

# Final Model - death\_2year

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

```
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. education\_level
4. underlying\_heart\_disease
5. heart\_disease
6. nyha\_basal
7. hypertension
8. prior\_mi
9. 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. admission\_t0\_emergency  
30. aco  
31. antiarritmico  
32. ieca\_bra  
33. dva  
34. digoxina  
35. estatina  
36. diuretico  
37. vasodilatador  
38. insuf\_cardiaca  
39. espironolactona  
40. antiplaquetario\_ev  
41. insulina  
42. psicofarmacos  
43. antifungico  
44. antiviral  
45. classe\_meds\_qtde  
46. meds\_cardiovasc\_qtde  
47. meds\_antimicrobianos  
48. vni  
49. ventilacao\_mecanica  
50. transplante\_cardiaco  
51. outros\_proced\_cirurgicos  
52. icp  
53. angioplastia  
54. cateterismo  
55. cateter\_venoso\_central  
56. proced\_invasivos\_qtde  
57. transfusao  
58. interconsulta  
59. equipe\_multiprof  
60. holter  
61. teste\_esforco  
62. tilt\_teste  
63. metodos\_graficos\_qtde  
64. laboratorio  
65. cultura  
66. analises\_clinicas\_qtde  
67. citologia  
68. histopatologia\_qtde  
69. angio\_tc  
70. cintilografia  
71. ecocardiograma  
72. endoscopia  
73. flebografia  
74. pet\_ct  
75. ultrassom  
76. tomografia  
77. ressonancia  
78. exames\_imagem\_qtde  
79. bic  
80. 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: 0.762"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.773"

Features with zero importance on the initial model:
unimportant_features <- setdiff(features, full_model$importance$Feature)

unimportant_features %>%
  gluedown::md_order()

1. underlying_heart_disease
2. heart_disease
3. hemodialysis
4. dialysis_t0
5. antiplaquetario_ev
6. antiviral
7. vni
8. transplante_cardiaco
9. angioplastia
10. cateter_venoso_central
11. transfusao
12. teste_esforco
13. tilt_teste
14. citologia
15. histopatologia_qtde
16. endoscopia
17. pet_ct

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: 0.751"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.757"

selection_results <- tibble::tribble(
  ~`Number of Features`, ~`CV AUC`, ~`CV AUC Std Error`, ~`AUC Loss`, ~`Least Important Feature`,
  length(features), full_model$cv_auc, full_model$cv_auc_std_err, 0, tail(full_model$importance$Feature, 1)
)

```

```

if (full_model$cv_auc - trimmed_model$cv_auc < max_auc_loss) {
  current_features <- trimmed_features
  current_model <- trimmed_model
  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(trimmed_features),
            `CV AUC` = current_model$cv_auc,
            `CV AUC Std Error` = current_model$cv_auc_std_err,
            `AUC Loss` = current_auc_loss,
            `Least Important Feature` = current_least_important)
} else {
  current_features <- features
  current_model <- full_model
  current_least_important <- tail(current_model$importance$Feature, 1)
  current_auc_loss <- 0
}

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(df_train, 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),
            `CV AUC` = current_model$cv_auc,
            `CV AUC Std Error` = current_model$cv_auc_std_err,
            `AUC Loss` = current_auc_loss,
            `Least Important Feature` = current_least_important)

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

selection_results %>% niceFormatting(digits = 4, label = 1)

```

Table 1:

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
80	0.7621	0.0044	0.0000	copd
79	0.7610	0.0050	0.0011	cateter_venoso_central
78	0.7592	0.0047	0.0028	nyha_basal
77	0.7593	0.0042	0.0027	antifungico
76	0.7590	0.0044	0.0030	flebografia
75	0.7587	0.0044	0.0034	insulina
74	0.7595	0.0041	0.0026	cardiac_arrest
73	0.7589	0.0045	0.0031	underlying_heart_disease
72	0.7559	0.0044	0.0061	bic
71	0.7564	0.0037	0.0057	outros_proced_cirurgicos
70	0.7565	0.0041	0.0056	ressonancia
69	0.7554	0.0040	0.0066	procedure_type_new
68	0.7562	0.0033	0.0059	angio_tc
67	0.7552	0.0046	0.0069	holter
66	0.7543	0.0036	0.0078	reop_type_1

Table 1: (*continued*)

