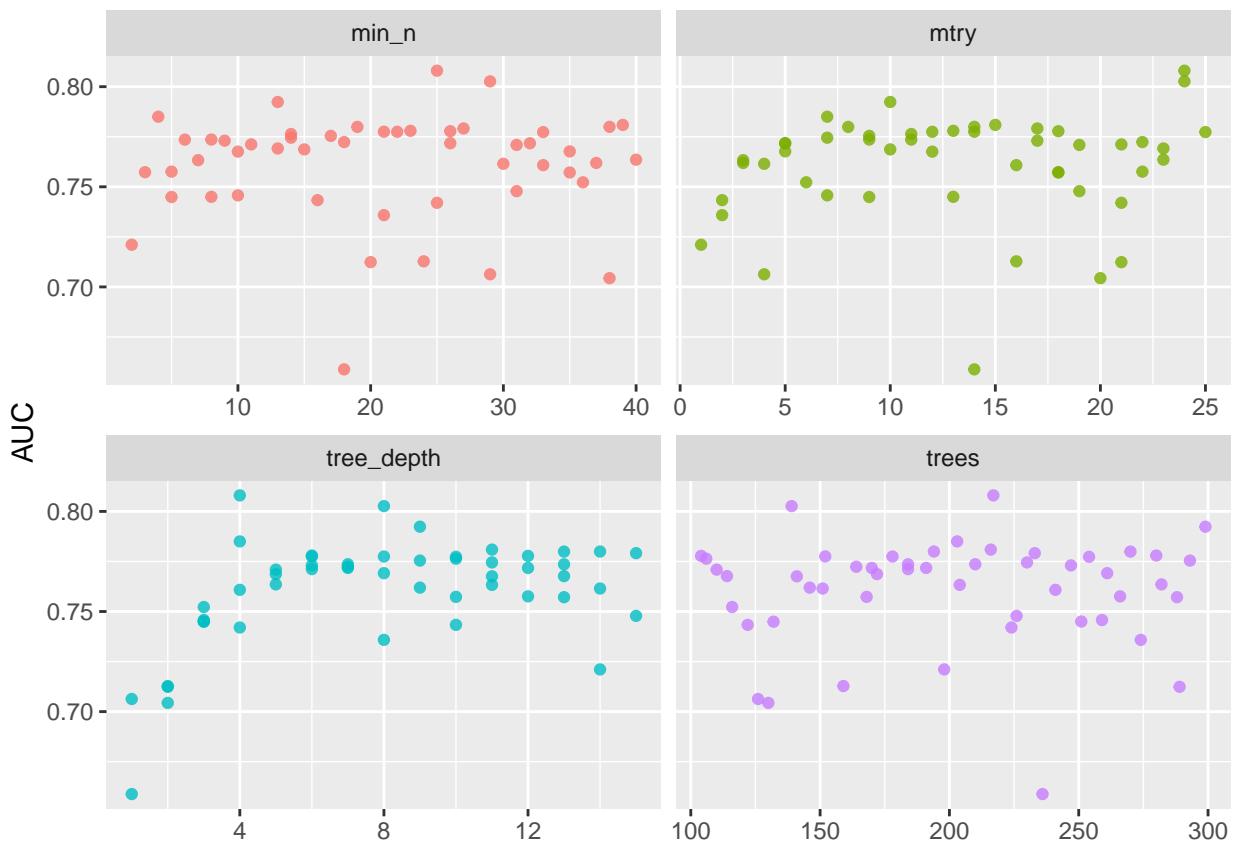
lightgbm_tune %>%
show_best("roc_auc")

## # 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    24    217    25        4  0.0163      7.67e- 3 roc_auc binary  0.808    4  0.00531 Preprocess
## 2    24    139    29        8  0.0331      6.66e- 5 roc_auc binary  0.803    4  0.00579 Preprocess
## 3    10    299    13        9  0.00629     8.60e-10 roc_auc binary  0.792    4  0.00515 Preprocess
## 4     7    203     4        4  0.0196      7.46e+ 0 roc_auc binary  0.785    4  0.00636 Preprocess
## 5    15    216    39       11  0.00000000263  1.32e- 3 roc_auc binary  0.781    4  0.00492 Preprocess

