

Final Model - death_3year

Eduardo Yuki Yada

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)
```

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))
```

Eligible features

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

exception_columns = c('death_intraop', 'death_intraop_1')

correlated_columns = c('year_procedure_1', # com year_adm_t0
                      'age_surgery_1', # com age
                      'admission_t0', # com admission_pre_t0_count
                      '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)
}

```

Starting features:

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

1. sex
2. age
3. race
4. education_level
5. underlying_heart_disease
6. heart_disease
7. nyha_basal
8. hypertension
9. prior_mi
10. heart_failure
11. af
12. cardiac_arrest
13. valvopathy
14. diabetes
15. renal_failure
16. hemodialysis
17. stroke
18. copd
19. cancer
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. antiaritmico
34. ieca_bra
35. dva
36. digoxina
37. estatina
38. diuretico
39. vasodilatador
40. insuf_cardiaca
41. espironolactona
42. antiplaquetario_ev
43. insulina
44. anticonvulsivante
45. psicofarmacos
46. antifungico
47. classe_meds_qtde
48. meds_cardiovasc_qtde
49. meds_antimicrobianos
50. outros_proced_cirurgicos
51. icp

52. angioplastia
53. cateterismo
54. eletrofisiologia
55. cateter_venoso_central
56. proced_invasivos_qtde
57. transfusao
58. equipe_multiprof
59. ecg
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. angiografia
71. cintilografia
72. ecocardiograma
73. endoscopia
74. flebografia
75. pet_ct
76. ultrassom
77. tomografia
78. radiografia
79. ressonancia
80. exames_imagem_qtde
81. bic

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) %>% dplyr::select(all_of(c(features, outcome_column)))
df_test <- testing(df_split) %>% dplyr::select(all_of(c(features, outcome_column)))
```

Global parameters

```
k <- 4 # Number of folds for cross validation
grid_size <- 50 # Number of parameter combination to tune on each model

set.seed(234)
df_folds <- vfold_cv(df_train, v = k,
                      strata = all_of(outcome_column))

max_auc_loss <- 0.01
```

Functions

```
niceFormatting = function(df, caption="", digits = 2, font_size = NULL){
  df %>%
    kbl(booktabs = T, longtable = T, caption = caption, digits = digits, format = "latex") %>%
    kable_styling(font_size = font_size,
                  latex_options = c("striped", "HOLD_position", "repeat_header"))
}

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)
}
```

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") %>%
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]))
}

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

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: 0.793"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.797"

Features with zero importance on the initial model:

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

unimportant_features %>%
gluedown::md_order()

```

1. heart_disease
2. cardiac_arrest
3. stroke
4. angioplastia
5. eletrofisiologia

```

6. cateter_venoso_central
7. transfusao
8. tilt_teste
9. angiografia
10. endoscopia
11. flebografia
12. 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: 0.786"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.800"
selection_results <- tibble::tribble(
  ~`Number of Features`, ~`AUC Loss`, ~`Least Important Feature`,
  length(features), 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),
           `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),
           `AUC Loss` = current_auc_loss,
           `Least Important Feature` = current_least_important)

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

selection_results %>% niceFormatting(digits = 4)

```

Table 1:

Number of Features	AUC Loss	Least Important Feature
81	0.0000	antiplaquetario_ev
69	0.0069	hemodialysis
68	0.0023	outros_proced_cirurgicos
67	0.0052	antiplaquetario_ev
66	0.0023	angio_tc
65	-0.0026	cultura
64	0.0007	procedure_type_1
63	-0.0001	diabetes
62	0.0068	cancer
61	0.0036	teste_esforco
60	0.0023	cateterismo
59	0.0053	cintilografia
58	0.0026	dialysis_t0
57	0.0012	antifungico
56	0.0023	insulina
55	0.0014	prior_mi
54	0.0020	ressonancia
53	0.0072	copd
52	-0.0005	icp
51	0.0022	procedure_type_new
50	0.0055	histopatologia_qtde
49	0.0032	admission_t0_emergency
48	0.0061	anticonvulsivante
47	0.0037	race
46	0.0000	citologia
45	0.0088	hypertension
44	0.0001	heart_failure
43	0.0062	ecocardiograma
42	0.0040	tomografia
41	0.0084	holter
40	0.0020	proced_invasivos_qtde
39	0.0036	radiografia
38	0.0062	af
37	0.0002	ultrassom
36	0.0057	aco
35	0.0012	admission_pre_t0_180d
34	0.0014	bic
33	0.0036	sex
32	0.0023	renal_failure
31	0.0046	cied_final_1
30	0.0083	valvopathy
29	0.0066	digoxina
28	0.0039	analises_clinicas_qtde
27	0.0023	underlying_heart_disease
26	0.0047	cied_final_group_1
25	0.0081	dva
24	0.0048	antiarritmico
23	0.0058	estatina
22	0.0012	metodos_graficos_qtde
21	0.0058	psicofarmacos
20	0.0070	exames_imagem_qtde
19	0.0050	reop_type_1
18	0.0064	nyha_basal

Table 1: (continued)

Number of Features	AUC Loss	Least Important Feature
17	0.0231	equipe_multiprof

