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

jean wills

13/07/2020

# read the file and set up train and test

library(plyr)  
library(dplyr)  
BM\_mini <- read.csv("/Users/jeanwills/Desktop/CKME136/1\_data/bank.csv", header=T, sep = ";", stringsAsFactors = T, na.strings = "NA")  
BM<-BM\_mini  
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 1140 111  
## yes 56 50  
##   
## Accuracy : 0.8769   
## 95% CI : (0.8583, 0.894)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.7099   
##   
## Kappa : 0.3095   
##   
## Mcnemar's Test P-Value : 2.933e-05   
##   
## Sensitivity : 0.9532   
## Specificity : 0.3106   
## Pos Pred Value : 0.9113   
## Neg Pred Value : 0.4717   
## Prevalence : 0.8814   
## Detection Rate : 0.8401   
## Detection Prevalence : 0.9219   
## Balanced Accuracy : 0.6319   
##   
## '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] 0.3630216

# 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 1140 95  
## yes 56 66  
##   
## Accuracy : 0.8887   
## 95% CI : (0.8708, 0.905)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.213699   
##   
## Kappa : 0.4056   
##   
## Mcnemar's Test P-Value : 0.001985   
##   
## Sensitivity : 0.9532   
## Specificity : 0.4099   
## Pos Pred Value : 0.9231   
## Neg Pred Value : 0.5410   
## Prevalence : 0.8814   
## Detection Rate : 0.8401   
## Detection Prevalence : 0.9101   
## Balanced Accuracy : 0.6816   
##   
## '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] 0.4674824

# 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 23:35:18 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 3164 cases (17 attributes) from undefined.data  
##   
## Decision tree:  
##   
## poutcome = success:  
## :...balance > 7635: no (5)  
## : balance <= 7635:  
## : :...duration > 174: yes (68/15)  
## : duration <= 174:  
## : :...housing = yes: no (7)  
## : housing = no:  
## : :...age <= 44: yes (6/1)  
## : age > 44: no (5)  
## poutcome in {failure,other,unknown}:  
## :...duration > 375:  
## :...duration <= 645: no (382/75)  
## : duration > 645:  
## : :...contact in {telephone,unknown}:  
## : :...age > 59: yes (5)  
## : : age <= 59:  
## : : :...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 <= 15: no (3)  
## : : day > 15: yes (2)  
## : job = retired:  
## : :...poutcome = failure: yes (1)  
## : : poutcome in {other,unknown}: no (3)  
## : job = student:  
## : :...age <= 29: yes (2)  
## : : age > 29: no (2)  
## : job = unemployed:  
## : :...balance <= 462: no (3)  
## : : balance > 462: 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 <= 34: no (3)  
## : : : age > 34: yes (3)  
## : : month = jul:  
## : : :...age <= 42: yes (2)  
## : : : age > 42: no (4)  
## : : month = nov:  
## : : :...age <= 36: no (3)  
## : : age > 36: yes (2)  
## : job = management:  
## : :...month in {apr,jun,nov}: no (9/3)  
## : month in {dec,feb,jan,mar,oct,sep}: yes (2)  
## : month = may:  
## : :...balance <= 322: no (3)  
## : : balance > 322: yes (3)  
## : month = aug:  
## : :...duration > 800: yes (7)  
## : : duration <= 800:  
## : : :...day <= 17: yes (2)  
## : : day > 17: no (3)  
## : month = jul:  
## : :...campaign > 3: yes (4)  
## : campaign <= 3:  
## : :...balance <= 343: yes (3/1)  
## : balance > 343: no (3)  
## duration <= 375:  
## :...month in {aug,jan,jul,jun,may,nov}: no (2121/54)  
## month in {mar,oct}:  
## :...duration <= 88: no (11)  
## : duration > 88:  
## : :...job in {blue-collar,entrepreneur,retired,self-employed,  
## : : unknown}: no (14/3)  
## : job in {housemaid,services,student,unemployed}: yes (3)  
## : job = admin.:  
## : :...duration <= 143: no (4)  
## : : duration > 143: 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 > 20:  
## :...duration > 239: yes (12/1)  
## : duration <= 239:  
## : :...balance <= 11: yes (3)  
## : balance > 11: no (28/1)  
## day <= 20:  
## :...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 > 9: yes (7)  
## day <= 9:  
## :...balance <= 2139: no (45)  
## balance > 2139:  
## :...pdays <= 28: no (7/1)  
## pdays > 28: yes (2)  
##   
##   
## Evaluation on training data (3164 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 58 207( 6.5%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 2760 44 (a): class no  
## 163 197 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% poutcome  
## 99.84% duration  
## 80.03% month  
## 13.72% job  
## 9.20% day  
## 7.21% contact  
## 6.10% balance  
## 3.60% age  
## 3.00% marital  
## 0.57% housing  
## 0.32% campaign  
## 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 1159 106  
## yes 37 55  
##   
## Accuracy : 0.8946   
## 95% CI : (0.8771, 0.9105)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.06912   
##   
## Kappa : 0.3814   
##   
## Mcnemar's Test P-Value : 1.297e-08   
##   
## Sensitivity : 0.9691   
## Specificity : 0.3416   
## Pos Pred Value : 0.9162   
## Neg Pred Value : 0.5978   
## Prevalence : 0.8814   
## Detection Rate : 0.8541   
## Detection Prevalence : 0.9322   
## Balanced Accuracy : 0.6553   
##   
## '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] 0.4176098

