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]

# FS2 RFE+RF - keep only 5: # 3, 4, 14, 15, 16 => delete 5 to 13, 2, 1  
temp<-BM\_mini\_sc  
rm(BM\_mini\_sc)  
temp<- temp[-13]  
temp<- temp[-12]  
temp<- temp[-11]  
temp<- temp[-10]  
temp<- temp[-9]  
temp<- temp[-8]  
temp<- temp[-7]  
temp<- temp[-6]  
temp<- temp[-5]  
temp<- temp[-2]  
temp<- temp[-1]  
BM<- temp  
summary(BM)

## day duration contact month   
## Min. :0.0000 Min. :0.0000 cellular :2896 may :1398   
## 1st Qu.:0.2667 1st Qu.:0.0879 telephone: 301 jul : 706   
## Median :0.5000 Median :0.1717 unknown :1324 aug : 633   
## Mean :0.4972 Mean :0.2450 jun : 531   
## 3rd Qu.:0.6667 3rd Qu.:0.3206 nov : 389   
## Max. :1.0000 Max. :1.0000 apr : 293   
## (Other): 571   
## 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 6....  
x=smote\_train[,-6]  
trainsv=smote\_train  
train=smote\_train  
y=smote\_train$y  
test\_noy=BM\_test[,-6]  
test\_labels=BM\_test$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 1037 54  
## yes 159 107  
##   
## Accuracy : 0.843   
## 95% CI : (0.8226, 0.862)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4146   
##   
## Mcnemar's Test P-Value : 1.034e-12   
##   
## Sensitivity : 0.8671   
## Specificity : 0.6646   
## Pos Pred Value : 0.9505   
## Neg Pred Value : 0.4023   
## Prevalence : 0.8814   
## Detection Rate : 0.7642   
## Detection Prevalence : 0.8040   
## Balanced Accuracy : 0.7658   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nab\_mod, test[-6])  
 actual<- test$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] 0.4971124

# 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 1044 40  
## yes 152 121  
##   
## Accuracy : 0.8585   
## 95% CI : (0.8388, 0.8766)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9951   
##   
## Kappa : 0.48   
##   
## Mcnemar's Test P-Value : 1.14e-15   
##   
## Sensitivity : 0.8729   
## Specificity : 0.7516   
## Pos Pred Value : 0.9631   
## Neg Pred Value : 0.4432   
## Prevalence : 0.8814   
## Detection Rate : 0.7693   
## Detection Prevalence : 0.7988   
## Balanced Accuracy : 0.8122   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(c5\_mod, test[,-6])  
 actual<- test$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] 0.5614554

