Biostat 203B Homework 5

Due Mar 20 @ 11:59PM

AUTHOR
Tanya Wang, 605587605

Predicting ICU duration

Using the ICU cohort `mimiciv_icu_cohort.rds` you built in Homework 4, develop at least three machine learning approaches (logistic regression with enet regularization, random forest, boosting, SVM, MLP, etc) plus a model stacking approach for predicting whether a patient's ICU stay will be longer than 2 days. You should use the `los_long` variable as the outcome. You algorithms can use patient demographic information (gender, age at ICU `intime`, marital status, race), ICU admission information (first care unit), the last lab measurements before the ICU stay, and first vital measurements during ICU stay as features. You are welcome to use any feature engineering techniques you think are appropriate; but make sure to not use features that are not available at an ICU stay's `intime`. For instance, `last_careunit` cannot be used in your algorithms.

Question 1

Data preprocessing and feature engineering.

Solution:

Load the dataset and convert categorical variables to factor:

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.4     ✓ readr     2.1.5
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ ggplot2   3.5.1     ✓ tibble    3.2.1
✓ lubridate 1.9.4     ✓ tidyr     1.3.1
✓ purrr     1.0.4
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```
library(caret)
```

```
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':
```

```
    lift
```

```
library(data.table)
```

```
Attaching package: 'data.table'

The following objects are masked from 'package:lubridate':

    hour, isoweek, mday, minute, month, quarter, second, wday, week,
    yday, year

The following objects are masked from 'package:dplyr':

    between, first, last

The following object is masked from 'package:purrr':

    transpose
```

```
mimic_icu_cohort <-
  readRDS("~/203b-hw/hw4/mimiciv_shiny/mimic_icu_cohort.rds") |>
  filter(!is.na(los_long)) |>
  select(-los, -last_careunit, -dod, -discharge_location,
         -hospital_expire_flag, -intime, -outtime, -admittime, -dischtime,
         -deathtime, -edregtime, -edouttime, -admit_provider_id, -anchor_year,
         -anchor_age, -anchor_year_group) |>
  mutate(across(
    c(los_long,
      insurance,
      language,
      marital_status,
      gender), as.factor))

colnames(mimic_icu_cohort)
```

```
 [1] "subject_id"
 [2] "hadm_id"
 [3] "stay_id"
 [4] "first_careunit"
 [5] "admission_type"
 [6] "admission_location"
 [7] "insurance"
 [8] "language"
 [9] "marital_status"
[10] "race"
[11] "gender"
[12] "age_intime"
```

```
[13] "bicarbonate"
[14] "chloride"
[15] "creatinine"
[16] "glucose"
[17] "potassium"
[18] "sodium"
[19] "hematocrit"
[20] "wbc"
[21] "heart_rate"
[22] "non_invasive_blood_pressure_diastolic"
[23] "non_invasive_blood_pressure_systolic"
[24] "respiratory_rate"
[25] "temperature_fahrenheit"
[26] "los_long"
```

```
str(mimic_icu_cohort)
```

```
tibble [94,444 × 26] (S3: tbl_df/tbl/data.frame)
 $ subject_id                          : int [1:94444] 10000032 10000690 10000980 10001217
10001217 10001725 10001843 10001884 10002013 10002114 ...
 $ hadm_id                             : int [1:94444] 29079034 25860671 26913865 24597018
27703517 25563031 26133978 26184834 23581541 27793700 ...
 $ stay_id                             : int [1:94444] 39553978 37081114 39765666 37067082
34592300 31205490 39698942 37510196 39060235 34672098 ...
 $ first_careunit                      : Factor w/ 5 levels "Cardiac Vascular Intensive Care
Unit (CVICU)",..: 2 2 2 4 4 3 3 2 1 5 ...
 $ admission_type                      : Factor w/ 5 levels "EW EMER.","OBSERVATION ADMIT",..: 1
1 1 1 5 1 4 2 3 2 ...
 $ admission_location                  : Factor w/ 5 levels "EMERGENCY ROOM",..: 1 1 1 1 2 5 3 1
2 2 ...
 $ insurance                           : Factor w/ 5 levels "Medicaid","Medicare",..: 1 2 2 5 5
5 2 2 2 1 ...
 $ language                            : Factor w/ 25 levels "American Sign Language",..: 7 7 7
17 17 7 7 7 7 7 ...
 $ marital_status                      : Factor w/ 4 levels "DIVORCED","MARRIED",..: 4 4 2 2 2 2
3 2 3 NA ...
 $ race                                : Factor w/ 5 levels "OTHER","ASIAN",..: 5 5 3 5 5 5 5 3
1 1 ...
 $ gender                              : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 2 1 1 2 ...
 $ age_intime                          : int [1:94444] 52 86 76 55 55 46 76 77 57 56 ...
 $ bicarbonate                         : num [1:94444] 25 26 21 22 30 NA 28 30 24 18 ...
 $ chloride                            : num [1:94444] 95 100 109 108 104 98 97 88 102 NA ...
 $ creatinine                          : num [1:94444] 0.7 1 2.3 0.6 0.5 NA 1.3 1.1 0.9 3.1 ...
 $ glucose                             : num [1:94444] 102 85 89 112 87 NA 131 141 288 95 ...
 $ potassium                           : num [1:94444] 6.7 4.8 3.9 4.2 4.1 4.1 3.9 4.5 3.5 6.5
...
 $ sodium                              : num [1:94444] 126 137 144 142 142 139 138 130 137 125
...
 $ hematocrit                          : num [1:94444] 41.1 36.1 27.3 38.1 37.4 NA 31.4 39.7
34.9 34.3 ...
```

```
 $ wbc                                : num [1:94444] 6.9 7.1 5.3 15.7 5.4 NA 10.4 12.2 7.2
16.8 ...
 $ heart_rate                         : num [1:94444] 91 78 76 86 79.3 ...
 $ non_invasive_blood_pressure_diastolic: num [1:94444] 48 56.5 102 90 93.3 ...
 $ non_invasive_blood_pressure_systolic : num [1:94444] 84 106 154 151 156 ...
 $ respiratory_rate                   : num [1:94444] 24 24.3 23.5 18 14 ...
 $ temperature_fahrenheit             : num [1:94444] 98.7 97.7 98 98.5 97.6 97.7 97.9 98.1
97.2 97.9 ...
 $ los_long                           : Factor w/ 2 levels "TRUE","FALSE": 2 1 2 2 2 2 2 1 2 1
...
```

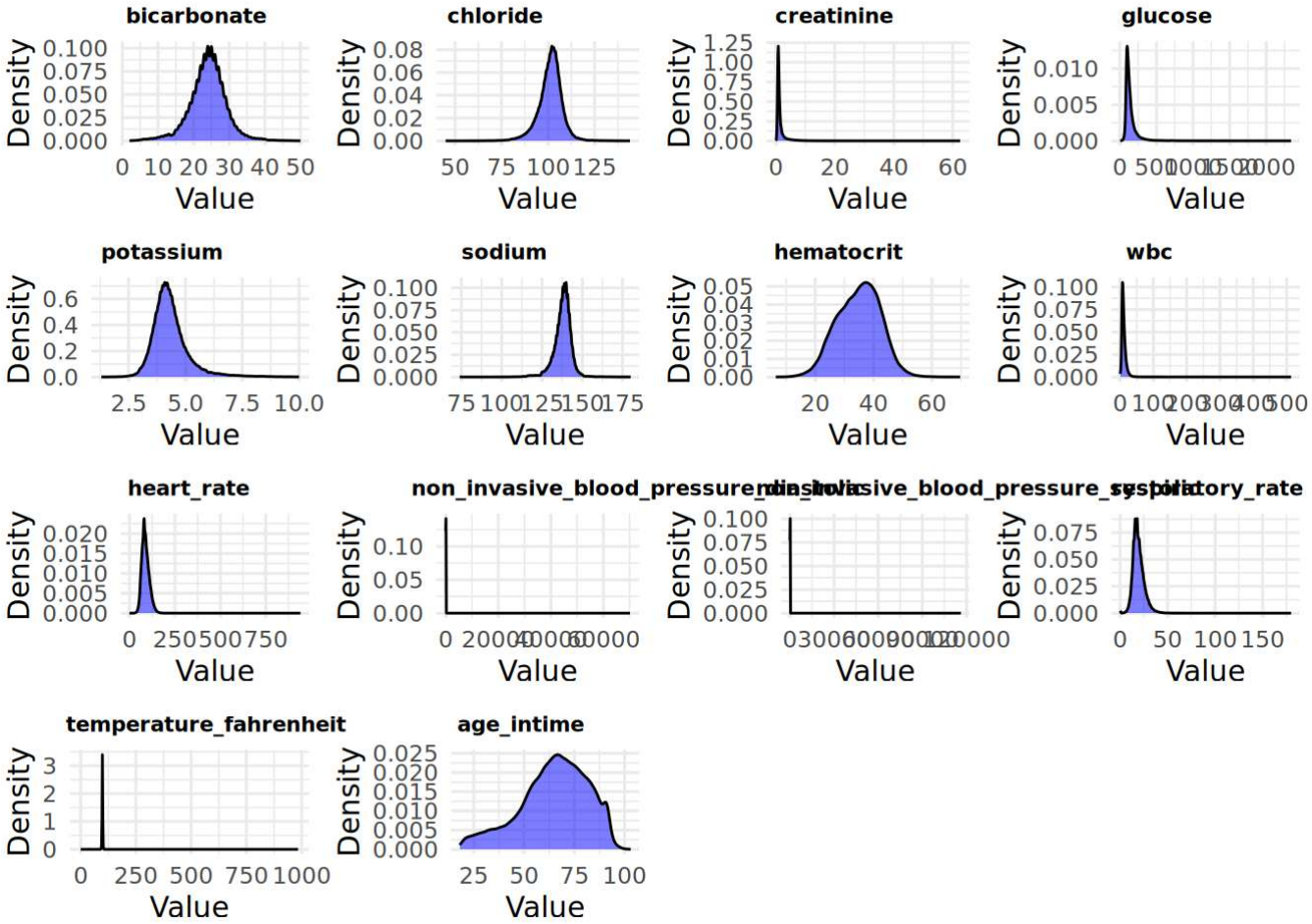
```
library(ggplot2)
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```
# Select only continuous variables
continuous_vars <- mimic_icu_cohort |>
  select(
    bicarbonate, chloride, creatinine, glucose, potassium, sodium,
    hematocrit, wbc, heart_rate,
    non_invasive_blood_pressure_diastolic,
    non_invasive_blood_pressure_systolic,
    respiratory_rate, temperature_fahrenheit, age_intime
  )

plot_list <- list()

for (var in colnames(continuous_vars)) {
  p <- ggplot(mimic_icu_cohort, aes_string(x = var)) +
    geom_density(fill = "blue", alpha = 0.5, na.rm = TRUE) +
    labs(title = var, x = "Value", y = "Density") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.1, face = "bold", size = 8))

  plot_list[[var]] <- p
}
```

```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
ℹ Please use tidy evaluation idioms with `aes()`.
ℹ See also `vignette("ggplot2-in-packages")` for more information.
```

