Final Model - readmission_1year

Eduardo Yuki Yada

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)
```

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))
```

Eligible features

```
eligible_columns = df_names %>%
  filter(momento.aquisicao == 'Admissão t0') %>%
  .$variable.name

exception_columns = c('death_intraop', 'death_intraop_1')

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
                       )

eligible_features = eligible_columns %>%
  base::intersect(c(columns_list$categorical_columns, columns_list$numerical_columns)) %>%
  setdiff(c(exception_columns, correlated_columns))
```

```r
if (is.null(features_list)) {
  features = eligible_features
} else {
  features = base::intersect(eligible_features, features_list)
}
```

Starting features:

```r
gluedown::md_order(features, seq = TRUE, pad = TRUE)
```

1. sex
2. age
3. race
4. education_level
5. patient_state
6. underlying_heart_disease
7. heart_disease
8. nyha_basal
9. prior_mi
10. heart_failure
11. af
12. cardiac_arrest
13. transplant
14. valvopathy
15. endocardites
16. diabetes
17. renal_failure
18. hemodialysis
19. copd
20. comorbidities_count
21. procedure_type_1
22. reop_type_1
23. procedure_type_new
24. cied_final_1
25. cied_final_group_1
26. admission_pre_t0_count
27. admission_pre_t0_180d
28. year_adm_t0
29. icu_t0
30. dialysis_t0
31. admission_t0_emergency
32. aco
33. antiarritmico
34. betabloqueador
35. ieca_bra
36. dva
37. digoxina
38. estatina
39. diuretico
40. vasodilatador
41. insuf_cardiaca
42. espironolactona
43. bloq_calcio
44. antiplaquetario_ev
45. insulina
46. anticonvulsivante
47. psicofarmacos
48. antifungico
49. antiviral
50. antiretroviral
51. classe_meds_qtde

52. meds_cardiovasc_qtde
53. meds_antimicrobianos
54. cec
55. transplante_cardiaco
56. cir_toracica
57. outros_proced_cirurgicos
58. icp
59. intervencao_cv
60. angioplastia
61. cateterismo
62. eletrofisiologia
63. cateter_venoso_central
64. proced_invasivos_qtde
65. cve_desf
66. transfusao
67. interconsulta
68. equipe_multiprof
69. ecg
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. aortografia
84. arteriografia
85. cintilografia
86. ecocardiograma
87. endoscopia
88. flebografia
89. pet_ct
90. ultrassom
91. tomografia
92. radiografia
93. ressonancia
94. exames_imagem_qtde
95. bic
96. mpp

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) %>% dplyr::select(all_of(c(features, outcome_column)))
df_test <- testing(df_split) %>% dplyr::select(all_of(c(features, outcome_column)))
```

Global parameters

```
k <- 4 # Number of folds for cross validation
grid_size <- 50 # Number of parameter combination to tune on each model

set.seed(234)
df_folds <- vfold_cv(df_train, v = k,
                     strata = all_of(outcome_column))

max_auc_loss <- 0.01
```

Functions

```
niceFormatting = function(df, caption="", digits = 2, font_size = NULL){
  df %>%
    kbl(booktabs = T, longtable = T, caption = caption, digits = digits, format = "latex") %>%
    kable_styling(font_size = font_size,
                  latex_options = c("striped", "HOLD_position", "repeat_header"))
}
```

```
validation = function(model_fit, new_data, plot=TRUE) {
  library(pROC)
  library(caret)

  test_predictions_prob <-
    predict(model_fit, new_data = new_data, type = "prob") %>%
    rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
    .$`1`

  pROC_obj <- roc(
    new_data[[outcome_column]],
    test_predictions_prob,
    direction = "<",
    levels = c(0, 1),
    smoothed = TRUE,
    ci = TRUE,
    ci.alpha = 0.9,
    stratified = FALSE,
    plot = plot,
    auc.polygon = TRUE,
    max.auc.polygon = TRUE,
    grid = TRUE,
    print.auc = TRUE,
    show.thres = TRUE
  )

  test_predictions_class <-
    predict(model_fit, new_data = new_data, type = "class") %>%
    rename_at(vars(starts_with(".pred_")), ~ str_remove(., ".pred_")) %>%
    .$class

  conf_matrix <- table(test_predictions_class, new_data[[outcome_column]])

  if (plot) {
    sens.ci <- ci.se(pROC_obj)
    plot(sens.ci, type = "shape", col = "lightblue")
    plot(sens.ci, type = "bars")

    confusionMatrix(conf_matrix) %>% print
  }
```

```
  return(pROC_obj)
}
```

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") %>%
    step_impute_mean(all_numeric_predictors()) %>%
    step_zv(all_predictors())

  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)

  return(list(cv_auc = collect_metrics(model_fit_rs) %>% filter(.metric == 'roc_auc') %>% .$mean,
              importance = feature_importance,
              auc = as.numeric(model_auc$auc),
              auc_lower = model_auc$ci[1],
              auc_upper = model_auc$ci[3]))
}
```

```
hyperparameters <- readRDS(
  sprintf(
    "../EDA/auxiliar/model_selection/hyperparameters/lightgbm_%s.rds",
    outcome_column
  )
)

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: 0.711"
```

Features with zero importance on the initial model:

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

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

1. transplant
2. copd
3. dialysis_t0
4. antiplaquetario_ev
5. antiretroviral
6. cec
7. cir_toracica
8. intervencao_cv
9. angioplastia
10. teste_esforco
11. tilt_teste
12. citologia
13. angio_rm
14. aortografia
15. arteriografia
16. 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: 0.721"
```

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

```
## [1] "Trimmed Model Test AUC: 0.710"
```

```
selection_results <- tibble::tribble(
  ~`Number of Features`, ~`AUC Loss`, ~`Least Important Feature`,
  length(features), 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),
            `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),
            `AUC Loss` = current_auc_loss,
            `Least Important Feature` = current_least_important)