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
65	0.7522	0.0037	0.0099	endoscopia
64	0.7518	0.0037	0.0103	procedure_type_1

```

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)

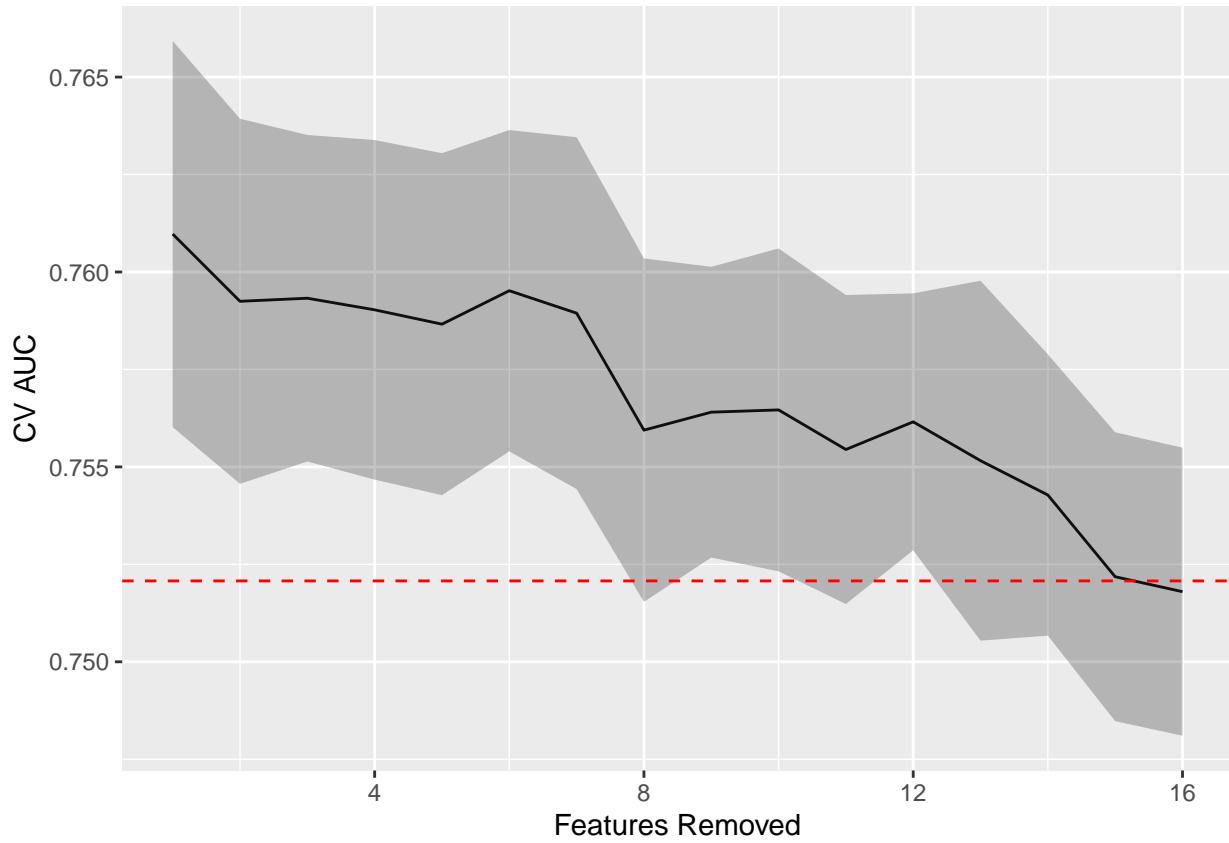
## [1] "Selected Model CV Train AUC: 0.753"
sprintf('Selected Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Selected Model Test AUC: 0.761"

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 %>%
  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")

