

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: 0.007

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))

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. race
## 04. education_level
## 05. patient_state
## 06. underlying_heart_disease
## 07. heart_disease
## 08. nyha_basal
## 09. prior_mi
## 10. heart_failure
## 11. af
## 12. cardiac_arrest
## 13. transplant
## 14. valvopathy
## 15. endocardites
## 16. diabetes
## 17. renal_failure
## 18. hemodialysis
## 19. copd
## 20. comorbidities_count
## 21. procedure_type_1
## 22. reop_type_1
## 23. procedure_type_new
## 24. cied_final_1
## 25. cied_final_group_1
## 26. admission_t0
## 27. admission_pre_t0_180d
## 28. year_adm_t0
## 29. icu_t0
## 30. dialysis_t0
## 31. disch_outcomes_t0
## 32. admission_t0_emergency
## 33. aco
## 34. antiarritmico
## 35. betabloqueador
## 36. ieca_bra
## 37. dva
## 38. digoxina
## 39. estatina
## 40. diuretico
## 41. vasodilatador

```

```
## 42. insuf_cardiaca
## 43. espironolactona
## 44. bloq_calcio
## 45. antiplaquetario_ev
## 46. insulina
## 47. anticonvulsivante
## 48. psicofarmacos
## 49. antifungico
## 50. antiviral
## 51. antiretroviral
## 52. classe_meds_qtde
## 53. meds_cardiovasc_qtde
## 54. meds_antimicrobianos
## 55. vni
## 56. cec
## 57. transplante_cardiaco
## 58. outros_proced_cirurgicos
## 59. icp
## 60. intervencao_cv
## 61. angioplastia
## 62. cateterismo
## 63. eletrofisiologia
## 64. cateter_venoso_central
## 65. proced_invasivos_qtde
## 66. cve_desf
## 67. transfusao
## 68. interconsulta
## 69. equipe_multiprof
## 70. ecg
## 71. holter
## 72. teste_esforco
## 73. espiro_ergoespiro
## 74. tilt_teste
## 75. metodos_graficos_qtde
## 76. laboratorio
## 77. cultura
## 78. analises_clinicas_qtde
## 79. citologia
## 80. biopsia
## 81. histopatologia_qtde
## 82. angio_rm
## 83. angio_tc
## 84. arteriografia
## 85. cintilografia
## 86. ecocardiograma
## 87. endoscopia
## 88. flebografia
## 89. pet_ct
## 90. ultrassom
## 91. tomografia
## 92. radiografia
## 93. ressonancia
## 94. exames_imagem_qtde
## 95. bic
## 96. mpp
```

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] 0.6999239
dim(df_test)[1] / dim(df)[1]

## [1] 0.3000761

Minutes to run: 0.005
```

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[2]))
}

```

```

        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: 0.721"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.705"

Minutes to run: 0.218

Features with zero importance on the initial model:

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

unimportant_features %>%
  gluedown::md_order()

## 01. heart_disease
## 02. transplant
## 03. hemodialysis
## 04. dialysis_t0
## 05. disch_outcomes_t0
## 06. antiplaquetario_ev
## 07. antiviral
## 08. antiretroviral
## 09. vni
## 10. cec
## 11. transplante_cardiaco
## 12. intervencao_cv
## 13. angioplastia
## 14. cve_desf
## 15. transfusao
## 16. espiro_ergoespiro
## 17. tilt_teste
## 18. citologia
## 19. angio_rm
## 20. arteriografia
## 21. flebografia
## 22. mpp

