Final Model - readmission_30d

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. education_level
2. underlying_heart_disease
3. heart_disease
4. nyha_basal
5. prior_mi
6. heart_failure
7. transplant
8. endocardites
9. hemodialysis
10. comorbidities_count
11. procedure_type_1
12. reop_type_1
13. procedure_type_new
14. cied_final_1
15. cied_final_group_1
16. admission_pre_t0_count
17. admission_pre_t0_180d
18. icu_t0
19. dialysis_t0
20. admission_t0_emergency
21. aco
22. antiarritmico
23. betabloqueador
24. ieca_bra

25. dva
26. digoxina
27. estatina
28. diuretico
29. vasodilatador
30. insuf_cardiaca
31. espironolactona
32. bloq_calcio
33. antiplaquetario_ev
34. insulina
35. anticonvulsivante
36. psicofarmacos
37. antifungico
38. antiviral
39. classe_meds_qtde
40. meds_cardiovasc_qtde
41. meds_antimicrobianos
42. ventilacao_mecanica
43. cec
44. transplante_cardiaco
45. outros_proced_cirurgicos
46. icp
47. intervencao_cv
48. cateterismo
49. eletrofisiologia
50. cateter_venoso_central
51. proced_invasivos_qtde
52. cve_desf
53. transfusao
54. equipe_multiprof
55. holter
56. metodos_graficos_qtde
57. laboratorio
58. cultura
59. analises_clinicas_qtde
60. citologia
61. biopsia
62. histopatologia_qtde
63. angio_rm
64. angio_tc
65. cintilografia
66. ecocardiograma
67. endoscopia
68. flebografia
69. pet_ct
70. ultrassom
71. tomografia
72. ressonancia
73. exames_imagem_qtde
74. bic
75. mpp
76. 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: 0.682"
```

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

```
## [1] "Full Model Test AUC: 0.698"
```

Features with zero importance on the initial model:

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

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

1. transplant
2. endocardites
3. hemodialysis
4. dialysis_t0
5. antiviral
6. transplante_cardiaco
7. icp
8. intervencao_cv
9. citologia
10. endoscopia
11. 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: 0.681"
```

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

```
## [1] "Trimmed Model Test AUC: 0.697"
```

```
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
76	0.6822	0.0068	0.0000	angio_rm
65	0.6807	0.0082	0.0015	education_level
64	0.6809	0.0080	0.0013	biopsia
63	0.6774	0.0064	0.0048	heart_disease
62	0.6807	0.0093	0.0015	nyha_basal
61	0.6791	0.0079	0.0031	angio_tc
60	0.6787	0.0069	0.0035	antiplaquetario_ev
59	0.6816	0.0076	0.0006	cec
58	0.6791	0.0070	0.0031	cateter_venoso_central
57	0.6807	0.0069	0.0015	cve_desf
56	0.6770	0.0075	0.0052	transfusao
55	0.6810	0.0081	0.0012	underlying_heart_disease
54	0.6806	0.0086	0.0016	insulina
53	0.6787	0.0072	0.0035	bloq_calcio
52	0.6775	0.0077	0.0047	eletrofisiologia
51	0.6781	0.0071	0.0041	flebografia
50	0.6805	0.0073	0.0017	cintilografia
49	0.6780	0.0063	0.0042	mpp
48	0.6796	0.0076	0.0026	ressonancia
47	0.6779	0.0075	0.0043	procedure_type_1
46	0.6813	0.0075	0.0009	cateterismo
45	0.6794	0.0071	0.0028	histopatologia_qtde
44	0.6798	0.0078	0.0024	outros_proced_cirurgicos
43	0.6773	0.0070	0.0049	antifungico
42	0.6830	0.0077	-0.0008	cied_final_1
41	0.6788	0.0079	0.0034	holter
40	0.6825	0.0080	-0.0003	cultura
39	0.6796	0.0079	0.0026	reop_type_1

Table 1: *(continued)*

Number of Features	CV AUC	CV AUC Std Error	AUC Loss	Least Important Feature
38	0.6784	0.0074	0.0038	prior_mi
37	0.6792	0.0059	0.0030	tomografia
36	0.6788	0.0067	0.0034	heart_failure
35	0.6787	0.0071	0.0035	betabloqueador
34	0.6782	0.0073	0.0040	ultrassom
33	0.6777	0.0065	0.0045	ecocardiograma
32	0.6795	0.0071	0.0027	aco
31	0.6773	0.0075	0.0049	ventilacao_mecanica
30	0.6773	0.0077	0.0049	procedure_type_new
29	0.6759	0.0062	0.0063	cied_final_group_1
28	0.6761	0.0056	0.0061	admission_t0_emergency
27	0.6735	0.0071	0.0087	admission_pre_t0_180d
26	0.6765	0.0065	0.0057	espironolactona
25	0.6762	0.0079	0.0060	proced_invasivos_qtde
24	0.6778	0.0072	0.0044	ieca_bra
23	0.6758	0.0083	0.0064	laboratorio
22	0.6752	0.0080	0.0070	dva
21	0.6717	0.0085	0.0105	insuf_cardiaca

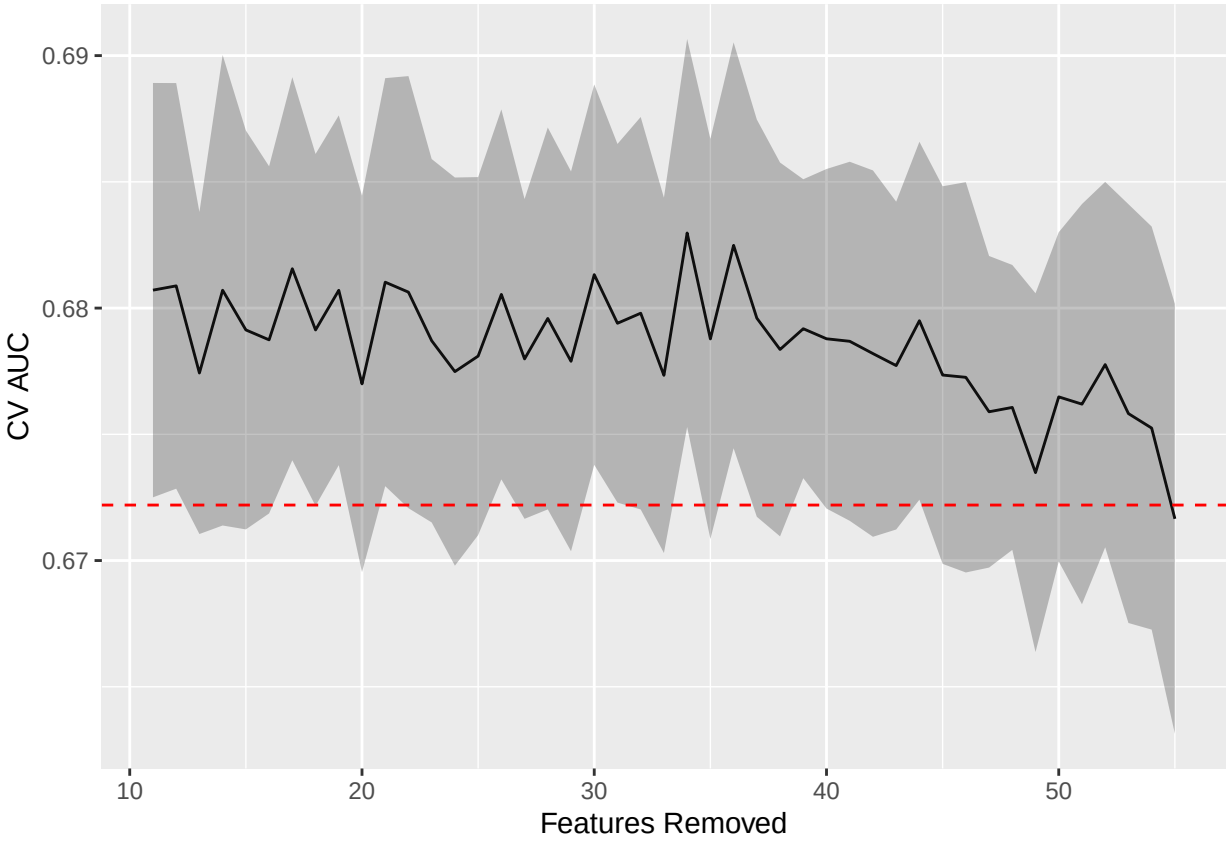
```
if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_droped)
} else {
  selected_features <- current_features
}

