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
```

```r
library(GGally)
```

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

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

```
Attaching package: 'xgboost'

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

    slice
```

```r
library(vip)
```

```
Attaching package: 'vip'

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

    vi
```

```r
# 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 = factor(los >= 2, levels = c(FALSE, TRUE),
                          labels = c("FALSE", "TRUE")))

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, Glucose)

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, Glucose)

##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          Glucose 
               0                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 × 22
  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 15 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>, Glucose <dbl>
```

```
dim(icu_train)
```

```
[1] 37569    22
```

```
head(icu_test)
```

```
# A tibble: 6 × 22
  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 15 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>, Glucose <dbl>
```

```
dim(icu_test)
```

```
[1] 37571    22
```

```
##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 + Glucose,
  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: 18

── 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()
```

```
ℹ The workflow being saved contains a recipe, which is 5.23 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.

   user  system elapsed
 64.342   9.197  74.823
```

```
logit_tune
```

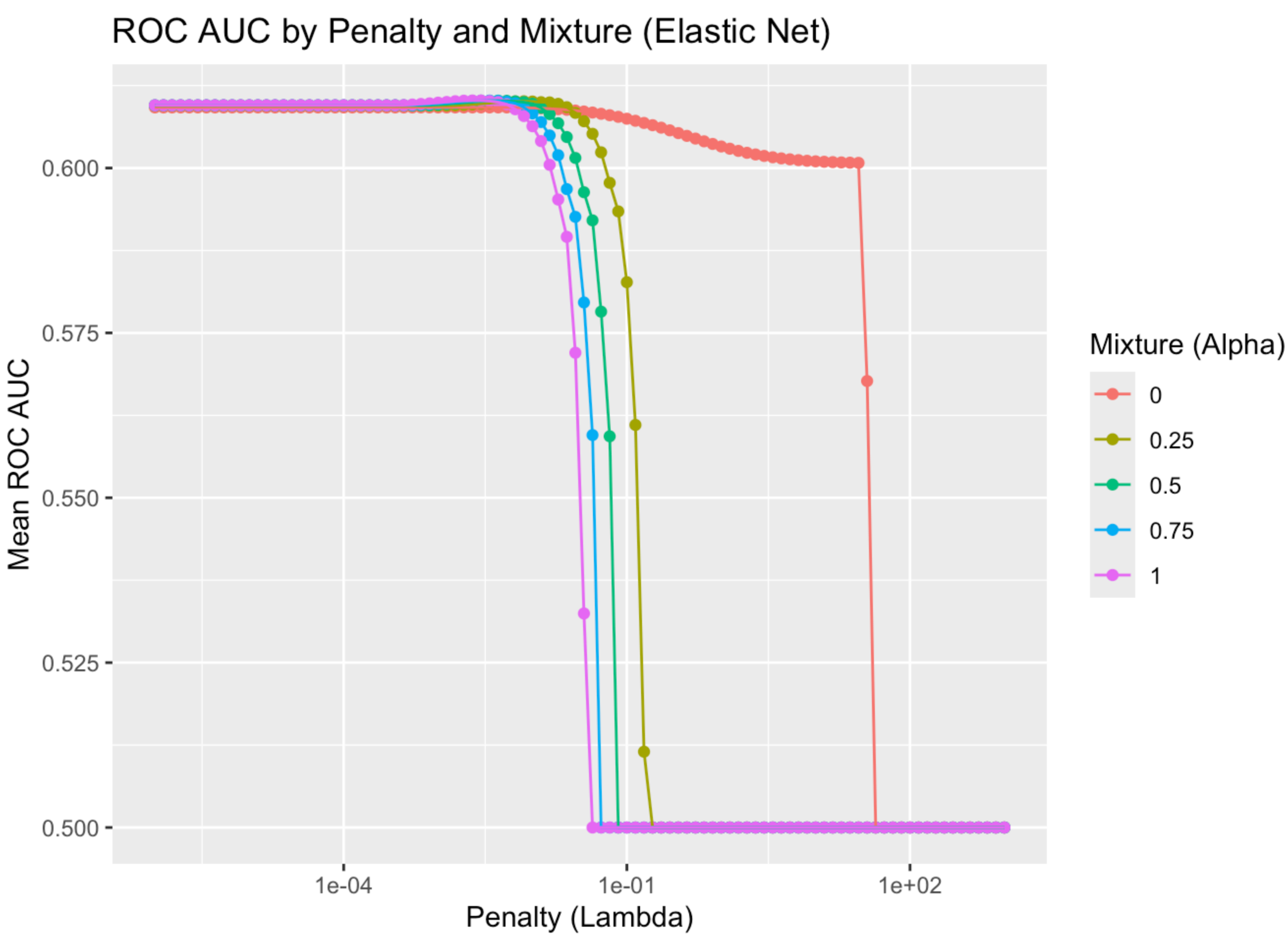
```
# Tuning results
# 3-fold cross-validation
# A tibble: 3 × 5
  splits                id    .metrics             .notes           .predictions
  <list>                <chr> <list>               <list>           <list>
1 <split [25046/12523]> Fold1 <tibble [1,000 × 6]> <tibble [0 × 4]> <tibble>
2 <split [25046/12523]> Fold2 <tibble [1,000 × 6]> <tibble [0 × 4]> <tibble>
3 <split [25046/12523]> 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     0.609     3 0.00192 Preprocessor1_Mode…
 2 0.00000123      0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
```

```
 3 0.00000152       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
 4 0.00000187       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
 5 0.00000231       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
 6 0.00000285       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
 7 0.00000351       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
 8 0.00000433       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
 9 0.00000534       0 roc_auc binary     0.609     3 0.00192 Preprocessor1_Mode…
10 0.00000658       0 roc_auc binary     0.609     3 0.00192 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 0.00285    1    roc_auc binary     0.610     3 0.00216 Preprocessor1_Model439
2 0.00231    1    roc_auc binary     0.610     3 0.00213 Preprocessor1_Model438
3 0.00351    0.75 roc_auc binary     0.610     3 0.00215 Preprocessor1_Model340
4 0.00285    0.75 roc_auc binary     0.610     3 0.00210 Preprocessor1_Model339
5 0.00433    0.5  roc_auc binary     0.610     3 0.00210 Preprocessor1_Model241
```

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

```
# A tibble: 1 × 3
  penalty mixture .config
    <dbl>   <dbl> <chr>
1 0.00285       1 Preprocessor1_Model439
```

```
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         0.607 Preprocessor1_Model1
3 brier_class binary         0.241 Preprocessor1_Model1
```

```
predictions <- collect_predictions(final_logit_fit)

predictions %>%
  select(los_long, .pred_TRUE, .pred_class) %>%
  head(10)
```

```
# A tibble: 10 × 3
   los_long .pred_TRUE .pred_class
   <fct>         <dbl> <fct>
 1 FALSE         0.449 FALSE
 2 FALSE         0.419 FALSE
 3 FALSE         0.385 FALSE
 4 FALSE         0.514 TRUE
 5 TRUE          0.314 FALSE
 6 FALSE         0.472 FALSE
 7 TRUE          0.494 FALSE
 8 TRUE          0.504 TRUE
 9 TRUE          0.425 FALSE
10 FALSE         0.396 FALSE
```

```
conf_mat(predictions, truth = los_long, estimate = .pred_class)
```

```
          Truth
Prediction FALSE  TRUE
     FALSE 12655  9197
     TRUE   6608  9111
```

```
final_model <- extract_fit_parsnip(final_logit_fit$.workflow[[1]])
tidy(final_model) %>%
  arrange(desc(estimate)) %>%
  print(n = Inf)
```

```
# A tibble: 66 × 3
   term                                                           estimate penalty
   <chr>                                                             <dbl>   <dbl>
 1 first_careunit_Neuro.Intermediate                               1.54e-1 0.00285
 2 Heart_Rate                                                      1.44e-1 0.00285
 3 Respiratory_Rate                                                1.22e-1 0.00285
 4 age_intime                                                      1.01e-1 0.00285
 5 WBC                                                             6.44e-2 0.00285
 6 first_careunit_Surgery.Vascular.Intermediate                    6.39e-2 0.00285
 7 Creatinine                                                      4.52e-2 0.00285
 8 gender_M                                                        3.43e-2 0.00285
 9 race_UNKNOWN                                                    3.04e-2 0.00285
10 Bicarbonate                                                     2.35e-2 0.00285
11 race_UNABLE.TO.OBTAIN                                           2.20e-2 0.00285
12 first_careunit_Neuro.Stepdown                                   1.84e-2 0.00285
13 first_careunit_Neuro.Surgical.Intensive.Care.Unit..Neuro.SI…    9.03e-3 0.00285
14 Glucose                                                         7.96e-3 0.00285
15 race_PORTUGUESE                                                 4.98e-3 0.00285
16 first_careunit_Medicine                                         4.00e-3 0.00285
17 race_HISPANIC.LATINO...DOMINICAN                                3.68e-3 0.00285
18 first_careunit_Surgery.Trauma                                   3.67e-3 0.00285
19 race_SOUTH.AMERICAN                                             2.73e-3 0.00285
20 first_careunit_Intensive.Care.Unit..ICU.                        1.41e-3 0.00285
```

