Pima Indians

Viraj Bhalala

CS 498 Applied ML

HW 1 problem 1

```
library(klaR)
```

```
## Loading required package: MASS
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
setwd("../PrimaIndians/")
wdat <- read.table("pima-indians-diabetes.data", sep = ",")
# Source: http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabete
s/
# Col names of the data set
# 1. Number of times pregnant
# 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
# 3. Diastolic blood pressure (mm Hg)
# 4. Triceps skin fold thickness (mm)
# 5. 2-Hour serum insulin (mu U/ml)
# 6. Body mass index (weight in kg/(height in m)^2)
# 7. Diabetes pedigree function
# 8. Age (years)
# 9. Class variable (0 or 1)
```

Part A

```
bigx<-wdat[,-c(9)]
bigy<-wdat[,9]
trscore <- array(dim = 10)
tescore <- array(dim = 10)


#####################
#Part A
#############
for (wi in 1:10) {
  wtd <- createDataPartition(y = bigy, p = 0.8, list = FALSE)
  nbx <- bigx
  ntrbx <- nbx[wtd, ]
  ntrby <- bigy[wtd]

  trposflag <- ntrby > 0
  ptregs <- ntrbx[trposflag, ]
  ntregs <- ntrbx[!trposflag, ]

  ntebx <- nbx[-wtd, ]
  nteby <- bigy[-wtd]

  ptrmean <- sapply(ptregs, mean, na.rm = T)
  ntrmean <- sapply(ntregs, mean, na.rm = T)
  ptrsd   <- sapply(ptregs, sd, na.rm = T)
  ntrsd   <- sapply(ntregs, sd, na.rm = T)

  ptroffsets <- t(t(ntrbx) - ptrmean)
  ptrscales  <- t(t(ptroffsets) / ptrsd)
  ptrlogs    <- -(1/2) * rowSums(apply(ptrscales, c(1,2),
                function(x) x^2), na.rm = T) - sum(log(ptrsd))+log(NROW(ptregs)/NROW(ntr
by))
  ntroffsets <- t(t(ntrbx) - ntrmean)
  ntrscales  <- t(t(ntroffsets) / ntrsd)
  ntrlogs    <- -(1/2) * rowSums(apply(ntrscales, c(1,2)
                                       , function(x) x^2), na.rm = T) - sum(log(ntrsd))
+log(NROW(ntregs)/NROW(ntrby))


  lvwtr      <- ptrlogs > ntrlogs
  gotrighttr <- lvwtr == ntrby
  trscore[wi]<- sum(gotrighttr)/(sum(gotrighttr)+sum(!gotrighttr))

  pteoffsets <- t(t(ntebx)-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)) +log
(NROW(ptregs)/NROW(ntrby))

  nteoffsets <- t(t(ntebx)-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)) +log
(NROW(ntregs)/NROW(ntrby))

  lvwte<-ptelogs>ntelogs
  gotright<-lvwte==nteby
  tescore[wi]<-sum(gotright)/(sum(gotright)+sum(!gotright))  # Accuracy on the test set
}
print(tescore)
```

```
##  [1] 0.8039216 0.8235294 0.7516340 0.7516340 0.7581699 0.7385621 0.7254902
##  [8] 0.7712418 0.7124183 0.7516340
```

```
print(paste("Average Accuracy",mean(tescore)))
```

```
## [1] "Average Accuracy 0.758823529411765"
```

