

Final Model - readmission_180d

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))
```

```

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. antiaritmico
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. 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. arteriografia
84. cintilografia
85. ecocardiograma
86. endoscopia
87. flebografia
88. pet_ct
89. ultrassom
90. tomografia
91. radiografia
92. ressonancia
93. exames_imagem_qtde
94. bic
95. 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

```
nicerFormatting = 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: 0.715"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.695"

```

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. antiretroviral
6. transplante_cardiaco
7. cir_toracica
8. intervencao_cv
9. angioplastia
10. transfusao
11. espiro_ergoespiro
12. tilt_teste
13. angio_rm
14. arteriografia
15. 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: 0.713"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.695"

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_dropped <- 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
95	0.0000	renal_failure
80	0.0015	renal_failure
79	0.0014	n_procedure_t0
78	0.0009	antifungico
77	0.0013	citologia
76	0.0018	mpp
75	0.0009	antiplaquetario_ev
74	0.0026	teste_esforco
73	0.0011	angio_tc
72	0.0009	heart_disease
71	0.0018	vni
70	0.0021	cec
69	0.0021	valvopathy
68	0.0028	insulina
67	0.0029	cve_desf
66	0.0019	prior_mi
65	0.0024	icp
64	0.0030	cateter_venoso_central
63	0.0026	heart_failure
62	0.0033	cardiac_arrest
61	0.0017	eletrofisiologia
60	0.0025	flebografia
59	0.0019	ultrassom
58	0.0030	tomografia
57	0.0028	cateterismo
56	0.0036	ressonancia
55	0.0030	underlying_heart_disease
54	0.0022	antiviral
53	0.0023	cintilografia
52	0.0015	biopsia
51	0.0033	cultura
50	0.0035	cied_final_group_1
49	0.0032	holter
48	0.0029	diabetes
47	0.0020	anticonvulsivante
46	0.0034	procedure_type_new
45	0.0040	interconsulta
44	0.0036	bic
43	0.0033	af
42	0.0047	bloq_calcio

Table 1: (*continued*)

