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)  
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  
#  
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:

## ROSE SAMPLE MAJOR/MINOR - each for training \*\*\* y changes to Class for some \*\*\* FIX FOR ALL

# install.packages("ROSE")  
library(ROSE)  
set.seed(123)  
rose\_train <- ROSE(y ~ ., data = BM\_train)$data   
table(rose\_train$y)

##   
## no yes   
## 1598 1566

# step 3b - now run the following for each separate balanced data….FIX FOR ALL

#use these now in each model if needed:  
x=rose\_train[,-17]  
trainsv=rose\_train  
train=rose\_train  
y=rose\_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, verboseIter=FALSE))  
# 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 1024 74  
## yes 172 87  
##   
## Accuracy : 0.8187   
## 95% CI : (0.7972, 0.8389)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3139   
##   
## Mcnemar's Test P-Value : 6.23e-10   
##   
## Sensitivity : 0.8562   
## Specificity : 0.5404   
## Pos Pred Value : 0.9326   
## Neg Pred Value : 0.3359   
## Prevalence : 0.8814   
## Detection Rate : 0.7546   
## Detection Prevalence : 0.8091   
## Balanced Accuracy : 0.6983   
##   
## '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] 0.4133345

# 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 990 49  
## yes 206 112  
##   
## Accuracy : 0.8121   
## 95% CI : (0.7903, 0.8325)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3681   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8278   
## Specificity : 0.6957   
## Pos Pred Value : 0.9528   
## Neg Pred Value : 0.3522   
## Prevalence : 0.8814   
## Detection Rate : 0.7296   
## Detection Prevalence : 0.7657   
## Balanced Accuracy : 0.7617   
##   
## '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] 0.4708581

