Final Model - readmission_180d

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. patient_state
5. underlying_heart_disease
6. heart_disease
7. nyha_basal
8. prior_mi
9. heart_failure
10. af
11. cardiac_arrest
12. transplant
13. valvopathy
14. endocardites
15. diabetes
16. renal_failure
17. hemodialysis
18. copd
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. icu_t0
28. dialysis_t0
29. n_procedure_t0
30. admission_t0_emergency
31. aco
32. antiarritmico
33. betabloqueador
34. ieca_bra
35. dva
36. digoxina
37. estatina
38. diuretico
39. vasodilatador
40. insuf_cardiaca
41. espironolactona
42. bloq_calcio
43. antiplaquetario_ev
44. insulina
45. anticonvulsivante
46. psicofarmacos
47. antifungico
48. antiviral
49. antiretroviral
50. classe_meds_qtde
51. meds_cardiovasc_qtde
52. meds_antimicrobianos
53. vni
54. ventilacao_mecanica
55. cec
56. transplante_cardiaco
57. cir_toracica
58. outros_proced_cirurgicos
59. icp
60. intervencao_cv
61. angioplastia
62. cateterismo
63. eletrofisiologia
64. cateter_venoso_central
65. proced_invasivos_qtde
66. cve_desf
67. transfusao
68. interconsulta
69. equipe_multiprof
70. holter
71. teste_esforco
72. espiro_ergoespiro
73. tilt_teste
74. metodos_graficos_qtde
75. laboratorio
76. cultura
77. analises_clinicas_qtde
78. citologia
79. biopsia
80. histopatologia_qtde
81. angio_rm
82. angio_tc
83. arteriografia
84. cintilografia
85. ecocardiograma

86. endoscopia
87. flebografia
88. pet_ct
89. ultrassom
90. tomografia
91. ressonancia
92. exames_imagem_qtde
93. bic
94. mpp
95. 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: 0.720"
```

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

```
## [1] "Full Model Test AUC: 0.700"
```

Features with zero importance on the initial model:

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

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

1. hemodialysis
2. antiretroviral
3. transplante_cardiaco
4. cir_toracica
5. angioplastia
6. transfusao
7. tilt_teste
8. angio_rm
9. arteriografia
10. 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: 0.719"
```