```
grid.arrange(grobs = plot_list, ncol = 4)
```



Handle missing values (for continuous variables, handle skewed data with median imputation and normally distributed data with mean imputation, and for categorical variables, replace missing values with the mode):

```
library(moments)

skewness_values <- sapply(mimic_icu_cohort,
                          function(x)
                            if(is.numeric(x))
                              skewness(x, na.rm = TRUE)
                            else
                              NA)

# Impute with median
skewed_vars <- names(skewness_values[abs(skewness_values) > 0.5])
# Impute with mean
normal_vars <- names(skewness_values[abs(skewness_values) <= 0.5])

impute_missing <- function(df) {
  for (col in colnames(df)) {
    if (is.numeric(df[[col]])) {
      if (col %in% skewed_vars) {
        df[[col]][is.na(df[[col]])] <-
          median(df[[col]], na.rm = TRUE)  # Median for skewed data
      } else {
```

```
        df[[col]][is.na(df[[col]])] <-
          mean(df[[col]], na.rm = TRUE)  # Mean for normal data
      }
    } else {
      # Impute categorical variables with mode
      df[[col]][is.na(df[[col]])] <-
        as.character(names(sort(table(df[[col]]), decreasing = TRUE))[1])
    }
  }
  return(df)
}

mimic_icu_cohort <- impute_missing(mimic_icu_cohort)

colSums(is.na(mimic_icu_cohort))
```

```
                          subject_id                                hadm_id
                                   0                                       0
                             stay_id                           first_careunit
                                   0                                       0
                      admission_type                      admission_location
                                   0                                       0
                           insurance                                language
                                   0                                       0
                      marital_status                                    race
                                   0                                       0
                              gender                               age_intime
                                   0                                       0
                         bicarbonate                                chloride
                                   0                                       0
                          creatinine                                 glucose
                                   0                                       0
                           potassium                                  sodium
                                   0                                       0
                          hematocrit                                     wbc
                                   0                                       0
                          heart_rate non_invasive_blood_pressure_diastolic
                                   0                                       0
non_invasive_blood_pressure_systolic                        respiratory_rate
                                   0                                       0
               temperature_fahrenheit                                los_long
                                   0                                       0
```

Quetion 2

Partition data into 50% training set and 50% test set. Stratify partitioning according to `los_long`. For grading purpose, sort the data by `subject_id`, `hadm_id`, and `stay_id` and use the seed `203` for the initial data split. Below is the sample code.

```
set.seed(203)

# sort
mimiciv_icu_cohort <- mimiciv_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id)

data_split <- initial_split(
  mimiciv_icu_cohort,
  # stratify by los_long
  strata = "los_long",
  prop = 0.5
  )
```

Solution:

```
library(tidymodels)
```

```
── Attaching packages ─────────────────────────────────────── tidymodels 1.3.0 ──

✓ broom        1.0.7     ✓ rsample      1.2.1
✓ dials        1.4.0     ✓ tune         1.3.0
✓ infer        1.0.7     ✓ workflows    1.2.0
✓ modeldata    1.4.0     ✓ workflowsets 1.1.0
✓ parsnip      1.3.1     ✓ yardstick    1.3.2
✓ recipes      1.1.1

── Conflicts ────────────────────────────────────────── tidymodels_conflicts() ──
✖ data.table::between()    masks dplyr::between()
✖ gridExtra::combine()     masks dplyr::combine()
✖ scales::discard()        masks purrr::discard()
✖ dplyr::filter()          masks stats::filter()
✖ data.table::first()      masks dplyr::first()
✖ recipes::fixed()         masks stringr::fixed()
✖ dplyr::lag()             masks stats::lag()
✖ data.table::last()       masks dplyr::last()
✖ caret::lift()            masks purrr::lift()
✖ yardstick::precision()   masks caret::precision()
✖ yardstick::recall()      masks caret::recall()
✖ yardstick::sensitivity() masks caret::sensitivity()
✖ yardstick::spec()        masks readr::spec()
✖ yardstick::specificity() masks caret::specificity()
✖ recipes::step()          masks stats::step()
✖ data.table::transpose()  masks purrr::transpose()
```

```
set.seed(203)

# Sort dataset by subject_id, hadm_id, stay_id
mimic_icu_cohort <- mimic_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id)
```

```
# Split the dataset
data_split <- initial_split(
  mimic_icu_cohort,
  strata = "los_long",
  prop = 0.5
)

train_data <- training(data_split) |>
  select(-subject_id, -hadm_id, -stay_id)

test_data <- testing(data_split) |>
  select(-subject_id, -hadm_id, -stay_id)

dim(train_data)
```

```
[1] 47221    23
```

```
dim(test_data)
```

```
[1] 47223    23
```

Question 3

Train and tune the models using the training set.

Logistic regression (with enet regularization)

```
logit_recipe <- recipe(
  los_long ~ .,
  data = train_data
  ) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_numeric_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  print()
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 22
```

```
── Operations
```

```
• Dummy variables from: all_nominal_predictors()
```

```
• Zero variance filter on: all_numeric_predictors()
```

```
• Centering and scaling for: all_numeric_predictors()
```

```
logit_mod <-
  logistic_reg(
    penalty = tune(),
    mixture = tune()
  ) |>
  set_engine("glmnet", standardize = FALSE) |>
  print()
```

```
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```
logit_wf <- workflow() |>
  add_recipe(logit_recipe) |>
  add_model(logit_mod) |>
  print()
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
3 Recipe Steps

• step_dummy()
• step_zv()
• step_normalize()

── Model ───────────────────────────────────────────────────────────────────────
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()
```

```
Engine-Specific Arguments:
  standardize = FALSE

Computational engine: glmnet
```

```
param_grid <- grid_regular(
  penalty(range = c(-6, 3)),
  mixture(),
  levels = c(5, 5)
  ) |>
  print()
```

```
# A tibble: 25 × 2
       penalty mixture
         <dbl>   <dbl>
 1    0.000001    0
 2    0.000178    0
 3    0.0316      0
 4    5.62        0
 5 1000           0
 6    0.000001    0.25
 7    0.000178    0.25
 8    0.0316      0.25
 9    5.62        0.25
10 1000           0.25
# ℹ 15 more rows
```

Set cross-validation partitions.

```
set.seed(203)

folds <- vfold_cv(train_data, v = 5)
folds
```

```
#  5-fold cross-validation
# A tibble: 5 × 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37777/9444]> Fold2
3 <split [37777/9444]> Fold3
4 <split [37777/9444]> Fold4
5 <split [37777/9444]> Fold5
```

Fit cross-validation.

```
library(glmnet)
```

```
Loading required package: Matrix


Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loaded glmnet 4.1-8
```

```
(logit_fit <- logit_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid,
    metrics = metric_set(roc_auc, accuracy)
    )) |>
  system.time()
```

