Final Model

Eduardo Yuki Yada

Imports

```r
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

```r
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

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

df %>% dim
```

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

Minutes to run: 0.007

Eligible features

```r
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_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. education_level
## 02. underlying_heart_disease
## 03. heart_disease
## 04. nyha_basal
## 05. prior_mi
## 06. heart_failure
## 07. transplant
## 08. endocardites
## 09. hemodialysis
## 10. comorbidities_count
## 11. procedure_type_1
## 12. reop_type_1
## 13. procedure_type_new
## 14. cied_final_1
## 15. cied_final_group_1
## 16. admission_t0
## 17. admission_pre_t0_180d
## 18. icu_t0
## 19. dialysis_t0
## 20. disch_outcomes_t0
## 21. admission_t0_emergency
## 22. aco
## 23. antiarritmico
## 24. betabloqueador
## 25. ieca_bra
## 26. dva
## 27. digoxina
## 28. estatina
## 29. diuretico
## 30. vasodilatador
## 31. insuf_cardiaca
## 32. espironolactona
## 33. bloq_calcio
## 34. antiplaquetario_ev
## 35. insulina
## 36. anticonvulsivante
## 37. psicofarmacos
## 38. antifungico
## 39. antiviral
## 40. classe_meds_qtde
## 41. meds_cardiovasc_qtde
## 42. meds_antimicrobianos
## 43. cec
## 44. transplante_cardiaco
## 45. outros_proced_cirurgicos
## 46. icp
## 47. intervencao_cv
## 48. cateterismo
## 49. eletrofisiologia
## 50. cateter_venoso_central
## 51. proced_invasivos_qtde
## 52. cve_desf
## 53. transfusao
## 54. equipe_multiprof
## 55. ecg
## 56. holter
## 57. tilt_teste
## 58. metodos_graficos_qtde
## 59. laboratorio
## 60. cultura
## 61. analises_clinicas_qtde
## 62. citologia
## 63. biopsia
## 64. histopatologia_qtde
## 65. angio_rm
## 66. angio_tc
## 67. cintilografia
## 68. ecocardiograma
## 69. endoscopia
## 70. flebografia
## 71. pet_ct
## 72. ultrassom
## 73. tomografia
## 74. radiografia
## 75. ressonancia
## 76. exames_imagem_qtde
## 77. bic
## 78. mpp
```

Minutes to run: 0

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
```

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

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

Minutes to run: 0.004

Global parameters

```r
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

```r
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: 0.684"
```

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

```
## [1] "Full Model Test AUC: 0.688"
```

Minutes to run: 0.2

Features with zero importance on the initial model:

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

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

```
## 01. education_level
## 02. heart_disease
## 03. transplant
## 04. endocardites
## 05. hemodialysis
## 06. dialysis_t0
## 07. disch_outcomes_t0
## 08. antiviral
## 09. cec
## 10. intervencao_cv
## 11. transfusao
## 12. tilt_teste
## 13. citologia
## 14. biopsia
## 15. angio_rm
## 16. endoscopia
## 17. pet_ct
```

Minutes to run: 0

```r
trimmed_features <- full_model$importance$Feature

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: 0.685"
```

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

```
## [1] "Trimmed Model Test AUC: 0.687"
```

Minutes to run: 0.183

```r
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] 60.000000000  0.001324373
## [1] 5.900000e+01 7.489589e-07
## [1] 58.0000000000 -0.0001925203
## [1] 5.700000e+01 8.378743e-04
## [1] 56.000000000  0.001117807
## [1] 55.000000000  0.001122421
## [1] 54.0000000000 -0.0001646392
## [1] 53.000000000  0.001817384
## [1] 52.000000000  0.001854988
## [1] 51.00000000  0.00197999
## [1] 50.0000000000 -0.0001322111
## [1] 49.000000000  0.003804764
## [1] 48.000000000  0.003741697
## [1] 47.00000000  0.00228927
## [1] 46.0000000  0.0010442
## [1] 45.000000000  0.003268508
## [1] 44.000000000  0.004814327
## [1] 43.00000000  0.00402861
## [1] 42.000000000  0.005607631
## [1] 41.000000000  0.004854522
## [1] 40.000000000  0.005778619
## [1] 39.000000000  0.005246672
## [1] 38.000000000  0.008529093
## [1] 37.000000000  0.005775771
## [1] 36.000000000  0.007268955
## [1] 35.000000000  0.009223764
## [1] 34.000000000  0.009395815
## [1] 33.00000000  0.01055508
```

