Trabajo PIBMadrid - Técnicas de Clasificación

Daniel Corral, Antonio Pascual, Diego Senso

03/12/2020

Contents

Objetivo

El objetivo del presente informe es estudiar el PIB per cápita de los diferentes municipios de Madrid, convirtiéndo este valor en una variable categórica (mediante la discretización por cuartiles) y tratando de clasificarla según diferentes modelos a estimar.

En primer lugar, se cargan las librerías necesarias para el estudio:

Carga y tratamiento del dataset

Se procede a cargar el dataset y se elimina la columna de "municipios", pues no será relevante para el estudio posterior.

```
#Carga del dataset
pibmadrid <- read.csv("pibmadrid.csv", sep = ";")

str(pibmadrid)
```

```
## 'data.frame':    179 obs. of  16 variables:
##  $ municipio        : chr  "Acebeda (La)" "Ajalvir" "Alameda del Valle" "Alamo (El)" ...
##  $ pib_percapita    : num  53662 62255 22436 12812 26805 ...
##  $ empadronados     : num  67 4.26 248 8.85 204.82 ...
##  $ paro             : num  11.94 7.53 6.05 11.32 10.66 ...
##  $ afiliados        : num  227 915 187 163 252 ...
##  $ declaraciones    : num  23 1.91 102 3.57 88.62 ...
##  $ catastro         : num  23.4 82.8 45.3 39.6 125.5 ...
##  $ turismos         : num  500 2232 484 437 459 ...
##  $ distancia_capital: num  88 27 91 38 31 15 13 49 30 43 ...
##  $ agricultura      : num  10.53 0.2 10.71 1.25 0.34 ...
##  $ energia          : num  0 38.87 0 9.42 14.4 ...
##  $ construccion     : num  5.26 13.61 3.57 14.59 8 ...
##  $ hosteleria       : num  26.3 21.5 57.1 28.8 32.1 ...
##  $ finanzas         : num  15.79 14.78 8.93 10.5 17.27 ...
##  $ otros            : num  42.1 12.4 19.6 40 30.3 ...
##  $ natalidad        : num  15.38 13.32 12.35 9.88 9.11 ...
```

```
#Eliminación de la columna municipios
pibmadrid <- pibmadrid[,-1]
```

Diccionario de variables

El dataset con el que se va a trabajar ha sido creado desde un inicio acudiendo a datos publicados por el Instituto de Estadística de la Comunidad de Madrid. Para conformar los datos, se ha seleccionado la variable a explicar (el PIB per cápita de cada municipio de la Comunidad de Madrid) y una serie de variables que se ha considerado que podrían ser importantes para estudiar su evolución.

Así pues, el dataset "pibmadrid" cuenta con un total de 179 observaciones en las que cada una de ellas representa una combinación de datos para ese municipio acerca de diferentes cuestiones.

Las variables con las que cuenta el dataset son las siguientes:

- **municipio**: el nombre del municipio en cuestión. Se ha eliminado esta variable al inicio pues no va a ser relevante para el análisis.
- **pib_percapita**: el dato del PIB per cápita de cada municipio de la Comunidad de Madrid, expresado en euros.
- **empadronados**: número de personas empadronadas en el municipio en cuestión.
- **paro**: número de parados del municipio por cada 1000 habitantes.

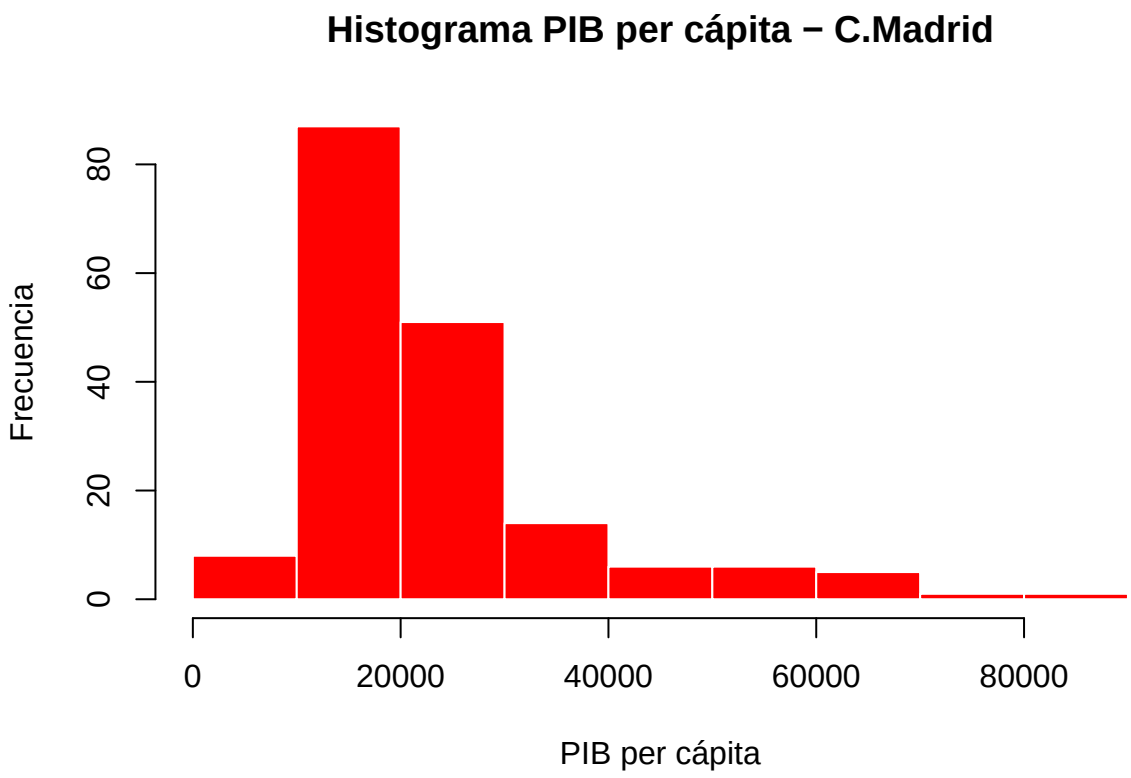
- **afiliados**: número de afiliados a la Seguridad Social por cada 1000 habitantes.
- **declaraciones**: número de delcaraciones de la renta producidas en ese municipio.

