Biostat 203B Homework 5

Due March 20nd, 2025 @ 11:59PM

AUTHOR
Luke Hodges 906182810

Loading in the necessary libraries and data file

```
# Load required libraries
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(tidymodels)
```

```
── Attaching packages ────────────────────────────────────── tidymodels 1.3.0 ──
✔ broom        1.0.7          ✔ rsample      1.2.1.9000
✔ dials        1.4.0.9000     ✔ tune         1.3.0.9000
✔ infer        1.0.7          ✔ workflows    1.2.0.9000
✔ modeldata    1.4.0          ✔ workflowsets 1.1.0
✔ parsnip      1.3.1.9000     ✔ yardstick    1.3.2
✔ recipes      1.1.1.9000
── Conflicts ───────────────────────────────────────── tidymodels_conflicts() ──
✖ scales::discard() masks purrr::discard()
✖ dplyr::filter()   masks stats::filter()
✖ recipes::fixed()  masks stringr::fixed()
✖ dplyr::lag()      masks stats::lag()
✖ yardstick::spec() masks readr::spec()
✖ recipes::step()   masks stats::step()
```

```
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
```

```
library(GGally)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
library(ranger)
library(gtsummary)
library(stacks)
library(xgboost)
```

```
Attaching package: 'xgboost'

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

    slice
```

```
library(vip)
```

```
Attaching package: 'vip'

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

    vi
```

```
# Load the MIMIC-IV dataset

mimic_icu_cohort <- readRDS("../homework4/mimiciv_shiny/mimic_icu_cohort.rds")

mimic_icu_cohort <- mimic_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id)

mimic_icu_cohort <- mimic_icu_cohort |>
  mutate(los_long = los >= 2)

mimic_icu_cohort$los_long <- as.factor(mimic_icu_cohort$los_long)
```

Logistic Regression

Data preprocessing and feature engineering.

```
#Remove ID Columns and then Select the Predictors
icu_data <- mimic_icu_cohort %>%
  select(subject_id, hadm_id, stay_id, los_long, first_careunit, gender,
         age_intime, marital_status, race, Heart_Rate, DiaBP, SysBP,
         Respiratory_Rate, Temp, Creatinine, Potassium, Chloride, Bicarbonate,
         Hematocrit, WBC, Sodium)

icu_data <- icu_data %>%
  arrange(subject_id, hadm_id, stay_id)


##We need to make sure that los_long is a factor as we got this as an error
icu_data$los_long <- as.factor(icu_data$los_long)

##Now we need to remove all the NA values
icu_data <- icu_data %>%
  drop_na(first_careunit, gender, age_intime, marital_status, race, Heart_Rate,
          DiaBP, SysBP, Respiratory_Rate, Temp, Creatinine, Potassium,
          Chloride, Bicarbonate, Hematocrit, WBC, Sodium)

##We can check to see if there are any NAs here
colSums(is.na(icu_data))
```

```
      subject_id          hadm_id          stay_id         los_long
               0                0                0                0
  first_careunit           gender       age_intime   marital_status
               0                0                0                0
            race       Heart_Rate            DiaBP            SysBP
               0                0                0                0
Respiratory_Rate             Temp       Creatinine        Potassium
               0                0                0                0
        Chloride      Bicarbonate       Hematocrit              WBC
               0                0                0                0
          Sodium
               0
```

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.

```
set.seed(203)

# Stratified split by los_long
icu_split <- initial_split(
  icu_data,
  strata = los_long,
  prop = 0.5
)
```

```r
icu_train <- training(icu_split)
icu_test <- testing(icu_split)
```

```r
head(icu_train)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000980 26913865 39765666 FALSE    Medical Intensive Car… F              76
2   10002155 20345487 32358465 FALSE    Medical Intensive Car… F              83
3   10003019 22774359 30676350 FALSE    Medical/Surgical Inte… M              73
4   10003502 29011269 35796366 FALSE    Coronary Care Unit (C… F              94
5   10004457 23251352 31494479 FALSE    Cardiac Vascular Inte… M              66
6   10005348 25239799 34629895 FALSE    Cardiac Vascular Inte… M              78
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

```r
dim(icu_train)
```

```
[1] 37719    21
```

```r
head(icu_test)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000032 29079034 39553978 FALSE    Medical Intensive Car… F              52
2   10001217 24597018 37067082 FALSE    Surgical Intensive Ca… F              55
3   10001217 27703517 34592300 FALSE    Surgical Intensive Ca… F              55
4   10001843 26133978 39698942 FALSE    Medical/Surgical Inte… M              76
5   10001884 26184834 37510196 TRUE     Medical Intensive Car… F              77
6   10002013 23581541 39060235 FALSE    Cardiac Vascular Inte… F              57
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

```r
dim(icu_test)
```

```
[1] 37720    21
```

```r
##Now, let us make the logit_recipe for the logistic regression model

logit_recipe <- recipe(
  los_long ~ first_careunit + gender + age_intime + marital_status + race +
    Heart_Rate + DiaBP + SysBP + Respiratory_Rate + Temp +
```

```
    Creatinine + Potassium + Chloride + Bicarbonate + Hematocrit +
    WBC + Sodium,
  data = icu_train
) %>%
  step_impute_median(all_numeric_predictors()) %>%
  step_impute_mode(all_nominal_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_numeric_predictors())

logit_recipe
```

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

── Inputs

Number of variables by role

outcome:    1
predictor: 17

── Operations

• Median imputation for: all_numeric_predictors()

• Mode imputation for: all_nominal_predictors()

• Novel factor level assignment for: all_nominal_predictors()

• Dummy variables from: all_nominal_predictors()

• Zero variance filter on: all_predictors()

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

Train and tune the models using the training set.

```
logit_mod <- logistic_reg(
  penalty = tune(),
  mixture = tune()
) %>%
  set_engine("glmnet", standardize = TRUE) %>%
  set_mode("classification") %>%
  print()
```

```
Logistic Regression Model Specification (classification)
```

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

Engine-Specific Arguments:
  standardize = TRUE

Computational engine: glmnet
```

```r
logit_mod
```

```
Logistic Regression Model Specification (classification)

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

Engine-Specific Arguments:
  standardize = TRUE

Computational engine: glmnet
```

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

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

── Preprocessor ────────────────────────────────────────────────────────────────
6 Recipe Steps

• step_impute_median()
• step_impute_mode()
• step_novel()
• step_dummy()
• step_zv()
• step_normalize()

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

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

Engine-Specific Arguments:
```

```
  standardize = TRUE

Computational engine: glmnet
```

```r
logit_wf
```

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

── Preprocessor ────────────────────────────────────────────────────────────────
6 Recipe Steps

• step_impute_median()
• step_impute_mode()
• step_novel()
• step_dummy()
• step_zv()
• step_normalize()

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

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

Engine-Specific Arguments:
  standardize = TRUE

Computational engine: glmnet
```

```r
param_grid <- grid_regular(
  penalty(range = c(-6, 3)),   # log10 scale
  mixture(),
  levels = c(100, 5)
) %>%
  print()
```

```
# A tibble: 500 × 2
      penalty mixture
        <dbl>   <dbl>
 1 0.000001         0
 2 0.00000123       0
 3 0.00000152       0
 4 0.00000187       0
 5 0.00000231       0
 6 0.00000285       0
 7 0.00000351       0
```

```
 8 0.00000433       0
 9 0.00000534       0
10 0.00000658       0
# ℹ 490 more rows
```

```
#We are going to use a v = 3 since my laptop takes a while to load v = 3. We need to make

set.seed(203)
cv_folds <- vfold_cv(icu_train, v = 3)
```

```
(logit_tune <- tune_grid(
  object = logit_wf,
  resamples = cv_folds,
  grid = param_grid,
  metrics = metric_set(roc_auc, accuracy),
  control = control_stack_grid()
)) |>
  system.time()
```

```
   user  system elapsed
 71.484   7.517  79.609
```

```
logit_tune
```

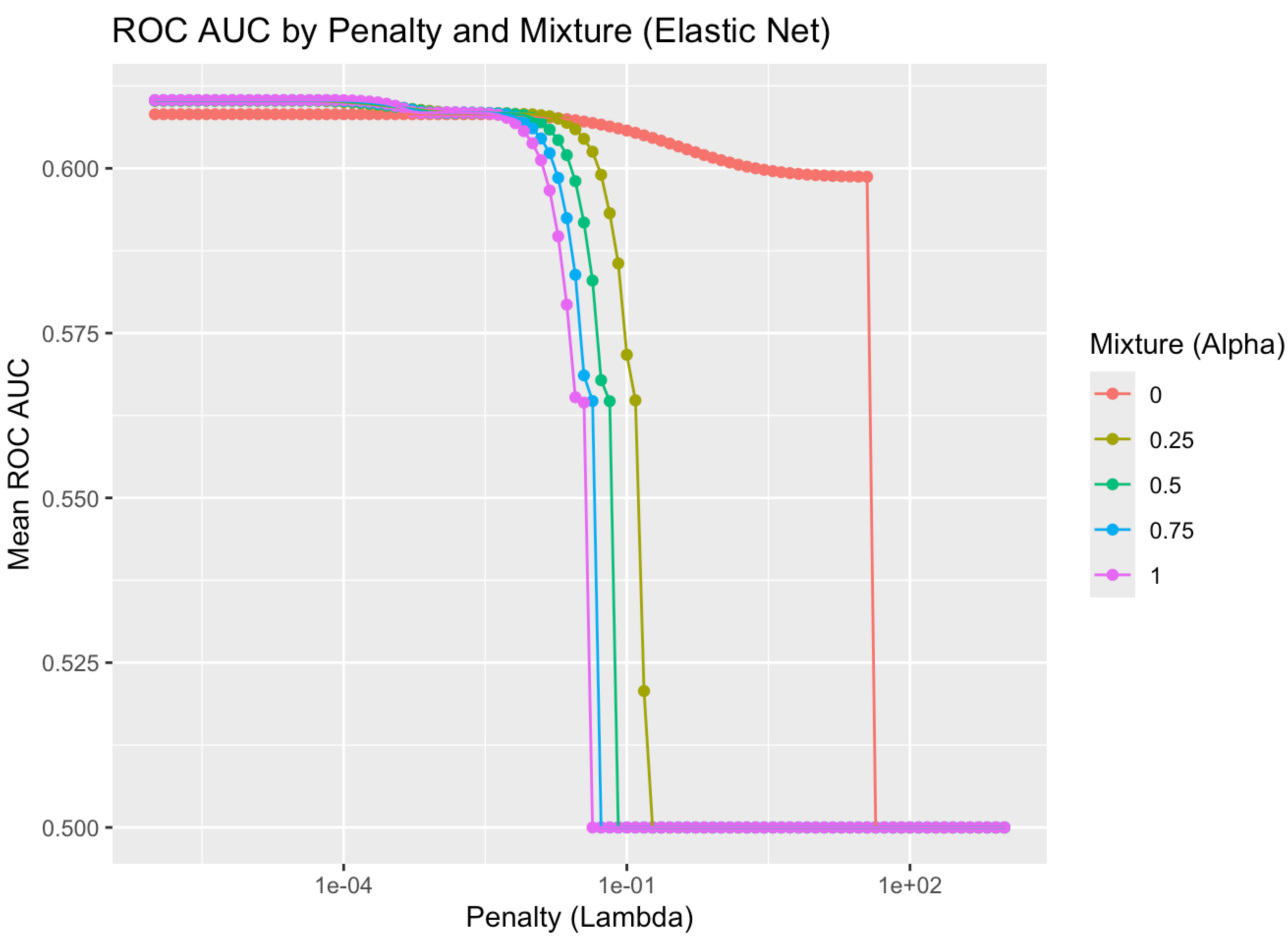
```
# Tuning results
# 3-fold cross-validation
# A tibble: 3 × 5
  splits                id    .metrics             .notes           .predictions
  <list>                <chr> <list>               <list>           <list>
1 <split [25146/12573]> Fold1 <tibble [1,000 × 6]> <tibble [0 × 4]> <tibble>
2 <split [25146/12573]> Fold2 <tibble [1,000 × 6]> <tibble [0 × 4]> <tibble>
3 <split [25146/12573]> Fold3 <tibble [1,000 × 6]> <tibble [0 × 4]> <tibble>
```

```
logit_tune_roc <- logit_tune |>
  collect_metrics() |>
  filter(.metric == "roc_auc")

logit_tune_roc
```

```
# A tibble: 500 × 8
      penalty mixture .metric .estimator  mean     n std_err .config
        <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
 1 0.000001         0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 2 0.00000123       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 3 0.00000152       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 4 0.00000187       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 5 0.00000231       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 6 0.00000285       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
```

```
 7 0.00000351       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 8 0.00000433       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
 9 0.00000534       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
10 0.00000658       0 roc_auc binary     0.608     3 0.00378 Preprocessor1_Mode…
# ℹ 490 more rows
```

```
logit_tune_roc |>
  ggplot(aes(x = penalty, y = mean, color = factor(mixture))) +
  geom_point() +
  geom_line() +
  labs(
    title = "ROC AUC by Penalty and Mixture (Elastic Net)",
    x = "Penalty (Lambda)",
    y = "Mean ROC AUC",
    color = "Mixture (Alpha)"
  ) +
  scale_x_log10()
```



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?