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: 0.721"

```

```

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

## [1] "Trimmed Model Test AUC: 0.700"

Minutes to run: 0.219

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_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] 7.300000e+01 6.934639e-04
## [1] 7.200000e+01 6.934639e-04
## [1] 7.100000e+01 -3.417968e-05
## [1] 70.0000000000 -0.0005846241
## [1] 6.900000e+01 -5.481202e-05
## [1] 6.800000e+01 3.957346e-04
## [1] 6.700000e+01 2.697581e-04
## [1] 6.600000e+01 2.830393e-04
## [1] 6.500000e+01 4.957651e-04
## [1] 6.400000e+01 9.878075e-06
## [1] 63.0000000000 -0.0007553996
## [1] 62.0000000000 -0.0004728753
## [1] 61.0000000000 -0.0004728753
## [1] 60.0000000000 -0.0005392654
## [1] 59.0000000000 -0.0003871522
## [1] 58.0000000000 -0.0003871522
## [1] 57.0000000000 -0.0006746929
## [1] 56.0000000000 -0.0004313135
## [1] 55.0000000000 -0.0002622399
## [1] 54.0000000000 -0.0005217077
## [1] 53.0000000000 -0.0005366937
## [1] 5.200000e+01 5.276145e-04
## [1] 51.0000000000 -0.0005956396
## [1] 50.0000000000 -0.0007742357
## [1] 49.0000000000 -0.0006117594
## [1] 48.0000000000 -0.0004024132
## [1] 47.0000000000 -0.0001596606

```

```

## [1] 4.600000e+01 6.222833e-04
## [1] 4.500000e+01 1.591366e-04
## [1] 4.400000e+01 -3.641992e-05
## [1] 4.300000e+01 -5.440799e-05
## [1] 4.200000e+01 1.502005e-04
## [1] 41.000000000 -0.001210487
## [1] 40.000000000 -0.0005355236
## [1] 39.000000000 -0.000883536
## [1] 38.000000000 -0.0007657857
## [1] 37.000000000 -0.0001815185
## [1] 36.000000000 -0.0001290128
## [1] 3.500000e+01 3.232348e-04
## [1] 3.400000e+01 7.304867e-04
## [1] 33.000000000 0.001463984
## [1] 32.000000000 0.002086323
## [1] 31.000000000 0.002353282
## [1] 30.000000000 0.002550518
## [1] 29.000000000 0.002237473
## [1] 28.000000000 0.001514214
## [1] 27.000000000 0.001952952
## [1] 26.000000000 0.001837346
## [1] 25.000000000 0.001650065
## [1] 24.000000000 0.002557833
## [1] 23.000000000 0.00240737
## [1] 22.000000000 0.002675185
## [1] 21.000000000 0.004060353
## [1] 20.000000000 0.005335882
## [1] 19.000000000 0.004925148
## [1] 18.0000000 0.0041134
## [1] 17.000000000 0.00347279
## [1] 16.000000000 0.004671835
## [1] 15.0000000 0.00589168
## [1] 14.000000000 0.007462429
## [1] 13.000000000 0.00910642
## [1] 12.000000000 0.009298177
## [1] 11.000000000 0.01105005

```

`selection_results`

```

## # A tibble: 65 x 3
##   `Number of Features` `AUC Loss` `Least Important Feature`
##   <int>      <dbl> <chr>
## 1 96        0       pet_ct
## 2 74        0.000195 diabetes
## 3 73        0.000693 pet_ct
## 4 72        0.000693 prior_mi
## 5 71        -0.0000342 renal_failure
## 6 70        -0.000585 interconsulta
## 7 69        -0.0000548 cultura
## 8 68        0.000396 angio_tc
## 9 67        0.000270 icp
## 10 66       0.000283 insulina
## # ... with 55 more rows
## # i Use `print(n = ...)` to see more rows

```

Minutes to run: 11.517

