P8451 Machine Learning in Public Health - Assignment 9

2023-3-28

As instructed, this analysis will be modeled based on the demonstration code from session 9. In preparation for all the analyses below, we will load the following libraries:

library(lattice)  
library(NHANES)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(caret)

## Loading required package: ggplot2

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.2.3

## randomForest 4.7-1.1

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

##   
## Attaching package: 'randomForest'

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

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

# Part 0: Data Preprocessing

## Data Import and Cleaning

We will begin by importing the **NHANES** data and processing it.

1. Subsetting the data to only include the relevant features
2. Removing observations with missing values

Remember from our prior assignment the data are imbalanced, so we will need to deal with this during our analysis.

data ("NHANES")  
table(NHANES$Diabetes)

##   
## No Yes   
## 9098 760

keep.var<-names(NHANES) %in% c("Age", "Race1", "Education", "Poverty", "Weight", "Height", "Pulse", "Diabetes", "BMI", "PhysActive", "Smoke100", "BPSysAve", "BPDiaAve", "TotChol")  
  
NHANES.subset<-NHANES[keep.var]  
  
str(NHANES.subset)

## tibble [10,000 × 14] (S3: tbl\_df/tbl/data.frame)  
## $ Age : int [1:10000] 34 34 34 4 49 9 8 45 45 45 ...  
## $ Race1 : Factor w/ 5 levels "Black","Hispanic",..: 4 4 4 5 4 4 4 4 4 4 ...  
## $ Education : Factor w/ 5 levels "8th Grade","9 - 11th Grade",..: 3 3 3 NA 4 NA NA 5 5 5 ...  
## $ Poverty : num [1:10000] 1.36 1.36 1.36 1.07 1.91 1.84 2.33 5 5 5 ...  
## $ Weight : num [1:10000] 87.4 87.4 87.4 17 86.7 29.8 35.2 75.7 75.7 75.7 ...  
## $ Height : num [1:10000] 165 165 165 105 168 ...  
## $ BMI : num [1:10000] 32.2 32.2 32.2 15.3 30.6 ...  
## $ Pulse : int [1:10000] 70 70 70 NA 86 82 72 62 62 62 ...  
## $ BPSysAve : int [1:10000] 113 113 113 NA 112 86 107 118 118 118 ...  
## $ BPDiaAve : int [1:10000] 85 85 85 NA 75 47 37 64 64 64 ...  
## $ TotChol : num [1:10000] 3.49 3.49 3.49 NA 6.7 4.86 4.09 5.82 5.82 5.82 ...  
## $ Diabetes : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ PhysActive: Factor w/ 2 levels "No","Yes": 1 1 1 NA 1 NA NA 2 2 2 ...  
## $ Smoke100 : Factor w/ 2 levels "No","Yes": 2 2 2 NA 2 NA NA 1 1 1 ...

#Remove missing values and then remove duplicates  
NHANES.subset<-na.omit(NHANES.subset)  
NHANES.subset<-unique(NHANES.subset)  
  
#Check distributions  
summary(NHANES.subset)

## Age Race1 Education Poverty   
## Min. :20.00 Black : 531 8th Grade : 266 Min. :0.000   
## 1st Qu.:32.00 Hispanic: 258 9 - 11th Grade: 511 1st Qu.:1.240   
## Median :46.00 Mexican : 390 High School : 828 Median :2.635   
## Mean :47.25 White :2390 Some College :1194 Mean :2.783   
## 3rd Qu.:60.00 Other : 311 College Grad :1081 3rd Qu.:4.650   
## Max. :80.00 Max. :5.000   
## Weight Height BMI Pulse   
## Min. : 37.00 Min. :139.9 Min. :15.02 Min. : 40.00   
## 1st Qu.: 67.50 1st Qu.:161.3 1st Qu.:24.20 1st Qu.: 64.00   
## Median : 79.80 Median :168.6 Median :27.80 Median : 72.00   
## Mean : 82.55 Mean :168.7 Mean :28.95 Mean : 72.45   
## 3rd Qu.: 93.90 3rd Qu.:175.8 3rd Qu.:32.34 3rd Qu.: 80.00   
## Max. :230.70 Max. :200.4 Max. :81.25 Max. :128.00   
## BPSysAve BPDiaAve TotChol Diabetes PhysActive  
## Min. : 78 Min. : 0.00 Min. : 1.530 No :3437 No :1850   
## 1st Qu.:109 1st Qu.: 63.00 1st Qu.: 4.290 Yes: 443 Yes:2030   
## Median :119 Median : 70.00 Median : 4.990   
## Mean :121 Mean : 69.93 Mean : 5.051   
## 3rd Qu.:130 3rd Qu.: 78.00 3rd Qu.: 5.690   
## Max. :226 Max. :116.00 Max. :13.650   
## Smoke100   
## No :2161   
## Yes:1719   
##   
##   
##   
##

