

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)

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')

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

Global parameters

```
k <- 5 # 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(
    "./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: 0.712"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.693"

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. n_procedure_t0
6. antiretroviral
7. cir_toracica
8. intervencao_cv
9. angioplastia
10. tilt_teste
11. citologia
12. angio_rm
13. arteriografia
14. 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: 0.712"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

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

selection_results <- 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	heart_disease
81	0.0003	renal_failure
80	-0.0008	antiplaquetario_ev
79	0.0005	heart_disease
78	-0.0010	espiro_ergoespiro
77	0.0016	transplante_cardiaco
76	-0.0008	teste_esforco
75	-0.0003	angio_tc
74	0.0015	transfusao
73	0.0001	icp
72	0.0011	mpp
71	0.0003	cve_desf
70	0.0005	antifungico
69	0.0004	valvopathy
68	0.0006	cec
67	-0.0006	underlying_heart_disease
66	0.0001	prior_mi
65	-0.0002	cateterismo
64	0.0009	cateter Venoso_Central
63	-0.0024	cardiac_arrest
62	-0.0029	felografia
61	-0.0007	insulina
60	-0.0009	sex

Table 1: (*continued*)

Number of Features	AUC Loss	Least Important Feature
59	-0.0014	eletrofisiologia
58	0.0004	diabetes
57	-0.0006	ressonancia
56	-0.0014	cintilografia
55	0.0004	cultura
54	0.0006	holter
53	0.0002	tomografia
52	0.0013	copd
51	0.0001	heart_failure
50	0.0001	interconsulta
49	-0.0020	ultrassom
48	0.0004	anticonvulsivante
47	-0.0007	vni
46	0.0004	nyha_basal
45	0.0003	bloq_calcio
44	0.0021	bic
43	0.0014	antiviral
42	-0.0001	af
41	-0.0006	cied_final_group_1
40	0.0000	outros_proced_cirurgicos
39	0.0004	aco
38	0.0011	endoscopia
37	0.0016	procedure_type_1
36	0.0006	biopsia
35	0.0001	admission_t0_emergency
34	0.0010	proced_invasivos_qtde
33	0.0015	betabloqueador
32	0.0001	procedure_type_new
31	0.0013	digoxina
30	0.0023	comorbidities_count
29	0.0023	ecg
28	0.0015	patient_state
27	0.0032	analises_clinicas_qtde
26	0.0028	education_level
25	0.0050	radiografia
24	0.0056	diuretico
23	0.0049	ecocardiograma
22	0.0078	insuf_cardiaca
21	0.0086	estatina
20	0.0076	meds_antimicrobianos
19	0.0097	espironolactona
18	0.0129	psicofarmacos

```

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

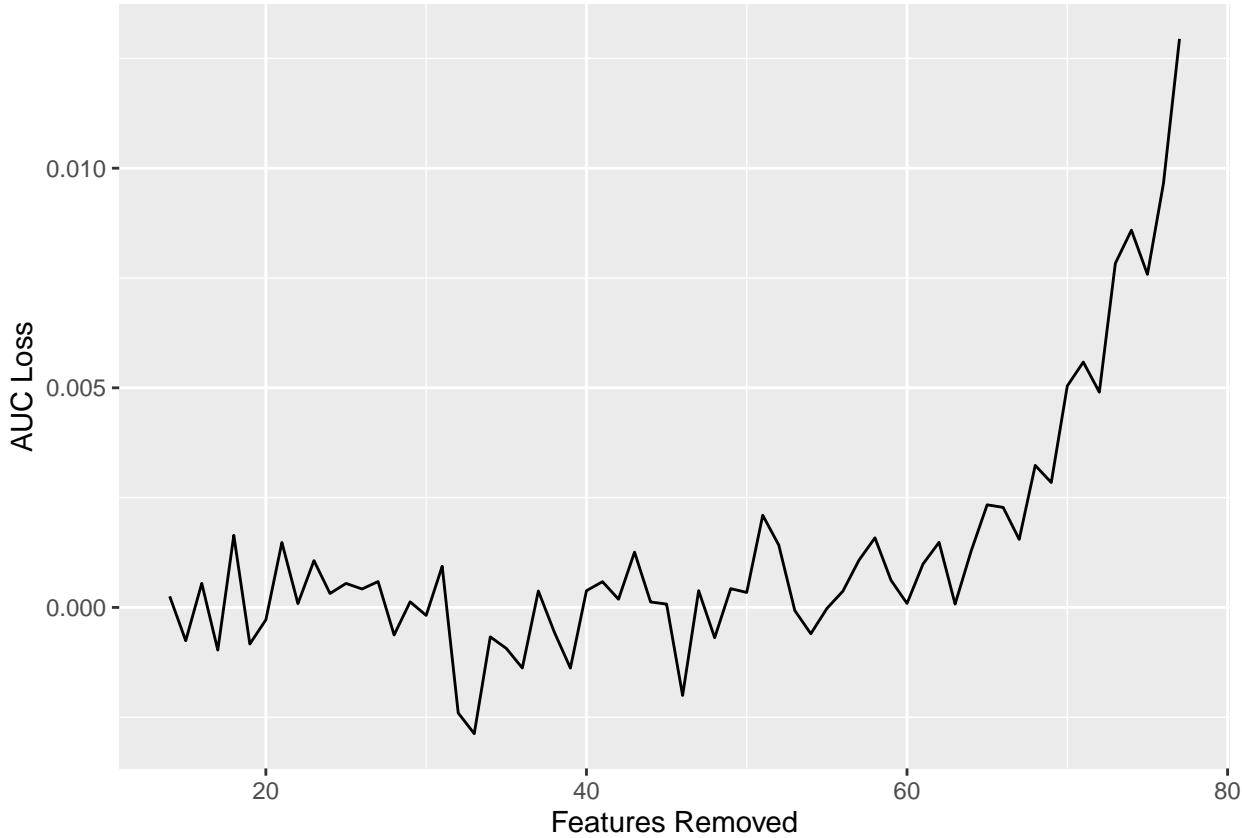
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.700"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.684"

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. age
5. classe_meds_qtde
6. vasodilatador
7. icu_t0
8. metodos_graficos_qtde
9. antiaritmico
10. admission_pre_t0_180d
11. exames_imagem_qtde
12. cied_final_1

```

13. equipe_multiprof
14. reop_type_1
15. psicofarmacos
16. dva
17. histopatologia_qtde
18. ieca_bra

# doParallel::registerDoParallel(8)

