

Final Model - readmission_60d

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. age
2. education_level
3. underlying_heart_disease
4. heart_disease
5. nyha_basal
6. prior_mi
7. heart_failure
8. af
9. cardiac_arrest
10. transplant
11. valvopathy
12. diabetes
13. hemodialysis
14. comorbidities_count
15. procedure_type_1
16. reop_type_1
17. procedure_type_new
18. cied_final_1
19. cied_final_group_1
20. admission_pre_t0_count
21. admission_pre_t0_180d
22. icu_t0
23. dialysis_t0
24. admission_t0_emergency
25. aco
26. antiarritmico
27. betabloqueador
28. ieca_bra
29. dva
30. digoxina
31. estatina
32. diuretico
33. vasodilatador
34. insuf_cardiaca
35. espironolactona
36. bloq_calcio
37. antiplaquetario_ev

38. insulina
39. anticonvulsivante
40. psicofarmacos
41. antifungico
42. antiviral
43. classe_meds_qtde
44. meds_cardiovasc_qtde
45. meds_antimicrobianos
46. cec
47. transplante_cardiaco
48. cir_toracica
49. outros_proced_cirurgicos
50. icp
51. angioplastia
52. cateterismo
53. eletrofisiologia
54. cateter_venoso_central
55. proced_invasivos_qtde
56. cve_desf
57. transfusao
58. interconsulta
59. equipe_multiprof
60. ecg
61. holter
62. teste_esforco
63. espiro_ergoespiro
64. tilt_teste
65. metodos_graficos_qtde
66. laboratorio
67. cultura
68. analises_clinicas_qtde
69. citologia
70. biopsia
71. histopatologia_qtde
72. angio_rm
73. angio_tc
74. arteriografia
75. cintilografia
76. ecocardiograma
77. endoscopia
78. pet_ct
79. ultrassom
80. tomografia
81. radiografia
82. ressonancia
83. exames_imagem_qtde
84. bic

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: 0.707"
sprintf('Full Model Test AUC: %.3f' ,full_model$auc)

## [1] "Full Model Test AUC: 0.685"

Features with zero importance on the initial model:
unimportant_features <- setdiff(features, full_model$importance$Feature)

unimportant_features %>%
  gluedown::md_order()

1. heart_disease
2. transplant
3. hemodialysis
4. dialysis_t0
5. antiplaquetario_ev
6. cir_toracica
7. icp
8. angioplastia
9. espiro_ergoespiro
10. tilt_teste
11. angio_rm
12. arteriografia

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: 0.706"
sprintf('Trimmed Model Test AUC: %.3f' ,trimmed_model$auc)

## [1] "Trimmed Model Test AUC: 0.686"

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
84	0.0000	pet_ct
72	0.0014	cateter_venoso_central
71	0.0050	valvopathy
70	0.0022	underlying_heart_disease
69	0.0029	teste_esforco
68	0.0014	cec
67	0.0021	diabetes
66	0.0020	transplante_cardiaco
65	0.0037	education_level
64	0.0013	nyha_basal
63	0.0032	ressonancia
62	0.0017	angio_tc
61	0.0034	endoscopia
60	0.0019	eletrofisiologia
59	0.0032	citologia
58	0.0029	cardiac_arrest
57	0.0028	holter
56	0.0046	pet_ct
55	0.0013	tomografia
54	0.0002	cateterismo
53	0.0040	insulina
52	0.0045	prior_mi
51	0.0042	biopsia
50	0.0038	antifungico
49	0.0045	cultura
48	0.0044	interconsulta
47	0.0034	cve_desf
46	0.0054	procedure_type_1
45	0.0054	digoxina
44	0.0042	cintilografia
43	0.0043	transfusao
42	0.0047	ultrassom
41	0.0047	outros_proced_cirurgicos
40	0.0037	procedure_type_new
39	0.0039	antiviral
38	0.0054	comorbidities_count
37	0.0062	ecocardiograma
36	0.0071	aco

Table 1: (*continued*)

Number of Features	AUC Loss	Least Important Feature
35	0.0048	af
34	0.0061	cied_final_1
33	0.0072	bic
32	0.0056	heart_failure
31	0.0059	insuf_cardiaca
30	0.0064	dva
29	0.0065	histopatologia_qtde
28	0.0064	estatina
27	0.0097	radiografia
26	0.0114	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)

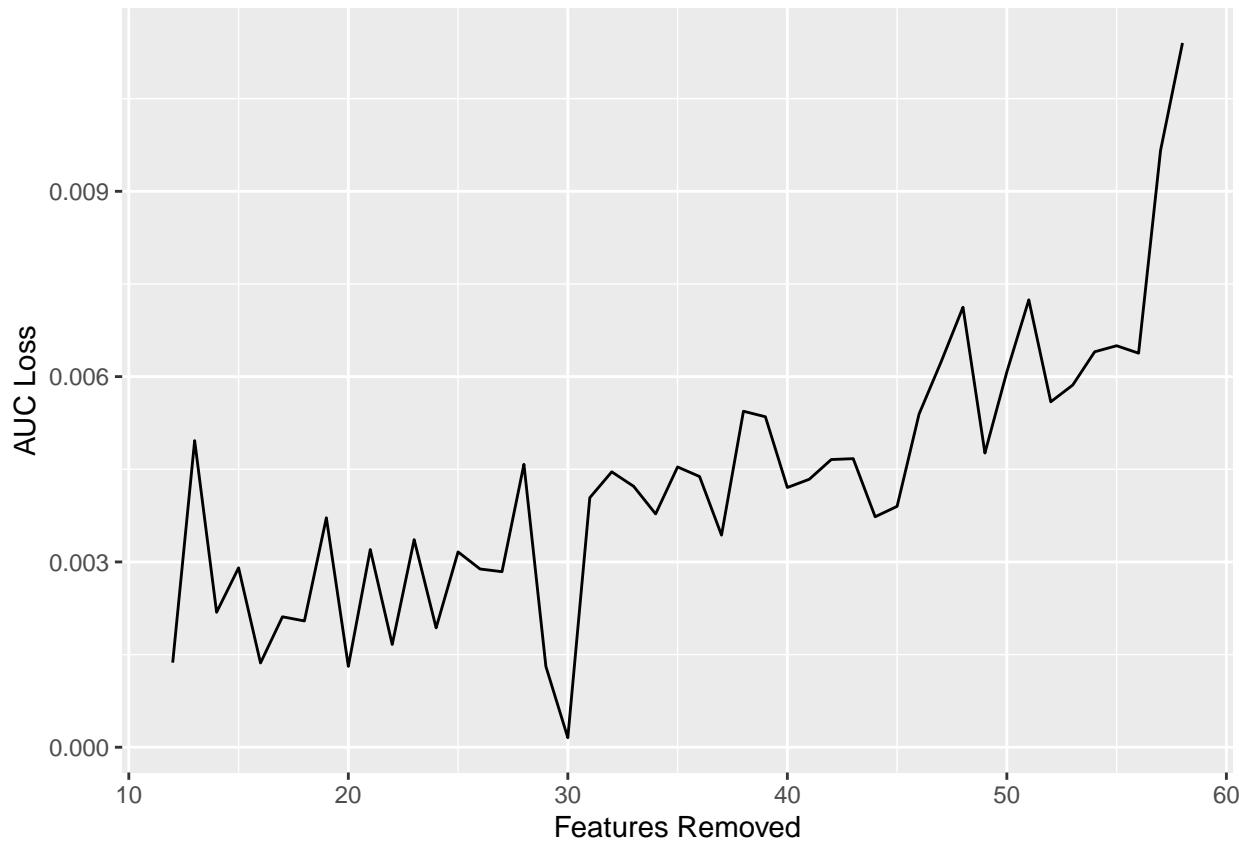
sprintf('Trimmed Model CV Train AUC: %.3f', feature_selected_model$cv_auc)

