

# Final Model - death\_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

```
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. cancer
19. comorbidities\_count
20. procedure\_type\_1
21. reop\_type\_1
22. procedure\_type\_new
23. cied\_final\_1
24. cied\_final\_group\_1

25. admission\_pre\_t0\_count  
26. admission\_pre\_t0\_180d  
27. year\_adm\_t0  
28. icu\_t0  
29. dialysis\_t0  
30. admission\_t0\_emergency  
31. aco  
32. antiaritmico  
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. antiviral  
46. classe\_meds\_qtde  
47. meds\_cardiovasc\_qtde  
48. meds\_antimicrobianos  
49. vni  
50. ventilacao\_mecanica  
51. transplante\_cardiaco  
52. cir\_toracica  
53. outros\_proced\_cirurgicos  
54. icp  
55. cateterismo  
56. cateter\_venoso\_central  
57. proced\_invasivos\_qtde  
58. transfusao  
59. interconsulta  
60. equipe\_multiprof  
61. holter  
62. teste\_esforco  
63. tilt\_teste  
64. metodos\_graficos\_qtde  
65. laboratorio  
66. cultura  
67. analises\_clinicas\_qtde  
68. citologia  
69. histopatologia\_qtde  
70. angio\_tc  
71. angiografia  
72. aortografia  
73. cintilografia  
74. ecocardiograma  
75. endoscopia  
76. flebografia  
77. pet\_ct  
78. ultrassom  
79. tomografia  
80. ressonancia  
81. exames\_imagem\_qtde  
82. bic  
83. 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: 0.806"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.810"

Features with zero importance on the initial model:

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

unimportant_features %>%
  gluedown::md_order()

1. cardiac_arrest
2. vni
3. cir_toracica
4. transfusao
5. teste_esforco
6. aortografia
7. 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: 0.805"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.814"

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
83	0.8060	0.0128	0.0000	heart_disease
76	0.8049	0.0109	0.0011	endoscopia
75	0.8072	0.0097	-0.0011	tilt_teste
74	0.8073	0.0096	-0.0012	angio_tc
73	0.8067	0.0105	-0.0007	underlying_heart_disease
72	0.8058	0.0113	0.0002	procedure_type_1
71	0.8057	0.0104	0.0003	dialysis_t0
70	0.8072	0.0111	-0.0012	antiviral
69	0.8075	0.0109	-0.0015	heart_disease
68	0.8069	0.0107	-0.0009	antifungico
67	0.8042	0.0121	0.0018	icp
66	0.8071	0.0103	-0.0010	holter
65	0.8068	0.0095	-0.0008	ressonancia
64	0.8074	0.0099	-0.0014	histopatologia_qtde
63	0.8098	0.0112	-0.0037	angiografia
62	0.8081	0.0107	-0.0021	cateter_venoso_central
61	0.8109	0.0104	-0.0049	antiplaquetario_ev
60	0.8059	0.0097	0.0001	flebografia
59	0.8067	0.0119	-0.0007	insulina
58	0.8086	0.0100	-0.0025	hemodialysis
57	0.8069	0.0105	-0.0008	copd
56	0.8075	0.0106	-0.0015	bic
55	0.8075	0.0097	-0.0014	heart_failure
54	0.8072	0.0103	-0.0012	transplante_cardiaco
53	0.8088	0.0112	-0.0028	cied_final_1

Table 1: (continued)

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
52	0.8090	0.0097	-0.0030	sex
51	0.8107	0.0106	-0.0046	procedure_type_new
50	0.8080	0.0100	-0.0020	nyha_basal
49	0.8071	0.0100	-0.0011	renal_failure
48	0.8065	0.0102	-0.0004	outros_proced_cirurgicos
47	0.8071	0.0097	-0.0011	aco
46	0.8057	0.0115	0.0003	interconsulta
45	0.8059	0.0098	0.0001	cancer
44	0.8088	0.0105	-0.0028	prior_mi
43	0.8068	0.0091	-0.0008	stroke
42	0.8060	0.0087	0.0001	cintilografia
41	0.8084	0.0104	-0.0024	diabetes
40	0.8102	0.0101	-0.0042	tomografia
39	0.8055	0.0109	0.0005	cateterismo
38	0.8052	0.0107	0.0009	valvopathy
37	0.8045	0.0096	0.0016	admission_pre_t0_180d
36	0.8061	0.0089	-0.0001	cultura
35	0.8044	0.0090	0.0016	ecocardiograma
34	0.8046	0.0100	0.0014	reop_type_1
33	0.8010	0.0084	0.0050	admission_t0_emergency
32	0.8010	0.0106	0.0050	hypertension
31	0.8035	0.0105	0.0025	ultrassom
30	0.8009	0.0107	0.0051	ventilacao_mecanica
29	0.8006	0.0106	0.0054	analises_clinicas_qtde
28	0.8005	0.0102	0.0055	citologia
27	0.7978	0.0107	0.0082	digoxina
26	0.7966	0.0104	0.0094	af
25	0.7970	0.0108	0.0090	proced_invasivos_qtde
24	0.7957	0.0106	0.0103	dva

