# 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. antiarritmico
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(
    "./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: 0.792"
```

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

```
## [1] "Full Model Test AUC: 0.800"
```

Features with zero importance on the initial model:

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

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

1. angiografia

```
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)
```

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

```
## [1] "Trimmed Model CV Train AUC: 0.795"
```

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

```
## [1] "Trimmed Model Test AUC: 0.800"
```

```r
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_droped <- 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 | angioplastia |
| 80 | -0.0028 | angioplastia |
| 79 | -0.0011 | histopatologia_qtde |
| 78 | -0.0007 | teste_esforco |
| 77 | -0.0008 | heart_disease |
| 76 | -0.0004 | cateter_venoso_central |
| 75 | -0.0026 | hemodialysis |
| 74 | -0.0009 | holter |

Table 1: *(continued)*

| Number of Features | AUC Loss | Least Important Feature |
|---|---|---|
| 73 | -0.0013 | transfusao |
| 72 | -0.0001 | angio_tc |
| 71 | -0.0010 | tilt_teste |
| 70 | -0.0010 | dialysis_t0 |
| 69 | -0.0021 | stroke |
| 68 | -0.0002 | endoscopia |
| 67 | -0.0029 | antiplaquetario_ev |
| 66 | 0.0000 | pet_ct |
| 65 | -0.0013 | eletrofisiologia |
| 64 | -0.0029 | cintilografia |
| 63 | -0.0028 | cancer |
| 62 | 0.0001 | cardiac_arrest |
| 61 | -0.0032 | icp |
| 60 | -0.0015 | copd |
| 59 | 0.0004 | antifungico |
| 58 | -0.0005 | ressonancia |
| 57 | -0.0039 | procedure_type_new |
| 56 | -0.0024 | flebografia |
| 55 | 0.0000 | procedure_type_1 |
| 54 | -0.0006 | outros_proced_cirurgicos |
| 53 | 0.0000 | cateterismo |
| 52 | 0.0002 | citologia |
| 51 | -0.0006 | aco |
| 50 | -0.0012 | diabetes |
| 49 | -0.0018 | prior_mi |
| 48 | 0.0029 | insulina |
| 47 | 0.0000 | sex |
| 46 | 0.0007 | ecocardiograma |
| 45 | -0.0016 | heart_failure |
| 44 | 0.0008 | cultura |
| 43 | 0.0007 | admission_t0_emergency |
| 42 | -0.0005 | anticonvulsivante |
| 41 | 0.0006 | renal_failure |
| 40 | -0.0001 | hypertension |
| 39 | 0.0012 | race |
| 38 | 0.0021 | underlying_heart_disease |
| 37 | 0.0043 | bic |
| 36 | 0.0015 | valvopathy |
| 35 | 0.0025 | dva |
| 34 | 0.0019 | proced_invasivos_qtde |
| 33 | 0.0010 | digoxina |
| 32 | 0.0031 | tomografia |
| 31 | 0.0036 | af |
| 30 | 0.0050 | radiografia |
| 29 | 0.0028 | analises_clinicas_qtde |
| 28 | 0.0012 | cied_final_group_1 |
| 27 | 0.0011 | admission_pre_t0_180d |
| 26 | 0.0030 | ultrassom |
| 25 | 0.0026 | cied_final_1 |
| 24 | 0.0066 | antiarritmico |
| 23 | 0.0049 | ecg |
| 22 | 0.0038 | nyha_basal |
| 21 | 0.0128 | estatina |

