P8451\_HW9

Ruixi Li

2024-03-20

# Data Preparation

library(lattice)  
library(NHANES)  
library(dplyr)  
library(caret)  
library(randomForest)  
library(pROC)  
  
  
data ("NHANES")  
table(NHANES$Diabetes)# the data is strongly imbalanced

##   
## 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]  
  
# check the coding and refernce group of the outcome  
contrasts(NHANES.subset$Diabetes)

## Yes  
## No 0  
## Yes 1

# Since the reference group is assigned correctly, I don't need to change it.  
  
# Set up the reference group for the prediction outcome  
# NHANES.subset = NHANES.subset |> mutate(Diabetes = relevel(Diabetes,ref = "No"))  
  
skimr::skim(NHANES.subset) # all variables are numeric or factor, have missing

Data summary

|  |  |
| --- | --- |
| Name | NHANES.subset |
| Number of rows | 10000 |
| Number of columns | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 5 |
| numeric | 9 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| Race1 | 0 | 1.00 | FALSE | 5 | Whi: 6372, Bla: 1197, Mex: 1015, Oth: 806 |
| Education | 2779 | 0.72 | FALSE | 5 | Som: 2267, Col: 2098, Hig: 1517, 9 -: 888 |
| Diabetes | 142 | 0.99 | FALSE | 2 | No: 9098, Yes: 760 |
| PhysActive | 1674 | 0.83 | FALSE | 2 | Yes: 4649, No: 3677 |
| Smoke100 | 2765 | 0.72 | FALSE | 2 | No: 4024, Yes: 3211 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Age | 0 | 1.00 | 36.74 | 22.40 | 0.00 | 17.00 | 36.00 | 54.00 | 80.00 | ▇▇▇▆▅ |
| Poverty | 726 | 0.93 | 2.80 | 1.68 | 0.00 | 1.24 | 2.70 | 4.71 | 5.00 | ▅▅▃▃▇ |
| Weight | 78 | 0.99 | 70.98 | 29.13 | 2.80 | 56.10 | 72.70 | 88.90 | 230.70 | ▂▇▂▁▁ |
| Height | 353 | 0.96 | 161.88 | 20.19 | 83.60 | 156.80 | 166.00 | 174.50 | 200.40 | ▁▁▁▇▂ |
| BMI | 366 | 0.96 | 26.66 | 7.38 | 12.88 | 21.58 | 25.98 | 30.89 | 81.25 | ▇▆▁▁▁ |
| Pulse | 1437 | 0.86 | 73.56 | 12.16 | 40.00 | 64.00 | 72.00 | 82.00 | 136.00 | ▂▇▃▁▁ |
| BPSysAve | 1449 | 0.86 | 118.15 | 17.25 | 76.00 | 106.00 | 116.00 | 127.00 | 226.00 | ▃▇▂▁▁ |
| BPDiaAve | 1449 | 0.86 | 67.48 | 14.35 | 0.00 | 61.00 | 69.00 | 76.00 | 116.00 | ▁▁▇▇▁ |
| TotChol | 1526 | 0.85 | 4.88 | 1.08 | 1.53 | 4.11 | 4.78 | 5.53 | 13.65 | ▂▇▁▁▁ |

#Remove missings and then remove duplicates  
NHANES.subset<-na.omit(NHANES.subset)  
NHANES.subset<-unique(NHANES.subset)  
  
#Check distributions  
skimr::skim(NHANES.subset)