```
selection_results
```

```
## # A tibble: 30 x 3
##    `Number of Features`   `AUC Loss` `Least Important Feature`
##                   <int>        <dbl> <chr>
##  1                   78  0           underlying_heart_disease
##  2                   61 -0.000390    cve_desf
##  3                   60  0.00132     antiplaquetario_ev
##  4                   59  0.000000749 angio_tc
##  5                   58 -0.000193    insulina
##  6                   57  0.000838    cateter_venoso_central
##  7                   56  0.00112     bloq_calcio
##  8                   55  0.00112     underlying_heart_disease
##  9                   54 -0.000165    nyha_basal
## 10                   53  0.00182     procedure_type_1
## # ... with 20 more rows
## # i Use `print(n = ...)` to see more rows
```

Minutes to run: 4.818

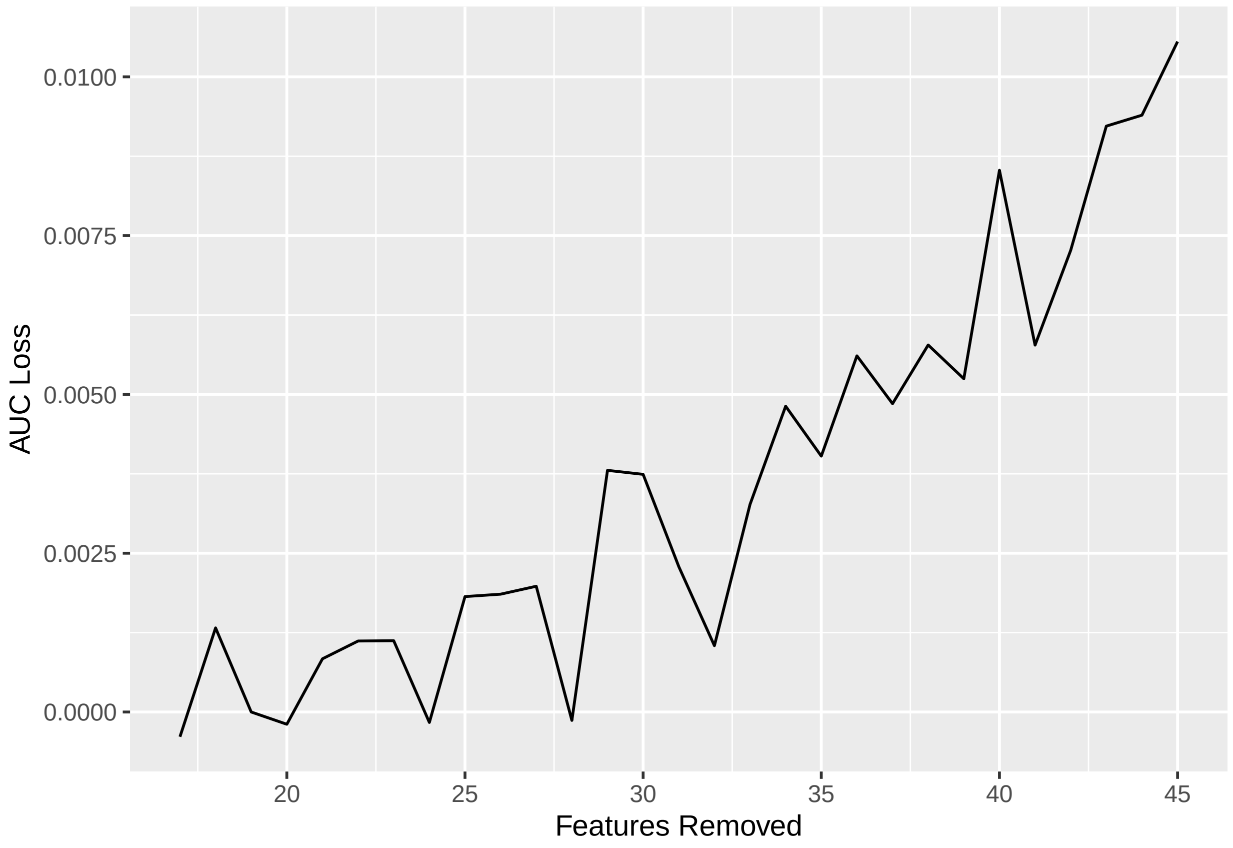
```
selected_features <- c(current_features, last_feature_droped)

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: 0.675"
```

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

