Biostat 203B Homework 5

Due Mar 22 @ 11:59PM

AUTHOR
Zongzhe Lin UID:206328707

Predicting ICU duration

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

```
# A vector of required packages
required_packages <- c("dplyr", "caret", "glmnet", "pROC",
                       "rsample", "randomForest", "doParallel", "gbm")

# Loop through each package to check if it is installed; install it if not
for (pkg in required_packages) {
  if (!require(pkg, character.only = TRUE, quietly = TRUE)) {
    install.packages(pkg)
    library(pkg, character.only = TRUE)
  }
}
```

```
载入程辑包：'dplyr'

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

    filter, lag

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

    intersect, setdiff, setequal, union

Warning: 程辑包'ggplot2'是用R版本4.3.3 来建造的

Warning: 程辑包'lattice'是用R版本4.3.3 来建造的

Warning: 程辑包'Matrix'是用R版本4.3.3 来建造的

Loaded glmnet 4.1-8

Type 'citation("pROC")' for a citation.
```

```
载入程辑包：'pROC'

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

    cov, smooth, var

Warning: 程辑包'rsample'是用R版本4.3.3 来建造的

randomForest 4.7-1.1

Type rfNews() to see new features/changes/bug fixes.


载入程辑包：'randomForest'

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

    margin

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

    combine

Warning: 程辑包'doParallel'是用R版本4.3.3 来建造的

Loaded gbm 2.1.9

This version of gbm is no longer under development. Consider transitioning to gbm3,
https://github.com/gbm-developers/gbm3
```

```r
# Load the libraries
library(dplyr)
library(caret)
library(glmnet)
library(pROC)
library(rsample)
library(randomForest)
library(doParallel)
library(gbm)
```

1. Data preprocessing and feature engineering.

```r
# Load the dataset
mimiciv_icu_cohort <- readRDS("mimic_icu_cohort.rds")
```

```r
# Access the actual data frame
mimiciv_icu_cohort <- mimiciv_icu_cohort$inputs$data

# Select specific columns
mimiciv_icu_cohort <- mimiciv_icu_cohort[, c(
  "gender", "anchor_age", "marital_status", "race",
  "first_careunit", "hematocrit", "chloride",
```

```
  "sodium", "glucose", "bicarbonate", "white_blood_cell_count",
  "potassium", "heart_rate", "systolic_bp", "diastolic_bp",
  "temperature_f", "respiratory_rate", "subject_id", "hadm_id",
  "stay_id", "los_long"
)]
```

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

```
set.seed(203)

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

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

training_set <- training(data_split)
testing_set <- testing(data_split)

# Remove NA values
training_set <- na.omit(training_set)
testing_set <- na.omit(testing_set)

# Convert outcome to a binary factor
training_set$los_long <- factor(
  training_set$los_long,
  levels = c(FALSE, TRUE),
  labels = c("ShortStay", "LongStay"))

testing_set$los_long <- factor(
  testing_set$los_long, levels = c(FALSE, TRUE),
  labels = c("ShortStay", "LongStay"))
```