```
show_best(logit_tune, 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         1 roc_auc binary     0.610     3 0.00411 Preprocessor1_Model…
2 0.00000123       1 roc_auc binary     0.610     3 0.00411 Preprocessor1_Model…
3 0.00000152       1 roc_auc binary     0.610     3 0.00411 Preprocessor1_Model…
4 0.00000187       1 roc_auc binary     0.610     3 0.00411 Preprocessor1_Model…
5 0.00000231       1 roc_auc binary     0.610     3 0.00411 Preprocessor1_Model…
```

```
best_logit <- select_best(logit_tune, metric = "roc_auc")
best_logit
```

```
# A tibble: 1 × 3
   penalty mixture .config
     <dbl>   <dbl> <chr>
1 0.000001       1 Preprocessor1_Model401
```

```
final_logit_wf <- finalize_workflow(
  logit_wf,
  best_logit
)

final_logit_fit <- last_fit(
  final_logit_wf,
  split = icu_split
)

# Collect metrics on the test set
collect_metrics(final_logit_fit)
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.579 Preprocessor1_Model1
2 roc_auc     binary         0.608 Preprocessor1_Model1
3 brier_class binary         0.241 Preprocessor1_Model1
```

```
predictions <- collect_predictions(final_logit_fit)
conf_mat(predictions, truth = los_long, estimate = .pred_class)
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 12622  9169
     TRUE   6715  9214
```

```
final_model <- extract_fit_parsnip(final_logit_fit$.workflow[[1]])
tidy(final_model) %>%
```

```
  arrange(desc(estimate)) %>%
  print(n = Inf)
```

```
# A tibble: 64 × 3
   term                                                     estimate penalty
   <chr>                                                       <dbl>   <dbl>
 1 Heart_Rate                                                1.74e-1    1e-6
 2 first_careunit_Neuro.Intermediate                         1.64e-1    1e-6
 3 Respiratory_Rate                                          1.28e-1    1e-6
 4 age_intime                                                1.24e-1    1e-6
 5 WBC                                                       1.11e-1    1e-6
 6 first_careunit_Surgery.Vascular.Intermediate              7.88e-2    1e-6
 7 Creatinine                                                5.77e-2    1e-6
 8 first_careunit_Medicine                                   4.28e-2    1e-6
 9 Sodium                                                    3.09e-2    1e-6
10 gender_M                                                  2.71e-2    1e-6
11 marital_status_SINGLE                                     2.61e-2    1e-6
12 race_HISPANIC.LATINO...CENTRAL.AMERICAN                   2.26e-2    1e-6
13 race_HISPANIC.LATINO...DOMINICAN                          2.11e-2    1e-6
14 race_UNKNOWN                                              1.98e-2    1e-6
15 first_careunit_Intensive.Care.Unit..ICU.                  1.95e-2    1e-6
16 Temp                                                      1.90e-2    1e-6
17 Bicarbonate                                               1.51e-2    1e-6
18 first_careunit_Neuro.Surgical.Intensive.Care.Unit..Neuro.SI… 1.46e-2    1e-6
19 race_ASIAN...KOREAN                                       1.30e-2    1e-6
20 race_PORTUGUESE                                           1.09e-2    1e-6
21 first_careunit_Neuro.Stepdown                             1.02e-2    1e-6
22 race_UNABLE.TO.OBTAIN                                     9.43e-3    1e-6
23 marital_status_MARRIED                                    8.68e-3    1e-6
24 race_BLACK.CARIBBEAN.ISLAND                               7.25e-3    1e-6
25 race_HISPANIC.LATINO...PUERTO.RICAN                       6.40e-3    1e-6
26 Potassium                                                 4.13e-3    1e-6
27 race_MULTIPLE.RACE.ETHNICITY                              3.76e-3    1e-6
28 race_WHITE...BRAZILIAN                                    2.88e-3    1e-6
29 race_HISPANIC.LATINO...HONDURAN                           2.07e-3    1e-6
30 race_BLACK.AFRICAN.AMERICAN                               0          1e-6
31 race_OTHER                                                0          1e-6
32 race_NATIVE.HAWAIIAN.OR.OTHER.PACIFIC.ISLANDER           -3.49e-4    1e-6
33 race_ASIAN...ASIAN.INDIAN                                -2.89e-3    1e-6
34 first_careunit_Surgery.Trauma                            -3.44e-3    1e-6
35 race_ASIAN                                               -3.56e-3    1e-6
36 race_BLACK.AFRICAN                                       -3.59e-3    1e-6
37 first_careunit_PACU                                      -4.46e-3    1e-6
38 race_HISPANIC.LATINO...COLUMBIAN                         -4.85e-3    1e-6
39 race_ASIAN...SOUTH.EAST.ASIAN                            -7.41e-3    1e-6
40 race_PATIENT.DECLINED.TO.ANSWER                          -7.63e-3    1e-6
41 race_WHITE                                               -8.32e-3    1e-6
42 race_HISPANIC.LATINO...MEXICAN                           -9.12e-3    1e-6
43 race_HISPANIC.LATINO...SALVADORAN                        -9.54e-3    1e-6
44 race_SOUTH.AMERICAN                                      -1.08e-2    1e-6
```

```
45 race_HISPANIC.LATINO...GUATEMALAN                                 -1.10e-2    1e-6
46 race_WHITE...RUSSIAN                                              -1.17e-2    1e-6
47 race_HISPANIC.LATINO...CUBAN                                      -1.24e-2    1e-6
48 race_WHITE...EASTERN.EUROPEAN                                     -1.65e-2    1e-6
49 race_WHITE...OTHER.EUROPEAN                                       -1.84e-2    1e-6
50 race_BLACK.CAPE.VERDEAN                                           -1.97e-2    1e-6
51 first_careunit_Med.Surg                                           -2.03e-2    1e-6
52 marital_status_WIDOWED                                            -2.54e-2    1e-6
53 race_HISPANIC.OR.LATINO                                           -2.57e-2    1e-6
54 race_ASIAN...CHINESE                                              -2.60e-2    1e-6
55 first_careunit_Surgical.Intensive.Care.Unit..SICU.                -2.76e-2    1e-6
56 DiaBP                                                             -3.24e-2    1e-6
57 first_careunit_Coronary.Care.Unit..CCU.                           -3.75e-2    1e-6
58 (Intercept)                                                       -6.26e-2    1e-6
59 Chloride                                                          -7.55e-2    1e-6
60 first_careunit_Trauma.SICU..TSICU.                                -8.84e-2    1e-6
61 Hematocrit                                                        -1.28e-1    1e-6
62 first_careunit_Medical.Intensive.Care.Unit..MICU.                 -1.72e-1    1e-6
63 first_careunit_Medical.Surgical.Intensive.Care.Unit..MICU.S… -2.04e-1    1e-6
64 SysBP                                                             -2.41e+0    1e-6
```

The Accuracy is 0.579; the ROC AUC is 0.607; and the Brier score is 0.241. Regrading accuracy, this means that 57.9% of the time, the model correctly predicts whether or not a patient has a long or short ICU stay based on the features examined. This means that this is better than just randomly guessing, which would give more of a 50-50 split.

The ROC AUC indicates that the model is able to distinguish ICU stays between long and short in a modest way.

The Top Five Most Important Features Are: Heart Rate (+0.174), Respiratory Rate (+0.128), Age (+0.124), Neuro Intermediate Care Service (+0.164). In other words, those with a higher heart_rate, respiratory rate, age, and also placed into the neuro intermediate care service had a higher chance of having a longer ICU stay. This makes sense considering that higher_heart rate and respiratory rates are a sign of distress, while older age is linked to frailty and needs more attention than individuals that are younger. It should also be noted that the SysBP is -2.413, which means higher SysBP is indicated with lower length of stays. This is quiet odd and is not something I would have expected as high blood pressure (although that does involve looking at the DiaBP as well). Looking into it some more, a higher SysBP could indicate more stability cardiovascularly, so theyw ould not need as much attention

Random Forest

Data Preprocessing and engineering

```
#Remove ID Columns and then Select the Predictors
icu_data <- mimic_icu_cohort %>%
  select(subject_id, hadm_id, stay_id, los_long, first_careunit, gender,
         age_intime, marital_status, race, Heart_Rate, DiaBP, SysBP,
         Respiratory_Rate, Temp, Creatinine, Potassium, Chloride, Bicarbonate,
```

```
         Hematocrit, WBC, Sodium)

icu_data <- icu_data %>%
  arrange(subject_id, hadm_id, stay_id)


##We need to make sure that los_long is a factor as we got this as an error
icu_data$los_long <- as.factor(icu_data$los_long)

##Now we need to remove all the NA values
icu_data <- icu_data %>%
  drop_na(first_careunit, gender, age_intime, marital_status, race, Heart_Rate,
          DiaBP, SysBP, Respiratory_Rate, Temp, Creatinine, Potassium,
          Chloride, Bicarbonate, Hematocrit, WBC, Sodium)

##We can check to see if there are any NAs here
colSums(is.na(icu_data))
```

```
      subject_id          hadm_id          stay_id         los_long 
               0                0                0                0 
  first_careunit           gender       age_intime   marital_status 
               0                0                0                0 
            race       Heart_Rate            DiaBP            SysBP 
               0                0                0                0 
Respiratory_Rate             Temp       Creatinine        Potassium 
               0                0                0                0 
        Chloride      Bicarbonate       Hematocrit              WBC 
               0                0                0                0 
          Sodium 
               0 
```

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.

```
set.seed(203)

# Stratified split by los_long
icu_split <- initial_split(
  icu_data,
  strata = los_long,
  prop = 0.5
)

icu_train <- training(icu_split)
icu_test <- testing(icu_split)

head(icu_train)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
```

```
      <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000980 26913865 39765666 FALSE    Medical Intensive Car… F              76
2   10002155 20345487 32358465 FALSE    Medical Intensive Car… F              83
3   10003019 22774359 30676350 FALSE    Medical/Surgical Inte… M              73
4   10003502 29011269 35796366 FALSE    Coronary Care Unit (C… F              94
5   10004457 23251352 31494479 FALSE    Cardiac Vascular Inte… M              66
6   10005348 25239799 34629895 FALSE    Cardiac Vascular Inte… M              78
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

```r
head(icu_test)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000032 29079034 39553978 FALSE    Medical Intensive Car… F              52
2   10001217 24597018 37067082 FALSE    Surgical Intensive Ca… F              55
3   10001217 27703517 34592300 FALSE    Surgical Intensive Ca… F              55
4   10001843 26133978 39698942 FALSE    Medical/Surgical Inte… M              76
5   10001884 26184834 37510196 TRUE     Medical Intensive Car… F              77
6   10002013 23581541 39060235 FALSE    Cardiac Vascular Inte… F              57
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

Train and tune the models using the training set.

```r
rf_recipe <- recipe(
  los_long ~ first_careunit + gender + age_intime + marital_status + race +
    Heart_Rate + DiaBP + SysBP + Respiratory_Rate + Temp +
    Creatinine + Potassium + Chloride + Bicarbonate + Hematocrit +
    WBC + Sodium,
  data = icu_train
) %>%
  step_impute_median(all_numeric_predictors()) %>%
  step_impute_mode(all_nominal_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())
```

```r
rf_mod <- rand_forest(
  mode = "classification",
  mtry = tune(),
  trees = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification") %>%
  print()
```

```
Random Forest Model Specification (classification)

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

Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```
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 ────────────────────────────────────────────────────────────────
5 Recipe Steps

• step_impute_median()
• step_impute_mode()
• step_novel()
• step_dummy()
• step_zv()

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

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

```
Engine-Specific Arguments:
  importance = impurity

Computational engine: ranger
```

```r
rf_params <- hardhat::extract_parameter_set_dials(rf_mod)

# Define a smaller tuning grid for faster search

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

```r
set.seed(203)

#We are going to keep at it 3 for consistency, as mentioned before.

cv_folds <- vfold_cv(icu_train, v = 3)

rf_tune <- tune_grid(
  object = rf_wf,
  resamples = cv_folds,
  grid = rf_grid,
  metrics = metric_set(roc_auc, accuracy),
  control = control_stack_grid()
)

rf_tune
```

```
# Tuning results
# 3-fold cross-validation
# A tibble: 3 × 5
  splits                id    .metrics          .notes           .predictions
  <list>                <chr> <list>            <list>           <list>
1 <split [25146/12573]> Fold1 <tibble [50 × 6]> <tibble [0 × 4]> <tibble>
2 <split [25146/12573]> Fold2 <tibble [50 × 6]> <tibble [0 × 4]> <tibble>
3 <split [25146/12573]> Fold3 <tibble [50 × 6]> <tibble [0 × 4]> <tibble>
```

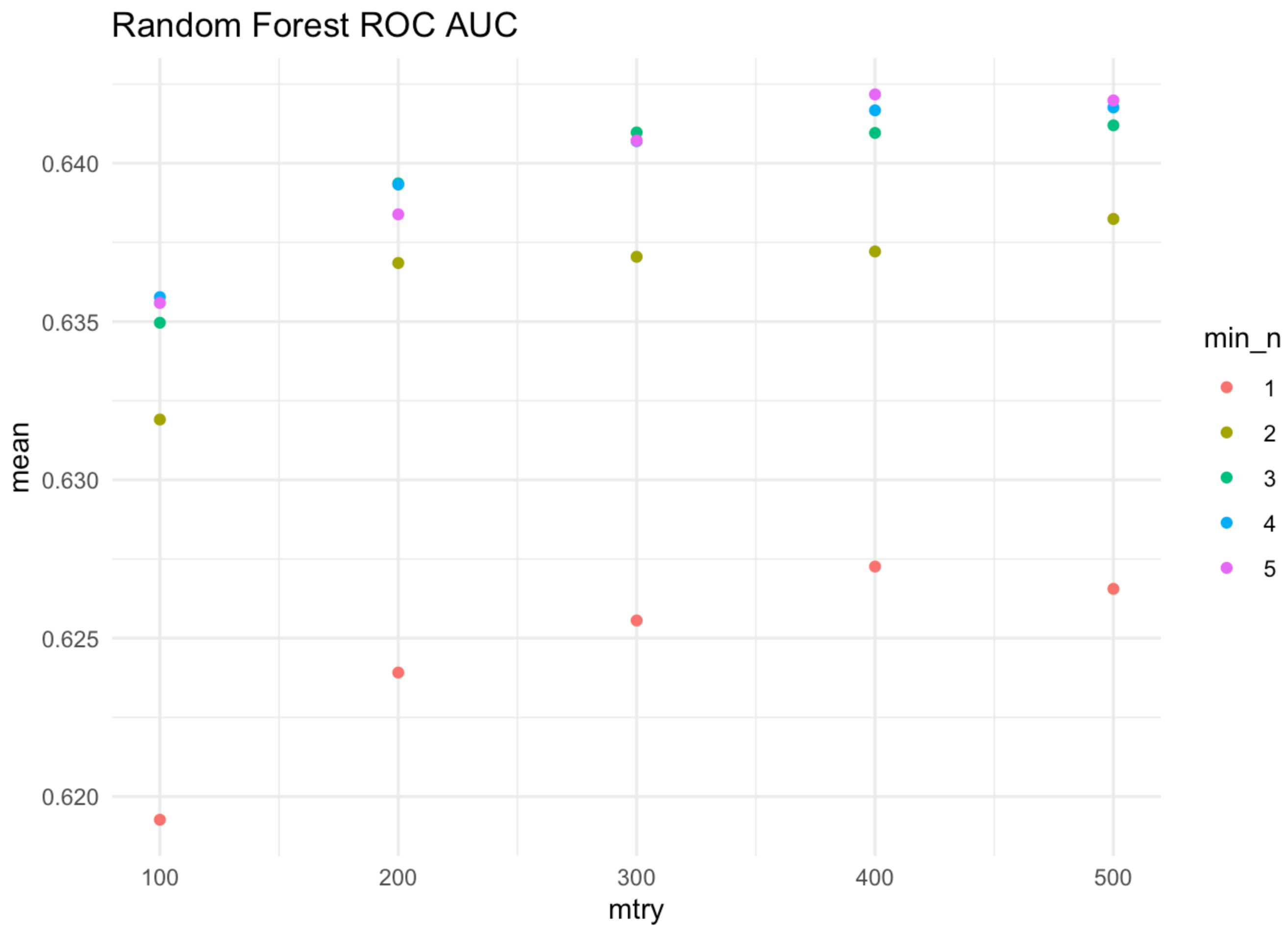
```r
collect_metrics(rf_tune)
```