# 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, CF = 0.25, minCases = 2, fuzzyThreshold = FALSE, sample =  
## 0, earlyStopping = TRUE, label = "outcome", seed = 2372L), verbose = FALSE)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Wed Jul 29 01:24:58 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 3164 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## duration > 0.3900346:  
## :...contact = unknown:  
## : :...duration <= 0.7508941:  
## : : :...job in {housemaid,self-employed,student,unemployed,  
## : : : : unknown}: no (8)  
## : : : job = retired: yes (2)  
## : : : job = entrepreneur:  
## : : : :...age <= 0.1399173: yes (2)  
## : : : : age > 0.1399173: no (2)  
## : : : job = services:  
## : : : :...age <= 0.142028: yes (3)  
## : : : : age > 0.142028: no (13)  
## : : : job = blue-collar:  
## : : : :...education in {primary,tertiary,unknown}: no (3)  
## : : : : education = secondary:  
## : : : : :...loan = no: yes (21/8)  
## : : : : loan = yes: no (3)  
## : : : job = management:  
## : : : :...month in {apr,dec,feb,jan,jul,mar,nov,oct,sep}: yes (0)  
## : : : : month = may: no (4)  
## : : : : month in {aug,jun}:  
## : : : : :...education = secondary: no (1)  
## : : : : education in {primary,tertiary,unknown}: yes (13/2)  
## : : : job = technician:  
## : : : :...day <= 0.6043191: no (8)  
## : : : : day > 0.6043191:  
## : : : : :...campaign <= 0.02732627: no (2)  
## : : : : campaign > 0.02732627: yes (5)  
## : : : job = admin.:  
## : : : :...month in {apr,aug,dec,feb,jan,jul,mar,nov,oct,sep}: yes (0)  
## : : : month = jun: no (2)  
## : : : month = may:  
## : : : :...education in {primary,tertiary}: yes (3)  
## : : : education = unknown: no (1)  
## : : : education = secondary:  
## : : : :...loan = no: no (2)  
## : : : loan = yes: yes (5)  
## : : duration > 0.7508941:  
## : : :...month in {apr,aug,dec,feb,jan,mar,nov,oct,sep}: yes (0)  
## : : month = jul: no (2)  
## : : month in {jun,may}:  
## : : :...age > 0.3032653: yes (43/1)  
## : : age <= 0.3032653:  
## : : :...loan = yes: no (2)  
## : : loan = no:  
## : : :...marital in {divorced,single}: yes (16/1)  
## : : marital = married:  
## : : :...duration <= 0.9020767: yes (5/1)  
## : : duration > 0.9020767:  
## : : :...education in {primary,secondary,  
## : : : unknown}: no (7)  
## : : education = tertiary: yes (2)  
## : contact in {cellular,telephone}:  
## : :...poutcome = success:  
## : :...contact = cellular: yes (102)  
## : : contact = telephone:  
## : : :...month = apr: no (2)  
## : : month in {aug,dec,feb,jan,jul,jun,mar,may,nov,oct,  
## : : sep}: yes (6)  
## : poutcome in {failure,other,unknown}:  
## : :...duration > 0.7970093:  
## : :...contact = cellular: yes (244/10)  
## : : contact = telephone:  
## : : :...job in {blue-collar,housemaid,services}: no (4)  
## : : job in {admin.,entrepreneur,management,retired,  
## : : self-employed,student,technician,unemployed,  
## : : unknown}: yes (17)  
## : duration <= 0.7970093:  
## : :...previous <= -0.1801277: yes (53/1)  
## : previous > -0.1801277:  
## : :...month in {dec,feb,jun,mar,oct}: yes (132/7)  
## : month in {apr,aug,may,sep}:  
## : :...loan = yes:  
## : : :...previous > 0.555738: yes (10)  
## : : : previous <= 0.555738:  
## : : : :...job in {admin.,blue-collar,entrepreneur,  
## : : : : housemaid,management,retired,  
## : : : : self-employed,student,unemployed,  
## : : : : unknown}: no (10)  
## : : : job in {services,technician}: yes (4/1)  
## : : loan = no:  
## : : :...job in {admin.,entrepreneur,housemaid,management,  
## : : : self-employed,unemployed,  
## : : : unknown}: yes (109/18)  
## : : job = student:  
## : : :...housing = no: yes (2)  
## : : : housing = yes: no (3)  
## : : job = services:  
## : : :...education = primary: yes (0)  
## : : : education in {tertiary,unknown}: no (2)  
## : : : education = secondary:  
## : : : :...day <= 0.2576606: no (2)  
## : : : day > 0.2576606: yes (10)  
## : : job = technician:  
## : : :...poutcome = other: yes (0)  
## : : : poutcome = failure: no (3)  
## : : : poutcome = unknown:  
## : : : :...day <= 0.8112814: yes (43/6)  
## : : : day > 0.8112814: no (2)  
## : : job = blue-collar:  
## : : :...marital = divorced: no (1)  
## : : : marital = single: yes (13)  
## : : : marital = married:  
## : : : :...balance > 0.3078804: yes (7)  
## : : : balance <= 0.3078804:  
## : : : :...age <= 0.577369: no (13/1)  
## : : : age > 0.577369: yes (2)  
## : : job = retired:  
## : : :...education in {primary,tertiary,  
## : : : unknown}: yes (6)  
## : : education = secondary:  
## : : :...marital = single: no (0)  
## : : marital = divorced: yes (2)  
## : : marital = married:  
