All 7 models - nb, C5.0, JRip, glm, kknn, nnet, svmlinear

jean wills

13/07/2020

THIS FILE IS TO SEE EFFECTS OF FEATURE SELECTION METHOD x5

## step 1a - need to get to “BM\_mini\_sc” (added 95% CI + numeric scaled)

library(plyr)  
library(dplyr)  
BM\_mini <- read.csv("/Users/jeanwills/Desktop/CKME136/1\_data/bank.csv", header=T, sep = ";", stringsAsFactors = T, na.strings = "NA")  
# Step 1 - NUMERIC DATA Cleaning - change numeric data outside the 2.5% and the 97.5% percentiles to this maximum/minimum value  
BM<-BM\_mini  
  
Lq\_bal<- quantile(BM$balance, probs=c(0.025))  
Hq\_bal<- quantile(BM$balance, probs=c(0.975))  
#Lq\_bal # -393  
#Hq\_bal # 8969  
Lq\_dur<- quantile(BM$duration, probs=c(0.025))  
Hq\_dur<- quantile(BM$duration, probs=c(0.975))  
#Lq\_dur # 19  
#Hq\_dur # 986  
Lq\_cam<- quantile(BM$campaign, probs=c(0.025))  
Hq\_cam<- quantile(BM$campaign, probs=c(0.975))  
#Lq\_cam # 1  
#Hq\_cam # 11  
Lq\_days<- quantile(BM$pdays, probs=c(0.025))  
Hq\_days<- quantile(BM$pdays, probs=c(0.975))  
#Lq\_days # -1  
#Hq\_days # 356  
Lq\_prv<- quantile(BM$previous, probs=c(0.025))  
Hq\_prv<- quantile(BM$previous, probs=c(0.975))  
#Lq\_prv # 0  
#Hq\_prv # 5  
  
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 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 have file BM ..

## step 1b - BM -> BM\_mini\_sc"

# KEEP: age-1, balance-6, day-10, duration-12, campaign-13, pdays-14, previous-15  
BMS<- BM  
# 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  
rm(BMS)  
rm(BM\_s)  
rm(BM)

## Step 1c - NOW DELETE THE ATTRIBUTES HERE AND CHANGE ANY data where there is [-17]

temp<-BM\_mini\_sc  
  
  
# FS4: ANOVA - keep only 7: # 4, 8, 10, 11, 14, 15, 16 => delete 13,12,9,7,6,5,3,2,1  
temp<-BM\_mini\_sc  
temp<- temp[-13]  
temp<- temp[-12]  
temp<- temp[-9]  
temp<- temp[-7]  
temp<- temp[-6]  
temp<- temp[-5]  
temp<- temp[-3]  
temp<- temp[-2]  
temp<- temp[-1]  
  
BM<- temp  
summary(BM)

## duration job education default   
## Min. :0.0000 management :969 primary : 678 no :4445   
## 1st Qu.:0.0879 blue-collar:946 secondary:2306 yes: 76   
## Median :0.1717 technician :768 tertiary :1350   
## Mean :0.2450 admin. :478 unknown : 187   
## 3rd Qu.:0.3206 services :417   
## Max. :1.0000 retired :230   
## (Other) :713   
## contact month poutcome y   
## cellular :2896 may :1398 failure: 490 no :4000   
## telephone: 301 jul : 706 other : 197 yes: 521   
## unknown :1324 aug : 633 success: 129   
## jun : 531 unknown:3705   
## nov : 389   
## apr : 293   
## (Other): 571

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

# str(BM\_mini\_sc)  
# rename dataset here:  
# BM<- BM\_mini\_sc  
set.seed(30)  
# get train and test datasets  
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 3 RUN SMOTE for training

# install.packages("DMwR")  
library(DMwR)  
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   
# \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
x=smote\_train[,-8]  
trainsv=smote\_train  
train=smote\_train  
y=smote\_train$y  
test\_noy=BM\_test[,-8]  
test\_labels=BM\_test$y

