

# Final Model - readmission\_1year

Eduardo Yuki Yada

## Global parameters

```
k <- 5 # Number of folds for cross validation
grid_size <- 30 # Number of parameter combination to tune on each model
max_auc_loss <- 0.01 # Max accepted loss of AUC for reducing num of features
```

## Imports

```
library(tidyverse)
library(yaml)
library(tidymodels)
library(usemodels)
library(vip)
library(kableExtra)
library(SHAPforxgboost)
library(xgboost)
library(Matrix)
library(mltools)
library(bonsai)
library(lightgbm)
library(pROC)
library(caret)
library(themis)

source("aux_functions.R")

select <- dplyr::select
```

## 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

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

dir.create(file.path("./auxiliar/final_model/hyperparameters/"),
          showWarnings = FALSE,
          recursive = TRUE)

dir.create(file.path("./auxiliar/final_model/performance/"),
          showWarnings = FALSE,
          recursive = TRUE)
```

```
dir.create(file.path("./auxiliar/final_model/selected_features/"),
          showWarnings = FALSE,
          recursive = TRUE)
```

## Eligible features

```
cat_features_list = readRDS(sprintf(
  "./auxiliar/significant_columns/categorical_%s.rds",
  outcome_column
))

num_features_list = readRDS(sprintf(
  "./auxiliar/significant_columns/numerical_%s.rds",
  outcome_column
))

features_list = c(cat_features_list, num_features_list)

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

exception_columns <- c('death_intraop', 'death_intraop_1', 'disch_outcomes_t0')

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,
                      'radiografia', # com exames_imagem_qtde
                      'ecg' # com metodos_graficos_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)
}
```

Starting features:

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

1. sex
2. age
3. race
4. education\_level
5. patient\_state
6. underlying\_heart\_disease
7. heart\_disease
8. nyha\_basal
9. prior\_mi
10. heart\_failure
11. af
12. cardiac\_arrest

13. transplant  
14. valvopathy  
15. endocarditis  
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\_pre\_t0\_count  
27. admission\_pre\_t0\_180d  
28. year\_adm\_t0  
29. icu\_t0  
30. dialysis\_t0  
31. admission\_t0\_emergency  
32. aco  
33. antiarritmico  
34. betabloqueador  
35. ieca\_bra  
36. dva  
37. digoxina  
38. estatina  
39. diuretico  
40. vasodilatador  
41. insuf\_cardiaca  
42. espironolactona  
43. bloq\_calcio  
44. antiplaquetario\_ev  
45. insulina  
46. anticonvulsivante  
47. psicofarmacos  
48. antifungico  
49. antiviral  
50. antiretroviral  
51. classe\_meds\_qtde  
52. meds\_cardiovasc\_qtde  
53. meds\_antimicrobianos  
54. ventilacao\_mecanica  
55. cec  
56. transplante\_cardiaco  
57. cir\_toracica  
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. holter  
71. teste\_esforco  
72. espiro\_ergoespiro  
73. tilt\_teste

74. metodos\_graficos\_qtde  
 75. laboratorio  
 76. cultura  
 77. analises\_clinicas\_qtde  
 78. citologia  
 79. biopsia  
 80. histopatologia\_qtde  
 81. angio\_rm  
 82. angio\_tc  
 83. aortografia  
 84. arteriografia  
 85. cintilografia  
 86. ecocardiograma  
 87. endoscopia  
 88. flebografia  
 89. pet\_ct  
 90. ultrassom  
 91. tomografia  
 92. ressonancia  
 93. exames\_imagem\_qtde  
 94. dieta\_parenteral  
 95. bic  
 96. mpp  
 97. hospital\_stay

## Train test split (70%/30%)

```

set.seed(42)

if (outcome_column == 'readmission_30d') {
  df_split <- readRDS("dataset/split_object.rds")
} else {
  df_split <- initial_split(df, prop = .7, strata = all_of(outcome_column))
}

df_train <- training(df_split) %>% select(all_of(c(features, outcome_column)))
df_test <- testing(df_split) %>% select(all_of(c(features, outcome_column)))

df_folds <- vfold_cv(df_train, v = k,
                      strata = all_of(outcome_column))

```

## Feature Selection

```

model_fit_wf <- function(df_train, 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")

  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)

cv_results <- collect_metrics(model_fit_rs) %>% filter(.metric == 'roc_auc')

return(
  list(
    cv_auc = cv_results$mean,
    cv_auc_std_err = cv_results$std_err,
    importance = feature_importance,
    auc = as.numeric(model_auc$auc),
    auc_lower = model_auc$ci[1],
    auc_upper = model_auc$ci[3]
  )
)
}
}

```

```

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

hyperparameters$sample_size <- 1

full_model <- model_fit_wf(df_train, features, outcome_column, hyperparameters)

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

## [1] "Full Model CV Train AUC: 0.715"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.714"

```

Features with zero importance on the initial model:

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

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

1. transplant
2. endocardites
3. renal\_failure
4. hemodialysis
5. copd
6. dialysis\_t0
7. antiplaquetario\_ev
8. antiviral
9. antiretroviral

```

10. cec
11. transplante_cardiaco
12. cir_toracica
13. icp
14. intervencao_cv
15. angioplastia
16. cve_desf
17. transfusao
18. teste_esforco
19. espiro_ergoespiro
20. tilt_teste
21. citologia
22. biopsia
23. angio_rm
24. angio_tc
25. aortografia
26. arteriografia
27. endoscopia
28. flebografia
29. pet_ct
30. dieta_parenteral
31. mpp