```
21 race_WHITE...BRAZILIAN                                   7.66e-4 0.00285
22 Sodium                                                   4.14e-4 0.00285
23 DiaBP                                                    0       0.00285
24 Temp                                                     0       0.00285
25 Potassium                                                0       0.00285
26 first_careunit_Med.Surg                                  0       0.00285
27 first_careunit_Neurology                                 0       0.00285
28 first_careunit_PACU                                      0       0.00285
29 first_careunit_Surgical.Intensive.Care.Unit..SICU.       0       0.00285
30 marital_status_MARRIED                                   0       0.00285
31 marital_status_SINGLE                                    0       0.00285
32 race_ASIAN                                               0       0.00285
33 race_ASIAN...CHINESE                                     0       0.00285
34 race_ASIAN...KOREAN                                      0       0.00285
35 race_ASIAN...SOUTH.EAST.ASIAN                            0       0.00285
36 race_BLACK.AFRICAN                                       0       0.00285
37 race_BLACK.CAPE.VERDEAN                                  0       0.00285
38 race_BLACK.CARIBBEAN.ISLAND                              0       0.00285
39 race_HISPANIC.LATINO...CENTRAL.AMERICAN                  0       0.00285
40 race_HISPANIC.LATINO...COLUMBIAN                         0       0.00285
41 race_HISPANIC.LATINO...CUBAN                             0       0.00285
42 race_HISPANIC.LATINO...GUATEMALAN                        0       0.00285
43 race_HISPANIC.LATINO...HONDURAN                          0       0.00285
44 race_HISPANIC.LATINO...MEXICAN                           0       0.00285
45 race_HISPANIC.LATINO...PUERTO.RICAN                      0       0.00285
46 race_HISPANIC.LATINO...SALVADORAN                        0       0.00285
47 race_MULTIPLE.RACE.ETHNICITY                             0       0.00285
48 race_NATIVE.HAWAIIAN.OR.OTHER.PACIFIC.ISLANDER           0       0.00285
49 race_OTHER                                               0       0.00285
50 race_WHITE                                               0       0.00285
51 race_WHITE...RUSSIAN                                     0       0.00285
52 race_WHITE...OTHER.EUROPEAN                             -9.85e-6 0.00285
53 race_PATIENT.DECLINED.TO.ANSWER                         -1.20e-3 0.00285
54 race_WHITE...EASTERN.EUROPEAN                           -2.91e-3 0.00285
55 race_ASIAN...ASIAN.INDIAN                               -2.98e-3 0.00285
56 race_BLACK.AFRICAN.AMERICAN                             -4.59e-3 0.00285
57 race_HISPANIC.OR.LATINO                                 -5.67e-3 0.00285
58 first_careunit_Coronary.Care.Unit..CCU.                 -1.39e-2 0.00285
59 marital_status_WIDOWED                                  -2.52e-2 0.00285
60 first_careunit_Trauma.SICU..TSICU.                      -4.27e-2 0.00285
61 (Intercept)                                             -5.07e-2 0.00285
62 Chloride                                                -5.29e-2 0.00285
63 SysBP                                                   -1.15e-1 0.00285
64 Hematocrit                                              -1.16e-1 0.00285
65 first_careunit_Medical.Intensive.Care.Unit..MICU.       -1.17e-1 0.00285
66 first_careunit_Medical.Surgical.Intensive.Care.Unit..MICU.S… -1.75e-1 0.00285
```

```r
levels(predictions$los_long)
```

```
[1] "FALSE" "TRUE"
```

```
roc_auc(predictions, truth = los_long, .pred_TRUE, event_level = "second")
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.607
```

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: first_careunit_Neuro.intermediate, heart_rate, respiratory_rate, age_intime, and WBC. 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. Those with a higher white blood cell count also makes sense as this indicates an infection is being fought within the body. It is worth noting that the negative predictors of this model include, SysBP, Hematocrit, First_careunit_medical.intensive care unit, and then the same, but surgical

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, Glucose)

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, Glucose)
```

```r
##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          Glucose 
                0                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)

# 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 × 22
  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 15 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>, Glucose <dbl>
```

```r
head(icu_test)
```

```
# A tibble: 6 × 22
  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 15 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>, Glucose <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 + Glucose,
  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
```

```r
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
```

```
#We Received an error regarding the following command, so let us do it to satsify it

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)
)
```

```
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()
)
```

```
ℹ The workflow being saved contains a recipe, which is 5.23 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

```
rf_tune
```

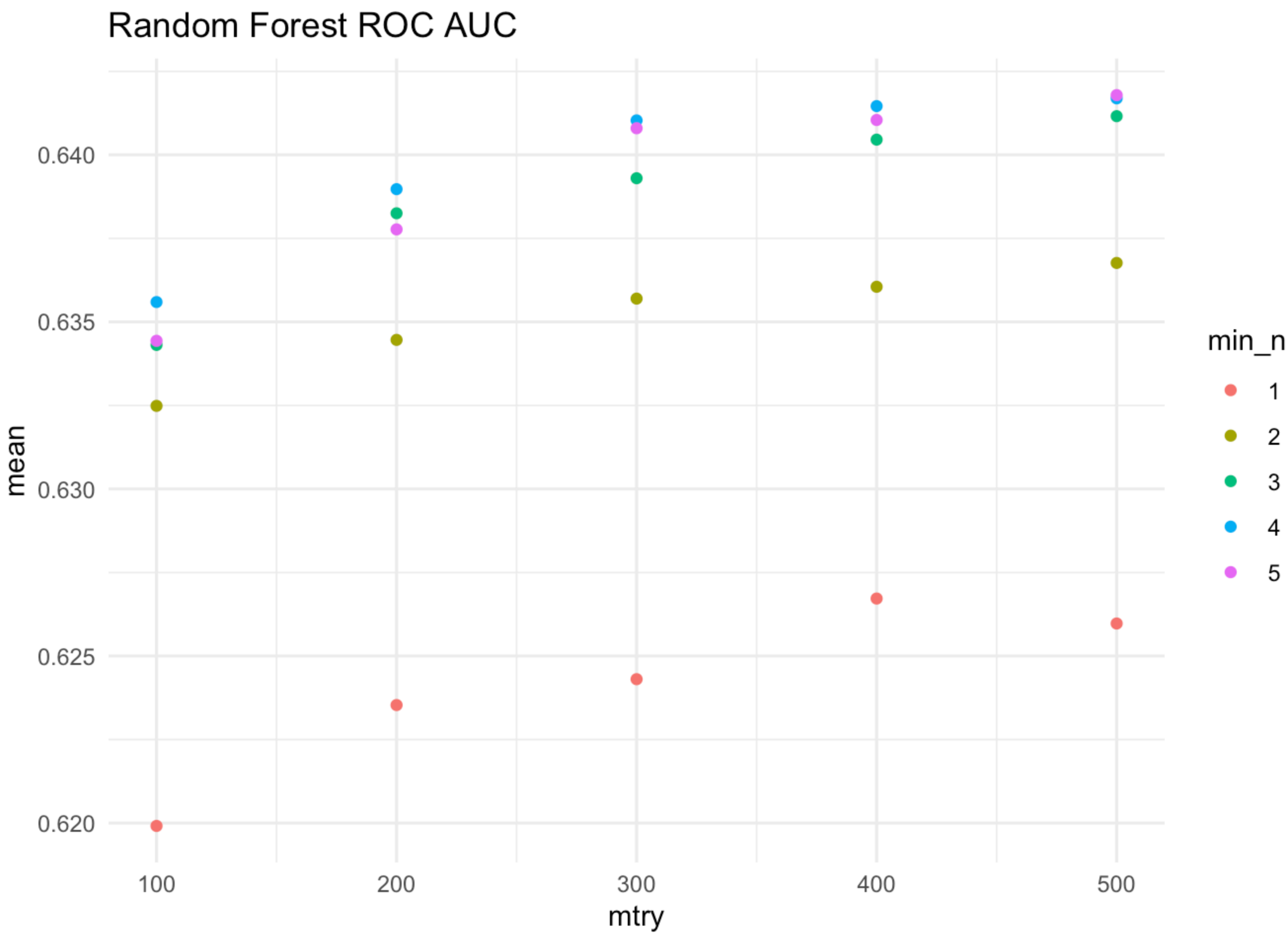
```
# Tuning results
# 3-fold cross-validation
# A tibble: 3 × 5
  splits                id    .metrics          .notes           .predictions
  <list>                <chr> <list>            <list>           <list>
1 <split [25046/12523]> Fold1 <tibble [50 × 6]> <tibble [0 × 4]> <tibble>
2 <split [25046/12523]> Fold2 <tibble [50 × 6]> <tibble [0 × 4]> <tibble>
3 <split [25046/12523]> Fold3 <tibble [50 × 6]> <tibble [0 × 4]> <tibble>
```

```
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 0.00575 Preprocessor1_Model01
 2     1   100 roc_auc  binary     0.620     3 0.00139 Preprocessor1_Model01
 3     1   200 accuracy binary     0.569     3 0.00900 Preprocessor1_Model02
 4     1   200 roc_auc  binary     0.624     3 0.00156 Preprocessor1_Model02
 5     1   300 accuracy binary     0.568     3 0.00528 Preprocessor1_Model03
 6     1   300 roc_auc  binary     0.624     3 0.00128 Preprocessor1_Model03
 7     1   400 accuracy binary     0.572     3 0.00481 Preprocessor1_Model04
 8     1   400 roc_auc  binary     0.627     3 0.00215 Preprocessor1_Model04
 9     1   500 accuracy binary     0.571     3 0.00643 Preprocessor1_Model05
10     1   500 roc_auc  binary     0.626     3 0.00240 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 0.00575 Preprocessor1_Model01
 2     1   100 roc_auc  binary     0.620     3 0.00139 Preprocessor1_Model01
 3     1   200 accuracy binary     0.569     3 0.00900 Preprocessor1_Model02
 4     1   200 roc_auc  binary     0.624     3 0.00156 Preprocessor1_Model02
 5     1   300 accuracy binary     0.568     3 0.00528 Preprocessor1_Model03
 6     1   300 roc_auc  binary     0.624     3 0.00128 Preprocessor1_Model03
 7     1   400 accuracy binary     0.572     3 0.00481 Preprocessor1_Model04
 8     1   400 roc_auc  binary     0.627     3 0.00215 Preprocessor1_Model04
 9     1   500 accuracy binary     0.571     3 0.00643 Preprocessor1_Model05
10     1   500 roc_auc  binary     0.626     3 0.00240 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   500 roc_auc binary     0.642     3 0.00144 Preprocessor1_Model25
2     4   500 roc_auc binary     0.642     3 0.00163 Preprocessor1_Model20
3     4   400 roc_auc binary     0.641     3 0.00195 Preprocessor1_Model19
4     3   500 roc_auc binary     0.641     3 0.00179 Preprocessor1_Model15
5     5   400 roc_auc binary     0.641     3 0.00203 Preprocessor1_Model24
```

The best metric seems to be when the mtry is five and the trees are 500.