Part B

```r
wdat2 <-wdat
#replace 0 to NA
wdat2$V3[wdat2$V3 == 0] <- NA
wdat2$V4[wdat2$V4 == 0] <- NA
wdat2$V6[wdat2$V6 == 0] <- NA
wdat2$V8[wdat2$V8 == 0] <- NA

bigx<-wdat2[,-c(9)]
bigy<-wdat2[,9]
trscore <- array(dim = 10)
tescore <- array(dim = 10)


for (wi in 1:10) {
  wtd <- createDataPartition(y = bigy, p = 0.8, list = FALSE)
  nbx <- bigx
  ntrbx <- nbx[wtd, ]
  ntrby <- bigy[wtd]

  trposflag <- ntrby > 0
  ptregs <- ntrbx[trposflag, ]
  ntregs <- ntrbx[!trposflag, ]

  ntebx <- nbx[-wtd, ]
  nteby <- bigy[-wtd]

  ptrmean <- sapply(ptregs, mean, na.rm = T)
  ntrmean <- sapply(ntregs, mean, na.rm = T)
  ptrsd   <- sapply(ptregs, sd, na.rm = T)
  ntrsd   <- sapply(ntregs, sd, na.rm = T)

  ptroffsets <- t(t(ntrbx) - ptrmean)
  ptrscales  <- t(t(ptroffsets) / ptrsd)
  ptrlogs    <- -(1/2) * rowSums(apply(ptrscales, c(1,2),
                function(x) x^2), na.rm = T) - sum(log(ptrsd))+log(NROW(ptregs)/NROW(ntr
by))
  ntroffsets <- t(t(ntrbx) - ntrmean)
  ntrscales  <- t(t(ntroffsets) / ntrsd)
  ntrlogs    <- -(1/2) * rowSums(apply(ntrscales, c(1,2)
                                       , function(x) x^2), na.rm = T) - sum(log(ntrsd))
+log(NROW(ntregs)/NROW(ntrby))


  lvwtr      <- ptrlogs > ntrlogs
  gotrighttr <- lvwtr == ntrby
  trscore[wi]<- sum(gotrighttr)/(sum(gotrighttr)+sum(!gotrighttr))

  pteoffsets <- t(t(ntebx)-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)) +log
(NROW(ptregs)/NROW(ntrby))
```

```
  nteoffsets <- t(t(ntebx)-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)) +log
(NROW(ntregs)/NROW(ntrby))

  lvwte<-ptelogs>ntelogs
  gotright<-lvwte==nteby
  tescore[wi]<-sum(gotright)/(sum(gotright)+sum(!gotright))
}
print(tescore)
```

```
##  [1] 0.7647059 0.7385621 0.7712418 0.7254902 0.7647059 0.7908497 0.7647059
##  [8] 0.7647059 0.7058824 0.7973856
```

```
print(paste("Average Accuracy",mean(tescore)))
```

```
## [1] "Average Accuracy 0.758823529411765"
```

Part C

```
##Model with Klar
trainPartition <- createDataPartition(y = wdat[,9], p = 0.8, list = FALSE)
train <- wdat[trainPartition,]
test <-wdat[-trainPartition,]
model1 <- klaR::NaiveBayes(as.factor(V9)~.,data= train )
pred1 <- predict(object = model1, test[,-c(9)])
gotright = test[,9] ==pred1$class
accuracy =  sum(gotright)/(sum(gotright)+sum(!gotright))
print(paste("accuracy using klar", accuracy))
```

```
## [1] "accuracy using klar 0.745098039215686"
```

```
#Model with caret
model2 <- caret::train(x = train[,-c(9)], y= as.factor(train[,c(9)]),"nb", trControl=tra
inControl(method='cv',number=10)  )
pred2 <- predict(model2, test[,-c(9)])
gotright = test[,9] ==pred2
accuracy =  sum(gotright)/(sum(gotright)+sum(!gotright))
print(paste("accuracy using caret", accuracy))
```

```
## [1] "accuracy using caret 0.745098039215686"
```

Part D

```
svmModel <-  svmlight(V9 ~ ., data=train, pathsvm='../../svm_light_osx.8.4_i7/')
pred3 <- predict(svmModel, test[,-c(9)])
gotright = test[,9] ==pred3$class
accuracy =  sum(gotright)/(sum(gotright)+sum(!gotright))
print(paste("SVM accuracy", accuracy))
```

```
## [1] "SVM accuracy 0.712418300653595"
```