INTRODUCCION AL ANALISIS MULTIVARIADO

Lab. No.5 - KNN - SVM

Se quiere predecir el tipo de cliente (bueno o malo) usando las siguientes variables que se analizan antes de otorgar un crédito:

- edad: edad del cliente en años cumplidos.
- antig_laboral: antiguedad laboral del cliente en años.
- vivienda: tipo de vivienda (propia, padres, alquilada, contrato privado y otros).
- estado_civil: estado de civil del cliente (soltero, casado, separado, divorciado y viudo).
- trabajo: tipo de trabajo del cliente (asalariado, independiente, temporal y otros).
- ingreso: salario mensual del cliente.
- gasto: gasto mensual del cliente.
- deuda: deuda mensual del cliente.
- ahorro: deuda mensual del cliente.
- patrimonio: valor del patrimonio de cliente.
- porc_deuda: $\frac{deuda}{ingreso} * 100$
- porc_ahorro: $\frac{ahorro}{ingreso} * 100$
- porc_gasto: $\frac{gasto}{ingreso} * 100$
- sobreendeudado: 1 si el cliente esta sobreendeudado y 0 si no.
- plazo: plazo del préstamo solicitado.
- monto: monto del préstamo solicitado.
- garantia: valor de la garantia.
- montoGarantia: $\frac{monto}{garantia} * 100$

K VECINOS MAS CERCANOS

1. Cargue los paquetes `caret` , `DT` , `ROCR` , `class` , `kknn` , `e1071` , `adabag` y `randomForest` .

```
suppressMessages(library(caret))
suppressMessages(library(DT))
suppressMessages(library(ROCR))
suppressMessages(library(class))
suppressMessages(library(kknn))
suppressMessages(library(e1071))
suppressMessages(library(plotly))
suppressMessages(library(adabag))
suppressMessages(library(randomForest))
```

2. Procedimiento manual:

a. Use los primeros 50 datos de base2 para entrenamiento y los siguientes 10 para validación. Forme las dos bases.

```
load("Credit.Rdata")
train=base2[1:50,-(3:6)]
test=base2[51:60,-(3:6)]
```

b. Haga la clasificación manualmente usando 3 vecinos más cercanos de los 10 datos de validación, usando las distancias a los 50 datos de entrenamiento. Use la distancia euclídea.

```
pred=c()
for(i in 1:10){
  d=as.matrix(dist(rbind(train[,-1],test[i,-1])))[51,-51]
  o=order(d)
  clas=train[o,][1:3,1]
  tab=table(clas)
  pred[i]= names(tab)[which(tab==max(tab))]
}
pred
```

```
##  [1] "malo"  "bueno" "bueno" "malo"  "malo"  "malo"  "bueno" "bueno" "bueno"
## [10] "bueno"
```

c. Use la función `knn` de la librería `class` de la siguiente forma `knn(train,test,clase,k = 3)`, donde `train` son los datos de entrenamiento (sin la variable objetivo), `test` son los datos de validación y `clase` es la variable objetivo de la base de entrenamiento.

```
pred2=knn(train[,-1],test[,-1],train$cliente,k = 3)
```

d. Verifique que la clasificación manual coincide con la realizada con la función `knn`.

```
table(pred,pred2)
```

```
##        pred2
## pred    bueno malo
##   bueno     6    0
##   malo      0    4
```

3. Validación entrenamiento/prueba:

a. Escoja solo las variables numéricas de base2 y estandarícelas. Luego pegue la variable objetivo y llame a esta base3.

```
base3=data.frame(scale(base2[,-c(1,3:6)]))
base3$cliente=base2$cliente
names(base3)
```