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

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

## [1] "Trimmed Model CV Train AUC: 0.712"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.710"

selection_results <- tibble::tribble(
  ~`Tested Feature`, ~`Dropped`, ~`Number of Features`, ~`CV AUC`, ~`CV AUC Std Error`, ~`Total AUC Loss`, ~`Instant AUC Loss`,
  'None', TRUE, length(features), full_model$cv_auc, full_model$cv_auc_std_err, 0, 0
)

whitelist <- c()

if (full_model$cv_auc - trimmed_model$cv_auc < max_auc_loss) {
  current_features <- trimmed_features
  current_model <- trimmed_model
  current_auc_loss <- full_model$cv_auc - current_model$cv_auc
  instant_auc_loss <- full_model$cv_auc - current_model$cv_auc

  selection_results <- selection_results %>%
    add_row(`Tested Feature` = 'All unimportant',
           `Dropped` = TRUE,
           `Number of Features` = length(trimmed_features),
           `CV AUC` = current_model$cv_auc,
           `CV AUC Std Error` = current_model$cv_auc_std_err,
           `Total AUC Loss` = current_auc_loss,
           `Instant AUC Loss` = instant_auc_loss)
} else {
  current_features <- features
  current_model <- full_model
  current_auc_loss <- 0
}

```

```

while (current_auc_loss < max_auc_loss | mean(current_features %in% whitelist) == 1) {
  current_least_important <-
    tail(setdiff(current_model$importance$Feature, whitelist), 1)
  test_features <-
    setdiff(current_features, current_least_important)
  hyperparameters$mtry <-
    min(hyperparameters$mtry, length(test_features))
  current_model <-
    model_fit_wf(df_train, test_features, outcome_column, hyperparameters)
  instant_auc_loss <-
    tail(selection_results %>% filter(Dropped) %>% .`CV AUC`, n = 1) - current_model$cv_auc
  current_auc_loss <- full_model$cv_auc - current_model$cv_auc

  if (instant_auc_loss < max_auc_loss / 5 &
      current_auc_loss < max_auc_loss) {
    dropped <- TRUE
    current_features <- test_features
  } else {
    dropped <- FALSE
    whitelist <- c(whitelist, current_least_important)
  }

  selection_results <- selection_results %>%
    add_row(
      `Tested Feature` = current_least_important,
      `Dropped` = dropped,
      `Number of Features` = length(test_features),
      `CV AUC` = current_model$cv_auc,
      `CV AUC Std Error` = current_model$cv_auc_std_err,
      `Total AUC Loss` = current_auc_loss,
      `Instant AUC Loss` = instant_auc_loss
    )
}

print(c(
  length(current_features),
  round(current_auc_loss, 4),
  round(instant_auc_loss, 4),
  current_least_important
))
}

## [1] "65"      "0.0037" "0.001"  "af"
## [1] "64"      "0.0024" "-0.0013" "insulina"
## [1] "63"      "0.0036" "0.0013" "race"
## [1] "62"      "0.0035" "-1e-04"   "cintilografia"
## [1] "61"      "0.0028" "-7e-04"   "antifungico"
## [1] "61"      "0.0051" "0.0023"   "cateter_venoso_central"
## [1] "60"      "0.0004" "0.0011"   "tomografia"
## [1] "60"      "0.0061" "0.0022"   "bic"
## [1] "59"      "0.0047" "8e-04"    "cateterismo"
## [1] "58"      "0.0051" "4e-04"    "cardiac_arrest"
## [1] "57"      "0.0059" "8e-04"    "interconsulta"
## [1] "56"      "0.0054" "-5e-04"
## [4] "underlying_heart_disease"
## [1] "55"      "0.0065" "0.0011"   "digoxina"
## [1] "54"      "0.0062" "-3e-04"   "diabetes"
## [1] "53"      "0.0073" "0.0011"   "heart_failure"
## [1] "52"      "0.0093" "0.002"    "ecocardiograma"
## [1] "52"      "0.0107" "0.0014"   "ultrassom"

```

```

selection_results %>%
  rename(Features = `Number of Features`) %>%
  niceFormatting(digits = 4, label = 1)

```

Table 1:

Tested Feature	Dropped	Features	CV AUC	CV AUC Std Error	Total AUC Loss	Instant AUC Loss
None	TRUE	97	0.7150	0.0068	0.0000	0.0000
All unimportant	TRUE	66	0.7123	0.0064	0.0026	0.0026
af	TRUE	65	0.7113	0.0059	0.0037	0.0010
insulina	TRUE	64	0.7126	0.0060	0.0024	-0.0013
race	TRUE	63	0.7114	0.0060	0.0036	0.0013
cintilografia	TRUE	62	0.7115	0.0054	0.0035	-0.0001
antifungico	TRUE	61	0.7121	0.0055	0.0028	-0.0007
cateter_venoso_central	FALSE	60	0.7099	0.0059	0.0051	0.0023
tomografia	TRUE	60	0.7110	0.0057	0.0040	0.0011
bic	FALSE	59	0.7088	0.0051	0.0061	0.0022
cateterismo	TRUE	59	0.7102	0.0051	0.0047	0.0008
cardiac_arrest	TRUE	58	0.7098	0.0050	0.0051	0.0004
interconsulta	TRUE	57	0.7091	0.0056	0.0059	0.0008
underlying_heart_disease	TRUE	56	0.7096	0.0053	0.0054	-0.0005
digoxina	TRUE	55	0.7085	0.0052	0.0065	0.0011
diabetes	TRUE	54	0.7088	0.0052	0.0062	-0.0003
heart_failure	TRUE	53	0.7076	0.0050	0.0073	0.0011
ecocardiograma	TRUE	52	0.7057	0.0052	0.0093	0.0020
ultrassom	FALSE	51	0.7042	0.0067	0.0107	0.0014

```

if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_dropped)
} else {
  selected_features <- current_features
}

con <- file(sprintf('./auxiliar/final_model/selected_features/%s.yaml', outcome_column), "w")
write_yaml(selected_features, con)
close(con)

feature_selected_model <- model_fit_wf(df_train, selected_features,
                                         outcome_column, hyperparameters)

