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  
# 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:

## SMOTE SAMPLE MAJOR/MINOR - each 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  
x=smote\_train[,-17]  
trainsv=smote\_train  
train=smote\_train  
y=smote\_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))  
# 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 1066 73  
## yes 130 88  
##   
## Accuracy : 0.8504   
## 95% CI : (0.8303, 0.869)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9997   
##   
## Kappa : 0.3797   
##   
## Mcnemar's Test P-Value : 8.479e-05   
##   
## Sensitivity : 0.8913   
## Specificity : 0.5466   
## Pos Pred Value : 0.9359   
## Neg Pred Value : 0.4037   
## Prevalence : 0.8814   
## Detection Rate : 0.7856   
## Detection Prevalence : 0.8394   
## Balanced Accuracy : 0.7189   
##   
## '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] 0.4590605

# 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 1024 48  
## yes 172 113  
##   
## Accuracy : 0.8379   
## 95% CI : (0.8172, 0.8571)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4186   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8562   
## Specificity : 0.7019   
## Pos Pred Value : 0.9552   
## Neg Pred Value : 0.3965   
## Prevalence : 0.8814   
## Detection Rate : 0.7546   
## Detection Prevalence : 0.7900   
## Balanced Accuracy : 0.7790   
##   
## '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] 0.508775