- **catastro**: valor catastral por unidad urbana. Se calcula con la división del valor catastral urbano entre las unidades urbanas.
- **turismos**: número de vehículos que poseen los habitantes de ese municipio por cada 1000 habitantes.
- **distancia_capital**: kilómetros de distancia entre el municipio y la ciudad de Madrid.
- **agricultura**: porcentaje de personas del municipio dedicados al sector de la agricultura del total de ocupados.
- **energia**: porcentaje de personas del municipio dedicados al sector de la energía del total de ocupados.
- **construccion**: : porcentaje de personas del municipio dedicados al sector de la construcción del total de ocupados.
- **hosteleria**: porcentaje de personas del municipio dedicados al sector de la hostelería del total de ocupados.
- **finanzas**: porcentaje de personas del municipio dedicados al sector de las finanzas del total de ocupados.
- **otros**: porcentaje de personas del municipio dedicados a otros sectores del total de ocupados.
- **natalidad**: nacimientos por cada 1000 habitantes.

Observación del dataset (y de la variable a predecir)

Realizamos una primera aproximación al dataset mediante la función "summary" para ganar una idea de cuál es el comportamiento de las diferentes variables. Además observamos gráficamente la variable a estudiar mediante un histograma.

```
##  pib_percapita    empadronados          paro          afiliados
##  Min.   : 7334   Min.   :      1   Min.   : 0.000   Min.   : 63.49
##  1st Qu.:14234   1st Qu.:      4   1st Qu.: 7.820   1st Qu.:138.41
##  Median :19288   Median :     13   Median : 9.740   Median :178.06
##  Mean   :23176   Mean   :  18042   Mean   : 9.727   Mean   :223.75
##  3rd Qu.:27492   3rd Qu.:    125   3rd Qu.:11.620   3rd Qu.:242.40
##  Max.   :83698   Max.   :3207247   Max.   :16.470   Max.   :914.93
##  declaraciones          catastro         turismos       distancia_capital
##  Min.   :      1.0   Min.   : 18.06   Min.   :  336.7   Min.   :  0.00
##  1st Qu.:      3.6   1st Qu.: 40.20   1st Qu.:  435.7   1st Qu.: 32.00
##  Median :     27.2   Median : 76.32   Median :  479.5   Median : 47.00
##  Mean   :   9212.0   Mean   : 86.11   Mean   :  977.4   Mean   : 49.05
##  3rd Qu.:    241.5   3rd Qu.:125.40   3rd Qu.:  529.3   3rd Qu.: 62.00
##  Max.   :1618472.0   Max.   :236.26   Max.   :19731.7   Max.   :105.00
##   agricultura        energia       construccion      hosteleria
##  Min.   : 0.000   Min.   : 0.00   Min.   : 0.000   Min.   : 3.77
##  1st Qu.: 0.510   1st Qu.: 2.69   1st Qu.: 7.615   1st Qu.:21.84
##  Median : 1.490   Median : 6.20   Median :11.110   Median :27.07
##  Mean   : 3.458   Mean   :10.11   Mean   :12.118   Mean   :27.60
##  3rd Qu.: 5.030   3rd Qu.:15.04   3rd Qu.:14.300   3rd Qu.:34.08
##  Max.   :26.770   Max.   :64.15   Max.   :42.310   Max.   :69.23
##     finanzas          otros         natalidad
##  Min.   : 0.000   Min.   : 8.68   Min.   : 0.000
##  1st Qu.: 8.645   1st Qu.:24.93   1st Qu.: 8.375
##  Median :13.450   Median :33.37   Median :10.460
##  Mean   :14.556   Mean   :34.92   Mean   :10.223
##  3rd Qu.:18.865   3rd Qu.:42.87   3rd Qu.:12.765
##  Max.   :50.980   Max.   :79.45   Max.   :26.440
```



Como se puede observar, el PIB per cápita más repetido se encuentra entre los 10.000 y los 20.000, seguido del intervalo entre 20.000 y 30.000.

Discretización del PIB per cápita por cuartiles

Para estudiar la variable y poder clasificarla con algunos de los modelos, se sacan los cuartiles de la variable "pib_percapita". Según esos valores se creará una nueva columna ("pib_categ") con las categorías de 1 a 4.

```
#Cuantiles
quantile(pibmadrid$pib_percapita)
```

```
##      0%     25%     50%     75%    100%
##  7334.0 14234.5 19288.0 27492.0 83698.0
```

```
#Creación de nueva columna y cambio de números de categorías
pibmadrid[,'pib_categ'] <- cut(pibmadrid$pib_percapita, breaks = c(7333.9, 14234.5, 19288.0, 27492.0, 83
```

Creación del set de entrenamiento y test

Procedemos a crear el set de entrenamiento y el de test para ver la calidad de los modelos que posteriormente configuraremos. Seleccionamos que la parte de entrenamiento será del 80%, mientras que la del test un 20% del total de la muestra.

```r
set.seed(123)
entrenamiento <- sample(x = nrow(pibmadrid), size = nrow(pibmadrid)*0.8, replace = FALSE)

# Subgrupo de datos de entrenamiento
train <- pibmadrid[entrenamiento,]

# Subgrupo de datos de test
test <- pibmadrid[-entrenamiento,]
```