```

Smote

```

library(themis)

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(), one_hot = TRUE) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_smote(!!(sym(outcome_column)))

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)

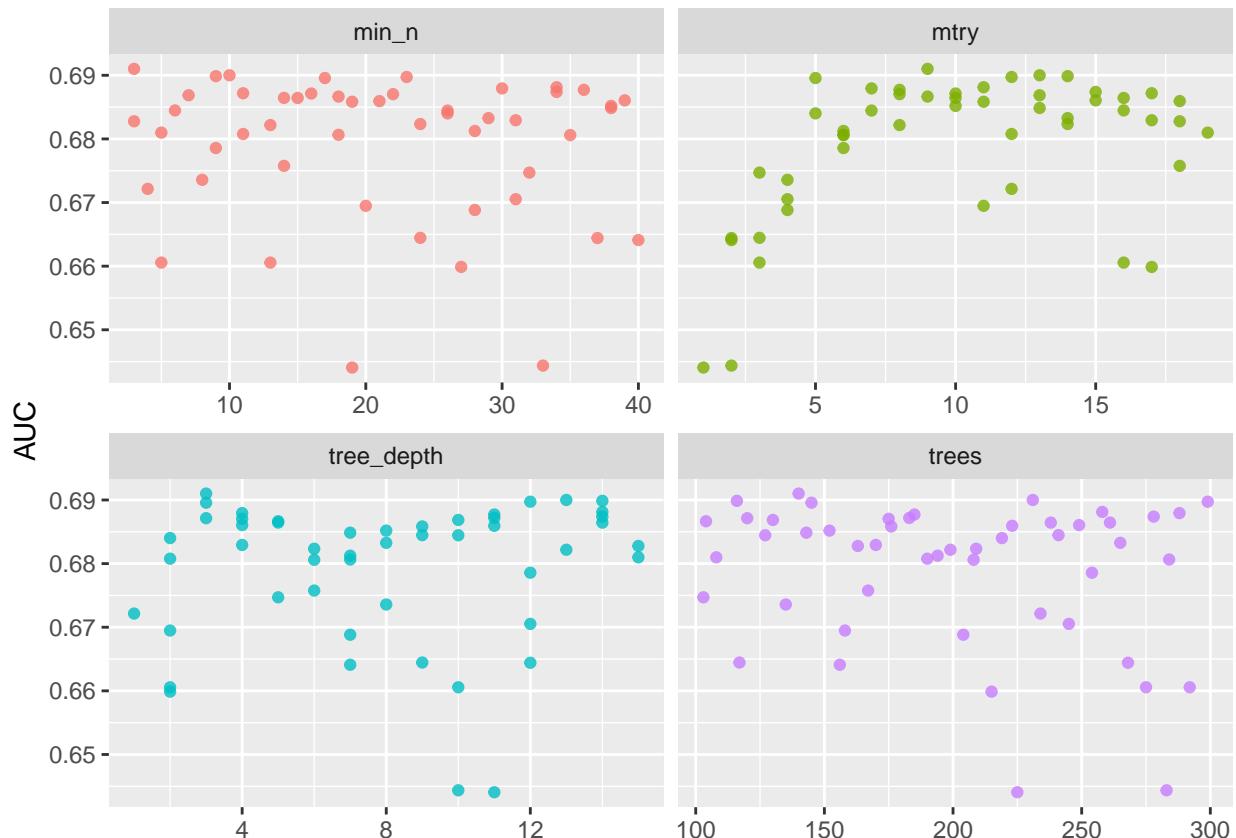
```

Table 2:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
9	140	3	3	0e+00	0.00265	roc_auc	binary	0.69101	5	0.00790	Preprocessor1
13	231	10	13	0e+00	0.00008	roc_auc	binary	0.69000	5	0.00835	Preprocessor1
14	116	9	14	0e+00	0.00000	roc_auc	binary	0.68986	5	0.00899	Preprocessor1
12	299	23	12	0e+00	0.00016	roc_auc	binary	0.68973	5	0.00857	Preprocessor1
5	145	17	3	4e-05	0.00490	roc_auc	binary	0.68957	5	0.00876	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)

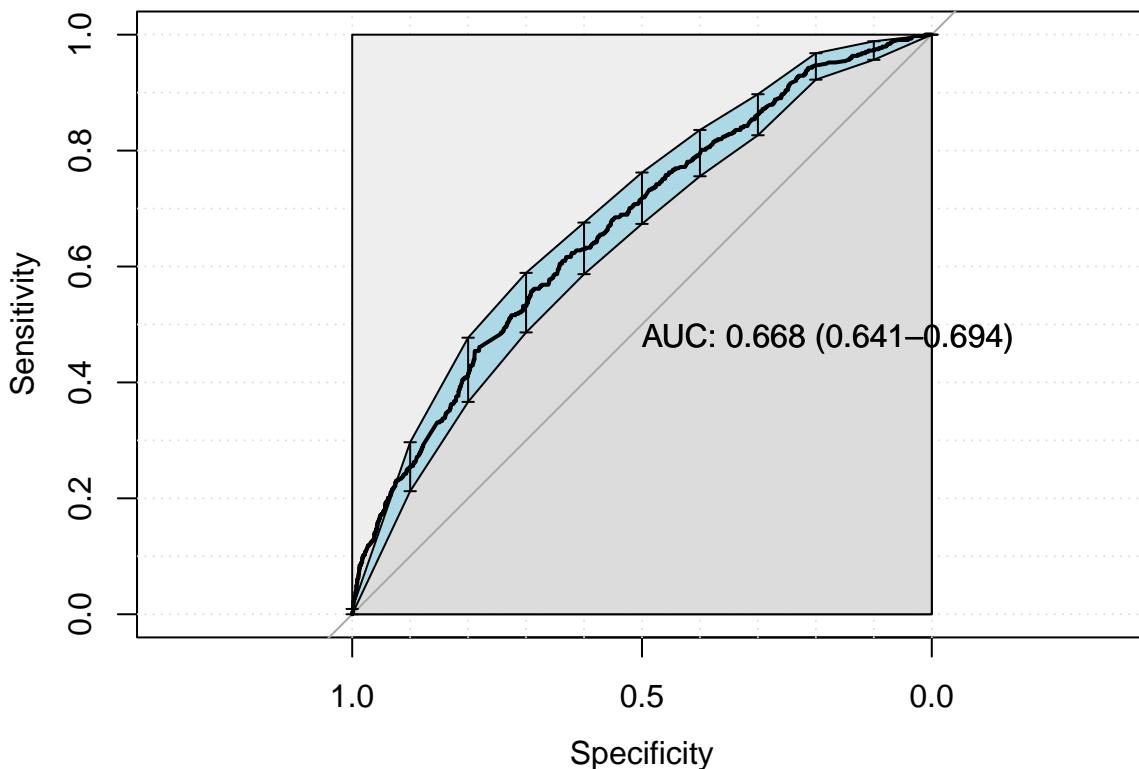
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)
```