## : : :...housing = no: no (5)  
## : : housing = yes: yes (2)  
## : month in {jan,jul,nov}:  
## : :...default = yes: yes (6)  
## : default = no:  
## : :...previous > 0.7248725: yes (9)  
## : previous <= 0.7248725:  
## : :...marital = divorced:  
## : :...loan = no: yes (14/2)  
## : : loan = yes: no (4/1)  
## : marital in {married,single}:  
## : :...job in {admin.,retired,unemployed,  
## : : unknown}: no (14/1)  
## : job = student: yes (2)  
## : job = entrepreneur:  
## : :...duration <= 0.566397: no (2)  
## : : duration > 0.566397: yes (5)  
## : job = housemaid:  
## : :...housing = no: yes (4)  
## : : housing = yes: no (2)  
## : job = services:  
## : :...day <= 0.379189: yes (2)  
## : : day > 0.379189: no (9)  
## : job = blue-collar:  
## : :...marital = married: no (10)  
## : : marital = single:  
## : : :...duration <= 0.7770826: yes (4)  
## : : duration > 0.7770826: no (2)  
## : job = self-employed:  
## : :...education = primary: yes (1)  
## : : education in {secondary,  
## : : : unknown}: no (4)  
## : : education = tertiary:  
## : : :...month in {jan,nov}: yes (3)  
## : : month = jul: no (1)  
## : job = technician:  
## : :...housing = no:  
## : : :...month in {jan,jul}: no (4)  
## : : : month = nov: yes (1)  
## : : housing = yes: [S1]  
## : job = management:  
## : :...poutcome = failure: no (3)  
## : poutcome = other: yes (1)  
## : poutcome = unknown:  
## : :...campaign <= 0.1914849: no (16/2)  
## : campaign > 0.1914849:  
## : :...age <= 0.09940609: no (2)  
## : age > 0.09940609: yes (9)  
## duration <= 0.3900346:  
## :...poutcome = success:  
## :...contact = telephone: yes (17)  
## : contact = unknown: no (1)  
## : contact = cellular:  
## : :...job = entrepreneur: yes (0)  
## : job in {technician,unknown}:  
## : :...month in {apr,aug,dec,feb,mar,may,oct}: no (14/1)  
## : : month in {jan,jul,jun,nov,sep}: yes (11)  
## : job in {admin.,blue-collar,housemaid,management,retired,  
## : : self-employed,services,student,unemployed}:  
## : :...education in {secondary,tertiary,unknown}: yes (121/7)  
## : education = primary:  
## : :...duration <= 0.261616: no (5)  
## : duration > 0.261616: yes (3)  
## poutcome in {failure,other,unknown}:  
## :...month in {apr,dec,feb,mar,oct,sep}:  
## :...job in {blue-collar,entrepreneur,housemaid}:  
## : :...marital = divorced:  
## : : :...education in {primary,tertiary}: no (3)  
## : : : education in {secondary,unknown}: yes (6)  
## : : marital in {married,single}:  
## : : :...contact = cellular: no (44/1)  
## : : contact in {telephone,unknown}:  
## : : :...housing = no: no (6)  
## : : housing = yes: yes (7/1)  
## : job in {admin.,management,retired,self-employed,services,student,  
## : : technician,unemployed,unknown}:  
## : :...loan = yes:  
## : :...month in {apr,dec,feb,mar,sep}: no (17)  
## : : month = oct: yes (10/2)  
## : loan = no:  
## : :...month in {mar,oct}:  
## : :...poutcome in {failure,unknown}: yes (105/8)  
## : : poutcome = other:  
## : : :...balance <= 0.1939913: yes (2)  
## : : balance > 0.1939913: no (4)  
## : month in {apr,dec,feb,sep}:  
## : :...pdays > 0.6836674: no (21/3)  
## : pdays <= 0.6836674:  
## : :...poutcome = other: yes (19/1)  
## : poutcome in {failure,unknown}:  
## : :...day > 0.5303066: yes (80/14)  
## : day <= 0.5303066:  
## : :...job in {admin.,self-employed,services,  
## : : student,unknown}: no (19)  
## : job in {management,retired,technician,  
## : : unemployed}:  
## : :...housing = yes:  
## : :...month in {apr,feb}: no (14/3)  
## : : month = sep: yes (1)  
## : : month = dec:  
## : : :...job = technician: no (1)  
## : : job in {management,retired,  
## : : unemployed}: yes (3)  
## : housing = no: [S2]  
## month in {aug,jan,jul,jun,may,nov}:  
## :...age <= 0.0915713:  
## :...contact = unknown:  
## : :...poutcome = failure: no (0)  
## : : poutcome = other: yes (1)  
## : : poutcome = unknown:  
## : : :...job in {admin.,blue-collar,housemaid,management,  
## : : : retired,self-employed,services,student,  
## : : : technician,unemployed,unknown}: no (20)  
## : : job = entrepreneur: yes (2)  
## : contact in {cellular,telephone}:  
## : :...marital = divorced: yes (2)  
## : marital = married:  
## : :...contact = telephone: yes (2)  
## : : contact = cellular:  
## : : :...previous <= 0.555738: no (21/5)  
## : : previous > 0.555738: yes (4)  
## : marital = single:  
## : :...pdays > 0.3829445: no (4)  
## : pdays <= 0.3829445:  
## : :...housing = no: yes (36/5)  
## : housing = yes:  
## : :...previous <= -0.1685526: yes (3)  
## : previous > -0.1685526: no (4)  
## age > 0.0915713:  
## :...contact = unknown:  
## :...month in {aug,jan,jul,jun,may}:  
## : :...poutcome = other: yes (2)  
## : : poutcome in {failure,unknown}: no (411/12)  
## : month = nov:  
## : :...job in {admin.,blue-collar,entrepreneur,housemaid,  
## : : management,retired,services,student,technician,  
## : : unemployed,unknown}: yes (4)  
## : job = self-employed: no (2)  
## contact in {cellular,telephone}:  
## :...month = jun:  
## :...balance > 0.2708378: yes (14)  
## : balance <= 0.2708378:  
## : :...marital in {divorced,single}: yes (17/5)  
## : marital = married:  
## : :...education in {primary,secondary}: no (12)  
## : education = unknown: yes (1)  
## : education = tertiary:  
## : :...loan = no: no (4)  
## : loan = yes: yes (2)  
## month in {aug,jan,jul,may,nov}:  
## :...job in {admin.,blue-collar,entrepreneur,housemaid,  
## : management,self-employed,services,technician,  
## : unemployed}:  
## :...duration > 0.2428025:  
## : :...pdays <= 0.2074154:  
## : : :...previous <= -0.1478059: yes (17/6)  
## : : : previous > -0.1478059:  
## : : : :...default = no: no (133/23)  
## : : : default = yes:  
## : : : :...marital = married: no (2)  
## : : : marital in {divorced,  
## : : : single}: yes (4)  
## : : pdays > 0.2074154:  
## : : :...month in {aug,jan}: yes (13)  
## : : month in {jul,may,nov}:  
## : : :...job in {blue-collar,  
## : : : self-employed}: no (3)  
## : : job in {entrepreneur,housemaid,  
## : : : technician,  
## : : : unemployed}: yes (4)  
## : : job = admin.:  
## : : :...pdays <= 0.8578955: no (2)  
## : : : pdays > 0.8578955: yes (2)  
## : : job = services:  
## : : :...housing = no: yes (4)  
## : : : housing = yes: no (3)  
## : : job = management:  
## : : :...marital = divorced: no (2)  
## : : marital = single: yes (1)  
## : : marital = married:  
## : : :...campaign <= 0.05794393: no (3)  
## : : campaign > 0.05794393: yes (4)  
## : duration <= 0.2428025:  
## : :...previous <= -0.1780963:  
## : :...contact = cellular: no (25/8)  
## : : contact = telephone: yes (3)  
## : previous > -0.1780963:  
## : :...loan = yes: no (97/2)  
## : loan = no:  
## : :...poutcome in {other,  
## : : unknown}: no (387/30)  
## : poutcome = failure:  
## : :...month in {aug,jul}: yes (8/1)  
## : month in {jan,may,nov}: [S3]  
## job in {retired,student,unknown}:  
## :...loan = yes: no (9)  
## loan = no:  
## :...education in {primary,unknown}:  
## :...previous <= 0.7117147: no (21/1)  
## : previous > 0.7117147: yes (3)  
## education in {secondary,tertiary}:  
## :...marital in {divorced,single}: yes (20/2)  
## marital = married:  
## :...campaign > 0.3416926: no (7)  
## campaign <= 0.3416926:  
## :...job = student: no (3)  
## job in {retired,unknown}: [S4]  
##   
## SubTree [S1]  
##   
## education in {primary,secondary,unknown}: yes (9)  
## education = tertiary: no (3/1)  
##   
## SubTree [S2]  
##   
## education in {primary,secondary,unknown}: yes (24/2)  
## education = tertiary:  
## :...poutcome = failure: yes (7)  
## poutcome = unknown:  
## :...contact in {cellular,unknown}: no (17/5)  
## contact = telephone: yes (3)  
##   
## SubTree [S3]  
##   
## job in {blue-collar,entrepreneur,housemaid,self-employed,services,technician,  
## : unemployed}: no (37)  
## job in {admin.,management}:  
## :...education in {primary,tertiary,unknown}: no (15/3)  
## education = secondary:  
## :...month = may: no (3)  
## month in {jan,nov}: yes (5)  
##   
## SubTree [S4]  
##   
## poutcome = failure: no (3)  
## poutcome in {other,unknown}:  
## :...contact = telephone: yes (7)  
## contact = cellular:  
## :...day <= 0.6711588: yes (19/2)  
## day > 0.6711588: no (5)  
##   
##   
## Evaluation on training data (3164 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 168 217( 6.9%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 1484 114 (a): class no  
## 103 1463 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% duration  
## 94.31% poutcome  
## 89.38% contact  
## 79.80% month  
## 61.54% job  
## 49.40% age  
## 43.43% previous  
## 42.16% loan  
## 14.66% marital  
## 14.32% pdays  
## 13.59% education  
## 9.01% default  
## 8.72% day  
## 5.31% housing  
## 2.69% campaign  
## 2.47% balance  
##   
##   
## Time: 0.1 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 978 27  
## yes 218 134  
##   
## Accuracy : 0.8195   
## 95% CI : (0.7979, 0.8396)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4295   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8177   
## Specificity : 0.8323   
## Pos Pred Value : 0.9731   
## Neg Pred Value : 0.3807   
## Prevalence : 0.8814   
## Detection Rate : 0.7207   
## Detection Prevalence : 0.7406   
## Balanced Accuracy : 0.8250   
##   
## '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] 0.5232393