```
# 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 [37569/37571]> 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         0.606 Preprocessor1_Model1
2 roc_auc     binary         0.646 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 12600  8126
     TRUE   6663 10182
```

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: 65 × 2
   feature                                                         importance
   <chr>                                                                <dbl>
 1 SysBP                                                                 690.
 2 Hematocrit                                                            674.
 3 WBC                                                                   660.
 4 Heart_Rate                                                            654.
 5 Glucose                                                               631.
 6 age_intime                                                            622.
 7 DiaBP                                                                 610.
```

```
 8 Respiratory_Rate                                                   586.
 9 Temp                                                               583.
10 Creatinine                                                         525.
11 Potassium                                                          510.
12 Bicarbonate                                                        497.
13 Chloride                                                           487.
14 Sodium                                                             478.
15 first_careunit_Neuro.Intermediate                                  121.
16 first_careunit_Medical.Surgical.Intensive.Care.Unit..MICU.SICU.    104.
17 gender_M                                                            96.4
18 race_WHITE                                                          90.8
19 first_careunit_Medical.Intensive.Care.Unit..MICU.                   88.5
20 marital_status_MARRIED                                              84.3
21 marital_status_SINGLE                                               75.6
22 first_careunit_Surgical.Intensive.Care.Unit..SICU.                  68.8
23 race_BLACK.AFRICAN.AMERICAN                                         66.6
24 first_careunit_Coronary.Care.Unit..CCU.                             65.4
25 first_careunit_Trauma.SICU..TSICU.                                  64.2
26 marital_status_WIDOWED                                              64.1
27 race_OTHER                                                          46.5
28 race_UNKNOWN                                                        42.8
29 race_WHITE...OTHER.EUROPEAN                                         41.7
30 race_HISPANIC.LATINO...PUERTO.RICAN                                 28.6
31 first_careunit_Neuro.Stepdown                                       28.0
32 first_careunit_Neuro.Surgical.Intensive.Care.Unit..Neuro.SICU.      27.6
33 race_ASIAN...CHINESE                                                27.3
34 race_ASIAN                                                          26.0
35 race_WHITE...RUSSIAN                                                25.0
36 race_HISPANIC.LATINO...DOMINICAN                                    19.5
37 race_BLACK.CAPE.VERDEAN                                             19.0
38 race_UNABLE.TO.OBTAIN                                               18.9
39 race_HISPANIC.OR.LATINO                                             18.7
40 race_BLACK.CARIBBEAN.ISLAND                                         17.1
41 race_PORTUGUESE                                                     14.1
42 race_BLACK.AFRICAN                                                  13.7
43 race_PATIENT.DECLINED.TO.ANSWER                                     13.3
44 race_ASIAN...SOUTH.EAST.ASIAN                                       13.0
45 race_WHITE...EASTERN.EUROPEAN                                       10.1
46 first_careunit_Surgery.Vascular.Intermediate                        10.1
47 race_HISPANIC.LATINO...GUATEMALAN                                    8.88
48 race_ASIAN...ASIAN.INDIAN                                            7.96
49 race_WHITE...BRAZILIAN                                               7.08
50 race_HISPANIC.LATINO...SALVADORAN                                    6.62
51 race_SOUTH.AMERICAN                                                  5.92
52 first_careunit_PACU                                                  4.67
53 race_HISPANIC.LATINO...COLUMBIAN                                     4.62
54 race_HISPANIC.LATINO...MEXICAN                                       4.16
55 race_HISPANIC.LATINO...CUBAN                                         3.68
56 race_HISPANIC.LATINO...HONDURAN                                      3.51
57 race_NATIVE.HAWAIIAN.OR.OTHER.PACIFIC.ISLANDER                       3.49
58 race_HISPANIC.LATINO...CENTRAL.AMERICAN                              3.08
```

```
59 race_MULTIPLE.RACE.ETHNICITY                          2.69
60 race_ASIAN...KOREAN                                   1.98
61 first_careunit_Medicine                               0.742
62 first_careunit_Intensive.Care.Unit..ICU.              0.507
63 first_careunit_Surgery.Trauma                         0.249
64 first_careunit_Med.Surg                               0.0864
65 first_careunit_Neurology                              0.0834
```

Let us graph it for better visualization using GGPlot

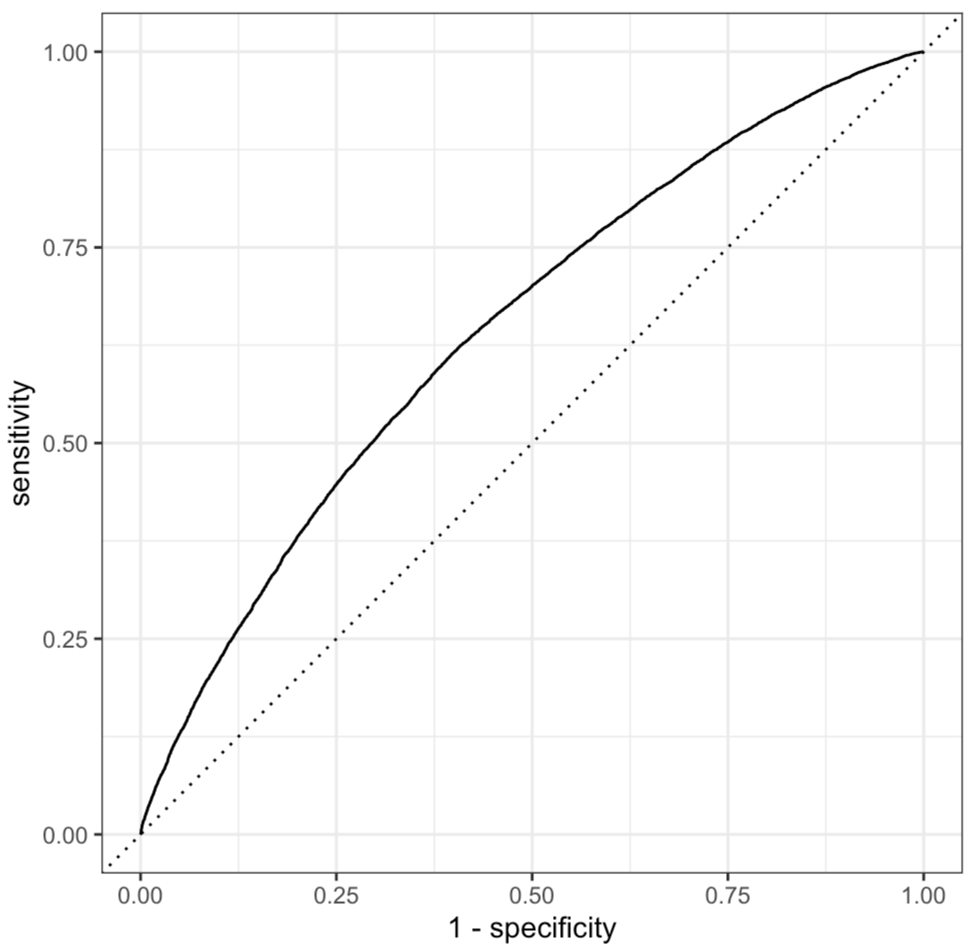
```
levels(predictions_rf$los_long)
```

```
[1] "FALSE" "TRUE"
```