```

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)

```

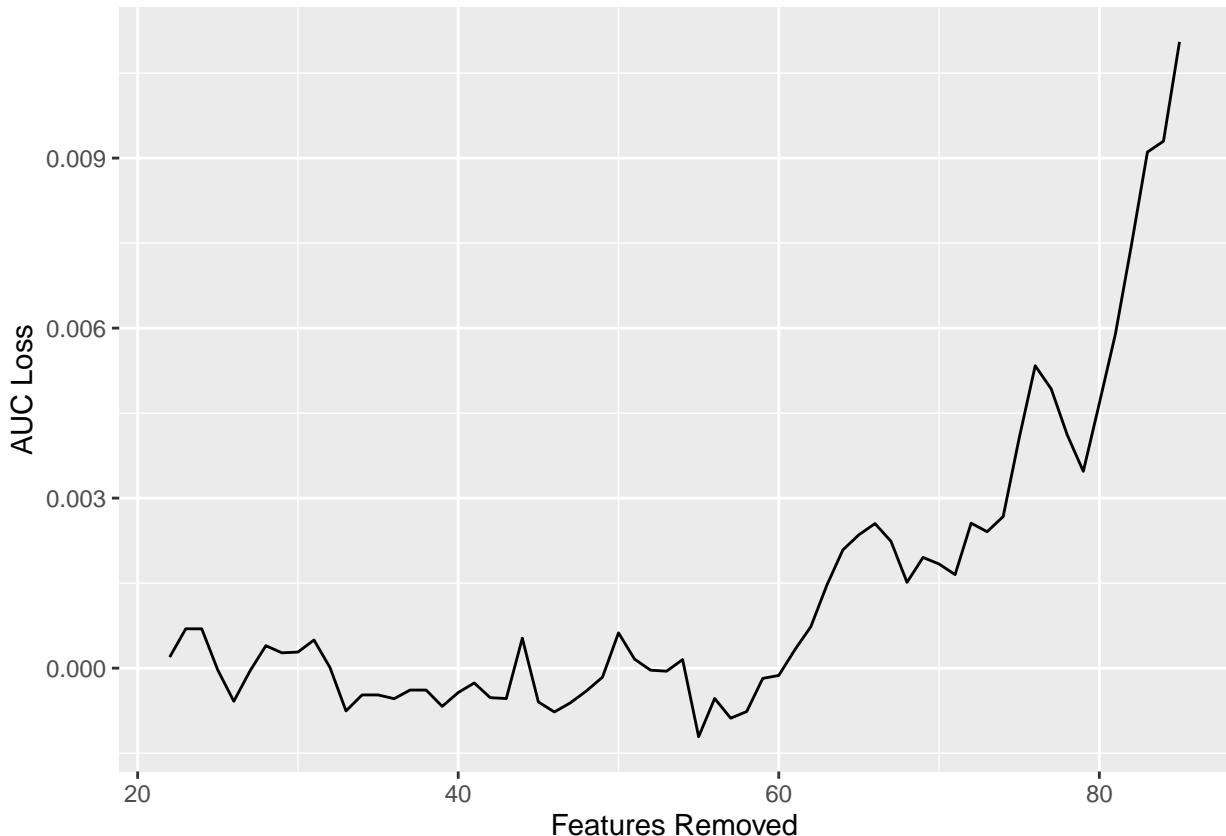
```

## [1] "Trimmed Model CV Train AUC: 0.713"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.701"

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:

0.175

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)

