Final Model - death_180days

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. 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. 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. cateterismo
54. cateter_venoso_central
55. proced_invasivos_qtde
56. transfusao
57. interconsulta
58. equipe_multiprof
59. holter
60. teste_esforco
61. metodos_graficos_qtde
62. laboratorio
63. cultura
64. analises_clinicas_qtde
65. citologia
66. histopatologia_qtde
67. angiografia
68. aortografia
69. arteriografia
70. cintilografia
71. ecocardiograma
72. endoscopia
73. ultrassom
74. tomografia
75. ressonancia
76. exames_imagem_qtde
77. bic
78. 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: 0.778"
```

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

```
## [1] "Full Model Test AUC: 0.808"
```

Features with zero importance on the initial model:

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

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

1. education_level
2. underlying_heart_disease
3. heart_disease
4. hemodialysis
5. cied_final_1
6. dialysis_t0
7. vni
8. transplante_cardiaco
9. outros_proced_cirurgicos
10. icp
11. cateter_venoso_central
12. transfusao
13. teste_esforco
14. angiografia
15. aortografia
16. arteriografia

```
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: 0.776"
```

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

```
## [1] "Trimmed Model Test AUC: 0.803"
```

```
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
78	0.7778	0.0186	0.0000	endoscopia
62	0.7762	0.0170	0.0016	antiplaquetario_ev
61	0.7751	0.0168	0.0026	procedure_type_1
60	0.7693	0.0169	0.0085	bic
59	0.7736	0.0174	0.0042	aco
58	0.7706	0.0163	0.0072	valvopathy
57	0.7745	0.0165	0.0033	histopatologia_qtde
56	0.7724	0.0173	0.0053	ventilacao_mecanica
55	0.7709	0.0174	0.0069	antifungico
54	0.7716	0.0158	0.0062	insulina
53	0.7717	0.0165	0.0061	ressonancia
52	0.7715	0.0173	0.0063	cardiac_arrest
51	0.7673	0.0157	0.0105	diabetes

```
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)
```

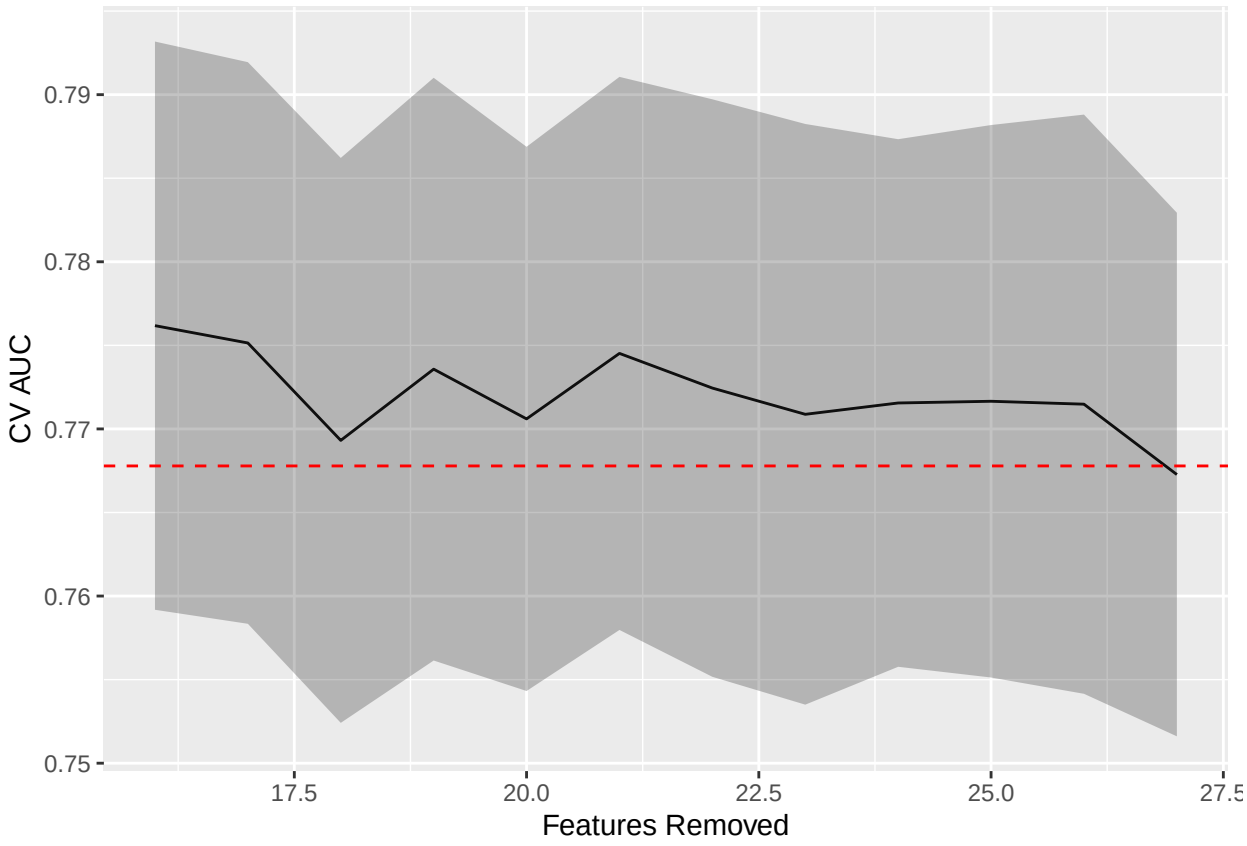
```
close(con)

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: 0.766"
```