  # print(c(length(current_features), current_auc_loss))
}

selection_results %>% niceFormatting(digits = 4)
```

Table 1:

Number of Features	AUC Loss	Least Important Feature
96	0.0000	heart_disease
80	-0.0008	anticonvulsivante
79	-0.0010	mpp
78	-0.0003	hemodialysis
77	-0.0002	eletrofisiologia
76	-0.0007	heart_disease
75	-0.0007	renal_failure
74	-0.0005	cve_desf
73	-0.0010	antifungico
72	-0.0013	espiro_ergoespiro
71	-0.0007	icp
70	-0.0015	af
69	-0.0008	outros_proced_cirurgicos
68	-0.0003	sex
67	0.0000	tomografia
66	-0.0004	analises_clinicas_qtde
65	0.0000	procedure_type_new
64	-0.0001	angio_tc
63	-0.0002	diabetes
62	-0.0001	ressonancia
61	-0.0005	endocardites
60	-0.0003	cateterismo
59	0.0000	flebografia
58	0.0001	holter
57	-0.0004	heart_failure
56	-0.0003	cintilografia
55	-0.0003	race
54	-0.0004	insulina
53	-0.0003	cateter_venoso_central
52	0.0000	endoscopia
51	-0.0001	cultura
50	-0.0003	biopsia
49	-0.0003	cied_final_group_1
48	-0.0001	bic
47	-0.0006	aco
46	-0.0008	prior_mi
45	-0.0009	ultrassom
44	-0.0014	cardiac_arrest
43	-0.0012	underlying_heart_disease

Table 1: *(continued)*

Number of Features	AUC Loss	Least Important Feature
42	-0.0022	valvopathy
41	-0.0016	exames_imagem_qtde
40	-0.0017	interconsulta
39	-0.0015	betabloqueador
38	-0.0017	proced_invasivos_qtde
37	-0.0020	nyha_basal
36	-0.0005	comorbidities_count
35	-0.0001	radiografia
34	-0.0008	insuf_cardiaca
33	-0.0007	bloq_calcio
32	-0.0011	ecocardiograma
31	-0.0015	antiviral
30	-0.0013	transplante_cardiaco
29	-0.0012	ecg
28	-0.0012	psicofarmacos
27	-0.0017	diuretico
26	-0.0012	laboratorio
25	-0.0028	admission_t0_emergency
24	-0.0025	transfusao
23	-0.0026	ieca_bra
22	-0.0014	education_level
21	-0.0018	histopatologia_qtde
20	-0.0013	digoxina
19	-0.0012	patient_state
18	0.0015	estatina
17	0.0009	dva
16	0.0014	equipe_multiprof
15	0.0022	procedure_type_1
14	0.0022	meds_antimicrobianos
13	0.0035	espironolactona
12	0.0038	admission_pre_t0_180d
11	0.0041	antiarritmico
10	0.0047	vasodilatador
9	0.0048	cied_final_1
8	0.0114	icu_t0

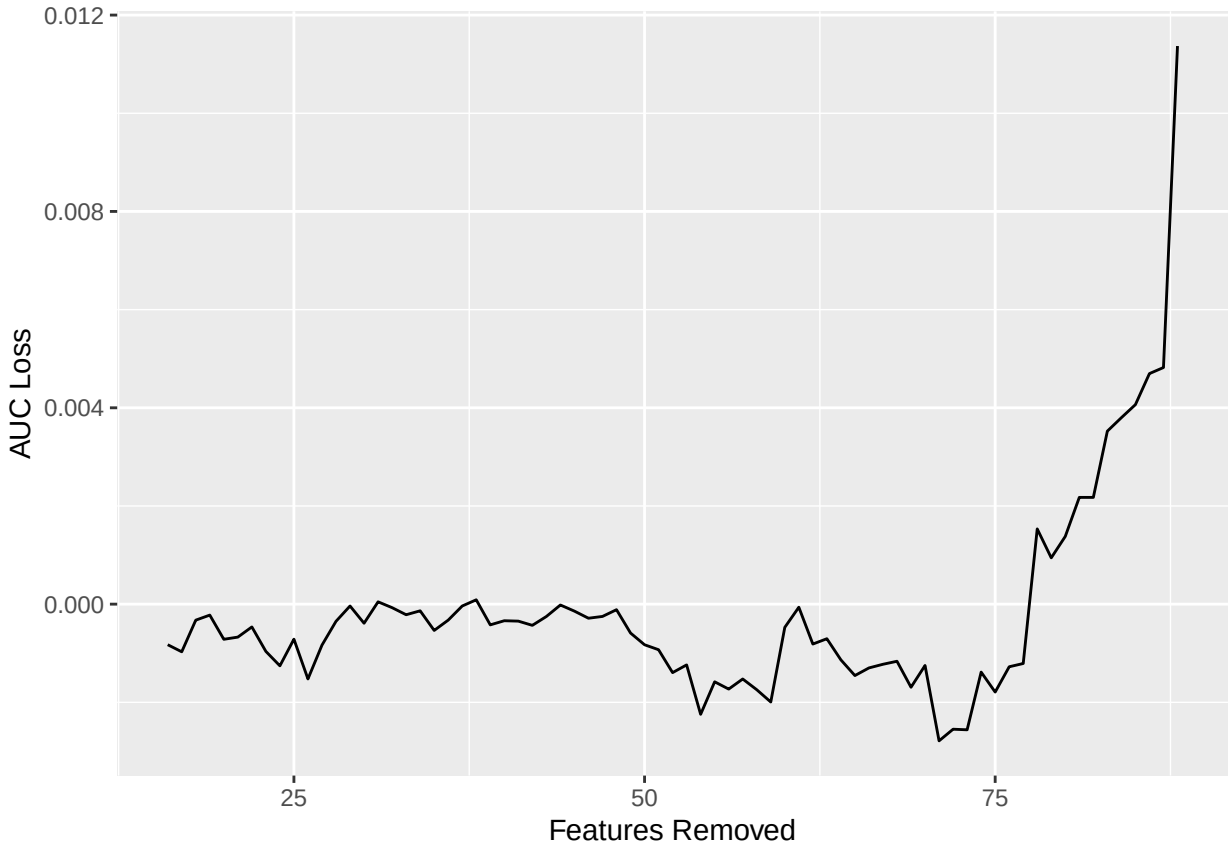
```
if (exists('last_feature_dropped')) {
  selected_features <- c(current_features, last_feature_droped)
} else {
  selected_features <- current_features
}

con <- file(sprintf('../EDA/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('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)
```