```

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

con <- file(sprintf('../EDA/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)

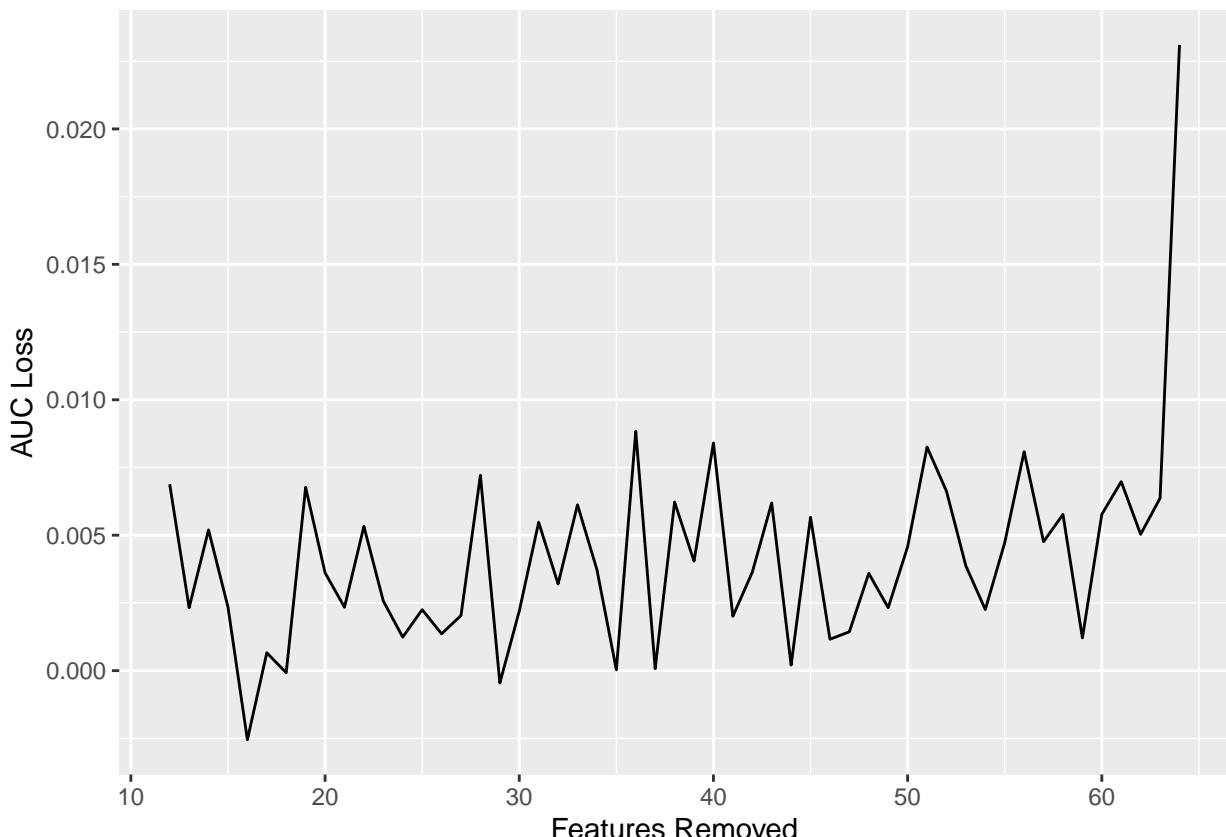
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.770"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.787"