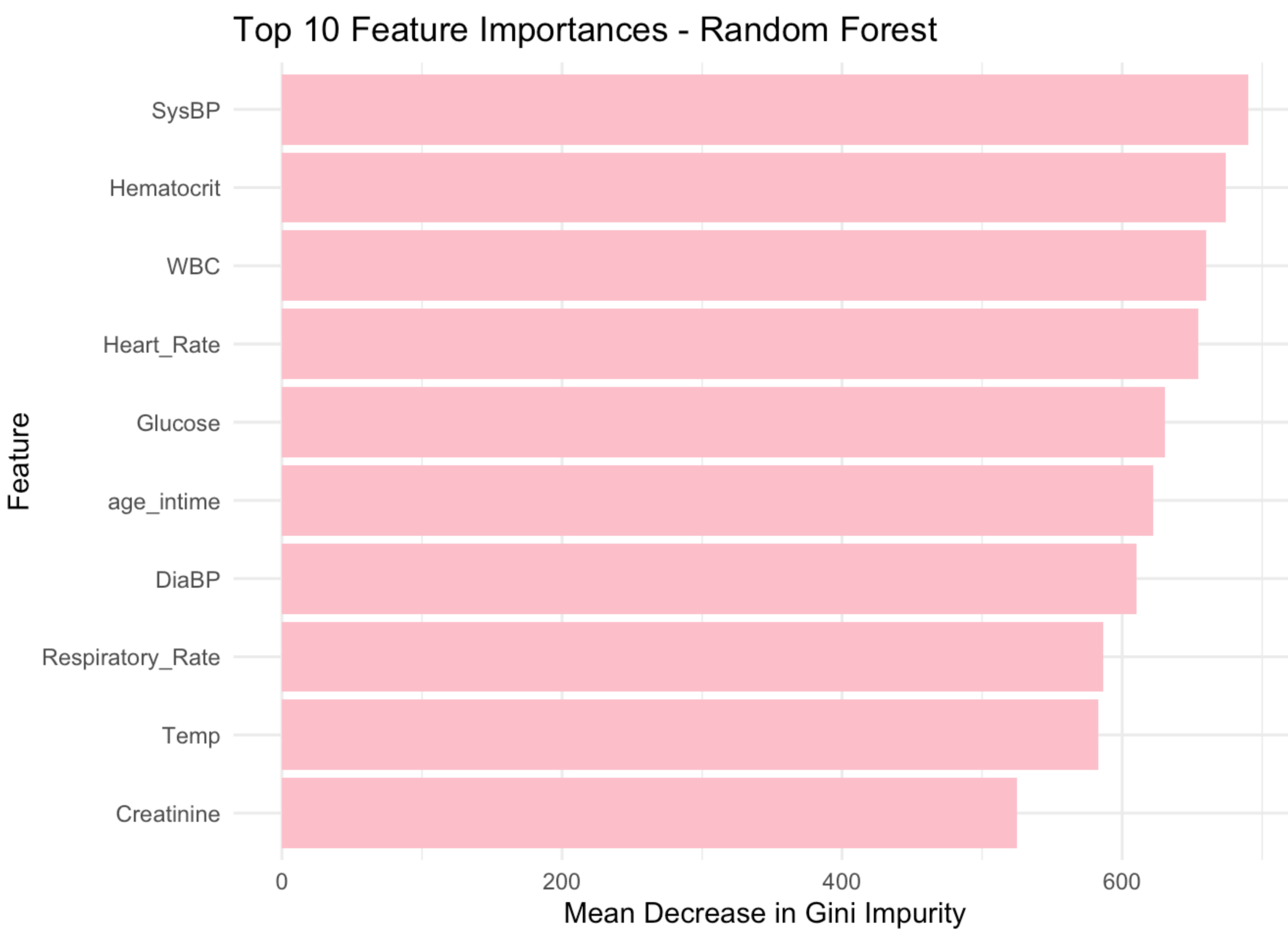
```
roc_auc(predictions_rf, truth = los_long, .pred_TRUE, event_level = "second")
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.646
```

```
roc_curve(predictions_rf, truth = los_long,
          .pred_TRUE, event_level = "second") %>%
  autoplot()
```



```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 0.606, while the ROC AUC is 0.6459. 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 0.58, respectively, the Random Forest model is better at predicting long ICU stays.

Looking at the importance of variables, SysBP, Hematocrit, WBC, Heart_Rate, and glucose 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 importance 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, Glucose is an indicator of length of stay >= 2 as well. This could be indicative of organs like the pancreas working less so more glucose is building up in the blood stream, causing stress induced hyperglycemia. See below for a more specific reasoning as well

Comparing this to the Logistic Regression as well, the only similarities is Heart_Rate. This makes sense as the logistic regression penalizes coefficients when it does not add onto the model itself. Random Forest makes it so the most variables that split the trees (are involved the most) are highlighted with more importance. The logistic regression also had SysBP as a negative predictor, so it is interesting to see it arise with the most importance in the Random Forest

XGBoost

Data Pre-Processing 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, Glucose)

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)

levels(icu_data$los_long)
```

```
[1] "FALSE" "TRUE"
```

```
##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, Glucose)

##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          Glucose 
              0                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 × 22
  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 15 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>, Glucose <dbl>
```

```r
head(icu_test)
```

```
# A tibble: 6 × 22
  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 15 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>, Glucose <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 + Glucose,
  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

```
#We received a similar error as before, so use hardhat to remove it

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()
)
```

```
ℹ The workflow being saved contains a recipe, which is 5.23 Mb in ℹ memory. If
this was not intentional, please set the control setting ℹ `save_workflow =
FALSE`.
```

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

```
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         0.601 Preprocessor1_Model1
2 roc_auc     binary         0.641 Preprocessor1_Model1
3 brier_class binary         0.234 Preprocessor1_Model1
```

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

```
          Truth
Prediction FALSE  TRUE
     FALSE 12929  8654
     TRUE   6334  9654
```

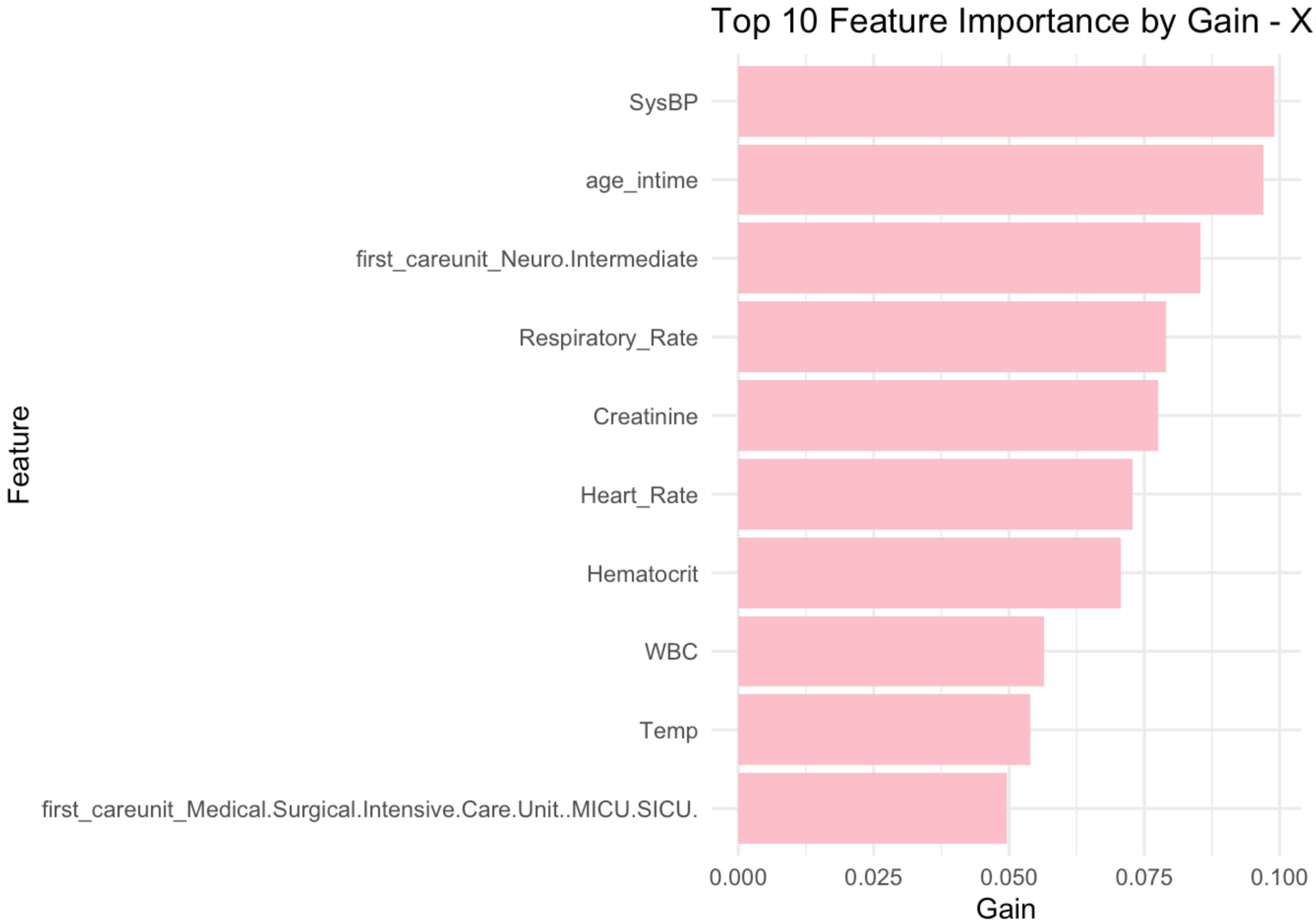
```
final_modelxgb <- extract_fit_parsnip(final_xgb_fit$.workflow[[1]])

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: 37 × 4
   Feature                                                  Gain   Cover Frequency
   <chr>                                                   <dbl>   <dbl>     <dbl>
 1 SysBP                                                 9.90e-2 5.13e-2  0.0568
 2 age_intime                                            9.70e-2 6.72e-2  0.0843
 3 first_careunit_Neuro.Intermediate                     8.53e-2 4.42e-2  0.0352
 4 Respiratory_Rate                                      7.90e-2 6.87e-2  0.0606
 5 Creatinine                                            7.76e-2 8.60e-2  0.0710
```

```
 6 Heart_Rate                                         7.29e-2 6.43e-2  0.0781
 7 Hematocrit                                         7.06e-2 8.03e-2  0.0781
 8 WBC                                                5.65e-2 4.56e-2  0.0473
 9 Temp                                               5.39e-2 6.60e-2  0.0568
10 first_careunit_Medical.Surgical.Intensive.Care.Uni… 4.96e-2 5.63e-2  0.0328
11 Bicarbonate                                        4.17e-2 4.73e-2  0.0470
12 Glucose                                            4.08e-2 3.55e-2  0.0689
13 first_careunit_Medical.Intensive.Care.Unit..MICU.   3.56e-2 4.28e-2  0.0272
14 Sodium                                             2.98e-2 6.36e-2  0.0538
15 DiaBP                                              2.17e-2 2.58e-2  0.0438
16 Chloride                                           1.73e-2 1.99e-2  0.0308
17 Potassium                                          1.39e-2 4.79e-3  0.0284
18 first_careunit_Surgery.Vascular.Intermediate       1.26e-2 3.83e-2  0.0172
19 gender_M                                           9.23e-3 9.85e-3  0.0163
20 first_careunit_Trauma.SICU..TSICU.                 6.95e-3 1.74e-2  0.0101
21 race_UNKNOWN                                       5.13e-3 1.33e-2  0.00887
22 first_careunit_Neuro.Stepdown                      4.84e-3 1.73e-2  0.00858
23 race_BLACK.AFRICAN.AMERICAN                        4.56e-3 6.01e-3  0.00621
24 first_careunit_Coronary.Care.Unit..CCU.            2.02e-3 6.10e-3  0.00414
25 marital_status_WIDOWED                             1.98e-3 2.06e-3  0.00385
26 marital_status_MARRIED                             1.89e-3 4.89e-4  0.00325
27 race_UNABLE.TO.OBTAIN                              1.88e-3 1.04e-2  0.00473
28 first_careunit_Neuro.Surgical.Intensive.Care.Unit.… 1.23e-3 3.86e-3  0.00296
29 race_HISPANIC.LATINO...DOMINICAN                   1.06e-3 1.67e-3  0.00266
30 first_careunit_Surgical.Intensive.Care.Unit..SICU.  9.01e-4 3.70e-4  0.00148
31 race_WHITE                                         8.63e-4 7.87e-5  0.00118
32 race_ASIAN...CHINESE                               7.59e-4 1.04e-3  0.00266
33 race_WHITE...RUSSIAN                               6.20e-4 1.97e-4  0.00148
34 race_HISPANIC.OR.LATINO                            5.22e-4 3.82e-5  0.00118
35 marital_status_SINGLE                              3.49e-4 1.70e-5  0.000887
36 race_WHITE...BRAZILIAN                             2.93e-4 2.61e-4  0.000592
37 race_WHITE...EASTERN.EUROPEAN                      2.43e-4 1.78e-3  0.000887
```

Let us plot it in GGPlot for better visualization

```r
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 0.601 and 0.641, respectively. This indicates that the random_forest was better able to predict los_long >= 2 days, but the XGBoost 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 first_careunit_Neuro.Intermediate.