## Partitioning Data

For the purposes of this analysis, we will partition the data into training and testing using a 70/30 split. This process involves applying the createDataPartition function to generate a set of training and testing data with equal proportion of individual with the outcome of interest, i.e., Diabetes. The new object train.data contains all the indexes of the rows in the original data set contained in the 70% split. The rows indexed to be in the 70% is assigned to a new training data set, and the remaining 30% is assigned to a new test.data object.

set.seed(123)  
  
training.data<-createDataPartition(NHANES.subset$Diabetes, p=0.7, list=F)  
train.data<-NHANES.subset[training.data, ]  
test.data<-NHANES.subset[-training.data, ]

# Part I: Creating Three Different Models

For the purposes of this analysis, we will create and compare the following models:

1. Random Forest Model with 3 values of mtry and 3 values of ntree
2. Support Vector Classifier Model
3. Logistic Regression Model

## 1.1 Model 1: Random Forest Model with 3 values of mtry and 3 values of ntree

As directed, *up sampling* was used in efforts in the following analysis to improve model performance.

# Try mtry of all, half of all, sqrt of all,   
# Try ntree of 100, 300, 500  
feat.count<-c((ncol(train.data)-1), (ncol(train.data)-1)/2, sqrt(ncol(train.data)-1))  
grid.rf<-expand.grid(mtry=feat.count)  
  
control.obj<-trainControl(method="cv", number=5, sampling="up")  
  
tree.num<-seq(100,500, by=200)  
results.trees<-list()  
for (ntree in tree.num){  
 set.seed(123)  
 rf.nhanes<-train(Diabetes~., data=train.data, method="rf", trControl=control.obj, metric="Accuracy", tuneGrid=grid.rf, importance=TRUE, ntree=ntree)  
 index<-toString(ntree)  
 results.trees[[index]]<-rf.nhanes$results  
}  
  
output.nhanes<-bind\_rows(results.trees, .id = "ntrees")  
best.tune<-output.nhanes[which.max(output.nhanes[,"Accuracy"]),]  
best.tune$mtry

## [1] 3.605551

results.trees

## $`100`  
## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 3.605551 0.8825926 0.1891744 0.005541275 0.06003981  
## 2 6.500000 0.8781768 0.1938987 0.004865396 0.04582566  
## 3 13.000000 0.8678786 0.1982621 0.014519506 0.07584329  
##   
## $`300`  
## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 3.605551 0.8825933 0.1725960 0.004509350 0.05543532  
## 2 6.500000 0.8789162 0.1964447 0.007288846 0.04962271  
## 3 13.000000 0.8711915 0.2172433 0.016515768 0.09099971  
##   
## $`500`  
## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 3.605551 0.8803874 0.1579985 0.006125591 0.04286762  
## 2 6.500000 0.8803874 0.2029017 0.007028040 0.04885534  
## 3 13.000000 0.8722938 0.2170273 0.013465096 0.07086335

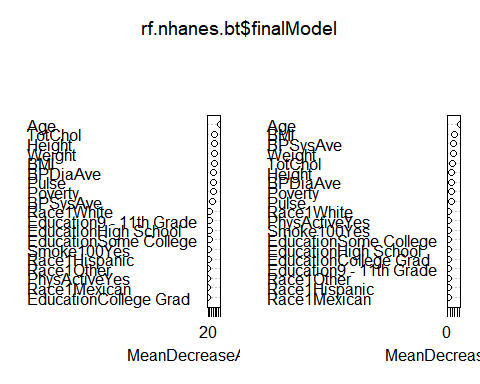
mtry.grid<-expand.grid(.mtry=best.tune$mtry)  
  
set.seed(123)  
 rf.nhanes.bt<-train(Diabetes~., data=train.data, method="rf", trControl=control.obj, metric="Accuracy", tuneGrid=mtry.grid, importance=TRUE, ntree=as.numeric(best.tune$ntrees))  
  
confusionMatrix(rf.nhanes.bt)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 86.5 9.4  
## Yes 2.0 2.0  
##   
## Accuracy (average) : 0.8855

varImp(rf.nhanes.bt)