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

```
## [1] "Trimmed Model Test AUC: 0.701"
```

```r
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
95	0.7204	0.0137	0.0000	dialysis_t0
85	0.7189	0.0134	0.0015	cec
84	0.7219	0.0135	-0.0015	citologia
83	0.7204	0.0140	0.0000	renal_failure
82	0.7211	0.0133	-0.0007	endocardites
81	0.7239	0.0136	-0.0034	teste_esforco
80	0.7207	0.0124	-0.0003	dialysis_t0
79	0.7220	0.0120	-0.0016	angio_tc
78	0.7215	0.0136	-0.0010	heart_disease
77	0.7212	0.0145	-0.0008	antifungico

Table 1: *(continued)*

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
76	0.7184	0.0142	0.0021	intervencao_cv
75	0.7228	0.0136	-0.0023	transplant
74	0.7233	0.0136	-0.0029	cve_desf
73	0.7212	0.0124	-0.0007	vni
72	0.7204	0.0132	0.0001	espiro_ergoespiro
71	0.7228	0.0136	-0.0024	ventilacao_mecanica
70	0.7222	0.0138	-0.0018	valvopathy
69	0.7222	0.0131	-0.0018	biopsia
68	0.7216	0.0128	-0.0012	mpp
67	0.7229	0.0133	-0.0025	prior_mi
66	0.7221	0.0131	-0.0017	insulina
65	0.7210	0.0141	-0.0006	ressonancia
64	0.7215	0.0138	-0.0010	cateter_venoso_central
63	0.7217	0.0127	-0.0012	cintilografia
62	0.7183	0.0135	0.0022	icp
61	0.7204	0.0141	0.0001	cateterismo
60	0.7235	0.0130	-0.0031	heart_failure
59	0.7202	0.0142	0.0003	procedure_type_1
58	0.7230	0.0121	-0.0025	antiviral
57	0.7189	0.0126	0.0016	cardiac_arrest
56	0.7234	0.0124	-0.0029	eletrofisiologia
55	0.7215	0.0127	-0.0011	tomografia
54	0.7225	0.0128	-0.0021	outros_proced_cirurgicos
53	0.7224	0.0120	-0.0019	cultura
52	0.7229	0.0117	-0.0025	ultrassom
51	0.7224	0.0118	-0.0020	antiplaquetario_ev
50	0.7233	0.0113	-0.0028	bloq_calcio
49	0.7219	0.0114	-0.0014	procedure_type_new
48	0.7194	0.0115	0.0011	interconsulta
47	0.7232	0.0106	-0.0028	flebografia
46	0.7228	0.0106	-0.0024	aco
45	0.7231	0.0112	-0.0026	n_procedure_t0
44	0.7212	0.0106	-0.0008	diabetes
43	0.7236	0.0116	-0.0031	copd
42	0.7233	0.0112	-0.0028	holter
41	0.7237	0.0122	-0.0032	cied_final_group_1
40	0.7222	0.0118	-0.0017	analises_clinicas_qtde
39	0.7217	0.0129	-0.0013	endoscopia
38	0.7206	0.0130	-0.0002	underlying_heart_disease
37	0.7216	0.0121	-0.0012	digoxina
36	0.7226	0.0124	-0.0022	af
35	0.7234	0.0120	-0.0029	admission_t0_emergency
34	0.7240	0.0122	-0.0036	nyha_basal
33	0.7201	0.0117	0.0004	bic
32	0.7209	0.0119	-0.0004	anticonvulsivante
31	0.7199	0.0126	0.0005	sex
30	0.7217	0.0135	-0.0013	education_level
29	0.7218	0.0109	-0.0014	patient_state
28	0.7139	0.0099	0.0066	ecocardiograma
27	0.7140	0.0106	0.0064	insuf_cardiaca
26	0.7152	0.0100	0.0052	betabloqueador
25	0.7131	0.0105	0.0074	admission_pre_t0_180d
24	0.7132	0.0111	0.0073	reop_type_1

Table 1: *(continued)*

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
23	0.7101	0.0102	0.0104	espironolactona

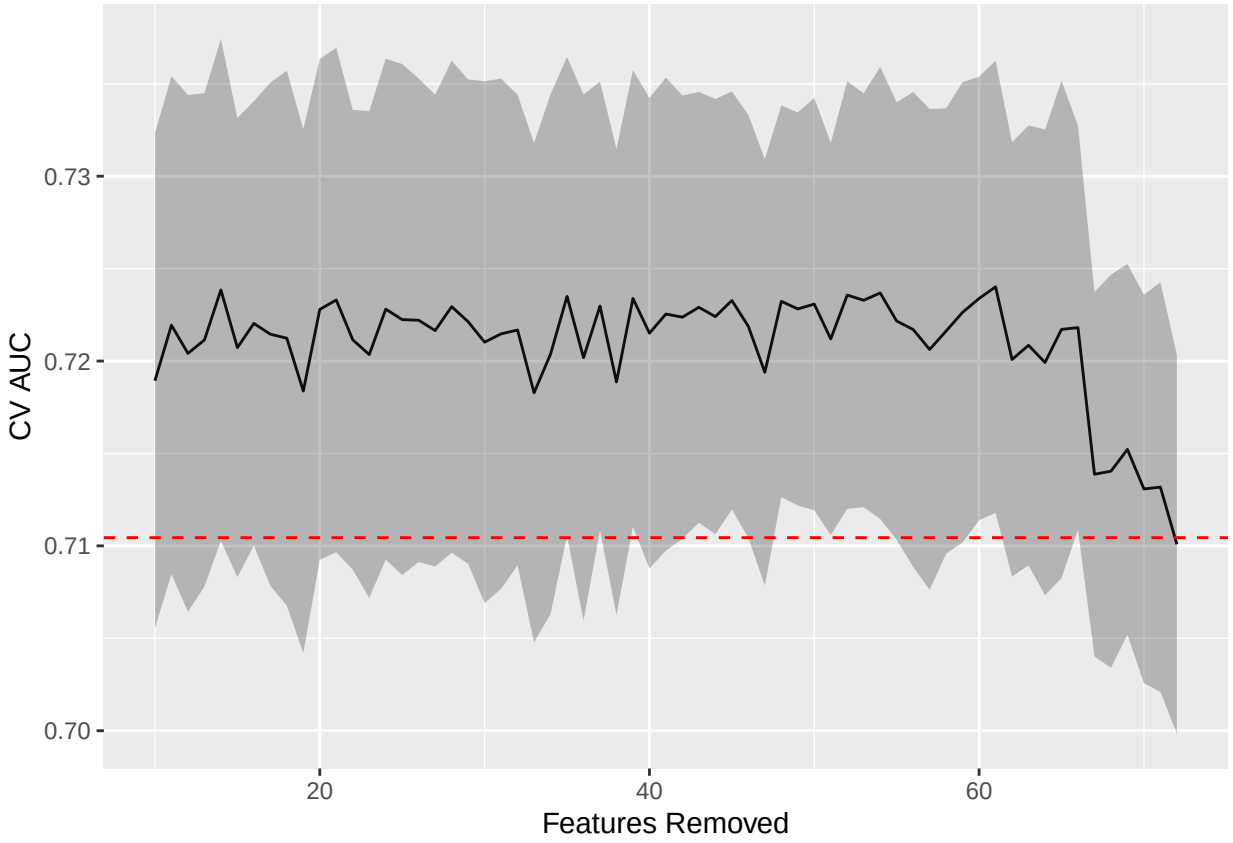
```
if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_droped)
} else {
  selected_features <- current_features
}

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: 0.710"
```

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