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")

```



```

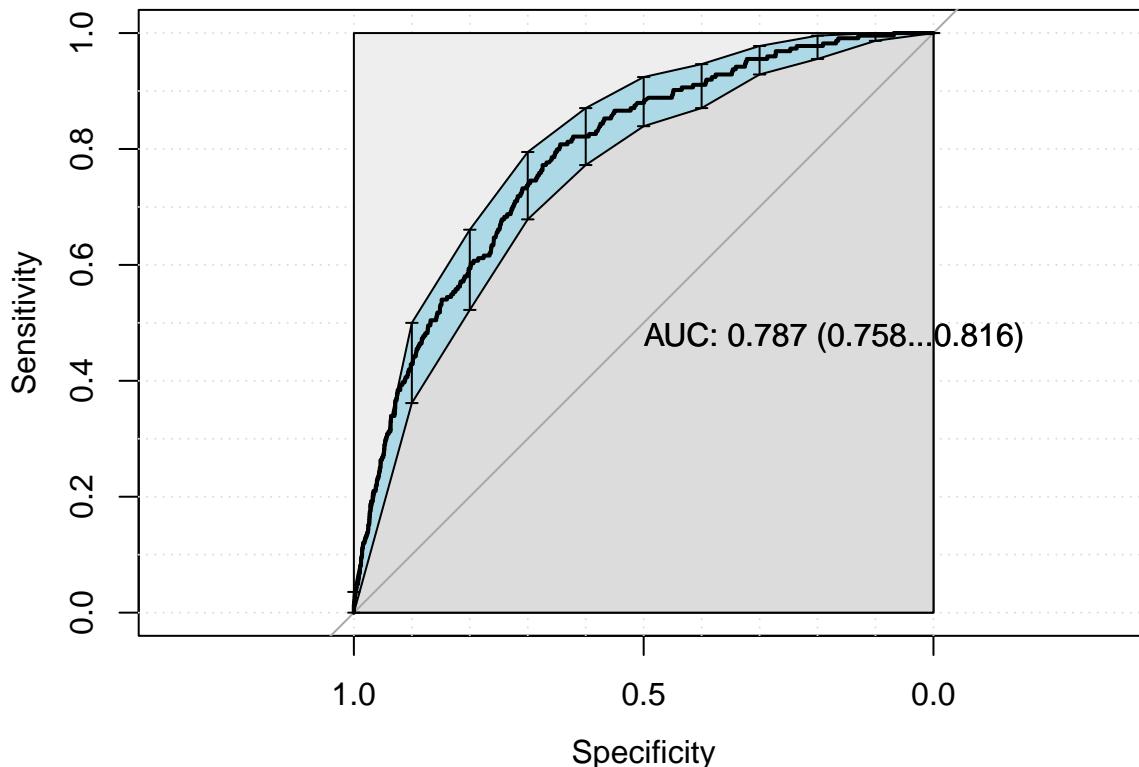
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4502  220
##                      1     4     4
## 
##                         Accuracy : 0.9526
##                         95% CI : (0.9462, 0.9585)
## No Information Rate : 0.9526
## P-Value [Acc > NIR] : 0.5178
## 
##                         Kappa : 0.0313
## 
## Mcnemar's Test P-Value : <2e-16
## 
##                         Sensitivity : 0.99911
##                         Specificity : 0.01786
## Pos Pred Value : 0.95341
## Neg Pred Value : 0.50000
## Prevalence : 0.95264
## Detection Rate : 0.95180
## Detection Prevalence : 0.99831
## Balanced Accuracy : 0.50848
## 
## '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: 1.673

