Untitled

jean wills

29/07/2020

# Step 3 - Feature Selection Methods

Note that these models are NOT being tested for metrics, but rather as a ’one-off" to see if we can reduce the number of attributes

### 1: PCA

### 2. Recursive feature elimination (RFE) with Random Forest

### 3. Boruta

### 4. neural network LVQ method

### 5. Regression - ANOVA, AIC

### 6. Regression - AIC stepwise

## this file uses “BM\_mini\_num\_sc”

## Get the data in the proper format first:

### read in dataset - on “BM\_mini”

library(plyr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

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

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

BM\_mini <- read.csv("/Users/jeanwills/Desktop/CKME136/1\_data/bank.csv", header=T, sep = ";", stringsAsFactors = T, na.strings = "NA")

### NUMERIC DATA Cleaning - change numeric data to 95% CI

BM<-BM\_mini  
  
Lq\_bal<- quantile(BM$balance, probs=c(0.025))  
Hq\_bal<- quantile(BM$balance, probs=c(0.975))  
Lq\_dur<- quantile(BM$duration, probs=c(0.025))  
Hq\_dur<- quantile(BM$duration, probs=c(0.975))  
Lq\_cam<- quantile(BM$campaign, probs=c(0.025))  
Hq\_cam<- quantile(BM$campaign, probs=c(0.975))  
Lq\_days<- quantile(BM$pdays, probs=c(0.025))  
Hq\_days<- quantile(BM$pdays, probs=c(0.975))  
Lq\_prv<- quantile(BM$previous, probs=c(0.025))  
Hq\_prv<- quantile(BM$previous, probs=c(0.975))  
BM$balance[BM$balance < Lq\_bal] <- Lq\_bal  
BM$balance[BM$balance > Hq\_bal] <- Hq\_bal  
BM$duration[BM$duration < Lq\_dur] <- Lq\_dur  
BM$duration[BM$duration > Hq\_dur] <- Hq\_dur  
BM$campaign[BM$campaign < Lq\_cam] <- Lq\_cam  
BM$campaign[BM$campaign > Hq\_cam] <- Hq\_cam  
BM$pdays[BM$pdays < Lq\_days] <- Lq\_days  
BM$pdays[BM$pdays > Hq\_days] <- Hq\_days  
BM$previous[BM$previous < Lq\_prv] <- Lq\_prv  
BM$previous[BM$previous > Hq\_prv] <- Hq\_prv  
  
# now have file BM ...

### now make minor adjsutments

# switch -1 -> 0 in 'pdays'  
BM$pdays<- ifelse(BM$pdays == -1, 0, BM$pdays)  
# switch duration in seconds to minutes for easier use  
BM$duration<- BM$duration/60

### now switch categorical to get numeric - RESULT is “BM\_mini\_num”

# numeric still NOT normalized/scaled,   
# actually ALL DATA not normalized/scaled EXCEPT default, housing, loan  
BM\_n <- BM  
BM\_n$job<- as.numeric(BM\_n$job) #12  
# marital: 1-single, 2-married, 3-divorced  
BM\_n$marital<- ifelse(BM\_n$marital == c("single"), 1,   
 ifelse(BM\_n$marital== c("married"), 2, 3))  
# education: 0:unknown, 1: primary, 2:secondary, 3:divorced  
BM\_n$education<- ifelse(BM\_n$education == c("unknown"), 0,   
 ifelse(BM\_n$education == c("primary"), 1,   
 ifelse(BM\_n$education == c("secondary"), 2, 3)))  
# default, housing, loan: if yes then 0 else 1  
# BM\_num$housing<- as.numeric(BM\_num$housing) #2  
BM\_n$default<- ifelse(BM\_n$default == c("yes"), 0, 1) #2  
BM\_n$housing<- ifelse(BM\_n$housing == c("yes"), 0, 1) #2  
BM\_n$loan<- ifelse(BM\_n$loan == c("yes"), 0, 1) #2  
BM\_n$contact<- as.numeric(BM\_n$contact) #3  
# month: jan:1, feb:2.....dec:12  
BM\_n$month<- ifelse(BM\_n$month == "jan", 1,   
 ifelse(BM\_n$month == "feb", 2,   
 ifelse(BM\_n$month == "mar", 3,  
 ifelse(BM\_n$month == "apr", 4,   
 ifelse(BM\_n$month == "may", 5,   
 ifelse(BM\_n$month == "jun", 6,  
 ifelse(BM\_n$month == "jul", 7,  
 ifelse(BM\_n$month == "aug", 8,  
 ifelse(BM\_n$month == "sep", 9,  
 ifelse(BM\_n$month == "oct", 10,  
 ifelse(BM\_n$month == "nov", 11, 12)))))))))))  
# poutcome: 0:unknown,other, 1:failure, 2: success  
BM\_n$poutcome<- ifelse(BM\_n$poutcome == c("failure"), 1, ifelse(BM\_n$poutcome== c("success"), 2, 0))   
# result: BM\_num with only numeric data (NOT scaled)  
# extra step for BM\_mini  
# BM\_n

