Final Model - readmission_60d

Eduardo Yuki Yada

Global parameters

```
k <- params$k # Number of folds for cross validation
grid_size <- params$grid_size # Number of parameter combination to tune on each model
max_auc_loss <- params$max_auc_loss # Max accepted loss of AUC for reducing num of features
repeats <- params$repeats
Hmisc::list.tree(params)
```

```
##  params = list 5 (952 bytes)
## .  max_auc_loss = double 1= 0.01
## .  outcome_column = character 1= readmission_60d
## .  k = double 1= 10
## .  grid_size = double 1= 50
## .  repeats = double 1= 2
```

Imports

```
library(tidyverse)
library(yaml)
library(tidymodels)
library(usemodels)
library(vip)
library(kableExtra)
library(SHAPforxgboost)
library(xgboost)
library(Matrix)
library(mltools)
library(bonsai)
library(lightgbm)
library(pROC)
library(caret)
library(themis)

source("aux_functions.R")

select <- dplyr::select
predict <- stats::predict
```

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
```

```r
df[columns_list$outcome_columns] <- lapply(df[columns_list$outcome_columns], factor)
df <- mutate(df, across(where(is.character), as.factor))
```

```r
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)

dir.create(file.path("./auxiliar/final_model/auroc_plots/"),
           showWarnings = FALSE,
           recursive = TRUE)

dir.create(file.path("./auxiliar/final_model/shap_plots/"),
           showWarnings = FALSE,
           recursive = TRUE)
```

Eligible features

```r
cat_features_list = read_yaml(sprintf(
  "./auxiliar/significant_columns/categorical_%s.yaml",
  outcome_column
))

num_features_list = read_yaml(sprintf(
  "./auxiliar/significant_columns/numerical_%s.yaml",
  outcome_column
))

features_list = c(cat_features_list, num_features_list)
```

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

exception_columns <- c('death_intraop', 'death_intraop_1', 'disch_outcomes_t0')

correlated_columns = c('year_procedure_1', # com year_adm_t0
                       'age_surgery_1', # com age
                       'admission_t0', # com admission_pre_t0_count
                       'atb', # com meds_antimicrobianos
                       'classe_meds_cardio_qtde', # com classe_meds_qtde
                       'suporte_hemod', # com proced_invasivos_qtde,
                       'radiografia', # com exames_imagem_qtde
                       'ecg' # com metodos_graficos_qtde
                       )

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

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. ventilacao_mecanica
47. cec
48. transplante_cardiaco
49. cir_toracica
50. outros_proced_cirurgicos
51. icp
52. angioplastia
53. cateterismo
54. eletrofisiologia
55. cateter_venoso_central
56. proced_invasivos_qtde
57. cve_desf
58. transfusao
59. interconsulta

60. equipe_multiprof
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. ressonancia
82. exames_imagem_qtde
83. bic
84. hospital_stay

Train test split (70%/30%)

```
set.seed(42)

if (outcome_column == 'readmission_30d') {
  df_split <- readRDS("dataset/split_object.rds")
} else {
  df_split <- initial_split(df, prop = .7, strata = all_of(outcome_column))
}

df_train <- training(df_split) %>% select(all_of(c(features, outcome_column)))
df_test <- testing(df_split) %>% select(all_of(c(features, outcome_column)))

df_folds <- vfold_cv(df_train, v = k,
                     strata = all_of(outcome_column),
                     repeats = repeats)
```

Feature Selection

```
custom_dummy_names <- function(var, lvl, ordinal = FALSE) {
  dummy_names(var, lvl, ordinal = FALSE, sep = "__")
}

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_dummy(all_nominal_predictors(), naming = custom_dummy_names)

  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) %>%
    separate(Feature, c("Feature","value"), "__", fill = 'right') %>%
    group_by(Feature) %>%
    summarise(Gain = sum(Gain),
              Cover = sum(Cover),
              Frequency = sum(Frequency)) %>%
    ungroup() %>%
    arrange(desc(Gain))

  cv_results <- collect_metrics(model_fit_rs) %>% filter(.metric == 'roc_auc')

  return(
    list(
      cv_auc =  cv_results$mean,
      cv_auc_std_err = cv_results$std_err,
      importance = feature_importance,
      auc = as.numeric(model_auc$auc),
      auc_lower = model_auc$ci[1],
      auc_upper = model_auc$ci[3]
    )
  )
}
```

```
hyperparameters <- read_yaml(
  sprintf(
    "./auxiliar/model_selection/hyperparameters/%s.yaml",
    outcome_column
  )
)

hyperparameters$sample_size <- 1

full_model <- model_fit_wf(df_train, features, outcome_column, hyperparameters)

sprintf('Full Model CV Train AUC: %.3f' ,full_model$cv_auc)
```

```
## [1] "Full Model CV Train AUC: 0.710"
```

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

```
## [1] "Full Model Test AUC: 0.682"
```

Features with zero importance on the initial model:

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

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

1. heart_failure
2. af
3. cardiac_arrest
4. transplant
5. valvopathy
6. diabetes
7. hemodialysis
8. dialysis_t0
9. antiplaquetario_ev
10. antifungico
11. cec
12. transplante_cardiaco
13. cir_toracica
14. outros_proced_cirurgicos
15. icp
16. angioplastia
17. cateterismo
18. eletrofisiologia
19. cateter_venoso_central
20. cve_desf
21. teste_esforco
22. espiro_ergoespiro
23. tilt_teste
24. cultura
25. citologia
26. biopsia
27. angio_rm
28. arteriografia
29. pet_ct
30. ressonancia