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

### to see model results:

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

##   
## Call:  
## (function (x, y, trials = 1, rules = FALSE, weights = NULL, control  
## = FALSE, sample = 0, earlyStopping = TRUE, label = "outcome", seed =  
## 2372L), verbose = FALSE)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Wed Jul 29 01:41:58 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 2520 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## duration <= 0.1995863:  
## :...month in {dec,mar,oct,sep}:  
## : :...campaign > 0.303829: no (4)  
## : : campaign <= 0.303829:  
## : : :...duration <= 0.03826267: no (3)  
## : : duration > 0.03826267: yes (48/6)  
## : month in {apr,aug,feb,jan,jul,jun,may,nov}:  
## : :...poutcome = success:  
## : :...pdays <= 0.3879324: yes (11)  
## : : pdays > 0.3879324: no (11/3)  
## : poutcome in {failure,other,unknown}:  
## : :...month in {apr,feb}:  
## : :...loan = yes: no (11)  
## : : loan = no:  
## : : :...age > 0.5740556: yes (5)  
## : : age <= 0.5740556:  
## : : :...marital = divorced:  
## : : :...housing = yes: yes (3)  
## : : : housing = no:  
## : : : :...balance <= 0.2205725: no (6)  
## : : : balance > 0.2205725: yes (4)  
## : : marital in {married,single}:  
## : : :...contact = unknown: yes (2)  
## : : contact in {cellular,telephone}:  
## : : :...pdays > 0.1573034: no (27)  
## : : pdays <= 0.1573034:  
## : : :...pdays <= 0.004498994: no (40/6)  
## : : pdays > 0.004498994: yes (3)  
## : month in {aug,jan,jul,jun,may,nov}:  
## : :...pdays <= 0.01306889: no (706/18)  
## : pdays > 0.01306889:  
## : :...poutcome = unknown: yes (10)  
## : poutcome in {failure,other}:  
## : :...balance <= 0.1500748:  
## : :...pdays <= 0.1800783: yes (2)  
## : : pdays > 0.1800783: no (80)  
## : balance > 0.1500748:  
## : :...previous > 0.4983276: no (13)  
## : previous <= 0.4983276:  
## : :...previous <= 0.2897677:  
## : :...poutcome = failure: no (14/1)  
## : : poutcome = other: yes (3/1)  
## : previous > 0.2897677:  
## : :...day <= 0.2152035: no (2)  
## : day > 0.2152035: yes (7)  
## duration > 0.1995863:  
## :...previous > 0:  
## :...poutcome in {success,unknown}: yes (318/8)  
## : poutcome in {failure,other}:  
## : :...previous <= 0.1992719: yes (67)  
## : previous > 0.1992719:  
## : :...month = jan: no (6)  
## : month in {dec,jul}: yes (11/1)  
## : month = aug:  
## : :...job in {admin.,blue-collar,entrepreneur,management,retired,  
## : : : technician,unemployed,unknown}: yes (22)  
## : : job in {housemaid,self-employed,services,student}: no (5)  
## : month = mar:  
## : :...age <= 0.6351154: yes (4)  
## : : age > 0.6351154: no (2)  
## : month = nov:  
## : :...poutcome = failure: no (14/3)  
## : : poutcome = other: yes (14/2)  
## : month = oct:  
## : :...balance <= 0.4998358: yes (22)  
## : : balance > 0.4998358: no (2)  
## : month = sep:  
## : :...age <= 0.4113255: yes (7)  
## : : age > 0.4113255: no (2)  
## : month = apr:  
## : :...duration > 0.6232215: yes (7)  
## : : duration <= 0.6232215:  
## : : :...campaign <= 0.2182717: no (21/1)  
## : : campaign > 0.2182717: yes (5)  
## : month = feb:  
## : :...campaign > 0.2: no (2)  
## : : campaign <= 0.2:  
## : : :...contact in {cellular,unknown}: yes (12/1)  
## : : contact = telephone: no (1)  
## : month = jun:  
## : :...poutcome = other: yes (9)  
## : : poutcome = failure:  
## : : :...duration <= 0.3830353: no (3)  
## : : duration > 0.3830353: yes (5)  
## : month = may:  
## : :...campaign > 0.1019187: yes (21/1)  
## : campaign <= 0.1019187:  
## : :...loan = yes: yes (7/1)  
## : loan = no:  
## : :...balance <= 0.0867336: no (12)  
## : balance > 0.0867336:  
## : :...education in {primary,secondary}: yes (11/1)  
## : education in {tertiary,unknown}:  
## : :...poutcome = failure: no (7)  
## : poutcome = other: yes (1)  
## previous <= 0:  
## :...duration <= 0.4707389:  
## :...month in {dec,mar,oct}: yes (33/2)  
## : month in {apr,aug,feb,jan,jul,jun,may,nov,sep}:  
## : :...default = yes:  
## : :...month in {apr,aug,feb,jan,nov,sep}: yes (6)  
## : : month in {jul,jun,may}: no (5)  
## : default = no:  
## : :...contact = unknown:  
## : :...month in {apr,aug,feb,nov}: yes (8)  
## : : month in {jan,jul,jun,may,sep}: no (134/8)  
## : contact = cellular:  
## : :...month = jun: yes (10)  
## : : month in {apr,aug,feb,jan,jul,may,nov,sep}:  
## : : :...age <= 0.5886043: no (236/50)  
## : : age > 0.5886043:  
## : : :...day > 0.5333334: yes (7)  
## : : day <= 0.5333334:  
## : : :...duration <= 0.2537265: yes (5/1)  
## : : duration > 0.2537265: no (5)  
## : contact = telephone:  
## : :...balance > 0.4303635: no (7)  
## : balance <= 0.4303635:  
## : :...education = tertiary: yes (10)  
## : education in {primary,secondary,unknown}:  
## : :...duration > 0.3383687: yes (6)  
## : duration <= 0.3383687:  
## : :...campaign <= 0.1490633: no (10)  
## : campaign > 0.1490633: yes (5/2)  
## duration > 0.4707389:  
## :...default = yes: yes (21/1)  
## default = no:  
## :...duration <= 0.7042399:  
## :...job = unknown: yes (0)  
## : job in {entrepreneur,self-employed,student,  
## : : unemployed}: no (20/1)  
## : job in {admin.,blue-collar,housemaid,management,retired,  
## : : services,technician}:  
## : :...month in {dec,feb,jun,mar}: yes (27/3)  
## : month in {jan,sep}: no (4/2)  
## : month = oct:  
## : :...age <= 0.4819314: yes (2)  
## : : age > 0.4819314: no (3)  
## : month = aug:  
## : :...loan = no:  
## : : :...campaign <= 0.04983635: no (3/1)  
## : : : campaign > 0.04983635: yes (21/2)  
## : : loan = yes:  
## : : :...balance <= 0.06515702: no (4)  
## : : balance > 0.06515702: yes (3)  
## : month = nov:  
## : :...contact in {telephone,unknown}: yes (5)  
## : : contact = cellular:  
## : : :...marital in {divorced,single}: no (6/1)  
## : : marital = married: yes (4/1)  
## : month = apr:  
## : :...housing = no: yes (6)  
## : : housing = yes:  
## : : :...day <= 0.4539885: yes (2)  
## : : day > 0.4539885:  
## : : :...contact in {cellular,unknown}: no (3)  
## : : contact = telephone: yes (1)  
## : month = jul:  
## : :...day > 0.6912588: no (9/1)  
## : : day <= 0.6912588:  
## : : :...marital in {divorced,single}: yes (7)  
## : : marital = married:  
## : : :...contact in {cellular,  
## : : : telephone}: no (8/1)  
## : : contact = unknown: yes (1)  
## : month = may:  
## : :...marital = divorced: yes (3)  
## : marital = married:  
## : :...job in {admin.,management}: yes (12/3)  
## : : job in {blue-collar,housemaid,retired,  
## : : : services}: no (14/1)  
## : : job = technician:  
## : : :...balance <= 0.0377252: yes (2)  
## : : balance > 0.0377252: no (4)  
## : marital = single:  
## : :...contact in {cellular,telephone}: yes (4)  
## : contact = unknown:  
## : :...age <= 0.1617647: yes (2)  
## : age > 0.1617647: no (3)  
## duration > 0.7042399:  
## :...loan = no: yes (141/15)  
## loan = yes:  
## :...balance > 0.09335612: yes (23)  
## balance <= 0.09335612:  
## :...contact = telephone: no (4)  
## contact in {cellular,unknown}:  
## :...balance > 0.0907458: no (5)  
## balance <= 0.0907458:  
## :...age <= 0.153907: no (3)  
## age > 0.153907:  
## :...duration <= 0.7497415: no (3/1)  
## duration > 0.7497415: yes (15)  
##   
##   
## Evaluation on training data (2520 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 103 151( 6.0%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 1388 52 (a): class no  
## 99 981 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% duration  
## 75.40% month  
## 62.70% poutcome  
## 61.27% previous  
## 36.87% pdays  
## 33.81% default  
## 23.61% contact  
## 15.44% age  
## 14.44% loan  
## 11.51% balance  
## 8.33% job  
## 7.70% campaign  
## 6.15% marital  
## 2.26% day  
## 1.98% education  
## 0.99% housing  
##   
##   
## Time: 0.0 secs

