

# Final Model - death\_180days

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. 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. classe\_meds\_qtd
46. meds\_cardiovasc\_qtd
47. meds\_antimicrobianos
48. vni
49. outros\_proced\_cirurgicos
50. icp
51. cateterismo

```

52. cateter_venoso_central
53. proced_invasivos_qtde
54. transfusao
55. interconsulta
56. equipe_multiprof
57. ecg
58. holter
59. teste_esforco
60. metodos_graficos_qtde
61. laboratorio
62. cultura
63. analises_clinicas_qtde
64. citologia
65. histopatologia_qtde
66. angiografia
67. aortografia
68. arteriografia
69. cintilografia
70. ecocardiograma
71. endoscopia
72. ultrassom
73. tomografia
74. radiografia
75. ressonancia
76. exames_imagem_qtde
77. 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(
    "./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: 0.765"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

```

## [1] "Full Model Test AUC: 0.817"

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. hemodialysis
3. dialysis\_t0
4. icp
5. transfusao
6. teste\_esforco
7. angiografia
8. aortografia
9. arteriografia
10. endoscopia

```

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: 0.764"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.819"

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
77	0.0000	outros_proced_cirurgicos
67	0.0010	vni
66	-0.0002	underlying_heart_disease
65	0.0029	nyha_basal

Table 1: (*continued*)

Number of Features	AUC Loss	Least Important Feature
64	0.0054	cardiac_arrest
63	0.0049	holter
62	0.0027	outros_proced_cirurgicos
61	0.0020	education_level
60	0.0042	cateter_venoso_central
59	0.0029	antifungico
58	0.0039	procedure_type_new
57	0.0033	digoxina
56	0.0000	antiplaquetario_ev
55	0.0078	aco
54	0.0057	cied_final_1
53	0.0049	heart_failure
52	0.0050	hypertension
51	0.0048	reop_type_1
50	0.0022	ressonancia
49	0.0069	cancer
48	0.0062	insulina
47	0.0074	valvopathy
46	0.0095	admission_t0_emergency
45	0.0052	histopatologia_qtde
44	0.0095	procedure_type_1
43	0.0103	bic

```

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)

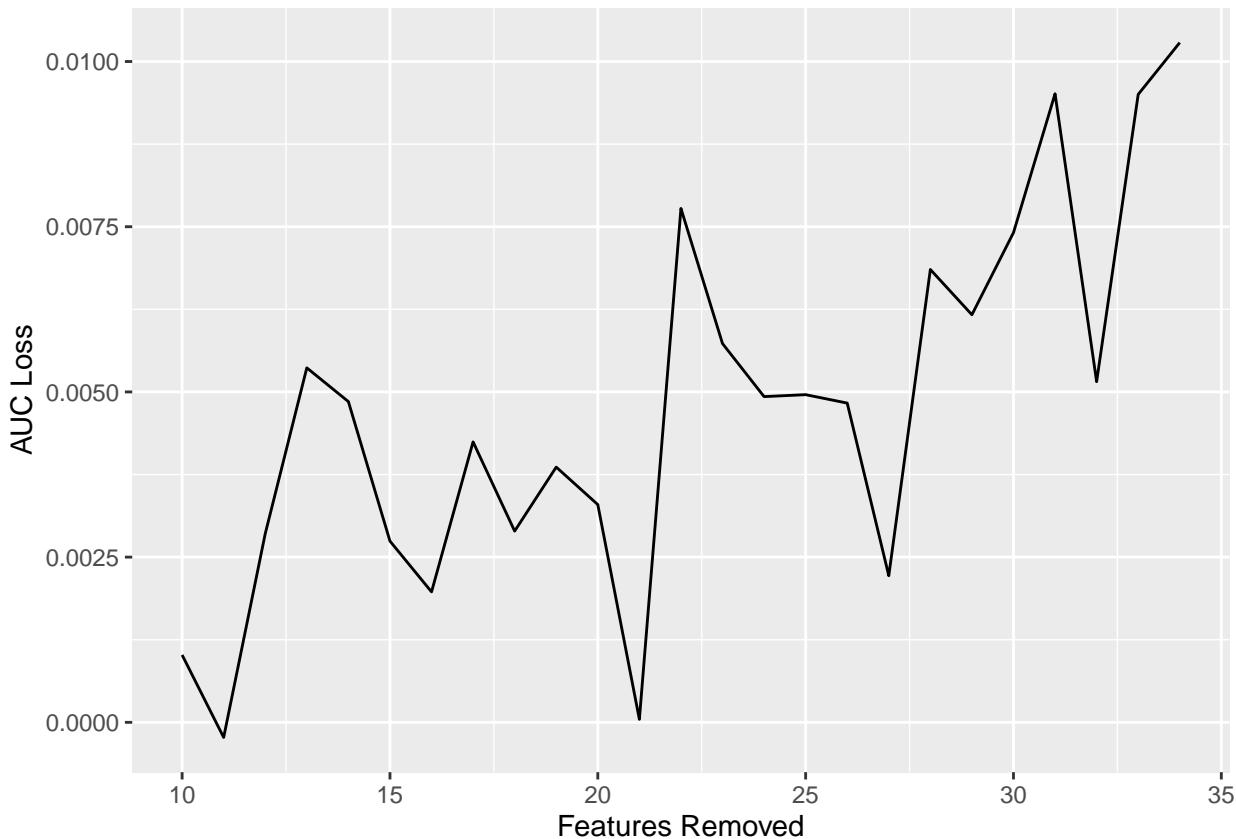
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.756"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.806"