```r
trimmed_features <- full_model$importance$Feature
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: 0.711"
```

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

```
## [1] "Trimmed Model Test AUC: 0.682"
```

```r
selection_results <- tibble::tribble(
  ~`Tested Feature`, ~`Dropped`, ~`Number of Features`, ~`CV AUC`, ~`CV AUC Std Error`, ~`Total AUC Loss`, ~`Ins
  'None', TRUE, length(features), full_model$cv_auc, full_model$cv_auc_std_err, 0, 0
)

whitelist <- c()

if (full_model$cv_auc - trimmed_model$cv_auc < max_auc_loss) {
  current_features <- trimmed_features
  current_model <- trimmed_model
  current_auc_loss <- full_model$cv_auc - current_model$cv_auc
  instant_auc_loss <- full_model$cv_auc - current_model$cv_auc

  selection_results <- selection_results %>%
    add_row(`Tested Feature` = 'All unimportant',
```

```
            `Dropped` = TRUE,
            `Number of Features` = length(trimmed_features),
            `CV AUC` = current_model$cv_auc,
            `CV AUC Std Error` = current_model$cv_auc_std_err,
            `Total AUC Loss` = current_auc_loss,
            `Instant AUC Loss` = instant_auc_loss)
} else {
  current_features <- features
  current_model <- full_model
  current_auc_loss <- 0
}

while (current_auc_loss < max_auc_loss & mean(current_features %in% whitelist) < 1) {
  zero_importance_features <-
    setdiff(current_features, current_model$importance$Feature) %>%
    setdiff(whitelist)
  if (length(zero_importance_features) > 0) {
    current_least_important <- zero_importance_features[1]
  } else {
    current_least_important <-
      tail(setdiff(current_model$importance$Feature, whitelist), 1)
  }
  test_features <-
    setdiff(current_features, current_least_important)
  current_model <-
    model_fit_wf(df_train, test_features, outcome_column, hyperparameters)
  instant_auc_loss <-
    tail(selection_results %>% filter(Dropped) %>% .$`CV AUC`, n = 1) - current_model$cv_auc

  if (instant_auc_loss < max_auc_loss / 5 &
      current_auc_loss < max_auc_loss) {
    dropped <- TRUE
    current_features <- test_features
    current_auc_loss <- full_model$cv_auc - current_model$cv_auc
  } else {
    dropped <- FALSE
    whitelist <- c(whitelist, current_least_important)
  }

  selection_results <- selection_results %>%
    add_row(
      `Tested Feature` = current_least_important,
      `Dropped` = dropped,
      `Number of Features` = length(test_features),
      `CV AUC` = current_model$cv_auc,
      `CV AUC Std Error` = current_model$cv_auc_std_err,
      `Total AUC Loss` = current_auc_loss,
      `Instant AUC Loss` = instant_auc_loss
    )

  print(c(
    length(current_features),
    round(current_auc_loss, 4),
    round(instant_auc_loss, 4),
    current_least_important
  ))
}
```

```
## [1] "53"          "-0.001"      "0"           "reop_type_1"
## [1] "52"                  "-9e-04"              "1e-04"
## [4] "ventilacao_mecanica"
```

```
## [1] "51"         "-4e-04"     "5e-04"      "endoscopia"
## [1] "50"       "-8e-04"   "-4e-04"   "prior_mi"
## [1] "49"            "-8e-04"        "0"             "interconsulta"
## [1] "48"         "-0.0011"    "-3e-04"     "transfusao"
## [1] "47"         "-0.0015"    "-4e-04"     "nyha_basal"
## [1] "46"         "-0.0016"    "-2e-04"     "tomografia"
## [1] "45"       "-0.0018"  "-2e-04"   "angio_tc"
## [1] "44"        "-0.0023"   "-5e-04"    "antiviral"
## [1] "43"                 "-0.0016"            "7e-04"
## [4] "cied_final_group_1"
## [1] "42"      "-0.0027" "-0.001"  "bic"
## [1] "41"      "-0.0029" "-2e-04"  "holter"
## [1] "40"                      "-0.0033"
## [3] "-4e-04"                  "underlying_heart_disease"
## [1] "39"      "-0.0036" "-3e-04"  "aco"
## [1] "38"            "-0.0034"       "2e-04"         "cintilografia"
## [1] "37"       "-0.0029"  "4e-04"    "insulina"
## [1] "36"                 "-0.0029"            "1e-04"
## [4] "procedure_type_new"
## [1] "35"              "-0.0039"         "-0.001"          "espironolactona"
## [1] "34"        "-0.0045"   "-6e-04"    "ultrassom"
## [1] "33"             "-0.0046"        "-1e-04"         "insuf_cardiaca"
## [1] "32"       "-0.005"   "-4e-04"   "estatina"
## [1] "31"             "-0.0049"        "1e-04"          "ecocardiograma"
## [1] "30"                  "-0.005"             "-1e-04"
## [4] "comorbidities_count"
## [1] "29"            "-0.0049"       "1e-04"         "psicofarmacos"
## [1] "28"                     "-0.0052"               "-3e-04"
## [4] "analises_clinicas_qtde"
## [1] "27"                    "-0.006"               "-8e-04"
## [4] "metodos_graficos_qtde"
## [1] "26"           "-0.0063"      "-3e-04"       "cied_final_1"
## [1] "25"             "-0.0056"        "7e-04"          "betabloqueador"
## [1] "24"            "-0.0052"       "4e-04"         "heart_disease"
## [1] "23"                  "-0.0058"             "-6e-04"
## [4] "histopatologia_qtde"
## [1] "22"      "-0.0053" "5e-04"   "dva"
## [1] "21"               "-0.005"           "3e-04"            "procedure_type_1"
## [1] "20"       "-0.005"   "0"        "digoxina"
## [1] "19"                    "-0.0045"               "5e-04"
## [4] "proced_invasivos_qtde"
## [1] "18"          "-0.005"      "-5e-04"      "laboratorio"
## [1] "17"            "-0.0051"       "-1e-04"        "vasodilatador"
## [1] "16"       "-0.0056"  "-5e-04"   "ieca_bra"
## [1] "15"                     "-0.0052"               "4e-04"
## [4] "admission_t0_emergency"
## [1] "14"                    "-0.005"               "2e-04"
## [4] "admission_pre_t0_180d"
## [1] "13"                "-0.005"            "1e-04"
## [4] "anticonvulsivante"
## [1] "12"                   "-0.0067"              "-0.0018"
## [4] "meds_cardiovasc_qtde"
## [1] "11"          "-0.0056"     "0.0012"      "bloq_calcio"
## [1] "10"                 "-0.0066"             "-0.001"
## [4] "exames_imagem_qtde"
## [1] "9"             "-0.0051"       "0.0015"        "antiarritmico"
## [1] "8"         "-0.0041"   "0.001"     "diuretico"
## [1] "7"       "-0.0048" "-7e-04"  "icu_t0"
## [1] "6"               "-0.0045"         "3e-04"           "education_level"
## [1] "6"                "-0.0045"          "0.002"             "equipe_multiprof"
## [1] "5"                   "-0.0055"              "-0.001"
```