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

## model 3 - JRip (rule learner)

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## JR\_pred no yes  
## no 1019 44  
## yes 177 117  
##   
## Accuracy : 0.8371   
## 95% CI : (0.8164, 0.8564)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4263   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8520   
## Specificity : 0.7267   
## Pos Pred Value : 0.9586   
## Neg Pred Value : 0.3980   
## Prevalence : 0.8814   
## Detection Rate : 0.7509   
## Detection Prevalence : 0.7833   
## Balanced Accuracy : 0.7894   
##   
## '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] 0.5144209

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.282506) and (pdays >= 0.000579) => .outcome=yes (475.0/50.0)  
## (duration >= 0.206945) and (pdays >= 0.000551) => .outcome=yes (131.0/32.0)  
## (duration >= 0.704746) => .outcome=yes (199.0/29.0)  
## (duration >= 0.416552) and (campaign >= 0.1044) => .outcome=yes (62.0/15.0)  
## (duration >= 0.154179) and (campaign <= 0.094979) and (campaign >= 0.000568) => .outcome=yes (61.0/0.0)  
## (duration >= 0.344364) and (balance >= 0.088825) and (day <= 0.266197) => .outcome=yes (29.0/7.0)  
## (duration >= 0.154179) and (campaign <= 0.000018) and (day >= 0.640731) and (month = apr) => .outcome=yes (19.0/0.0)  
## (duration >= 0.146033) and (month = oct) and (age <= 0.529784) => .outcome=yes (23.0/1.0)  
## (duration >= 0.185109) and (age >= 0.617647) => .outcome=yes (27.0/9.0)  
## (pdays >= 0.002809) and (pdays <= 0.280899) and (age >= 0.193369) => .outcome=yes (34.0/5.0)  
## (duration >= 0.426545) and (balance <= 0.03998) => .outcome=yes (18.0/4.0)  
## => .outcome=no (1442.0/154.0)  
##   
## Number of Rules : 12

# all for outcome yes

## model 4 - Logistic Regression

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## log\_pred no yes  
## no 1040 43  
## yes 156 118  
##   
## Accuracy : 0.8534   
## 95% CI : (0.8334, 0.8718)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9992   
##   
## Kappa : 0.4621   
##   
## Mcnemar's Test P-Value : 2.03e-15   
##   
## Sensitivity : 0.8696   
## Specificity : 0.7329   
## Pos Pred Value : 0.9603   
## Neg Pred Value : 0.4307   
## Prevalence : 0.8814   
## Detection Rate : 0.7664   
## Detection Prevalence : 0.7981   
## Balanced Accuracy : 0.8012   
##   
## '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] 0.5416793

