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  
  
# now have file BM ..

## step 1b - BM -> BM\_dummy

## Part 3 - step 2b: Conversion B- Categorical to Numeric - as dummy variables - RESULT is “BM\_dummy or BM\_mini\_dummy”

# original numeric still NOT normalized/scaled  
# dummy coding  
# keep y as factor  
BM\_d <- BM  
# BM\_dummy <- BM  
# now create new attributes for each component in attribute less 1 category  
# for example, marital has 3 attributes so we need 2 dummy variables (each of 0,1)  
# BM\_fact$y<- ifelse(BM\_fact$y==c("yes"), 0, 1)  
#  
BM\_d$job1 <- ifelse(BM\_d$job == c("admin."), 1, 0)  
BM\_d$job2 <- ifelse(BM\_d$job == c("blue-collar"), 1, 0)  
BM\_d$job3 <- ifelse(BM\_d$job == c("entrepreneur"), 1, 0)  
BM\_d$job4 <- ifelse(BM\_d$job == c("housemaid"), 1, 0)  
BM\_d$job5 <- ifelse(BM\_d$job == c("management"), 1, 0)  
BM\_d$job6 <- ifelse(BM\_d$job == c("retired"), 1, 0)  
BM\_d$job7 <- ifelse(BM\_d$job == c("self-employed"), 1, 0)  
BM\_d$job8 <- ifelse(BM\_d$job == c("services"), 1, 0)  
BM\_d$job9 <- ifelse(BM\_d$job == c("student"), 1, 0)  
BM\_d$job10 <- ifelse(BM\_d$job == c("technician"), 1, 0)  
BM\_d$job11 <- ifelse(BM\_d$job == c("unemployed"), 1, 0)  
BM\_d$job12 <- ifelse(BM\_d$job == c("unknown"), 1, 0)  
#  
BM\_d$mar1 <- ifelse(BM\_d$marital== c("divorced"), 1, 0)  
BM\_d$mar2 <- ifelse(BM\_d$marital== c("married"), 1, 0)  
BM\_d$mar3 <- ifelse(BM\_d$marital== c("single"), 1, 0)  
#  
BM\_d$ed1 <- ifelse(BM\_d$education == c("primary"), 1, 0)  
BM\_d$ed2 <- ifelse(BM\_d$education == c("secondary"), 1, 0)  
BM\_d$ed3 <- ifelse(BM\_d$education == c("tertiary"), 1, 0)  
BM\_d$ed4 <- ifelse(BM\_d$education == c("unknown"), 1, 0)  
  
