Parte I

# EVALUACIÓN MINERÍA DE DATOS Y MODELIZACIÓN PREDICTIVA

María Yacobi Quilez

14 Abril, 2020

# 1.INTRODUCCIÓN

Se cuenta con una base de datos formada por diferentes variables demográficas y económicas y resultados de las elecciones en municipios concretos. Con esta información, se busca predecir dos cosas: por un lado, el porcentaje votos a partidos de derechas que se dará en un municipio a través de la variable porcentaje de derechas (Dcha_Pct), y por otro, si habrá una abstención alta en las elecciones en cada municipio a través de la variable dicotómica de abstención alta (AbstencionAlta), que toma valor 1 si la abstención es superior al 30% o 0 si es inferior al 30%.

# 2.DEPURACIÓN DE DATOS

Empezaremos primero importando los datos para inspeccionar si se ha asignado correctamente el tipo de cada variable. Cargo las librerías necesarias[1] para la depuración de datos, así como las funciones y los datos.

```
datos<-read_xlsx("/Users/mariayacobi/Desktop/Master_Clases/Mineria_de_datos_apuntes/Mineria_de_datos_ejercicios/DatosEleccionesEspana.xlsx")
```

Antes de nada, se procede a eliminar las variables que no se van a utilizar. Como solo nos quedamos con porcentaje de derechas y abstención alta, elimino porcentaje de abstención, porcentaje de izquierda, porcentaje otros, izquierda y derecha. También elimino *Name*, ya que es el identificador y no aporta información al modelo.[2]

```
datos<-subset(datos,select = -c(AbstentionPtge,Izda_Pct,Otros_Pct,Izquierda,Derecha,Name))
```

Tras ver más detenidamente los porcentajes de edad, veo que la variable *Age_under19_Ptge* ya incluye a todos entre 0 y 19, así que no necesito *Age_0-4_Ptge*. No me aporta más información de la que ya tengo así que la elimino también.

```
datos<-datos[,-(7)]
```

A primera vista y tras inspeccionar la asignación de variables tengo claro que la variable *AbstenciónAlta* tiene que transformarse en dicotómica y que las variables *Densidad* y *ActividadPpal* tienen que ser factores. Sin embargo voy a confirmar que no se me escapa nada mirando el número de valores distintos de cada variable numérica.[3]

```
sapply(Filter(is.numeric,datos),function(x) length(unique(x)))
```

Veo que en principio ninguna de las variables clasificadas como numéricas actualmente tiene pocos valores únicos por lo que de momento se quedan así. Sin embargo, en el caso de *CodProvincia* y *CCAA*, creo que es necesario transformarlas en factor, ya que el código representa una categoría: a qué provincia pertenece el municipio. Además, para poder hacer esto utilizaremos un método de tramificación para reducir el número de factores.

[1] Esto se ha incluido en el documento RMD como código de *setup*, por eso no aparece aquí.

[2] Ver Anexo A.1

[3] Ver Anexo A.2 para ver la salida del código

En consecuencia: solo tengo que cambiar a factor las variables *AbstencionAlta*, *Densidad*, *Actividad Principal*, *CCAA* y *CodProvincia*. Además, tras ver resúmenes estadísticos básicos de cada variable con la función summary[4], obtengo pistas de cosas que puedo depurar en este proceso.

Por un lado, veo que algunas variables como las de desempleo tienen de valor máximo 100%, lo cual es extraño. Estos podrían ser outliers. Otras variables que pueden tener outliers porque parecen tener una distribución con mucha cola son las de número de empresas, población o censo y explotaciones. También veo valores faltantes en muchos casos (declarados y no declarados) y valores fuera de rango en la variable *SameComAutonPtge.*

```
summary(datos)
```

A continuación profundizo en la exporación inicial de las variables con la función describe del paquete psych.[5]

```
psych::describe(Filter(is.numeric,datos))
```

Al observar la asimetría de cada variable, vemos que hay algunas variables extremadamente asimétricas, como población, censo, total de empresas (y todas las variables de número de empresas por sector) y número de inmuebles. Realmente casi todas muestran algo de asimetría aunque las mencionadas anteriormente son las que más asimetría presentan. Además, vemos muchos valores elevados de la medida de curtosis, especialmente en las variables que ya había mencionado anteriormente que sospechaba que podían tener bastante cola (población, censo, total de empresas y por tipo, etc). Posiblemente aprendamos más de lo que ocurre al estudiar la presencia de outliers en estas variables. Además, como ya habíamos visto con la función summary, hay datos faltantes.

```
any(is.na(datos))

## [1] TRUE
```

En conclusion, vemos bastantes cosas que se pueden arreglar, datos faltantes, variables mal codificadas y outliers que inspeccionaremos más adelante.

## 2.2 Recodificación de tipos de variables, recategorización y tratamiendo de valores fuera de rango

Comenzamos creando un árbol de decisión para tramificar CCAA de cara a su respuesta frente a la variable objetivo continua y la variable objetivo binaria. Crearé dos árboles: uno para cada variable objetivo ya que los grupos pueden no comportarse de la misma forma para ambas variables.

```
varObjCont<-datos$Dcha_Pct
varObjBin<-datos$AbstencionAlta
tree<-rpart(varObjCont~CCAA, data=datos)
rpart.plot(tree, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```

Como se puede ver en el gráfico[6], se han creado cuatro grupos. A continuación veré los resultados de agrupar las CCAA en estos cuatro grupos a través de la diferencia entre medias de cada grupo.

[4] Ver Anexo A.4

[5] Ver Anexo A.5

[6] Ver Anexo A.5

```
datos$CCAA<-factor(datos$CCAA)
da<-cbind(var=datos$CCAA,tree=tree$where,obj=varObjCont)
```

```
aggregate(var~tree, data=da, mean)

##   tree       var
## 1    2 10.885643
## 2    4  7.677843
## 3    6  8.133117
## 4    7  7.004889
```

```
aggregate(obj~tree, data=da, mean)

##   tree      obj
## 1    2 14.29624
## 2    4 43.94544
## 3    6 55.64049
## 4    7 64.85848
```

Efectivamente, vemos que hay diferencias entre las medias de votos a porcentajes de derecha en los cuatro grupos por lo que cambio los niveles de la variable. Además voy a mantener la variable CCAA y creo una nueva llamada CCAACont para repetir el proceso para la variable binaria. Para poder ver qué CCAA está en qué grupo, veo tree y observo que un grupo (2) es Cataluña y País Vasco, otro (6) es Andalucía, Asturias, Baleares, Canarias, Valencia, Extremadura y Navarra, otro (14) es Aragón, Cantabria, Castilla la Mancha, Galicia, Madrid, Murcia y Rioja y un último grupo (15) que son CastillaLeón, Ceuta y Melilla. A continuación recodifico la variable.