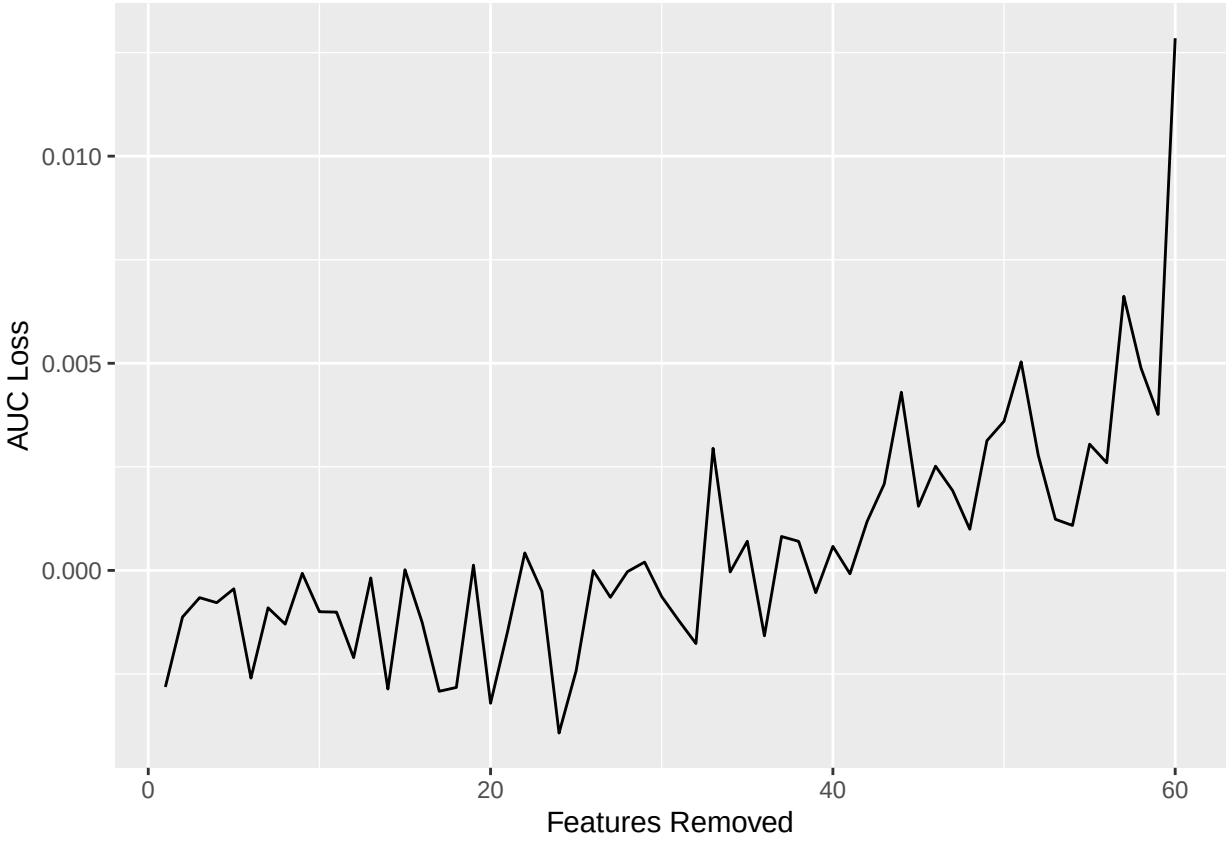
```
if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_droped)
} else {
  selected_features <- current_features
}

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: 0.779"
```

```
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)
```

```
## [1] "Trimmed Model Test AUC: 0.786"
```

```
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. admission_pre_t0_count

3. year_adm_t0
4. classe_meds_qtde
5. espironolactona
6. laboratorio
7. comorbidities_count
8. diuretico
9. education_level
10. icu_t0
11. vasodilatador
12. meds_cardiovasc_qtde
13. ieca_bra
14. insuf_cardiaca
15. equipe_multiprof
16. meds_antimicrobianos
17. exames_imagem_qtde
18. psicofarmacos
19. metodos_graficos_qtde
20. estatina
21. reop_type_1

