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]

# FS3 Boratu - Dtemp<- DELETE only 2 - job and loan - # 8, 13 => delete, 13, 8  
temp<-BM\_mini\_sc  
temp<- temp[-13]  
temp<- temp[-8]  
BM<- temp  
summary(BM)

## age balance day duration   
## Min. :0.0000 Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.2059 1st Qu.:0.04935 1st Qu.:0.2667 1st Qu.:0.0879   
## Median :0.2941 Median :0.08940 Median :0.5000 Median :0.1717   
## Mean :0.3260 Mean :0.17805 Mean :0.4972 Mean :0.2450   
## 3rd Qu.:0.4412 3rd Qu.:0.20006 3rd Qu.:0.6667 3rd Qu.:0.3206   
## Max. :1.0000 Max. :1.00000 Max. :1.0000 Max. :1.0000   
##   
## campaign pdays previous marital   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 divorced: 528   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 married :2797   
## Median :0.1000 Median :0.0000 Median :0.00000 single :1196   
## Mean :0.1642 Mean :0.1105 Mean :0.09082   
## 3rd Qu.:0.2000 3rd Qu.:0.0000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000   
##   
## education default housing contact month   
## primary : 678 no :4445 no :1962 cellular :2896 may :1398   
## secondary:2306 yes: 76 yes:2559 telephone: 301 jul : 706   
## tertiary :1350 unknown :1324 aug : 633   
## unknown : 187 jun : 531   
## nov : 389   
## apr : 293   
## (Other): 571   
## poutcome y   
## failure: 490 no :4000   
## other : 197 yes: 521   
## success: 129   
## unknown:3705   
##   
##   
##

## 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 15....  
x=smote\_train[,-15]  
trainsv=smote\_train  
train=smote\_train  
y=smote\_train$y  
test\_noy=BM\_test[,-15]  
test\_labels=BM\_test$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 1045 62  
## yes 151 99  
##   
## Accuracy : 0.843   
## 95% CI : (0.8226, 0.862)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3943   
##   
## Mcnemar's Test P-Value : 1.643e-09   
##   
## Sensitivity : 0.8737   
## Specificity : 0.6149   
## Pos Pred Value : 0.9440   
## Neg Pred Value : 0.3960   
## Prevalence : 0.8814   
## Detection Rate : 0.7701   
## Detection Prevalence : 0.8158   
## Balanced Accuracy : 0.7443   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nab\_mod, test[-15])  
 actual<- test$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] 0.4764012

# 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 1031 36  
## yes 165 125  
##   
## Accuracy : 0.8519   
## 95% CI : (0.8319, 0.8704)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9995   
##   
## Kappa : 0.4741   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8620   
## Specificity : 0.7764   
## Pos Pred Value : 0.9663   
## Neg Pred Value : 0.4310   
## Prevalence : 0.8814   
## Detection Rate : 0.7598   
## Detection Prevalence : 0.7863   
## Balanced Accuracy : 0.8192   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(c5\_mod, test[,-15])  
 actual<- test$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] 0.556841