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. year_adm_t0
3. classe_meds_qtde
4. admission_pre_t0_count
5. ieca_bra
6. espironolactona
7. meds_antimicrobianos
8. icu_t0
9. education_level
10. diuretico
11. ecg
12. insuf_cardiaca
13. comorbidities_count
14. vasodilatador
15. equipo_multiprof
16. laboratorio
17. meds_cardiovasc_qtde

```
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)

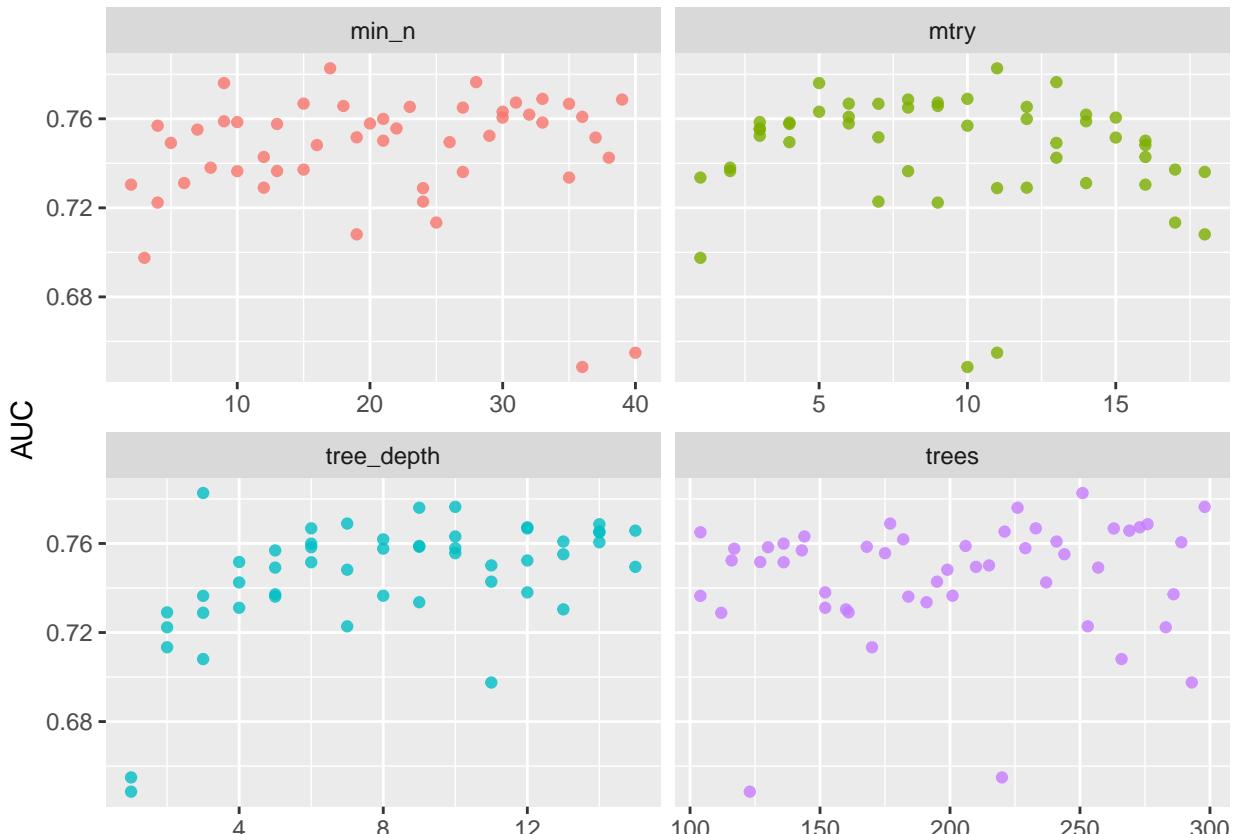
lightgbm_tune %>%
  show_best("roc_auc") %>%
  niceFormatting(digits = 5)
```

Table 2:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
11	251	17	3	0.02131	0.00006	roc_auc	binary	0.78266	4	0.00391	Preprocessor1
13	298	28	10	0.01370	0.41556	roc_auc	binary	0.77646	4	0.00625	Preprocessor1
5	226	9	9	0.00549	0.00000	roc_auc	binary	0.77605	4	0.00562	Preprocessor1
10	177	33	7	0.00005	0.00131	roc_auc	binary	0.76895	4	0.00491	Preprocessor1
8	276	39	14	0.00000	0.00000	roc_auc	binary	0.76865	4	0.00455	Preprocessor1

```
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")
```



```

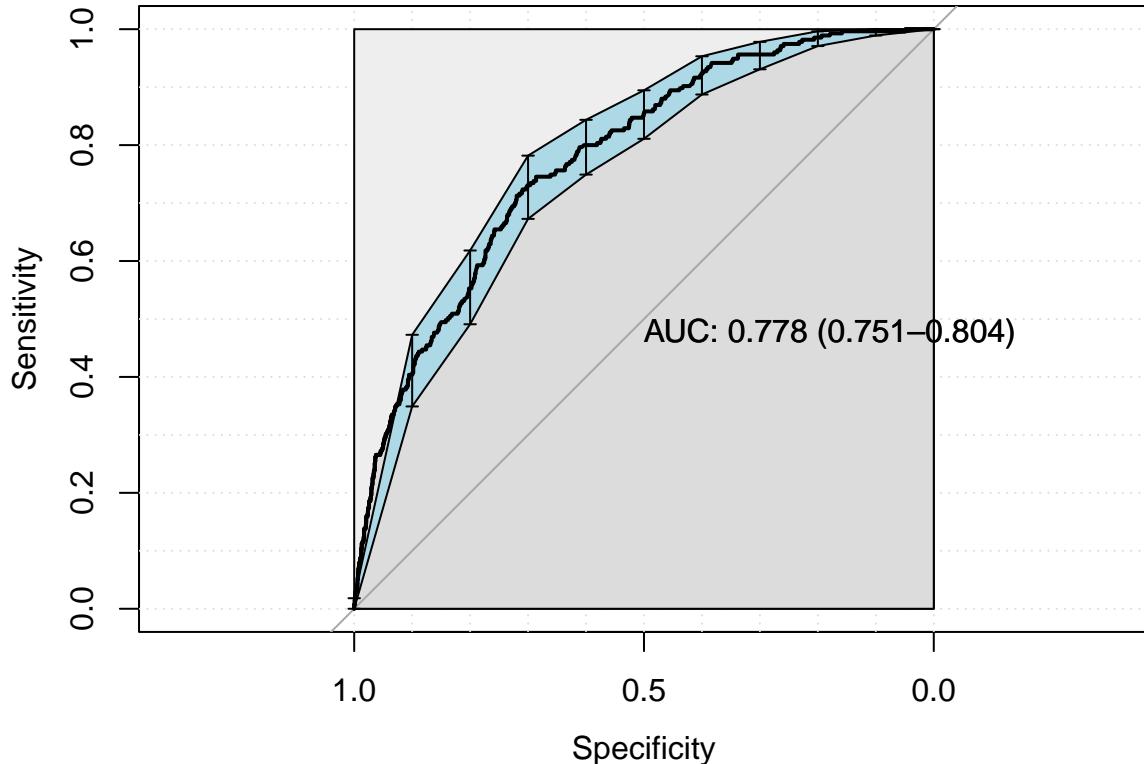
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4454  273
##                      1     1     2
## 
##          Accuracy : 0.9421
##          95% CI : (0.935, 0.9486)
##  No Information Rate : 0.9419
##  P-Value [Acc > NIR] : 0.4913
## 
##          Kappa : 0.0132
## 
##  Mcnemar's Test P-Value : <2e-16
## 
##          Sensitivity : 0.999776
##          Specificity  : 0.007273
##  Pos Pred Value  : 0.942247
##  Neg Pred Value  : 0.666667
##          Prevalence  : 0.941860