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

```
## [1] "Selected Model Test AUC: 0.782"
```

```
 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)
```

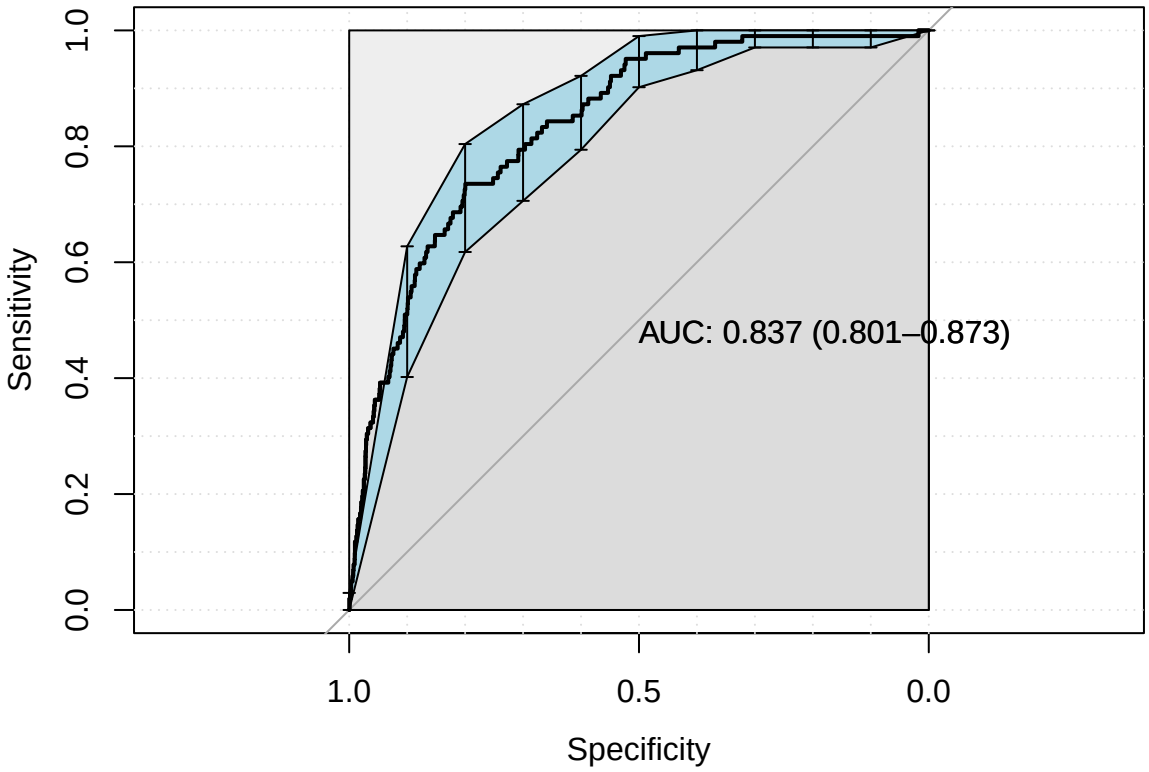
1. hospital_stay
2. age
3. laboratorio
4. vasodilatador
5. espironolactona
6. year_adm_t0
7. ieca_bra
8. analises_clinicas_qtde
9. admission_pre_t0_count
10. comorbidities_count
11. icu_t0
12. equipe_multiprof
13. meds_cardiovasc_qtde
14. meds_antimicrobianos
15. diuretico
16. classe_meds_qtde
17. antiarritmico
18. estatina
19. admission_pre_t0_180d
20. dva
21. insuf_cardiaca
22. psicofarmacos
23. exames_imagem_qtde
24. ecocardiograma
25. stroke
26. metodos_graficos_qtde
27. cied_final_group_1
28. admission_t0_emergency
29. cateterismo
30. citologia
31. interconsulta
32. proced_invasivos_qtde
33. cintilografia
34. renal_failure
35. ultrassom
36. af
37. sex
38. cancer
39. copd
40. tomografia
41. diabetes
42. cultura
43. prior_mi
44. holter
45. reop_type_1
46. heart_failure
47. hypertension
48. procedure_type_new
49. digoxina
50. nyha_basal
51. endoscopia