# 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] Tue Jul 28 23:53:39 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 2520 cases (15 attributes) from undefined.data  
##   
## Decision tree:  
##   
## duration <= 0.1706308:  
## :...month in {apr,feb,mar,oct,sep}:  
## : :...poutcome = success: yes (6)  
## : : poutcome in {failure,other,unknown}:  
## : : :...month in {apr,feb}:  
## : : :...contact in {cellular,telephone}: no (81/13)  
## : : : contact = unknown: yes (2)  
## : : month in {mar,oct,sep}:  
## : : :...duration <= 0.06928645: no (8/1)  
## : : duration > 0.06928645: yes (29/8)  
## : month in {aug,dec,jan,jul,jun,may,nov}:  
## : :...poutcome = other: no (41/8)  
## : poutcome = success:  
## : :...duration <= 0.08790072: no (4)  
## : : duration > 0.08790072: yes (8/1)  
## : poutcome in {failure,unknown}:  
## : :...previous <= 0.000833591: no (634/9)  
## : previous > 0.000833591:  
## : :...poutcome = failure: no (56/3)  
## : poutcome = unknown: yes (4)  
## duration > 0.1706308:  
## :...duration > 0.6080662: yes (423/61)  
## duration <= 0.6080662:  
## :...pdays > 0.005617978:  
## :...poutcome in {success,unknown}: yes (281/3)  
## : poutcome in {failure,other}:  
## : :...previous <= 0.1938858: yes (40)  
## : previous > 0.1938858:  
## : :...month = jan: no (11)  
## : month in {aug,dec,jul,jun,nov,oct,sep}: yes (85/14)  
## : month = mar:  
## : :...age <= 0.6618519: yes (2)  
## : : age > 0.6618519: no (2)  
## : month = apr:  
## : :...housing = yes: no (20/2)  
## : : housing = no:  
## : : :...age <= 0.4697259: yes (4)  
## : : age > 0.4697259: no (4)  
## : month = feb:  
## : :...campaign > 0.2: no (5)  
## : : campaign <= 0.2:  
## : : :...balance <= 0.6885281: yes (11/2)  
## : : balance > 0.6885281: no (2)  
## : month = may:  
## : :...day > 0.5709449: yes (6)  
## : day <= 0.5709449:  
## : :...duration <= 0.3702172: no (19/1)  
## : duration > 0.3702172:  
## : :...pdays <= 0.9831461: yes (8)  
## : pdays > 0.9831461:  
## : :...duration <= 0.4881076: yes (2)  
## : duration > 0.4881076: no (6)  
## pdays <= 0.005617978:  
## :...poutcome in {failure,success}: yes (2)  
## poutcome = other: no (5/1)  
## poutcome = unknown:  
## :...month in {dec,mar,oct}: yes (32/2)  
## month in {apr,feb,sep}:  
## :...day > 0.6479374: yes (22)  
## : day <= 0.6479374:  
## : :...contact = unknown: yes (4)  
## : contact = telephone:  
## : :...age <= 0.5349576: yes (10/1)  
## : : age > 0.5349576: no (4)  
## : contact = cellular:  
## : :...duration > 0.4736298: yes (11/2)  
## : duration <= 0.4736298:  
## : :...age <= 0.576771: no (39/4)  
## : age > 0.576771: yes (3)  
## month in {aug,jan,jul,jun,may,nov}:  
## :...duration <= 0.42606:  
## :...contact = unknown:  
## : :...month in {aug,nov}: yes (4)  
## : : month in {jan,jul,jun,may}: no (161/2)  
## : contact = telephone:  
## : :...balance > 0.30047: no (8)  
## : : balance <= 0.30047:  
## : : :...month in {jun,may,nov}: no (7/2)  
## : : month in {aug,jan}: yes (8)  
## : : month = jul:  
## : : :...balance <= 0.1063875: no (5/1)  
## : : balance > 0.1063875: yes (6)  
## : contact = cellular:  
## : :...month = jun: yes (14/2)  
## : month in {aug,jan,jul,may,nov}:  
## : :...age > 0.1385546:  
## : :...age <= 0.5918233: no (199/22)  
## : : age > 0.5918233: yes (14/6)  
## : age <= 0.1385546:  
## : :...balance <= 0.06099124: no (6)  
## : balance > 0.06099124:  
## : :...day <= 0.2799734: no (2)  
## : day > 0.2799734: yes (9)  
## duration > 0.42606:  
## :...balance > 0.5401624: no (9)  
## balance <= 0.5401624:  
## :...balance > 0.2910224: yes (11)  
## balance <= 0.2910224:  
## :...campaign <= 0.08691704:  
## :...campaign > 0.01073316: yes (10)  
## : campaign <= 0.01073316:  
## : :...marital = divorced: yes (8/1)  
## : marital in {married,single}:  
## : :...balance <= 0.06686606: yes (13/2)  
## : balance > 0.06686606: no (28/6)  
## campaign > 0.08691704:  
## :...marital = divorced: no (6/1)  
## marital = single:  
## :...education in {primary,secondary,  
## : : unknown}: no (17/6)  
## : education = tertiary: yes (6/1)  
## marital = married:  
## :...contact = telephone: yes (2)  
## contact in {cellular,unknown}:  
## :...month in {aug,jan,jul,may,  
## : nov}: no (29/2)  
## month = jun: yes (2)  
##   
##   
## Evaluation on training data (2520 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 64 190( 7.5%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 1334 106 (a): class no  
## 84 996 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% duration  
## 83.21% poutcome  
## 70.20% month  
## 48.57% pdays  
## 36.55% previous  
## 25.00% contact  
## 11.83% age  
## 8.13% balance  
## 5.75% day  
## 5.52% campaign  
## 4.40% marital  
## 1.11% housing  
## 0.91% 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 1040 39  
## yes 156 122  
##   
## Accuracy : 0.8563   
## 95% CI : (0.8365, 0.8745)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9976   
##   
## Kappa : 0.4773   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8696   
## Specificity : 0.7578   
## Pos Pred Value : 0.9639   
## Neg Pred Value : 0.4388   
## Prevalence : 0.8814   
## Detection Rate : 0.7664   
## Detection Prevalence : 0.7951   
## Balanced Accuracy : 0.8137   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(JR\_mod, test[-15])  
 actual<- test$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] 0.5549976

