Final Model - death_3year

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. 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. ventilacao_mecanica
51. transplante_cardiaco
52. outros_proced_cirurgicos
53. icp
54. angioplastia
55. cateterismo
56. eletrofisiologia
57. cateter_venoso_central
58. proced_invasivos_qtde
59. transfusao
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. cintilografia
73. ecocardiograma
74. endoscopia
75. flebografia
76. pet_ct
77. ultrassom
78. tomografia
79. ressonancia
80. exames_imagem_qtde
81. bic
82. 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: 0.799"
```

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

```
## [1] "Full Model Test AUC: 0.793"
```

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. procedure_type_new
4. antiplaquetario_ev
5. insulina
6. transplante_cardiaco
7. angioplastia
8. transfusao
9. holter
10. tilt_teste
11. histopatologia_qtde
12. angiografia
13. cintilografia
14. flebografia
15. 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: 0.798"
```

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

```
## [1] "Trimmed Model Test AUC: 0.796"
```

```
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_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),
            `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
82	0.7990	0.0065	0.0000	heart_failure
67	0.7975	0.0065	0.0015	endoscopia
66	0.7999	0.0066	-0.0008	outros_proced_cirurgicos
65	0.7979	0.0071	0.0011	antifungico
64	0.7989	0.0071	0.0001	cultura
63	0.7988	0.0070	0.0003	cateter_venoso_central
62	0.7999	0.0060	-0.0008	angio_tc
61	0.7988	0.0072	0.0002	ecocardiograma
60	0.7994	0.0072	-0.0004	eletrofisiologia
59	0.7993	0.0070	-0.0003	hypertension
58	0.7990	0.0070	0.0000	copd
57	0.7988	0.0071	0.0003	tomografia