Data summary

|  |  |
| --- | --- |
| Name | NHANES.subset |
| Number of rows | 3880 |
| Number of columns | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 5 |
| numeric | 9 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| Race1 | 0 | 1 | FALSE | 5 | Whi: 2390, Bla: 531, Mex: 390, Oth: 311 |
| Education | 0 | 1 | FALSE | 5 | Som: 1194, Col: 1081, Hig: 828, 9 -: 511 |
| Diabetes | 0 | 1 | FALSE | 2 | No: 3437, Yes: 443 |
| PhysActive | 0 | 1 | FALSE | 2 | Yes: 2030, No: 1850 |
| Smoke100 | 0 | 1 | FALSE | 2 | No: 2161, Yes: 1719 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Age | 0 | 1 | 47.25 | 17.30 | 20.00 | 32.00 | 46.00 | 60.00 | 80.00 | ▇▇▇▅▅ |
| Poverty | 0 | 1 | 2.78 | 1.67 | 0.00 | 1.24 | 2.63 | 4.65 | 5.00 | ▅▆▃▃▇ |
| Weight | 0 | 1 | 82.55 | 21.33 | 37.00 | 67.50 | 79.80 | 93.90 | 230.70 | ▆▇▁▁▁ |
| Height | 0 | 1 | 168.66 | 10.06 | 139.90 | 161.30 | 168.60 | 175.80 | 200.40 | ▁▆▇▅▁ |
| BMI | 0 | 1 | 28.95 | 6.84 | 15.02 | 24.20 | 27.80 | 32.34 | 81.25 | ▇▆▁▁▁ |
| Pulse | 0 | 1 | 72.45 | 11.85 | 40.00 | 64.00 | 72.00 | 80.00 | 128.00 | ▁▇▅▁▁ |
| BPSysAve | 0 | 1 | 121.02 | 17.15 | 78.00 | 109.00 | 119.00 | 130.00 | 226.00 | ▃▇▂▁▁ |
| BPDiaAve | 0 | 1 | 69.93 | 12.68 | 0.00 | 63.00 | 70.00 | 78.00 | 116.00 | ▁▁▇▇▁ |
| TotChol | 0 | 1 | 5.05 | 1.06 | 1.53 | 4.29 | 4.99 | 5.69 | 13.65 | ▂▇▁▁▁ |

# Set up: Partition data into training/testing

set.seed(123)  
  
train.indices <- NHANES.subset %>%  
 pull(Diabetes) %>%  
 createDataPartition(p = 0.7, list = FALSE)  
  
train.data <- NHANES.subset %>%  
 slice(train.indices)

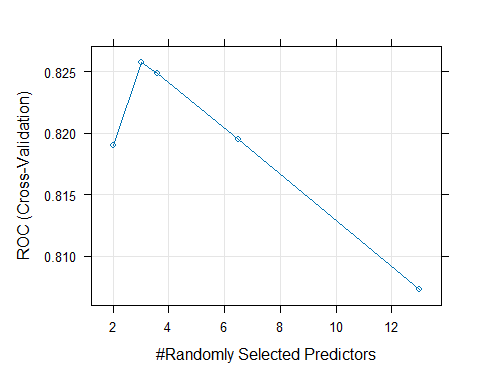
## Warning: Slicing with a 1-column matrix was deprecated in dplyr 1.1.0.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

test.data <- NHANES.subset %>%  
 slice(-train.indices)

# Model building and hyperparameter tuning

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

control.obj<-trainControl(method="cv",   
 number=5,   
 sampling="up",   
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
#I use 5-fold cv here to reduce computational load  
# The twoClassSummary function is designed for binary classification problems and will provide metrics like sensitivity, specificity, and Area Under the ROC Curve (AUC).  
# ClassProbs=TRUE is necessary for twoClassSummary to work because it requires class probabilities to calculate AUC.  
  
# hyperparameter tuning  
# 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),2,3)  
# I want to try more mtry, but it takes too long. Since the model have poor performace when mtry=1, I just add 2, 3 as candidate mtry.  
  
grid.rf<-expand.grid(mtry=feat.count)  
  
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="ROC",   
 tuneGrid=grid.rf,   
 importance=TRUE, # the model will calculate variable importance measures  
 ntree=ntree)  
 index<-toString(ntree)  
 results.trees[[index]]<-rf.nhanes$results  
}  
  
plot(rf.nhanes)