```



```
# 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

Selected features:

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

1. sex
2. age
3. education\_level
4. heart\_disease
5. hypertension
6. prior\_mi
7. heart\_failure
8. af
9. valvopathy
10. diabetes
11. renal\_failure
12. hemodialysis
13. stroke
14. comorbidities\_count
15. procedure\_type\_1
16. cied\_final\_1
17. cied\_final\_group\_1
18. admission\_pre\_t0\_count
19. admission\_pre\_t0\_180d
20. year\_adm\_t0
21. icu\_t0

22. dialysis\_t0  
 23. admission\_t0\_emergency  
 24. aco  
 25. antiarritmico  
 26. ieca\_bra  
 27. dva  
 28. digoxina  
 29. estatina  
 30. diuretico  
 31. vasodilatador  
 32. insuf\_cardiaca  
 33. espironolactona  
 34. antiplaquetario\_ev  
 35. psicofarmacos  
 36. antiviral  
 37. classe\_meds\_qtde  
 38. meds\_cardiovasc\_qtde  
 39. meds\_antimicrobianos  
 40. vni  
 41. ventilacao\_mecanica  
 42. transplante\_cardiaco  
 43. icp  
 44. angioplastia  
 45. cateterismo  
 46. proced\_invasivos\_qtde  
 47. transfusao  
 48. interconsulta  
 49. equipe\_multiprof  
 50. teste\_esforco  
 51. tilt\_teste  
 52. metodos\_graficos\_qtde  
 53. laboratorio  
 54. cultura  
 55. analises\_clinicas\_qtde  
 56. citologia  
 57. histopatologia\_qtde  
 58. cintilografia  
 59. ecocardiograma  
 60. pet\_ct  
 61. ultrassom  
 62. tomografia  
 63. exames\_imagem\_qtde  
 64. hospital\_stay

## 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_hypercube(
    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)

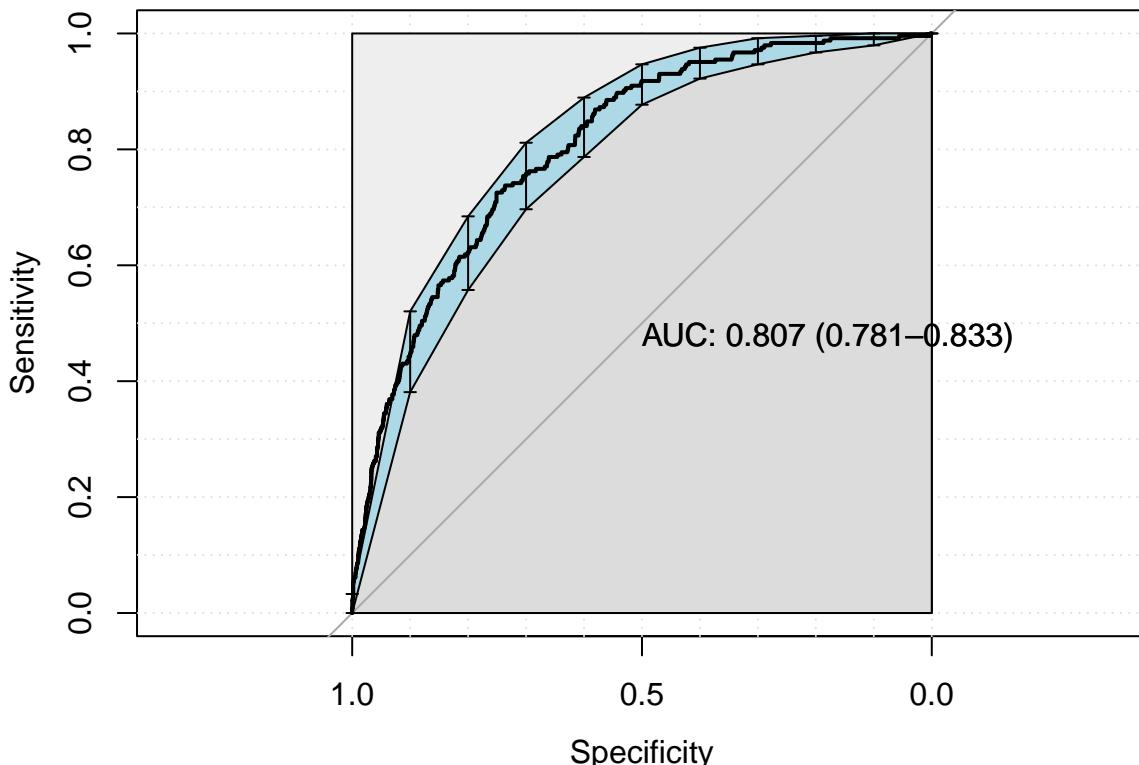
lightgbm_auc <- validation(final_lightgbm_fit, df_test)

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: 0.04"
## Confusion Matrix and Statistics

