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

jean wills

13/07/2020

# 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  
# now have file BM ..

## step 1b - BM -> BM\_scale or 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  
# 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)

# 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  
# 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:

## for unbalanced, rename the files to make it easy….

table(BM\_train$y)

##   
## no yes   
## 2804 360

x=BM\_train[,-17]  
trainsv=BM\_train  
train=BM\_train  
y=BM\_train$y  
test\_noy=BM\_test[,-17]  
test\_labels=BM\_test$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))  
# 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 1140 111  
## yes 56 50  
##   
## Accuracy : 0.8769   
## 95% CI : (0.8583, 0.894)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.7099   
##   
## Kappa : 0.3095   
##   
## Mcnemar's Test P-Value : 2.933e-05   
##   
## Sensitivity : 0.9532   
## Specificity : 0.3106   
## Pos Pred Value : 0.9113   
## Neg Pred Value : 0.4717   
## Prevalence : 0.8814   
## Detection Rate : 0.8401   
## Detection Prevalence : 0.9219   
## Balanced Accuracy : 0.6319   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(nab\_mod, test[-17])  
 actual<- test$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] 0.3630216

# 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 1140 95  
## yes 56 66  
##   
## Accuracy : 0.8887   
## 95% CI : (0.8708, 0.905)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.213699   
##   
## Kappa : 0.4056   
##   
## Mcnemar's Test P-Value : 0.001985   
##   
## Sensitivity : 0.9532   
## Specificity : 0.4099   
## Pos Pred Value : 0.9231   
## Neg Pred Value : 0.5410   
## Prevalence : 0.8814   
## Detection Rate : 0.8401   
## Detection Prevalence : 0.9101   
## Balanced Accuracy : 0.6816   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(c5\_mod, test[-17])  
 actual<- test$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] 0.4674824

# 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  
## 0.25, minCases = 2, fuzzyThreshold = FALSE, sample = 0, earlyStopping =  
## TRUE, label = "outcome", seed = 2372L), verbose = FALSE)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Tue Jul 28 23:14:53 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 3164 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## poutcome = success:  
## :...balance > 0.146951: no (5)  
## : balance <= 0.146951:  
## : :...duration > 0.05627276: yes (68/15)  
## : duration <= 0.05627276:  
## : :...housing = yes: no (7)  
## : housing = no:  
## : :...age <= 0.3676471: yes (6/1)  
## : age > 0.3676471: no (5)  
## poutcome in {failure,other,unknown}:  
## :...duration > 0.122807:  
## :...duration <= 0.2121814: no (382/75)  
## : duration > 0.2121814:  
## : :...contact in {telephone,unknown}:  
## : :...age > 0.5882353: yes (5)  
## : : age <= 0.5882353:  
## : : :...marital in {divorced,single}: yes (27/13)  
## : : marital = married: no (50/9)  
## : contact = cellular:  
## : :...job in {entrepreneur,unknown}: no (10/4)  
## : job in {self-employed,services,technician}: yes (38/8)  
## : job = admin.:  
## : :...marital in {divorced,single}: yes (8/2)  
## : : marital = married: no (3)  
## : job = housemaid:  
## : :...day <= 0.4666667: no (3)  
## : : day > 0.4666667: yes (2)  
## : job = retired:  
## : :...poutcome = failure: yes (1)  
## : : poutcome in {other,unknown}: no (3)  
## : job = student:  
## : :...age <= 0.1470588: yes (2)  
## : : age > 0.1470588: no (2)  
## : job = unemployed:  
## : :...balance <= 0.05067046: no (3)  
## : : balance > 0.05067046: yes (2)  
## : job = blue-collar:  
## : :...month in {aug,jun,may}: yes (10/2)  
## : : month in {dec,feb,jan,mar,oct,sep}: no (3)  
## : : month = apr:  
## : : :...age <= 0.2205882: no (3)  
## : : : age > 0.2205882: yes (3)  
## : : month = jul:  
## : : :...age <= 0.3382353: yes (2)  
## : : : age > 0.3382353: no (4)  
## : : month = nov:  
## : : :...age <= 0.25: no (3)  
## : : age > 0.25: yes (2)  
## : job = management:  
## : :...month in {apr,jun,nov}: no (9/3)  
## : month in {dec,feb,jan,mar,oct,sep}: yes (2)  
## : month = may:  
## : :...balance <= 0.04877787: no (3)  
## : : balance > 0.04877787: yes (3)  
## : month = aug:  
## : :...duration > 0.2634889: yes (7)  
## : : duration <= 0.2634889:  
## : : :...day <= 0.5333334: yes (2)  
## : : day > 0.5333334: no (3)  
## : month = jul:  
## : :...campaign > 0.04081633: yes (4)  
## : campaign <= 0.04081633:  
## : :...balance <= 0.04907317: yes (3/1)  
## : balance > 0.04907317: no (3)  
## duration <= 0.122807:  
## :...month in {aug,jan,jul,jun,may,nov}: no (2121/54)  
## month in {mar,oct}:  
## :...duration <= 0.02780536: no (11)  
## : duration > 0.02780536:  
## : :...job in {blue-collar,entrepreneur,retired,self-employed,  
## : : unknown}: no (14/3)  
## : job in {housemaid,services,student,unemployed}: yes (3)  
## : job = admin.:  
## : :...duration <= 0.04601125: no (4)  
## : : duration > 0.04601125: yes (7/1)  
## : job = management:  
## : :...month = mar: yes (5)  
## : : month = oct: no (10/4)  
## : job = technician:  
## : :...marital in {divorced,married}: yes (5)  
## : marital = single: no (2)  
## month in {apr,dec,feb,sep}:  
## :...day > 0.6333333:  
## :...duration > 0.07778881: yes (12/1)  
## : duration <= 0.07778881:  
## : :...balance <= 0.04460343: yes (3)  
## : balance > 0.04460343: no (28/1)  
## day <= 0.6333333:  
## :...job in {admin.,blue-collar,housemaid,services,student,  
## : unknown}: no (113)  
## job in {entrepreneur,management,retired,self-employed,  
## : technician,unemployed}:  
## :...month in {apr,dec,sep}: no (64/9)  
## month = feb:  
## :...day > 0.2666667: yes (7)  
## day <= 0.2666667:  
## :...balance <= 0.07318023: no (45)  
## balance > 0.07318023:  
## :...pdays <= 0.03325688: no (7/1)  
## pdays > 0.03325688: yes (2)  
##   
##   
## Evaluation on training data (3164 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 58 207( 6.5%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 2760 44 (a): class no  
## 163 197 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% poutcome  
## 99.84% duration  
## 80.03% month  
## 13.72% job  
## 9.20% day  
## 7.21% contact  
## 6.10% balance  
## 3.60% age  
## 3.00% marital  
## 0.57% housing  
## 0.32% campaign  
## 0.28% pdays  
##   
##   
## 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 1160 108  
## yes 36 53  
##   
## Accuracy : 0.8939   
## 95% CI : (0.8763, 0.9098)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.08141   
##   
## Kappa : 0.3709   
##   
## Mcnemar's Test P-Value : 3.285e-09   
##   
## Sensitivity : 0.9699   
## Specificity : 0.3292   
## Pos Pred Value : 0.9148   
## Neg Pred Value : 0.5955   
## Prevalence : 0.8814   
## Detection Rate : 0.8548   
## Detection Prevalence : 0.9344   
## Balanced Accuracy : 0.6495   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(JR\_mod, test[-17])  
 actual<- test$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] 0.4044588