```

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

## [1] "Selected Model Test AUC: 0.805"

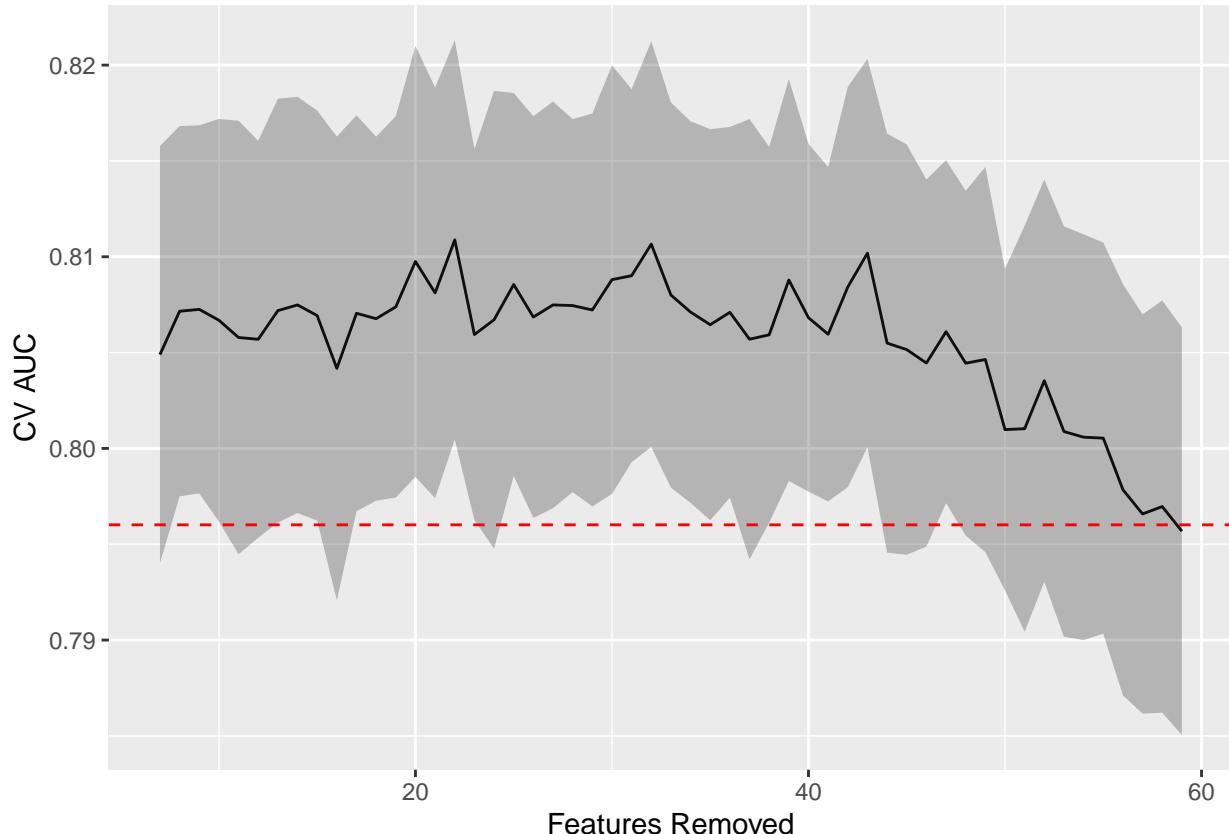
selection_results <- selection_results %>%
  filter(`Number of Features` < length(features)) %>%
  mutate(`Features Removed` = length(features) - `Number of Features`,
    `CV AUC Low` = `CV AUC` - `CV AUC Std Error`,
    `CV AUC High` = `CV AUC` + `CV AUC Std Error`)

```

```

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. age
2. hospital\_stay
3. year\_adm\_t0
4. comorbidities\_count
5. admission\_pre\_t0\_count
6. vasodilatador
7. laboratorio
8. espironolactona
9. metodos\_graficos\_qtde
10. meds\_cardiovasc\_qtde
11. classe\_meds\_qtde
12. psicofarmacos
13. meds\_antimicrobianos

```

14. ieca_bra
15. icu_t0
16. diuretico
17. estatina
18. insuf_cardiaca
19. equipe_multiprof
20. exames_imagem_qtde
21. antiaritmico
22. education_level
23. cied_final_group_1
24. dva