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: 0.673"
```

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

```
## [1] "Selected Model Test AUC: 0.688"
```

```
 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:

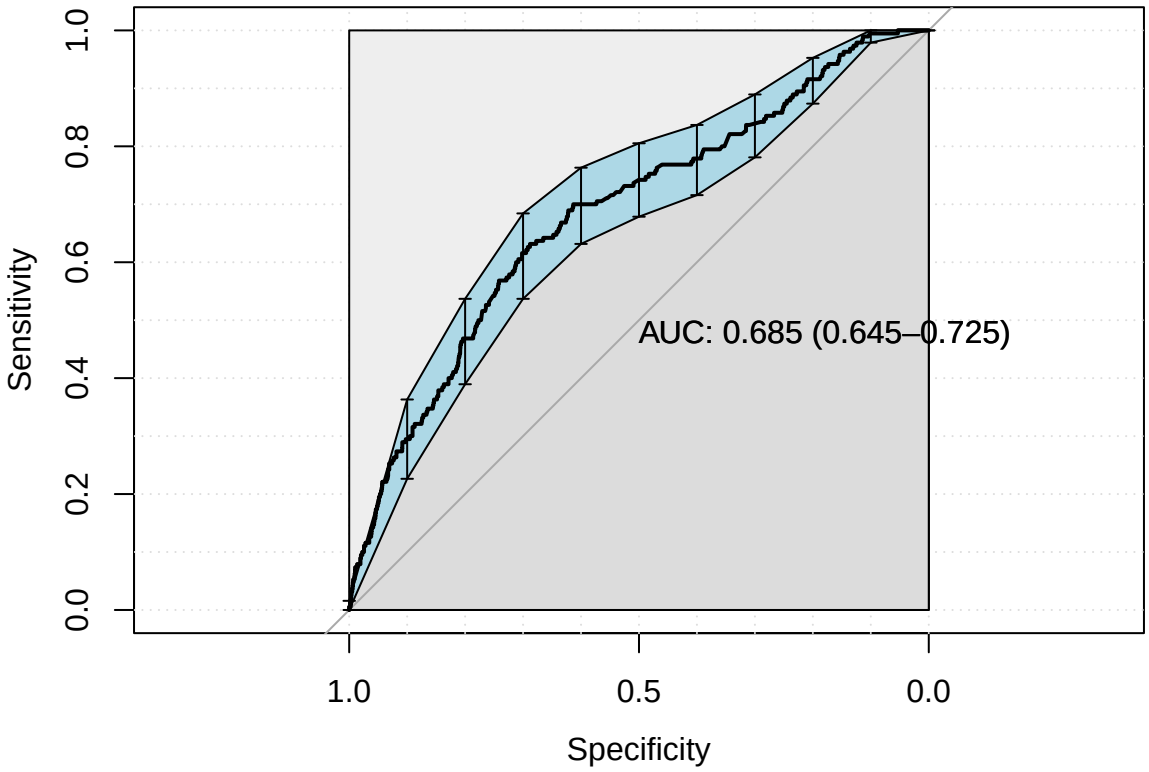
```
gluedown::md_order(selected_features, seq = TRUE, pad = TRUE)
```