```

```

##           Detection Rate : 0.941649
##   Detection Prevalence : 0.999366
##   Balanced Accuracy : 0.503524
##
##      '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

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

```

SHAP values

```

lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

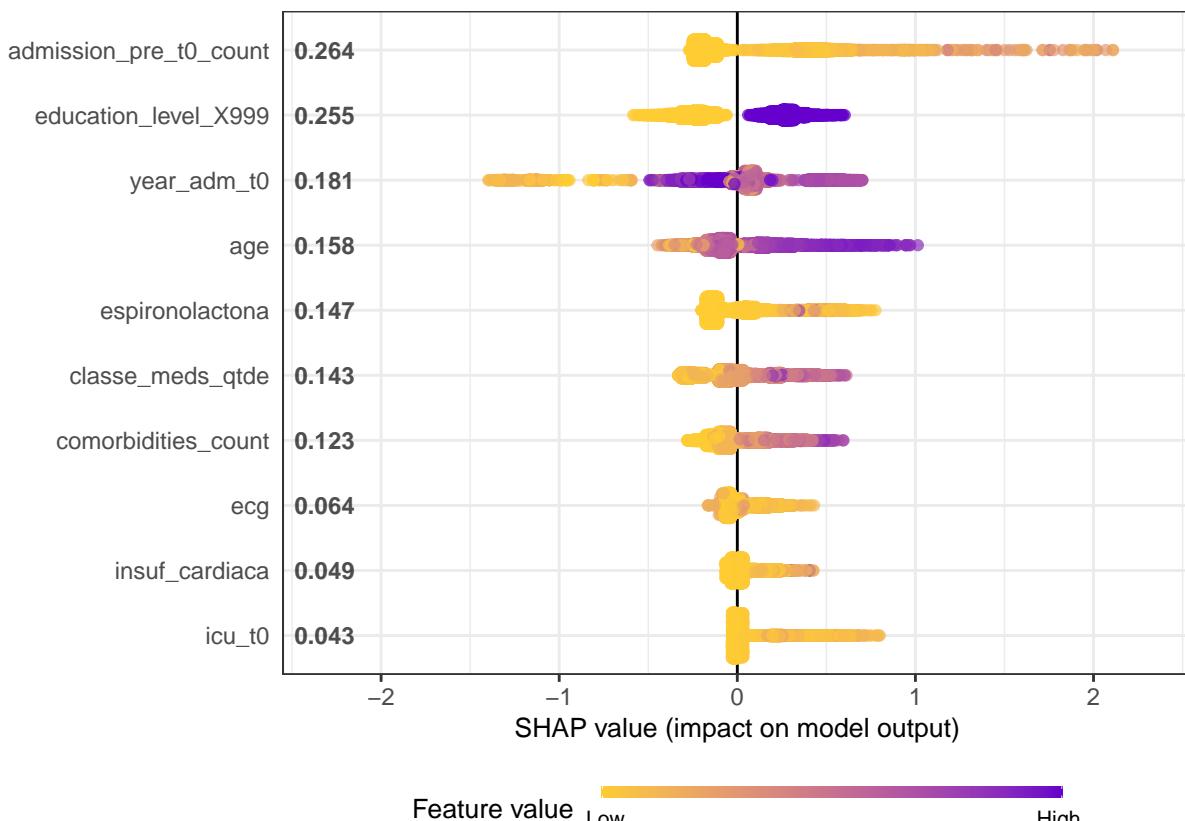
trained_rec <- prep(lightgbm_recipe, training = df_train)

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)))