```
## [4] "meds_antimicrobianos"
## [1] "4"       "-0.007"  "-0.0015" "age"
## [1] "4"                 "-0.007"           "0.0095"           "classe_meds_qtde"
## [1] "4"                      "-0.007"                "0.0095"
## [4] "admission_pre_t0_count"
## [1] "4"             "-0.007"        "0.0444"        "hospital_stay"
```

```
selection_results %>%
  rename(Features = `Number of Features`) %>%
  niceFormatting(digits = 4, label = 1)
```

Table 1:

Tested Feature	Dropped	Features	CV AUC	CV AUC Std Error	Total AUC Loss	Instant AUC Loss
None	TRUE	84	0.7104	0.0095	0.0000	0.0000
All unimportant	TRUE	54	0.7114	0.0094	-0.0010	-0.0010
reop_type_1	TRUE	53	0.7114	0.0094	-0.0010	0.0000
ventilacao_mecanica	TRUE	52	0.7113	0.0094	-0.0009	0.0001
endoscopia	TRUE	51	0.7108	0.0094	-0.0004	0.0005
prior_mi	TRUE	50	0.7112	0.0091	-0.0008	-0.0004
interconsulta	TRUE	49	0.7112	0.0092	-0.0008	0.0000
transfusao	TRUE	48	0.7115	0.0093	-0.0011	-0.0003
nyha_basal	TRUE	47	0.7119	0.0094	-0.0015	-0.0004
tomografia	TRUE	46	0.7120	0.0095	-0.0016	-0.0002
angio_tc	TRUE	45	0.7122	0.0094	-0.0018	-0.0002
antiviral	TRUE	44	0.7127	0.0094	-0.0023	-0.0005
cied_final_group_1	TRUE	43	0.7120	0.0094	-0.0016	0.0007
bic	TRUE	42	0.7131	0.0093	-0.0027	-0.0010
holter	TRUE	41	0.7133	0.0093	-0.0029	-0.0002
underlying_heart_disease	TRUE	40	0.7137	0.0093	-0.0033	-0.0004
aco	TRUE	39	0.7140	0.0093	-0.0036	-0.0003
cintilografia	TRUE	38	0.7138	0.0093	-0.0034	0.0002
insulina	TRUE	37	0.7133	0.0093	-0.0029	0.0004
procedure_type_new	TRUE	36	0.7133	0.0092	-0.0029	0.0001
espironolactona	TRUE	35	0.7143	0.0090	-0.0039	-0.0010
ultrassom	TRUE	34	0.7149	0.0092	-0.0045	-0.0006
insuf_cardiaca	TRUE	33	0.7150	0.0091	-0.0046	-0.0001
estatina	TRUE	32	0.7154	0.0093	-0.0050	-0.0004
ecocardiograma	TRUE	31	0.7153	0.0091	-0.0049	0.0001
comorbidities_count	TRUE	30	0.7154	0.0093	-0.0050	-0.0001
psicofarmacos	TRUE	29	0.7153	0.0092	-0.0049	0.0001
analises_clinicas_qtde	TRUE	28	0.7157	0.0091	-0.0052	-0.0003