```
# 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     0.570     3 0.00610 Preprocessor1_Model01
 2     1   100 roc_auc  binary     0.619     3 0.00197 Preprocessor1_Model01
 3     1   200 accuracy binary     0.575     3 0.00620 Preprocessor1_Model02
```

```
 4     1   200 roc_auc  binary     0.624     3 0.00210 Preprocessor1_Model02
 5     1   300 accuracy binary     0.573     3 0.00658 Preprocessor1_Model03
 6     1   300 roc_auc  binary     0.626     3 0.00290 Preprocessor1_Model03
 7     1   400 accuracy binary     0.576     3 0.00705 Preprocessor1_Model04
 8     1   400 roc_auc  binary     0.627     3 0.00302 Preprocessor1_Model04
 9     1   500 accuracy binary     0.572     3 0.00598 Preprocessor1_Model05
10     1   500 roc_auc  binary     0.627     3 0.00351 Preprocessor1_Model05
# ℹ 40 more rows
```

```r
# Plot ROC AUC vs mtry and min_n

rf_tune %>%
  collect_metrics() |>
  print(width = Inf) |>
  filter(.metric == "roc_auc") |>
  ggplot(mapping = aes(x = trees, y = mean, color = factor(mtry))) +
  geom_point() +
  labs(
    title = "Random Forest ROC AUC",
    x = "mtry",
    color = "min_n"
  ) +
  theme_minimal()
```

```
# 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     0.570     3 0.00610 Preprocessor1_Model01
 2     1   100 roc_auc  binary     0.619     3 0.00197 Preprocessor1_Model01
 3     1   200 accuracy binary     0.575     3 0.00620 Preprocessor1_Model02
 4     1   200 roc_auc  binary     0.624     3 0.00210 Preprocessor1_Model02
 5     1   300 accuracy binary     0.573     3 0.00658 Preprocessor1_Model03
 6     1   300 roc_auc  binary     0.626     3 0.00290 Preprocessor1_Model03
 7     1   400 accuracy binary     0.576     3 0.00705 Preprocessor1_Model04
 8     1   400 roc_auc  binary     0.627     3 0.00302 Preprocessor1_Model04
 9     1   500 accuracy binary     0.572     3 0.00598 Preprocessor1_Model05
10     1   500 roc_auc  binary     0.627     3 0.00351 Preprocessor1_Model05
# ℹ 40 more rows
```



This indicates that when the mtry is around 400 to 500, we received the highest ROC AUC of about 0.6425.

```
# Select best hyperparameters (corrected)
best_rf <- select_best(rf_tune, metric = "roc_auc")

rf_tune |>
  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     5   400 roc_auc binary     0.642     3 0.00251 Preprocessor1_Model24
2     5   500 roc_auc binary     0.642     3 0.00286 Preprocessor1_Model25
3     4   500 roc_auc binary     0.642     3 0.00251 Preprocessor1_Model20
4     4   400 roc_auc binary     0.642     3 0.00306 Preprocessor1_Model19
5     3   500 roc_auc binary     0.641     3 0.00304 Preprocessor1_Model15
```

The best metric seems to be when the mtry is five and the trees are 400. The Roc_AUC mean is 0.642.

```
# Finalize the workflow with the best hyperparameters
final_rf_wf <- finalize_workflow(rf_wf, best_rf)

# Fit the final model on the training set and evaluate on the test set
final_rf_fit <- final_rf_wf |>
```

```
  last_fit(icu_split)

final_rf_fit
```

```
# Resampling results
# Manual resampling
# A tibble: 1 × 6
  splits                id              .metrics .notes   .predictions .workflow
  <list>                <chr>           <list>   <list>   <list>       <list>
1 <split [37719/37720]> train/test sp… <tibble> <tibble> <tibble>     <workflow>
```

```
# Collect metrics on the test set
collect_metrics(final_rf_fit)
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.603 Preprocessor1_Model1
2 roc_auc     binary         0.643 Preprocessor1_Model1
3 brier_class binary         0.235 Preprocessor1_Model1
```

```
# Generate predictions on the test set
predictions_rf <- collect_predictions(final_rf_fit)
conf_mat(predictions_rf, truth = los_long, estimate = .pred_class)
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 12555  8194
     TRUE   6782 10189
```

Now, let us extract the importance of each of the variables

```
final_modelrf <- extract_fit_parsnip(final_rf_fit$.workflow[[1]])

importance_df <- final_modelrf$fit$variable.importance %>%
  enframe(name = "feature", value = "importance") %>%
  arrange(desc(importance))

print(importance_df, n = Inf)
```

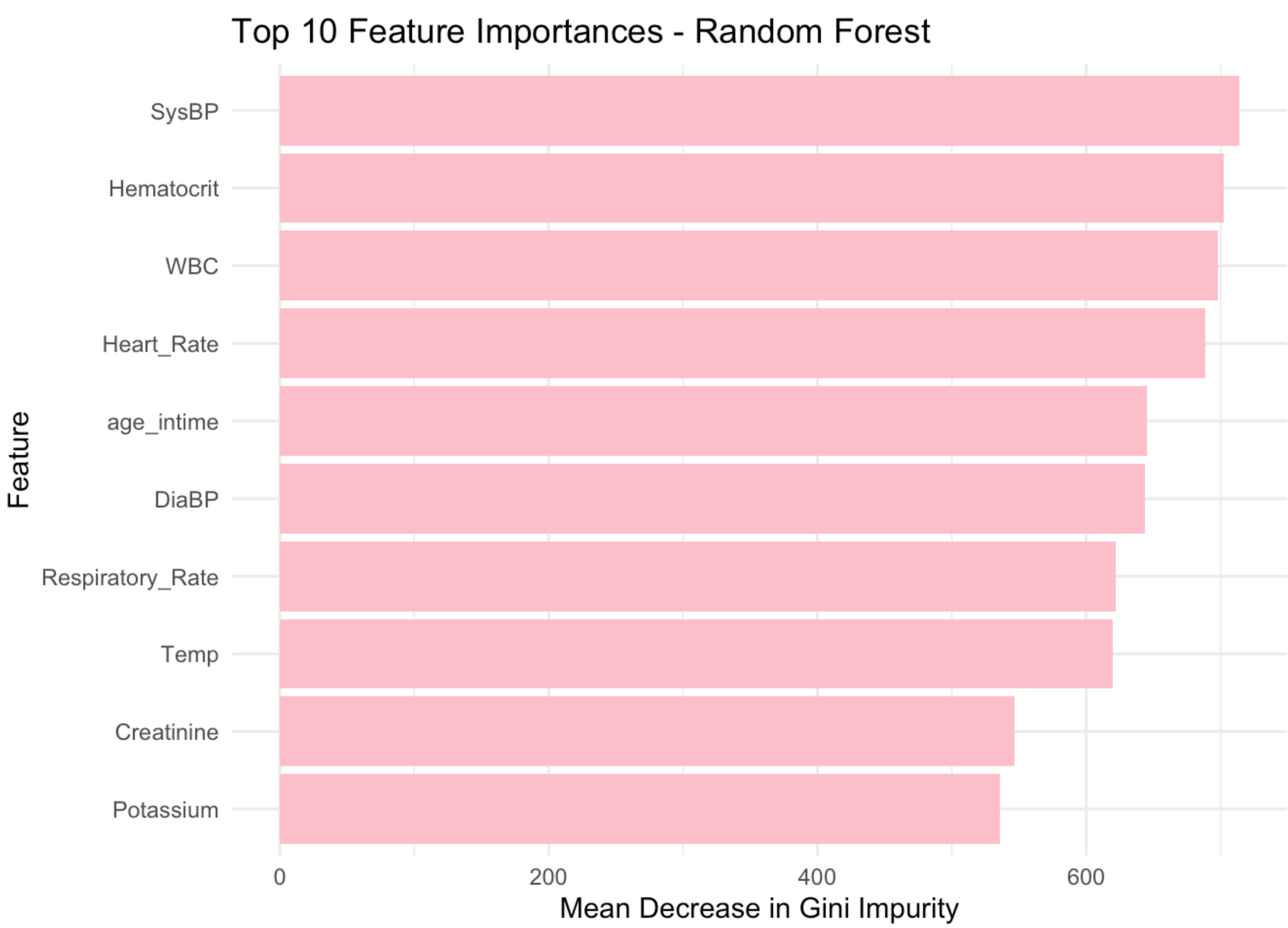
```
# A tibble: 63 × 2
   feature                                                           importance
   <chr>                                                                  <dbl>
 1 SysBP                                                                   714.
 2 Hematocrit                                                              702.
 3 WBC                                                                     698.
 4 Heart_Rate                                                              688.
 5 age_intime                                                              645.
 6 DiaBP                                                                   644.
 7 Respiratory_Rate                                                        622.
```

```
 8 Temp                                                        620.
 9 Creatinine                                                  547.
10 Potassium                                                   536.
11 Bicarbonate                                                 514.
12 Chloride                                                    511.
13 Sodium                                                      501.
14 first_careunit_Neuro.Intermediate                           129.
15 gender_M                                                    102.
16 race_WHITE                                                   95.3
17 first_careunit_Medical.Surgical.Intensive.Care.Unit..MICU.SICU.  92.4
18 first_careunit_Medical.Intensive.Care.Unit..MICU.            90.4
19 marital_status_MARRIED                                       88.5
20 marital_status_SINGLE                                        81.9
21 race_BLACK.AFRICAN.AMERICAN                                  72.3
22 first_careunit_Surgical.Intensive.Care.Unit..SICU.           69.4
23 marital_status_WIDOWED                                       66.1
24 first_careunit_Coronary.Care.Unit..CCU.                      63.8
25 first_careunit_Trauma.SICU..TSICU.                           62.7
26 race_OTHER                                                   50.4
27 race_WHITE...OTHER.EUROPEAN                                  43.4
28 race_UNKNOWN                                                 42.3
29 first_careunit_Neuro.Surgical.Intensive.Care.Unit..Neuro.SICU.  28.8
30 race_ASIAN                                                   28.7
31 race_HISPANIC.LATINO...PUERTO.RICAN                          28.5
32 race_ASIAN...CHINESE                                         28.0
33 first_careunit_Neuro.Stepdown                                27.0
34 race_WHITE...RUSSIAN                                         25.8
35 race_HISPANIC.LATINO...DOMINICAN                             21.6
36 race_BLACK.CAPE.VERDEAN                                      21.1
37 race_HISPANIC.OR.LATINO                                      19.7
38 race_UNABLE.TO.OBTAIN                                        18.5
39 race_BLACK.CARIBBEAN.ISLAND                                  17.1
40 race_BLACK.AFRICAN                                           16.0
41 race_ASIAN...SOUTH.EAST.ASIAN                                15.4
42 race_PORTUGUESE                                              14.7
43 race_PATIENT.DECLINED.TO.ANSWER                              14.1
44 race_ASIAN...ASIAN.INDIAN                                    10.4
45 race_WHITE...EASTERN.EUROPEAN                                10.2
46 first_careunit_Surgery.Vascular.Intermediate                  9.04
47 race_WHITE...BRAZILIAN                                        8.64
48 race_HISPANIC.LATINO...GUATEMALAN                             8.03
49 race_HISPANIC.LATINO...SALVADORAN                             6.24
50 race_SOUTH.AMERICAN                                           5.47
51 first_careunit_PACU                                           4.66
52 race_HISPANIC.LATINO...COLUMBIAN                              4.62
53 race_HISPANIC.LATINO...CUBAN                                  4.53
54 race_HISPANIC.LATINO...MEXICAN                                4.44
55 race_HISPANIC.LATINO...CENTRAL.AMERICAN                       3.65
56 race_MULTIPLE.RACE.ETHNICITY                                  3.45
57 race_NATIVE.HAWAIIAN.OR.OTHER.PACIFIC.ISLANDER                3.19
58 race_HISPANIC.LATINO...HONDURAN                               3.08
```

```
59 race_ASIAN...KOREAN                                    2.45
60 first_careunit_Intensive.Care.Unit..ICU.               1.33
61 first_careunit_Medicine                                0.500
62 first_careunit_Surgery.Trauma                          0.299
63 first_careunit_Med.Surg                                0.186
```

Let us graph it for better visualization using GGPlot