Number of Features	AUC Loss	Least Important Feature
41	0.0054	procedure_type_1
40	0.0042	sex
39	0.0062	aco
38	0.0067	analises_clinicas_qtde
37	0.0056	copd
36	0.0061	proced_invasivos_qtde
35	0.0082	admission_t0_emergency
34	0.0066	outros_proced_cirurgicos
33	0.0065	endoscopia
32	0.0077	nyha_basal
31	0.0091	digoxina
30	0.0093	ecg
29	0.0086	comorbidities_count
28	0.0071	betabloqueador
27	0.0072	ecocardiograma
26	0.0058	patient_state
25	0.0095	education_level
24	0.0134	insuf_cardiaca

```

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

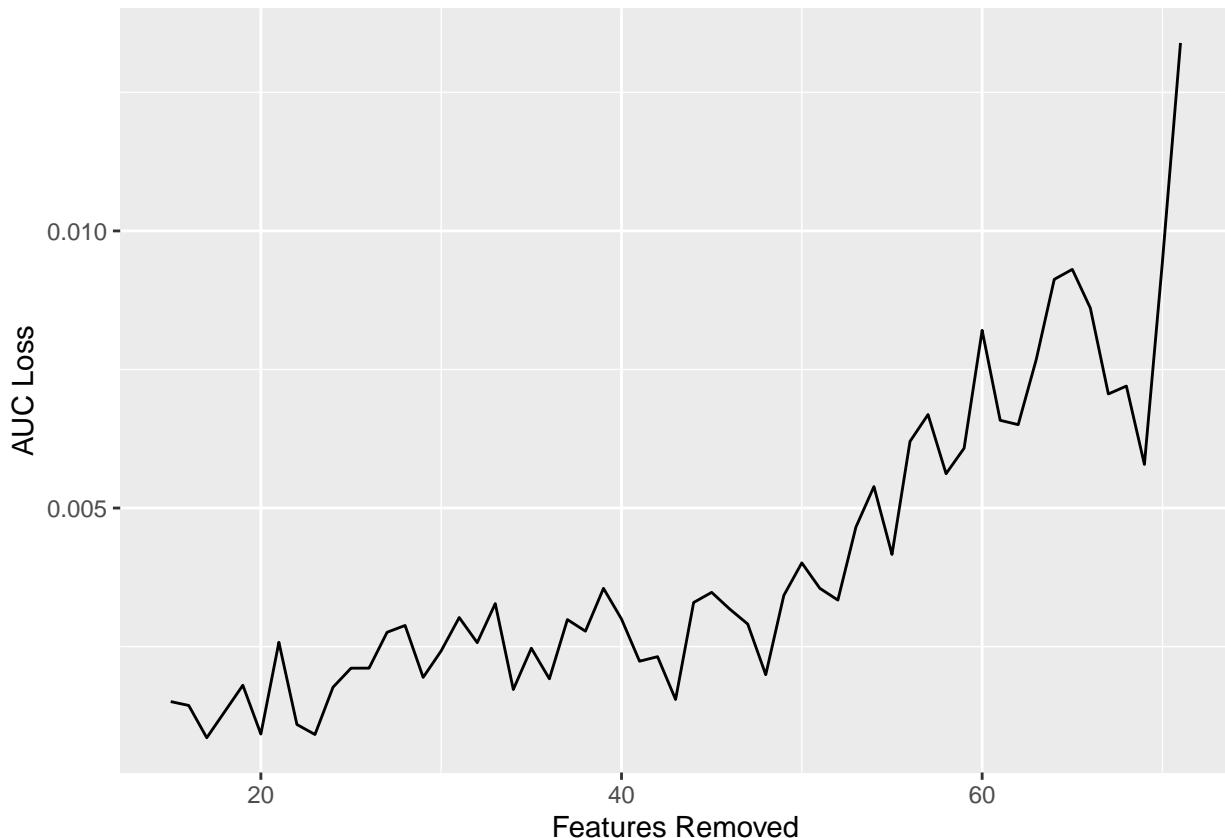
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.701"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.688"

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. meds_cardiovasc_qtde
2. admission_pre_t0_count
3. laboratorio
4. icu_t0
5. age
6. metodos_graficos_qtde
7. classe_meds_qtde
8. antiarritmico
9. admission_pre_t0_180d
10. vasodilatador
11. diuretico
12. cied_final_1
13. equipe_multiprof
14. reop_type_1
15. espironolactona
16. meds_antimicrobianos
17. histopatologia_qtde
18. exames_imagem_qtde
19. dva
20. insuf_cardiaca
21. ieca_bra
22. psicofarmacos
23. estatina
24. radiografia

```
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
7	175	27	11	0.01180	0.00000	roc_auc	binary	0.71304	4	0.01044	Preprocessor1
8	187	31	10	0.01696	0.00014	roc_auc	binary	0.71166	4	0.00899	Preprocessor1
10	160	32	12	0.00000	0.00000	roc_auc	binary	0.71082	4	0.01087	Preprocessor1
10	149	23	14	0.00025	0.23126	roc_auc	binary	0.71062	4	0.01101	Preprocessor1
12	248	31	9	0.00035	0.00000	roc_auc	binary	0.71057	4	0.01128	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")

```