## [1] "Trimmed Model CV Train AUC: 0.698"
sprintf('Trimmed Model Test AUC: %.3f', feature_selected_model$auc)

## [1] "Trimmed Model Test AUC: 0.681"

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. laboratorio
3. admission_pre_t0_count
4. exames_imagem_qtde
5. analises_clinicas_qtde
6. equipe_multiprof
7. age
8. icu_t0
9. classe_meds_qtde
10. ecg
11. metodos_graficos_qtde
12. meds_antimicrobianos
13. psicofarmacos
14. antiaritmico
15. admission_t0_emergency
16. vasodilatador
17. diuretico
18. bloq_calcio
19. admission_pre_t0_180d
20. espironolactona
21. anticonvulsivante
22. ieca_bra
23. reop_type_1
24. proced_invasivos_qtde
25. cied_final_group_1
26. betabloqueador

```
# 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:

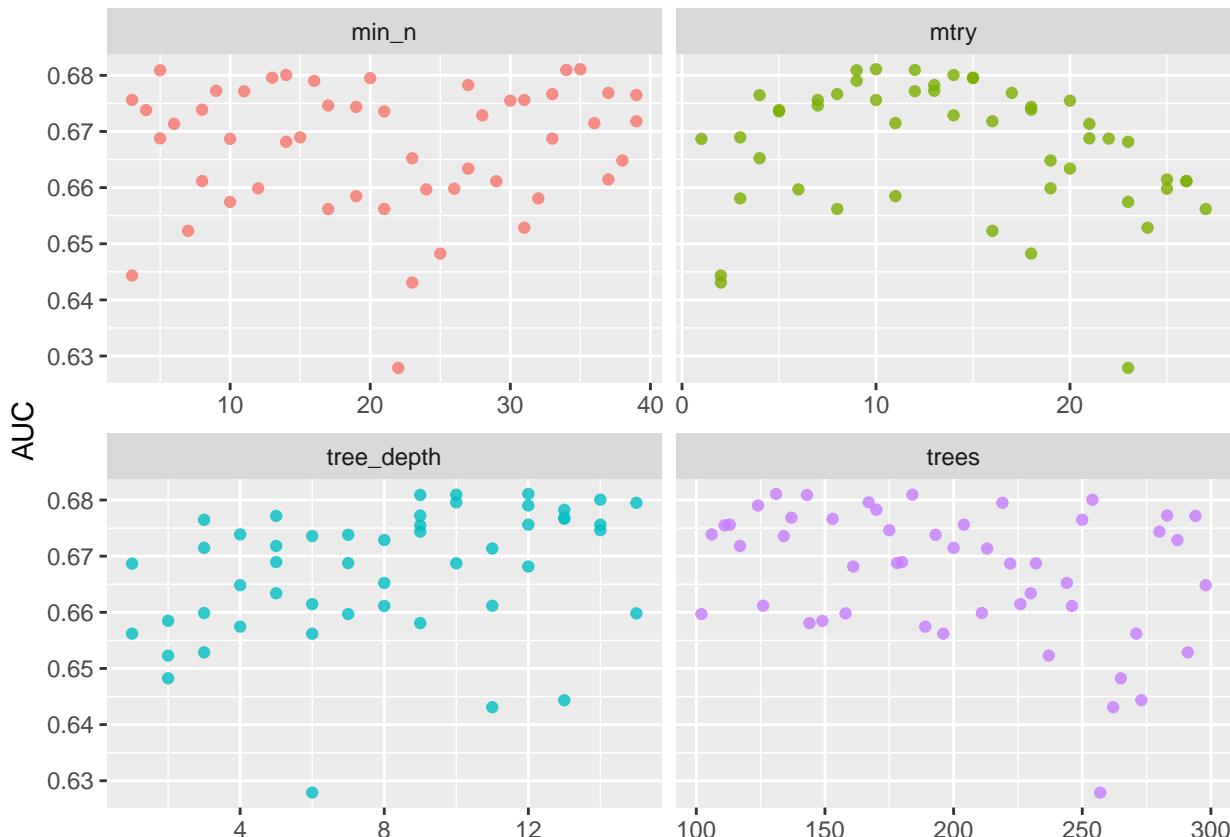
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
10	131	35	12	4e-05	0.00000	roc_auc	binary	0.68108	5	0.00968	Preprocessor1
12	184	34	10	1e-05	0.00000	roc_auc	binary	0.68096	5	0.00888	Preprocessor1
9	143	5	9	1e-05	0.03876	roc_auc	binary	0.68090	5	0.00900	Preprocessor1
14	254	14	14	0e+00	0.00376	roc_auc	binary	0.68006	5	0.01062	Preprocessor1

Table 2: (continued)

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
15	167	13	10	0e+00	0.00005	roc_auc	binary	0.67958	5	0.00963	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")
```

