pruning.R

shiwam

2022-01-30

df=read.csv("new\_train.csv");head(df)

## age job marital education default housing loan contact  
## 1 49 blue-collar married basic.9y unknown no no cellular  
## 2 37 entrepreneur married university.degree no no no telephone  
## 3 78 retired married basic.4y no no no cellular  
## 4 36 admin. married university.degree no yes no telephone  
## 5 59 retired divorced university.degree no no no cellular  
## 6 29 admin. single university.degree no no no cellular  
## month day\_of\_week duration campaign pdays previous poutcome y  
## 1 nov wed 227 4 999 0 nonexistent no  
## 2 nov wed 202 2 999 1 failure no  
## 3 jul mon 1148 1 999 0 nonexistent yes  
## 4 may mon 120 2 999 0 nonexistent no  
## 5 jun tue 368 2 999 0 nonexistent no  
## 6 aug wed 256 2 999 0 nonexistent no

df[,c(2,3,4,5,6,7,8,9,10,15,16)]=lapply(df[,c(2,3,4,5,6,7,8,9,10,15,16)], FUN = as.factor)

str(df)

## 'data.frame': 32950 obs. of 16 variables:  
## $ age : int 49 37 78 36 59 29 26 30 50 33 ...  
## $ job : Factor w/ 12 levels "admin.","blue-collar",..: 2 3 6 1 6 1 9 2 2 1 ...  
## $ marital : Factor w/ 4 levels "divorced","married",..: 2 2 2 2 1 3 3 2 2 3 ...  
## $ education : Factor w/ 8 levels "basic.4y","basic.6y",..: 3 7 1 7 7 7 3 1 1 4 ...  
## $ default : Factor w/ 3 levels "no","unknown",..: 2 1 1 1 1 1 1 1 2 1 ...  
## $ housing : Factor w/ 3 levels "no","unknown",..: 1 1 1 3 1 1 1 3 1 3 ...  
## $ loan : Factor w/ 3 levels "no","unknown",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ contact : Factor w/ 2 levels "cellular","telephone": 1 2 1 2 1 1 2 1 2 1 ...  
## $ month : Factor w/ 10 levels "apr","aug","dec",..: 8 8 4 7 5 2 2 8 7 4 ...  
## $ day\_of\_week: Factor w/ 5 levels "fri","mon","thu",..: 5 5 2 2 4 5 5 5 1 4 ...  
## $ duration : int 227 202 1148 120 368 256 449 126 574 498 ...  
## $ campaign : int 4 2 1 2 2 2 1 2 1 5 ...  
## $ pdays : int 999 999 999 999 999 999 999 999 999 999 ...  
## $ previous : int 0 1 0 0 0 0 0 0 0 0 ...  
## $ poutcome : Factor w/ 3 levels "failure","nonexistent",..: 2 1 2 2 2 2 2 2 2 2 ...  
## $ y : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 1 1 1 ...

levels(df$job)=0:(length(levels(df$job))-1)  
levels(df$marital)=0:(length(levels(df$marital))-1)  
levels(df$contact)=0:(length(levels(df$contact))-1)  
levels(df$education)=0:(length(levels(df$education))-1)  
levels(df$default)=0:(length(levels(df$default))-1)  
levels(df$housing)=0:(length(levels(df$housing))-1)  
levels(df$loan)=0:(length(levels(df$loan))-1)  
levels(df$month)=0:(length(levels(df$month))-1)  
levels(df$day\_of\_week)=0:(length(levels(df$day\_of\_week))-1)  
levels(df$poutcome)=0:(length(levels(df$poutcome))-1)  
levels(df$y)=0:(length(levels(df$y))-1)  
df[,c(2,3,4,5,6,7,8,9,10,15,16)]=lapply(df[,c(2,3,4,5,6,7,8,9,10,15,16)], FUN = as.character)  
df[,c(2,3,4,5,6,7,8,9,10,15,16)]=lapply(df[,c(2,3,4,5,6,7,8,9,10,15,16)], FUN = as.numeric)  
head(df)#NOW ALL ARE IN NUMERIC VARIABLE

## age job marital education default housing loan contact month day\_of\_week  
## 1 49 1 1 2 1 0 0 0 7 4  
## 2 37 2 1 6 0 0 0 1 7 4  
## 3 78 5 1 0 0 0 0 0 3 1  
## 4 36 0 1 6 0 2 0 1 6 1  
## 5 59 5 0 6 0 0 0 0 4 3  
## 6 29 0 2 6 0 0 0 0 1 4  
## duration campaign pdays previous poutcome y  
## 1 227 4 999 0 1 0  
## 2 202 2 999 1 0 0  
## 3 1148 1 999 0 1 1  
## 4 120 2 999 0 1 0  
## 5 368 2 999 0 1 0  
## 6 256 2 999 0 1 0

df1=df[,-c(13,14)]  
dftest=read.csv("new\_test.csv");head(dftest)