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 646) and (contact = cellular) => .outcome=yes (155.0/66.0)  
## (duration >= 212) and (pdays >= 38) and (poutcome = success) => .outcome=yes (56.0/12.0)  
## => .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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(log\_mod, test[-17])  
 actual<- test$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] 0.4424991

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

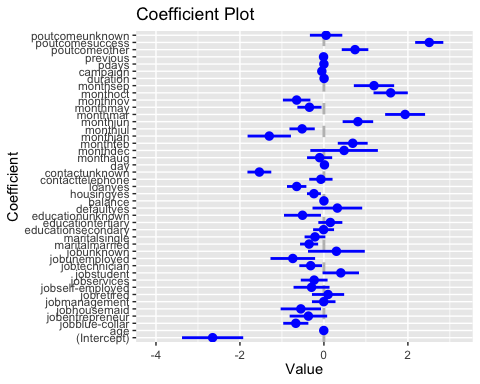
### model results

# 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:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.651e+00 7.286e-01 -3.638 0.000274 \*\*\*  
## age -3.651e-03 8.408e-03 -0.434 0.664118   
## jobblue-collar -6.698e-01 2.994e-01 -2.237 0.025296 \*   
## jobentrepreneur -3.697e-01 4.468e-01 -0.828 0.407949   
## jobhousemaid -5.497e-01 4.810e-01 -1.143 0.253084   
## jobmanagement -4.758e-03 2.805e-01 -0.017 0.986464   
## jobretired 9.876e-02 3.835e-01 0.258 0.796769   
## jobself-employed -2.948e-01 4.288e-01 -0.687 0.491853   
## jobservices -2.321e-01 3.192e-01 -0.727 0.467156   
## jobstudent 4.022e-01 4.330e-01 0.929 0.352984   
## jobtechnician -3.151e-01 2.706e-01 -1.165 0.244182   
## jobunemployed -7.412e-01 5.314e-01 -1.395 0.163112   
## jobunknown 2.975e-01 6.761e-01 0.440 0.659917   
## maritalmarried -3.529e-01 2.154e-01 -1.639 0.101270   
## maritalsingle -2.129e-01 2.475e-01 -0.860 0.389696   
## educationsecondary -7.986e-03 2.495e-01 -0.032 0.974465   
## educationtertiary 1.528e-01 2.860e-01 0.534 0.593126   
## educationunknown -5.106e-01 4.399e-01 -1.161 0.245798   
## defaultyes 3.202e-01 5.918e-01 0.541 0.588501   
## balance 2.520e-07 1.939e-05 0.013 0.989630   
## housingyes -2.380e-01 1.653e-01 -1.439 0.150025   
## loanyes -6.485e-01 2.310e-01 -2.807 0.005001 \*\*   
## contacttelephone -7.227e-02 2.777e-01 -0.260 0.794684   
## contactunknown -1.538e+00 2.839e-01 -5.416 6.10e-08 \*\*\*  
## day 1.282e-02 9.782e-03 1.311 0.189966   
## monthaug -1.024e-01 2.992e-01 -0.342 0.732031   
## monthdec 4.807e-01 8.044e-01 0.598 0.550122   
## monthfeb 6.862e-01 3.559e-01 1.928 0.053877 .   
## monthjan -1.302e+00 5.163e-01 -2.521 0.011701 \*   
## monthjul -5.195e-01 3.014e-01 -1.723 0.084800 .   
## monthjun 8.096e-01 3.636e-01 2.226 0.025981 \*   
## monthmar 1.934e+00 4.756e-01 4.066 4.78e-05 \*\*\*  
## monthmay -3.446e-01 2.877e-01 -1.197 0.231137   
## monthnov -6.493e-01 3.276e-01 -1.982 0.047491 \*   
## monthoct 1.590e+00 4.076e-01 3.900 9.63e-05 \*\*\*  
## monthsep 1.193e+00 4.802e-01 2.484 0.012993 \*   
## 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   
## poutcomeother 7.412e-01 3.166e-01 2.341 0.019221 \*   
## poutcomesuccess 2.509e+00 3.353e-01 7.482 7.34e-14 \*\*\*  
## poutcomeunknown 5.191e-02 3.859e-01 0.135 0.892994   
## ---  
## 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 1171 113  
## yes 25 48  
##   
## Accuracy : 0.8983   
## 95% CI : (0.881, 0.9139)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.02753   
##   
## Kappa : 0.3631   
##   
## Mcnemar's Test P-Value : 1.302e-13   
##   
## Sensitivity : 0.9791   
## Specificity : 0.2981   
## Pos Pred Value : 0.9120   
## Neg Pred Value : 0.6575   
## Prevalence : 0.8814   
## Detection Rate : 0.8629   
## Detection Prevalence : 0.9462   
## Balanced Accuracy : 0.6386   
##   
## '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] NaN

# 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 1196 161  
## yes 0 0  
##   
## Accuracy : 0.8814   
## 95% CI : (0.863, 0.8981)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.521   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8814   
## Neg Pred Value : NaN   
## Prevalence : 0.8814   
## Detection Rate : 0.8814   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## '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] 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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 pred<- predict(sv\_m, test[-17])  
 actual<- test$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

### END