```r
importance_df %>%
  top_n(10, wt = importance) %>%
  ggplot(aes(x = reorder(feature, importance), y = importance)) +
  geom_col(fill = "pink") +
  coord_flip() +
  labs(
    title = "Top 10 Feature Importances - Random Forest",
    x = "Feature",
    y = "Mean Decrease in Gini Impurity"
  ) +
  theme_minimal()
```



Based on all the results, the accuracy of the RF model is 0.60, while the ROC AUC is 0.643. This means that the model correctly predicted whether or not a patient was going to stay at longer than or equal to two days about 60% of the time. The ROC AUC illustrates that the model is somewhat effective at distinguishing between long and short stays, but is not the most effective at doing so. Considering that the Logit

Regression's ROC AUC and Accuracy was 0.61 and 0.579, respectively, the Random Forest model is slightly better at predicting long ICU stays.

Looking at the importance of variables, SysBP, Hematocrit, WBC, Heart_Rate, and Age_intime were the most important features in determining whether or not the los_long was greater than or equal to two days. SysBP makes sense considering that this could indicate cardiovascular problems, while Hematocrit may indicate anemia from lack of red blood cells needed for oxygen transportation and survival. However, as we saw with the logistic regression, the SysBP was indicative of a lower LOS, so the importantce makes sense as well as it would influence the model results. WBC counts could also indicate infection if there is an elevated amount as well, which would make an individual stay longer. Lastly, age_intime makes sense considering older patients are more frail and need to be watched more closely, the same would go hand-in-hand with Heart_rate as well; higher heart_rates indicate some sort of stress occurring in the body, such as an infection or cardiovascular condition.

Comparing this to the Logistic Regression as well, the only similarities are: Heart_Rate, age_intime, WBC, and SysBP with varying degrees of importance to each respective model.

XGBoost

Data Preprocessing and engineering

```
#Remove ID Columns and then Select the Predictors
icu_data <- mimic_icu_cohort %>%
  select(subject_id, hadm_id, stay_id, los_long, first_careunit, gender,
         age_intime, marital_status, race, Heart_Rate, DiaBP, SysBP,
         Respiratory_Rate, Temp, Creatinine, Potassium, Chloride, Bicarbonate,
         Hematocrit, WBC, Sodium)

icu_data <- icu_data %>%
  arrange(subject_id, hadm_id, stay_id)


##We need to make sure that los_long is a factor as we got this as an error
icu_data$los_long <- as.factor(icu_data$los_long)

##Now we need to remove all the NA values
icu_data <- icu_data %>%
  drop_na(first_careunit, gender, age_intime, marital_status, race, Heart_Rate,
          DiaBP, SysBP, Respiratory_Rate, Temp, Creatinine, Potassium,
          Chloride, Bicarbonate, Hematocrit, WBC, Sodium)

##We can check to see if there are any NAs here
colSums(is.na(icu_data))
```

```
     subject_id          hadm_id          stay_id         los_long
              0                0                0                0
 first_careunit           gender       age_intime   marital_status
```

```
              0                0                0                0
           race       Heart_Rate            DiaBP            SysBP
              0                0                0                0
Respiratory_Rate             Temp       Creatinine        Potassium
              0                0                0                0
       Chloride      Bicarbonate       Hematocrit              WBC
              0                0                0                0
         Sodium
              0
```

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.

```r
set.seed(203)

icu_split <- initial_split(
  icu_data,
  strata = los_long,
  prop = 0.5
)

icu_train <- training(icu_split)
icu_test <- testing(icu_split)

head(icu_train)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000980 26913865 39765666 FALSE    Medical Intensive Car… F              76
2   10002155 20345487 32358465 FALSE    Medical Intensive Car… F              83
3   10003019 22774359 30676350 FALSE    Medical/Surgical Inte… M              73
4   10003502 29011269 35796366 FALSE    Coronary Care Unit (C… F              94
5   10004457 23251352 31494479 FALSE    Cardiac Vascular Inte… M              66
6   10005348 25239799 34629895 FALSE    Cardiac Vascular Inte… M              78
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

```r
head(icu_test)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000032 29079034 39553978 FALSE    Medical Intensive Car… F              52
2   10001217 24597018 37067082 FALSE    Surgical Intensive Ca… F              55
3   10001217 27703517 34592300 FALSE    Surgical Intensive Ca… F              55
4   10001843 26133978 39698942 FALSE    Medical/Surgical Inte… M              76
5   10001884 26184834 37510196 TRUE     Medical Intensive Car… F              77
```

```
6   10002013 23581541 39060235 FALSE    Cardiac Vascular Inte… F             57
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

Train and Tune the Models Using the Training Set

```
XGBoostRecipe <- recipe(
  los_long ~ first_careunit + gender + age_intime + marital_status + race +
    Heart_Rate + DiaBP + SysBP + Respiratory_Rate + Temp +
    Creatinine + Potassium + Chloride + Bicarbonate + Hematocrit +
    WBC + Sodium,
  data = icu_train
) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_impute_mode(all_nominal_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())
```

Model specificaiton process with tuning paramters

```
xgb_mod <- boost_tree(
  mode = "classification",
  trees = tune(),
  tree_depth = tune(),
  learn_rate = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

xgb_mod
```

```
Boosted Tree Model Specification (classification)

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

Computational engine: xgboost
```

Now let us bundle the recipe we did wih our model into a workflow

```
xgb_wf <- workflow() |>
  add_recipe(XGBoostRecipe) |>
  add_model(xgb_mod)

xgb_wf
```

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

── Preprocessor ────────────────────────────────────────────────────────────
5 Recipe Steps

• step_impute_mean()
• step_impute_mode()
• step_novel()
• step_dummy()
• step_zv()

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

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

Computational engine: xgboost
```

Now we will define the grid for tuning

```
xgb_params <- hardhat::extract_parameter_set_dials(xgb_mod)

xgb_grid <- grid_regular(
  trees(range = c(100L, 500L)),
  tree_depth(range = c(1L, 3L)),
  learn_rate(range = c(-5, 2), trans = log10_trans()),
  levels = c(3, 3, 5)
  )
```

Now we will perform the tuning

```
set.seed(203)

#Kept it consistent; v = 3 was also approved by Dr. Zhou in office hours to speed up my r
xgbcv_folds <- vfold_cv(icu_train, v = 3)

xgb_tune <- tune_grid(
  object = xgb_wf,
  resamples = xgbcv_folds,
  grid = xgb_grid,
  metrics = metric_set(roc_auc, accuracy),
  control = control_stack_grid()
)
```

```r
best_xgb <- xgb_tune |>
  select_best(metric = "roc_auc")
```

```r
final_xgb_wf <- xgb_wf |>
  finalize_workflow(best_xgb)

final_xgb_fit <- final_xgb_wf |>
  last_fit(icu_split)

final_xgb_fit |>
  collect_metrics()
```

```
# A tibble: 3 × 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.599 Preprocessor1_Model1
2 roc_auc     binary         0.638 Preprocessor1_Model1
3 brier_class binary         0.235 Preprocessor1_Model1
```

```r
predictions_xgb <- collect_predictions(final_xgb_fit)
conf_mat(predictions_xgb, truth = los_long, estimate = .pred_class)
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 13072  8859
     TRUE   6265  9524
```

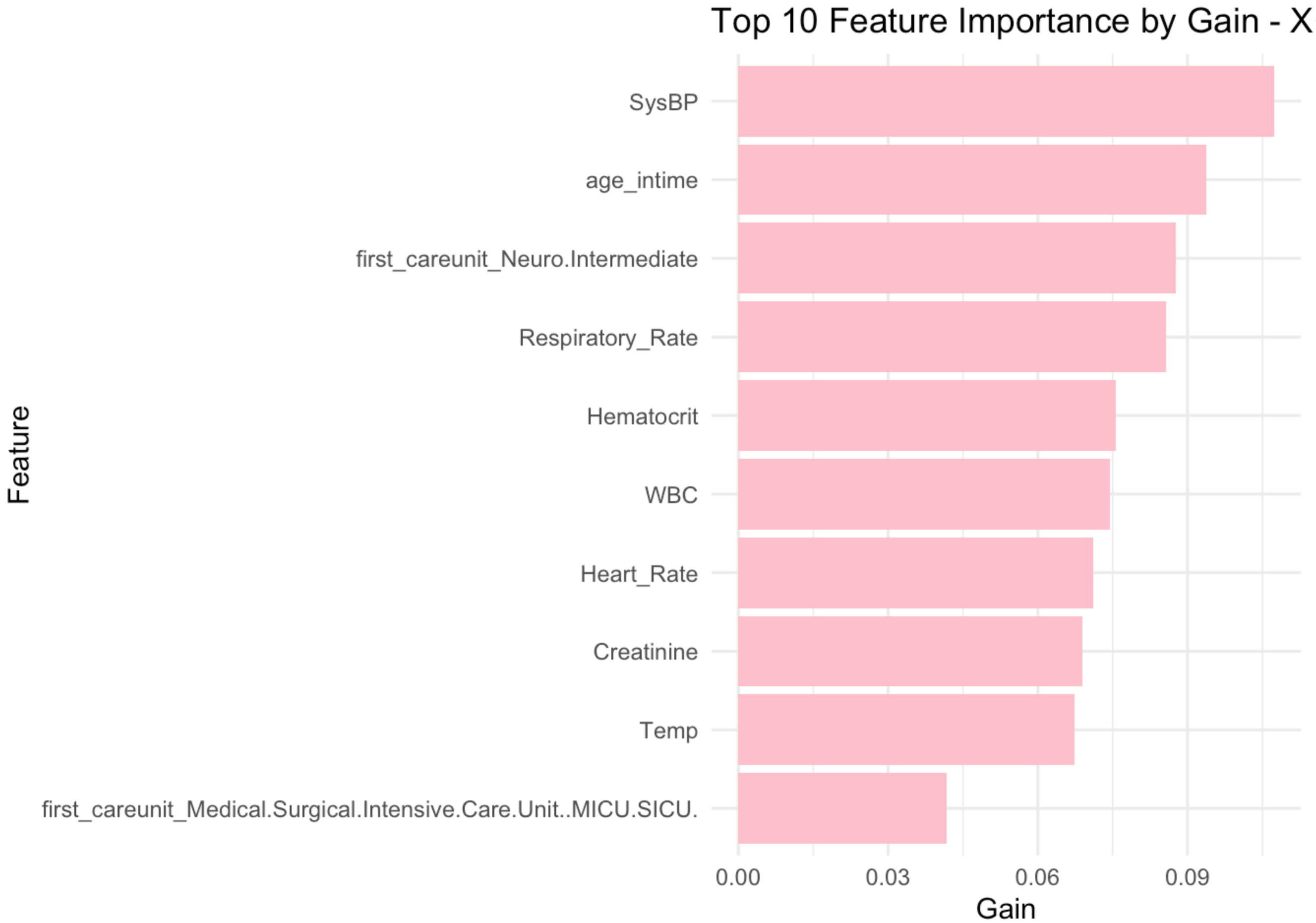
```r
final_modelxgb <- extract_fit_parsnip(final_xgb_fit$.workflow[[1]])

importance_df_xgb <- xgb.importance(model = final_modelxgb$fit) %>%
  as_tibble() %>%
  arrange(desc(Gain))

# Show all features and their importance (by Gain)
print(importance_df_xgb, n = Inf)
```

```
# A tibble: 38 × 4
   Feature                                                 Gain   Cover Frequency
   <chr>                                                  <dbl>   <dbl>     <dbl>
 1 SysBP                                                1.07e-1 5.91e-2   0.0749
 2 age_intime                                           9.37e-2 6.51e-2   0.0761
 3 first_careunit_Neuro.Intermediate                    8.76e-2 5.35e-2   0.0322
 4 Respiratory_Rate                                     8.56e-2 5.38e-2   0.0608
 5 Hematocrit                                           7.55e-2 8.13e-2   0.0935
 6 WBC                                                  7.45e-2 4.96e-2   0.0699
 7 Heart_Rate                                           7.11e-2 7.14e-2   0.0782
 8 Creatinine                                           6.89e-2 9.42e-2   0.0676
 9 Temp                                                 6.74e-2 7.98e-2   0.0664
```

```
10 first_careunit_Medical.Surgical.Intensive.Care.Uni… 4.17e-2 5.61e-2  0.0342
11 Sodium                                               3.92e-2 7.58e-2  0.0602
12 first_careunit_Medical.Intensive.Care.Unit..MICU.   3.82e-2 4.51e-2  0.0289
13 Bicarbonate                                          3.40e-2 5.30e-2  0.0531
14 DiaBP                                                2.93e-2 2.85e-2  0.0555
15 Chloride                                             2.18e-2 2.88e-2  0.0245
16 Potassium                                            1.83e-2 9.40e-3  0.0389
17 first_careunit_Surgery.Vascular.Intermediate         1.02e-2 3.45e-2  0.0153
18 first_careunit_Trauma.SICU..TSICU.                   1.01e-2 2.33e-2  0.0130
19 gender_M                                             4.30e-3 7.95e-4  0.00856
20 first_careunit_Neuro.Stepdown                        4.08e-3 7.11e-3  0.00531
21 race_UNKNOWN                                         3.71e-3 3.08e-3  0.00561
22 race_BLACK.AFRICAN.AMERICAN                          2.68e-3 3.29e-4  0.00561
23 first_careunit_Neuro.Surgical.Intensive.Care.Unit.… 2.48e-3 4.09e-3  0.00413
24 marital_status_SINGLE                                1.39e-3 6.08e-5  0.00413
25 race_HISPANIC.LATINO...DOMINICAN                     1.28e-3 9.03e-3  0.00413
26 marital_status_MARRIED                               1.06e-3 2.92e-4  0.00266
27 race_HISPANIC.OR.LATINO                              7.13e-4 5.88e-3  0.00266
28 first_careunit_Surgical.Intensive.Care.Unit..SICU.  6.85e-4 2.02e-4  0.00177
29 marital_status_WIDOWED                               6.73e-4 2.55e-4  0.00325
30 race_HISPANIC.LATINO...CENTRAL.AMERICAN              6.25e-4 2.96e-3  0.00177
31 race_PATIENT.DECLINED.TO.ANSWER                      3.29e-4 1.15e-4  0.000885
32 race_BLACK.CARIBBEAN.ISLAND                          3.25e-4 2.39e-5  0.00118
33 race_BLACK.AFRICAN                                   3.11e-4 1.77e-4  0.000590
34 race_ASIAN...SOUTH.EAST.ASIAN                        3.00e-4 4.66e-4  0.00118
35 race_ASIAN...CHINESE                                 2.12e-4 1.92e-3  0.000885
36 race_WHITE                                           1.54e-4 3.50e-6  0.00177
37 first_careunit_Coronary.Care.Unit..CCU.              1.53e-4 3.88e-4  0.000295
38 race_BLACK.CAPE.VERDEAN                              8.62e-5 6.17e-4  0.000295
```

Let us plot it in GGPlot for better visualization

```
importance_df_xgb %>%
  top_n(10, wt = Gain) %>%
  ggplot(aes(x = reorder(Feature, Gain), y = Gain)) +
  geom_col(fill = "pink") +
  coord_flip() +
  labs(
    title = "Top 10 Feature Importance by Gain - XGBoost",
    x = "Feature",
    y = "Gain"
  ) +
  theme_minimal()
```



The accuracy and ROC_AUC for the XGBoost model is 0.599 and 0.638, respectively. This indicates that the random_forest was better able to predict los_long >= 2 days, but was better than predicting this than the logit regression.