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

## [1] "Selected Model CV Train AUC: 0.707"
sprintf('Selected Model Test AUC: %.3f', feature_selected_model$auc)

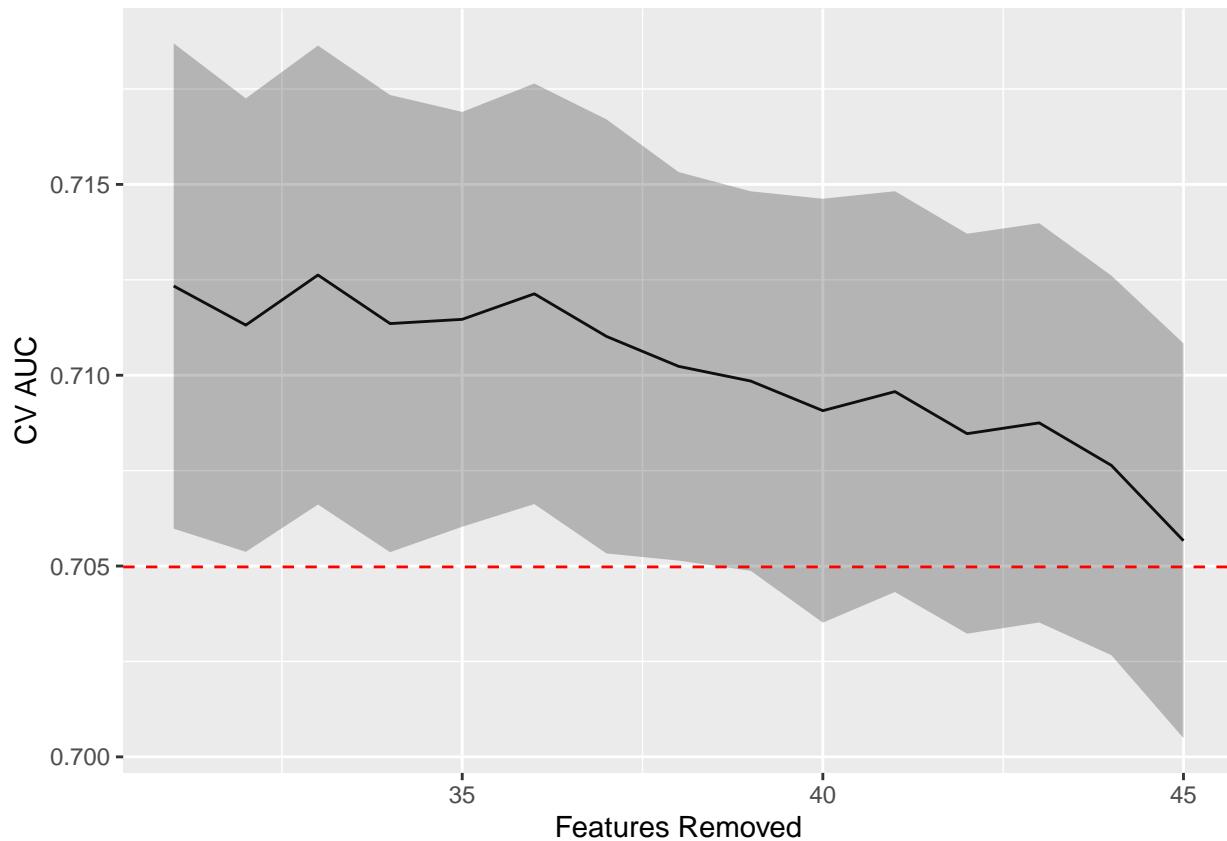
## [1] "Selected Model Test AUC: 0.707"

selection_results <- selection_results %>%
  filter(`Number of Features` < length(features)) %>%
  mutate(`Features Removed` = length(features) - `Number of Features`,
         `CV AUC Low` = `CV AUC` - `CV AUC Std Error`,
         `CV AUC High` = `CV AUC` + `CV AUC Std Error`)

selection_results %>%
  filter(Dropped) %>%
  ggplot(aes(x = `Features Removed`, y = `CV AUC`,
             ymin = `CV AUC Low`, ymax = `CV AUC High`)) +
  geom_line() +
  geom_ribbon(alpha = .3) +

```

```
geom_hline(yintercept = full_model$cv_auc - max_auc_loss,  
           linetype = "dashed", color = "red")
```



## Hyperparameter tuning

Selected features:

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

1. hospital\_stay
2. admission\_pre\_t0\_count
3. meds\_cardiovasc\_qtde
4. vasodilatador
5. admission\_pre\_t0\_180d
6. icu\_t0
7. classe\_meds\_qtde
8. diuretico
9. year\_adm\_t0
10. antiaritmico
11. cied\_final\_1
12. equipe\_multiprof
13. age
14. metodos\_graficos\_qtde
15. reop\_type\_1
16. admission\_t0\_emergency
17. ieca\_bra
18. insuf\_cardiaca
19. espironolactona
20. laboratorio
21. bloq\_calcio
22. meds\_antimicrobianos
23. cied\_final\_group\_1
24. procedure\_type\_new

```

25. estatina
26. dva
27. education_level
28. psicofarmacos
29. analises_clinicas_qtde
30. procedure_type_1
31. exames_imagem_qtde
32. proced_invasivos_qtde
33. sex
34. patient_state
35. aco
36. cultura
37. comorbidities_count
38. holter
39. betabloqueador
40. ultrassom
41. nyha_basal
42. prior_mi
43. valvopathy
44. bic
45. histopatologia_qtde
46. outros_proced_cirurgicos
47. ressonancia
48. cateter_venoso_central
49. ventilacao_mecanica
50. anticonvulsivante
51. eletrofisiologia
52. heart_disease

```

## Standard

```

lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% 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())

lightgbm_smote_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% 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()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_smote(!!sym(outcome_column))

lightgbm_upsample_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% 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()) %>%
  step_upsample(!!sym(outcome_column))

lightgbm_tuning <- function(recipe) {
  lightgbm_spec <- boost_tree(

```

```

mtry = tune(),
trees = tune(),
min_n = tune(),
tree_depth = tune(),
learn_rate = tune(),
loss_reduction = tune(),
sample_size = 1.0
) %>%
set_engine("lightgbm") %>%
set_mode("classification")

lightgbm_grid <- grid_latin_hypocube(
  mtry(range = c(1L, length(selected_features))),
  trees(range = c(100L, 300L)),
  min_n(),
  tree_depth(),
  learn_rate(),
  loss_reduction(),
  size = grid_size
)