## rf variable importance  
##   
## Importance  
## Age 100.0000  
## TotChol 48.5819  
## Weight 44.1984  
## Height 43.2151  
## BMI 42.6187  
## BPDiaAve 35.9389  
## BPSysAve 32.6963  
## Pulse 31.7347  
## Poverty 30.4601  
## Race1White 22.0407  
## Education9 - 11th Grade 12.2481  
## EducationHigh School 11.6640  
## EducationSome College 9.0631  
## Race1Mexican 6.2008  
## Smoke100Yes 6.0130  
## Race1Hispanic 4.1959  
## Race1Other 3.2143  
## EducationCollege Grad 0.9455  
## PhysActiveYes 0.0000

varImpPlot(rf.nhanes.bt$finalModel)

 Based on the output above, the average accuracy of Random Forest model is **0.8855**.

## 1.2 Model 2: Support Vector Classifier

To generate an SVC model, we will use the trainControl function to set our validation method. Next, we will incorporate different values for cost (C) into the model. We will also show information about the final model, and generate the metrics of accuracy from training using the confusionMatrix function.

set.seed(123)  
  
control.obj<-trainControl(method="cv", number=5, sampling="up", classProbs = TRUE)  
  
#Repeat expanding the grid search  
set.seed(123)  
  
svc.nhanes<-train(Diabetes ~ ., data=train.data, method="svmLinear", trControl=control.obj, preProcess=c("center", "scale"), probability=TRUE, tuneGrid=expand.grid(C=33.33))  
  
svc.nhanes$bestTune

## C  
## 1 33.33

svc.nhanes$results

## C Accuracy Kappa AccuracySD KappaSD  
## 1 33.33 0.7471529 0.2877784 0.01464678 0.02459951

confusionMatrix(svc.nhanes)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 66.0 2.8  
## Yes 22.5 8.7  
##   
## Accuracy (average) : 0.7471

Based on the output above, the average accuracy of Support Vector Classifier model is **0.7534**.

## 1.3 Model 3: Logistic Regression

We will employ a similar approach as demonstrated in Parts 1.1 and 1.2 to generate a logistic regression model. First, we will use the trainControl function to set our validation method, and we will train the algorithm by setting model = glm.

set.seed(123)  
  
control.obj<-trainControl(method="cv", number=5, sampling="up")  
  
log.nhanes<-train(Diabetes~., data=train.data, method="glm", family="binomial",preProcess=c("center", "scale"), trControl=control.obj)  
  
log.nhanes$results

## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.7449443 0.2872659 0.01293007 0.01994497

confusionMatrix(log.nhanes)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 65.7 2.7  
## Yes 22.8 8.8  
##   
## Accuracy (average) : 0.7449

coef(log.nhanes$finalModel)

## (Intercept) Age Race1Hispanic   
## -0.0299598258 1.2711654901 -0.0019044612   
## Race1Mexican Race1White Race1Other   
## 0.0779643578 -0.3066013994 0.2074473138   
## `Education9 - 11th Grade` `EducationHigh School` `EducationSome College`   
## -0.1720856801 -0.1382293524 0.0035292225   
## `EducationCollege Grad` Poverty Weight   
## -0.0779982329 -0.2201333076 -1.8476597303   
## Height BMI Pulse   
## 1.0231555185 2.4774811814 0.2526903162   
## BPSysAve BPDiaAve TotChol   
## 0.1691937444 0.0011036880 -0.1933529959   
## PhysActiveYes Smoke100Yes   
## -0.0009498227 0.2307703716

Based on the output above, the accuracy of the Logistic Regression model is **0.7449**.