Modelo de regresión lineal

Pasando a la configuración de modelos, en primer lugar se ha decidido realizar un simple modelo de regresión lineal con el fin de observar cuáles de las variables son buenas para explicar la Y (aún en euros).

```r
modelo_lineal <- lm(pib_percapita ~. -pib_categ, data = pibmadrid)
summary(modelo_lineal)
```

```
##
## Call:
## lm(formula = pib_percapita ~ . - pib_categ, data = pibmadrid)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35480  -4519   -773   3607  46130
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -2.546e+04  1.508e+04  -1.688   0.0932 .
## empadronados       2.053e+00  1.356e+00   1.515   0.1318
## paro              -3.971e+02  3.213e+02  -1.236   0.2183
## afiliados          5.362e+01  6.059e+00   8.850 1.35e-15 ***
## declaraciones     -4.062e+00  2.687e+00  -1.512   0.1325
## catastro          -1.780e+01  2.061e+01  -0.864   0.3891
## turismos           5.110e-01  2.926e-01   1.746   0.0826 .
## distancia_capital  3.114e+02  6.038e+01   5.157 7.15e-07 ***
## agricultura        2.800e+02  2.603e+02   1.076   0.2837
## energia            3.839e+02  1.492e+02   2.574   0.0110 *
## construccion       1.547e+02  1.642e+02   0.942   0.3476
## hosteleria         2.349e+02  1.365e+02   1.721   0.0872 .
## finanzas           6.515e+02  1.354e+02   4.812 3.37e-06 ***
## otros              2.213e+02  1.533e+02   1.443   0.1509
## natalidad         -3.703e+02  1.684e+02  -2.199   0.0293 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9033 on 164 degrees of freedom
## Multiple R-squared:  0.5785, Adjusted R-squared:  0.5425
## F-statistic: 16.08 on 14 and 164 DF,  p-value: < 2.2e-16
```

A continuación y habiendo observado los resultados, se contrasta con stepAIC la mejor combinación de variables:

```
stepAIC(modelo_lineal, direction = "both")
```

Creamos un modelo con la configuración derivada por el contraste stepAIC.

```
modelo_lineal2 <- lm(pib_percapita ~ empadronados + afiliados + declaraciones +
    turismos + distancia_capital + energia + finanzas + natalidad, data = train)
summary(modelo_lineal2)
```

```
##
## Call:
## lm(formula = pib_percapita ~ empadronados + afiliados + declaraciones +
##     turismos + distancia_capital + energia + finanzas + natalidad,
##     data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28799  -4091   -539   3659  45528
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -1.536e+04  5.101e+03  -3.011  0.00312 **
## empadronados       3.105e+00  1.337e+00   2.323  0.02171 *
## afiliados          4.490e+01  6.872e+00   6.534 1.23e-09 ***
## declaraciones     -6.145e+00  2.649e+00  -2.320  0.02186 *
## turismos           6.978e-01  4.289e-01   1.627  0.10610
## distancia_capital  4.052e+02  5.463e+01   7.417 1.23e-11 ***
## energia            1.465e+02  7.996e+01   1.832  0.06917 .
## finanzas           7.041e+02  1.265e+02   5.567 1.36e-07 ***
## natalidad         -3.249e+02  1.866e+02  -1.741  0.08392 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9135 on 134 degrees of freedom
## Multiple R-squared:  0.5125, Adjusted R-squared:  0.4834
## F-statistic: 17.61 on 8 and 134 DF,  p-value: < 2.2e-16
```

No existen unas grandes diferencias. El R ajustado asciende aunque no es un aumento demasiado significativo. Por otro lado, las variables que antes tenían calidad explicativa parecen conservarla.