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

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

lightgbm_tune %>%
  show_best("roc_auc") %>%
  niceFormatting(digits = 5, label = 4)

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)

```

```

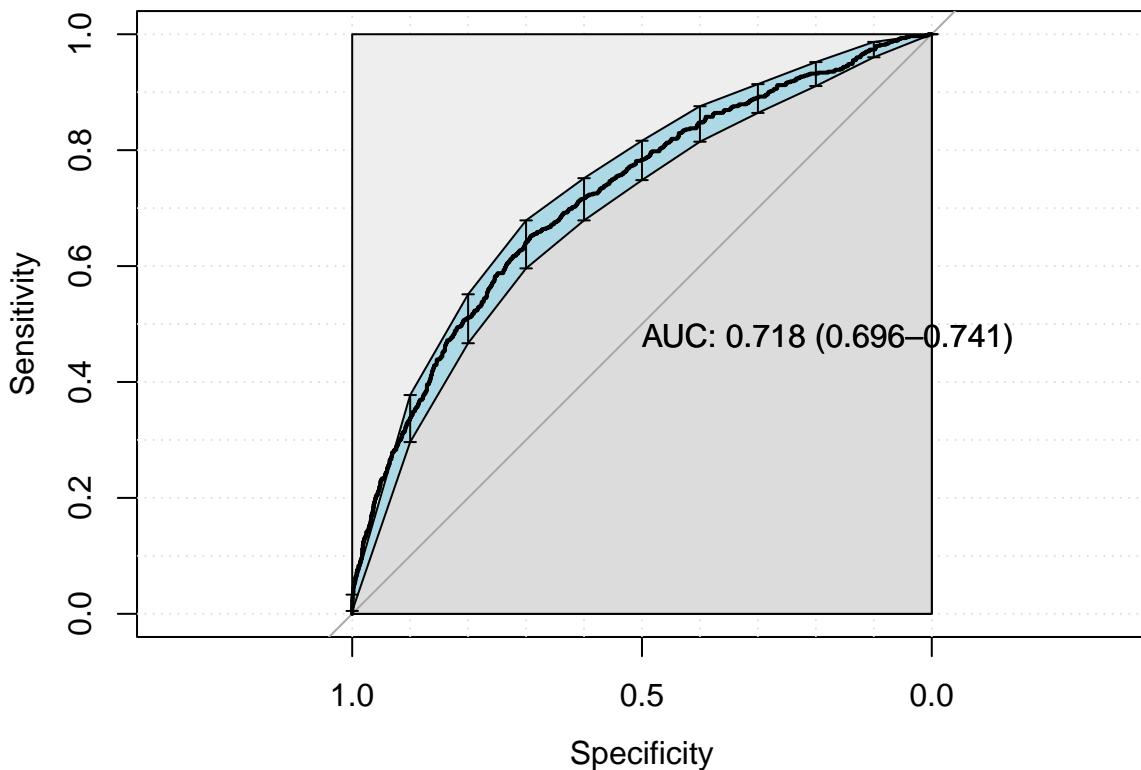
lightgbm_auc <- validation(final_lightgbm_fit, df_test)

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

return(list(auc = as.numeric(lightgbm_auc$auc),
            auc_lower = lightgbm_auc$ci[1],
            auc_upper = lightgbm_auc$ci[3],
            parameters = lightgbm_parameters,
            fit = final_lightgbm_fit))
}

standard_results <- lightgbm_tuning(lightgbm_recipe)

```



```

## [1] "Optimal Threshold: 0.12"
## Confusion Matrix and Statistics
##
##    reference
## data      0     1
##   0 2856  210
##   1 1271  394
##
##                  Accuracy : 0.687
##                             95% CI : (0.6735, 0.7002)
##   No Information Rate : 0.8723
##   P-Value [Acc > NIR] : 1
##
##                  Kappa : 0.1968
##
##   Mcnemar's Test P-Value : <2e-16
##
```

```

##           Sensitivity : 0.6920
##           Specificity : 0.6523
##           Pos Pred Value : 0.9315
##           Neg Pred Value : 0.2366
##           Prevalence : 0.8723
##           Detection Rate : 0.6037
##   Detection Prevalence : 0.6481
##           Balanced Accuracy : 0.6722
##
##           'Positive' Class : 0
##
# smote_results <- lightgbm_tuning(lightgbm_smote_recipe)
# upsample_results <- lightgbm_tuning(lightgbm_upsample_recipe)

final_lightgbm_fit <- standard_results$fit
lightgbm_parameters <- standard_results$parameters

# saveRDS(
#   lightgbm_parameters,
#   file = sprintf(
#     "./auxiliar/final_model/hyperparameters/lightgbm_%s.rds",
#     outcome_column
#   )
# )

```

## SHAP values

```

lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

trained_rec <- prep(lightgbm_recipe, training = df_train)