```
datos$CCAACont<-factor(tree$where)
levels(datos$CCAACont)[levels(datos$CCAACont)=="2"] <- "Cat-PVasco"
levels(datos$CCAACont)[levels(datos$CCAACont)=="4"] <- "And-CVal-Ext-Bal-Cana-Ast-Nav"
levels(datos$CCAACont)[levels(datos$CCAACont)=="6"] <- "CMancha-Gal-Arag-Rioja-Mad-Mur-Cant"
levels(datos$CCAACont)[levels(datos$CCAACont)=="7"] <- "CLeon-Ceuta-Melilla"
```

Ahora voy a hacer lo mismo para mi variable objetivo binaria y CCAA para usarla más adelante en la regresión logística.

```
tree2<-rpart(varObjBin~CCAA, data=datos)
rpart.plot(tree2, box.palette="RdBu", shadow.col="gray", nn=TRUE)

datos$CCAA<-factor(datos$CCAA)
da2<-cbind(var=datos$CCAA,tree=tree2$where,obj=varObjBin)
```

```
aggregate(var~tree, data=da2, mean)

##   tree      var
## 1    2 7.940680
## 2    4 8.693789
## 3    5 8.028765
```

```
aggregate(obj~tree, data=da2, mean)

##   tree       obj
## 1    2 0.1541205
## 2    4 0.4757764
## 3    5 0.7961083
```

Vemos que se han creado tres grupos[7]. Ahora factorizo y le cambio los niveles

```
datos$CCAABin<-factor(tree2$where)
levels(datos$CCAABin)[levels(datos$CCAABin)=="2"] <- "Arag-Cant-CLeon-CMancha-CVal-Ext-Mad-Mur-Rioja"
levels(datos$CCAABin)[levels(datos$CCAABin)=="4"] <- "And-Gal-Nav-PVasco"
levels(datos$CCAABin)[levels(datos$CCAABin)=="5"] <- "Ast-Bal-Cana-Cat-Ceuta-Melilla"
```

A continuación, voy a hacer exactamente lo mismo para la variable *CodigoProvicia*

```
tree3<-rpart(varObjCont~CodigoProvincia, data=datos)
rpart.plot(tree3, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```

No me merece la pena incluir la variable *CodigoProvincia*, ya que lo mejor que ha podido hacer el árbol es reducir a 22 grupos[8] que siguen siendo demasiados… Además, en cierto sentido *CCAA* y

[7] Ver gráfico en el anexo A.6

[8] Ver gráfico en el anexo A.7

*CodProvincia* están relacionadas, ya que la provincia es una subcategoría de *CCAA*. La elimino de mi dataset.

```
datos<-subset(datos,select = -c(CodigoProvincia))
```

Una vez que tengo estas variables recategorizadas y convertidas a factor, puedo continuar con el resto de variables en las que he encontrado errores de tipo de variable. Empezamos por convertir a factor aquellas variables cuyo tipo estaba mal codificado. Concretamente, estas son: *AbstenciónAlta*, *Densidad* y *ActividadPpal*.

```
datos[,c(4,26,30)]<-lapply(datos[,c(4,26,30)],factor)
```

Ya hemos reconvertido estas variables a factor. Además, veo que tengo algún *missing* no declarado en Densidad. Cambio estos valores a declarados.

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

## Recoded 92 values to NA.
```

Al hacer el summary anteriormente, también he encontrado algunos otros errores, como valores fuera de rango, valores missing declarados y outliers. Empiezo ahora por arreglar los valores fuera de rango. Para empezar, vemos que *SameComAutonPtge* tiene un valor de 127%, lo cual no debería ser. Además, *ForeignersPtge* tiene también un valor fuera de rango: un número negativo que en este caso no debería ser.

```
datos$ForeignersPtge<-replace(datos$ForeignersPtge,which(datos$ForeignersPtge<0),NA)
datos$SameComAutonPtge<-replace(datos$SameComAutonPtge,which(datos$SameComAutonPtge>100),NA)
```

## 2.3 Estudio de outliers

Antes de seguir, separo las variables objetivo de las input

```
varObjCont<-datos$Dcha_Pct
varObjBin<-datos$AbstencionAlta
inputdatos<-as.data.frame(datos[,-(4:5)])
```

Calculo primero el porcentaje de atípicos de cada variable. Veo muchas con un porcentaje muy bajo y otras que habría que examinar, aunque parece que ninguna supera el 11% de la variable servicios, por lo que los podemos gestionar transformando esos outliers en missings.[9]

```
sapply(Filter(is.numeric, inputdatos),function(x) atipicosAmissing(x)[[2]])/nrow(inputdatos)

inputdatos[,as.vector(which(sapply(inputdatos, class)=="numeric"))]<-sapply(Filter(is.numeric, inputdatos),function(x)
atipicosAmissing(x)[[1]])
sum(is.na(inputdatos))

## [1] 9972
```

He dado varias vueltas a la decisión de quitar outliers e imputarlos. En este caso, son outliers reales: el municipio de Madrid frente a pequeños municipios. Sé que imputando estos outliers en mi modelo de predicción, cuando tenga una caso real de un municipio como Madrid con 3 millones de habitantes, el modelo no va a poder predecirlo de una forma tan correcta. Pero como la mayoría de casos son municipios con mucha menor población, prefiero tener una buena capacidad de predicción para la mayoría de situaciones que se dan en este caso concreto.

[9] Ver Anexo A.8

## 2.3 Tratamiento de missings

A continuación realizaré un tratamiento de los valores faltantes en el dataset. Para ello, comienzo buscando un patrón en los missings.

```
corrplot(cor(is.na(inputdatos[colnames(inputdatos)[colSums(is.na(inputdatos))>0]])),method = "ellipse",type = "upper")
```

Vemos[10] que hay algo de correlación positiva (en muchos casos, una alta correlación positiva) entre los valores missing de las variables población y total de empresas (y población y cada una de los variables de número de empresas por sector), así como entre censo y total de empresas (y censo y cada una de las variables de número de empresas por sector) y entre las variables de número de empresas por sector. A continuación, veremos la proporción de missings por variable y observación[11].

```
inputdatos$prop_missings<-apply(is.na(inputdatos),1,mean)
(prop_missingsVars<-apply(is.na(inputdatos),2,mean))
```

En este caso vemos unas cuantas variables que rondan o superan el 10% de observaciones missing, aunque ninguna super el 50% por lo que no debo eliminar nada. En el caso de las observaciones, el valor máximo es del 35.2% por lo que tampoco ejecuto el código para eliminar observaciones. Como no hemos eliminado variables ni observaciones, a continuación vamos a imputarlas utilizando el método aleatorio y reviso que no queden missings.[12]

```
any(is.na(inputdatos))

## [1] FALSE
```

Guardo los datos depurados.

```
saveRDS(cbind(varObjBin,varObjCont,inputdatos),"datosEleccionesDep")
```

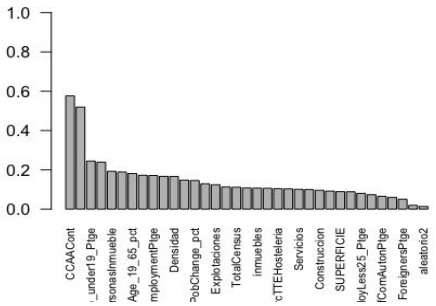
# 3.EXPLORACIÓN VISUAL DE LOS DATOS

Antes de nada, quito CCAA, ya que ya tengo mis dos variables de CCAA, una para cada variable objetivo y vuelvo a separar las target de las input. Genero dos variables aleatorias para ayudarme a discernir entre las variables que realmente tienen una relación con la target y las que no.