```

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_smote_auc <- validation(final_lightgbm_fit, df_test)

```



```

## Confusion Matrix and Statistics
##
## test_predictions_class      0      1
##                      0 3972  339
##                      1  320   99
##
##          Accuracy : 0.8607
##          95% CI : (0.8505, 0.8704)
##  No Information Rate : 0.9074
##  P-Value [Acc > NIR] : 1.0000
##
##          Kappa : 0.1545
##
##  Mcnemar's Test P-Value : 0.4832
##
##          Sensitivity : 0.9254
##          Specificity  : 0.2260
##  Pos Pred Value  : 0.9214
##  Neg Pred Value  : 0.2363
##          Prevalence  : 0.9074
##          Detection Rate : 0.8397
##  Detection Prevalence : 0.9114
##          Balanced Accuracy : 0.5757
##
##          '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(
    "./auxiliar/final_model/hyperparameters/lightgbm_smote_%s.rds",
    outcome_column
  )
)

```

Upsample

```

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(), one_hot = TRUE) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_upsample(!!sym(outcome_column))

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)

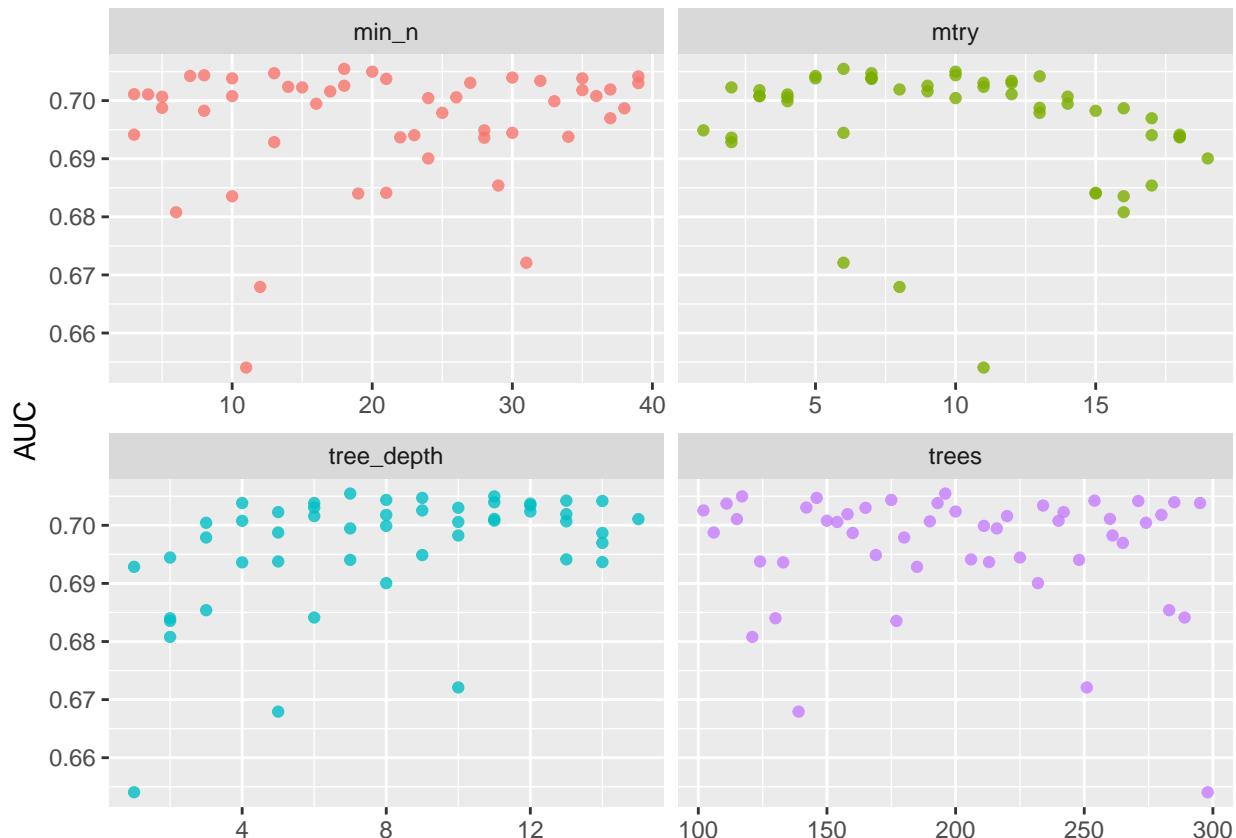
```

Table 3:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
6	196	18	7	0.00057	0.00089	roc_auc	binary	0.70546	5	0.00604	Preprocessor1
10	117	20	11	0.00013	0.00000	roc_auc	binary	0.70498	5	0.00519	Preprocessor1
7	146	13	9	0.00000	0.00000	roc_auc	binary	0.70472	5	0.00520	Preprocessor1
10	175	8	8	0.00000	0.20944	roc_auc	binary	0.70437	5	0.00446	Preprocessor1
5	254	7	13	0.00152	0.02256	roc_auc	binary	0.70424	5	0.00533	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)

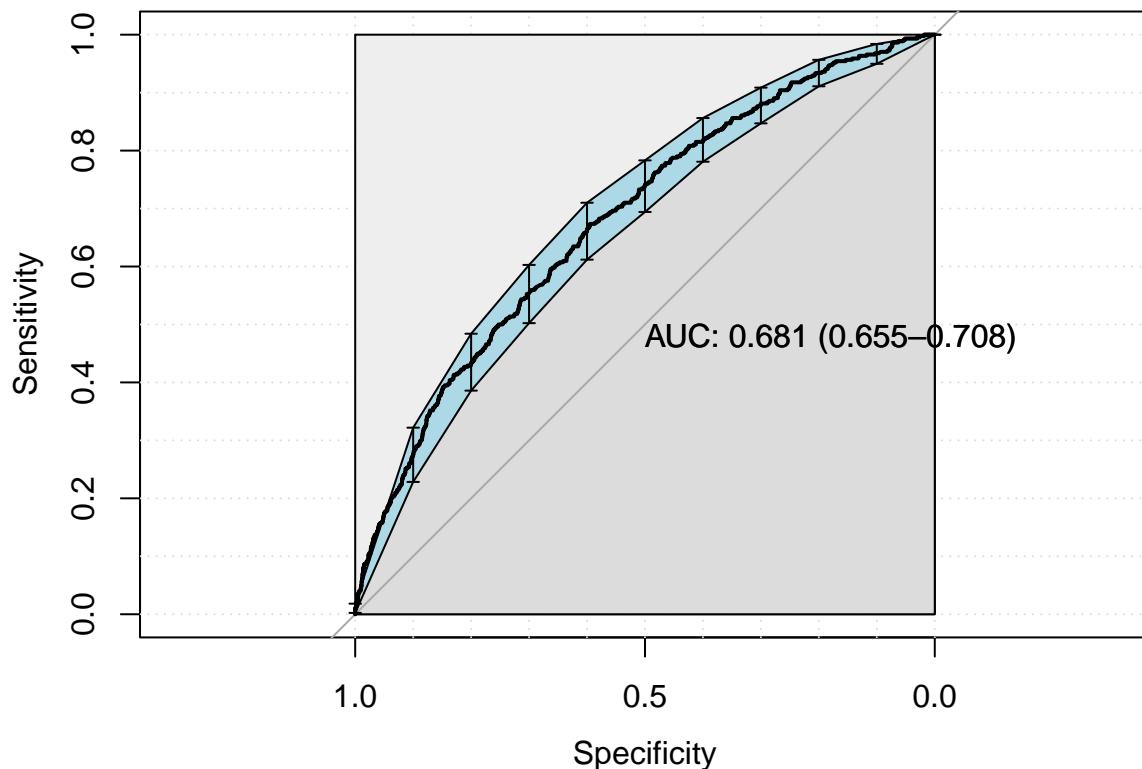
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)
```

```

final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

lightgbm_upsample_auc <- validation(final_lightgbm_fit, df_test)