```

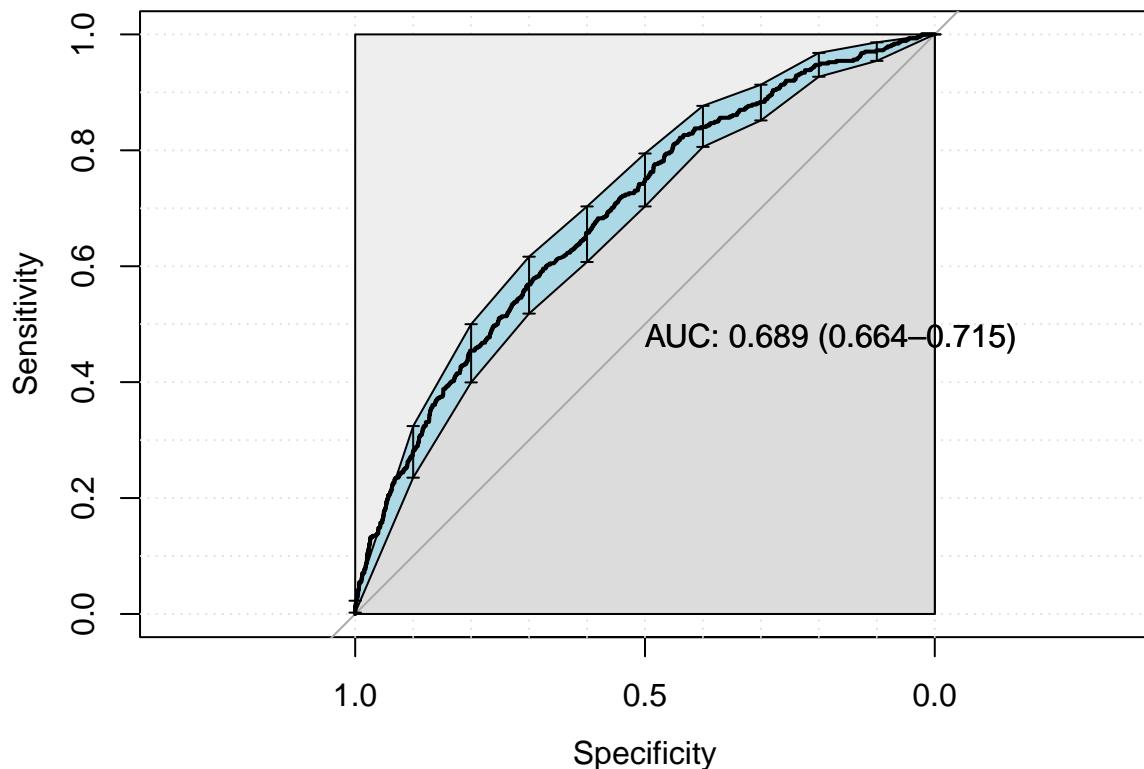
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4292  433
##                      1     0     5
##
##          Accuracy : 0.9085
## 95% CI : (0.8999, 0.9165)
##  No Information Rate : 0.9074
## P-Value [Acc > NIR] : 0.4132
##
##          Kappa : 0.0205
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.00000