```
EleccionesDep<-readRDS("datosEleccionesDep")
EleccionesDep<-EleccionesDep[,-(3)]

EleccionesObjCont<-EleccionesDep$varObjCont
EleccionesObjBin<-EleccionesDep$varObjBin
inputElecciones<-EleccionesDep[,-(1:2)]
inputElecciones$aleatorio1<-runif(nrow(inputElecciones))
inputElecciones$aleatorio2<-runif(nrow(inputElecciones))
```

Ahora que ya tengo estos pasos previos completados, comienzo con el análisis descriptivo de las variables, empezando por gráficos que representan la V de cramer para todas las variables input contra las targets (tanto la binaria como la continua).

```
graficoVcramer(inputElecciones,EleccionesObjCont)
```




[10] Ver Anexo A.9 para ver el gráfico de correlación

[11] Ver Anexo A.10

[12] Ver anexo A. 11 para ver el código

Bajo este criterio podemos observar que las 5 variables más importantes que explican la variable Dcha_Ptge son:

1. Comunidad autónoma (la pensada para continua)
2. Porcentaje de menores de 19 años
3. Porcentaje de mayores de 65 años
4. Número medio de personas por inmueble
5. Tasa de desempleo para mayores de 40 años

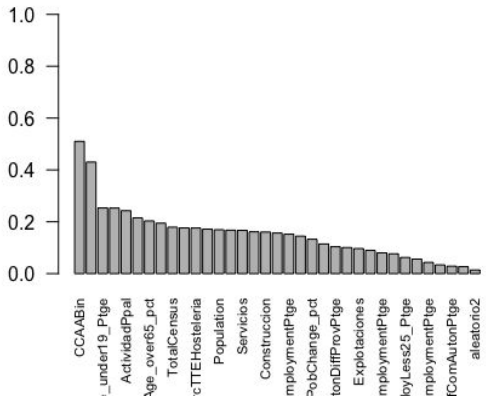
Otras que pueden tener algo de importancia son aquellas relacionadas con el porcentaje de desempleo (en el sector servicios y en el tramo de edad entre 25 y 40) así como la actividad principal y la densidad de población.

```
graficoVcramer(inputElecciones,EleccionesObjBin)
```

En cuanto a la variable binaria, bajo este criterio vemos que pueden tener importancia las siguientes cinco variables:

1. Comunidad autónoma (la pensada para binaria)
2. Porcentaje de menores de 19 años
3. Actividad principal
4. Porcentaje de habitantes mayores de 65 años
5. Número de empresas en el sector industrial.

Además, en este caso vemos que las relacionadas con el desempleo no parecen tener tanta importancia como en el caso de la target continua.




## 2.1. Exploración de relación de variables predictoras frente a variable objetivo binaria

Hacemos gráficos de mosaico para ver como responde cada categoría frente a la targ. binaria.

```
par(mfrow=c(1,2))
mosaico_targetbinaria(inputElecciones$CCAABin,EleccionesObjBin,"CCAA") #Solo hago en este caso con CCAABin ya que estoy mirando cómo influyen frente a la binaria. Sí influye
mosaico_targetbinaria(inputElecciones$Densidad,EleccionesObjBin,"Densidad de población") #Esta parece que también influye: vemos menos abstención a medida que aumenta la densidad de población

mosaico_targetbinaria(inputElecciones$ActividadPpal,EleccionesObjBin,"Actividad Principal")#También parece que tenga influencia. En aquellos municipios con un sector u otro de actividad principal podemos esperar ver una respuesta diferente.
```

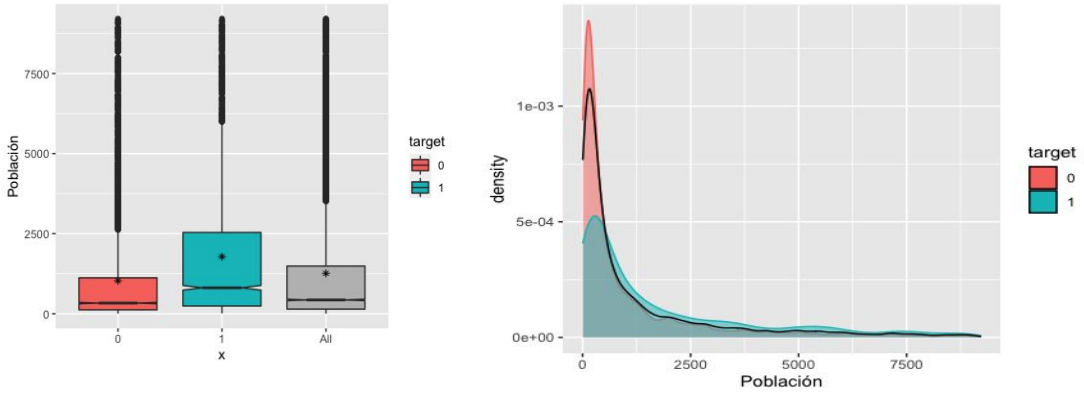
A continuación hago lo mismo pero con gráficos de barras.

```
barras_targetbinaria(inputElecciones$ActividadPpal,EleccionesObjBin,"Act.Ppal") #Bastante diferencia en la distribución de categorías de actividad principal y abstención alta

barras_targetbinaria(inputElecciones$Densidad,EleccionesObjBin,"Densidad") #En este caso no me queda tan claro.Es verdad que vemos que en municipios de densidad muy baja hay más de la mitad de los municipios tienen abstención alta y en densidad Alta al revés pero la distribución parece estar más equilibrada que en el caso anterior.

barras_targetbinaria(inputElecciones$CCAABin,EleccionesObjBin,"CCAA") #En este caso sí que veo bastante diferencia.
```

Ahora voy a analizar visualmente el comportamiento de las variables numéricas frente a la variable objetivo binaria mediante boxplots e histogramas. Dejo tan solo un ejemplo del código y el gráfico

aquí para cada uno de los gráficos porque son muchos y todos iguales. Se ha realizado el mismo proceso para cada una de las variables continuas. Más abajo las conclusiones recogidas.

