P8451\_HW7\_ML

Ruixi Li

2024-02-28

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.3 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.3 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(pROC)

## Type 'citation("pROC")' for a citation.  
##   
## Attaching package: 'pROC'  
##   
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(rpart.plot)

## Loading required package: rpart

# Data preparation

# load the data, remove dupplicate and clean variable names  
mi = read\_csv("mi.data.csv") |> distinct() |> janitor::clean\_names()|> select(-id)  
  
#mi |> group\_by(fc) |> count()  
  
skimr::skim(mi)# all variables are numeric

Data summary

|  |  |
| --- | --- |
| Name | mi |
| Number of rows | 1700 |
| Number of columns | 15 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| numeric | 15 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| age | 0 | 1 | 61.87 | 11.24 | 26.00 | 54.00 | 63.00 | 70.00 | 92.0 | ▁▃▇▆▁ |
| sex | 0 | 1 | 0.63 | 0.49 | 0.00 | 0.00 | 1.00 | 1.00 | 2.0 | ▅▁▇▁▁ |
| sodium | 0 | 1 | 136.56 | 6.49 | 117.00 | 133.00 | 136.00 | 140.00 | 169.0 | ▁▇▇▁▁ |
| alt | 0 | 1 | 0.47 | 0.38 | 0.03 | 0.23 | 0.38 | 0.61 | 3.0 | ▇▂▁▁▁ |
| wbc | 0 | 1 | 8.80 | 3.44 | 2.00 | 6.40 | 8.00 | 10.50 | 27.9 | ▆▇▂▁▁ |
| esr | 0 | 1 | 13.48 | 11.17 | 1.00 | 5.00 | 10.00 | 19.00 | 140.0 | ▇▁▁▁▁ |
| sbp | 0 | 1 | 138.85 | 34.49 | 0.00 | 120.00 | 140.00 | 160.00 | 260.0 | ▁▂▇▃▁ |
| dbp | 0 | 1 | 82.10 | 19.13 | 0.00 | 70.00 | 80.00 | 90.00 | 190.0 | ▁▃▇▁▁ |
| pulm\_adema | 0 | 1 | 0.07 | 0.25 | 0.00 | 0.00 | 0.00 | 0.00 | 1.0 | ▇▁▁▁▁ |
| fc | 0 | 1 | 1.22 | 1.04 | 0.00 | 0.00 | 2.00 | 2.00 | 4.0 | ▆▁▇▁▁ |
| arr | 0 | 1 | 0.02 | 0.16 | 0.00 | 0.00 | 0.00 | 0.00 | 1.0 | ▇▁▁▁▁ |
| diab | 0 | 1 | 0.13 | 0.34 | 0.00 | 0.00 | 0.00 | 0.00 | 1.0 | ▇▁▁▁▁ |
| obesity | 0 | 1 | 0.02 | 0.16 | 0.00 | 0.00 | 0.00 | 0.00 | 1.0 | ▇▁▁▁▁ |
| asthma | 0 | 1 | 0.02 | 0.15 | 0.00 | 0.00 | 0.00 | 0.00 | 1.0 | ▇▁▁▁▁ |
| readmission | 0 | 1 | 0.09 | 0.29 | 0.00 | 0.00 | 0.00 | 0.00 | 1.0 | ▇▁▁▁▁ |

# In machine learning, we don't need to recode the character variables into its 'label', instead we convert them as factor with set reference.  
cate\_var = c("sex","pulm\_adema","fc","arr","diab","obesity","asthma","readmission")  
# convert all categorical variables into factors  
mi = mi |> mutate(across(all\_of(cate\_var),as.factor))  
  
# set all categorical variables' reference group as '0'  
mi = mi |> mutate(across(all\_of(cate\_var), ~relevel(.x, ref = "0")))  
  
skimr::skim(mi)

Data summary

|  |  |
| --- | --- |
| Name | mi |
| Number of rows | 1700 |
| Number of columns | 15 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 8 |
| numeric | 7 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