```
## [1] "Trimmed Model CV Train AUC: 0.709"
```

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

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

```
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. admission_pre_t0_count
2. meds_cardiovasc_qtde
3. metodos_graficos_qtde
4. icu_t0
5. year_adm_t0
6. classe_meds_qtde
7. reop_type_1
8. age

```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% dplyr::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(), one_hot = TRUE) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_zv(all_predictors())
```

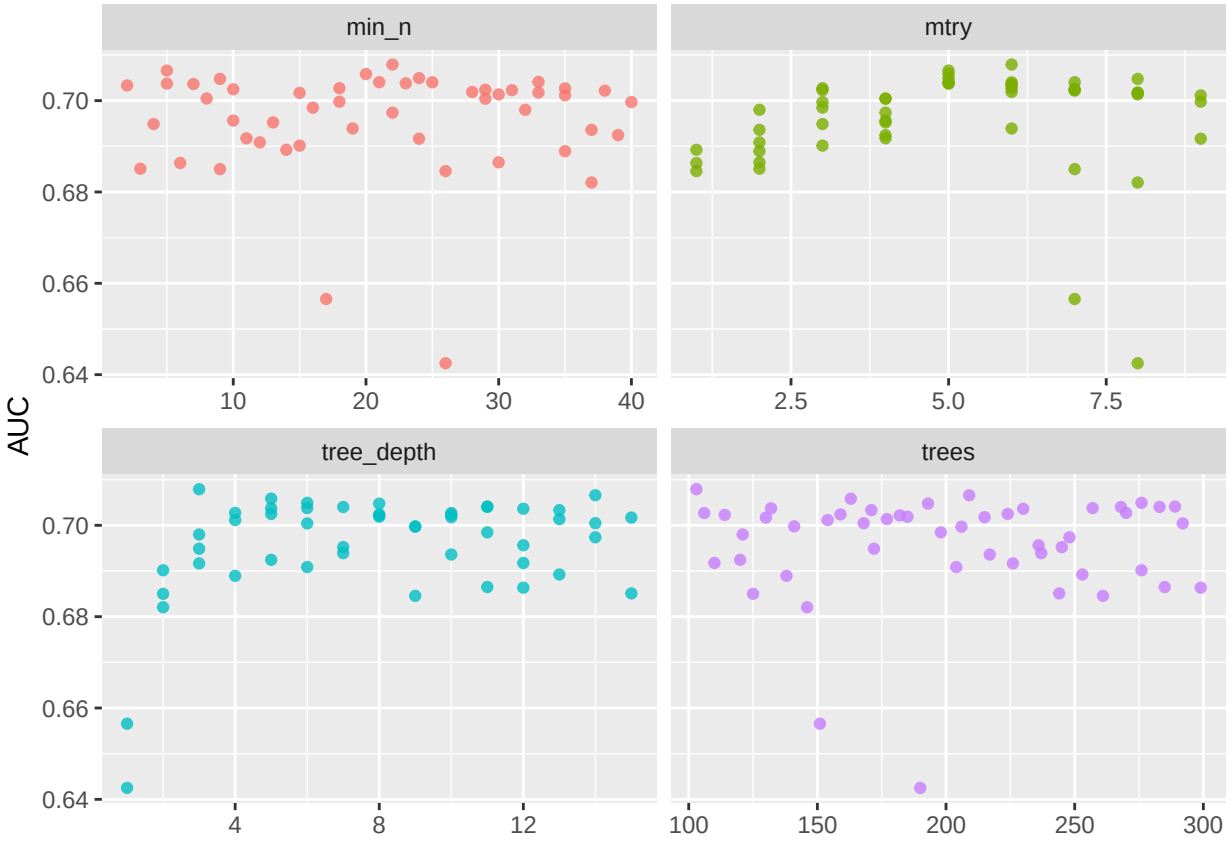
```
lightgbm_spec <- boost_tree(
  mtry = tune(),
  trees = tune(),
  min_n = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  loss_reduction = tune()
) %>%
  set_engine("lightgbm") %>%
  set_mode("classification")

lightgbm_grid <- grid_latin_hypercube(
  finalize(mtry(),
           df_train %>% dplyr::select(all_of(c(selected_features, outcome_column)))),
  dials::trees(range = c(100L, 300L)),
  min_n(),
  tree_depth(),
  learn_rate(),
  loss_reduction(),
  size = grid_size
)

lightgbm_workflow <-
  workflow() %>%
  add_recipe(lightgbm_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)
```

Table 2:

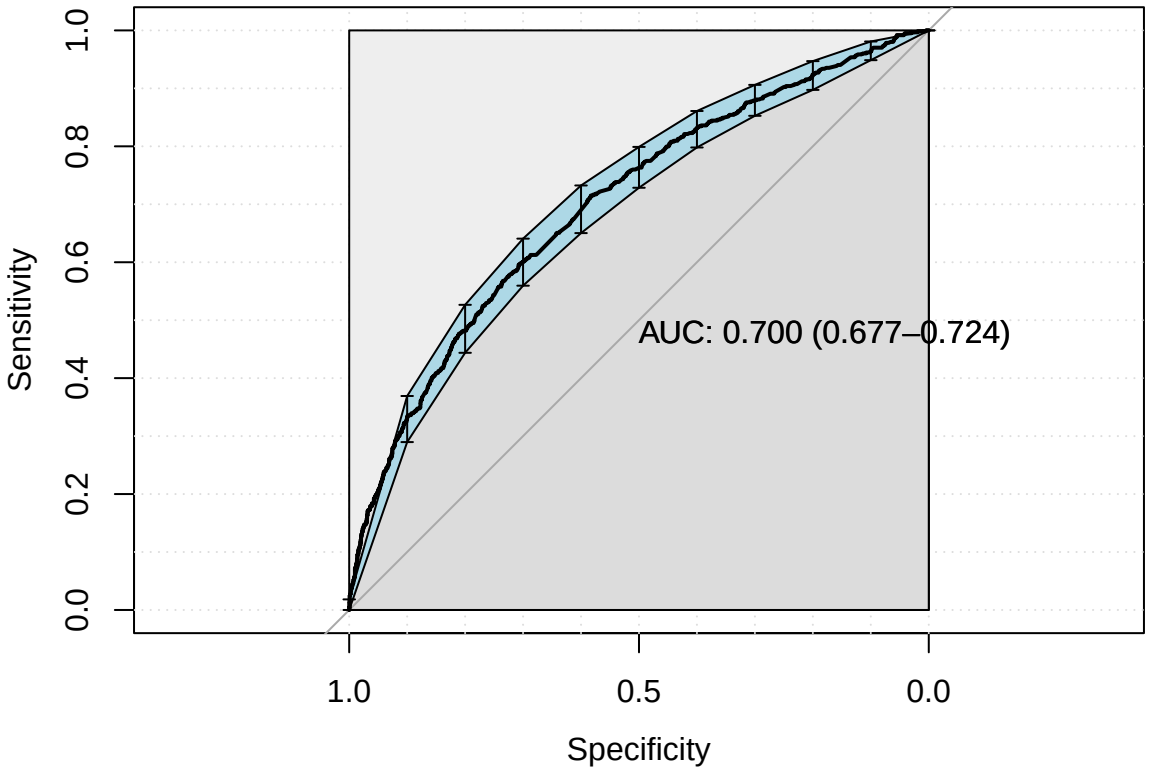
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
6	103	22	3	0.03124	0	roc_auc	binary	0.70790	4	0.00524	Preprocessor1_
5	209	5	14	0.01292	0	roc_auc	binary	0.70659	4	0.00265	Preprocessor1_
5	163	20	5	0.00265	0	roc_auc	binary	0.70583	4	0.00548	Preprocessor1_
5	276	24	6	0.00019	0	roc_auc	binary	0.70493	4	0.00500	Preprocessor1_
8	193	9	8	0.00497	0	roc_auc	binary	0.70476	4	0.00534	Preprocessor1_

```
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)
```



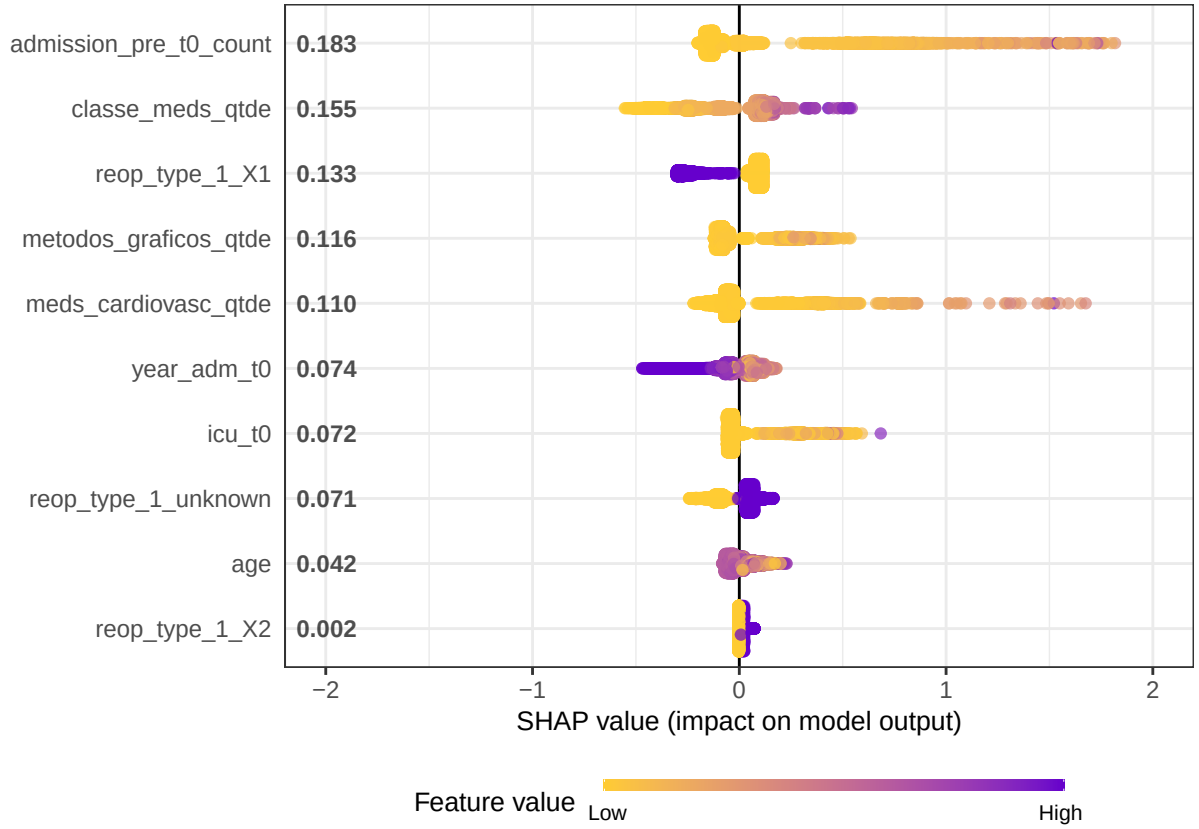

```
## Confusion Matrix and Statistics
##
##
## test_predictions_class    0    1
##                       0 4118  587
##                       1    9   17
##
##                Accuracy : 0.874
##                  95% CI : (0.8642, 0.8834)
##     No Information Rate : 0.8723
##     P-Value [Acc > NIR] : 0.3738
##
##                   Kappa : 0.0439
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99782
##             Specificity : 0.02815
##          Pos Pred Value : 0.87524
##          Neg Pred Value : 0.65385
##              Prevalence : 0.87233
##          Detection Rate : 0.87043
##    Detection Prevalence : 0.99450
##       Balanced Accuracy : 0.51298
##
##        'Positive' Class : 0
##
```

```
lightgbm_parameters <- lightgbm_tune %>%
  show_best("roc_auc", n = 1) %>%
  select(trees, mtry, min_n, tree_depth, learn_rate, loss_reduction) %>%
  as.list

saveRDS(
```

```
  lightgbm_parameters,
  file = sprintf(
    "../EDA/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)))

shap.plot.summary.wrap1(model = lightgbm_model, X = matrix_train, top_n = 10, dilute = F)
```




```
# Crunch SHAP values
shap <- shap.prep(lightgbm_model, X_train = matrix_test)

for (x in shap.importance(shap, names_only = TRUE)[1:5]) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = TRUE,
    jitter_width = 0.01,
    alpha = 0.3
```