```
   user  system elapsed 
 19.512   0.167  18.996 
```

```
logit_fit
```

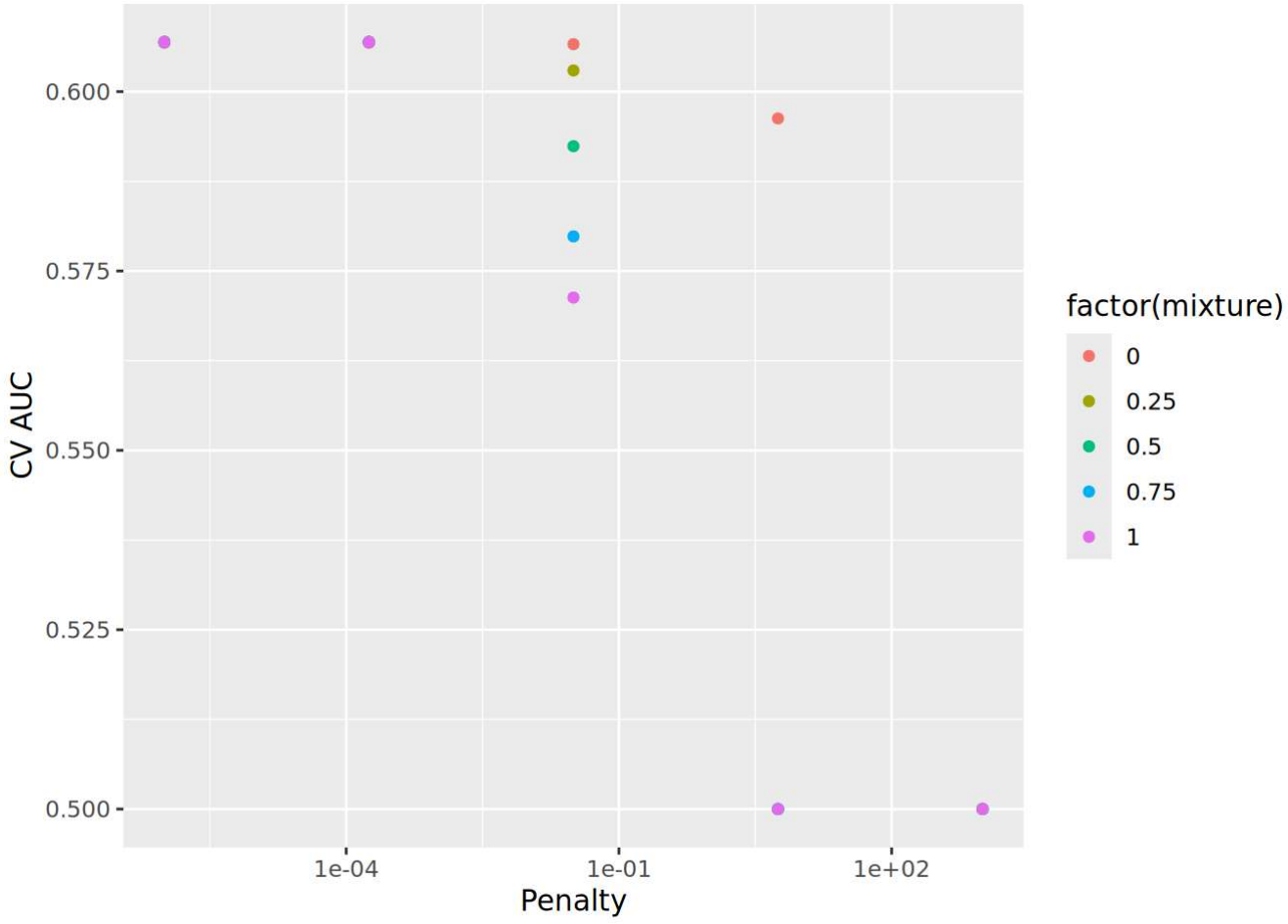
```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 × 4
  splits               id    .metrics          .notes
  <list>               <chr> <list>            <list>
1 <split [37776/9445]> Fold1 <tibble [50 × 6]> <tibble [0 × 3]>
2 <split [37777/9444]> Fold2 <tibble [50 × 6]> <tibble [0 × 3]>
3 <split [37777/9444]> Fold3 <tibble [50 × 6]> <tibble [0 × 3]>
4 <split [37777/9444]> Fold4 <tibble [50 × 6]> <tibble [0 × 3]>
5 <split [37777/9444]> Fold5 <tibble [50 × 6]> <tibble [0 × 3]>
```

Visualize CV results:

```
logit_fit |>
  # aggregate metrics from K folds
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = penalty, y = mean, color = factor(mixture))) +
  geom_point() +
  labs(x = "Penalty", y = "CV AUC") +
  scale_x_log10()
```

```
# A tibble: 50 × 8
       penalty mixture .metric  .estimator  mean     n std_err
         <dbl>   <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
```

```
   1    0.000001       0 accuracy binary     0.577     5 0.00222
   2    0.000001       0 roc_auc  binary     0.607     5 0.00124
   3    0.000178       0 accuracy binary     0.577     5 0.00222
   4    0.000178       0 roc_auc  binary     0.607     5 0.00124
   5    0.0316         0 accuracy binary     0.576     5 0.00234
   6    0.0316         0 roc_auc  binary     0.607     5 0.00107
   7    5.62           0 accuracy binary     0.527     5 0.00394
   8    5.62           0 roc_auc  binary     0.596     5 0.00144
   9 1000              0 accuracy binary     0.509     5 0.00164
  10 1000              0 roc_auc  binary     0.5       5 0
     .config
     <chr>
   1 Preprocessor1_Model01
   2 Preprocessor1_Model01
   3 Preprocessor1_Model02
   4 Preprocessor1_Model02
   5 Preprocessor1_Model03
   6 Preprocessor1_Model03
   7 Preprocessor1_Model04
   8 Preprocessor1_Model04
   9 Preprocessor1_Model05
  10 Preprocessor1_Model05
  # ℹ 40 more rows
```



Show the top 5 models.

```
logit_fit |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
   penalty mixture .metric .estimator  mean     n std_err .config
     <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 0.000001    0.5  roc_auc binary     0.607     5 0.00126 Preprocessor1_Model11
2 0.000178    0.5  roc_auc binary     0.607     5 0.00126 Preprocessor1_Model12
3 0.000001    1    roc_auc binary     0.607     5 0.00126 Preprocessor1_Model21
4 0.000178    1    roc_auc binary     0.607     5 0.00126 Preprocessor1_Model22
5 0.000001    0.75 roc_auc binary     0.607     5 0.00126 Preprocessor1_Model16
```

Select the best model.

```
best_logit <- logit_fit |>
  select_best(metric = "roc_auc")
best_logit
```

```
# A tibble: 1 × 3
   penalty mixture .config
     <dbl>   <dbl> <chr>
1 0.000001     0.5 Preprocessor1_Model11
```

Fit this final model to the whole training data and use test data to estimate the model performance.

```
# Final workflow
final_wf_logit <- logit_wf |>
  finalize_workflow(best_logit)
final_wf_logit
```

```
══ Workflow ════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────
3 Recipe Steps

• step_dummy()
• step_zv()
• step_normalize()

── Model ───────────────────────────────────────────────────────────────
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = 1e-06
  mixture = 0.5

Engine-Specific Arguments:
```

```
  standardize = FALSE

Computational engine: glmnet
```

```
# Fit the whole training set, then predict the test cases
final_fit_logit <-
  final_wf_logit |>
  last_fit(data_split)
final_fit_logit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits                id             .metrics .notes   .predictions .workflow
  <list>                <chr>          <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp… <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
final_fit_logit |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.576 Preprocessor1_Model1
2 roc_auc     binary         0.605 Preprocessor1_Model1
3 brier_class binary         0.242 Preprocessor1_Model1
```

Random Forest

```
rf_recipe <-
  recipe(
    los_long ~ .,
    data = train_data
  ) |>
  step_zv(all_numeric_predictors())
```

```
rf_mod <-
  rand_forest(
    mode = "classification",
    # Number of predictors randomly sampled in each split
    mtry = tune(),
    # Number of trees in ensemble
    trees = tune()
  ) |>
  set_engine("ranger", importance = "impurity")
rf_mod
```

```
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```
rf_wf <- workflow() |>
  add_recipe(rf_recipe) |>
  add_model(rf_mod)
rf_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────────────────────────
1 Recipe Step

• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```
param_grid <- grid_regular(
  trees(range = c(100L, 500L)),
  mtry(range = c(1L, 5L)),
  levels = c(5, 5)
  )
param_grid
```

```
# A tibble: 25 × 2
   trees  mtry
   <int> <int>
 1   100     1
 2   200     1
```

```
 3   300     1
 4   400     1
 5   500     1
 6   100     2
 7   200     2
 8   300     2
 9   400     2
10   500     2
# ℹ 15 more rows
```

Set cross-validation partitions.

```
set.seed(203)

folds <- vfold_cv(train_data, v = 5)
folds
```

```
#  5-fold cross-validation
# A tibble: 5 × 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37777/9444]> Fold2
3 <split [37777/9444]> Fold3
4 <split [37777/9444]> Fold4
5 <split [37777/9444]> Fold5
```

Fit cross-validation.

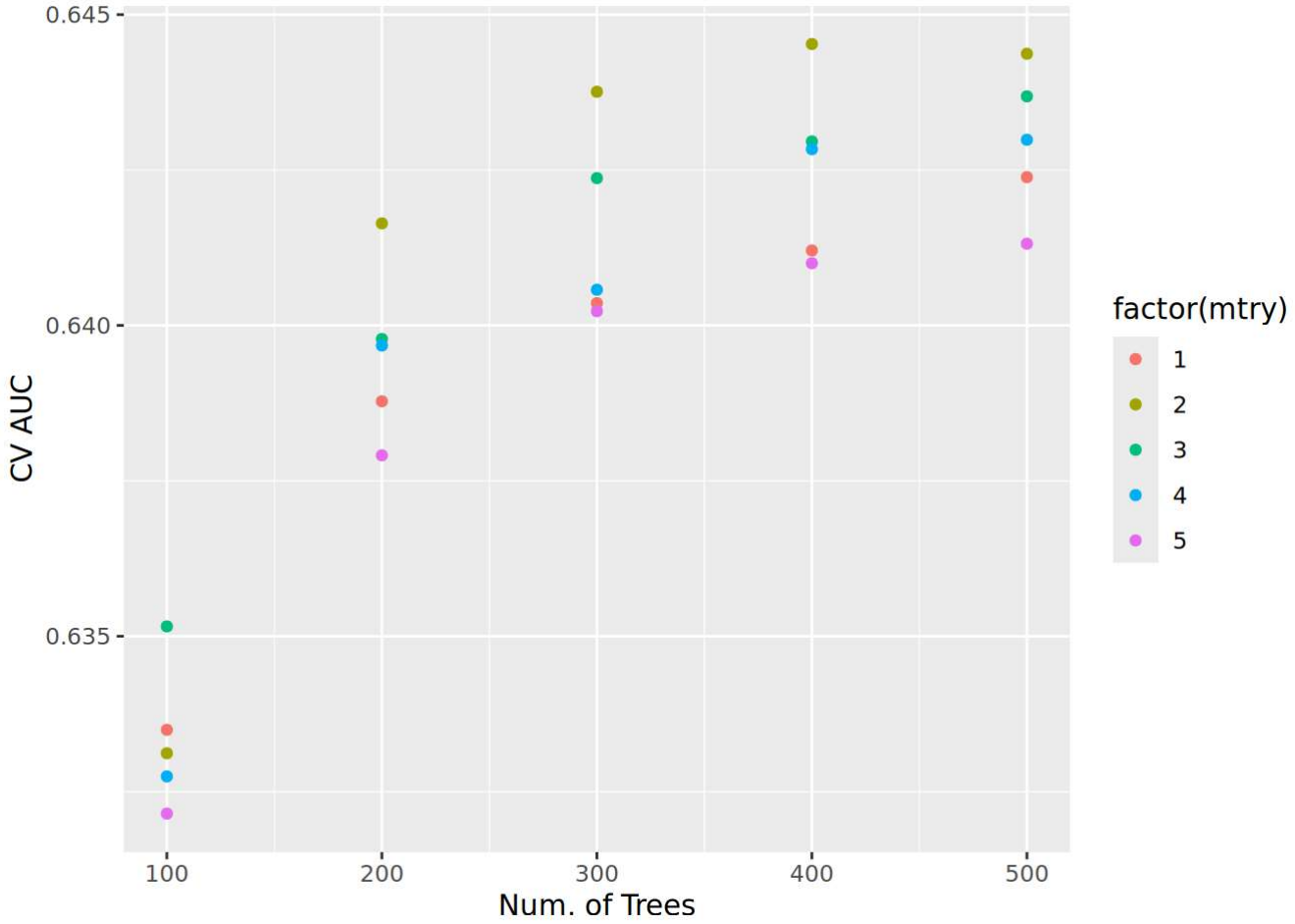
```
library(ranger)