56	0.7995	0.0068	-0.0005	admission_pre_t0_180d
55	0.7990	0.0070	0.0000	af
54	0.7993	0.0070	-0.0002	digoxina
53	0.8000	0.0068	-0.0010	analises_clinicas_qtde
52	0.8002	0.0069	-0.0011	teste_esforco

Table 1: *(continued)*

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
51	0.8009	0.0071	-0.0019	proced_invasivos_qtde
50	0.8024	0.0061	-0.0033	ressonancia
49	0.8019	0.0060	-0.0028	bic
48	0.8019	0.0064	-0.0028	icp
47	0.8018	0.0063	-0.0028	anticonvulsivante
46	0.8024	0.0064	-0.0033	dialysis_t0
45	0.8023	0.0061	-0.0032	reop_type_1
44	0.8020	0.0060	-0.0029	procedure_type_1
43	0.8024	0.0061	-0.0034	cancer
42	0.8022	0.0057	-0.0031	cateterismo
41	0.8020	0.0061	-0.0030	dva
40	0.8024	0.0063	-0.0033	sex
39	0.8026	0.0067	-0.0035	prior_mi
38	0.8028	0.0069	-0.0038	diabetes
37	0.8024	0.0068	-0.0034	race
36	0.8006	0.0071	-0.0015	citologia
35	0.8007	0.0071	-0.0017	estatina
34	0.8015	0.0070	-0.0024	admission_t0_emergency