```

```

##      reference
## data    0     1
##      0 3369   67
##      1 1117  177
##
##                         Accuracy : 0.7497
##                         95% CI : (0.7371, 0.762)
##      No Information Rate : 0.9484
##      P-Value [Acc > NIR] : 1
##
##                         Kappa : 0.157
##
##      Mcnemar's Test P-Value : <2e-16
##
##      Sensitivity : 0.7510
##      Specificity : 0.7254
##      Pos Pred Value : 0.9805
##      Neg Pred Value : 0.1368
##      Prevalence : 0.9484
##      Detection Rate : 0.7123
##      Detection Prevalence : 0.7264
##      Balanced Accuracy : 0.7382
##
##      '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)

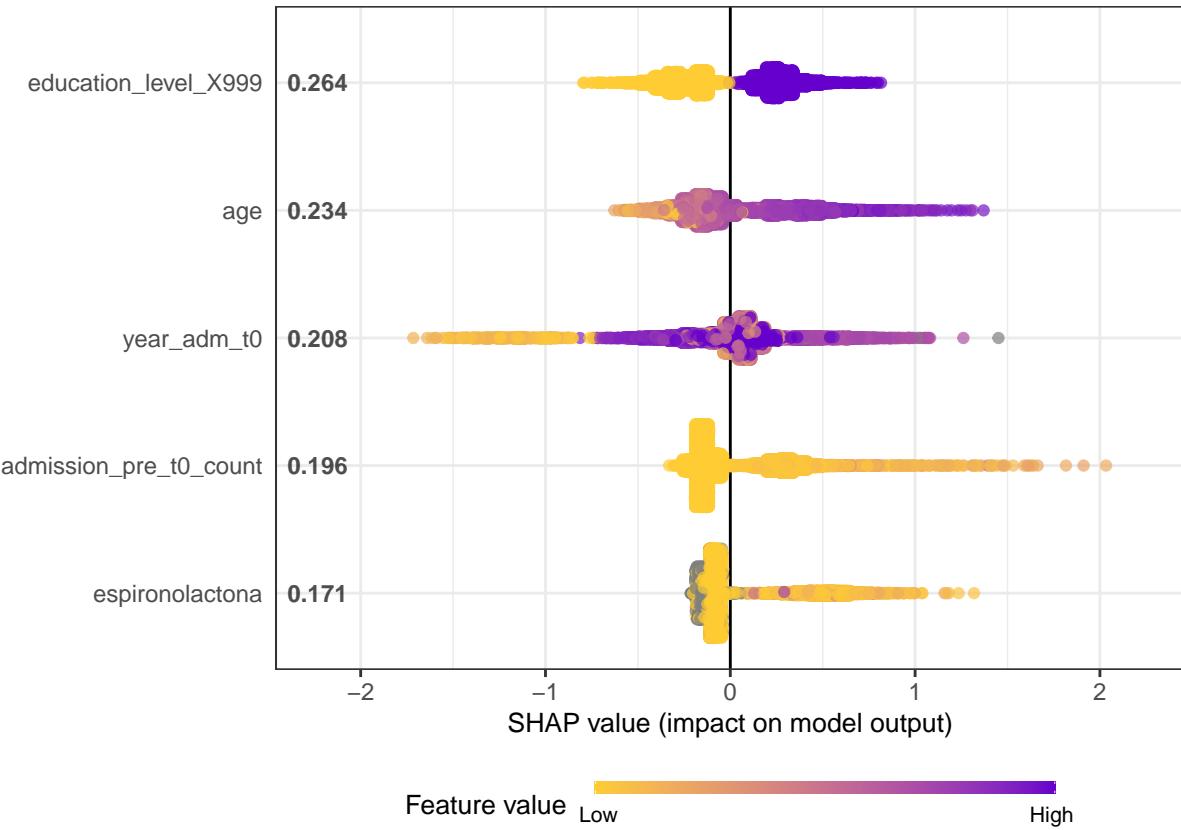
df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

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)