```
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)
```

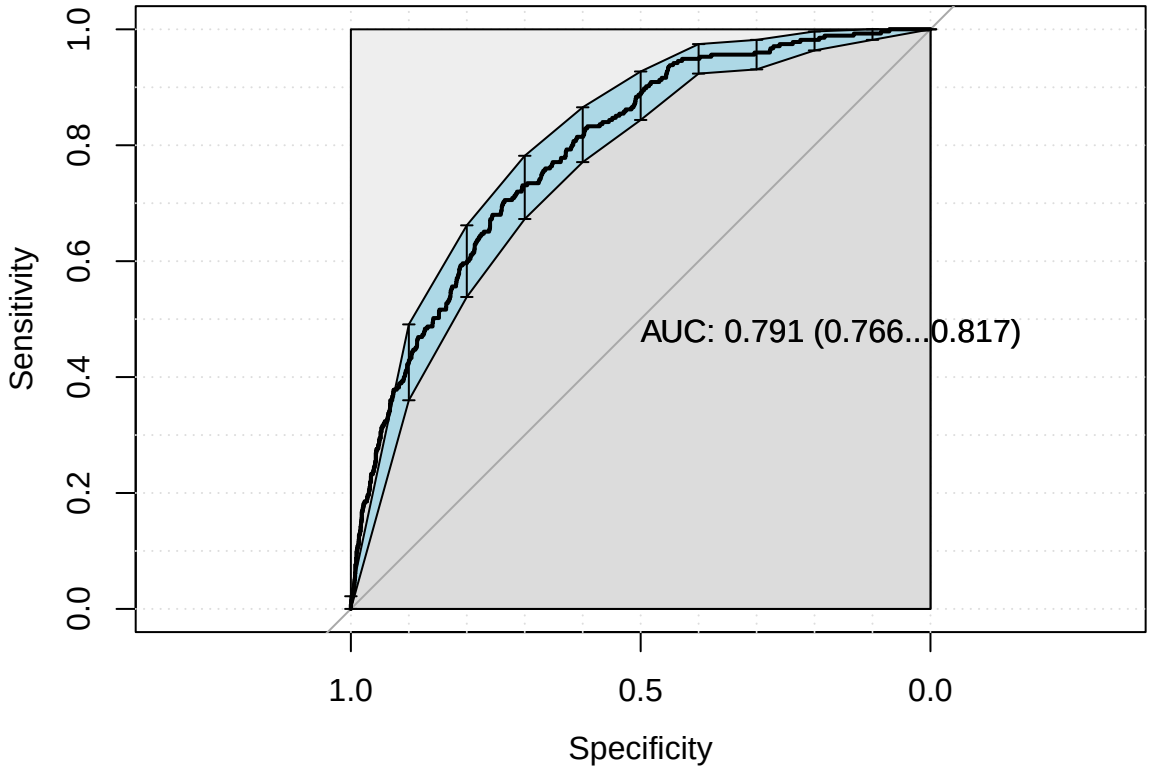
Table 2:

| mtry | trees | min_n | tree_depth | learn_rate | loss_reduction | .metric | .estimator | mean | n | std_err | .config |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | 253 | 34 | 4 | 0.02199 | 0.00001 | roc_auc | binary | 0.78603 | 4 | 0.00503 | Preprocessor1_ |
| 10 | 148 | 33 | 10 | 0.01183 | 0.00000 | roc_auc | binary | 0.78393 | 4 | 0.00593 | Preprocessor1_ |
| 13 | 164 | 11 | 2 | 0.04898 | 0.00086 | roc_auc | binary | 0.77629 | 4 | 0.00400 | Preprocessor1_ |
| 17 | 158 | 24 | 14 | 0.00822 | 0.00011 | roc_auc | binary | 0.77497 | 4 | 0.00470 | Preprocessor1_ |
| 3 | 116 | 11 | 4 | 0.04320 | 0.35641 | roc_auc | binary | 0.77392 | 4 | 0.00705 | 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)
```