This outcome is quiet interesting, as I expected the XGBoost model to perform better than the random forest model. However, considering that learn_rate is on the lower end, it is not surprising; this means that the XGBoost needed more values to be able to learn the model correctly. The low value that I gave it was not enough, but necessary (and allowed by Dr. Zhou) due to the slow nature of the processing (>= 2ish hours). Given a stronger computer, I would have made this value higher to see if this was the same or better than the random forest model

Looking into the gain, cover, and frequency meanings. it looks like SysBP had the highest importance to the model, followed by age_intime and then respiratory rate. However, Hematocrit showed up the most (frequency) with 9.18%, indicating it was used in about 9.18% of all splits there were done. Hematocrit also had the highest cover, which indicates that it impacted the observations the most.

Interestingly, SysBP, age_intime, Respiratory_Rate, first_careunit being the intermediate Neuro one, and Hematocrit, were the most important features in determining whether or not the los_long was greater than or equal to two days. This is interesting as the XGBoost model was able to pick up on some of same features as the random forest model (SysBP, Hematocrit, and Age_intime), but in a different order. This could be due to the fact that the XGBoost model is more sensitive to the features and is able to pick up on the nuances of

the data better than the random forest model. However as I stated before, this could also be because of the varying learn_rate, so there is error as well.

Comparing XGBoost with Random Forest, both had the SysBP as the highest feature for prediction. Otherwise, they also both had age_intime and Hematocrit in their top five for feature importance. Comparing XGBoost to the Logistic Regression Model, however, the similarities included: neuro.intermediate as the first careunit, age_intime, and respiratory_rate, and technically SysBP as well. In essence, the common features out of all three of the features was age_intime and SysBP.

Model Stacking Log. Regression, Random Forest, and XGBoost

Data Preprocessing and engineering

```
set.seed(203)

# Stratified split by los_long
icu_split <- initial_split(
  icu_data,
  strata = los_long,
  prop = 0.5
)

icu_train <- training(icu_split)
icu_test <- testing(icu_split)

head(icu_train)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000980 26913865 39765666 FALSE    Medical Intensive Car… F              76
2   10002155 20345487 32358465 FALSE    Medical Intensive Car… F              83
3   10003019 22774359 30676350 FALSE    Medical/Surgical Inte… M              73
4   10003502 29011269 35796366 FALSE    Coronary Care Unit (C… F              94
5   10004457 23251352 31494479 FALSE    Cardiac Vascular Inte… M              66
6   10005348 25239799 34629895 FALSE    Cardiac Vascular Inte… M              78
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

```
head(icu_test)
```

```
# A tibble: 6 × 21
  subject_id  hadm_id  stay_id los_long first_careunit         gender age_intime
       <int>    <int>    <int> <fct>    <chr>                  <chr>       <int>
1   10000032 29079034 39553978 FALSE    Medical Intensive Car… F              52
2   10001217 24597018 37067082 FALSE    Surgical Intensive Ca… F              55
```

```
3   10001217 27703517 34592300 FALSE    Surgical Intensive Ca… F              55
4   10001843 26133978 39698942 FALSE    Medical/Surgical Inte… M              76
5   10001884 26184834 37510196 TRUE     Medical Intensive Car… F              77
6   10002013 23581541 39060235 FALSE    Cardiac Vascular Inte… F              57
# ℹ 14 more variables: marital_status <chr>, race <chr>, Heart_Rate <dbl>,
#   DiaBP <dbl>, SysBP <dbl>, Respiratory_Rate <dbl>, Temp <dbl>,
#   Creatinine <dbl>, Potassium <dbl>, Chloride <dbl>, Bicarbonate <dbl>,
#   Hematocrit <dbl>, WBC <dbl>, Sodium <dbl>
```

```r
##Now, let us make the logit_recipe for the logistic regression model

modelstacking <- recipe(
  los_long ~ first_careunit + gender + age_intime + marital_status + race +
    Heart_Rate + DiaBP + SysBP + Respiratory_Rate + Temp +
    Creatinine + Potassium + Chloride + Bicarbonate + Hematocrit +
    WBC + Sodium,
  data = icu_train
) %>%
  step_impute_median(all_numeric_predictors()) %>%
  step_impute_mode(all_nominal_predictors()) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_numeric_predictors())
```

```r
set.seed(203)
foldsSTACK <- vfold_cv(icu_train, v = 3)
```

Final Model Stacking:

```r
#The penalty was decided in office hours with Dr. Zhou to speed up the loading process of

icu_stack <-
  stacks() %>%
  add_candidates(logit_tune) %>%
  add_candidates(rf_tune) %>%
  add_candidates(xgb_tune) %>%
  blend_predictions(
    penalty = 10^(-6:2),
    metrics = c("roc_auc", "accuracy")
  ) |>
  fit_members()
```

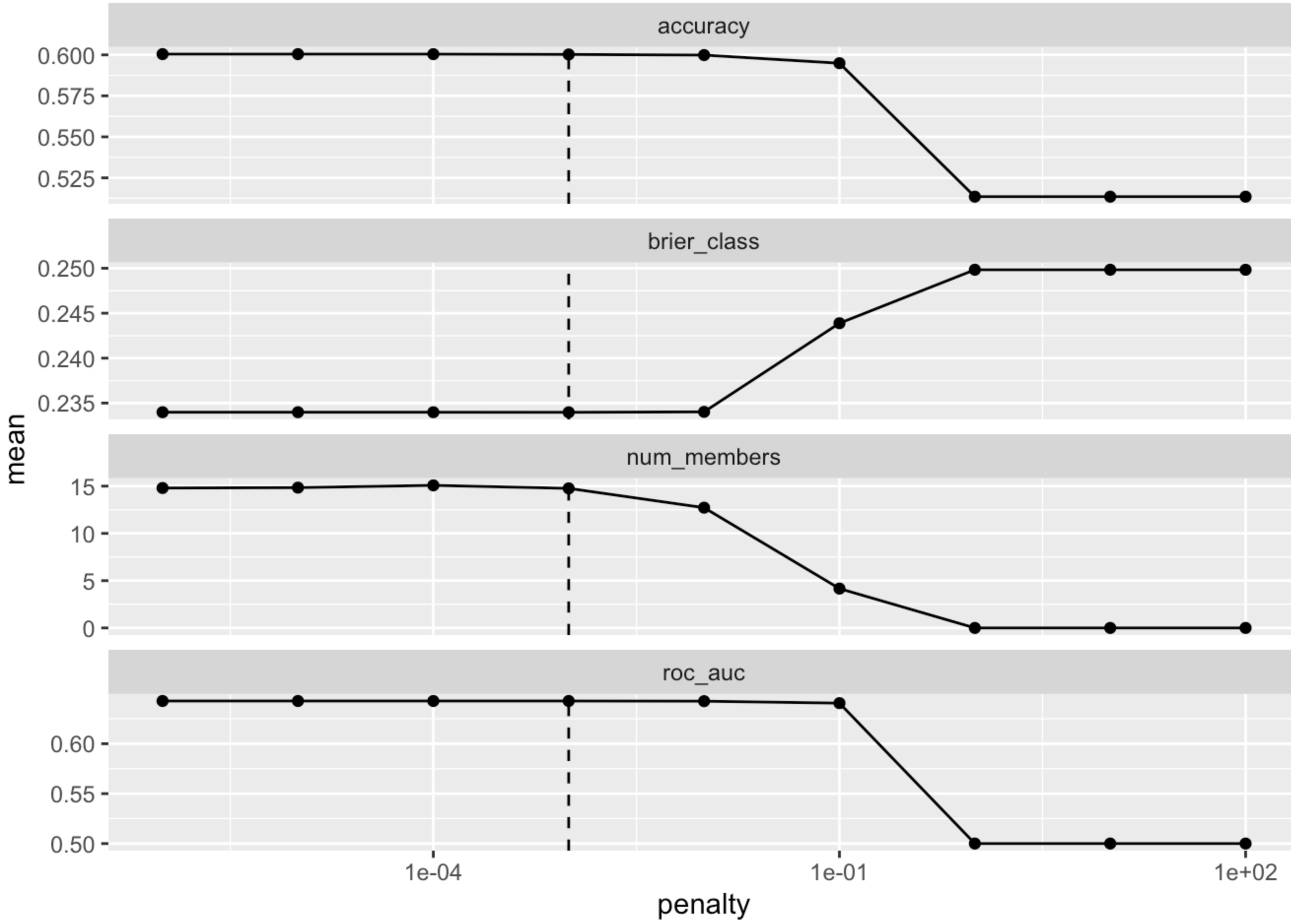
```
Warning: Predictions from 634 candidates were identical to those from existing
candidates and were removed from the data stack.

Warning: Predictions from 12 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'
```

```
icu_stack
```

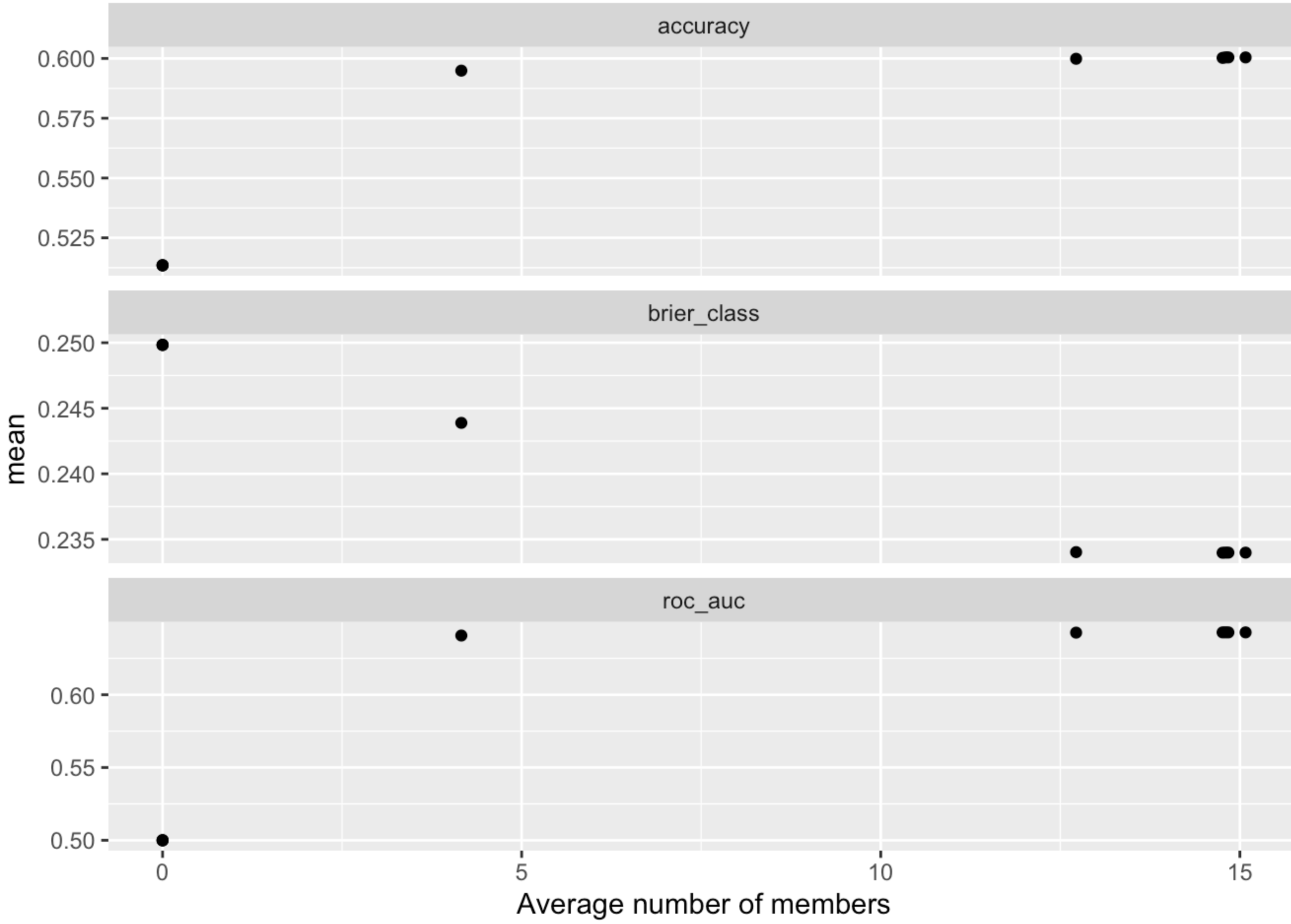
```
── A stacked ensemble model ─────────────────────────────────────


Out of 247 possible candidate members, the ensemble retained 15.

Penalty: 0.001.

Mixture: 1.


