

Final Model - death_2year

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
```

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

```

cat_features_list = readRDS(sprintf(
  "./auxiliar/significant_columns/categorical_%s.rds",
  outcome_column
))

num_features_list = readRDS(sprintf(
  "./auxiliar/significant_columns/numerical_%s.rds",
  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))

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. comorbidities_count
19. procedure_type_1
20. reop_type_1
21. procedure_type_new
22. cied_final_1
23. cied_final_group_1
24. admission_pre_t0_count
25. admission_pre_t0_180d
26. year_adm_t0
27. icu_t0
28. dialysis_t0
29. admission_t0_emergency
30. aco
31. antiarritmico
32. ieca_bra
33. dva
34. digoxina
35. estatina
36. diuretico
37. vasodilatador
38. insuf_cardiaca
39. espironolactona
40. antiplaquetario_ev
41. insulina
42. psicofarmacos
43. antifungico
44. antiviral
45. classe_meds_qtde
46. meds_cardiovasc_qtde
47. meds_antimicrobianos
48. vni
49. ventilacao_mecanica
50. transplante_cardiaco
51. outros_proced_cirurgicos
52. icp
53. angioplastia
54. cateterismo
55. cateter_venoso_central
56. proced_invasivos_qtde
57. transfusao
58. interconsulta
59. equipe_multiprof
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. cintilografia
71. ecocardiograma
72. endoscopia

73. flebografia
 74. pet_ct
 75. ultrassom
 76. tomografia
 77. ressonancia
 78. exams_imagem_qtde
 79. bic
 80. 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))

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: 0.792"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.807"

# full_model$importance %>%
#   filter(str_detect(Feature, 'education'))
#
# full_model$importance %>%
#   filter(str_detect(Feature, 'education')) %>%
#   summarise(across(where(is.numeric), ~ sum(.x, na.rm = TRUE)))
#
# full_model$importance %>%
#   separate(Feature, c("Feature", "value"), "_") %>%
#   group_by(Feature) %>%
#   summarise(Gain = sum(Gain),
#             Cover = sum(Cover),
#             Frequency = sum(Frequency))

```

Features with zero importance on the initial model:

```

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

unimportant_features %>%
  gluedown::md_order()

```

```

1. sex
2. heart_failure
3. hemodialysis
4. reop_type_1
5. antiviral
6. vni
7. transplante_cardiaco
8. angioplastia
9. transfusao

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: 0.792"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.807"

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
current_auc_loss <- full_model$cv_auc - current_model$cv_auc

if (instant_auc_loss < max_auc_loss / 5 &
    current_auc_loss < max_auc_loss) {
  dropped <- TRUE
  current_features <- test_features
} 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] "71"           "0.0144"        "0.0143"        "year_adm_t0"

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	80	0.7925	0.0057	0.0000	0.0000
All unimportant	TRUE	71	0.7923	0.0063	0.0002	0.0002
year_adm_t0	FALSE	70	0.7781	0.0060	0.0144	0.0143

```

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

## [1] "Selected Model CV Train AUC: 0.792"

