Linear Classification

Daniel Ferreira Zanchetta y Lais Silva Almeida Zanchetta

26/02/2020

Operaciones abajo son para el pre-processing de los datos:

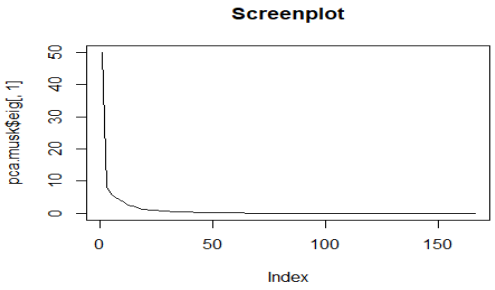
Para el pre-processing de los datos en la Musk.data, hemos realizado los siguientes procedimientos: Asignamos nombres a las columnas, tal cual estaba definido en Musk2.info; Quitamos del dataset los dos primeros atributos, tal y como decía en la Musk2.info para no ser utilizados para predecir class; Pasamos el atributo "class" a factor y cambiar las labels de 0 a "No" (para No Musk) y de 1 a "Yes" para el caso de Musk; Quitamos un total de 17 filas duplicadas; Hemos estandarizado los valores numericos, pues no estabamos seguros de que todos teniam la misma unidad; Aplicamos la función de PCA, y nos fijamos que no hay "codos" que nos digan con certidumbre cuantos componentes significativas hay. Se nota que hay un pequeño codo convexo a un valor aproximado a 15 pero no se podría decir con toda certeza que esta seria la cantidad de componentes significativos.

Por lo tanto, definimos el numero de componentes significativas igual a 25 pues acumula 90,08% de dimensiones significativas.

Obs.: No obstante, aunque la analisis de PCA ha ayudado a identificar el numero de dimensiones significativas, pero nos estaba ocasionando errores que no fuimos capaces de solucionar. Por otro lado, como bien intenta expresar los resultados finales, suponemos haber conseguido llegar a conclusiones interesantes con los algoritmos.

```
library(dplyr)

library(tidyr)
Musk <- read.table("Musk2.data", sep = ",") %>% select(-c(V1,V2))
head(Musk,5)
colnames(Musk) <-
c('f1','f2','f3','f4','f5','f6','f7','f8','f9','f10','f11','f12','f13','f
14','f15','f16','f17','f18','f19','f20','f21','f22','f23','f24','f25','f2
6','f27','f28','f29','f30','f31','f32','f33','f34','f35','f36','f37','f38
','f39','f40','f41','f42','f43','f44','f45','f46','f47','f48','f49','f50'
,'f51','f52','f53','f54','f55','f56','f57','f58','f59','f60','f61','f62',
'f63','f64','f65','f66','f67','f68','f69','f70','f71','f72','f73','f74','
f75','f76','f77','f78','f79','f80','f81','f82','f83','f84','f85','f86','f
87','f88','f89','f90','f91','f92','f93','f94','f95','f96','f97','f98','f9
9','f100','f101','f102','f103','f104','f105','f106','f107','f108','f109',
'f110','f111','f112','f113','f114','f115','f116','f117','f118','f119','f1
20','f121','f122','f123','f124','f125','f126','f127','f128','f129','f130'
,'f131','f132','f133','f134','f135','f136','f137','f138','f139','f140','f
141','f142','f143','f144','f145','f146','f147','f148','f149','f150','f151
','f152','f153','f154','f155','f156','f157','f158','f159','f160','f161','
f162','f163','f164','f165','f166', 'class')

Musk$class <- factor(Musk$class, labels=c("No", "Yes"))

Musk <- Musk[!duplicated(Musk),]
```

```
funumeric <- function(value){
  if(is.numeric(value)){as.vector(scale(value))} else{value}}
Musk <- data.frame(lapply(Musk,funumeric))
summary(Musk)

library(FactoMineR)
pca.musk <- PCA(Musk,ind.sup = c(1:1017),quali.sup = 167)

plot(pca.musk$eig[,1],type="l",main="Screenplot")
```




```
pca.musk$var$cor

nd = 25
pca.musk.sign <- PCA(Musk, ncp=nd,
quali.sup=167, scale.unit = FALSE)

pca.musk <-
```

```
(cbind(as.data.frame(pca.musk.sign$ind$coord),Musk[167]))
```

Operaciones para aplicar los algoritmos de clasificación linear:

En primer lugar, hemos utilizado, tal y como recomendado, los algoritmos de Logistic Regression, LDA y QDA, intentando buscar entre los tres lo que tenia la más pequeña Training Error y Test Error. Para llegar a los resultados de comparacion entre los tres metodos, primero hemos dividido el dataset de Musk entre Training y Test, dejando aproximadamente el 66% para Training.

Logistic Regression

Para la parte del metodo de logistic regression, primero hemos aplicado el GLM (generalized linear models), con lo cual hemos descubierto que algunas variables, como f1, f17, f32, f33, pueden no ser estadisticamente interesantes, sin embargo no las hemos ignorado de momento. Enseguida, intentamos simplificar un poco el modelo quitando las variables no significativas. Utilizamos función de step(). Pero, por tardar mucho en ejecutarse, al final no lo hemos considerado y utilizamos el resultado de la función GLM.