The 10 highest weighted member classes are:

# A tibble: 10 × 3
   member                   type        weight
   <chr>                    <chr>        <dbl>
 1 .pred_TRUE_rf_tune_1_24  rand_forest  1.19
 2 .pred_TRUE_xgb_tune_1_39 boost_tree   1.07
 3 .pred_TRUE_rf_tune_1_23  rand_forest  0.653
 4 .pred_TRUE_rf_tune_1_13  rand_forest  0.449
 5 .pred_TRUE_rf_tune_1_25  rand_forest  0.437
 6 .pred_TRUE_rf_tune_1_21  rand_forest  0.413
 7 .pred_TRUE_rf_tune_1_18  rand_forest  0.367
 8 .pred_TRUE_rf_tune_1_19  rand_forest  0.296
 9 .pred_TRUE_xgb_tune_1_10 boost_tree   0.270
10 .pred_TRUE_rf_tune_1_16  rand_forest  0.181
```

```
autoplot(icu_stack)
```



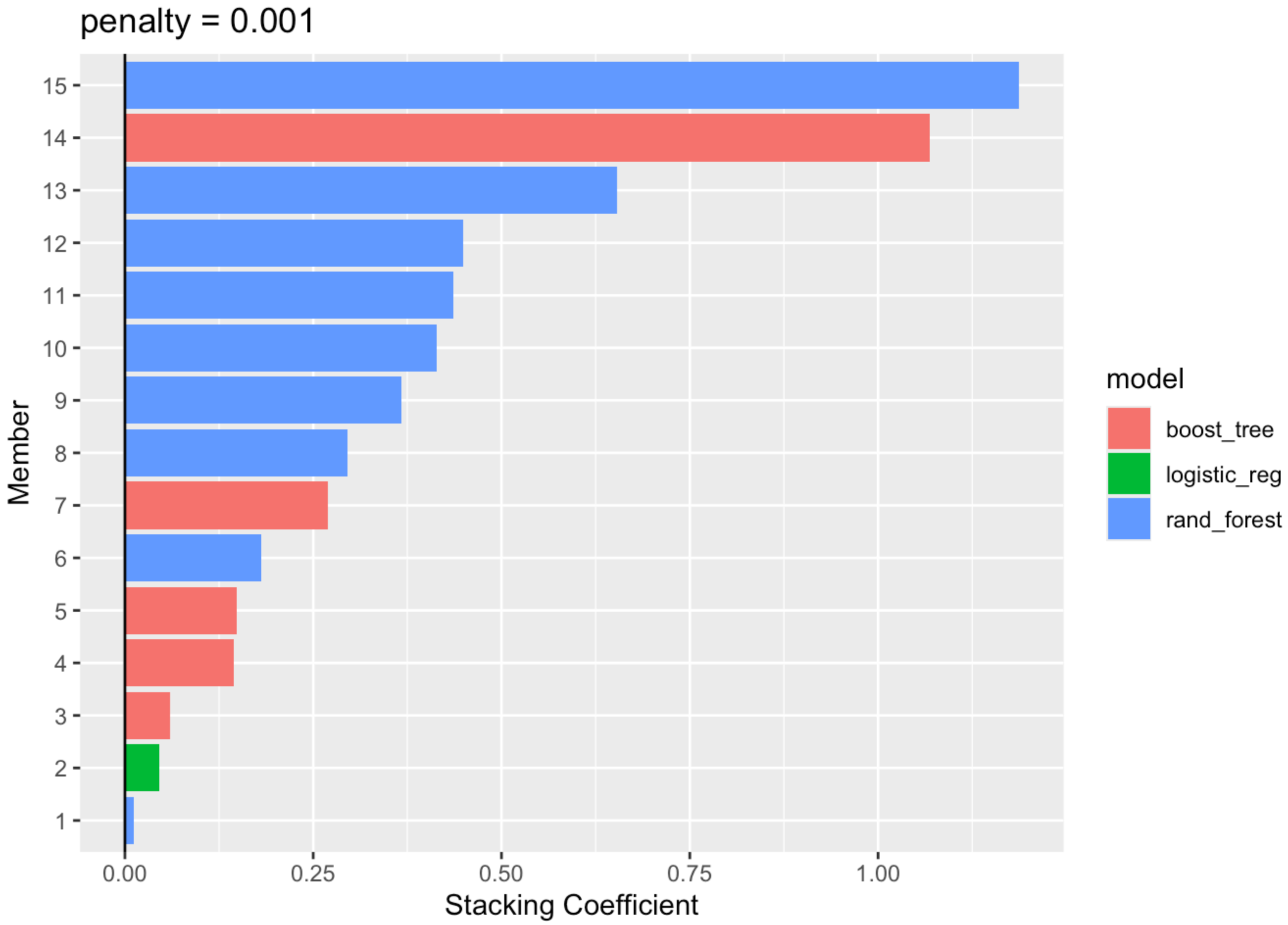
Looking at this picture, the accuracy of 0.6 occurred at the lowest penalty of where the dotted line is at; as the penalty increased, the accuracy decreased as well. The brier_class increased while the penalty increased as well, which is not what we want since a lower brier_class is better. The ROC_AUC was the highest at the location of the dotted line (1e-03 probably) and then was moderately the same until about 1e-01, but then decreased to 0.50 as the penalties increased. With all of this in mind, it is suggested that in the future that a penalty of 1e-03 is used for better results and performance

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



this makes sense, higher number of members would create higher accuracy, which is seen, better calibration, which is seen by the brier_class, and higher ROC AUC. In essence, higher average number of members created better performance of the model.

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



This graph illustrates that random forest had the highest weight in the model and that the stacked model had a better time utilizing random forest compared to XGBoost or Log. Reg.. Surprisingly, Log. Reg is rarely on there at all. However, considering that XGBoost and RF had better accuracy and ROC_AUC compared to Log. Reg., this is believable.

```
collect_parameters(icu_stack, "rf_tune") |>
  arrange(desc(coef)) |>
  print(n = Inf)
```

```
# A tibble: 25 × 5
   member        mtry trees terms                      coef
   <chr>        <int> <int> <chr>                     <dbl>
 1 rf_tune_1_24     5   400 .pred_TRUE_rf_tune_1_24 1.19
 2 rf_tune_1_23     5   300 .pred_TRUE_rf_tune_1_23 0.653
 3 rf_tune_1_13     3   300 .pred_TRUE_rf_tune_1_13 0.449
 4 rf_tune_1_25     5   500 .pred_TRUE_rf_tune_1_25 0.437
 5 rf_tune_1_21     5   100 .pred_TRUE_rf_tune_1_21 0.413
 6 rf_tune_1_18     4   300 .pred_TRUE_rf_tune_1_18 0.367
 7 rf_tune_1_19     4   400 .pred_TRUE_rf_tune_1_19 0.296
 8 rf_tune_1_16     4   100 .pred_TRUE_rf_tune_1_16 0.181
 9 rf_tune_1_17     4   200 .pred_TRUE_rf_tune_1_17 0.0115
10 rf_tune_1_01     1   100 .pred_TRUE_rf_tune_1_01 0
11 rf_tune_1_02     1   200 .pred_TRUE_rf_tune_1_02 0
```

```
12 rf_tune_1_03     1   300 .pred_TRUE_rf_tune_1_03 0
13 rf_tune_1_04     1   400 .pred_TRUE_rf_tune_1_04 0
14 rf_tune_1_05     1   500 .pred_TRUE_rf_tune_1_05 0
15 rf_tune_1_06     2   100 .pred_TRUE_rf_tune_1_06 0
16 rf_tune_1_07     2   200 .pred_TRUE_rf_tune_1_07 0
17 rf_tune_1_08     2   300 .pred_TRUE_rf_tune_1_08 0
18 rf_tune_1_09     2   400 .pred_TRUE_rf_tune_1_09 0
19 rf_tune_1_10     2   500 .pred_TRUE_rf_tune_1_10 0
20 rf_tune_1_11     3   100 .pred_TRUE_rf_tune_1_11 0
21 rf_tune_1_12     3   200 .pred_TRUE_rf_tune_1_12 0
22 rf_tune_1_14     3   400 .pred_TRUE_rf_tune_1_14 0
23 rf_tune_1_15     3   500 .pred_TRUE_rf_tune_1_15 0
24 rf_tune_1_20     4   500 .pred_TRUE_rf_tune_1_20 0
25 rf_tune_1_22     5   200 .pred_TRUE_rf_tune_1_22 0
```

The most important memebrs seemed to have come from the random_forest model. Specifically, the first three had the highest coef. indicating a higher weight/influence on the final stacked model prdiction

```
collect_parameters(icu_stack, "logit_tune") |>
  arrange(desc(coef)) |>
  print(n = Inf)
```

```
# A tibble: 183 × 5
    member              penalty mixture terms                          coef
    <chr>                 <dbl>   <dbl> <chr>                         <dbl>
  1 logit_tune_1_101  0.000001     0.25 .pred_TRUE_logit_tune_1_101 0.0453
  2 logit_tune_1_001  0.000001     0    .pred_TRUE_logit_tune_1_001 0
  3 logit_tune_1_041  0.00433      0    .pred_TRUE_logit_tune_1_041 0
  4 logit_tune_1_042  0.00534      0    .pred_TRUE_logit_tune_1_042 0
  5 logit_tune_1_043  0.00658      0    .pred_TRUE_logit_tune_1_043 0
  6 logit_tune_1_044  0.00811      0    .pred_TRUE_logit_tune_1_044 0
  7 logit_tune_1_045  0.01         0    .pred_TRUE_logit_tune_1_045 0
  8 logit_tune_1_046  0.0123       0    .pred_TRUE_logit_tune_1_046 0
  9 logit_tune_1_047  0.0152       0    .pred_TRUE_logit_tune_1_047 0
 10 logit_tune_1_048  0.0187       0    .pred_TRUE_logit_tune_1_048 0
 11 logit_tune_1_049  0.0231       0    .pred_TRUE_logit_tune_1_049 0
 12 logit_tune_1_050  0.0285       0    .pred_TRUE_logit_tune_1_050 0
 13 logit_tune_1_051  0.0351       0    .pred_TRUE_logit_tune_1_051 0
 14 logit_tune_1_052  0.0433       0    .pred_TRUE_logit_tune_1_052 0
 15 logit_tune_1_053  0.0534       0    .pred_TRUE_logit_tune_1_053 0
 16 logit_tune_1_054  0.0658       0    .pred_TRUE_logit_tune_1_054 0
 17 logit_tune_1_055  0.0811       0    .pred_TRUE_logit_tune_1_055 0
 18 logit_tune_1_056  0.1          0    .pred_TRUE_logit_tune_1_056 0
 19 logit_tune_1_057  0.123        0    .pred_TRUE_logit_tune_1_057 0
 20 logit_tune_1_058  0.152        0    .pred_TRUE_logit_tune_1_058 0
 21 logit_tune_1_059  0.187        0    .pred_TRUE_logit_tune_1_059 0
 22 logit_tune_1_060  0.231        0    .pred_TRUE_logit_tune_1_060 0
 23 logit_tune_1_061  0.285        0    .pred_TRUE_logit_tune_1_061 0
 24 logit_tune_1_062  0.351        0    .pred_TRUE_logit_tune_1_062 0
 25 logit_tune_1_063  0.433        0    .pred_TRUE_logit_tune_1_063 0
 26 logit_tune_1_064  0.534        0    .pred_TRUE_logit_tune_1_064 0
```

```
27 logit_tune_1_065  0.658        0    .pred_TRUE_logit_tune_1_065 0
28 logit_tune_1_066  0.811        0    .pred_TRUE_logit_tune_1_066 0
29 logit_tune_1_067  1            0    .pred_TRUE_logit_tune_1_067 0
30 logit_tune_1_068  1.23         0    .pred_TRUE_logit_tune_1_068 0
31 logit_tune_1_069  1.52         0    .pred_TRUE_logit_tune_1_069 0
32 logit_tune_1_070  1.87         0    .pred_TRUE_logit_tune_1_070 0
33 logit_tune_1_071  2.31         0    .pred_TRUE_logit_tune_1_071 0
34 logit_tune_1_072  2.85         0    .pred_TRUE_logit_tune_1_072 0
35 logit_tune_1_073  3.51         0    .pred_TRUE_logit_tune_1_073 0
36 logit_tune_1_074  4.33         0    .pred_TRUE_logit_tune_1_074 0
37 logit_tune_1_075  5.34         0    .pred_TRUE_logit_tune_1_075 0
38 logit_tune_1_076  6.58         0    .pred_TRUE_logit_tune_1_076 0
39 logit_tune_1_077  8.11         0    .pred_TRUE_logit_tune_1_077 0
40 logit_tune_1_078 10            0    .pred_TRUE_logit_tune_1_078 0
41 logit_tune_1_079 12.3          0    .pred_TRUE_logit_tune_1_079 0
42 logit_tune_1_080 15.2          0    .pred_TRUE_logit_tune_1_080 0
43 logit_tune_1_081 18.7          0    .pred_TRUE_logit_tune_1_081 0
44 logit_tune_1_082 23.1          0    .pred_TRUE_logit_tune_1_082 0
45 logit_tune_1_083 28.5          0    .pred_TRUE_logit_tune_1_083 0
46 logit_tune_1_084 35.1          0    .pred_TRUE_logit_tune_1_084 0
47 logit_tune_1_120  0.0000534    0.25 .pred_TRUE_logit_tune_1_120 0
48 logit_tune_1_121  0.0000658    0.25 .pred_TRUE_logit_tune_1_121 0
49 logit_tune_1_122  0.0000811    0.25 .pred_TRUE_logit_tune_1_122 0
50 logit_tune_1_123  0.0001       0.25 .pred_TRUE_logit_tune_1_123 0
51 logit_tune_1_124  0.000123     0.25 .pred_TRUE_logit_tune_1_124 0
52 logit_tune_1_125  0.000152     0.25 .pred_TRUE_logit_tune_1_125 0
53 logit_tune_1_126  0.000187     0.25 .pred_TRUE_logit_tune_1_126 0
54 logit_tune_1_127  0.000231     0.25 .pred_TRUE_logit_tune_1_127 0
55 logit_tune_1_128  0.000285     0.25 .pred_TRUE_logit_tune_1_128 0
56 logit_tune_1_129  0.000351     0.25 .pred_TRUE_logit_tune_1_129 0
57 logit_tune_1_130  0.000433     0.25 .pred_TRUE_logit_tune_1_130 0
58 logit_tune_1_131  0.000534     0.25 .pred_TRUE_logit_tune_1_131 0
59 logit_tune_1_132  0.000658     0.25 .pred_TRUE_logit_tune_1_132 0
60 logit_tune_1_133  0.000811     0.25 .pred_TRUE_logit_tune_1_133 0
61 logit_tune_1_134  0.001        0.25 .pred_TRUE_logit_tune_1_134 0
62 logit_tune_1_135  0.00123      0.25 .pred_TRUE_logit_tune_1_135 0
63 logit_tune_1_136  0.00152      0.25 .pred_TRUE_logit_tune_1_136 0
64 logit_tune_1_137  0.00187      0.25 .pred_TRUE_logit_tune_1_137 0
65 logit_tune_1_138  0.00231      0.25 .pred_TRUE_logit_tune_1_138 0
66 logit_tune_1_139  0.00285      0.25 .pred_TRUE_logit_tune_1_139 0
67 logit_tune_1_140  0.00351      0.25 .pred_TRUE_logit_tune_1_140 0
68 logit_tune_1_141  0.00433      0.25 .pred_TRUE_logit_tune_1_141 0
69 logit_tune_1_142  0.00534      0.25 .pred_TRUE_logit_tune_1_142 0
70 logit_tune_1_143  0.00658      0.25 .pred_TRUE_logit_tune_1_143 0
71 logit_tune_1_144  0.00811      0.25 .pred_TRUE_logit_tune_1_144 0
72 logit_tune_1_145  0.01         0.25 .pred_TRUE_logit_tune_1_145 0
73 logit_tune_1_146  0.0123       0.25 .pred_TRUE_logit_tune_1_146 0
74 logit_tune_1_147  0.0152       0.25 .pred_TRUE_logit_tune_1_147 0
75 logit_tune_1_148  0.0187       0.25 .pred_TRUE_logit_tune_1_148 0
76 logit_tune_1_149  0.0231       0.25 .pred_TRUE_logit_tune_1_149 0
77 logit_tune_1_150  0.0285       0.25 .pred_TRUE_logit_tune_1_150 0
```

