HW1\_2\_1a.R

neeraj

Mon Feb 1 19:44:07 2016

##Neeraj Asthana (nasthan2)  
##CS 498 HW1  
options(warn=-1)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

## Note: the specification for S3 class "family" in package 'MatrixModels' seems equivalent to one from package 'lme4': not turning on duplicate class definitions for this class.

library(klaR)

## Loading required package: MASS

##2.1a  
#load data  
setwd('/home/neeraj/Documents/UIUC/CS 498/CS498MachineLearning/HW1')  
raw\_data<-read.csv('pima.csv', header=FALSE)  
x\_vector <- raw\_data[-c(9)]  
y\_labels <- raw\_data[,9]  
  
#train naive bayes model  
trscore<-array(dim=10)  
tescore<-array(dim=10)  
for (wi in 1:10){  
 #create training and testing sets  
 datasplit <- createDataPartition(y=y\_labels, p=.8, list=FALSE)  
 trainx <- x\_vector[datasplit,]  
 trainy <- y\_labels[datasplit]  
 testx <- x\_vector[-datasplit,]  
 testy <- y\_labels[-datasplit]  
   
 #splitting positive and negative examples  
 trposflag<-trainy>0  
 positive\_examples <- trainx[trposflag, ]  
 negative\_examples <- trainx[!trposflag,]  
   
 #calculate means and sds  
 ptrmean<-sapply(positive\_examples, mean, na.rm=TRUE)  
 ntrmean<-sapply(negative\_examples, mean, na.rm=TRUE)  
 ptrsd<-sapply(positive\_examples, sd, na.rm=TRUE)  
 ntrsd<-sapply(negative\_examples, sd, na.rm=TRUE)  
   
 #calculate offsets and scales  
 ptroffsets<-t(t(trainx)-ptrmean)  
 ptrscales<-t(t(ptroffsets)/ptrsd)  
   
 pteoffsets<-t(t(testx)-ptrmean)  
 ptescales<-t(t(pteoffsets)/ptrsd)  
 ptelogs<--(1/2)\*rowSums(apply(ptescales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ptrsd))  
 nteoffsets<-t(t(testx)-ntrmean)  
 ntescales<-t(t(nteoffsets)/ntrsd)  
 ntelogs<--(1/2)\*rowSums(apply(ntescales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ntrsd))  
 lvwte<-ptelogs>ntelogs  
 gotright<-lvwte==testy  
 tescore[wi]<-sum(gotright)/(sum(gotright)+sum(!gotright))  
}  
accuracy <- sum(tescore) / length(tescore)  
#accuracy after cross validating 10 times  
accuracy

## [1] 0.7562092

##2.1b  
#replace '0' in columns   
x\_vector\_copy <- x\_vector  
for (i in c(3, 4, 6, 8)){  
 non\_values <- x\_vector[, i]==0  
 x\_vector\_copy[non\_values, i]=NA  
}  
  
#train naive bayes model  
trscore<-array(dim=10)  
tescore<-array(dim=10)  
for (wi in 1:10){  
 #create training and testing sets  
 datasplit <- createDataPartition(y=y\_labels, p=.8, list=FALSE)  
 trainx <- x\_vector\_copy[datasplit,]  
 trainy <- y\_labels[datasplit]  
 testx <- x\_vector\_copy[-datasplit,]  
 testy <- y\_labels[-datasplit]  
   
 #splitting positive and negative examples  
 trposflag<-trainy>0  
 positive\_examples <- trainx[trposflag, ]  
 negative\_examples <- trainx[!trposflag,]  
   
 #calculate means and sds  
 ptrmean<-sapply(positive\_examples, mean, na.rm=TRUE)  
 ntrmean<-sapply(negative\_examples, mean, na.rm=TRUE)  
 ptrsd<-sapply(positive\_examples, sd, na.rm=TRUE)  
 ntrsd<-sapply(negative\_examples, sd, na.rm=TRUE)  
   
 #calculate offsets and scales  
 ptroffsets<-t(t(trainx)-ptrmean)  
 ptrscales<-t(t(ptroffsets)/ptrsd)  
   
 pteoffsets<-t(t(testx)-ptrmean)  
 ptescales<-t(t(pteoffsets)/ptrsd)  
 ptelogs<--(1/2)\*rowSums(apply(ptescales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ptrsd))  
 nteoffsets<-t(t(testx)-ntrmean)  
 ntescales<-t(t(nteoffsets)/ntrsd)  
 ntelogs<--(1/2)\*rowSums(apply(ntescales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ntrsd))  
 lvwte<-ptelogs>ntelogs  
 gotright<-lvwte==testy  
 tescore[wi]<-sum(gotright)/(sum(gotright)+sum(!gotright))  
}  
accuracy <- sum(tescore) / length(tescore)  
#accuracy after cross validating 10 times  
accuracy

##accuracy decreases when NA's are removed

## [1] 0.7124183

##2.1c  
#create training and testing sets  
test\_accuracies <- array(dim=10)  
for (wi in 1:10){  
 datasplit <- createDataPartition(y=y\_labels, p=.8, list=FALSE)  
 trainx <- x\_vector[datasplit,]  
 trainy <- y\_labels[datasplit]  
 testx <- x\_vector[-datasplit,]  
 testy <- y\_labels[-datasplit]  
  
 #train naive bayes model using klaR package  
 tr <- trainControl(method='cv' , number=10)  
 model <- train (trainx , factor(trainy) , 'nb' , trControl=tr)  
  
 #prediction  
 predictions <- predict(model, newdata=testx)  
 correct <- length(testy[testy == predictions])  
 wrong <- length(testy[testy != predictions])  
 accuracy <- correct / (correct + wrong)  
 test\_accuracies[wi] <- accuracy  
}  
cross\_validation\_accuracy <- sum(test\_accuracies)/length(test\_accuracies)  
#accuracy after cross validating 10 times  
cross\_validation\_accuracy

## [1] 0.7712418

##2.1d  
#create data sets for training and testing  
datasplit<-createDataPartition(y=y\_labels, p=.8, list=FALSE)  
trainx <- x\_vector[datasplit,]  
trainy <- y\_labels[datasplit]  
testx <- x\_vector[-datasplit,]  
testy <- y\_labels[-datasplit]  
  
#train svm  
svm <- svmlight(trainx, factor(trainy), pathsvm="/home/neeraj/Documents/UIUC/svm\_light")  
labels <- predict(svm, testx)  
answers <- labels$class  
  
#accuracy  
correct <- sum(answers == testy)  
wrong <- sum(answers != testy)  
accuracy <- correct / (correct + wrong)  
accuracy

## [1] 0.7581699