#  
BM\_d$hous1 <- ifelse(BM\_d$housing == c("no"), 1, 0)  
BM\_d$def1 <- ifelse(BM\_d$default == c("no"), 1, 0)  
BM\_d$loan1 <- ifelse(BM\_d$loan == c("no"), 1, 0)  
#  
BM\_d$cont1 <- ifelse(BM\_d$contact == c("cellular"), 1, 0)  
BM\_d$cont2 <- ifelse(BM\_d$contact == c("telephone"), 1, 0)  
BM\_d$cont3 <- ifelse(BM\_d$contact == c("unknown"), 1, 0)  
#  
BM\_d$mon1 <- ifelse(BM\_d$month == c("jan"), 1, 0)  
BM\_d$mon2 <- ifelse(BM\_d$month == c("feb"), 1, 0)  
BM\_d$mon3 <- ifelse(BM\_d$month == c("mar"), 1, 0)  
BM\_d$mon4 <- ifelse(BM\_d$month == c("apr"), 1, 0)  
BM\_d$mon5 <- ifelse(BM\_d$month == c("may"), 1, 0)  
BM\_d$mon6 <- ifelse(BM\_d$month == c("jun"), 1, 0)  
BM\_d$mon7 <- ifelse(BM\_d$month == c("jul"), 1, 0)  
BM\_d$mon8 <- ifelse(BM\_d$month == c("aug"), 1, 0)  
BM\_d$mon9 <- ifelse(BM\_d$month == c("sep"), 1, 0)  
BM\_d$mon10 <- ifelse(BM\_d$month == c("oct"), 1, 0)  
BM\_d$mon11 <- ifelse(BM\_d$month == c("nov"), 1, 0)  
BM\_d$mon12 <- ifelse(BM\_d$month == c("dec"), 1, 0)  
#  
BM\_d$pout1 <- ifelse(BM\_d$poutcome == c("failure"), 1, 0)  
BM\_d$pout2 <- ifelse(BM\_d$poutcome == c("success"), 1, 0)  
BM\_d$pout3 <- ifelse(BM\_d$poutcome == c("other"), 1, 0)  
BM\_d$pout4 <- ifelse(BM\_d$poutcome == c("unknown"), 1, 0)  
# summary(BM\_dummy)  
# ok so we end up with a lot - then we have to delete the original Factor attributes  
## now take out the original attributes and the answer Y column and RESULT: BM\_dummy   
#  
# keep the y column in separate file for now  
BM\_y<- BM\_d[17]  
# now take out y in file - to place later at the end  
BM\_d<- BM\_d[-17]  
# now take out the remaining factors one at a time  
BM\_d<- BM\_d[-16]  
BM\_d<- BM\_d[-11]  
BM\_d<- BM\_d[-9]  
BM\_d<- BM\_d[-8]  
BM\_d<- BM\_d[-7]  
BM\_d<- BM\_d[-5]  
BM\_d<- BM\_d[-4]  
BM\_d<- BM\_d[-3]  
BM\_d<- BM\_d[-2]  
# add back y at the end   
# y is still a factor  
BM\_d<- cbind(BM\_d, BM\_y)  
# summary(BM\_dummy)  
# result: BM\_dummy with dummies for categorical = > all numeric (still not scaled)  
# convert  
# BM\_dummy<-BM\_d  
BM\_mini\_dm<-BM\_d  
rm(BM\_d)

# step 2 - run training and test datasets FOR ALL CONFIGURATIONS

# rename dataset here:  
BM<- BM\_mini\_dm  
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….chnage -17 to -49

# summary(BM\_mini\_dm)  
table(BM\_train$y)

##   
## no yes   
## 2804 360

x=BM\_train[,-49]  
trainsv=BM\_train  
train=BM\_train  
y=BM\_train$y  
test\_noy=BM\_test[,-49]  
test\_labels=BM\_test$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 1195 160  
## yes 1 1  
##   
## Accuracy : 0.8814   
## 95% CI : (0.863, 0.8981)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.521   
##   
## Kappa : 0.0094   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.999164   
## Specificity : 0.006211   
## Pos Pred Value : 0.881919   
## Neg Pred Value : 0.500000   
## Prevalence : 0.881356   
## Detection Rate : 0.880619   
## Detection Prevalence : 0.998526   
## Balanced Accuracy : 0.502688   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(nab\_mod, test[-49])  
 actual<- test$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] NaN

# 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 1143 93  
## yes 53 68  
##   
## Accuracy : 0.8924   
## 95% CI : (0.8747, 0.9084)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.110700   
##   
## Kappa : 0.4236   
##   
## Mcnemar's Test P-Value : 0.001248   
##   
## Sensitivity : 0.9557   
## Specificity : 0.4224   
## Pos Pred Value : 0.9248   
## Neg Pred Value : 0.5620   
## Prevalence : 0.8814   
## Detection Rate : 0.8423   
## Detection Prevalence : 0.9108   
## Balanced Accuracy : 0.6890   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(c5\_mod, test[-49])  
 actual<- test$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] 0.4807363