```



```

## Confusion Matrix and Statistics
##
## 
## test_predictions_class      0      1
##                      0 3278  226
##                      1 1014  212
## 
##          Accuracy : 0.7378
##          95% CI : (0.7251, 0.7503)
##  No Information Rate : 0.9074
##  P-Value [Acc > NIR] : 1
## 
##          Kappa : 0.1371
## 
##  Mcnemar's Test P-Value : <2e-16
## 
##          Sensitivity : 0.7637
##          Specificity  : 0.4840
##          Pos Pred Value : 0.9355
##          Neg Pred Value : 0.1729
##          Prevalence    : 0.9074
##          Detection Rate : 0.6930
##          Detection Prevalence : 0.7408
##          Balanced Accuracy : 0.6239
## 
##          '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(
    "./auxiliar/final_model/hyperparameters/lightgbm_upsample_%s.rds",
    outcome_column
  )
)

```

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(), 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)

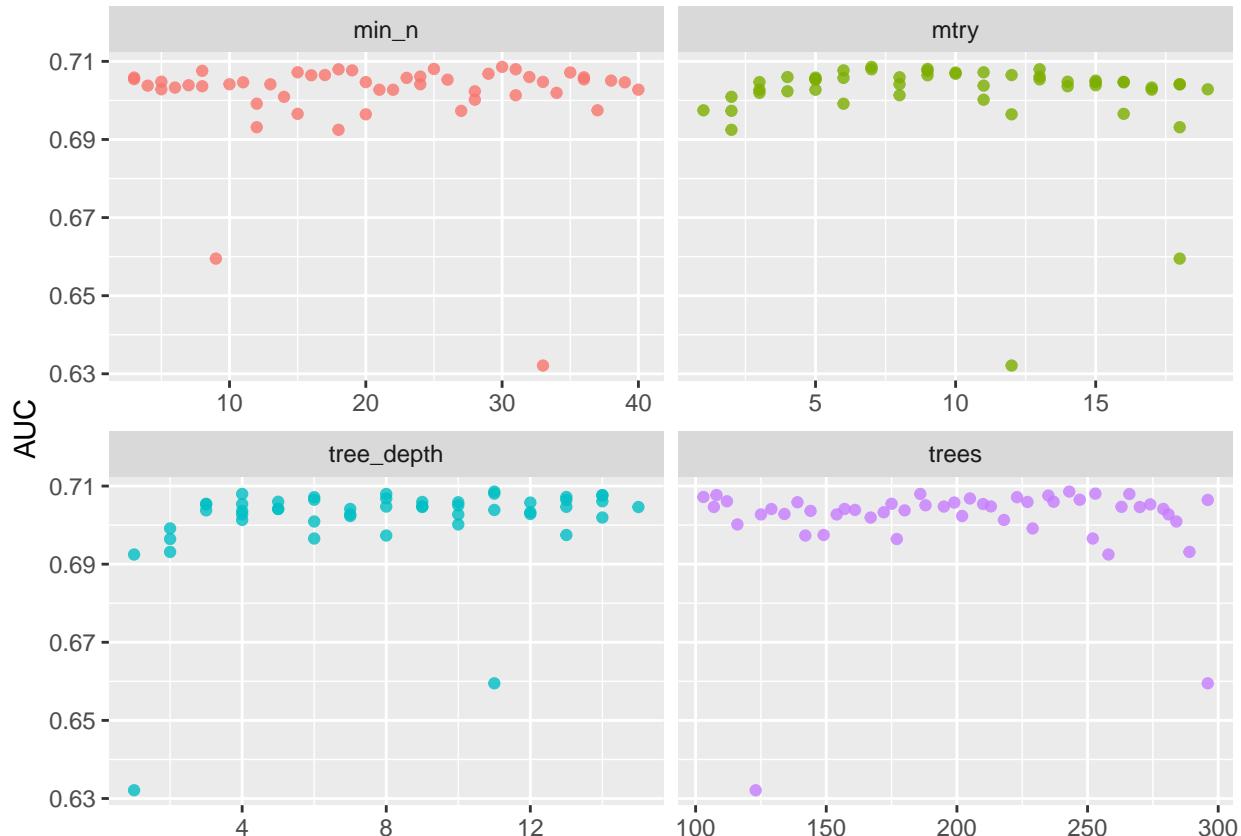
```

Table 4:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
7	243	30	11	0.00000	0.03714	roc_auc	binary	0.70859	5	0.00647	Preprocessor1
9	253	25	11	0.00000	0.00000	roc_auc	binary	0.70806	5	0.00575	Preprocessor1
13	186	31	4	0.00917	0.06557	roc_auc	binary	0.70798	5	0.00390	Preprocessor1
7	266	18	8	0.00018	0.00003	roc_auc	binary	0.70797	5	0.00640	Preprocessor1
6	108	19	14	0.00000	0.00007	roc_auc	binary	0.70770	5	0.00699	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")
```

