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

jean wills

13/07/2020

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

library(plyr)  
library(dplyr)  
BM\_mini <- read.csv("/Users/jeanwills/Desktop/CKME136/1\_data/bank.csv", header=T, sep = ";", stringsAsFactors = T, na.strings = "NA")  
# Step 1 - NUMERIC DATA Cleaning - change numeric data outside the 2.5% and the 97.5% percentiles to this maximum/minimum value  
BM<-BM\_mini  
  
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:

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

table(BM\_train$y)

##   
## no yes   
## 2804 360

x=BM\_train[,-17]  
trainsv=BM\_train  
train=BM\_train  
y=BM\_train$y  
test\_noy=BM\_test[,-17]  
test\_labels=BM\_test$y

# step 4 - now run models

## MODEL 1 NAIVE BAYES - uses caret

# FINAL PARAMETER VALUES USED WERE fL = 0, usekernel = TRUE and adjust = 1.  
library(klaR)  
library(caret)  
library(e1071)  
library(MASS)  
set.seed(520)  
nb\_grid <- expand.grid(fL= 0, usekernel= c("TRUE"), adjust=1)  
# nb\_grid <- expand.grid(fL= c(0,1), usekernel= c("TRUE", "FALSE"), adjust=c(0,1,2,3))  
nab\_mod<- train(x=x, y=y, method="nb", metric="ROC", tuneGrid=nb\_grid, trControl = trainControl(method="repeatedcv", number=10, repeats=10, classProbs=TRUE, summaryFunction = twoClassSummary))  
# to see model results:  
# nab\_mod  
# predict output using test set - even though we did 10x10 cv above, we are also using 'validation' set of 30% of the data:  
nab\_pred<- predict(nab\_mod, test\_noy)  
s<-table(nab\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## nab\_pred no yes  
## no 1160 119  
## yes 36 42  
##   
## Accuracy : 0.8858   
## 95% CI : (0.8676, 0.9022)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.3252   
##   
## Kappa : 0.297   
##   
## Mcnemar's Test P-Value : 4.506e-11   
##   
## Sensitivity : 0.9699   
## Specificity : 0.2609   
## Pos Pred Value : 0.9070   
## Neg Pred Value : 0.5385   
## Prevalence : 0.8814   
## Detection Rate : 0.8548   
## Detection Prevalence : 0.9425   
## Balanced Accuracy : 0.6154   
##   
## '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] 0.3410805

# 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 1137 94  
## yes 59 67  
##   
## Accuracy : 0.8873   
## 95% CI : (0.8692, 0.9036)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.266597   
##   
## Kappa : 0.4049   
##   
## Mcnemar's Test P-Value : 0.005983   
##   
## Sensitivity : 0.9507   
## Specificity : 0.4161   
## Pos Pred Value : 0.9236   
## Neg Pred Value : 0.5317   
## Prevalence : 0.8814   
## Detection Rate : 0.8379   
## Detection Prevalence : 0.9071   
## Balanced Accuracy : 0.6834   
##   
## '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] 0.4662806