rf_fit <- rf_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid,
    metrics = metric_set(roc_auc, accuracy)
    )
rf_fit
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 × 4
  splits               id    .metrics          .notes
  <list>               <chr> <list>            <list>
1 <split [37776/9445]> Fold1 <tibble [50 × 6]> <tibble [0 × 3]>
2 <split [37777/9444]> Fold2 <tibble [50 × 6]> <tibble [0 × 3]>
3 <split [37777/9444]> Fold3 <tibble [50 × 6]> <tibble [0 × 3]>
4 <split [37777/9444]> Fold4 <tibble [50 × 6]> <tibble [0 × 3]>
5 <split [37777/9444]> Fold5 <tibble [50 × 6]> <tibble [0 × 3]>
```

Visualize CV results:

```
rf_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = trees, y = mean, color = factor(mtry))) +
  geom_point() +
  # geom_line() +
  labs(x = "Num. of Trees", y = "CV AUC")
```

```
# A tibble: 50 × 8
    mtry trees .metric  .estimator  mean     n  std_err .config
   <int> <int> <chr>    <chr>      <dbl> <int>    <dbl> <chr>
 1     1   100 accuracy binary     0.596     5 0.00215  Preprocessor1_Model01
 2     1   100 roc_auc  binary     0.633     5 0.00215  Preprocessor1_Model01
 3     1   200 accuracy binary     0.598     5 0.000884 Preprocessor1_Model02
 4     1   200 roc_auc  binary     0.639     5 0.00155  Preprocessor1_Model02
 5     1   300 accuracy binary     0.599     5 0.00331  Preprocessor1_Model03
 6     1   300 roc_auc  binary     0.640     5 0.00312  Preprocessor1_Model03
 7     1   400 accuracy binary     0.600     5 0.00275  Preprocessor1_Model04
 8     1   400 roc_auc  binary     0.641     5 0.00252  Preprocessor1_Model04
 9     1   500 accuracy binary     0.600     5 0.00161  Preprocessor1_Model05
10     1   500 roc_auc  binary     0.642     5 0.00225  Preprocessor1_Model05
# ℹ 40 more rows
```



Show the top 5 models.

```
rf_fit |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
   mtry trees .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     2   400 roc_auc binary     0.645     5 0.00189 Preprocessor1_Model09
2     2   500 roc_auc binary     0.644     5 0.00164 Preprocessor1_Model10
3     2   300 roc_auc binary     0.644     5 0.00211 Preprocessor1_Model08
4     3   500 roc_auc binary     0.644     5 0.00196 Preprocessor1_Model15
5     4   500 roc_auc binary     0.643     5 0.00139 Preprocessor1_Model20
```

Select the best model.

```
best_rf <- rf_fit |>
  select_best(metric = "roc_auc")
best_rf
```

```
# A tibble: 1 × 3
   mtry trees .config
```

```
  <int> <int> <chr>
1     2   400 Preprocessor1_Model09
```

```
# Final workflow
final_wf_rf <- rf_wf |>
  finalize_workflow(best_rf)
final_wf_rf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────────────────────────
1 Recipe Step

• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Random Forest Model Specification (classification)

Main Arguments:
  mtry = 2
  trees = 400

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```
# Fit the whole training set, then predict the test cases
final_fit_rf <-
  final_wf_rf |>
  last_fit(data_split)
final_fit_rf
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits               id             .metrics .notes   .predictions .workflow
  <list>               <chr>          <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp… <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
final_fit_rf |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
```

```
1 accuracy    binary         0.598 Preprocessor1_Model1
2 roc_auc     binary         0.640 Preprocessor1_Model1
3 brier_class binary         0.235 Preprocessor1_Model1
```

XGBoost

```
library(xgboost)
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

```
gb_recipe <-
  recipe(
    los_long ~ .,
    data = train_data
  ) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_numeric_predictors()) |>
  print()
```

```
── Recipe ──────────────────────────────────────────────────────────────────────


── Inputs

Number of variables by role

outcome:    1
predictor: 22


── Operations

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_numeric_predictors()
```

```
gb_mod <-
  boost_tree(
    mode = "classification",
    trees = 500,
    tree_depth = tune(),
    learn_rate = tune()
  ) |>
```

```
  set_engine("xgboost")
gb_mod
```

```
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 500
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```
gb_wf <- workflow() |>
  add_recipe(gb_recipe) |>
  add_model(gb_mod)
gb_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: boost_tree()

── Preprocessor ────────────────────────────────────────────────────────────────
2 Recipe Steps

• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 500
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```
param_grid <- grid_regular(
  tree_depth(range = c(3L, 10L)),
  learn_rate(range = c(-3, 2), trans = log10_trans()),
  levels = c(3, 10)
  )
param_grid
```

```
# A tibble: 30 × 2
   tree_depth learn_rate
        <int>      <dbl>
 1          3    0.001
 2          6    0.001
```

```
 3         10    0.001
 4          3    0.00359
 5          6    0.00359
 6         10    0.00359
 7          3    0.0129
 8          6    0.0129
 9         10    0.0129
10          3    0.0464
# ℹ 20 more rows
```

Set cross-validation partitions.

```
set.seed(203)

folds <- vfold_cv(train_data, v = 5)
folds
```

```
#  5-fold cross-validation
# A tibble: 5 × 2
  splits               id
  <list>               <chr>
1 <split [37776/9445]> Fold1
2 <split [37777/9444]> Fold2
3 <split [37777/9444]> Fold3
4 <split [37777/9444]> Fold4
5 <split [37777/9444]> Fold5
```

Fit cross-validation.

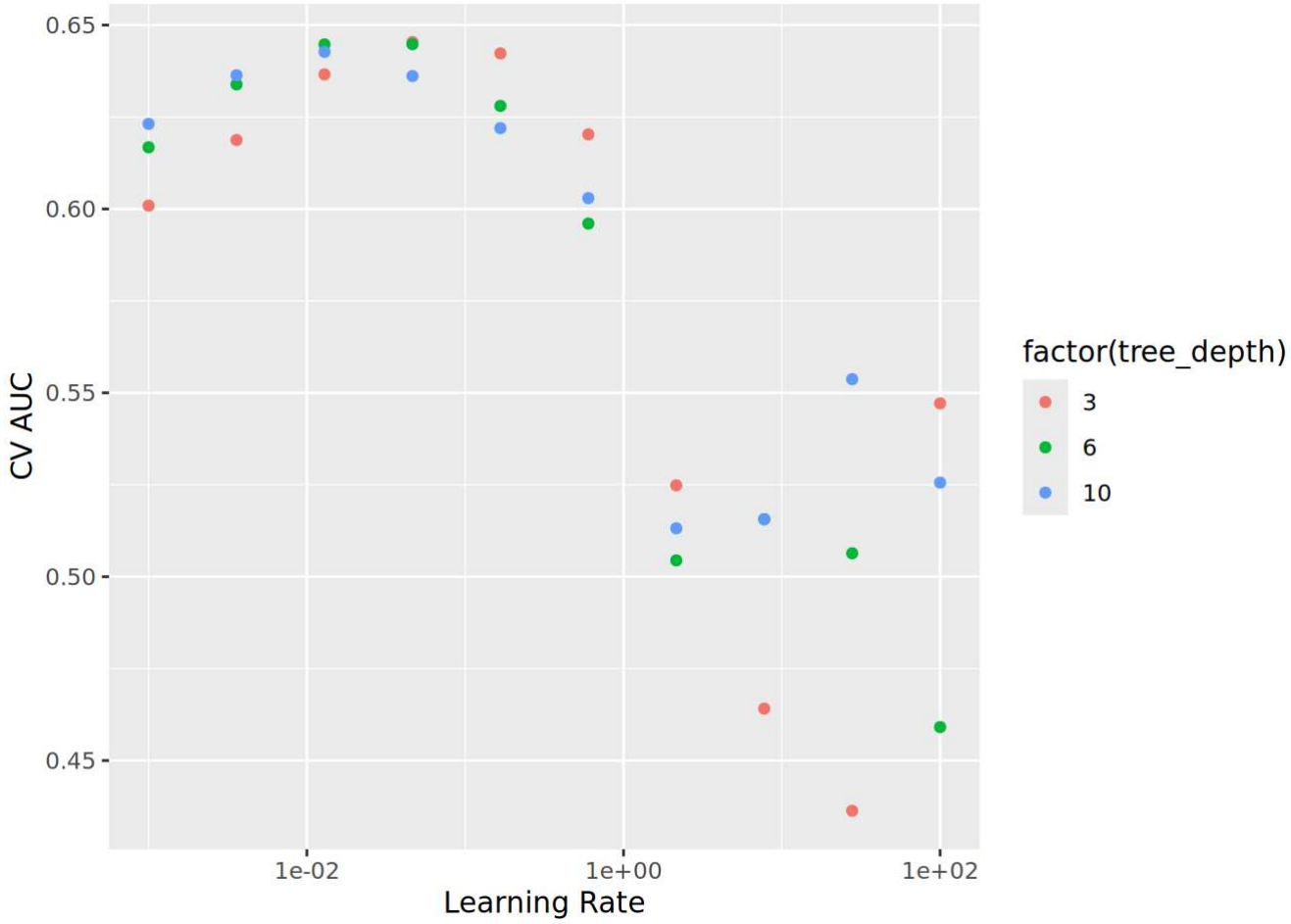
```
gb_fit <- gb_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid,
    metrics = metric_set(roc_auc, accuracy)
    )
gb_fit
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 × 4
  splits               id    .metrics          .notes
  <list>               <chr> <list>            <list>
1 <split [37776/9445]> Fold1 <tibble [60 × 6]> <tibble [0 × 3]>
2 <split [37777/9444]> Fold2 <tibble [60 × 6]> <tibble [0 × 3]>
3 <split [37777/9444]> Fold3 <tibble [60 × 6]> <tibble [0 × 3]>
4 <split [37777/9444]> Fold4 <tibble [60 × 6]> <tibble [0 × 3]>
5 <split [37777/9444]> Fold5 <tibble [60 × 6]> <tibble [0 × 3]>
```