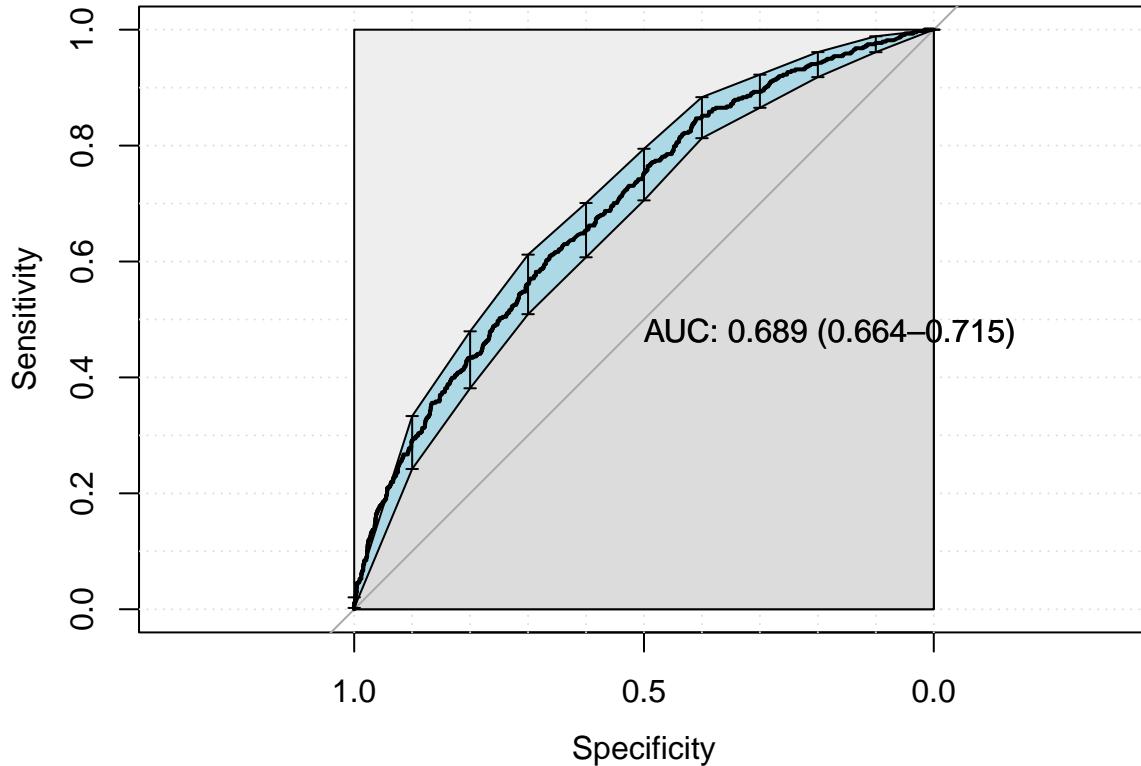


```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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  438
##                      1     0     0
##
##          Accuracy : 0.9074
## 95% CI : (0.8988, 0.9155)
## No Information Rate : 0.9074
## P-Value [Acc > NIR] : 0.5127
##
##          Kappa : 0
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.0000
##          Specificity  : 0.0000
## Pos Pred Value : 0.9074
## Neg Pred Value :      NaN
##          Prevalence : 0.9074
## Detection Rate : 0.9074
## Detection Prevalence : 1.0000
## Balanced Accuracy : 0.5000
##
## '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(
    "./auxiliar/final_model/hyperparameters/lightgbm_%s.rds",
    outcome_column
  )
)

```

SHAP values

```

lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

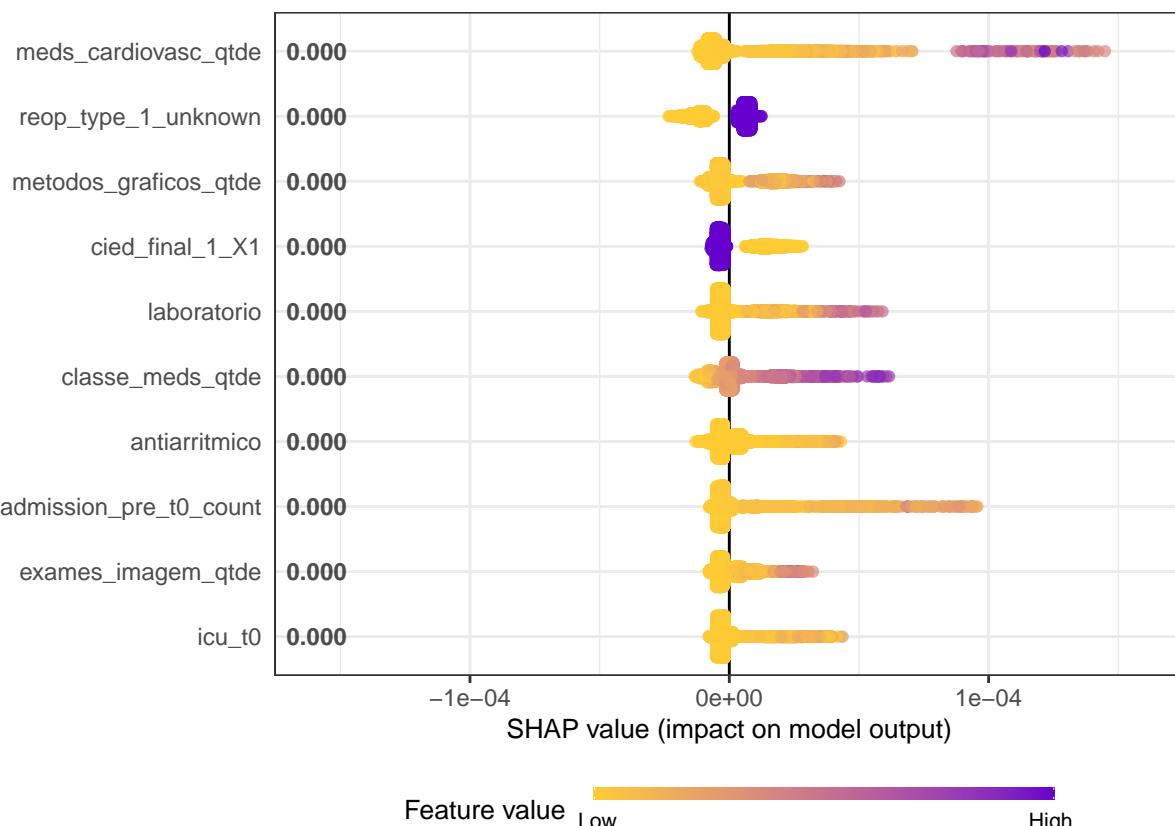
trained_rec <- prep(lightgbm_recipe, training = df_train)

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,