Modelo LDA

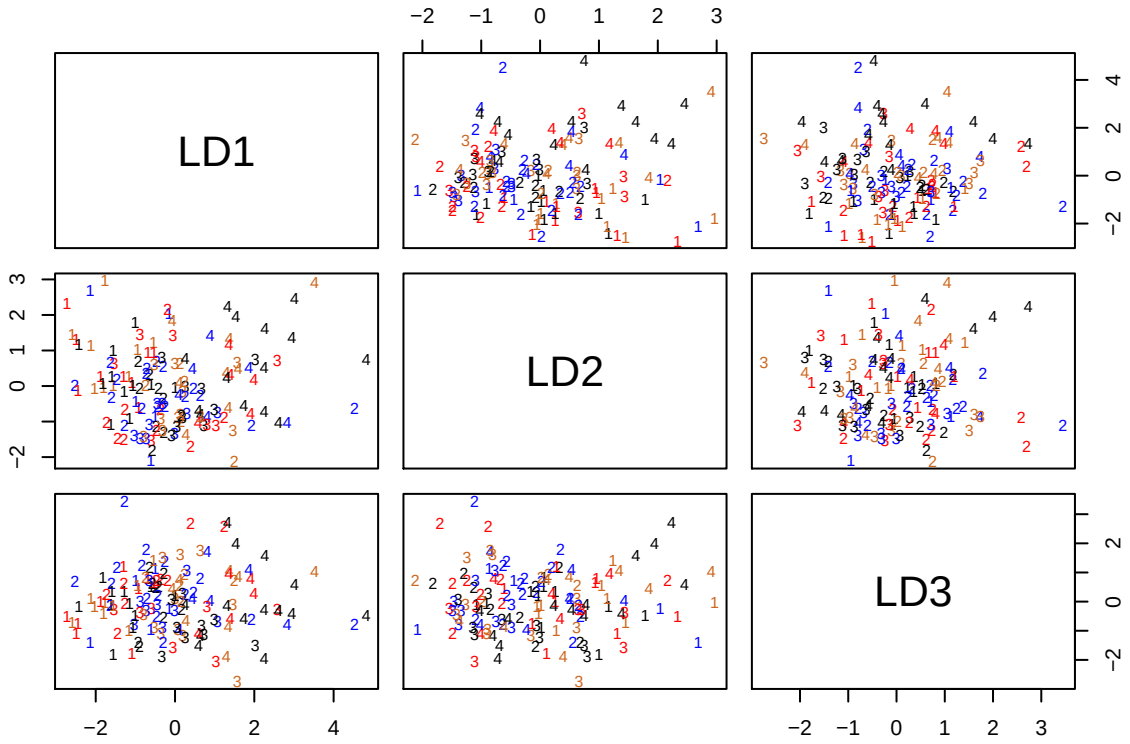
A partir de aquí, trabajamos ya con el PIB per cápita en formato categórico (valores de 1 a 4). Para predecir esta variable acudimos al modelo LDA. Se ha escogido introducir todas las variables iniciales pues pese a que el stepAIC sugería una configuración diferente, la mejoría del modelo era poco reseñable y se han incluido todas las variables para evitar acometer así una posible pérdida de información.

```
#Estimación del modelo
modelLDA <- lda(pib_categ ~. -pib_percapita, data=train)
modelLDA
```

```
## Call:
```

```
## lda(pib_categ ~ . - pib_percapita, data = train)
##
## Prior probabilities of groups:
##         1         2         3         4
## 0.2377622 0.2797203 0.2517483 0.2307692
##
## Group means:
##   empadronados      paro afiliados declaraciones catastro  turismos
## 1     115.5268 10.915588  138.1321      347.6171 69.84794  473.9056
## 2     123.1835 10.242000  190.4582      173.5958 86.39650  786.5165
## 3     160.6036  9.123611  241.4125      157.6456 98.17917  741.2883
## 4   97311.2467  8.330909  300.2561    49149.6839 83.06758 1386.2542
##   distancia_capital agricultura   energia construccion hosteleria finanzas
## 1          47.14706    5.730000  7.929412     15.15000   23.71412 13.23941
## 2          50.12500    2.957250  8.066000     13.10850   27.45550 13.01175
## 3          44.33333    2.470556 13.462222     11.70167   27.83194 13.94306
## 4          56.84848    3.519394  9.972121     10.33576   28.59273 16.99606
##      otros natalidad
## 1 36.64500 12.089118
## 2 37.45625  9.574250
## 3 32.62583  9.399167
## 4 35.46515  8.758182
##
## Coefficients of linear discriminants:
##                               LD1           LD2           LD3
## empadronados       0.0004276382 -5.464173e-04  8.666982e-04
## paro              -0.0645327442 -9.140791e-02  3.546034e-01
## afiliados          0.0068381377  1.625970e-03 -1.375978e-03
## declaraciones     -0.0008460434  1.084272e-03 -1.716294e-03
## catastro          -0.0009004753 -6.122137e-03  9.711145e-03
## turismos           0.0001630649 -1.963481e-05  2.690378e-05
## distancia_capital  0.0529867242  1.664168e-02  5.704172e-02
## agricultura       -0.0338723169  2.082253e-01 -1.397719e-01
## energia            0.0486271235  4.360182e-02 -8.267940e-02
## construccion      -0.0108726443  4.767543e-02 -1.473184e-03
## hosteleria         0.0284794338  1.675924e-02 -3.086119e-03
## finanzas           0.0641043796  1.074105e-01  3.611032e-02
## otros              0.0076934540  4.783863e-02 -9.459475e-03
## natalidad         -0.0407711764  8.529812e-02  4.699193e-02
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.7613 0.1523 0.0864
```

```
##Gráfico
plot(modelLDA, pch=16, col = c("red","blue", "chocolate3", "black")[pibmadrid$pib_categ])
```




```
# Prediccion respuesta
ldaResult <- predict(modelLDA, newdata = test)

# Matriz de confusion
tldamod <- table(ldaResult$class, test$pib_categ)
tldamod
```

```
##
##      1 2 3 4
##    1 3 0 1 0
##    2 7 4 3 2
##    3 1 0 2 4
##    4 0 1 2 6
```

```
# Precision
sum(diag(tldamod))/sum(tldamod)*100 #Precisión
```

```
## [1] 41.66667
```

```
# Gráficos de partición (por separado para reducir tiempos de computación)
#library(klaR)
#partimat(pibmadrid[,1:5],pibmadrid$pib_categ,data=pibmadrid,method="lda",main="Gráficos de partición")
#partimat(pibmadrid[,6:9],pibmadrid$pib_categ,data=pibmadrid,method="lda",main="Gráficos de partición")
#partimat(pibmadrid[,10:15],pibmadrid$pib_categ,data=pibmadrid,method="lda",main="Gráficos de partición
```

A juzgar por los resultados, la precisión del modelo es reducida. No logra clasificar correctamente muchas de las observaciones dentro de cada uno de los cuatro grupos.

