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

jean wills

13/07/2020

THIS FILE IS TO SEE EFFECTS OF FEATURE SELECTION METHOD x5

## 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\_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  
rm(BMS)  
rm(BM\_s)  
rm(BM)

## Step 1c - NOW DELETE THE ATTRIBUTES HERE AND CHANGE ANY data where there is [-17]

temp<-BM\_mini\_sc  
  
# FS5: AIC step keep 10: # 1, 4, 6, 7, 10, 11, 12, 13, 14, 16 +> delete 15,9,8,5,3,2  
temp<-BM\_mini\_sc  
temp<- temp[-15]  
temp<- temp[-9]  
temp<- temp[-8]  
temp<- temp[-5]  
temp<- temp[-3]  
temp<- temp[-2]  
  
BM<- temp  
summary(BM)

## age duration pdays previous   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.2059 1st Qu.:0.0879 1st Qu.:0.0000 1st Qu.:0.00000   
## Median :0.2941 Median :0.1717 Median :0.0000 Median :0.00000   
## Mean :0.3260 Mean :0.2450 Mean :0.1105 Mean :0.09082   
## 3rd Qu.:0.4412 3rd Qu.:0.3206 3rd Qu.:0.0000 3rd Qu.:0.00000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.00000   
## education default housing loan contact   
## primary : 678 no :4445 no :1962 no :3830 cellular :2896   
## secondary:2306 yes: 76 yes:2559 yes: 691 telephone: 301   
## tertiary :1350 unknown :1324   
## unknown : 187   
##   
##   
## poutcome y   
## failure: 490 no :4000   
## other : 197 yes: 521   
## success: 129   
## unknown:3705   
##   
##

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

# str(BM\_mini\_sc)  
# rename dataset here:  
# BM<- BM\_mini\_sc  
set.seed(30)  
# get train and test datasets  
BM\_train\_index <- sample(nrow(BM), 0.7 \* nrow(BM))  
BM\_train<- BM[BM\_train\_index, ]  
BM\_test <- BM[-BM\_train\_index, ]  
# BM\_train\_labels <- BM[BM\_train\_index, 17]  
# BM\_test\_labels <- BM[-BM\_train\_index, 17]  
# note that we only balance the training sets  
# and leave test set as is.

## step 3 RUN SMOTE 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   
# \*\*\*\*\*\*\*\*\*- change y from 17 to 11....  
x=smote\_train[,-11]  
trainsv=smote\_train  
train=smote\_train  
y=smote\_train$y  
test\_noy=BM\_test[,-11]  
test\_labels=BM\_test$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 1075 84  
## yes 121 77  
##   
## Accuracy : 0.8489   
## 95% CI : (0.8288, 0.8676)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.99985   
##   
## Kappa : 0.343   
##   
## Mcnemar's Test P-Value : 0.01193   
##   
## Sensitivity : 0.8988   
## Specificity : 0.4783   
## Pos Pred Value : 0.9275   
## Neg Pred Value : 0.3889   
## Prevalence : 0.8814   
## Detection Rate : 0.7922   
## Detection Prevalence : 0.8541   
## Balanced Accuracy : 0.6885   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nab\_mod, test[-11])  
 actual<- test$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] 0.4244273

# 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 1006 49  
## yes 190 112  
##   
## Accuracy : 0.8239   
## 95% CI : (0.8025, 0.8438)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3893   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8411   
## Specificity : 0.6957   
## Pos Pred Value : 0.9536   
## Neg Pred Value : 0.3709   
## Prevalence : 0.8814   
## Detection Rate : 0.7413   
## Detection Prevalence : 0.7775   
## Balanced Accuracy : 0.7684   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(c5\_mod, test[,-11])  
 actual<- test$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] 0.4841707