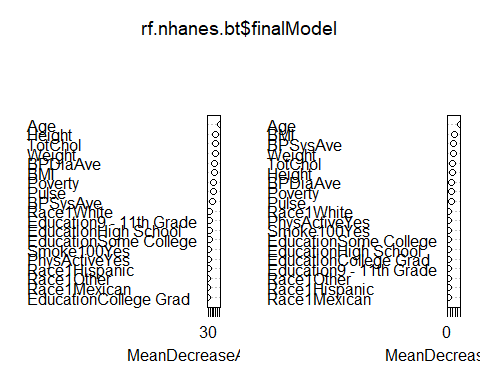
output.nhanes<-bind\_rows(results.trees, .id = "ntrees")  
best.tune<-output.nhanes[which.max(output.nhanes[,"ROC"]),]  
best.tune$mtry

## [1] 3

# results.trees-no need to output  
mtry.grid<-expand.grid(.mtry=best.tune$mtry)# choose the best tune to retrain the model  
  
set.seed(123)  
 rf.nhanes.bt<-train(  
 Diabetes~.,   
 data=train.data,   
 method="rf",   
 trControl=control.obj,   
 metric="ROC",   
 tuneGrid=mtry.grid,   
 importance=TRUE,  
 ntree=as.numeric(best.tune$ntrees))  
  
  
varImp(rf.nhanes.bt)

## rf variable importance  
##   
## Importance  
## Age 100.000  
## TotChol 50.432  
## Height 44.750  
## Weight 43.117  
## BMI 42.418  
## BPDiaAve 38.666  
## BPSysAve 31.223  
## Poverty 30.344  
## Pulse 29.462  
## Race1White 20.569  
## EducationHigh School 11.712  
## Education9 - 11th Grade 11.465  
## EducationSome College 7.319  
## Race1Hispanic 2.709  
## Smoke100Yes 2.564  
## Race1Other 2.496  
## Race1Mexican 2.273  
## PhysActiveYes 1.103  
## EducationCollege Grad 0.000

varImpPlot(rf.nhanes.bt$finalModel)

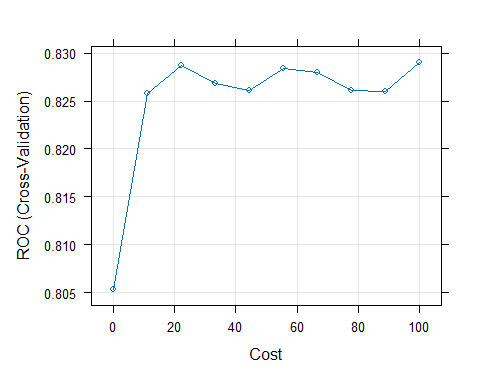


* Random forest have the function of feature selection, we might consider focusing on the most important variables for a more parsimonious model or for insight into the features most strongly associated with the outcome variable. Given the variable importance plot, some Education, race category is not that important(contribution <10%), but they can not be excluded separately.Ideally, I want to take a subset of features that contribute the most to model accuracy and purity to achieve the best trade-off between model complexity and predictive power.
* Instead of removing features,a more common approach would be to take the features until you reach a point of diminishing returns, where adding more features does not significantly increase the model’s performance.

## Model 2: Support Vector Classifier

set.seed(123)  
  
control.obj<-trainControl(method="cv",   
 number=5,   
 sampling="up",  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
  
#Repeat expanding the grid search  
set.seed(123)  
  
svc.nhanes<-train(  
 Diabetes ~ .,   
 data=train.data,   
 method="svmLinear",   
 metric="ROC",   
 trControl=control.obj,   
 preProcess=c("center", "scale"),   
 probability=TRUE,   
 importance=TRUE,  
 tuneGrid=expand.grid(C=seq(0.0001,100, length=10)))  
# for random forest, preProcess and probability is not necessary  
# svc.nhanes$results  
plot(svc.nhanes)

svc.nhanes.bt<-train(  
 Diabetes ~ .,   
 data=train.data,   
 method="svmLinear",   
 metric="ROC",   
 trControl=control.obj,   
 preProcess=c("center", "scale"),   
 probability=TRUE,   
 importance=TRUE,  
 tuneLength=11.1112)