1. hospital_stay
2. analises_clinicas_qtde
3. meds_antimicrobianos
4. vasodilatador
5. classe_meds_qtde
6. metodos_graficos_qtde
7. exames_imagem_qtde
8. icu_t0
9. meds_cardiovasc_qtde
10. admission_pre_t0_count
11. estatina
12. diuretico
13. bic
14. equipe_multiprof
15. psicofarmacos
16. anticonvulsivante
17. antiarritmico
18. comorbidities_count
19. insuf_cardiaca
20. digoxina
21. angio_rm

Standard

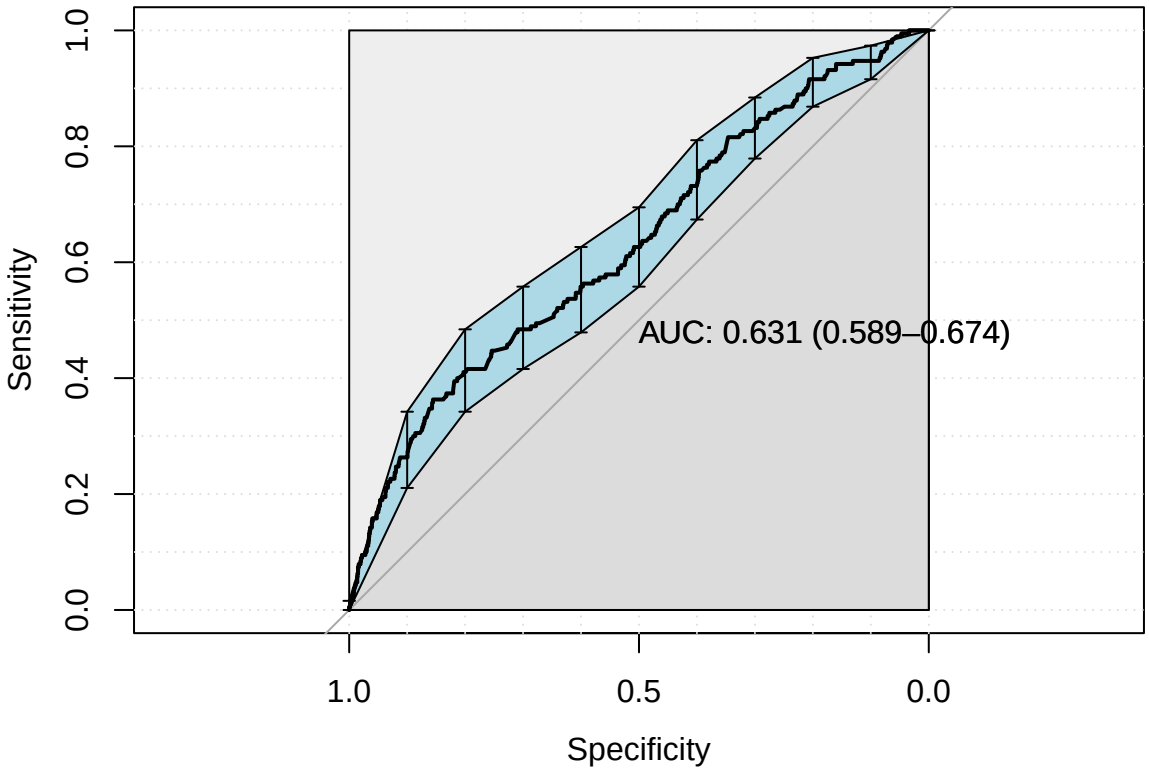
```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% select(all_of(c(selected_features, outcome_column)))) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(),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



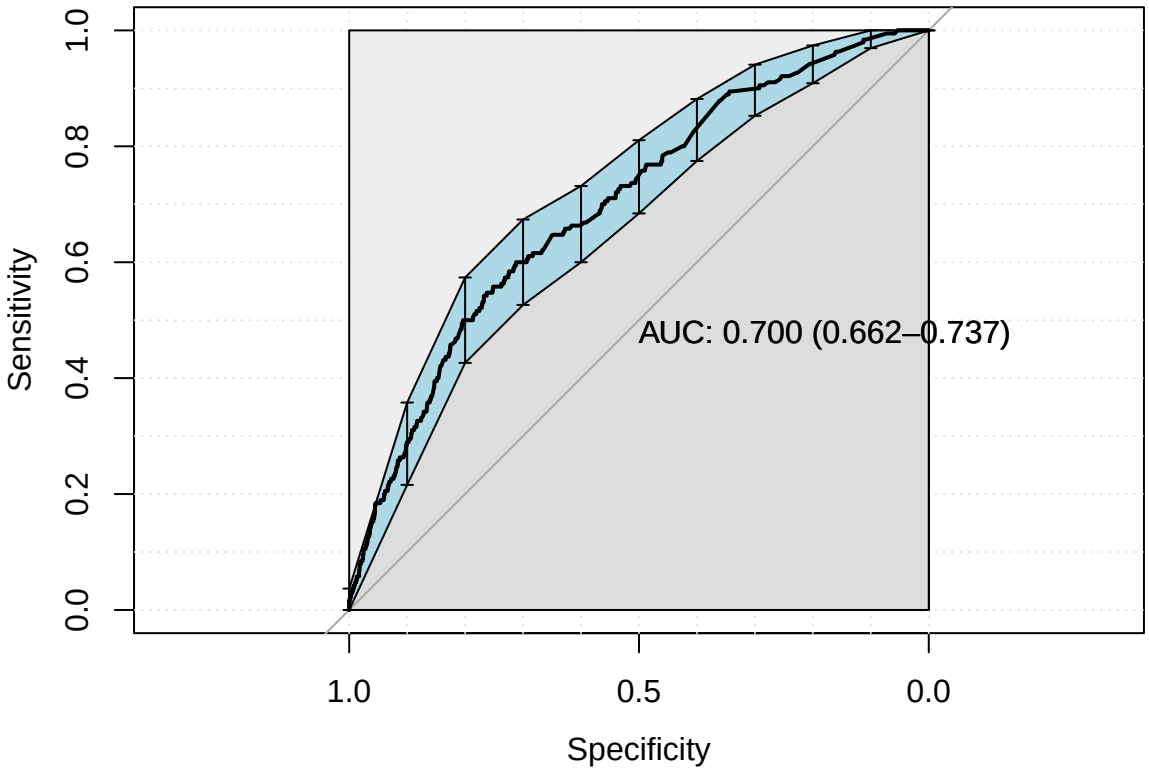