```
boxplot_targetbinaria(inputElecciones$Population,EleccionesObjBin,"Población")

hist_targetbinaria(inputElecciones$Population,EleccionesObjBin,"Población")
```




En población no veo una diferencia significativa en el comportamiento de la variable y abstención alta, así como en censo. En el caso de porcentaje de menores de 19 años, no es muy marcada la diferencia, pero cuando el porcentaje de población en ese tramo de edad se sitúa entre 0 y 15, hay menor incidencia de abstención alta, mientras que entre 15 y 25 hay mayor incidencia de abstención alta.

En el caso de porcentaje entre 19 y 65 sí vemos que entre el 55 y el 65 % hay mayor incidencia de abstención alta. También en el caso de porcentaje de mayores de 65 vemos que entre un 15 y un 25% hay mayor incidencia de abstención alta y del 25 al 60% menor incidencia de abstención alta.

En el caso de porcentaje de mujeres no veo una diferencia significativa. En porcentaje de extranjeros tampoco, ni en tasa de desempleo en menores de 25.En general, en ninguna variable de desempleo por tramos de edad hay diferencias. Tampoco se ven diferencias de comportamiento en las que expresan el porcentaje de habitantes que provienen de otra CCAA u otra provincia en la misma CCAA. Tampoco ninguna diferencia en variables de desempleo por sector ni en las variables de numero total de empresas y por sector (numero total de empresas). En el resto ninguna diferencia.

## 2.2. Exploración de relación de variables predictoras frente a variable objetivo continua

A continuación, estudio visualmente la relación entre variables input y la variable objetivo continua. Empiezo por ver las numéricas frente a la target continua. Luego categóricas frente la target continua.

```
corrplot(cor(cbind(EleccionesObjCont,Filter(is.numeric, inputElecciones)), use="pairwise", method="pearson"), method =
"number",type = "upper",tl.cex = 0.3,number.cex = 0.5)
```

Vemos[13] que ninguna variable tiene una gran correlación con el porcentaje de votos a partidos de derecha, aunque de mayor a menor correlación tenemos: (apuntes cuaderno) - hacer tabla en anexo

[13] Ver Anexo B.3 para el gráfico de correlación

Además, vemos que en varios casos hay una muy alta correlación entre predictores. Para ello vuelvo a aplicar el gráfico de correlación pero esta vez agrupando por grupos de correlación.[14]

```
corMat <- cor(Filter(is.numeric, inputElecciones))
corrplot(corMat, order = "hclust", tl.cex = 0.7)
```

Busco exactamente estas variables (las que tengan un grado de correlación superior a 0.8) y las elimino.

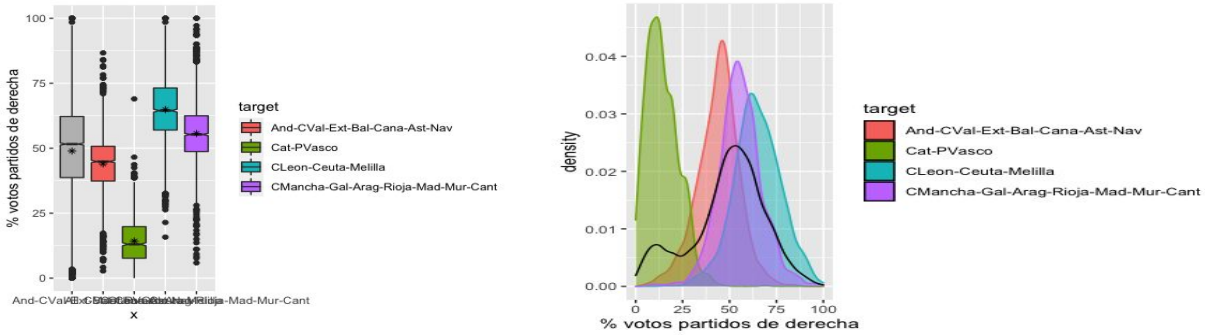
```
highlyCor <- colnames(inputElecciones)[findCorrelation(corMat, cutoff = 0.8, verbose = TRUE)]

input_cor <- inputElecciones[, which(!colnames(inputElecciones) %in% highlyCor)]
ncol(input_cor)

## [1] 31
```

No estoy del todo satisfecha con como me ha gestionado la colinealidad esto[15]. Las variables que más me preocupan son censo-población, los tramos de edades y las variables que indican el porcentaje de habitantes que son de la misma provincia, misma CCAA y diferente provincia y diferente CCAA. Me gusta que haya quitado población, porque me quedo con censo y porcentaje de cambio de población y pob 2010 que me da una imagen completa, pero con edad, me ha dejado solo una: porcentaje de 19 a 65 años, la cual es la única de las tres de tramos de edad que no aparece en las cinco más importantes para las target en mis gráficos V de cramer. Por eso, manualmente voy a quitar población, porcentaje entre 19 y 65 y una de las de Provincia y CCAA (en estos dos últimos casos me quedo con dos variables de tres, ya que solo es necesario quitar una para librarnos de la colinealidad).

```
input_cor<-inputElecciones[,-c(1,4,8)]
```

Seguimos con las variables categóricas input frente a la target continua. He decidido usar las funciones de boxplot e histograma creadas originalmente para ver visualmente la target binaria frente a los predictores categóricos porque el principio es el mismo, cambiando de orden las variables: pongo la continua en el eje x y la categórica n el eje y.

```
boxplot_targetbinaria(EleccionesObjCont,inputElecciones$CCAACont,"% votos partidos de derecha") #mucha diferencia de comportamineto

hist_targetbinaria(EleccionesObjCont,inputElecciones$CCAACont,"% votos partidos de derecha")#Muchas diferencias
```

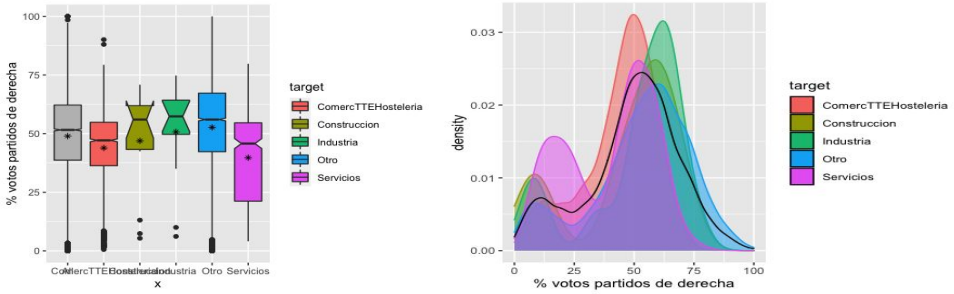



```
boxplot_targetbinaria(EleccionesObjCont,inputElecciones$ActividadPpal,"% votos partidos de derecha")
```

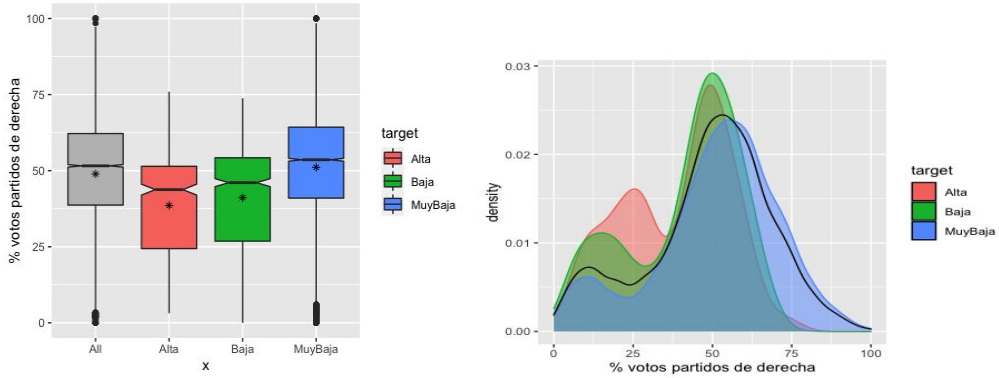
[14] Ver Anexo B.4 para el gráfico de correlación

[15] Ver anexo B.5

```
hist_targetbinaria(EleccionesObjCont,inputElecciones$ActividadPpal,"% votos partidos de derecha")
```




```
boxplot_targetbinaria(EleccionesObjCont,inputElecciones$Densidad,"% votos partidos de derecha")

hist_targetbinaria(EleccionesObjCont,inputElecciones$Densidad,"% votos partidos de derecha")
```




## 4.TRANSFORMACIÓN DE VARIABLES

Antes de comenzar la regresión voy a aplicar transformaciones a mis variables para ver si alguna de ellas tiene una relación no lineal con la target. Primero quito la variable proporción de *missings* que no me sirve para mucho.