```
## [1] "Trimmed Model Test AUC: 0.678"
```

```r
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()
```



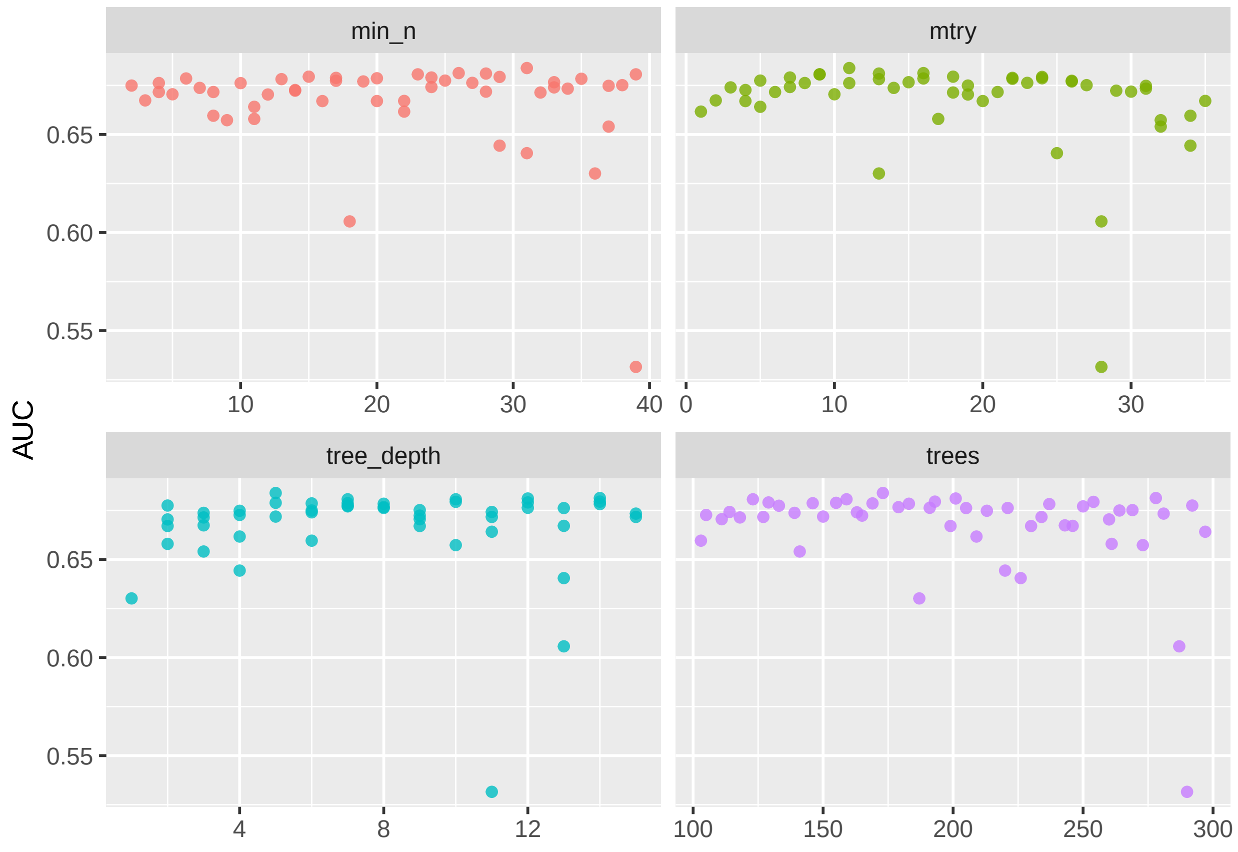
Minutes to run: 0.156

Hyperparameter tuning

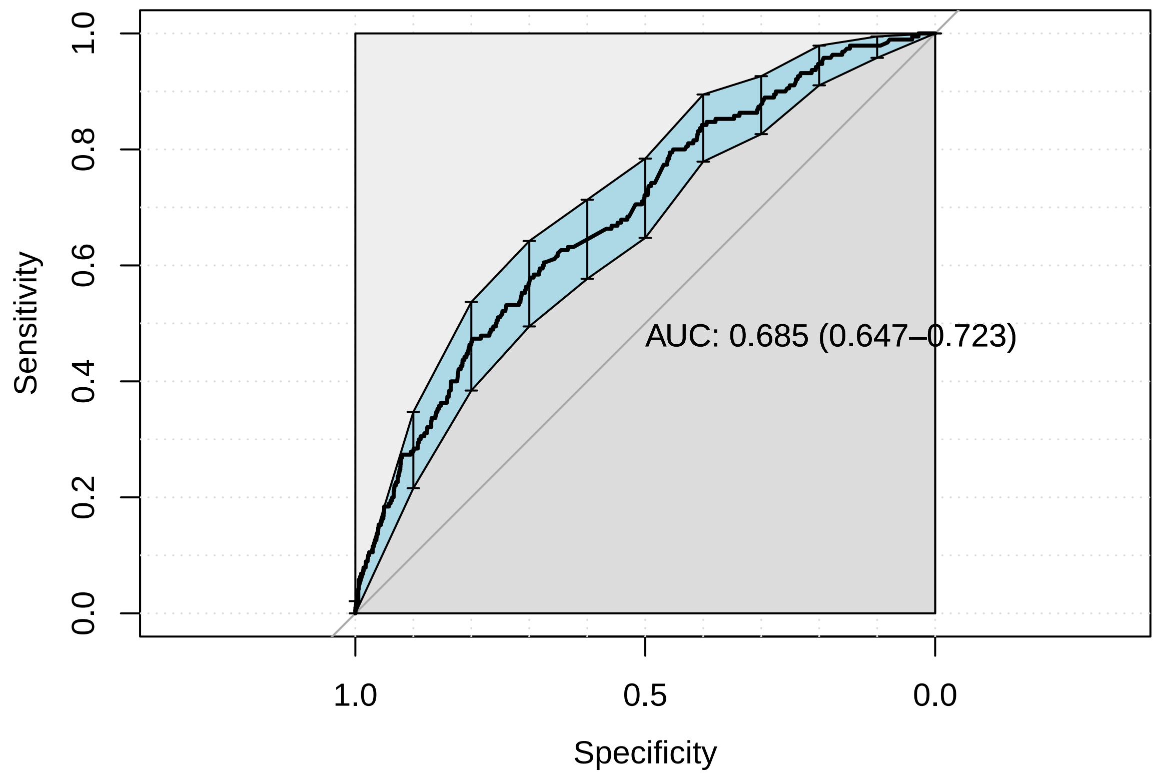
```r
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% dplyr::select(all_of(c(selected_features, outcome_column)))) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), 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_~1 loss_r~2 .metric .esti~3  mean
##   <int> <int> <int>      <int>    <dbl>    <dbl> <chr>   <chr>   <dbl>
## 1    11   173    31          5 1.52e-10 4.59e- 4 roc_auc binary  0.684
## 2    16   278    26         14 8.44e- 6 1.03e- 8 roc_auc binary  0.681
## 3    13   201    28         12 3.54e- 5 5.46e-10 roc_auc binary  0.681
## 4     9   159    39         10 1.30e- 7 1.02e- 4 roc_auc binary  0.681
## 5     9   123    23          7 7.14e-10 1.97e- 2 roc_auc binary  0.681
## # ... 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
```