```

## 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)

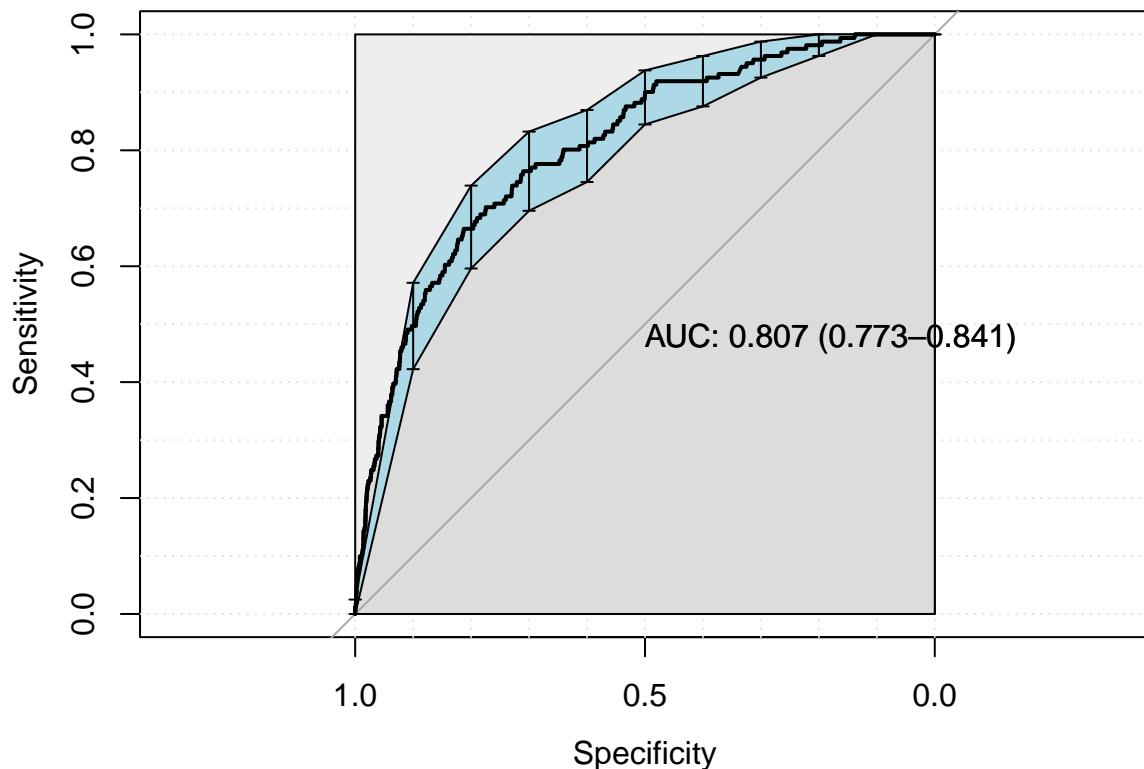
lightgbm_auc <- validation(final_lightgbm_fit, df_test)

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 3714    54
##   1   855   107
##
##                  Accuracy : 0.8078
##                     95% CI : (0.7963, 0.819)
##      No Information Rate : 0.966
##      P-Value [Acc > NIR] : 1
##
##                  Kappa : 0.1404
##
## Mcnemar's Test P-Value : <2e-16
##
##      Sensitivity : 0.8129
##      Specificity : 0.6646
##      Pos Pred Value : 0.9857
##      Neg Pred Value : 0.1112
##      Prevalence : 0.9660
##      Detection Rate : 0.7852
##      Detection Prevalence : 0.7966
##      Balanced Accuracy : 0.7387
##
##      '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)