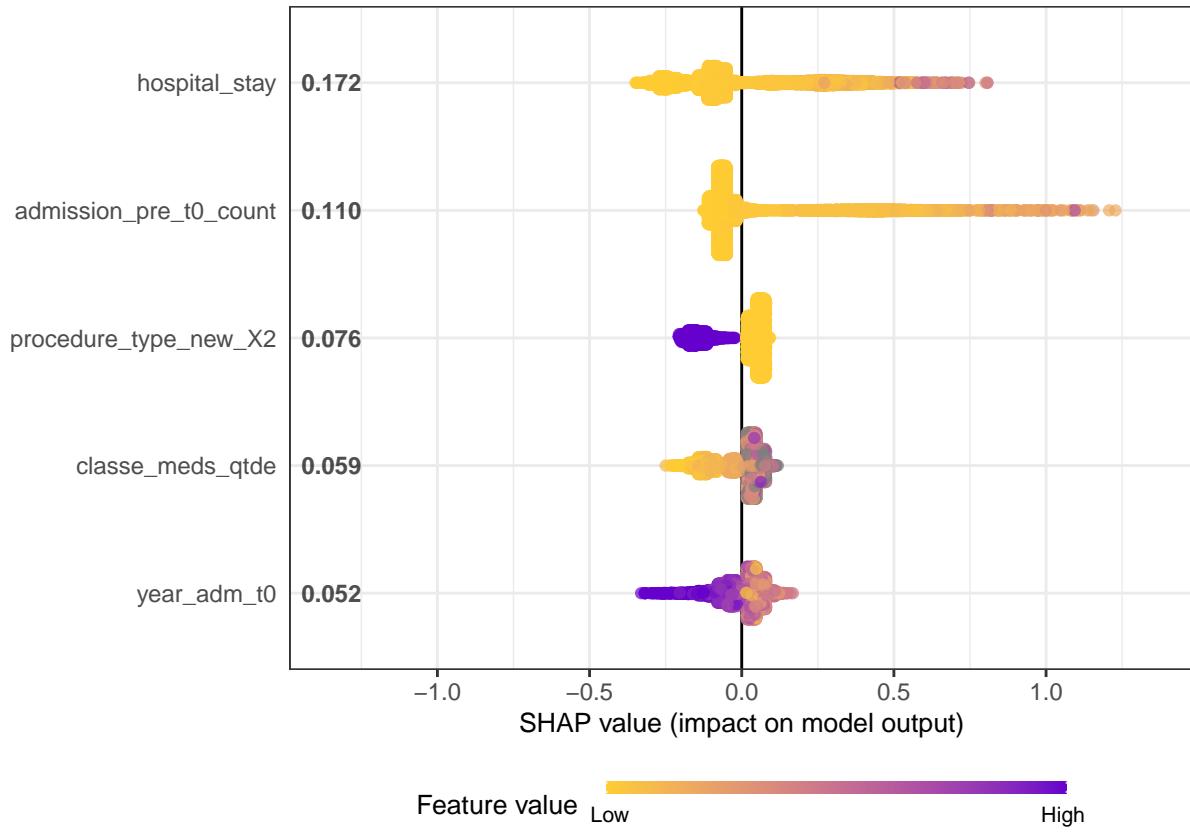
df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

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

n_plots <- min(5, length(selected_features))