3. Train and tune the models using the training set.

Logistic Regression with Elastic Net Regularization (Enet):

```
# Set up cross-validation for model tuning
set.seed(203)
train_control <- trainControl(method = "cv", number = 10,
                              classProbs = TRUE,
                              summaryFunction = twoClassSummary)

# Train the Elastic Net model
enet_model <- train(los_long ~ .,
                    data = training_set,
                    method = "glmnet",
```

```
                    trControl = train_control,
                    preProcess = c("center", "scale", "medianImpute"),
                    tuneLength = 5,
                    metric = "ROC")
```

```
# Predictions on the test set
testing_predictions <- predict(enet_model, newdata=testing_set, type="raw")

# Ensure that the actual outcomes are a factor with the correct levels
testing_set$los_long <- factor(testing_set$los_long,
                               levels=levels(training_set$los_long))

# Ensure predictions are a factor with the same levels as actual outcomes
testing_predictions <- factor(testing_predictions,
                              levels=levels(testing_set$los_long))

# Use the confusionMatrix function
confusion_matrix <- confusionMatrix(data=testing_predictions,
                                    reference=testing_set$los_long)
print(confusion_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  ShortStay LongStay
  ShortStay     10316     7645
  LongStay       5060     6472

               Accuracy : 0.5692
                 95% CI : (0.5635, 0.5749)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1303

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6709
            Specificity : 0.4585
         Pos Pred Value : 0.5744
         Neg Pred Value : 0.5612
             Prevalence : 0.5213
         Detection Rate : 0.3498
   Detection Prevalence : 0.6090
      Balanced Accuracy : 0.5647

       'Positive' Class : ShortStay
```

```
# ROC and AUC on the test set
testing_probs <- predict(enet_model, newdata=testing_set, type="prob")
```

```
ROC_result_testing <- roc(response=testing_set$los_long,
                          predictor=testing_probs$LongStay)
```

```
Setting levels: control = ShortStay, case = LongStay

Setting direction: controls < cases
```

```
auc_value <- auc(ROC_result_testing)
print(auc_value)
```

```
Area under the curve: 0.5927
```

```
# Check the variable importance
variable_importance <- varImp(enet_model, scale=FALSE)
print(variable_importance)
```

```
glmnet variable importance

  only 20 most important variables shown (out of 28)

                                                           Overall
first_careunitMedical/Surgical Intensive Care Unit (MICU/SICU) 0.19488
first_careunitMedical Intensive Care Unit (MICU)               0.19326
hematocrit                                                     0.18024
chloride                                                       0.15760
respiratory_rate                                               0.14120
sodium                                                         0.13952
heart_rate                                                     0.12783
white_blood_cell_count                                         0.11017
anchor_age                                                     0.10442
temperature_f                                                  0.08699
diastolic_bp                                                   0.07502
potassium                                                      0.04906
bicarbonate                                                    0.04028
marital_statusWIDOWED                                          0.03640
marital_statusSINGLE                                           0.03582
first_careunitSurgical Intensive Care Unit (SICU)              0.03543
marital_statusMARRIED                                          0.03040
genderM                                                        0.02110
raceBLACK                                                      0.01734
hadm_id                                                        0.01424
```

Elastic Net Model Evaluation: The Elastic Net model exhibits modest predictive power, with an AUC of 0.5927—suggesting it slightly surpasses random chance in distinguishing between long and short ICU stays. Its accuracy hovers at 56.92%, only marginally better than the baseline no-information rate. This indicates that while the model does offer some predictive insights, its performance is limited.

Key Predictive Variables: The model identifies the initial care unit type—specifically, Medical/Surgical and Medical ICUs—as top predictors, alongside physiological measurements like hematocrit and respiratory rate. These variables seem to be the most significant in discerning the length of ICU stays.

Model Comparison and Interpretability: its strength lies in interpretability; the Elastic Net approach aids in pinpointing influential predictors by balancing feature selection and regularization, which is valuable for

understanding model decisions.

