Minería de Datos y Modelización Predictiva (I)

Fernández Hernández, Alberto. 54003003S

12/28/2020

Contents

1. Depuración de los datos

1.1 Análisis del conjunto de datos

Inicialmente nos encontramos ante un conjunto de datos con la información demográfica de los diferentes municipios en España así como sus últimos resultados electorales. Por ello, antes de elegir las variables objetivo y elaborar el modelo de regresión, echemos un primer vistazo a los datos proporcionados. En primer lugar, y antes de analizar las variables independientes, debemos **recategorizar las variables cualitativas como factor**, dado que el formato establecido por defecto es numérico. Sin contar el campo *CodigoProvincia* (el cual mencionaremos más adelante) existen un total de 6 variables categóricas, incluyendo las variables objetivo cualitativas (AbstencionAlta, Izquierda y Derecha), además de los campos CCAA, ActividadPpal y Densidad, dado que contienen un número limitado de valores únicos:

```
# c(3, 7, 11, 12, 34, 38) => CCAA, AbstencionAlta, Izquierda, Derecha, ActividadPpal y Densidad
datos[,c(3, 7, 11, 12, 34, 38)] <- lapply(datos[,c(3, 7, 11, 12, 34, 38)], factor)
```

Por otro lado, podemos observar que los datos proporcionados contienen un total de 41 variables, de las cuales cabe destacar el campo identificador *Name*, un campo con el nombre de cada municipio:

```
## Nombres de municipio unicos:  8102  de  8119 filas. Numero columnas:  41
```

Salvo excepciones, en las que el nombre del municipio coincide, se trata de un campo que podríamos considerar como identificativo, por lo que no nos aportará información relevante al modelo y por ello lo eliminamos:

```
datos <- datos[, -c(1)] # Eliminamos el campo identificador
```

Por otro lado, nos encontramos con el campo *CodigoProvincia* que, a diferencia del anterior, el número de valores diferentes es significativamente menor (52 valores únicos). No obstante, nos encontramos ante la siguiente duda ¿Mantenemos los datos en formato numérico o los recategorizamos a *factor*? Por un lado, recategorizarlo como una variable cualitativa puede llegar a entorpecer la elaboración del modelo, en especial si una o varias de las categorías no está lo suficientemente representada y debe ser agrupada. Por otro lado, si queremos mantener la variable como un campo numérico tiene que aportar "sentido" al modelo, esto es, asegurar que la media del código de provincia es de 26.67, por ejemplo, no aporta información, además de que no tiene sentido hablar de media o mediana en este campo. Por otro lado, si quisiéramos emplearlo como un campo más, deberá aportar un cierto grado de importancia, tanto a las variables objetivo cuantitativas como cualitativas:

```
# Columnas 5,7,8 y 9 correspondientes con las variables objetivo cuantitativas
cor(cbind(datos$CodigoProvincia, datos[, c(5,7,8,9)]), use = "complete.obs", method = "pearson")[1,-1]
```

```
## AbstentionPtge       Izda_Pct       Dcha_Pct      Otros_Pct
##    -0.05711613    -0.01653135     0.12657294    -0.09079252
```

```
# Mediante la V Cramer vemos la importancia sobre las variables objetivo cualitativas
salida <- c()
for(col in c("AbstencionAlta", "Izquierda", "Derecha")) {
  salida <- cbind(salida, sapply(datos[ , "CodigoProvincia"],function(x) Vcramer(x,unlist(datos[, col]))))
}
salida
```

```
##                       [,1]      [,2]      [,3]
## CodigoProvincia 0.1508031 0.1609849 0.2371896
```

Como podemos observar en ambas salidas, la correlación tanto en las variables cuantitativas como cualitativas no superan el 0.12 y 0.23 respectivamente, valores bastante bajos, por lo que podemos descartar la variables para ambos modelos. Antes de continuar, de cara a valorar la calidad de la depuración final guardamos en una variable los valores de correlación originales, con el objetivo de compararlos con los del conjunto de datos ya depurado:

```
corr.previa <- cor(datos[,unlist(lapply(datos, is.numeric))], use="complete.obs", method="pearson")
```

1.1 Valores erróneos o no declarados

A continuación, procedemos a eliminar aquellos valores no declarados en las variables, así como posible valores fuera de rango:

1. *ForeignersPtge* negativos. Porcentajes de extranjeros menores a cero:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -8.96    1.06    3.59    5.62    8.18   71.47
```

```
datos$ForeignersPtge<-replace(datos$ForeignersPtge, which(datos$ForeignersPtge < 0), NA)  # Min < 0
```

2. Porcentajes de *SameComAutonPtge* y *PobChange_pct* superiores al 100 % (en el caso de *PobChange_pct* según la documentación es posible la aparición de porcentajes negativos, pero no superiores al 100 %):

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   75.81   84.49   81.63   90.46  127.16
```

```
datos$SameComAutonPtge <-replace(datos$SameComAutonPtge, which(datos$SameComAutonPtge > 100), NA)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -52.2700 -10.4000  -4.9600  -4.8974   0.0925 138.4600        7
```

```
datos$PobChange_pct <-replace(datos$PobChange_pct, which(datos$PobChange_pct > 100), NA) # Max > 100
```

3. Valores a 99.999 en la columna *Explotaciones*, posible indicativo de la ausencia de valores en estos casos:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1      22      52    2447     137   99999
```

```
datos$Explotaciones<-replace(datos$Explotaciones,which(datos$Explotaciones==99999),NA) # Max == 99999
```

4. Categoría "?" sin declarar en *Densidad*, por lo que lo recategorizamos a *NA*:

```
##         ?  Alta Baja MuyBaja
## n    92.0 557.0 1053    6417
## %     1.1   6.9   13      79
## val%  1.1   6.9   13      79
```

```
datos$Densidad<-recode.na(datos$Densidad,"?")
```

```
## Recoded 92 values to NA.
```

1.2 Análisis de valores atípicos

Una vez corregidos los errores detectados, analicemos los valores atípicos más destacados empleando la función *describe*:

```
##                           sd  skew kurtosis
## Population          46215.20 45.98  2814.43
## TotalCensus         34428.89 46.49  2888.34
## totalEmpresas        4219.37 53.68  3472.00
## ComercTTEHosteleria  1233.02 45.40  2646.94
## Servicios            2446.81 57.48  3830.74
## inmuebles           24314.71 44.53  2643.65
## Pob2010             47535.68 47.15  2939.56
## SUPERFICIE           9218.19  6.07    62.28
```

Como podemos observar en la salida anterior, las columnas con la población, el censo total, el número total de empresas, así como la superficie son los que mayor desviación presentan con respecto a su media, lo que se traduce en, además de un coeficiente de simetría mayor a 1 (asimetría) y una elevada curtosis, **claros indicios de la presencia de valores atípicos** (algo lógico si nos planteamos el caso de la Población, con municipios que no superan el millar de habitantes en contraste con grandes municipios como Madrid o Barcelona). Por ello, comenzamos analizando el porcentaje maximo de valores atipicos en nuestro conjunto de datos:

```
## Servicios
##  11.87338
```

En este caso, el maximo porcentaje corresponde con el campo *Servicios*, con un 11.87 %, una de las variables con elevada desviación típica que habíamos planteado previamente. Por ello, dado que el máximo porcentaje de valores atípicos no supera el 12 % podemos marcalos como ausentes sin problema alguno para posteriormente imputarlos.

1.3 Análisis de valores missings (NA). Imputaciones

Tras recodificar los valores atípicos como ausentes, debemos analizar la proporción de valores atípicos tanto por observacion como por variable. Para ello, obtenemos el valor maximo de *missings* tanto por fila como por columna:

```
##   Por.observacion Por.variable
## 1            37.5     12.63702
```

Aparentemente, mientras que el porcentaje de *missings* por variable es del 12.64 %, por observaciones detectamos un mayor numero (37.5 %). No obstante, si empleamos la función *summary*:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   3.874   3.125  37.500
```

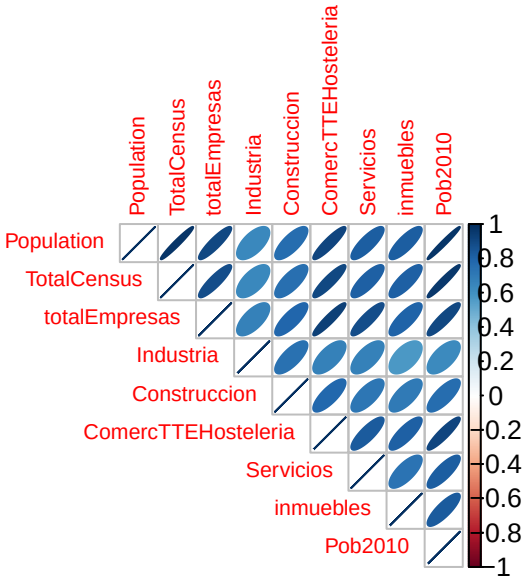
Vemos que el 75 % de las observaciones contienen aproximadamente un 3 % de valores *missings* o menos, por lo que no parece tratarse varias filas (de hecho, solo el 25 % presenta un porcentaje de *missings* superior al 3 %). Por otro lado, la pérdida de información en ambos casos no supera el 50 %, por lo que en lugar de eliminar dichos valores podemos imputarlos. De forma previa a la imputación, existen determinados campos que pueden ser imputados sin necesidad de emplear una media, mediana o de forma aleatoria:

1. *Age_19_65_pct*, cuyo porcentaje de edad puede calcularse a partir de la suma de *Age_under19_Ptge* y *Age_over65_pct* menos el 100 %:

```
x["Age_19_65_pct"] <- 100 - (as.numeric(x["Age_under19_Ptge"]) + as.numeric(x["Age_over65_pct"]))
# Ejemplo demostrativo de que 100 - Age_under19_Ptge + Age_over65_pct = Age_19_65_pct
```

```
##   Age_19_65_pct X100.Age_under_19.Age_over65
## 1        55.059                       55.060
## 2        56.643                       56.641
## 3        54.834                       54.833
## 4        60.098                       60.096
## 5        59.391                       59.389
```