# 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 00:18:14 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 2520 cases (11 attributes) from undefined.data  
##   
## Decision tree:  
##   
## duration <= 0.2109617:  
## :...poutcome = success:  
## : :...education in {primary,unknown}: no (3)  
## : : education in {secondary,tertiary}:  
## : : :...duration > 0.1305113: yes (25)  
## : : duration <= 0.1305113:  
## : : :...pdays <= 0.2592432: yes (6)  
## : : pdays > 0.2592432: no (6/1)  
## : poutcome in {failure,other,unknown}:  
## : :...duration <= 0.07342296: no (340/4)  
## : duration > 0.07342296:  
## : :...previous <= 0.003241458: no (539/61)  
## : previous > 0.003241458:  
## : :...poutcome = unknown: yes (20)  
## : poutcome in {failure,other}:  
## : :...contact = unknown: yes (7)  
## : contact in {cellular,telephone}:  
## : :...pdays > 0.3704492: no (94/8)  
## : pdays <= 0.3704492:  
## : :...age <= 0.2647059: no (16/2)  
## : age > 0.2647059: yes (17/5)  
## duration > 0.2109617:  
## :...previous > 0:  
## :...poutcome in {other,success,unknown}: yes (443/21)  
## : poutcome = failure:  
## : :...previous <= 0.1982166: yes (16)  
## : previous > 0.1982166:  
## : :...loan = yes: yes (17/3)  
## : loan = no:  
## : :...housing = no:  
## : :...age <= 0.5181711: yes (29/6)  
## : : age > 0.5181711:  
## : : :...age <= 0.7903481: no (12/3)  
## : : age > 0.7903481: yes (4)  
## : housing = yes:  
## : :...previous > 0.2405902: yes (46/21)  
## : previous <= 0.2405902:  
## : :...pdays <= 0.991573: no (13)  
## : pdays > 0.991573: yes (4/1)  
## previous <= 0:  
## :...duration <= 0.450879:  
## :...contact = telephone: yes (40/14)  
## : contact = unknown:  
## : :...age <= 0.370143: no (95/4)  
## : : age > 0.370143:  
## : : :...age <= 0.418384: yes (8/2)  
## : : age > 0.418384: no (31/4)  
## : contact = cellular:  
## : :...education = primary:  
## : :...age <= 0.4920191: no (25/1)  
## : : age > 0.4920191: yes (5/1)  
## : education = tertiary:  
## : :...loan = no: no (80/30)  
## : : loan = yes: yes (11/3)  
## : education = unknown:  
## : :...housing = no: yes (4/1)  
## : : housing = yes: no (5)  
## : education = secondary:  
## : :...default = yes:  
## : :...housing = no: yes (3)  
## : : housing = yes: no (2)  
## : default = no:  
## : :...loan = no:  
## : :...age <= 0.1239145: yes (7/1)  
## : : age > 0.1239145: no (102/24)  
## : loan = yes:  
## : :...duration <= 0.4157187: no (18/4)  
## : duration > 0.4157187: yes (4/1)  
## duration > 0.450879:  
## :...duration > 0.8082834: yes (156/21)  
## duration <= 0.8082834:  
## :...education in {tertiary,unknown}: yes (87/24)  
## education = secondary:  
## :...age <= 0.181017: no (31/8)  
## : age > 0.181017: yes (102/33)  
## education = primary:  
## :...loan = yes:  
## :...age <= 0.2352941: no (2)  
## : age > 0.2352941: yes (6)  
## loan = no:  
## :...duration > 0.6779224: no (16/2)  
## duration <= 0.6779224:  
## :...contact in {cellular,telephone}: yes (13/2)  
## contact = unknown:  
## :...duration <= 0.5066705: yes (3)  
## duration > 0.5066705: no (7/1)  
##   
##   
## Evaluation on training data (2520 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 46 317(12.6%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 1280 160 (a): class no  
## 157 923 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% duration  
## 84.92% previous  
## 65.75% poutcome  
## 23.69% contact  
## 22.74% education  
## 19.52% age  
## 15.63% loan  
## 6.19% pdays  
## 5.40% default  
## 4.84% 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 1025 42  
## yes 171 119  
##   
## Accuracy : 0.843   
## 95% CI : (0.8226, 0.862)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4427   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8570   
## Specificity : 0.7391   
## Pos Pred Value : 0.9606   
## Neg Pred Value : 0.4103   
## Prevalence : 0.8814   
## Detection Rate : 0.7553   
## Detection Prevalence : 0.7863   
## Balanced Accuracy : 0.7981   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(JR\_mod, test[-11])  
 actual<- test$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] 0.5275397

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.256463) and (pdays >= 0.000614) => .outcome=yes (503.0/54.0)  
## (duration >= 0.453507) => .outcome=yes (423.0/125.0)  
## (duration >= 0.156153) and (pdays >= 0.001386) and (pdays <= 0.383167) and (previous <= 0.599395) => .outcome=yes (65.0/2.0)  
## (duration >= 0.198675) and (contact = telephone) and (age <= 0.459495) => .outcome=yes (35.0/7.0)  
## (duration >= 0.130511) and (poutcome = success) => .outcome=yes (32.0/3.0)  
## (duration >= 0.155119) and (age >= 0.617647) => .outcome=yes (21.0/5.0)  
## => .outcome=no (1441.0/197.0)  
##   
## Number of Rules : 7

# 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 1061 54  
## yes 135 107  
##   
## Accuracy : 0.8607   
## 95% CI : (0.8412, 0.8787)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9905   
##   
## Kappa : 0.4531   
##   
## Mcnemar's Test P-Value : 5.915e-09   
##   
## Sensitivity : 0.8871   
## Specificity : 0.6646   
## Pos Pred Value : 0.9516   
## Neg Pred Value : 0.4421   
## Prevalence : 0.8814   
## Detection Rate : 0.7819   
## Detection Prevalence : 0.8217   
## Balanced Accuracy : 0.7759   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(log\_mod, test[-11])  
 actual<- test$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] 0.5242524