## age job marital education default housing loan contact month day\_of\_week  
## 1 32 4 0 6 0 0 0 0 3 3  
## 2 37 10 3 6 0 0 0 0 4 3  
## 3 55 5 0 5 1 2 0 0 3 2  
## 4 44 2 1 0 1 0 0 1 4 3  
## 5 28 0 2 3 0 0 0 0 5 0  
## 6 45 10 1 2 0 0 0 0 1 0  
## duration campaign poutcome  
## 1 131 5 1  
## 2 100 1 1  
## 3 131 2 1  
## 4 48 2 1  
## 5 144 2 1  
## 6 126 3 1

library(rpart)

## Warning: package 'rpart' was built under R version 4.0.5

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.5

####NOW PARTITIONG THE TRAINING DATASET INTO VALIDATION PART BECAUSE THE GIVEN DATASET IS MUCH LARGER  
partidx=sample(1:nrow(df1),26000,replace = F)#20% of train dataset convert to validation dataset  
df1train=df1[partidx,];head(df1train)#training

## age job marital education default housing loan contact month day\_of\_week  
## 26528 30 7 1 3 0 2 0 1 6 1  
## 17866 32 6 1 2 0 0 0 1 6 0  
## 25105 34 0 1 2 0 0 0 1 4 2  
## 15671 28 4 2 6 0 0 0 1 4 4  
## 19419 38 0 1 3 0 2 0 1 6 2  
## 25013 47 7 0 3 0 0 0 1 6 1  
## duration campaign poutcome y  
## 26528 92 2 1 0  
## 17866 101 2 1 0  
## 25105 238 1 1 0  
## 15671 361 1 1 0  
## 19419 26 3 1 0  
## 25013 143 1 1 0

partidx1=sample(1:nrow(df1[-partidx]),6950,replace=F)

df1valid=df1[partidx1,]  
head(df1valid)#VALIDATION

## age job marital education default housing loan contact month day\_of\_week  
## 10501 52 4 1 6 0 0 0 0 3 4  
## 20059 49 1 1 1 0 2 0 1 6 4  
## 1341 31 9 2 6 0 0 0 0 7 3  
## 30470 36 2 0 6 0 2 0 0 7 1  
## 27051 60 6 1 6 0 2 0 0 3 4  
## 25151 49 0 1 3 0 0 0 0 3 0  
## duration campaign poutcome y  
## 10501 119 2 1 0  
## 20059 97 1 1 0  
## 1341 110 1 0 0  
## 30470 292 1 1 0  
## 27051 416 4 1 1  
## 25151 366 2 1 0

#BUILD MODEL ON TRAINING PARTITION  
mod1=rpart(y~.,method = "class",data=df1train,control = rpart.control(  
 cp=0,minsplit = 2,minbucket = 1,maxcompete = 0,maxsurrogate = 0,xval = 0,  
 parms=list(split="gini")))  
predicted\_y=predict(mod1,dftest,type = "class")  
df1test=cbind(dftest,predicted\_y);head(df1test)

## age job marital education default housing loan contact month day\_of\_week  
## 1 32 4 0 6 0 0 0 0 3 3  
## 2 37 10 3 6 0 0 0 0 4 3  
## 3 55 5 0 5 1 2 0 0 3 2  
## 4 44 2 1 0 1 0 0 1 4 3  
## 5 28 0 2 3 0 0 0 0 5 0  
## 6 45 10 1 2 0 0 0 0 1 0  
## duration campaign poutcome predicted\_y  
## 1 131 5 1 0  
## 2 100 1 1 0  
## 3 131 2 1 0  
## 4 48 2 1 0  
## 5 144 2 1 1  
## 6 126 3 1 0

##########PRUNING PROCESS##################  
  
#AVOID OVERFITTING  
#\* FULL GROWN TREE LEADS TO COMPLETE OVERFITTING OF DATA  
#\* POOR PERFORMANCE ON NEW DATA  
#\* PRUNE THE FULL GROEN TREE BACK TO A LEVEL WHERE IT DOESN'T OVERFIT THE DATA OR FIT NOISE  
  