```
## [1] "Optimal Threshold: 0.04"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3124   70
##    1 1416  120
##
##                Accuracy : 0.6858
##                  95% CI : (0.6724, 0.6991)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0728
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.68811
##             Specificity : 0.63158
##          Pos Pred Value : 0.97808
##          Neg Pred Value : 0.07813
##              Prevalence : 0.95983
##          Detection Rate : 0.66047
##    Detection Prevalence : 0.67526
##       Balanced Accuracy : 0.65984
##
##        'Positive' Class : 0
##
```

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




```
## [1] "Optimal Threshold: 0.50"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3884  121
##    1  656   69
##
##                Accuracy : 0.8357
##                  95% CI : (0.8249, 0.8462)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0931
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.85551
##             Specificity : 0.36316
##          Pos Pred Value : 0.96979
##          Neg Pred Value : 0.09517
##              Prevalence : 0.95983
##          Detection Rate : 0.82114
##    Detection Prevalence : 0.84672
##       Balanced Accuracy : 0.60933
##
##        'Positive' Class : 0
##
```

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



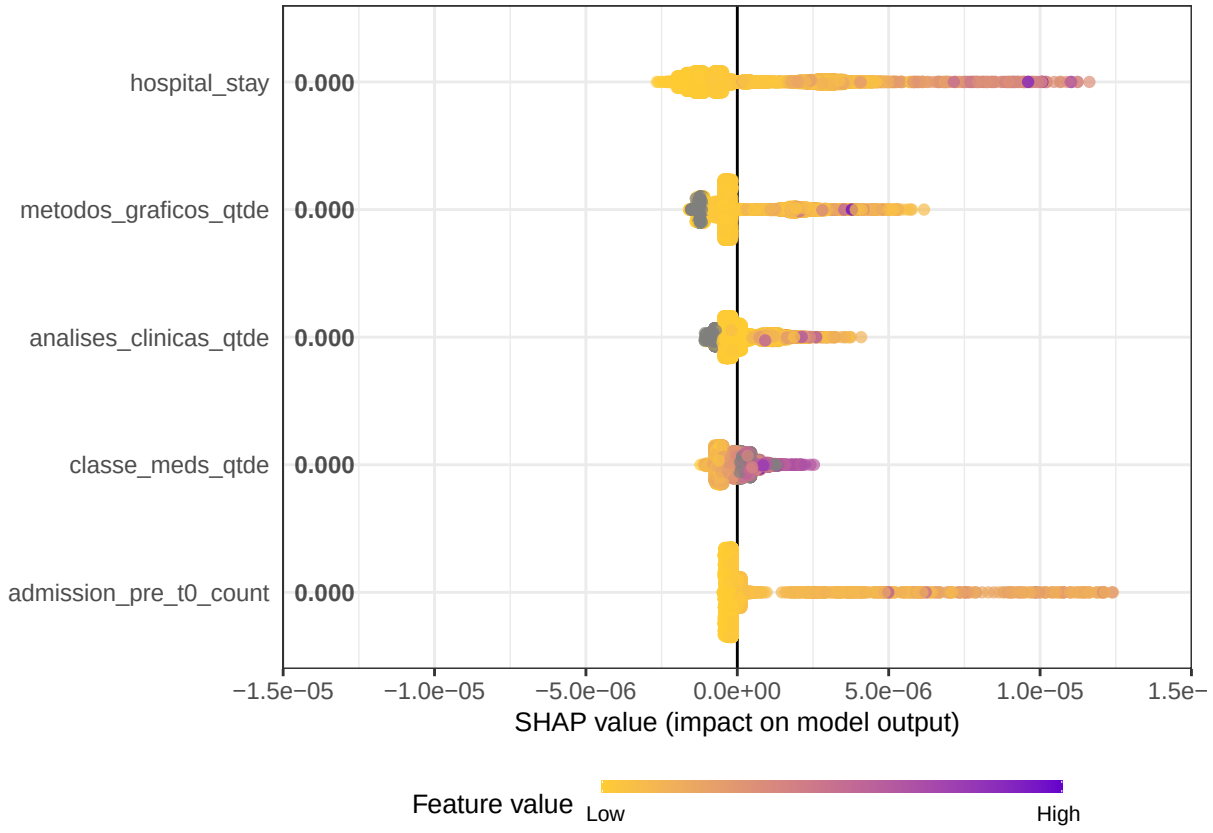