2. *totalEmpresas* ¿Podría calcularse a partir de la suma del número de empresas de cada sector? *Industria*, *Construcción*, *ComercioTTEHostelería* y *Servicios*. Una primera prueba para comprobar si el campo *totalEmpresas* es la suma de cada columna podría ser analizando la matriz de correlación de valores *missings*: **Si el datos de cualquier sector falta, el de totalEmpresas tampoco debería aparecer al no poder calcularse**:



Efectivamente, **detectamos una correlación entre los valores *missing* de *totalEmpresas* con cada sector**. De hecho, no solo existe correlación entre *totalEmpresas*, sino incluso entre cada sector. A modo de ejemplo, si el número de empresas dedicadas al comercio no aparece, el número de empresas dedicadas a la construcción tampoco. Incluso si la población no aparece, tampoco suelen aparecer el número de empresas, el censo total (algo razonable

ya que la población con derecho a voto depende de la población del municipio), e incluso tampoco suele aparece la población registrada en el año 2010 (No se tratan de valores *missing* aleatorios, siguen un patrón). Por tanto, el campo *totalEmpresas* puede calcularse a partir de la suma de cada sector. Como última prueba, realicemos una comprobación manual, sumando cada columna para comprobar si coincide con *totalEmpresas*:

```
x["totalEmpresas"] <- as.numeric(x["Industria"]) + as.numeric(x["Construccion"]) +
                      as.numeric(x["ComercTTEHosteleria"]) + as.numeric(x["Servicios"])
# Ejemplo demostrativo de la suma de las empresas de cada sector, en funcion de ActividadPpal
```

```
##               ActividadPpal
## Coincide_Suma. ComercTTEHosteleria Construccion Industria Otro Servicios
##             NO                   0            0         0 3712         0
##             SI                1700           11         9 1215       195
```

Por lo general, cuando la actividad principal es *Otro*, la suma de cada columna no suele coincidir con *totalEmpresas* (al ser predominante otro tipo de Actividad). No obstante, tal como se puede comprobar en la siguiente salida, de los valores *missing* de *totalEmpresas*, sólo existen 5 filas con la actividad principal a *Otro*, por lo que podemos realizar la imputación sin problema alguno:

```
##
## ComercTTEHosteleria        Construccion           Industria                Otro
##                 527                   0                   0                   5
##           Servicios
##                 326
```

3. *Densidad*, cuyo valor puede obtenerse a través del cociente entre *Population* y *SUPERFICIE*: si la proporcion es menor a 1 decimos que la densidad es "MuyBaja"; si está entre 1 y 5 decimos que es "Baja" y si es mayor a 5 diremos que es "Alta":

```
ifelse(proporcion < 1, densidad <- "MuyBaja", ifelse(proporcion >= 1 & proporcion <= 5,
      densidad <- "Baja", densidad <- "Alta"))
# Ejemplo demostrativo del cociente entre Population y SUPERFICIE
```

```
##          Cociente_Pob_SUP
## Densidad  Alta Baja MuyBaja
##   Alta     140    0       0
##   Baja       0  755       0
##   MuyBaja    0    0    6152
```

Para el proceso de imputación en el resto de variables se ha dividido en un total de dos fases por el siguiente motivo: hay demasiados valores *missing* consecutivos. Esto supone que, a la hora de realizar la imputación con valores aleatorios (mediante una interpolación) muchos de los valores *missing* quedan sin imputarse, dado que muchos de ellos parecen ser consecutivos, lo que impide calcular su valor. Para ello, se realiza una primera imputación de forma aleatoria para, a continuación, imputar los valores restantes mediante la mediana (dado que muchos de los valores atípicos marcados a *missing* presentaban una elevada desviación típica, lo que hace que la mediana sea mucho más representativa que la media):

```
## 10064  valores missing
```

```
datos[,columnas] <- sapply(datos[, columnas],function(x) ImputacionCuant(x,"aleatorio"))
```

```
## 1191  valores missing tras la imputacion aleatoria (NO se ha eliminado todos)
```

```
datos[,columnas] <- sapply(datos[, columnas],function(x) ImputacionCuant(x,"mediana"))
```

```
## 974  valores missing tras la imputacion con la mediana
```

Como podemos observar, hemos conseguido reducir el porcentaje de *missings*. A continuación, si imputamos manualmente las tres columnas mencionadas anteriormente a partir del resto de variables:

```
## 0  valores missing
```

Tras la imputación final, veamos qué porcentaje de correlación se ha perdido comparando la correlación del conjunto de datos inicial con respecto al conjunto de datos depurado:

```r
corr.posterior <- cor(datos[,unlist(lapply(datos, is.numeric))], use = "complete.obs" , method="pearson")
comparacion.corr <- corr.posterior[-33, -33] - corr.previa # Eliminamos el campo prop_missings
sum(abs(comparacion.corr) < 0.2) * 100 / (dim(comparacion.corr)[1] * dim(comparacion.corr)[2])
```

```
## [1] 82.5528
```

Podemos observar que aproximadamente un 82.55 % de las variables se ha visto reducida en menos de 0.2 con respecto a la correlación original, bastante mayor con respecto a una imputación únicamente con la media o con la mediana. Tras eliminar los valores *missing* debemos preguntarnos ¿Qué variables escojo como variables objetivo? Para ello, se ha empleado como criterio **aquella variable objetivo** con mayor suma de correlaciones con respecto al resto de variables:

```
## AbstentionPtge . Suma correlaciones:  3.311757
## Izda_Pct . Suma correlaciones:  4.72752
## Dcha_Pct . Suma correlaciones:  7.711944
## Otros_Pct . Suma correlaciones:  8.56923
```

```
## AbstencionAlta . Suma correlaciones:  4.910279
## Izquierda . Suma correlaciones:  5.432872
## Derecha . Suma correlaciones:  8.085535
```

Comenzando con las variables cuantitativas, en un principio elegiríamos como variable objetivo *Otros_Pct*, de no ser por un inconveniente: el elevado número de valores atípicos que presenta, lo que puede llegar a dificultar la elaboración del modelo:

```r
sapply(Filter(is.numeric, datos[, c(4:8)]),function(x) atipicosAmissing(x)[[2]]) * 100/nrow(datos)
```

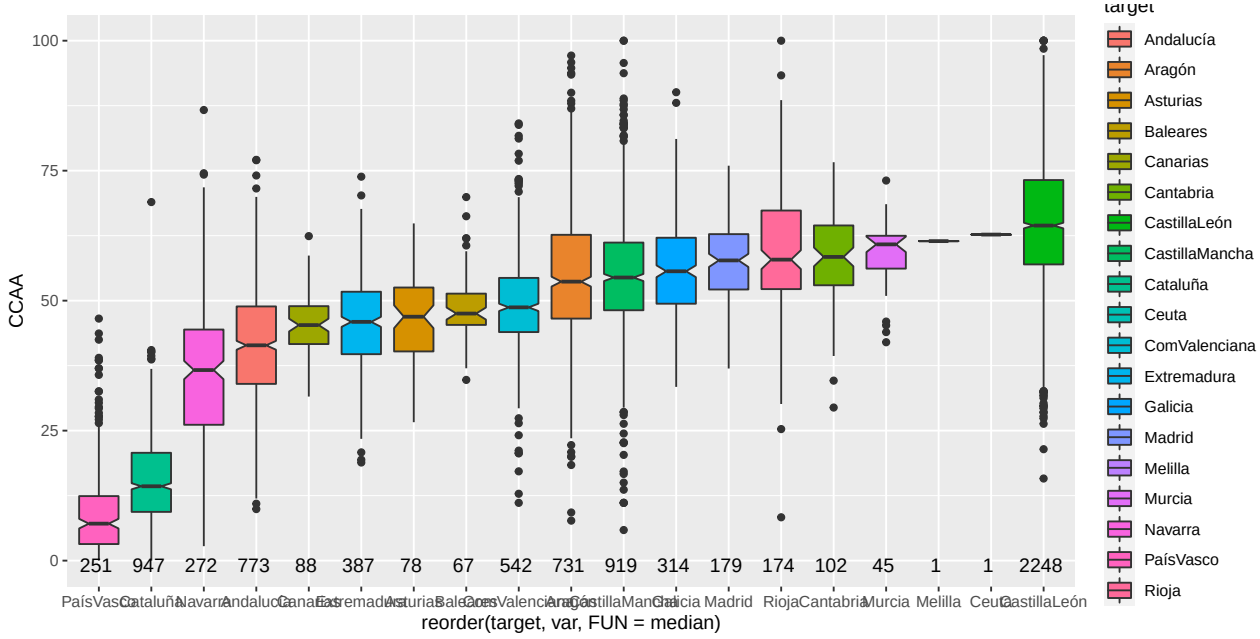
```
## AbstentionPtge       Izda_Pct       Dcha_Pct      Otros_Pct
##        0.00000        0.00000        0.00000       10.40769
```

Por ello, y con el objetivo de facilitar el desarrollo de un modelo lineal elegimos el porcentaje de votos a la derecha o *Dcha_Pct*, el cual también presenta un porcentaje de correlación elevado con respecto al resto de variables. En relación a la variable binaria, dada la elevada correlación acumulada que presenta, elegimos como variable objetivo binaria el campo *Derecha*. El resto de variables objetivo son eliminadas.