Interestingly, SysBP, age_intime, Respiratory_Rate, first_careunit being the intermediate Neuro one, and Creatinine, 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 one of same features as the random forest model (SysBP). 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. They did not share anything else otherwise. 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 other words, it looks like SysBP was a major component across all three models.

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 × 22
  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 15 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>, Glucose <dbl>
```

```
head(icu_test)
```

```
# A tibble: 6 × 22
  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 15 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>, Glucose <dbl>
```

```
##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 + Glucose,
  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())
```

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

Final Model Stacking:

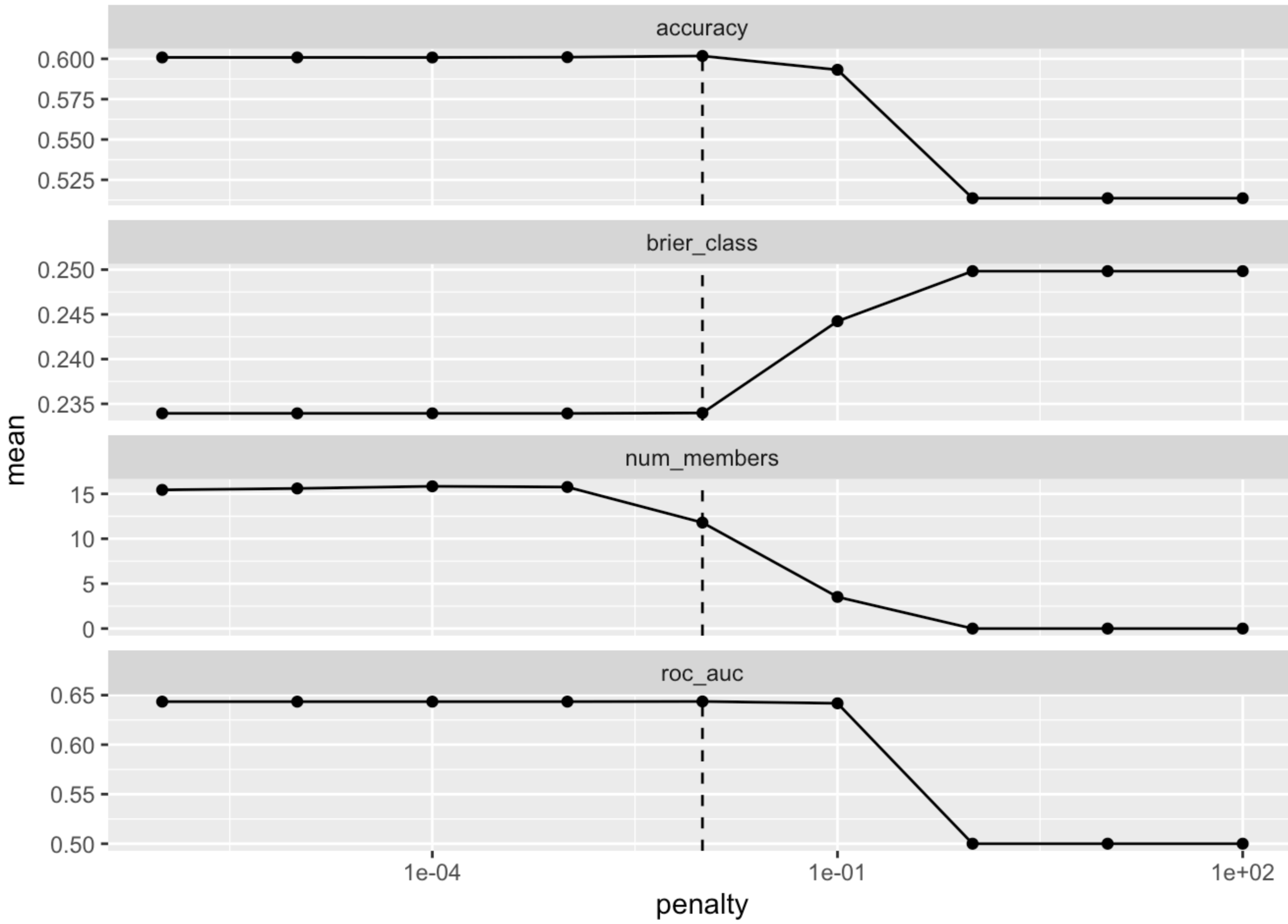
```
#The penalty was decided in office hours with Dr. Zhou to speed up the loading process of

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 724 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'
```

```r
icu_stack
```

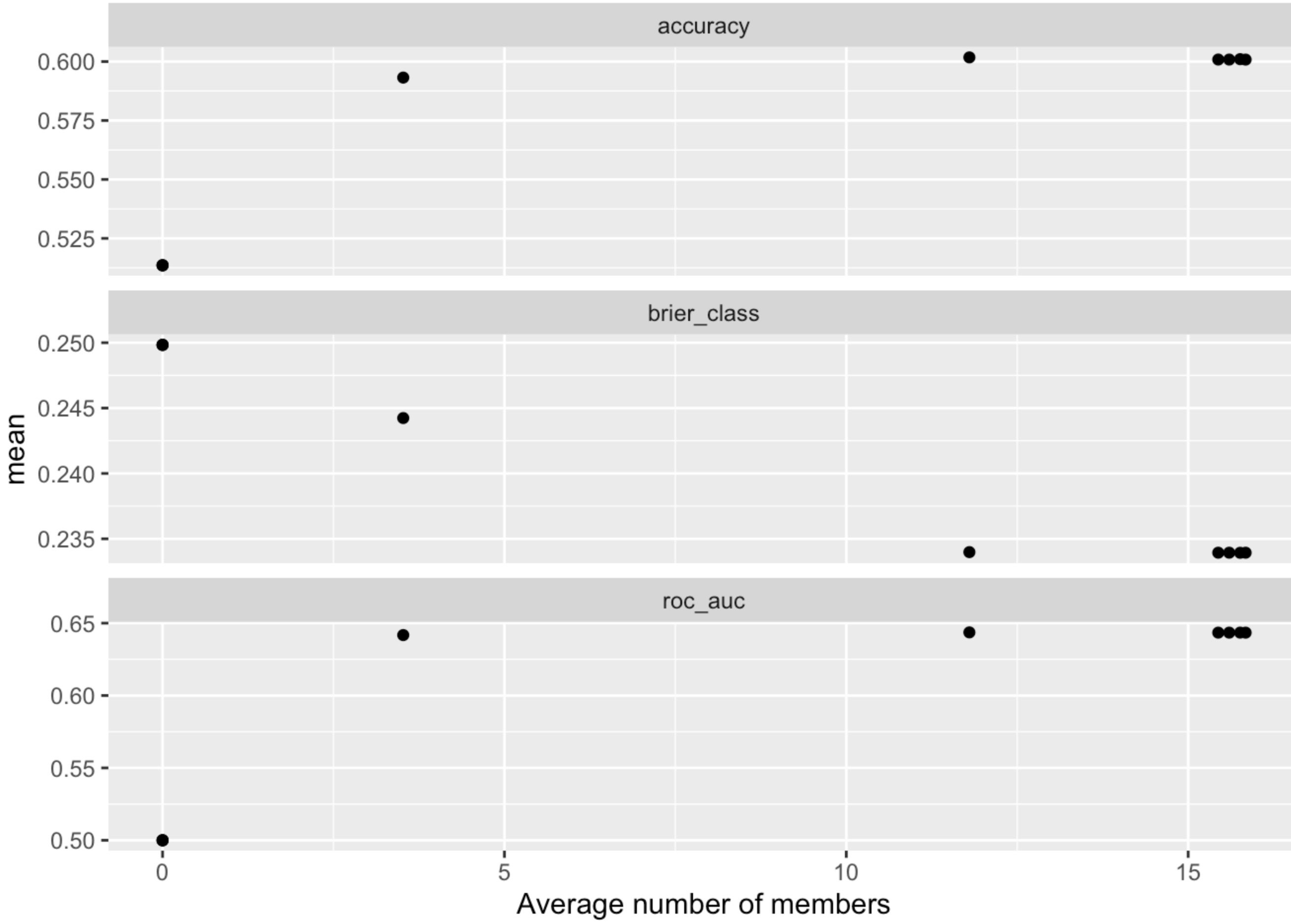
```
── A stacked ensemble model ─────────────────────────────────────


Out of 202 possible candidate members, the ensemble retained 9.

Penalty: 0.01.

Mixture: 1.