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

### to see model results:

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

##   
## Call:  
## (function (x, y, trials = 1, rules = FALSE, weights = NULL, control  
## 0.25, minCases = 2, fuzzyThreshold = FALSE, sample = 0, earlyStopping =  
## TRUE, label = "outcome", seed = 2372L), verbose = FALSE)  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Wed Jul 29 00:59:32 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 3164 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## poutcome = success:  
## :...balance > 0.8575091: no (5)  
## : balance <= 0.8575091:  
## : :...duration > 0.1602896: yes (68/15)  
## : duration <= 0.1602896:  
## : :...housing = yes: no (7)  
## : housing = no:  
## : :...age <= 0.3676471: yes (6/1)  
## : age > 0.3676471: no (5)  
## poutcome in {failure,other,unknown}:  
## :...duration > 0.3681489:  
## :...duration <= 0.647363: no (382/75)  
## : duration > 0.647363:  
## : :...contact in {telephone,unknown}:  
## : :...age > 0.5882353: yes (5)  
## : : age <= 0.5882353:  
## : : :...marital in {divorced,single}: yes (27/13)  
## : : marital = married: no (50/9)  
## : contact = cellular:  
## : :...job in {entrepreneur,unknown}: no (10/4)  
## : job in {self-employed,services,technician}: yes (38/8)  
## : job = admin.:  
## : :...marital in {divorced,single}: yes (8/2)  
## : : marital = married: no (3)  
## : job = housemaid:  
## : :...day <= 0.4666667: no (3)  
## : : day > 0.4666667: yes (2)  
## : job = retired:  
## : :...poutcome = failure: yes (1)  
## : : poutcome in {other,unknown}: no (3)  
## : job = student:  
## : :...age <= 0.1470588: yes (2)  
## : : age > 0.1470588: no (2)  
## : job = unemployed:  
## : :...balance <= 0.09132664: no (3)  
## : : balance > 0.09132664: yes (2)  
## : job = blue-collar:  
## : :...month in {aug,jun,may}: yes (10/2)  
## : : month in {dec,feb,jan,mar,oct,sep}: no (3)  
## : : month = apr:  
## : : :...age <= 0.2205882: no (3)  
## : : : age > 0.2205882: yes (3)  
## : : month = jul:  
## : : :...age <= 0.3382353: yes (2)  
## : : : age > 0.3382353: no (4)  
## : : month = nov:  
## : : :...age <= 0.25: no (3)  
## : : age > 0.25: yes (2)  
## : job = management:  
## : :...previous > 0.6: no (3)  
## : previous <= 0.6:  
## : :...duration > 0.8076525: yes (17/4)  
## : duration <= 0.8076525:  
## : :...day <= 0.5333334: yes (10/3)  
## : day > 0.5333334: no (9/3)  
## duration <= 0.3681489:  
## :...month in {aug,jan,jul,jun,may,nov}: no (2121/54)  
## month in {mar,oct}:  
## :...duration <= 0.0713547: no (11)  
## : duration > 0.0713547:  
## : :...job in {blue-collar,entrepreneur,retired,self-employed,  
## : : unknown}: no (14/3)  
## : job in {housemaid,services,student,unemployed}: yes (3)  
## : job = admin.:  
## : :...duration <= 0.1282316: no (4)  
## : : duration > 0.1282316: yes (7/1)  
## : job = management:  
## : :...month = mar: yes (5)  
## : : month = oct: no (10/4)  
## : job = technician:  
## : :...marital in {divorced,married}: yes (5)  
## : marital = single: no (2)  
## month in {apr,dec,feb,sep}:  
## :...day > 0.6333333:  
## :...duration > 0.2275078: yes (12/1)  
## : duration <= 0.2275078:  
## : :...balance <= 0.04315317: yes (3)  
## : balance > 0.04315317: no (28/1)  
## day <= 0.6333333:  
## :...job in {admin.,blue-collar,housemaid,services,student,  
## : unknown}: no (113)  
## job in {entrepreneur,management,retired,self-employed,  
## : technician,unemployed}:  
## :...month in {apr,dec,sep}: no (64/9)  
## month = feb:  
## :...day > 0.2666667: yes (7)  
## day <= 0.2666667:  
## :...balance <= 0.270455: no (45)  
## balance > 0.270455:  
## :...pdays <= 0.07865169: no (7/1)  
## pdays > 0.07865169: yes (2)  
##   
##   
## Evaluation on training data (3164 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 52 213( 6.7%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 2754 50 (a): class no  
## 163 197 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% poutcome  
## 99.84% duration  
## 78.79% month  
## 13.72% job  
## 9.64% day  
## 7.21% contact  
## 5.72% balance  
## 3.60% age  
## 3.00% marital  
## 1.23% previous  
## 0.57% housing  
## 0.28% pdays  
##   
##   
## Time: 0.0 secs