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. laboratorio
2. vasodilatador
3. analises\_clinicas\_qtde
4. age
5. meds\_cardiovasc\_qtde
6. year\_adm\_t0
7. comorbidities\_count
8. ieca\_bra
9. icu\_t0
10. admission\_pre\_t0\_count
11. espironolactona
12. equipe\_multiprof
13. estatina
14. diuretico
15. antiaritmico
16. admission\_pre\_t0\_180d
17. classe\_meds\_qtde
18. exames\_imagem\_qtde
19. insuf\_cardiaca
20. interconsulta
21. dva
22. metodos\_graficos\_qtde
23. meds\_antimicrobianos
24. citologia
25. radiografia
26. ecg
27. cied\_final\_group\_1
28. cateterismo

```

29. proced_invasivos_qtde
30. af
31. cintilografia
32. ecocardiograma
33. stroke
34. psicofarmacos
35. cultura
36. ultrassom
37. renal_failure
38. diabetes
39. sex
40. tomografia
41. copd
42. bic
43. prior_mi

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)

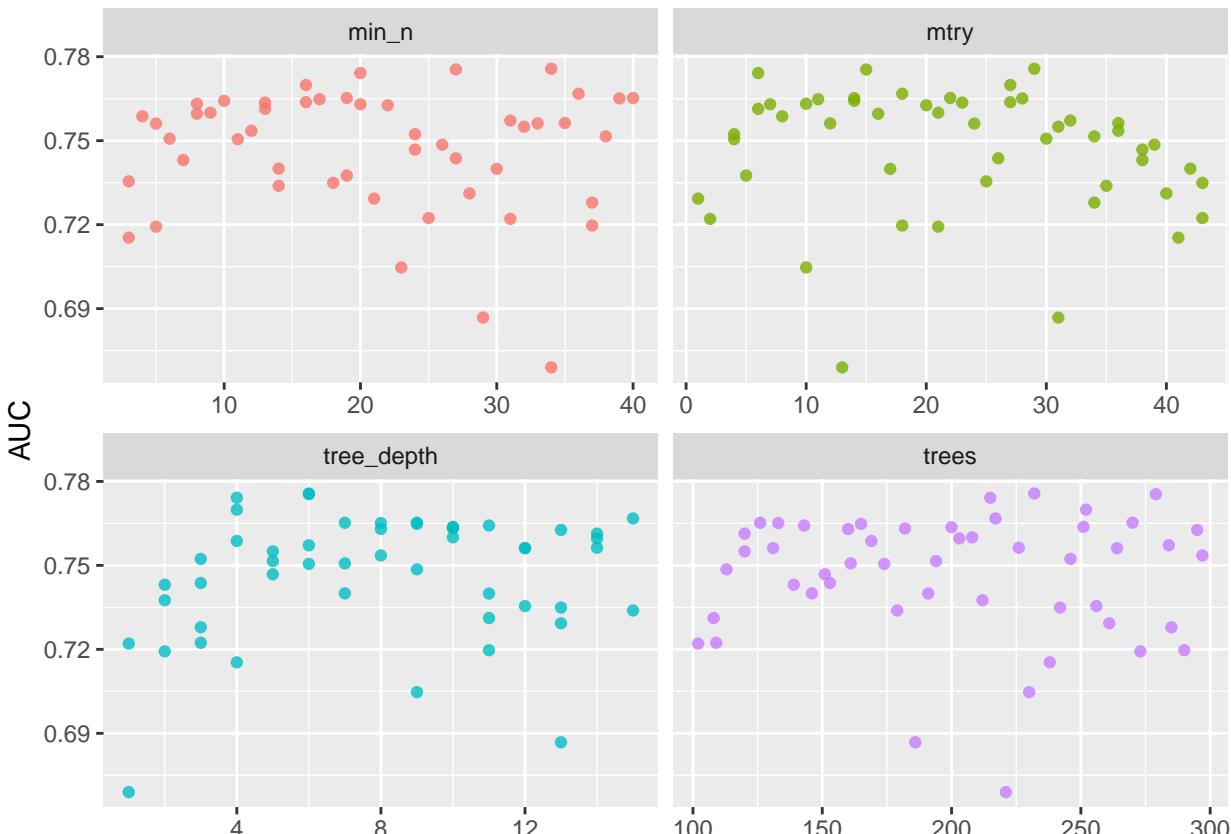
```