# step 4 - now run models

## MODEL 1 NAIVE BAYES - uses caret

# FINAL PARAMETER VALUES USED WERE fL = 0, usekernel = TRUE and adjust = 1.  
library(klaR)  
library(caret)  
library(e1071)  
library(MASS)  
set.seed(520)  
nb\_grid <- expand.grid(fL= 0, usekernel= c("TRUE"), adjust=1)  
# nb\_grid <- expand.grid(fL= c(0,1), usekernel= c("TRUE", "FALSE"), adjust=c(0,1,2,3))  
nab\_mod<- train(x=x, y=y, method="nb", metric="ROC", tuneGrid=nb\_grid, trControl = trainControl(method="repeatedcv", number=10, repeats=10, classProbs=TRUE, summaryFunction = twoClassSummary, verboseIter=FALSE))  
# to see model results:  
# nab\_mod  
# predict output using test set - even though we did 10x10 cv above, we are also using 'validation' set of 30% of the data:  
nab\_pred<- predict(nab\_mod, test\_noy)  
s<-table(nab\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## nab\_pred no yes  
## no 1038 62  
## yes 158 99  
##   
## Accuracy : 0.8379   
## 95% CI : (0.8172, 0.8571)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3838   
##   
## Mcnemar's Test P-Value : 1.505e-10   
##   
## Sensitivity : 0.8679   
## Specificity : 0.6149   
## Pos Pred Value : 0.9436   
## Neg Pred Value : 0.3852   
## Prevalence : 0.8814   
## Detection Rate : 0.7649   
## Detection Prevalence : 0.8106   
## Balanced Accuracy : 0.7414   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nab\_mod, test[-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.4728056

# print the average of the 10 F1 results for test set

## model 2 - C5.0 Decision Tree algorithm

strengths: numeric/nominal, easy interpretation weaknesses: biased splits, overfitting

# The final tuning parameters used for the original model were trials = 1, model = tree and winnow = FALSE.  
# Trials = an integer specifying the number of boosting iterations. A value of one indicates that a single model is used.  
# winnow: A logical: should predictor winnowing (i.e feature selection) be used.  
library(caret)  
set.seed(40)  
# c5\_grid <- expand.grid(trials=c(1,3,5), model = c("tree", "rules"), winnow = c(TRUE, FALSE))  
c5\_grid <- expand.grid(trials = 1, model = "tree", winnow = FALSE)  
c5\_ctrl<- trainControl(method="repeatedcv", number = 10, repeats=10, classProbs=TRUE, summaryFunction = twoClassSummary)  
c5\_mod<- train(x,y, method="C5.0", metric="ROC", tuneGrid=c5\_grid, trControl = c5\_ctrl, verbose=FALSE)  
c\_pred<- predict(c5\_mod, test\_noy)  
# Testing the result output  
s<-table(c\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## c\_pred no yes  
## no 1004 52  
## yes 192 109  
##   
## Accuracy : 0.8202   
## 95% CI : (0.7987, 0.8403)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3753   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8395   
## Specificity : 0.6770   
## Pos Pred Value : 0.9508   
## Neg Pred Value : 0.3621   
## Prevalence : 0.8814   
## Detection Rate : 0.7399   
## Detection Prevalence : 0.7782   
## Balanced Accuracy : 0.7582   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(c5\_mod, test[,-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.478013

# print the average of the 10 F1 results for test set

### to see model results:

#c5\_mod  
#c5\_mod$finalModel  
# another way to see the tree's decisions and best attributes:  
summary(c5\_mod)