metodos_graficos_qtde	TRUE	27	0.7164	0.0091	-0.0060	-0.0008
cied_final_1	TRUE	26	0.7167	0.0090	-0.0063	-0.0003
betabloqueador	TRUE	25	0.7160	0.0091	-0.0056	0.0007
heart_disease	TRUE	24	0.7156	0.0093	-0.0052	0.0004
histopatologia_qtde	TRUE	23	0.7162	0.0091	-0.0058	-0.0006
dva	TRUE	22	0.7157	0.0090	-0.0053	0.0005
procedure_type_1	TRUE	21	0.7154	0.0094	-0.0050	0.0003
digoxina	TRUE	20	0.7154	0.0094	-0.0050	0.0000
proced_invasivos_qtde	TRUE	19	0.7149	0.0092	-0.0045	0.0005
laboratorio	TRUE	18	0.7154	0.0092	-0.0050	-0.0005
vasodilatador	TRUE	17	0.7155	0.0092	-0.0051	-0.0001
ieca_bra	TRUE	16	0.7160	0.0094	-0.0056	-0.0005
admission_t0_emergency	TRUE	15	0.7156	0.0096	-0.0052	0.0004
admission_pre_t0_180d	TRUE	14	0.7154	0.0096	-0.0050	0.0002
anticonvulsivante	TRUE	13	0.7154	0.0096	-0.0050	0.0001
meds_cardiovasc_qtde	TRUE	12	0.7171	0.0099	-0.0067	-0.0018

Table 1: *(continued)*

Tested Feature	Dropped	Features	CV AUC	CV AUC Std Error	Total AUC Loss	Instant AUC Loss
bloq_calcio	TRUE	11	0.7160	0.0098	-0.0056	0.0012
exames_imagem_qtde	TRUE	10	0.7170	0.0096	-0.0066	-0.0010
antiarritmico	TRUE	9	0.7155	0.0097	-0.0051	0.0015
diuretico	TRUE	8	0.7145	0.0092	-0.0041	0.0010
icu_t0	TRUE	7	0.7152	0.0094	-0.0048	-0.0007
education_level	TRUE	6	0.7149	0.0091	-0.0045	0.0003
equipe_multiprof	FALSE	5	0.7129	0.0089	-0.0045	0.0020
meds_antimicrobianos	TRUE	5	0.7159	0.0087	-0.0055	-0.0010
age	TRUE	4	0.7174	0.0093	-0.0070	-0.0015
classe_meds_qtde	FALSE	3	0.7079	0.0085	-0.0070	0.0095
admission_pre_t0_count	FALSE	3	0.7079	0.0092	-0.0070	0.0095
hospital_stay	FALSE	3	0.6731	0.0092	-0.0070	0.0444

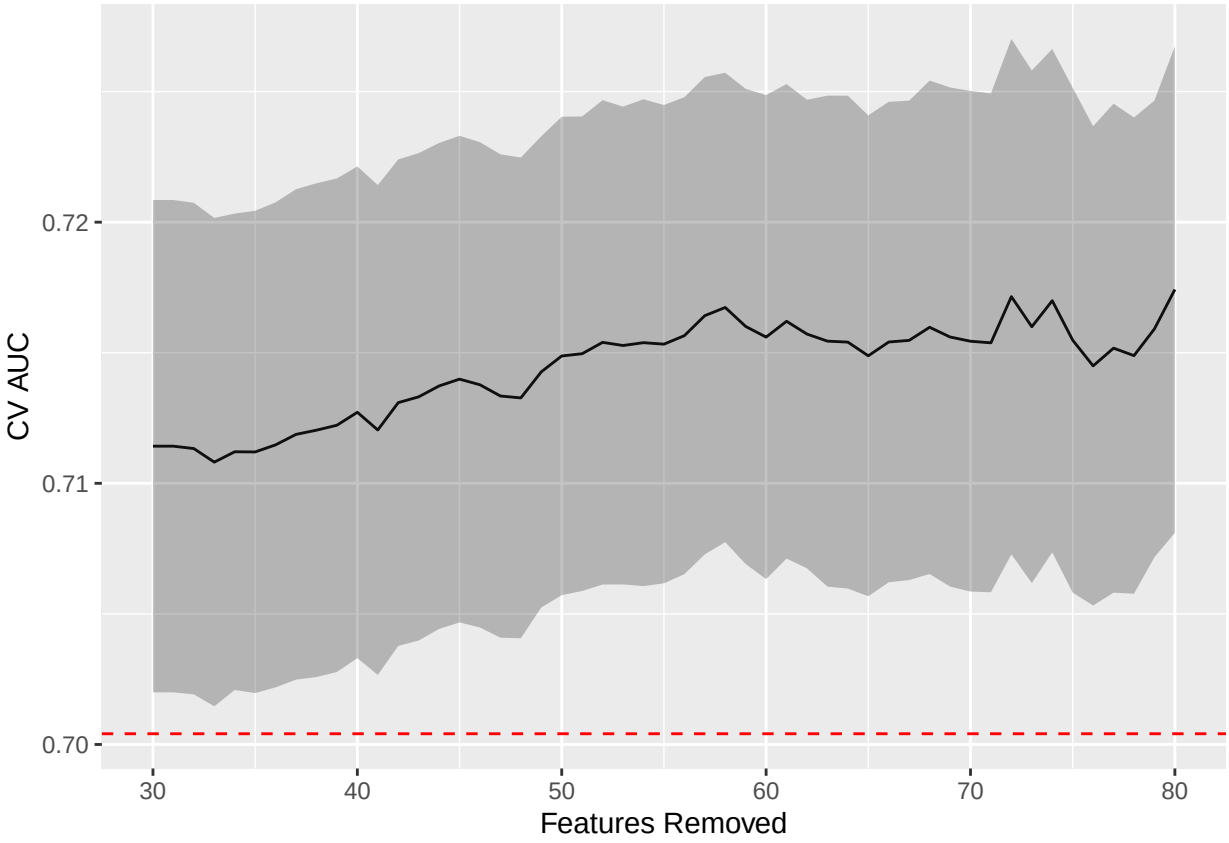
```
selected_features <- current_features

con <- file(sprintf('./auxiliar/final_model/selected_features/%s.yaml', outcome_column), "w")
write_yaml(selected_features, con)
close(con)

feature_selected_model <- model_fit_wf(df_train, selected_features,
                                       outcome_column, hyperparameters)