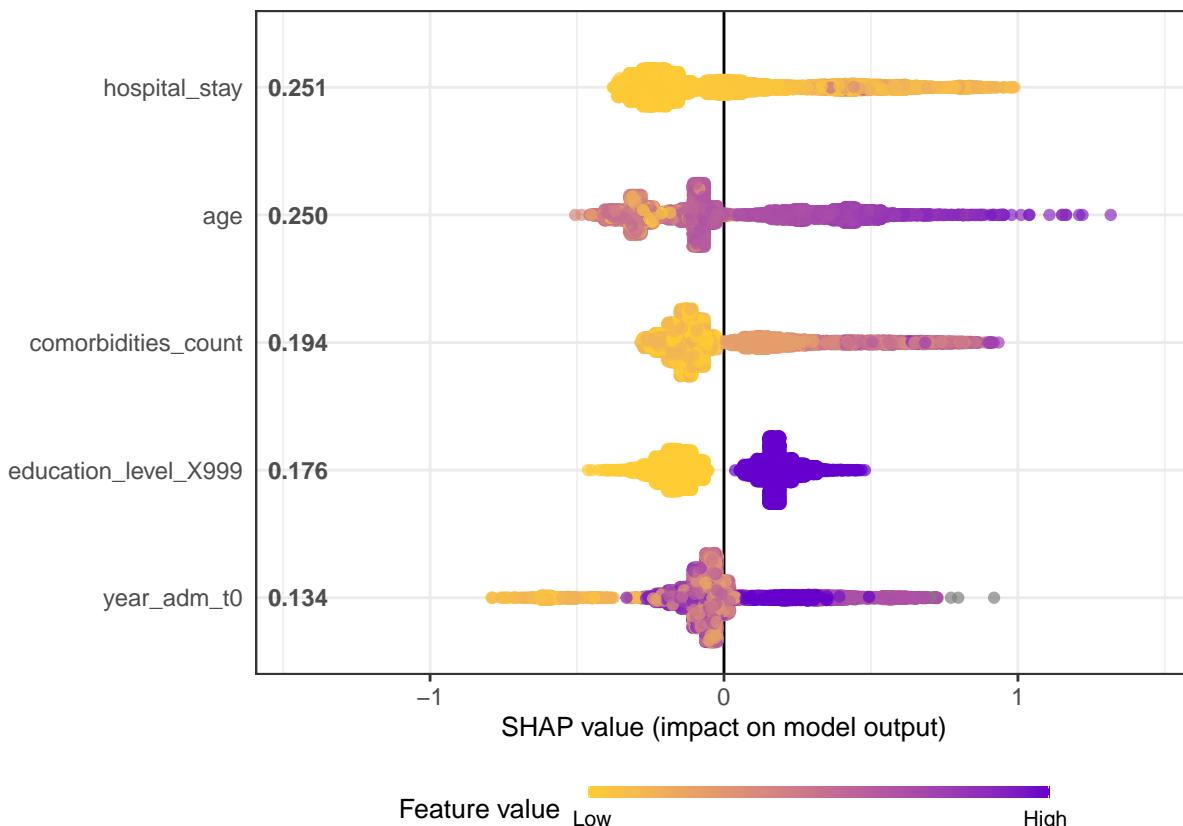
df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

