

# Final Model - death\_1year

Eduardo Yuki Yada

## Global parameters

```
k <- params$k # Number of folds for cross validation
grid_size <- params$grid_size # Number of parameter combination to tune on each model
max_auc_loss <- params$max_auc_loss # Max accepted loss of AUC for reducing num of features
repeats <- params$repeats
Hmisc::list.tree(params)

##  params = list 5 (952 bytes)
## . max_auc_loss = double 1= 0.01
## . outcome_column = character 1= death_1year
## . k = double 1= 10
## . grid_size = double 1= 50
## . repeats = double 1= 2
```

## 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
predict <- stats::predict
```

## 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)

dir.create(file.path("./auxiliar/final_model/auroc_plots/"),
           showWarnings = FALSE,
           recursive = TRUE)

dir.create(file.path("./auxiliar/final_model/shap_plots/"),
           showWarnings = FALSE,
           recursive = TRUE)

```

## Eligible features

```

cat_features_list = read_yaml(sprintf(
  "./auxiliar/significant_columns/categorical_%s.yaml",
  outcome_column
))

num_features_list = read_yaml(sprintf(
  "./auxiliar/significant_columns/numerical_%s.yaml",
  outcome_column
))

features_list = c(cat_features_list, num_features_list)

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))

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. antiarritmico
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),
                      repeats = repeats)

```

## Feature Selection

```

custom_dummy_names <- function(var, lvl, ordinal = FALSE) {
  dummy_names(var, lvl, ordinal = FALSE, sep = "___")
}

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_dummy(all_nominal_predictors(), naming = custom_dummy_names)

  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) %>%
separate(Feature, c("Feature", "value"), ___, fill = 'right') %>%
group_by(Feature) %>%
summarise(Gain = sum(Gain),
Cover = sum(Cover),
Frequency = sum(Frequency)) %>%
ungroup() %>%
arrange(desc(Gain))

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 <- read_yaml(
  sprintf(
    "./auxiliar/model_selection/hyperparameters/%s.yaml",
    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: 0.811"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

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

```

Features with zero importance on the initial model:

```

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

unimportant_features %>%
  gluedown::md_order()

1. dialysis_t0
2. vni
3. transplante_cardiaco
4. teste_esforco
5. tilt_teste
6. histopatologia_qtde

trimmed_features <- full_model$importance$Feature
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: 0.811"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.809"

selection_results <- tibble::tribble(
  ~`Tested Feature`, ~`Dropped`, ~`Number of Features`, ~`CV AUC`, ~`CV AUC Std Error`, ~`Total AUC Loss`, ~`Instant AUC Loss`,
  'None', TRUE, length(features), full_model$cv_auc, full_model$cv_auc_std_err, 0, 0
)

whitelist <- c()

if (full_model$cv_auc - trimmed_model$cv_auc < max_auc_loss) {
  current_features <- trimmed_features
  current_model <- trimmed_model
  current_auc_loss <- full_model$cv_auc - current_model$cv_auc
  instant_auc_loss <- full_model$cv_auc - current_model$cv_auc

  selection_results <- selection_results %>%
    add_row(`Tested Feature` = 'All unimportant',
            `Dropped` = TRUE,
            `Number of Features` = length(trimmed_features),
            `CV AUC` = current_model$cv_auc,
            `CV AUC Std Error` = current_model$cv_auc_std_err,
            `Total AUC Loss` = current_auc_loss,
            `Instant AUC Loss` = instant_auc_loss)
} else {
  current_features <- features
  current_model <- full_model
  current_auc_loss <- 0
}

while (current_auc_loss < max_auc_loss & mean(current_features %in% whitelist) < 1) {
  zero_importance_features <-
    setdiff(current_features, current_model$importance$Feature) %>%
    setdiff(whitelist)
  if (length(zero_importance_features) > 0) {
    current_least_important <- zero_importance_features[1]
  } else {
    current_least_important <-
      tail(setdiff(current_model$importance$Feature, whitelist), 1)
  }
  test_features <-
    setdiff(current_features, current_least_important)
}