##   
## Call:  
## (function (x, y, trials = 1, rules = FALSE, weights = NULL, control  
## = FALSE, sample = 0, earlyStopping = TRUE, label = "outcome", seed =  
## 2372L), verbose = FALSE)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Wed Jul 29 00:08:59 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 2520 cases (8 attributes) from undefined.data  
##   
## Decision tree:  
##   
## duration <= 0.1706308:  
## :...month in {mar,oct}: yes (43/12)  
## : month in {apr,aug,dec,feb,jan,jul,jun,may,nov,sep}:  
## : :...poutcome in {failure,unknown}: no (781/39)  
## : poutcome = success: yes (20/6)  
## : poutcome = other:  
## : :...contact in {cellular,telephone}: no (44/7)  
## : contact = unknown:  
## : :...month in {apr,aug,dec,feb,jan,jul,jun,may,sep}: yes (5)  
## : month = nov: no (3)  
## duration > 0.1706308:  
## :...poutcome in {other,success}:  
## :...contact in {telephone,unknown}: yes (113/1)  
## : contact = cellular:  
## : :...poutcome = success: yes (148/10)  
## : poutcome = other:  
## : :...duration <= 0.2699069: no (22/8)  
## : duration > 0.2699069: yes (58/4)  
## poutcome in {failure,unknown}:  
## :...duration > 0.5522234:  
## :...default = yes: yes (11)  
## : default = no:  
## : :...contact in {cellular,telephone}: yes (301/59)  
## : contact = unknown:  
## : :...poutcome = failure: yes (7)  
## : poutcome = unknown:  
## : :...duration > 0.9820951: yes (29/2)  
## : duration <= 0.9820951:  
## : :...month in {apr,feb,jan,mar,nov,oct}: yes (12)  
## : month in {aug,dec,jul,jun,may,sep}: no (50/18)  
## duration <= 0.5522234:  
## :...month in {mar,oct,sep}: yes (89/10)  
## month in {apr,aug,dec,feb,jan,jul,jun,may,nov}:  
## :...contact = telephone:  
## :...month in {aug,dec,feb,jun,may}: yes (49/8)  
## : month = jan: no (2/1)  
## : month = apr:  
## : :...education in {primary,tertiary,unknown}: yes (6)  
## : : education = secondary: no (3)  
## : month = jul:  
## : :...duration <= 0.2585315: no (10)  
## : : duration > 0.2585315: yes (17/1)  
## : month = nov:  
## : :...duration <= 0.2099276: no (2)  
## : duration > 0.2099276: yes (4)  
## contact in {cellular,unknown}:  
## :...job in {admin.,entrepreneur,housemaid,management,retired,  
## : self-employed,student,unemployed,unknown}:  
## :...contact = unknown:  
## : :...month in {dec,feb,jan}: no (0)  
## : : month in {apr,aug,nov}: yes (8)  
## : : month in {jul,jun,may}:  
## : : :...duration <= 0.3733195: no (58/1)  
## : : duration > 0.3733195:  
## : : :...education in {primary,  
## : : : tertiary}: yes (13/4)  
## : : education in {secondary,  
## : : unknown}: no (10/2)  
## : contact = cellular:  
## : :...month in {jan,jul}:  
## : :...job in {admin.,entrepreneur,self-employed,  
## : : : unemployed}: no (17/2)  
## : : job in {student,unknown}: yes (3)  
## : : job = housemaid:  
## : : :...duration <= 0.3619442: no (4)  
## : : : duration > 0.3619442: yes (2)  
## : : job = retired:  
## : : :...education in {secondary,unknown}: yes (2)  
## : : : education in {primary,tertiary}: no (4)  
## : : job = management:  
## : : :...duration <= 0.3288521: no (18)  
## : : duration > 0.3288521:  
## : : :...duration <= 0.4866104: yes (5)  
## : : duration > 0.4866104: no (2)  
## : month in {apr,aug,dec,feb,jun,may,nov}:  
## : :...education in {primary,unknown}: yes (37/9)  
## : education = secondary:  
## : :...job in {admin.,housemaid,management,  
## : : : unknown}: yes (33/6)  
## : : job = entrepreneur:  
## : : :...poutcome = failure: yes (2)  
## : : : poutcome = unknown: no (5/1)  
## : : job = retired:  
## : : :...poutcome = failure: no (3)  
## : : : poutcome = unknown: yes (11/5)  
## : : job = self-employed:  
## : : :...month in {apr,aug,dec,feb,jun,  
## : : : : may}: yes (4)  
## : : : month = nov: no (2)  
## : : job = student:  
## : : :...duration <= 0.4216879: yes (6)  
## : : : duration > 0.4216879: no (2)  
## : : job = unemployed:  
## : : :...duration <= 0.2502585: yes (2)  
## : : duration > 0.2502585: no (3)  
## : education = tertiary:  
## : :...default = yes: no (2)  
## : default = no:  
## : :...month = dec: no (2)  
## : month in {feb,may}: yes (31/10)  
## : month = apr:  
## : :...poutcome = failure: no (4)  
## : : poutcome = unknown: yes (6)  
## : month = aug:  
## : :...poutcome = failure: yes (3/1)  
## : : poutcome = unknown: no (24/4)  
## : month = jun:  
## : :...duration <= 0.2202689: no (2)  
## : : duration > 0.2202689: yes (8)  
## : month = nov:  
## : :...poutcome = failure: yes (4)  
## : poutcome = unknown: no (8/1)  
## job in {blue-collar,services,technician}:  
## :...contact = unknown:  
## :...month in {apr,aug,dec,feb,jan}: no (0)  
## : month = nov: yes (2)  
## : month in {jul,jun,may}:  
## : :...duration <= 0.434333: no (94)  
## : duration > 0.434333:  
## : :...month = jun: no (0)  
## : month = jul: yes (1)  
## : month = may:  
## : :...duration <= 0.4601862: yes (2)  
## : duration > 0.4601862: no (12/2)  
## contact = cellular:  
## :...duration <= 0.2008211: no (34)  
## duration > 0.2008211:  
## :...education in {primary,secondary,  
## : unknown}: no (165/30)  
## education = tertiary:  
## :...job = blue-collar: no (2/1)  
## job = services: yes (3)  
## job = technician:  
## :...duration <= 0.4157187:  
## :...duration <= 0.2202689: yes (4)  
## : duration > 0.2202689: no (18/4)  
## duration > 0.4157187:  
## :...month = aug: no (1)  
## month in {apr,dec,feb,jan,jul,jun,  
## may,nov}: yes (3)  
##   
##   
## Evaluation on training data (2520 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 74 269(10.7%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 1292 148 (a): class no  
## 121 959 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% duration  
## 98.29% poutcome  
## 72.66% month  
## 62.54% contact  
## 27.42% job  
## 20.00% default  
## 17.38% education  
##   
##   
## Time: 0.0 secs