The 9 highest weighted member classes are:

# A tibble: 9 × 3
  member                   type          weight
  <chr>                    <chr>          <dbl>
1 .pred_TRUE_xgb_tune_1_39 boost_tree   1.20
2 .pred_TRUE_rf_tune_1_24  rand_forest  1.05
3 .pred_TRUE_rf_tune_1_25  rand_forest  0.998
4 .pred_TRUE_rf_tune_1_23  rand_forest  0.771
5 .pred_TRUE_rf_tune_1_18  rand_forest  0.698
6 .pred_TRUE_rf_tune_1_19  rand_forest  0.531
7 .pred_TRUE_rf_tune_1_20  rand_forest  0.136
8 .pred_TRUE_rf_tune_1_21  rand_forest  0.102
9 .pred_TRUE_rf_tune_1_16  rand_forest  0.00623
```

```r
autoplot(icu_stack)
```



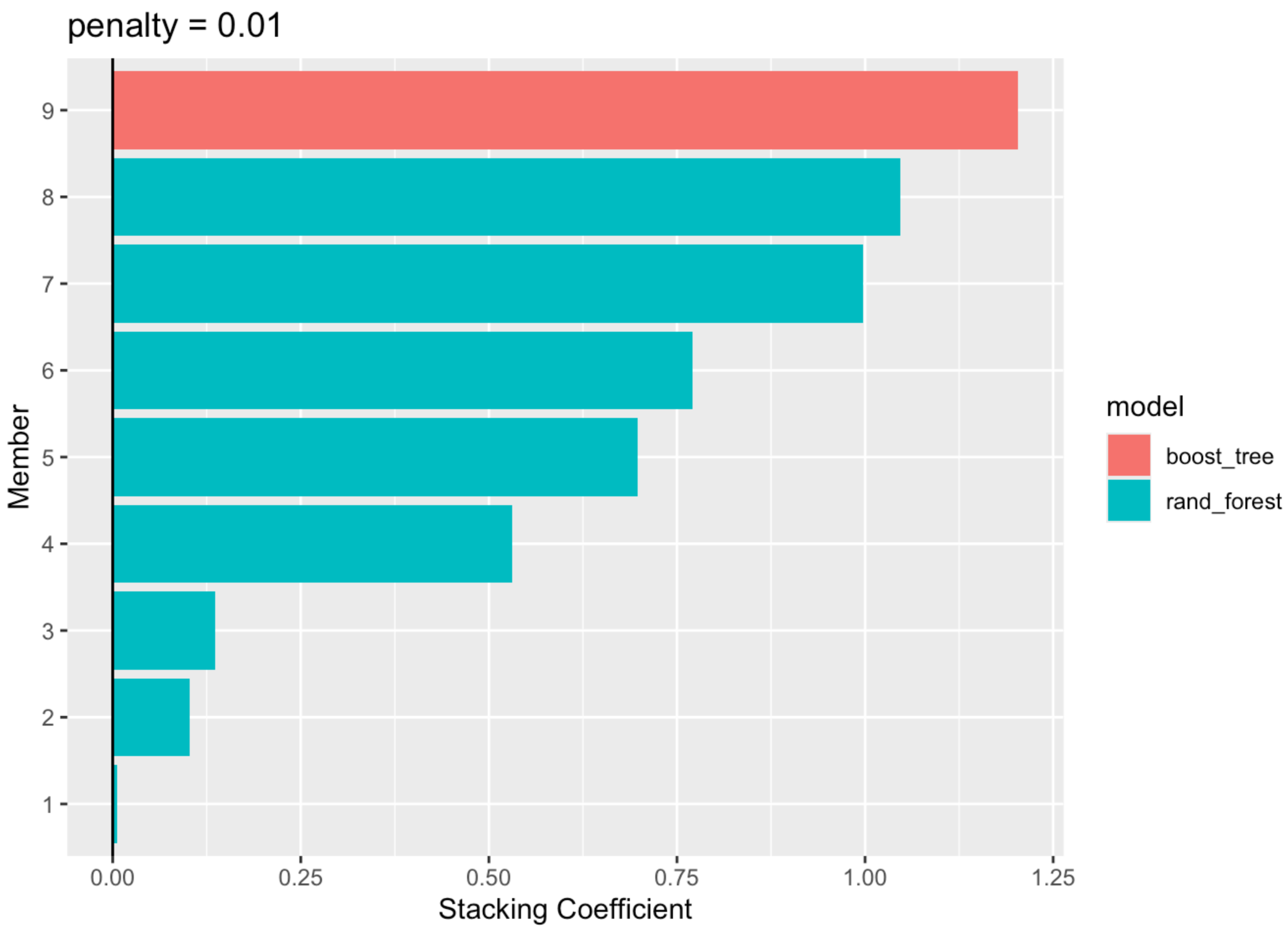
Looking at this picture, the accuracy of 0.6 occurred at the lowest penalty of where the dotted line is at; as the penalty increased, the accuracy decreased as well. The brier_class increased while the penalty increased as well, which is not what we want since a lower brier_class is better. The ROC_AUC was the highest at the location of the dotted line (1e-02 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-02 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 boost_tree had the highest weight in the model and that the stacked model had a better time utilizing random forest compared to random forest or Log. Reg.. Surprisingly, Log. Reg is not 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 1.05
 2 rf_tune_1_25     5   500 .pred_TRUE_rf_tune_1_25 0.998
 3 rf_tune_1_23     5   300 .pred_TRUE_rf_tune_1_23 0.771
 4 rf_tune_1_18     4   300 .pred_TRUE_rf_tune_1_18 0.698
 5 rf_tune_1_19     4   400 .pred_TRUE_rf_tune_1_19 0.531
 6 rf_tune_1_20     4   500 .pred_TRUE_rf_tune_1_20 0.136
 7 rf_tune_1_21     5   100 .pred_TRUE_rf_tune_1_21 0.102
 8 rf_tune_1_16     4   100 .pred_TRUE_rf_tune_1_16 0.00623
 9 rf_tune_1_01     1   100 .pred_TRUE_rf_tune_1_01 0
10 rf_tune_1_02     1   200 .pred_TRUE_rf_tune_1_02 0
11 rf_tune_1_03     1   300 .pred_TRUE_rf_tune_1_03 0
```

```
12 rf_tune_1_04      1   400 .pred_TRUE_rf_tune_1_04 0
13 rf_tune_1_05      1   500 .pred_TRUE_rf_tune_1_05 0
14 rf_tune_1_06      2   100 .pred_TRUE_rf_tune_1_06 0
15 rf_tune_1_07      2   200 .pred_TRUE_rf_tune_1_07 0
16 rf_tune_1_08      2   300 .pred_TRUE_rf_tune_1_08 0
17 rf_tune_1_09      2   400 .pred_TRUE_rf_tune_1_09 0
18 rf_tune_1_10      2   500 .pred_TRUE_rf_tune_1_10 0
19 rf_tune_1_11      3   100 .pred_TRUE_rf_tune_1_11 0
20 rf_tune_1_12      3   200 .pred_TRUE_rf_tune_1_12 0
21 rf_tune_1_13      3   300 .pred_TRUE_rf_tune_1_13 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_17      4   200 .pred_TRUE_rf_tune_1_17 0
25 rf_tune_1_22      5   200 .pred_TRUE_rf_tune_1_22 0
```

The most important members from the random_forest are showed above. Specifically, the first three had the highest coef. indicating a higher weight/influence on the final stacked model prediction. It looks like when the mtry was 5 and the trees were 400, the coefficient was highest (1.05)