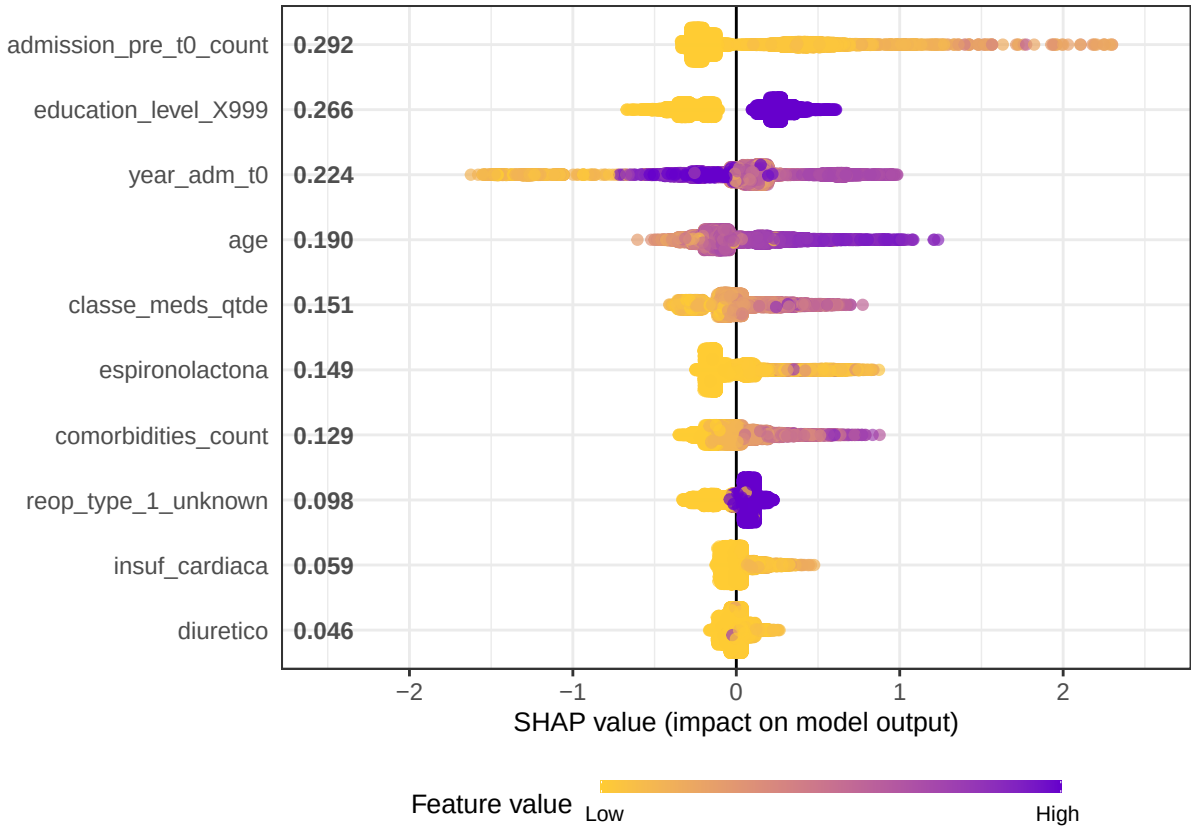



```
##   |
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                      0 4449  271
##                      1    6    4
##
##                Accuracy : 0.9414
##                  95% CI : (0.9344, 0.948)
##     No Information Rate : 0.9419
##     P-Value [Acc > NIR] : 0.5652
##
##                   Kappa : 0.0241
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99865
##             Specificity : 0.01455
##          Pos Pred Value : 0.94258
##          Neg Pred Value : 0.40000
##              Prevalence : 0.94186
##          Detection Rate : 0.94059
##    Detection Prevalence : 0.99789
##       Balanced Accuracy : 0.50660
##
##        '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

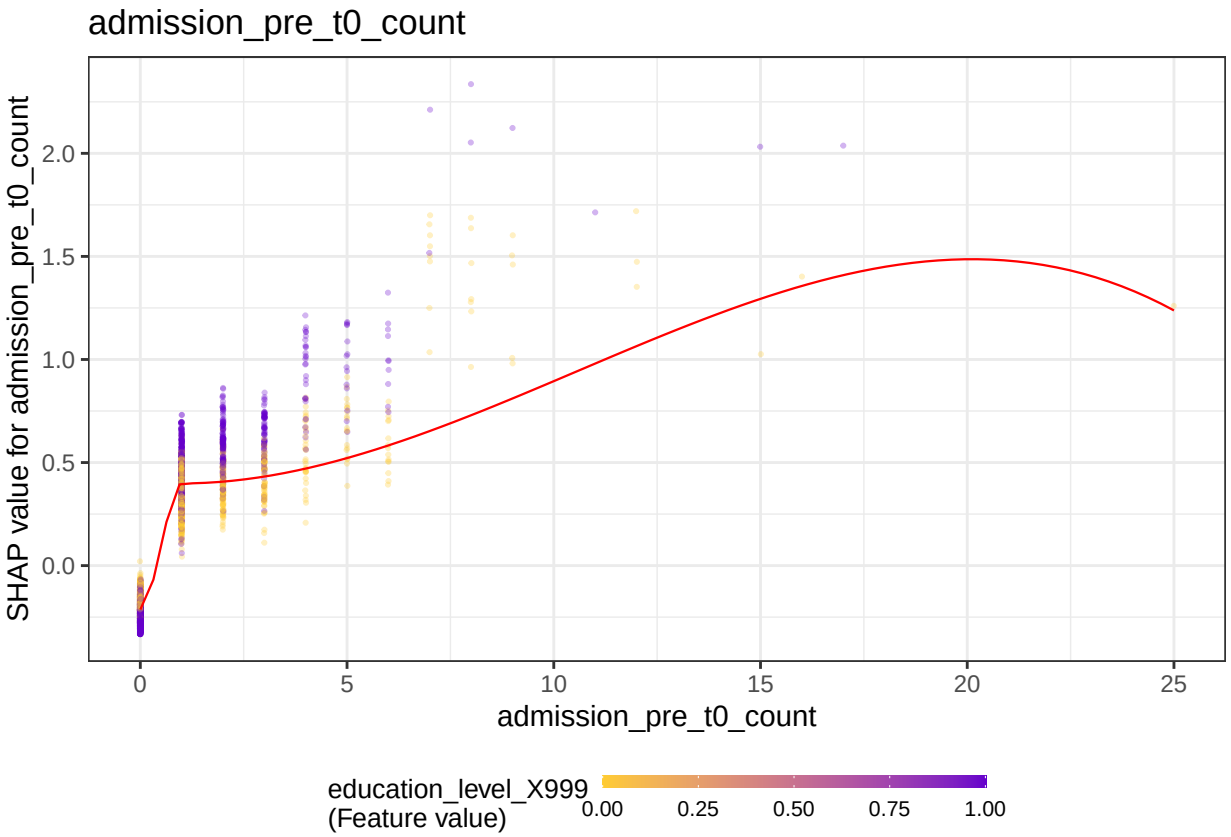
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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)
```