# we see the decision trees and....  
# ... and best attributes: see below at bottom....

## model 3 - JRip (rule learner)

# The final values used for the model were NumOpt = 10, NumFolds = 10 and MinWeights = 10.  
library(caret)  
library(RWeka)  
set.seed(50)  
# JR\_grid <- expand.grid(NumOpt=c(1,3,5,10), NumFolds=c(1,3,5,10),MinWeights=c(1,3,5,10))  
JR\_grid <- expand.grid(NumOpt=10, NumFolds=10, MinWeights=10)  
JR\_ctrl<- trainControl(method="repeatedcv", number = 10, repeats = 10, classProbs=TRUE, summaryFunction = twoClassSummary)  
JR\_mod<- train(x, y, method="JRip", metric="ROC", tuneGrid=JR\_grid, trControl = JR\_ctrl)  
JR\_pred<- predict(JR\_mod, test\_noy)  
s<-table(JR\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## JR\_pred no yes  
## no 1019 39  
## yes 177 122  
##   
## Accuracy : 0.8408   
## 95% CI : (0.8203, 0.8599)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4448   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8520   
## Specificity : 0.7578   
## Pos Pred Value : 0.9631   
## Neg Pred Value : 0.4080   
## Prevalence : 0.8814   
## Detection Rate : 0.7509   
## Detection Prevalence : 0.7797   
## Balanced Accuracy : 0.8049   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(JR\_mod, test[-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.5323727