#PRUNING PROCESS  
#VALIDATION PARTITION: MISCLASSIFICATION ERROR VS NUMBER OF DECISION NODES  
#TOTAL NUMBER OF NODES IN FULL GROWN TREES  
nrow(mod1$frame)

## [1] 4831

#number of decision nodes  
nrow(mod1$splits)

## [1] 2415

#no of terminal nodes  
nrow(mod1$frame)-nrow(mod1$splits)

## [1] 2416

#NODE NUMBER  
head(row.names(mod1$frame))

## [1] "1" "2" "4" "8" "16" "32"

#COERCION TO INTEGER  
tosses1=as.integer(row.names(mod1$frame))  
tosses2=sort(tosses1);head(tosses2)

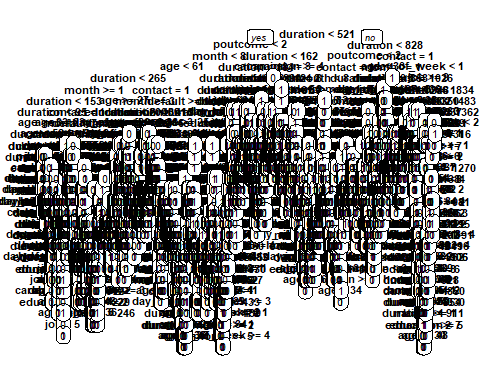
## [1] 1 2 3 4 5 6

#COUNTER FOR NODES TO BE SNIPPED OFF  
i=1  
mod1splitv=NULL  
mod1strainv=NULL  
mod1svalidv=NULL  
errtrainv=NULL  
errvalidv=NULL  
for (y in mod1$frame$var) {  
 if (as.character(y)!="<leaf>" & i<length(tosses2)) {  
 tosses3=tosses2[(i+1):length(tosses2)]  
 mod1split=snip.rpart(mod1,toss = tosses3)  
 mod1splitv=c(mod1splitv,mod1split)  
 mod1strain=predict(mod1split,df1train,type="class")  
 mod1strainv=c(mod1strainv,mod1strain)  
 #for validation dataset   
 mod1svalid=predict(mod1split,df1valid,type="class")  
 mod1svalidv=c(mod1svalidv,mod1svalid)  
 errtrain=mean(mod1strain!=df1train$y)  
 errtrainv=c(errtrainv,errtrain)  
 errvalid=mean(mod1svalid!=df1valid$y)  
 errvalidv=c(errvalidv,errvalid)  
 }  
 i=i+1  
}  
  
#ERROR RATE VS NO OF SPLITS  
DF=data.frame("DECISION\_NODES"=1:nrow(mod1$splits),"ERROR\_TRAINING"=errtrainv,  
 "ERROR\_VALIDATION"=errvalidv,check.names=F)  
  
head(DF,20)#HERE WE SEE RATE OF DECREASE IN THE TRAINING PARTITION OR VALIDATION PARTITION

## DECISION\_NODES ERROR\_TRAINING ERROR\_VALIDATION  
## 1 1 1.133462e-01 0.11597122  
## 2 2 1.055769e-01 0.10589928  
## 3 3 9.880769e-02 0.10000000  
## 4 4 9.880769e-02 0.10000000  
## 5 5 9.580769e-02 0.09697842  
## 6 6 9.350000e-02 0.09539568  
## 7 7 9.350000e-02 0.09539568  
## 8 8 9.350000e-02 0.09539568  
## 9 9 9.273077e-02 0.09539568  
## 10 10 9.273077e-02 0.09539568  
## 11 11 9.273077e-02 0.09539568  
## 12 12 9.273077e-02 0.09539568  
## 13 13 9.273077e-02 0.09539568  
## 14 14 9.192308e-02 0.09525180  
## 15 15 9.146154e-02 0.09352518  
## 16 16 9.146154e-02 0.09352518  
## 17 17 9.126923e-02 0.09410072  
## 18 18 9.061538e-02 0.09424460  
## 19 19 9.019231e-02 0.09395683  
## 20 20 8.976923e-02 0.09366906

##TREE AFTER LAST SNIP  
prp(mod1split,varlen = 0,cex = 0.7,extra = 0,compress = T,Margin = 0,digits = 0)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



nrow(mod1split$frame)

## [1] 4831

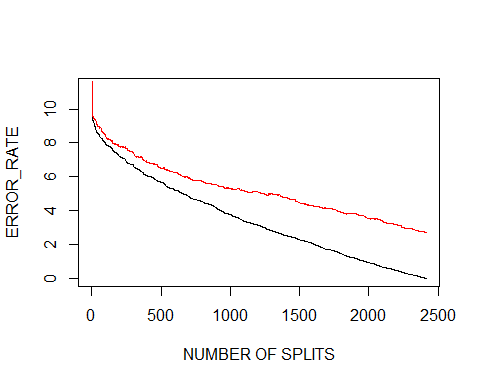
nrow(mod1split$splits)