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

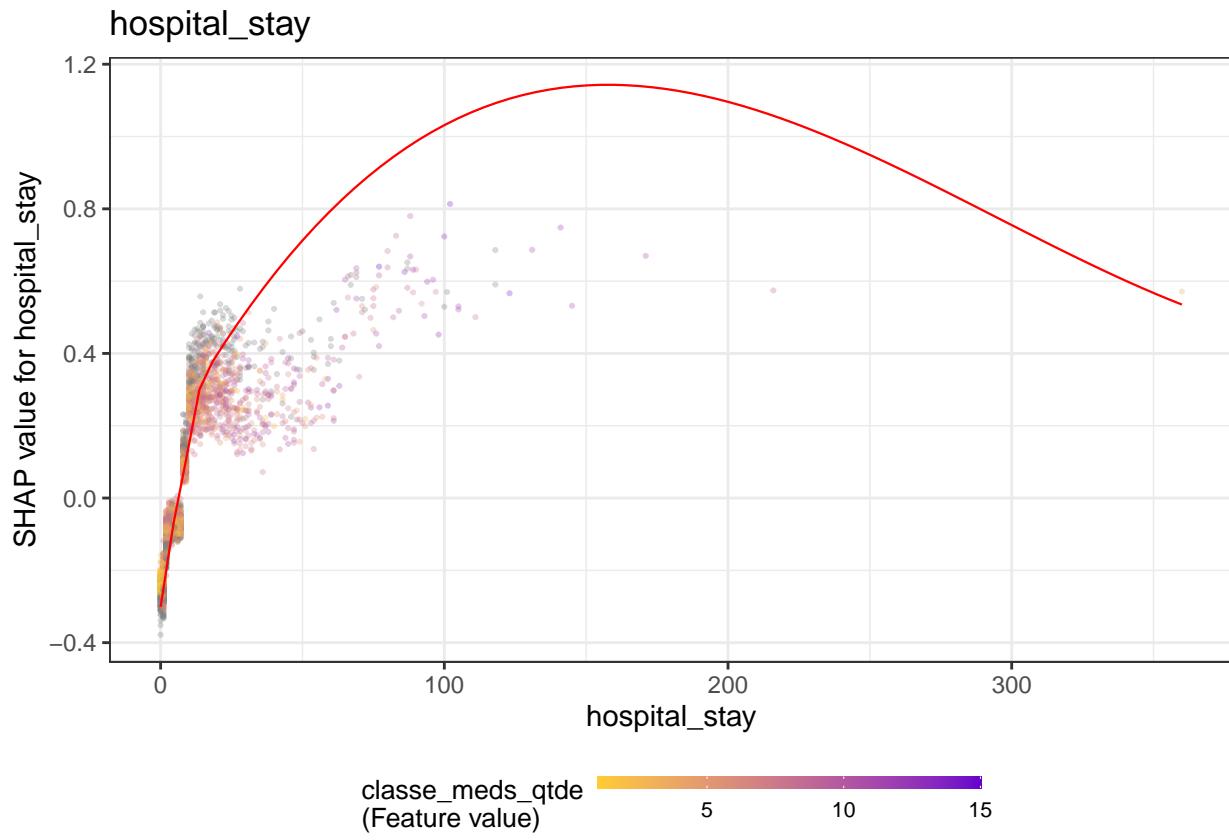
```



```
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

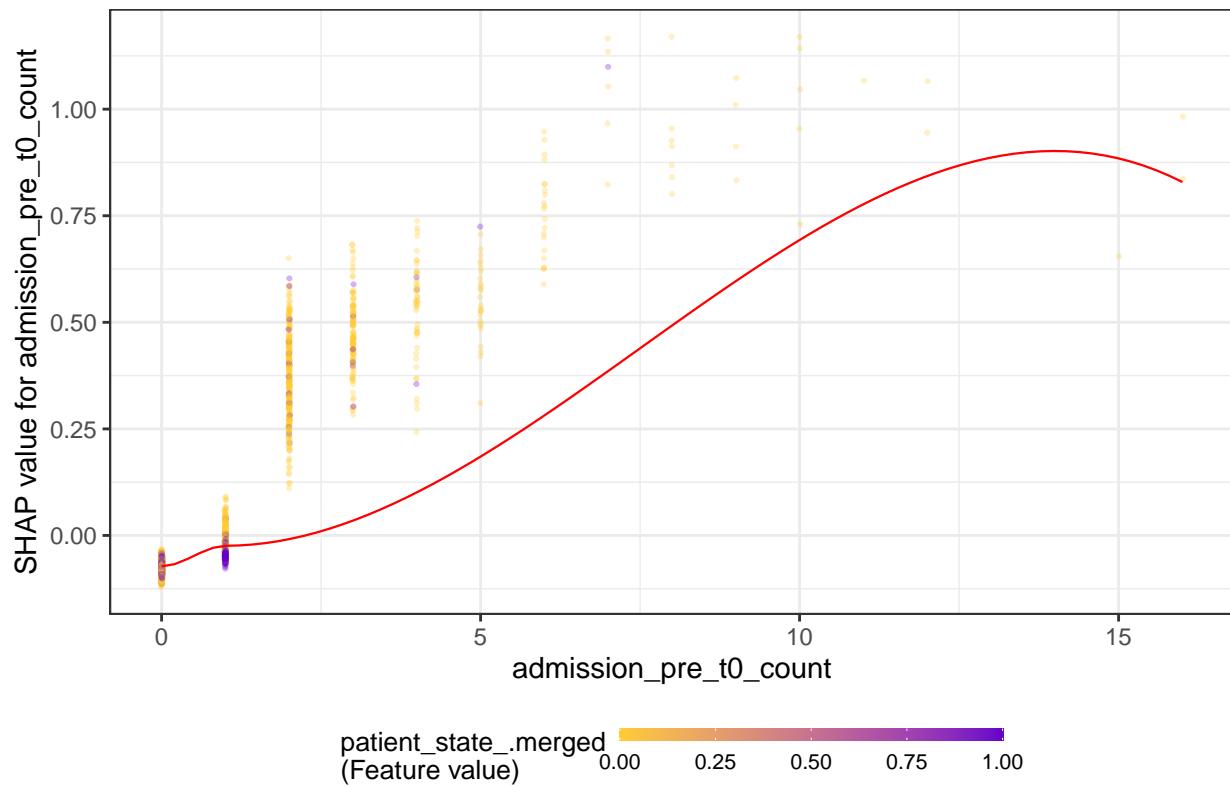
for (x in shap.importance(shap, names_only = TRUE)[1:n_plots]) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = TRUE,
    jitter_width = 0.01,
    alpha = 0.3
  ) +
    labs(title = x)
  print(p)
}

## `geom_smooth()` using formula 'y ~ x'
```



```
## `geom_smooth()` using formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -0.08
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.08
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 4.7124e-29
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1
```

## admission\_pre\_t0\_count

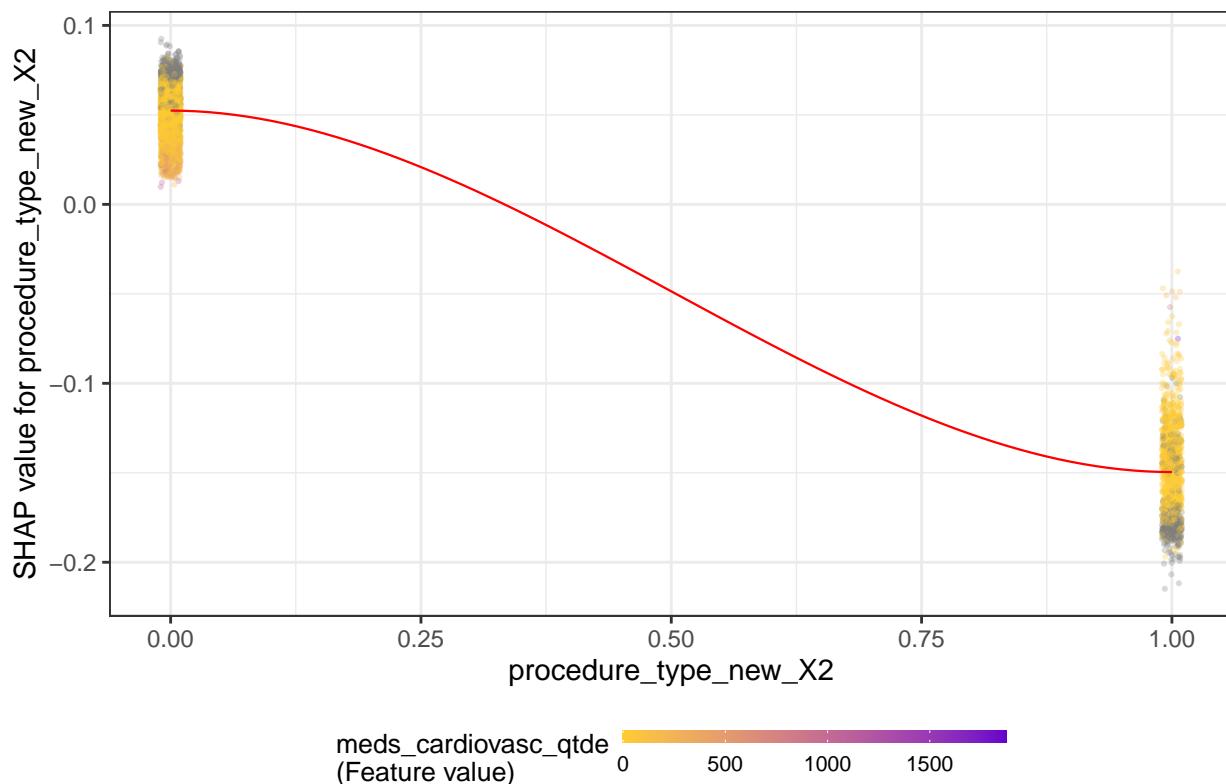