# 

### scale only original numeric data

# KEEP: age-1, balance-6, day-10, duration-12, campaign-13, pdays-14, previous-15  
BMS<- BM\_n  
# BMS<-BM\_mini  
normalize<- function(x) {  
 return ((x - min(x)) / (max(x) - min(x)))  
}  
BM\_s<- as.data.frame(lapply(BMS[,c(1,6,10,12:15)], normalize))  
# now recombine dataframes with the nominal components  
BM\_s$job<-BMS$job  
BM\_s$marital<-BMS$marital  
BM\_s$education<-BMS$education  
BM\_s$default<-BMS$default  
BM\_s$housing<-BMS$housing  
BM\_s$loan<-BMS$loan  
BM\_s$contact<-BMS$contact  
BM\_s$month<-BMS$month  
BM\_s$poutcome<-BMS$poutcome  
BM\_s$y<-BMS$y  
# convert  
BM\_mini\_sc<-BM\_s  
# result used BM\_num file but now BM\_num\_scale with normalized numeric data  
# and y is factor  
# to convert y to numeric use next line  
# BM\_scale$y<- ifelse(BM\_scale$y==c("yes"), 1, 0)  
rm(BMS)  
rm(BM\_s)  
rm(BM)  
rm(BM\_n)

## we are using SMOTE since it was found to be better so data will be rebalanced with SMOTE first

# step 2 - run training and test datasets FOR ALL CONFIGURATIONS

# str(BM\_mini\_sc)  
# rename datasets  
BM<-BM\_mini\_sc  
set.seed(30)  
# get train and test datasets  
# note that if we use cross validation, we can use the complete dataset for training or keep a portion for validation after  
#  
BM\_train\_index <- sample(nrow(BM), 0.7 \* nrow(BM))  
BM\_train<- BM[BM\_train\_index, ]  
BM\_test <- BM[-BM\_train\_index, ]  
BM\_train\_labels <- BM[BM\_train\_index, 17]  
BM\_test\_labels <- BM[-BM\_train\_index, 17]  
# note that we only balance the training sets  
# and leave test set as is.

# step 3a - run SPECIFIC Balancing step to get balanced data version:

## SMOTE SAMPLE MAJOR/MINOR - each for training

# install.packages("DMwR")  
library(DMwR)

## Loading required package: lattice

## Loading required package: grid

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

##   
## Attaching package: 'DMwR'

## The following object is masked from 'package:plyr':  
##   
## join

library(grid)  
# this one is slow  
set.seed(50)  
smote\_train <- SMOTE(y ~ ., data = BM\_train)   
table(smote\_train$y)

##   
## no yes   
## 1440 1080

# now we name datasets that are used in models below  
train=smote\_train  
  
BMX<- smote\_train  
data<- smote\_train  
RFEdata<-smote\_train  
BMX<-smote\_train  
dataset<- smote\_train  
# train<- smote\_train  
pca\_trainset = smote\_train   
BMY<- smote\_train

### #2 Recursive Feature Elimination with Random Forest

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. A Random Forest algorithm is used on each iteration to evaluate the model. Method = cross-validation, number=10-fold, repeated 10 times.

set.seed(7)  
library(mlbench)  
library(caret)

## Loading required package: ggplot2

# library(lattice)  
# library(ggplot2)  
# define the control using a random forest selection function rfFuncs  
control <- rfeControl(functions=rfFuncs, method="repeatedcv", number=10, repeats=10)  
# run the RFE algorithm  
results <- rfe(RFEdata[,1:16], RFEdata[,17], sizes=c(1:16), rfeControl=control)  
# summarize the results  
print(results)

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (10 fold, repeated 10 times)   
##   
## Resampling performance over subset size:  
##   
## Variables Accuracy Kappa AccuracySD KappaSD Selected  
## 1 0.7638 0.5068 0.02724 0.05722   
## 2 0.8481 0.6858 0.02156 0.04487   
## 3 0.8604 0.7135 0.02889 0.05883   
## 4 0.8861 0.7666 0.02270 0.04651   
## 5 0.8958 0.7868 0.02100 0.04272   
## 6 0.9040 0.8038 0.01913 0.03916   
## 7 0.9072 0.8108 0.01883 0.03842   
## 8 0.9124 0.8211 0.01923 0.03933   
## 9 0.9129 0.8221 0.01924 0.03928   
## 10 0.9164 0.8292 0.01780 0.03630   
## 11 0.9167 0.8298 0.01807 0.03692   
## 12 0.9177 0.8319 0.01767 0.03606   
## 13 0.9201 0.8369 0.01773 0.03621   
## 14 0.9210 0.8386 0.01768 0.03597   
## 15 0.9231 0.8429 0.01932 0.03943 \*  
## 16 0.9219 0.8406 0.01744 0.03554   
##   
## The top 5 variables (out of 15):  
## duration, month, day, contact, balance

# list the chosen features  
predictors(results)

## [1] "duration" "month" "day" "contact" "balance" "poutcome"   
## [7] "age" "job" "education" "campaign" "marital" "housing"   
## [13] "previous" "pdays" "loan"

# plot the results  
plot(results, type=c("g", "o"))