```

```

current_model <-
  model_fit_wf(df_train, test_features, outcome_column, hyperparameters)
instant_auc_loss <-
  tail(selection_results %>% filter(Dropped) %>% .$`CV AUC`, n = 1) - current_model$cv_auc

if (instant_auc_loss < max_auc_loss / 5 &
    current_auc_loss < max_auc_loss) {
  dropped <- TRUE
  current_features <- test_features
  current_auc_loss <- full_model$cv_auc - current_model$cv_auc
} else {
  dropped <- FALSE
  whitelist <- c(whitelist, current_least_important)
}

selection_results <- selection_results %>%
  add_row(
    `Tested Feature` = current_least_important,
    `Dropped` = dropped,
    `Number of Features` = length(test_features),
    `CV AUC` = current_model$cv_auc,
    `CV AUC Std Error` = current_model$cv_auc_std_err,
    `Total AUC Loss` = current_auc_loss,
    `Instant AUC Loss` = instant_auc_loss
  )

print(c(
  length(current_features),
  round(current_auc_loss, 4),
  round(instant_auc_loss, 4),
  current_least_important
))
}

## [1] "76"      "-1e-04"   "1e-04"     "endoscopia"
## [1] "75"      "4e-04"    "5e-04"     "transfusao"
## [1] "74"      "6e-04"    "2e-04"     "aortografia"
## [1] "73"      "0.0017"   "0.0012"   "holter"
## [1] "72"      "0.001"    "-7e-04"   "copd"
## [1] "71"      "-7e-04"   "-0.0018"   "pet_ct"
## [1] "70"      "0"        "8e-04"    "cir_toracica"
## [1] "69"      "0.0011"   "0.001"    "insulina"
## [1] "68"      "0"        "-0.0011"   "antiviral"
## [1] "67"      "-1e-04"   "0"        "cultura"
## [1] "66"      "-5e-04"   "-4e-04"   "aco"
## [1] "65"      "-2e-04"   "3e-04"    "heart_failure"
## [1] "64"      "-1e-04"   "1e-04"    "ressonancia"
## [1] "63"      "0.0018"   "0.0019"   "sex"
## [1] "62"      "0.0015"   "-3e-04"   "cancer"
## [1] "61"      "0.0015"   "0"        "hemodialysis"
## [1] "60"      "-0.0012"  "-0.0027"
## [4] "procedure_type_new"
## [1] "59"      "-0.0014"  "-2e-04"   "stroke"
## [1] "58"      "-0.0014"  "0"        "procedure_type_1"
## [1] "57"      "-0.0013"  "1e-04"    "bic"
## [1] "56"      "-9e-04"   "4e-04"    "cardiac_arrest"
## [1] "55"      "-0.0016"  "-7e-04"   "angio_tc"
## [1] "54"      "-8e-04"   "8e-04"    "icp"
## [1] "53"      "-0.0025"  "-0.0017"   "ecocardiograma"
## [1] "53"      "-0.0025"  "0.0028"   "valvopathy"
## [1] "53"      "-0.0025"  "0.0031"

```

```

## [4] "antiplaquetario_ev"
## [1] "52"      "-0.0012"   "0.0013"    "cateterismo"
## [1] "51"      "-7e-04"    "5e-04"     "antifungico"
## [1] "50"      "-0.0024"   "-0.0017"   "angiografia"
## [1] "49"      "-0.0037"   "0.0013"
## [4] "ventilacao_mecanica"
## [1] "48"      "-0.0031"   "5e-04"     "heart_disease"
## [1] "48"      "-0.0031"   "0.0021"
## [4] "admission_pre_t0_180d"
## [1] "47"      "-0.0025"   "6e-04"     "interconsulta"
## [1] "46"      "-0.0013"   "0.0013"    "tomografia"
## [1] "45"      "-0.0043"   "-0.0031"
## [4] "cateter Venoso_Central"
## [1] "44"      "-0.003"    "0.0013"    "prior_mi"
## [1] "43"      "-0.0051"   "-0.0022"   "diabetes"
## [1] "42"      "-0.0054"   "-3e-04"    "flebografia"
## [1] "41"      "-0.0048"   "6e-04"     "renal_failure"
## [1] "40"      "-0.0063"
## [3] "-0.0015"  "outros_proced_cirurgicos"
## [1] "39"      "-0.0073"   "-0.001"    "cintilografia"
## [1] "38"      "-0.0054"   "0.0019"    "digoxina"
## [1] "38"      "-0.0054"   "0.0038"    "hypertension"
## [1] "38"      "-0.0054"   "0.003"     "ultrassom"
## [1] "37"      "-0.0036"   "0.0018"    "reop_type_1"
## [1] "36"      "-0.0055"   "-0.0019"   "cied_final_1"
## [1] "35"      "-0.0037"   "0.0018"    "af"
## [1] "34"      "-0.0042"   "0.001"     "-5e-04"
## [4] "admission_t0_emergency"
## [1] "33"      "-0.0035"   "7e-04"     "citologia"
## [1] "32"      "-0.0041"   "-6e-04"    "estatina"
## [1] "31"      "-0.0047"   "-5e-04"    "dva"
## [1] "30"      "-0.0037"   "0.001"     "equipe_multiprof"
## [1] "30"      "-0.0037"   "0.0024"
## [4] "cied_final_group_1"
## [1] "29"      "-0.0026"   "0.0011"
## [4] "analises_clinicas_qtde"
## [1] "28"      "-0.0037"   "-0.0012"
## [4] "exames_imagem_qtde"
## [1] "27"      "-0.0047"
## [3] "-9e-04"   "underlying_heart_disease"
## [1] "26"      "-0.0053"   "-6e-04"    "icu_t0"
## [1] "26"      "-0.0053"   "0.0022"
## [4] "proced_invasivos_qtde"
## [1] "26"      "-0.0053"   "0.0046"    "antiarritmico"
## [1] "25"      "-0.0056"   "-3e-04"    "insuf_cardiaca"
## [1] "24"      "-0.0043"   "0.0014"
## [4] "metodos_graficos_qtde"
## [1] "23"      "-0.0043"   "0"         "diuretico"
## [1] "22"      "-0.005"    "-7e-04"    "psicofarmacos"
## [1] "21"      "-0.0056"   "-6e-04"    "nyha_basal"
## [1] "20"      "-0.0058"   "-2e-04"
## [4] "meds_antimicrobianos"
## [1] "20"      "-0.0058"   "0.0032"    "ieca_bra"
## [1] "19"      "-0.0051"   "7e-04"     "classe_meds_qtde"
## [1] "18"      "-0.0033"   "0.0018"    "vasodilatador"
## [1] "17"      "-0.0067"   "-0.0034"
## [4] "meds_cardiovasc_qtde"
## [1] "17"      "-0.0067"   "0.0161"    "education_level"
## [1] "17"      "-0.0067"   "0.0087"
## [4] "admission_pre_t0_count"
## [1] "17"      "-0.0067"   "0.0075"    "espironolactona"