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.276681) and (pdays >= 0.000674) => .outcome=yes (455.0/40.0)  
## (duration >= 0.621421) => .outcome=yes (274.0/49.0)  
## (duration >= 0.168791) and (pdays >= 0.018038) and (pdays <= 0.393532) => .outcome=yes (109.0/10.0)  
## (duration >= 0.171579) and (campaign <= 0.09322) and (campaign >= 0.004931) => .outcome=yes (49.0/0.0)  
## (duration >= 0.210962) and (contact = telephone) and (balance <= 0.288293) and (age <= 0.527269) => .outcome=yes (33.0/3.0)  
## (duration >= 0.13677) and (month = oct) => .outcome=yes (27.0/3.0)  
## (duration >= 0.210962) and (contact = cellular) and (month = jun) => .outcome=yes (21.0/3.0)  
## (pdays >= 0.000306) and (pdays <= 0.280899) => .outcome=yes (50.0/18.0)  
## (duration >= 0.192347) and (age >= 0.617647) => .outcome=yes (23.0/7.0)  
## (duration >= 0.425026) and (campaign <= 0.099774) and (balance <= 0.060137) => .outcome=yes (19.0/4.0)  
## (duration >= 0.1542) and (month = apr) and (day >= 0.666667) => .outcome=yes (17.0/1.0)  
## (duration >= 0.492314) and (marital = single) => .outcome=yes (24.0/8.0)  
## (duration >= 0.163392) and (balance >= 0.150182) and (pdays >= 0.497096) => .outcome=yes (21.0/7.0)  
## => .outcome=no (1398.0/111.0)  
##   
## Number of Rules : 14

# 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 1039 44  
## yes 157 117  
##   
## Accuracy : 0.8519   
## 95% CI : (0.8319, 0.8704)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9995   
##   
## Kappa : 0.4567   
##   
## Mcnemar's Test P-Value : 2.792e-15   
##   
## Sensitivity : 0.8687   
## Specificity : 0.7267   
## Pos Pred Value : 0.9594   
## Neg Pred Value : 0.4270   
## Prevalence : 0.8814   
## Detection Rate : 0.7657   
## Detection Prevalence : 0.7981   
## Balanced Accuracy : 0.7977   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(log\_mod, test[-15])  
 actual<- test$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] 0.535786