```

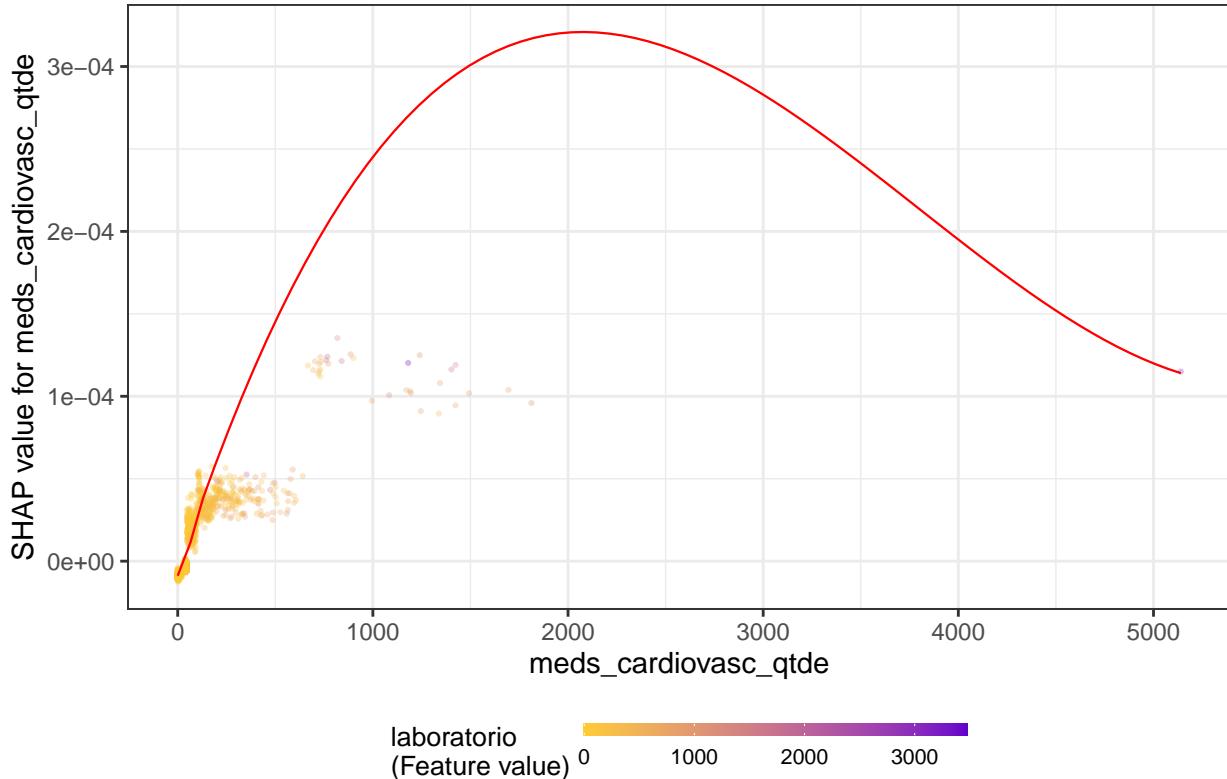
```

    color_feature = "auto",
    smooth = TRUE,
    jitter_width = 0.01,
    alpha = 0.3
) +
  labs(title = x)
print(p)
}

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

```

meds_cardiovasc_qtde

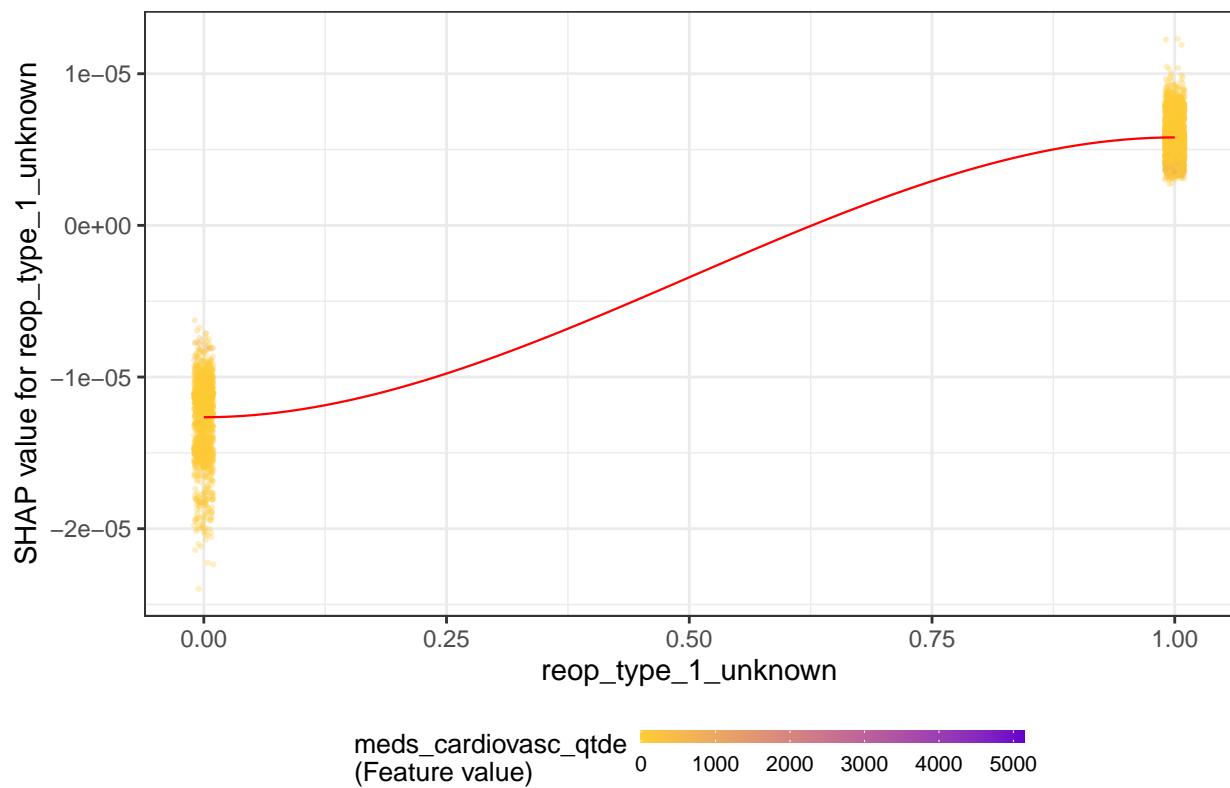


```

## `geom_smooth()` using formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudo-inverse
## 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

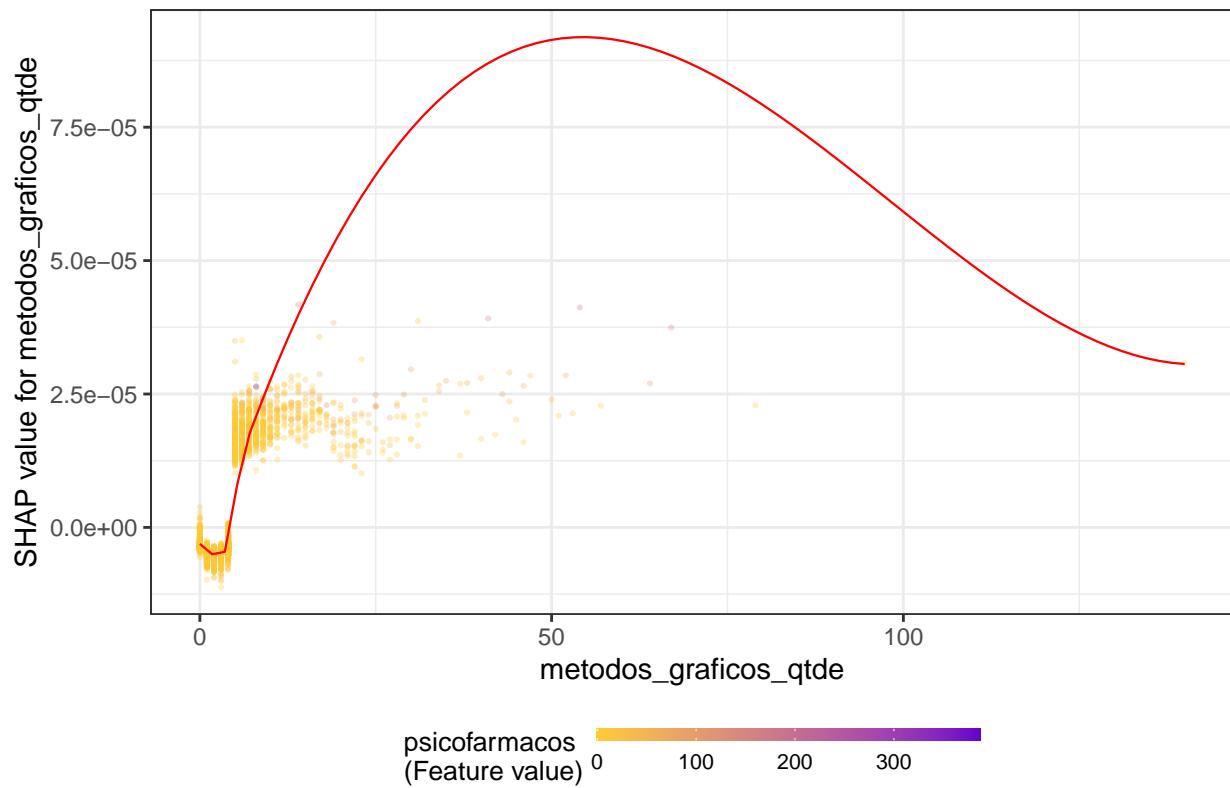
```