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",

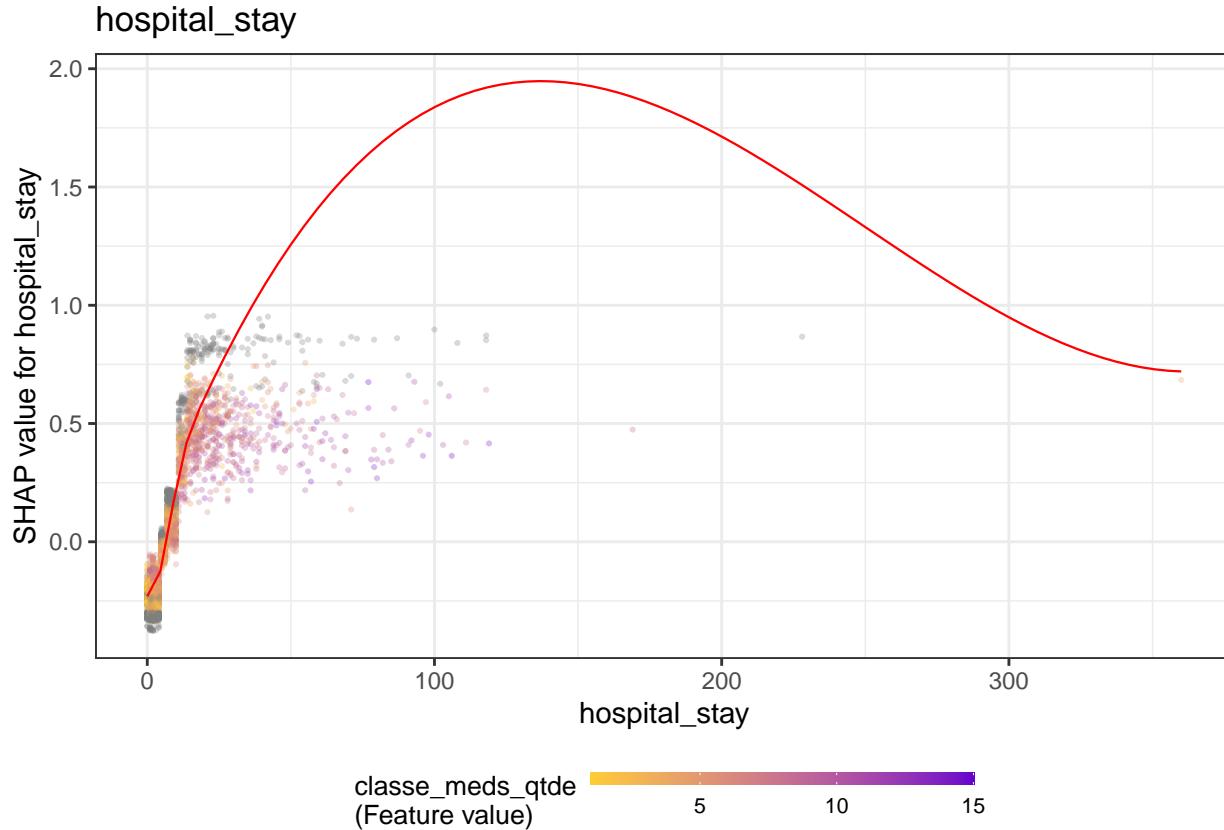
```

```

    smooth = TRUE,
    jitter_width = 0.01,
    alpha = 0.3
) +
  labs(title = x)
print(p)
}