```r
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)
```

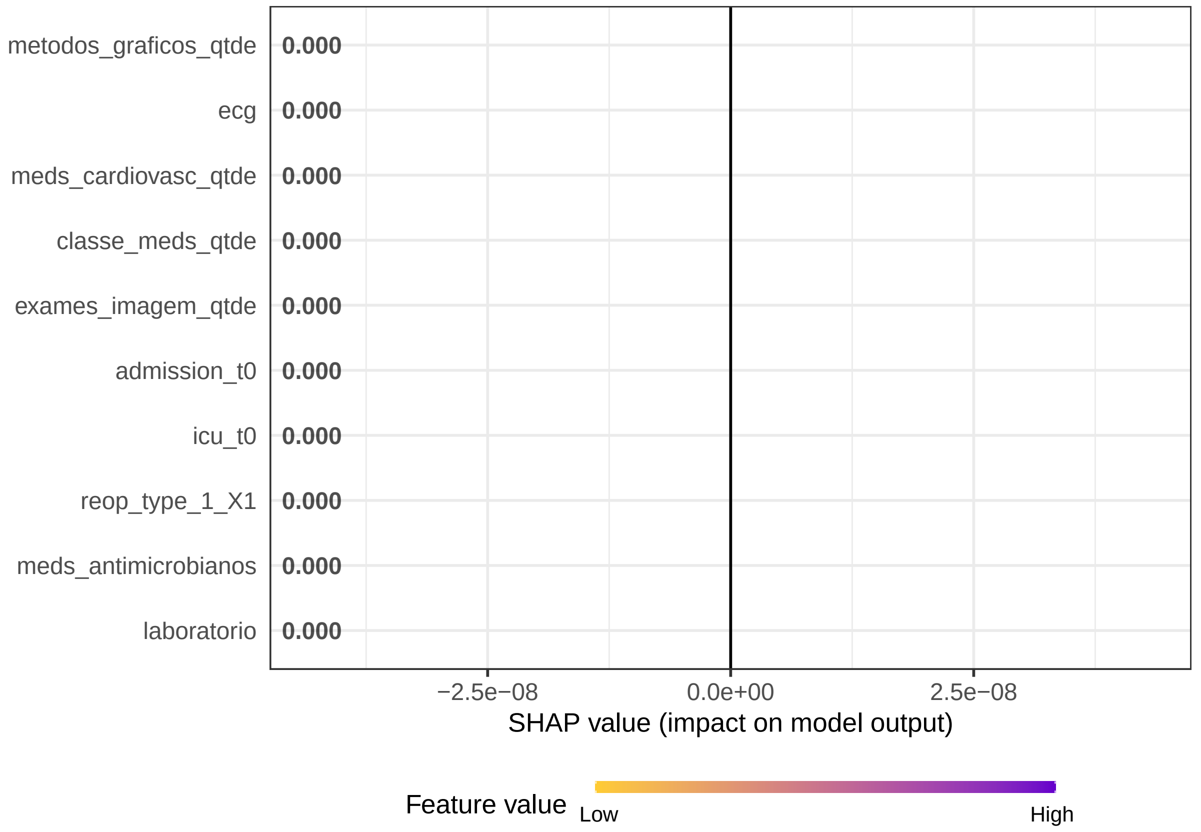