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

```

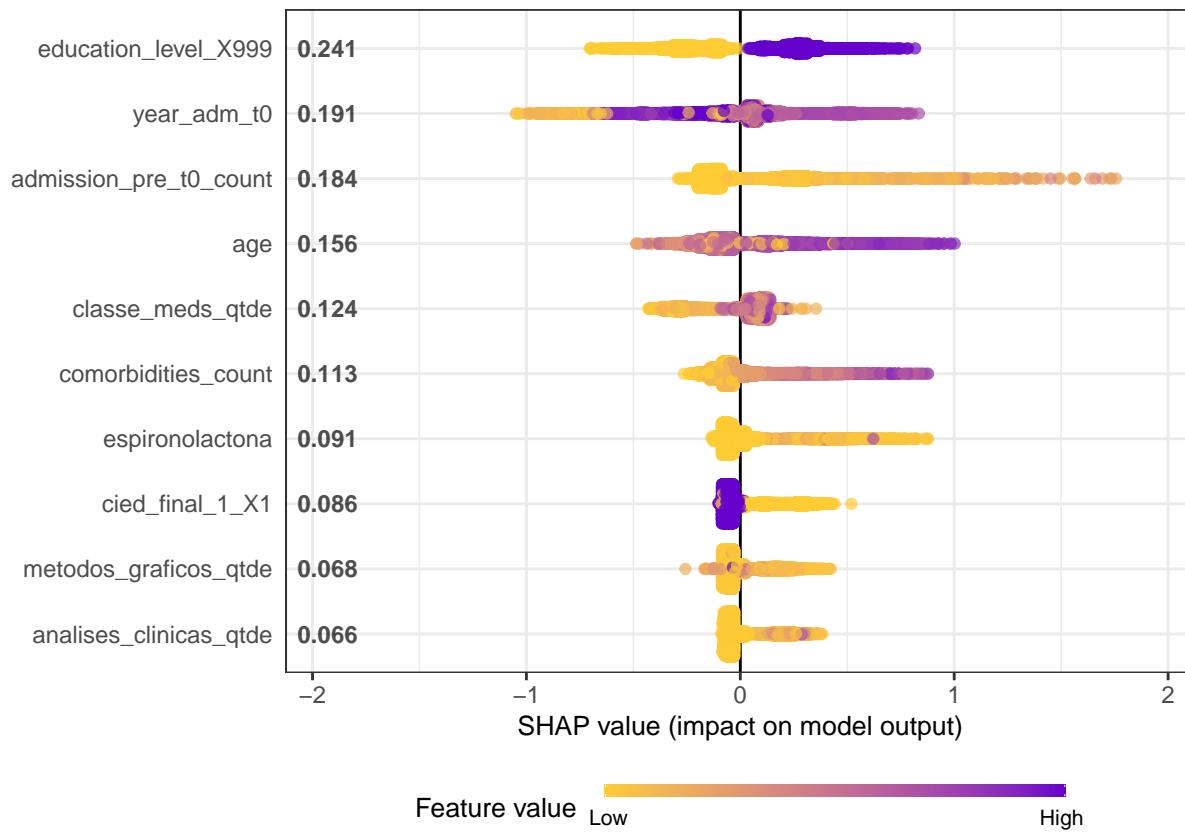
trained_rec <- prep(lightgbm_recipe, training = df_train)

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)