Visualize CV results:

```
gb_fit |>
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = learn_rate, y = mean, color = factor(tree_depth))) +
  geom_point() +
  labs(x = "Learning Rate", y = "CV AUC") +
  scale_x_log10()
```

```
# A tibble: 60 × 8
   tree_depth learn_rate .metric  .estimator  mean     n std_err
        <int>      <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
 1          3    0.001   accuracy binary     0.571     5 0.00294
 2          3    0.001   roc_auc  binary     0.601     5 0.00282
 3          6    0.001   accuracy binary     0.582     5 0.00255
 4          6    0.001   roc_auc  binary     0.617     5 0.00358
 5         10    0.001   accuracy binary     0.586     5 0.00293
 6         10    0.001   roc_auc  binary     0.623     5 0.00364
 7          3    0.00359 accuracy binary     0.582     5 0.00126
 8          3    0.00359 roc_auc  binary     0.619     5 0.00229
 9          6    0.00359 accuracy binary     0.593     5 0.00312
10          6    0.00359 roc_auc  binary     0.634     5 0.00237
   .config
   <chr>
 1 Preprocessor1_Model01
 2 Preprocessor1_Model01
 3 Preprocessor1_Model02
 4 Preprocessor1_Model02
 5 Preprocessor1_Model03
 6 Preprocessor1_Model03
 7 Preprocessor1_Model04
 8 Preprocessor1_Model04
 9 Preprocessor1_Model05
10 Preprocessor1_Model05
# ℹ 50 more rows
```



Show the top 5 models.

```
gb_fit |>
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 × 8
  tree_depth learn_rate .metric .estimator  mean     n std_err .config
       <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1          3     0.0464 roc_auc binary     0.645     5 0.00123 Preprocessor1_Mo…
2          6     0.0464 roc_auc binary     0.645     5 0.00197 Preprocessor1_Mo…
3          6     0.0129 roc_auc binary     0.645     5 0.00175 Preprocessor1_Mo…
4         10     0.0129 roc_auc binary     0.643     5 0.00129 Preprocessor1_Mo…
5          3     0.167  roc_auc binary     0.642     5 0.00199 Preprocessor1_Mo…
```

Select the best model.

```
best_gb <- gb_fit |>
  select_best(metric = "roc_auc")
best_gb
```

```
# A tibble: 1 × 3
  tree_depth learn_rate .config
```

```
      <int>      <dbl> <chr>
1         3     0.0464 Preprocessor1_Model10
```

```
# Final workflow
final_wf_gb <- gb_wf |>
  finalize_workflow(best_gb)
final_wf_gb
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: boost_tree()

── Preprocessor ────────────────────────────────────────────────────────────────
2 Recipe Steps

• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 500
  tree_depth = 3
  learn_rate = 0.0464158883361278

Computational engine: xgboost
```

```
# Fit the whole training set, then predict the test cases
final_fit_gb <-
  final_wf_gb |>
  last_fit(data_split)
final_fit_gb
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits               id             .metrics .notes   .predictions .workflow
  <list>               <chr>          <list>   <list>   <list>       <list>
1 <split [47221/47223]> train/test sp… <tibble> <tibble> <tibble>     <workflow>
```

```
# Test metrics
final_fit_gb |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.600 Preprocessor1_Model1
```

```
2 roc_auc     binary         0.640 Preprocessor1_Model1
3 brier_class binary         0.235 Preprocessor1_Model1
```

Model Stacking

```
library(stacks)
library(keras)
```

```
Attaching package: 'keras'

The following object is masked from 'package:yardstick':

    get_weights
```

```
library(kernlab)
```

```
Attaching package: 'kernlab'

The following object is masked from 'package:dials':

    buffer

The following object is masked from 'package:scales':

    alpha

The following object is masked from 'package:purrr':

    cross

The following object is masked from 'package:ggplot2':

    alpha
```

```
ms_recipe <-
  recipe(
    los_long ~ .,
    data = train_data
  ) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_predictors())
ms_recipe
```

```
── Recipe ──────────────────────────────────────────────────────────────────────
```

```
── Inputs

Number of variables by role

outcome:    1
predictor: 22


── Operations

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_predictors()
```

```
set.seed(203)
folds <- vfold_cv(train_data, v = 5)
```

Logistic Regression

```
logit_mod <-
  logistic_reg(
    penalty = tune(),
    mixture = tune()
  ) |>
  set_engine("glmnet", standardize = TRUE)
logit_mod
```

```
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = TRUE

Computational engine: glmnet
```

```
logit_wf <- workflow() |>
  add_recipe(ms_recipe) |>
  add_model(logit_mod)
logit_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: logistic_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
2 Recipe Steps
```

```
• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Logistic Regression Model Specification (classification)

Main Arguments:
  penalty = tune()
  mixture = tune()

Engine-Specific Arguments:
  standardize = TRUE

Computational engine: glmnet
```

```
logit_grid <- grid_regular(
  penalty(range = c(-6, 3)),
  mixture(),
  levels = c(5, 5)
  )

logit_res <-
  tune_grid(
    object = logit_wf,
    resamples = folds,
    grid = logit_grid,
    control = control_stack_grid()
  )
```

```
ℹ The workflow being saved contains a recipe, which is 6.74 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```
logit_res
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 × 5
  splits               id    .metrics          .notes           .predictions
  <list>               <chr> <list>            <list>           <list>
1 <split [37776/9445]> Fold1 <tibble [75 × 6]> <tibble [0 × 3]> <tibble>
2 <split [37777/9444]> Fold2 <tibble [75 × 6]> <tibble [0 × 3]> <tibble>
3 <split [37777/9444]> Fold3 <tibble [75 × 6]> <tibble [0 × 3]> <tibble>
4 <split [37777/9444]> Fold4 <tibble [75 × 6]> <tibble [0 × 3]> <tibble>
5 <split [37777/9444]> Fold5 <tibble [75 × 6]> <tibble [0 × 3]> <tibble>
```

Random forest

```
rf_mod <-
  rand_forest(
    mode = "classification",
    # Number of predictors randomly sampled in each split
    mtry = tune(),
    # Number of trees in ensemble
    trees = tune()
  ) |>
  set_engine("ranger")
rf_mod
```

```
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Computational engine: ranger
```

```
rf_wf <- workflow() |>
  add_recipe(ms_recipe) |>
  add_model(rf_mod)
rf_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: rand_forest()

── Preprocessor ────────────────────────────────────────────────────────────────
2 Recipe Steps

• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Random Forest Model Specification (classification)

Main Arguments:
  mtry = tune()
  trees = tune()

Computational engine: ranger
```

```
rf_grid <- grid_regular(
  trees(range = c(100L, 500L)),
  mtry(range = c(1L, 5L)),
  levels = c(2, 5)
  )
```

```
rf_res <-
  tune_grid(
    object = rf_wf,
    resamples = folds,
    grid = rf_grid,
    control = control_stack_grid()
  )
```

```
ℹ The workflow being saved contains a recipe, which is 6.74 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```
rf_res
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 × 5
  splits               id    .metrics          .notes           .predictions
  <list>               <chr> <list>            <list>           <list>
1 <split [37776/9445]> Fold1 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
2 <split [37777/9444]> Fold2 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
3 <split [37777/9444]> Fold3 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
4 <split [37777/9444]> Fold4 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
5 <split [37777/9444]> Fold5 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
```

XGBoost

```
gb_mod <-
  boost_tree(
    mode = "classification",
    trees = 500,
    tree_depth = tune(),
    learn_rate = tune()
  ) |>
  set_engine("xgboost")
gb_mod
```

```
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 500
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```
gb_wf <- workflow() |>
  add_recipe(ms_recipe) |>
```

```
  add_model(gb_mod)
gb_wf
```

```
══ Workflow ════════════════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: boost_tree()

── Preprocessor ────────────────────────────────────────────────────────────────
2 Recipe Steps

• step_dummy()
• step_zv()

── Model ───────────────────────────────────────────────────────────────────────
Boosted Tree Model Specification (classification)

Main Arguments:
  trees = 500
  tree_depth = tune()
  learn_rate = tune()

Computational engine: xgboost
```

```
gb_grid <- grid_regular(
  tree_depth(range = c(3L, 10L)),
  learn_rate(range = c(-3, 2), trans = log10_trans()),
  levels = c(2, 5)
  )

gb_res <-
  tune_grid(
    object = gb_wf,
    resamples = folds,
    grid = gb_grid,
    control = control_stack_grid()
  )
```

```
ℹ The workflow being saved contains a recipe, which is 6.74 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```
gb_res
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 × 5
  splits               id    .metrics          .notes           .predictions
  <list>               <chr> <list>            <list>           <list>
1 <split [37776/9445]> Fold1 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
```

```
2 <split [37777/9444]> Fold2 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
3 <split [37777/9444]> Fold3 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
4 <split [37777/9444]> Fold4 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
5 <split [37777/9444]> Fold5 <tibble [30 × 6]> <tibble [0 × 3]> <tibble>
```

Model stacking

```
mimic_model_st <-
  # initialize the stack
  stacks() |>
  # add candidate members
  add_candidates(logit_res) |>
  add_candidates(rf_res) |>
  add_candidates(gb_res) |>
  # determine how to combine their predictions
  blend_predictions(
    penalty = 10^(-6:2),
    metrics = c("roc_auc")
    ) |>
  # fit the candidates with nonzero stacking coefficients
  fit_members()
```