# 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] Tue Jul 28 23:46:54 2020  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 2520 cases (6 attributes) from undefined.data  
##   
## Decision tree:  
##   
## poutcome = success: yes (233/15)  
## poutcome in {failure,other,unknown}:  
## :...duration <= 0.2099276:  
## :...month in {aug,dec,jan,jul,jun,may,nov}: no (869/50)  
## : month in {mar,oct,sep}:  
## : :...duration <= 0.0713547:  
## : : :...poutcome in {failure,unknown}: no (12/1)  
## : : : poutcome = other: yes (2)  
## : : duration > 0.0713547:  
## : : :...day <= 0.1164786: no (6/1)  
## : : day > 0.1164786: yes (73/13)  
## : month in {apr,feb}:  
## : :...contact = unknown: yes (5)  
## : contact = telephone:  
## : :...day <= 0.2666667: no (4)  
## : : day > 0.2666667: yes (5)  
## : contact = cellular:  
## : :...day <= 0.2773848: no (37/2)  
## : day > 0.2773848:  
## : :...month = apr: no (43/7)  
## : month = feb:  
## : :...day <= 0.649235: yes (11/1)  
## : day > 0.649235: no (4)  
## duration > 0.2099276:  
## :...contact = telephone: yes (229/22)  
## contact in {cellular,unknown}:  
## :...month in {dec,mar,oct}: yes (75/5)  
## month in {apr,aug,feb,jan,jul,jun,may,nov,sep}:  
## :...duration > 0.4304888:  
## :...duration > 0.7766287: yes (148/18)  
## : duration <= 0.7766287:  
## : :...poutcome in {failure,other}: yes (80/19)  
## : poutcome = unknown:  
## : :...contact = unknown:  
## : :...month in {apr,aug,feb,jan,jul,nov,  
## : : : sep}: yes (5)  
## : : month = jun:  
## : : :...day <= 0.048866: yes (2)  
## : : : day > 0.048866: no (16/3)  
## : : month = may:  
## : : :...duration <= 0.4715615: yes (3)  
## : : duration > 0.4715615: no (35/10)  
## : contact = cellular:  
## : :...month in {aug,jan,jun,may}: yes (81/20)  
## : month in {jul,sep}: no (38/17)  
## : month = feb:  
## : :...duration <= 0.4680654: no (3)  
## : : duration > 0.4680654: yes (10/2)  
## : month = nov:  
## : :...day <= 0.6153817: no (11/1)  
## : : day > 0.6153817: yes (3)  
## : month = apr:  
## : :...day > 0.5832622: yes (4)  
## : day <= 0.5832622:  
## : :...day > 0.4640105: no (5)  
## : day <= 0.4640105:  
## : :...duration <= 0.5418821: no (5/1)  
## : duration > 0.5418821: yes (3)  
## duration <= 0.4304888:  
## :...contact = unknown:  
## :...month in {aug,nov}: yes (4)  
## : month in {apr,feb,jan,jul,jun,may,sep}: no (118/5)  
## contact = cellular:  
## :...month in {feb,jun,sep}:  
## :...day > 0.4473079: yes (16)  
## : day <= 0.4473079:  
## : :...month = feb: no (15/5)  
## : month = jun: yes (17/6)  
## : month = sep:  
## : :...day <= 0.3069219: yes (6/2)  
## : day > 0.3069219: no (3)  
## month in {apr,aug,jan,jul,may,nov}:  
## :...day > 0.7082826:  
## :...month = jul: no (19/4)  
## : month in {apr,nov}: yes (10)  
## : month = aug:  
## : :...day <= 0.8165982: no (4/1)  
## : : day > 0.8165982: yes (7/1)  
## : month = jan:  
## : :...day <= 0.8470786: yes (2)  
## : : day > 0.8470786: no (6)  
## : month = may:  
## : :...poutcome in {failure,other}: yes (4)  
## : poutcome = unknown: no (5/1)  
## day <= 0.7082826:  
## :...day <= 0.1287784:  
## :...month = apr: no (3)  
## : month in {aug,jan,jul,may,  
## : nov}: yes (17/5)  
## day > 0.1287784:  
## :...month = jan: yes (1)  
## month = may: no (50/5)  
## month = apr:  
## :...day <= 0.649235: no (28/5)  
## : day > 0.649235: yes (3)  
## month = jul:  
## :...poutcome = failure: yes (3)  
## : poutcome in {other,  
## : unknown}: no (37/4)  
## month = nov:  
## :...day <= 0.515869: yes (7)  
## : day > 0.515869: no (27/4)  
## month = aug:  
## :...poutcome in {failure,  
## : other}: yes (3)  
## poutcome = unknown:  
## :...day > 0.2159222: no (36/5)  
## day <= 0.2159222:  
## :...day <= 0.1365742: no (2)  
## day > 0.1365742: yes (7/2)  
##   
##   
## Evaluation on training data (2520 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 61 263(10.4%) <<  
##   
##   
## (a) (b) <-classified as  
## ---- ----  
## 1309 131 (a): class no  
## 132 948 (b): class yes  
##   
##   
## Attribute usage:  
##   
## 100.00% poutcome  
## 90.75% duration  
## 81.67% month  
## 52.58% contact  
## 22.62% day  
##   
##   
## 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 1032 35  
## yes 164 126  
##   
## Accuracy : 0.8534   
## 95% CI : (0.8334, 0.8718)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9992   
##   
## Kappa : 0.4793   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8629   
## Specificity : 0.7826   
## Pos Pred Value : 0.9672   
## Neg Pred Value : 0.4345   
## Prevalence : 0.8814   
## Detection Rate : 0.7605   
## Detection Prevalence : 0.7863   
## Balanced Accuracy : 0.8227   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(JR\_mod, test[-6])  
 actual<- test$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] 0.5604332

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

### JRip rules compiled…

JR\_mod$finalModel

## JRIP rules:  
## ===========  
##   
## (duration >= 0.210274) and (contact = telephone) => .outcome=yes (290.0/22.0)  
## (duration >= 0.431231) => .outcome=yes (525.0/142.0)  
## (poutcome = success) => .outcome=yes (129.0/15.0)  
## (duration >= 0.160927) and (month = oct) => .outcome=yes (40.0/7.0)  
## (duration >= 0.137539) and (month = mar) => .outcome=yes (31.0/1.0)  
## (duration >= 0.171665) and (contact = cellular) and (month = jun) => .outcome=yes (28.0/8.0)  
## (duration >= 0.210274) and (contact = cellular) and (day >= 0.646957) and (month = apr) => .outcome=yes (13.0/0.0)  
## (duration >= 0.136307) and (month = sep) and (day >= 0.24198) => .outcome=yes (19.0/4.0)  
## => .outcome=no (1445.0/204.0)  
##   
## Number of Rules : 9

# 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 1046 48  
## yes 150 113  
##   
## Accuracy : 0.8541   
## 95% CI : (0.8342, 0.8725)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9989   
##   
## Kappa : 0.4524   
##   
## Mcnemar's Test P-Value : 7.086e-13   
##   
## Sensitivity : 0.8746   
## Specificity : 0.7019   
## Pos Pred Value : 0.9561   
## Neg Pred Value : 0.4297   
## Prevalence : 0.8814   
## Detection Rate : 0.7708   
## Detection Prevalence : 0.8062   
## Balanced Accuracy : 0.7882   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(log\_mod, test[-6])  
 actual<- test$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] 0.5282255