Tras eliminar el resto de variables, debemos hacernos la siguiente pregunta. De las variables cualitativas ¿Podemos agrupar alguna de sus categorías? Salvo el campo *Densidad*, donde la frecuencia de cada categoría está repartida de forma equitativa:

```
##      MuyBaja Baja  Alta
## n     6509.0 1053 557.0
## %       80.2   13   6.9
## val%    80.2   13   6.9
```

Tanto en el campo *CCAA* como *ActividadPpal* debemos agrupar algunas de las categorías. Comenzando con las Comunidades Autónomas, dado que disponemos de 19 valores diferentes (algunos de los cuales como Ceuta o Melilla con una única representación), tal y como se muestra a continuación:



Para agrupar las Comunidades Autónomas, no solo agruparemos aquellas categorías con un menor número de variables sino además aquellas Comunidades **cuya amplitud en el diagrama de caja y bigotes sea similar**: **País Vasco y Cataluña**; **Navarra y Andalucía**; **ComValenciana, Extremadura, Asturias, Baleares y Canarias**; **Aragón y Castilla la Mancha**; así como **Galicia, Cantabria, Madrid, La Rioja, Ceuta, Melilla y Murcia'**. En el caso de Castilla y León, dado que se trata de la CCAA con mayor número de observaciones, no la agruparemos con otra comunidad. No obstante, de cara a la creación de los modelos es importante tener en cuenta que se trata de la CCAA con la mayor distribución de votos hacia la derecha, además de ser la única categoría que no ha sido agrupada, por lo que lo consideraremos como la **categoría de referencia**, recodificando su nombre a *AA_CL* (de esta manera la categoría será elegida como referencia por orden alfabético):

```
datos$CCAA <- recode(datos$CCAA, "c('Navarra', 'Andalucía') = 'AN_NA'; c('Cataluña', 'PaísVasco') = 'CAT_PV';
c('ComValenciana', 'Extremadura', 'Asturias', 'Baleares', 'Canarias') = 'CV_EX_AS_BA_CA';
c('Aragón', 'CastillaMancha') = 'AR_CM'; c('CastillaLeón') = 'AA_CL';
c('Galicia', 'Cantabria', 'Madrid', 'Rioja', 'Ceuta', 'Melilla', 'Murcia') = 'MA_CA_RI_CE_ME_MU_GA';")
```

En contraposición, nos encontramos con el campo *ActividadPpal*:

```
##      ComercTTEHosteleria Construccion Industria   Otro Servicios
## n                 2540.0         14.0      13.0 4932.0     620.0
## %                   31.3          0.2       0.2   60.7       7.6
## val%                31.3          0.2       0.2   60.7       7.6
```

En este campo, nos encontramos con los campo *Construccion* e *Industria* con apenas 14 y 13 apariciones, respectivamente. Por ello, dado que solo hay que agrupar dos categorías con poca representación (a diferencia de las CCAA donde teníamos 19), lo agruparemos con la categoría con mayor representación: *Otro*, dado que dispone de una amplitud similar en el diagrama de caja y bigotes:

```
datos$ActividadPpal <- recode(datos$ActividadPpal,
                      "c('Construccion', 'Industria', 'Otro') = 'Construccion_Industria_Otro';")
```

1.4 Transformaciones de variables y relaciones con las variables objetivo

Una vez recategoriazadas las variables, puede ser necesario realizar alguna transformación en las variables para poder plasmar de este modo la verdadera relación de las variables independientes con la variable objetivo:

```
input_cont<-data.frame(varObjCont,datos,Transf_Auto(Filter(is.numeric, datos),varObjCont))
input_bin<-data.frame(varObjBin,datos,Transf_Auto(Filter(is.numeric, datos),varObjBin))
```

Sin embargo, debemos recordar un detalle fundamental: **cuantas más variables empleemos para elaborar**

nuestros modelos, más costoso (computacionalmente) será obtenerlo. Por tanto, de todas las transformaciones que hemos obtenido ¿Algunas de ellas mejoran la correlación con las variables objetivo? Comenzando con la variable objetivo continua, **compararemos las dos matrices de correlación: las variables originales y sus transformadas**. Sobre la diferencia filtraremos aquellas cuya diferencia sea igual o inferior a 0.01 (prácticamente idénticas):

```
correlaciones  <- round(cor(Filter(is.numeric, input_cont), use="pairwise", method="pearson")[1,32:61]
                        - cor(Filter(is.numeric, input_cont), use="pairwise", method="pearson")[1,2:31], 2)
# Filtramos aquellas variables transformadas que apenas hayan aumentado su correlacion
colnames(input_cont)[colnames(input_cont) %in% names(correlaciones[correlaciones > -0.01])]
```

```
##  [1] "xAge_under19_Ptge"         "sqrtxAge_19_65_pct"
##  [3] "xAge_over65_pct"           "xWomanPopulationPtge"
##  [5] "expxSameComAutonPtge"      "xSameComAutonDiffProvPtge"
##  [7] "logxSUPERFICIE"            "xPersonasInmueble"
##  [9] "sqrtxExplotaciones"        "sqrtxprop_missings"
```

Como podemos observar, 10 de las transformadas no aportan apenas mejoría al modelo. Por tanto eliminamos dichas transformadas, junto con aquellas variables originales que aportan menor correlación con respecto a sus transformadas.

Con respecto a la variable objetivo binaria, emplearemos el criterio del **Valor de la información** o *Information Value*, criterio que nos permite medir la capacidad predictiva que presenta una variable (agrupada en intervalos) para predecir con precisión los valores 1,0 de la variable objetivo. El objetivo será calcular la diferencia entre el *Information Value* de cada una de las variables originales (agrupadas por intervalos) con respecto a las variables transformadas:

```
salida.woe <- woebin(input_bin, "varObjBin", print_step = 0)
```

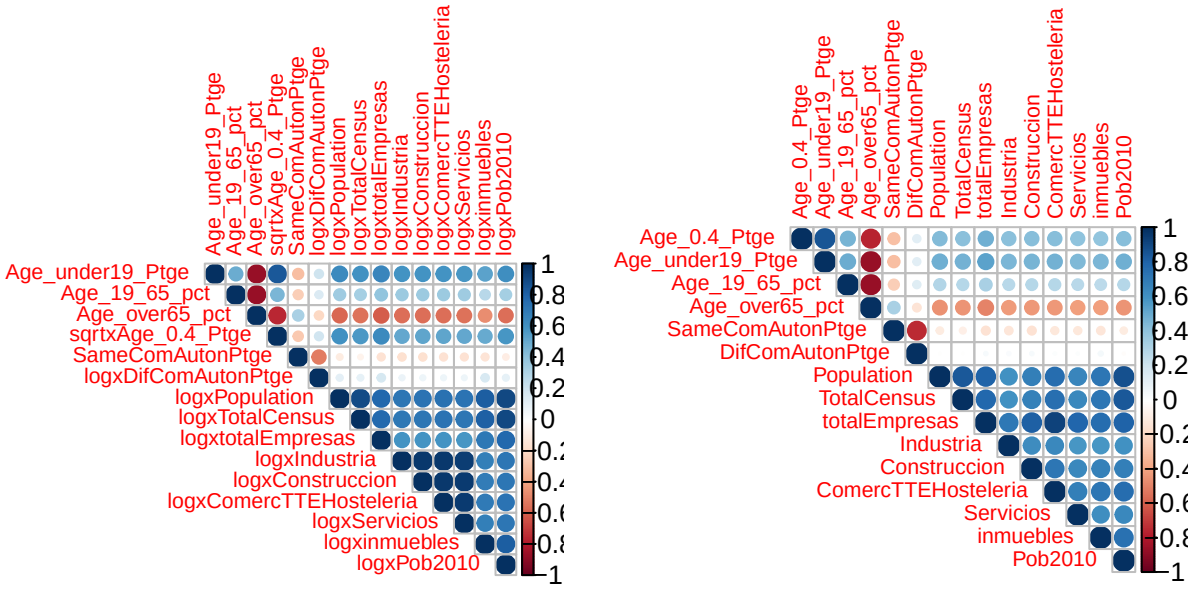
```
summary(sapply(salida.woe[c(2:24,26:28,30:33)], function(x) x$total_iv[1]) -
          sapply(salida.woe[c(34:63)], function(x) x$total_iv[1]))
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -3.306e-02 -1.824e-03 -9.050e-06 -1.217e-04  2.324e-03  1.676e-02
```

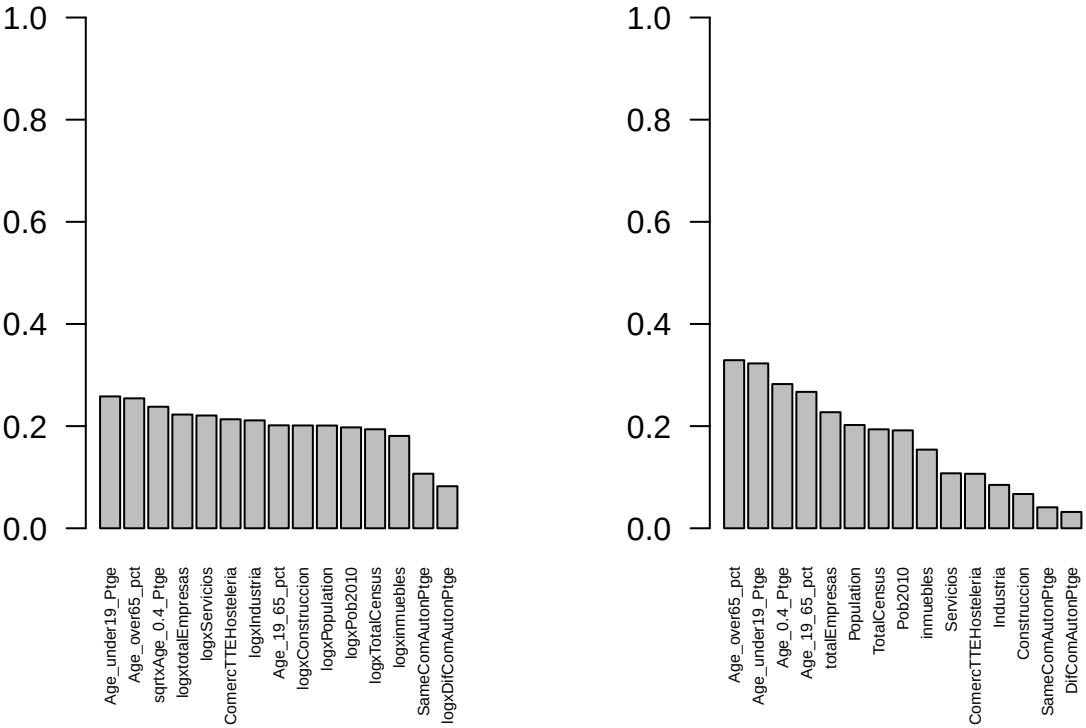
Si nos fijamos en la salida anterior, prácticamente ninguna de las variables transformadas mejora significativamente con respecto a la variable original, donde en el mejor de los casos el aporte máximo al valor de información es de 0.015. Por otro lado, el tercer cuartil nos indica que el 75 % de las variables ve mejorado su valor de información en 0.0025 o menos, por lo que apenas supone una mejoría. Por tanto, para el modelo de regresión logístico **mantendremos las variables origiales** (eliminamos las transformadas).

