Homework 3.1 - Imputations and Imbalanced data

Hamed

2/23/2020

# Performance measurement, Imbalance correction and Imputing missing values

## 1. Load the Wisconsin breast cancer dataset from blackboard

suppressWarnings(library(tidyverse))

## -- Attaching packages ------------------------------------------------- tidyverse 1.3.0 --

## <U+2713> ggplot2 3.2.1 <U+2713> purrr 0.3.3  
## <U+2713> tibble 2.1.3 <U+2713> dplyr 0.8.3  
## <U+2713> tidyr 1.0.0 <U+2713> stringr 1.4.0  
## <U+2713> readr 1.3.1 <U+2713> forcats 0.4.0

## -- Conflicts ---------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

suppressWarnings(library(caret))

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

suppressWarnings(library(rpart.plot))

## Loading required package: rpart

suppressWarnings(library(randomForest))

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

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

suppressWarnings(library(e1071))  
suppressWarnings(library(ROCR))

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

suppressWarnings(library(pROC))

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

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

suppressWarnings(library(RCurl))

## Loading required package: bitops

##   
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':  
##   
## complete

suppressWarnings(library(gbm))

## Loaded gbm 2.1.5

breast\_cancer<-read.csv('WisconsinBreastCancerwithmissing.csv',header = T)  
  
str(breast\_cancer) #check the variables

## 'data.frame': 699 obs. of 10 variables:  
## $ classes : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...  
## $ clump\_thickness : int NA 5 3 NA NA 8 1 2 2 4 ...  
## $ uniformity\_of\_cell\_size : int NA 4 1 8 1 10 1 1 1 2 ...  
## $ uniformity\_of\_cell\_shape : int 1 4 1 8 1 10 1 2 1 1 ...  
## $ marginal\_adhesion : int 1 5 1 1 3 8 1 1 1 1 ...  
## $ single\_epithelial\_cell\_size: int 2 7 2 3 2 7 2 2 2 2 ...  
## $ bare\_nuclei : int 1 10 2 4 1 10 10 1 1 1 ...  
## $ bland\_chromatin : int 3 3 3 3 3 9 3 3 1 2 ...  
## $ normal\_nucleoli : int 1 2 1 7 1 7 1 1 1 1 ...  
## $ mitosis : int 1 1 1 1 1 1 1 1 5 1 ...

dim(breast\_cancer) #check the size of dataframe

## [1] 699 10

## 2. How many missing data points are there?

#Shows the number of missing values  
sum(is.na(breast\_cancer))

## [1] 234

#In which columns are they missing?  
colnames(breast\_cancer)[colSums(is.na(breast\_cancer)) > 0]

## [1] "clump\_thickness" "uniformity\_of\_cell\_size"  
## [3] "bare\_nuclei"

## 3. Impute for the first column(‘clump\_thickness’) with missing values using the mean and round to an integer

breast\_cancer$clump\_thickness[is.na(breast\_cancer$clump\_thickness)]= round(mean(breast\_cancer$clump\_thickness, na.rm=TRUE),digits = 0)  
  
#Check again for number of missing values  
sum(is.na(breast\_cancer$clump\_thickness))

## [1] 0

## 4. Impute for the second column(‘uniformity\_of\_cell\_size’) with missing values using KNN with three nearest neighbors and round to integer

#Check for tge number of missing values before imputation  
sum(is.na(breast\_cancer$uniformity\_of\_cell\_size))

## [1] 63

#Imputation  
library(VIM)

## Loading required package: colorspace

##   
## Attaching package: 'colorspace'

## The following object is masked from 'package:pROC':  
##   
## coords

## Loading required package: grid

## Loading required package: data.table

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

## The following object is masked from 'package:purrr':  
##   
## transpose

## VIM is ready to use.   
## Since version 4.0.0 the GUI is in its own package VIMGUI.  
##   
## Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':  
##   
## sleep

breast\_cancer<-kNN(data = breast\_cancer, variable = 'uniformity\_of\_cell\_size',  
 k=3, imp\_var=FALSE)  
  
  
#Check for the number of missing values after imputation  
sum(is.na(breast\_cancer$uniformity\_of\_cell\_size))

## [1] 0

## 5. Impute for the third column(‘bare\_nuclei’) with missing values using a regression to predict the third column based on uniformity\_of\_cell\_shape, marginal\_adhesion and normal\_nucleoli. Round to integers.