## [1] 2415

nrow(mod1split$frame)-nrow(mod1split$splits)

## [1] 2416

#plot of error rate VS number of splits  
nsplits=1:nrow(mod1$split)  
plot(nsplits,100\*errtrainv,type = "l",xlab = "NUMBER OF SPLITS",ylab = "ERROR\_RATE")  
lines(nsplits,100\*errvalidv,col="red")



#MINIMUM ERROR TREE AND BEST PRUNED TREEE  
min(errvalidv)

## [1] 0.02733813

MET=min(nsplits[which(errvalidv==min(errvalidv))]);MET

## [1] 2402

#STANDARD ERROR  
sqrt(var(errvalidv)/length(errvalidv))

## [1] 0.0003299524

#####################  
#BEST PRUNED TREE NEAR FIRST MINIMA WITHIN STANDARD ERROR  
met1std=min(errvalidv)+sqrt(var(errvalidv)/length(errvalidv));met1std

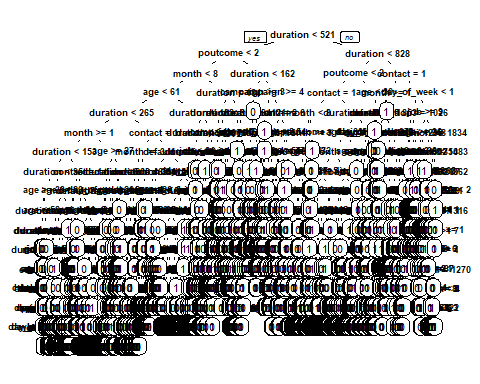
## [1] 0.02766808

BPT=DF[which(errvalidv > min(errvalidv) & errvalidv < met1std&nsplits,MET),][1,1]  
BPT#here the resulting decision node are required so we can removing remaining node after this resulting node

## [1] 2367

#BUT THIS IS NOT EXACT ANSWER 2-3 NODES GIVES BETTER RESULT THAN THIS ABOVE RESULTING NODE IN BPT  
  
toss3=tosses2[(BPT+1):length(tosses2)]  
mod1best=snip.rpart(mod1,toss = toss3)  
prp(mod1best,varlen = 0,cex=0.7,extra=0,compress = T,Margin = 0,digits = 0,split.cex = 0.8,under.cex = 0.8)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



#SIMILARLY WE FIND CLASSIFICATION ACCURACY & MISCLASSIFICATION ERROR FOR TRAINING, VALIDATION & TESTING DATASET  
bmodtrain1=predict(mod1best,df1train,type="class")  
table("PREDICTED\_VALUE"=bmodtrain1,"ACTUAL\_VALUE"=df1train$y)

## ACTUAL\_VALUE  
## PREDICTED\_VALUE 0 1  
## 0 22804 576  
## 1 249 2371

#CLASSIFICATION ACCURACY  
mean(bmodtrain1==df1train$y)

## [1] 0.9682692

#CLASSIFICATION ERROR  
mean(bmodtrain1!=df1train$y)

## [1] 0.03173077

######## in validation partition#######  
bmodvalid1=predict(mod1best,df1valid,type="class")  
table("ACTUAL\_VALUE"=df1valid$y,"PREDICTED\_VALUE"=bmodvalid1)

## PREDICTED\_VALUE  
## ACTUAL\_VALUE 0 1  
## 0 6007 137  
## 1 219 587

#CLASSIFICATION ACCURACY  
mean(bmodvalid1==df1valid$y)

## [1] 0.948777

#CLASSIFICATION ERROR  
mean(bmodvalid1!=df1valid$y)

## [1] 0.05122302

##########in testing partition########  
bmodtest1=predict(mod1best,df1test,type="class")  
table("ACTUAL\_VALUE"=df1test$predicted\_y,"PREDICTED\_VALUE"=bmodtest1)

## PREDICTED\_VALUE  
## ACTUAL\_VALUE 0 1  
## 0 7396 24  
## 1 241 577

#CLASSIFICATION ACCURACY  
mean(bmodtest1==df1test$predicted\_y)

## [1] 0.967832

#CLASSIFICATION ERROR  
mean(bmodtest1!=df1test$predicted\_y)

## [1] 0.032168