1.5 Detección de las relaciones entre las variables input y objetivo

Como último paso en el proceso de depuración, debemos analizar qué relación existen entre las variables independientes y objetivo, pero incluso algo mucho más importante ¿Hay correlación entre las variables independientes? Esto último debe tenerse en cuenta, dado que una alta colinealidad puede reducir significativamente la calidad de ambos modelos. Para ello, quisiera destacar cuatro casos en los que se produce colinealidad:



Tanto en las variables originales como incluso con aquellas transformadas, nos encontramos con grupos de variables que presentan una elevada correlación entre sí: **los grupos de edad** (entre 0 y 4 años, menor a 19, entre 19 y 65 y más de 65 años); **si residen o no en la misma CCAA** (*SameComAutomPtge* y *DifComAutonPtge*); el total de empresas con respecto al número de empresas de cada sector (Industria, Construcción, Comercio, Servicios, así como el número de inmuebles y la población en el año 2010); e incluso una elevada correlación entre el campo *Population* y *totalCensus*. Para solventar el problema, empleando la **V de Cramer** escojemos aquellas variables de cada grupo que tenga una mayor relevancia con respecto a la variable objetivo, mientras que el resto las eliminamos:



Como podemos observar en los gráfico anteriores, para el modelo lineal *input_cont* nos quedaremos con las variables *Age_under_19_Ptge* y *Age_19_65_pct* (no están muy correlacionadas entre sí), *SameComAutonPtge*, *logxPopulation* y *logxtotalEmpresas*. Con respecto a *input_bin*, conservaremos *Age_over65_pct*, *SameComAutonPtge*, *Population* y *totalEmpresas*.

2. Construcción del modelo de regresion lineal

2.1 Elaboración del modelo inicial

Una vez realizada la depuración de los datos, procedemos a elaborar el modelo de regresión lineal. En primera instancia, **ejecutaremos un primer modelo con todas las variables y todas sus transformaciones**, con el objetivo de echar un primer vistazo al modelo inicial (aunque no sea el definitivo):

```
formInt<-formulaInteracciones(input_cont,1)
modelo1<-lm(formInt,data=data_train)
# Funcion que devuelve los valores Train,Test,AIC y SBC
mostrar.estadisticas(modelo1, data_train, data_test, "lm", "varObjCont")
```

```
## Train:  0.7497424 ; Test:  0.7057302 ; Dif. (Train-Test):  0.0440122 ; AIC: 48774.44 ; SBC:  50367.49
## Numero de variables:  234
```

Inicialmente, con 234 variables nos encontramos con un modelo con valores de AIC y BIC bastante elevados, aunque con una diferencia entre el train y test de solo 0.04. Esto supone, en primer lugar, que no todas las variables presentan la misma importancia en el modelo (incluso puede que muchas de ellas no aporten prácticamente información). Para comprobarlo, analizaremos las estadísticas de cada variable empleando la función *modelEffectSizes*:

```
variacion.r2 <- modelEffectSizes(modelo1, Print = FALSE)
summary(variacion.r2$Effects[, 4])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
## 0.0000001 0.0000361 0.0001190 0.0003016 0.0003355 0.0077325         1
```

Si recordamos de la refactorización de las CCAA, las provincias de MA_CA_RI_CE_ME_MU_GA presentaban un menor número de filas ¿Y si los agrupamos con la categoría con la mediana más similar? En este caso, CCAAAR_CM:

```
Train:  0.7467209 ; Test:  0.7026724 ; Dif. (Train-Test):  0.04404844 ; AIC: 48802.39 ; SBC:  50225.97
Numero de variables:  209
```

En este caso, pese a que el SBC disminuya, tanto el criterio AIC como la diferencia entre el R2 train-test aumenta ligeramente, por lo que es posible que estemos perdidendo interacciones significativas con el resto de variables, por lo que mantedremos las categorías originales.

Analizando la salida del primer modelo, podemos comprobar que el 75 % de las variables del modelo inicial presentan una importancia en el R2 de 0.0003 o menos, es decir, existe solo un 25 % de las variables que aportan más de 0.0003, por lo que debemos centrarnos en estas últimas (de cara a facilitar la selección clásica). Por ello, elegimos del modelo original aquellas variables atípicas (aportan más de 0.0003 al R2):

```
variables.mas.imp <- names(boxplot(variacion.r2$Effects[, 4], plot = FALSE)$out)
variables.mas.imp
```

```
## [1] "CCAA:SameComAutonPtge"   "CCAA:PersonasInmueble"
## [3] "CCAA:logxForeignersPtge"
```

En este caso, las variables con mayor importancia corresponden con interacciones con el campo CCAA. Por tanto, de cara a un segundo modelo conservaremos las transformaciones más importantes junto con las columnas originales, dado que algunas de las variables pueden proporcionar más información al modelo si no están incluidas en ninguna interacción:

```
formInt <- paste0("varObjCont~",paste0(colnames(input_cont[-1]), collapse = "+"),"+",
                  paste0(variables.mas.imp, collapse = "+"))
modelo1.2<-lm(as.formula(formInt),data=data_train)
mostrar.estadisticas(modelo1.2, data_train, data_test, "lm", "varObjCont")
```

```
## Train:  0.7279723 ; Test:  0.7226042 ; Dif. (Train-Test):  0.005368091 ; AIC: 48938.29 ; SBC:  49250.12
## Numero de variables:  45
```

Reduciendo el número de parámetros a 45, pese a aumentar el AIC en más de 100 puntos, el criterio SBC consigue verse reducido en más de 1000, además de recortar la diferencia entre el R2 obtenido en los datos de entrenamiento y *test*.

2.2 Selección de variables clásica

A continuación, con la formula del segundo modelo, podemos partir como base para la selección de variables clásica:

```
estadisticas.modelos <- seleccion.clasica(formInt, data_train, data_test, "lm")
```

```
##              R.2.train  R.2.test   Diferencia      AIC      SBC N.Parametros
## AIC-both     0.7260226 0.7229653 0.0030572632 48950.68 49147.27           28
## SBC-both     0.7122427 0.7116922 0.0005504356 49253.45 49395.81           20
## AIC-forward  0.7261075 0.7227644 0.0033430844 48952.66 49162.81           30
## SBC-forward  0.7122427 0.7116922 0.0005504356 49253.45 49395.81           20
## AIC-backward 0.7277198 0.7229281 0.0047917427 48924.31 49168.35           35
## SBC-backward 0.7249481 0.7227676 0.0021804760 48966.11 49128.80           23
```

Una vez calculada la selección clásica, realizamos la validación cruzada:

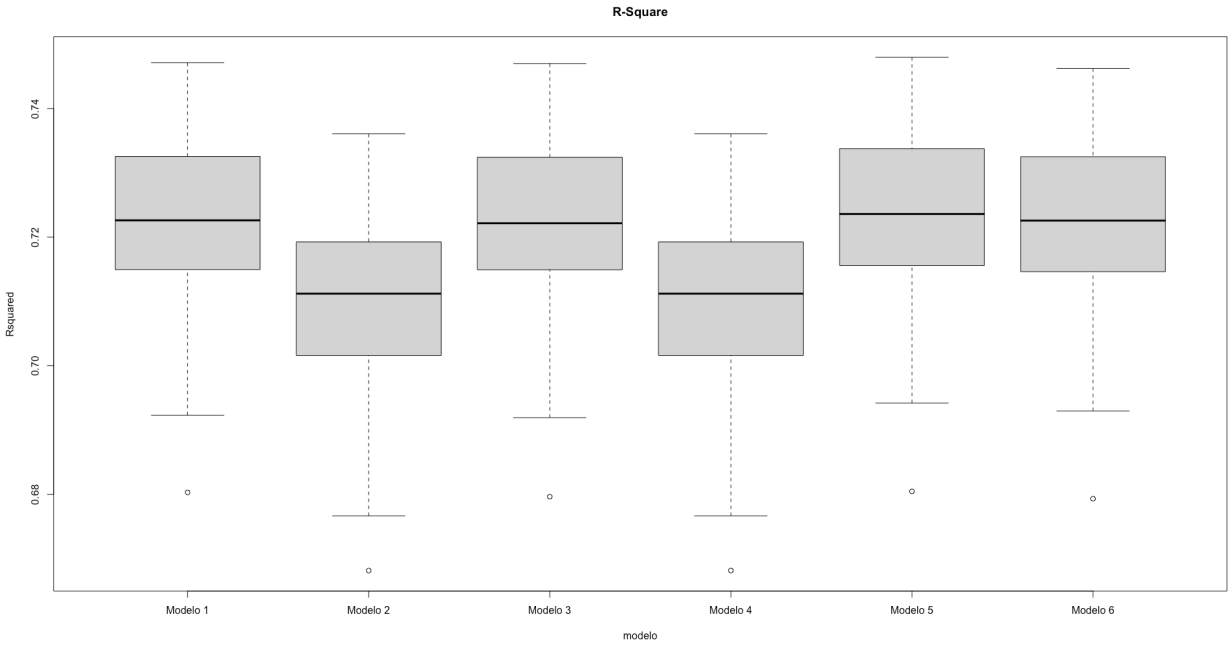


Figure 1: Validación cruzada en el modelo de selección clásica

```
estadisticas.modelos.final # Mostramos la media y desviacion tipica obtenida en la validacion cruzada
```

```
##     modelo     media         sd
## 1 Modelo 1 0.7228260 0.01340819
## 2 Modelo 2 0.7101190 0.01399851
## 3 Modelo 3 0.7226791 0.01341586
## 4 Modelo 4 0.7101190 0.01399851
## 5 Modelo 5 0.7237219 0.01348659
## 6 Modelo 6 0.7222440 0.01340825
```