```
input_cor<- input_cor[,-(30)]
input_cont<-cbind(input_cor,Transf_Auto(Filter(is.numeric, input_cor),EleccionesObjCont))
sapply(Filter(is.numeric, input_cor)[,-ncol(Filter(is.numeric, input_cor))],function(x) length(unique(x)))

input_bin<-cbind(input_cor,Transf_Auto(Filter(is.numeric, input_cor),EleccionesObjBin))
```

Guardo mis datos para utilizarlos en la regresión lineal y logística

```
saveRDS(data.frame(input_bin,EleccionesObjBin),"todo_bin_e")
saveRDS(data.frame(input_cont,EleccionesObjCont),"todo_cont_e")
```

## 5.REGRESIÓN LINEAL

Para hacer la regresión lineal, primero voy a hacer varios modelos manuales con el método forward hasta seleccionar el que mejor funcione de los mejores. A continuación, utilizaré métodos de selección automática de variables con el método stepwise y luego analizaré el mejor modelo obtenido a través de validación cruzada de los mejores modelos. Cargo los datos depurados. Además, le quito la variable CCAABin que era específica para la regresión logística

```
datoscont<-readRDS("todo_cont_e")
datoscont<-subset(datoscont,select = -c(CCAABin)) #Estos datos los usaré para la generación automática de modelos. En la
```

```
parte manual, uso input_cor que contiene los datos originales.
datosorig <- data.frame(EleccionesObjCont,input_cor[,-(29)])

set.seed(12345678)
trainIndex <- createDataPartition(datosorig$EleccionesObjCont, p=0.8, list=FALSE)
data_train <- datosorig[trainIndex,]
data_test <- datosorig[-trainIndex,]
```

## 5.1. Método Manual

Empiezo el método manual de selección de variables con el modelo completo. Después introduciré poco a poco las que parezcan explicar más el modelo según VCramer.

```
modelomanual1<-lm(EleccionesObjCont~.,data=data_train)
summary(modelomanual1)
```

```
Rsq(modelomanual1,"EleccionesObjCont",data_train)

## $r2
## [1] 0.7009629
##
## $r2_adj
## [1] 0.6992496
```

```
Rsq(modelomanual1,"EleccionesObjCont",data_test)

## $r2
## [1] 0.6968516
##
## $r2_adj
## [1] 0.6897749
```

En este modelo[16] no veo una mala $R^2$ pero hay muchísimas variables que parecen tener un efecto poco significativo en la predicción. Por ello, ahora voy a meter solo las top 5 según VdeCramer, a ver qué tal sale.

```
modelomanual2<-lm(EleccionesObjCont~CCAACont+Age_over65_pct+Age_under19_Ptge+UnemployMore40_Ptge+PersonasInmueble,data=da
ta_train)
summary(modelomanual2)
```

```
Rsq(modelomanual2,"EleccionesObjCont",data_train)

## $r2
## [1] 0.6832073
##
## $r2_adj
## [1] 0.6828166
```

```
Rsq(modelomanual2,"EleccionesObjCont",data_test)

## $r2
## [1] 0.681936
##
## $r2_adj
## [1] 0.6803594
```

Obtengo peores resultados[17]. Aunque estas variables me salían como importantes en la exploración de datos, en el primer modelo completo muchas de ellas no eran significativas, como por ejemplo, porcentaje de menores de 19 años, y no me salían en el top 5 algunas que sí parecían significativas en el modelo completo. Voy a hacer un modelo teniendo en cuenta solo estas que considero significativas según el modelo completo.

```
modelomanual3<-lm(EleccionesObjCont~CCAACont+Age_over65_pct+ForeignersPtge+AgricultureUnemploymentPtge+IndustryUnemployme
ntPtge+ServicesUnemploymentPtge+ActividadPpal+Densidad,data=data_train)
summary(modelomanual3) #En este modelo todas son bastante significativas.
```

[16] Ver resumen en anexo C.1

[17] Ver resumen en anexo C.2

```
Rsq(modelomanual3,"EleccionesObjCont",data_train)

## $r2
## [1] 0.6983701
##
## $r2_adj
## [1] 0.6976719
```

```
Rsq(modelomanual3,"EleccionesObjCont",data_test)

## $r2
## [1] 0.6954684
##
## $r2_adj
## [1] 0.6926258
```

No hay[18] mucha diferencia entre train y test en mi $R^2$ y esta es un poco mejor que en el modelo anterior. Como son muchas variables y cuesta a veces ver el efecto que tiene meter y quitarlas del modelo, voy a proceder a realizar una selección automática de variables y luego testearé los mejores modelos (de la parte manual, me quedo con el último, que tiene buen $R^2$ y no tantas variables como el modelo completo).

Vuelvo a hacer la partición test para usar todo_cont_e que tiene las transformaciones también y así lo hago todo del tirón. He decidido hacer la selección automática con el método stepwise mirando siempre tanto el criterio de akaike como el criterio de información bayesiano para elegir el mejor modelo que me saque del mejor de cada método.

```
set.seed(12345678)
trainIndex <- createDataPartition(datoscont$EleccionesObjCont, p=0.8, list=FALSE)
data_train <- datoscont[trainIndex,]
data_test <- datoscont[-trainIndex,]
```

## 5.2. Selección automática de variables: sin transformaciones ni interacciones

Empezamos con una selección de variables con las variables originales: ninguna transformación ni interacción.