Modelo QDA

La misma variable categórica la pasamos a predecir con el modelo QDA.

```
modelQDA <- qda(pib_categ ~. -pib_percapita, data=train)
modelQDA
```

```
## Call:
## qda(pib_categ ~ . - pib_percapita, data = train)
##
## Prior probabilities of groups:
##         1         2         3         4
## 0.2377622 0.2797203 0.2517483 0.2307692
##
## Group means:
##   empadronados      paro afiliados declaraciones catastro  turismos
## 1     115.5268 10.915588  138.1321      347.6171 69.84794  473.9056
## 2     123.1835 10.242000  190.4582      173.5958 86.39650  786.5165
## 3     160.6036  9.123611  241.4125      157.6456 98.17917  741.2883
## 4   97311.2467  8.330909  300.2561    49149.6839 83.06758 1386.2542
##   distancia_capital agricultura   energia construccion hosteleria finanzas
## 1          47.14706    5.730000  7.929412     15.15000   23.71412 13.23941
## 2          50.12500    2.957250  8.066000     13.10850   27.45550 13.01175
## 3          44.33333    2.470556 13.462222     11.70167   27.83194 13.94306
## 4          56.84848    3.519394  9.972121     10.33576   28.59273 16.99606
##      otros natalidad
## 1 36.64500 12.089118
## 2 37.45625  9.574250
## 3 32.62583  9.399167
## 4 35.46515  8.758182
```

```
# Prediccion respuesta
qdaResult <- predict(modelQDA, newdata = test)

# Matriz de confusion
tqdamod <- table(qdaResult$class, test$pib_categ)
tqdamod
```

```
##
##     1 2 3 4
##   1 4 0 1 0
##   2 4 2 3 2
##   3 3 2 3 5
##   4 0 1 1 5
```

```
# Precisión
sum(diag(tqdamod))/sum(tqdamod)*100 # Precisión (41,67 %)
```

```
## [1] 38.88889
```

```
#Gráficos de partición
#partimat(pibmadrid[,1:5],pibmadrid$pib_categ,data=pibmadrid,method="qda",main="Gráficos de partición")
#partimat(pibmadrid[,6:9],pibmadrid$pib_categ,data=pibmadrid,method="qda",main="Gráficos de partición")
#partimat(pibmadrid[,10:15],pibmadrid$pib_categ,data=pibmadrid,method="qda",main="Gráficos de partición
```

El porcentaje de acierto es algo inferior al del modelo LDA. Seguimos sacando la conclusión de que al tener que acertar entre cuatro valores de la categórica, el modelo tiene un acierto bastante pequeño. Por ello, en el siguiente apartado realizaremos los modelos para una variable de dos valores posibles.

PIBMadrid como categórica de valores 0 y 1

Dada la baja precisión que hemos obtenido en los modelos anteriores, hemos decidido crear una nueva columna (pib_categ2) para separar el PIB en dos grupos. De esta forma, podemos contrastar si los resultados anteriores son debidos a la mala calidad de los modelos, o que estos no eran suficientemente buenos como para actuar ante una categórica de cuatro valores. Adicionalmente, este cambio nos permite acometer otros contrastes, como la regresión logística.

```
#Creación de una nueva columna con valores 0 y 1 según el PIB per cápita
pibmadrid[,'pib_categ2'] <- cut(pibmadrid$pib_percapita, breaks = c(7333.9, 19287.0, 83698.0), labels =
```

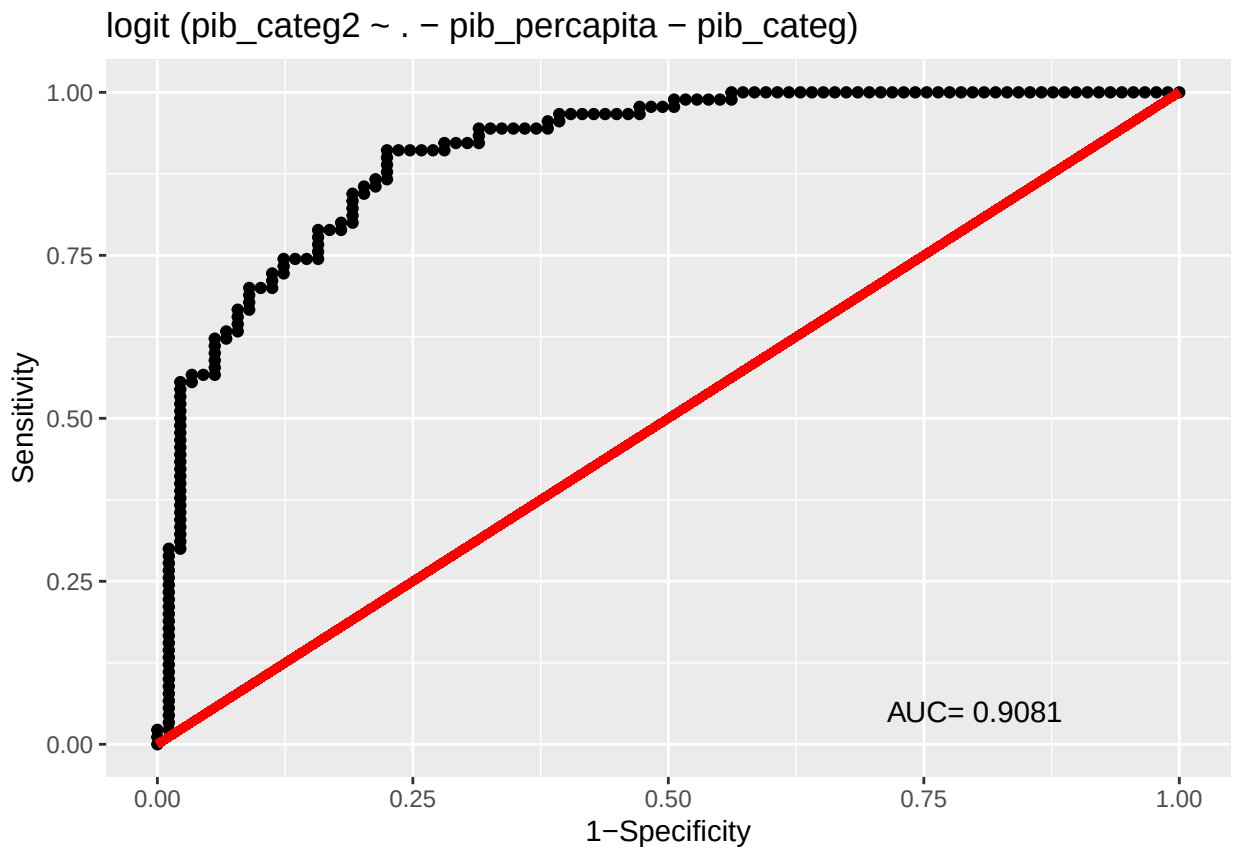
Configuramos de nuevo la muestra de entrenamiento y de test.

```
#Se selecciona una muestra aleatoria y se selecciona el 80% para el entrenamiento.
set.seed(123)
entrenamiento <- sample(x = nrow(pibmadrid), size = nrow(pibmadrid)*0.8, replace = FALSE)

# Subgrupo de datos de entrenamiento
train <- pibmadrid[entrenamiento,]

# Subgrupo de datos de test
test <- pibmadrid[-entrenamiento,]
```