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

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

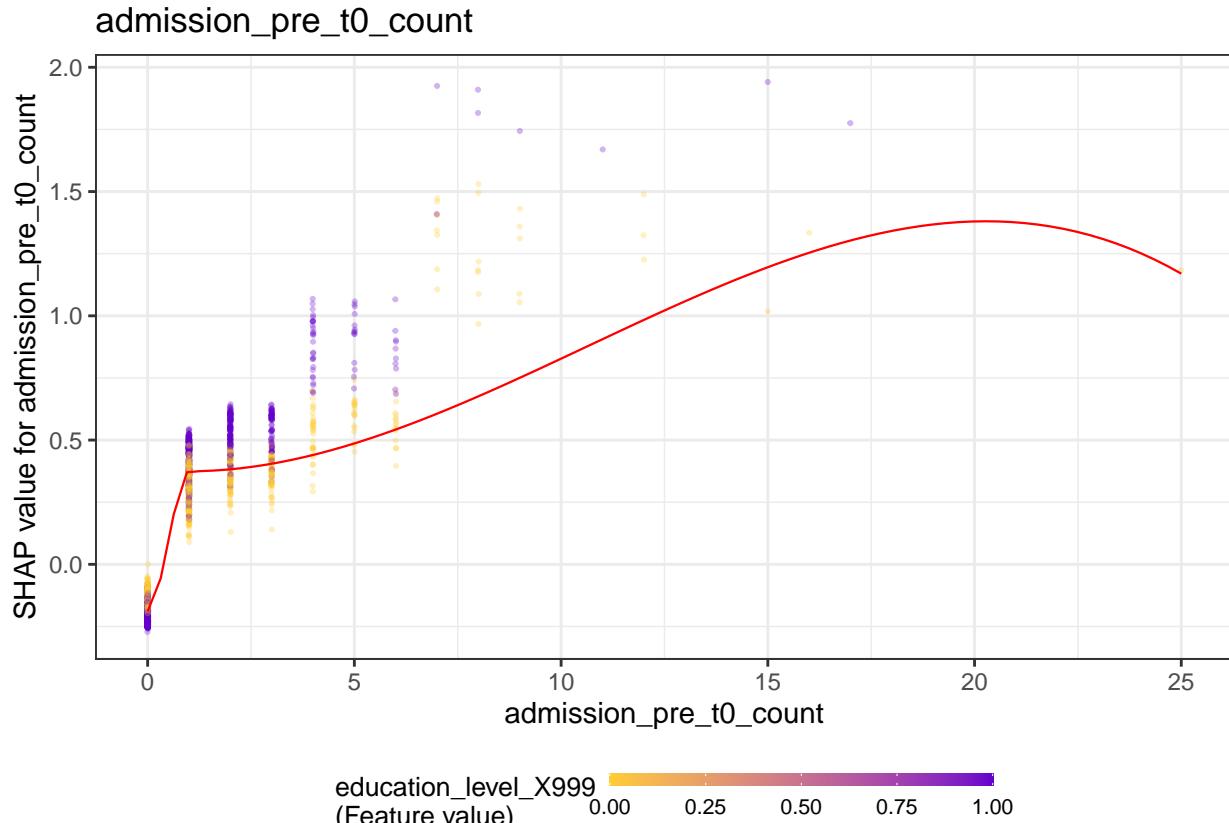
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -0.125

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.125

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 1.3776e-27

## 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

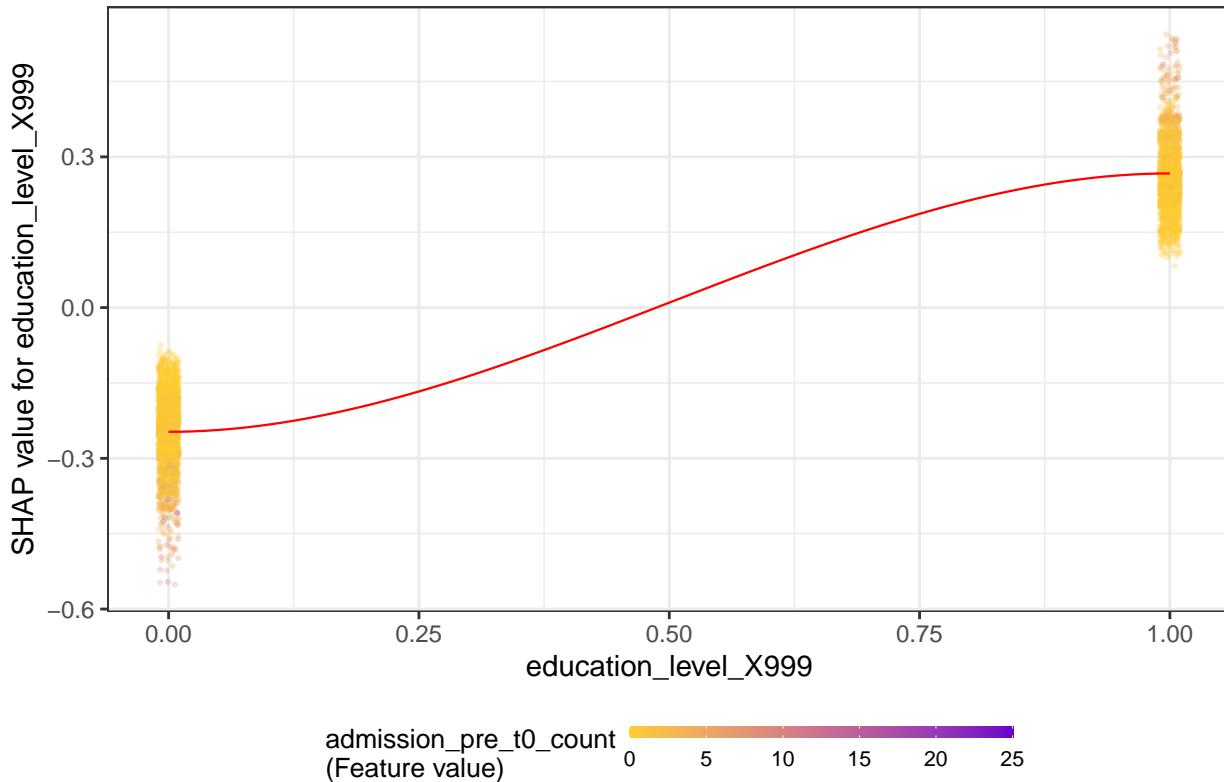
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.005