33	0.8013	0.0069	-0.0023	aco
32	0.8014	0.0068	-0.0023	equipe_multiprof
31	0.8013	0.0066	-0.0023	ultrassom
30	0.8009	0.0064	-0.0019	ventilacao_mecanica
29	0.8009	0.0063	-0.0019	renal_failure
28	0.8009	0.0064	-0.0019	valvopathy
27	0.8016	0.0062	-0.0026	psicofarmacos
26	0.8026	0.0058	-0.0036	cied_final_1
25	0.8006	0.0072	-0.0015	underlying_heart_disease
24	0.8017	0.0067	-0.0027	exames_imagem_qtde
23	0.8013	0.0069	-0.0023	antiarritmico
22	0.8008	0.0076	-0.0018	icu_t0
21	0.8019	0.0075	-0.0029	metodos_graficos_qtde
20	0.8014	0.0071	-0.0023	meds_cardiovasc_qtde
19	0.8018	0.0072	-0.0027	heart_failure
18	0.8021	0.0072	-0.0031	cardiac_arrest
17	0.8021	0.0076	-0.0031	cied_final_group_1
16	0.8008	0.0069	-0.0017	vasodilatador
15	0.8005	0.0067	-0.0015	meds_antimicrobianos
14	0.8018	0.0060	-0.0028	laboratorio
13	0.8026	0.0063	-0.0036	insuf_cardiaca
12	0.8030	0.0065	-0.0039	diuretico