Standard

```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
```

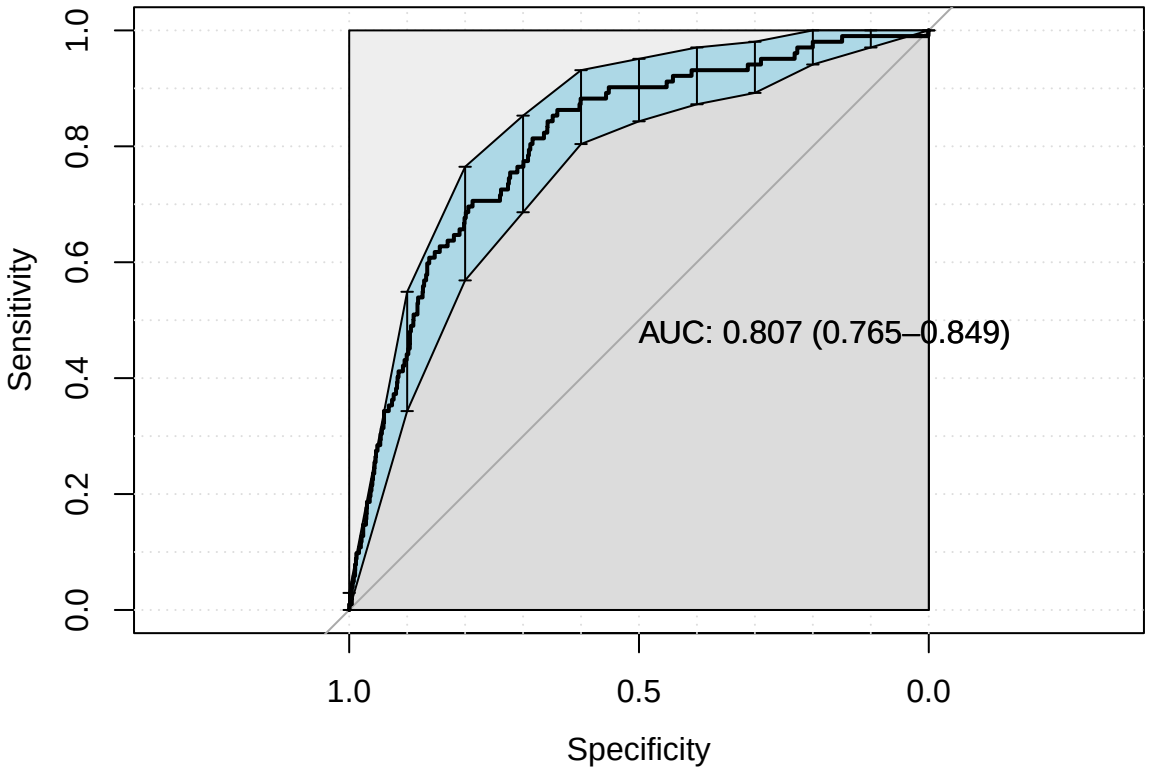
```
        data = df_train %>% select(all_of(c(selected_features, outcome_column)))) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors())

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)
```



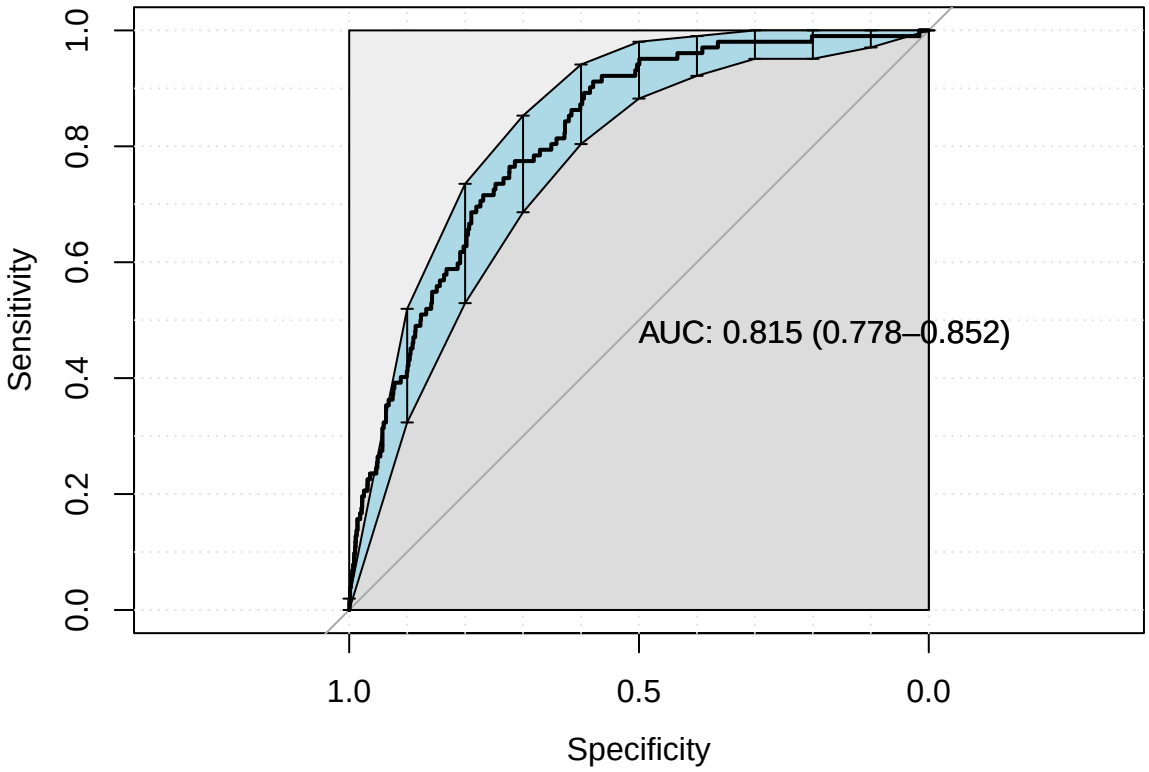

```
## [1] "Optimal Threshold: 0.02"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3701   27
##    1  927   75
##
##                Accuracy : 0.7983
##                  95% CI : (0.7866, 0.8097)
##     No Information Rate : 0.9784
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1007
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.79970
##             Specificity : 0.73529
##          Pos Pred Value : 0.99276
##          Neg Pred Value : 0.07485
##              Prevalence : 0.97844
##          Detection Rate : 0.78245
##    Detection Prevalence : 0.78816
##       Balanced Accuracy : 0.76750
##
##        'Positive' Class : 0
##
```

```
smote_results <- lightgbm_tuning(lightgbm_smote_recipe)
```




```
## [1] "Optimal Threshold: 0.50"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 2966   14
##    1 1662   88
##
##                Accuracy : 0.6457
##                  95% CI : (0.6318, 0.6593)
##     No Information Rate : 0.9784
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0566
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.64088
##             Specificity : 0.86275
##          Pos Pred Value : 0.99530
##          Neg Pred Value : 0.05029
##              Prevalence : 0.97844
##          Detection Rate : 0.62706
##    Detection Prevalence : 0.63002
##       Balanced Accuracy : 0.75181
##
##        'Positive' Class : 0
##
```

```
upsample_results <- lightgbm_tuning(lightgbm_upsample_recipe)
```