```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal  
## condition number 2.1666e-28
```

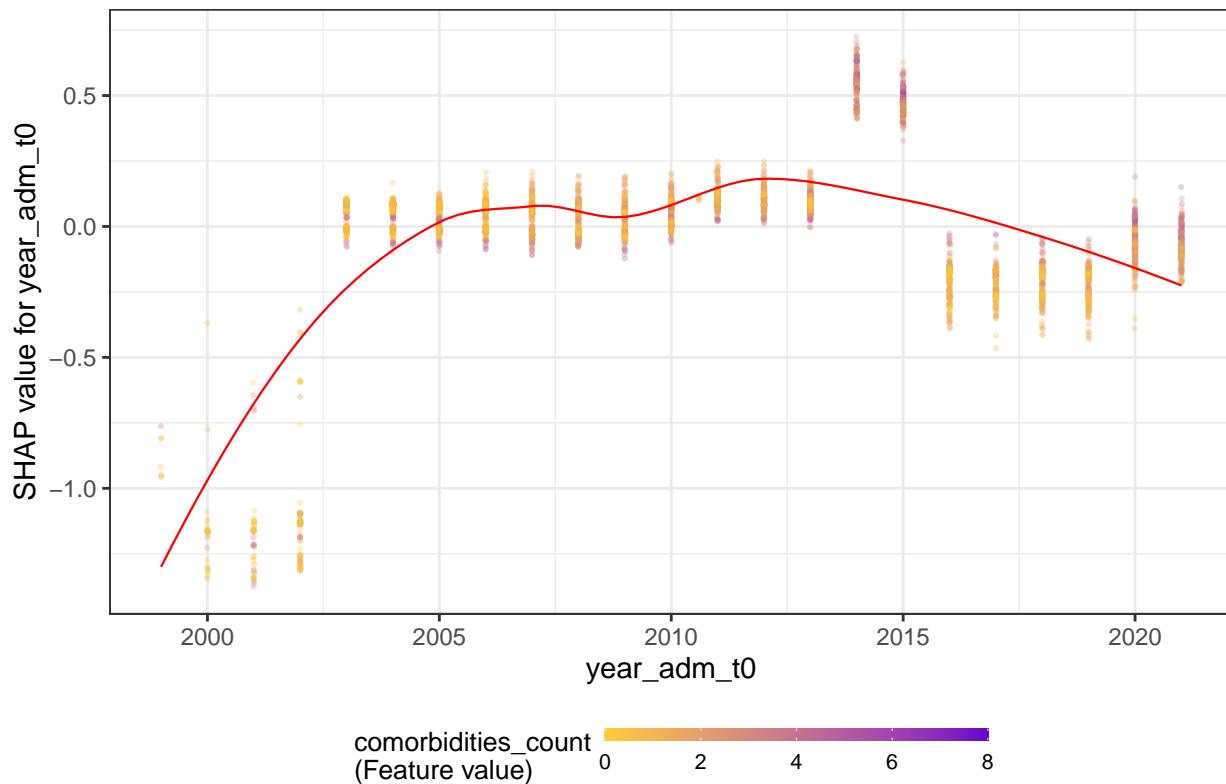
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are  
## other near singularities as well. 1.01
```