11	0.8019	0.0067	-0.0029	nyha_basal
10	0.7912	0.0091	0.0078	ieca_bra
9	0.7900	0.0090	0.0090	classe_meds_qtde
8	0.7909	0.0105	0.0081	education_level
7	0.7780	0.0107	0.0210	comorbidities_count

```
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")
```

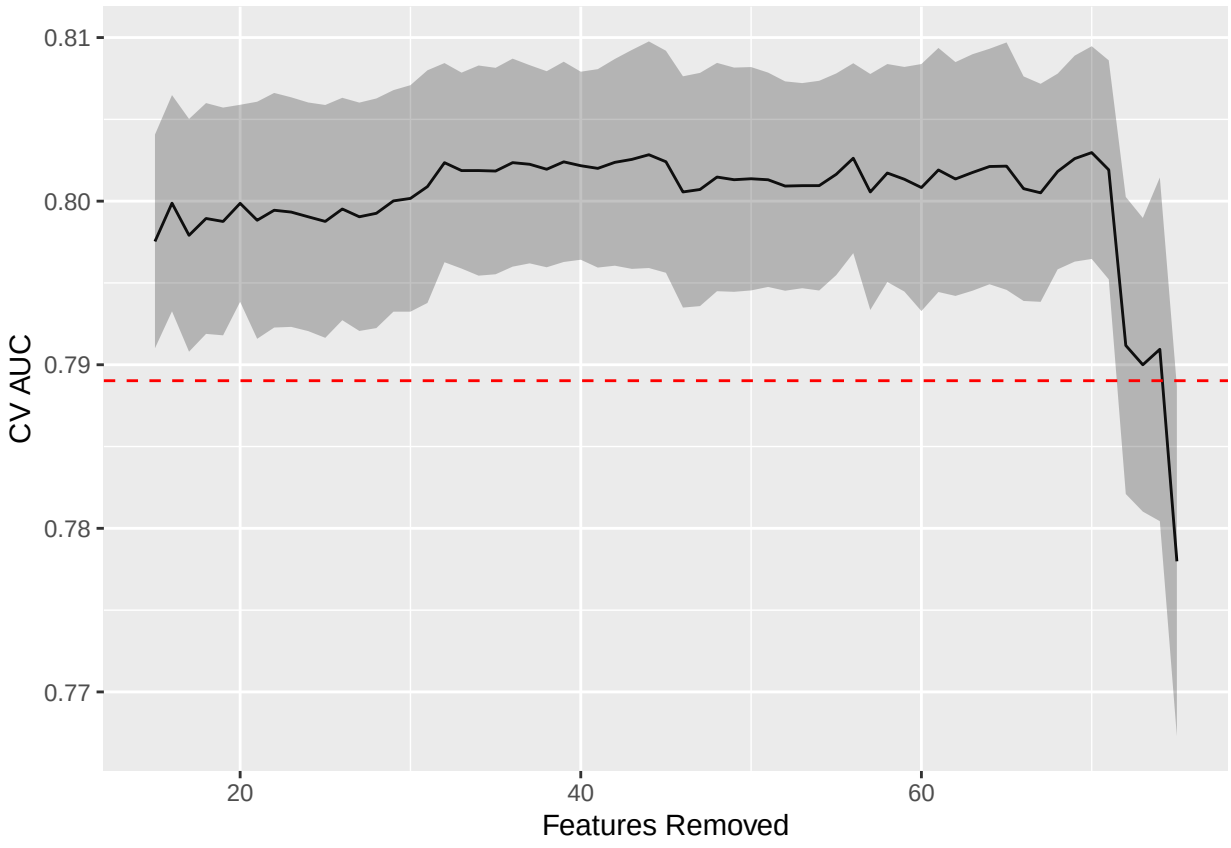
```
write_yaml(selected_features, con)
close(con)

feature_selected_model <- model_fit_wf(df_train, selected_features,
                                       outcome_column, hyperparameters)

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

```
## [1] "Selected Model CV Train AUC: 0.778"
```

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

```
## [1] "Selected Model Test AUC: 0.747"
```

```
 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:

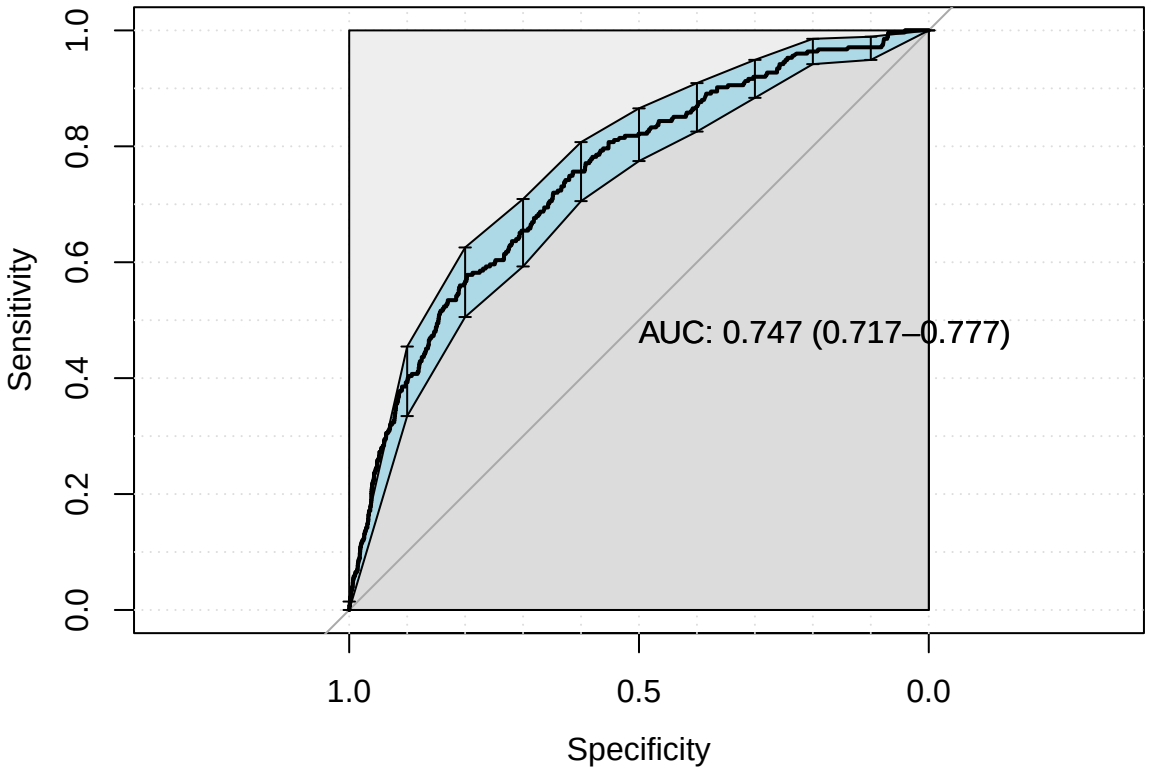
```
gluedown::md_order(selected_features, seq = TRUE, pad = TRUE)
```

1. hospital_stay