Analizando los resultados obtenidos en la selección aleatoria y en la validacion cruzada, los mejores modelos obtenidos han sido el número 1 y el número 6 en términos de desviación de típica (0.01340819 y 0.01340825, respectivamente), ya que el modelo 2, pese a tener menos parámetros, su media de R2 es bastante menor 0.7101190. Dado que la bondad de ajuste es muy similar en el gráfico (tamaño de la caja como de los bigote), analicemos las estadísticas obtenidas por ambos modelos: pese a que el AIC en el modelo 1 es ligeramente mayor, tanto el criterio SBC como la diferencia entre el R2 obtenido en train y test da una mayor ventaja al modelo 6. Sin embargo, la media de R2 no parece dejar claro cual es el modelo ganador, ya que ambas son similares (0.7228260 en el modelo 1 y 0.7222440 en el modelo 6). No obstante, si nos guiamos por el principio de parsimonia, **ante dos posibles explicaciones en igualdad de condiciones, la teoría más simple tiene más probabilidades de ser correcta**. A modo de ejemplo, ¿Qué variables diferencian a ambos modelos? Veamos:

```r
# ¿Que parametros tiene el modelo 1 que no presente el modelo 6?
Reduce(setdiff, strsplit(c(as.character(estadisticas.modelos[1]$`AIC-both`$call)[2],
                           as.character(estadisticas.modelos[6]$`SBC-backward`$call)[2]), split = " "))
```

```
## [1] "ActividadPpal"                    "WomanPopulationPtge"
## [3] "logxConstructionUnemploymentPtge" "Explotaciones"
```

La principal diferencia entre ambos modelos es el número de parámetros: **la diferencia entre el modelo 1 y el modelo 6 es de cuatro variables (desglosando la ActividadPpal, 5 variables)**. Es decir, empleando menos parámetros, el modelo 6 **es capaz de obtener un mejor SBC y una desviación típica**. Por ello, de cara a la comparación con los modelos aleatorios escogemos el modelo 6.

2.3 Selección de variables aleatoria

Una vez escogido, realizamos una selección de variables aleatoria, comparando los resultados con el modelo anterior:

```
## Modelo aleatorio  1
## Train:  0.71425 ; Test:  0.7136172 ; Dif. (Train-Test):  0.0006327691 ; AIC: 49211.98 ; SBC:  49367.89
## Numero de variables:  22
## Modelo aleatorio  2
## Train:  0.7243206 ; Test:  0.7213802 ; Dif. (Train-Test):  0.002940464 ; AIC: 48978.91 ; SBC:  49134.82
## Numero de variables:  22
## Modelo aleatorio  3
## Train:  0.7135244 ; Test:  0.7094126 ; Dif. (Train-Test):  0.00411189 ; AIC: 49236.45 ; SBC:  49419.48
## Numero de variables:  26
```

```
##     modelo     media         sd
## 1 Modelo 1 0.7222440 0.01340825
## 2 Modelo 2 0.7120494 0.01391400
## 3 Modelo 3 0.7216570 0.01339689
## 4 Modelo 4 0.7107515 0.01420169
```

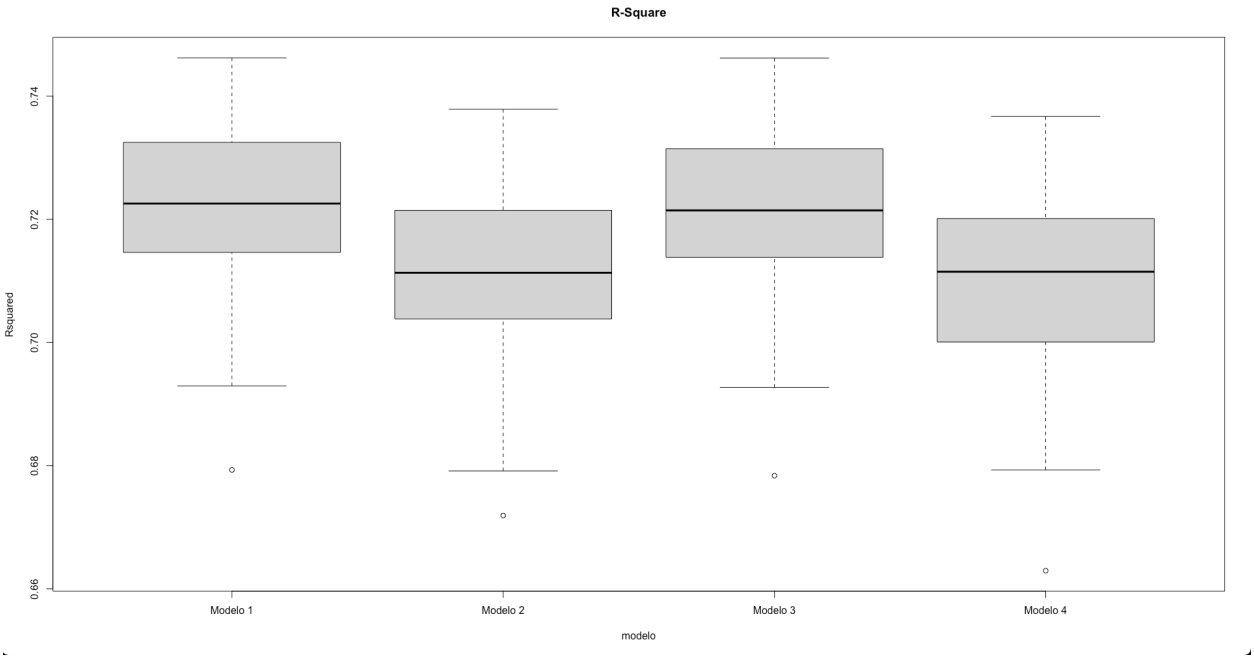


Figure 2: Validación cruzada en el modelo de selección aleatoria + modelo 6

2.4 Selección del modelo ganador

Nuevamente, las desviaciones típicas en los modelos aleatorios son mayores con respecto al modelo 6 salvo en el segundo caso, donde es ligeramente menor (0.01339689 frente a 0.01340825 obtenido en el modelo 6). Sin embargo, y pese a que el principio de parsimonia nos lleve a elegir el segundo modelo aleatorio, **el modelo 6 obtenido en la selección clásica, pese a tener una variable adicional, refleja un mejor resultado en todos los sentidos, tanto en AIC como SBC, además de que la diferencia entre ambas desviaciones típicas no es demasiado grande**:

```
Modelo aleatorio 2 => (Train-Test):  0.002940464 ; AIC: 48978.91 ; SBC:  49134.82 ; sd: 0.01391400
Modelo clasico 6   => (Train-Test):  0.002180476 ; AIC: 48966.11 ; SBC:  49128.8  ; sd: 0.01340825
```

Por tanto, elegimos como modelo ganador al modelo 6. A continuación, evaluamos su resultado:

ESTADÍSTICAS DEL MODELO:

```
# Evaluamos las estadisticas del modelo ganador
mostrar.estadisticas(modelo.final, data_train, data_test, "lm", "varObjCont")
```

```
## Train:  0.7249481 ; Test:  0.7227676 ; Dif. (Train-Test):  0.002180476 ; AIC: 48966.11 ; SBC:  49128.8
## Numero de variables:  23
```

```
estadisticas.modelos.final[6,]
```

```
##     modelo    media         sd
## 6 Modelo 6 0.722244 0.01340825
```

2.5 Interpretación de los coeficientes de dos variables

A modo de ejemplo, interpretamos los coeficientes de dos variables incluidas en el modelo:

```
##
## Call:
## lm(formula = as.formula(formula.final), data = data_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -54.401  -6.010   0.090   6.413  44.633
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                 55.77179    2.84195  19.624
## CCAAAN_NA                                   10.50797    4.01811   2.615
## CCAAAR_CM                                    2.61079    2.83661   0.920
## CCAACAT_PV                                  19.28835    3.93165   4.906
## CCAACV_EX_AS_BA_CA                           8.71309    3.33822   2.610
## CCAAMA_CA_RI_CE_ME_MU_GA                    14.14547    3.88530   3.641
## Age_19_65_pct                               -0.20096    0.02236  -8.988
## SameComAutonPtge                             0.20010    0.02804   7.135
## prop_missings                                0.06889    0.01793   3.843
## logxForeignersPtge                          -0.01564    0.07755  -0.202
## logxIndustryUnemploymentPtge                -0.20842    0.03878  -5.375
## logxServicesUnemploymentPtge                -0.34993    0.05794  -6.040
## logxtotalEmpresas                           -0.37271    0.05935  -6.280
## CCAAAN_NA:SameComAutonPtge                  -0.36278    0.04775  -7.597
## CCAAAR_CM:SameComAutonPtge                  -0.12646    0.03471  -3.643
## CCAACAT_PV:SameComAutonPtge                 -0.82285    0.04839 -17.004
## CCAACV_EX_AS_BA_CA:SameComAutonPtge         -0.26626    0.04077  -6.531
## CCAAMA_CA_RI_CE_ME_MU_GA:SameComAutonPtge   -0.21978    0.04775  -4.603
## CCAAAN_NA:logxForeignersPtge                 1.53400    0.23239   6.601
## CCAAAR_CM:logxForeignersPtge                 0.40798    0.12431   3.282
## CCAACAT_PV:logxForeignersPtge               -0.65428    0.32043  -2.042
## CCAACV_EX_AS_BA_CA:logxForeignersPtge        0.63594    0.22306   2.851
## CCAAMA_CA_RI_CE_ME_MU_GA:logxForeignersPtge  0.25172    0.24858   1.013
```

```
##                                          Pr(>|t|)
## (Intercept)                               < 2e-16 ***
## CCAAAN_NA                                0.008939 **
## CCAAAR_CM                                0.357403
## CCAACAT_PV                               9.53e-07 ***
## CCAACV_EX_AS_BA_CA                       0.009072 **
## CCAAMA_CA_RI_CE_ME_MU_GA                 0.000274 ***
## Age_19_65_pct                             < 2e-16 ***
## SameComAutonPtge                         1.07e-12 ***
## prop_missings                            0.000123 ***
## logxForeignersPtge                       0.840185
## logxIndustryUnemploymentPtge             7.94e-08 ***
## logxServicesUnemploymentPtge             1.63e-09 ***
## logxtotalEmpresas                        3.61e-10 ***
## CCAAAN_NA:SameComAutonPtge               3.46e-14 ***
## CCAAAR_CM:SameComAutonPtge               0.000272 ***
## CCAACAT_PV:SameComAutonPtge               < 2e-16 ***
## CCAACV_EX_AS_BA_CA:SameComAutonPtge      7.03e-11 ***
## CCAAMA_CA_RI_CE_ME_MU_GA:SameComAutonPtge 4.24e-06 ***
## CCAAAN_NA:logxForeignersPtge             4.41e-11 ***
## CCAAAR_CM:logxForeignersPtge             0.001037 **
## CCAACAT_PV:logxForeignersPtge            0.041201 *
## CCAACV_EX_AS_BA_CA:logxForeignersPtge    0.004372 **
## CCAAMA_CA_RI_CE_ME_MU_GA:logxForeignersPtge 0.311278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.47 on 6473 degrees of freedom
## Multiple R-squared:  0.7249, Adjusted R-squared:  0.724
## F-statistic: 775.5 on 22 and 6473 DF,  p-value: < 2.2e-16
```