education_level_X999

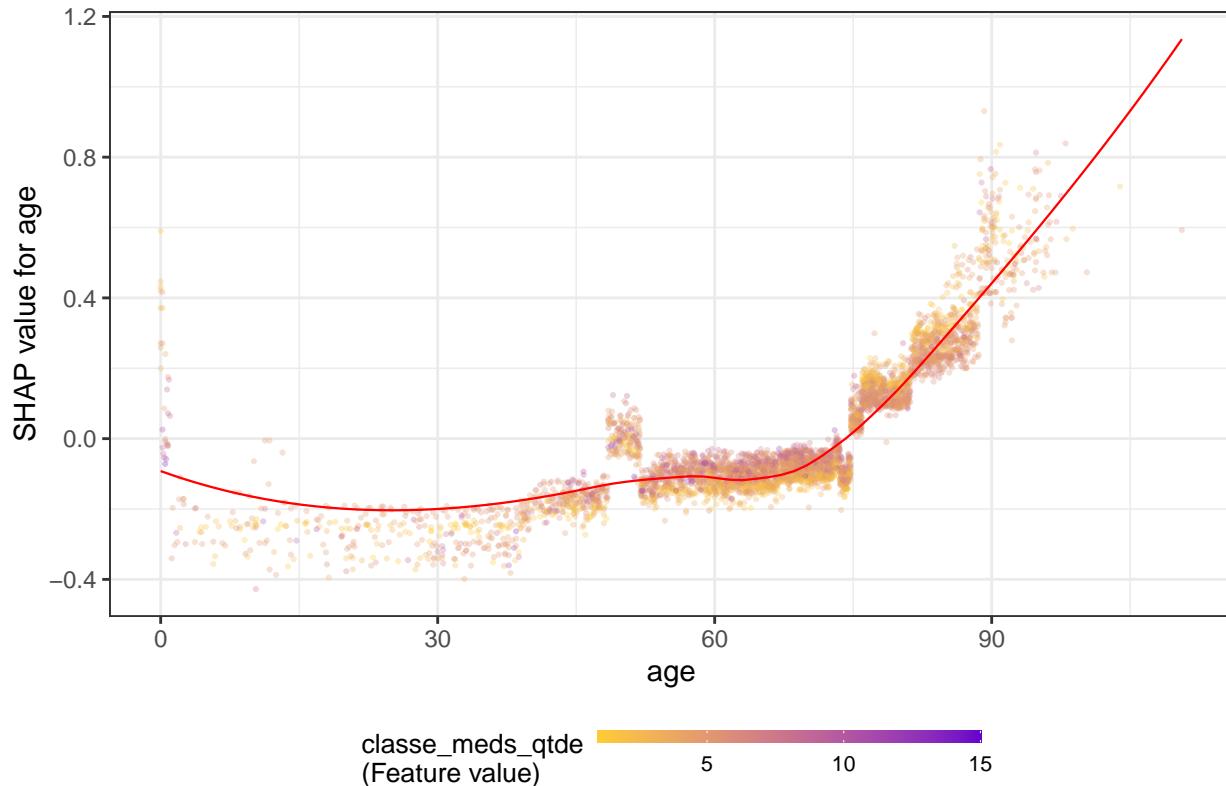


```
## `geom_smooth()` using formula 'y ~ x'
```

