Part 4

jean wills

07/07/2020

## Project: Portuguese Bank Marketing Data

# TO FINISH - CHECK RESEARCH AND COMPARE

# PART 4 - Feature Selection Methods

Feature Selection Methods 1 - 3: # 1: Use correlations -see step 7c above - no change # 2: Rank Features By Importance - i.e. LVQ - see below # 3: Use Random Forest - see below # 4: Use PCA

NOTE: we can do feature selection on its own or as part of models \*\*\*

# Part 4 - #2 Feature Selection - Neural Network method - ALL numeric data

Learning Vector Quantization algorithm (LVQ) is an artificial neural network algorithm number=10-fold, repeated 10 times.

## convert BM\_mini to BM\_num first

BM\_mini <- read.csv("/Users/jeanwills/Desktop/CKME136/bank.csv", header=T, sep = ";", stringsAsFactors = T, na.strings = "NA")  
BM<- BM\_mini  
# step 2 - make changes  
# may delete later - attribute 'default' in column 5  
# BM<- select(BM,-5)  
# specific deletion where BM$previous == 275   
BM<- BM[grep("275", BM$previous, invert=TRUE),]  
# 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$duration/60  
#  
# convert BM to BM\_num now  
#  
BM\_num <- BM  
BM\_num$job<- as.numeric(BM\_num$job) #12  
# marital: 1-single, 2-married, 3-divorced  
BM\_num$marital<- ifelse(BM\_num$marital == c("single"), 1,   
 ifelse(BM\_num$marital== c("married"), 2, 3))  
# education: 0:unknown, 1: primary, 2:secondary, 3:divorced  
BM\_num$education<- ifelse(BM\_num$education == c("unknown"), 0,   
 ifelse(BM\_num$education == c("primary"), 1,   
 ifelse(BM\_num$education == c("secondary"), 2, 3)))  
# default, housing, loan: if yes then 0 else 1  
# BM\_num$housing<- as.numeric(BM\_num$housing) #2  
BM\_num$default<- ifelse(BM\_num$default == c("yes"), 0, 1) #2  
BM\_num$housing<- ifelse(BM\_num$housing == c("yes"), 0, 1) #2  
BM\_num$loan<- ifelse(BM\_num$loan == c("yes"), 0, 1) #2  
BM\_num$contact<- as.numeric(BM\_num$contact) #3  
# month: jan:1, feb:2.....dec:12  
BM\_num$month<- ifelse(BM\_num$month == "jan", 1,   
 ifelse(BM\_num$month == "feb", 2,   
 ifelse(BM\_num$month == "mar", 3,  
 ifelse(BM\_num$month == "apr", 4,   
 ifelse(BM\_num$month == "may", 5,   
 ifelse(BM\_num$month == "jun", 6,  
 ifelse(BM\_num$month == "jul", 7,  
 ifelse(BM\_num$month == "aug", 8,  
 ifelse(BM\_num$month == "sep", 9,  
 ifelse(BM\_num$month == "oct", 10,  
 ifelse(BM\_num$month == "nov", 11, 12)))))))))))  
# poutcome: 0:unknown,other, 1:failure, 2: success  
BM\_num$poutcome<- ifelse(BM\_num$poutcome == c("failure"), 1, ifelse(BM\_num$poutcome== c("success"), 2, 0))   
# result: BM\_num with only numeric data (NOT scaled)

## now ready:

# extra:  
# num<- subset(BM\_mini\_num, select = c("age", "balance", "day", "duration", "campaign", "pdays", "previous", "y"))  
# y stays as a category  
# parallel processing is supported by caret package  
library(parallel)  
library(doMC)

## Loading required package: foreach

## Loading required package: iterators

library(caret)

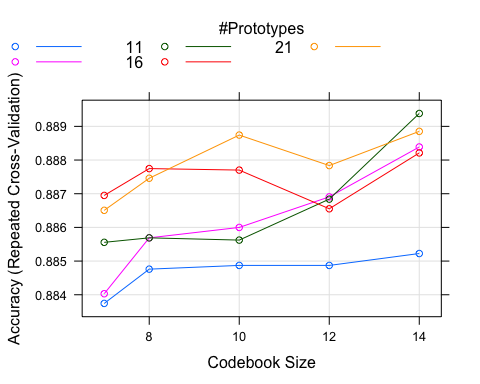
## Loading required package: lattice

## Loading required package: ggplot2

# get 4 cores:   
numCores<- detectCores()  
registerDoMC(cores = numCores)  
set.seed(7)  
library(mlbench)  
# prepare training scheme  
control <- trainControl(method="repeatedcv", number=10, repeats=10)  
# manual example: grid<- expand.grid(size=c(5,10,20,50), k=c(1,2,3,4,5)) and add tuneGrid=grid as model parameter and delete tuneLength  
# scale = normalizing the dataset  
# tuneLength creates a large tuning grid of hyperparameters with different configurations and gives best model based on optimizing metric  
# that makes most sense for the model  
model <- train(y~., data=BM\_num, method="lvq", preProcess="scale", trControl=control, tuneLength=5)  
# using tunelength, final values used for model were size=14 and k=11  
# estimate variable importance  
print(model)