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.750894) => .outcome=yes (407.0/36.0)  
## (duration >= 0.325744) and (previous >= 0.197061) => .outcome=yes (274.0/37.0)  
## (duration >= 0.452823) and (loan = no) => .outcome=yes (324.0/79.0)  
## (pdays >= 0.192901) and (poutcome = success) => .outcome=yes (130.0/21.0)  
## (duration >= 0.222715) and (pdays >= 0.221832) => .outcome=yes (121.0/40.0)  
## (previous >= 0.113664) and (poutcome = success) => .outcome=yes (26.0/1.0)  
## (contact = cellular) and (month = oct) => .outcome=yes (63.0/11.0)  
## (housing = no) and (loan = no) and (balance >= 0.200361) and (month = feb) => .outcome=yes (34.0/3.0)  
## (previous <= -0.223501) => .outcome=yes (52.0/11.0)  
## (contact = cellular) and (month = mar) => .outcome=yes (30.0/5.0)  
## (contact = cellular) and (month = apr) and (day >= 0.767556) => .outcome=yes (43.0/7.0)  
## (housing = no) and (age <= 0.205767) and (job = student) => .outcome=yes (34.0/8.0)  
## (contact = cellular) and (month = jun) and (housing = no) => .outcome=yes (31.0/4.0)  
## (duration >= 0.231419) and (age >= 0.639134) => .outcome=yes (26.0/6.0)  
## (duration >= 0.272981) and (pdays <= -0.198905) => .outcome=yes (23.0/7.0)  
## (previous >= 0.205955) and (month = aug) and (pdays >= 0.432551) => .outcome=yes (10.0/0.0)  
## => .outcome=no (1536.0/214.0)  
##   
## Number of Rules : 17