```
 78 logit_tune_1_151  0.0351       0.25 .pred_TRUE_logit_tune_1_151 0
 79 logit_tune_1_152  0.0433       0.25 .pred_TRUE_logit_tune_1_152 0
 80 logit_tune_1_153  0.0534       0.25 .pred_TRUE_logit_tune_1_153 0
 81 logit_tune_1_154  0.0658       0.25 .pred_TRUE_logit_tune_1_154 0
 82 logit_tune_1_155  0.0811       0.25 .pred_TRUE_logit_tune_1_155 0
 83 logit_tune_1_156  0.1          0.25 .pred_TRUE_logit_tune_1_156 0
 84 logit_tune_1_157  0.123        0.25 .pred_TRUE_logit_tune_1_157 0
 85 logit_tune_1_158  0.152        0.25 .pred_TRUE_logit_tune_1_158 0
 86 logit_tune_1_201  0.000001     0.5  .pred_TRUE_logit_tune_1_201 0
 87 logit_tune_1_221  0.0000658    0.5  .pred_TRUE_logit_tune_1_221 0
 88 logit_tune_1_222  0.0000811    0.5  .pred_TRUE_logit_tune_1_222 0
 89 logit_tune_1_223  0.0001       0.5  .pred_TRUE_logit_tune_1_223 0
 90 logit_tune_1_224  0.000123     0.5  .pred_TRUE_logit_tune_1_224 0
 91 logit_tune_1_225  0.000152     0.5  .pred_TRUE_logit_tune_1_225 0
 92 logit_tune_1_226  0.000187     0.5  .pred_TRUE_logit_tune_1_226 0
 93 logit_tune_1_227  0.000231     0.5  .pred_TRUE_logit_tune_1_227 0
 94 logit_tune_1_228  0.000285     0.5  .pred_TRUE_logit_tune_1_228 0
 95 logit_tune_1_229  0.000351     0.5  .pred_TRUE_logit_tune_1_229 0
 96 logit_tune_1_230  0.000433     0.5  .pred_TRUE_logit_tune_1_230 0
 97 logit_tune_1_231  0.000534     0.5  .pred_TRUE_logit_tune_1_231 0
 98 logit_tune_1_232  0.000658     0.5  .pred_TRUE_logit_tune_1_232 0
 99 logit_tune_1_233  0.000811     0.5  .pred_TRUE_logit_tune_1_233 0
100 logit_tune_1_234  0.001        0.5  .pred_TRUE_logit_tune_1_234 0
101 logit_tune_1_235  0.00123      0.5  .pred_TRUE_logit_tune_1_235 0
102 logit_tune_1_236  0.00152      0.5  .pred_TRUE_logit_tune_1_236 0
103 logit_tune_1_237  0.00187      0.5  .pred_TRUE_logit_tune_1_237 0
104 logit_tune_1_238  0.00231      0.5  .pred_TRUE_logit_tune_1_238 0
105 logit_tune_1_239  0.00285      0.5  .pred_TRUE_logit_tune_1_239 0
106 logit_tune_1_240  0.00351      0.5  .pred_TRUE_logit_tune_1_240 0
107 logit_tune_1_241  0.00433      0.5  .pred_TRUE_logit_tune_1_241 0
108 logit_tune_1_242  0.00534      0.5  .pred_TRUE_logit_tune_1_242 0
109 logit_tune_1_243  0.00658      0.5  .pred_TRUE_logit_tune_1_243 0
110 logit_tune_1_244  0.00811      0.5  .pred_TRUE_logit_tune_1_244 0
111 logit_tune_1_245  0.01         0.5  .pred_TRUE_logit_tune_1_245 0
112 logit_tune_1_246  0.0123       0.5  .pred_TRUE_logit_tune_1_246 0
113 logit_tune_1_247  0.0152       0.5  .pred_TRUE_logit_tune_1_247 0
114 logit_tune_1_248  0.0187       0.5  .pred_TRUE_logit_tune_1_248 0
115 logit_tune_1_249  0.0231       0.5  .pred_TRUE_logit_tune_1_249 0
116 logit_tune_1_250  0.0285       0.5  .pred_TRUE_logit_tune_1_250 0
117 logit_tune_1_251  0.0351       0.5  .pred_TRUE_logit_tune_1_251 0
118 logit_tune_1_252  0.0433       0.5  .pred_TRUE_logit_tune_1_252 0
119 logit_tune_1_253  0.0534       0.5  .pred_TRUE_logit_tune_1_253 0
120 logit_tune_1_254  0.0658       0.5  .pred_TRUE_logit_tune_1_254 0
121 logit_tune_1_301  0.000001     0.75 .pred_TRUE_logit_tune_1_301 0
122 logit_tune_1_322  0.0000811    0.75 .pred_TRUE_logit_tune_1_322 0
123 logit_tune_1_323  0.0001       0.75 .pred_TRUE_logit_tune_1_323 0
124 logit_tune_1_324  0.000123     0.75 .pred_TRUE_logit_tune_1_324 0
125 logit_tune_1_325  0.000152     0.75 .pred_TRUE_logit_tune_1_325 0
126 logit_tune_1_326  0.000187     0.75 .pred_TRUE_logit_tune_1_326 0
127 logit_tune_1_327  0.000231     0.75 .pred_TRUE_logit_tune_1_327 0
128 logit_tune_1_328  0.000285     0.75 .pred_TRUE_logit_tune_1_328 0
```

```
129 logit_tune_1_329  0.000351     0.75 .pred_TRUE_logit_tune_1_329 0
130 logit_tune_1_330  0.000433     0.75 .pred_TRUE_logit_tune_1_330 0
131 logit_tune_1_331  0.000534     0.75 .pred_TRUE_logit_tune_1_331 0
132 logit_tune_1_332  0.000658     0.75 .pred_TRUE_logit_tune_1_332 0
133 logit_tune_1_333  0.000811     0.75 .pred_TRUE_logit_tune_1_333 0
134 logit_tune_1_334  0.001        0.75 .pred_TRUE_logit_tune_1_334 0
135 logit_tune_1_335  0.00123      0.75 .pred_TRUE_logit_tune_1_335 0
136 logit_tune_1_336  0.00152      0.75 .pred_TRUE_logit_tune_1_336 0
137 logit_tune_1_337  0.00187      0.75 .pred_TRUE_logit_tune_1_337 0
138 logit_tune_1_338  0.00231      0.75 .pred_TRUE_logit_tune_1_338 0
139 logit_tune_1_339  0.00285      0.75 .pred_TRUE_logit_tune_1_339 0
140 logit_tune_1_340  0.00351      0.75 .pred_TRUE_logit_tune_1_340 0
141 logit_tune_1_341  0.00433      0.75 .pred_TRUE_logit_tune_1_341 0
142 logit_tune_1_342  0.00534      0.75 .pred_TRUE_logit_tune_1_342 0
143 logit_tune_1_343  0.00658      0.75 .pred_TRUE_logit_tune_1_343 0
144 logit_tune_1_344  0.00811      0.75 .pred_TRUE_logit_tune_1_344 0
145 logit_tune_1_345  0.01         0.75 .pred_TRUE_logit_tune_1_345 0
146 logit_tune_1_346  0.0123       0.75 .pred_TRUE_logit_tune_1_346 0
147 logit_tune_1_347  0.0152       0.75 .pred_TRUE_logit_tune_1_347 0
148 logit_tune_1_348  0.0187       0.75 .pred_TRUE_logit_tune_1_348 0
149 logit_tune_1_349  0.0231       0.75 .pred_TRUE_logit_tune_1_349 0
150 logit_tune_1_350  0.0285       0.75 .pred_TRUE_logit_tune_1_350 0
151 logit_tune_1_351  0.0351       0.75 .pred_TRUE_logit_tune_1_351 0
152 logit_tune_1_352  0.0433       0.75 .pred_TRUE_logit_tune_1_352 0
153 logit_tune_1_401  0.000001     1    .pred_TRUE_logit_tune_1_401 0
154 logit_tune_1_423  0.0001       1    .pred_TRUE_logit_tune_1_423 0
155 logit_tune_1_424  0.000123     1    .pred_TRUE_logit_tune_1_424 0
156 logit_tune_1_425  0.000152     1    .pred_TRUE_logit_tune_1_425 0
157 logit_tune_1_426  0.000187     1    .pred_TRUE_logit_tune_1_426 0
158 logit_tune_1_427  0.000231     1    .pred_TRUE_logit_tune_1_427 0
159 logit_tune_1_428  0.000285     1    .pred_TRUE_logit_tune_1_428 0
160 logit_tune_1_429  0.000351     1    .pred_TRUE_logit_tune_1_429 0
161 logit_tune_1_430  0.000433     1    .pred_TRUE_logit_tune_1_430 0
162 logit_tune_1_431  0.000534     1    .pred_TRUE_logit_tune_1_431 0
163 logit_tune_1_432  0.000658     1    .pred_TRUE_logit_tune_1_432 0
164 logit_tune_1_433  0.000811     1    .pred_TRUE_logit_tune_1_433 0
165 logit_tune_1_434  0.001        1    .pred_TRUE_logit_tune_1_434 0
166 logit_tune_1_435  0.00123      1    .pred_TRUE_logit_tune_1_435 0
167 logit_tune_1_436  0.00152      1    .pred_TRUE_logit_tune_1_436 0
168 logit_tune_1_437  0.00187      1    .pred_TRUE_logit_tune_1_437 0
169 logit_tune_1_438  0.00231      1    .pred_TRUE_logit_tune_1_438 0
170 logit_tune_1_439  0.00285      1    .pred_TRUE_logit_tune_1_439 0
171 logit_tune_1_440  0.00351      1    .pred_TRUE_logit_tune_1_440 0
172 logit_tune_1_441  0.00433      1    .pred_TRUE_logit_tune_1_441 0
173 logit_tune_1_442  0.00534      1    .pred_TRUE_logit_tune_1_442 0
174 logit_tune_1_443  0.00658      1    .pred_TRUE_logit_tune_1_443 0
175 logit_tune_1_444  0.00811      1    .pred_TRUE_logit_tune_1_444 0
176 logit_tune_1_445  0.01         1    .pred_TRUE_logit_tune_1_445 0
177 logit_tune_1_446  0.0123       1    .pred_TRUE_logit_tune_1_446 0
178 logit_tune_1_447  0.0152       1    .pred_TRUE_logit_tune_1_447 0
179 logit_tune_1_448  0.0187       1    .pred_TRUE_logit_tune_1_448 0
```

```
180 logit_tune_1_449  0.0231       1    .pred_TRUE_logit_tune_1_449 0
181 logit_tune_1_450  0.0285       1    .pred_TRUE_logit_tune_1_450 0
182 logit_tune_1_451  0.0351       1    .pred_TRUE_logit_tune_1_451 0
183 logit_tune_1_452  0.0433       1    .pred_TRUE_logit_tune_1_452 0
```

**This shows that the logit_tune contributed once to the model, with a coef of 0.045.