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

```
# A tibble: 138 × 5
    member             penalty mixture terms                        coef
    <chr>                <dbl>   <dbl> <chr>                       <dbl>
  1 logit_tune_1_001  0.000001     0   .pred_TRUE_logit_tune_1_001     0
  2 logit_tune_1_042  0.00534      0   .pred_TRUE_logit_tune_1_042     0
  3 logit_tune_1_043  0.00658      0   .pred_TRUE_logit_tune_1_043     0
  4 logit_tune_1_044  0.00811      0   .pred_TRUE_logit_tune_1_044     0
  5 logit_tune_1_045  0.01         0   .pred_TRUE_logit_tune_1_045     0
  6 logit_tune_1_046  0.0123       0   .pred_TRUE_logit_tune_1_046     0
  7 logit_tune_1_047  0.0152       0   .pred_TRUE_logit_tune_1_047     0
  8 logit_tune_1_048  0.0187       0   .pred_TRUE_logit_tune_1_048     0
  9 logit_tune_1_049  0.0231       0   .pred_TRUE_logit_tune_1_049     0
 10 logit_tune_1_050  0.0285       0   .pred_TRUE_logit_tune_1_050     0
 11 logit_tune_1_051  0.0351       0   .pred_TRUE_logit_tune_1_051     0
 12 logit_tune_1_052  0.0433       0   .pred_TRUE_logit_tune_1_052     0
 13 logit_tune_1_053  0.0534       0   .pred_TRUE_logit_tune_1_053     0
 14 logit_tune_1_054  0.0658       0   .pred_TRUE_logit_tune_1_054     0
 15 logit_tune_1_055  0.0811       0   .pred_TRUE_logit_tune_1_055     0
 16 logit_tune_1_056  0.1          0   .pred_TRUE_logit_tune_1_056     0
 17 logit_tune_1_057  0.123        0   .pred_TRUE_logit_tune_1_057     0
 18 logit_tune_1_058  0.152        0   .pred_TRUE_logit_tune_1_058     0
 19 logit_tune_1_059  0.187        0   .pred_TRUE_logit_tune_1_059     0
 20 logit_tune_1_060  0.231        0   .pred_TRUE_logit_tune_1_060     0
 21 logit_tune_1_061  0.285        0   .pred_TRUE_logit_tune_1_061     0
 22 logit_tune_1_062  0.351        0   .pred_TRUE_logit_tune_1_062     0
 23 logit_tune_1_063  0.433        0   .pred_TRUE_logit_tune_1_063     0
 24 logit_tune_1_064  0.534        0   .pred_TRUE_logit_tune_1_064     0
 25 logit_tune_1_065  0.658        0   .pred_TRUE_logit_tune_1_065     0
```

```
26 logit_tune_1_066  0.811       0    .pred_TRUE_logit_tune_1_066     0
27 logit_tune_1_067  1           0    .pred_TRUE_logit_tune_1_067     0
28 logit_tune_1_068  1.23        0    .pred_TRUE_logit_tune_1_068     0
29 logit_tune_1_069  1.52        0    .pred_TRUE_logit_tune_1_069     0
30 logit_tune_1_070  1.87        0    .pred_TRUE_logit_tune_1_070     0
31 logit_tune_1_071  2.31        0    .pred_TRUE_logit_tune_1_071     0
32 logit_tune_1_072  2.85        0    .pred_TRUE_logit_tune_1_072     0
33 logit_tune_1_073  3.51        0    .pred_TRUE_logit_tune_1_073     0
34 logit_tune_1_074  4.33        0    .pred_TRUE_logit_tune_1_074     0
35 logit_tune_1_075  5.34        0    .pred_TRUE_logit_tune_1_075     0
36 logit_tune_1_076  6.58        0    .pred_TRUE_logit_tune_1_076     0
37 logit_tune_1_077  8.11        0    .pred_TRUE_logit_tune_1_077     0
38 logit_tune_1_078 10           0    .pred_TRUE_logit_tune_1_078     0
39 logit_tune_1_079 12.3         0    .pred_TRUE_logit_tune_1_079     0
40 logit_tune_1_080 15.2         0    .pred_TRUE_logit_tune_1_080     0
41 logit_tune_1_081 18.7         0    .pred_TRUE_logit_tune_1_081     0
42 logit_tune_1_082 23.1         0    .pred_TRUE_logit_tune_1_082     0
43 logit_tune_1_083 28.5         0    .pred_TRUE_logit_tune_1_083     0
44 logit_tune_1_084 35.1         0    .pred_TRUE_logit_tune_1_084     0
45 logit_tune_1_101  0.000001    0.25 .pred_TRUE_logit_tune_1_101     0
46 logit_tune_1_136  0.00152     0.25 .pred_TRUE_logit_tune_1_136     0
47 logit_tune_1_137  0.00187     0.25 .pred_TRUE_logit_tune_1_137     0
48 logit_tune_1_138  0.00231     0.25 .pred_TRUE_logit_tune_1_138     0
49 logit_tune_1_139  0.00285     0.25 .pred_TRUE_logit_tune_1_139     0
50 logit_tune_1_140  0.00351     0.25 .pred_TRUE_logit_tune_1_140     0
51 logit_tune_1_141  0.00433     0.25 .pred_TRUE_logit_tune_1_141     0
52 logit_tune_1_142  0.00534     0.25 .pred_TRUE_logit_tune_1_142     0
53 logit_tune_1_143  0.00658     0.25 .pred_TRUE_logit_tune_1_143     0
54 logit_tune_1_144  0.00811     0.25 .pred_TRUE_logit_tune_1_144     0
55 logit_tune_1_145  0.01        0.25 .pred_TRUE_logit_tune_1_145     0
56 logit_tune_1_146  0.0123      0.25 .pred_TRUE_logit_tune_1_146     0
57 logit_tune_1_147  0.0152      0.25 .pred_TRUE_logit_tune_1_147     0
58 logit_tune_1_148  0.0187      0.25 .pred_TRUE_logit_tune_1_148     0
59 logit_tune_1_149  0.0231      0.25 .pred_TRUE_logit_tune_1_149     0
60 logit_tune_1_150  0.0285      0.25 .pred_TRUE_logit_tune_1_150     0
61 logit_tune_1_151  0.0351      0.25 .pred_TRUE_logit_tune_1_151     0
62 logit_tune_1_152  0.0433      0.25 .pred_TRUE_logit_tune_1_152     0
63 logit_tune_1_153  0.0534      0.25 .pred_TRUE_logit_tune_1_153     0
64 logit_tune_1_154  0.0658      0.25 .pred_TRUE_logit_tune_1_154     0
65 logit_tune_1_155  0.0811      0.25 .pred_TRUE_logit_tune_1_155     0
66 logit_tune_1_156  0.1         0.25 .pred_TRUE_logit_tune_1_156     0
67 logit_tune_1_157  0.123       0.25 .pred_TRUE_logit_tune_1_157     0
68 logit_tune_1_158  0.152       0.25 .pred_TRUE_logit_tune_1_158     0
69 logit_tune_1_201  0.000001    0.5  .pred_TRUE_logit_tune_1_201     0
70 logit_tune_1_233  0.000811    0.5  .pred_TRUE_logit_tune_1_233     0
71 logit_tune_1_234  0.001       0.5  .pred_TRUE_logit_tune_1_234     0
72 logit_tune_1_235  0.00123     0.5  .pred_TRUE_logit_tune_1_235     0
73 logit_tune_1_236  0.00152     0.5  .pred_TRUE_logit_tune_1_236     0
74 logit_tune_1_237  0.00187     0.5  .pred_TRUE_logit_tune_1_237     0
75 logit_tune_1_238  0.00231     0.5  .pred_TRUE_logit_tune_1_238     0
76 logit_tune_1_239  0.00285     0.5  .pred_TRUE_logit_tune_1_239     0
```

```
 77 logit_tune_1_240  0.00351     0.5  .pred_TRUE_logit_tune_1_240     0
 78 logit_tune_1_241  0.00433     0.5  .pred_TRUE_logit_tune_1_241     0
 79 logit_tune_1_242  0.00534     0.5  .pred_TRUE_logit_tune_1_242     0
 80 logit_tune_1_243  0.00658     0.5  .pred_TRUE_logit_tune_1_243     0
 81 logit_tune_1_244  0.00811     0.5  .pred_TRUE_logit_tune_1_244     0
 82 logit_tune_1_245  0.01        0.5  .pred_TRUE_logit_tune_1_245     0
 83 logit_tune_1_246  0.0123      0.5  .pred_TRUE_logit_tune_1_246     0
 84 logit_tune_1_247  0.0152      0.5  .pred_TRUE_logit_tune_1_247     0
 85 logit_tune_1_248  0.0187      0.5  .pred_TRUE_logit_tune_1_248     0
 86 logit_tune_1_249  0.0231      0.5  .pred_TRUE_logit_tune_1_249     0
 87 logit_tune_1_250  0.0285      0.5  .pred_TRUE_logit_tune_1_250     0
 88 logit_tune_1_251  0.0351      0.5  .pred_TRUE_logit_tune_1_251     0
 89 logit_tune_1_252  0.0433      0.5  .pred_TRUE_logit_tune_1_252     0
 90 logit_tune_1_253  0.0534      0.5  .pred_TRUE_logit_tune_1_253     0
 91 logit_tune_1_254  0.0658      0.5  .pred_TRUE_logit_tune_1_254     0
 92 logit_tune_1_301  0.000001    0.75 .pred_TRUE_logit_tune_1_301     0
 93 logit_tune_1_331  0.000534    0.75 .pred_TRUE_logit_tune_1_331     0
 94 logit_tune_1_332  0.000658    0.75 .pred_TRUE_logit_tune_1_332     0
 95 logit_tune_1_333  0.000811    0.75 .pred_TRUE_logit_tune_1_333     0
 96 logit_tune_1_334  0.001       0.75 .pred_TRUE_logit_tune_1_334     0
 97 logit_tune_1_335  0.00123     0.75 .pred_TRUE_logit_tune_1_335     0
 98 logit_tune_1_336  0.00152     0.75 .pred_TRUE_logit_tune_1_336     0
 99 logit_tune_1_337  0.00187     0.75 .pred_TRUE_logit_tune_1_337     0
100 logit_tune_1_338  0.00231     0.75 .pred_TRUE_logit_tune_1_338     0
101 logit_tune_1_339  0.00285     0.75 .pred_TRUE_logit_tune_1_339     0
102 logit_tune_1_340  0.00351     0.75 .pred_TRUE_logit_tune_1_340     0
103 logit_tune_1_341  0.00433     0.75 .pred_TRUE_logit_tune_1_341     0
104 logit_tune_1_342  0.00534     0.75 .pred_TRUE_logit_tune_1_342     0
105 logit_tune_1_343  0.00658     0.75 .pred_TRUE_logit_tune_1_343     0
106 logit_tune_1_344  0.00811     0.75 .pred_TRUE_logit_tune_1_344     0
107 logit_tune_1_345  0.01        0.75 .pred_TRUE_logit_tune_1_345     0
108 logit_tune_1_346  0.0123      0.75 .pred_TRUE_logit_tune_1_346     0
109 logit_tune_1_347  0.0152      0.75 .pred_TRUE_logit_tune_1_347     0
110 logit_tune_1_348  0.0187      0.75 .pred_TRUE_logit_tune_1_348     0
111 logit_tune_1_349  0.0231      0.75 .pred_TRUE_logit_tune_1_349     0
112 logit_tune_1_350  0.0285      0.75 .pred_TRUE_logit_tune_1_350     0
113 logit_tune_1_351  0.0351      0.75 .pred_TRUE_logit_tune_1_351     0
114 logit_tune_1_352  0.0433      0.75 .pred_TRUE_logit_tune_1_352     0
115 logit_tune_1_401  0.000001    1    .pred_TRUE_logit_tune_1_401     0
116 logit_tune_1_430  0.000433    1    .pred_TRUE_logit_tune_1_430     0
117 logit_tune_1_431  0.000534    1    .pred_TRUE_logit_tune_1_431     0
118 logit_tune_1_432  0.000658    1    .pred_TRUE_logit_tune_1_432     0
119 logit_tune_1_433  0.000811    1    .pred_TRUE_logit_tune_1_433     0
120 logit_tune_1_434  0.001       1    .pred_TRUE_logit_tune_1_434     0
121 logit_tune_1_435  0.00123     1    .pred_TRUE_logit_tune_1_435     0
122 logit_tune_1_436  0.00152     1    .pred_TRUE_logit_tune_1_436     0
123 logit_tune_1_437  0.00187     1    .pred_TRUE_logit_tune_1_437     0
124 logit_tune_1_438  0.00231     1    .pred_TRUE_logit_tune_1_438     0
125 logit_tune_1_439  0.00285     1    .pred_TRUE_logit_tune_1_439     0
126 logit_tune_1_440  0.00351     1    .pred_TRUE_logit_tune_1_440     0
127 logit_tune_1_441  0.00433     1    .pred_TRUE_logit_tune_1_441     0
```

```
128 logit_tune_1_442  0.00534     1    .pred_TRUE_logit_tune_1_442     0
129 logit_tune_1_443  0.00658     1    .pred_TRUE_logit_tune_1_443     0
130 logit_tune_1_444  0.00811     1    .pred_TRUE_logit_tune_1_444     0
131 logit_tune_1_445  0.01        1    .pred_TRUE_logit_tune_1_445     0
132 logit_tune_1_446  0.0123      1    .pred_TRUE_logit_tune_1_446     0
133 logit_tune_1_447  0.0152      1    .pred_TRUE_logit_tune_1_447     0
134 logit_tune_1_448  0.0187      1    .pred_TRUE_logit_tune_1_448     0
135 logit_tune_1_449  0.0231      1    .pred_TRUE_logit_tune_1_449     0
136 logit_tune_1_450  0.0285      1    .pred_TRUE_logit_tune_1_450     0
137 logit_tune_1_451  0.0351      1    .pred_TRUE_logit_tune_1_451     0
138 logit_tune_1_452  0.0433      1    .pred_TRUE_logit_tune_1_452     0
```