##          Specificity  : 0.01142
##  Pos Pred Value  : 0.90836
##  Neg Pred Value  : 1.00000
##          Prevalence   : 0.90740
##  Detection Rate  : 0.90740
## Detection Prevalence : 0.99894
##  Balanced Accuracy : 0.50571
##
##  '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)

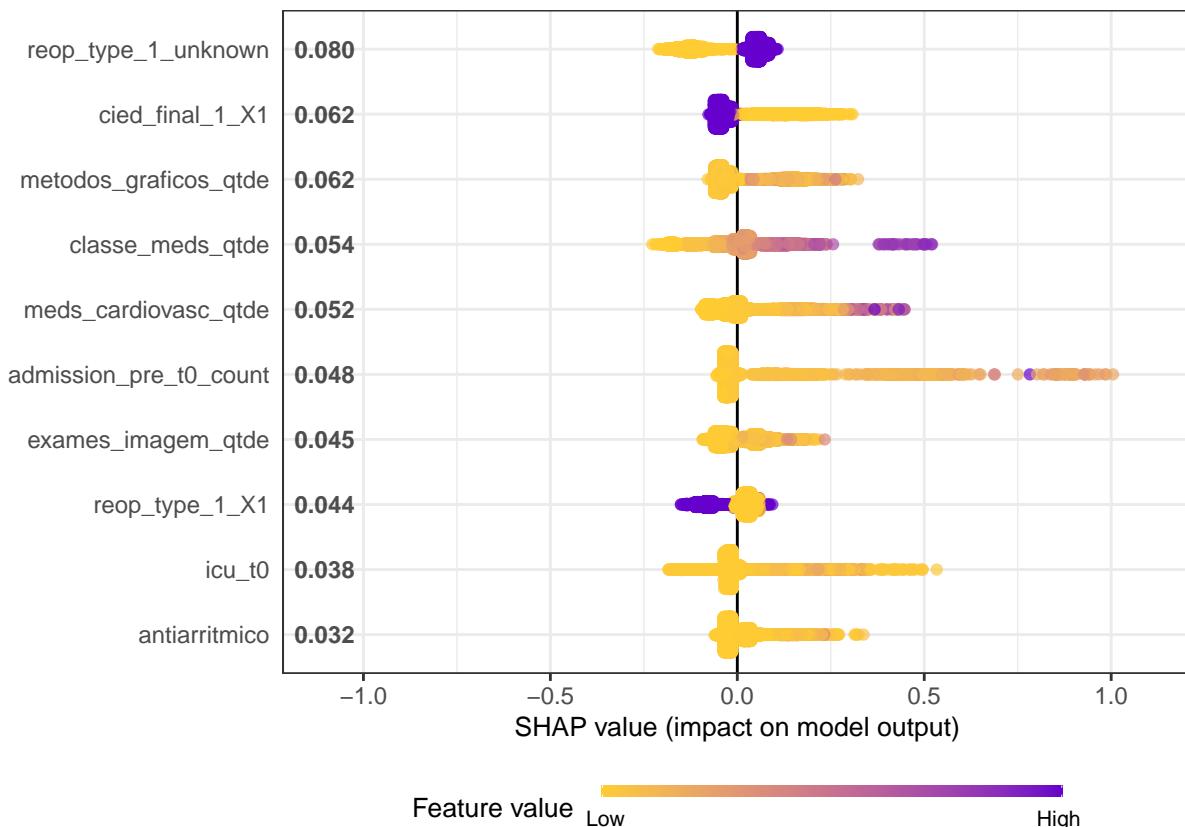
trained_rec <- prep(lightgbm_recipe, training = df_train)