```

```

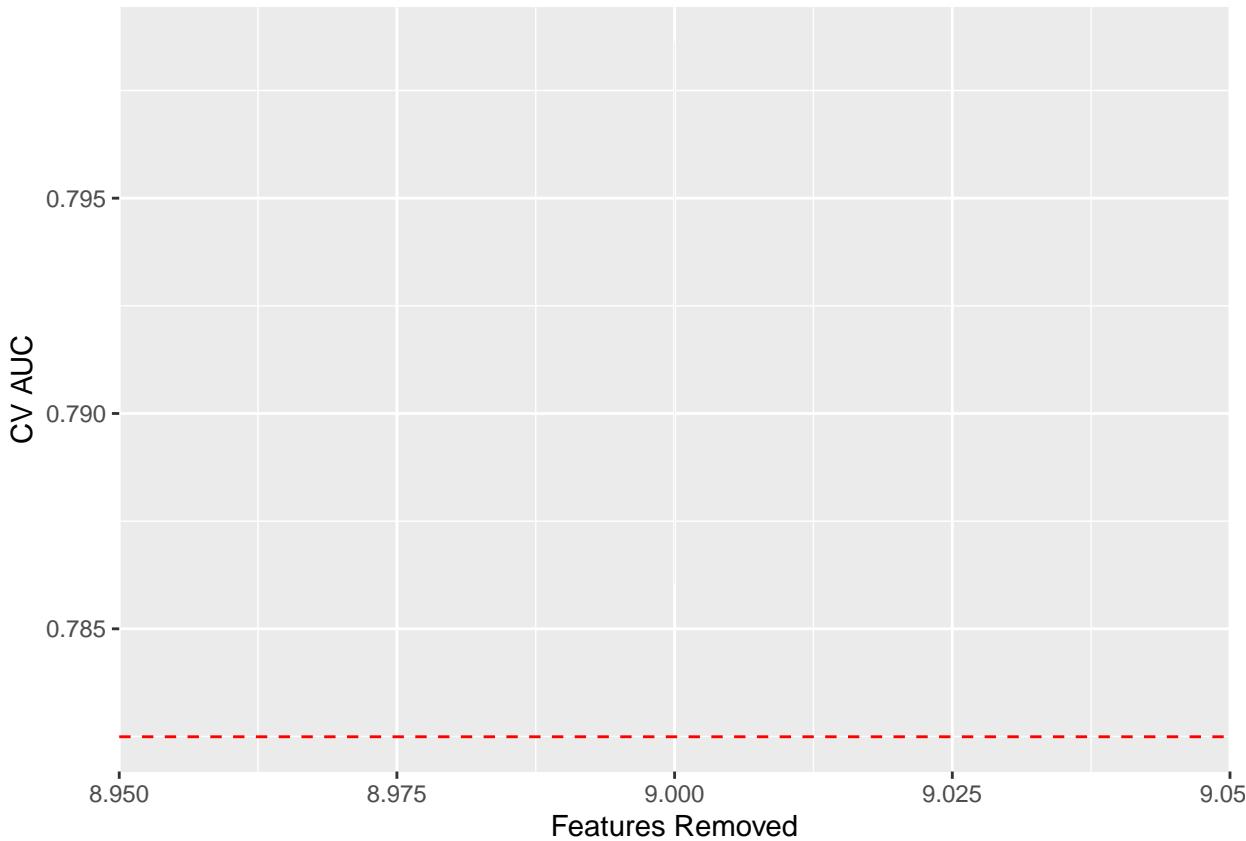
sprintf('Selected Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Selected Model Test AUC: 0.807"

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. aco
2. admission_pre_t0_180d
3. admission_pre_t0_count
4. admission_t0_emergency
5. af
6. age
7. analises_clinicas_qtde
8. angio_tc
9. antiarritmico

10. antifungico
11. antiplaquetario_ev
12. bic
13. cardiac_arrest
14. cateter_venoso_central
15. cateterismo
16. cied_final_1
17. cied_final_group_1
18. cintilografia
19. citologia
20. classe_meds_qtde
21. comorbidities_count
22. copd
23. cultura
24. diabetes
25. dialysis_t0
26. digoxina
27. diuretico
28. dva
29. ecocardiograma
30. education_level
31. endoscopia
32. equipe_multiprof
33. espironolactona
34. estatina
35. exames_imagem_qtde
36. flebografia
37. heart_disease
38. histopatologia_qtde
39. holter
40. hospital_stay
41. hypertension
42. icp
43. icu_t0
44. ieca_bra
45. insuf_cardiaca
46. insulina
47. interconsulta
48. laboratorio
49. meds_antimicrobianos
50. meds_cardiovasc_qtde
51. metodos_graficos_qtde
52. nyha Basal
53. outros_proced_cirurgicos
54. pet_ct
55. prior_mi
56. proced_invasivos_qtde
57. procedure_type_1
58. procedure_type_new
59. psicofarmacos