```
## [1] "Optimal Threshold: 0.49"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3233   76
##    1 1307  114
##
##                Accuracy : 0.7076
##                  95% CI : (0.6944, 0.7205)
##     No Information Rate : 0.9598
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0761
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71211
##             Specificity : 0.60000
##          Pos Pred Value : 0.97703
##          Neg Pred Value : 0.08023
##              Prevalence : 0.95983
##          Detection Rate : 0.68351
##    Detection Prevalence : 0.69958
##       Balanced Accuracy : 0.65606
##
##        '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

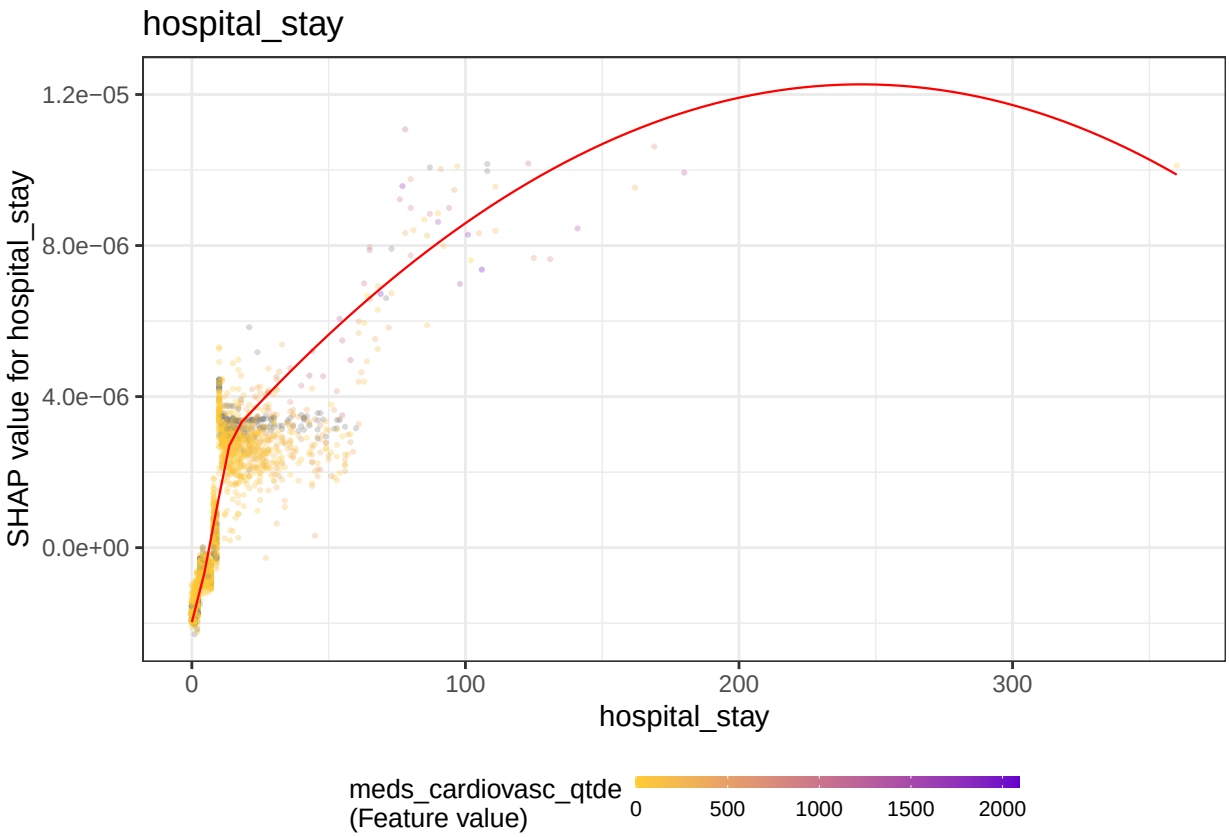
```
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

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)
```