```
## [1] "Selected Model Test AUC: 0.671"
```

```
 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)
```

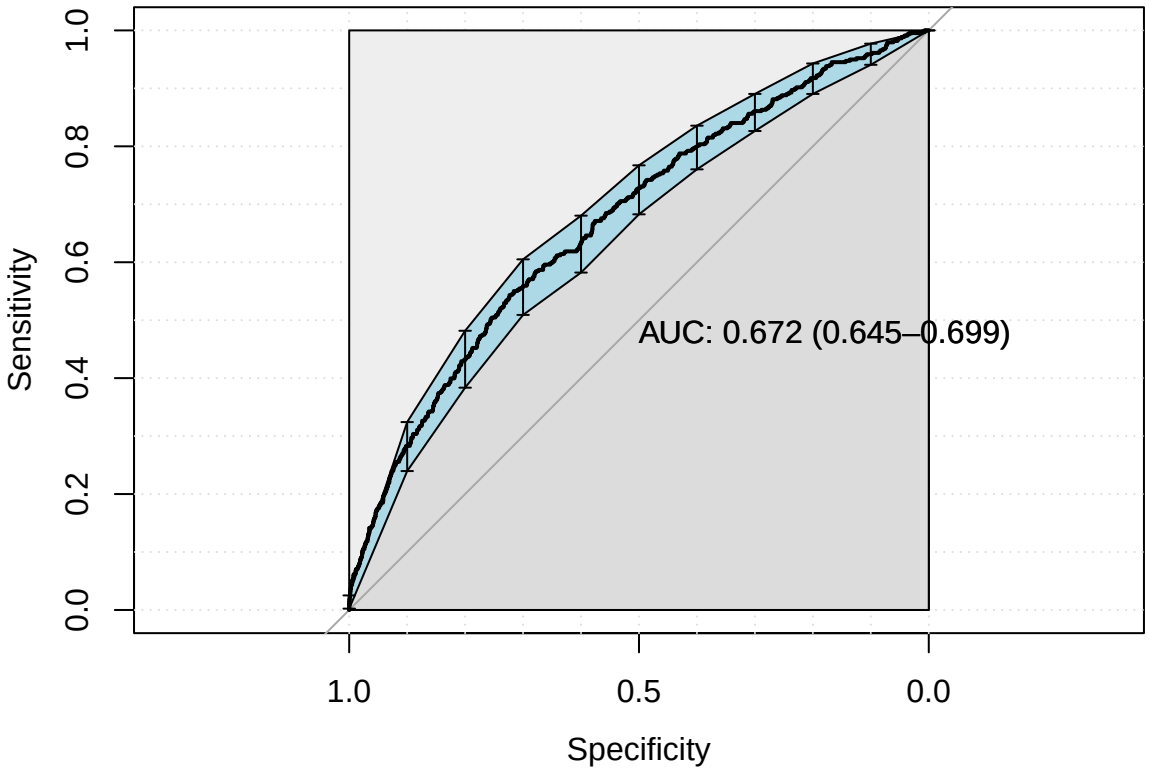
1. hospital_stay
2. age
3. meds_cardiovasc_qtde
4. admission_pre_t0_count
5. icu_t0
6. laboratorio
7. ieca_bra
8. classe_meds_qtde
9. meds_antimicrobianos
10. vasodilatador
11. metodos_graficos_qtde
12. antiarritmico
13. exames_imagem_qtde
14. diuretico
15. psicofarmacos
16. equipe_multiprof
17. estatina
18. cied_final_1
19. dva
20. histopatologia_qtde
21. comorbidities_count

22. espironolactona
23. proced_invasivos_qtde

Standard

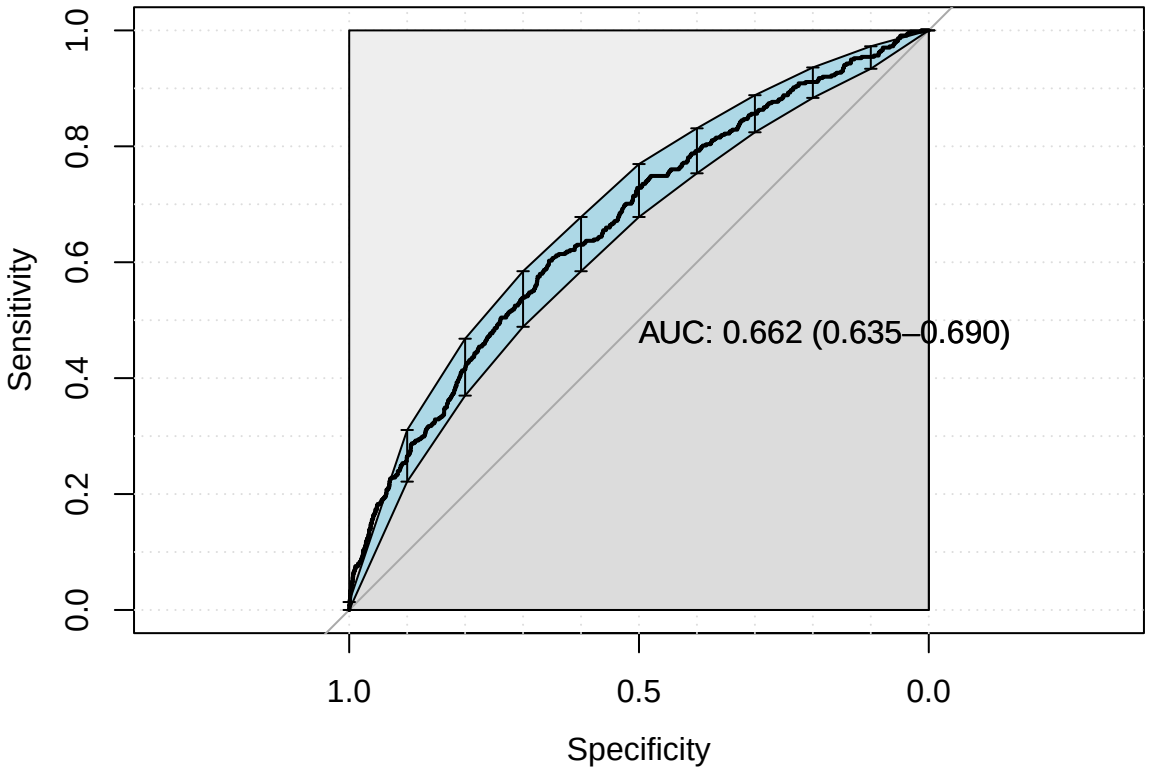
```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% select(all_of(c(selected_features, outcome_column)))) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), 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)
```



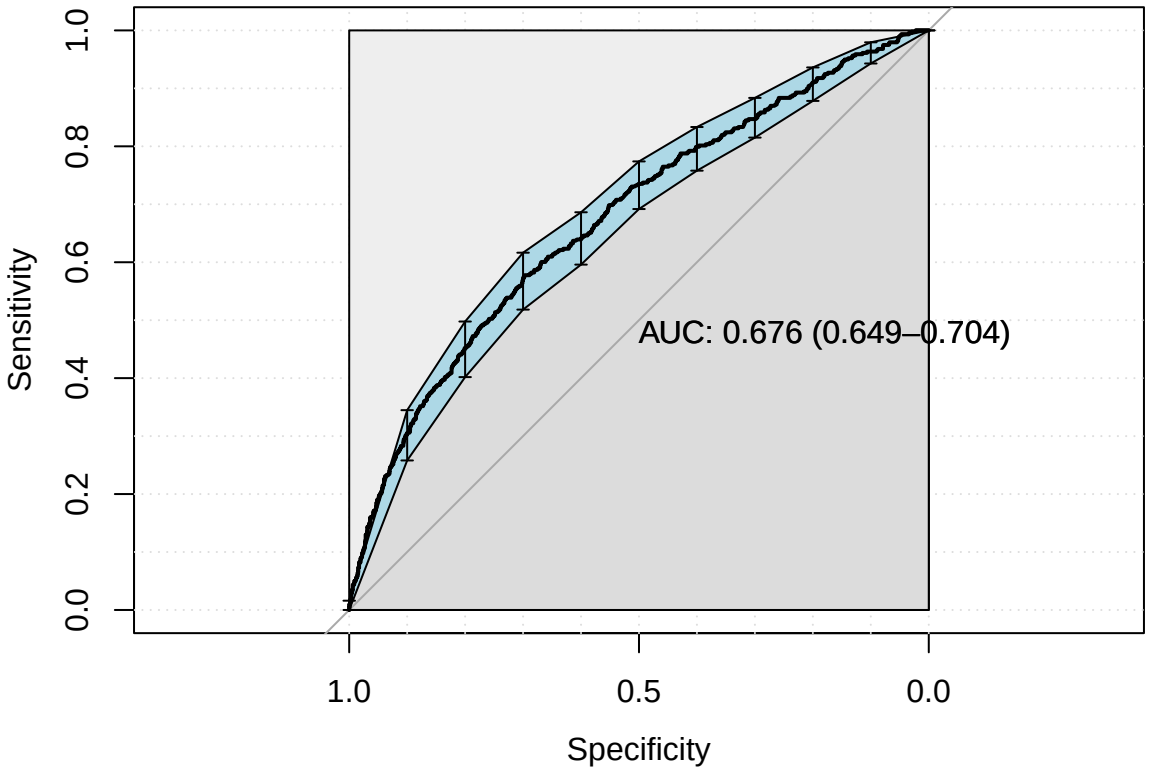