```

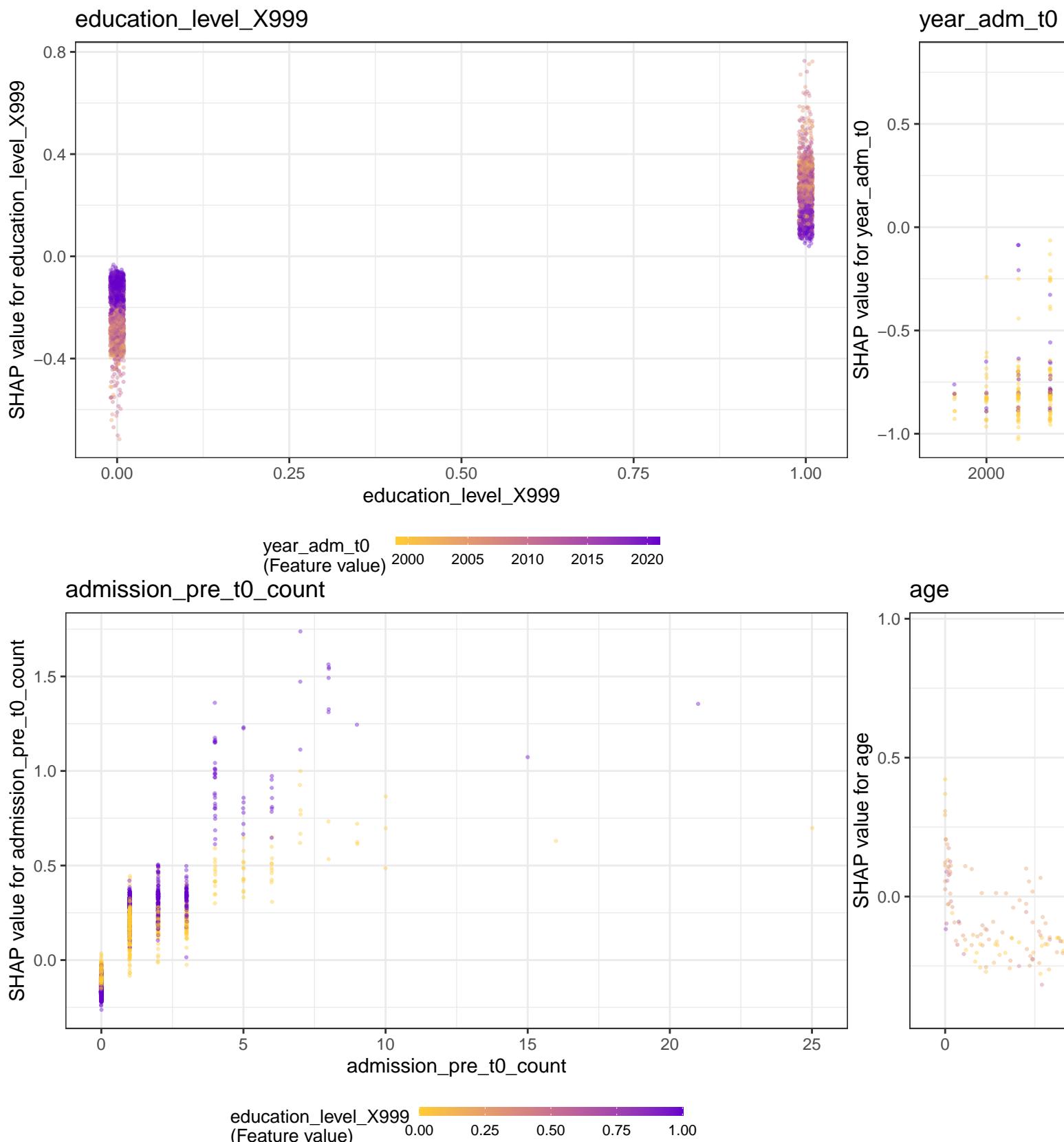


```

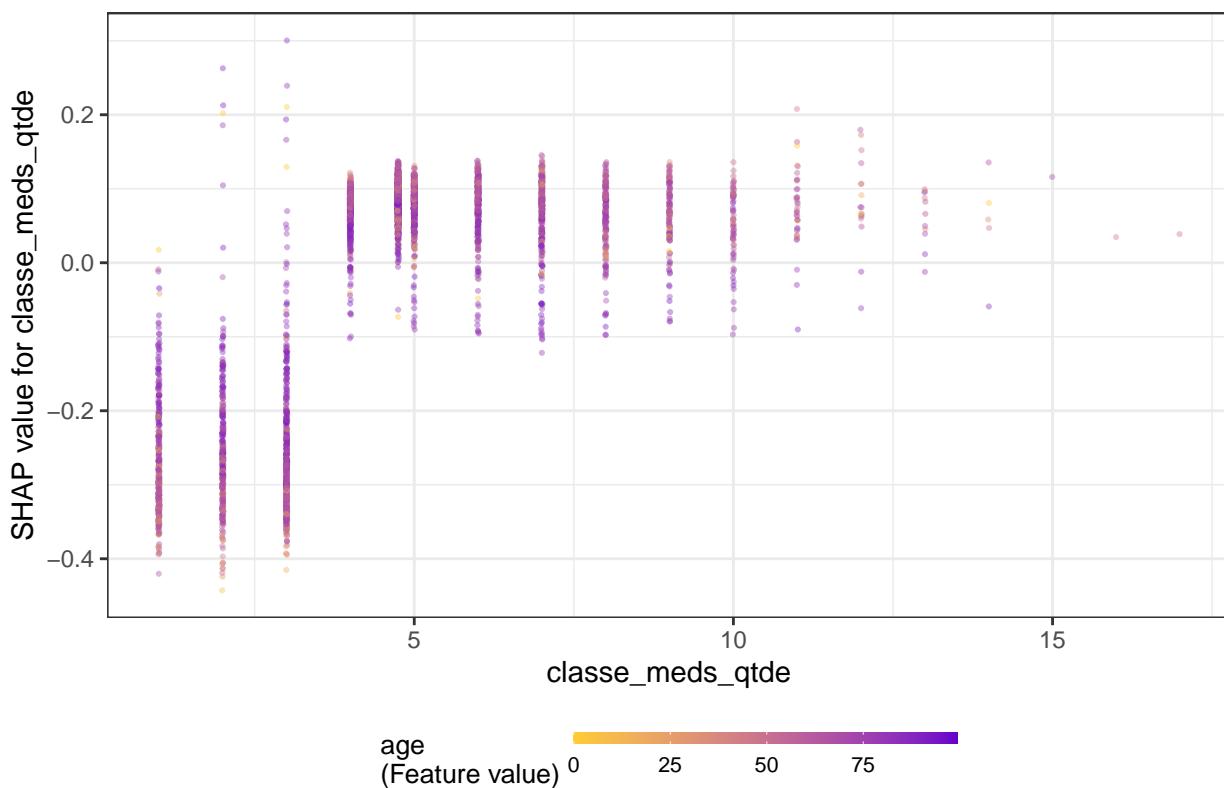
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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)
}