60. renal_failure
61. ressonancia
62. stroke
63. teste_esforco
64. tilt_teste
65. tomografia
66. ultrassom
67. underlying_heart_disease
68. valvopathy
69. vasodilatador
70. ventilacao_mecanica

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(),
    loss_reduction = tune(),
    sample_size = 1.0
  ) %>%
    set_engine("lightgbm") %>%
    set_mode("classification")

  lightgbm_grid <- grid_latin_hypercube(
    trees(range = c(50L, 300L)),
    min_n(),
    tree_depth(),
    learn_rate(range = c(0.01, 0.3), trans = NULL),
    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")

  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)

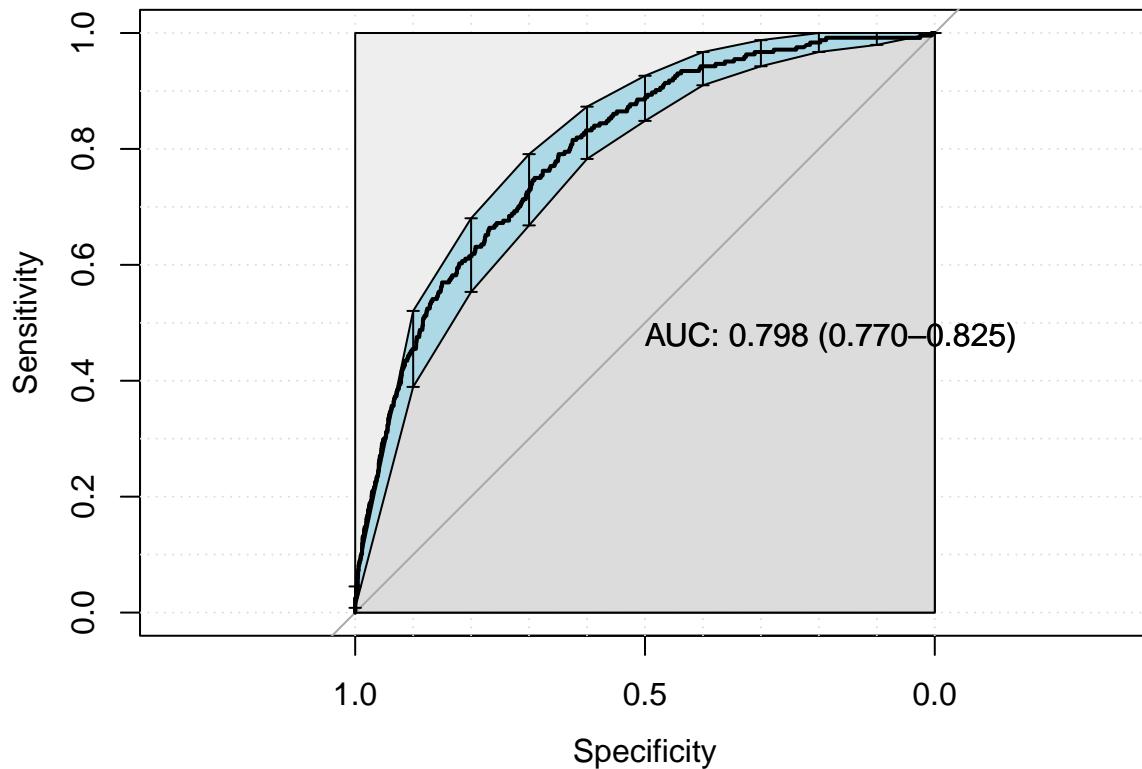
lightgbm_auc <- validation(final_lightgbm_fit, df_test)

lightgbm_parameters <- lightgbm_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(trees, min_n, tree_depth, learn_rate, 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: 0.04"
## Confusion Matrix and Statistics
##
##      reference
## data      0     1
##   0 2804    45
##   1 1682   199
##
##                  Accuracy : 0.6349
##                  95% CI : (0.621, 0.6486)
##      No Information Rate : 0.9484
##      P-Value [Acc > NIR] : 1
##
##                  Kappa : 0.1056
## 