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
  )
}

```

```

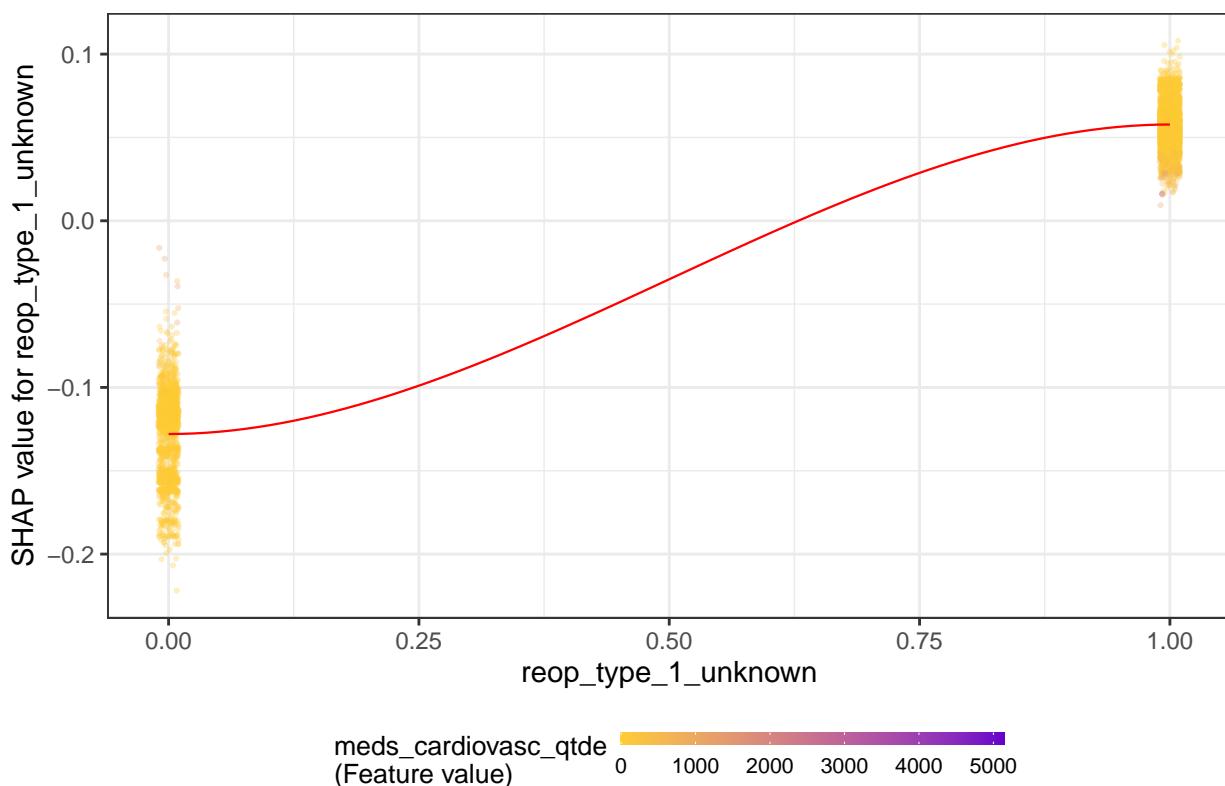
) +
  labs(title = x)
print(p)
}

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

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.005
## 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 5.0206e-29
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 1.01

```

`reop_type_1_unknown`

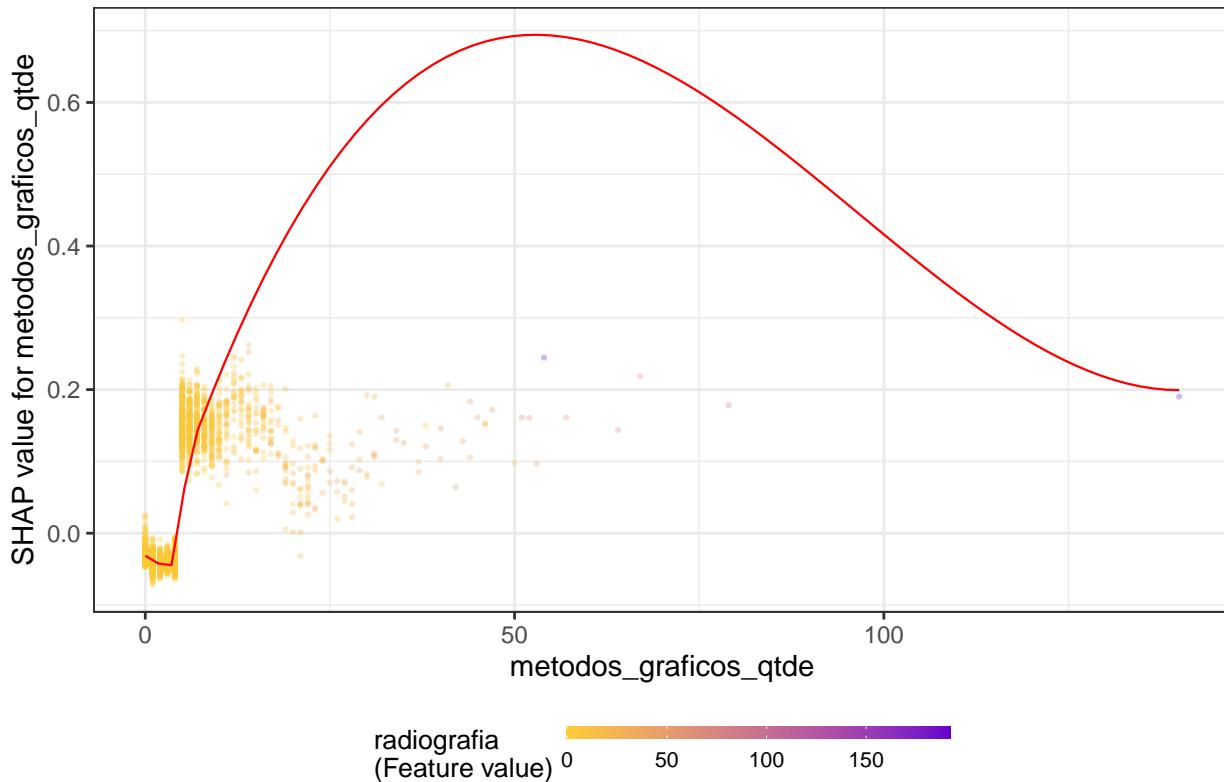