```

```

## [1] "17"           "-0.0067"          "0.0056"
## [4] "comorbidities_count"
## [1] "16"           "-0.0069"         "-2e-04"        "laboratorio"
## [1] "16"           "-0.0069"         "0.025"        "year_adm_t0"
## [1] "16"           "-0.0069"         "0.0114"       "hospital_stay"
## [1] "16"           "-0.0069"         "0.013"        "age"

selection_results %>%
  rename(Features = `Number of Features` ) %>%
  niceFormatting(digits = 4, label = 1)

```

Table 1:

Tested Feature	Dropped	Features	CV AUC	CV AUC Std Error	Total AUC Loss	Instant AUC Loss
None	TRUE	83	0.8109	0.0067	0.0000	0.0000
All unimportant	TRUE	77	0.8111	0.0072	-0.0002	-0.0002
endoscopia	TRUE	76	0.8110	0.0069	-0.0001	0.0001
transfusao	TRUE	75	0.8105	0.0068	0.0004	0.0005
aortografia	TRUE	74	0.8103	0.0070	0.0006	0.0002
holter	TRUE	73	0.8092	0.0071	0.0017	0.0012
copd	TRUE	72	0.8099	0.0072	0.0010	-0.0007
pet_ct	TRUE	71	0.8116	0.0070	-0.0007	-0.0018
cir_toracica	TRUE	70	0.8108	0.0066	0.0000	0.0008
insulina	TRUE	69	0.8098	0.0070	0.0011	0.0010
antiviral	TRUE	68	0.8109	0.0069	0.0000	-0.0011
cultura	TRUE	67	0.8110	0.0069	-0.0001	0.0000
aco	TRUE	66	0.8113	0.0067	-0.0005	-0.0004
heart_failure	TRUE	65	0.8111	0.0072	-0.0002	0.0003
ressonancia	TRUE	64	0.8110	0.0069	-0.0001	0.0001
sex	TRUE	63	0.8090	0.0072	0.0018	0.0019
cancer	TRUE	62	0.8094	0.0070	0.0015	-0.0003
hemodialysis	TRUE	61	0.8094	0.0069	0.0015	0.0000
procedure_type_new	TRUE	60	0.8121	0.0072	-0.0012	-0.0027
stroke	TRUE	59	0.8123	0.0073	-0.0014	-0.0002
procedure_type_1	TRUE	58	0.8123	0.0073	-0.0014	0.0000
bic	TRUE	57	0.8122	0.0070	-0.0013	0.0001
cardiac_arrest	TRUE	56	0.8118	0.0076	-0.0009	0.0004
angio_tc	TRUE	55	0.8125	0.0074	-0.0016	-0.0007
icp	TRUE	54	0.8117	0.0072	-0.0008	0.0008
ecocardiograma	TRUE	53	0.8134	0.0075	-0.0025	-0.0017
valvopathy	FALSE	52	0.8107	0.0073	-0.0025	0.0028
antiplaquetario_ev	FALSE	52	0.8103	0.0072	-0.0025	0.0031
cateterismo	TRUE	52	0.8121	0.0070	-0.0012	0.0013
antifungico	TRUE	51	0.8116	0.0073	-0.0007	0.0005
angiografia	TRUE	50	0.8132	0.0071	-0.0024	-0.0017
ventilacao_mecanica	TRUE	49	0.8146	0.0070	-0.0037	-0.0013
heart_disease	TRUE	48	0.8140	0.0068	-0.0031	0.0005
admission_pre_t0_180d	FALSE	47	0.8120	0.0072	-0.0031	0.0021
interconsulta	TRUE	47	0.8134	0.0071	-0.0025	0.0006
tomografia	TRUE	46	0.8121	0.0070	-0.0013	0.0013
cateter Venoso_Central	TRUE	45	0.8152	0.0071	-0.0043	-0.0031
prior_mi	TRUE	44	0.8139	0.0073	-0.0030	0.0013
diabetes	TRUE	43	0.8160	0.0071	-0.0051	-0.0022
flebografia	TRUE	42	0.8163	0.0072	-0.0054	-0.0003
renal_failure	TRUE	41	0.8157	0.0066	-0.0048	0.0006
outros_proced_cirurgicos	TRUE	40	0.8172	0.0067	-0.0063	-0.0015
cintilografia	TRUE	39	0.8182	0.0070	-0.0073	-0.0010
digoxina	TRUE	38	0.8163	0.0071	-0.0054	0.0019