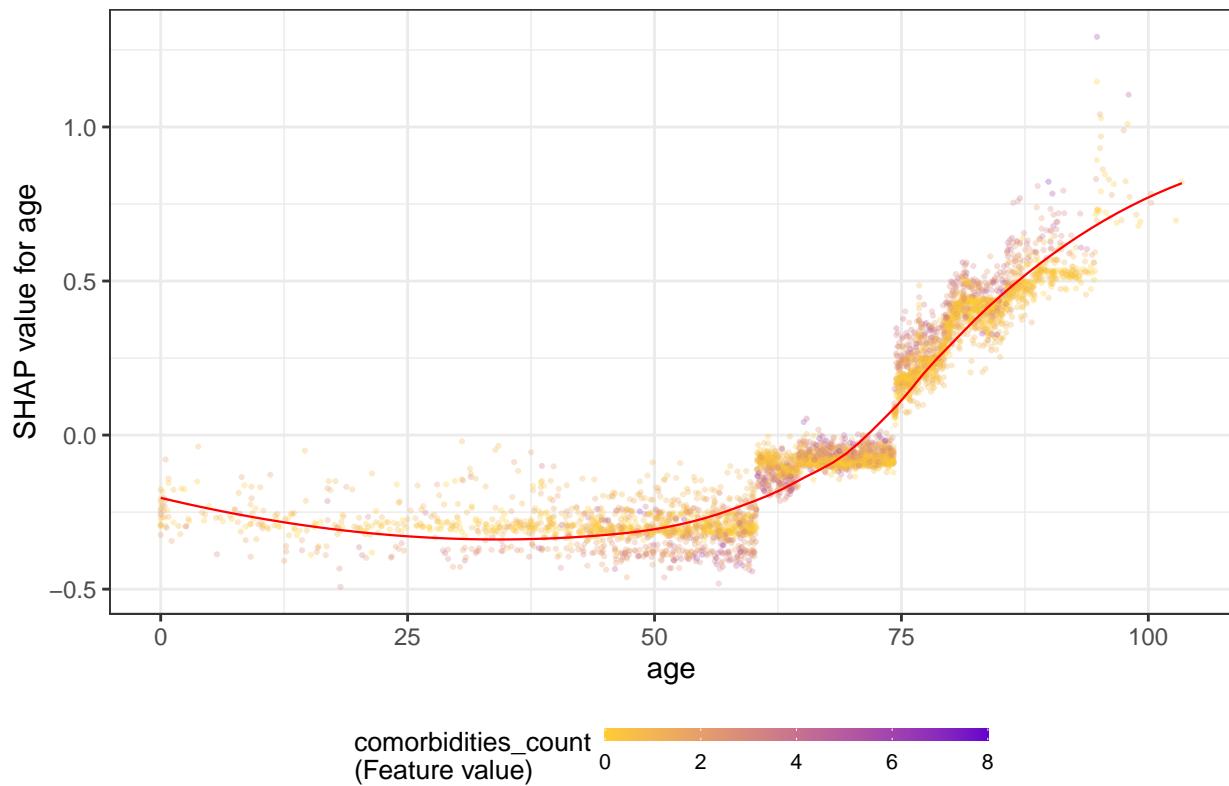
```

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

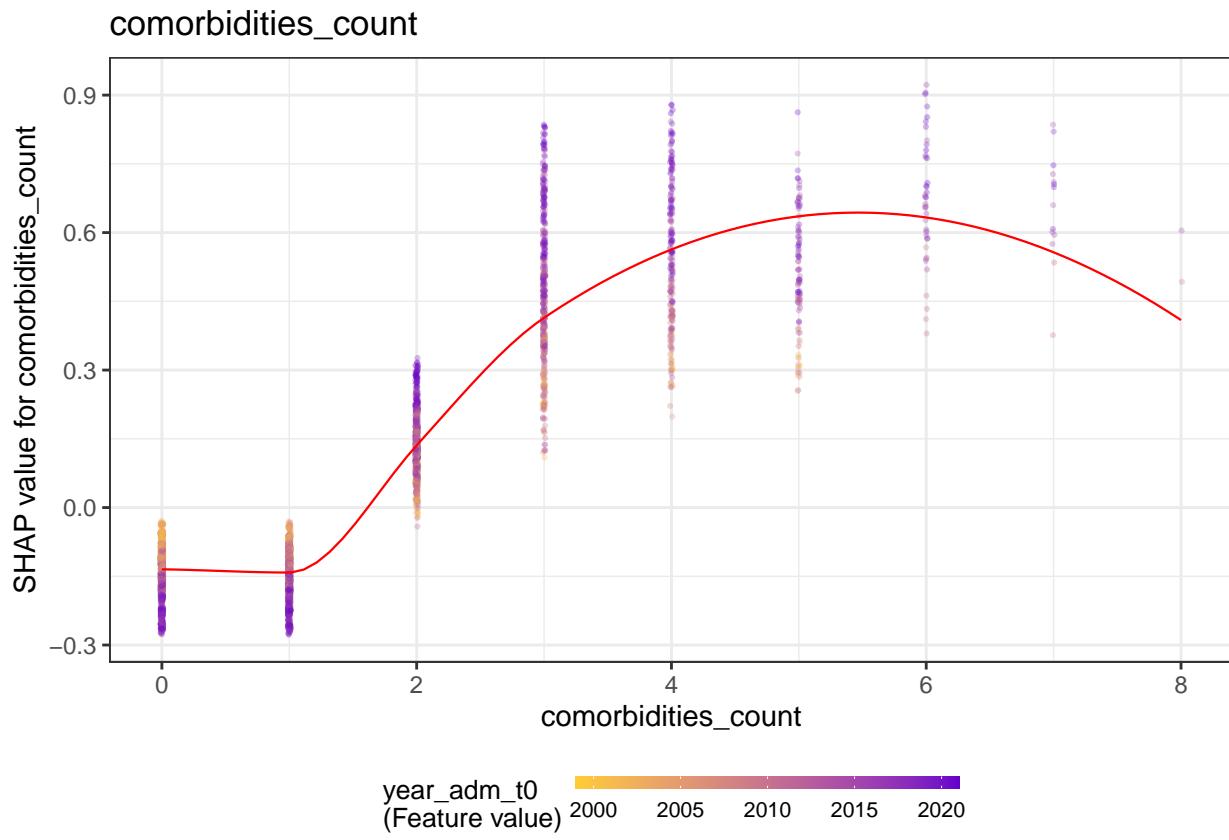