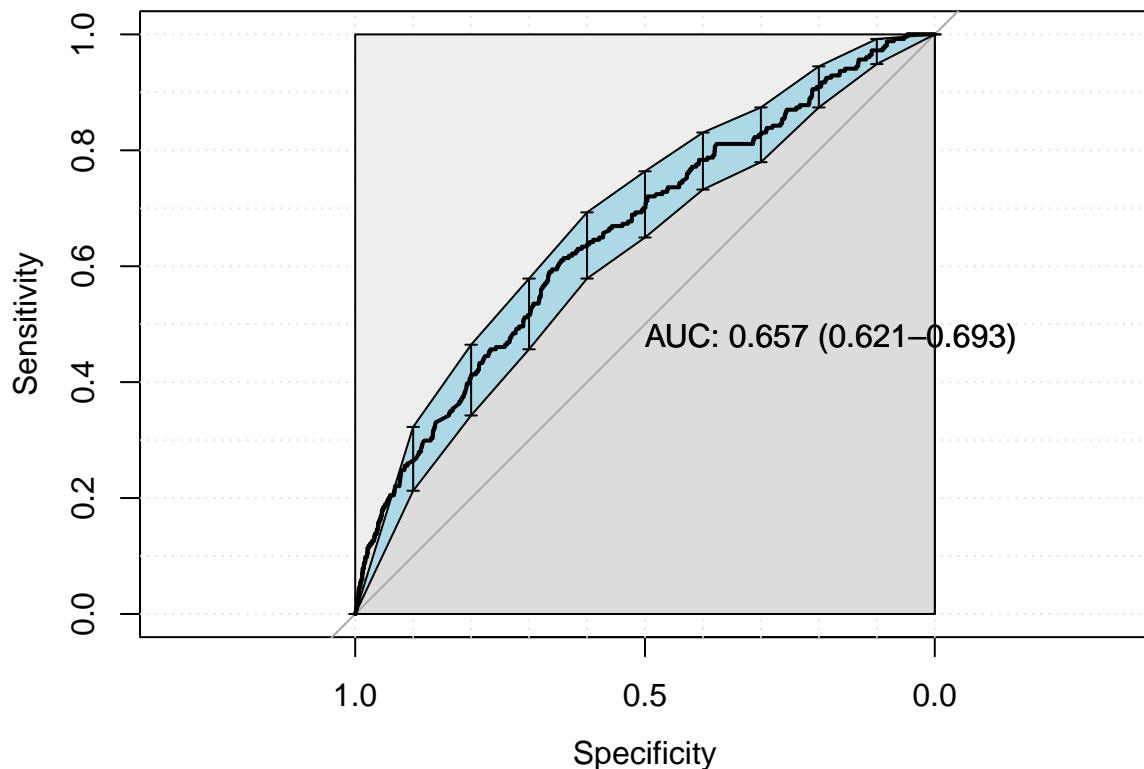


```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4324  219
##                      1 152   35
##
##          Accuracy : 0.9216
## 95% CI : (0.9135, 0.9291)
## No Information Rate : 0.9463
## P-Value [Acc > NIR] : 1.0000000
##
##          Kappa : 0.1186
##
## McNemar's Test P-Value : 0.0006113
##
##          Sensitivity : 0.9660
##          Specificity  : 0.1378
## Pos Pred Value  : 0.9518
## Neg Pred Value  : 0.1872
## Prevalence       : 0.9463
## Detection Rate  : 0.9142
## Detection Prevalence: 0.9605
## Balanced Accuracy : 0.5519
##
## '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:

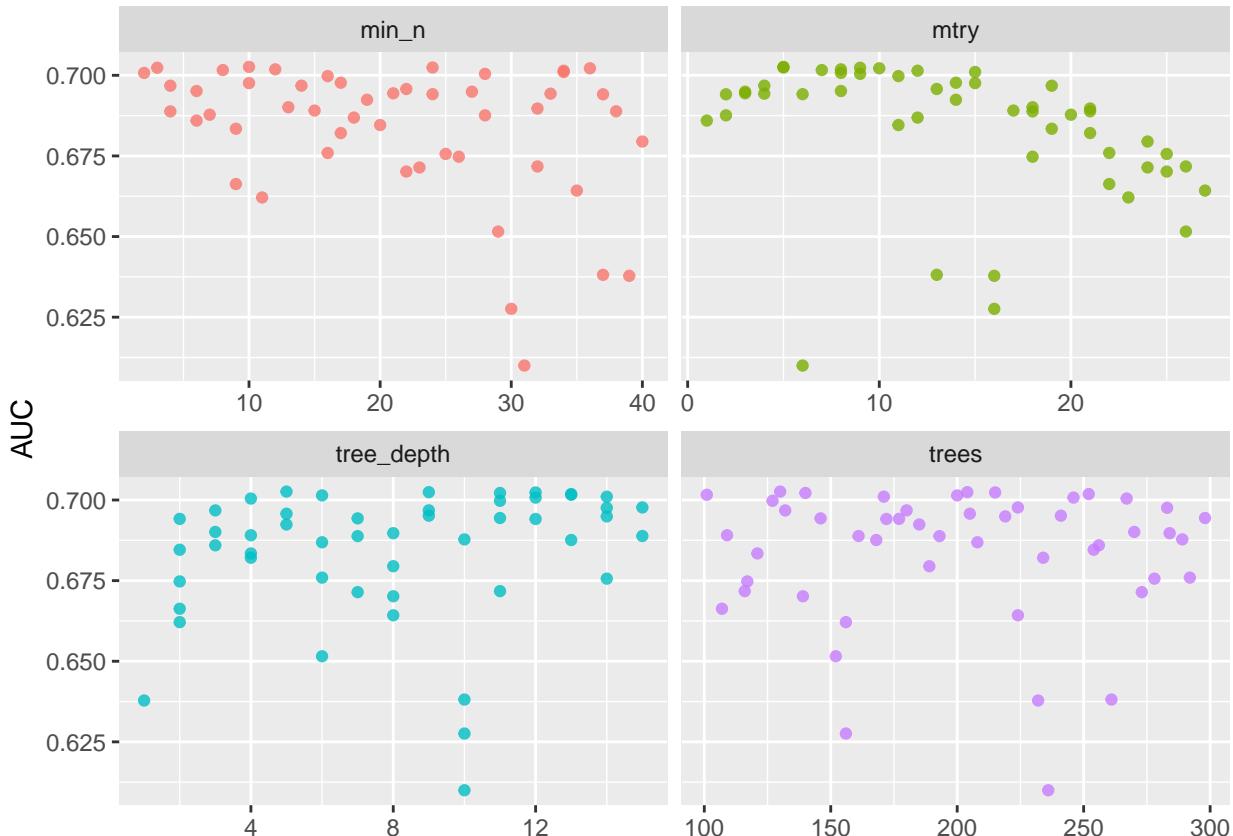
mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
5	130	10	5	0.00000	3.40764	roc_auc	binary	0.70261	5	0.00794	Preprocessor1

Table 3: (continued)

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
5	204	24	9	0.00074	0.01322	roc_auc	binary	0.70243	5	0.00526	Preprocessor1