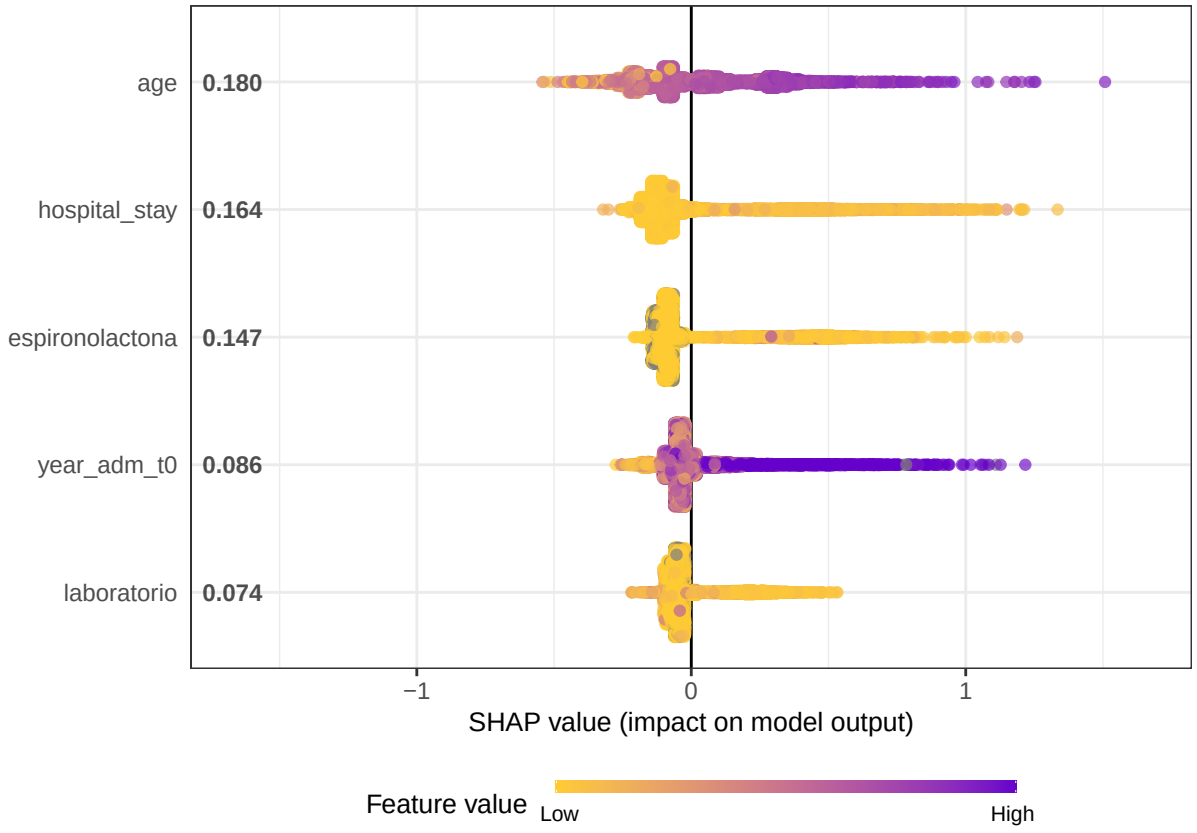

```
## [1] "Optimal Threshold: 0.50"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 2680    9
##    1 1948   93
##
##                Accuracy : 0.5863
##                  95% CI : (0.5721, 0.6003)
##     No Information Rate : 0.9784
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0477
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.57908
##             Specificity : 0.91176
##          Pos Pred Value : 0.99665
##          Neg Pred Value : 0.04557
##              Prevalence : 0.97844
##          Detection Rate : 0.56660
##    Detection Prevalence : 0.56850
##       Balanced Accuracy : 0.74542
##
##        '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

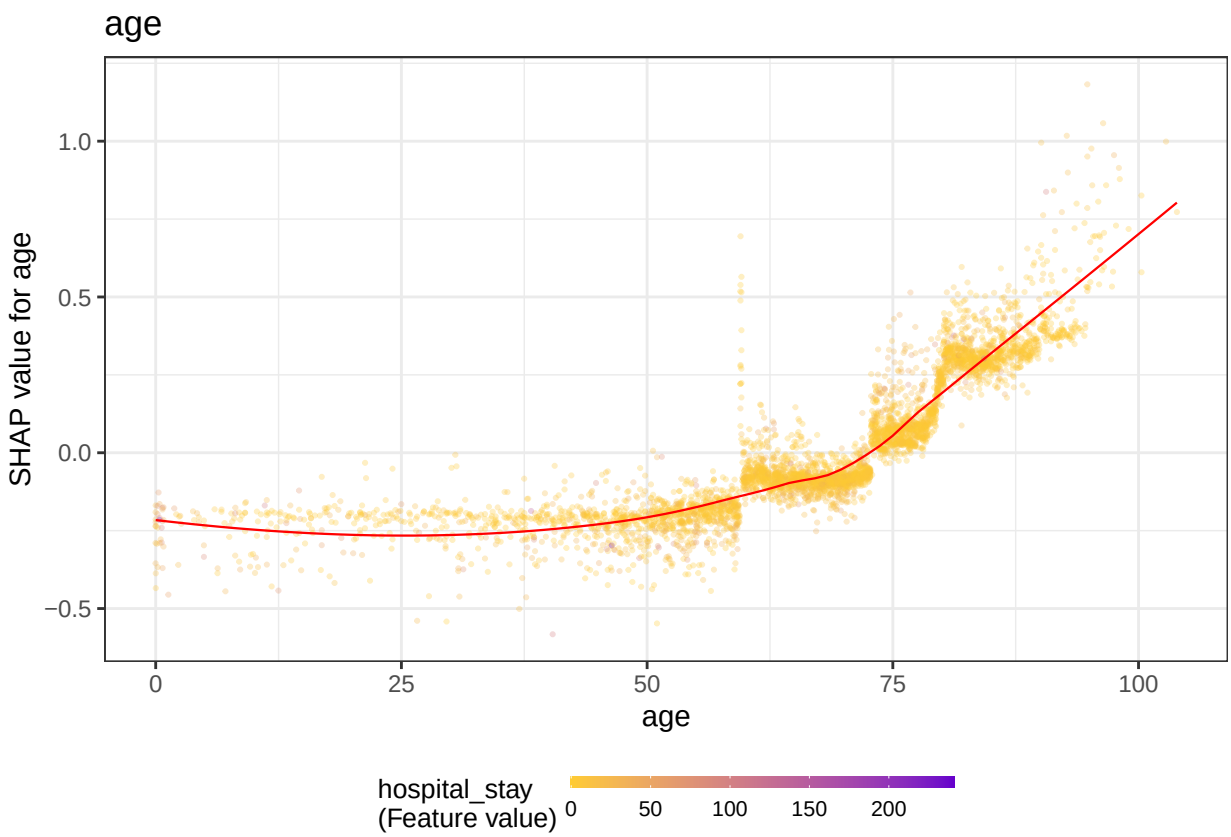
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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)
```