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.214167) and (contact = cellular) => .outcome=yes (152.0/64.0)  
## (duration >= 0.071831) and (pdays >= 0.044725) and (poutcome = success) => .outcome=yes (55.0/13.0)  
## => .outcome=no (2957.0/230.0)  
##   
## Number of Rules : 3

# 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 1165 104  
## yes 31 57  
##   
## Accuracy : 0.9005   
## 95% CI : (0.8833, 0.9159)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.01455   
##   
## Kappa : 0.4082   
##   
## Mcnemar's Test P-Value : 5.763e-10   
##   
## Sensitivity : 0.9741   
## Specificity : 0.3540   
## Pos Pred Value : 0.9180   
## Neg Pred Value : 0.6477   
## Prevalence : 0.8814   
## Detection Rate : 0.8585   
## Detection Prevalence : 0.9352   
## Balanced Accuracy : 0.6641   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(log\_mod, test[-17])  
 actual<- test$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] 0.4424991

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

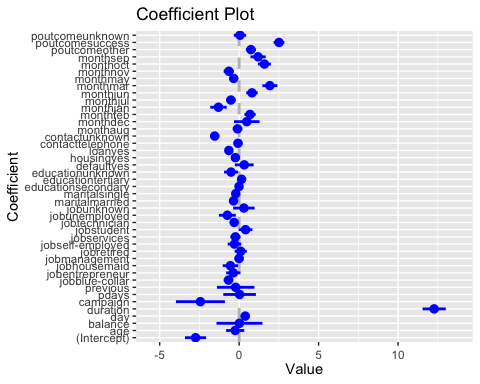
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.8382 -0.3827 -0.2576 -0.1565 3.0061   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.741812 0.662354 -4.139 3.48e-05 \*\*\*  
## age -0.248271 0.571745 -0.434 0.6641   
## balance 0.018772 1.444250 0.013 0.9896   
## day 0.384633 0.293461 1.311 0.1900   
## duration 12.273995 0.730255 16.808 < 2e-16 \*\*\*  
## campaign -2.437929 1.540679 -1.582 0.1136   
## pdays 0.029031 1.026607 0.028 0.9774   
## previous -0.219749 1.178410 -0.186 0.8521   
## jobblue-collar -0.669800 0.299439 -2.237 0.0253 \*   
## jobentrepreneur -0.369725 0.446793 -0.828 0.4079   
## jobhousemaid -0.549748 0.481015 -1.143 0.2531   
## jobmanagement -0.004758 0.280455 -0.017 0.9865   
## jobretired 0.098758 0.383480 0.258 0.7968   
## jobself-employed -0.294752 0.428815 -0.687 0.4919   
## jobservices -0.232068 0.319163 -0.727 0.4672   
## jobstudent 0.402220 0.433046 0.929 0.3530   
## jobtechnician -0.315138 0.270599 -1.165 0.2442   
## jobunemployed -0.741155 0.531415 -1.395 0.1631   
## jobunknown 0.297507 0.676110 0.440 0.6599   
## maritalmarried -0.352923 0.215364 -1.639 0.1013   
## maritalsingle -0.212932 0.247547 -0.860 0.3897   
## educationsecondary -0.007986 0.249498 -0.032 0.9745   
## educationtertiary 0.152811 0.285995 0.534 0.5931   
## educationunknown -0.510559 0.439903 -1.161 0.2458   
## defaultyes 0.320166 0.591793 0.541 0.5885   
## housingyes -0.237956 0.165311 -1.439 0.1500   
## loanyes -0.648523 0.231041 -2.807 0.0050 \*\*   
## contacttelephone -0.072272 0.277719 -0.260 0.7947   
## contactunknown -1.537566 0.283897 -5.416 6.10e-08 \*\*\*  
## monthaug -0.102446 0.299177 -0.342 0.7320   
## monthdec 0.480674 0.804370 0.598 0.5501   
## monthfeb 0.686165 0.355927 1.928 0.0539 .   
## monthjan -1.301680 0.516327 -2.521 0.0117 \*   
## monthjul -0.519478 0.301410 -1.723 0.0848 .   
## monthjun 0.809638 0.363638 2.226 0.0260 \*   
## monthmar 1.933844 0.475576 4.066 4.78e-05 \*\*\*  
## monthmay -0.344559 0.287747 -1.197 0.2311   
## monthnov -0.649344 0.327638 -1.982 0.0475 \*   
## monthoct 1.589671 0.407647 3.900 9.63e-05 \*\*\*  
## monthsep 1.192695 0.480156 2.484 0.0130 \*   
## poutcomeother 0.741236 0.316602 2.341 0.0192 \*   
## poutcomesuccess 2.508580 0.335294 7.482 7.34e-14 \*\*\*  
## poutcomeunknown 0.051914 0.385931 0.135 0.8930   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2242.3 on 3163 degrees of freedom  
## Residual deviance: 1521.2 on 3121 degrees of freedom  
## AIC: 1607.2  
##   
## 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 1171 113  
## yes 25 48  
##   
## Accuracy : 0.8983   
## 95% CI : (0.881, 0.9139)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.02753   
##   
## Kappa : 0.3631   
##   
## Mcnemar's Test P-Value : 1.302e-13   
##   
## Sensitivity : 0.9791   
## Specificity : 0.2981   
## Pos Pred Value : 0.9120   
## Neg Pred Value : 0.6575   
## Prevalence : 0.8814   
## Detection Rate : 0.8629   
## Detection Prevalence : 0.9462   
## Balanced Accuracy : 0.6386   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(knn\_mod, test[-17])  
 actual<- test$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] NaN

# 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 1132 99  
## yes 64 62  
##   
## Accuracy : 0.8799   
## 95% CI : (0.8614, 0.8967)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.587083   
##   
## Kappa : 0.366   
##   
## Mcnemar's Test P-Value : 0.007743   
##   
## Sensitivity : 0.9465   
## Specificity : 0.3851   
## Pos Pred Value : 0.9196   
## Neg Pred Value : 0.4921   
## Prevalence : 0.8814   
## Detection Rate : 0.8342   
## Detection Prevalence : 0.9071   
## Balanced Accuracy : 0.6658   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(nn\_mod, test[-17])  
 actual<- test$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] 0.4297859

# 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, verbose=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 1183 136  
## yes 13 25  
##   
## Accuracy : 0.8902   
## 95% CI : (0.8723, 0.9063)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.1673   
##   
## Kappa : 0.2157   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9891   
## Specificity : 0.1553   
## Pos Pred Value : 0.8969   
## Neg Pred Value : 0.6579   
## Prevalence : 0.8814   
## Detection Rate : 0.8718   
## Detection Prevalence : 0.9720   
## Balanced Accuracy : 0.5722   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(sv\_m, test[-17])  
 actual<- test$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] 0.244443

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