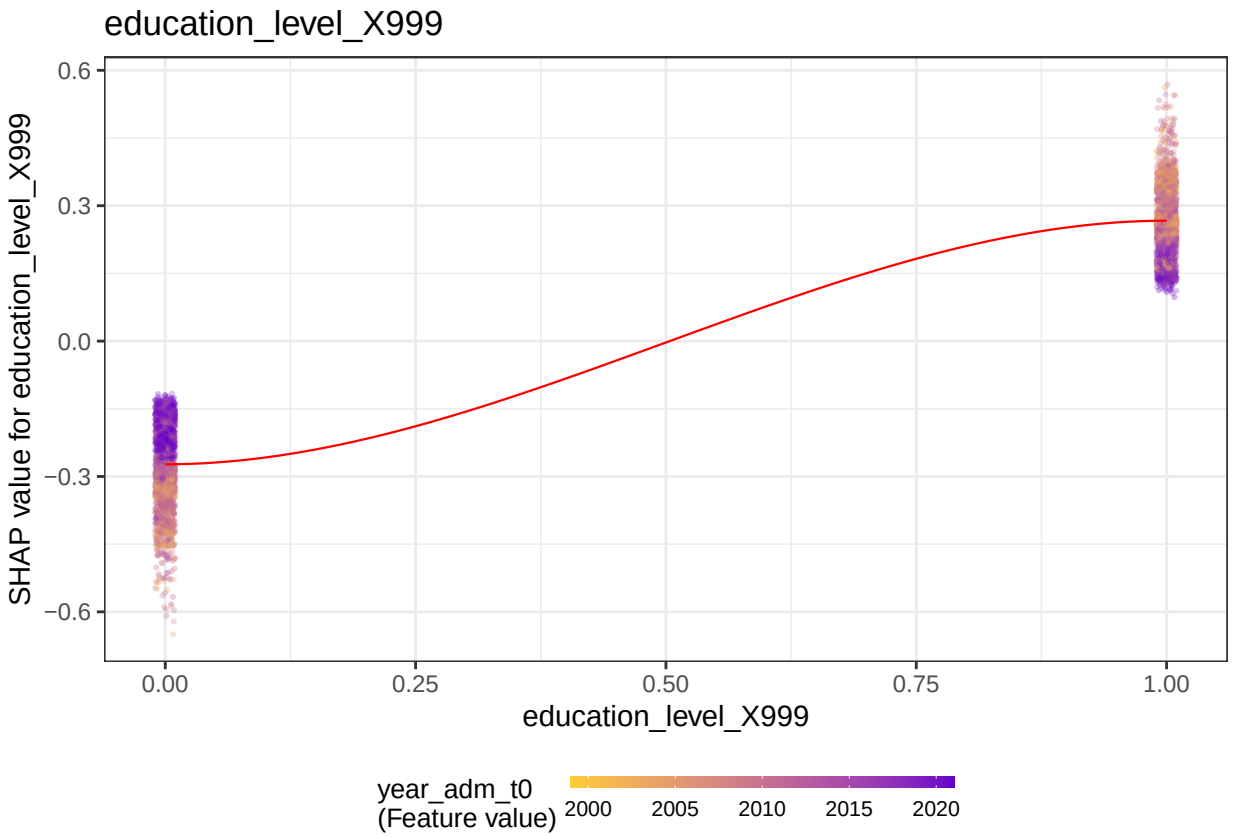

```
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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)
}
```