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

## model 3 - JRip (rule learner)

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## JR\_pred no yes  
## no 1155 103  
## yes 41 58  
##   
## Accuracy : 0.8939   
## 95% CI : (0.8763, 0.9098)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.08141   
##   
## Kappa : 0.3911   
##   
## Mcnemar's Test P-Value : 3.709e-07   
##   
## Sensitivity : 0.9657   
## Specificity : 0.3602   
## Pos Pred Value : 0.9181   
## Neg Pred Value : 0.5859   
## Prevalence : 0.8814   
## Detection Rate : 0.8511   
## Detection Prevalence : 0.9270   
## Balanced Accuracy : 0.6630   
##   
## '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] 0.4360599

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.199586) and (pdays >= 0.106742) and (poutcome = success) => .outcome=yes (65.0/17.0)  
## (duration >= 0.78697) => .outcome=yes (130.0/54.0)  
## => .outcome=no (2969.0/236.0)  
##   
## Number of Rules : 3

# all for outcome yes

## model 4 - Logistic Regression

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## log\_pred no yes  
## no 1157 97  
## yes 39 64  
##   
## Accuracy : 0.8998   
## 95% CI : (0.8826, 0.9152)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.01813   
##   
## Kappa : 0.4323   
##   
## Mcnemar's Test P-Value : 1.02e-06   
##   
## Sensitivity : 0.9674   
## Specificity : 0.3975   
## Pos Pred Value : 0.9226   
## Neg Pred Value : 0.6214   
## Prevalence : 0.8814   
## Detection Rate : 0.8526   
## Detection Prevalence : 0.9241   
## Balanced Accuracy : 0.6825   
##   
## '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] 0.4699654

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

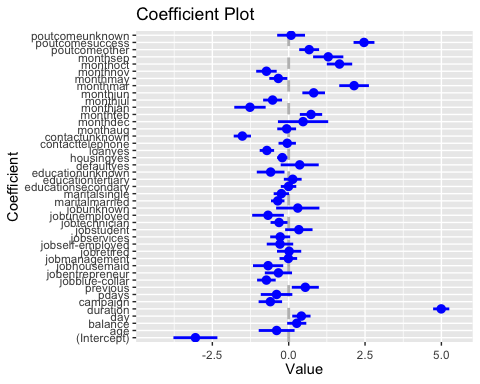
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.9844 -0.3578 -0.2318 -0.1387 3.0677   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.054544 0.720012 -4.242 2.21e-05 \*\*\*  
## age -0.394788 0.587040 -0.673 0.50126   
## balance 0.263278 0.313407 0.840 0.40088   
## day 0.415660 0.299384 1.388 0.16502   
## duration 4.994198 0.264956 18.849 < 2e-16 \*\*\*  
## campaign -0.600947 0.384369 -1.563 0.11794   
## pdays -0.397519 0.516871 -0.769 0.44184   
## previous 0.543751 0.442628 1.228 0.21927   
## jobblue-collar -0.730066 0.304449 -2.398 0.01649 \*   
## jobentrepreneur -0.336412 0.445615 -0.755 0.45029   
## jobhousemaid -0.674314 0.494799 -1.363 0.17294   
## jobmanagement -0.011299 0.288364 -0.039 0.96875   
## jobretired 0.013945 0.396373 0.035 0.97194   
## jobself-employed -0.285468 0.433229 -0.659 0.50994   
## jobservices -0.278705 0.326904 -0.853 0.39390   
## jobstudent 0.332176 0.448745 0.740 0.45916   
## jobtechnician -0.316011 0.277186 -1.140 0.25426   
## jobunemployed -0.677087 0.518774 -1.305 0.19184   
## jobunknown 0.297041 0.708324 0.419 0.67495   
## maritalmarried -0.358928 0.220324 -1.629 0.10329   
## maritalsingle -0.239683 0.252277 -0.950 0.34207   
## educationsecondary -0.005298 0.252901 -0.021 0.98329   
## educationtertiary 0.132391 0.291206 0.455 0.64938   
## educationunknown -0.591393 0.453282 -1.305 0.19200   
## defaultyes 0.360717 0.622646 0.579 0.56237   
## housingyes -0.209800 0.167967 -1.249 0.21165   
## loanyes -0.713785 0.236169 -3.022 0.00251 \*\*   
## contacttelephone -0.048248 0.284621 -0.170 0.86539   
## contactunknown -1.517847 0.282609 -5.371 7.84e-08 \*\*\*  
## monthaug -0.066254 0.308196 -0.215 0.82979   
## monthdec 0.470997 0.822989 0.572 0.56712   
## monthfeb 0.730207 0.365789 1.996 0.04591 \*   
## monthjan -1.268208 0.513578 -2.469 0.01354 \*   
## monthjul -0.527439 0.309724 -1.703 0.08858 .   
## monthjun 0.817838 0.370491 2.207 0.02728 \*   
## monthmar 2.141497 0.485878 4.407 1.05e-05 \*\*\*  
## monthmay -0.339708 0.295200 -1.151 0.24983   
## monthnov -0.728728 0.335720 -2.171 0.02996 \*   
## monthoct 1.663709 0.415494 4.004 6.22e-05 \*\*\*  
## monthsep 1.294313 0.494933 2.615 0.00892 \*\*   
## poutcomeother 0.669971 0.326248 2.054 0.04002 \*   
## poutcomesuccess 2.465824 0.341918 7.212 5.52e-13 \*\*\*  
## poutcomeunknown 0.078652 0.455811 0.173 0.86300   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2242.3 on 3163 degrees of freedom  
## Residual deviance: 1459.5 on 3121 degrees of freedom  
## AIC: 1545.5  
##   
## 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 1158 103  
## yes 38 58  
##   
## Accuracy : 0.8961   
## 95% CI : (0.8786, 0.9118)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.04884   
##   
## Kappa : 0.398   
##   
## Mcnemar's Test P-Value : 7.055e-08   
##   
## Sensitivity : 0.9682   
## Specificity : 0.3602   
## Pos Pred Value : 0.9183   
## Neg Pred Value : 0.6042   
## Prevalence : 0.8814   
## Detection Rate : 0.8534   
## Detection Prevalence : 0.9293   
## Balanced Accuracy : 0.6642   
##   
## '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] 0.4345147