* At very low cost values (near 0), the model performance is poorer, with ROC values significantly lower than the rest of the cost values. This indicates underfitting;The ROC performance increase sharply as cost increases from 0 to 10; Beyond a certain cost threshold (around 10 to 20), the ROC value levels off and becomes relatively stable despite further increases in cost. This suggests that the model has reached a point where adding more complexity (lower regularization) does not significantly improve cross-validated performance. There is some variability in ROC scores at higher cost values, but it does not show a clear trend of improvement or degradation. This slight variation is typical in cross-validation results due to the randomness inherent in the partitioning of data. I would say the best tune might be 11.1112.

## Model 3: Logistic Regression

set.seed(123)  
  
control.obj<-trainControl(method="cv",   
 number=5,   
 sampling="up",  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
  
logit.nhanes<-train(  
 Diabetes~.,   
 data=train.data,   
 method="glm",   
 family="binomial",  
 metric="ROC",   
 preProcess=c("center", "scale"),   
 trControl=control.obj)  
# importance=TRUE is not applicable in logistic regression  
  
logit.nhanes$results

## parameter ROC Sens Spec ROCSD SensSD SpecSD  
## 1 none 0.828858 0.7423159 0.7651306 0.0255589 0.01913421 0.05827666

coef(logit.nhanes$finalModel)

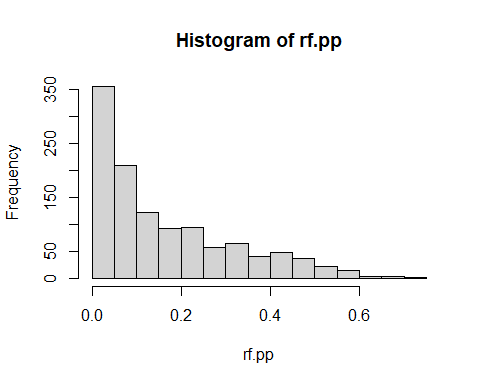
## (Intercept) Age Race1Hispanic   
## -0.0299598258 1.2711654901 -0.0019044612   
## Race1Mexican Race1White Race1Other   
## 0.0779643578 -0.3066013994 0.2074473138   
## `Education9 - 11th Grade` `EducationHigh School` `EducationSome College`   
## -0.1720856801 -0.1382293524 0.0035292225   
## `EducationCollege 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

# plot(logit.nhanes$finalModel,select=3)   
# The plot() function for a glm object can produce several types of diagnostic plots to assess the fit of the model. The select argument specifies which type of plot to produce:  
  
# select=1: Residuals vs Fitted  
# select=2: Normal Q-Q  
# select=3: Scale-Location (also known as Spread-Location or Standardized residuals vs. Fitted)  
# select=4: Cook's Distance plot  
# select=5: Residuals vs Leverage plot that helps us to find influential cases

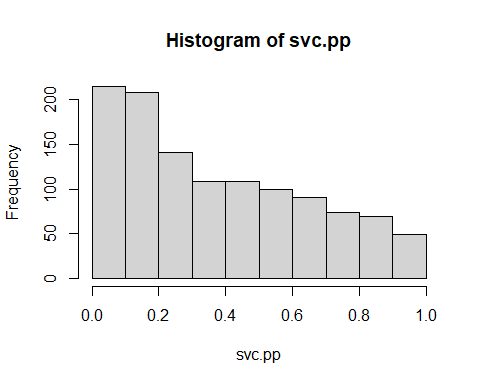
# Calibration

## Get predicted propabilities

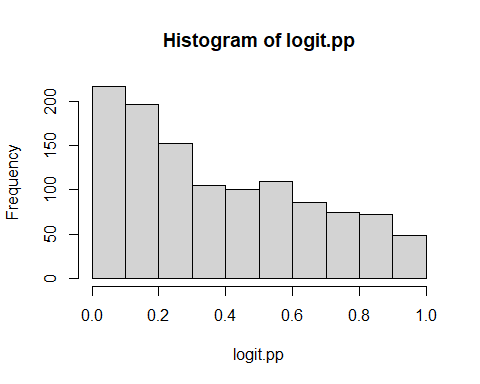
#Predict in test-set and output probabilities  
rf.probs<-predict(rf.nhanes, test.data, type="prob")  
  