```
## [1] "Optimal Threshold: 0.09"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3105  200
##    1 1187  238
##
##                Accuracy : 0.7068
##                  95% CI : (0.6936, 0.7197)
##     No Information Rate : 0.9074
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1326
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7234
##             Specificity : 0.5434
##          Pos Pred Value : 0.9395
##          Neg Pred Value : 0.1670
##              Prevalence : 0.9074
##          Detection Rate : 0.6564
##    Detection Prevalence : 0.6987
##       Balanced Accuracy : 0.6334
##
##        'Positive' Class : 0
##
```

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




```
## [1] "Optimal Threshold: 0.12"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 2812  174
##    1 1480  264
##
##                Accuracy : 0.6503
##                  95% CI : (0.6365, 0.6639)
##     No Information Rate : 0.9074
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1103
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6552
##             Specificity : 0.6027
##          Pos Pred Value : 0.9417
##          Neg Pred Value : 0.1514
##              Prevalence : 0.9074
##          Detection Rate : 0.5945
##    Detection Prevalence : 0.6313
##       Balanced Accuracy : 0.6290
##
##        'Positive' Class : 0
##
```

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



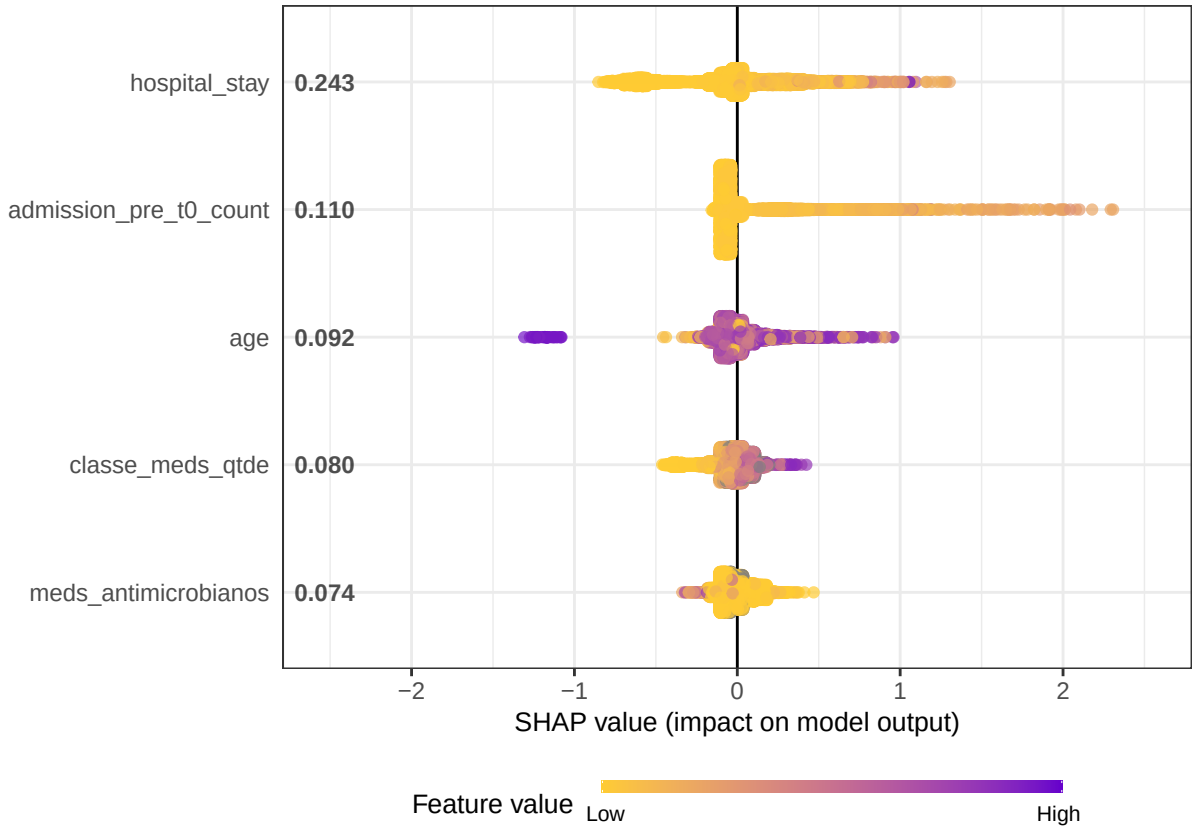