2. year_adm_t0
3. admission_pre_t0_count
4. age
5. espironolactona
6. comorbidities_count
7. stroke

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)
```



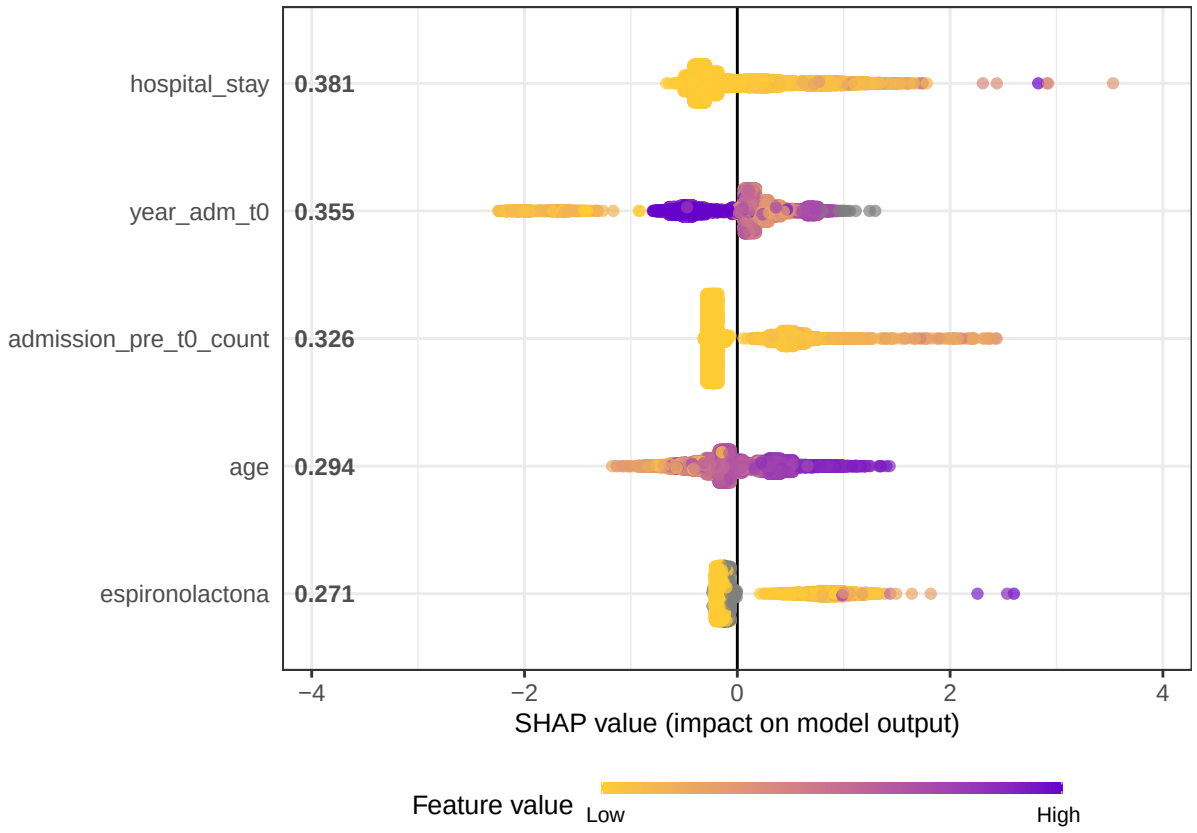

```
## [1] "Optimal Threshold: 0.08"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3550  116
##    1  905  159
##
##                Accuracy : 0.7841
##                  95% CI : (0.7721, 0.7958)
##     No Information Rate : 0.9419
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1599
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7969
##             Specificity : 0.5782
##          Pos Pred Value : 0.9684
##          Neg Pred Value : 0.1494
##              Prevalence : 0.9419
##          Detection Rate : 0.7505
##    Detection Prevalence : 0.7751
##       Balanced Accuracy : 0.6875
##
##        '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

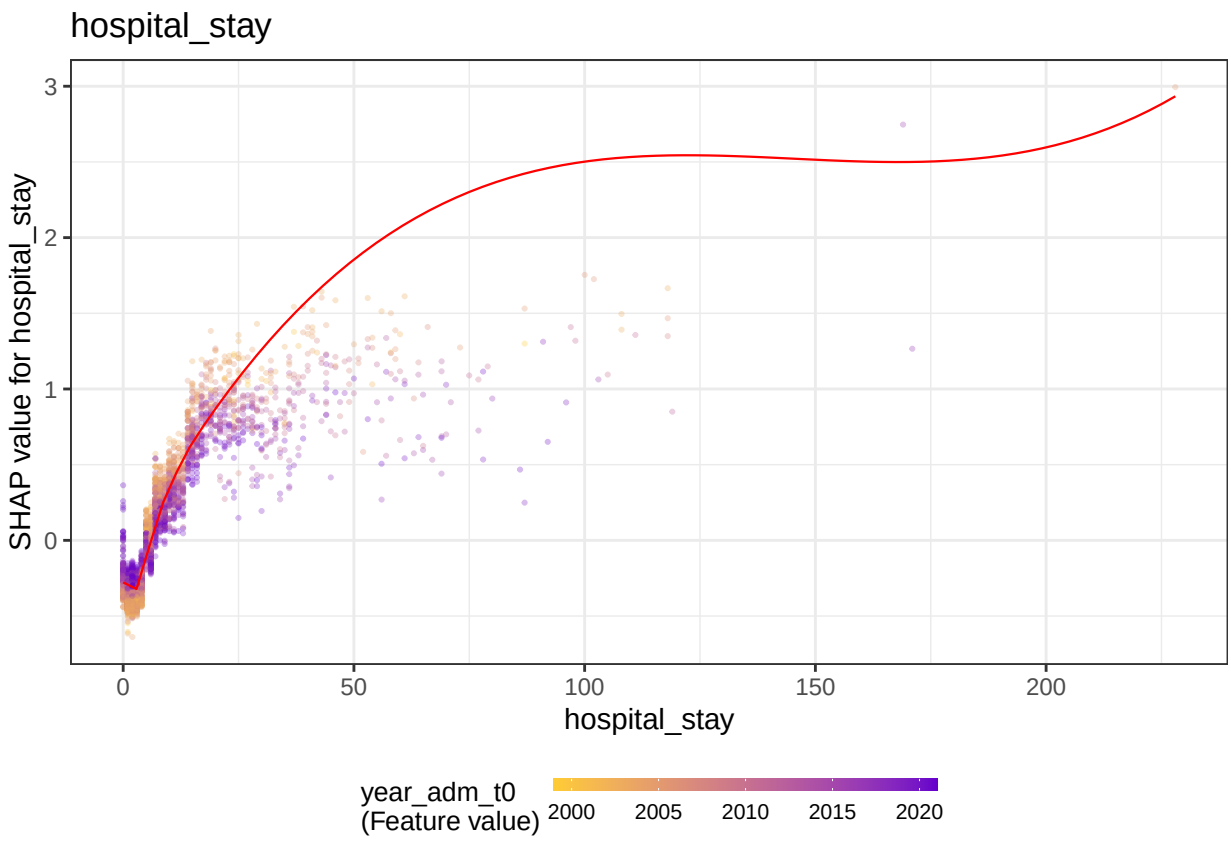
