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

jean wills

13/07/2020

# knitr::opts\_chunk$set(echo = TRUE, message=FALSE, warning=FALSE)

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

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")  
# 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\_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:

## UNDERSAMPLE MAJOR/MINOR - 360 each for training

set.seed(30)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

down\_train <- downSample(x = BM\_train[, -ncol(BM\_train)], y = BM\_train$y)  
table(down\_train$Class)

##   
## no yes   
## 360 360

#use these now in each model if needed:  
x=down\_train[,-17]  
trainsv=down\_train  
y=down\_train$Class  
test\_noy=BM\_test[,-17]  
test\_labels=BM\_test$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)

## Loading required package: MASS

##   
## Attaching package: 'MASS'

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

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))

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 38  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 38

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 69  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 69

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 64

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 64

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 63  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 63

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 44  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 44

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 66  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 66

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 65  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 65

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 60  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 60

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 66  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 66

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 66  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 66

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 62  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 62

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 63  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 63

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37  
  
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 37

# 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)

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 962

s<-table(nab\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## nab\_pred no yes  
## no 963 58  
## yes 233 103  
##   
## Accuracy : 0.7856   
## 95% CI : (0.7627, 0.8071)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3026   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8052   
## Specificity : 0.6398   
## Pos Pred Value : 0.9432   
## Neg Pred Value : 0.3065   
## Prevalence : 0.8814   
## Detection Rate : 0.7097   
## Detection Prevalence : 0.7524   
## Balanced Accuracy : 0.7225   
##   
## '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)  
 })

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 95

#  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.4128003

# 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 892 17  
## yes 304 144  
##   
## Accuracy : 0.7634   
## 95% CI : (0.7399, 0.7858)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3614   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7458   
## Specificity : 0.8944   
## Pos Pred Value : 0.9813   
## Neg Pred Value : 0.3214   
## Prevalence : 0.8814   
## Detection Rate : 0.6573   
## Detection Prevalence : 0.6699   
## Balanced Accuracy : 0.8201   
##   
## '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] 0.4756645

# 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] Mon Aug 3 17:27:00 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 720 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## duration > 0.1985522:  
## :...contact = unknown:  
## : :...duration <= 0.4312306: no (38/2)  
## : : duration > 0.4312306:  
## : : :...loan = no: yes (38/11)  
## : : loan = yes:  
## : : :...age <= 0.3676471: no (5)  
## : : age > 0.3676471: yes (2)  
## : contact in {cellular,telephone}:  
## : :...duration > 0.4498449: yes (159/14)  
## : duration <= 0.4498449:  
## : :...previous > 0: yes (78/10)  
## : previous <= 0:  
## : :...job in {blue-collar,entrepreneur,services}: no (40/7)  
## : job in {admin.,housemaid,management,retired,self-employed,  
## : : student,technician,unemployed,unknown}:  
## : :...month in {apr,dec,feb,jun,mar,nov,oct}: yes (42/5)  
## : month = sep: no (2/1)  
## : month = aug:  
## : :...age <= 0.2352941: no (7)  
## : : age > 0.2352941: yes (15/6)  
## : month = jan:  
## : :...balance <= 0.07113864: no (2)  
## : : balance > 0.07113864: yes (2)  
## : month = jul:  
## : :...contact = cellular: no (10/2)  
## : : contact = telephone: yes (5/1)  
## : month = may:  
## : :...housing = no: yes (4)  
## : housing = yes: no (5/1)  
## duration <= 0.1985522:  
## :...poutcome = success: yes (12/2)  
## poutcome in {failure,other,unknown}:  
## :...month in {aug,dec,jan,jul,jun,may}: no (183/12)  
## month in {apr,feb,mar,nov,oct,sep}:  
## :...duration <= 0.05791106: no (14)  
## duration > 0.05791106:  
## :...day > 0.7: yes (7)  
## day <= 0.7:  
## :...loan = yes: no (9/1)  
## loan = no:  
## :...housing = no: yes (19/6)  
## housing = yes:  
## :...marital = divorced: yes (2)  
## marital in {married,single}: no (20/4)  
##   
##   
## Evaluation on training data (720 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 25 85(11.8%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 305 55 (a): class no  
## 30 330 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% duration  
## 63.06% contact  
## 48.33% month  
## 36.94% poutcome  
## 29.44% previous  
## 18.61% job  
## 13.19% loan  
## 7.92% day  
## 6.94% housing  
## 4.03% age  
## 3.06% marital  
## 0.56% balance  
##   
##   
## 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 928 38  
## yes 268 123  
##   
## Accuracy : 0.7745   
## 95% CI : (0.7513, 0.7965)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3337   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7759   
## Specificity : 0.7640   
## Pos Pred Value : 0.9607   
## Neg Pred Value : 0.3146   
## Prevalence : 0.8814   
## Detection Rate : 0.6839   
## Detection Prevalence : 0.7119   
## Balanced Accuracy : 0.7699   
##   
## '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] 0.4459825