# Part II: Comparing Three Different Models

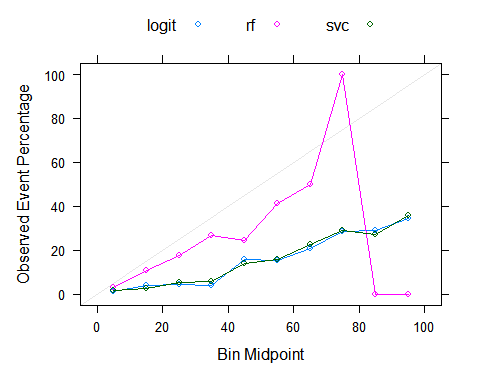
We will now generate predicted probabilities from each of the three models applied within the testing dataset, and plot and compare calibration curves across the three algorithms.

## 2.1 Generating Predicted Probabilities

#Predict in test-set and output probabilities  
rf.probs<-predict(rf.nhanes, test.data, type="prob")  
  
#Pull out predicted probabilities for Diabetes=Yes  
rf.pp<-rf.probs[,2]  
  
svc.probs<-predict(svc.nhanes,test.data, type="prob")  
svc.pp<-svc.probs[,2]  
  
#Predict in test-set using response type  
log.probs<-predict(log.nhanes, test.data, type="prob")  
logit.pp<-log.probs[,2]

## 2.2 Plot & Compare Calibration Curves

pred.prob<-data.frame(Class=test.data$Diabetes, logit=logit.pp, rf=rf.pp, svc=svc.pp)  
  
calplot<-(calibration(Class ~ logit+rf+svc, data=pred.prob, class="Yes", cuts=10))  
  
xyplot(calplot, auto.key=list(columns=3))



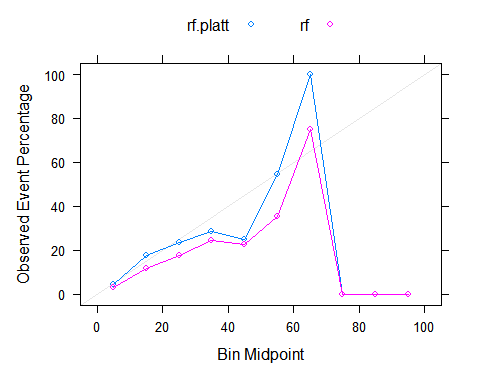
# Part III: Calibrating Probabilities

Below we calibrate the probabilites of all three models generated above in Part I. To do so, we partition testing data into 2 sets: set to train calibration and then set to evaluate results. We will employ the Platt’s Scaling method to train a LR model on the outputs of the classifier.

set.seed(123)  
cal.data.index<-test.data$Diabetes%>% createDataPartition(p=0.5, list=F)  
cal.data<-test.data[cal.data.index, ]  
final.test.data<-test.data[-cal.data.index, ]

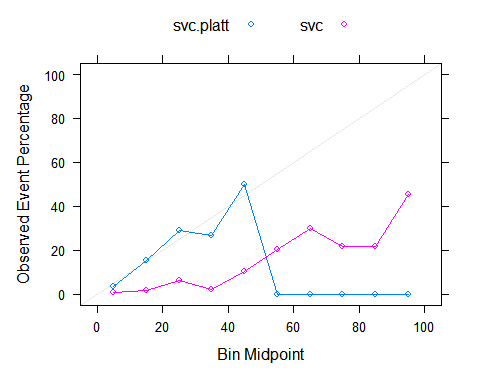
## 3.1 Calibration of Random Forest Model

#Predict on test-set without scaling to obtain raw pred prob in test set  
rf.probs.nocal<-predict(rf.nhanes, final.test.data, type="prob")  
rf.pp.nocal<-rf.probs.nocal[,2]  
  
#Apply model developed on training data to calibration dataset to obtain predictions  
rf.probs.cal<-predict(rf.nhanes, cal.data, type="prob")  
rf.pp.cal<-rf.probs.cal[,2]  
  
#Add to dataset with actual values from calibration data  
calib.rf.data.frame<-data.frame(rf.pp.cal, cal.data$Diabetes)  
colnames(calib.rf.data.frame)<-c("x", "y")  
  
#Use logistic regression to model predicted probabilities from calibration data to actual vales  
calib.rf.model<-glm(y ~ x, data=calib.rf.data.frame, family = binomial)  
  