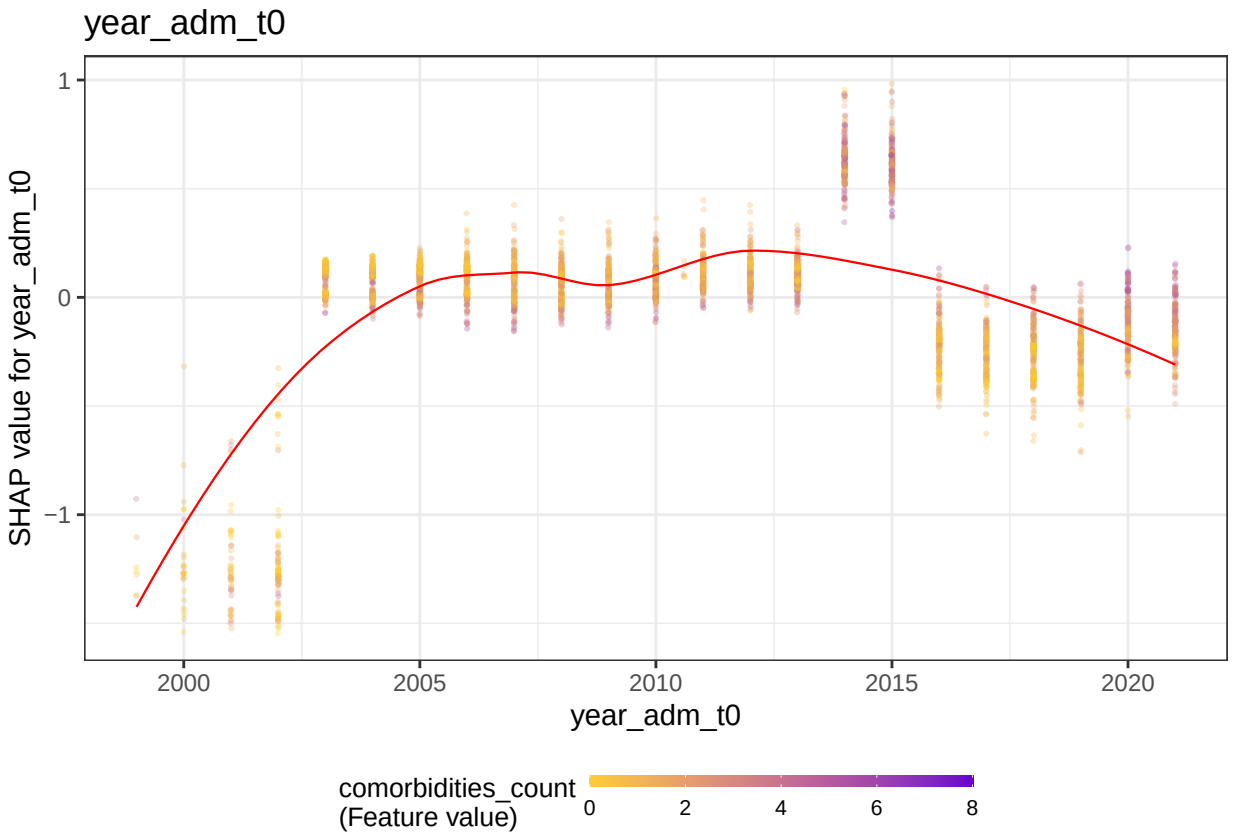
```
## `geom_smooth()` using formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at -0.12
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius 1.125
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition number
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near singul
## well. 1
```



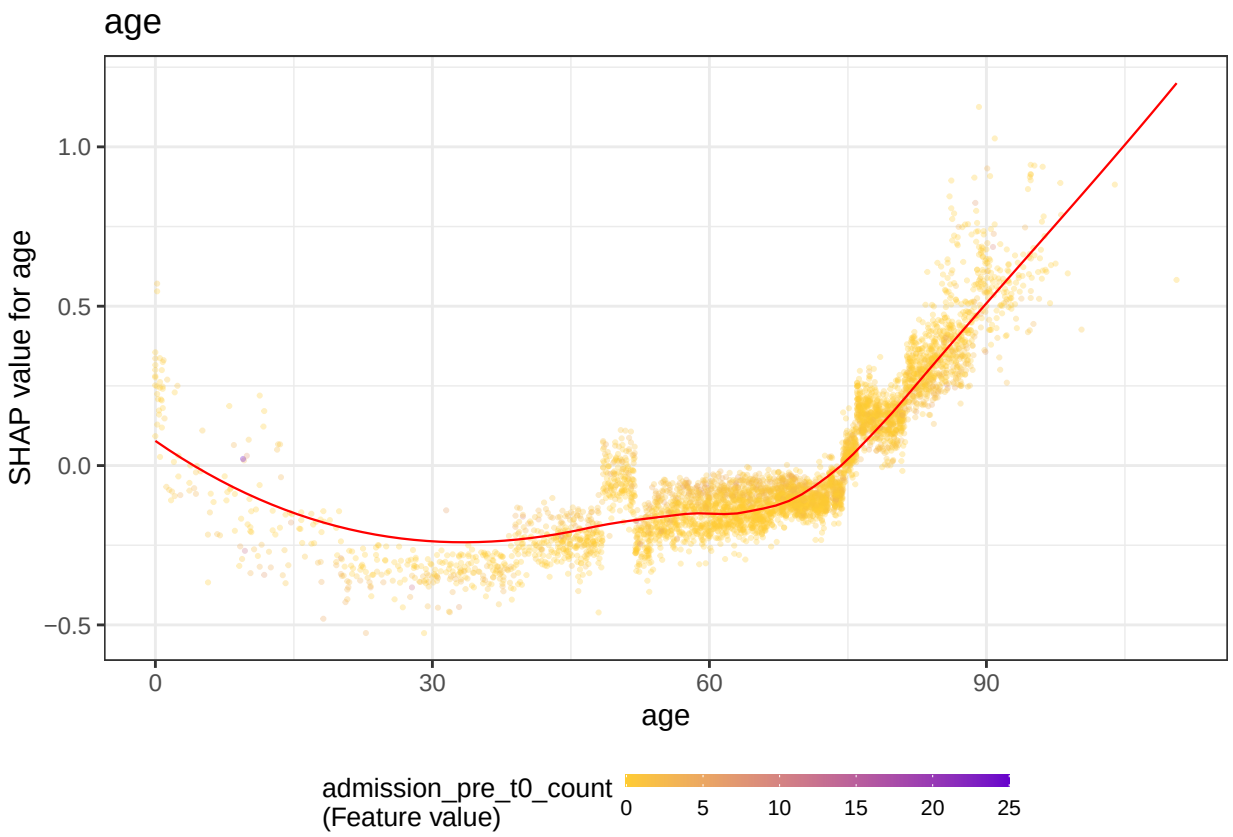

```
## `geom_smooth()` using formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at -0.00
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius 1.005
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition number
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near singul
## well. 1.01
```