reop_type_1_unknown



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

metodos_graficos_qtde



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

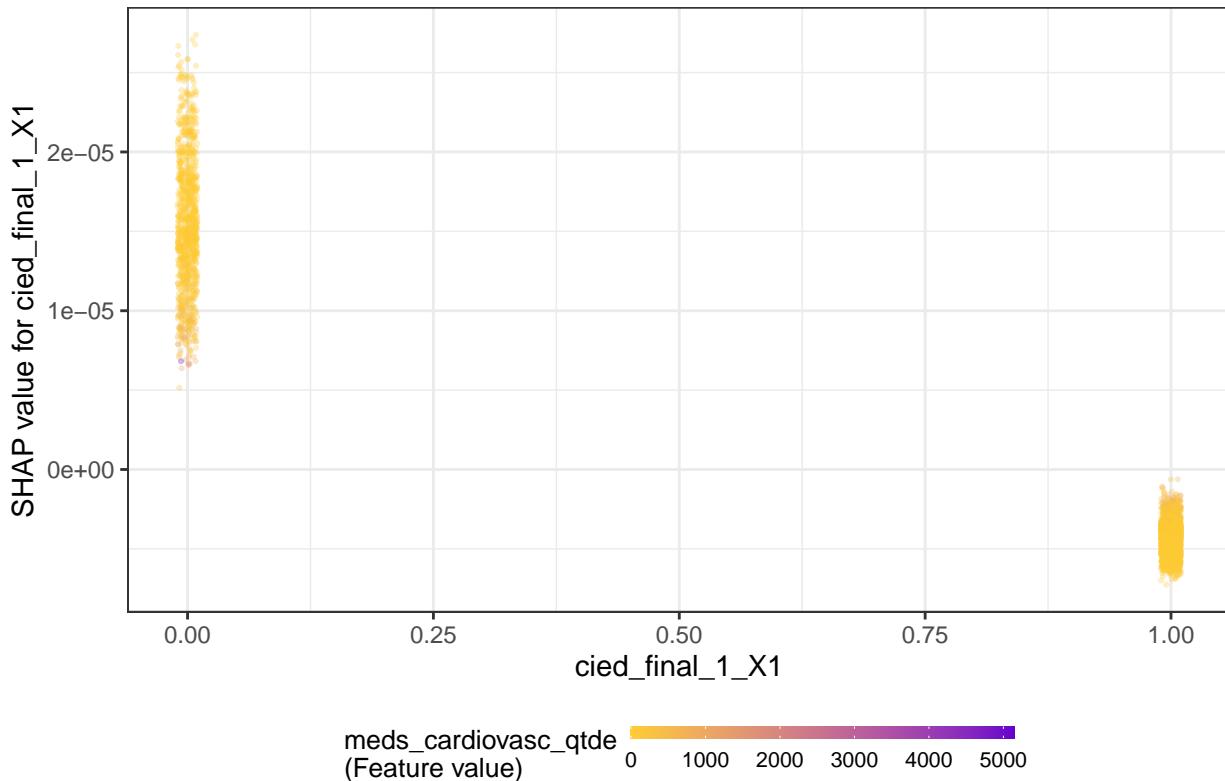
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric, : pseudoinverse
## used at -0.005
```