Table 1: (continued)

Tested Feature	Dropped	Features	CV AUC	CV AUC Std Error	Total AUC Loss	Instant AUC Loss
hypertension	FALSE	37	0.8125	0.0070	-0.0054	0.0038
ultrassom	FALSE	37	0.8132	0.0071	-0.0054	0.0030
reop_type_1	TRUE	37	0.8145	0.0069	-0.0036	0.0018
cied_final_1	TRUE	36	0.8163	0.0071	-0.0055	-0.0019
af	TRUE	35	0.8146	0.0068	-0.0037	0.0018
admission_t0_emergency	TRUE	34	0.8151	0.0067	-0.0042	-0.0005
citologia	TRUE	33	0.8144	0.0065	-0.0035	0.0007
estatina	TRUE	32	0.8150	0.0067	-0.0041	-0.0006
dva	TRUE	31	0.8155	0.0067	-0.0047	-0.0005
equipe_multiprof	TRUE	30	0.8145	0.0068	-0.0037	0.0010
cied_final_group_1	FALSE	29	0.8122	0.0077	-0.0037	0.0024
analises_clinicas_qtde	TRUE	29	0.8134	0.0071	-0.0026	0.0011
exames_imagem_qtde	TRUE	28	0.8146	0.0072	-0.0037	-0.0012
underlying_heart_disease	TRUE	27	0.8155	0.0074	-0.0047	-0.0009
icu_t0	TRUE	26	0.8162	0.0071	-0.0053	-0.0006
proced_invasivos_qtde	FALSE	25	0.8140	0.0074	-0.0053	0.0022
antiarritmico	FALSE	25	0.8116	0.0072	-0.0053	0.0046
insuf_cardiaca	TRUE	25	0.8165	0.0068	-0.0056	-0.0003
metodos_graficos_qtde	TRUE	24	0.8152	0.0072	-0.0043	0.0014
diuretico	TRUE	23	0.8151	0.0077	-0.0043	0.0000
psicofarmacos	TRUE	22	0.8159	0.0074	-0.0050	-0.0007
nyha_basal	TRUE	21	0.8165	0.0079	-0.0056	-0.0006
meds_antimicrobianos	TRUE	20	0.8167	0.0080	-0.0058	-0.0002
ieca_bra	FALSE	19	0.8135	0.0081	-0.0058	0.0032
classe_meds_qtde	TRUE	19	0.8160	0.0072	-0.0051	0.0007
vasodilatador	TRUE	18	0.8142	0.0069	-0.0033	0.0018
meds_cardiovasc_qtde	TRUE	17	0.8175	0.0069	-0.0067	-0.0034
education_level	FALSE	16	0.8014	0.0070	-0.0067	0.0161
admission_pre_t0_count	FALSE	16	0.8089	0.0076	-0.0067	0.0087
espironolactona	FALSE	16	0.8101	0.0071	-0.0067	0.0075
comorbidities_count	FALSE	16	0.8119	0.0075	-0.0067	0.0056
laboratorio	TRUE	16	0.8178	0.0082	-0.0069	-0.0002
year_adm_t0	FALSE	15	0.7927	0.0083	-0.0069	0.0250
hospital_stay	FALSE	15	0.8064	0.0085	-0.0069	0.0114
age	FALSE	15	0.8048	0.0076	-0.0069	0.0130

```

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: 0.818"
sprintf('Selected Model Test AUC: %.3f', feature_selected_model$auc)

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

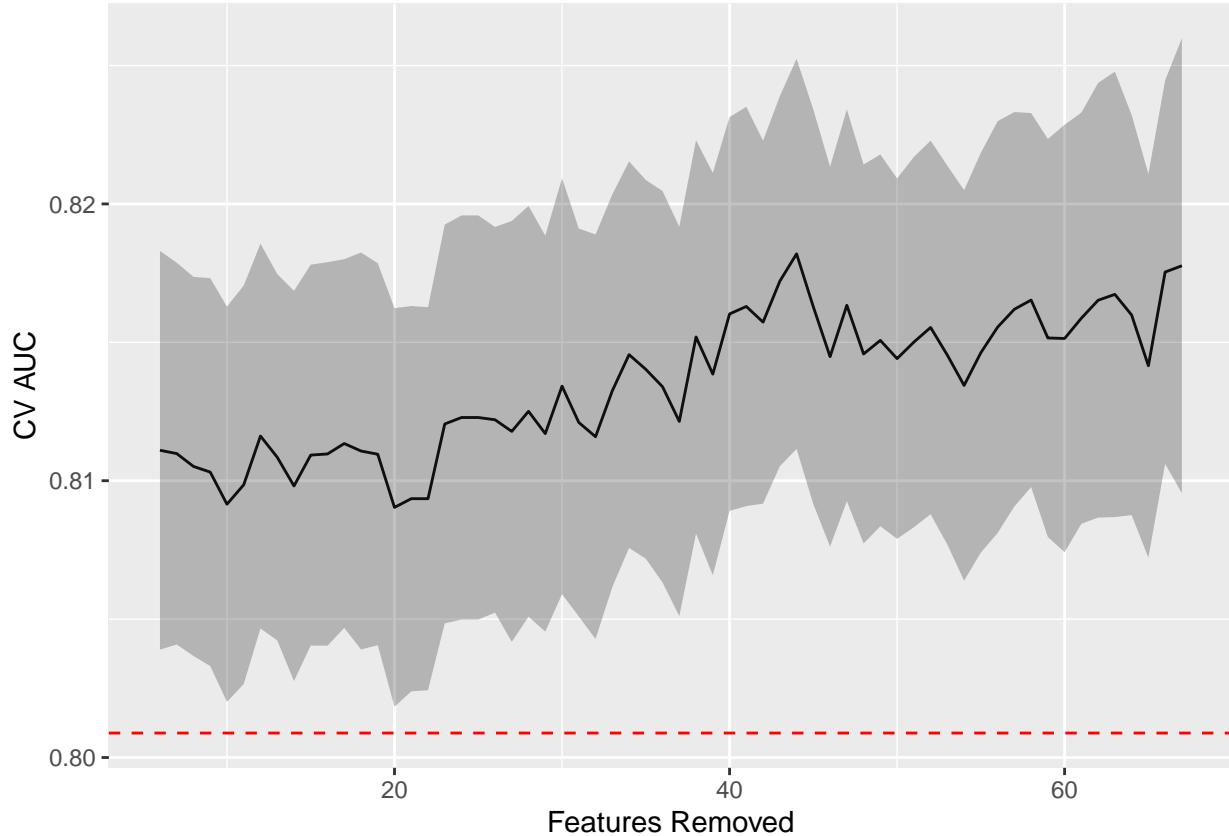
selection_results <- selection_results %>%
  filter(`Number of Features` < length(features)) %>%
  mutate(`Features Removed` = length(features) - `Number of Features`,
```
```