# 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] Tue Jul 28 22:45:27 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 3164 cases (49 attributes) from undefined.data  
##   
## Decision tree:  
##   
## pout2 > 0: yes (91/33)  
## pout2 <= 0:  
## :...duration <= 375:  
## :...mon3 <= 0: no (2437/103)  
## : mon3 > 0:  
## : :...hous1 <= 0: yes (6/1)  
## : hous1 > 0: no (20/6)  
## duration > 375:  
## :...duration > 645:  
## :...cont3 <= 0: yes (165/72)  
## : cont3 > 0: no (63/20)  
## duration <= 645:  
## :...cont3 > 0: no (105/10)  
## cont3 <= 0:  
## :...mon6 > 0: yes (12/3)  
## mon6 <= 0:  
## :...hous1 <= 0: no (135/20)  
## hous1 > 0:  
## :...mon4 <= 0: no (122/30)  
## mon4 > 0:  
## :...pdays <= 64: yes (6)  
## pdays > 64: no (2)  
##   
##   
## Evaluation on training data (3164 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 12 298( 9.4%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 2695 109 (a): class no  
## 189 171 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% pout2  
## 97.12% duration  
## 77.84% mon3  
## 19.28% cont3  
## 9.20% hous1  
## 8.75% mon6  
## 4.11% mon4  
## 0.25% pdays  
##   
##   
## 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 1158 106  
## yes 38 55  
##   
## Accuracy : 0.8939   
## 95% CI : (0.8763, 0.9098)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.08141   
##   
## Kappa : 0.3791   
##   
## Mcnemar's Test P-Value : 2.36e-08   
##   
## Sensitivity : 0.9682   
## Specificity : 0.3416   
## Pos Pred Value : 0.9161   
## Neg Pred Value : 0.5914   
## Prevalence : 0.8814   
## Detection Rate : 0.8534   
## Detection Prevalence : 0.9315   
## Balanced Accuracy : 0.6549   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(JR\_mod, test[-49])  
 actual<- test$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] 0.416744

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 212) and (pout2 >= 1) => .outcome=yes (65.0/17.0)  
## (duration >= 646) and (cont1 >= 1) => .outcome=yes (146.0/61.0)  
## => .outcome=no (2953.0/227.0)  
##   
## 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 1165 104  
## yes 31 57  
##   
## Accuracy : 0.9005   
## 95% CI : (0.8833, 0.9159)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.01455   
##   
## Kappa : 0.4082   
##   
## Mcnemar's Test P-Value : 5.763e-10   
##   
## Sensitivity : 0.9741   
## Specificity : 0.3540   
## Pos Pred Value : 0.9180   
## Neg Pred Value : 0.6477   
## Prevalence : 0.8814   
## Detection Rate : 0.8585   
## Detection Prevalence : 0.9352   
## Balanced Accuracy : 0.6641   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(log\_mod, test[-49])  
 actual<- test$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] 0.4424991