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

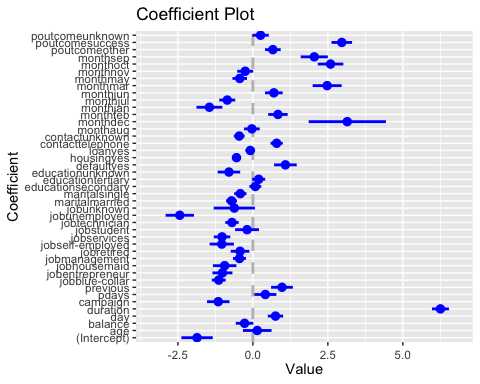
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.4463 -0.5266 -0.2373 0.4875 2.5820   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.86189 0.52327 -3.558 0.000373 \*\*\*  
## age 0.14538 0.47805 0.304 0.761049   
## balance -0.27698 0.29250 -0.947 0.343658   
## day 0.75437 0.25600 2.947 0.003211 \*\*   
## duration 6.25523 0.28256 22.137 < 2e-16 \*\*\*  
## campaign -1.15435 0.37455 -3.082 0.002056 \*\*   
## pdays 0.41614 0.36814 1.130 0.258314   
## previous 0.97033 0.36376 2.667 0.007642 \*\*   
## jobblue-collar -1.13923 0.23157 -4.920 8.67e-07 \*\*\*  
## jobentrepreneur -1.01592 0.33359 -3.045 0.002324 \*\*   
## jobhousemaid -0.94099 0.39208 -2.400 0.016394 \*   
## jobmanagement -0.44601 0.21605 -2.064 0.038979 \*   
## jobretired -0.42724 0.31117 -1.373 0.169742   
## jobself-employed -1.03696 0.40430 -2.565 0.010322 \*   
## jobservices -1.02890 0.27502 -3.741 0.000183 \*\*\*  
## jobstudent -0.19390 0.39807 -0.487 0.626184   
## jobtechnician -0.69664 0.22253 -3.131 0.001744 \*\*   
## jobunemployed -2.43765 0.47252 -5.159 2.48e-07 \*\*\*  
## jobunknown -0.61864 0.68741 -0.900 0.368142   
## maritalmarried -0.71126 0.18247 -3.898 9.70e-05 \*\*\*  
## maritalsingle -0.41839 0.20499 -2.041 0.041254 \*   
## educationsecondary 0.08179 0.19853 0.412 0.680348   
## educationtertiary 0.19376 0.21786 0.889 0.373812   
## educationunknown -0.79788 0.37429 -2.132 0.033031 \*   
## defaultyes 1.08434 0.37813 2.868 0.004136 \*\*   
## housingyes -0.54613 0.12938 -4.221 2.43e-05 \*\*\*  
## loanyes -0.08551 0.15863 -0.539 0.589820   
## contacttelephone 0.79320 0.20342 3.899 9.65e-05 \*\*\*  
## contactunknown -0.45536 0.18015 -2.528 0.011483 \*   
## monthaug -0.03311 0.26095 -0.127 0.899019   
## monthdec 3.14807 1.28720 2.446 0.014458 \*   
## monthfeb 0.83540 0.32524 2.569 0.010212 \*   
## monthjan -1.44953 0.43162 -3.358 0.000784 \*\*\*  
## monthjul -0.85445 0.26786 -3.190 0.001423 \*\*   
## monthjun 0.69972 0.29400 2.380 0.017312 \*   
## monthmar 2.47684 0.48358 5.122 3.03e-07 \*\*\*  
## monthmay -0.43815 0.24228 -1.808 0.070537 .   
## monthnov -0.25574 0.26637 -0.960 0.337014   
## monthoct 2.59035 0.42566 6.086 1.16e-09 \*\*\*  
## monthsep 2.04689 0.45214 4.527 5.98e-06 \*\*\*  
## poutcomeother 0.66641 0.26111 2.552 0.010704 \*   
## poutcomesuccess 2.96503 0.34330 8.637 < 2e-16 \*\*\*  
## poutcomeunknown 0.25639 0.27358 0.937 0.348674   
## ---  
## 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: 1866.3 on 2477 degrees of freedom  
## AIC: 1952.3  
##   
## 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 1041 58  
## yes 155 103  
##   
## Accuracy : 0.843   
## 95% CI : (0.8226, 0.862)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4047   
##   
## Mcnemar's Test P-Value : 4.774e-11   
##   
## Sensitivity : 0.8704   
## Specificity : 0.6398   
## Pos Pred Value : 0.9472   
## Neg Pred Value : 0.3992   
## Prevalence : 0.8814   
## Detection Rate : 0.7671   
## Detection Prevalence : 0.8099   
## Balanced Accuracy : 0.7551   
##   
## '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] 0.4945885

# 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 999 47  
## yes 197 114  
##   
## Accuracy : 0.8202   
## 95% CI : (0.7987, 0.8403)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3872   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8353   
## Specificity : 0.7081   
## Pos Pred Value : 0.9551   
## Neg Pred Value : 0.3666   
## Prevalence : 0.8814   
## Detection Rate : 0.7362   
## Detection Prevalence : 0.7708   
## Balanced Accuracy : 0.7717   
##   
## '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] 0.4854039

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

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

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

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

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## sp no yes  
## no 1039 46  
## yes 157 115  
##   
## Accuracy : 0.8504   
## 95% CI : (0.8303, 0.869)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9997   
##   
## Kappa : 0.4491   
##   
## Mcnemar's Test P-Value : 1.159e-14   
##   
## Sensitivity : 0.8687   
## Specificity : 0.7143   
## Pos Pred Value : 0.9576   
## Neg Pred Value : 0.4228   
## Prevalence : 0.8814   
## Detection Rate : 0.7657   
## Detection Prevalence : 0.7996   
## Balanced Accuracy : 0.7915   
##   
## '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] 0.5334185

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