Table 2:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
29	232	34	6	0.00516	0.00000	roc_auc	binary	0.77569	4	0.02665	Preprocessor1_
15	279	27	6	0.00618	0.00000	roc_auc	binary	0.77544	4	0.02815	Preprocessor1_
6	215	20	4	0.02241	0.00002	roc_auc	binary	0.77417	4	0.02497	Preprocessor1_
27	252	16	4	0.01244	0.00018	roc_auc	binary	0.76990	4	0.02792	Preprocessor1_
18	217	36	15	0.00034	0.00041	roc_auc	binary	0.76677	4	0.02409	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")
```

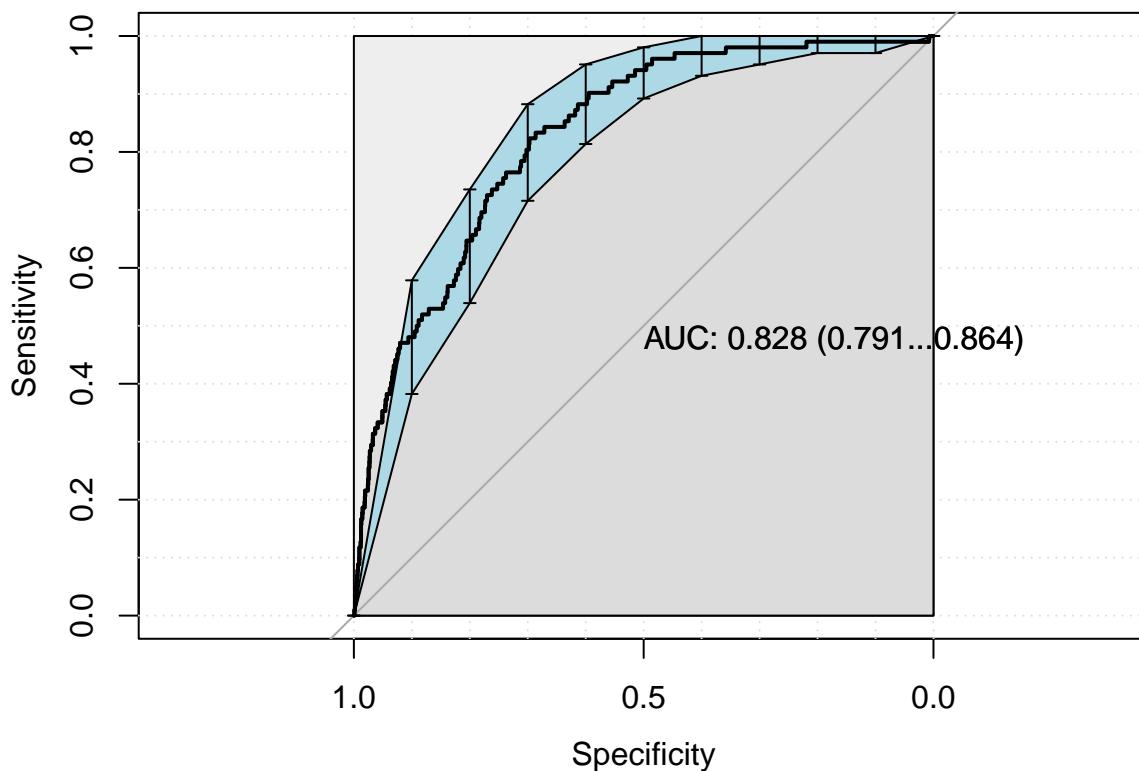


```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)
```

```
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 4628  102  
##                      1      0      0  
##  
##          Accuracy : 0.9784  
##          95% CI  : (0.9739, 0.9824)  
##          No Information Rate : 0.9784  
##          P-Value [Acc > NIR] : 0.5263  
##  
##          Kappa : 0  
##  
##  McNemar's Test P-Value : <2e-16  
##  
##          Sensitivity : 1.0000  
##          Specificity  : 0.0000  
##          Pos Pred Value : 0.9784  
##          Neg Pred Value :     NaN  
##          Prevalence   : 0.9784  
##          Detection Rate : 0.9784  
##          Detection Prevalence : 1.0000  
##          Balanced Accuracy : 0.5000  
##  
##          'Positive' Class : 0  
##  
lightgbm_parameters <- lightgbm_tune %>%  
  show_best("roc_auc", n = 1) %>%  
  select(trees, mtry, min_n, tree_depth, learn_rate, loss_reduction) %>%  
  as.list
```