```



```

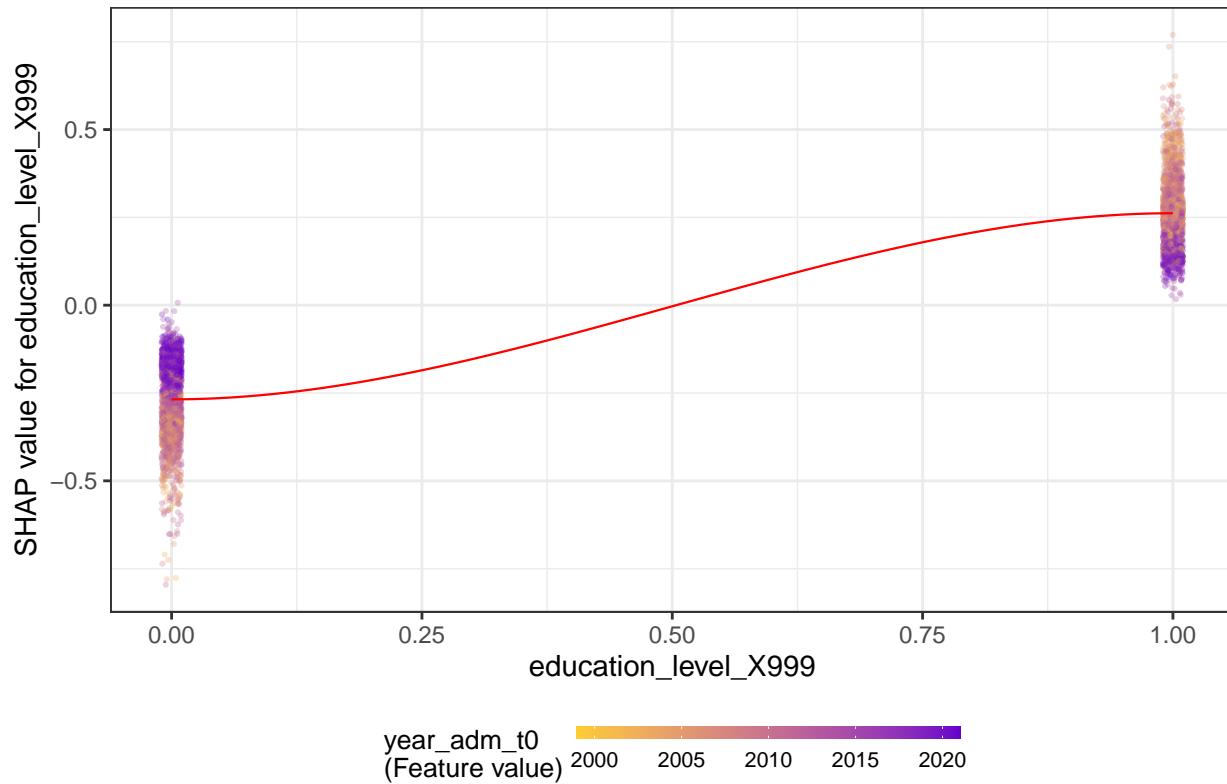
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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'
## 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 1.5385e-29
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1.01

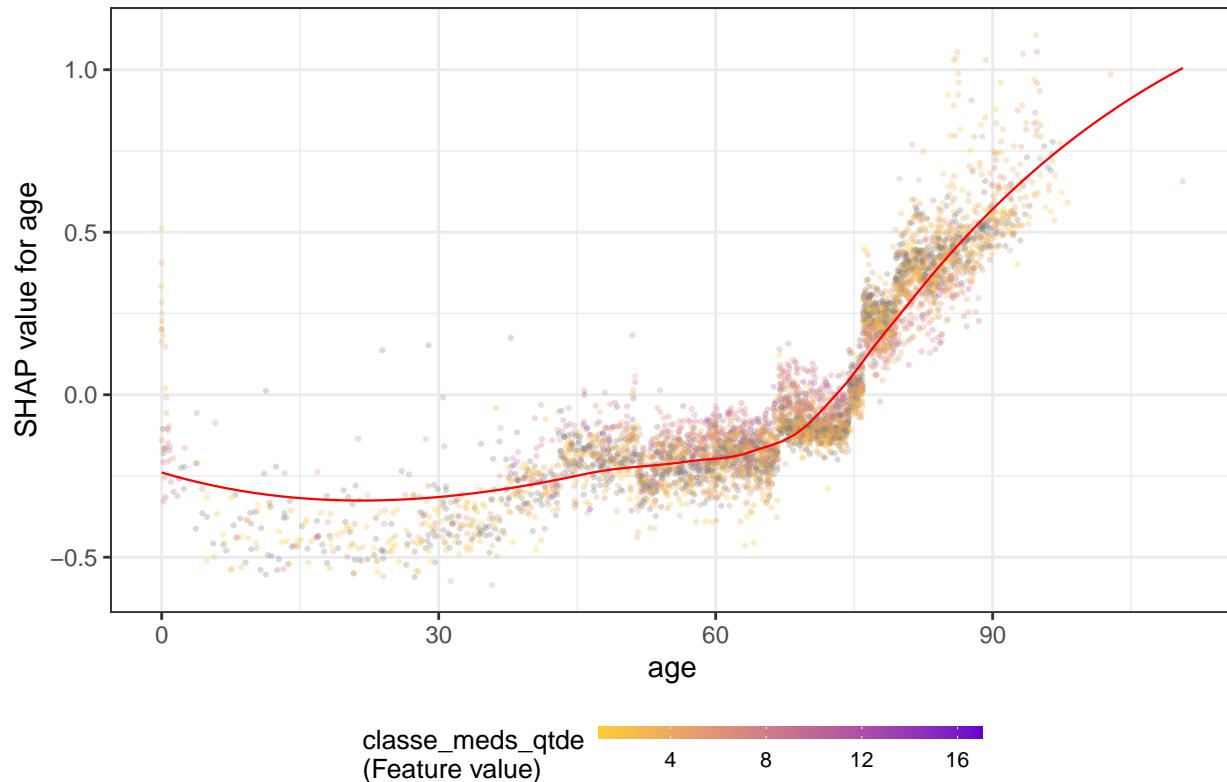
```