```
## `geom_smooth()` using formula 'y ~ x'
```

age



```
## `geom_smooth()` using formula 'y ~ x'  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at  
## 0  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius 2  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition  
## number 3.344e-15  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near  
## singularities as well. 1
```

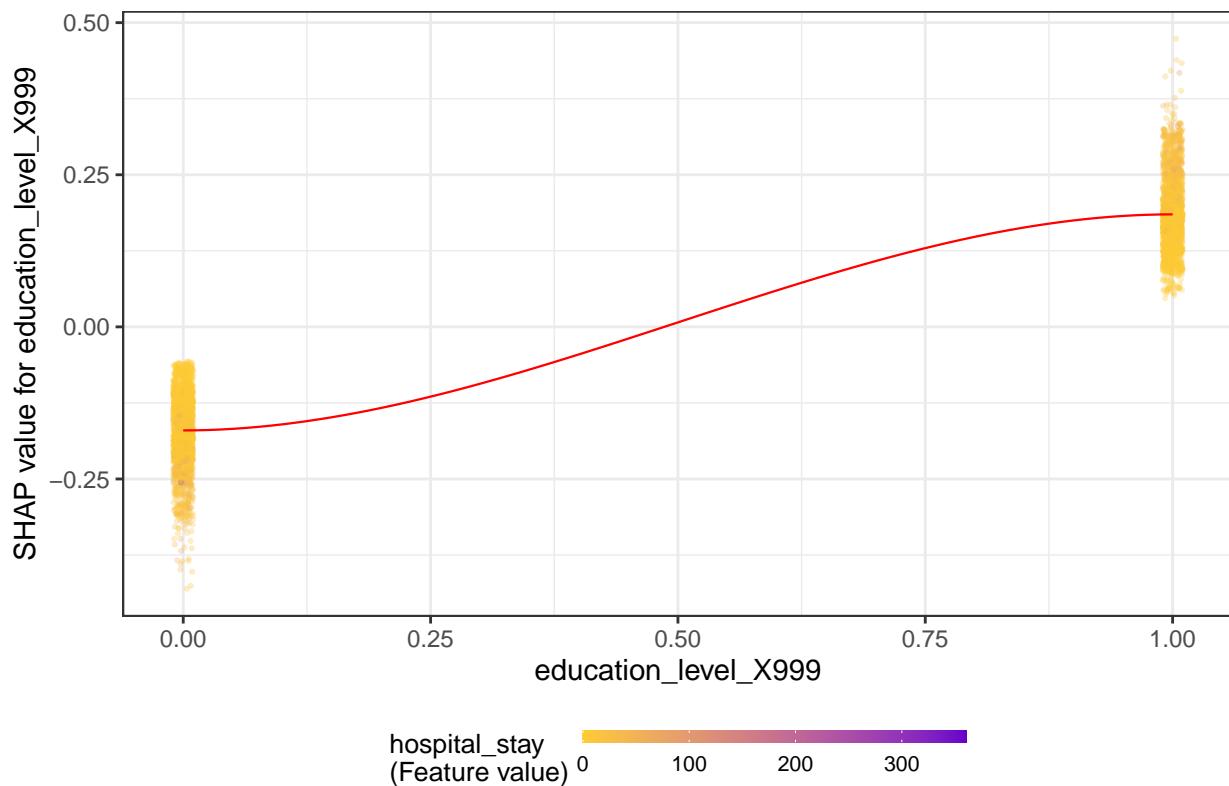


```

## `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 4.7919e-28
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1.01