```

## `geom_smooth()` using formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -0.005
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.005
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 1.1644e-28
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1.01

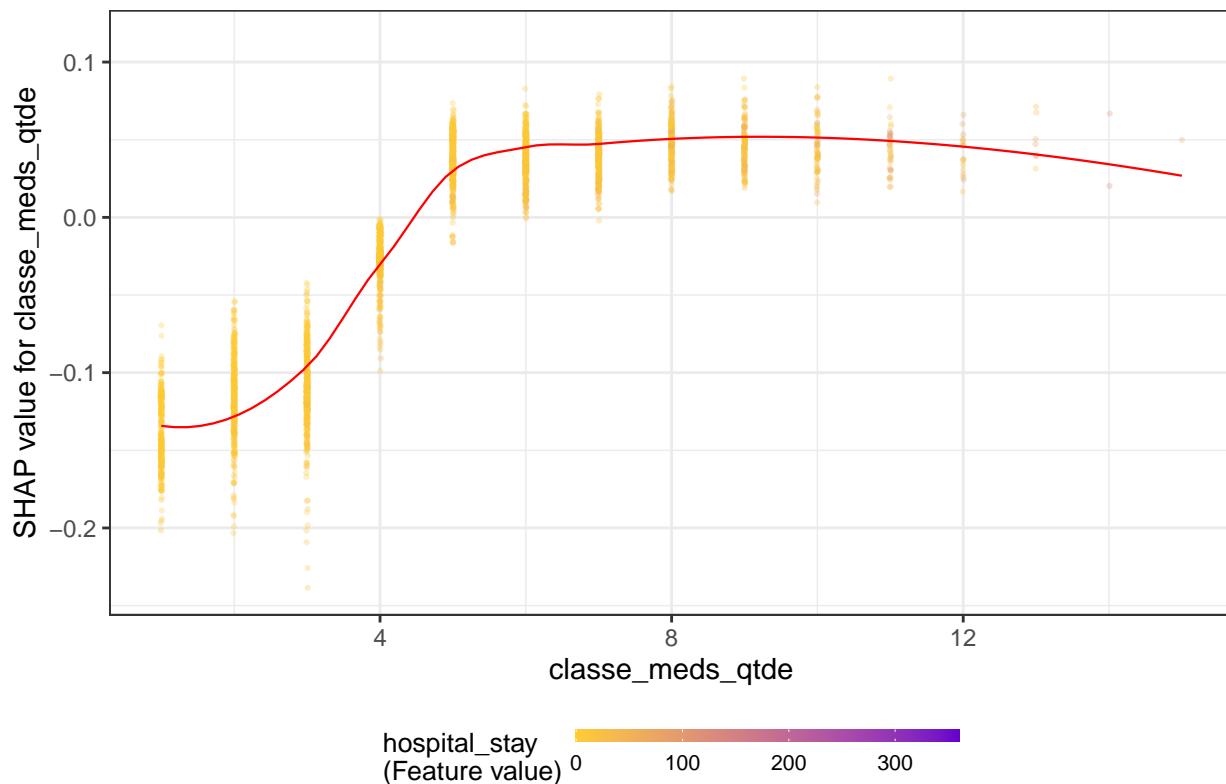
```

procedure\_type\_new\_X2



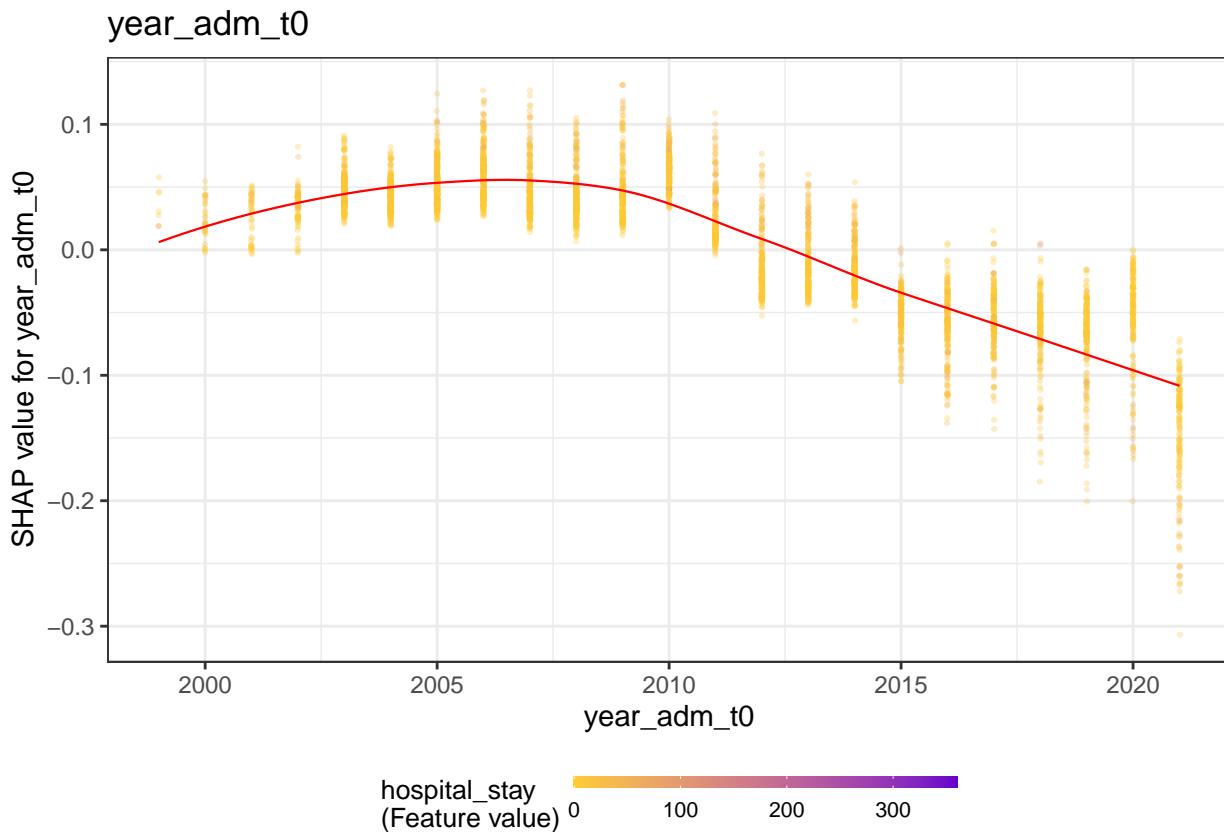
```
## `geom_smooth()` using formula 'y ~ x'  
## Warning: Removed 1507 rows containing non-finite values (stat_smooth).  
## Warning: Removed 1507 rows containing missing values (geom_point).
```

classe\_meds\_qtde