#Apply calibration model above to raw predicted probabilities from test set  
data.test.rf<-data.frame(rf.pp.nocal)  
colnames(data.test.rf)<-c("x")  
platt.data.rf<-predict(calib.rf.model, data.test.rf, type="response")  
  
platt.prob.rf<-data.frame(Class=final.test.data$Diabetes, rf.platt=platt.data.rf, rf=rf.pp.nocal)  
  
calplot.rf<-(calibration(Class ~ rf.platt+rf, data=platt.prob.rf, class="Yes", cuts=10))  
xyplot(calplot.rf, auto.key=list(columns=2))



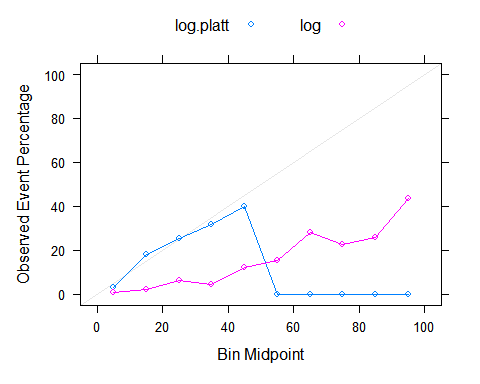
## 3.2 Calibration of Support Vector Classifier

#Predict on test-set without scaling  
svc.probs.nocal<-predict(svc.nhanes,final.test.data, type="prob")  
svc.pp.nocal<-svc.probs.nocal[,2]  
  
#Apply model developed on training data to calibration dataset to obtain predictions  
svc.probs.cal<-predict(svc.nhanes,cal.data, type="prob")  
svc.pp.cal<-svc.probs.cal[,2]  
  
#Add to dataset with actual values from calibration data  
calib.svc.data.frame<-data.frame(svc.pp.cal, cal.data$Diabetes)  
colnames(calib.svc.data.frame)<-c("x", "y")  
calib.svc.model<-glm(y ~ x, data=calib.svc.data.frame, family = binomial)  
  
#Predict on test set using model developed in calibration  
data.test.svc<-data.frame(svc.pp.nocal)  
colnames(data.test.svc)<-c("x")  
platt.data.svc<-predict(calib.svc.model, data.test.svc, type="response")  
  
platt.prob.svc<-data.frame(Class=final.test.data$Diabetes, svc.platt=platt.data.svc, svc=svc.pp.nocal)  
  
calplot<-(calibration(Class ~ svc.platt+svc, data=platt.prob.svc, class="Yes", cuts=10))  
xyplot(calplot, auto.key=list(columns=2))



## 3.3 Calibration of Logistic Regression

#Predict on test-set without scaling to obtain raw pred prob in test set  
log.probs.nocal<-predict(log.nhanes, final.test.data, type="prob")  
log.pp.nocal<-log.probs.nocal[,2]  
  
#Apply model developed on training data to calibration dataset to obtain predictions  
log.probs.cal<-predict(log.nhanes, cal.data, type="prob")  
log.pp.cal<-log.probs.cal[,2]  
  
#Add to dataset with actual values from calibration data  
calib.log.data.frame<-data.frame(log.pp.cal, cal.data$Diabetes)  
colnames(calib.log.data.frame)<-c("x", "y")  
  
#Use logistic regression to model predicted probabilities from calibration data to actual vales  
calib.log.model<-glm(y ~ x, data=calib.log.data.frame, family = binomial)  
  
#Apply calibration model above to raw predicted probabilities from test set  
data.test.log<-data.frame(log.pp.nocal)  
colnames(data.test.log)<-c("x")  
platt.data.log<-predict(calib.log.model, data.test.log, type="response")  
  
platt.prob.log<-data.frame(Class=final.test.data$Diabetes, log.platt=platt.data.log, log=log.pp.nocal)  
  
calplot.log<-(calibration(Class ~ log.platt+log, data=platt.prob.log, class="Yes", cuts=10))  
xyplot(calplot.log, auto.key=list(columns=2))



# Part IV: Discussion

## 4.1 Choosing the “Optimal” Model

An indicator of an optimal model is if the slope of the line in the calibration curve is equal to 1. More specifically, the value of Bin Midpoint is equal to the value of the Observed Event Percentage at all values. Referring to the calibration curves generated above, we can see that *none* of the three models are “optimal” since all lines in the curves are very irregular, and do not follow a slope = 1 pattern.

## 4.2 Additional Evaluation for Clinical Settings

The additional evaluation I would apply to this model to make it more appropriate for clinical settings is to use a 10-fold cross validation method.