```

`~CV AUC Low` = `~CV AUC` - `~CV AUC Std Error`,
`~CV AUC High` = `~CV AUC` + `~CV AUC Std Error`)

selection_results %>%
  filter(Dropped) %>%
  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")

```



## Hyperparameter tuning

Selected features:

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

1. hospital\_stay
2. age
3. year\_adm\_t0
4. comorbidities\_count
5. admission\_pre\_t0\_count
6. espironolactona
7. education\_level
8. antiaritmico
9. ieca\_bra
10. cied\_final\_group\_1
11. proced\_invasivos\_qtde
12. hypertension
13. ultrassom
14. admission\_pre\_t0\_180d
15. antiplaquetario\_ev

## 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_tuning <- function(recipe) {

  lightgbm_spec <- boost_tree(
    trees = tune(),
    min_n = tune(),
    tree_depth = tune(),
    learn_rate = tune(),
    sample_size = 1.0
  ) %>%
    set_engine("lightgbm",
               nthread = 8) %>%
    set_mode("classification")

  lightgbm_grid <- grid_latin_hypercube(
    trees(range = c(25L, 150L)),
    min_n(range = c(2L, 100L)),
    tree_depth(range = c(2L, 15L)),
    learn_rate(range = c(-3, -1), trans = log10_trans()),
    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")

  autoplot(lightgbm_tune, metric = "roc_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)