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

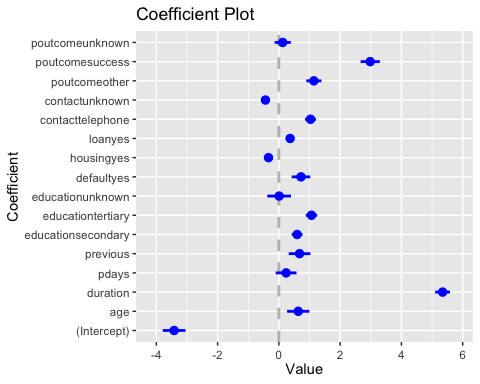
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.4864 -0.6436 -0.3764 0.5858 2.4038   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.419255 0.370400 -9.231 < 2e-16 \*\*\*  
## age 0.632348 0.364886 1.733 0.083095 .   
## duration 5.344458 0.241406 22.139 < 2e-16 \*\*\*  
## pdays 0.235585 0.340607 0.692 0.489149   
## previous 0.675134 0.353971 1.907 0.056480 .   
## educationsecondary 0.595934 0.175344 3.399 0.000677 \*\*\*  
## educationtertiary 1.064271 0.186236 5.715 1.10e-08 \*\*\*  
## educationunknown 0.007916 0.387483 0.020 0.983701   
## defaultyes 0.721567 0.304226 2.372 0.017701 \*   
## housingyes -0.338463 0.111817 -3.027 0.002470 \*\*   
## loanyes 0.367285 0.130814 2.808 0.004990 \*\*   
## contacttelephone 1.032481 0.179751 5.744 9.25e-09 \*\*\*  
## contactunknown -0.438198 0.140716 -3.114 0.001845 \*\*   
## poutcomeother 1.141999 0.246162 4.639 3.50e-06 \*\*\*  
## poutcomesuccess 2.980974 0.314441 9.480 < 2e-16 \*\*\*  
## poutcomeunknown 0.125012 0.265170 0.471 0.637326   
## ---  
## 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: 2201.2 on 2504 degrees of freedom  
## AIC: 2233.2  
##   
## Number of Fisher Scoring iterations: 5

### plot to see the most important attributes (those that “stand out” at far left or right)

require(coefplot)  
coefplot(log\_mod)



# to reinterpret data properly  
# invlogit<- function (x) {1/(1+exp(-x))}  
# invlogit(log\_mod$coefficients)  
# results of plot: duration, campaign, balance, poutcomesuccess, some months, , some jobs

## Model #5 - K-Nearest Neighbours

# The final values used for the model were kmax = 9, distance = 2 and kernel = optimal  
library(caret)  
library(lattice)  
library(ggplot2)  
knn\_ctrl<- trainControl(method="repeatedcv", number = 10, repeats = 10, classProbs=TRUE, summaryFunction = twoClassSummary)  
# knn\_grid<-expand.grid(kmax=c(5,7,9,13), distance=c(1,2,4,6), kernel=c("rectangular","rank","optimal"))  
knn\_grid<-expand.grid(kmax=9, distance=2, kernel="optimal")  
knn\_mod<- train(x=x,y=y, method="kknn", metric="ROC", tuneGrid=knn\_grid, trControl=knn\_ctrl, verbose=FALSE)  
knn\_pred<- predict(knn\_mod, test\_noy)  
# Testing the result output  
s<-table( knn\_pred, test\_labels)  
# Confusion matrix  
print(confusionMatrix(s))

## Confusion Matrix and Statistics  
##   
## test\_labels  
## knn\_pred no yes  
## no 1001 53  
## yes 195 108  
##   
## Accuracy : 0.8172   
## 95% CI : (0.7956, 0.8375)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3675   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8370   
## Specificity : 0.6708   
## Pos Pred Value : 0.9497   
## Neg Pred Value : 0.3564   
## Prevalence : 0.8814   
## Detection Rate : 0.7377   
## Detection Prevalence : 0.7767   
## Balanced Accuracy : 0.7539   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(knn\_mod, test[-11])  
 actual<- test$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] 0.4621066

# 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 1035 54  
## yes 161 107  
##   
## Accuracy : 0.8416   
## 95% CI : (0.821, 0.8606)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4116   
##   
## Mcnemar's Test P-Value : 4.861e-13   
##   
## Sensitivity : 0.8654   
## Specificity : 0.6646   
## Pos Pred Value : 0.9504   
## Neg Pred Value : 0.3993   
## Prevalence : 0.8814   
## Detection Rate : 0.7627   
## Detection Prevalence : 0.8025   
## Balanced Accuracy : 0.7650   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nn\_mod, test[-11])  
 actual<- test$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] 0.4974057

# 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 1060 56  
## yes 136 105  
##   
## Accuracy : 0.8585   
## 95% CI : (0.8388, 0.8766)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9951   
##   
## Kappa : 0.4432   
##   
## Mcnemar's Test P-Value : 1.189e-08   
##   
## Sensitivity : 0.8863   
## Specificity : 0.6522   
## Pos Pred Value : 0.9498   
## Neg Pred Value : 0.4357   
## Prevalence : 0.8814   
## Detection Rate : 0.7811   
## Detection Prevalence : 0.8224   
## Balanced Accuracy : 0.7692   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(sv\_m, test[-11])  
 actual<- test$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] 0.5208462

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