# 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 982 34  
## yes 214 127  
##   
## Accuracy : 0.8172   
## 95% CI : (0.7956, 0.8375)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.411   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8211   
## Specificity : 0.7888   
## Pos Pred Value : 0.9665   
## Neg Pred Value : 0.3724   
## Prevalence : 0.8814   
## Detection Rate : 0.7237   
## Detection Prevalence : 0.7487   
## Balanced Accuracy : 0.8049   
##   
## 'Positive' Class : no   
##

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

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(log\_mod, test[-17])  
 actual<- test$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] 0.505032

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

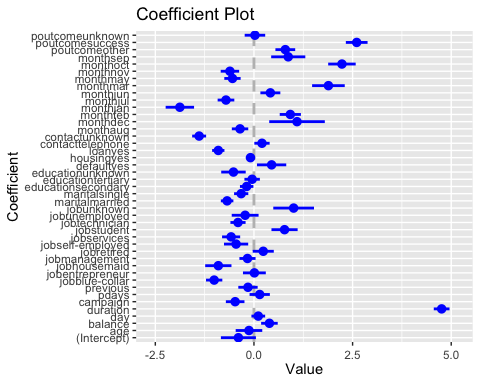
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2476 -0.6504 -0.1632 0.6223 2.6878   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.39412 0.44375 -0.888 0.374455   
## age -0.12636 0.33686 -0.375 0.707580   
## balance 0.39205 0.21036 1.864 0.062363 .   
## day 0.10983 0.17412 0.631 0.528177   
## duration 4.75724 0.19959 23.835 < 2e-16 \*\*\*  
## campaign -0.47629 0.23575 -2.020 0.043352 \*   
## pdays 0.14580 0.25709 0.567 0.570635   
## previous -0.15146 0.24517 -0.618 0.536727   
## jobblue-collar -1.00847 0.20665 -4.880 1.06e-06 \*\*\*  
## jobentrepreneur 0.01082 0.28949 0.037 0.970192   
## jobhousemaid -0.90297 0.33291 -2.712 0.006680 \*\*   
## jobmanagement -0.16403 0.20460 -0.802 0.422727   
## jobretired 0.23554 0.26711 0.882 0.377871   
## jobself-employed -0.45286 0.30689 -1.476 0.140043   
## jobservices -0.57888 0.22930 -2.525 0.011584 \*   
## jobstudent 0.77532 0.33172 2.337 0.019426 \*   
## jobtechnician -0.40443 0.19201 -2.106 0.035181 \*   
## jobunemployed -0.22326 0.34019 -0.656 0.511642   
## jobunknown 1.00580 0.51614 1.949 0.051333 .   
## maritalmarried -0.68041 0.15898 -4.280 1.87e-05 \*\*\*  
## maritalsingle -0.32359 0.18013 -1.796 0.072426 .   
## educationsecondary -0.18466 0.17067 -1.082 0.279262   
## educationtertiary -0.04605 0.19560 -0.235 0.813884   
## educationunknown -0.51945 0.31025 -1.674 0.094068 .   
## defaultyes 0.44732 0.37112 1.205 0.228074   
## housingyes -0.08794 0.11514 -0.764 0.445008   
## loanyes -0.90411 0.15580 -5.803 6.51e-09 \*\*\*  
## contacttelephone 0.20348 0.19401 1.049 0.294260   
## contactunknown -1.38952 0.17669 -7.864 3.72e-15 \*\*\*  
## monthaug -0.35270 0.20814 -1.695 0.090162 .   
## monthdec 1.09158 0.70444 1.550 0.121243   
## monthfeb 0.92158 0.26832 3.435 0.000593 \*\*\*  
## monthjan -1.87818 0.36012 -5.215 1.83e-07 \*\*\*  
## monthjul -0.70914 0.21144 -3.354 0.000797 \*\*\*  
## monthjun 0.41593 0.25272 1.646 0.099801 .   
## monthmar 1.88747 0.41363 4.563 5.04e-06 \*\*\*  
## monthmay -0.54456 0.20708 -2.630 0.008547 \*\*   
## monthnov -0.60909 0.23363 -2.607 0.009132 \*\*   
## monthoct 2.23002 0.34860 6.397 1.58e-10 \*\*\*  
## monthsep 0.86911 0.43274 2.008 0.044603 \*   
## poutcomeother 0.79474 0.25077 3.169 0.001529 \*\*   
## poutcomesuccess 2.60434 0.27532 9.459 < 2e-16 \*\*\*  
## poutcomeunknown 0.02486 0.25994 0.096 0.923807   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4385.9 on 3163 degrees of freedom  
## Residual deviance: 2685.3 on 3121 degrees of freedom  
## AIC: 2771.3  
##   
## Number of Fisher Scoring iterations: 5

### 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 1001 80  
## yes 195 81  
##   
## Accuracy : 0.7973   
## 95% CI : (0.775, 0.8184)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2598   
##   
## Mcnemar's Test P-Value : 6.223e-12   
##   
## Sensitivity : 0.8370   
## Specificity : 0.5031   
## Pos Pred Value : 0.9260   
## Neg Pred Value : 0.2935   
## Prevalence : 0.8814   
## Detection Rate : 0.7377   
## Detection Prevalence : 0.7966   
## Balanced Accuracy : 0.6700   
##   
## '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] 0.3671161

# 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 996 58  
## yes 200 103  
##   
## Accuracy : 0.8099   
## 95% CI : (0.788, 0.8304)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.342   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8328   
## Specificity : 0.6398   
## Pos Pred Value : 0.9450   
## Neg Pred Value : 0.3399   
## Prevalence : 0.8814   
## Detection Rate : 0.7340   
## Detection Prevalence : 0.7767   
## Balanced Accuracy : 0.7363   
##   
## '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] 0.4466008

# 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 986 36  
## yes 210 125  
##   
## Accuracy : 0.8187   
## 95% CI : (0.7972, 0.8389)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4094   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8244   
## Specificity : 0.7764   
## Pos Pred Value : 0.9648   
## Neg Pred Value : 0.3731   
## Prevalence : 0.8814   
## Detection Rate : 0.7266   
## Detection Prevalence : 0.7531   
## Balanced Accuracy : 0.8004   
##   
## '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] 0.5068865

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