```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                      0 4540  190
##                      1    0    0
##
##                Accuracy : 0.9598
##                  95% CI : (0.9538, 0.9652)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 0.5193
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9598
##          Neg Pred Value :    NaN
##              Prevalence : 0.9598
##          Detection Rate : 0.9598
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

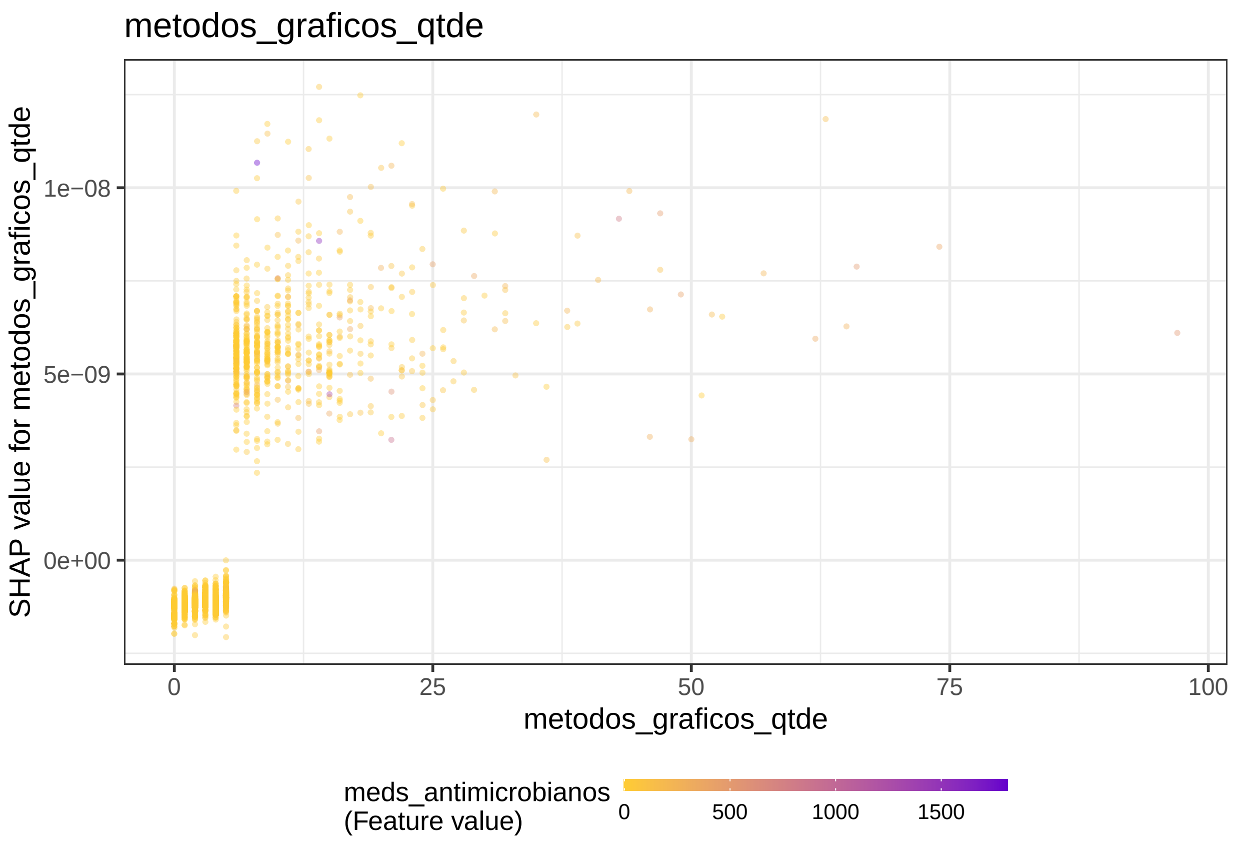
```
lightgbm_parameters <- lightgbm_tune %>%
  show_best("roc_auc", n=1) %>%
  select(trees, mtry, min_n, tree_depth, learn_rate, loss_reduction) %>%
  as.list
```