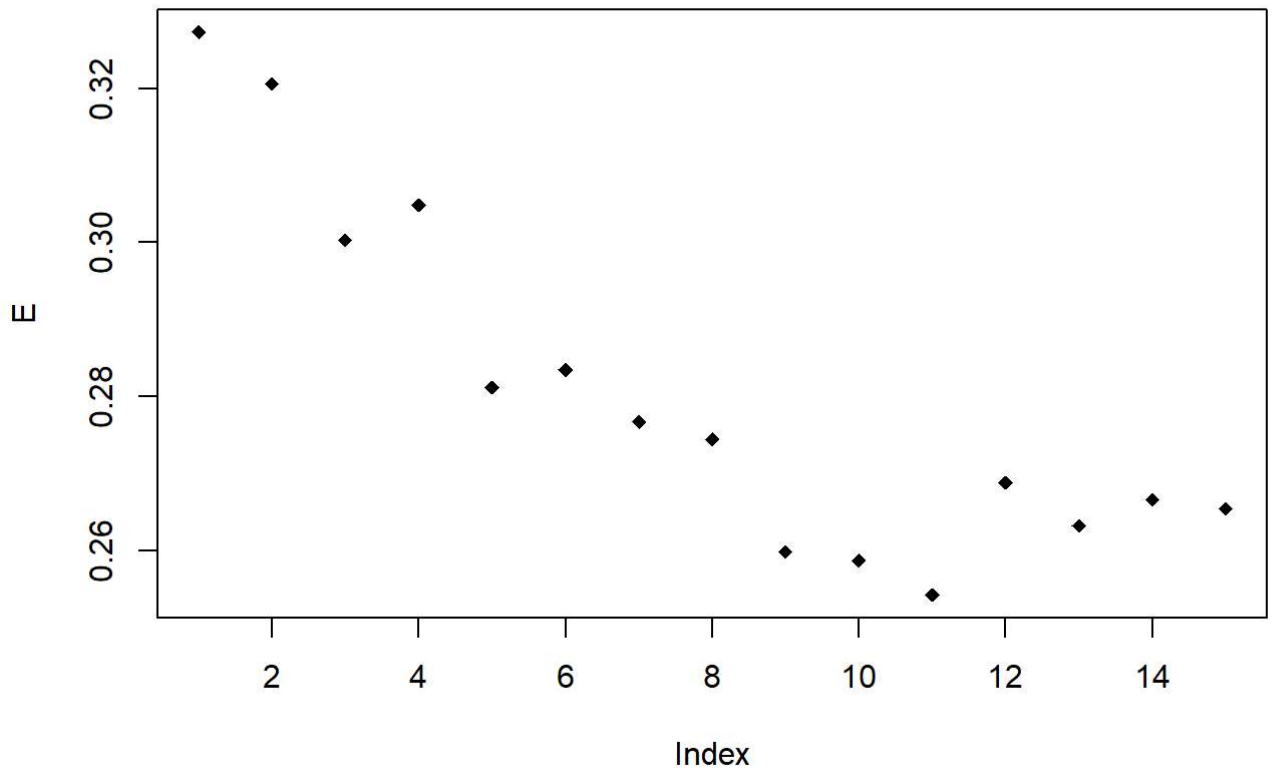
```
##  [1] "edad"             "ingreso"          "gasto"            "deuda"
##  [5] "ahorro"           "patrimonio"       "porc_deuda"       "porc_ahorro"
##  [9] "porc_gasto"       "sobreendeudado"   "plazo"            "monto"
## [13] "garantia"         "monto_financiado" "cliente"
```

b. Haga una base de entrenamiento con el 80% de los datos y una base de prueba con el resto.

```
set.seed(10)
n=nrow(base3)
s=sample(1:n,round(0.8*n))
train=base3[s,]
test=base3[-s,]
```

c. Lleve a cabo la clasificación usando diferentes cantidades de vecinos y almacene el error de clasificación. Decida un número adecuado de vecinos.

```
E=c()
for(k in 1:15){
  pred = knn(train[,-15],test[,-15],train$cliente,k = k)
  confu= table(test$cliente,pred)
  error= 1-sum(diag(confu))/sum(confu)
  E[k]=error
}
plot(E,pch=18)
```



Entre 9 y 11 vecinos.