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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)
```



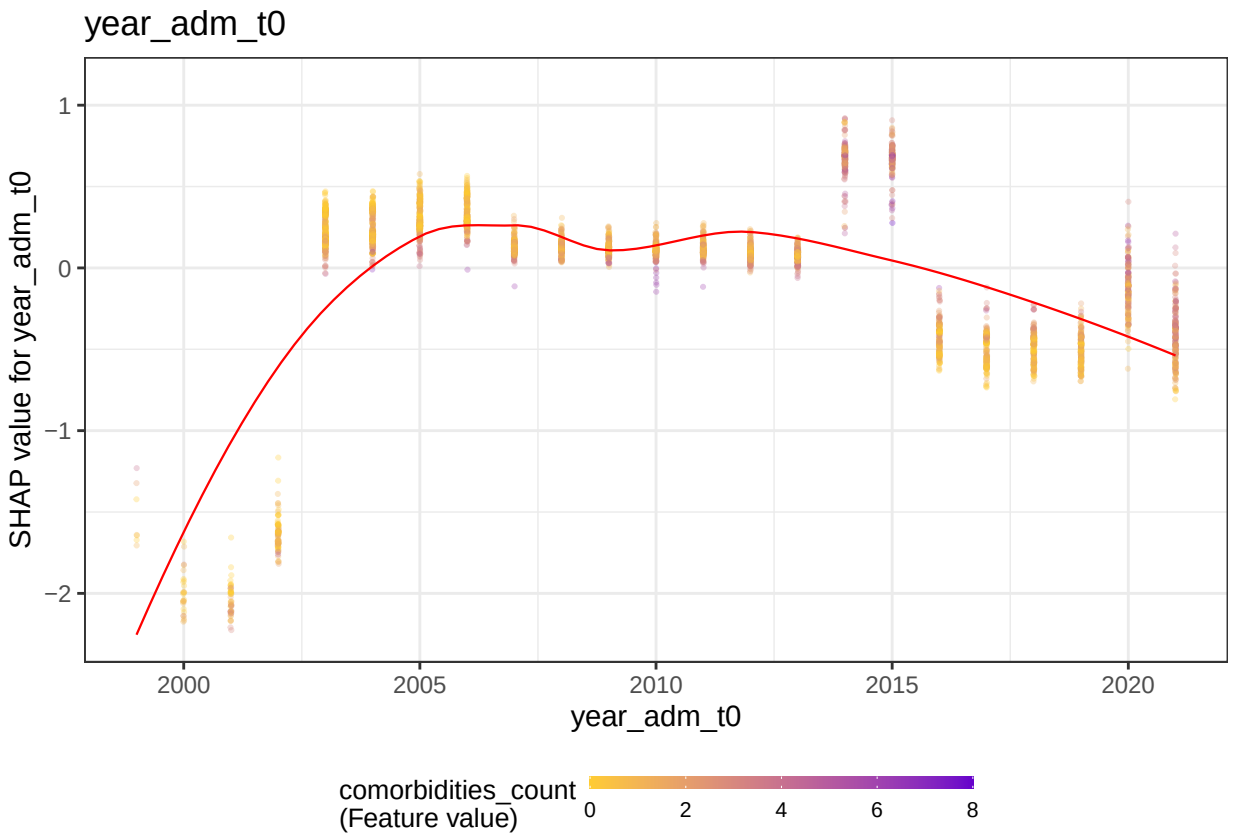

```
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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)
}
```

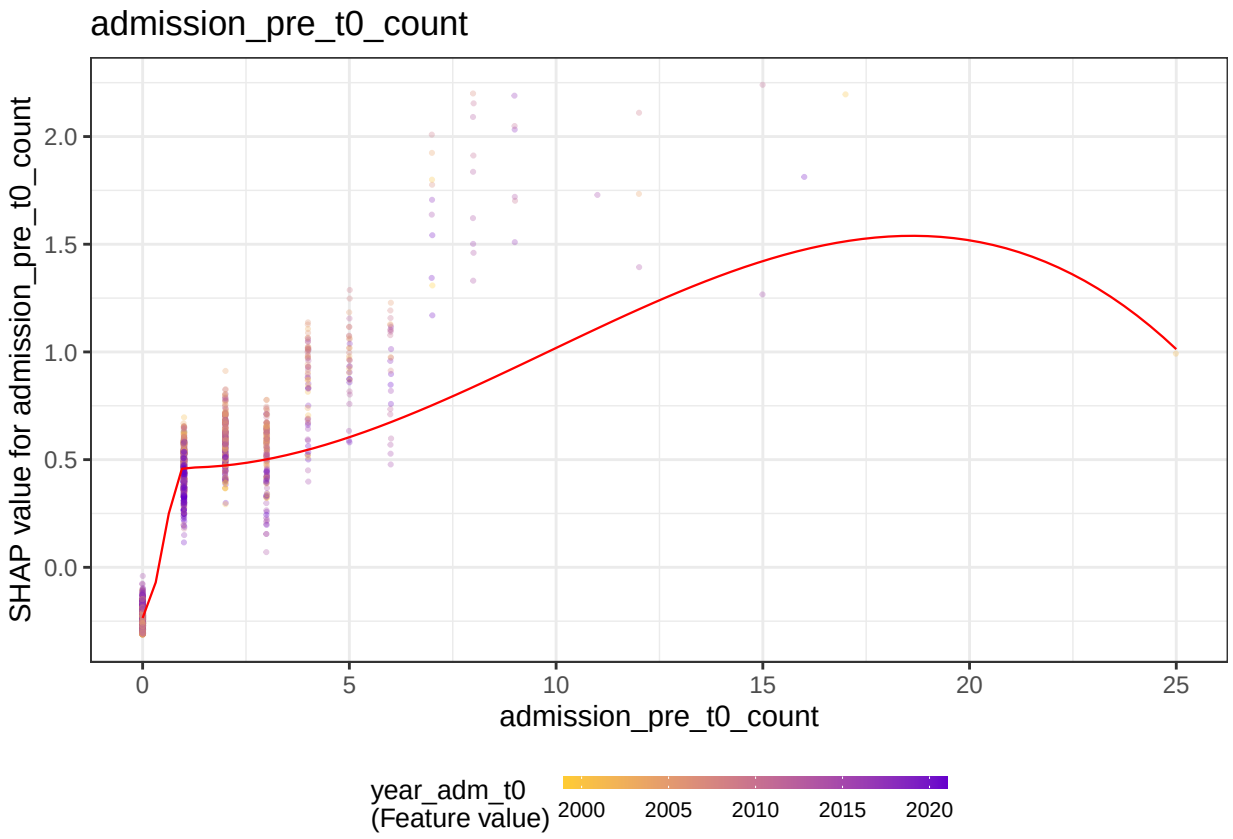
```
## `geom_smooth()` using formula 'y ~ x'
```



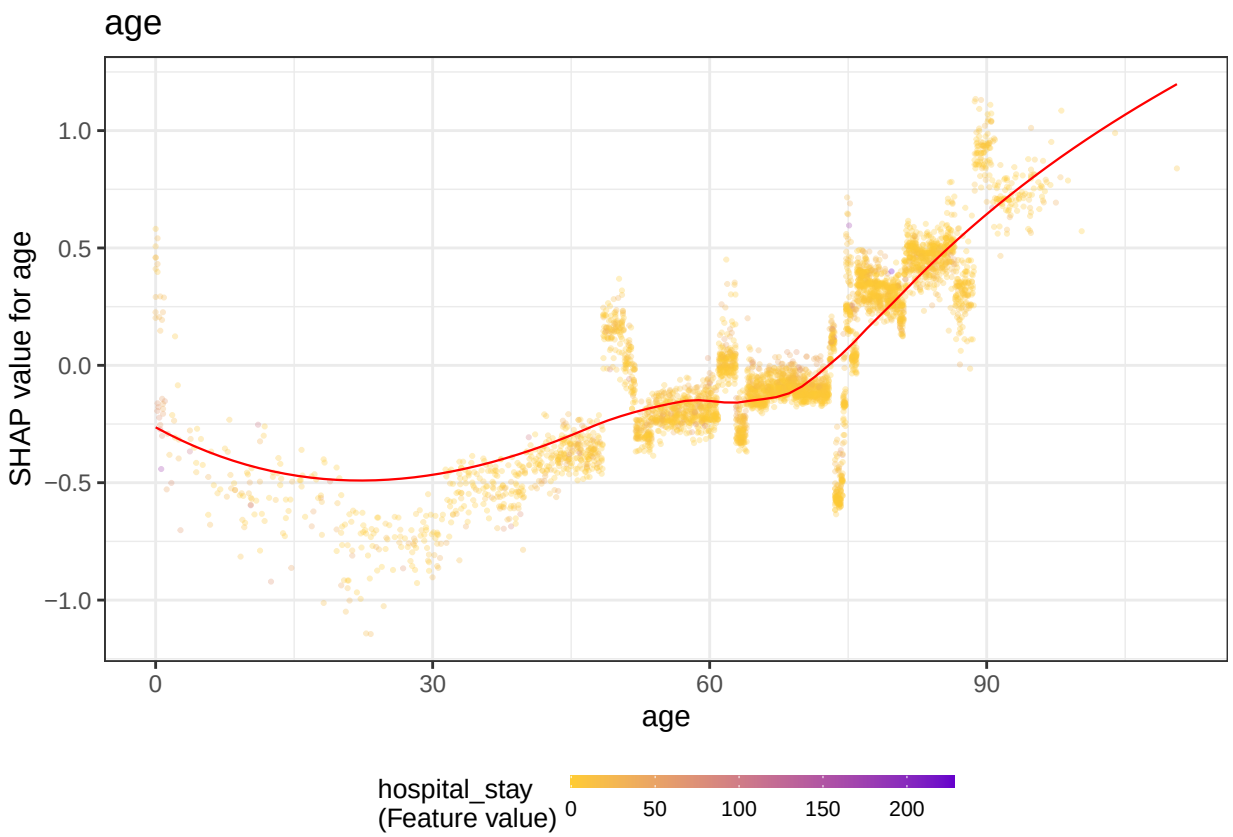

```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 6 rows containing non-finite values (stat_smooth).
## Warning: Removed 6 rows containing missing values (geom_point).
```




```
## `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
```

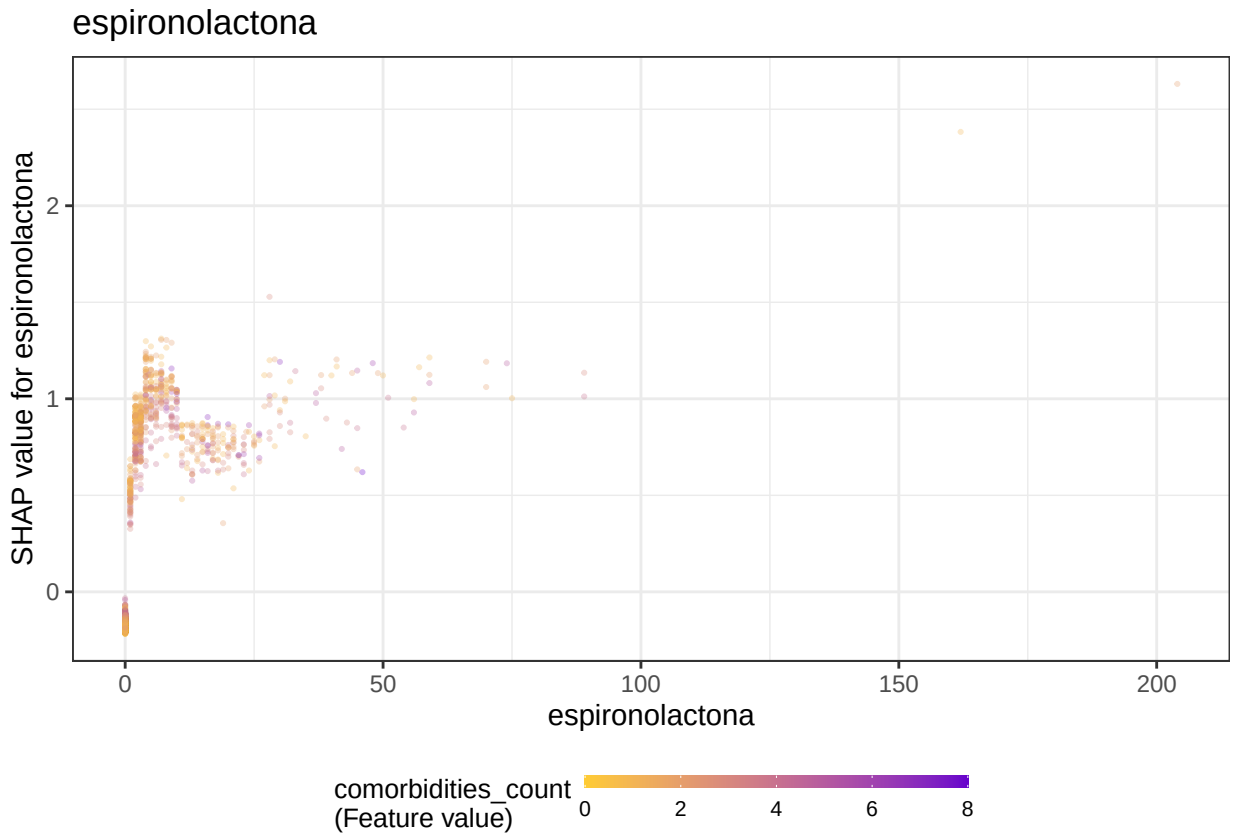



```
## `geom_smooth()` using formula 'y ~ x'
```




```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 1070 rows containing non-finite values (stat_smooth).