Modelo de regresión logística

Realizamos un modelo de regresión logística y graficamos la curva ROC. Este modelo permite la estimación de una variable categórica binaria en función de variables cuantitativas, como es nuestro caso.

```
#Construcción del modelo
model2 <- glm(pib_categ2 ~.-pib_percapita-pib_categ, data=pibmadrid, family = binomial(), na.action=na.

#Graficar la curva ROC
rocplot(model2)
```



El valor del AUC (que asciende a 0.9081) y la gráfica nos muestran buenos resultados ya que cuanto más cerca esté el valor de 1 mayor será la calidad del modelo.

LDA

Se repite la estimación del modelo LDA pero esta vez con la nueva categórica.

```
#Creación del modelo
modelLDA2 <- lda(pib_categ2 ~. -pib_percapita-pib_categ, data=train)
modelLDA2
```

```
## Call:
## lda(pib_categ2 ~ . - pib_percapita - pib_categ, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.5104895 0.4895105
##
## Group means:
##   empadronados     paro afiliados declaraciones catastro  turismos
## 0     114.6884 10.59397  166.5734      254.5316 79.45534  645.0682
## 1   45964.7981  8.72600  267.9180    23254.2401 90.08757 1041.6606
##   distancia_capital agricultura  energia construccion hosteleria finanzas
## 0          48.28767    4.185616  8.10137     13.89356   25.83575 13.23849
## 1          50.78571    3.037714 11.63657     11.25071   28.05714 15.24314
```

```
##      otros natalidad
## 0 37.03890 10.876712
## 1 34.07457  8.962714
##
## Coefficients of linear discriminants:
##                              LD1
## empadronados       7.857113e-05
## paro              -1.947579e-01
## afiliados          6.025286e-03
## declaraciones     -1.551299e-04
## catastro          -4.445968e-03
## turismos           1.213737e-04
## distancia_capital  2.138387e-02
## agricultura        9.426272e-03
## energia            6.449277e-02
## construccion      -9.429918e-03
## hosteleria         1.732636e-02
## finanzas           3.115089e-02
## otros              1.659341e-03
## natalidad         -6.770790e-02
```

```
# Prediccion respuesta
ldaResult <- predict(modelLDA2, newdata = test)

# Matriz de confusion
tldamod2 <- table(ldaResult$class, test$pib_categ2)
tldamod2
```

```
##
##      0  1
##   0 14  6
##   1  2 14
```

```
# Precision
sum(diag(tldamod2))/sum(tldamod2)*100
```

```
## [1] 77.77778
```

Al tener que predecir ahora entre dos valores posibles, el modelo ofrece un acierto considerablemente superior que anteriormente.

QDA

También pasamos a estimar un modelo discriminante cuadrático con esta nueva configuración.

```
modelQDA2 <- qda(pib_categ2 ~. -pib_percapita-pib_categ, data=train)
modelQDA2
```

```
## Call:
## qda(pib_categ2 ~ . - pib_percapita - pib_categ, data = train)
##
```

```
## Prior probabilities of groups:
##         0         1
## 0.5104895 0.4895105
##
## Group means:
##   empadronados     paro afiliados declaraciones catastro  turismos
## 0     114.6884 10.59397  166.5734      254.5316 79.45534  645.0682
## 1   45964.7981  8.72600  267.9180    23254.2401 90.08757 1041.6606
##   distancia_capital agricultura  energia construccion hosteleria finanzas
## 0          48.28767    4.185616  8.10137     13.89356   25.83575 13.23849
## 1          50.78571    3.037714 11.63657     11.25071   28.05714 15.24314
##      otros natalidad
## 0 37.03890 10.876712
## 1 34.07457  8.962714
```

```
# Prediccion respuesta
qdaResult <- predict(modelQDA2, newdata = test)

# Matriz de confusion
tqdamod <- table(qdaResult$class, test$pib_categ2)
tqdamod
```

```
##
##      0  1
##   0 15 12
##   1  1  8
```

```
# Precisión
sum(diag(tqdamod))/sum(tqdamod)*100 # Precisión
```

```
## [1] 63.88889
```

Lo mismo ocurre con el modelo QDA, que ofrece peor resultado que el LDA pero ampliamente mejor que el QDA estimado para 4 valores de la categórica.