```
null<-lm(EleccionesObjCont~1, data=data_train)
full<-lm(EleccionesObjCont~., data=data_train[,c(1:30,58)])
```

| ModeloStepAIC | ModeloStepBIC |
|---|---|
| `modeloStepAIC<-step(null, scope=list(lower=null, upper=full), direction="both")`<br><br>`summary(modeloStepAIC)` | `modeloStepBIC<-step(null, scope=list(lower=null, upper=full), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modeloStepBIC)` |
| `Rsq(modeloStepAIC,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.6973234`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.6931616` | `Rsq(modeloStepBIC,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.6949415`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.6924768` |
| `modeloStepAIC$rank`<br><br>`## [1] 22` | `modeloStepBIC$rank`<br><br>`## [1] 13` |

[18] Ver resumen en anexo C.3

El modeloStepBIC[19] tiene peor $R^2$ pero lo prefiero porque la diferencia en $R^2$ no es tan grande como para asumir 10 parámetros de más en el modeloStepAIC.

## 5.3. Selección automática de variables: con interacciones

```
formInt<-formulaInteracciones(datoscont[,c(1:30,58)],31)
fullInt<-lm(formInt,data = data_train)
```

| ModeloStepAIC_int | ModeloStepBIC_int |
|---|---|
| `modeloStepAIC_int<-step(null, scope=list(lower=null, upper=fullInt), direction="both")`<br><br>`summary(modeloStepAIC_int)` | `modeloStepBIC_int<-step(null, scope=list(lower=null, upper=fullInt), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modeloStepBIC_int)` |
| `Rsq(modeloStepAIC_int,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.593015`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.567695` | `Rsq(modeloStepBIC_int,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.7056839`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.7018234` |
| `modeloStepAIC_int$rank`<br><br>`## [1] 95` | `modeloStepBIC_int$rank`<br><br>`## [1] 21` |

ModeloStepAIC_int tiene[20] un muy buen $R^2$ en el summary pero hay bastante diferencia con test. El $R^2$ es un pelín[21] peor que en modeloStepBIC_int y sigue habiendo diferencia con test, pero lo prefiero porque tiene menos parámetros (muchos menos).

## 5.4. Selección automática de variables: con transformaciones

```
fullT<-lm(EleccionesObjCont~., data=data_train)
```

| ModeloStepAIC_trans | ModeloStepBIC_trans |
|---|---|
| `modeloStepAIC_trans<-step(null, scope=list(lower=null, upper=fullT), direction="both")`<br><br>`summary(modeloStepAIC_trans)` | `modeloStepBIC_trans<-step(null, scope=list(lower=null, upper=fullT), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modeloStepBIC_trans)` |
| `Rsq(modeloStepAIC_trans,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.7049441`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.6997612` | `Rsq(modeloStepBIC_trans,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.7018062`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.6986477` |

[19] Ver Anexo C.5

[20] Ver Anexo C.6

[21] Ver Anexo C.7

| | |
|---|---|
| `modeloStepAIC_trans$rank`<br><br>`## [1] 28` | `modeloStepBIC_trans$rank`<br><br>`## [1] 17` |

Tenemos[22] un $R^2$ un poco peor en modeloStepAIC_trans que con interacciones pero es más estable ya que hay menos diferencia con data_test. El $R^2$ es[23] prácticamente igual en StepBIC_trans y la diferencia con test nula. Algo mejora entonces este modelo. Prefiero modeloStepBIC_trans porque tiene menos parámetros y mayor estabilidad entre train y test.

## 5.5. Selección automática de variables: con transformaciones e interacciones

```
formIntT<-formulaInteracciones(datoscont,58)
fullIntT<-lm(formIntT, data=data_train)
```

| ModeloStepAIC_transInt | ModeloStepBIC_transInt |
|---|---|
| `modeloStepAIC_transInt<-step(null, scope=list(lower=null, upper=fullIntT), direction="both")`<br><br>`summary(modeloStepAIC_transInt)` | `modeloStepBIC_transInt<-step(null, scope=list(lower=null, upper=fullIntT), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modeloStepBIC_transInt)` |
| `Rsq(modeloStepAIC_transInt,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.6958331`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.6710941` | `Rsq(modeloStepBIC_transInt,"EleccionesObjCont",data_test)`<br><br>`## $r2`<br>`## [1] 0.7131392`<br>`##`<br>`## $r2_adj`<br>`## [1] 0.7082833` |
| `modeloStepAIC_transInt$rank`<br><br>`## [1] 122` | `modeloStepBIC_transInt$rank`<br><br>`## [1] 27` |

Vemos un $R^2$ muy alto[24] en summary en el modeloStepAIC_transInt, pero luego vemos que hay bastante diferencia con test. El $R^2$ es peor[25] en modeloSTepBIC_transInt, pero bueno, no hay mucha diferencia con test. Por el principio de parsimonia preferimos el modeloStepBIC_transInt.

## 5.6 Validación cruzada repetida[26]

Cojo los mejores modelos de cada uno de los pasos anteriores y el modelomanual3 y aplico validación cruzada repetida para elegir el mejor.

```
boxplot(Rsquared~modelo,data=total,main="R-Square")
```

| | |
|---|---|
| `aggregate(Rsquared~modelo, data = total, mean)` | `aggregate(Rsquared~modelo, data = total, sd)` |

[22] Ver Anexo C.8

[23] Ver anexo C.9

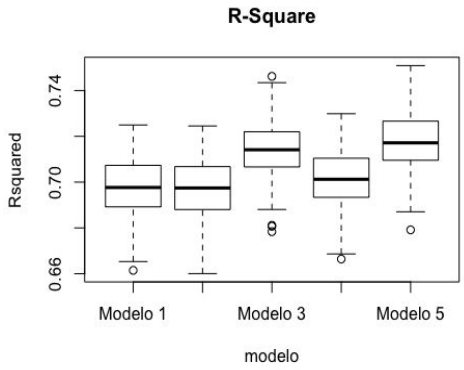
[24] Ver Anexo C.10

[25] Ver Anexo C.11

[26] Ver en Anexo C.13 para ver código de la validación cruzada repetida

```
##     modelo  Rsquared
## 1 Modelo 1 0.6971954
## 2 Modelo 2 0.6966518
## 3 Modelo 3 0.7140051
## 4 Modelo 4 0.7011693
## 5 Modelo 5 0.7171645
```

```
##     modelo   Rsquared
## 1 Modelo 1 0.01365704
## 2 Modelo 2 0.01372225
## 3 Modelo 3 0.01323422
## 4 Modelo 4 0.01362309
## 5 Modelo 5 0.01347646
```




El modelo 3 (modeloStepBIC_int) y el modelo 5 (modeloStepBIC_transInt) parecen los mejores.No hay mucha diferencia entre el $R^2$ de estos dos modelos y además, en variabilidad son parecidos también (y a todos los demás). Ahora compruebo el número de parámetros de estos dos modelos preseleccionados.

```
length(coef(modeloStepBIC_int))

## [1] 21

length(coef(modeloStepBIC_transInt))

## [1] 27
```

## 5.7 Selección del modelo ganador

El que contiene transformaciones e interacciones (modeloStepBIC_transInt) tiene siete parámetros menos. Sin embargo, el aumento de $R^2$ no es lo suficiente como para justificar el tener que interpretar variables transformadas con logaritmos, que es mucho más complicado. Me quedo con el modeloStepBIC_int. Mi argumentación en general es la siguiente: de todos los modelos generados (manuales y automáticos) me he quedado con los que tenían menos parámetros en cada caso. También he analizado la estabilidad de los modelos a través de su diferencia en $R^2$ entre los datasets de train y test. En este caso tengo algo de diferencia pero no la suficiente como para preocuparme. Además, todas las variables son bastante significativas en el modelo.