1. **CCAAAN__NA** (Andalucía y Navarra): 10.50. Es decir, **el porcentaje de votos a la derecha aumenta en un 10.50 % si la Comunidad Autónoma a la que pertenece el municipio es Andalucía o Navarra, con respecto a la CCAA de referencia (Castilla y León)**.
2. **logxServicesUnemploymentPtge**: -0.34. Es decir, **por cada incremento unitario en el porcentaje de votos a la Derecha, el porcentaje de parados en el sector Servicios (en escala logarítimica) se ve reducido en -0.34 unidades**

En última instancia, realizamos un análisis de la importancia de las variables del modelo final:

```
## lm(formula = as.formula(formula.final), data = data_train)
##
## Coefficients
##                                    SSR df pEta-sqr dR-sqr
## (Intercept)                  42180.9434  1   0.0562     NA
## CCAA                          4139.2617  5   0.0058 0.0016
## Age_19_65_pct                 8848.2877  1   0.0123 0.0034
## SameComAutonPtge              5576.0420  1   0.0078 0.0022
## prop_missings                 1617.2920  1   0.0023 0.0006
## logxForeignersPtge               4.4543  1   0.0000 0.0000
## logxIndustryUnemploymentPtge  3163.9506  1   0.0044 0.0012
## logxServicesUnemploymentPtge  3995.2732  1   0.0056 0.0016
## logxtotalEmpresas             4319.5381  1   0.0061 0.0017
## CCAA:SameComAutonPtge        35943.0996  5   0.0483 0.0139
## CCAA:logxForeignersPtge       6502.1247  5   0.0091 0.0025
##
## Sum of squared errors (SSE): 708964.9
## Sum of squared total  (SST): 2577567.7
```

En este caso, nos encontramos con que las variables con mayor con dos variables que apenas aportan valor al R2:

logxForeignersPtge y **prop_missings**. No obstante, la eliminación de alguna de ellas no aporta mejoría al modelo. En el caso de **logxForeignersPtge**, al eliminarlo supondría que el efecto del porcentaje de extranjeros sobre el porcentaje de votos a la derecha dependiense únicamente de las interacciones de dicha variable con las CCAA. Sin embargo, la interacción entre Castilla y León (la cual se empleaba previamente como categoría de referencia), tampoco aporta información al modelo. Por otro lado, aunque el campo **prop_missings** suponga una pérdida de tan solo el 0.006 en el R2, al eliminarlo tanto el valor de AIC como SBC aumentan con respecto al modelo anterior. De hecho, eliminando **prop_missings** obtenemos justamente el segundo modelo aleatorio (22 variables), lo que significa que dicha variable **pone de manifiesto que el efecto de la proporción de valores *missings* es significativo**.

```
# AIC anterior: 48966.11 ---- SBC anterior: 49128.8
Train:  0.7243206 ; Test:  0.7213802 ; Dif. (Train-Test):  0.002940464 ; AIC: 48978.91 ; SBC:  49134.82
Numero de variables:  22
```

Además, el p-valor obtenido en el *summary* asegura la importancia de dicha variable al 95 % de confianza, por lo que no la eliminaremos. Por último, cabe destacar el p-valor obtenido en algunas variables en las que interviene el campo CCAA.

3. Construcción del modelo de regresion logística

3.1. Elaboración del modelo inicial

A continuación, una vez elaborado el modelo de regresión lineal nos centramos en la regresión logística. En primer lugar, y al igual que en el apartado anterior, **ejecutaremos un primer modelo con todas las variables y todas sus transformaciones**, aunque no sea el modelo definitivo:

```
modelo1.bin<-glm(formInt.bin,data=data_train.bin, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
mostrar.estadisticas(modelo1.bin, data_train.bin, data_test.bin, "glm", "varObjBin")
```

```
## Train:  0.4836808 ; Test:  0.2885679 ; Dif. (Train-Test):  0.1951128 ; AIC: 4898.755 ; SBC:  6417.238
## Numero de variables:  224
```

Analizando un primer modelo, llama la atención el mensaje de advertencia que devuelve R: *fitted probabilities numerically 0 or 1 occurred*, esto es, **el modelo está prediciendo probabilidades absolutas (0,1)**. Esto último puede deberse, principalmente, a un exceso en el número de parámetros, lo que se traduce en una quasi-separación, es decir, mientras que un pequeño conjunto de las variables es capaz de predecir perfectamente la variable objetivo, el resto (probablemente por la menor importancia que presentan o por la poca influencia hacia la variable objetivo) les resultará mucho más difícil, teniendo en muchas ocasiones coeficientes tan grandes o tan pequeños como los que se muestran a continuación:

```
summary(summary(modelo1.bin)$coefficients[, 4]) # Desde -10555.99 hasta 47767.97
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0001006 0.2438835 0.5630272 0.5257464 0.8339119 0.9996603
```

Dicho de otro modo, **sobran variables**, en especial interacciones. Por tanto, debemos conservar únicamente aquellas interacciones que aportan "significancia" al modelo. Para ello, mediante la función *impVariablesLog* filtramos aquellas con mayor relevancia con respecto a la variable dicotómica. Una primera aproximación sería **recategorizar alguna variable cualitativa**, concretamente las CCAA. Para ello, emplearemos nuevamente la función *woebin*, el cual permite permite generar la unión óptima de las variables, concretamente de las CCAA:

```
## [1] "AA_CL"
## [2] "AN_NA"
## [3] "AR_CM"
## [4] "CAT_PV"
## [5] "CV_EX_AS_BA_CA%,%MA_CA_RI_CE_ME_MU_GA"
```

Concretamente, el criterio *woe* nos indica una posible unión entre *CV_EX_AS_BA_CA* con *MA_CA_RI_CE_ME_MU_GA*. Siguiendo su recomendación, realicemos una pequeña prueba con el fin de comprobar si mejora el modelo:

```
input_bin_copia$CCAA <- recode(input_bin_copia$CCAA, "c('CV_EX_AS_BA_CA','MA_CA_RI_CE_ME_MU_GA') =
                       'CV_EX_AS_BA_CA_MA_CA_RI_CE_ME_MU_GA_CM'")
```

```
## Train:  0.4676321 ; Test:  0.2810729 ; Dif. (Train-Test):  0.1865592 ; AIC: 4989.097 ; SBC:  6344.885
## Numero de variables:  200
```

Con respecto al modelo inicial, bien es cierto que la diferencia entre el train y el test se ve reducido ligeramente, así como el valor SBC No obstante, y del mismo modo que ocurría con la regresión lineal, la disminución no es muy significativa, del mismo modo que aumenta además el error AIC. Por otra parte, si recategorizamos ambas comunidades puede ocurrir que estemos perdiendo información relevante al unir ambas categorías, por lo que mantedremos el modelo inicial.

```
# importancia.var <- impVariablesLog(modelo1.bin, "varObjBin", data_train.bin)
summary(importancia.var$V5)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 5.224e-06 1.096e-04 3.358e-04 5.588e-04 8.550e-04 2.685e-03
```

Volviendo con el *summary*, y a diferencia del modelo lineal, el tercer cuartil nos indica que **un 75 % de las interacciones presentan una importancia con respecto al R2 de 0.0008 o menos, un valor significativamente mayor que en relación al tercer cuartil del modelo lineal**. Como consecuencia, en lugar de filtrar únicamente aquellas interacciones con valores atípicos (*outliers*), dado que el tercer cuartil es mucho mayor **filtramos aquellas interacciones a partir de 0.0008 en adelante**, es decir, nos quedamos con el 25 % de las interacciones más relevantes (junto con las columnas originales):

```
variables.mas.imp <- importancia.var[which(importancia.var$V5 > 8.550e-04), "V2"]
formInt.bin <- paste0("varObjBin~",paste0(colnames(input_bin)[-1], collapse = "+"),"+",
                      paste0(unlist(variables.mas.imp), collapse = "+"))
modelo1.2.bin<-glm(formInt.bin,data=data_train.bin, family = binomial)
mostrar.estadisticas(modelo1.2.bin, data_train.bin, data_test.bin, "glm", "varObjBin")
```

```
## Train:  0.4634901 ; Test:  0.3443494 ; Dif. (Train-Test):  0.1191407 ; AIC: 4822.802 ; SBC:  5493.917
## Numero de variables:  99
```

Analizando las estadísticas obtenidas, comprobamos que el modelo mejora prácticamente en todos los aspectos, desde la diferencia entre el valor R2 train-test, hasta los valores AIC y SBC, tan solo reduciendo el número de parámetros, sin recategorizar ninguna variable. No obstante, siguen siendo demasiados parámetros, por lo que este último modelo servirá como punto de partida para la selección clásica.