```

```

## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.6251
##          Specificity : 0.8156
##          Pos Pred Value : 0.9842
##          Neg Pred Value : 0.1058
##          Prevalence : 0.9484
##          Detection Rate : 0.5928
##          Detection Prevalence : 0.6023
##          Balanced Accuracy : 0.7203
##
##          'Positive' Class : 0
##
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)

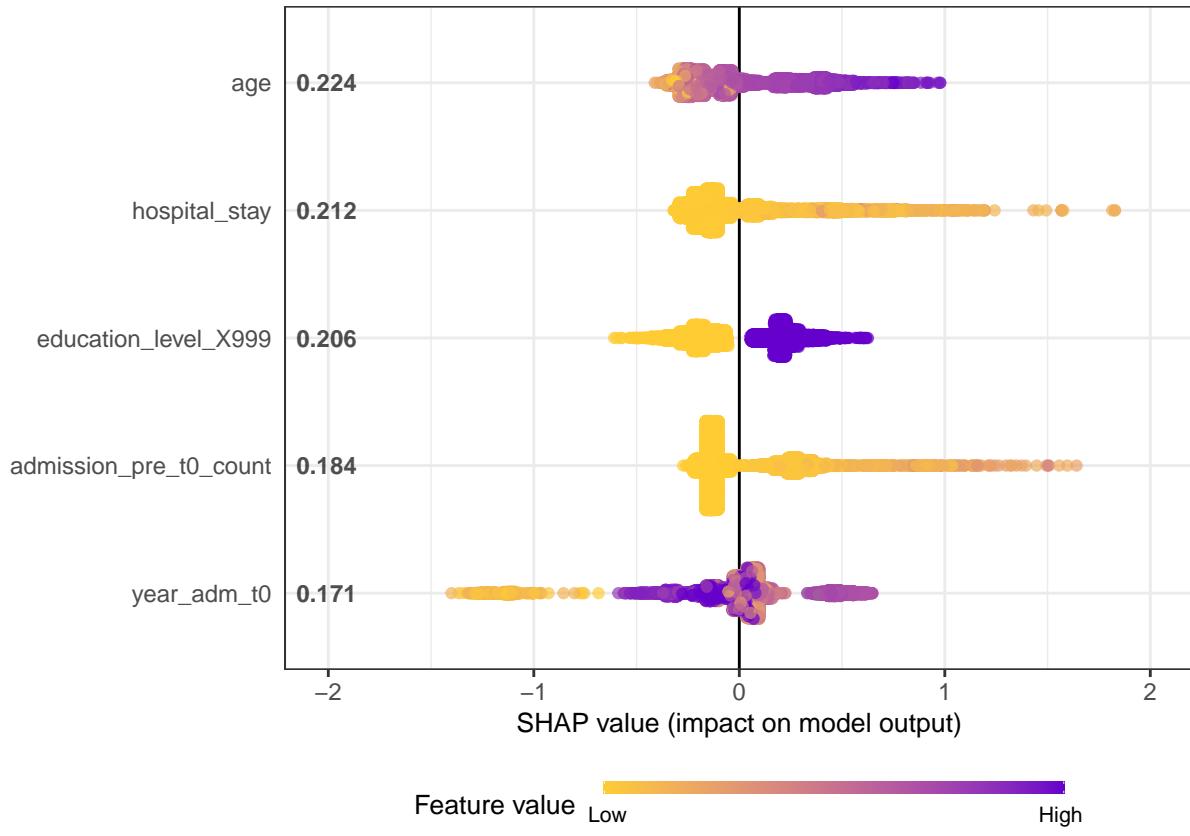
df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

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))

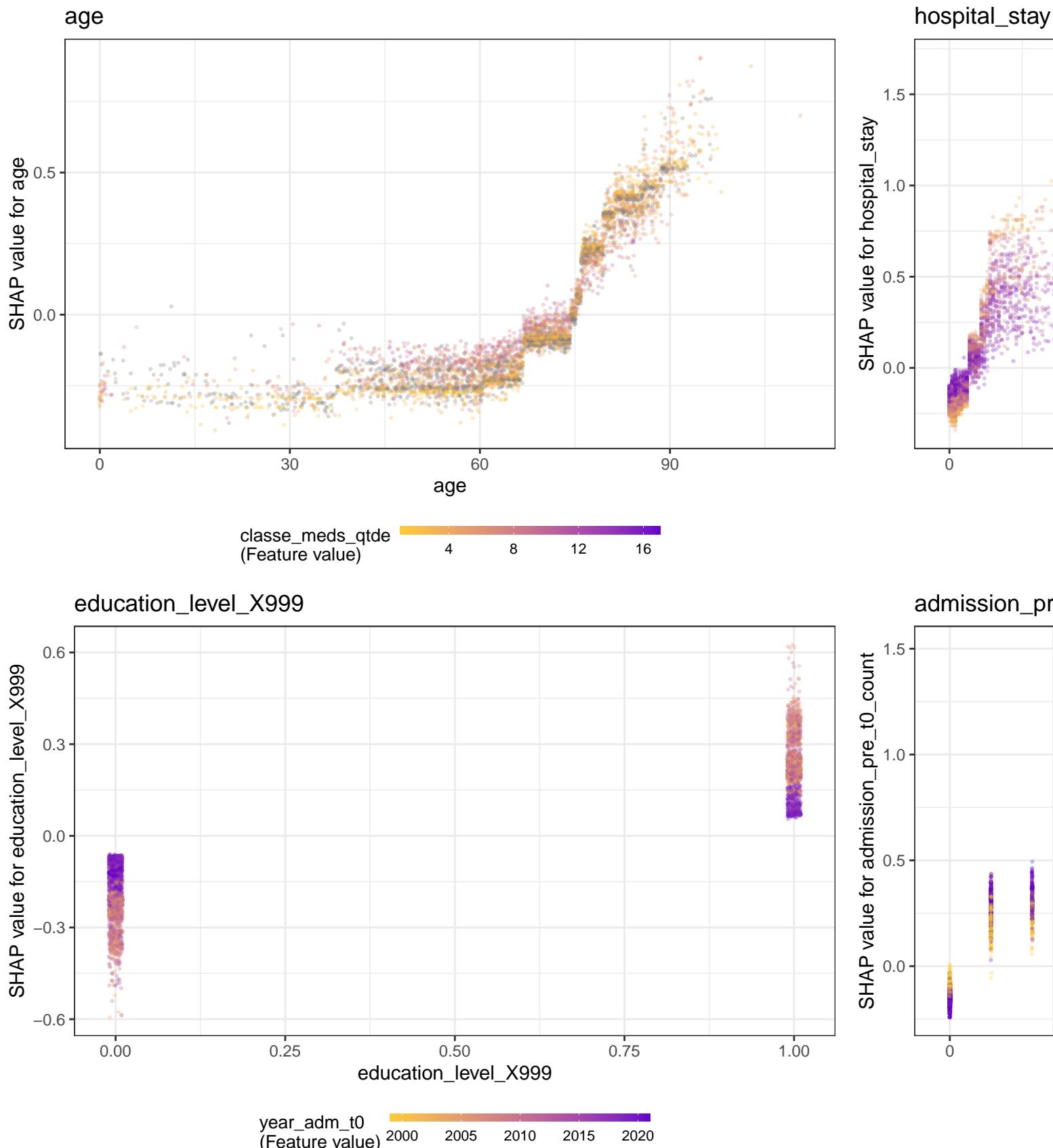
shap.plot.summary.wrap1(model = lightgbm_model, X = matrix_train,
                       top_n = n_plots, dilute = F)