Naive-Bayes

El Naive-Bayes permite clasificar en la medida que calcula la probabilidad de que la variable Y en cuestión pertenezca al valor 0 ó 1 en función del valor de cada una de las variables existentes.

```
#Creación del modelo
modelo_bayes <- naiveBayes(formula= pib_categ2 ~. -pib_percapita- pib_categ, data = train)
modelo_bayes
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
```

```
## Y
##         0         1
## 0.5104895 0.4895105
##
## Conditional probabilities:
##    empadronados
## Y         [,1]        [,2]
##   0   114.6884    252.0311
##   1 45964.7981 383321.5704
##
##    paro
## Y       [,1]     [,2]
##   0 10.59397 2.649614
##   1  8.72600 2.567557
##
##    afiliados
## Y       [,1]      [,2]
##   0 166.5734  92.36043
##   1 267.9180 141.26443
##
##    declaraciones
## Y         [,1]        [,2]
##   0   254.5316    334.2108
##   1 23254.2401 193428.3679
##
##    catastro
## Y       [,1]     [,2]
##   0 79.45534 45.14971
##   1 90.08757 61.17261
##
##    turismos
## Y        [,1]      [,2]
##   0  645.0682  820.0375
##   1 1041.6606 2517.7546
##
##    distancia_capital
## Y       [,1]     [,2]
##   0 48.28767 13.35153
##   1 50.78571 26.89588
##
##    agricultura
## Y       [,1]     [,2]
##   0 4.185616 4.960352
##   1 3.037714 3.714672
##
##    energia
## Y       [,1]      [,2]
##   0  8.10137  6.918985
##   1 11.63657 12.932413
##
##    construccion
## Y       [,1]     [,2]
##   0 13.89356 6.449068
##   1 11.25071 8.316244
```

```
##
##    hosteleria
## Y       [,1]      [,2]
##   0 25.83575  7.622795
##   1 28.05714 11.216729
##
##    finanzas
## Y       [,1]     [,2]
##   0 13.23849  6.49191
##   1 15.24314 10.75993
##
##    otros
## Y       [,1]     [,2]
##   0 37.03890 11.67341
##   1 34.07457 16.37412
##
##    natalidad
## Y        [,1]     [,2]
##   0 10.876712 4.922423
##   1  8.962714 4.719213
```

```
#Predicción
prediccion2 <- predict(modelo_bayes, newdata=test, type = "class")

#Matriz de confusión
matrizconfusion <- table(test$pib_categ2, prediccion2)
matrizconfusion
```

```
##    prediccion2
##      0  1
##   0 15  1
##   1 16  4
```

```
# Porcentaje de aciertos
sum(diag(matrizconfusion))/sum(matrizconfusion)
```

```
## [1] 0.5277778
```

En la salida de código se puede observar lo anteriormente comentado. En cuanto al acierto, ofrece un 52,78%, el cual parece mejorable.

Árboles de decisión

Se ha decidido también utilizar árboles de decisión con el PIB en categórica de valores 0 y 1.

```
#Construcción del árbol
arbolrpart <- rpart(pib_categ2 ~. -pib_percapita -pib_categ, method = "class", data =pibmadrid)
print(arbolrpart)
```

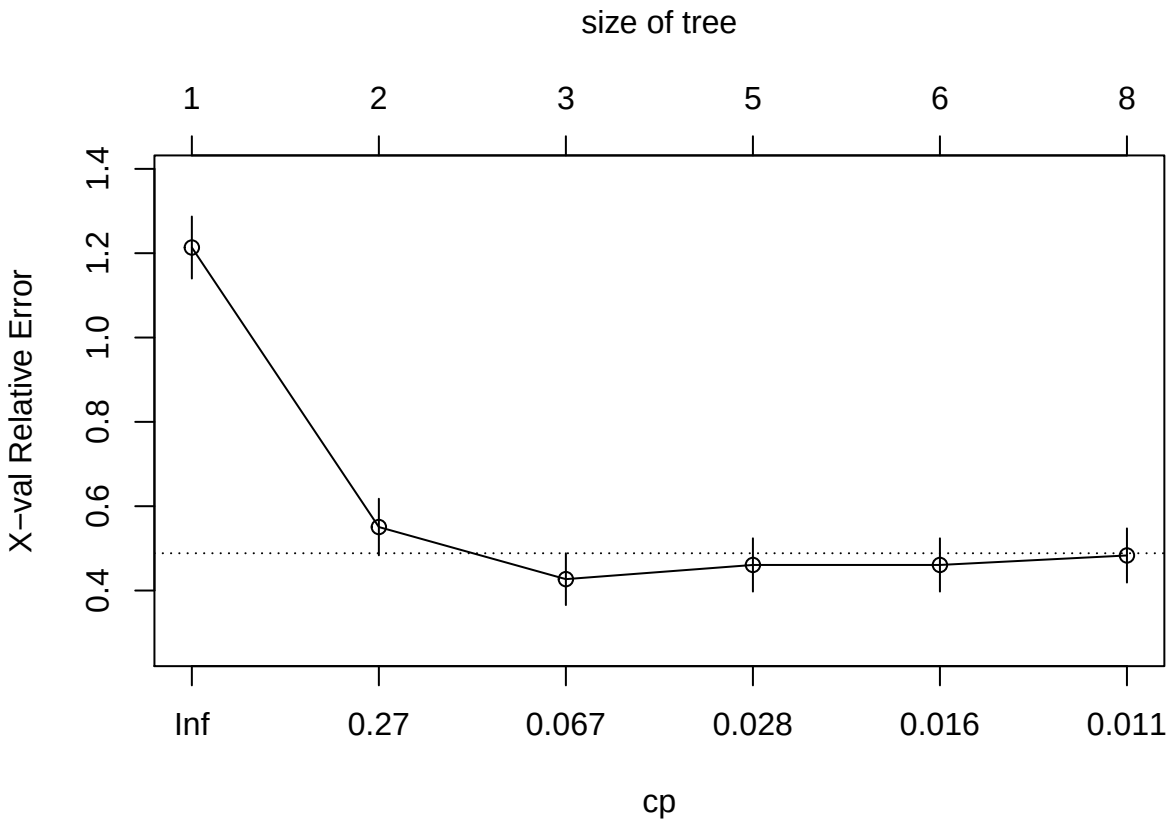
```
## n= 179
##
```

```
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 179 89 1 (0.49720670 0.50279330)
##    2) afiliados< 191.815 102 27 0 (0.73529412 0.26470588)
##      4) distancia_capital< 81.5 88 14 0 (0.84090909 0.15909091)
##        8) distancia_capital< 53.5 55  3 0 (0.94545455 0.05454545) *
##        9) distancia_capital>=53.5 33 11 0 (0.66666667 0.33333333)
##         18) turismos< 469.645 21  2 0 (0.90476190 0.09523810) *
##         19) turismos>=469.645 12  3 1 (0.25000000 0.75000000) *
##      5) distancia_capital>=81.5 14  1 1 (0.07142857 0.92857143) *
##    3) afiliados>=191.815 77 14 1 (0.18181818 0.81818182)
##      6) declaraciones< 2.555 8  3 0 (0.62500000 0.37500000) *
##      7) declaraciones>=2.555 69  9 1 (0.13043478 0.86956522)
##       14) otros>=30.36 24  7 1 (0.29166667 0.70833333)
##         28) finanzas>=14.025 12  5 0 (0.58333333 0.41666667) *
##         29) finanzas< 14.025 12  0 1 (0.00000000 1.00000000) *
##       15) otros< 30.36 45  2 1 (0.04444444 0.95555556) *
```

```
# Estadisticas de resultados
printcp(arbolrpart)
```

```
##
## Classification tree:
## rpart(formula = pib_categ2 ~ . - pib_percapita - pib_categ, data = pibmadrid,
##     method = "class")
##
## Variables actually used in tree construction:
## [1] afiliados         declaraciones     distancia_capital finanzas
## [5] otros             turismos
##
## Root node error: 89/179 = 0.49721
##
## n= 179
##
##         CP nsplit rel error  xerror     xstd
## 1 0.539326      0   1.00000 1.21348 0.073540
## 2 0.134831      1   0.46067 0.55056 0.067028
## 3 0.033708      2   0.32584 0.42697 0.061473
## 4 0.022472      4   0.25843 0.46067 0.063171
## 5 0.011236      5   0.23596 0.46067 0.063171
## 6 0.010000      7   0.21348 0.48315 0.064222
```

```
# Evolución del error a medida que se incrementan los nodos
plotcp(arbolrpart)
```




```
# Forma automática de realizar la "poda"
parbolrpart<- prune(arbolrpart, cp= arbolrpart$cptable[which.min(arbolrpart$cptable[,"xerror"]),"CP"])
printcp(parbolrpart)
```

```
##
## Classification tree:
## rpart(formula = pib_categ2 ~ . - pib_percapita - pib_categ, data = pibmadrid,
##     method = "class")
##
## Variables actually used in tree construction:
## [1] afiliados         distancia_capital
##
## Root node error: 89/179 = 0.49721
##
## n= 179
##
##         CP nsplit rel error  xerror     xstd
## 1 0.539326      0   1.00000 1.21348 0.073540
## 2 0.134831      1   0.46067 0.55056 0.067028
## 3 0.033708      2   0.32584 0.42697 0.061473
```

```
#Predicción
predrpart <- predict(parbolrpart, newdata = train, type = "class")

# Matriz de confusión
```

```r
t1<-table(predrpart, train$pib_categ2)
t1
```

```
##
## predrpart  0  1
##         0 60 11
##         1 13 59
```

```r
# Porcentaje de aciertos
sum(diag(t1))/sum(t1)
```

```
## [1] 0.8321678
```