3.2. Selección de variables clásica

```
estadisticas.modelos.bin <- seleccion.clasica(formInt.bin, data_train.bin, data_test.bin, "glm")
```

```
##              R.2.train  R.2.test  Diferencia      AIC      SBC N.Parametros
## AIC-both     0.4556352 0.4436611 0.011974103 4852.512 5394.827           80
## SBC-both     0.4186636 0.4142750 0.004388596 5049.213 5178.013           19
## AIC-forward  0.4556352 0.4436611 0.011974103 4852.512 5394.827           80
## SBC-forward  0.4186636 0.4142750 0.004388596 5049.213 5178.013           19
## AIC-backward 0.4632562 0.3452324 0.118023873 4818.818 5469.596           96
## SBC-backward 0.4400553 0.4334371 0.006618201 4906.813 5177.971           40
```

Una vez calculada la selección clásica, realizamos la validación cruzada:

```
estadisticas.modelos.final.bin # Mostramos la media y desviacion tipica obtenida en la validacion cruzada
```

```
##     modelo     media          sd
## 1 Modelo 1 0.8895489 0.008087545
## 2 Modelo 2 0.8834345 0.008597503
## 3 Modelo 3 0.8883197 0.008204140
```

Dado que los modelos 1 y 3, así como el 2 y 4 son iguales, además de que el modelo 5 presenta un elevado número de outliers que dificulta la lectura de los diagramas de caja y bigotes, se ha decidido mostrar únicamente el modelo 1, 2 y 6. Analizando los resultados obtenidos, podemos observar como el modelo 1, pese a ser el que menor desviación típica presenta, el número de parámetros continúa siendo elevado; mientras que por el contrario los modelos 2 y 6 presentan un menor número de variables. La pregunta es, ¿Cuál de ellos es mejor modelo? Según el diagrama, la anchura de la caja en el modelo 6 es menor, lo que indica una menor dispersión de los valores ROC (menor desviación típica). No obstante, pese a que los valores de AIC y SBC sean ligeramente menores en el modelo 6 (4906 y 5177,

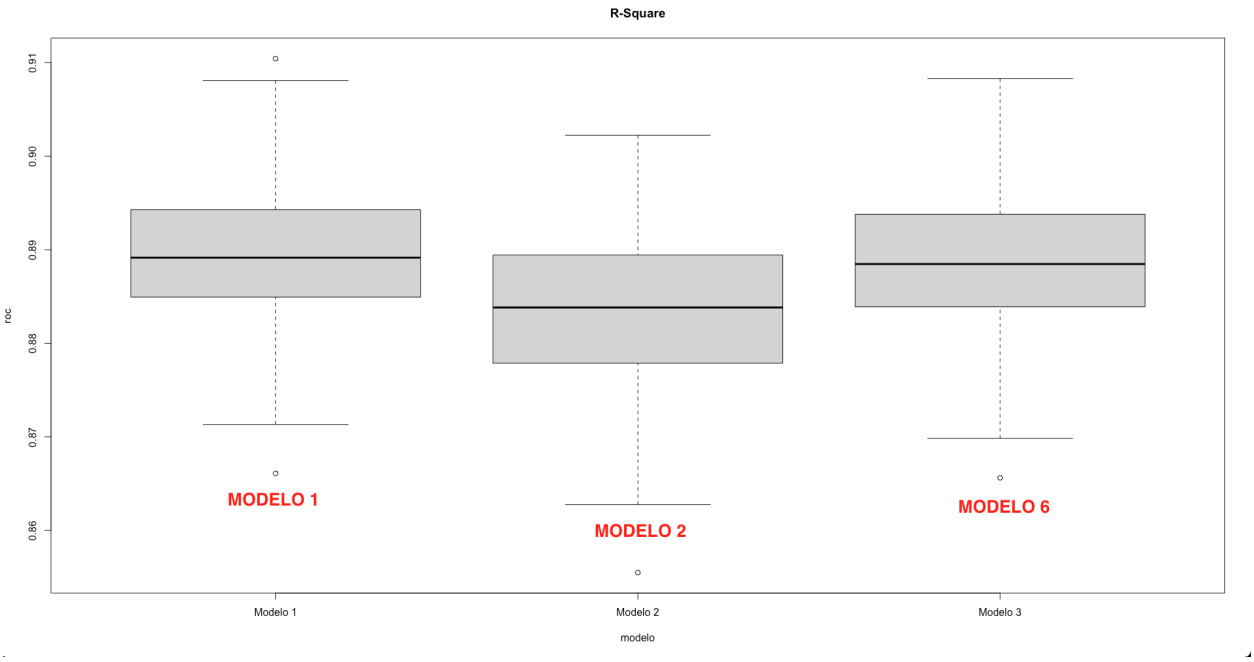


Figure 3: Validación cruzada en el modelo de selección clásica

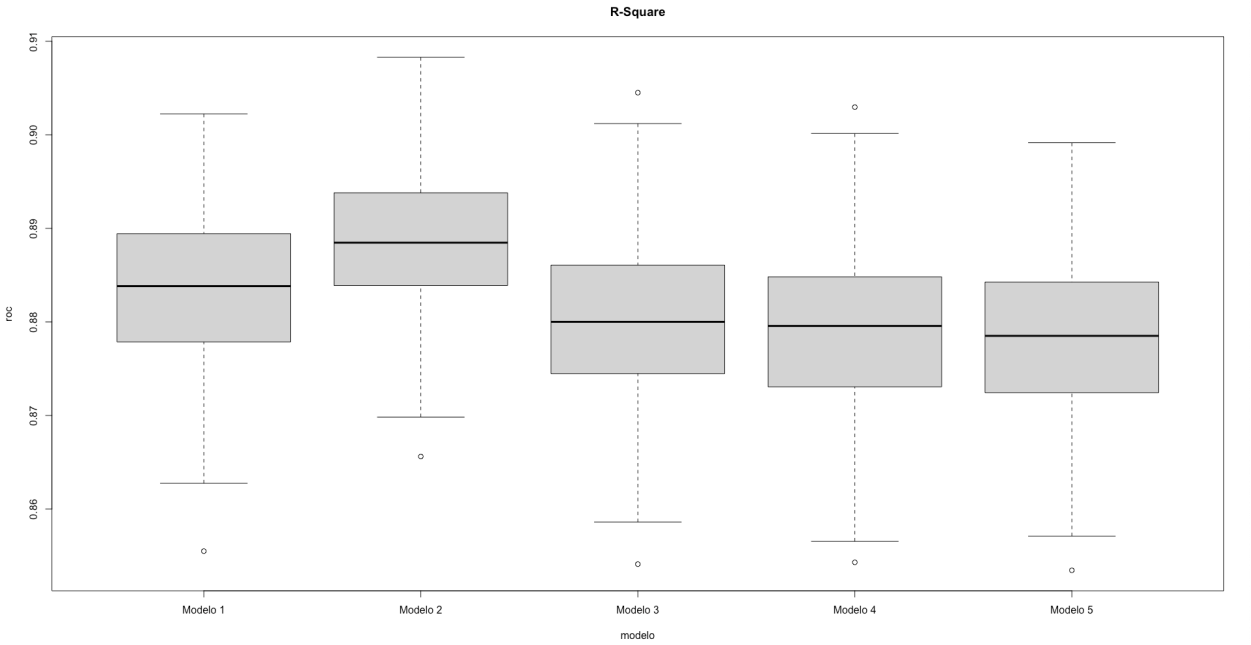
respectivamente), bien es cierto que la diferencia no es muy significativa (143 puntos en AIC y apenas 1 punto en SBC), además de que la diferencia entre ambas desviaciones típicas no es muy elevada: 0.0085 en el modelo 2 y 0.0082 en el modelo 6, es decir, de tan solo 0.0003, incluso la diferencia entre las medias ROC de ambos modelos es de tan solo 0.05, teniendo un pseudo-R2 mayor en el modelo 6 (0.44 frente a 0.41). Además, la diferencia en el número de parámetros supone una gran diferencia: 40 frente a 13 coeficientes.

Por tanto, a simple vista (y pese a las estadísticas favorables al modelo 6) no está tan claro cual es el modelo vencedor. Por ello, de cara a la selección aleatoria comparamos ambos modelos con los obtenidos aleatoriamente.

3.3. Selección de variables aleatoria

```
## Train:  0.4103582 ; Test:  0.4045364 ; Dif. (Train-Test):  0.005821815 ; AIC: 5106.807 ; SBC:  5188.155
## Numero de variables:  12
## Train:  0.4088446 ; Test:  0.404895 ; Dif. (Train-Test):  0.003949538 ; AIC: 5117.855 ; SBC:  5192.423
## Numero de variables:  11
## Train:  0.4091012 ; Test:  0.4087142 ; Dif. (Train-Test):  0.0003869665 ; AIC: 5115.643 ; SBC:  5190.211
## Numero de variables:  11
```

```
##     modelo     media          sd
## 1 Modelo 1 0.8834345 0.008597503
## 2 Modelo 2 0.8883197 0.008204140
## 3 Modelo 3 0.8800970 0.008927627
## 4 Modelo 4 0.8793138 0.009081880
## 5 Modelo 5 0.8782888 0.008972804
```



3.4. Selección del modelo ganador

Analizando las salidas de los modelos aleatorios, tanto en AIC como SBC, ninguno de ellos aporta una mejoría en el modelo de regresión logística, donde el AIC y el SBC no disminuyen de los 5000 puntos, además de unas desviaciones típicas ligeramente superiores con respecto a los modelos 2 y 6 de la selección clásica. Por tanto, ¿Qué modelo debemos elegir, el modelo 2 o el modelo 6? Para ello, y dado que el modelo 6 presentaba un elevado número de coeficientes, veamos si podemos reducir dicho número, **filtrando de *impVariablesLog* aquellos coeficientes cuya importancia con respecto al R2 sea menor a 0.002**:

```
##                             V2          V5
## 1                   Population 0.001019342
## 2                PobChange_pct 0.001123748
## 3                   SUPERFICIE 0.001228155
## 4               Age_over65_pct 0.001610978
## 5 AgricultureUnemploymentPtge 0.001912597
```

Una vez eliminadas dichas variables, evaluemos nuevamente ambos modelos:

MODELO 2:

```
## Train:  0.4186636 ; Test:  0.414275 ; Dif. (Train-Test):  0.004388596 ; AIC: 5049.213 ; SBC:  5178.013
## Numero de variables:  19
```