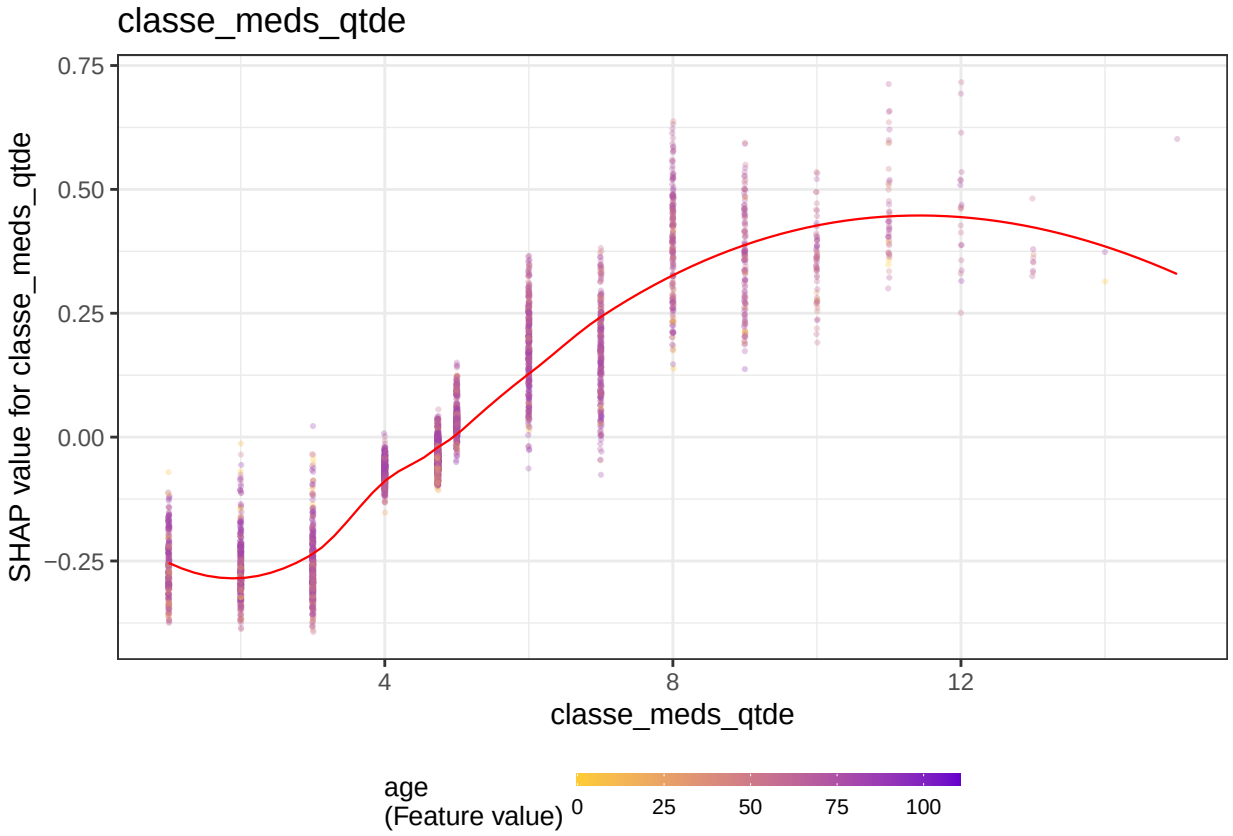

```
## `geom_smooth()` using formula 'y ~ x'
```