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

```
## [1] "Selected Model CV Train AUC: 0.717"
```

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

```
## [1] "Selected Model Test AUC: 0.674"
```

```
 selection_results <- selection_results %>%
  filter(`Number of Features` < length(features)) %>%
  mutate(`Features Removed` = length(features) - `Number of Features`,
         `CV AUC Low` = `CV AUC` - `CV AUC Std Error`,
         `CV AUC High` = `CV AUC` + `CV AUC Std Error`)

selection_results %>%
  filter(Dropped) %>%
  ggplot(aes(x = `Features Removed`, y = `CV AUC`,
             ymin = `CV AUC Low`, ymax = `CV AUC High`)) +
  geom_line() +
  geom_ribbon(alpha = .3) +
  geom_hline(yintercept = full_model$cv_auc - max_auc_loss,
             linetype = "dashed", color = "red")
```



Hyperparameter tuning

Selected features:

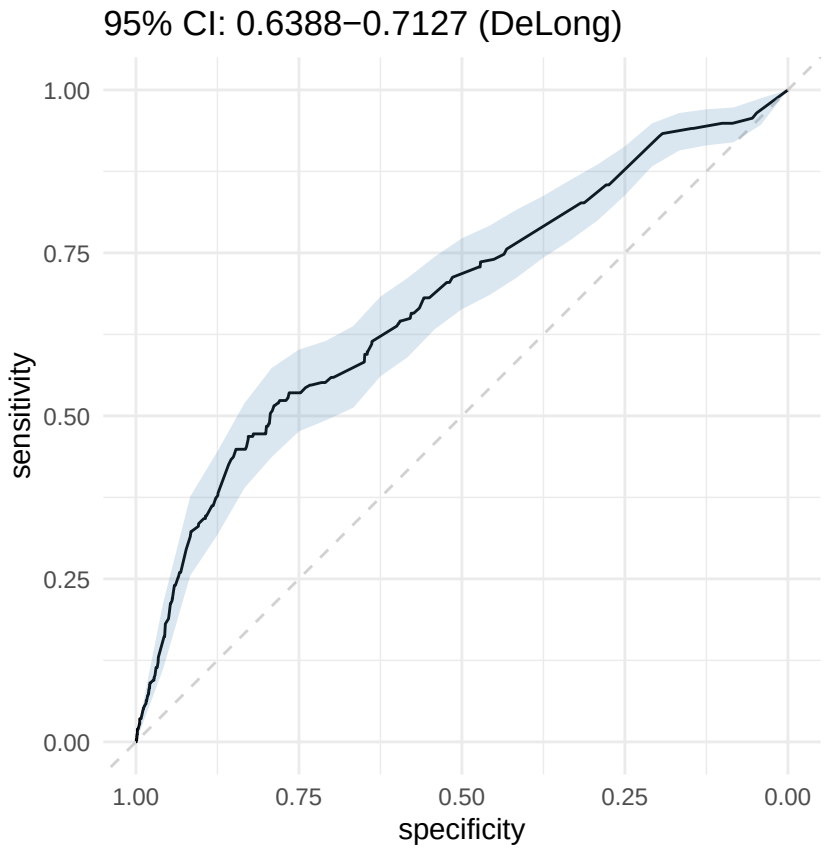
```
gluedown::md_order(selected_features, seq = TRUE, pad = TRUE)
```

1. hospital_stay
2. admission_pre_t0_count
3. classe_meds_qtde
4. equipe_multiprof