```
## [1] "Optimal Threshold: 0.50"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 2993  185
##    1 1299  253
##
##                Accuracy : 0.6863
##                  95% CI : (0.6728, 0.6995)
##     No Information Rate : 0.9074
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1284
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6973
##             Specificity : 0.5776
##          Pos Pred Value : 0.9418
##          Neg Pred Value : 0.1630
##              Prevalence : 0.9074
##          Detection Rate : 0.6328
##    Detection Prevalence : 0.6719
##       Balanced Accuracy : 0.6375
##
##        '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

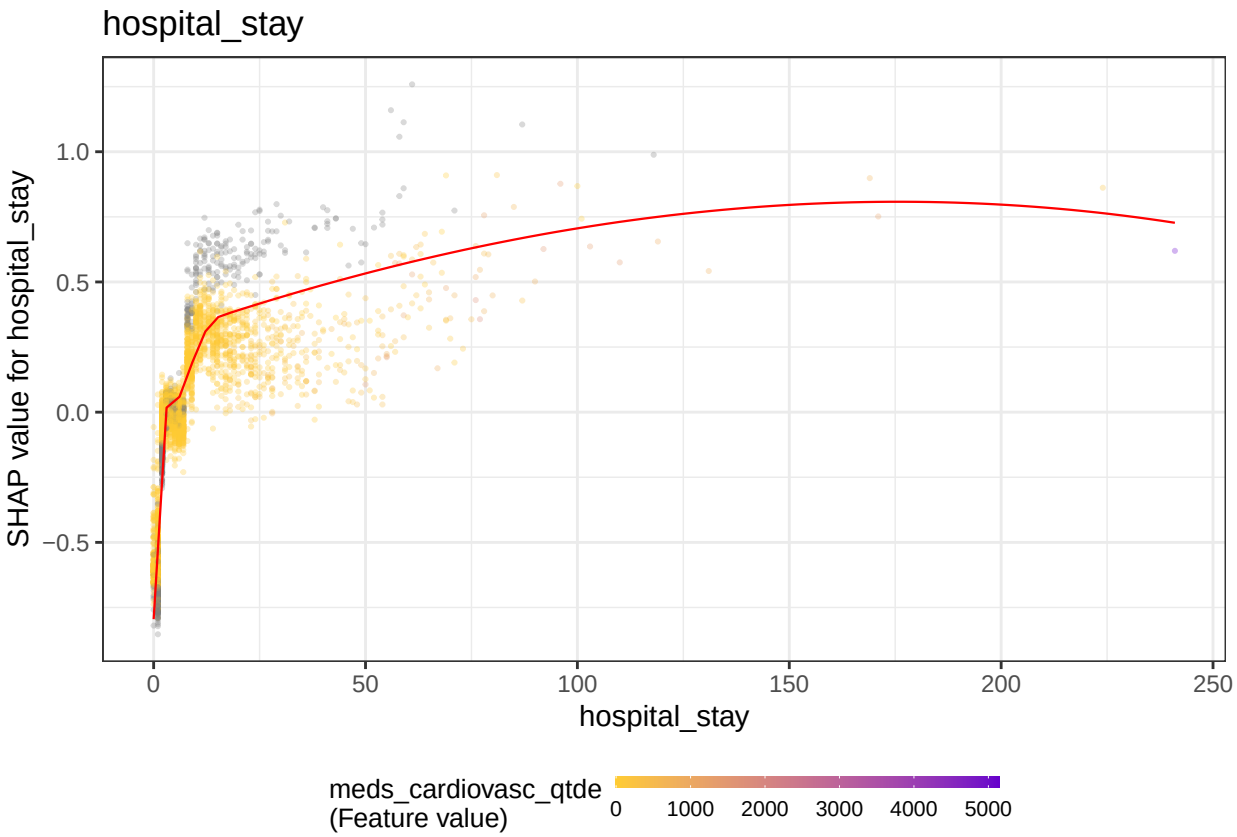
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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'
```



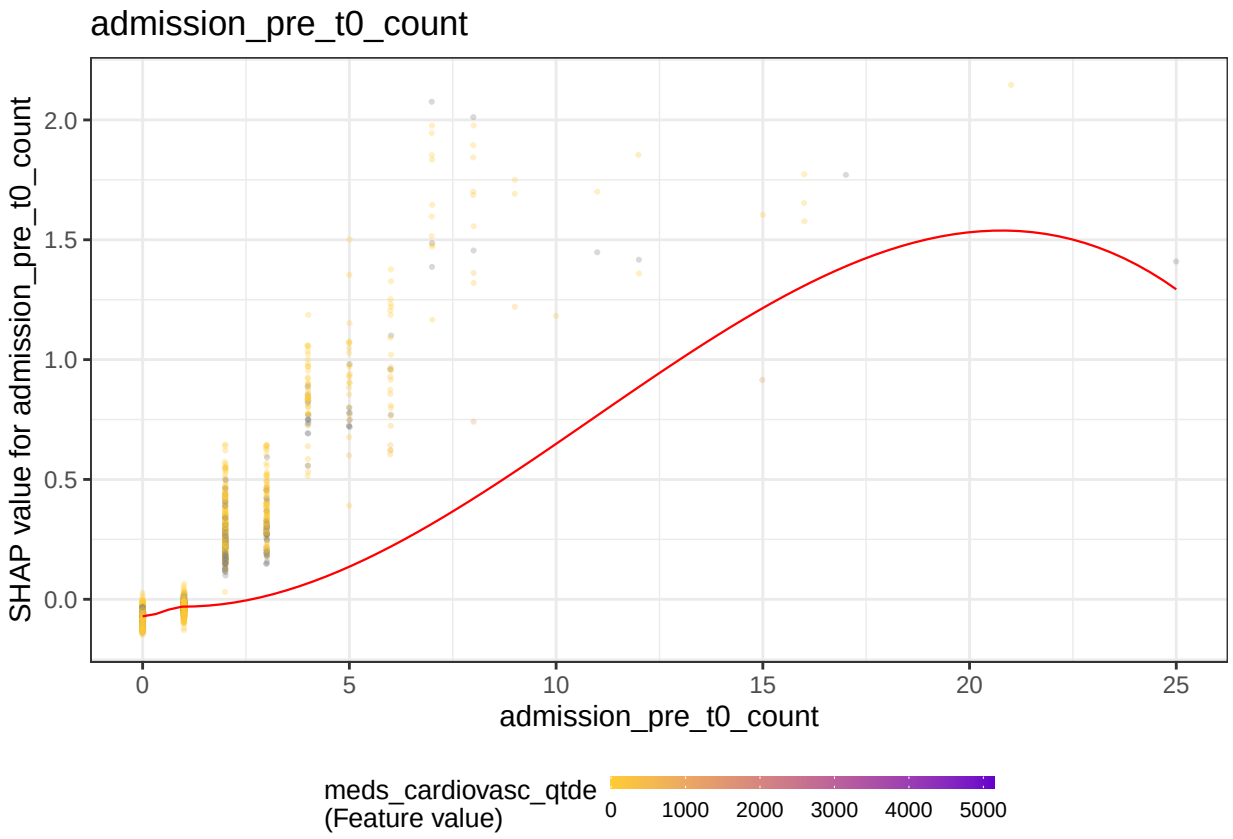

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -0.125
```