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration <= 0.198552) => .outcome=no (266.0/49.0)  
## (duration <= 0.441572) and (pdays <= 0) and (contact = unknown) => .outcome=no (39.0/3.0)  
## (duration <= 0.449845) and (pdays <= 0) and (job = blue-collar) => .outcome=no (21.0/2.0)  
## (duration <= 0.380558) and (pdays <= 0) and (duration >= 0.241986) and (balance <= 0.093997) => .outcome=no (25.0/7.0)  
## => .outcome=yes (369.0/70.0)  
##   
## Number of Rules : 5

# 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 954 31  
## yes 242 130  
##   
## Accuracy : 0.7988   
## 95% CI : (0.7765, 0.8199)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3861   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7977   
## Specificity : 0.8075   
## Pos Pred Value : 0.9685   
## Neg Pred Value : 0.3495   
## Prevalence : 0.8814   
## Detection Rate : 0.7030   
## Detection Prevalence : 0.7259   
## Balanced Accuracy : 0.8026   
##   
## '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] 0.4892572

# 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   
## -3.3540 -0.5595 -0.0120 0.5377 2.4695   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.43187 1.15830 -0.373 0.709264   
## age -1.22500 0.97545 -1.256 0.209176   
## balance 0.11137 0.52946 0.210 0.833400   
## day 0.01022 0.46441 0.022 0.982443   
## duration 7.09570 0.60223 11.782 < 2e-16 \*\*\*  
## campaign -1.81217 0.62223 -2.912 0.003587 \*\*   
## pdays 0.34119 0.84473 0.404 0.686284   
## previous 0.98369 0.74275 1.324 0.185373   
## jobblue-collar -1.60768 0.48155 -3.339 0.000842 \*\*\*  
## jobentrepreneur -1.34318 0.65102 -2.063 0.039095 \*   
## jobhousemaid -1.31337 0.79255 -1.657 0.097491 .   
## jobmanagement -0.53040 0.46486 -1.141 0.253876   
## jobretired -1.00950 0.63276 -1.595 0.110626   
## jobself-employed -0.38300 0.75023 -0.511 0.609694   
## jobservices -1.46996 0.52901 -2.779 0.005458 \*\*   
## jobstudent -0.26856 0.74905 -0.359 0.719948   
## jobtechnician -1.04827 0.45112 -2.324 0.020141 \*   
## jobunemployed -1.40485 0.89429 -1.571 0.116202   
## jobunknown 0.77682 1.44233 0.539 0.590173   
## maritalmarried -0.02182 0.36633 -0.060 0.952501   
## maritalsingle -0.14601 0.41922 -0.348 0.727623   
## educationsecondary 0.09831 0.38925 0.253 0.800599   
## educationtertiary -0.09947 0.44876 -0.222 0.824583   
## educationunknown -0.78101 0.65934 -1.185 0.236202   
## defaultyes 0.43161 0.95996 0.450 0.652987   
## housingyes -0.54022 0.25400 -2.127 0.033434 \*   
## loanyes -1.46498 0.37713 -3.885 0.000103 \*\*\*  
## contacttelephone 0.29669 0.42640 0.696 0.486552   
## contactunknown -1.69400 0.40289 -4.205 2.62e-05 \*\*\*  
## monthaug -0.30639 0.46512 -0.659 0.510058   
## monthdec 14.96770 705.45976 0.021 0.983073   
## monthfeb 0.82446 0.59806 1.379 0.168026   
## monthjan -1.62003 0.72654 -2.230 0.025761 \*   
## monthjul -0.67857 0.47696 -1.423 0.154828   
## monthjun 0.46273 0.57947 0.799 0.424555   
## monthmar 1.59881 0.79531 2.010 0.044398 \*   
## monthmay -1.09559 0.46081 -2.378 0.017428 \*   
## monthnov -0.60997 0.54663 -1.116 0.264470   
## monthoct 1.98583 0.76089 2.610 0.009058 \*\*   
## monthsep 0.65910 0.79947 0.824 0.409703   
## poutcomeother 0.52976 0.52894 1.002 0.316569   
## poutcomesuccess 1.82025 0.58358 3.119 0.001814 \*\*   
## poutcomeunknown 0.46452 0.75443 0.616 0.538080   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 998.13 on 719 degrees of freedom  
## Residual deviance: 534.47 on 677 degrees of freedom  
## AIC: 620.47  
##   
## Number of Fisher Scoring iterations: 15