9	215	3	12	0.00000	0.00000	roc_auc	binary	0.70235	5	0.00769	Preprocessor1
10	140	36	11	0.00000	0.01073	roc_auc	binary	0.70218	5	0.00661	Preprocessor1
8	252	12	13	0.00009	9.32948	roc_auc	binary	0.70185	5	0.01043	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)

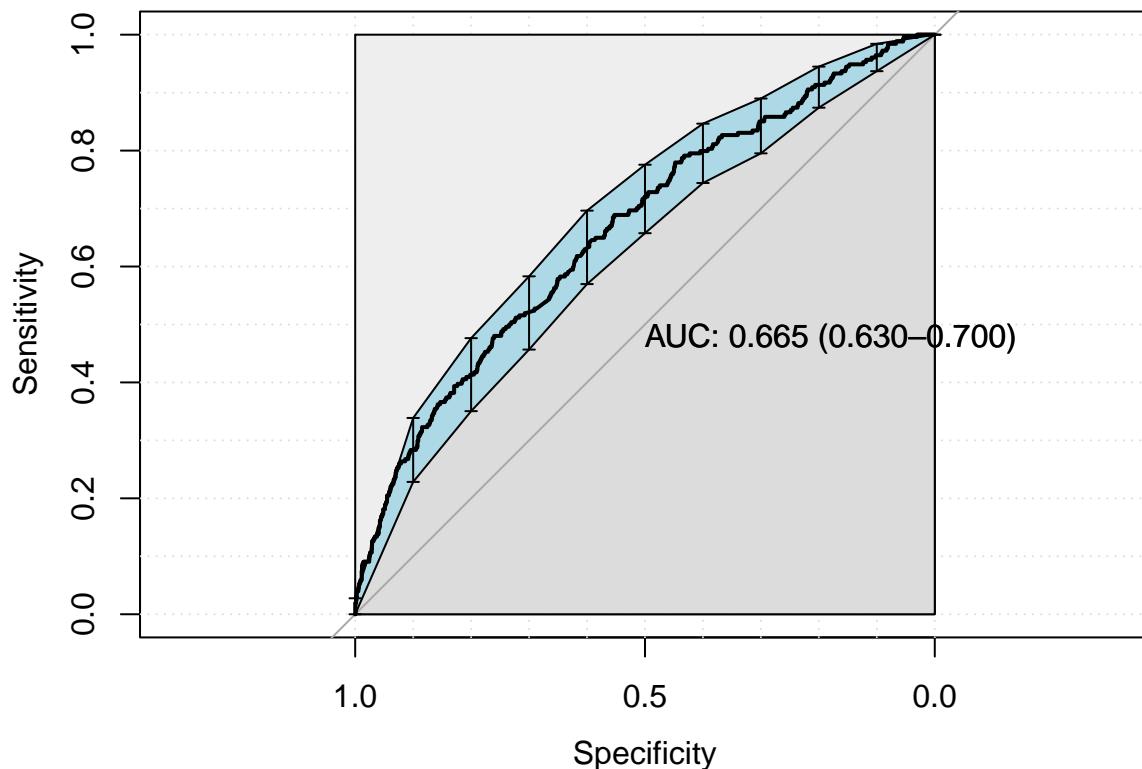
last_lightgbm_fit <-
  final_lightgbm_workflow %>%
  last_fit(df_split)
```

```

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 3498 141
##                      1  978 113
##
##          Accuracy : 0.7634
##          95% CI : (0.751, 0.7755)
##  No Information Rate : 0.9463
##  P-Value [Acc > NIR] : 1
##
##          Kappa : 0.0886
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.7815
##          Specificity : 0.4449
##  Pos Pred Value : 0.9613
##  Neg Pred Value : 0.1036
##          Prevalence : 0.9463
##  Detection Rate : 0.7395
## Detection Prevalence : 0.7693
##  Balanced Accuracy : 0.6132
##
## '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)

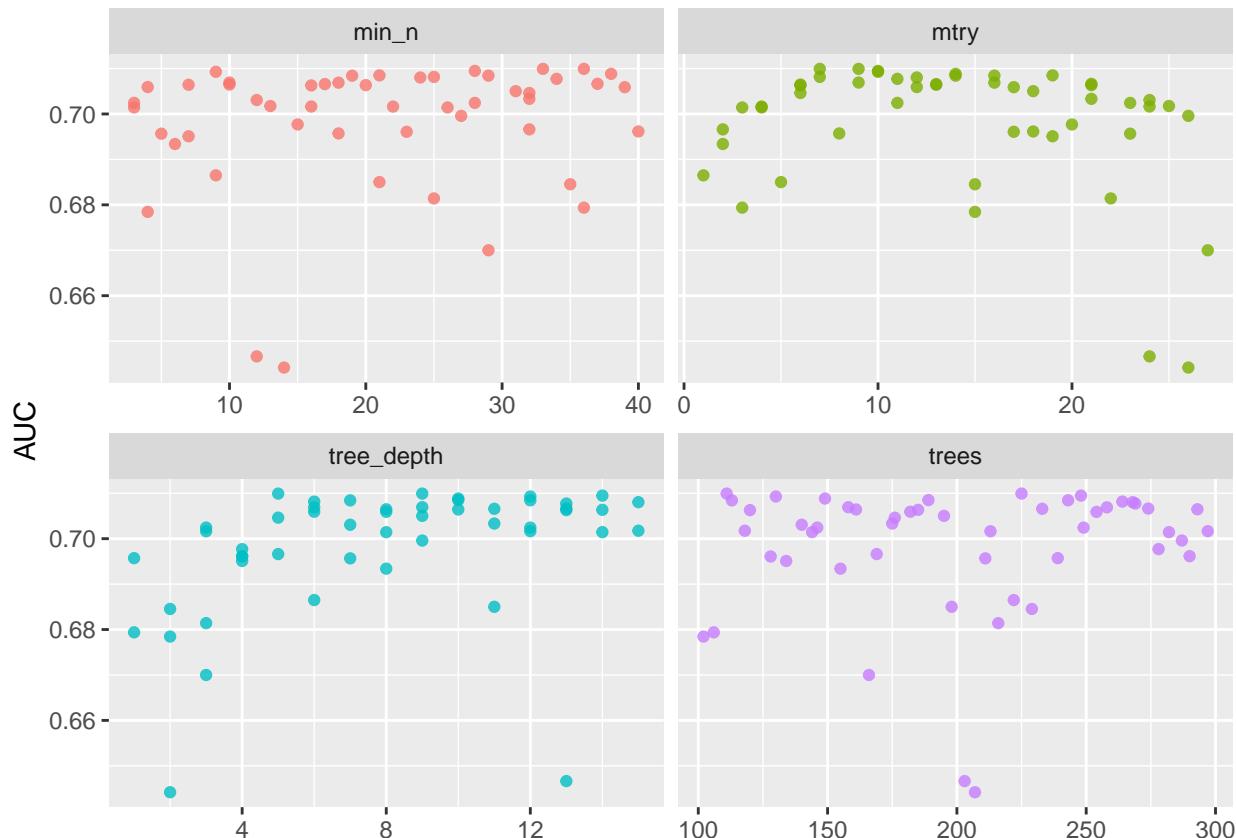
```