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

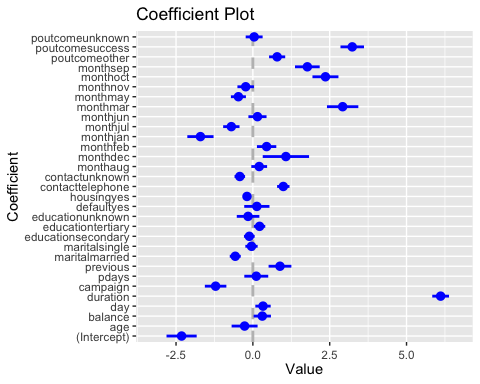
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.4587 -0.5501 -0.2822 0.4964 2.3792   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.31902 0.49068 -4.726 2.29e-06 \*\*\*  
## age -0.26921 0.42273 -0.637 0.524241   
## balance 0.30671 0.28158 1.089 0.276049   
## day 0.33001 0.24999 1.320 0.186795   
## duration 6.10989 0.27266 22.408 < 2e-16 \*\*\*  
## campaign -1.21240 0.35006 -3.463 0.000533 \*\*\*  
## pdays 0.11092 0.38939 0.285 0.775746   
## previous 0.88221 0.37084 2.379 0.017362 \*   
## maritalmarried -0.57297 0.17941 -3.194 0.001405 \*\*   
## maritalsingle -0.04465 0.20013 -0.223 0.823469   
## educationsecondary -0.11528 0.17486 -0.659 0.509746   
## educationtertiary 0.21825 0.18518 1.179 0.238555   
## educationunknown -0.15614 0.36773 -0.425 0.671139   
## defaultyes 0.12957 0.41057 0.316 0.752324   
## housingyes -0.19142 0.12463 -1.536 0.124576   
## contacttelephone 0.99038 0.20158 4.913 8.96e-07 \*\*\*  
## contactunknown -0.42524 0.17200 -2.472 0.013427 \*   
## monthaug 0.20626 0.25691 0.803 0.422063   
## monthdec 1.07457 0.75388 1.425 0.154047   
## monthfeb 0.44725 0.31526 1.419 0.156003   
## monthjan -1.70601 0.42680 -3.997 6.41e-05 \*\*\*  
## monthjul -0.70091 0.26587 -2.636 0.008383 \*\*   
## monthjun 0.14985 0.29505 0.508 0.611537   
## monthmar 2.92134 0.51040 5.724 1.04e-08 \*\*\*  
## monthmay -0.47052 0.24435 -1.926 0.054161 .   
## monthnov -0.23325 0.26903 -0.867 0.385936   
## monthoct 2.36170 0.42276 5.586 2.32e-08 \*\*\*  
## monthsep 1.77258 0.40175 4.412 1.02e-05 \*\*\*  
## poutcomeother 0.78953 0.26253 3.007 0.002635 \*\*   
## poutcomesuccess 3.23346 0.38251 8.453 < 2e-16 \*\*\*  
## poutcomeunknown 0.04198 0.27552 0.152 0.878910   
## ---  
## 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: 1929.8 on 2489 degrees of freedom  
## AIC: 1991.8  
##   
## 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 1038 58  
## yes 158 103  
##   
## Accuracy : 0.8408   
## 95% CI : (0.8203, 0.8599)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4001   
##   
## Mcnemar's Test P-Value : 1.627e-11   
##   
## Sensitivity : 0.8679   
## Specificity : 0.6398   
## Pos Pred Value : 0.9471   
## Neg Pred Value : 0.3946   
## Prevalence : 0.8814   
## Detection Rate : 0.7649   
## Detection Prevalence : 0.8077   
## Balanced Accuracy : 0.7538   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(knn\_mod, test[-15])  
 actual<- test$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] 0.4856178

# 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 1032 49  
## yes 164 112  
##   
## Accuracy : 0.843   
## 95% CI : (0.8226, 0.862)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4267   
##   
## Mcnemar's Test P-Value : 5.667e-15   
##   
## Sensitivity : 0.8629   
## Specificity : 0.6957   
## Pos Pred Value : 0.9547   
## Neg Pred Value : 0.4058   
## Prevalence : 0.8814   
## Detection Rate : 0.7605   
## Detection Prevalence : 0.7966   
## Balanced Accuracy : 0.7793   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nn\_mod, test[-15])  
 actual<- test$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] 0.5120622

# 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)

## line search fails -1.390461 0.05781246 1.056833e-05 8.787014e-07 -3.342991e-08 -1.048887e-08 -3.62515e-13

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 1041 45  
## yes 155 116  
##   
## Accuracy : 0.8526   
## 95% CI : (0.8326, 0.8711)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9994   
##   
## Kappa : 0.4561   
##   
## Mcnemar's Test P-Value : 1.283e-14   
##   
## Sensitivity : 0.8704   
## Specificity : 0.7205   
## Pos Pred Value : 0.9586   
## Neg Pred Value : 0.4280   
## Prevalence : 0.8814   
## Detection Rate : 0.7671   
## Detection Prevalence : 0.8003   
## Balanced Accuracy : 0.7954   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(sv\_m, test[-15])  
 actual<- test$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] 0.537363

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