```

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)

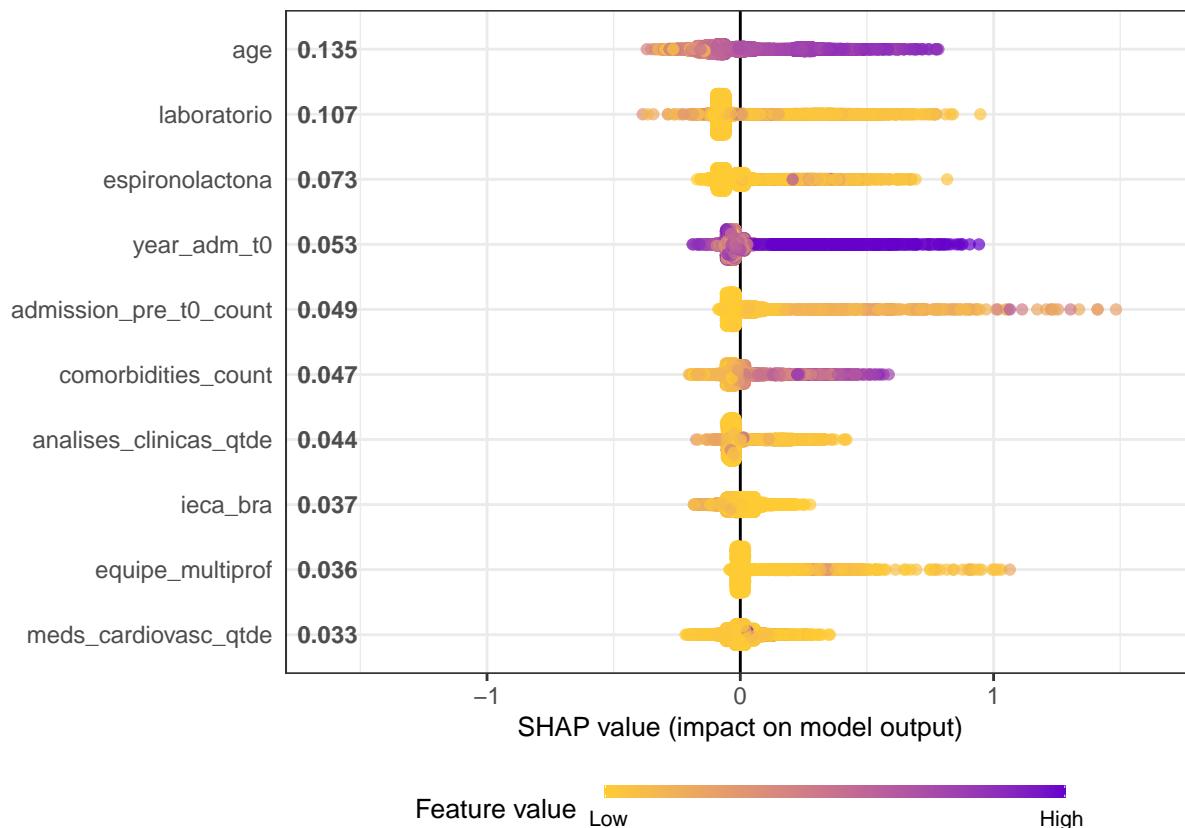
trained_rec <- prep(lightgbm_recipe, training = df_train)

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)))