Standard

```
lightgbm_recipe <-
  recipe(formula = sprintf("%s ~ .", outcome_column) %>% as.formula,
         data = df_train %>% select(all_of(c(selected_features, outcome_column)))) %>%
  step_novel(all_nominal_predictors()) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_other(all_nominal_predictors(), threshold = 0.05, other = ".merged") %>%
  step_dummy(all_nominal_predictors())

lightgbm_tuning <- function(recipe) {

  lightgbm_spec <- boost_tree(
    trees = tune(),
    min_n = tune(),
    tree_depth = tune(),
    learn_rate = tune(),
    sample_size = 1.0
  ) %>%
    set_engine("lightgbm",
               nthread = 8) %>%
    set_mode("classification")
```

```
  lightgbm_grid <- grid_latin_hypercube(
    trees(range = c(25L, 150L)),
    min_n(range = c(2L, 100L)),
    tree_depth(range = c(2L, 15L)),
    learn_rate(range = c(-3, -1), trans = log10_trans()),
    size = grid_size
  )

  lightgbm_workflow <-
    workflow() %>%
    add_recipe(recipe) %>%
    add_model(lightgbm_spec)

  lightgbm_tune <-
    lightgbm_workflow %>%
    tune_grid(resamples = df_folds,
              grid = lightgbm_grid)

  lightgbm_tune %>%
    show_best("roc_auc") %>%
    niceFormatting(digits = 5, label = 4)

  best_lightgbm <- lightgbm_tune %>%
    select_best("roc_auc")

  autoplot(lightgbm_tune, metric = "roc_auc")

  final_lightgbm_workflow <-
    lightgbm_workflow %>%
    finalize_workflow(best_lightgbm)

  last_lightgbm_fit <-
    final_lightgbm_workflow %>%
    last_fit(df_split)

  final_lightgbm_fit <- extract_workflow(last_lightgbm_fit)

  lightgbm_auc <- validation(final_lightgbm_fit, df_test)

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

  return(list(auc = as.numeric(lightgbm_auc$auc),
              auc_lower = lightgbm_auc$ci[1],
              auc_upper = lightgbm_auc$ci[3],
              parameters = lightgbm_parameters,
              fit = final_lightgbm_fit))
}

standard_results <- lightgbm_tuning(lightgbm_recipe)
```




```
## [1] "Optimal Threshold: 0.08"
## Confusion Matrix and Statistics
##
##     reference
## data    0    1
##    0 3528  123
##    1  948  131
##
##                Accuracy : 0.7736
##                  95% CI : (0.7614, 0.7854)
##     No Information Rate : 0.9463
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1201
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7882
##             Specificity : 0.5157
##          Pos Pred Value : 0.9663
##          Neg Pred Value : 0.1214
##              Prevalence : 0.9463
##          Detection Rate : 0.7459
##    Detection Prevalence : 0.7719
##       Balanced Accuracy : 0.6520
##
##        'Positive' Class : 0
##
```

```
final_lightgbm_fit <- standard_results$fit
lightgbm_parameters <- standard_results$parameters

con <- file(sprintf('./auxiliar/final_model/hyperparameters/%s.yaml',
                    outcome_column), "w")
```

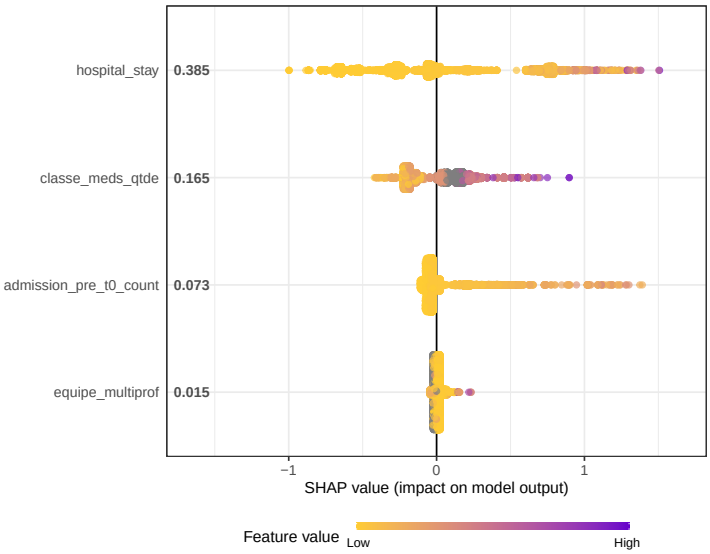
```r
write_yaml(lightgbm_parameters, con)
close(con)

# Save the final model. We need it for the calculator
lgb.save(
  parsnip::extract_fit_engine(final_lightgbm_fit),
  sprintf("./results/%s/final_model.txt", outcome_column)
)
saveRDS(final_lightgbm_fit,
        sprintf("./results/%s/final_model_wf.rds", outcome_column))
```