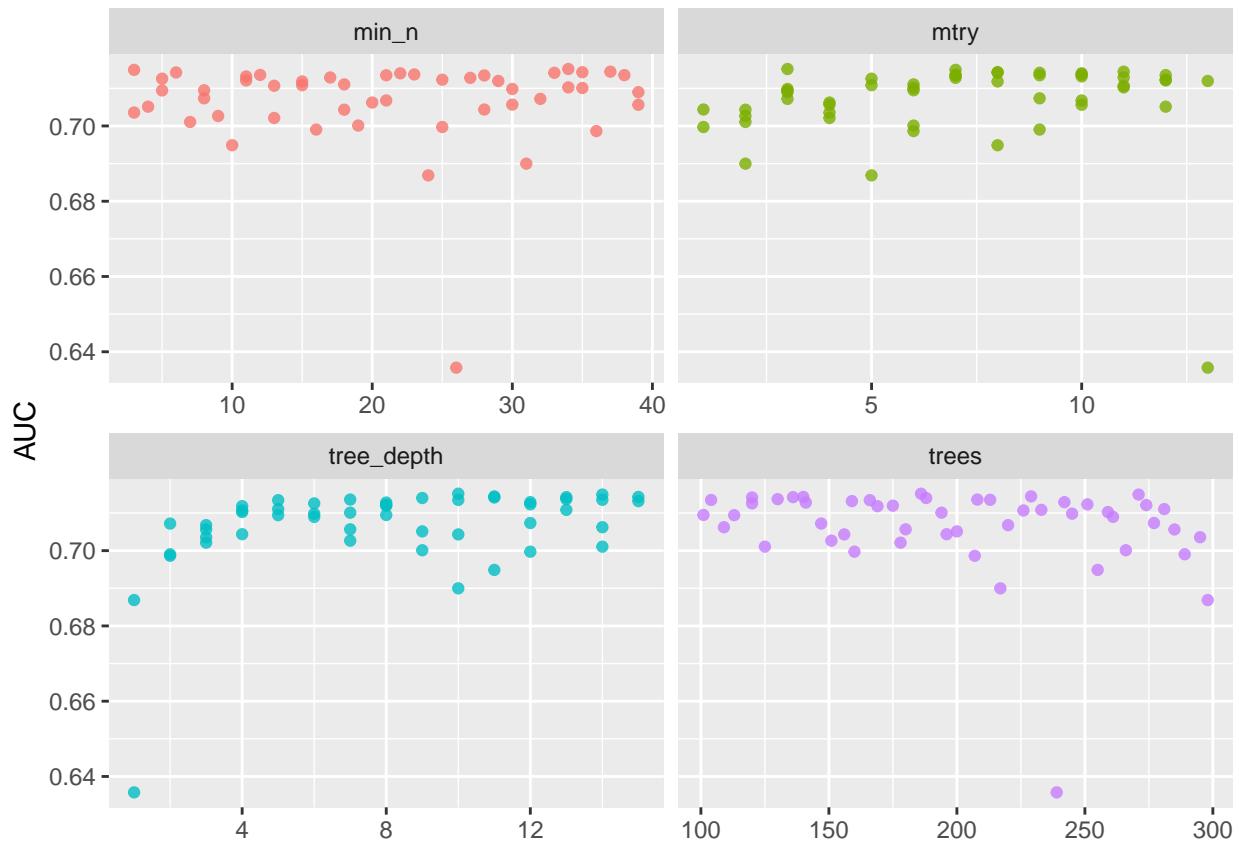
lightgbm_tune %>%
  show_best("roc_auc")

## # A tibble: 5 x 12
##   mtry trees min_n tree_depth learn_r~1 loss_~2 .metric .esti~3  mean
##   <int> <int> <int>      <int>    <dbl> <dbl> <chr> <chr> <dbl>
## 1     3    186     34        10  1.77e- 2 6.02e+0 roc_auc binary  0.715
## 2     7    271     3         14  2.35e- 5 2.79e-5 roc_auc binary  0.715
## 3    11    229     37        11  8.07e- 7 3.34e-1 roc_auc binary  0.714
## 4     8    140     35        15  5.01e- 6 4.55e-5 roc_auc binary  0.714
## 5     8    136      6        13  1.30e-10 1.62e-4 roc_auc binary  0.714
## # ... with 3 more variables: n <int>, std_err <dbl>, .config <chr>,
## #   and abbreviated variable names 1: learn_rate, 2: loss_reduction,
## #   3: .estimator
## # i Use `colnames()` to see all variable names