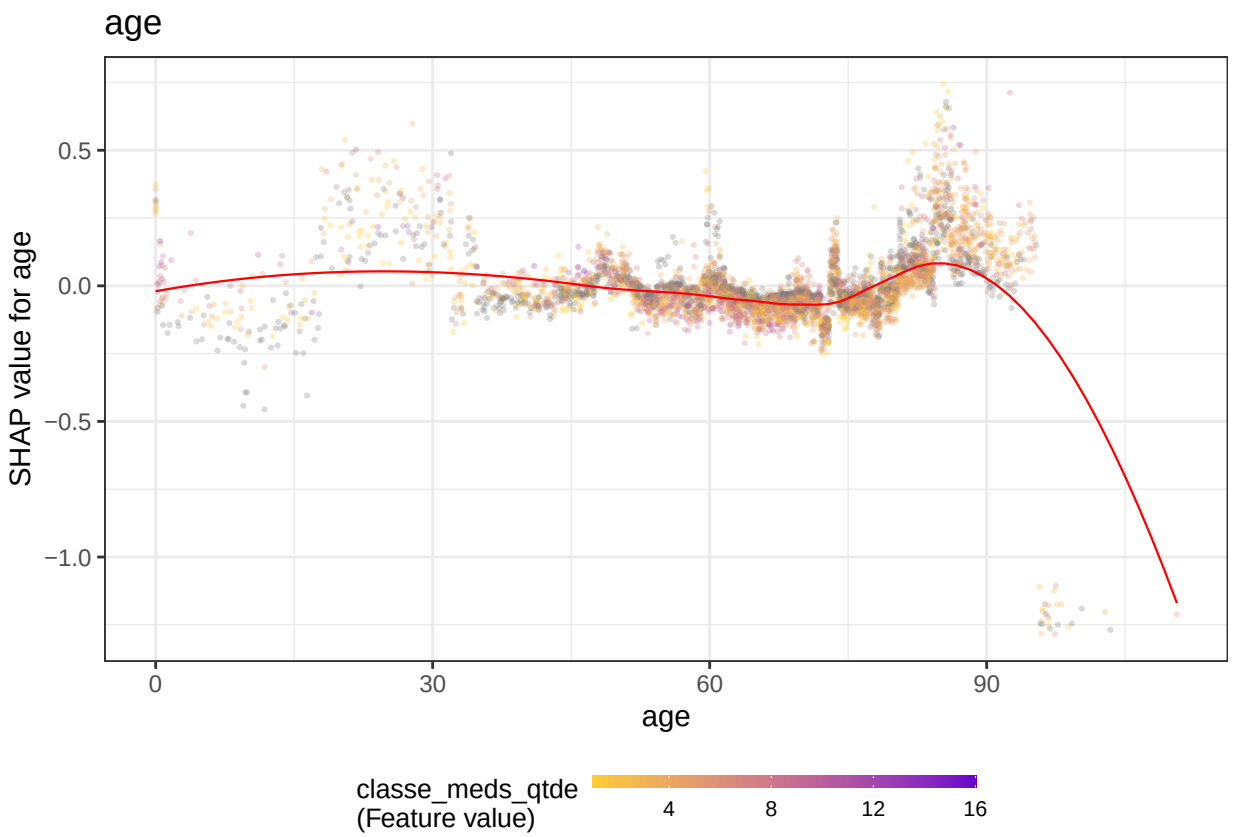
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.125
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 2.1864e-28
```

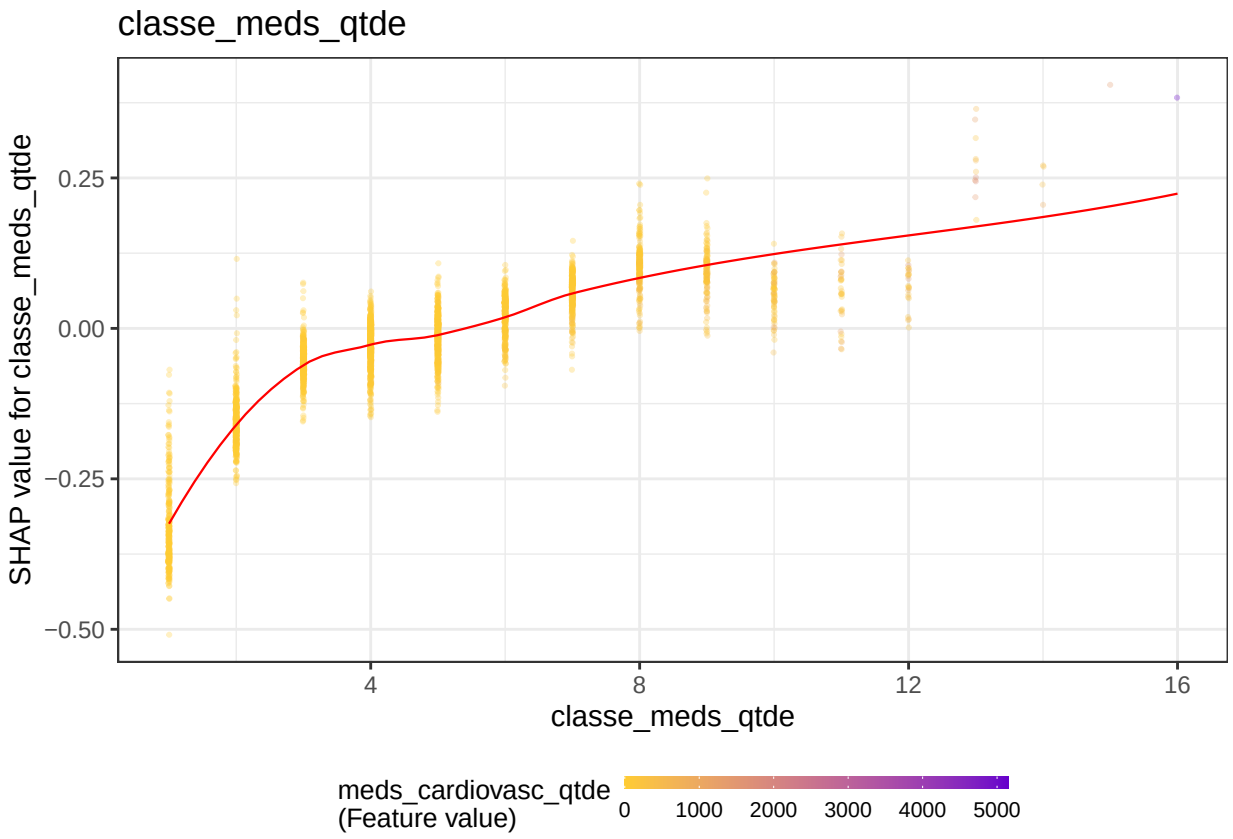
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are other near
## singularities as well. 1
```