### education\_level\_X999



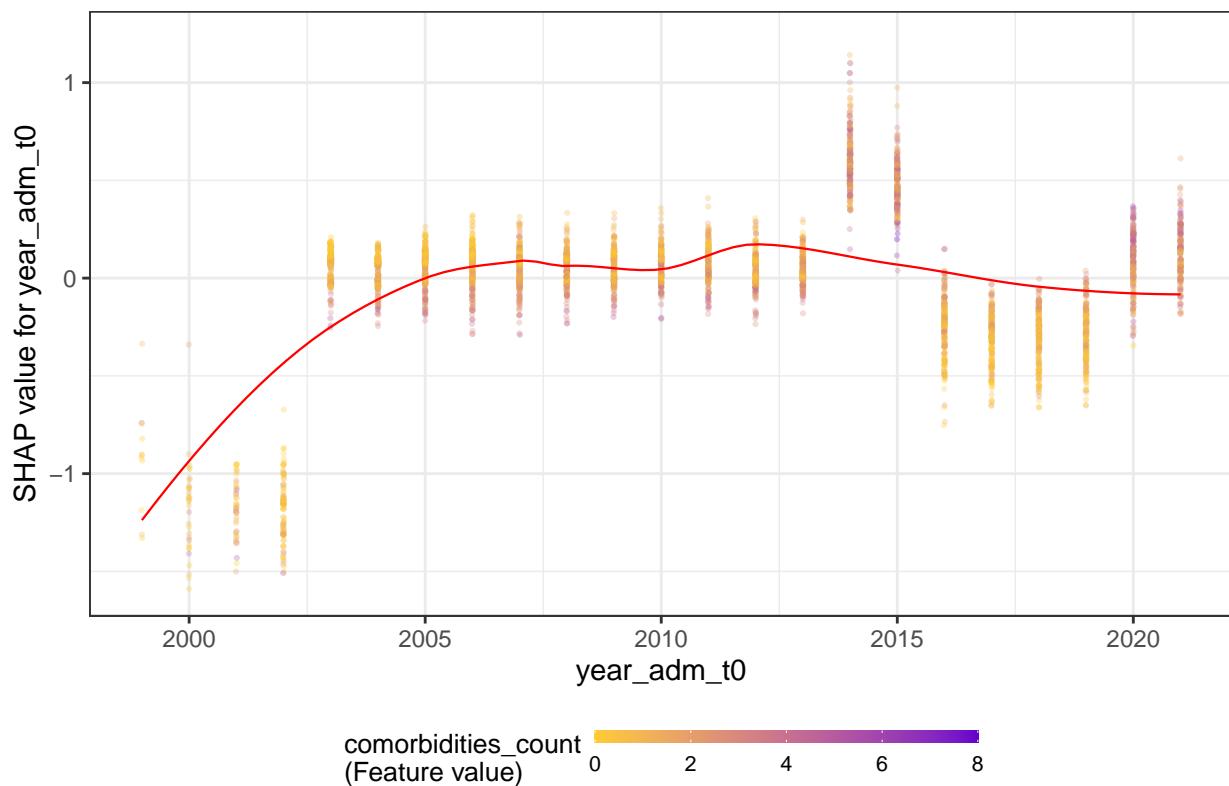
```
## `geom_smooth()` using formula 'y ~ x'
```