## Learning Vector Quantization   
##   
## 4521 samples  
## 16 predictor  
## 2 classes: 'no', 'yes'   
##   
## Pre-processing: scaled (16)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 4069, 4069, 4069, 4069, 4069, 4069, ...   
## Resampling results across tuning parameters:  
##   
## size k Accuracy Kappa   
## 7 1 0.8837429 0.1109742  
## 7 6 0.8840300 0.1180093  
## 7 11 0.8855566 0.1412831  
## 7 16 0.8869498 0.1643743  
## 7 21 0.8865071 0.1504003  
## 8 1 0.8847604 0.1235241  
## 8 6 0.8856896 0.1253541  
## 8 11 0.8856898 0.1296874  
## 8 16 0.8877465 0.1562352  
## 8 21 0.8874591 0.1541571  
## 10 1 0.8848711 0.1091688  
## 10 6 0.8859989 0.1312677  
## 10 11 0.8856228 0.1227119  
## 10 16 0.8877014 0.1443545  
## 10 21 0.8887416 0.1555325  
## 12 1 0.8848708 0.1004822  
## 12 6 0.8869064 0.1268625  
## 12 11 0.8868389 0.1240622  
## 12 16 0.8865526 0.1383382  
## 12 21 0.8878355 0.1368858  
## 14 1 0.8852244 0.1753003  
## 14 6 0.8883880 0.1943095  
## 14 11 0.8893830 0.2170989  
## 14 16 0.8882112 0.1973477  
## 14 21 0.8888515 0.2098529  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were size = 14 and k = 11.

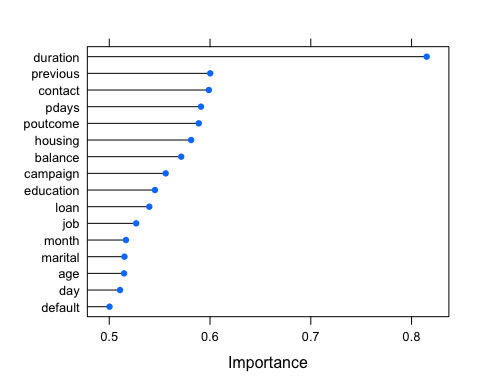
plot(model)



importance <- varImp(model, scale=FALSE)  
print(importance)

## ROC curve variable importance  
##   
## Importance  
## duration 0.8150  
## previous 0.6002  
## contact 0.5988  
## pdays 0.5910  
## poutcome 0.5888  
## housing 0.5812  
## balance 0.5714  
## campaign 0.5560  
## education 0.5453  
## loan 0.5397  
## job 0.5267  
## month 0.5166  
## marital 0.5151  
## age 0.5147  
## day 0.5107  
## default 0.5003

plot(importance)



## results: Keep above ~0.5 and all else could delete.

duration (0.815), previous (0.6), contact (0.6), pdays (0.59), poutcome (0.59), [housing (0.58), balance (0.57), campaign (0.556), education (0.54), loan (0.54), age (0.51), day (0.51)].

# Part 4 - #3 - Feature Selection - Random Forest - ALL numeric data BM\_num

Recursive Feature Elimination or RFE. Random forest function rfFuncs, method = cross-validation, number=10-fold, repeated 10 times.

# Kappa should be >0.6 “machine learning with R” pg.324. Kappa takes imbalance into account.

Takes too long to do- over 15 hours and still running - use mini version - results below.

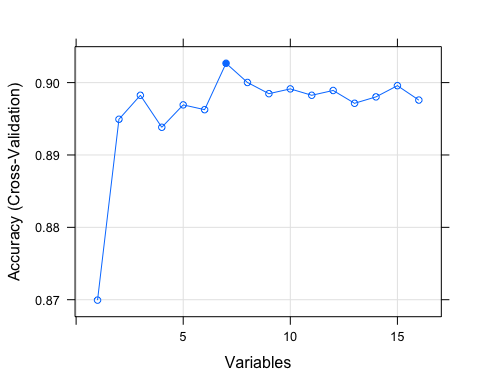
set.seed(7)  
# simple version:  
# r\_tree=randomForest(y ~.,data=BM,nodesize=25,ntree=200)  
# library(lattice)  
# library(ggplot2)  
# define the control using a random forest selection function rfFuncs  
control <- rfeControl(functions=rfFuncs, method="cv", number=10, repeats=10)  
# run the RFE algorithm  
results <- rfe(BM\_num[,1:16], BM\_num[,17], sizes=c(1:16), rfeControl=control)  
# summarize the results  
print(results)