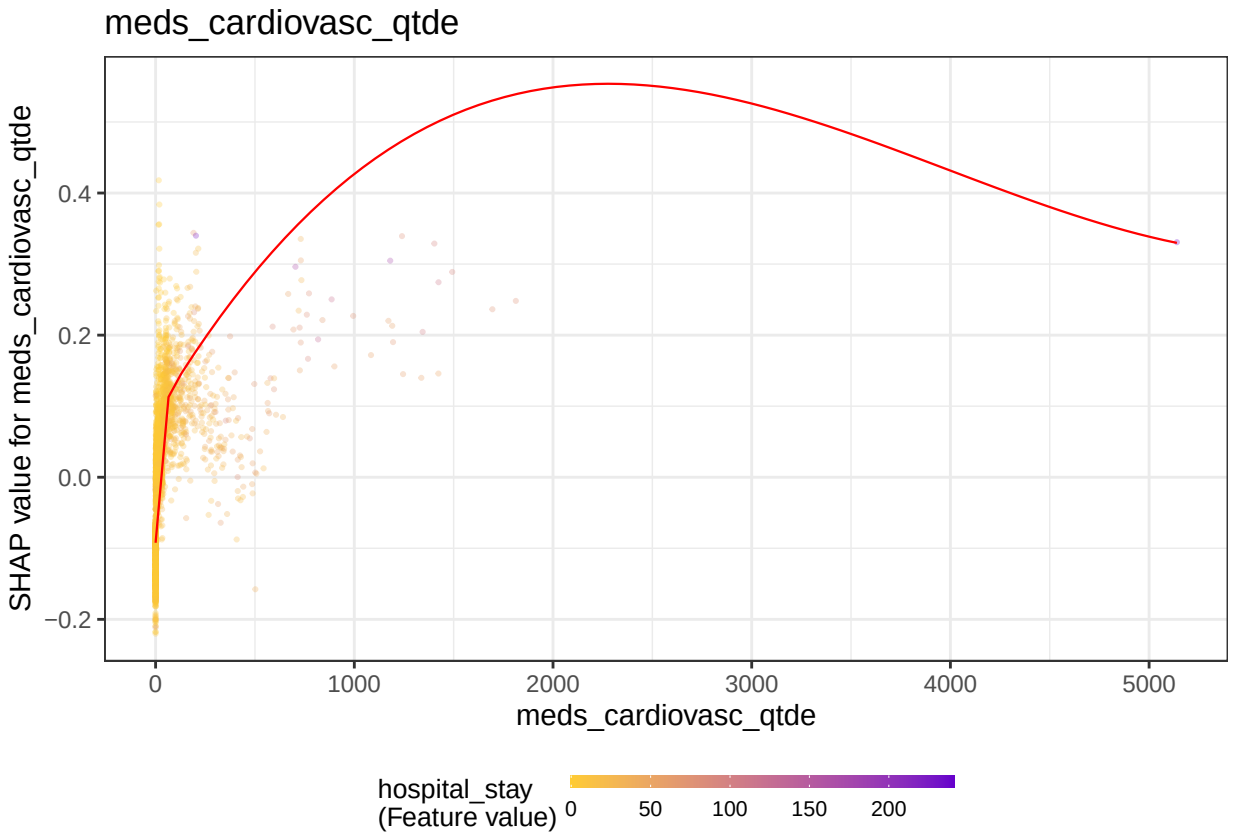

```
## `geom_smooth()` using formula 'y ~ x'
```




```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 1472 rows containing non-finite values (stat_smooth).
## Warning: Removed 1472 rows containing missing values (geom_point).
```