### age



```
## `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).
```

year\_adm\_t0

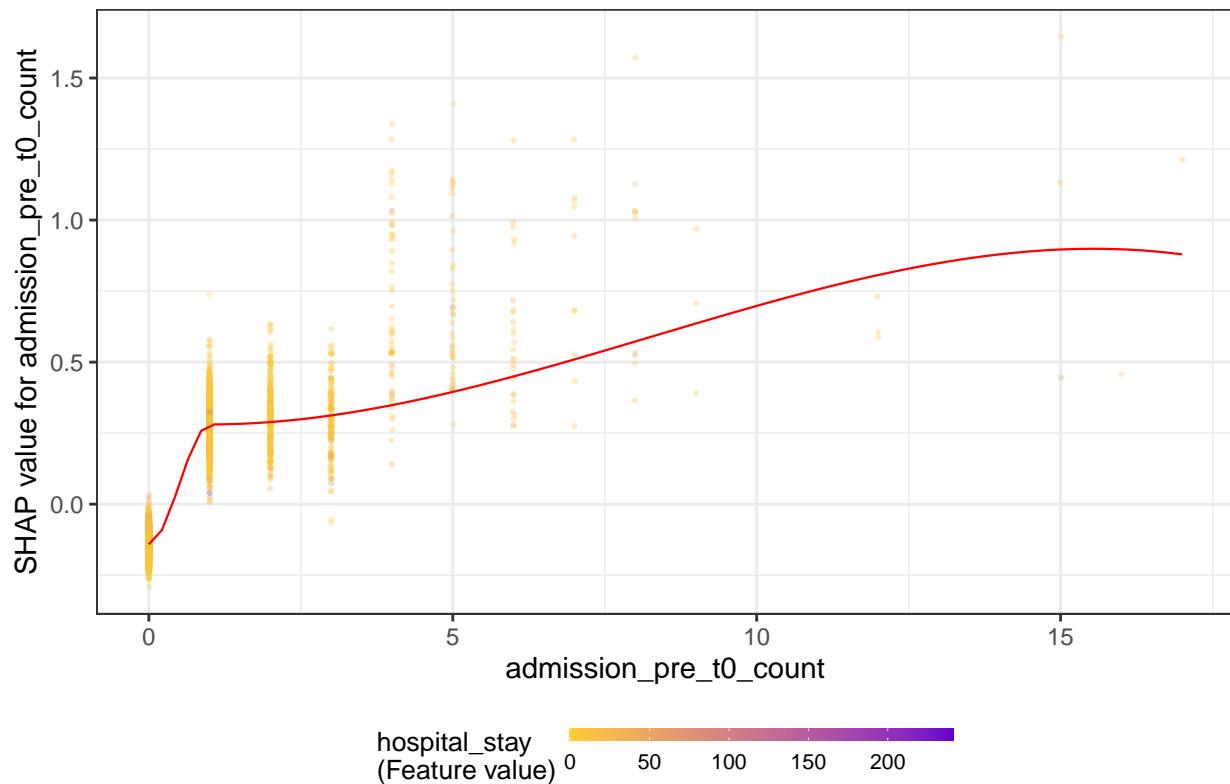


```

## `geom_smooth()` using formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -0.085
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.085
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 1.7903e-27
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1

```

admission\_pre\_t0\_count

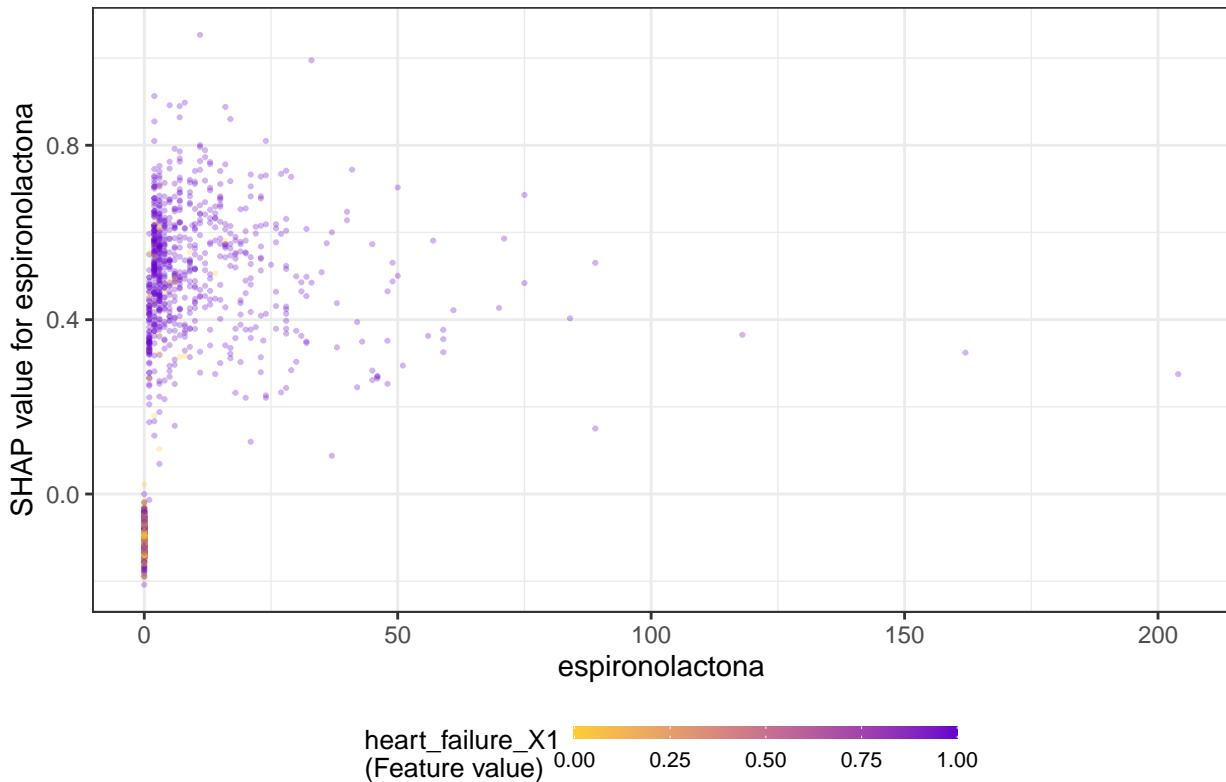


```

## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 1044 rows containing non-finite values (stat_smooth).
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : at -1.02
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : radius 1.0404
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : all data on boundary
## of neighborhood. make span bigger
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -1.02
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.02
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 1
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : zero-width
## neighborhood. make span bigger
## Warning: Computation failed in 'stat_smooth()':
## NA/NaN/Inf in foreign function call (arg 5)
## Warning: Removed 1044 rows containing missing values (geom_point).