Table 4:

mtry	trees	min_n	tree_depth	learn_rate	loss_reduction	.metric	.estimator	mean	n	std_err	.config
7	111	36	9	0.00001	0.00000	roc_auc	binary	0.70993	5	0.00727	Preprocessor1
9	225	33	5	0.01487	0.00000	roc_auc	binary	0.70992	5	0.01139	Preprocessor1
10	248	28	14	0.00000	0.84964	roc_auc	binary	0.70948	5	0.00724	Preprocessor1
10	130	9	12	0.00714	0.00009	roc_auc	binary	0.70929	5	0.00796	Preprocessor1
14	149	38	10	0.00000	0.00086	roc_auc	binary	0.70884	5	0.00720	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")
```

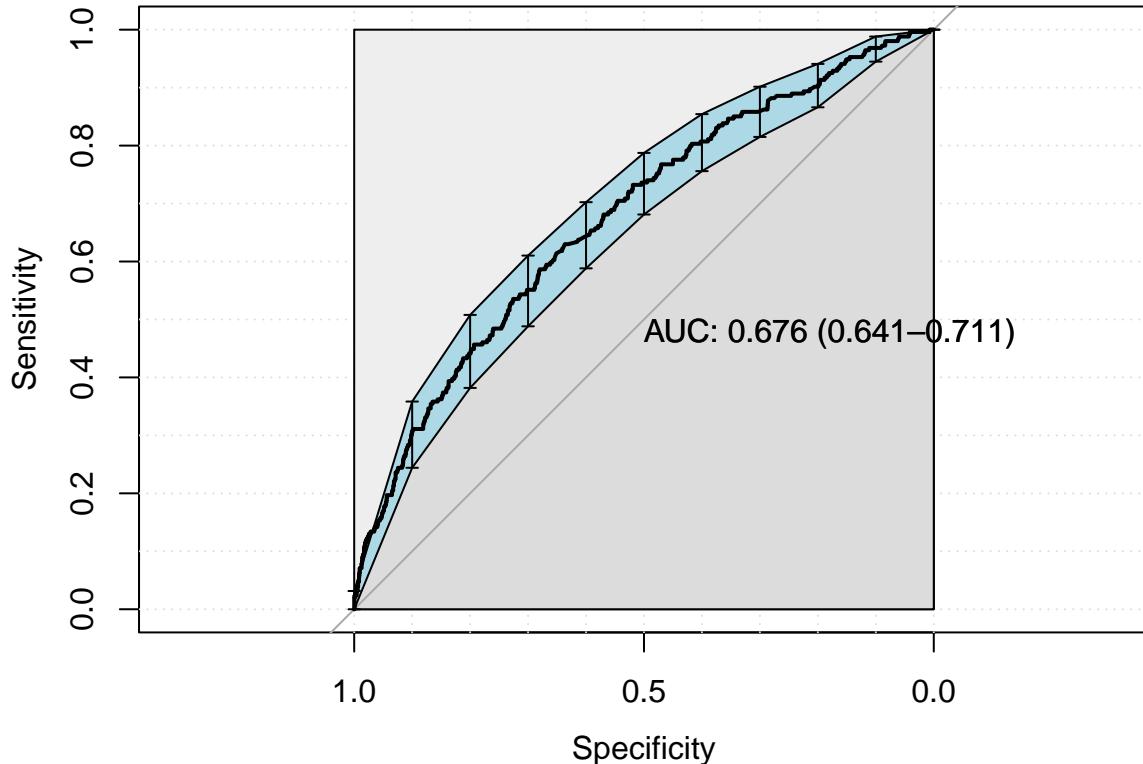


```
final_lightgbm_workflow <-
  lightgbm_workflow %>%
  finalize_workflow(best_lightgbm)

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 4476  254
##                      1     0     0
##
##          Accuracy : 0.9463
##          95% CI : (0.9395, 0.9526)
##  No Information Rate : 0.9463
##  P-Value [Acc > NIR] : 0.5167
##
##          Kappa : 0
##
## McNemar's Test P-Value : <2e-16
##
##          Sensitivity : 1.0000
##          Specificity  : 0.0000
##  Pos Pred Value : 0.9463
##  Neg Pred Value :      NaN
##          Prevalence : 0.9463
##  Detection Rate : 0.9463
## 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)

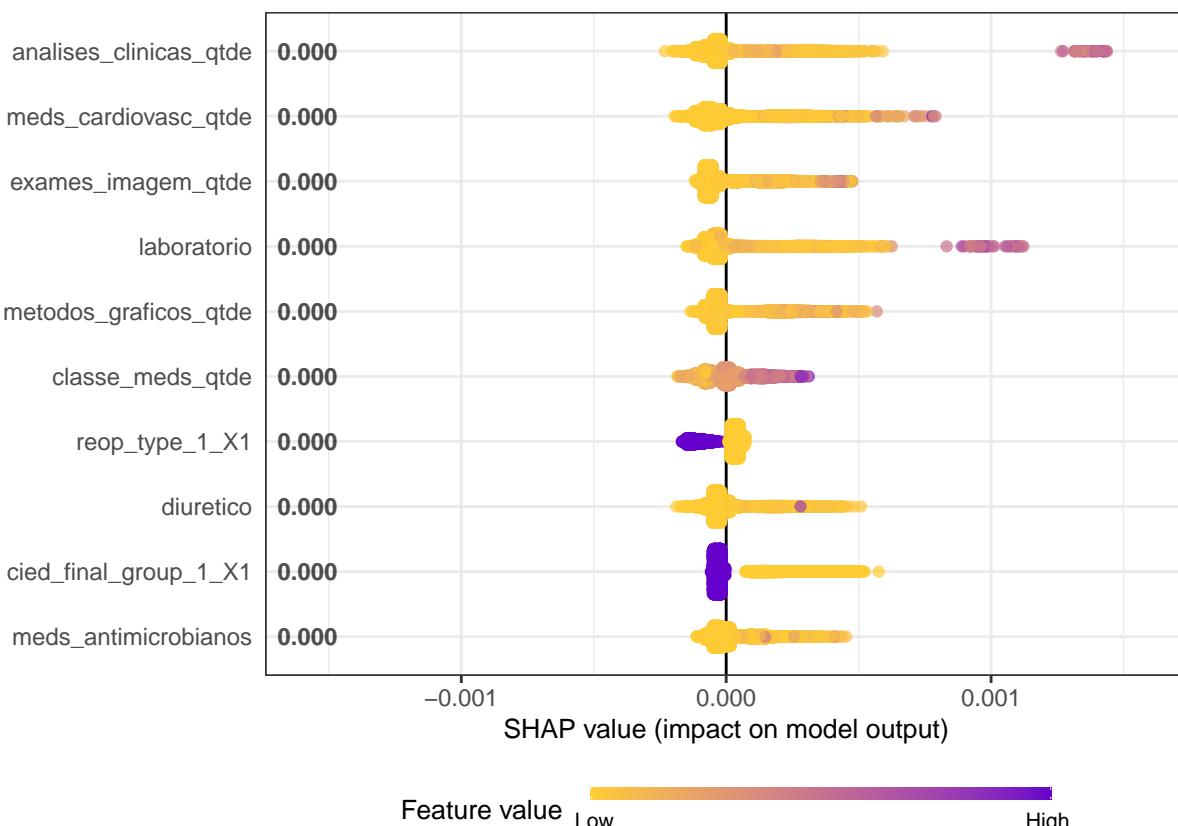
trained_rec <- prep(lightgbm_recipe, training = df_train)