```

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

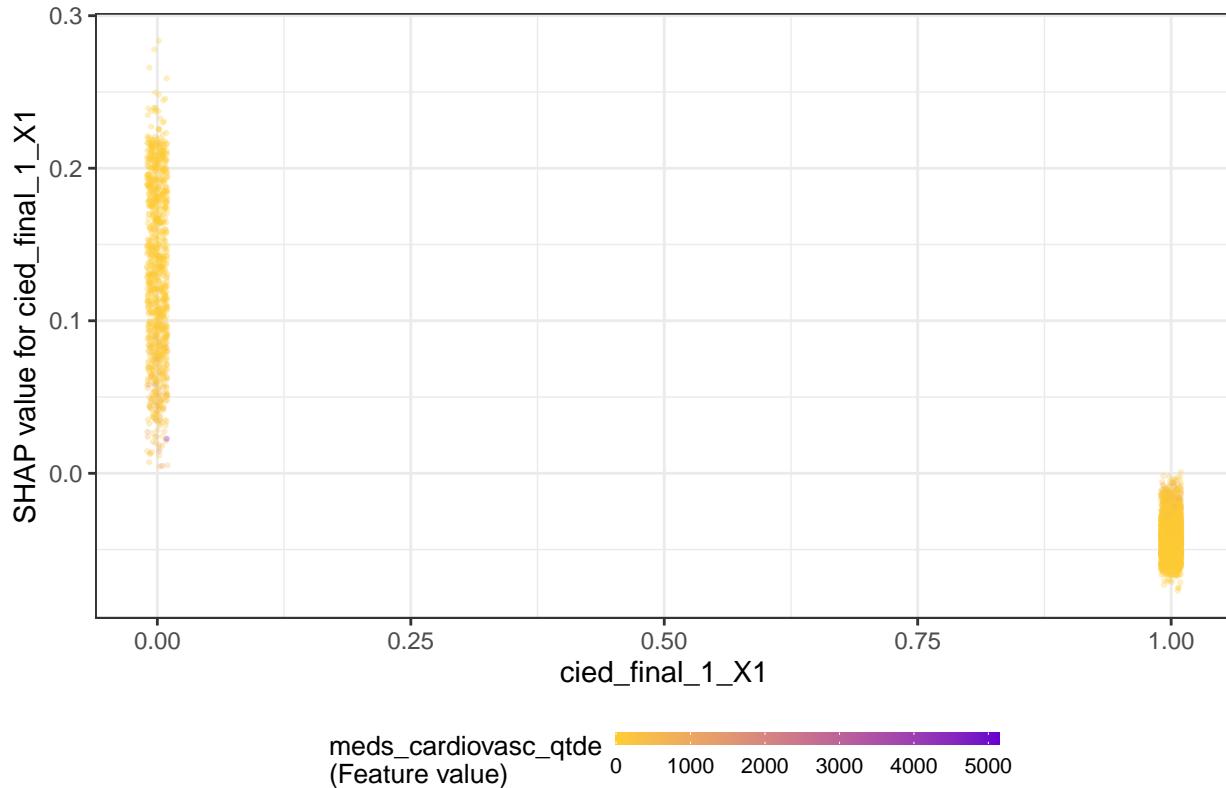
```

metodos_graficos_qtde



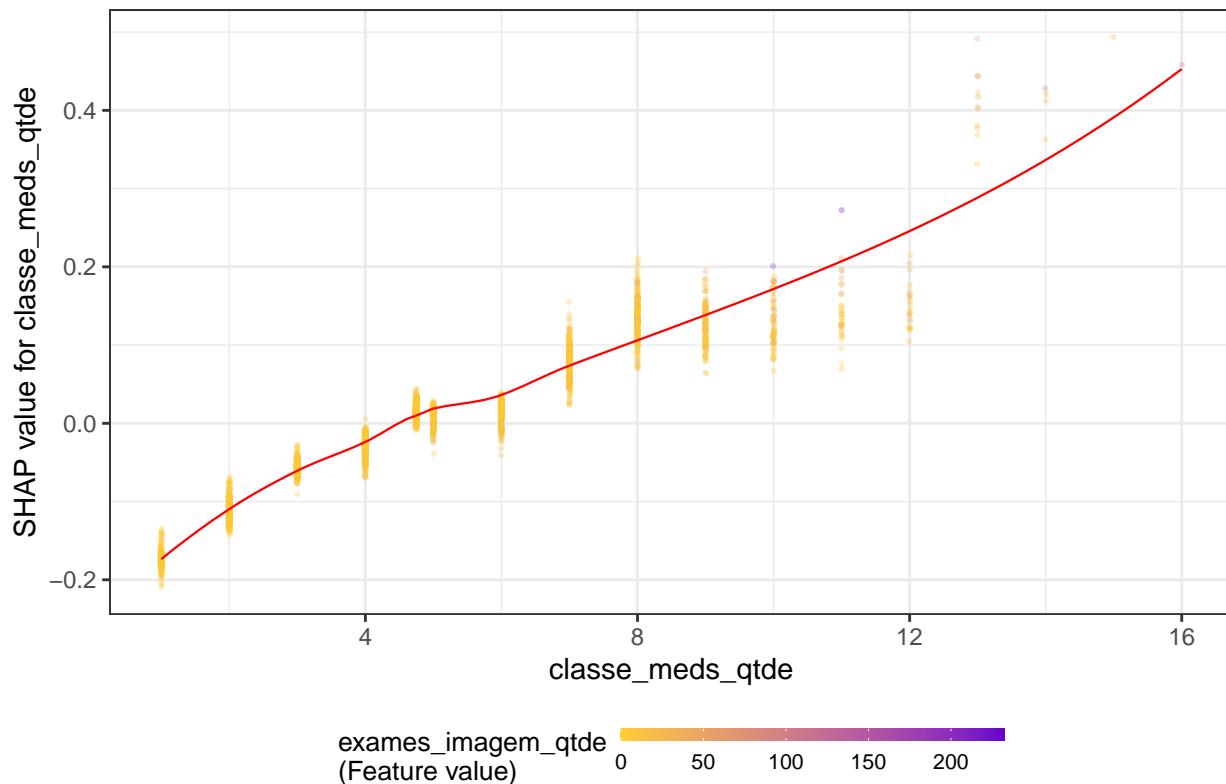
```
## `geom_smooth()` using formula 'y ~ x'  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : pseudoinverse used at -0.005  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : neighborhood radius 1.005  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : reciprocal condition number 2.429e-29  