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")

```



```

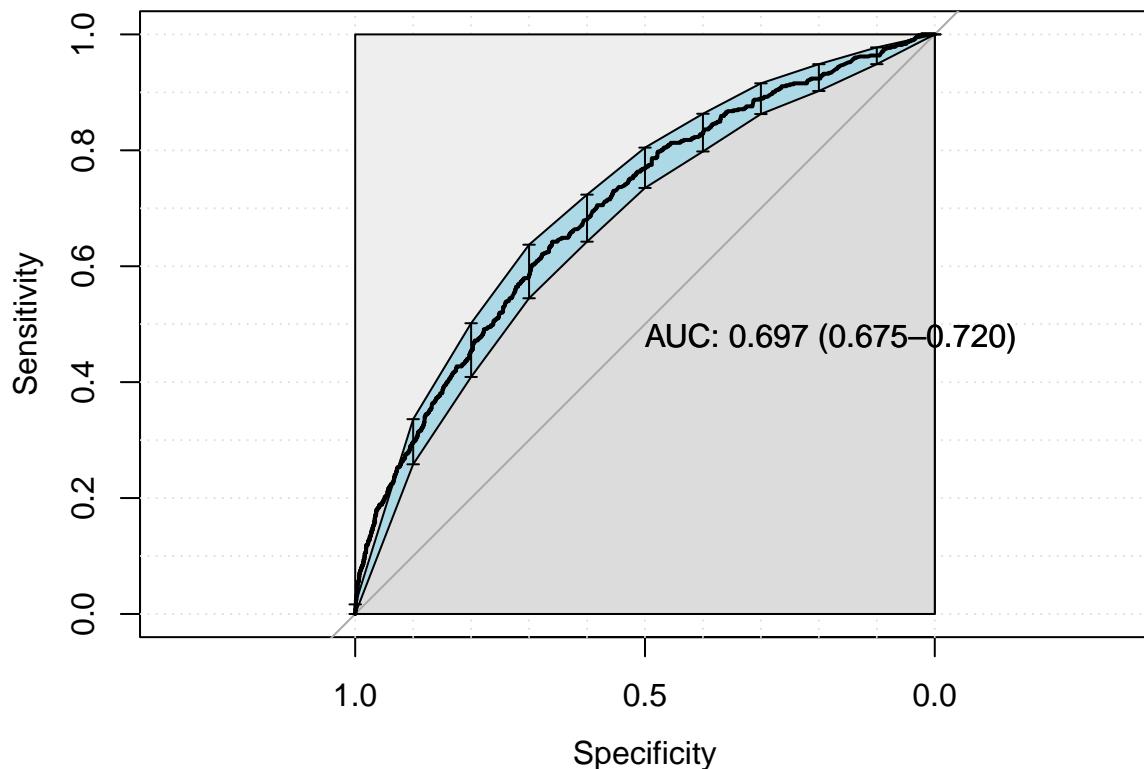
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4126  602
##                      1     1     2
##
##          Accuracy : 0.8725
## 95% CI : (0.8627, 0.8819)
##  No Information Rate : 0.8723
## P-Value [Acc > NIR] : 0.4935
##
##          Kappa : 0.0053
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.999758
##          Specificity  : 0.003311
##  Pos Pred Value  : 0.872673
##  Neg Pred Value  : 0.666667
##          Prevalence   : 0.872331
##          Detection Rate: 0.872120
##  Detection Prevalence: 0.999366
##          Balanced Accuracy: 0.501534
##
##          '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: 4.952

```

lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

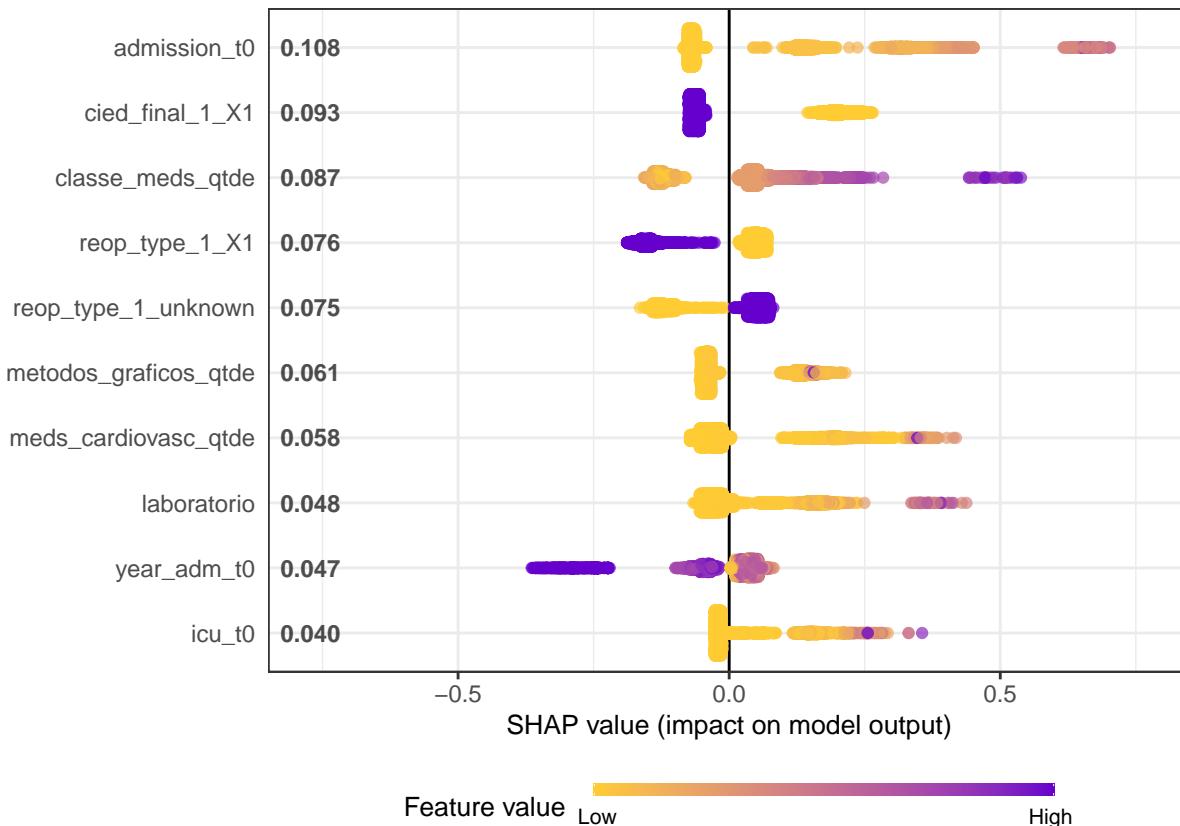
trained_rec <- prep(lightgbm_recipe, training = df_train)

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)