Random Forest:

```
# Register the parallel backend to use multiple cores
registerDoParallel(cores = detectCores())
```

```
# Set the seed for reproducibility
set.seed(203)

# Create a tuning grid specifying 'mtry' values
mtry_values <- round(seq(2, sqrt(ncol(training_set)), by = 2))

tuning_grid <- expand.grid(
  mtry = mtry_values
)

# Create a control function to specify the cross-validation method
train_control <- trainControl(
  method = "cv",          # Use k-fold cross-validation
  number = 10,            # Number of folds
  search = "grid",        # Parameter search method
  allowParallel = TRUE    # Allow parallel processing
)

# Train the random forest model with the correct tuning grid
rf_model <- train(
  los_long ~ .,
  data = training_set,
  method = "rf",                # Specify random forest
  trControl = train_control,
  ntree = 1000,                 # Number of trees
  tuneGrid = tuning_grid        # Correct tuning grid
)
```

```
# Predict on the test set
rf_predictions <- predict(rf_model, newdata=testing_set)

# Confusion Matrix and Accuracy
# Ensure predictions are a factor with the same levels as testing set
rf_predictions <- factor(rf_predictions, levels=levels(testing_set$los_long))

# Use the confusionMatrix function
rf_confusion_matrix <- confusionMatrix(data=rf_predictions,
                                       reference=testing_set$los_long)
print(rf_confusion_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  ShortStay LongStay
  ShortStay      9964     6683
```

```
  LongStay       5412     7434

               Accuracy : 0.5899
                 95% CI : (0.5843, 0.5955)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1753

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6480
            Specificity : 0.5266
         Pos Pred Value : 0.5985
         Neg Pred Value : 0.5787
             Prevalence : 0.5213
         Detection Rate : 0.3378
   Detection Prevalence : 0.5644
      Balanced Accuracy : 0.5873

       'Positive' Class : ShortStay
```

```r
# Calculate AUC
rf_probs <- predict(rf_model, newdata=testing_set, type="prob")
ROC_result_rf <- roc(response=testing_set$los_long,
                     predictor=rf_probs[, "LongStay"])
```

```
Setting levels: control = ShortStay, case = LongStay

Setting direction: controls < cases
```

```r
rf_auc <- auc(ROC_result_rf)
print(rf_auc)
```

```
Area under the curve: 0.6275
```

```r
# Variable Importance
rf_importance <- varImp(rf_model, scale=FALSE)
print(rf_importance)
```

```
rf variable importance

  only 20 most important variables shown (out of 28)

                       Overall
hematocrit               984.2
hadm_id                  933.2
stay_id                  933.0
white_blood_cell_count   931.1
subject_id               927.2
glucose                  895.0
```

```
systolic_bp              871.8
heart_rate               869.7
anchor_age               851.6
diastolic_bp             839.0
temperature_f            827.0
respiratory_rate         738.4
potassium                726.1
bicarbonate              699.7
chloride                 692.0
sodium                   673.8
genderM                  146.2
marital_statusMARRIED    128.7
raceWHITE                122.0
marital_statusSINGLE     114.7
```

The Random Forest model demonstrates a moderate level of accuracy, correctly predicting extended ICU stays approximately 58.96% of the time. With an AUC of 0.6275, the model's discriminative capacity is decent, surpassing mere chance yet indicating potential for enhancement. A closer examination of the confusion matrix reveals that the model more frequently identifies true negatives than false negatives, which is consistent for true positives over false positives. Despite this, the prevalence of errors is noteworthy and underscores the necessity for model refinement. The Kappa statistic stands at 0.1747, a modest figure that implies only slight congruence beyond random chance in the model's predictions when compared with actual outcomes. This statistic serves as a reminder of the opportunity to further advance the model's predictive capabilities.