A través de la función musk.accs (con la cual usa por defecto utiliza predicción probabilistica al 50%), hemos intentado hacer predicciones probabilisticas para los datos de training y los datos de test. Vemos un resultado de error en training de 4.12% y de error en test de 4.23%.

LDA

En la aplicación del LDA, vemos un error en training del 5,47% y error en test de 5,28%, que, en estos momentos, ha sido peor que lo de la logistic regression. Mismo utilizando Cross Validation Leave-one-out (CV=TRUE), obtenemos error en training de 5.97% y en test de 5,78%.

QDA

En la aplicación del QDA, vemos un error en training de 1,13% y error en test de 3.50%. En este caso entendemos que este metodo es el mejor en el caso de predicir si un nuevo dato es Musk o No Musk. Cuando hemos realizado la aplicación del mismo metodo con Cross Validation Leave-one-out (CV=TRUE), obtenemos error en training

de 3.16% y en test de 5,37% que, aun que sean mas grandes que sin CV leave-one-out, nos han parecido resultados mas interesantes que en los demas metodos.

```
set.seed(17)

N <- nrow(Musk)
learn <- sample(1:N, round(2*N/3))

nlearn <- length(learn)
ntest <- N - nlearn
# Logistic Regression.
musk.glm <- glm (class ~ ., data = Musk[learn,], family = "binomial")
summary(musk.glm)

#musk.glm.step <- step(musk.glm)

musk.accs <- function (P=0.5)
{
  muskM1.pred <- NULL
  muskM1.pred[musk.glm$fitted.values<P] <- 0
  muskM1.pred[musk.glm$fitted.values>=P] <- 1

  muskM1.pred <- factor(muskM1.pred, labels=c("No Musk","Musk"))

  cat("TRAINING ERROR:")
  print(M1.TRtable <- table(Truth=Musk[learn,]$class,Pred=muskM1.pred))

  print(100*(1-sum(diag(M1.TRtable))/nlearn))

  muskM1test <- predict(musk.glm, newdata=Musk[-learn,],type="response")
  muskM1.pred.test <- NULL
  muskM1.pred.test[muskM1test<P] <- 0
  muskM1.pred.test[muskM1test>=P] <- 1
  muskM1.pred.test <- factor(muskM1.pred.test, labels=c("No
Musk","Musk"))

  cat("TESTING ERROR:")
  print(M1.TEtable <- table(Truth=Musk[-
learn,]$class,Pred=muskM1.pred.test))

  print(100*(1-sum(diag(M1.TEtable))/ntest))
}

musk.accs()

## TRAINING ERROR:## [1] 4.125826
## TESTING ERROR: ## [1] 4.238833
library(MASS)

# 1.Aplicación del LDA
musk.lda.learn <- lda(class ~ ., Musk[learn,], CV = FALSE)
musk.lda.train <- predict(musk.lda.learn,newdata=Musk[learn,])

#Calcular los errores de LDA y las predicciones en training y test data

## LDA TRAINING ERROR: ## [1] 5.470709
```

```r
(tab <- table(Truth=Musk[learn,]$class, Pred=musk.lda.train$class))
(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

musk.lda.test <- predict(musk.lda.learn, newdata=Musk[-learn,], type =
"response")

## TESTING ERROR: [1] 5.287147

(tab <- table(Truth=Musk[-learn,]$class, Pred=musk.lda.test$class))

 (error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

#Leave-one-out
musk.lda.learn.loo <- lda(class ~ ., Musk[learn,], CV = TRUE)
(tab <- table(Truth=Musk[learn,]$class, Pred=musk.lda.learn.loo$class))

(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

## [1] 5.972191

musk.lda.test.loo <- lda(class ~ ., Musk[-learn,], CV = TRUE)
(tab <- table(Truth=Musk[-learn,]$class, Pred=musk.lda.test.loo$class))

(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

## [1] 5.788514

# 2.Quadratic Discriminant Analysis

musk.qda.learn <- qda (class ~ ., Musk[learn,], CV = FALSE)
musk.qda.train <- predict(musk.qda.learn)

#Calcular los errores de QDA y las predicciones en training y test data
musk.qda.train <- predict(musk.qda.learn,newdata=Musk[learn,])

## QDA TRAINING ERROR: ## [1] 1.139731

(tab <- table(Truth=Musk[learn,]$class, Pred=musk.qda.train$class))

(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

musk.qda.test <- predict(musk.qda.learn, newdata=Musk[-learn,], type =
"response")

## QDA TEST ERROR: ## [1] 3.509572

(tab <- table(Truth=Musk[-learn,]$class, Pred=musk.qda.test$class))
(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

#Leave-one-out
musk.qda.learn.loo <- qda(class ~ ., Musk[learn,], CV = TRUE)
(tab <- table(Truth=Musk[learn,]$class, Pred=musk.qda.learn.loo$class))

(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

## [1] 3.168452

musk.qda.test.loo <- qda(class ~ ., Musk[-learn,], CV = TRUE)
(tab <- table(Truth=Musk[-learn,]$class, Pred=musk.qda.test.loo$class))

(error.LOOCV <- 100*(1-sum(tab[row(tab)==col(tab)])/sum(tab)))

## [1] 5.378304
```