```

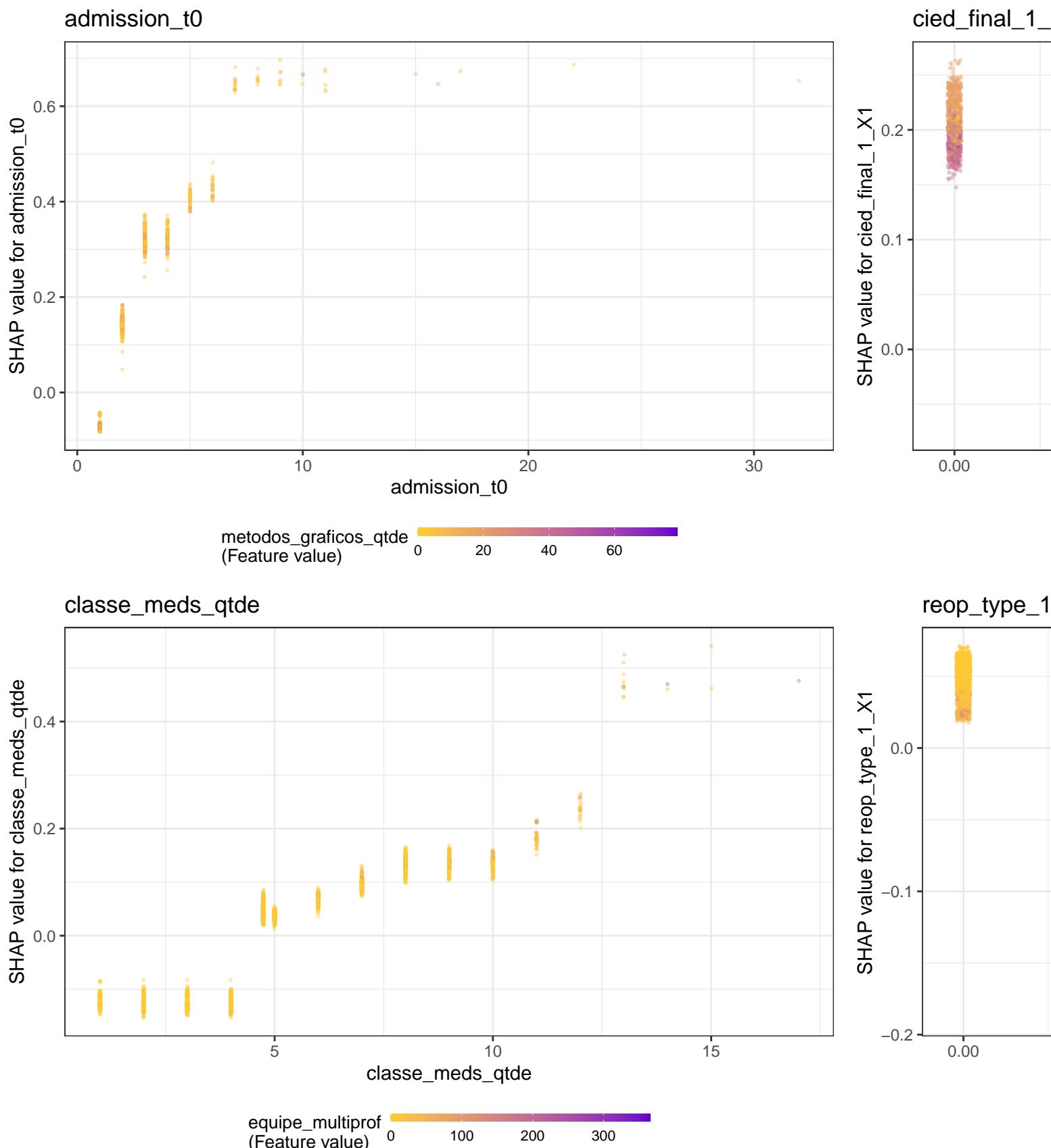


```

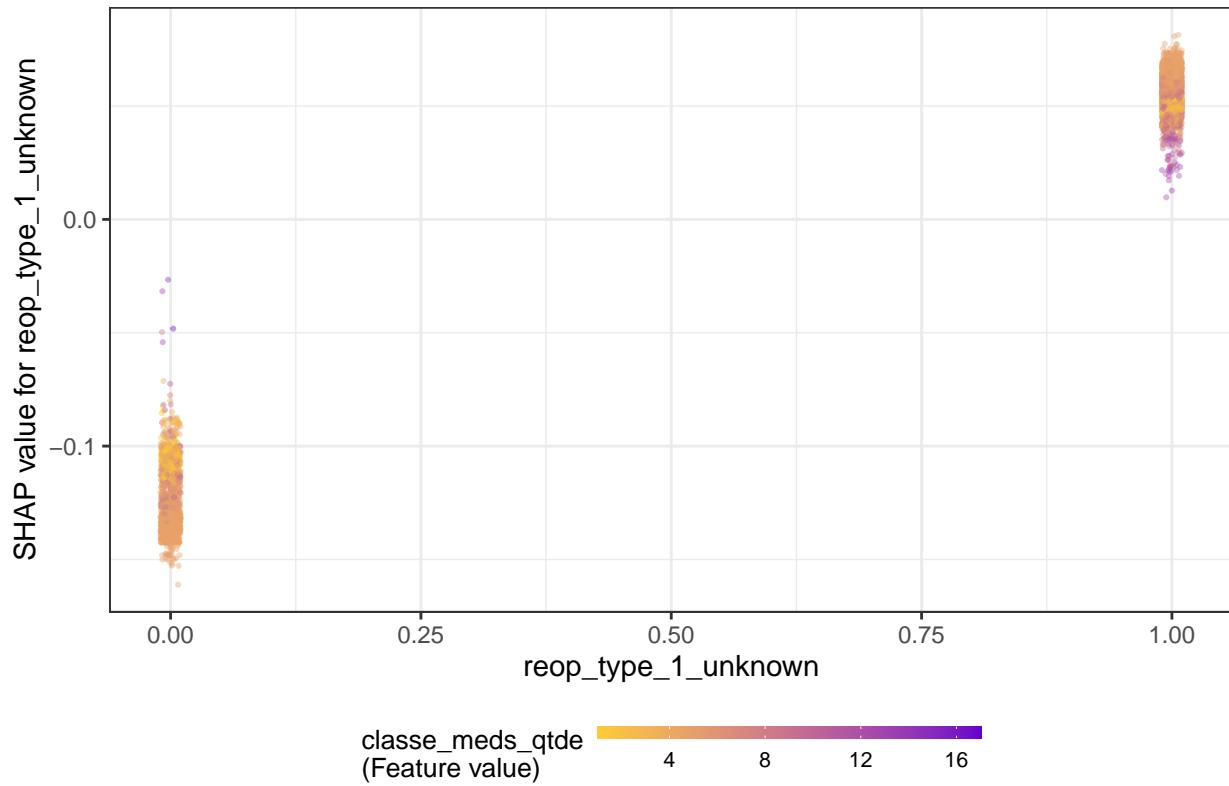
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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)
}