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,

```

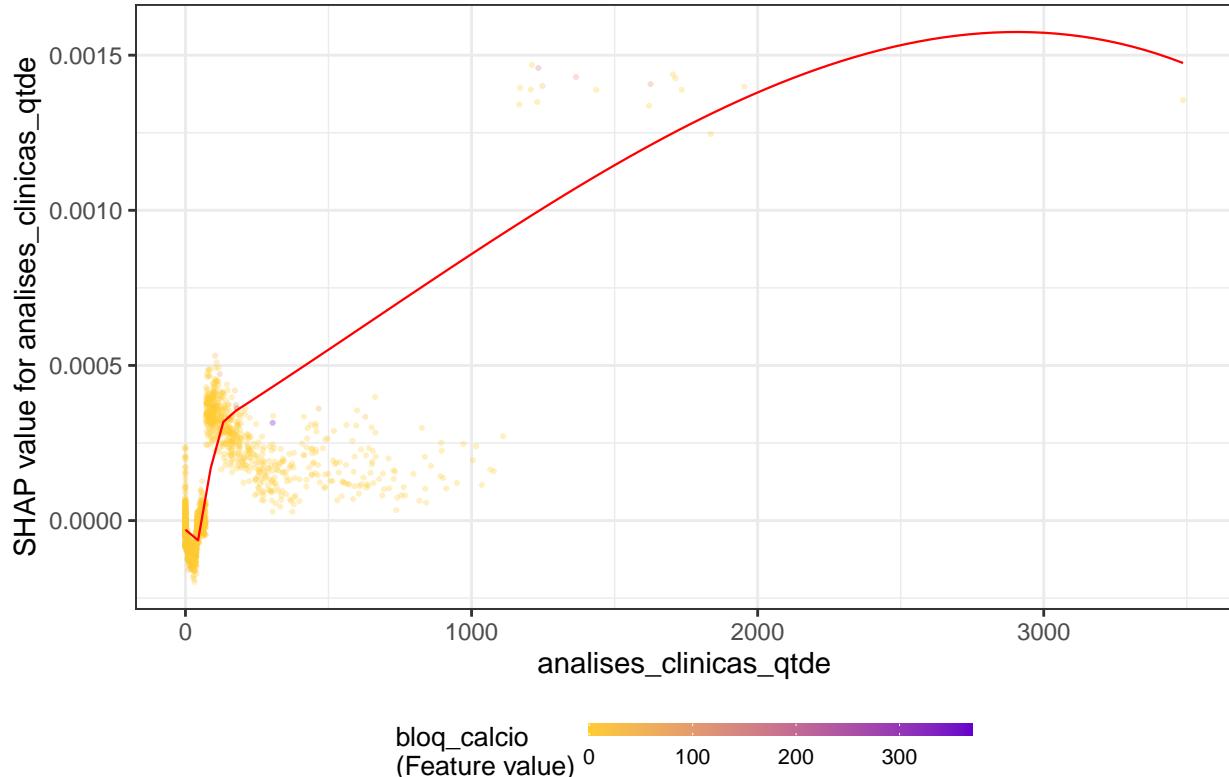
```