El resultado de predecir con el árbol de decisión arroja un acierto de algo más del 83%. En otras cuestiones, se puede observar gráfica y numéricamente cuando el error comienza a ascender, que es en el paso de la rama 3 a la 4, en el que se debería realizar la "poda" para evitar que el error siga subiendo.

Support Vector Machine

Por último, el SVM sirve también como método de clasficación binario. Se trata de un modelo que a través de una muestra de entrenamiento, representa los puntos de la muestra en el espacio separando las dos clases a espacios lo más separados posibles. Se ha procedido a crear dicha submuestra de entrenamiento, para posteriomente estimar el modelo y obtener la matriz de confusión y el porcentaje de precisión del modelo construido.

```r
#Creación del set de entrenamiento
set.seed(1234)
train<-sample(seq(length(pibmadrid$pib_categ2)), length(pibmadrid$pib_categ2)*0.80,replace=F)

#Configuración del modelo SVM
svm1 <- svm(pib_categ2~. -pib_percapita -pib_categ, data=pibmadrid, kernel="radial")
print(svm1)
```

```
##
## Call:
## svm(formula = pib_categ2 ~ . - pib_percapita - pib_categ, data = pibmadrid,
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  112
```

```r
summary(svm1)
```

```
##
## Call:
```

```
## svm(formula = pib_categ2 ~ . - pib_percapita - pib_categ, data = pibmadrid,
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  112
##
##  ( 60 52 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#Matriz de confusión
t1<-table(pibmadrid$pib_categ2, svm1$fitted)
t1
```

```
##
##      0  1
##   0 81  8
##   1 13 77
```

```
#Acierto
sum(diag(t1))/sum(t1)
```

```
## [1] 0.8826816
```

Este último modelo ofrece más del 88% de precisión. Es el más elevado que se ha obtenido durante la elaboración de este análisis.

Conclusiones

- De cara al futuro, quizá se hubieran obtenido mejores resultados al contar con datos plenamente actualizados. De la misma forma, una mayor cantidad de variables podría haber mejorado la calidad explicativa de nuestros modelos, aunque arriesgando a incluir problemas de multicolinealidad, por ejemplo.
- A la hora de clasificar, hemos observado claramente que es más probable obtener un mejor resultado al clasificar un PIB categórico de dos valores posibles antes que de cuatro. Es por ello, que se ha probado a trabajar de las dos formas y comparar los resultados.
- A modo de resumen, el SVM y los Árboles de decisión son los que nos han ofrecido una mayor precisión (por encima del 80%). En cuanto al resto, se ha observado de forma clara que al trabajar con una variable explicada de dos valores posibles la precisión de los modelos es manifiestamente superior a cuando las categorías posibles son cuatro.