```r
collect_parameters(icu_stack, "xgb_tune") |>
  arrange(desc(coef)) |>
  print(n = Inf)
```

```
# A tibble: 39 × 6
   member        trees tree_depth learn_rate terms                        coef
   <chr>         <int>      <int>      <dbl> <chr>                       <dbl>
 1 xgb_tune_1_39   500          3   0.0316   .pred_TRUE_xgb_tune_1_39 1.07
 2 xgb_tune_1_10   100          1   1.78     .pred_TRUE_xgb_tune_1_10 0.270
 3 xgb_tune_1_25   100          2   1.78     .pred_TRUE_xgb_tune_1_25 0.149
 4 xgb_tune_1_40   100          3   1.78     .pred_TRUE_xgb_tune_1_40 0.145
 5 xgb_tune_1_27   500          2   1.78     .pred_TRUE_xgb_tune_1_27 0.0594
 6 xgb_tune_1_01   100          1   0.00001  .pred_TRUE_xgb_tune_1_01 0
 7 xgb_tune_1_02   300          1   0.00001  .pred_TRUE_xgb_tune_1_02 0
 8 xgb_tune_1_03   500          1   0.00001  .pred_TRUE_xgb_tune_1_03 0
 9 xgb_tune_1_04   100          1   0.000562 .pred_TRUE_xgb_tune_1_04 0
10 xgb_tune_1_05   300          1   0.000562 .pred_TRUE_xgb_tune_1_05 0
11 xgb_tune_1_06   500          1   0.000562 .pred_TRUE_xgb_tune_1_06 0
12 xgb_tune_1_07   100          1   0.0316   .pred_TRUE_xgb_tune_1_07 0
13 xgb_tune_1_08   300          1   0.0316   .pred_TRUE_xgb_tune_1_08 0
14 xgb_tune_1_09   500          1   0.0316   .pred_TRUE_xgb_tune_1_09 0
15 xgb_tune_1_11   300          1   1.78     .pred_TRUE_xgb_tune_1_11 0
16 xgb_tune_1_12   500          1   1.78     .pred_TRUE_xgb_tune_1_12 0
17 xgb_tune_1_13   100          1 100        .pred_TRUE_xgb_tune_1_13 0
18 xgb_tune_1_16   100          2   0.00001  .pred_TRUE_xgb_tune_1_16 0
19 xgb_tune_1_17   300          2   0.00001  .pred_TRUE_xgb_tune_1_17 0
20 xgb_tune_1_18   500          2   0.00001  .pred_TRUE_xgb_tune_1_18 0
21 xgb_tune_1_19   100          2   0.000562 .pred_TRUE_xgb_tune_1_19 0
22 xgb_tune_1_20   300          2   0.000562 .pred_TRUE_xgb_tune_1_20 0
23 xgb_tune_1_21   500          2   0.000562 .pred_TRUE_xgb_tune_1_21 0
24 xgb_tune_1_22   100          2   0.0316   .pred_TRUE_xgb_tune_1_22 0
25 xgb_tune_1_23   300          2   0.0316   .pred_TRUE_xgb_tune_1_23 0
26 xgb_tune_1_24   500          2   0.0316   .pred_TRUE_xgb_tune_1_24 0
27 xgb_tune_1_26   300          2   1.78     .pred_TRUE_xgb_tune_1_26 0
28 xgb_tune_1_28   100          2 100        .pred_TRUE_xgb_tune_1_28 0
29 xgb_tune_1_31   100          3   0.00001  .pred_TRUE_xgb_tune_1_31 0
30 xgb_tune_1_32   300          3   0.00001  .pred_TRUE_xgb_tune_1_32 0
31 xgb_tune_1_33   500          3   0.00001  .pred_TRUE_xgb_tune_1_33 0
32 xgb_tune_1_34   100          3   0.000562 .pred_TRUE_xgb_tune_1_34 0
33 xgb_tune_1_35   300          3   0.000562 .pred_TRUE_xgb_tune_1_35 0
34 xgb_tune_1_36   500          3   0.000562 .pred_TRUE_xgb_tune_1_36 0
35 xgb_tune_1_37   100          3   0.0316   .pred_TRUE_xgb_tune_1_37 0
36 xgb_tune_1_38   300          3   0.0316   .pred_TRUE_xgb_tune_1_38 0
37 xgb_tune_1_41   300          3   1.78     .pred_TRUE_xgb_tune_1_41 0
```

```
38 xgb_tune_1_42   500           3   1.78      .pred_TRUE_xgb_tune_1_42 0
39 xgb_tune_1_43   100           3 100         .pred_TRUE_xgb_tune_1_43 0
```

XGB_tune contributed a little to the stacked model, more than the Log. Reg., but less than the random_forest.

```r
icu_stack_pred <- icu_test %>%
  bind_cols(predict(icu_stack, ., type = "prob")) %>%
  print(width = Inf)
```

```
# A tibble: 37,720 × 23
   subject_id  hadm_id  stay_id los_long
        <int>    <int>    <int> <fct>   
 1   10000032 29079034 39553978 FALSE   
 2   10001217 24597018 37067082 FALSE   
 3   10001217 27703517 34592300 FALSE   
 4   10001843 26133978 39698942 FALSE   
 5   10001884 26184834 37510196 TRUE    
 6   10002013 23581541 39060235 FALSE   
 7   10002428 23473524 35479615 TRUE    
 8   10002428 28662225 38875437 TRUE    
 9   10002443 21329021 35044219 TRUE    
10   10002930 25696644 37049133 FALSE   
   first_careunit                                   gender age_intime
   <chr>                                            <chr>       <int>
 1 Medical Intensive Care Unit (MICU)               F              52
 2 Surgical Intensive Care Unit (SICU)              F              55
 3 Surgical Intensive Care Unit (SICU)              F              55
 4 Medical/Surgical Intensive Care Unit (MICU/SICU) M              76
 5 Medical Intensive Care Unit (MICU)               F              77
 6 Cardiac Vascular Intensive Care Unit (CVICU)     F              57
 7 Medical Intensive Care Unit (MICU)               F              81
 8 Medical Intensive Care Unit (MICU)               F              81
 9 Coronary Care Unit (CCU)                         M              53
10 Medical Intensive Care Unit (MICU)               F              51
   marital_status race                   Heart_Rate DiaBP SysBP Respiratory_Rate
   <chr>          <chr>                       <dbl> <dbl> <dbl>            <dbl>
 1 WIDOWED        WHITE                        91    48    84               24  
 2 MARRIED        WHITE                        86    90   151               18  
 3 MARRIED        WHITE                        79.3  93.3 156               14  
 4 SINGLE         WHITE                       124.   78   110               16.5
 5 MARRIED        BLACK/AFRICAN AMERICAN       49    30.5 174.              13  
 6 SINGLE         OTHER                        80    62    98.5             14  
 7 WIDOWED        WHITE                        68.2  46    87               17.8
 8 WIDOWED        WHITE                       106.   51   102               25  
 9 SINGLE         WHITE                       106    99   140               12  
10 SINGLE         BLACK/AFRICAN AMERICAN       87    70   133               20  
    Temp Creatinine Potassium Chloride Bicarbonate Hematocrit   WBC Sodium
   <dbl>      <dbl>     <dbl>    <dbl>       <dbl>      <dbl> <dbl>  <dbl>
 1  98.7        0.7       6.7       95          25       41.1   6.9    126
 2  98.5        0.6       4.2      108          22       38.1  15.7    142
```

```
 3  97.6        0.5       4.1      104            30        37.4   5.4     142
 4  97.9        1.3       3.9       97            28        31.4  10.4     138
 5  98.1        1.1       4.5       88            30        39.7  12.2     130
 6  97.2        0.9       3.5      102            24        34.9   7.2     137
 7  97.2        0.3       3.5       95            37        29    16       136
 8  98.6        0.6       4.4      111            27        34.7  10.5     144
 9  96.7        0.9       5.3      106            18        43.1  16.9     135
10  99.2        0.4       4.1      107            16        26     4.8     134
   .pred_FALSE .pred_TRUE
         <dbl>      <dbl>
 1       0.464      0.536
 2       0.519      0.481
 3       0.685      0.315
 4       0.517      0.483
 5       0.613      0.387
 6       0.576      0.424
 7       0.310      0.690
 8       0.510      0.490
 9       0.592      0.408
10       0.574      0.426
# ℹ 37,710 more rows
```

This illustrates with the given values, the probability that the patient would have a stay longer than or equal to two days

```
yardstick::roc_auc(
  icu_stack_pred,
  truth = los_long,
  .pred_TRUE
)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.355
```

This ROC_AUC of the stacked model is 0.354. This means that this is worse than random guessing (which would be 50%). This is also the worst model compared to the Log. Reg., RF, and XGBoost.

```
icu_pred <-
  icu_test |>
  select(los_long) |>
  bind_cols(
    predict(
      icu_stack,
      icu_test,
      type = "class",
      members = TRUE
    )
```

```
) |>
  print(width = Inf)
```

```
# A tibble: 37,720 × 17
   los_long .pred_class .pred_class_logit_tune_1_101 .pred_class_rf_tune_1_13
   <fct>    <fct>       <fct>                        <fct>
 1 FALSE    TRUE        FALSE                        TRUE
 2 FALSE    FALSE       FALSE                        FALSE
 3 FALSE    FALSE       FALSE                        FALSE
 4 FALSE    FALSE       TRUE                         FALSE
 5 TRUE     FALSE       FALSE                        FALSE
 6 FALSE    FALSE       FALSE                        FALSE
 7 TRUE     TRUE        FALSE                        TRUE
 8 TRUE     FALSE       FALSE                        FALSE
 9 TRUE     FALSE       FALSE                        FALSE
10 FALSE    FALSE       FALSE                        FALSE
   .pred_class_rf_tune_1_16 .pred_class_rf_tune_1_17 .pred_class_rf_tune_1_18
   <fct>                    <fct>                    <fct>
 1 FALSE                    TRUE                     TRUE
 2 TRUE                     FALSE                    FALSE
 3 FALSE                    FALSE                    FALSE
 4 FALSE                    FALSE                    FALSE
 5 FALSE                    FALSE                    FALSE
 6 FALSE                    FALSE                    FALSE
 7 TRUE                     TRUE                     TRUE
 8 FALSE                    TRUE                     FALSE
 9 TRUE                     FALSE                    FALSE
10 FALSE                    FALSE                    FALSE
   .pred_class_rf_tune_1_19 .pred_class_rf_tune_1_21 .pred_class_rf_tune_1_23
   <fct>                    <fct>                    <fct>
 1 TRUE                     TRUE                     TRUE
 2 FALSE                    TRUE                     TRUE
 3 FALSE                    FALSE                    FALSE
 4 FALSE                    FALSE                    FALSE
 5 FALSE                    FALSE                    FALSE
 6 FALSE                    FALSE                    FALSE
 7 TRUE                     TRUE                     TRUE
 8 FALSE                    TRUE                     FALSE
 9 FALSE                    FALSE                    FALSE
10 FALSE                    FALSE                    FALSE
   .pred_class_rf_tune_1_24 .pred_class_rf_tune_1_25 .pred_class_xgb_tune_1_10
   <fct>                    <fct>                    <fct>
 1 TRUE                     TRUE                     TRUE
 2 TRUE                     FALSE                    FALSE
 3 FALSE                    FALSE                    FALSE
 4 FALSE                    FALSE                    FALSE
 5 FALSE                    FALSE                    FALSE
 6 FALSE                    FALSE                    FALSE
 7 TRUE                     TRUE                     TRUE
 8 TRUE                     TRUE                     FALSE
```

```
 9 FALSE                      FALSE                        FALSE
10 FALSE                      FALSE                        FALSE
   .pred_class_xgb_tune_1_25 .pred_class_xgb_tune_1_40 .pred_class_xgb_tune_1_27
   <fct>                      <fct>                      <fct>
 1 FALSE                      FALSE                      FALSE
 2 FALSE                      FALSE                      FALSE
 3 FALSE                      FALSE                      FALSE
 4 TRUE                       FALSE                      FALSE
 5 FALSE                      FALSE                      FALSE
 6 FALSE                      FALSE                      FALSE
 7 TRUE                       TRUE                       FALSE
 8 FALSE                      TRUE                       FALSE
 9 FALSE                      TRUE                       FALSE
10 FALSE                      TRUE                       FALSE
   .pred_class_xgb_tune_1_39
   <fct>
 1 TRUE
 2 FALSE
 3 FALSE
 4 FALSE
 5 FALSE
 6 FALSE
 7 TRUE
 8 FALSE
 9 FALSE
10 FALSE
# ℹ 37,710 more rows
```

```
icu_pred_accuracy <-
  map(
    colnames(icu_pred)[-1],
    ~mean(icu_pred$los_long == pull(icu_pred, .x))
  ) |>
  set_names(colnames(icu_pred)[-1]) |>
  as_tibble() |>
  pivot_longer(cols = everything(), names_to = "model", values_to = "accuracy")

icu_pred_accuracy
```

```
# A tibble: 16 × 2
   model                         accuracy
   <chr>                            <dbl>
 1 .pred_class                      0.604
 2 .pred_class_logit_tune_1_101     0.579
 3 .pred_class_rf_tune_1_13         0.601
 4 .pred_class_rf_tune_1_16         0.598
 5 .pred_class_rf_tune_1_17         0.600
 6 .pred_class_rf_tune_1_18         0.603
 7 .pred_class_rf_tune_1_19         0.604
 8 .pred_class_rf_tune_1_21         0.598
```

```
 9 .pred_class_rf_tune_1_23       0.602
10 .pred_class_rf_tune_1_24       0.603
11 .pred_class_rf_tune_1_25       0.603
12 .pred_class_xgb_tune_1_10      0.587
13 .pred_class_xgb_tune_1_25      0.553
14 .pred_class_xgb_tune_1_40      0.569
15 .pred_class_xgb_tune_1_27      0.524
16 .pred_class_xgb_tune_1_39      0.599
```

Looking at the .pred_class, the accuracy of the model is 0.6038. This is better than the accuracy of the RF model and the XGBoost model. However, this model is better than the RF model accuracy wise by 0.0007. This would also make it better than the logistic regression model as well.

Specifically focusing on performance, I would say that the RF model had the best performance out of the three. This is because it gave back both the highest ROC AUC and accuracy, while also taking a reasonable amount of time to load. Although the logistic regression was fast, it was not that accurate. The XGBoost was significantly slower and yield worse results than the RF Model, while the stacked model had good accuracy, but poor ROC AUC. My recommendation would be to use the random_forest if time permits, especially considering that the accuracy was only worse than the stacked model by 0.0007.

I think out of the four models, the logistic regression was the easiest one to interpret as you could just look at the estimates and see which one had the highest one and impacted the model the most. The random forest and XGBoost models were harder to interpret as they had features that impacted the model in different ways. The stacked model was also hard to interpret as it was a combination of the three models, so it was hard to see which model had the most impact on the final prediction, and it was very time consuming as well. I stick by my suggestion for random forest as it seems to have the best balance of performance and interpretability