#Pull out predicted probabilities for Diabetes=Yes  
rf.pp<-rf.probs[,2]  
  
  
svc.probs<-predict(svc.nhanes.bt,test.data, type="prob")  
svc.pp<-svc.probs[,2]  
  
  
#Predict in test-set using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



hist(svc.pp)



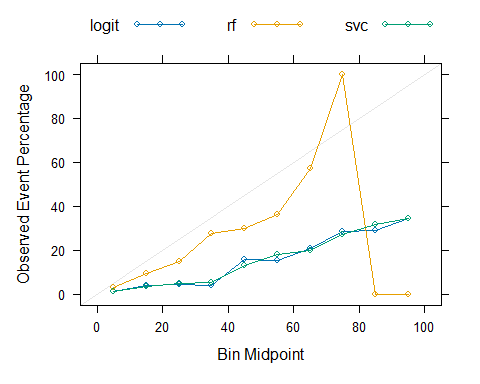
hist(logit.pp)



## Pre calibration plots

Plot and compare calibration curves across the three algorithms.

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))



* The diagonal grey line represents ideal calibration, where the predicted probabilities match the observed frequencies. A model’s line above the diagonal indicates underconfidence (the model’s predictions are too conservative), and a line below the diagonal indicates overconfidence (the model’s predictions are too optimistic). In this case, rf and logit model shows similar calibration, they are both overconfident across all bins(when the predicted probability is high); The rf model is relatively well-calibrated, especially when the predicted probability is low.

## Calibrate the probabilities from SVC and RF/Post calibration plots

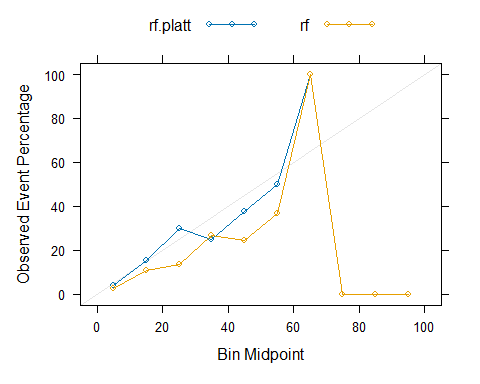
Partition testing data into 2 sets: set to train calibration and then set to evaluate results

Method 1: Platt’s Scaling-train a logistic regression model on the outputs of your classifier

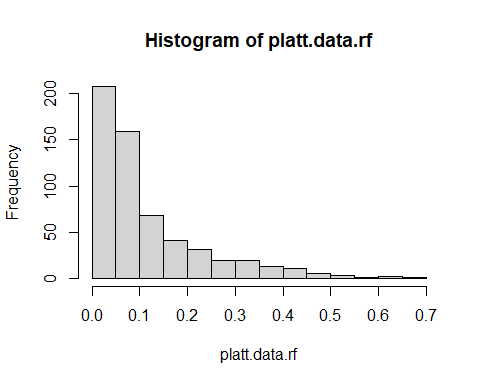
set.seed(123)  
cal.data.index<-test.data$Diabetes %>%   
 createDataPartition(p=0.5, list=F)  
  
cal.data<-test.data[cal.data.index, ]  
final.test.data<-test.data[-cal.data.index, ]

1. Calibration of RF

#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.data.frame.rf<-data.frame(rf.pp.cal, cal.data$Diabetes)  
colnames(calib.data.frame.rf)<-c("x", "y")  
  