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

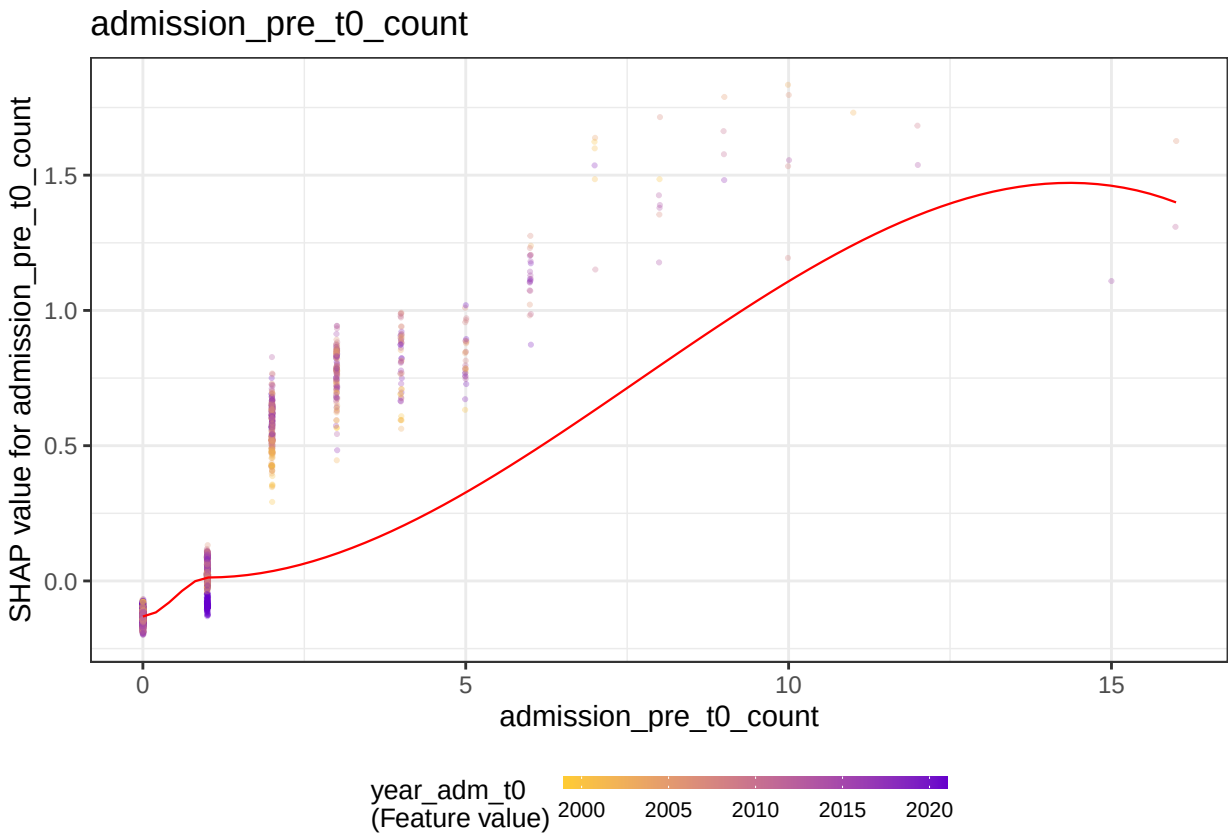
```
## `geom_smooth()` using formula 'y ~ x'
```