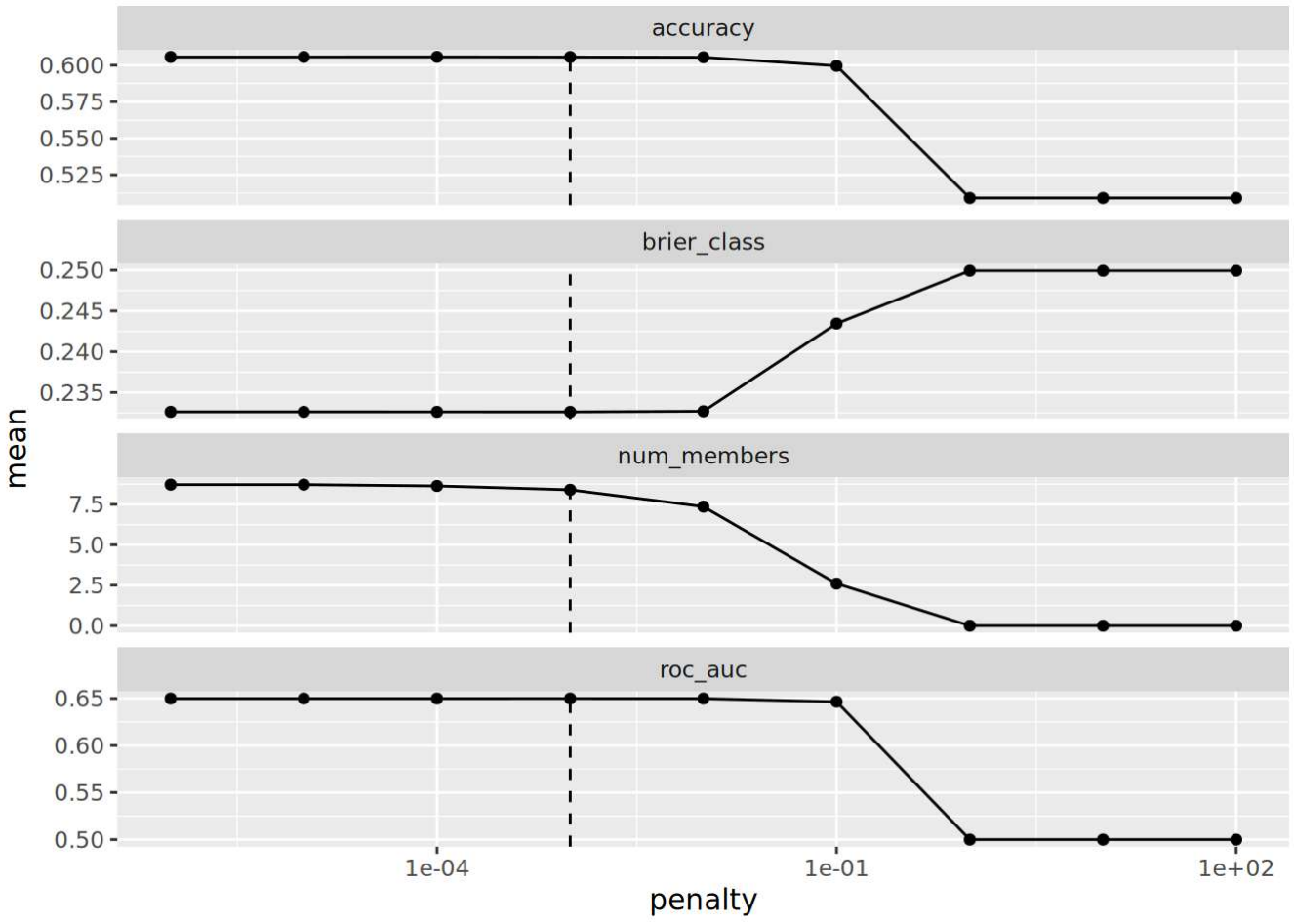
```
Warning: Predictions from 26 candidates were identical to those from existing candidates
and were removed from the data stack.

Warning: The `...` are not used in this function but one or more arguments were
passed: 'metrics'
```

```
mimic_model_st
```

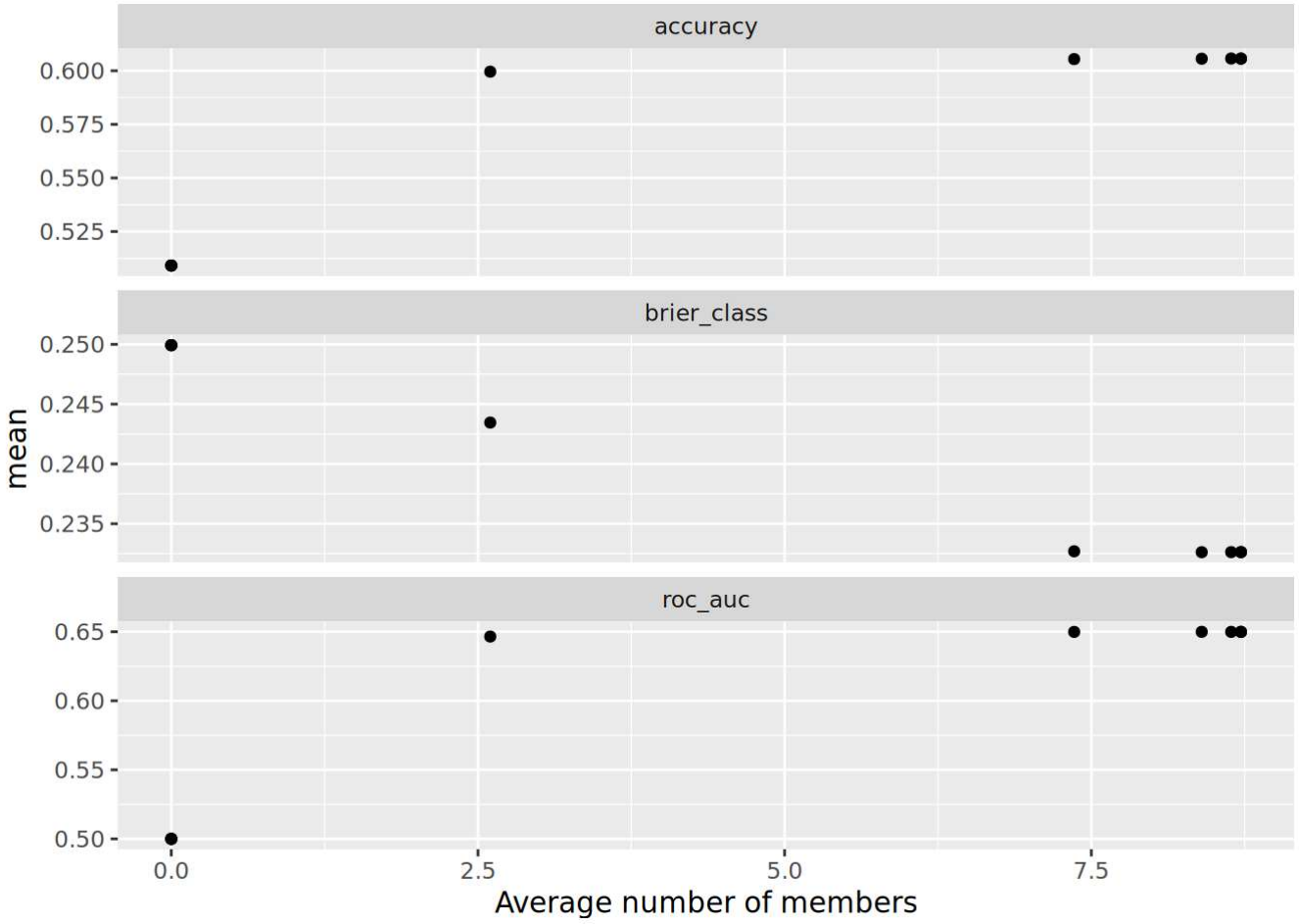
```
── A stacked ensemble model ─────────────────────────────────────


Out of 32 possible candidate members, the ensemble retained 8.

Penalty: 0.001.

Mixture: 1.


The 8 highest weighted member classes are:

# A tibble: 8 × 3
  member                  type         weight
  <chr>                   <chr>         <dbl>
1 .pred_FALSE_rf_res_1_08 rand_forest 1.50
2 .pred_FALSE_rf_res_1_10 rand_forest 1.24
3 .pred_FALSE_gb_res_1_04 boost_tree  0.853
4 .pred_FALSE_gb_res_1_05 boost_tree  0.641
5 .pred_FALSE_gb_res_1_03 boost_tree  0.525
6 .pred_FALSE_rf_res_1_07 rand_forest 0.458
```

```
7 .pred_FALSE_gb_res_1_06 boost_tree  0.128
8 .pred_FALSE_rf_res_1_06 rand_forest 0.00837
```

```
autoplot(mimic_model_st)
```