```

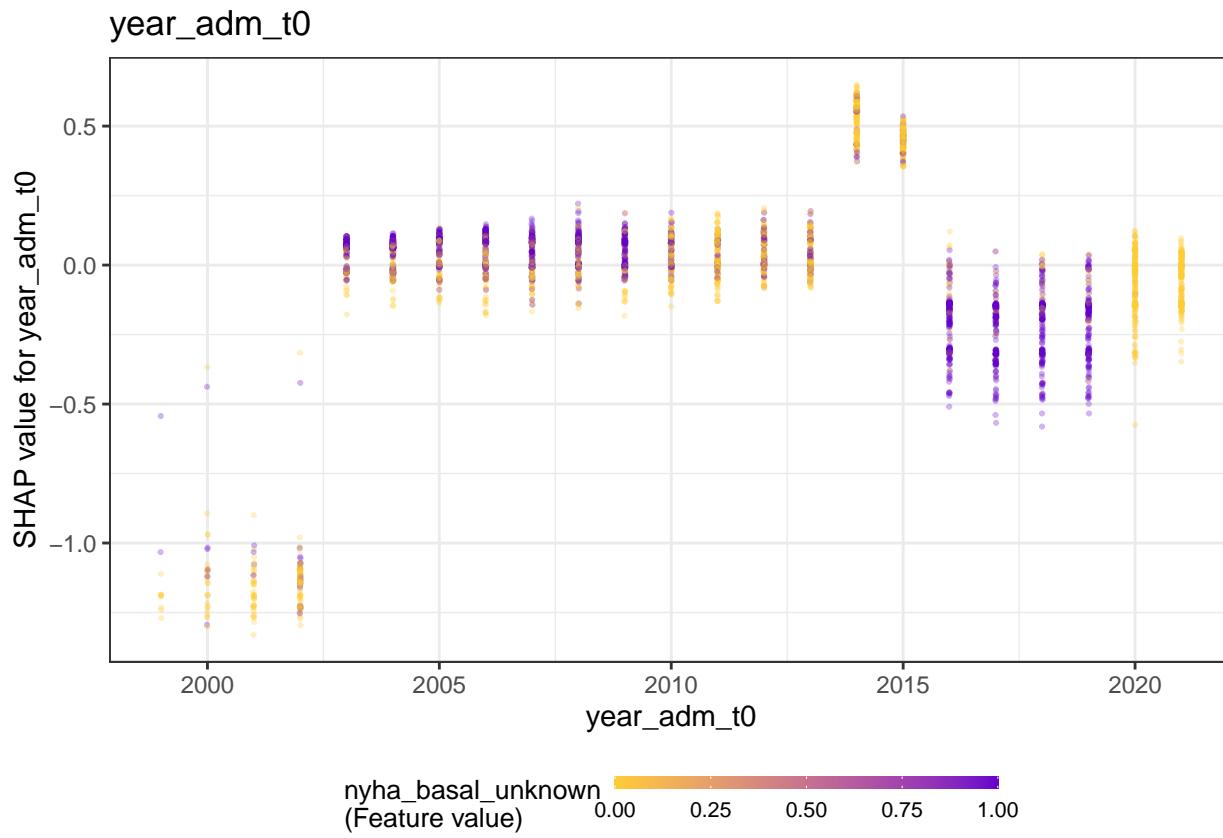


```
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

for (x in shap.importance(shap, names_only = TRUE)[1:n_plots]) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = FALSE,
    jitter_width = 0.01,
    alpha = 0.3
  ) +
    labs(title = x)
  print(p)
}
```



```
## Warning: Removed 5 rows containing missing values (geom_point).
```



```
## $num_iterations
## [1] 93
##
## $learning_rate
## [1] 0.05713298
##
## $max_depth
## [1] 3
##
## $feature_fraction
## [1] 1
##
## $min_data_in_leaf
## [1] 38
##
## $min_gain_to_split
## [1] 1.075483e-06
##
## $bagging_fraction
## [1] 1
##
## $num_class
## [1] 1
##
## $objective
## [1] "binary"
##
## $num_threads
## $num_threads$num_threads
## [1] 0
##
## $seed
```

```

## [1] 55923
##
## $deterministic
## [1] TRUE
##
## $verbose
## [1] -1
##
## $metric
## list()
##
## $interaction_constraints
## list()
##
## $feature_pre_filter
## [1] FALSE