```

```

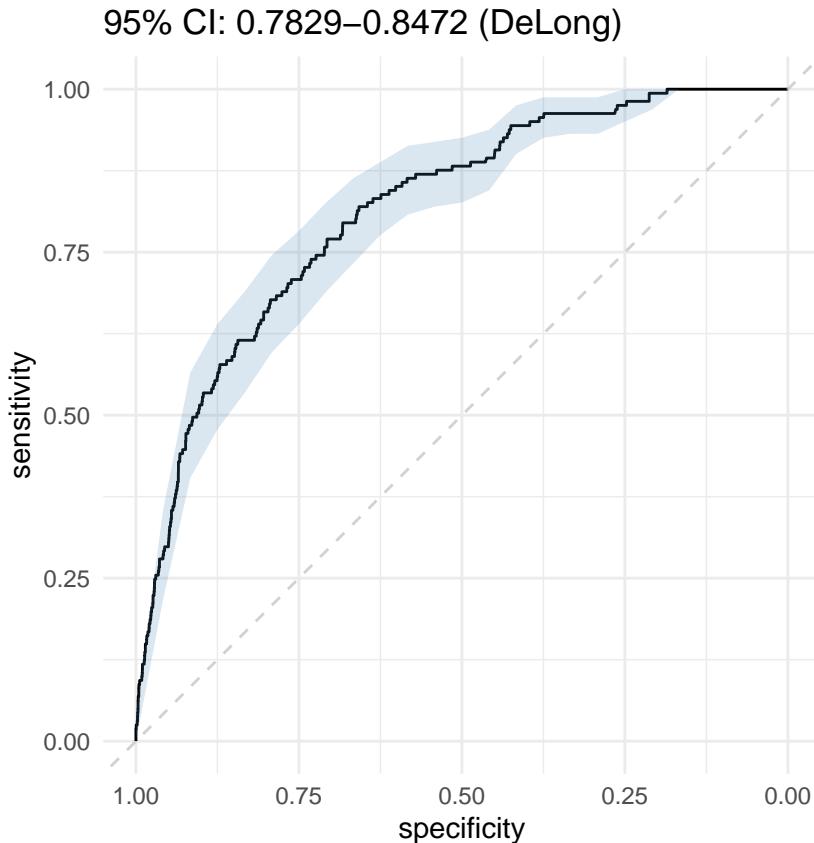
lightgbm_auc <- validation(final_lightgbm_fit, df_test)

lightgbm_parameters <- lightgbm_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(trees, min_n, tree_depth, learn_rate) %>%
  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: 0.02"
## Confusion Matrix and Statistics
##
##      reference
## data      0     1
##   0 3120    33
##   1 1449   128
##
##                  Accuracy : 0.6867
##                  95% CI : (0.6732, 0.6999)
##      No Information Rate : 0.966
##      P-Value [Acc > NIR] : 1
##
##                  Kappa : 0.0912
##
##  Mcnemar's Test P-Value : <2e-16
##

```

```

##           Sensitivity : 0.68286
##           Specificity : 0.79503
##           Pos Pred Value : 0.98953
##           Neg Pred Value : 0.08117
##           Prevalence : 0.96596
##           Detection Rate : 0.65962
##   Detection Prevalence : 0.66660
##           Balanced Accuracy : 0.73895
##
##           'Positive' Class : 0
##
final_lightgbm_fit <- standard_results$fit
lightgbm_parameters <- standard_results$parameters

con <- file(sprintf('./auxiliar/final_model/hyperparameters/%s.yaml',
                     outcome_column), "w")
write_yaml(lightgbm_parameters, con)
close(con)

# Save the final model. We need it for the calculator
lgb.save(
  parsnip::extract_fit_engine(final_lightgbm_fit),
  sprintf("./results/%s/final_model.txt", outcome_column)
)
saveRDS(final_lightgbm_fit,
        sprintf("./results/%s/final_model_wf.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(6, length(selected_features))
plotted <- 0

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)) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = FALSE,
    jitter_width = 0.01,
    alpha = 0.3
  ) +
    labs(title = x)

  if (plotted < n_plots) {