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

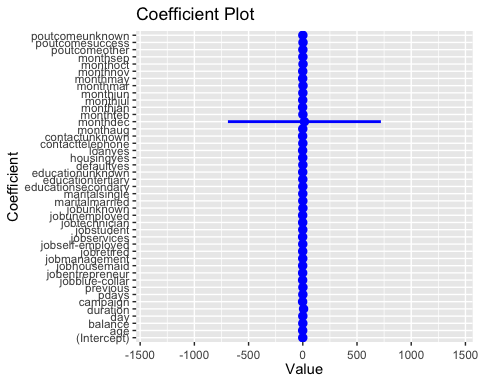
require(coefplot)

## Loading required package: coefplot

##   
## Attaching package: 'coefplot'

## The following object is masked from 'package:e1071':  
##   
## extractPath

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 908 37  
## yes 288 124  
##   
## Accuracy : 0.7605   
## 95% CI : (0.7369, 0.783)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3161   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7592   
## Specificity : 0.7702   
## Pos Pred Value : 0.9608   
## Neg Pred Value : 0.3010   
## Prevalence : 0.8814   
## Detection Rate : 0.6691   
## Detection Prevalence : 0.6964   
## Balanced Accuracy : 0.7647   
##   
## '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] 0.4328126

# 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)

## Loading required package: mlbench

require(caret)  
require (nnet)

## Loading required package: 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 931 28  
## yes 265 133  
##   
## Accuracy : 0.7841   
## 95% CI : (0.7612, 0.8057)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3693   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7784   
## Specificity : 0.8261   
## Pos Pred Value : 0.9708   
## Neg Pred Value : 0.3342   
## Prevalence : 0.8814   
## Detection Rate : 0.6861   
## Detection Prevalence : 0.7067   
## Balanced Accuracy : 0.8023   
##   
## '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] 0.477039

# 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)

##   
## Attaching package: 'kernlab'

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

fitctrl<- trainControl(method="repeatedcv", number = 10, repeats = 10, classProbs=TRUE, summaryFunction = twoClassSummary)  
grid<-expand.grid(C=1)  
sv\_m<- train(Class~., 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 960 34  
## yes 236 127  
##   
## Accuracy : 0.801   
## 95% CI : (0.7788, 0.822)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3834   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8027   
## Specificity : 0.7888   
## Pos Pred Value : 0.9658   
## Neg Pred Value : 0.3499   
## Prevalence : 0.8814   
## Detection Rate : 0.7074   
## Detection Prevalence : 0.7325   
## Balanced Accuracy : 0.7957   
##   
## '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] 0.4875208

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