```
Rsq(modeloStepBIC_int,"EleccionesObjCont",data_train)

## $r2
## [1] 0.7159949
##
## $r2_adj
## [1] 0.7150736
```

```
Rsq(modeloStepBIC_int,"EleccionesObjCont",data_test)

## $r2
## [1] 0.7056839
##
## $r2_adj
## [1] 0.7018234
```

## 5.8 Interpretación de coeficientes del modelo ganador

Los valores de los parámetros y sus valores se encuentran en el anexo C.

Por último, para interpretar coeficientes, en este caso elijo para interpretar el coeficiente de WomanPopulationPtge y el coeficiente de CCAAContCat-PVasco.

- WomanPopulationPtge: si el porcentaje (%) de mujeres en un municipio aumenta un 1%, el porcentaje de votos a partidos de derecha disminuye un 10,7%.
- CCAAContCat-PVasco: esta es un poco más compleja porque tiene interacciones, pero en el caso de que el municipio pertenezca a Cataluña o al País Vasco, el porcentaje de votos a

partidos de derecha disminuye un 40.75%, al que añadimos el efecto que aporta la interacción entre CCAAContCat-PVasco con el porcentaje de habitantes del municipio que nacieron en una CCAA diferente a la que pertenece el municipio, que por cada aumento unitario (un 1%) del porcentaje de habitantes que nacieron en una CCAA diferente a la del municipio y el municipio está en Cataluña o PVasco. Por lo que a ese 45,75% añadimos un 21,36% por cada aumento unitario de DifComAutonPtge.

# 6.REGRESIÓN LOGÍSTICA

```
datosbin<-readRDS("todo_bin_e")

datosbin<-subset(datosbin,select = -c(CCAACont))#Quito CCAACont y me quedo con CCAABin

set.seed(12345678)
trainIndex <- createDataPartition(datosbin$EleccionesObjBin, p=0.8, list=FALSE)
data_train <- datosbin[trainIndex,]
data_test  <- datosbin[-trainIndex,]
```

Comprobamos que ambos datasets tienen la misma distribución de 1 y 0 para la target.

```
freq(data_train$EleccionesObjBin)

##      n    % val%
## 0 4473 68.9 68.9
## 1 2023 31.1 31.1
```

```
freq(data_test$EleccionesObjBin)

##      n    % val%
## 0 1118 68.9 68.9
## 1  505 31.1 31.1
```

## 6.1. Selección automática de variables: sin transformaciones ni interacciones

Como hemos visto antes, la selección automática de variables es mucho más eficiente que la manual y para poder ahorrar espacio para cosas más importantes en esta regresión logística voy a pasar directamente a la selección automática de variables. Empezamos con una selección de variables con las variables originales: ninguna transformación ni interacción.

```
null<-glm(EleccionesObjBin~1, data=data_train, family = binomial)
full<-glm(EleccionesObjBin~., data=data_train[,c(1:30,58)],family = binomial)
```

| ModelogStepAIC | ModelogStepBIC |
|---|---|
| `modelogStepAIC<-step(null, scope=list(lower=null, upper=full), direction="both")`<br><br>`summary(modelogStepAIC)` | `modelogStepBIC<-step(null, scope=list(lower=null, upper=full), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modelogStepBIC)` |
| `pseudoR2(modelogStepAIC,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2335748`<br><br>`pseudoR2(modelogStepAIC,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.2179451` | `pseudoR2(modelogStepBIC,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2279437`<br><br>`pseudoR2(modelogStepBIC,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.210966` |
| `modelogStepAIC$rank`<br><br>`## [1] 20` | `modelogStepBIC$rank`<br><br>`## [1] 13` |

Casi todas las variables son significativas[27] en el modelogStepAIC aunque el pseudo$R^2$ se podría mejorar un poco. Además, hay bastante diferencia entre train y test. El modelogStepBIC tiene alguna variable[28] menos y en este caso, todas son muy siginificativas. El Pseudo$R^2$ es peor y sigue habiendo bastante diferencia entre train y test, aunque lo prefiero porque la diferencia en pseudo$R^2$ no es tan grande como para asumir 10 parámetros de más en el modelogStepAIC.

## 6.2. Selección automática de variables: con interacciones

```
formInt<-formulaInteracciones(datosbin[,c(1:30,58)],31)
fullInt<-glm(formInt,data = data_train,family = binomial)
```

| ModelogStepAIC_int | ModelogStepBIC_int |
|---|---|
| `modelogStepAIC_int<-step(null, scope=list(lower=null, upper=fullInt), direction="both")`<br><br>`summary(modelogStepAIC_int)` | `modelogStepBIC_int<-step(null, scope=list(lower=null, upper=fullInt), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modelogStepBIC_int)` |
| `pseudoR2(modelogStepAIC_int,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2676997`<br><br>`pseudoR2(modelogStepAIC_int,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.1830163` | `pseudoR2(modelogStepBIC_int,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2315921`<br><br>`pseudoR2(modelogStepBIC_int,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.213409` |
| `modelogStepAIC_int$rank`<br><br>`## [1] 67` | `modelogStepBIC_int$rank`<br><br>`## [1] 15` |

ModelogStepAIC_int t iene un montón de variables, muchas no son significativas[29]. Además es súper poco estable, hay muchísima diferencia entre train y test. ModelogStepBIC es un poco más estable que el anterior[30].

De todas formas, está muy claro cuál es el preferible. Aunque hemos sacado un 0.26 de pseudo$R^2$ en train en el modelogStepAIC_int, no es muy estable ya ue hay mucha diferencia con test, por lo que me fío más del modelogStepBic_int que además tiene muchísimos menos parámetros..

## 6.3. Selección automática de variables: con transformaciones.

```
fullT<-glm(EleccionesObjBin~., data=data_train, family = binomial)
```

| ModelogStepAIC_trans | ModelogStepBIC_trans |
|---|---|

[27] Ver Anexo D.1

[28] Ver Anexo D.2

[29] Ver Anexo D.3

[30] Ver Anexo D.4

| | |
|---|---|
| `modelogStepAIC_trans<-step(null, scope=list(lower=null, upper=fullT), direction="both")`<br><br>`summary(modelogStepAIC_trans)` | `modelogStepBIC_trans<-step(null, scope=list(lower=null, upper=fullT), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modelogStepBIC_trans)` |
| `pseudoR2(modelogStepAIC_trans,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2389063`<br><br>`pseudoR2(modelogStepAIC_trans,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.2204209` | `pseudoR2(modelogStepBIC_trans,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2299526`<br><br>`pseudoR2(modelogStepBIC_trans,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.2148213` |
| `modelogStepAIC_trans$rank`<br><br>`## [1] 25` | `modelogStepAIC_trans$rank`<br><br>`## [1] 25` |

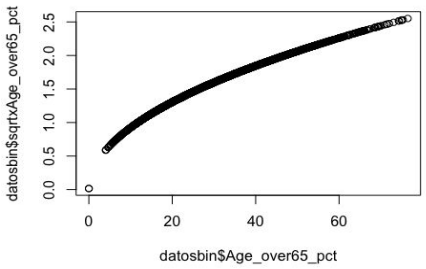
El modelo AIC con transformaciones es un poco más estable que el modelo AIC con interacciones[31]. El modelogStepBIC_trans tiene una estabilidad parecida al modelogstepAIC_trans pero pseudo$R^2$ un poco inferior[32]. Prefiero modeloStepBIC_trans porque tiene menos parámetros.

## 6.4. Selección automática de variables: con transformaciones e interacciones