# 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 1133 99  
## yes 63 62  
##   
## Accuracy : 0.8806   
## 95% CI : (0.8622, 0.8974)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.554252   
##   
## Kappa : 0.368   
##   
## Mcnemar's Test P-Value : 0.005962   
##   
## Sensitivity : 0.9473   
## Specificity : 0.3851   
## Pos Pred Value : 0.9196   
## Neg Pred Value : 0.4960   
## Prevalence : 0.8814   
## Detection Rate : 0.8349   
## Detection Prevalence : 0.9079   
## Balanced Accuracy : 0.6662   
##   
## '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] 0.4288036

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

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

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

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

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## sp no yes  
## no 1183 136  
## yes 13 25  
##   
## Accuracy : 0.8902   
## 95% CI : (0.8723, 0.9063)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.1673   
##   
## Kappa : 0.2157   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9891   
## Specificity : 0.1553   
## Pos Pred Value : 0.8969   
## Neg Pred Value : 0.6579   
## Prevalence : 0.8814   
## Detection Rate : 0.8718   
## Detection Prevalence : 0.9720   
## Balanced Accuracy : 0.5722   
##   
## 'Positive' Class : no   
##

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

# copy in files you need and use test dataset only   
banking<-BM\_test  
  
# the other way is to run 10-fold on the test dataset and take the average of the (10 times) F1 measure   
folds<- createFolds(banking$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(sv\_m, test[-17])  
 actual<- test$y  
 # PPV = TP/(TP+FP)  
 # pos<-posPredValue(table(pred, actual))  
 # I actually want: NPV= TN/(TN+FN) for precision of minority class  
 pr<-negPredValue(table(pred, actual))  
 # pr<-precision(table(pred, actual ))  
 # rec<- recall(table(pred, actual))  
 # i actually want specificity for recall of minority class  
 rec<- specificity(table(pred, actual))  
 F1<- 2 \* pr \* rec /(pr + rec)  
 return(F1)  
 })  
 #  
 # print(results)  
 value<-mean(unlist(results))  
 print(value)

## [1] 0.244443

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