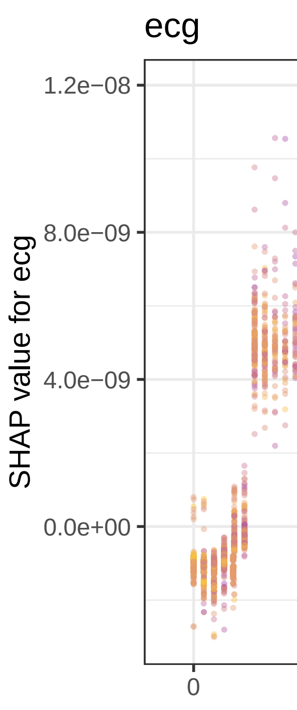
Minutes to run: 5.135

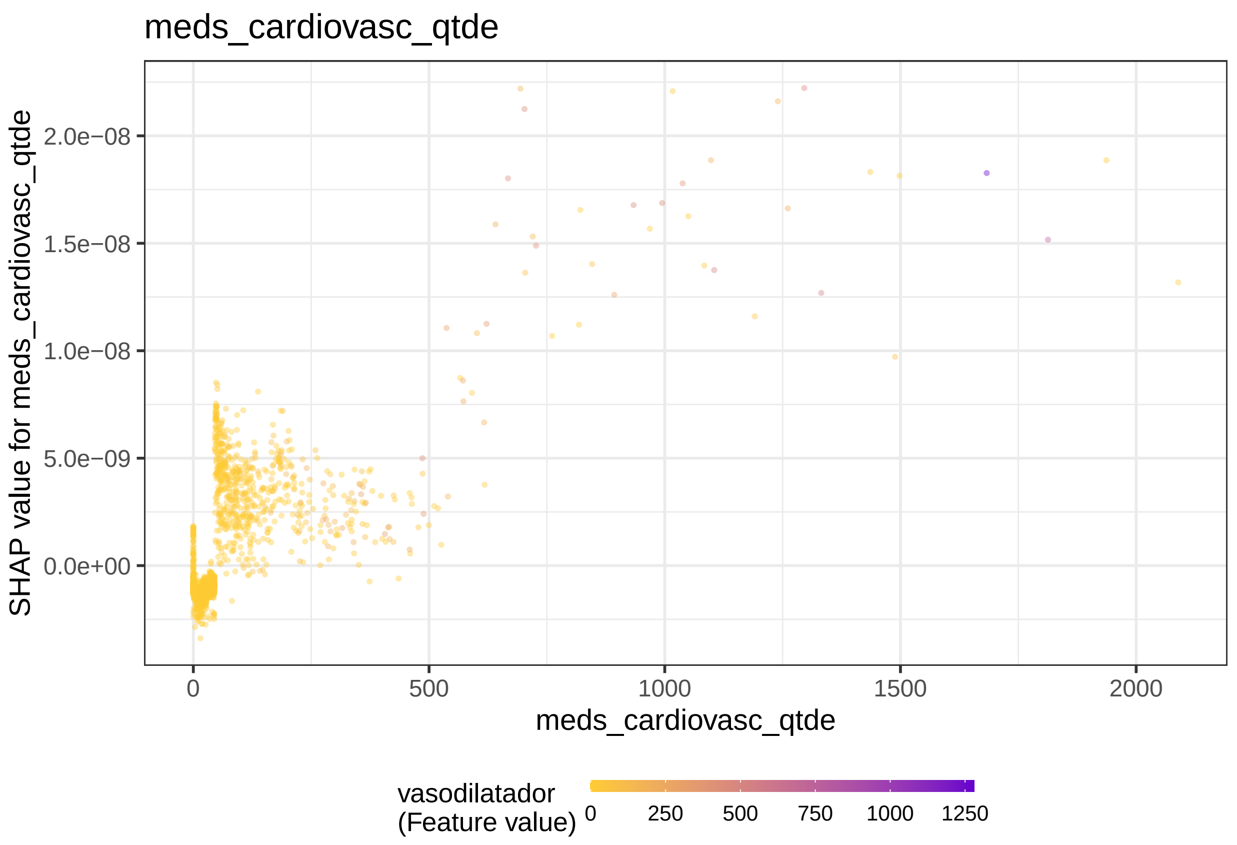
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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)
```

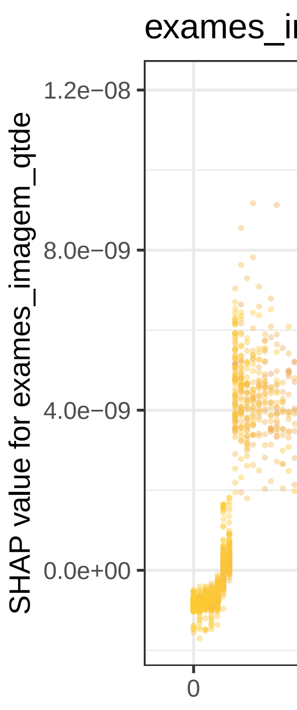


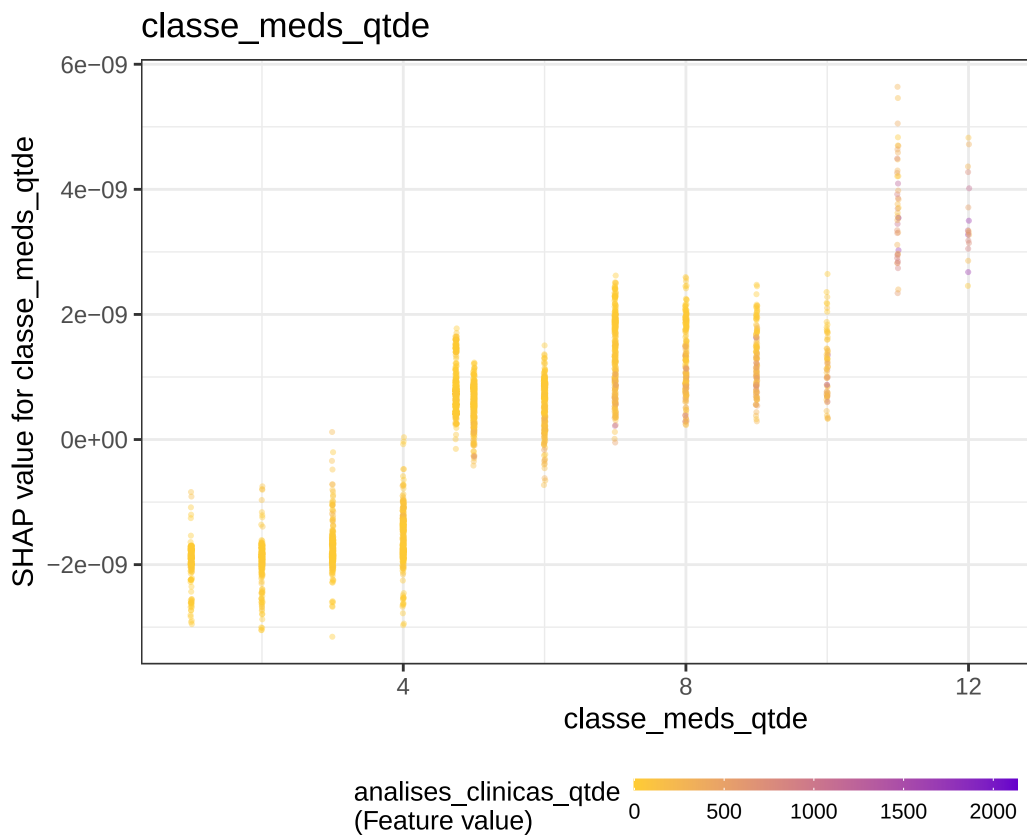
```