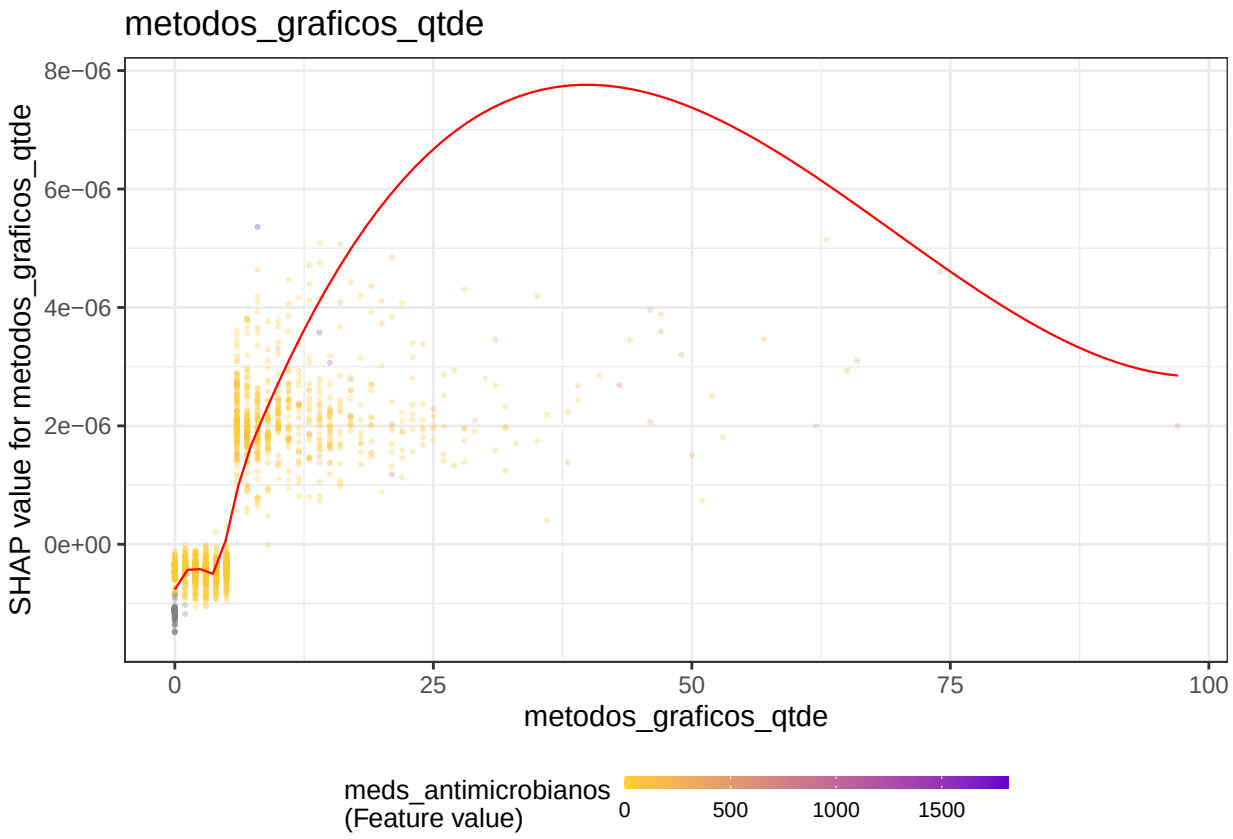

```
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

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'
```

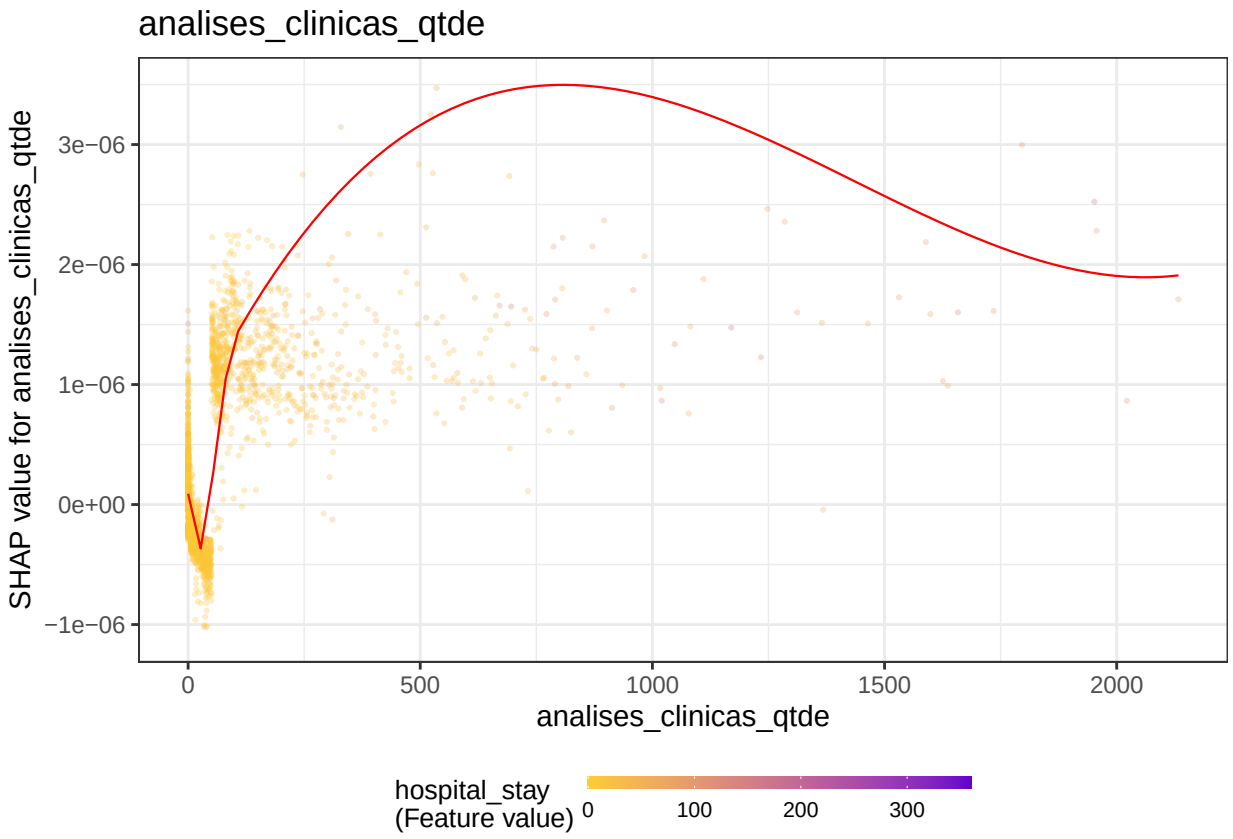



```
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 825 rows containing non-finite values (stat_smooth).
## Warning: Removed 825 rows containing missing values (geom_point).
```