#Use logistic regression to model predicted probabilities from calibration data to actual vales  
calib.model.rf<-glm(y ~ x, data=calib.data.frame.rf, 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.model.rf, 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))



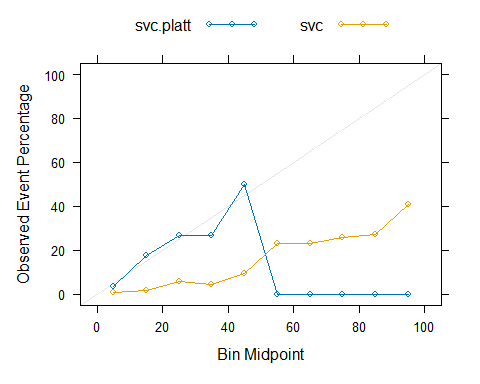
hist(platt.data.rf)



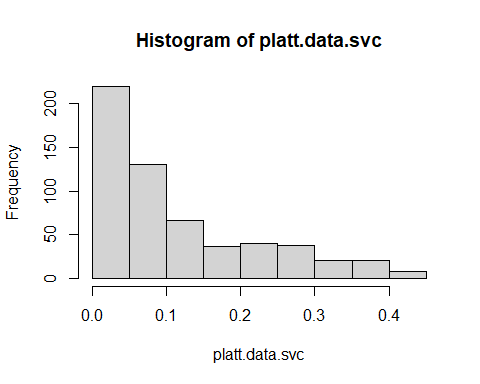
* Platt’s Scaling corrected the overconfident prediction when the bin midpoint < 60, improving the calibration of the RF model.

1. Calibration of SVC

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



hist(platt.data.svc)



* Platt’s Scaling corrected the overconfident prediction when the bin midpoint < 50, improving the calibration of the SVC model.

# Using resamples(multiple evaluation methods) to compare the three models

res = resamples(list(RF = rf.nhanes.bt,  
 SVC = svc.nhanes.bt,  
 GLM = logit.nhanes))  
summary(res)

##   
## Call:  
## summary.resamples(object = res)  
##   
## Models: RF, SVC, GLM   
## Number of resamples: 5   
##   
## ROC   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## RF 0.7984523 0.8055127 0.8301757 0.8213050 0.8331214 0.8392630 0  
## SVC 0.7996781 0.8096036 0.8315348 0.8290127 0.8378062 0.8664409 0  
## GLM 0.7999799 0.8105761 0.8269148 0.8288580 0.8425579 0.8642613 0  
##   
## Sens   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## RF 0.9729730 0.9792100 0.9812890 0.9804651 0.9813278 0.9875260 0  
## SVC 0.7406639 0.7422037 0.7442827 0.7510434 0.7463617 0.7817048 0  
## GLM 0.7302905 0.7318087 0.7318087 0.7423159 0.7422037 0.7754678 0  
##   
## Spec   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## RF 0.09677419 0.1290323 0.1451613 0.1477727 0.1774194 0.1904762 0  
## SVC 0.67741935 0.7580645 0.7741935 0.7715822 0.8095238 0.8387097 0  
## GLM 0.67741935 0.7419355 0.7741935 0.7651306 0.8095238 0.8225806 0

* I assume the purpose of this machine learning is to develop a tool to predict diabetes using above predictors. Estimated prevalence of diabetes in the United States is 11.6% in the population. So, we can just look at the calibration curve when the bin midpoint is low. Among the three models, rf had the best calibration at first. But after Platt’s scaling, SVC can achieve better calibration than rf.
* AUC value, which indicates the discriminative ability is important. Meanwhile, for most practical applications in clinical setting, a balance between sensitivity and specificity is desirable. Given the results, ROC of the three models are similar, RF’s specificity is too low whereas its sensitivity is higher than the other two models. So RF is not the optimal model. SVC had higher sensitivity, AUC compared with Logistic regression ,their specificity are the same. So, I would choose SVC as the optimal mode.