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (10 fold)   
##   
## Resampling performance over subset size:  
##   
## Variables Accuracy Kappa AccuracySD KappaSD Selected  
## 1 0.8699 0.1810 0.010291 0.06363   
## 2 0.8949 0.3570 0.014701 0.07489   
## 3 0.8982 0.3188 0.013007 0.08938   
## 4 0.8938 0.3770 0.011546 0.05959   
## 5 0.8969 0.3730 0.009646 0.05428   
## 6 0.8963 0.3526 0.011171 0.06231   
## 7 0.9027 0.3669 0.012005 0.09275 \*  
## 8 0.9000 0.3512 0.012181 0.09289   
## 9 0.8985 0.3883 0.013031 0.08490   
## 10 0.8991 0.3842 0.014412 0.09266   
## 11 0.8982 0.3712 0.013220 0.08625   
## 12 0.8989 0.3698 0.013669 0.09468   
## 13 0.8971 0.3433 0.012554 0.09034   
## 14 0.8980 0.3398 0.011232 0.08067   
## 15 0.8996 0.3478 0.012591 0.08863   
## 16 0.8976 0.3552 0.013941 0.09439   
##   
## The top 5 variables (out of 7):  
## duration, poutcome, month, day, contact

# list the chosen features  
predictors(results)

## [1] "duration" "poutcome" "month" "day" "contact" "pdays" "previous"

# plot the results  
plot(results, type=c("g", "o"))



#results show top 5 and 8: duration, month, poutcome, day, contact, pdays, previous, age  
# 2nd run top 9: duration, month, poutcome, day, pdays, contact, previous, age, housing  
# 2nd run: accuracy with duration is 0.8697 then keeps going up to 0.9038 to housing (#9)  
# 2nd run: but kappa at 0.4525 for the 9th so still not great- s/b over 0.6  
# 5 agree with above

## Part 4 - #4 - PCA

Principal component analysis (PCA) is a technique for reducing the dimensionality of datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. I think BM$y should not be in dataset?? just ignore it for now. REQUIRES SCALED NUMERIC DATA or option in formula to do that.

version 1: lets try with all numeric, not scaled -> BM\_num or BM\_dummy version 2: lets try with all numeric and numeric scaled (min-max) - BM\_num or BM\_dummy -> BM\_num\_scale version 3: lets try with all numeric and numeric scaled (z-scale) - BM\_num or BM\_dummy-> BM\_num\_z

we can use larger dataset, but for this example, let’s take BM\_num from above and scale it.

# all data should be numeric - if cor=TRUE then data is scaled and centered  
# BM\_num must be numeric  
BM\_num$y<- ifelse(BM\_num$y==c("yes"), 0, 1)  
# str(BM\_num)  
pc\_BM<- princomp(BM\_num, cor=TRUE)  
# pc\_BM$scores  
summary(pc\_BM)

## Importance of components:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5  
## Standard deviation 1.5522514 1.25646494 1.24016345 1.16243031 1.06916357  
## Proportion of Variance 0.1417344 0.09286495 0.09047091 0.07948495 0.06724181  
## Cumulative Proportion 0.1417344 0.23459933 0.32507024 0.40455519 0.47179700  
## Comp.6 Comp.7 Comp.8 Comp.9 Comp.10  
## Standard deviation 1.05404552 0.99892457 0.96637992 0.96090688 0.92732679  
## Proportion of Variance 0.06535364 0.05869708 0.05493471 0.05431424 0.05058441  
## Cumulative Proportion 0.53715065 0.59584772 0.65078244 0.70509667 0.75568108  
## Comp.11 Comp.12 Comp.13 Comp.14 Comp.15  
## Standard deviation 0.91332396 0.8802208 0.8180048 0.75402400 0.72677302  
## Proportion of Variance 0.04906827 0.0455758 0.0393607 0.03344425 0.03107053  
## Cumulative Proportion 0.80474936 0.8503252 0.8896859 0.92313010 0.95420063  
## Comp.16 Comp.17  
## Standard deviation 0.68624722 0.55466568  
## Proportion of Variance 0.02770207 0.01809729  
## Cumulative Proportion 0.98190271 1.00000000