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
  )
}

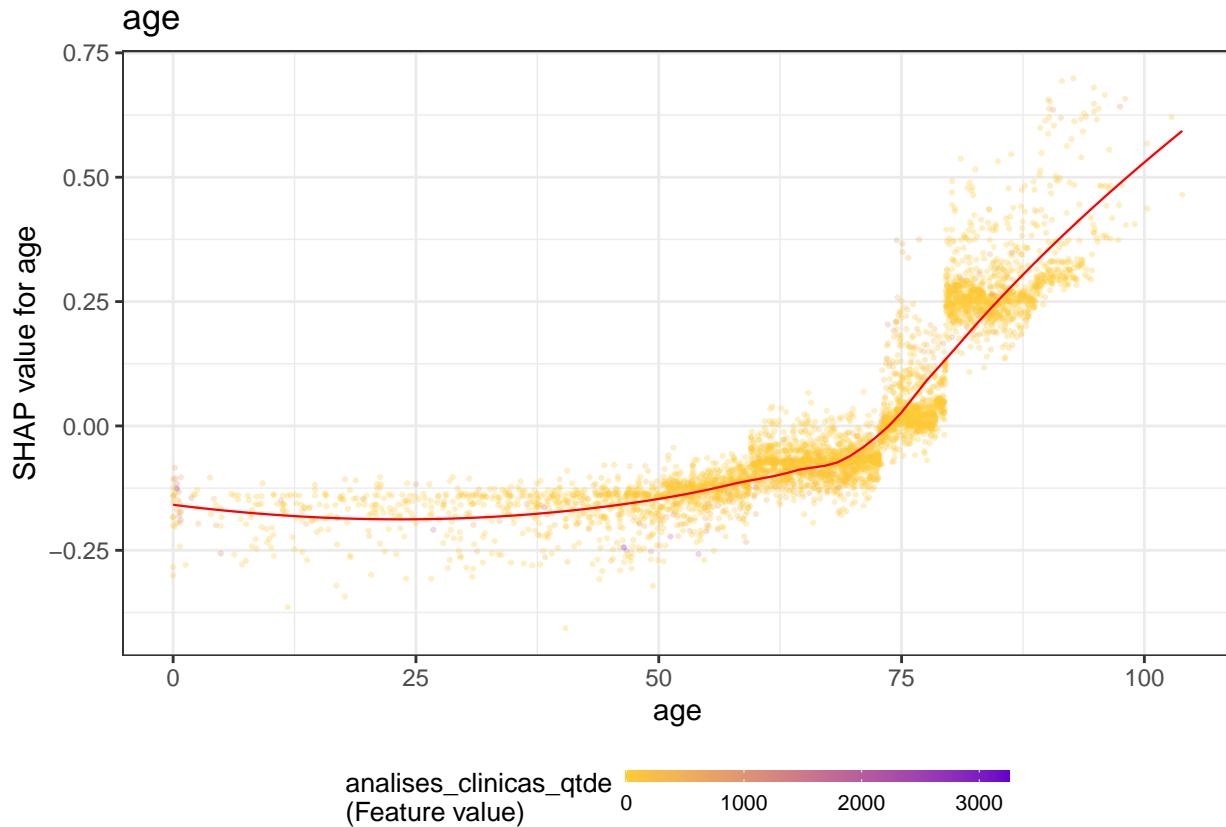
```

```

) +
  labs(title = x)
print(p)
}

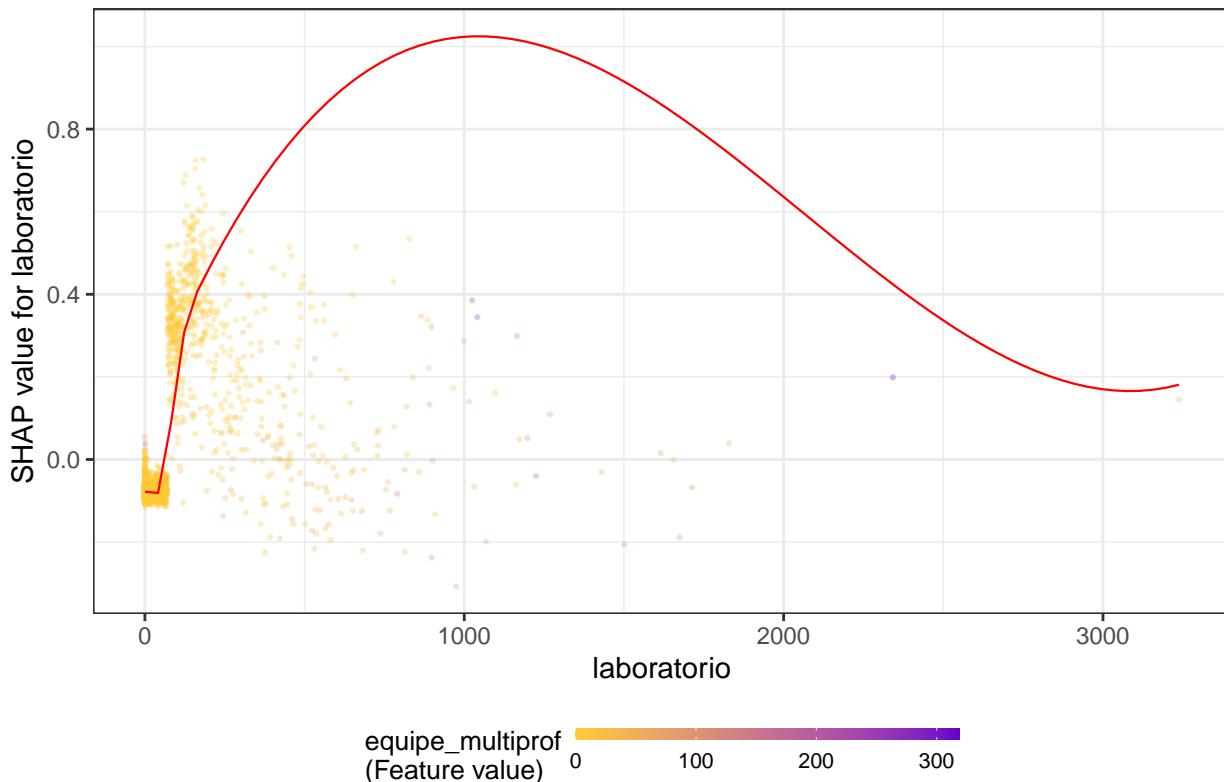
```

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



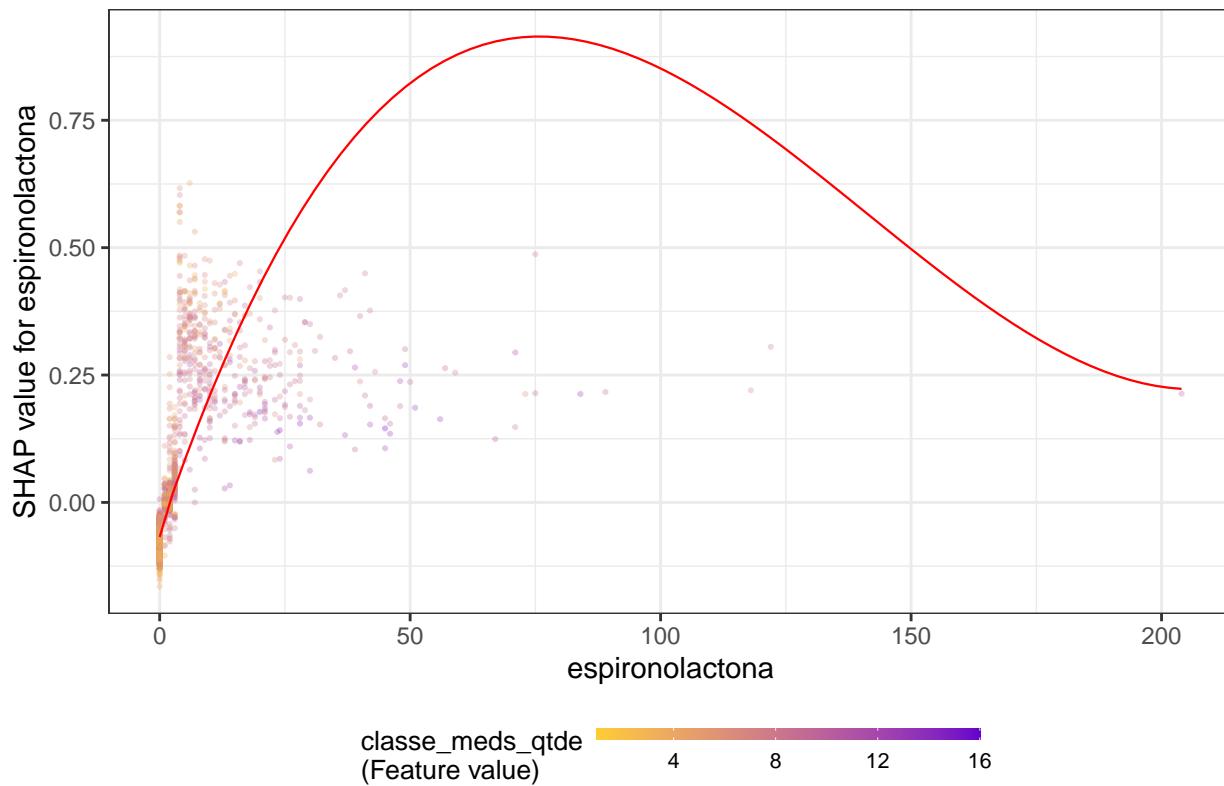
```