# print the average of the 10 F1 results for test set

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.452947) => .outcome=yes (657.0/158.0)  
## (duration >= 0.170925) and (poutcome = success) => .outcome=yes (151.0/4.0)  
## (duration >= 0.165783) and (contact = telephone) => .outcome=yes (117.0/25.0)  
## (contact = cellular) and (duration >= 0.276519) and (poutcome = other) => .outcome=yes (24.0/0.0)  
## (duration >= 0.165783) and (contact = cellular) and (month = mar) => .outcome=yes (21.0/0.0)  
## (duration >= 0.149948) and (month = oct) => .outcome=yes (38.0/4.0)  
## (duration >= 0.161356) and (contact = cellular) and (month = jun) and (poutcome = unknown) => .outcome=yes (17.0/1.0)  
## (duration >= 0.162584) and (contact = cellular) and (job = student) => .outcome=yes (14.0/3.0)  
## (month = mar) => .outcome=yes (27.0/8.0)  
## (contact = cellular) and (duration >= 0.336091) and (poutcome = failure) => .outcome=yes (18.0/4.0)  
## (duration >= 0.169764) and (month = apr) and (education = tertiary) => .outcome=yes (14.0/3.0)  
## => .outcome=no (1422.0/192.0)  
##   
## Number of Rules : 12

# all for outcome yes

## model 4 - Logistic Regression

# no parameters  
library(caret)  
set.seed(520)  
log\_mod<- train(x, y, method="glm", metric="ROC", family=binomial(link="logit"), trControl = trainControl(method="repeatedcv", number=10, repeats=10, verboseIter=FALSE, classProbs = TRUE, summaryFunction = twoClassSummary))  
log\_pred<- predict(log\_mod, test\_noy)  
s<-table(log\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## log\_pred no yes  
## no 1047 46  
## yes 149 115  
##   
## Accuracy : 0.8563   
## 95% CI : (0.8365, 0.8745)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9976   
##   
## Kappa : 0.4619   
##   
## Mcnemar's Test P-Value : 2.786e-13   
##   
## Sensitivity : 0.8754   
## Specificity : 0.7143   
## Pos Pred Value : 0.9579   
## Neg Pred Value : 0.4356   
## Prevalence : 0.8814   
## Detection Rate : 0.7716   
## Detection Prevalence : 0.8055   
## Balanced Accuracy : 0.7949   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(log\_mod, test[-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.5422712