```



classe_meds_qtde

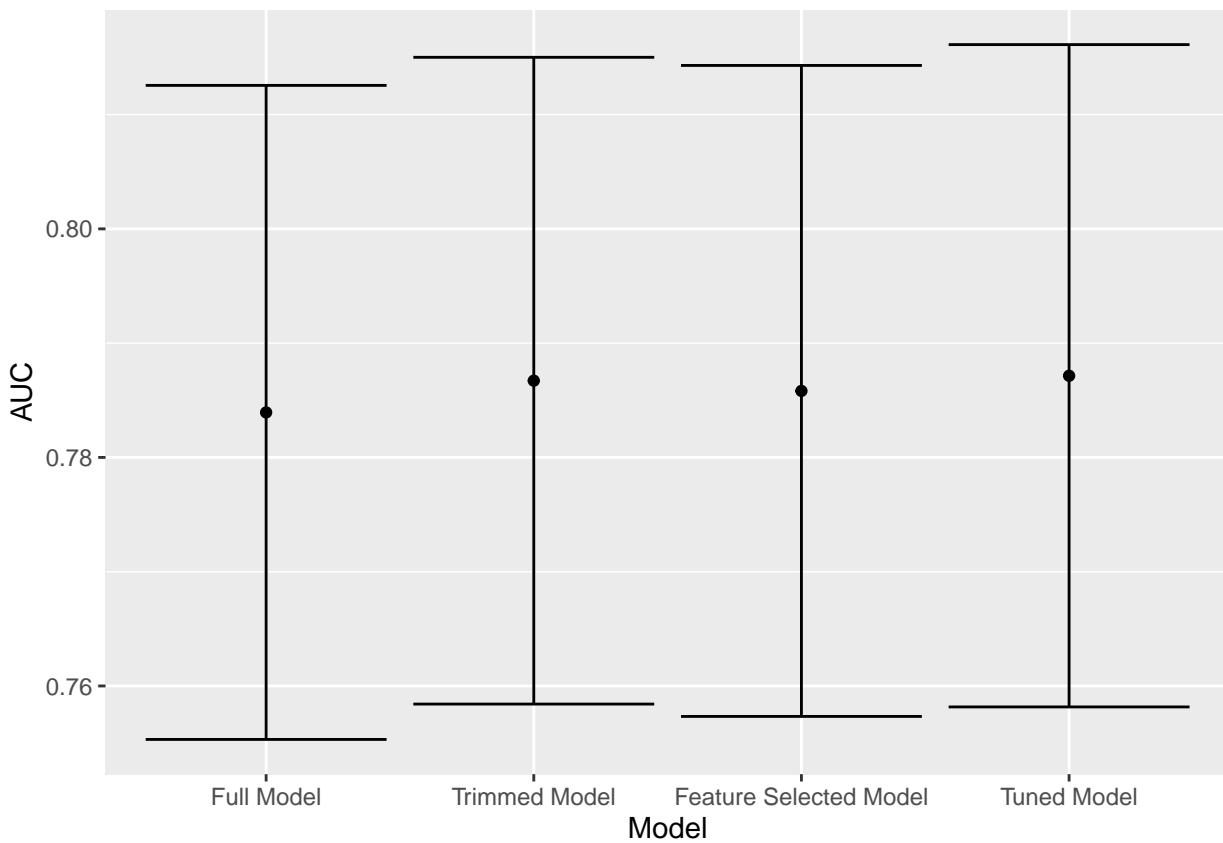


Minutes to run: 0

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$auc_upper,
  'Tuned Model', as.numeric(lightgbm_auc$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: 0.01