#Check for the number of missing values before imputing  
sum(is.na(breast\_cancer$bare\_nuclei))

## [1] 71

#Start by creating an indicator variable  
Ind<-function(t)  
{  
 x=dim(length(t))  
 x[which(!is.na(t))]=1  
 x[which(is.na(t))]=0  
 return(x)  
}  
breast\_cancer$I<-Ind(breast\_cancer$bare\_nuclei)  
  
  
#Fit a simple linear between third column and the mentioned columns  
mymodel <- lm(bare\_nuclei ~ uniformity\_of\_cell\_shape + marginal\_adhesion + normal\_nucleoli, data = breast\_cancer)  
  
  
#Impute missing value with the model output in a for loop  
for(i in 1:nrow(breast\_cancer))  
{  
 if(breast\_cancer$I==0)  
 {  
 breast\_cancer$bare\_nuclei[i] = mymodel$coefficients[1]+   
 mymodel$coefficients[2]\*breast\_cancer$uniformity\_of\_cell\_shape[i]+  
 mymodel$coefficients[3]\*breast\_cancer$marginal\_adhesion[i]+  
 mymodel$coefficients[4]\*breast\_cancer$normal\_nucleoli[i]  
 }  
}  
  
  
#check for the number of missing values after imputation  
sum(is.na(breast\_cancer$bare\_nuclei))

## [1] 71

### More data preparation before modeling, because the regression method of imputation had alot of warnings and did not impute missing data, thus I used KNN to impute any remaining missing data

breast\_cancer<-kNN(data = breast\_cancer, variable = 'bare\_nuclei',  
 k=3, imp\_var=FALSE)  
  
  
#Check for missing values after imputation  
sum(is.na(breast\_cancer$bare\_nuclei))

## [1] 0

# Drop the indicator variable 'I' to exclude it from the next modelling phase   
library(dplyr)  
breast\_cancer<-select(breast\_cancer,-c(I))

## 6. Build a decision tree model to predict class using 80% training data and five fold cross validation with three repeats. Hint: Use method =" rpart" in caret

#Load packages  
library(tidyverse)  
library(rpart)  
library(dplyr)  
  
# First split the dataset  
set.seed(123)  
## 80% of the sample size  
smp\_size <- floor(0.80 \* nrow(breast\_cancer))  
train\_ind <- sample(seq\_len(nrow(breast\_cancer)), size = smp\_size)  
train.cancer <- breast\_cancer[train\_ind, ]  
test.cancer <- breast\_cancer[-train\_ind, ]  
dim(train.cancer)

## [1] 559 10

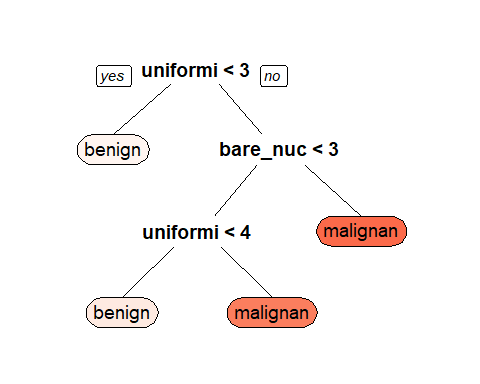
dim(test.cancer)

## [1] 140 10

# Fit the model on the training set  
model\_tree<-train(classes~.,data=train.cancer, method = "rpart", trControl = trainControl("cv", number = 5),tuneLength = 3)  
model\_tree

## CART   
##   
## 559 samples  
## 9 predictor  
## 2 classes: 'benign', 'malignant'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 447, 447, 447, 448, 447   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.03517588 0.9427606 0.8754786  
## 0.04271357 0.9320463 0.8550456  
## 0.80402010 0.8600869 0.6497040  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.03517588.