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

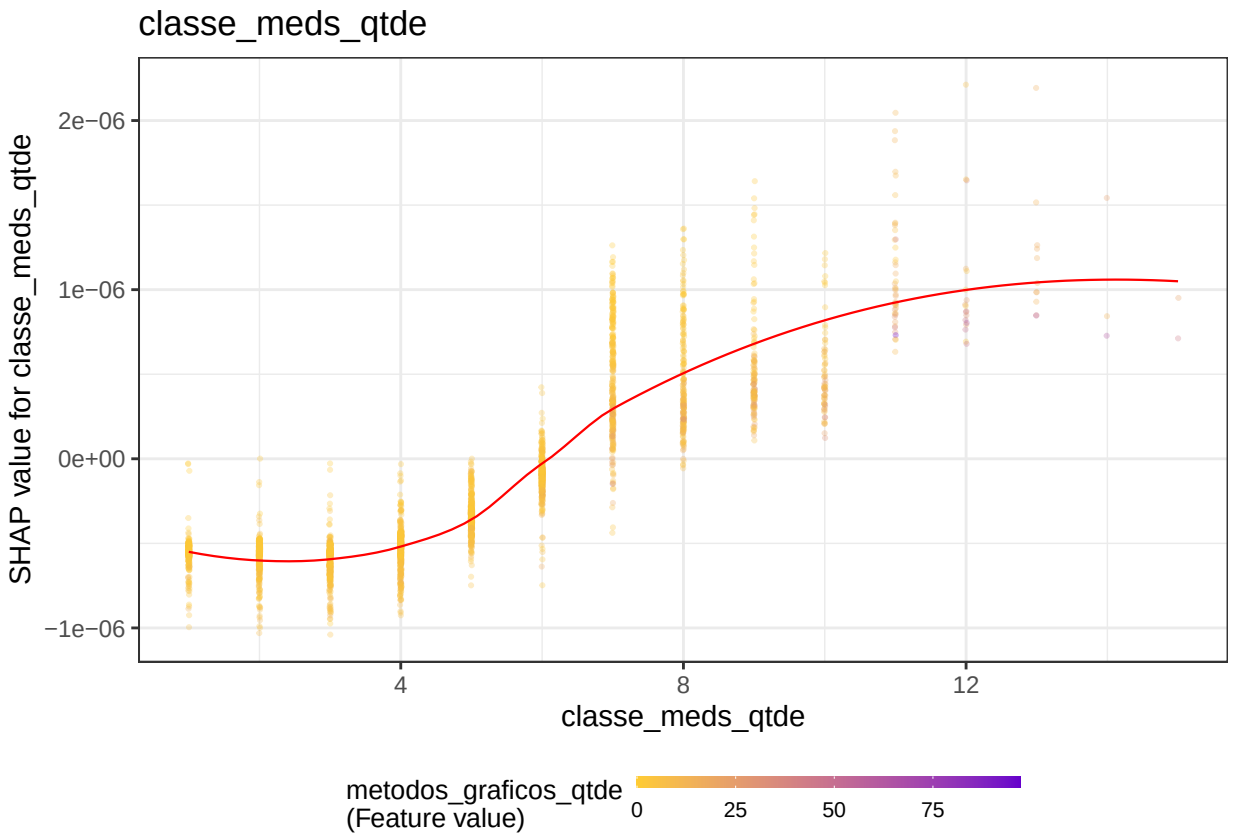
```
## Warning: Removed 825 rows containing non-finite values (stat_smooth).
## Removed 825 rows containing missing values (geom_point).
```




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

```
## Warning: Removed 1489 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1489 rows containing missing values (geom_point).
```