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

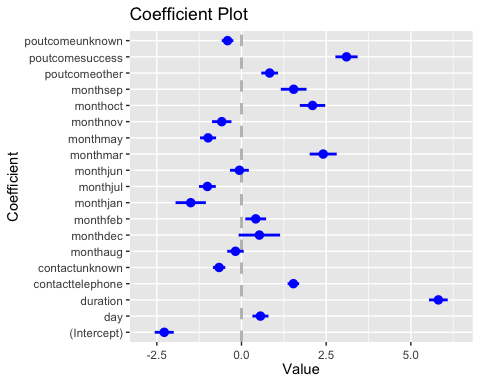
### model results

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

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2706 -0.5351 -0.2988 0.4910 2.7678   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.28120 0.28096 -8.119 4.69e-16 \*\*\*  
## day 0.55937 0.23581 2.372 0.017688 \*   
## duration 5.81340 0.27628 21.042 < 2e-16 \*\*\*  
## contacttelephone 1.53191 0.17049 8.985 < 2e-16 \*\*\*  
## contactunknown -0.66186 0.18228 -3.631 0.000282 \*\*\*  
## monthaug -0.17729 0.24248 -0.731 0.464691   
## monthdec 0.52730 0.61319 0.860 0.389833   
## monthfeb 0.42012 0.30719 1.368 0.171432   
## monthjan -1.49941 0.44741 -3.351 0.000804 \*\*\*  
## monthjul -1.00721 0.24863 -4.051 5.10e-05 \*\*\*  
## monthjun -0.06223 0.27902 -0.223 0.823499   
## monthmar 2.41235 0.39917 6.043 1.51e-09 \*\*\*  
## monthmay -0.98687 0.23688 -4.166 3.10e-05 \*\*\*  
## monthnov -0.58368 0.28660 -2.037 0.041692 \*   
## monthoct 2.09718 0.37394 5.608 2.04e-08 \*\*\*  
## monthsep 1.54079 0.37987 4.056 4.99e-05 \*\*\*  
## poutcomeother 0.83135 0.24728 3.362 0.000774 \*\*\*  
## poutcomesuccess 3.10091 0.32784 9.459 < 2e-16 \*\*\*  
## poutcomeunknown -0.41159 0.17155 -2.399 0.016427 \*   
## ---  
## 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: 1919.3 on 2501 degrees of freedom  
## AIC: 1957.3  
##   
## Number of Fisher Scoring iterations: 6

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

require(coefplot)  
coefplot(log\_mod)



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

## Model #5 - K-Nearest Neighbours

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

## Confusion Matrix and Statistics  
##   
## test\_labels  
## knn\_pred no yes  
## no 990 40  
## yes 206 121  
##   
## Accuracy : 0.8187   
## 95% CI : (0.7972, 0.8389)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4006   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8278   
## Specificity : 0.7516   
## Pos Pred Value : 0.9612   
## Neg Pred Value : 0.3700   
## Prevalence : 0.8814   
## Detection Rate : 0.7296   
## Detection Prevalence : 0.7590   
## Balanced Accuracy : 0.7897   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(knn\_mod, test[-6])  
 actual<- test$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] 0.4966353

# 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 1045 43  
## yes 151 118  
##   
## Accuracy : 0.857   
## 95% CI : (0.8373, 0.8752)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.997   
##   
## Kappa : 0.4702   
##   
## Mcnemar's Test P-Value : 1.564e-14   
##   
## Sensitivity : 0.8737   
## Specificity : 0.7329   
## Pos Pred Value : 0.9605   
## Neg Pred Value : 0.4387   
## Prevalence : 0.8814   
## Detection Rate : 0.7701   
## Detection Prevalence : 0.8018   
## Balanced Accuracy : 0.8033   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(nn\_mod, test[-6])  
 actual<- test$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] 0.5541607

# 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 1054 48  
## yes 142 113  
##   
## Accuracy : 0.86   
## 95% CI : (0.8404, 0.878)  
## No Information Rate : 0.8814   
## P-Value [Acc > NIR] : 0.9924   
##   
## Kappa : 0.4655   
##   
## Mcnemar's Test P-Value : 1.51e-11   
##   
## Sensitivity : 0.8813   
## Specificity : 0.7019   
## Pos Pred Value : 0.9564   
## Neg Pred Value : 0.4431   
## Prevalence : 0.8814   
## Detection Rate : 0.7767   
## Detection Prevalence : 0.8121   
## Balanced Accuracy : 0.7916   
##   
## '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$y, k=10)  
 # create a function to do 10 folds of the data and run the statistics...  
 results <- lapply(folds, function(x) {  
 test<- banking[x,]  
 # \*\*\*\*\*\*\*\*\*- change y from 17 to ....  
 pred<- predict(sv\_m, test[-6])  
 actual<- test$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] 0.5485336

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