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

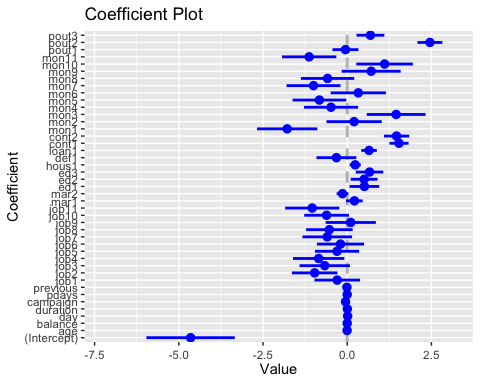
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.8382 -0.3827 -0.2576 -0.1565 3.0061   
##   
## Coefficients: (6 not defined because of singularities)  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.648e+00 1.313e+00 -3.541 0.000398 \*\*\*  
## age -3.651e-03 8.408e-03 -0.434 0.664118   
## balance 2.520e-07 1.939e-05 0.013 0.989630   
## day 1.282e-02 9.782e-03 1.311 0.189966   
## duration 4.063e-03 2.417e-04 16.808 < 2e-16 \*\*\*  
## campaign -4.975e-02 3.144e-02 -1.582 0.113564   
## pdays 3.329e-05 1.177e-03 0.028 0.977440   
## previous -8.790e-03 4.714e-02 -0.186 0.852069   
## job1 -2.975e-01 6.761e-01 -0.440 0.659917   
## job2 -9.673e-01 6.765e-01 -1.430 0.152746   
## job3 -6.672e-01 7.491e-01 -0.891 0.373074   
## job4 -8.473e-01 7.628e-01 -1.111 0.266699   
## job5 -3.023e-01 6.574e-01 -0.460 0.645647   
## job6 -1.987e-01 7.010e-01 -0.284 0.776788   
## job7 -5.923e-01 7.371e-01 -0.803 0.421713   
## job8 -5.296e-01 6.912e-01 -0.766 0.443565   
## job9 1.047e-01 7.452e-01 0.141 0.888246   
## job10 -6.126e-01 6.640e-01 -0.923 0.356187   
## job11 -1.039e+00 8.047e-01 -1.291 0.196790   
## job12 NA NA NA NA   
## mar1 2.129e-01 2.475e-01 0.860 0.389696   
## mar2 -1.400e-01 1.743e-01 -0.803 0.421755   
## mar3 NA NA NA NA   
## ed1 5.106e-01 4.399e-01 1.161 0.245798   
## ed2 5.026e-01 3.983e-01 1.262 0.207031   
## ed3 6.634e-01 4.082e-01 1.625 0.104142   
## ed4 NA NA NA NA   
## hous1 2.380e-01 1.653e-01 1.439 0.150025   
## def1 -3.202e-01 5.918e-01 -0.541 0.588501   
## loan1 6.485e-01 2.310e-01 2.807 0.005001 \*\*   
## cont1 1.538e+00 2.839e-01 5.416 6.10e-08 \*\*\*  
## cont2 1.465e+00 3.761e-01 3.896 9.80e-05 \*\*\*  
## cont3 NA NA NA NA   
## mon1 -1.782e+00 8.955e-01 -1.990 0.046547 \*   
## mon2 2.055e-01 8.198e-01 0.251 0.802080   
## mon3 1.453e+00 8.743e-01 1.662 0.096492 .   
## mon4 -4.807e-01 8.044e-01 -0.598 0.550122   
## mon5 -8.252e-01 7.978e-01 -1.034 0.300943   
## mon6 3.290e-01 8.200e-01 0.401 0.688286   
## mon7 -1.000e+00 8.014e-01 -1.248 0.212030   
## mon8 -5.831e-01 7.925e-01 -0.736 0.461859   
## mon9 7.120e-01 8.748e-01 0.814 0.415684   
## mon10 1.109e+00 8.397e-01 1.321 0.186576   
## mon11 -1.130e+00 8.075e-01 -1.399 0.161708   
## mon12 NA NA NA NA   
## pout1 -5.191e-02 3.859e-01 -0.135 0.892994   
## pout2 2.457e+00 3.703e-01 6.635 3.24e-11 \*\*\*  
## pout3 6.893e-01 4.136e-01 1.667 0.095601 .   
## pout4 NA NA NA NA   
## ---  
## 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: 1521.2 on 3121 degrees of freedom  
## AIC: 1607.2  
##   
## 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 1157 135  
## yes 39 26  
##   
## Accuracy : 0.8718   
## 95% CI : (0.8528, 0.8891)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.8709   
##   
## Kappa : 0.1737   
##   
## Mcnemar's Test P-Value : 5.937e-13   
##   
## Sensitivity : 0.9674   
## Specificity : 0.1615   
## Pos Pred Value : 0.8955   
## Neg Pred Value : 0.4000   
## Prevalence : 0.8814   
## Detection Rate : 0.8526   
## Detection Prevalence : 0.9521   
## Balanced Accuracy : 0.5644   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(knn\_mod, test[-49])  
 actual<- test$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] 0.2217622

# 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 1194 157  
## yes 2 4  
##   
## Accuracy : 0.8828   
## 95% CI : (0.8645, 0.8995)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.4541   
##   
## Kappa : 0.0397   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.99833   
## Specificity : 0.02484   
## Pos Pred Value : 0.88379   
## Neg Pred Value : 0.66667   
## Prevalence : 0.88136   
## Detection Rate : 0.87988   
## Detection Prevalence : 0.99558   
## Balanced Accuracy : 0.51159   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(nn\_mod, test[-49])  
 actual<- test$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] NaN

# 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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(sv\_m, test[-49])  
 actual<- test$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] NaN

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

### END