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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)
}
```


metodos_graficos_qtde
SHAP value for metodos_graficos_qtde
1e−08
5e−09
0e+00
0
25
50
75
100
metodos_graficos_qtde
meds_antimicrobianos (Feature value)
0
500
1000
1500


ecg
SHAP value for ecg
1.2e−08
8.0e−09
4.0e−09
0.0e+00
0


meds_cardiovasc_qtde
SHAP value for meds_cardiovasc_qtde
2.0e−08
1.5e−08
1.0e−08
5.0e−09
0.0e+00
0
500
1000
1500
2000
meds_cardiovasc_qtde
vasodilatador (Feature value)
0
250
500
750
1000
1250


exames_i
SHAP value for exames_imagem_qtde
1.2e−08
8.0e−09
4.0e−09
0.0e+00
0

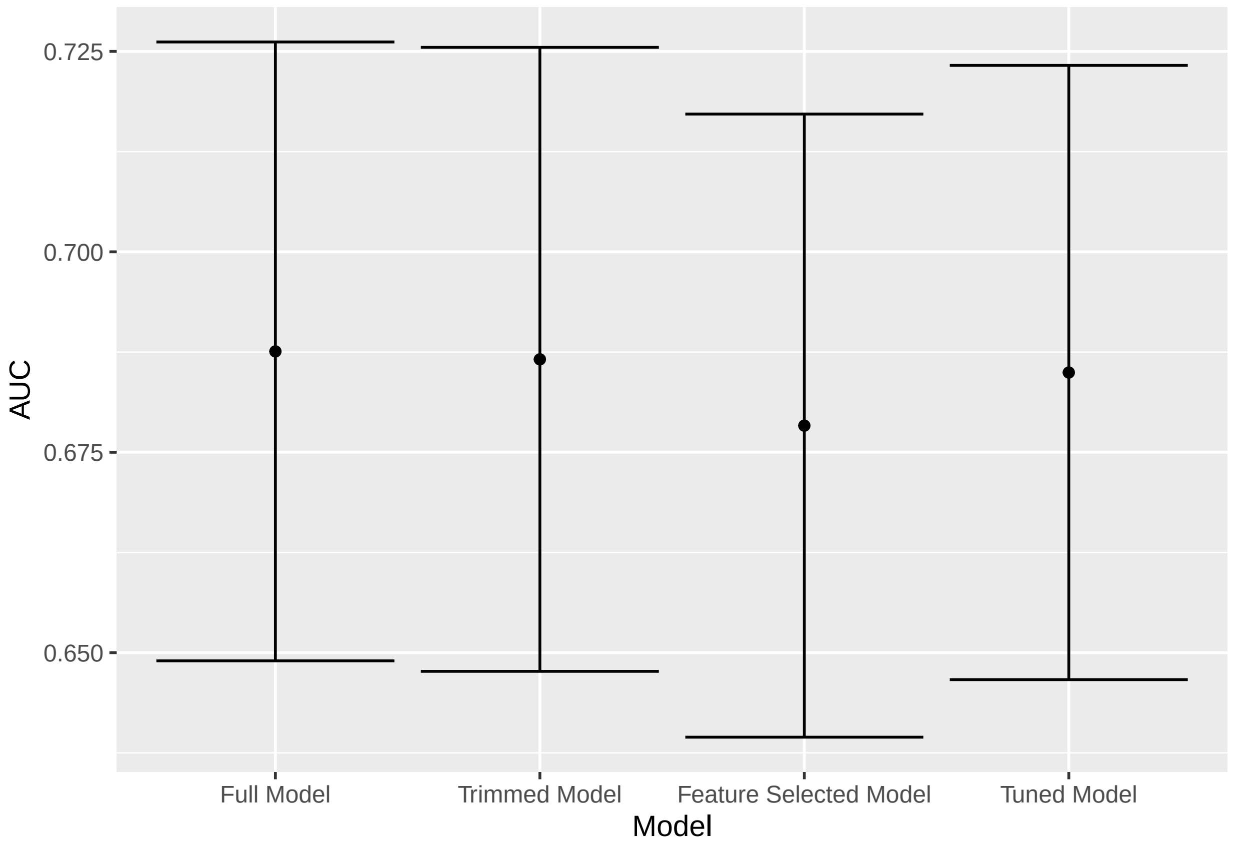


Minutes to run: 0.077

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))
```

Minutes to run: 0.002