```

## espironolactona

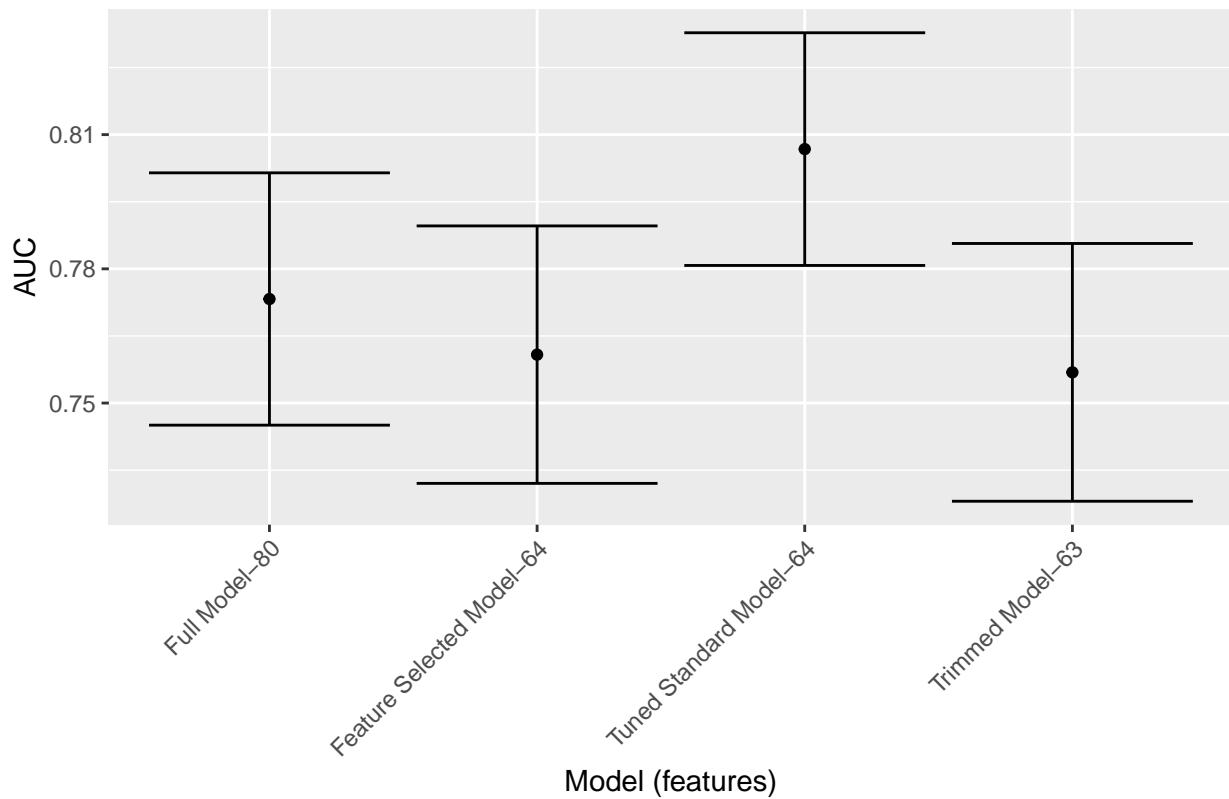


## Models Comparison

```
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$auc_upper,
  'Tuned Standard Model', standard_results$auc, standard_results$auc_lower, standard_results$auc_upper, length(selected_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(selected_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))
```

## death\_2year



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