```

## 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

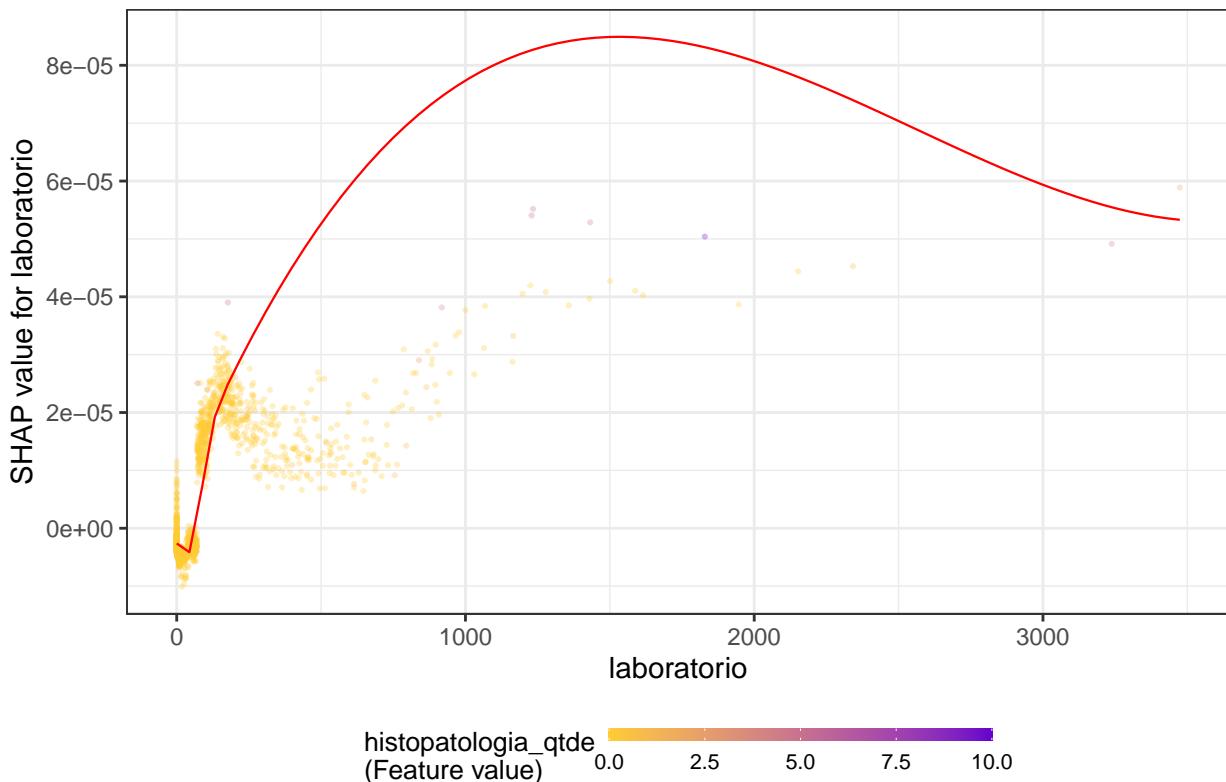


```

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

```

laboratorio



Models Comparison

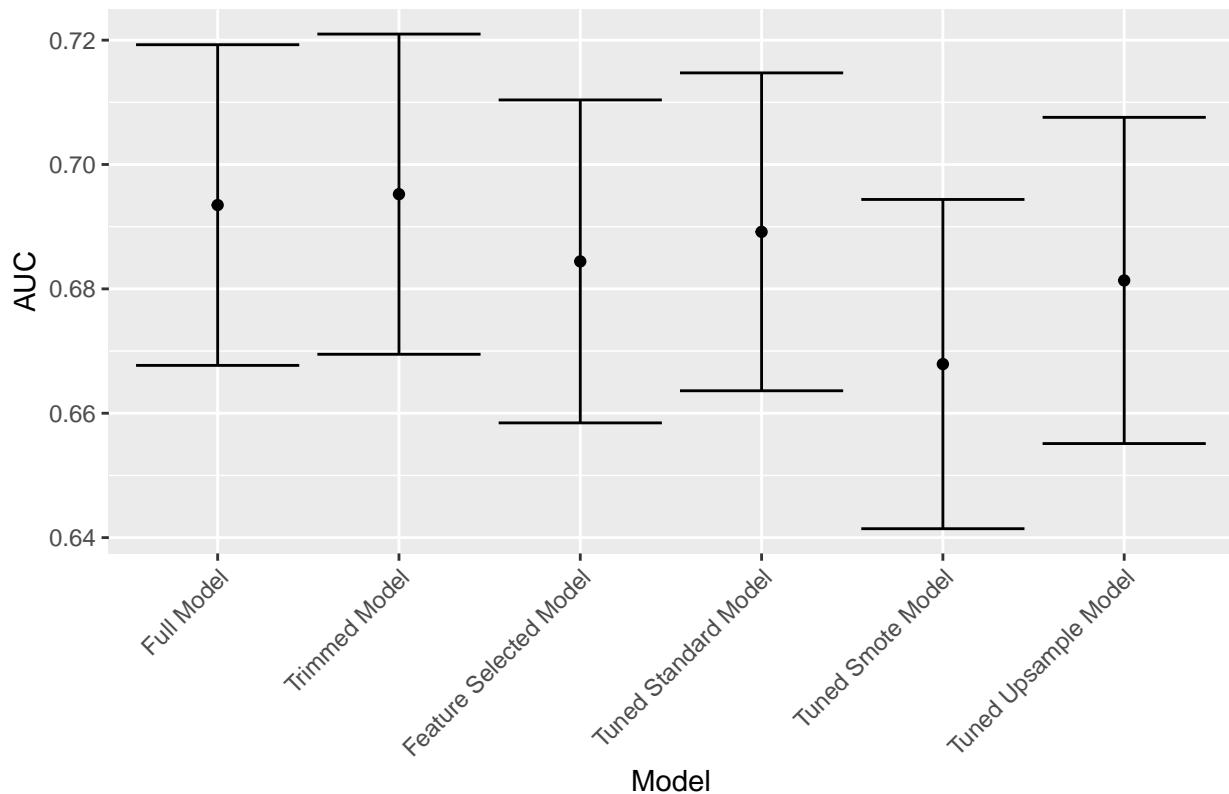
```

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$auc_upper,
  'Tuned Standard Model', as.numeric(lightgbm_auc$auc), lightgbm_auc$ci[1], lightgbm_auc$ci[3],
  'Tuned Smote Model', as.numeric(lightgbm_smote_auc$auc), lightgbm_smote_auc$ci[1], lightgbm_smote_auc$ci[3],
  'Tuned Upsample Model', as.numeric(lightgbm_upsample_auc$auc), lightgbm_upsample_auc$ci[1], lightgbm_upsample_auc$ci[3]
) %>%
  mutate(Target = outcome_column,
         Model = factor(Model,
                         levels = c('Full Model', 'Trimmed Model',
                                   'Feature Selected Model', 'Tuned Standard Model',
                                   'Tuned Smote Model', 'Tuned Upsample Model')))

df_auc %>%
  ggplot(aes(
    x = Model,
    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))

```

readmission_180d



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