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



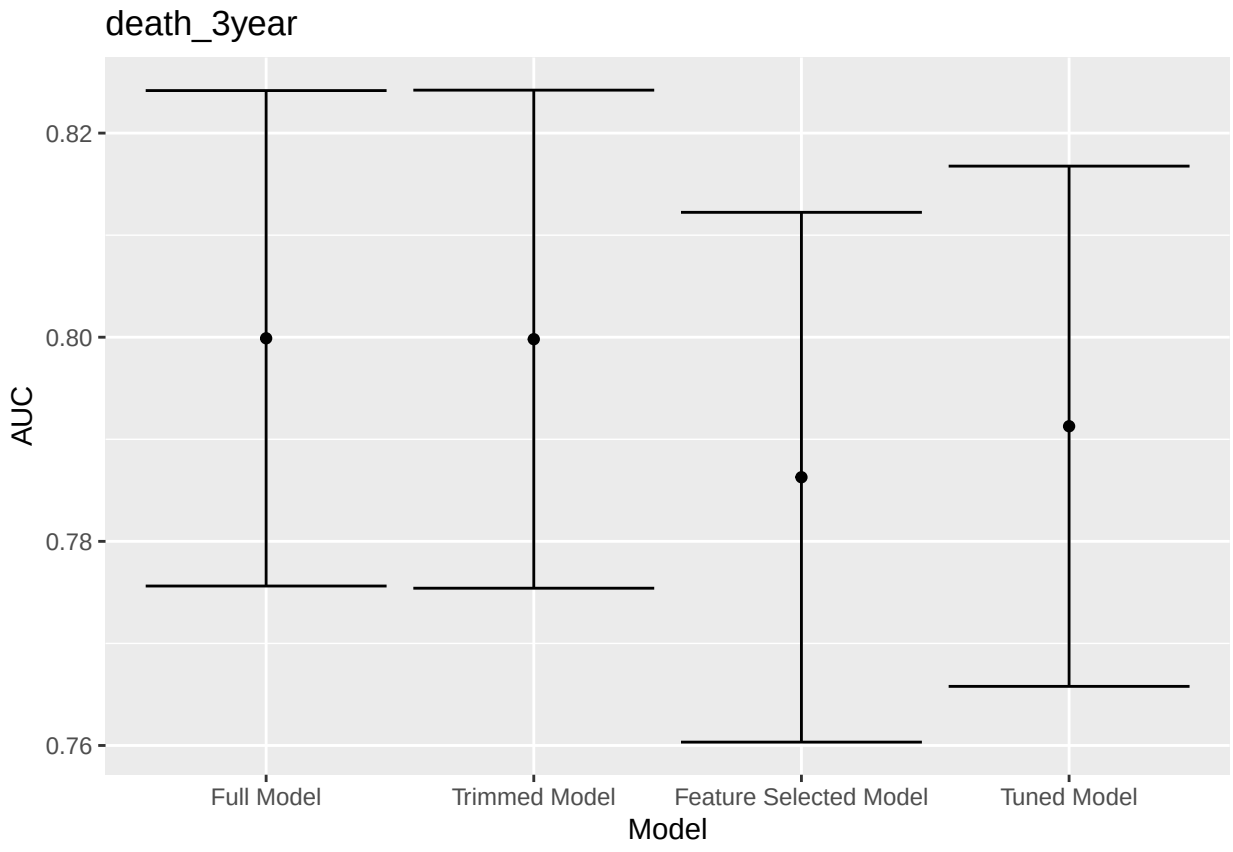

```
## `geom_smooth()` using formula 'y ~ x'
```




## 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$auc_lower, feature_selected_model
  'Tuned Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3],
  # 'Oversampled Model', as.numeric(oversampled_model$auc), oversampled_model$auc_lower, oversampled_model$auc_u
  # 'Undersampled Model', as.numeric(undersampled_model$auc), undersampled_model$auc_lower, undersampled_model$a
) %>%
  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))
```