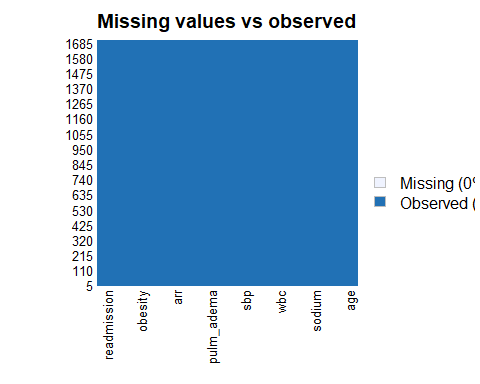
**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| sex | 0 | 1 | FALSE | 3 | 1: 1064, 0: 634, 2: 2 |
| pulm\_adema | 0 | 1 | FALSE | 2 | 0: 1589, 1: 111 |
| fc | 0 | 1 | FALSE | 5 | 2: 899, 0: 682, 3: 59, 1: 49 |
| arr | 0 | 1 | FALSE | 2 | 0: 1658, 1: 42 |
| diab | 0 | 1 | FALSE | 2 | 0: 1472, 1: 228 |
| obesity | 0 | 1 | FALSE | 2 | 0: 1658, 1: 42 |
| asthma | 0 | 1 | FALSE | 2 | 0: 1662, 1: 38 |
| readmission | 0 | 1 | FALSE | 2 | 0: 1541, 1: 159 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| age | 0 | 1 | 61.87 | 11.24 | 26.00 | 54.00 | 63.00 | 70.00 | 92.0 | ▁▃▇▆▁ |
| sodium | 0 | 1 | 136.56 | 6.49 | 117.00 | 133.00 | 136.00 | 140.00 | 169.0 | ▁▇▇▁▁ |
| alt | 0 | 1 | 0.47 | 0.38 | 0.03 | 0.23 | 0.38 | 0.61 | 3.0 | ▇▂▁▁▁ |
| wbc | 0 | 1 | 8.80 | 3.44 | 2.00 | 6.40 | 8.00 | 10.50 | 27.9 | ▆▇▂▁▁ |
| esr | 0 | 1 | 13.48 | 11.17 | 1.00 | 5.00 | 10.00 | 19.00 | 140.0 | ▇▁▁▁▁ |
| sbp | 0 | 1 | 138.85 | 34.49 | 0.00 | 120.00 | 140.00 | 160.00 | 260.0 | ▁▂▇▃▁ |
| dbp | 0 | 1 | 82.10 | 19.13 | 0.00 | 70.00 | 80.00 | 90.00 | 190.0 | ▁▃▇▁▁ |

mi |> Amelia::missmap(main = "Missing values vs observed")



summary(mi$readmission)

## 0 1   
## 1541 159

# the categorical outcome is strongly imbalanced

# Data Partitioning and cross-validation

set.seed(123)  
training.data = mi$readmission |> createDataPartition(p=0.7, list=F)  
train.data = mi[training.data, ]  
test.data = mi[-training.data, ]  
  
set.seed(123)  
control = trainControl(method="cv", number=10, sampling="up")  
# The professor said that we focus on proportion when deciding if we use over/uder sampling and focus on absolute sample size. I am not sure if there's golden rule for a "large/small" dataset, but for hw6, with 400+ "yes", I was advised to use upsampling. Maybe for this hw(150+ "yes"), I should also use upsampling.   
#Plus, the computational load is acceptable when using upsampling. I chose to use up sampling here.

# Model training

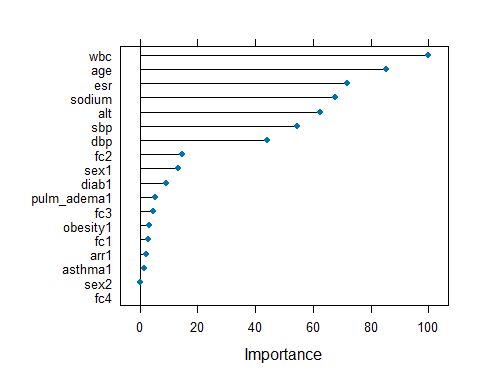
set.seed(123)  
  
# Define the tuning grid  
mtry = expand.grid(.mtry=seq(1, ncol(train.data), by = 1))  
  
# Setting mtry to the total number of features minus one is a fairly aggressive strategy, allowing each tree in the forest to consider almost all available features when making splits. This can be beneficial in certain datasets where most features contribute to predicting the target variable but can also increase the risk of overfitting, especially if some features are very predictive and consistently selected in the trees, leading to more correlated trees.So I chose to tuning thr mtry through cross validation  
  
# common ntree ranges from 100 to 1000, I tried 50, 100, 200, 300, ntree=100 produce the best performance, given other parameters constant.  
  
rf.model = train(  
 readmission ~.,   
 data=train.data,   
 method="rf",   
 metric="Accuracy",   
 tuneGrid=mtry,   
 ntree=100,   
 trControl=control)  
  
rf.model$bestTune

## mtry  
## 6 6

varImp(rf.model)

## rf variable importance  
##   
## Overall  
## wbc 1.000e+02  
## age 8.529e+01  
## esr 7.198e+01  
## sodium 6.751e+01  
## alt 6.251e+01  
## sbp 5.444e+01  
## dbp 4.396e+01  
## fc2 1.455e+01  
## sex1 1.328e+01  
## diab1 9.108e+00  
## pulm\_adema1 5.103e+00  
## fc3 4.421e+00  
## obesity1 3.295e+00  
## fc1 2.933e+00  
## arr1 2.156e+00  
## asthma1 1.626e+00  
## sex2 6.789e-03  
## fc4 0.000e+00

plot(varImp(rf.model))



confusionMatrix(rf.model)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction 0 1  
## 0 90.0 9.3  
## 1 0.6 0.1  
##   
## Accuracy (average) : 0.9009

set.seed(123)  
en.model = train(  
 readmission ~.,   
 data = train.data,   
 method = "glmnet",  
 trControl = control,   
 preProc=c("center", "scale"),  
 tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),   
 lambda = exp(seq(-3, 3, length = 100)))  
 )  
  