```

```

print(p)
plotted <- plotted + 1
}

ggsave(sprintf("./auxiliar/final_model/shap_plots/%s/%s.png",
  outcome_column, x),
  plot = p,
  dpi = 300)
}

## Saving 6.5 x 5 in image
## Saving 6.5 x 5 in image
## Saving 6.5 x 5 in image

## Warning: Removed 10 rows containing missing values ('geom_point()').

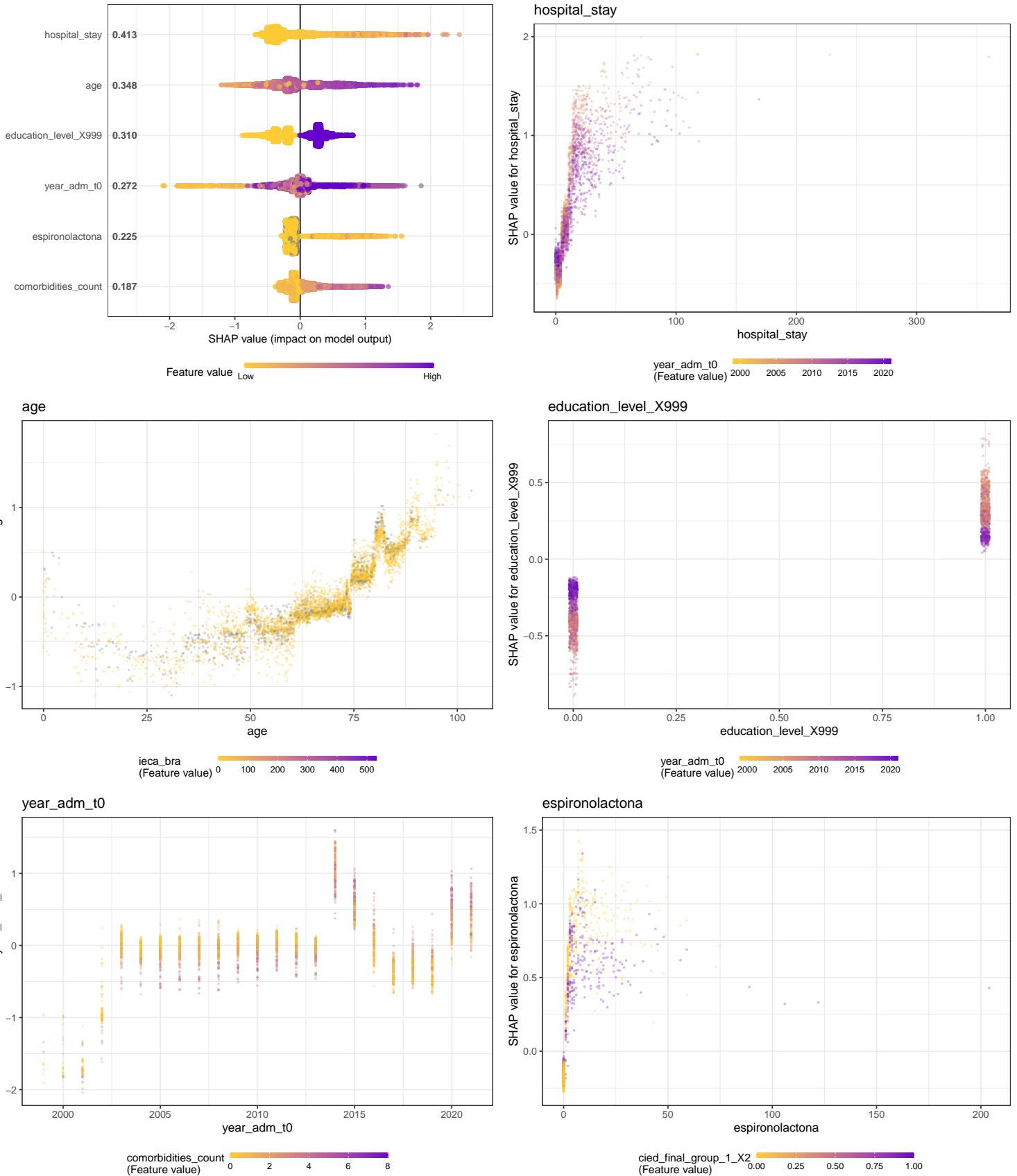
## Saving 6.5 x 5 in image
## Warning: Removed 10 rows containing missing values ('geom_point()').
## Warning: Removed 1044 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image
## Warning: Removed 1044 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image
## Saving 6.5 x 5 in image
## Saving 6.5 x 5 in image

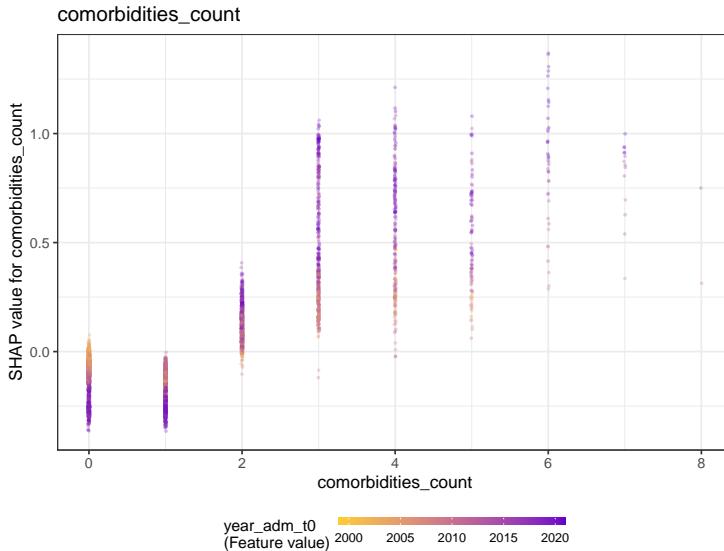
## Warning: Removed 1044 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image
## Saving 6.5 x 5 in image

## Warning: Removed 826 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image
## Warning: Removed 1044 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image
## Warning: Removed 826 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image

## Warning: Removed 1044 rows containing missing values ('geom_point()').
## Saving 6.5 x 5 in image