#

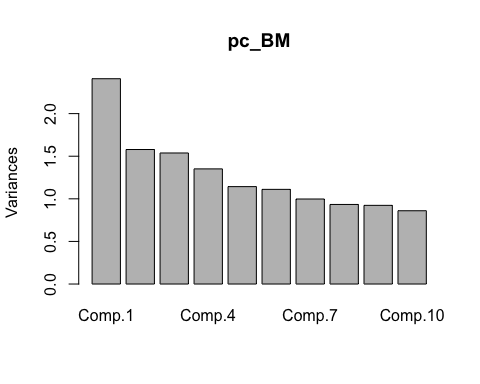
# RESULTS: keep Comp. st.dev. >= 1.0

## BM\_num (cor=TRUE): keep first 6 comp. - THIS EXAMPLE

BM\_num\_scale (cor=FALSE): keep only first 2 comp. BM\_num\_scale (cor=TRUE): keep first 7 comp. BM\_num\_z: (cor=FALSE): keep first 7 comp. BM\_num\_z (cor=TRUE): keep first 7 comp.

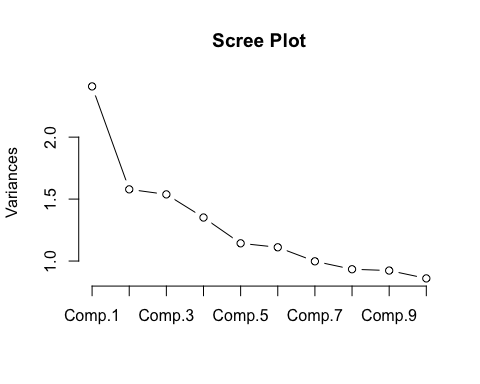
## then plot

plot(pc\_BM)



## scree plot

# print(pc\_BM)  
screeplot(pc\_BM, type="line", main="Scree Plot")



# again, use variance > 1.0 so keep first 7 components

## see actual component values

pc\_BM$loadings

##   
## Loadings:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9  
## age 0.138 0.646 0.112 0.118   
## job 0.285 -0.178 0.106 0.300 0.498 0.295 0.401  
## marital 0.566 -0.169 0.248 0.370  
## education 0.258 -0.402 0.126 -0.171 0.267 0.425  
## default 0.494 -0.269 -0.200 0.726 -0.314  
## balance 0.170 0.103 0.520 -0.479 -0.327 0.356  
## housing 0.506 0.135 0.205 0.217 -0.419  
## loan 0.585 -0.122 0.430 -0.322   
## contact 0.271 -0.386 0.102 -0.174 0.181 0.248  
## day 0.179 0.149 -0.127 -0.610 0.204   
## month 0.461 0.192 -0.122 -0.360 -0.144   
## duration 0.151 -0.672 -0.104 -0.171 -0.106   
## campaign 0.130 0.228 -0.122 -0.571 0.137 0.192  
## pdays -0.539 -0.175 0.145   
## previous -0.493 0.146   
## poutcome -0.538   
## y 0.231 -0.278 0.551 0.153   
## Comp.10 Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17  
## age 0.138 0.691   
## job 0.300 -0.180 -0.376 -0.148   
## marital -0.328 -0.104 -0.559   
## education -0.326 0.110 0.475 0.273 0.111 0.187   
## default   
## balance 0.354 -0.105 0.149 -0.211 -0.131   
## housing 0.236 0.319 0.303 0.187 -0.356 -0.129   
## loan -0.546 0.159   
## contact 0.238 0.738 -0.114 0.107   
## day -0.675 0.182   
## month -0.255 -0.627 0.324   
## duration -0.142 0.609 -0.197   
## campaign 0.265 0.663   
## pdays -0.298 0.746   
## previous 0.246 0.304 -0.143 0.712 -0.152   
## poutcome 0.162 -0.198 -0.504 -0.604   
## y -0.105 0.646 -0.253 -0.146   
##   
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9  
## SS loadings 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000  
## Proportion Var 0.059 0.059 0.059 0.059 0.059 0.059 0.059 0.059 0.059  
## Cumulative Var 0.059 0.118 0.176 0.235 0.294 0.353 0.412 0.471 0.529  
## Comp.10 Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17  
## SS loadings 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000  
## Proportion Var 0.059 0.059 0.059 0.059 0.059 0.059 0.059 0.059  
## Cumulative Var 0.588 0.647 0.706 0.765 0.824 0.882 0.941 1.000

# RESULTS:

## BM\_num (cor=TRUE): keep all attributes

BM\_num\_scale (cor=FALSE): keep only job, month BM\_num\_scale (cor=TRUE): keep all attributes BM\_num\_z (cor=TRUE): keep all attributes

BM\_num\_z (cor=FALSE): keep age, campaign, job, marital, education, contact, month ONLY (happen to be 7) use first 7 components, but there is much less data: comp.1 = 0.173*job2 -0.98*job comp.2 = 0.99*month etc…. comp.7 = 0.114*age + 0.781*campaign + 0.586*job7