#Print the values of alpha and lambda that gave best prediction  
en.model$bestTune

## alpha lambda  
## 311 0.15 0.09126943

# Model coefficients  
coef(en.model$finalModel, en.model$bestTune$lambda)

## 19 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) -0.0004641491  
## age 0.0879594286  
## sex1 -0.0310334926  
## sex2 .   
## sodium .   
## alt 0.0133950283  
## wbc 0.0556078868  
## esr .   
## sbp 0.0834484675  
## dbp 0.2008850188  
## pulm\_adema1 -0.0534854956  
## fc1 -0.0138962680  
## fc2 0.0984891773  
## fc3 0.0773679000  
## fc4 -0.0470085678  
## arr1 -0.0460763474  
## diab1 .   
## obesity1 .   
## asthma1 0.0164357328

confusionMatrix(en.model)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction 0 1  
## 0 55.8 3.7  
## 1 34.8 5.7  
##   
## Accuracy (average) : 0.6154

* Model Comparison:

1. The random forest model shows a higher accuracy(0.9001) compared to the Elastic Net model(0.6154)， so I chose random forest model as my final model.
2. While the random forest model shows a higher accuracy(0.9001) compared to the Elastic Net model(0.6154), using the accuracy solely can be misleading on imbalanced datasets, even after upsampling, because it doesn’t account for the distribution of the different classes; it simply reflects the proportion of total correct predictions. I evaluated the both of the models using ROC curve and found that the Elastic Net model had a higher balanced accuracy, higher AUC value(0.6063 vs. 0.5695) and the ROC curve of it also at the left top corner compared with the random forest model’s ROC curve. That is, the Elastic Net model offers a more balanced approach to handling both classes, making it the better choice in scenarios where both sensitivity and specificity are important. If we really want to develop a methods that can be implemented in the clinical setting. We would better choose Elastic Net model. But given that the professor said in the last lecture that we always compare and choose optimal model before applying it to test data, I just continued the following steps using random forest as my final model.

# Model Evaluation

#Create predictions in test set  
pred.rf = rf.model |>  
 predict(test.data)  
  
eval.results=confusionMatrix(pred.rf, test.data$readmission, positive = "1")  
print(eval.results)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 459 47  
## 1 3 0  
##   
## Accuracy : 0.9018   
## 95% CI : (0.8725, 0.9262)  
## No Information Rate : 0.9077   
## P-Value [Acc > NIR] : 0.7091   
##   
## Kappa : -0.0112   
##   
## Mcnemar's Test P-Value : 1.193e-09   
##   
## Sensitivity : 0.000000   
## Specificity : 0.993506   
## Pos Pred Value : 0.000000   
## Neg Pred Value : 0.907115   
## Prevalence : 0.092338   
## Detection Rate : 0.000000   
## Detection Prevalence : 0.005894   
## Balanced Accuracy : 0.496753   
##   
## 'Positive' Class : 1   
##

#Create predictions as probabilities on test set   
pred.rf.prob = rf.model |>   
 predict(test.data, type = "prob")

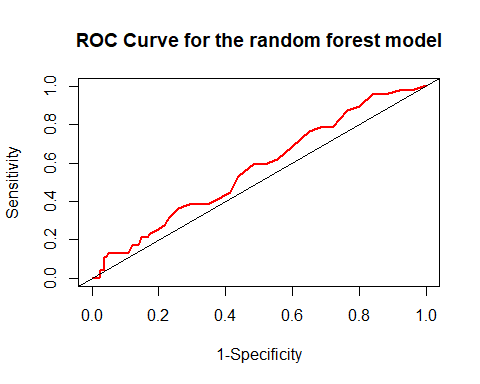
#Create predictions in test set  
#pred.en = en.model |>  
 #predict(test.data)  
  
#eval.results = confusionMatrix(pred.en, test.data$readmission, positive = "1")  
#print(eval.results)  
  
#Create predictions as probabilities on test set   
#pred.en.prob = en.model |>   
 #predict(test.data, type = "prob")

#Another potential evaluation: Area under the Receiver Operating Curve (AUROC)  
analysis\_rf = roc(response=test.data$readmission, predictor=pred.rf.prob[,2])

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

#analysis\_en = roc(response=test.data$readmission, predictor=pred.en.prob[,2])  
plot(1-analysis\_rf$specificities,analysis\_rf$sensitivities,type="l",  
ylab="Sensitivity",xlab="1-Specificity",col="red",lwd=2,  
main = "ROC Curve for the random forest model")  
#lines(1-analysis\_en$specificities, analysis\_en$sensitivities, col="blue", lwd=2)  
abline(a=0,b=1)



#legend("bottom", legend=c("random forest", "elastic net"),  
 #col=c("red", "blue"), lwd=2)  
auc\_rf = auc(analysis\_rf)  
auc\_rf

## Area under the curve: 0.5695

#auc\_en = auc(analysis\_en)  
#auc\_en

* AUC value for the random forest model is low, suggesting a weak discriniative ability of it.