#visulize the decision tree  
prp(model\_tree$finalModel, box.palette = "Reds", tweak = 1.2)

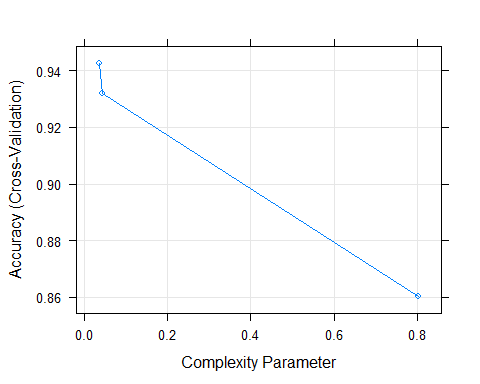


## 7. Show the accuracy results for each resample (fold) in the training data

model\_tree$results

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.03517588 0.9427606 0.8754786 0.01014002 0.02392457  
## 2 0.04271357 0.9320463 0.8550456 0.01611438 0.03265097  
## 3 0.80402010 0.8600869 0.6497040 0.11842918 0.36384705

# Plot model accuracy vs different values of cp (complexity parameter)  
plot(model\_tree)



## 8. Do the following performance measures for the test dataset

# (a).Compute all the accuracy measures (Accuracy, sensitivity, specificity etc)  
  
# Make predictions on the test data  
predicted.classes=model\_tree%>% predict(test.cancer)  
  
# Compute model accuracy rate on test data  
mean(predicted.classes == test.cancer$classes)

## [1] 0.9714286

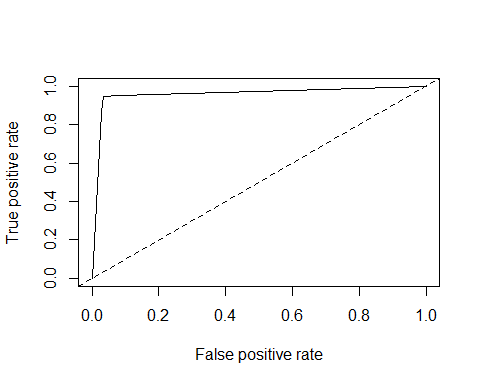
#calculate sensitivity  
sensitivity(predicted.classes,test.cancer$classes)

## [1] 0.9693878

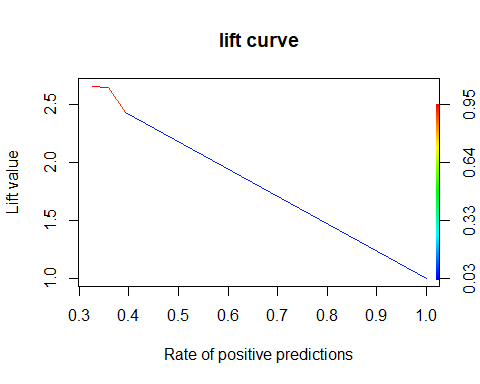
#calculate specificity  
specificity(predicted.classes,test.cancer$classes)

## [1] 0.9761905

# (b).Plot the ROC curve  
library(ROCR)  
pred <- prediction(predict(model\_tree, type = "prob")[, 2],train.cancer$classes)  
plot(performance(pred, "tpr", "fpr"))  
abline(0, 1, lty = 2)



# (c).Plot the lift curve  
perf <- performance(pred,"lift","rpp")  
plot(perf, main="lift curve", colorize=T)



# (d).Compute the AUC  
  
auc = performance(pred, 'auc')  
slot(auc, 'y.values')

## [[1]]  
## [1] 0.9605946

## 9. Is the cancer data imbalanced by the class feature? What is the percentage of the majority class and the percentage of the minority class?