Gradient Boosting Machines (GBM) / Boosting:

```
# Set the seed for reproducibility
set.seed(203)

# Create a control function to specify the CV method
train_control <- trainControl(method = "cv",
                              number = 10,
                              verboseIter = FALSE,
                              returnResamp = "all",
                              classProbs = TRUE,
                              summaryFunction = twoClassSummary,
                              allowParallel = TRUE)

# Define the tuning grid for GBM
gbmGrid <- expand.grid(interaction.depth = c(1, 3, 5),
                       n.trees = (1:5) * 50,
                       shrinkage = c(0.01, 0.1),
                       n.minobsinnode = c(10, 20))

# Train the GBM model
gbm_model <- train(los_long ~ .,
                   data = training_set,
                   method = "gbm",
                   trControl = train_control,
                   verbose = FALSE,
                   tuneGrid = gbmGrid,
                   metric = "ROC")
```

```r
# Predict on the test set
gbm_predictions <- predict(gbm_model, newdata=testing_set, type="raw")

# Confusion Matrix
gbm_confusion_matrix <- confusionMatrix(data=gbm_predictions,
                                        reference=testing_set$los_long)
print(gbm_confusion_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  ShortStay LongStay
  ShortStay     10086     6819
  LongStay       5290     7298

               Accuracy : 0.5894
                 95% CI : (0.5838, 0.5951)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1737

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6560
            Specificity : 0.5170
         Pos Pred Value : 0.5966
         Neg Pred Value : 0.5798
             Prevalence : 0.5213
         Detection Rate : 0.3420
   Detection Prevalence : 0.5732
      Balanced Accuracy : 0.5865

       'Positive' Class : ShortStay
```

```r
# ROC and AUC on the test set
gbm_probs <- predict(gbm_model, newdata=testing_set, type="prob")
ROC_result_gbm <- roc(response=testing_set$los_long,
                      predictor=gbm_probs[, "LongStay"])
```

```
Setting levels: control = ShortStay, case = LongStay

Setting direction: controls < cases
```

```r
gbm_auc <- auc(ROC_result_gbm)
print(gbm_auc)
```

```
Area under the curve: 0.6237
```

```r
# Check variable importance
gbm_importance <- varImp(gbm_model, scale=FALSE)
```

```
print(gbm_importance)
```

```
gbm variable importance

  only 20 most important variables shown (out of 28)
```

	Overall
hematocrit	321.19
respiratory_rate	268.98
temperature_f	265.75
anchor_age	247.83
white_blood_cell_count	244.40
heart_rate	188.30
glucose	160.39
systolic_bp	150.93
sodium	142.13
stay_id	139.84
subject_id	138.52
bicarbonate	138.03
chloride	129.39
first_careunitMedical Intensive Care Unit (MICU)	120.08
diastolic_bp	116.75
first_careunitMedical/Surgical Intensive Care Unit (MICU/SICU)	114.66
hadm_id	104.09
potassium	80.52
first_careunitOther	30.70
marital_statusWIDOWED	19.39

The Gradient Boosting Machine (GBM) model showcases a moderately effective prediction capability for the duration of ICU stays, achieving an accuracy rate close to 58.94%. This indicates that nearly 59% of the model's predictions correspond accurately to actual outcomes. With an Area Under the Curve (AUC) of 0.6237, the model exhibits a reasonable proficiency in differentiating between longer and shorter stays in the ICU, suggesting its effectiveness surpasses mere random guessing, yet highlighting a considerable scope for refinement.

Analysis of the confusion matrix reveals a balanced classification efficacy, with a higher frequency of correct predictions (true positives and true negatives) over incorrect ones. However, the existence of errors points to the GBM model's limitations in fully grasping the intricate dynamics within the dataset.

The model's variable importance evaluation pinpoints hematocrit, respiratory rate, and temperature as primary indicators. These findings underscore the critical nature of these factors in determining ICU stay lengths, offering valuable insights for medical practitioners aiming to optimize patient management strategies from the onset of ICU admission.