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

```

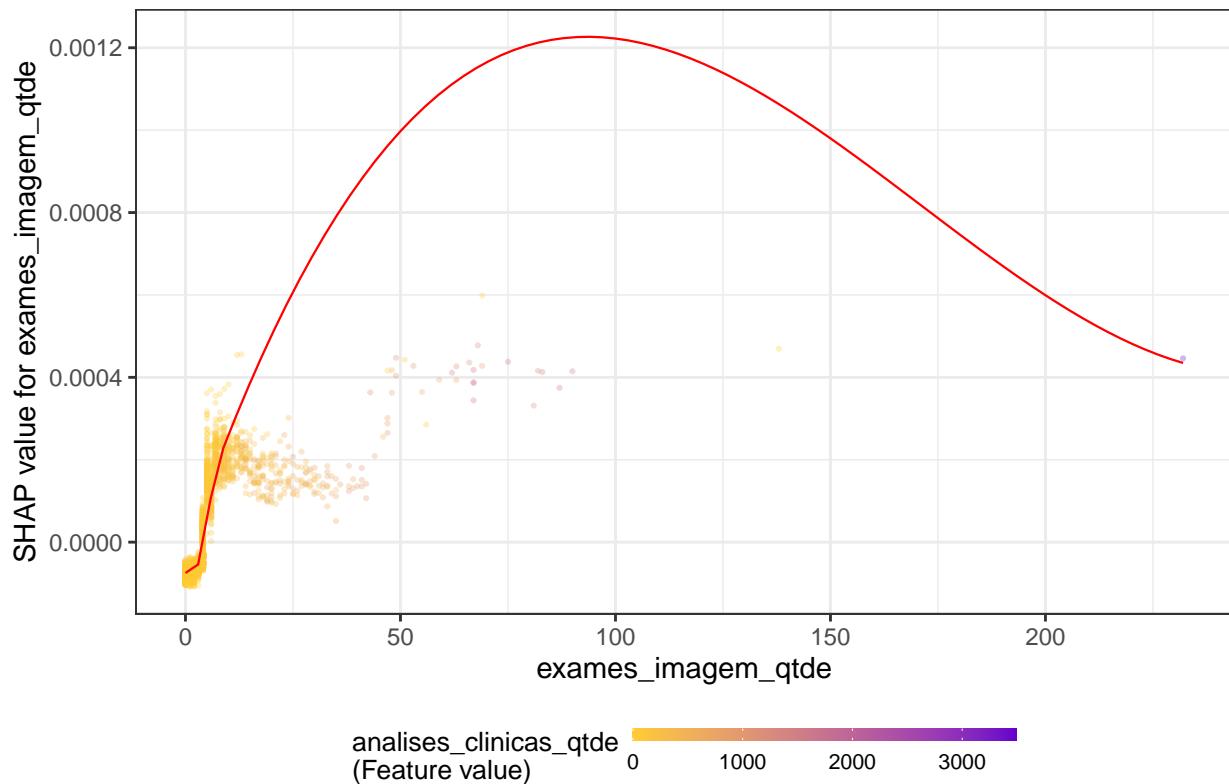
```
## `geom_smooth()` using formula 'y ~ x'
```

analises_clinicas_qtde



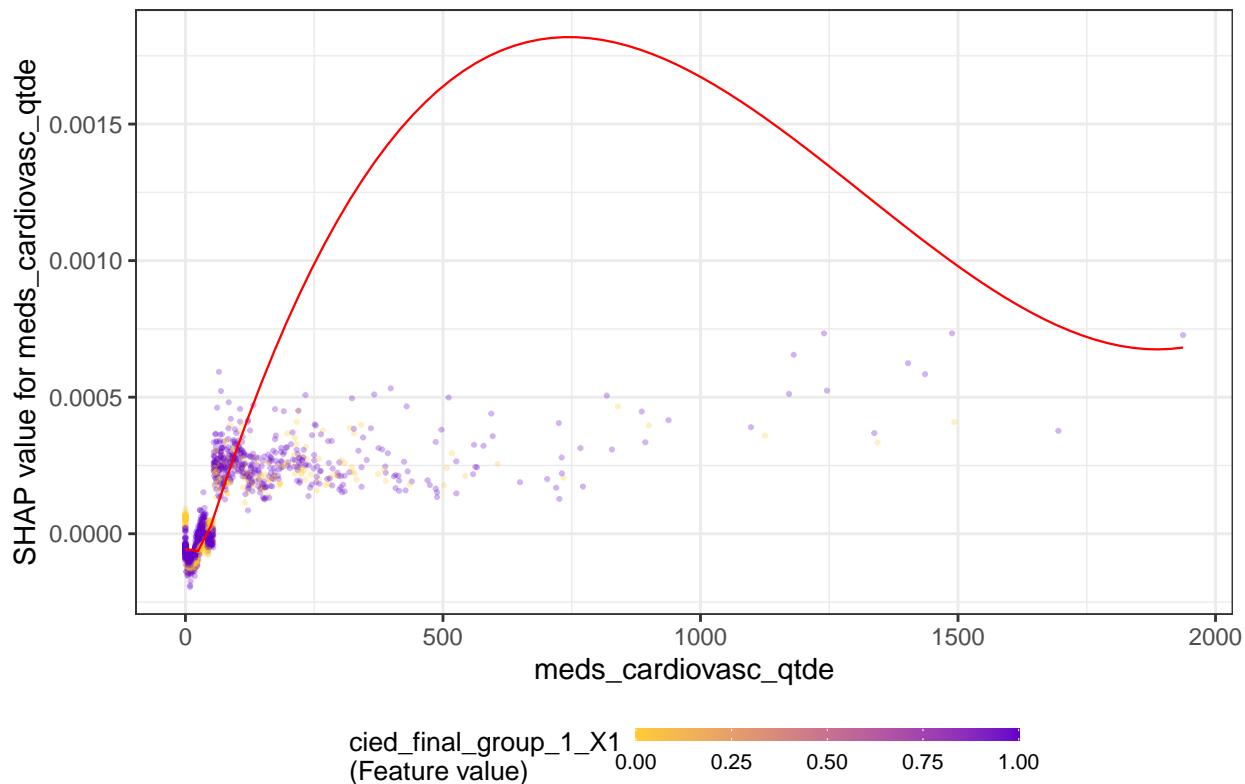
```
## `geom_smooth()` using formula 'y ~ x'
```

exames_imagem_qtde



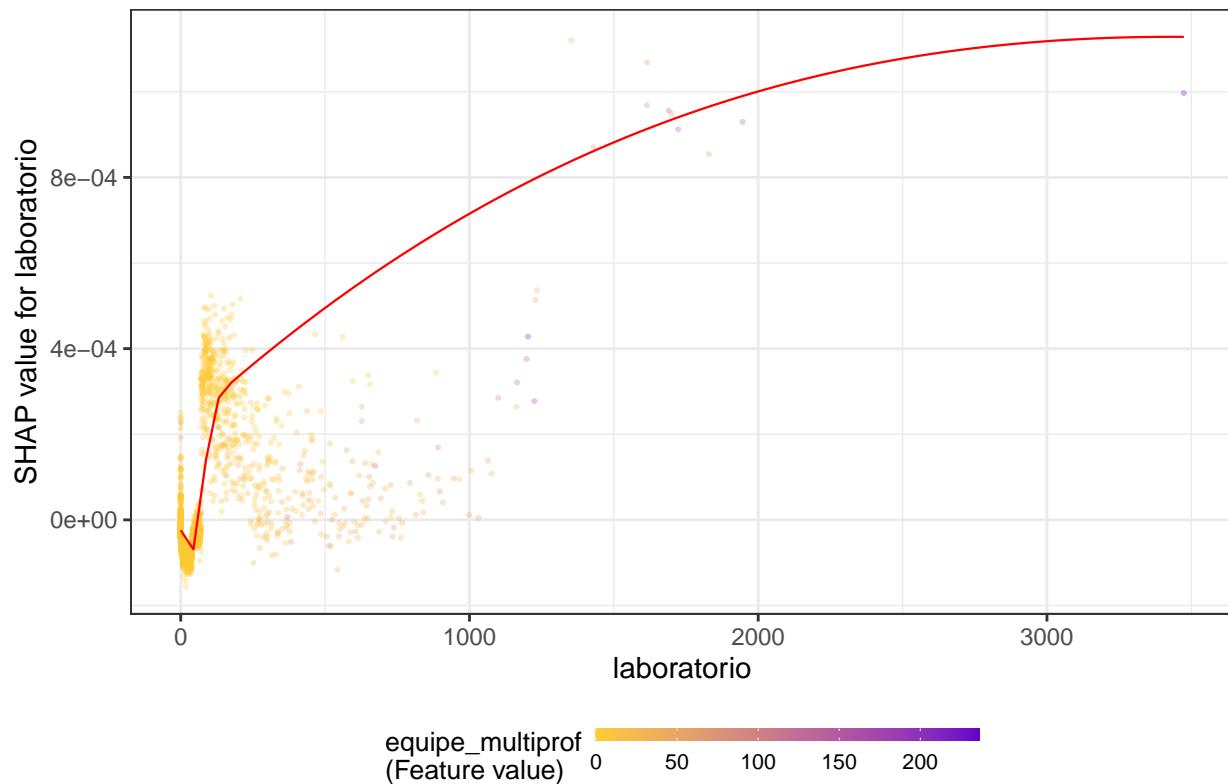
```
## `geom_smooth()` using formula 'y ~ x'
```