#Calculates the class frequencies to check for the majority class  
table(breast\_cancer$classes)

##   
## benign malignant   
## 458 241

#Gets the percentage: hence class "benign" is the majority while class "maligant" is the minority  
class\_percentage<- as.matrix(prop.table(table(breast\_cancer$classes)) \* 100)  
class\_percentage

## [,1]  
## benign 65.52217  
## malignant 34.47783

## 10. Now re-build the decision tree model above after correcting for imbalance using SMOTE.

# initialize imbalanced data  
imbal\_train=train.cancer  
imbal\_test=test.cancer  
  
# Set up control function for training  
ctrl <- trainControl(method = "repeatedcv",  
 number = 5,  
 repeats = 3,  
 classProbs = TRUE)  
  
  
#Build smote model  
  
ctrl$sampling <- "smote"  
  
smote\_fit <- train(classes ~ .,  
 data = imbal\_train,  
 method = "gbm",  
 verbose = FALSE,  
 metric = "ROC",  
 trControl = ctrl)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

##   
## Attaching package: 'DMwR'

## The following object is masked from 'package:VIM':  
##   
## kNN

smote\_fit

## Stochastic Gradient Boosting   
##   
## 559 samples  
## 9 predictor  
## 2 classes: 'benign', 'malignant'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 447, 448, 447, 447, 447, 447, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.9630309 0.9194193  
## 1 100 0.9606446 0.9139953  
## 1 150 0.9600440 0.9123142  
## 2 50 0.9636208 0.9207044  
## 2 100 0.9636261 0.9204065  
## 2 150 0.9594595 0.9111585  
## 3 50 0.9618350 0.9165948  
## 3 100 0.9612398 0.9150377  
## 3 150 0.9618350 0.9165255  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 100, interaction.depth =  
## 2, shrinkage = 0.1 and n.minobsinnode = 10.

## 11. Did any of the accuracy measures improve? If so, which ones?

Yes the SMOTE model showed an improvement in the accuracy measures by showing a higher AUC score

model\_list <- list(original = model\_tree,  
 SMOTE = smote\_fit)  
model\_list

## $original  
## CART   
##   
## 559 samples  
## 9 predictor  
## 2 classes: 'benign', 'malignant'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 447, 447, 447, 448, 447   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.03517588 0.9427606 0.8754786  
## 0.04271357 0.9320463 0.8550456  
## 0.80402010 0.8600869 0.6497040  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.03517588.  
##   
## $SMOTE  
## Stochastic Gradient Boosting   
##   
## 559 samples  
## 9 predictor  
## 2 classes: 'benign', 'malignant'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 3 times)   
## Summary of sample sizes: 447, 448, 447, 447, 447, 447, ...   
## Addtional sampling using SMOTE  
##   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.9630309 0.9194193  
## 1 100 0.9606446 0.9139953  
## 1 150 0.9600440 0.9123142  
## 2 50 0.9636208 0.9207044  
## 2 100 0.9636261 0.9204065  
## 2 150 0.9594595 0.9111585  
## 3 50 0.9618350 0.9165948  
## 3 100 0.9612398 0.9150377  
## 3 150 0.9618350 0.9165255  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 100, interaction.depth =  
## 2, shrinkage = 0.1 and n.minobsinnode = 10.

#First Build custom AUC function to extract AUC from the caret model object  
test\_roc <- function(model, data) {  
 roc(data$classes,  
 predict(model, data, type = "prob")[, "malignant"])  
}  
  
  
  
# Get the AUC for the original model with the original dataset.  
model\_tree %>%  
 test\_roc(data = test.cancer)%>%auc()

## Setting levels: control = benign, case = malignant

## Setting direction: controls < cases

## Area under the curve: 0.9712

# Get the AUC for the SMOTE model with the Imbalanced dataset.  
smote\_fit %>%  
 test\_roc(data = imbal\_test)%>%auc()

## Setting levels: control = benign, case = malignant  
## Setting direction: controls < cases

## Area under the curve: 0.9922