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : at 1.005  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : radius 2.5e-05  
## 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, : There are other near singularities as well. 2.5e-05  
## 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)
```

cied_final_1_X1



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

classe_meds_qtde



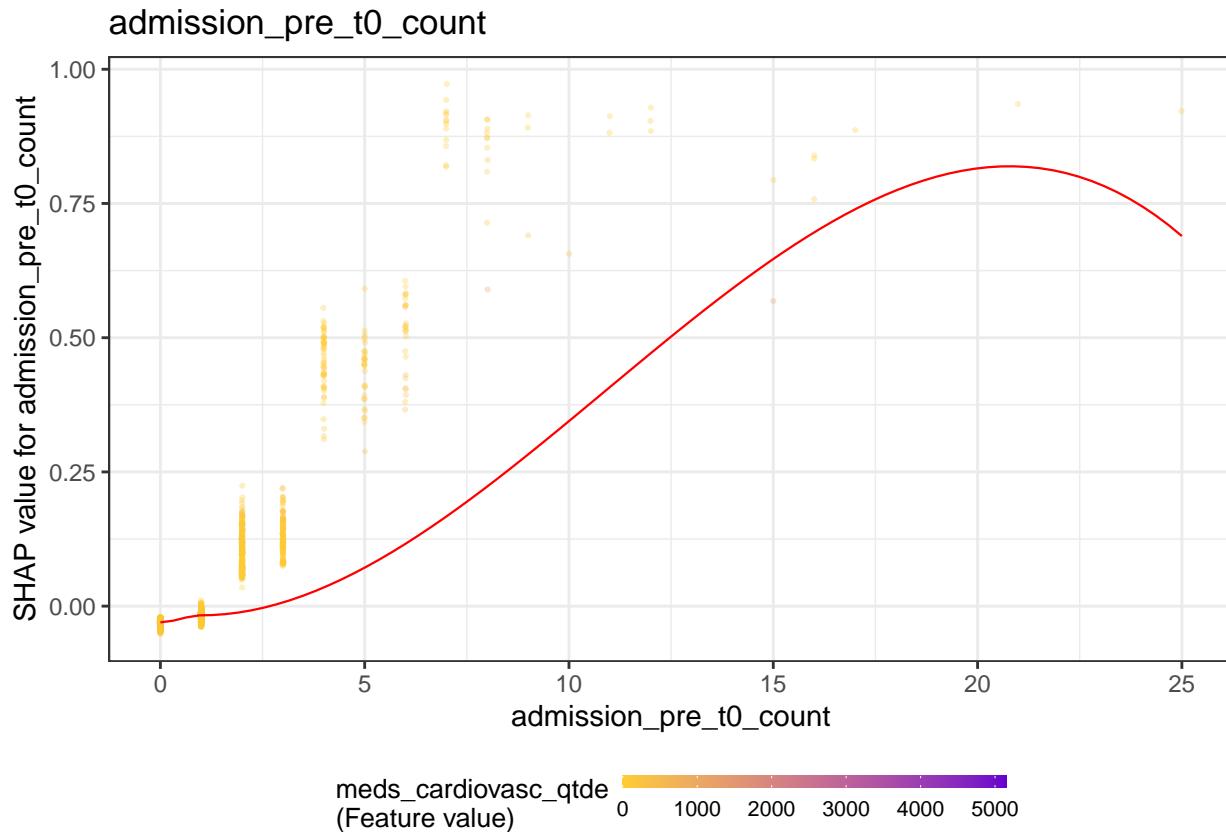
```
## `geom_smooth()` using formula 'y ~ x'
```