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

Stacking:

```
# Create a data frame to hold the predictions from each model
pred_gbm <- predict(gbm_model, newdata=testing_set, type="prob")
```

```r
pred_rf <- predict(rf_model, newdata=testing_set, type="prob")
pred_enet <- predict(enet_model, newdata=testing_set, type="prob")

# Combine the predictions into a single data frame
combined_preds <- data.frame(
  gbm=pred_gbm[, "LongStay"],
  rf=pred_rf[, "LongStay"],
  enet=pred_enet[, "LongStay"]
)

# Train a meta-model using the combined predictions
stacking_model <- train(
  los_long~.,
  data=cbind(combined_preds, los_long=testing_set$los_long),
  method="glm",
  trControl=trainControl(method="cv", number=10),
  family="binomial"
)

# Print the summary of the stacking model
print(summary(stacking_model))
```

```
Call:
NULL

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.49834    0.07684 -32.514  < 2e-16 ***
gbm          1.64522    0.21331   7.713 1.23e-14 ***
rf           3.40344    0.25202  13.505  < 2e-16 ***
enet        -0.02151    0.19308  -0.111    0.911
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 40832  on 29492  degrees of freedom
Residual deviance: 39226  on 29489  degrees of freedom
AIC: 39234

Number of Fisher Scoring iterations: 4
```

```r
# Predict with the stacking model on the combined predictions
stacked_predictions <- predict(stacking_model, newdata=combined_preds)

# Evaluate the performance of the stacking model
confusionMatrix(data=stacked_predictions, reference=testing_set$los_long)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  ShortStay LongStay
```

```
  ShortStay       9850     6507
  LongStay        5526     7610

               Accuracy : 0.592
                 95% CI : (0.5864, 0.5976)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1802

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6406
            Specificity : 0.5391
         Pos Pred Value : 0.6022
         Neg Pred Value : 0.5793
             Prevalence : 0.5213
         Detection Rate : 0.3340
   Detection Prevalence : 0.5546
      Balanced Accuracy : 0.5898

       'Positive' Class : ShortStay
```

```r
# Predict the class probabilities with the stacking model
stacked_probs <- predict(stacking_model, newdata=combined_preds, type="prob")
ROC_result_stacked <- roc(response=testing_set$los_long,
                          predictor=stacked_probs$LongStay)
```

```
Setting levels: control = ShortStay, case = LongStay

Setting direction: controls < cases
```

```r
auc_stacked <- auc(ROC_result_stacked)
print(auc_stacked)
```

```
Area under the curve: 0.6303
```

Method	AUC	Accuracy
Log Regression	0.59	0.57
Random Forest	0.63	0.59
GBM	0.63	0.59
Stacking	0.63	0.59

Based on the given results, all the machine learning algorithms and the model stacking method appear to have similar performance in terms of accuracy, all hovering around 59%. The area under the ROC curve (AUC) for logistic regression is slightly lower at 0.59, while random forest, GBM (Gradient Boosting Machine), and stacking have a slightly higher AUC of 0.63.

In the context of predicting long ICU stays:

The similar AUC and accuracy values suggest that there isn't a clear advantage to using more complex models over the simpler logistic regression in this case. This could mean that the underlying patterns in the data are not sufficiently captured by the additional complexity of the random forest or GBM, or it could indicate that all models are somewhat limited by the same factors, such as the quality of the input data or the inherent noise within it.

Hematocrit is the top predictor in Random Forest and GBM and also highly important in the Enet model. This suggests that the measurement of the proportion of blood that is made up of red blood cells is a strong indicator of patient health status and hence the length of ICU stay. Respiratory Rate and Heart Rate are vital signs that appear to be consistently important across the models. These are critical indicators of a patient's respiratory and cardiovascular systems, which are essential for monitoring in ICU settings. Age (denoted as anchor_age) is also a common predictor, indicating that patient age may be associated with the length of ICU stay, which aligns with clinical expectations that older patients may often have longer stays due to complex health issues. Laboratory tests and vitals such as white blood cell count, temperature, and glucose levels are also key features. High or abnormal values can indicate infection or other acute conditions, which could lead to extended ICU care.

Performance: All models show similar performance, suggesting that the chosen features and the amount of data available are giving all the models the same level of predictive capability. The fact that stacking did not outperform the individual models could indicate that the individual model predictions are too correlated or not diverse enough to benefit from the meta-modeling approach.

Interpretability: Logistic regression typically offers the highest interpretability, as the coefficients of the model can be directly translated into odds ratios for each feature, giving clear insights into the relationship between the features and the outcome. Tree-based models like random forest and GBM are less interpretable due to their complexity – they are considered "black box" models, where the exact reasoning for any given prediction is not as clear. Stacking further complicates interpretability since it combines multiple models.