## `geom_smooth()` using formula 'y ~ x'
```

**laboratorio**



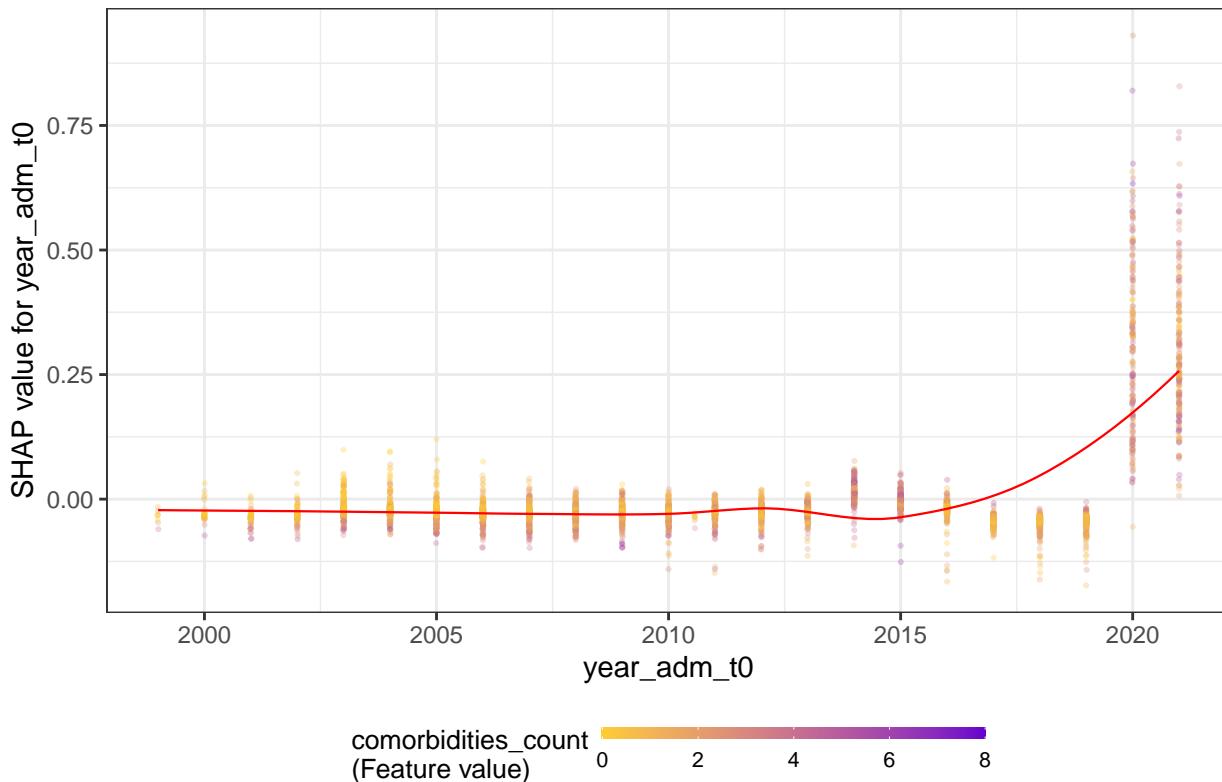
```
## `geom_smooth()` using formula 'y ~ x'
```

### espironolactona



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

### year\_adm\_t0



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

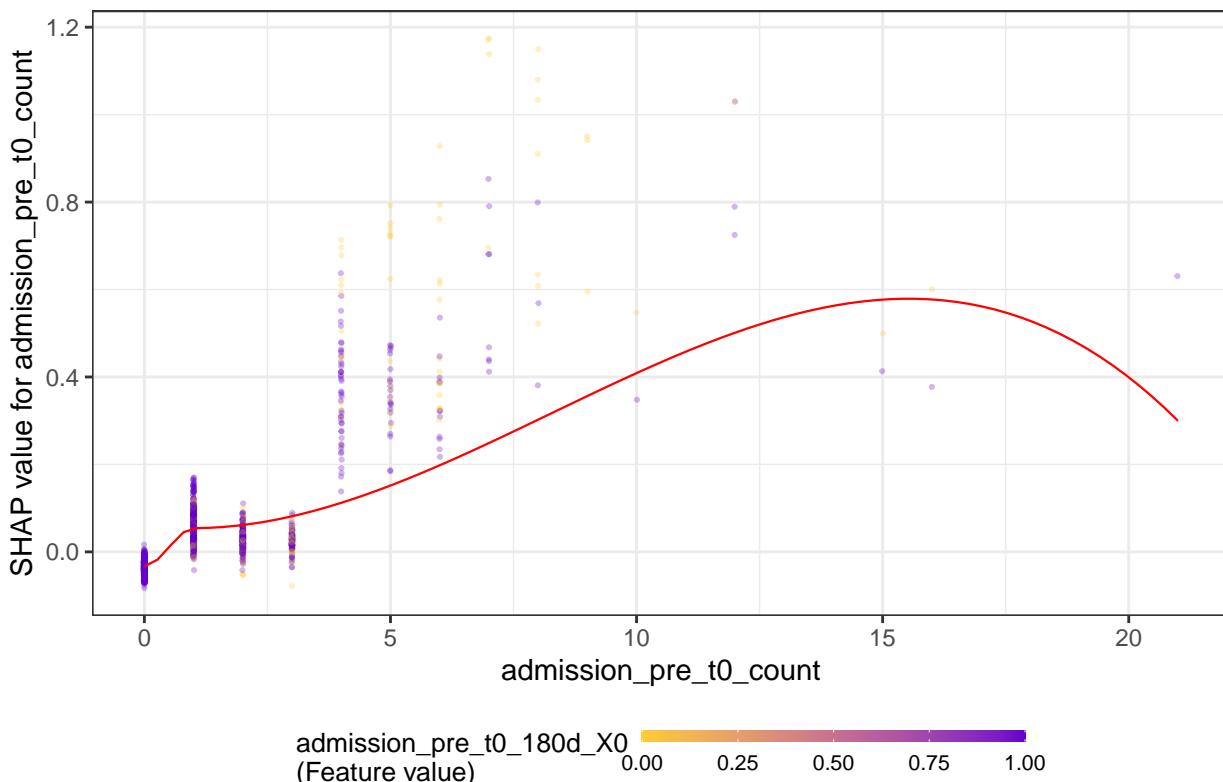
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at -0.10000000000000009
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius 1.105
```

```

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition number
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near singular
## well. 1