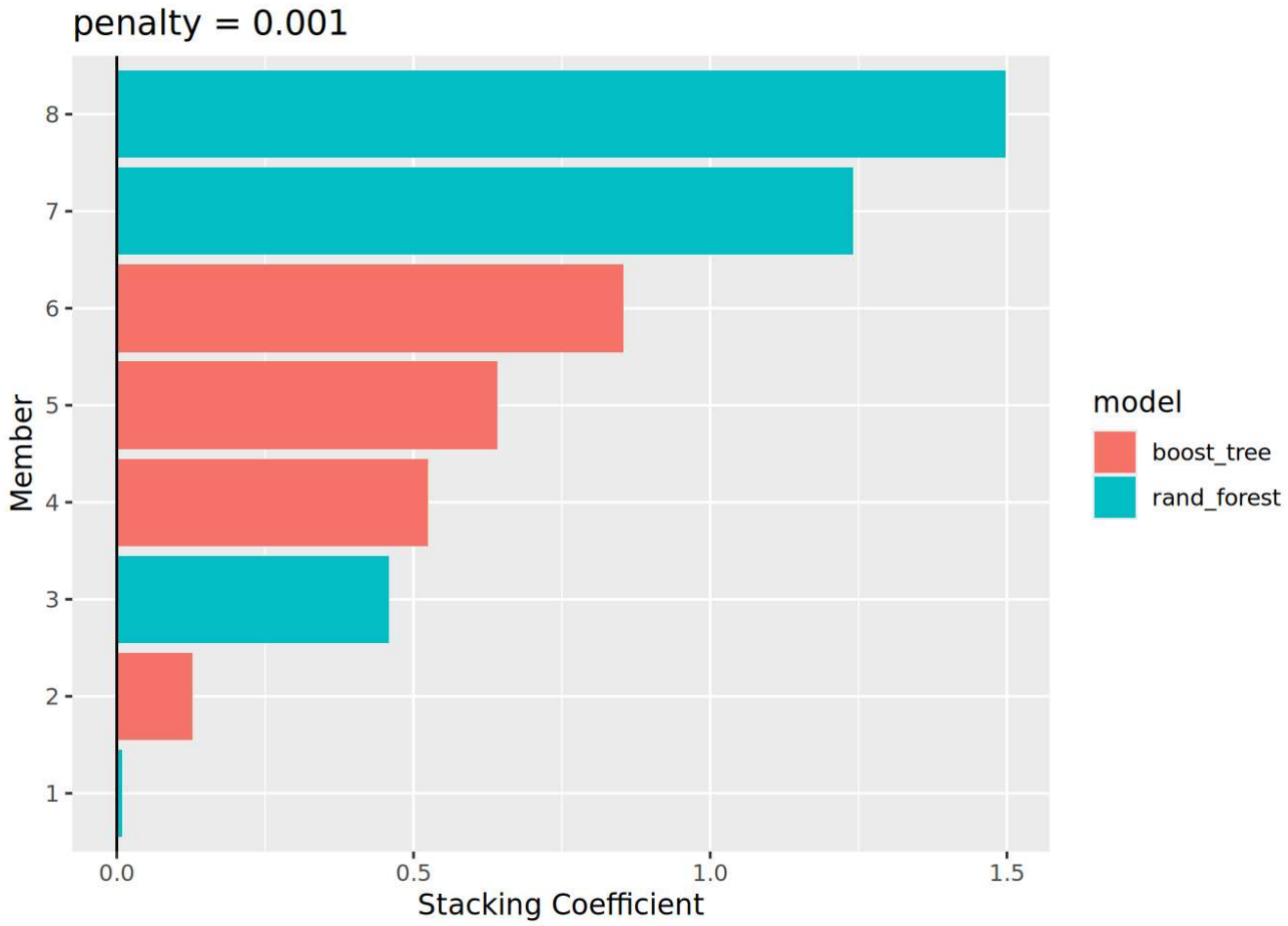



```
autoplot(mimic_model_st, type = "members")
```



Top results:

```
autoplot(mimic_model_st, type = "weights")
```




```
collect_parameters(mimic_model_st, "rf_res")
```

```
# A tibble: 10 × 5
   member       mtry trees terms                      coef
   <chr>       <int> <int> <chr>                     <dbl>
 1 rf_res_1_01     1   100 .pred_FALSE_rf_res_1_01 0
 2 rf_res_1_02     1   500 .pred_FALSE_rf_res_1_02 0
 3 rf_res_1_03     2   100 .pred_FALSE_rf_res_1_03 0
 4 rf_res_1_04     2   500 .pred_FALSE_rf_res_1_04 0
 5 rf_res_1_05     3   100 .pred_FALSE_rf_res_1_05 0
 6 rf_res_1_06     3   500 .pred_FALSE_rf_res_1_06 0.00837
 7 rf_res_1_07     4   100 .pred_FALSE_rf_res_1_07 0.458
 8 rf_res_1_08     4   500 .pred_FALSE_rf_res_1_08 1.50
 9 rf_res_1_09     5   100 .pred_FALSE_rf_res_1_09 0
10 rf_res_1_10     5   500 .pred_FALSE_rf_res_1_10 1.24
```

Final Classification

```
pred <- test_data %>%
  bind_cols(predict(mimic_model_st, ., type = "prob")) %>%
  print(width = Inf)
```

```
# A tibble: 47,223 × 25
   first_careunit                                  admission_type
   <fct>                                           <fct>
 1 Medical Intensive Care Unit (MICU)              EW EMER.
 2 Medical/Surgical Intensive Care Unit (MICU/SICU) EW EMER.
 3 Cardiac Vascular Intensive Care Unit (CVICU)     SURGICAL SAME DAY ADMISSION
 4 Other                                           OBSERVATION ADMIT
 5 Medical Intensive Care Unit (MICU)              EW EMER.
 6 Medical/Surgical Intensive Care Unit (MICU/SICU) EW EMER.
 7 Medical Intensive Care Unit (MICU)              EW EMER.
 8 Other                                           URGENT
 9 Other                                           EW EMER.
10 Cardiac Vascular Intensive Care Unit (CVICU)     OBSERVATION ADMIT
   admission_location     insurance language marital_status race  gender
   <fct>                  <fct>     <fct>    <fct>          <fct> <fct>
 1 EMERGENCY ROOM         Medicaid  English  WIDOWED        WHITE F
 2 Other                  Private   English  MARRIED        WHITE F
 3 PHYSICIAN REFERRAL     Medicare  English  SINGLE         OTHER F
 4 PHYSICIAN REFERRAL     Medicaid  English  MARRIED        OTHER M
 5 EMERGENCY ROOM         Medicare  English  MARRIED        WHITE F
 6 EMERGENCY ROOM         Medicare  English  MARRIED        WHITE F
 7 EMERGENCY ROOM         Medicare  English  WIDOWED        WHITE F
 8 TRANSFER FROM HOSPITAL Medicare  English  WIDOWED        WHITE M
 9 TRANSFER FROM HOSPITAL Private   English  SINGLE         WHITE M
10 PHYSICIAN REFERRAL     Private   English  MARRIED        WHITE F
   age_intime bicarbonate chloride creatinine glucose potassium sodium
        <dbl>       <dbl>    <dbl>      <dbl>   <dbl>     <dbl>  <dbl>
 1         52        25        95         0.7     102       6.7    126
 2         46        24.0      98         1       120       4.1    139
 3         57        24       102         0.9     288       3.5    137
 4         56        18       101.        3.1      95       6.5    125
 5         83        26        85         1.4     133       5.7    120
 6         82        23        98         2.8     117       4.9    135
 7         81        27       111         0.6     173       4.4    144
 8         90        23       102         1.9     105       4.4    140
 9         53        18       106         0.9     269       5.3    135
10         58        24.0     101.        1       120       4.2    138
   hematocrit   wbc heart_rate non_invasive_blood_pressure_diastolic
        <dbl> <dbl>      <dbl>                                 <dbl>
 1       41.1   6.9       91                                     48
 2       34.9   9.3       86                                     56
 3       34.9   7.2       80                                     62
 4       34.3  16.8      110.                                    80
 5       22.4   9.8      114                                     65
 6       25.5  17.9       91                                     51
 7       34.7  10.5      106.                                    51
 8       29.9   5.1       93.5                                   61
 9       43.1  16.9      106                                     99
10       34.9   9.3       80                                     72
   non_invasive_blood_pressure_systolic respiratory_rate temperature_fahrenheit
```

```
                                <dbl>            <dbl>                 <dbl>
 1                               84               24                    98.7
 2                               73               19                    97.7
 3                               98.5             14                    97.2
 4                              112               21                    97.9
 5                              109               24                    97.7
 6                              118               18                    96.9
 7                              102               25                    98.6
 8                              108               22.5                  98.1
 9                              140               12                    96.7
10                              109               17                    99
   los_long .pred_TRUE .pred_FALSE
   <fct>         <dbl>       <dbl>
 1 FALSE         0.596       0.404
 2 FALSE         0.406       0.594
 3 FALSE         0.376       0.624
 4 TRUE          0.633       0.367
 5 FALSE         0.514       0.486
 6 TRUE          0.490       0.510
 7 TRUE          0.456       0.544
 8 TRUE          0.587       0.413
 9 TRUE          0.610       0.390
10 FALSE         0.534       0.466
# ℹ 47,213 more rows
```

```
colnames(pred)
```

```
 [1] "first_careunit"
 [2] "admission_type"
 [3] "admission_location"
 [4] "insurance"
 [5] "language"
 [6] "marital_status"
 [7] "race"
 [8] "gender"
 [9] "age_intime"
[10] "bicarbonate"
[11] "chloride"
[12] "creatinine"
[13] "glucose"
[14] "potassium"
[15] "sodium"
[16] "hematocrit"
[17] "wbc"
[18] "heart_rate"
[19] "non_invasive_blood_pressure_diastolic"
[20] "non_invasive_blood_pressure_systolic"
[21] "respiratory_rate"
[22] "temperature_fahrenheit"
[23] "los_long"
```

```
[24] ".pred_TRUE"
[25] ".pred_FALSE"
```

```
yardstick::roc_auc(
  pred,
  truth = los_long,
  contains(".pred_TRUE")
  )
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.647
```

```
pred <-
  test_data |>
  select(los_long) |>
  bind_cols(
    predict(
      mimic_model_st,
      test_data,
      type = "class",
      members = TRUE
      )
    ) |>
  print(width = Inf)
```

```
# A tibble: 47,223 × 10
   los_long .pred_class .pred_class_rf_res_1_06 .pred_class_rf_res_1_07
   <fct>    <fct>       <fct>                   <fct>
 1 FALSE    TRUE        TRUE                    TRUE
 2 FALSE    FALSE       FALSE                   FALSE
 3 FALSE    FALSE       FALSE                   FALSE
 4 TRUE     TRUE        TRUE                    TRUE
 5 FALSE    TRUE        TRUE                    TRUE
 6 TRUE     FALSE       TRUE                    FALSE
 7 TRUE     FALSE       FALSE                   TRUE
 8 TRUE     TRUE        TRUE                    TRUE
 9 TRUE     TRUE        TRUE                    TRUE
10 FALSE    TRUE        TRUE                    TRUE
   .pred_class_rf_res_1_08 .pred_class_rf_res_1_10 .pred_class_gb_res_1_03
   <fct>                   <fct>                   <fct>
 1 TRUE                    TRUE                    TRUE
 2 FALSE                   FALSE                   FALSE
 3 FALSE                   FALSE                   FALSE
 4 TRUE                    TRUE                    TRUE
 5 TRUE                    TRUE                    TRUE
 6 TRUE                    TRUE                    TRUE
 7 FALSE                   FALSE                   TRUE
```

```
 8 TRUE                    TRUE                    TRUE
 9 TRUE                    TRUE                    TRUE
10 TRUE                    FALSE                   TRUE
   .pred_class_gb_res_1_05 .pred_class_gb_res_1_04 .pred_class_gb_res_1_06
   <fct>                   <fct>                   <fct>
 1 TRUE                    TRUE                    TRUE
 2 FALSE                   FALSE                   FALSE
 3 FALSE                   FALSE                   TRUE
 4 TRUE                    TRUE                    TRUE
 5 FALSE                   FALSE                   TRUE
 6 FALSE                   TRUE                    FALSE
 7 FALSE                   FALSE                   FALSE
 8 TRUE                    TRUE                    TRUE
 9 TRUE                    TRUE                    FALSE
10 TRUE                    TRUE                    TRUE
# ℹ 47,213 more rows
```

```
map(
  colnames(pred),
  ~mean(pred$los_long == pull(pred, .x))
  ) |>
  set_names(colnames(pred)) |>
  as_tibble() |>
  pivot_longer(c(everything(), -los_long))
```

```
# A tibble: 9 × 3
  los_long name                   value
     <dbl> <chr>                  <dbl>
1        1 .pred_class            0.604
2        1 .pred_class_rf_res_1_06 0.599
3        1 .pred_class_rf_res_1_07 0.596
4        1 .pred_class_rf_res_1_08 0.598
5        1 .pred_class_rf_res_1_10 0.600
6        1 .pred_class_gb_res_1_03 0.595
7        1 .pred_class_gb_res_1_05 0.598
8        1 .pred_class_gb_res_1_04 0.601
9        1 .pred_class_gb_res_1_06 0.580
```