meds_cardiovasc_qtde



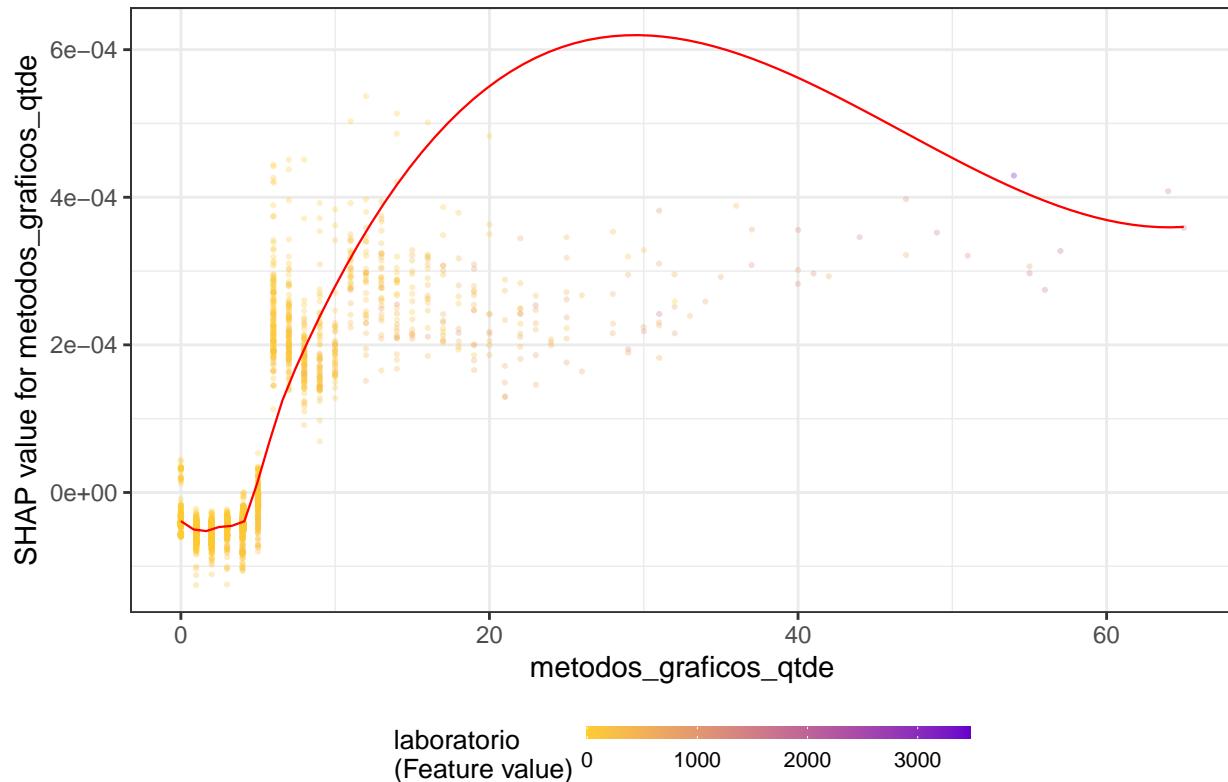
```
## `geom_smooth()` using formula 'y ~ x'
```

laboratorio



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

metodos_graficos_qtde



Models Comparison

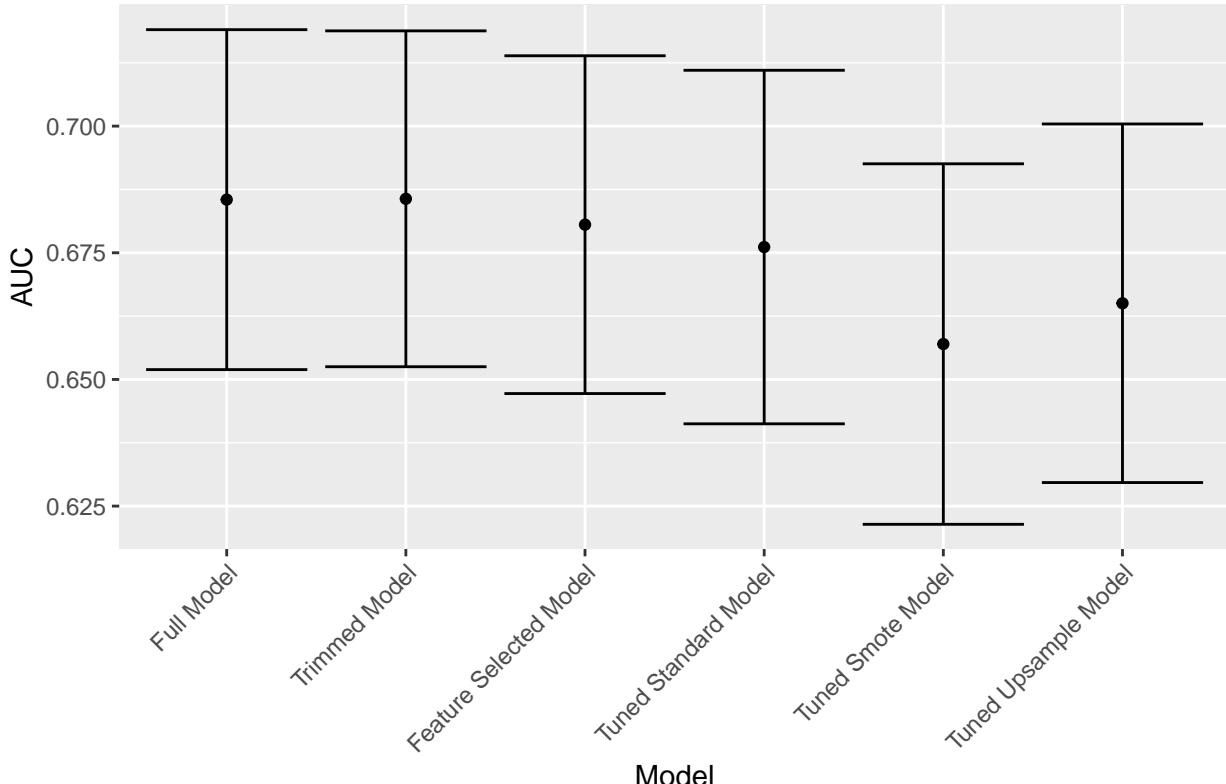
```

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_60d



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