```

admission\_pre\_t0\_count



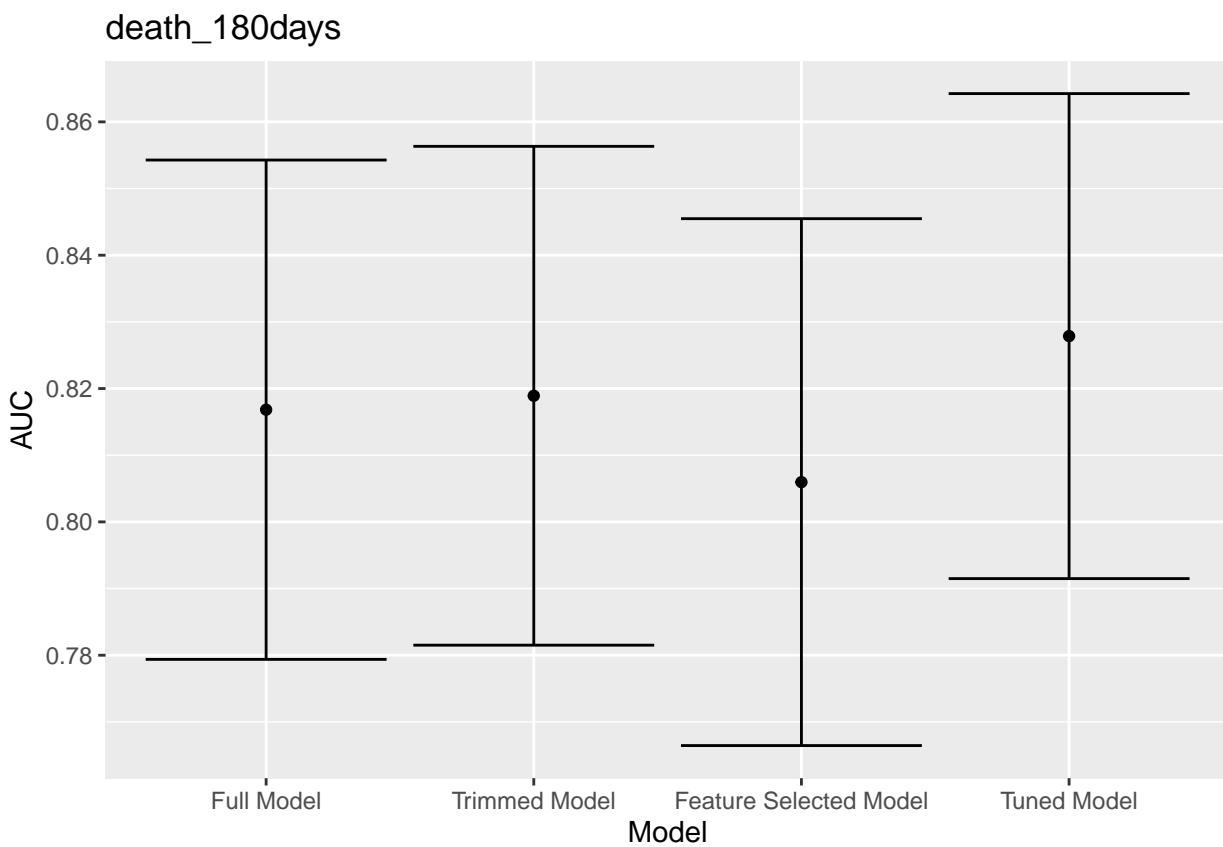
## Models Comparison

```

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)

```



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