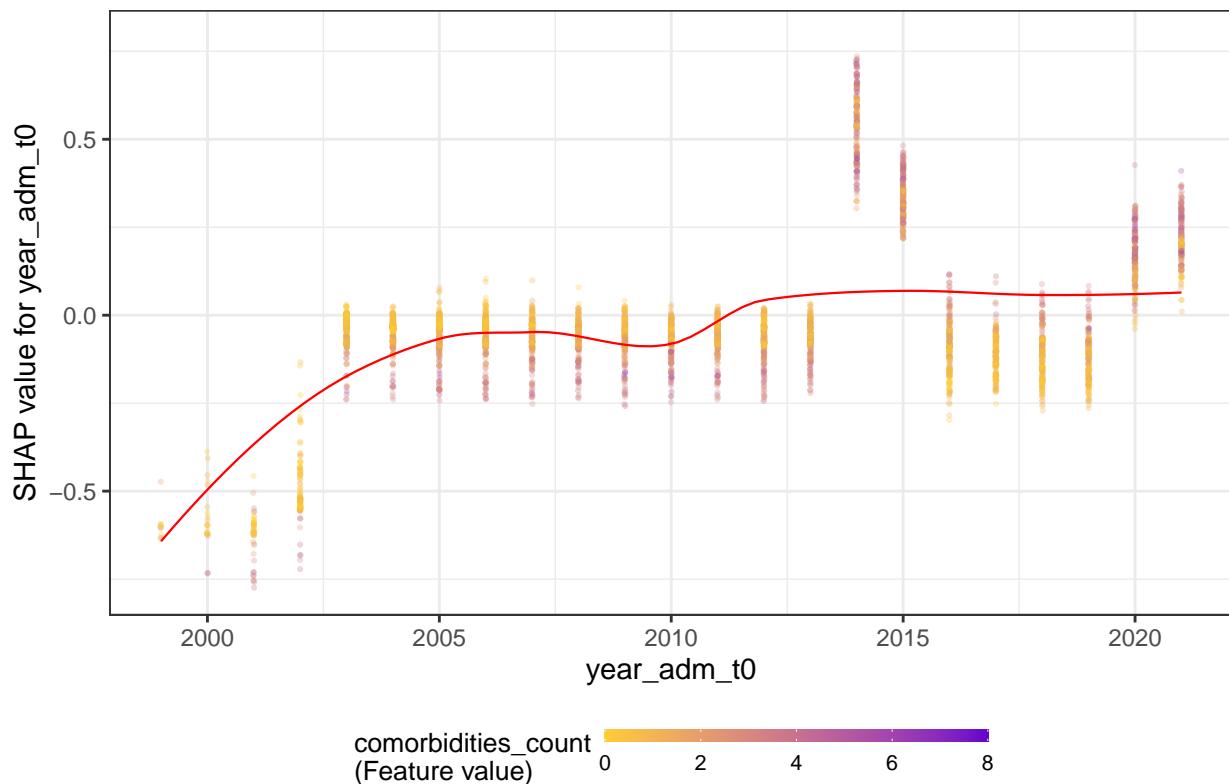
```

education\_level\_X999



```
## `geom_smooth()` using formula 'y ~ x'  
## Warning: Removed 10 rows containing non-finite values (stat_smooth).  
## Warning: Removed 10 rows containing missing values (geom_point).
```

year\_adm\_t0



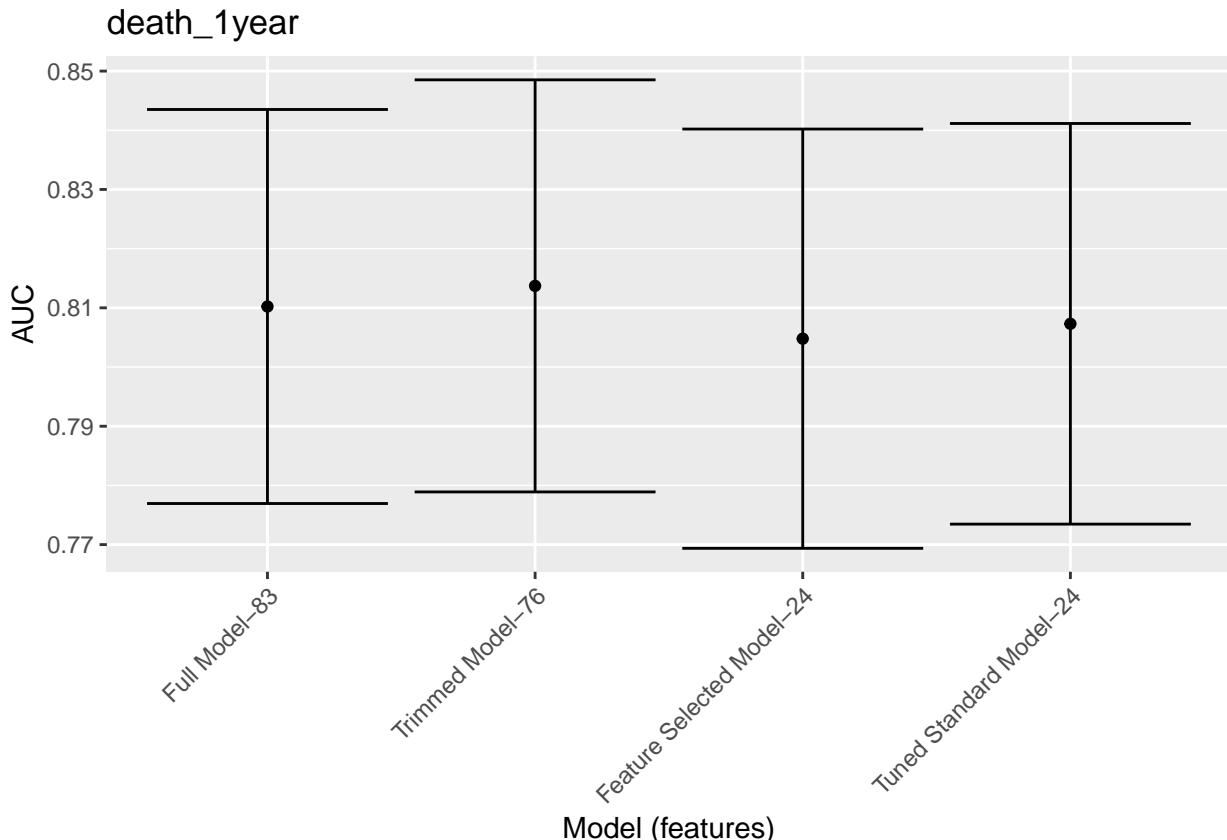
## Models Comparison

```

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(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))

```



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