MODELO 6:

```
## Train:  0.4335186 ; Test:  0.4332371 ; Dif. (Train-Test):  0.0002815281 ; AIC: 4953.16 ; SBC:  5190.423
## Numero de variables:  35
```

Con respecto al modelo 1 original, eliminando las variables anteriores el modelo no mejora con respecto al AIC y SBC, aunque la diferencia entre el train y test se reduce ligeramente. Analicemos las desviaciones típicas:

```
##     modelo     media          sd
## 1 Modelo 1 0.8834345 0.008597503
## 2 Modelo 2 0.8861644 0.008043842
```

La desviación típica, por el contrario, parece disminuir aunque no lo suficientes. Por tanto, haber reducido el número de variables no aclara cuál es mejor modelo. Por tanto, nos queda una última opción: **analizar el p-valor de las variables de cada modelo**. De este modo podremos comprobar la distribución del p-valor de los coeficientes de cada uno:

MODELO 2:

```r
summary(summary(modelo.final.bin)$coefficients[, 4])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000000 0.0000086 0.0338068 0.0010103 0.3607090
```

MODELO 6:

```r
summary(summary(modelo.final.bin.2)$coefficients[, 4])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0001302 0.0088571 0.1279645 0.1291336 0.8345302
```

Analizando los p-valores de cada modelo, podemos comprobar como, en el modelo 2, el 75 % de sus coeficientes presentan un p-valor de 0.001 o menos (3er cuartil), mientras que en el modelo 6 sólo el 50 % de las variables está por debajo de 0.008. Esto último supone que muchas de las variables (especialmente en las interacciones) no son significativas en el modelo 6, mientras que en el modelo 2 si lo son mayoritariamente. Por tanto, siguiendo no solo el principio de parsimonia (menor número de parámetros), sino además la importancia general de los coeficientes, **elegimos como modelo ganador el modelo 6**.

JUSTIFICACIÓN:

3.5. Punto de corte óptimo

Una vez elegido el modelo ganador, debemos determinar el punto de corte más óptimo. Para ello, generamos una rejilla con posibles puntos de corte. Sobre dicha rejilla buscaremos tanto el índice de que **maximice la tasa de aciertos** y el **índice de Youden**:

```r
rejilla$posiblesCortes[which.max(rejilla$Youden)]
```

```
## [1] 0.57
```

```r
rejilla$posiblesCortes[which.max(rejilla$Accuracy)]
```

```
## [1] 0.5
```

Obtenemos unos valores de índices bastante similares. No obstante, comparamos ambos puntos de corte:

```
## INDICE YOUDEN
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.8361060      0.9156746      0.7056911      0.8360507      0.8362235
```

```
## INDICE ACCURACY
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.8391867      0.9474206      0.6617886      0.8211522      0.8847826
```

Analizando las medidas de evaluación obtenidas, podemos observar como el índice de Youden presenta una sensitividad o tasa de verdaderos positivos del 91 % aproximadamente. Por otro lado, la especificidad mediante el índice de Youden es significativamente mayor con respecto al índice que maximiza la tasa de aciertos (70 % frente a un 66 %). El objetivo del modelo de regresión no solo es maximizar la tasa de verdaderos positivos, sino que además el modelo sea capaz de acertar, en la mejor medida posible, la tasa de verdaderos negativos, es decir, ser capaz de acertar no solo aquellos municipios con un predominio de los votos a la Derecha, sino también ser capaz de distinguir aquellos municipios con mayor predominio de la izquierda. Por tanto, **con el propósito de maximizar tanto la sensitividad como la especifidad**, sacrificamos el 94 % de máxima especificidad del segundo índice escogiendo el índice de Youden.

3.6. Interpretación de los coeficientes de dos variables

A continuación, interpretamos los coeficientes de dos variables incluidas en el modelo, tanto cuantitativa como cualitativa:

```
##
## Call:
## glm(formula = formula.final.bin, family = binomial, data = data_train.bin)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4994  -0.1696   0.3656   0.6340   3.4198
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          1.403e+00  2.006e-01   6.993 2.69e-12 ***
## CCAAAN_NA                           -3.221e+00  1.534e-01 -21.000  < 2e-16 ***
## CCAAAR_CM                           -1.743e+00  1.233e-01 -14.135  < 2e-16 ***
## CCAACAT_PV                          -7.164e+00  5.170e-01 -13.857  < 2e-16 ***
## CCAACV_EX_AS_BA_CA                  -1.831e+00  1.431e-01 -12.791  < 2e-16 ***
## CCAAMA_CA_RI_CE_ME_MU_GA            -7.395e-01  2.153e-01  -3.434 0.000594 ***
## ForeignersPtge                       5.489e-02  6.320e-03   8.685  < 2e-16 ***
## AgricultureUnemploymentPtge         -1.643e-02  3.692e-03  -4.450 8.58e-06 ***
## prop_missings                        3.094e-02  4.863e-03   6.362 1.99e-10 ***
## Age_over65_pct                       2.919e-02  4.606e-03   6.337 2.34e-10 ***
## PobChange_pct                        1.598e-02  4.552e-03   3.512 0.000445 ***
## SUPERFICIE                          -2.629e-05  6.754e-06  -3.892 9.93e-05 ***
## Population                          -1.364e-04  7.144e-05  -1.909 0.056289 .
## SameComAutonDiffProvPtge             3.182e-02  9.978e-03   3.189 0.001426 **
## CCAAAN_NA:Population                 9.955e-05  8.012e-05   1.243 0.214042
## CCAAAR_CM:Population                 4.145e-04  9.287e-05   4.464 8.06e-06 ***
## CCAACAT_PV:Population               -5.201e-04  5.690e-04  -0.914 0.360709
## CCAACV_EX_AS_BA_CA:Population        2.065e-04  7.873e-05   2.624 0.008700 **
## CCAAMA_CA_RI_CE_ME_MU_GA:Population  6.670e-04  1.500e-04   4.448 8.67e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8620.2  on 6495  degrees of freedom
## Residual deviance: 5011.2  on 6477  degrees of freedom
## AIC: 5049.2
##
## Number of Fisher Scoring iterations: 9
```

1. **CCAACAT_PV** (Cataluña y País Vasco): -7.1. Es decir, con un coeficiente negativo obtenemos la inversa del ODD Ratio $(1/e^{7.1}) =$ 8e-04, **por lo que diremos que el ODD de una mayoría de votos a la derecha si la comunidad autónoma es Cataluña o País Vasco es 8e-04 veces mayor que el ODD de una mayoría de votos a la derecha si la comunidad autónoma correspondiese con Castilla y León (categoría de referencia).** Por tanto, existe una muy alta probabilidad de que un municipio perteneciente a Cataluña o País Vasco tenga una mayoría de votos hacia la izquierda en comparación con municipios de Castilla y León.
2. **Age_over65_pct**: 0.02919. Es decir, **por cada incremento unitario en el porcentaje de población mayor a 65 años, la ODD de una mayoría de votos a la derecha aumenta en 0.02919 unidades**. Por tanto, podemos decir que el aumento en el porcentaje de población superior a 65 años es directamente proporción al aumento de la probabilidad de que en dicho municipio resulte ganador la derecha.

```
# Evaluamos las estadísticas del modelo ganador
mostrar.estadisticas(modelo.final.bin, data_train.bin, data_test.bin, "glm", "varObjBin")
```

```
## Train:  0.4186636 ; Test:  0.414275 ; Dif. (Train-Test):  0.004388596 ; AIC: 5049.213 ; SBC:  5178.013
## Numero de variables:  19
```

En última instancia, realizamos un análisis de la importancia de las variables del modelo final:

```
impVariablesLog(modelo.final.bin, "varObjBin", data_train.bin)
```

```
##                          V2          V5
```

```
## 1    SameComAutonDiffProvPtge 0.001228123
## 2               PobChange_pct 0.001448536
## 3                  SUPERFICIE 0.001750155
## 4 AgricultureUnemploymentPtge 0.002260586
## 5              Age_over65_pct 0.004801142
## 6               prop_missings 0.004882347
## 7             CCAA:Population 0.006088821
## 8              ForeignersPtge 0.009592236
```

Podemos comprobar cómo las variables más influyentes en el modelo son *ForeignersPtge*, la interacción entre las CCAA y *Population*, así como *prop_missings*. No obstante, también nos encontramos con variables cuya importancia en el modelo es mucho menor, variables como *SameComAutonDiffProvPtge*, *PobChange_pct* y *SUPERFICIE*, ¿Quizás sobren estas tres últimas? Si las eliminamos del modelo, obtendríamos el siguiente resultado:

```
Train:  0.4142889 ; Test:  0.4116508 ; Dif. (Train-Test):  0.002638034 ; AIC: 5080.924 ; SBC:  5189.387
Numero de variables:  16
```

Pese a que la diferencia del Pseudo-R2 entre los valores de train y test se ve reducido, los valores de AIC y SBC aumentan en ambos casos. Por tanto, lo más conveniente será mantener dichas variables en el modelo. Por otro lado, si analizamos el área bajo la curva (AUC), vemos que se sitúa en torno al 88 % con porcentajes muy similares tanto en los datos de entrenamiento como de prueba:

```
## Area under the curve: 0.8852
```

```
## Area under the curve: 0.8825
```

A continuación, analizamos las tasas obtenidas con el punto de corte tanto en los datos de entrenamiento como de prueba:

```
## TRAIN:
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.8371305      0.9137546      0.7114994      0.8385263      0.8342068
```

```
## TEST:
```

```
##       Accuracy    Sensitivity    Specificity Pos Pred Value Neg Pred Value
##      0.8361060      0.9156746      0.7056911      0.8360507      0.8362235
```

Podemos comprobar cómo las medidas de clasificación son similares entre los datos de entrenamiento y prueba, salvo ciertos casos como en la tasa de sensitividad o de valor predictivo negativo donde el porcentaje es levemente mayor en los datos de prueba que en los datos de entrenamiento. Por lo general, con el índice de Youden hemos conseguido no sólo acertar 9 de cada 10 municipios como verdaderos positivos, sino además ser capaz de distinguir (a un 71-70 %) verdaderos negativos.