Question 4

Compare model classification performance on the test set. Report both the area under ROC curve and accuracy for each machine learning algorithm and the model stacking. Interpret the results. What are the most important features in predicting long ICU stays? How do the models compare in terms of performance and interpretability?

```
library(yardstick)
library(dplyr)

logit_results <- collect_metrics(final_fit_logit) |>
  filter(.metric %in% c("roc_auc", "accuracy")) |>
```

```
  select(.metric, .estimate) |>
  pivot_wider(names_from = .metric, values_from = .estimate) |>
  mutate(model = "Logistic Regression")

rf_results <- collect_metrics(final_fit_rf) |>
  filter(.metric %in% c("roc_auc", "accuracy")) |>
  select(.metric, .estimate) |>
  pivot_wider(names_from = .metric, values_from = .estimate) |>
  mutate(model = "Random Forest")

gb_results <- collect_metrics(final_fit_gb) |>
  filter(.metric %in% c("roc_auc", "accuracy")) |>
  select(.metric, .estimate) |>
  pivot_wider(names_from = .metric, values_from = .estimate) |>
  mutate(model = "XGBoost")

stack_pred_class <- predict(mimic_model_st, new_data = test_data,
                            type = "class")$.pred_class
stack_pred_class <- factor(stack_pred_class,
                           levels = levels(test_data$los_long))

stack_auc <- yardstick::roc_auc(
    test_data |>
      bind_cols(predict(mimic_model_st,  new_data = test_data, type = "prob")),
    truth = los_long,
    contains(".pred_TRUE")
  )$.estimate

stack_acc <- mean(test_data$los_long == stack_pred_class)

stack_results <- tibble(
  model = "Model Stacking",
  accuracy = stack_acc,
  roc_auc = stack_auc
)

model_performance <- bind_rows(logit_results,
                               rf_results,
                               gb_results,
                               stack_results) |>
  select(model, roc_auc, accuracy)
print(model_performance)
```

```
# A tibble: 4 × 3
  model               roc_auc accuracy
  <chr>                 <dbl>    <dbl>
1 Logistic Regression   0.605    0.576
2 Random Forest         0.640    0.598
3 XGBoost               0.640    0.600
4 Model Stacking        0.647    0.604
```

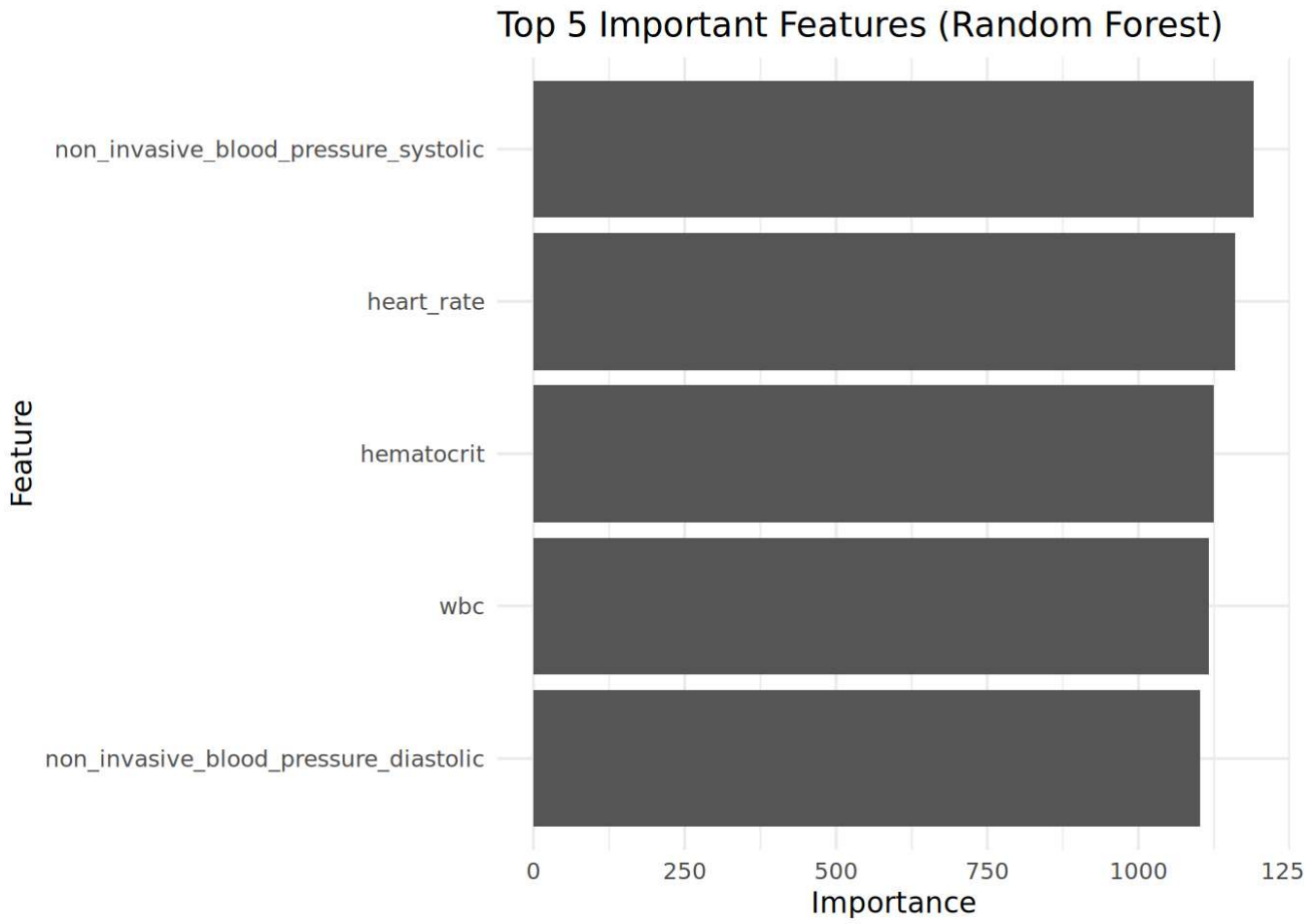
According to the results of model performace, Model Stacking performs the best with the highest roc_auc (0.6474) and accuracy (0.6039). XGBoost (0.6399 roc_auc, 0.5997 accuracy) and Random Forest (0.6396 roc_auc, 0.5995 accuracy) perform similarly and better than Logistic Regression. Logistic Regression performs the worst, with roc_auc (0.6051) and accuracy (0.5761). Model stacking leverages strengths of different models, thus improving the performance of model prediction.

XGBoost and Random Forest outperform Logistic Regression, indicating that non-linear relationships are important in predicting ICU stay length. Model Stacking outperforms all models, meaning a combination of multiple models leads to better predictions.

Besides, according to the Model Stacking Weights Plot, Random Forest dominates the stack, which means Random Forest models had the strongest predictive power in the ensemble. Furthermore, a lower penalty (0.001) means the model gives more weight to all members instead of aggressively shrinking coefficients. This suggests that multiple models contributed meaningfully.

Random Forest feature importance:

```
final_rf_fit <- final_wf_rf |>
  fit(train_data)

rf_model <- final_rf_fit |>
  extract_fit_engine()

rf_importance <- ranger::importance(rf_model)

rf_importance_df <- tibble(
  feature = names(rf_importance),
  importance = rf_importance
) |>
  arrange(desc(importance))

ggplot(rf_importance_df[1:5, ],
       aes(x = reorder(feature, importance), y = importance)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Top 5 Important Features (Random Forest)",
       x = "Feature", y = "Importance") +
  theme_minimal()
```



The top 5 most important features are non_invasive_blood_pressure_systolic, Heart rate, Hematocrit, wbc, and non_invasive_blood_pressure_diastolic.