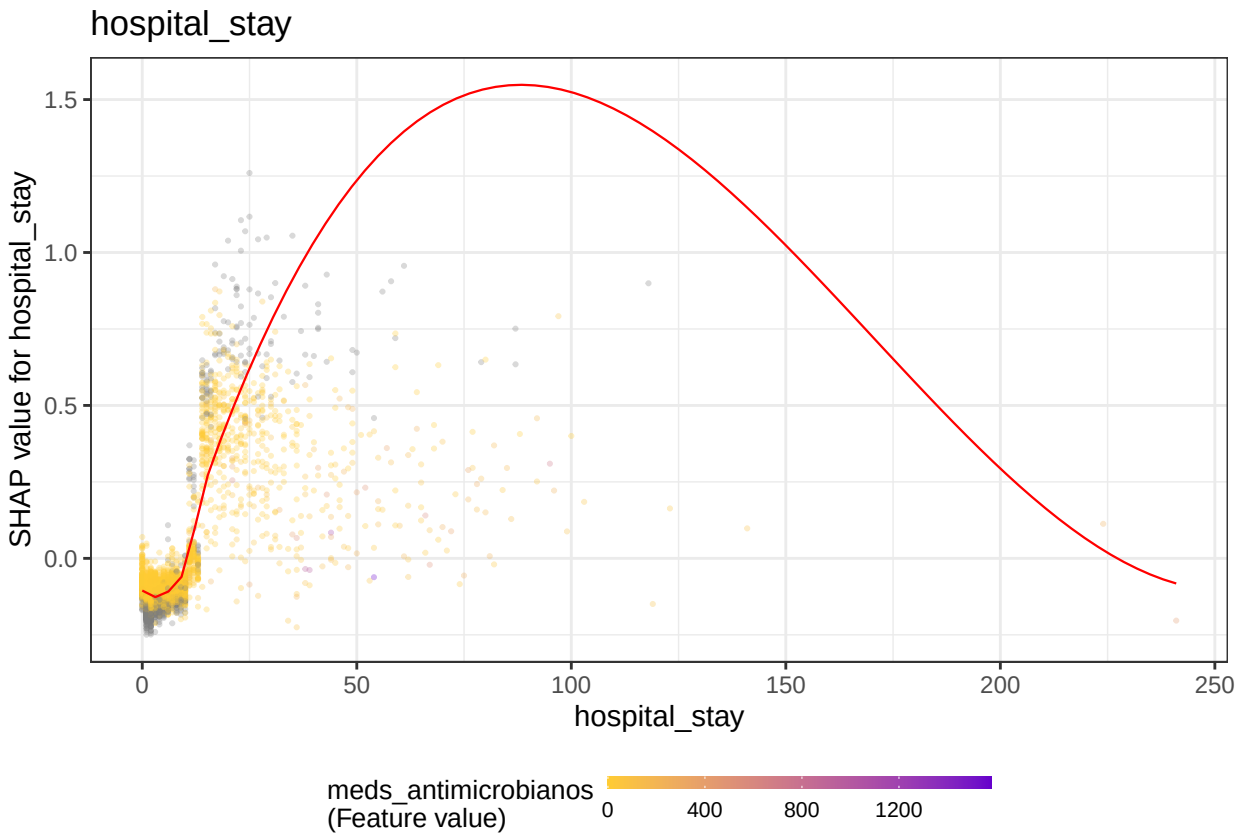

```
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

for (x in shap.importance(shap, names_only = TRUE)[1:n_plots]) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = TRUE,
    jitter_width = 0.01,
```

```
    alpha = 0.3
  ) +
    labs(title = x)
  print(p)
}
```