# print the average of the 10 F1 results for test set

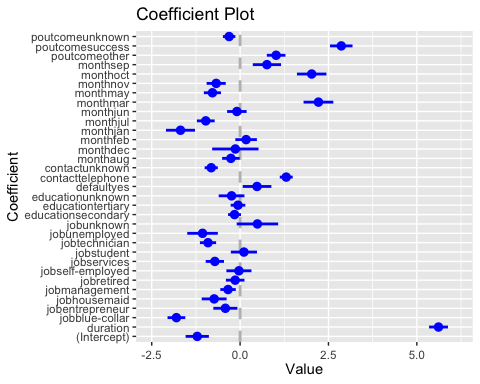
### model results

# to see model results:  
summary(log\_mod)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.7344 -0.5552 -0.2638 0.4903 2.6323   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.21221 0.32905 -3.684 0.000230 \*\*\*  
## duration 5.61303 0.26710 21.015 < 2e-16 \*\*\*  
## jobblue-collar -1.80158 0.25093 -7.179 7.00e-13 \*\*\*  
## jobentrepreneur -0.41652 0.34109 -1.221 0.222024   
## jobhousemaid -0.73375 0.35261 -2.081 0.037440 \*   
## jobmanagement -0.34063 0.21928 -1.553 0.120332   
## jobretired -0.13918 0.26297 -0.529 0.596607   
## jobself-employed -0.03319 0.35439 -0.094 0.925393   
## jobservices -0.71418 0.25859 -2.762 0.005749 \*\*   
## jobstudent 0.10810 0.36944 0.293 0.769831   
## jobtechnician -0.90766 0.22916 -3.961 7.47e-05 \*\*\*  
## jobunemployed -1.06245 0.43222 -2.458 0.013967 \*   
## jobunknown 0.49041 0.58700 0.835 0.403459   
## educationsecondary -0.15741 0.18281 -0.861 0.389191   
## educationtertiary -0.05895 0.20700 -0.285 0.775824   
## educationunknown -0.24098 0.36408 -0.662 0.508040   
## defaultyes 0.47883 0.40568 1.180 0.237877   
## contacttelephone 1.30536 0.18284 7.139 9.37e-13 \*\*\*  
## contactunknown -0.81712 0.18577 -4.398 1.09e-05 \*\*\*  
## monthaug -0.26079 0.24821 -1.051 0.293402   
## monthdec -0.13413 0.65594 -0.204 0.837978   
## monthfeb 0.17036 0.30489 0.559 0.576323   
## monthjan -1.68608 0.41100 -4.102 4.09e-05 \*\*\*  
## monthjul -0.97109 0.25126 -3.865 0.000111 \*\*\*  
## monthjun -0.09092 0.27596 -0.329 0.741807   
## monthmar 2.21637 0.42135 5.260 1.44e-07 \*\*\*  
## monthmay -0.78296 0.24189 -3.237 0.001209 \*\*   
## monthnov -0.67634 0.27028 -2.502 0.012338 \*   
## monthoct 2.02502 0.41703 4.856 1.20e-06 \*\*\*  
## monthsep 0.75927 0.40036 1.896 0.057898 .   
## poutcomeother 1.01906 0.26333 3.870 0.000109 \*\*\*  
## poutcomesuccess 2.86258 0.31828 8.994 < 2e-16 \*\*\*  
## poutcomeunknown -0.31014 0.17704 -1.752 0.079811 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 3441.9 on 2519 degrees of freedom  
## Residual deviance: 1906.9 on 2487 degrees of freedom  
## AIC: 1972.9  
##   
## Number of Fisher Scoring iterations: 6

### plot to see the most important attributes (those that “stand out” at far left or right)

require(coefplot)  
coefplot(log\_mod)



# to reinterpret data properly  
# invlogit<- function (x) {1/(1+exp(-x))}  
# invlogit(log\_mod$coefficients)  
# results of plot: duration, campaign, balance, poutcomesuccess, some months, , some jobs

## Model #5 - K-Nearest Neighbours

# The final values used for the model were kmax = 9, distance = 2 and kernel = optimal  
library(caret)  
library(lattice)  
library(ggplot2)  
knn\_ctrl<- trainControl(method="repeatedcv", number = 10, repeats = 10, classProbs=TRUE, summaryFunction = twoClassSummary)  
# knn\_grid<-expand.grid(kmax=c(5,7,9,13), distance=c(1,2,4,6), kernel=c("rectangular","rank","optimal"))  
knn\_grid<-expand.grid(kmax=9, distance=2, kernel="optimal")  
knn\_mod<- train(x=x,y=y, method="kknn", metric="ROC", tuneGrid=knn\_grid, trControl=knn\_ctrl, verbose=FALSE)  
knn\_pred<- predict(knn\_mod, test\_noy)  
# Testing the result output  
s<-table( knn\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## knn\_pred no yes  
## no 982 57  
## yes 214 104  
##   
## Accuracy : 0.8003   
## 95% CI : (0.778, 0.8213)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3284   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8211   
## Specificity : 0.6460   
## Pos Pred Value : 0.9451   
## Neg Pred Value : 0.3270   
## Prevalence : 0.8814   
## Detection Rate : 0.7237   
## Detection Prevalence : 0.7657   
## Balanced Accuracy : 0.7335   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(knn\_mod, test[-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.4320071

# print the average of the 10 F1 results for test set

## NEURAL NET - model 6

Positive: can be used for classification or numeric prediction, makes few assumptions about the data (doesnt have to be normalized). Negative: SLOW..can overfit, ‘black box’.