```

Models Comparison

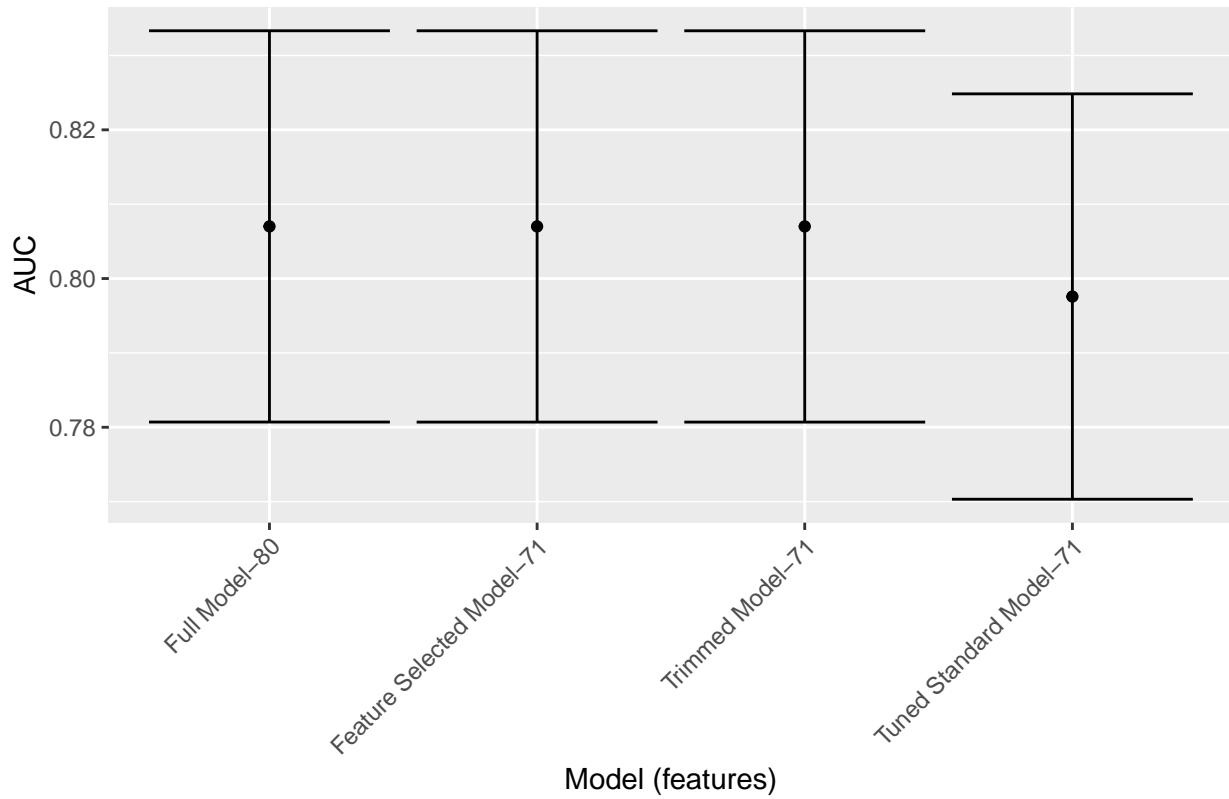
```

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))

```

death_2year



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