This shows that the logit_tune contributed not much to the model or anything since all the coefficients were zero.

```
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  1.20
 2 xgb_tune_1_01   100          1   0.00001  .pred_TRUE_xgb_tune_1_01  0
 3 xgb_tune_1_02   300          1   0.00001  .pred_TRUE_xgb_tune_1_02  0
 4 xgb_tune_1_03   500          1   0.00001  .pred_TRUE_xgb_tune_1_03  0
 5 xgb_tune_1_04   100          1   0.000562 .pred_TRUE_xgb_tune_1_04  0
 6 xgb_tune_1_05   300          1   0.000562 .pred_TRUE_xgb_tune_1_05  0
 7 xgb_tune_1_06   500          1   0.000562 .pred_TRUE_xgb_tune_1_06  0
 8 xgb_tune_1_07   100          1   0.0316   .pred_TRUE_xgb_tune_1_07  0
 9 xgb_tune_1_08   300          1   0.0316   .pred_TRUE_xgb_tune_1_08  0
10 xgb_tune_1_09   500          1   0.0316   .pred_TRUE_xgb_tune_1_09  0
11 xgb_tune_1_10   100          1   1.78     .pred_TRUE_xgb_tune_1_10  0
12 xgb_tune_1_11   300          1   1.78     .pred_TRUE_xgb_tune_1_11  0
13 xgb_tune_1_12   500          1   1.78     .pred_TRUE_xgb_tune_1_12  0
14 xgb_tune_1_13   100          1 100        .pred_TRUE_xgb_tune_1_13  0
15 xgb_tune_1_16   100          2   0.00001  .pred_TRUE_xgb_tune_1_16  0
16 xgb_tune_1_17   300          2   0.00001  .pred_TRUE_xgb_tune_1_17  0
17 xgb_tune_1_18   500          2   0.00001  .pred_TRUE_xgb_tune_1_18  0
18 xgb_tune_1_19   100          2   0.000562 .pred_TRUE_xgb_tune_1_19  0
19 xgb_tune_1_20   300          2   0.000562 .pred_TRUE_xgb_tune_1_20  0
20 xgb_tune_1_21   500          2   0.000562 .pred_TRUE_xgb_tune_1_21  0
21 xgb_tune_1_22   100          2   0.0316   .pred_TRUE_xgb_tune_1_22  0
22 xgb_tune_1_23   300          2   0.0316   .pred_TRUE_xgb_tune_1_23  0
23 xgb_tune_1_24   500          2   0.0316   .pred_TRUE_xgb_tune_1_24  0
24 xgb_tune_1_25   100          2   1.78     .pred_TRUE_xgb_tune_1_25  0
25 xgb_tune_1_26   300          2   1.78     .pred_TRUE_xgb_tune_1_26  0
26 xgb_tune_1_27   500          2   1.78     .pred_TRUE_xgb_tune_1_27  0
27 xgb_tune_1_28   100          2 100        .pred_TRUE_xgb_tune_1_28  0
28 xgb_tune_1_31   100          3   0.00001  .pred_TRUE_xgb_tune_1_31  0
29 xgb_tune_1_32   300          3   0.00001  .pred_TRUE_xgb_tune_1_32  0
```

```
30 xgb_tune_1_33   500          3   0.00001  .pred_TRUE_xgb_tune_1_33  0
31 xgb_tune_1_34   100          3   0.000562 .pred_TRUE_xgb_tune_1_34  0
32 xgb_tune_1_35   300          3   0.000562 .pred_TRUE_xgb_tune_1_35  0
33 xgb_tune_1_36   500          3   0.000562 .pred_TRUE_xgb_tune_1_36  0
34 xgb_tune_1_37   100          3   0.0316   .pred_TRUE_xgb_tune_1_37  0
35 xgb_tune_1_38   300          3   0.0316   .pred_TRUE_xgb_tune_1_38  0
36 xgb_tune_1_40   100          3   1.78     .pred_TRUE_xgb_tune_1_40  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 once to the stacked model, more than the Log. Reg., but less than the random_forest. The hihghest coefficient was seen at tree 500 with a tree_depth of 3. The coefficient was 1.204

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

```
# A tibble: 37,571 × 24
   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
   Glucose .pred_FALSE .pred_TRUE
     <dbl>       <dbl>      <dbl>
 1     102       0.458      0.542
 2     112       0.494      0.506
 3      87       0.673      0.327
 4     131       0.563      0.437
 5     141       0.540      0.460
 6     288       0.556      0.444
 7     113       0.390      0.610
 8     173       0.503      0.497
 9     269       0.575      0.425
10      58       0.572      0.428
# ℹ 37,561 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,
  event_level = "second"
)
```

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.648
```

This ROC_AUC of the stacked model is 0.647. This means that this is better than random guessing (which would be 50%). This is also the best models

```
icu_pred <-
  icu_test |>
```

```
  select(los_long) |>
  bind_cols(
    predict(
      icu_stack,
      icu_test,
      type = "class",
      members = TRUE
    )
  ) |>
  print(width = Inf)
```

```
# A tibble: 37,571 × 11
   los_long .pred_class .pred_class_rf_tune_1_16 .pred_class_rf_tune_1_18
   <fct>    <fct>       <fct>                    <fct>
 1 FALSE    TRUE        TRUE                     TRUE
 2 FALSE    TRUE        TRUE                     FALSE
 3 FALSE    FALSE       FALSE                    FALSE
 4 FALSE    FALSE       FALSE                    FALSE
 5 TRUE     FALSE       FALSE                    FALSE
 6 FALSE    FALSE       FALSE                    FALSE
 7 TRUE     TRUE        TRUE                     TRUE
 8 TRUE     FALSE       FALSE                    TRUE
 9 TRUE     FALSE       FALSE                    FALSE
10 FALSE    FALSE       TRUE                     FALSE
   .pred_class_rf_tune_1_19 .pred_class_rf_tune_1_20 .pred_class_rf_tune_1_21
   <fct>                    <fct>                    <fct>
 1 TRUE                     TRUE                     TRUE
 2 TRUE                     TRUE                     TRUE
 3 FALSE                    FALSE                    FALSE
 4 FALSE                    FALSE                    FALSE
 5 FALSE                    FALSE                    TRUE
 6 FALSE                    FALSE                    FALSE
 7 TRUE                     TRUE                     TRUE
 8 FALSE                    FALSE                    FALSE
 9 FALSE                    FALSE                    FALSE
10 FALSE                    FALSE                    FALSE
   .pred_class_rf_tune_1_23 .pred_class_rf_tune_1_24 .pred_class_rf_tune_1_25
   <fct>                    <fct>                    <fct>
 1 TRUE                     TRUE                     TRUE
 2 TRUE                     TRUE                     TRUE
 3 FALSE                    FALSE                    FALSE
 4 FALSE                    FALSE                    FALSE
 5 FALSE                    FALSE                    FALSE
 6 FALSE                    FALSE                    FALSE
 7 TRUE                     TRUE                     TRUE
 8 FALSE                    FALSE                    TRUE
 9 FALSE                    FALSE                    FALSE
10 FALSE                    FALSE                    FALSE
   .pred_class_xgb_tune_1_39
   <fct>
```

```
 1 TRUE
 2 FALSE
 3 FALSE
 4 FALSE
 5 FALSE
 6 FALSE
 7 TRUE
 8 TRUE
 9 FALSE
10 FALSE
# ℹ 37,561 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: 10 × 2
   model                       accuracy
   <chr>                          <dbl>
 1 .pred_class                    0.606
 2 .pred_class_rf_tune_1_16       0.600
 3 .pred_class_rf_tune_1_18       0.605
 4 .pred_class_rf_tune_1_19       0.605
 5 .pred_class_rf_tune_1_20       0.604
 6 .pred_class_rf_tune_1_21       0.598
 7 .pred_class_rf_tune_1_23       0.605
 8 .pred_class_rf_tune_1_24       0.606
 9 .pred_class_rf_tune_1_25       0.606
10 .pred_class_xgb_tune_1_39      0.601
```

Looking at the .pred_class, the accuracy of the model is 0.6064. 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 0.0001. 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 a high ROC AUC and accuracy (close to that of the stacked model) 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. 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 0.0001.

For the question: What are the most important features in predicting long ICU stays? Please see the other establishments from above. For example, I would say that one of the most important factors would include SysBP, however, there are others as well as mentioned before. Although none of them had very much common ones all around, it looks like SysBP, age_intime, heart_rate, WBC, Hematocrit, and Respiratory_Rate, were some of the most important features in determining whether or not the los_long was greater than or equal to two days.

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 as a result. However, interpretability is based on the individual's understanding on the models themselves. I think that a logistic regression is so commonly known about, it makes it easier to understand what it is stating. I think with more information and dissection of the components of all the models, that it could easily be up to the individual to decide which one is the most interpretable. To me, however, I think the logistic regression is the easiest to understand given all the information, but if its based on accuracy and predictability, then the stacked model would win because it had the highest out of the four. Out of convenience and efficacy, the random forest would win. In other words, it is up to the goal of the individual to figure out which one works best depending on the situation being examined/questioned