```
## 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

```



Models Comparison

```

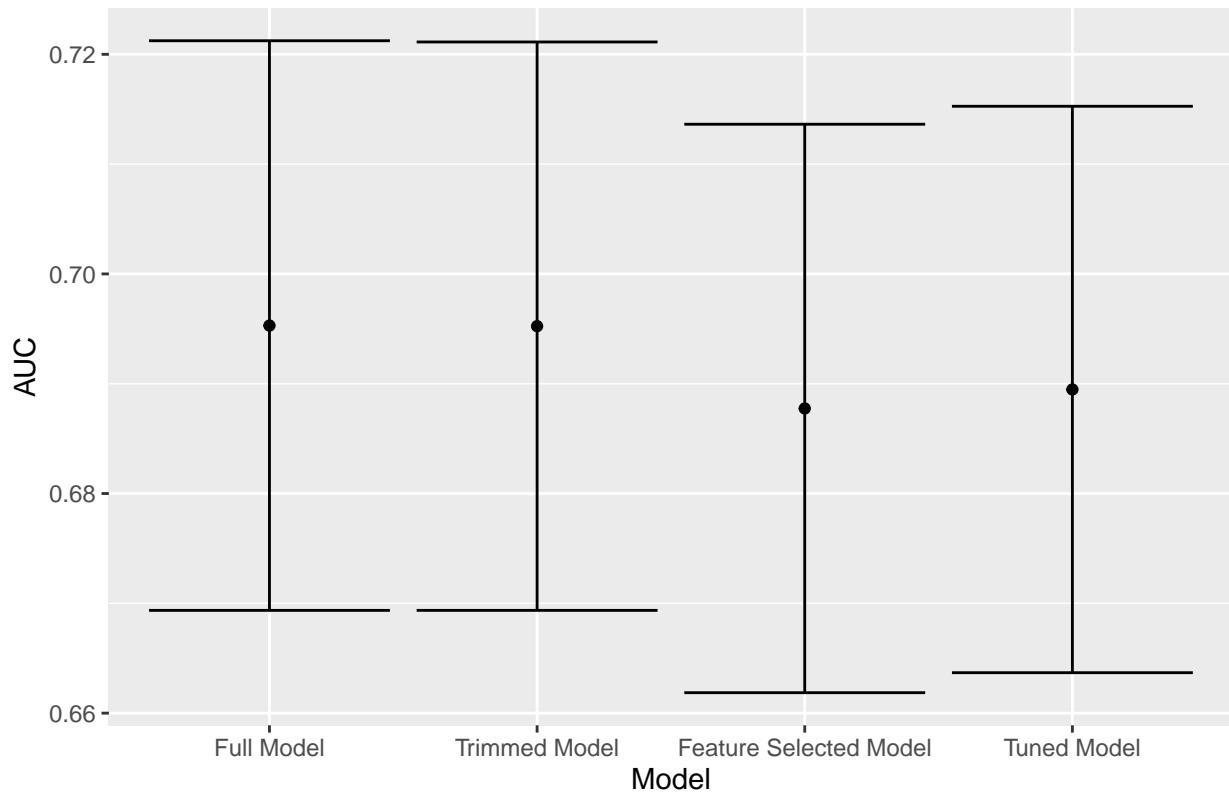
df_auc <- tibble::tribble(
  ~`Model`, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper,
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper,
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower,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

readmission_180d



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