```



reop_type_1_unknown

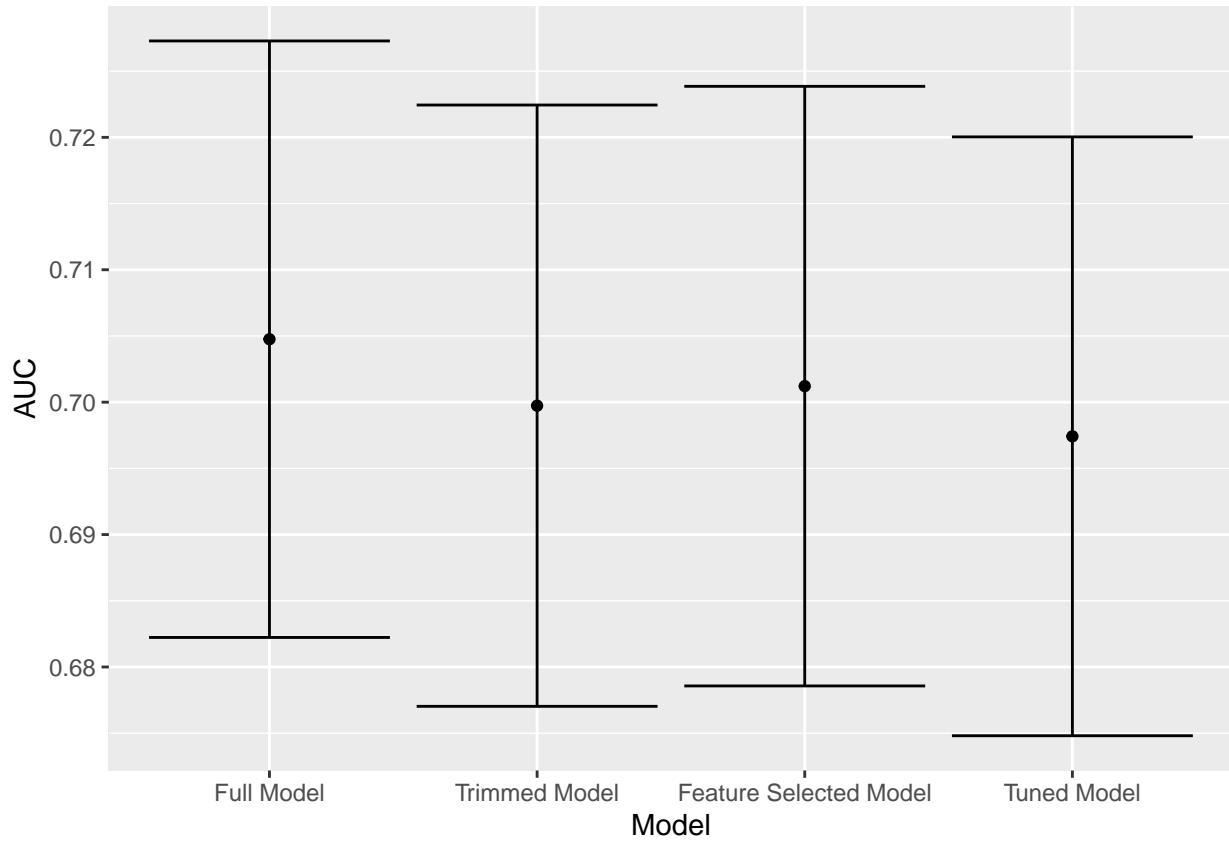


Minutes to run: 0

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))
```

Minutes to run: 0.002