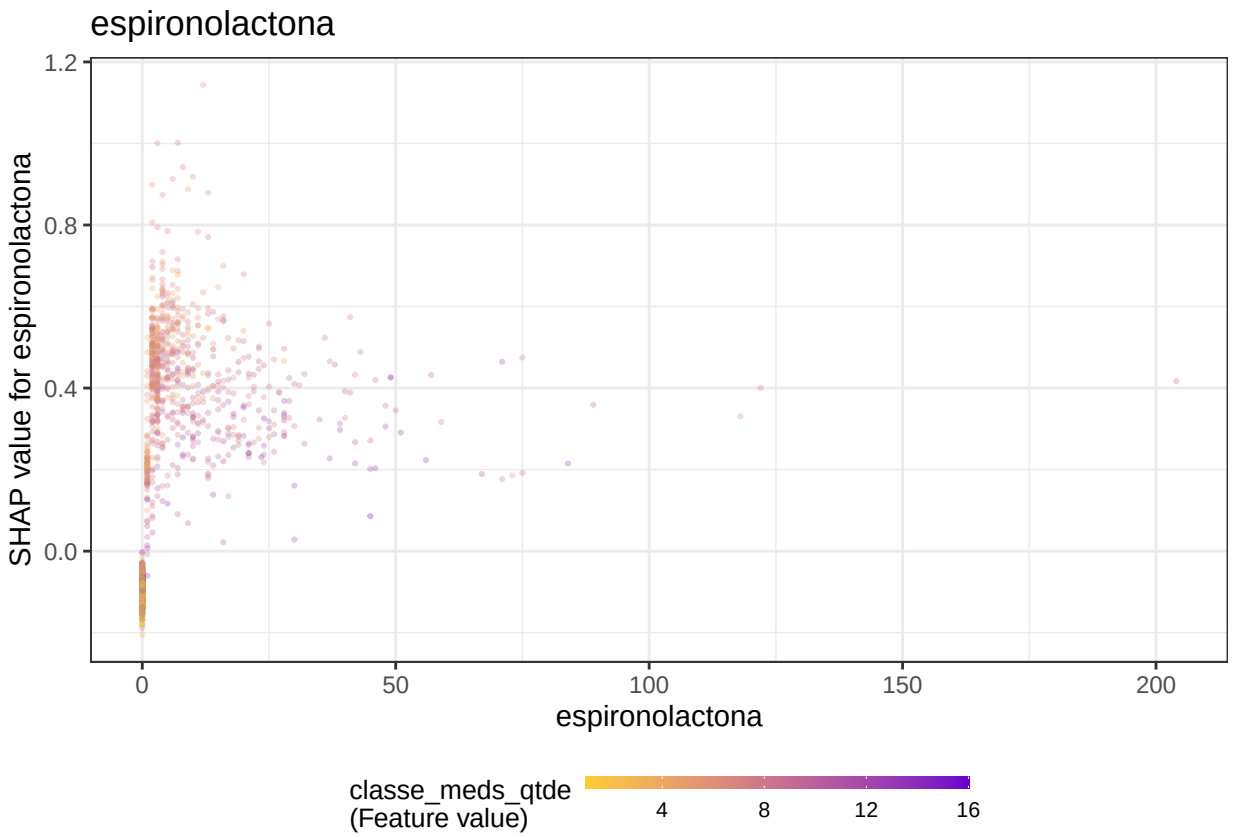
```
## `geom_smooth()` using formula 'y ~ x'
```