```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 5 rows containing non-finite values (stat_smooth).
## Warning: Removed 5 rows containing missing values (geom_point).
```

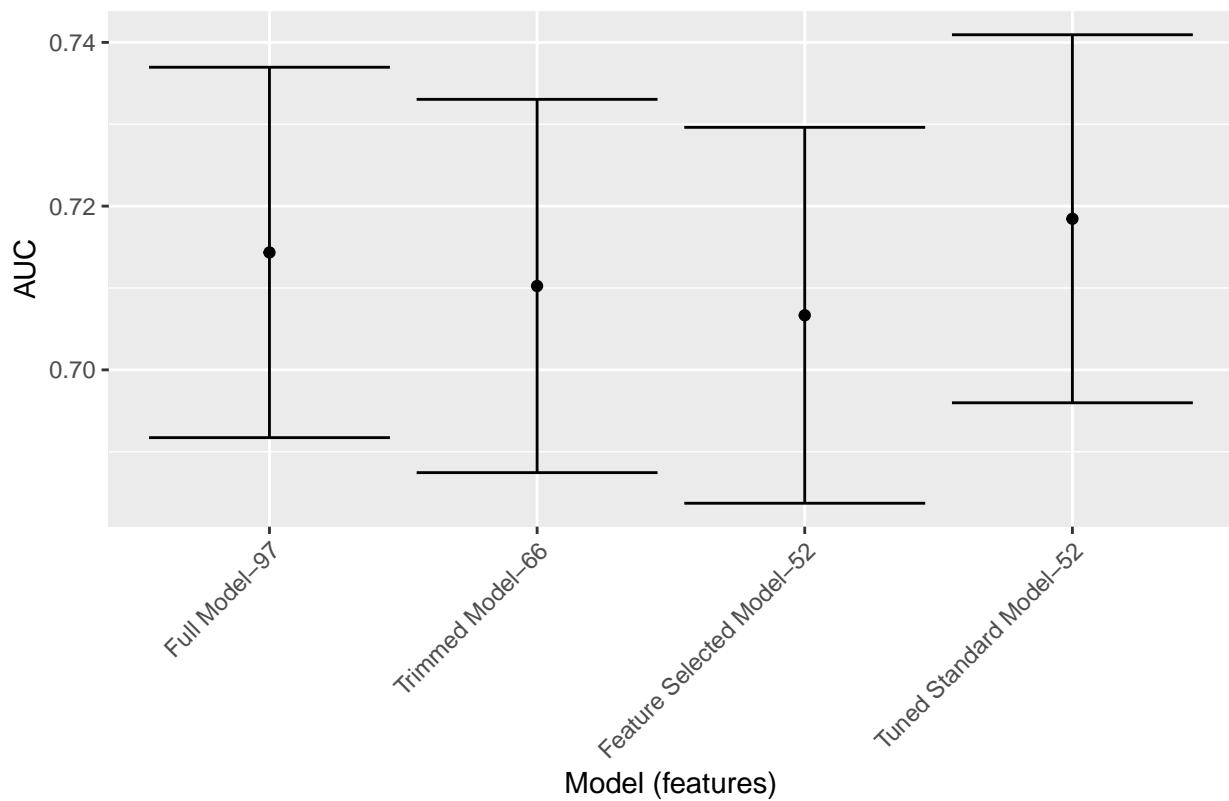


## Models Comparison

```
df_auc <- tibble::tribble(
  ~`Model`, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`, ~`Features`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper, length(features),
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper, length(trimmed_features),
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower, feature_selected_model$auc_upper, length(selected_features),
  'Tuned Standard Model', standard_results$auc, standard_results$auc_lower, standard_results$auc_upper, length(standard_features),
  # 'Tuned Smote Model', smote_results$auc, smote_results$auc_lower, smote_results$auc_upper, length(selected_features),
  # 'Tuned Upsample Model', upsample_results$auc, upsample_results$auc_lower, upsample_results$auc_upper, length(upsample_features)
) %>%
  mutate(Target = outcome_column,
    `Model (features)` = fct_reorder(paste0(Model, "-"), Features), -Features))

df_auc %>%
  ggplot(aes(
    x = `Model (features)` ,
    y = AUC,
    ymin = `Lower Limit` ,
    ymax = `Upper Limit` )
  ) +
  geom_point() +
  geom_errorbar() +
  labs(title = outcome_column) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

### readmission\_1year



```
saveRDS(df_auc, sprintf("./auxiliar/final_model/performance/%s.RData", outcome_column))
```