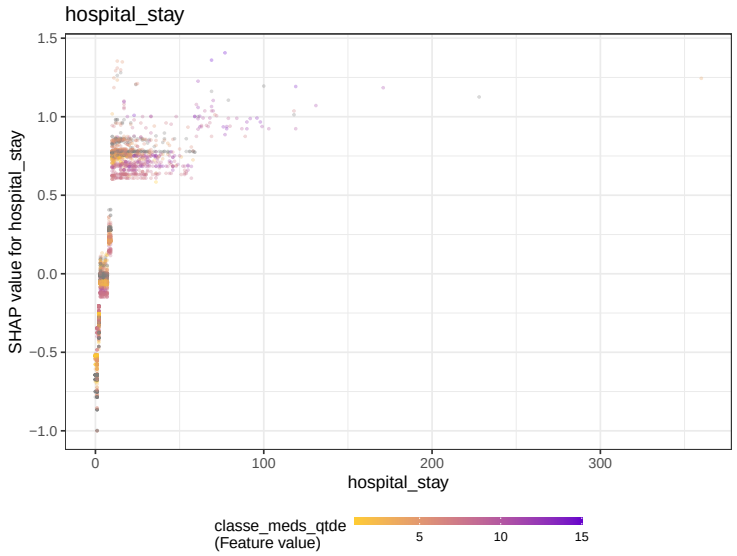
SHAP values

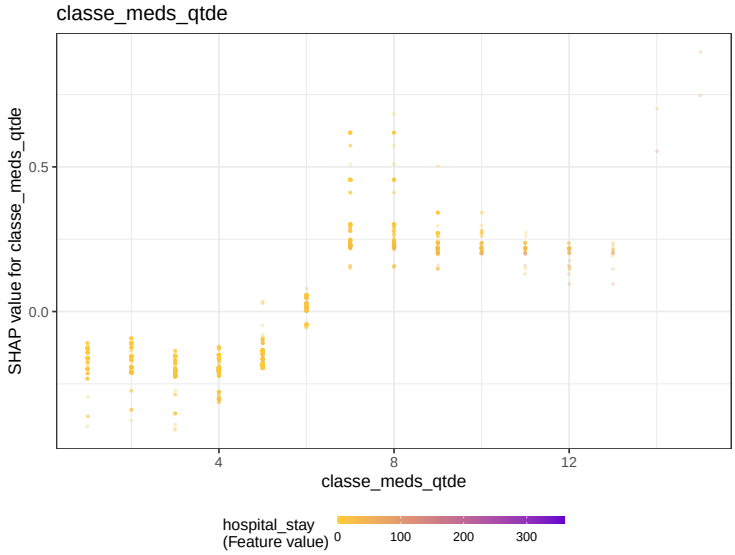
```r
lightgbm_model <- parsnip::extract_fit_engine(final_lightgbm_fit)

trained_rec <- prep(lightgbm_recipe, training = df_train)

df_train_baked <- bake(trained_rec, new_data = df_train)
df_test_baked <- bake(trained_rec, new_data = df_test)

matrix_train <- as.matrix(df_train_baked %>% select(-all_of(outcome_column)))
matrix_test <- as.matrix(df_test_baked %>% select(-all_of(outcome_column)))

n_plots <- min(6, length(selected_features))
plotted <- 0

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

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

for (x in shap.importance(shap, names_only = TRUE)) {
  p <- shap.plot.dependence(
    shap,
    x = x,
    color_feature = "auto",
    smooth = FALSE,
    jitter_width = 0.01,
    alpha = 0.3
  ) +
    labs(title = x)

  if (plotted < n_plots) {
    print(p)
    plotted <- plotted + 1
  }

  ggsave(sprintf("./auxiliar/final_model/shap_plots/%s/%s.png",
                 outcome_column, x),
         plot = p,
         dpi = 300)
}
```

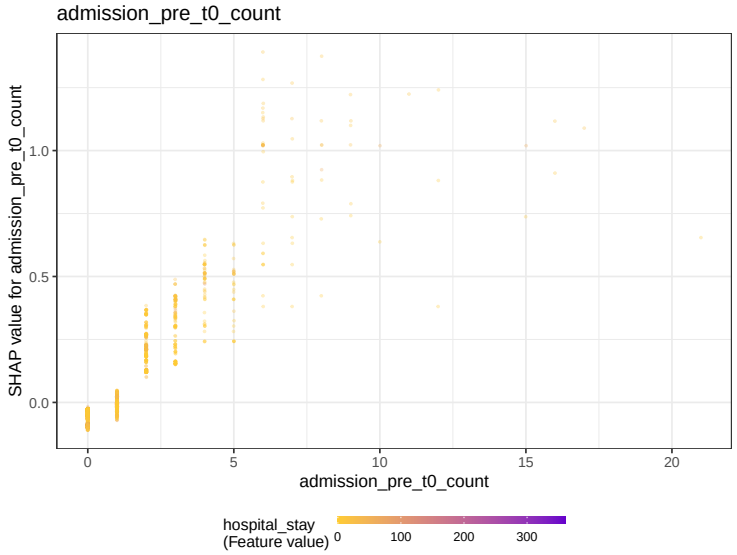
```
## Saving 6.5 x 5 in image
## Warning: Removed 1455 rows containing missing values (`geom_point()`).
## Saving 6.5 x 5 in image
## Warning: Removed 1455 rows containing missing values (`geom_point()`).
## Saving 6.5 x 5 in image
```

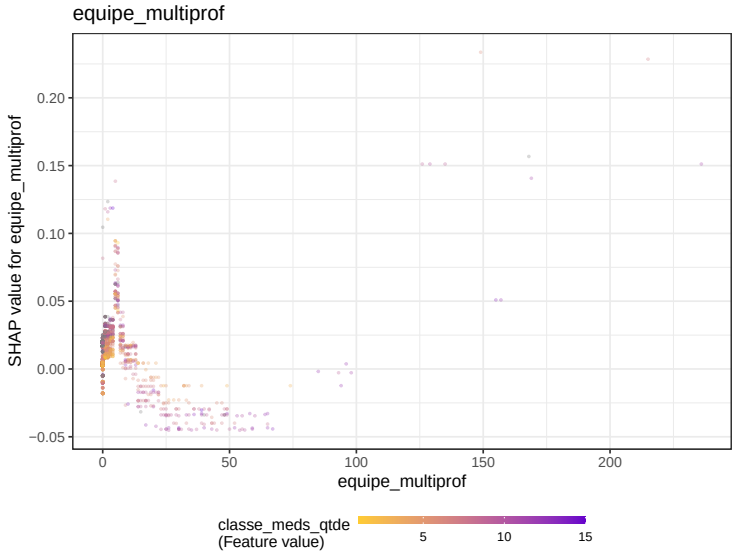
```
## Warning: Removed 802 rows containing missing values (‘geom_point()‘).
## Saving 6.5 x 5 in image
## Warning: Removed 802 rows containing missing values (‘geom_point()‘).
```