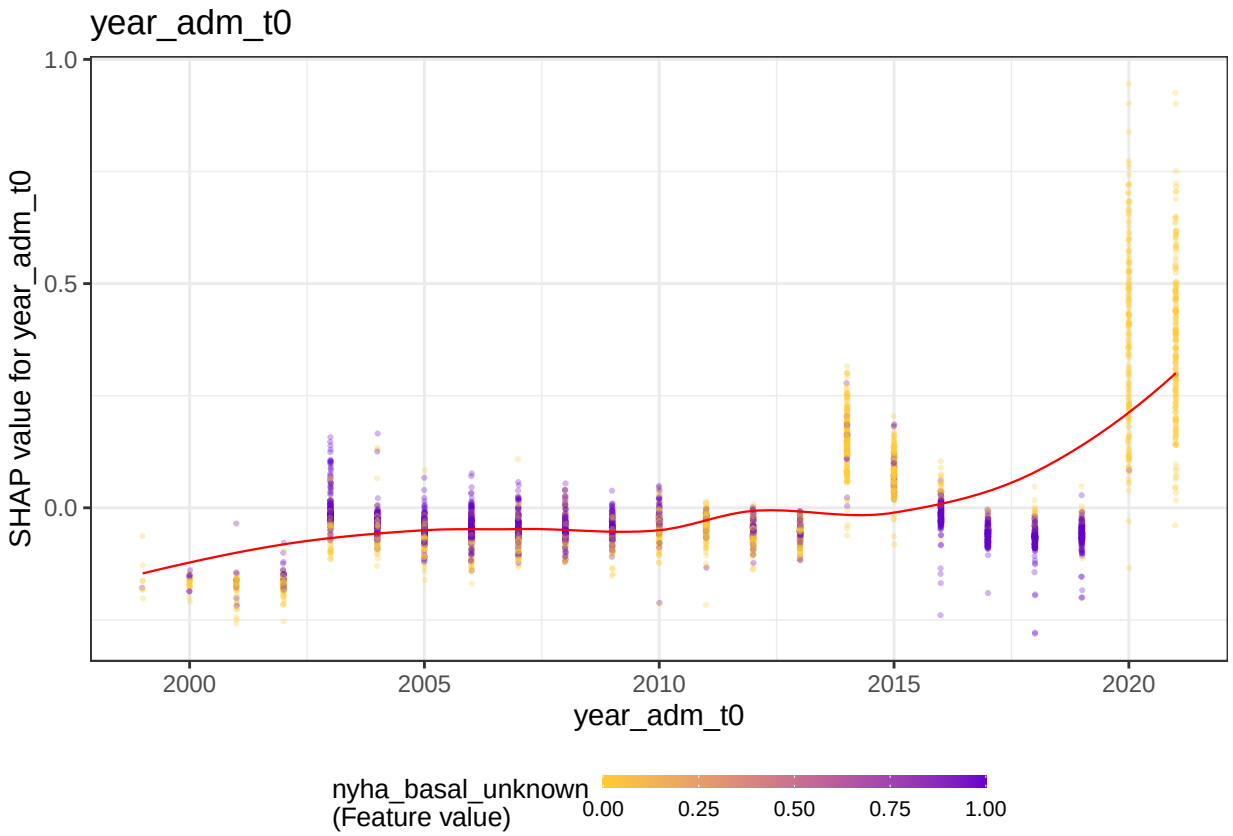

```
## `geom_smooth()` using formula 'y ~ x'
```




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

## Warning: Removed 1041 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 1041 rows containing missing values (geom_point).
```

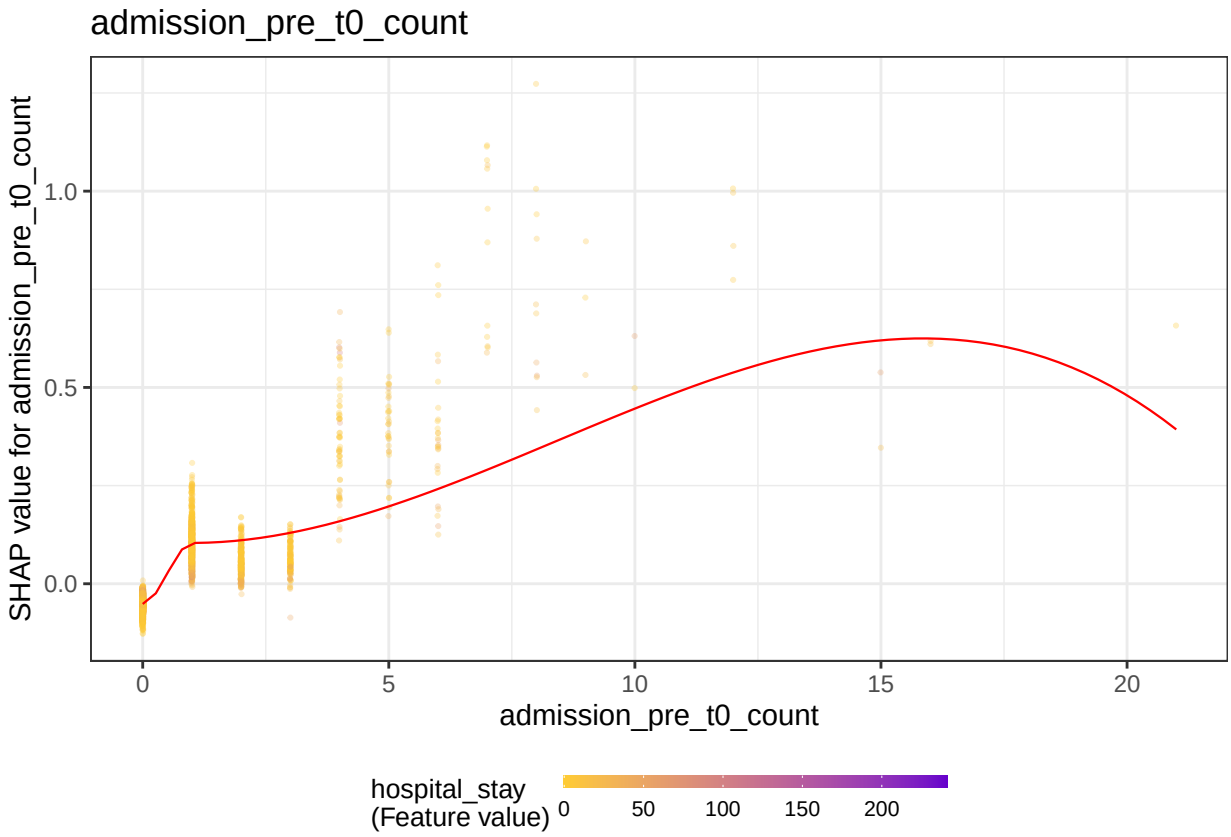