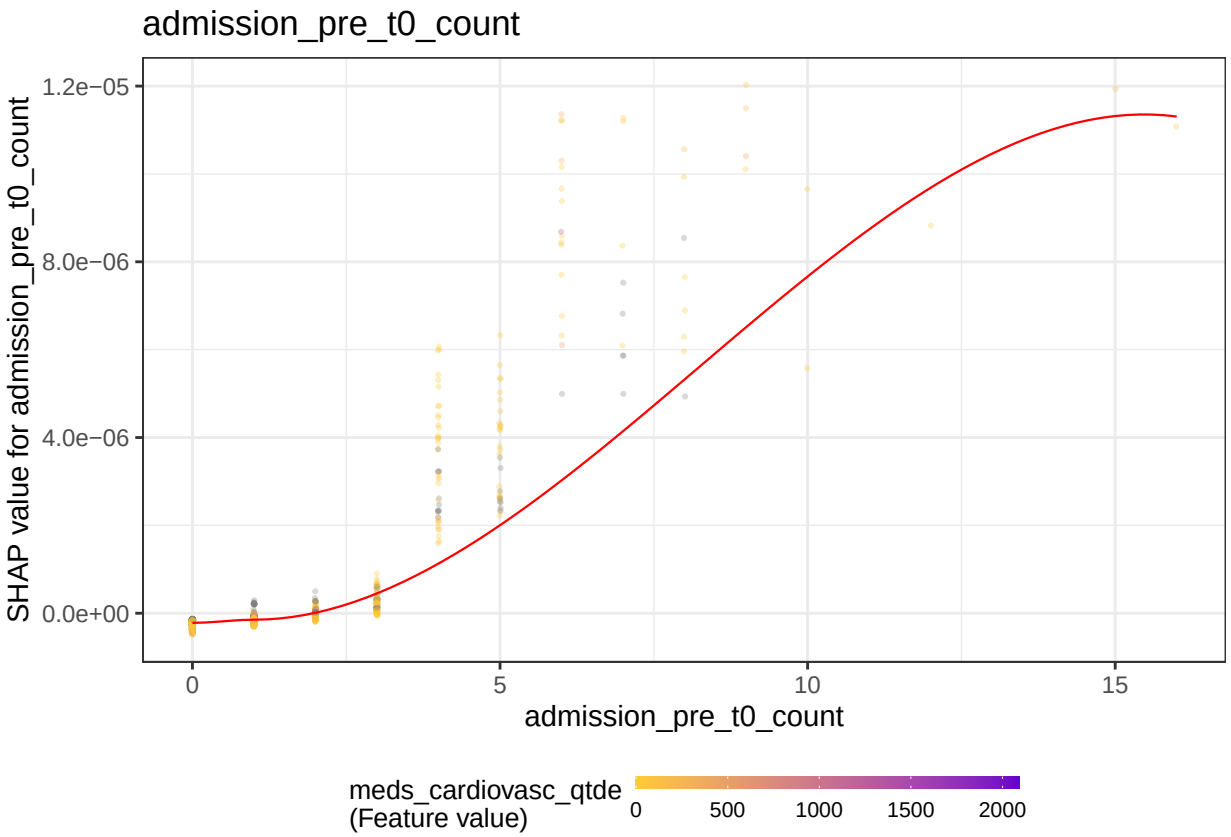

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse used at
## -0.08
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood radius
## 1.08
```

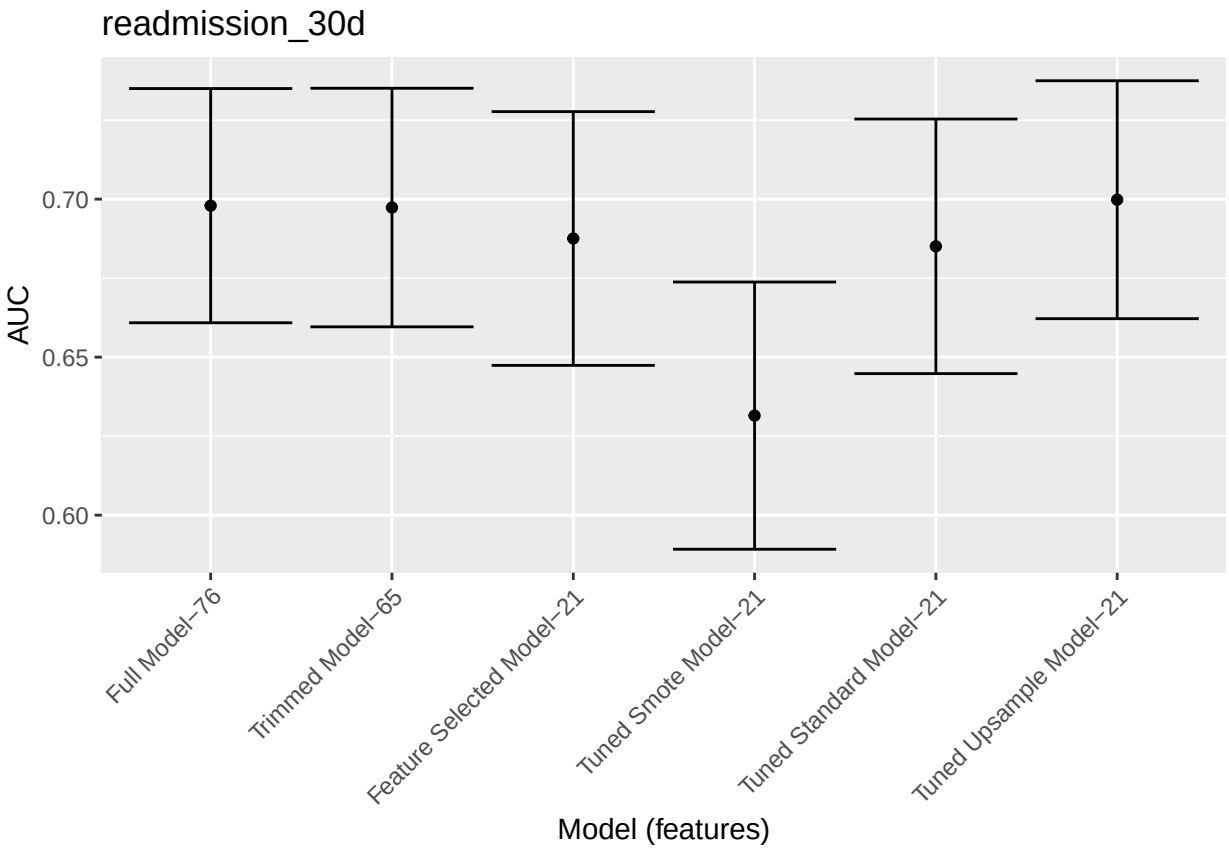
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal condition
## number 1.7723e-27
```

```
## 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))
```