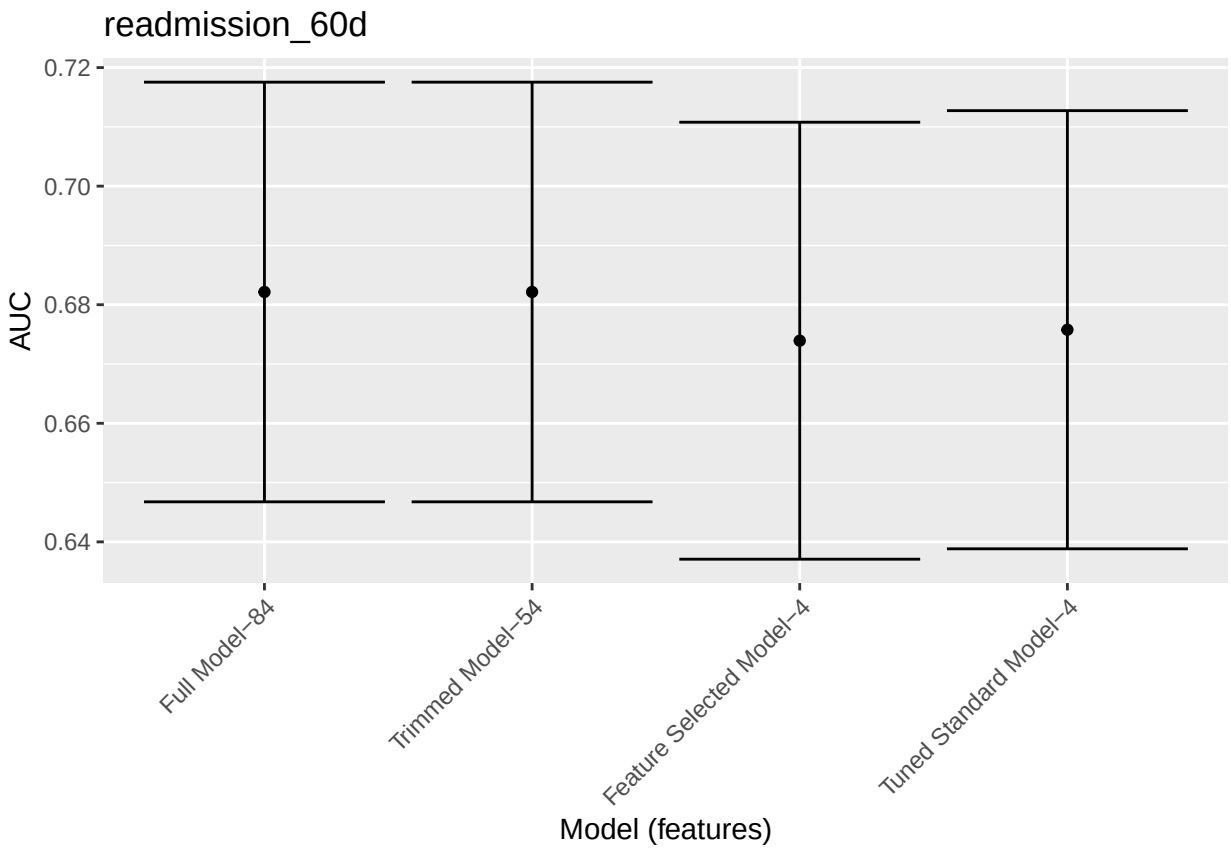

```
## $num_iterations
## [1] 63
```

```
##
## $learning_rate
## [1] 0.02654159
##
## $max_depth
## [1] 3
##
## $feature_fraction_bynode
## [1] 1
##
## $min_data_in_leaf
## [1] 9
##
## $min_gain_to_split
## [1] 0
##
## $bagging_fraction
## [1] 1
##
## $num_class
## [1] 1
##
## $objective
## [1] "binary"
##
## $num_threads
## $num_threads$num_threads
## [1] 0
##
##
## $nthread
## [1] 8
##
## $seed
## [1] 88564
##
## $deterministic
## [1] TRUE
##
## $verbose
## [1] -1
##
## $metric
## list()
##
## $interaction_constraints
## list()
##
## $feature_pre_filter
## [1] FALSE
```

Models Comparison

```
df_auc <- tibble::tribble(
  ~Model, ~`AUC`, ~`Lower Limit`, ~`Upper Limit`, ~`Features`,
  'Full Model', full_model$auc, full_model$auc_lower, full_model$auc_upper, length(features),
  'Trimmed Model', trimmed_model$auc, trimmed_model$auc_lower, trimmed_model$auc_upper, length(trimmed_features)
  'Feature Selected Model', feature_selected_model$auc, feature_selected_model$auc_lower, feature_selected_model
  'Tuned Standard Model', standard_results$auc, standard_results$auc_lower, standard_results$auc_upper, length(s
) %>%
```

```
  mutate(Target = outcome_column,
         `Model (features)` = fct_reorder(paste0(Model, "-", Features), -Features))

df_auc %>%
  ggplot(aes(
    x = `Model (features)`,
    y = AUC,
    ymin = `Lower Limit`,
    ymax = `Upper Limit`
  )) +
  geom_point() +
  geom_errorbar() +
  labs(title = outcome_column) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
write_csv(df_auc, sprintf("./auxiliar/final_model/performance/%s.csv", outcome_column))
```