```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 7 rows containing non-finite values (stat_smooth).
## Warning: Removed 7 rows containing missing values (geom_point).
```




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

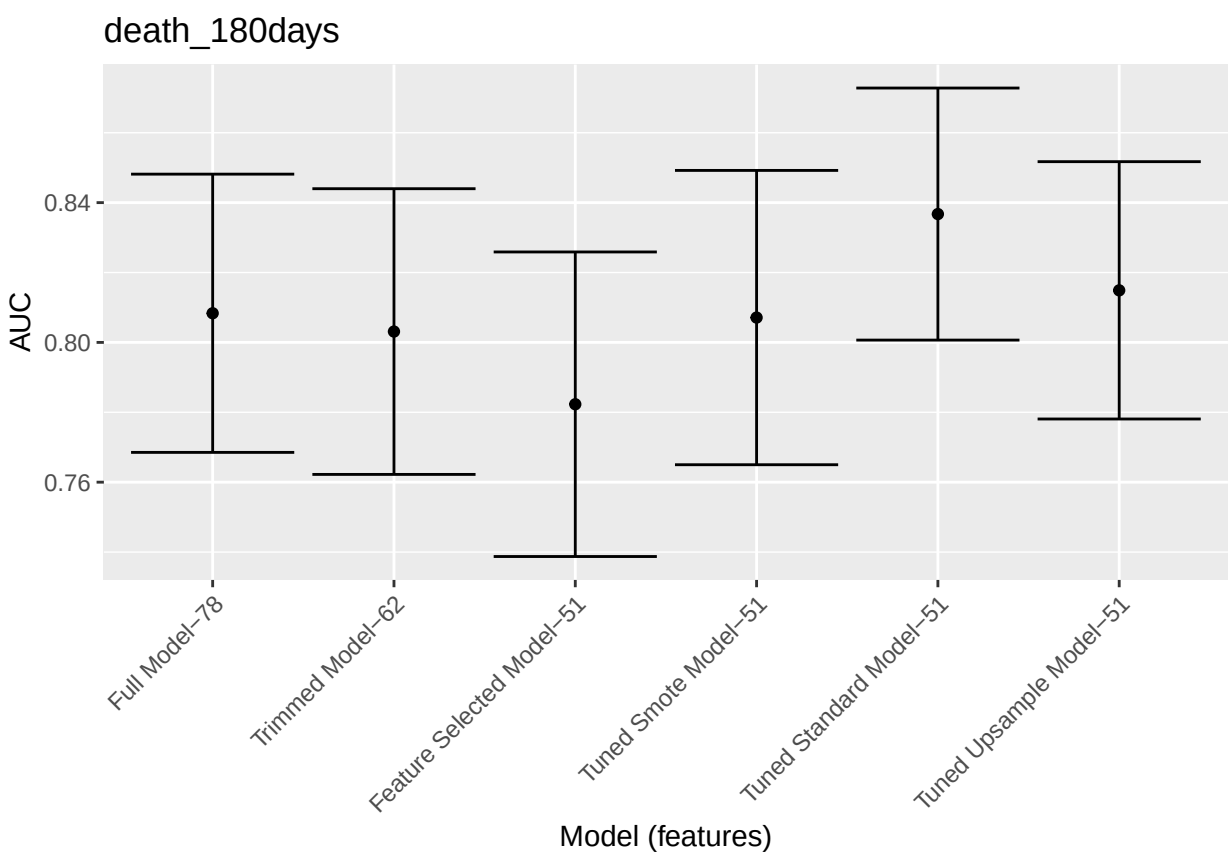
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -0.105

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.105

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 5.1924e-28

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1
```



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_feat
  'Tuned Upsample Model', upsample_results$auc, upsample_results$auc_lower, upsample_results$auc_upper, length(s
) %>%
  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))
```