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

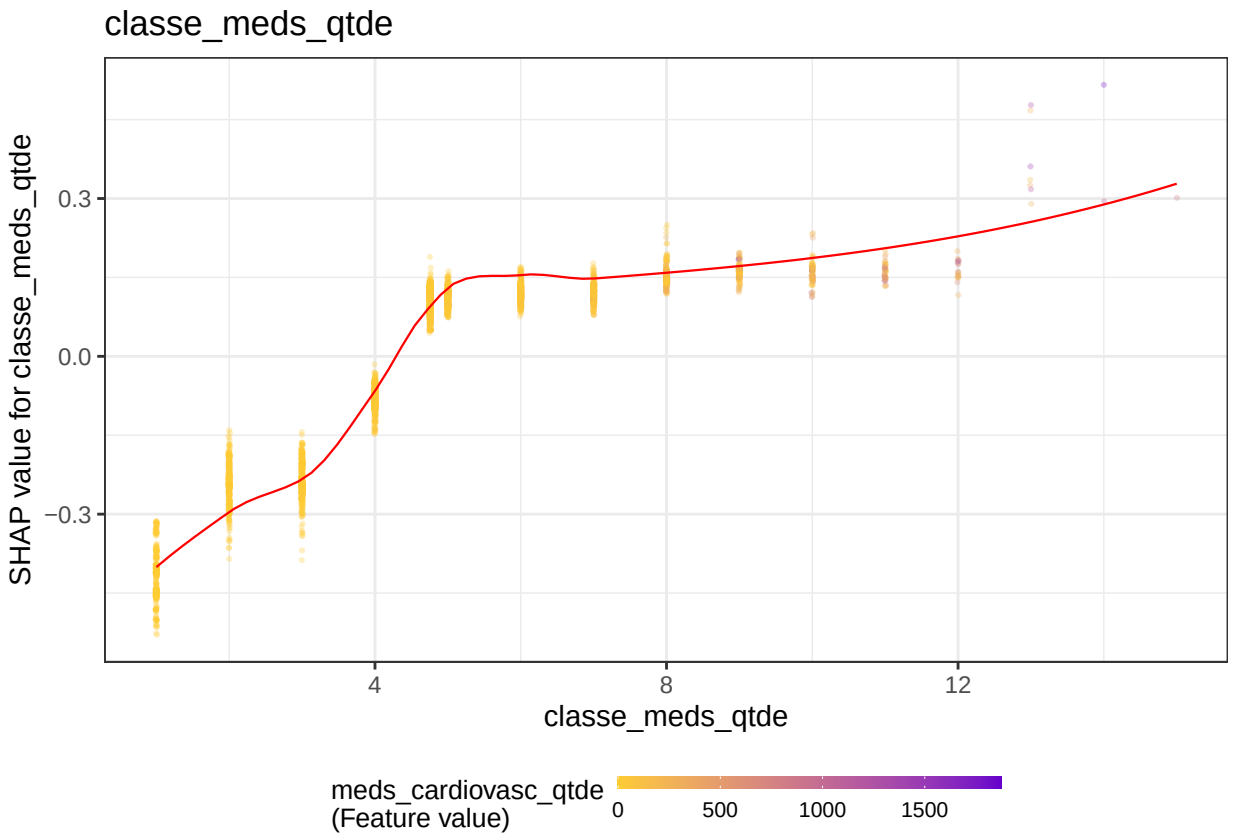
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.08
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 4.7124e-29
```

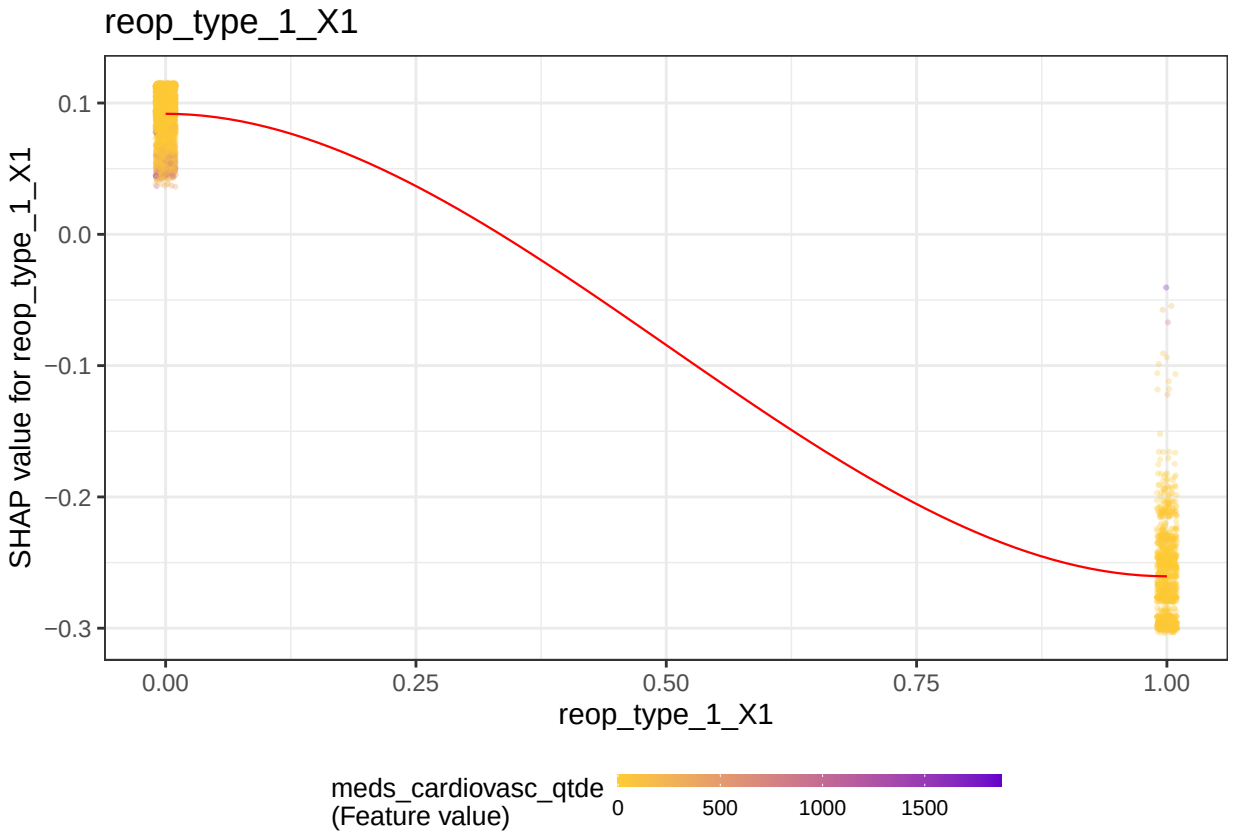
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are
## other near singularities as well. 1
```