```





```

## $num_iterations
## [1] 78
##
## $learning_rate
## [1] 0.09345235
##
## $max_depth
## [1] 4
##
## $feature_fraction_bynode
## [1] 1
##
## $min_data_in_leaf
## [1] 39
##
## $min_gain_to_split
## [1] 0
##
## $bagging_fraction
## [1] 1
##
## $num_class
## [1] 1
##
## $objective
## [1] "binary"
##
## $num_threads
## $num_threads$num_threads
## [1] 0
##
## $nthread
## [1] 8
##
## $seed
## [1] 38688
##
## $deterministic
## [1] TRUE
##
## $verbose
## [1] -1

```

```

##  

## $metric  

## list()  

##  

## $interaction_constraints  

## list()  

##  

## $feature_pre_filter  

## [1] FALSE

```

## 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,  

  'Tuned Standard Model', standard_results$auc, standard_results$auc_lower, standard_results$auc_upper, length(standard_results$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`  

)) +  

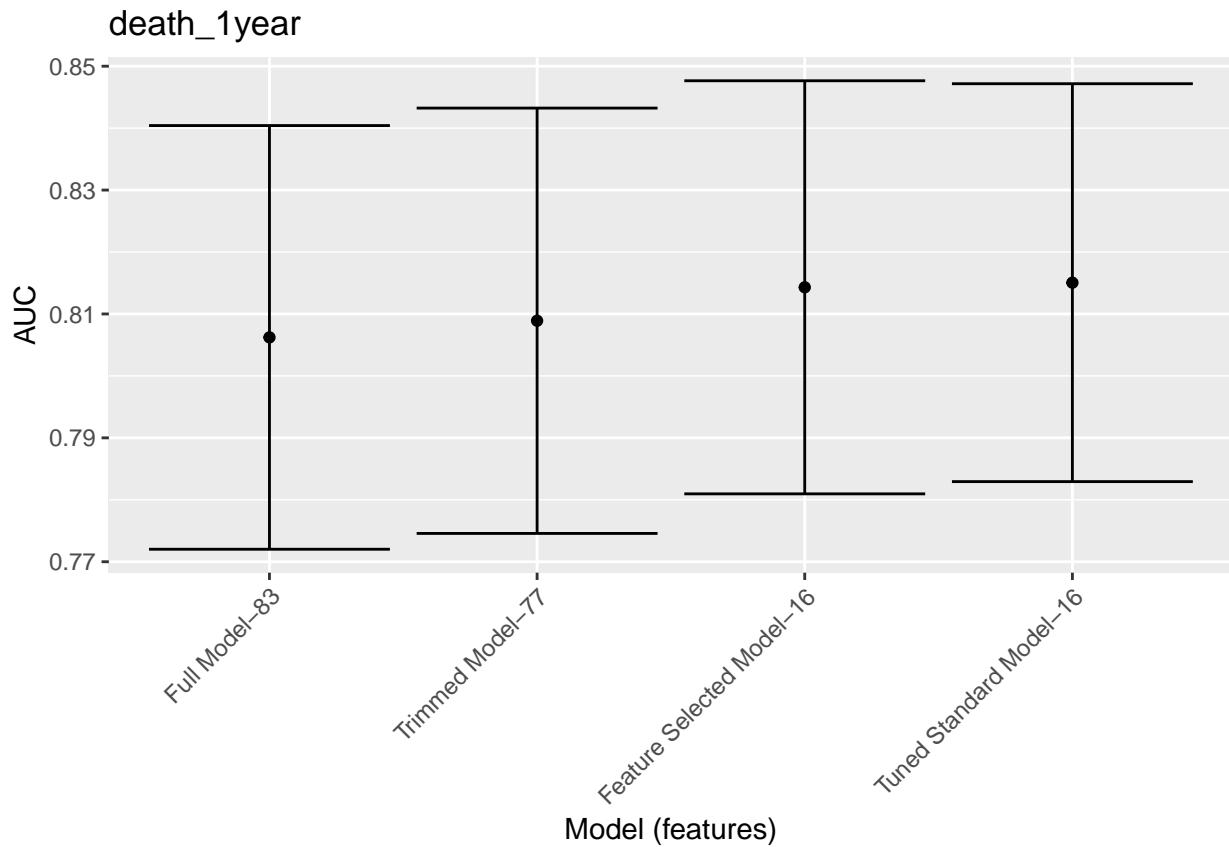
  geom_point() +  

  geom_errorbar() +  

  labs(title = outcome_column) +  

  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```



```
write_csv(df_auc, sprintf("./auxiliar/final_model/performance/%s.csv", outcome_column))
```