# #1: The final parameter values used for the model were size = 16 and decay = 0.1.  
# Size is the number of units in hidden layer (nnet fit a single hidden layer neural network) and   
# decay is the regularization parameter to avoid over-fitting.  
require(mlbench)  
require(caret)  
require (nnet)  
nnctrl = trainControl(method="repeatedcv", number=10, repeats = 10, classProbs=TRUE, summaryFunction = twoClassSummary)  
# initially, create a grid list to find best parameters:  
# nn\_grid = expand.grid(size=c(1,4,8,16),decay=c(0,0.1,0.2,0.3,0.4))  
nn\_grid = expand.grid(size=16, decay=0.1)  
nn\_mod <- train(x=x, y=y, method="nnet", metric="ROC", trControl=nnctrl, tuneGrid=nn\_grid, trace=FALSE)  
nn\_pred<- predict(nn\_mod, test\_noy)  
# Testing the result output  
s<-table( nn\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## nn\_pred no yes  
## no 1020 51  
## yes 176 110  
##   
## Accuracy : 0.8327   
## 95% CI : (0.8118, 0.8522)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4013   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8528   
## Specificity : 0.6832   
## Pos Pred Value : 0.9524   
## Neg Pred Value : 0.3846   
## Prevalence : 0.8814   
## Detection Rate : 0.7517   
## Detection Prevalence : 0.7892   
## Balanced Accuracy : 0.7680   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nn\_mod, test[-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.4888361

# print the average of the 10 F1 results for test set

## SVM model 7 - using caret package and linear kernel.

### Another supervised learning model. A SVM can be imagined as a surface thatc reates a boundary between points of data plotted in an n-space representing examples and their feature values. SVM creates a flat boundary called a hyperplane, which divides the space into homogeneous partitions on either side. This way, SVM combines nearest neighbour instance-based learning with linear regression modeling.

negatives: need to test various parameters and kernels to get best solution, can be slow, ’black box". positives: good for binary classification, classification/numeric prediction

# Linear (vanilla) kernel function.   
# The final values used for the model were Cost = 1 for classification.  
library(caret)  
library(kernlab)  
fitctrl<- trainControl(method="repeatedcv", number = 10, repeats = 10, classProbs=TRUE, summaryFunction = twoClassSummary)  
grid<-expand.grid(C=1)  
sv\_m<- train(y~., data=trainsv, method="svmLinear", metric="ROC", trControl=fitctrl, tunegrid=grid, trace=FALSE)  
sp<- predict(sv\_m, test\_noy)  
s<-table( sp, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## sp no yes  
## no 1053 50  
## yes 143 111  
##   
## Accuracy : 0.8578   
## 95% CI : (0.838, 0.8759)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9962   
##   
## Kappa : 0.4559   
##   
## Mcnemar's Test P-Value : 3.536e-11   
##   
## Sensitivity : 0.8804   
## Specificity : 0.6894   
## Pos Pred Value : 0.9547   
## Neg Pred Value : 0.4370   
## Prevalence : 0.8814   
## Detection Rate : 0.7760   
## Detection Prevalence : 0.8128   
## Balanced Accuracy : 0.7849   
##   
## 'Positive' Class : no   
##

### we can now run 10-fold on test dataset:

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(sv\_m, test[-8])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.5397985

# print the average of the 10 F1 results for test set