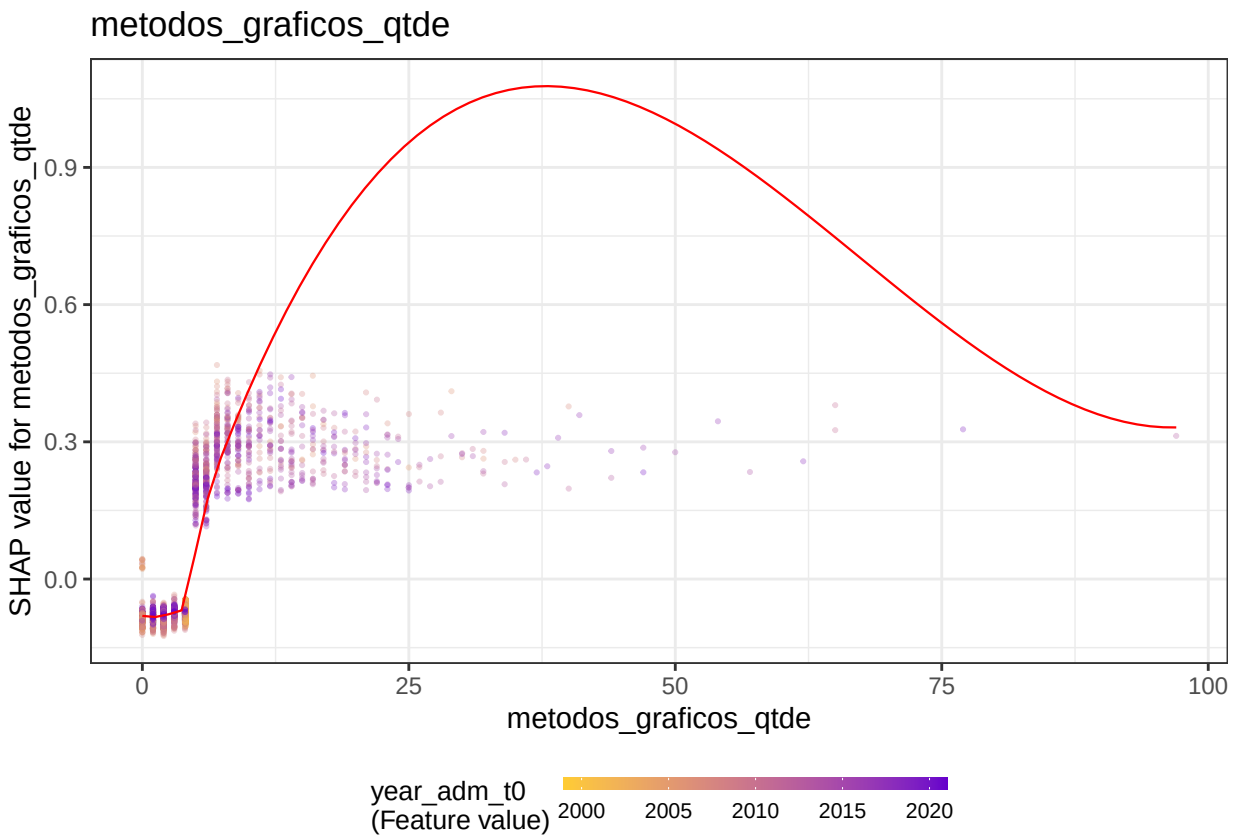

```
## `geom_smooth()` using formula 'y ~ x'
```



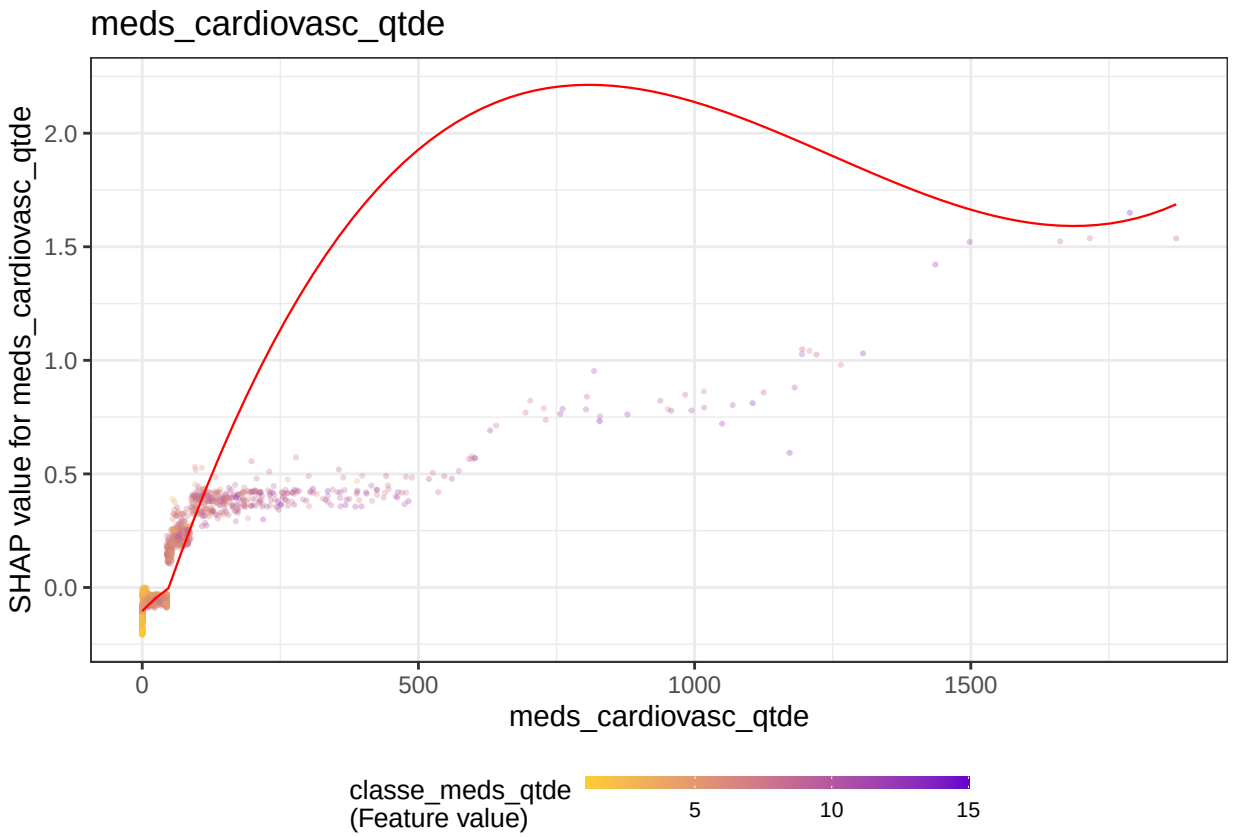

```
## `geom_smooth()` using formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -0.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : neighborhood
## radius 1.005

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : reciprocal
## condition number 1.1644e-28

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : There are
## other near singularities as well. 1.01
```