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : at -1.02
```

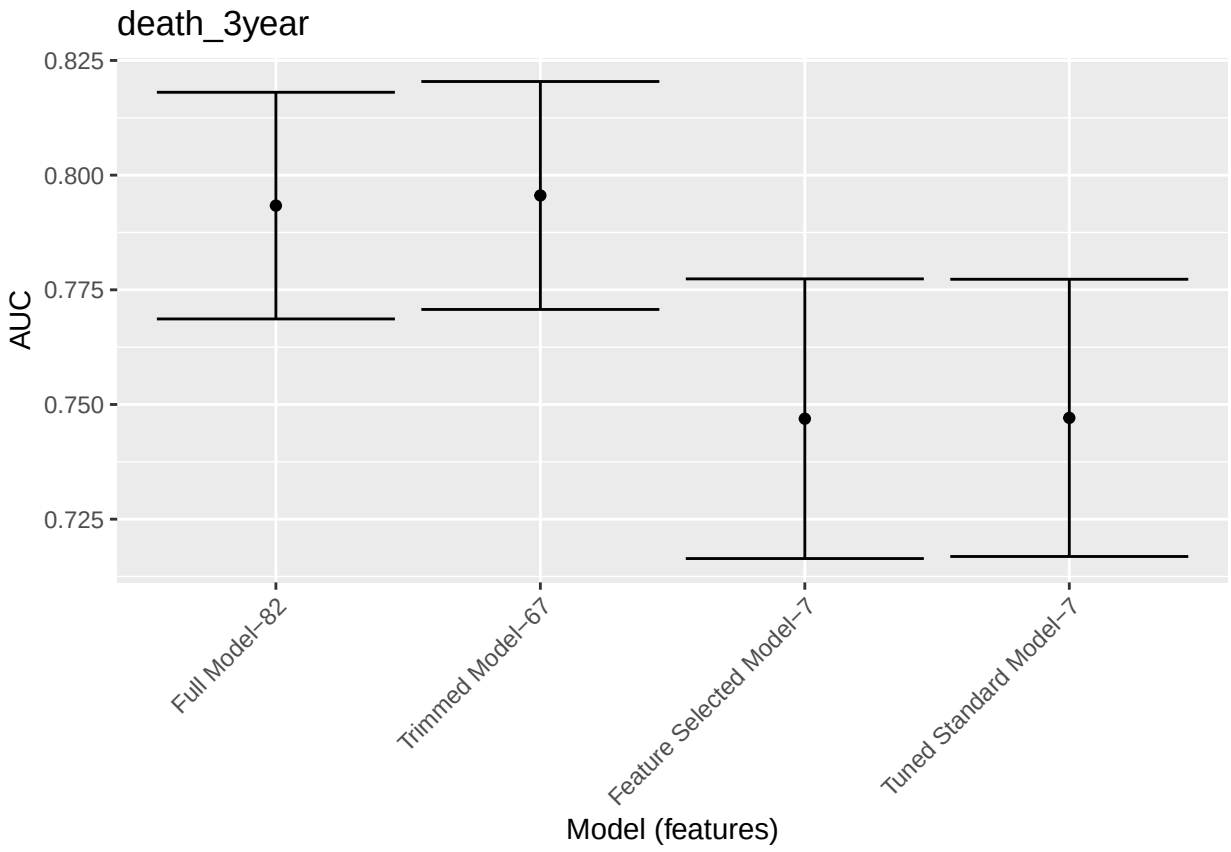
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : radius 1.0404

## 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 1070 rows containing missing values (geom_point).
```



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
  'Tuned Standard Model', standard_results$auc, standard_results$auc_lower, standard_results$auc_upper, length(s
  # 'Tuned Smote Model', smote_results$auc, smote_results$auc_lower, smote_results$auc_upper, length(selected_fe
  # 'Tuned Upsample Model', upsample_results$auc, upsample_results$auc_lower, upsample_results$auc_upper, length
) %>%
  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))
```