MAQUINAS VECTORIALES DE SOPORTE

4. Usando la misma base de entrenamiento anterior se realizará un SVM para predecir el tipo de cliente usando la función de Kernel Radial para la base de validación o prueba.

a. Haga un modelo clasificación usando la función `svm` de la librería `e1071`. Indique `kernel = "radial"` y `probability = TRUE`. Use la base de entrenamiento.

```
mod1=svm(cliente ~ ., kernel = "radial",
         probability = TRUE, data=train)
```

b. Haga la tabla de confusión con la base de validación

```
pred1=predict(mod1,newdata=test)
table(test$cliente,pred1)
```

```
##         pred1
##          bueno malo
##    bueno   637   23
##    malo    194   35
```

5. Repita los pasos anteriores:

a. Prediga el incumplimiento en la base de validación usando el kernel sigmoidal (`kernel = "sigmoid"`) con la base de entrenamiento.

```
mod2=svm(cliente ~ ., kernel = "sigmoid",
         probability = TRUE, data=train)
pred2=predict(mod2,newdata=test)
table(test$cliente,pred2)
```

```
##         pred2
##          bueno malo
##   bueno   509  151
##   malo    147   82
```

b. Repita el ejercicio usando el kernel lineal (`kernel = "linear"`).

```
mod3=svm(cliente ~ ., kernel = "linear",
         probability = TRUE, data=train)
pred3=predict(mod3,newdata=test)
table(test$cliente,pred3)
```

```
##         pred3
##          bueno malo
##   bueno   633   27
##   malo    197   32
```

6. Vea el summary de cada modelo y observe cuantos vectores de soporte se requirieron para hacer la función discriminante en cada caso.

```
summary(mod1)
```

```
##
## Call:
## svm(formula = cliente ~ ., data = train, kernel = "radial", probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  2094
##
##  ( 1110 984 )
##
##
## Number of Classes:  2
##
## Levels:
##  bueno malo
```

```
summary(mod2)
```

```
##
## Call:
## svm(formula = cliente ~ ., data = train, kernel = "sigmoid", probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  1
##      coef.0:  0
##
## Number of Support Vectors:  1421
##
##  ( 710 711 )
##
##
## Number of Classes:  2
##
## Levels:
##  bueno malo
```

```
summary(mod3)
```

```
##
## Call:
## svm(formula = cliente ~ ., data = train, kernel = "linear", probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  2153
##
##  ( 1151 1002 )
##
##
## Number of Classes:  2
##
## Levels:
##  bueno malo
```

- Obtenga la suma de la diagonal de cada matriz de confusión para determinar cuántos casos fueron bien clasificados en cada caso. ¿En cuál de ellos se hizo una mejor clasificación?

```
sum(diag(table(test$cliente,pred1)))
```

```
## [1] 672
```

```
sum(diag(table(test$cliente,pred2)))
```

```
## [1] 591
```

```
sum(diag(table(test$cliente,pred3)))
```

```
## [1] 665
```

7. Compare los resultados con los obtenidos con otros métodos:

a. Compárelo con el método de 9 vecinos más cercanos.

```
pred9=knn(train[,-15],test[,-15],train$cliente,k = 9)
sum(diag(table(test$cliente,pred9)))
```

```
## [1] 658
```

b. Compárelo con una regresión logística.

```
mod4=glm(cliente ~ ., family=binomial, data=train)
pred4=predict(mod4,newdata=test,type="response")>0.5
sum(diag(table(test$cliente,pred4)))
```

```
## [1] 677
```

c. Verifique si hay se puede asumir que las matrices de covarianza de los dos grupos son iguales.

```
library(biotools)
```

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

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:plotly':
##
##     select
```

```
## ---
## biotools version 4.2
```

```
boxM(train[,-15],train$cliente)
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  train[, -15]
## Chi-Sq (approx.) = 4283.1, df = 105, p-value < 2.2e-16
```

d. Obtenga la clasificación de los datos de validación con el análisis discriminante adecuado y compárela con los otros métodos. Se usa “lda” si es lineal y “qda” si es cuadrático.

Se debe usar el cuadrático porque las matrices de covarianza son diferentes.

```
library(MASS)
mod5=qda(cliente ~ ., prior=c(0.5,0.5), data=train)
pred5=predict(mod5,newdata=test)$class
sum(diag(table(test$cliente,pred5)))
```

```
## [1] 667
```