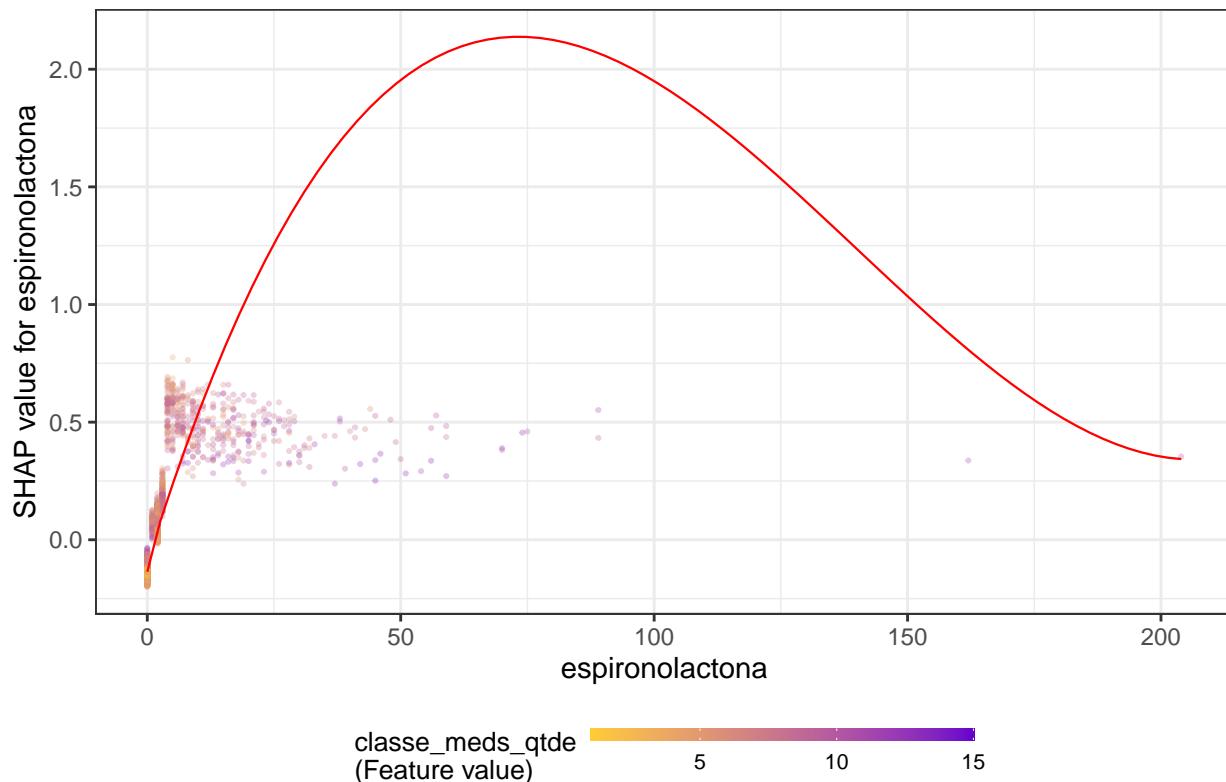
year_adm_t0



age



espironolactona

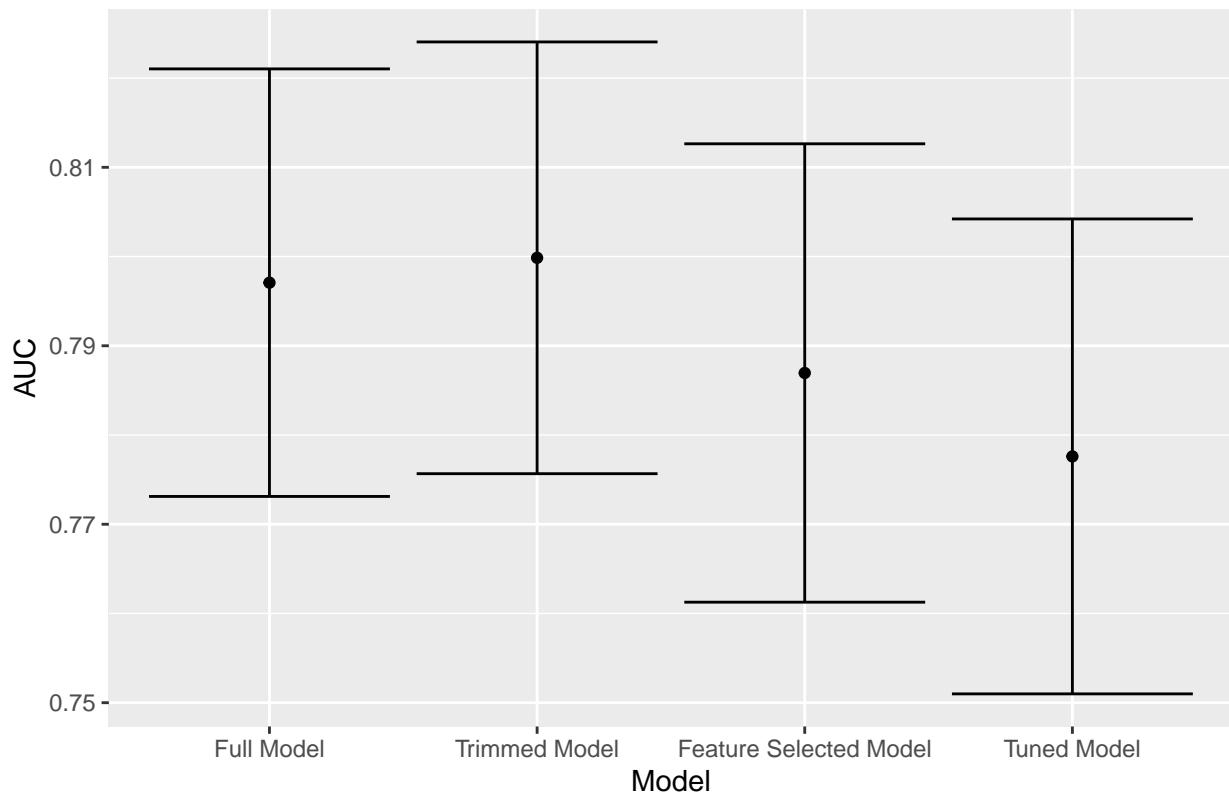


Models Comparison

```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper,
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper,
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower, feature_selected_model$auc_upper,
  'Tuned Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3],
  # 'Oversampled Model', as.numeric(oversampled_model$auc), oversampled_model$auc_lower, oversampled_model$auc_upper,
  # 'Undersampled Model', as.numeric(undersampled_model$auc), undersampled_model$auc_lower, undersampled_model$auc_upper
) %>%
  mutate(Target = outcome_column,
        Model = factor(Model,
                        levels = c('Full Model', 'Trimmed Model',
                                  'Feature Selected Model', 'Tuned Model',
                                  'Oversampled Model', 'Undersampled Model')))

df_auc %>%
  ggplot(aes(
    x = Model,
    y = AUC,
    ymin = `Lower Limit`,
    ymax = `Upper Limit`
  )) +
  geom_point() +
  geom_errorbar() +
  labs(title = outcome_column)
```

death_3year



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