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

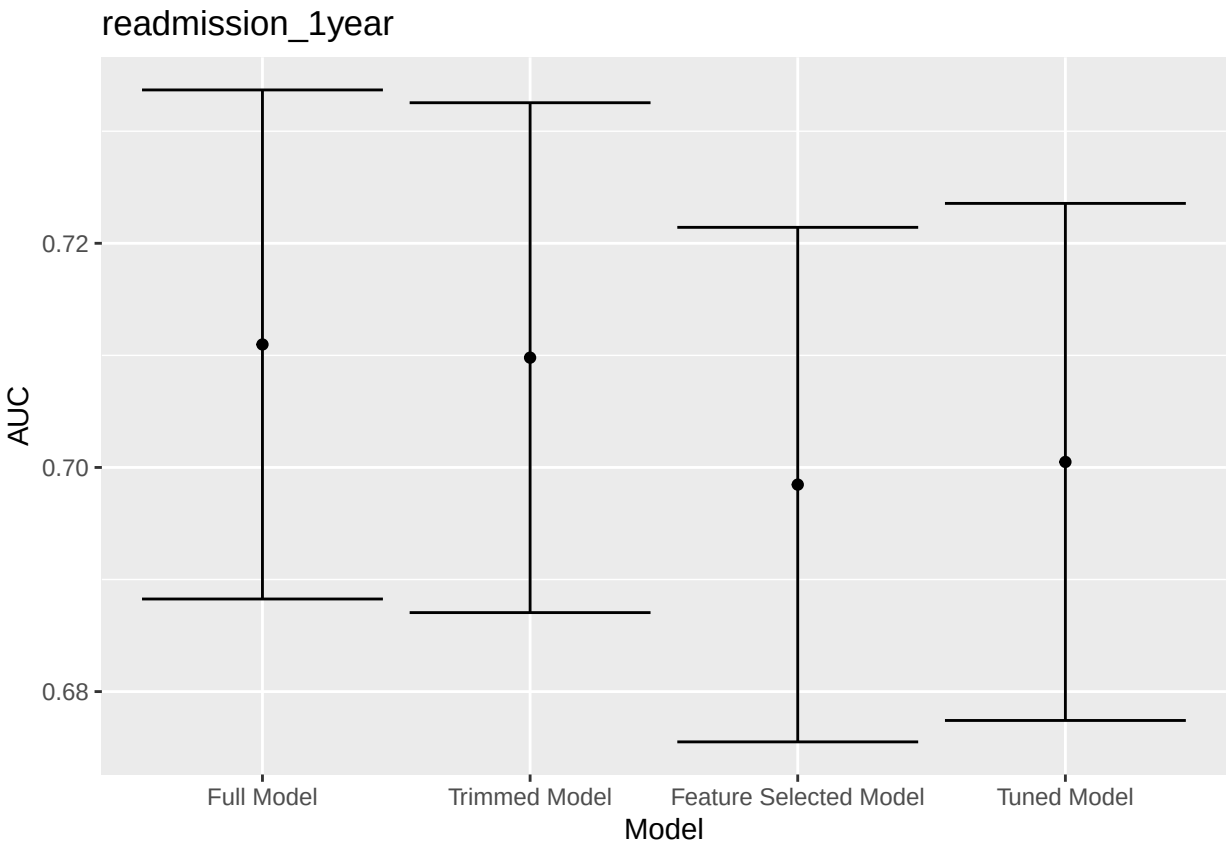



```
## `geom_smooth()` using formula 'y ~ x'
```



Models Comparison

```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper,
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper,
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower, feature_selected_model
  'Tuned Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3],
  # 'Oversampled Model', as.numeric(oversampled_model$auc), oversampled_model$auc_lower, oversampled_model$auc_u
  # 'Undersampled Model', as.numeric(undersampled_model$auc), undersampled_model$auc_lower, undersampled_model$a
) %>%
  mutate(Target = outcome_column,
         Model = factor(Model,
                        levels = c('Full Model', 'Trimmed Model',
                                   'Feature Selected Model', 'Tuned Model',
                                   'Oversampled Model', 'Undersampled Model')))

df_auc %>%
  ggplot(aes(
    x = Model,
    y = AUC,
    ymin = `Lower Limit`,
    ymax = `Upper Limit`
  )) +
  geom_point() +
  geom_errorbar() +
  labs(title = outcome_column)
```




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