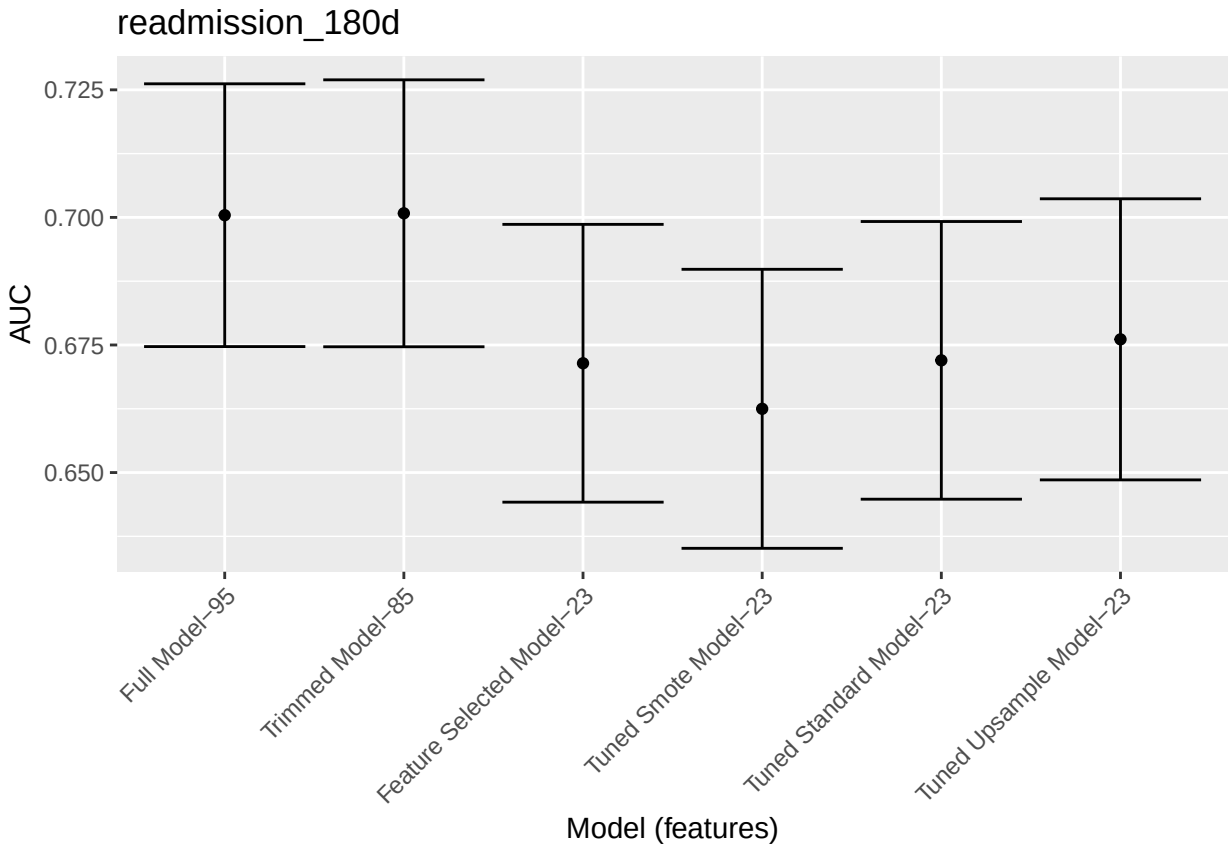



```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 1064 rows containing non-finite values (stat_smooth).
## Warning: Removed 1064 rows containing missing values (geom_point).
```



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_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))
```