```
formIntT<-formulaInteracciones(datosbin,58)
fullIntT<-glm(formIntT, data=data_train, family = binomial)
```

| ModelogStepAIC_transInt | ModelogStepBIC_transInt |
|---|---|
| `modelogStepAIC_transInt<-step(null, scope=list(lower=null, upper=fullIntT), direction="both")`<br><br>`summary(modelogStepAIC_transInt)` | `modelogStepBIC_transInt<-step(null, scope=list(lower=null, upper=fullIntT), direction="both",k=log(nrow(data_train)))`<br><br>`summary(modelogStepBIC_transInt)` |
| `pseudoR2(modelogStepAIC_transInt,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2718984`<br><br>`pseudoR2(modelogStepAIC_transInt,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.1663525` | `pseudoR2(modelogStepBIC_transInt,data_train, "EleccionesObjBin")`<br><br>`## [1] 0.2327496`<br><br>`pseudoR2(modelogStepBIC_transInt,data_test, "EleccionesObjBin")`<br><br>`## [1] 0.2155456` |

El modelogStepAIC_transInt tiene muchísimas variables[33] y muchas no parecen significativas. Además es muy inestable: hay muchísima diferencia entre train y test.




[31] Ver Anexo D.5

[32] Ver Anexo D.6

[33] Ver Anexo D.7

En el caso del modelogStepBIC_transInt vemos que es bastante más estable que el modeloStepAIC_transInt, pero tenemos un problema, y es que ha incluido en el modelo *Age_over65_pct* y *sqrtxAge_over65_pct* lo que causa multicolinealidad[34].

```
plot(datosbin$Age_over65_pct,datosbin$sqrtxAge_over65_pct)

cor(datosbin$Age_over65_pct,datosbin$sqrtxAge_over65_pct)

## [1] 0.9911631
```

Tienen una correlación prácticamente perfecta. Voy a ver cuál de ellas tiene un mayor impacto en la respuesta.

```
aggregate(Age_over65_pct ~EleccionesObjBin, data =
datosbin, mean)

##   EleccionesObjBin Age_over65_pct
## 1                0       30.66916
## 2                1       25.54475
```

```
aggregate(sqrtxAge_over65_pct ~EleccionesObjBin, data =
datosbin, mean)

##   EleccionesObjBin sqrtxAge_over65_pct
## 1                0            1.585469
## 2                1            1.441905
```

En este caso veo que las medias difieren más entre los dos subgrupos con la variable sin transformar, por lo que elimino la transformada.

```
modelogStepBIC_transInt_update<-update(modelogStepBIC_transInt, .~.-sqrtxAge_over65_pct)
summary(modelogStepBIC_transInt_update)

pseudoR2(modelogStepBIC_transInt_update,data_train, "EleccionesObjBin")

## [1] 0.2279511

pseudoR2(modelogStepBIC_transInt_update,data_test, "EleccionesObjBin")

## [1] 0.2127009
```

Al eliminar la multicolinealidad[35] ha bajado un poco la pseudoR2 pero el cambio es imperceptible.

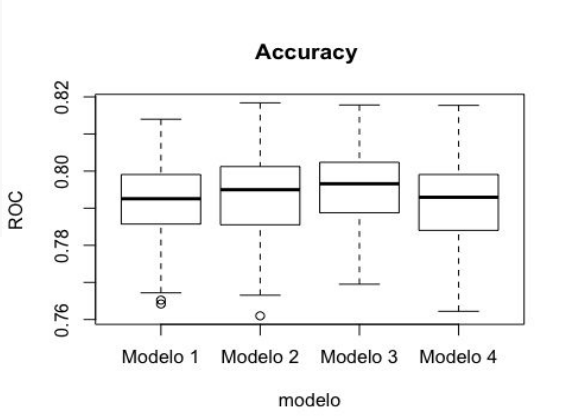
```
modelogStepAIC_transInt$rank

## [1] 77
```

```
modelogStepBIC_transInt_update$rank

## [1] 13
```

El segundo modelo es preferible por el principio de parsimonia (menos parámetros).

## 6.5 Validación cruzada repetida[36]

```
datosbin$EleccionesObjBin<-auxVarObj
boxplot(ROC ~modelo,data=total,main="Accuracy ")

aggregate(ROC ~modelo, data = total, mean)

##     modelo       ROC
## 1 Modelo 1 0.7921691
## 2 Modelo 2 0.7937817
## 3 Modelo 3 0.7954593
## 4 Modelo 4 0.7920582
```




[34] Ver Anexo D.8

[35] Ver Anexo D.9

[36] Ver Anexo D.10

```
aggregate(ROC ~modelo, data = total, sd)
```

```
##     modelo        ROC
## 1 Modelo 1 0.01125177
## 2 Modelo 2 0.01153132
## 3 Modelo 3 0.01075317
## 4 Modelo 4 0.01163319
```

Todos tienen un área debajo de la curva ROC bastante parecida aunque el modelo 4 parece ser el mejor. Aquí también vemos con la variabilidad que el modelo 4 es el que mejor la maneja aunque con poca diferencia. En cuanto a número de parámetros:

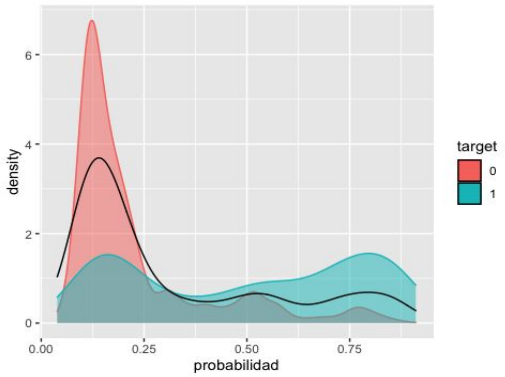
| | | | |
|---|---|---|---|
| `modelogStepBIC$rank`<br><br>`## [1] 13` | `modelogStepBIC_int$rank`<br><br>`## [1] 15` | `modelogStepBIC_trans$rank`<br><br>`## [1] 13` | `modelogStepBIC_transInt_update$rank`<br><br>`## [1] 13` |

Dada la poca diferencia en sesgo y variabilidad de los cuatro modelos seleccionados, finalmente optaré por el que menos parámetros tenga que en este caso es el modeloStepBIC sin transformaciones ni interacciones.

## 6.6 Selección del punto de corte

A continuación, busco el mejor punto de corte para este modelo seleccionado. Obtengo un gráfico de las probabilidades obtenidas:

```
hist_targetbinaria(predict(modelogStepBIC,
newdata=data_test,type="response"),data_test$EleccionesObjBin,"probabilidad")
```

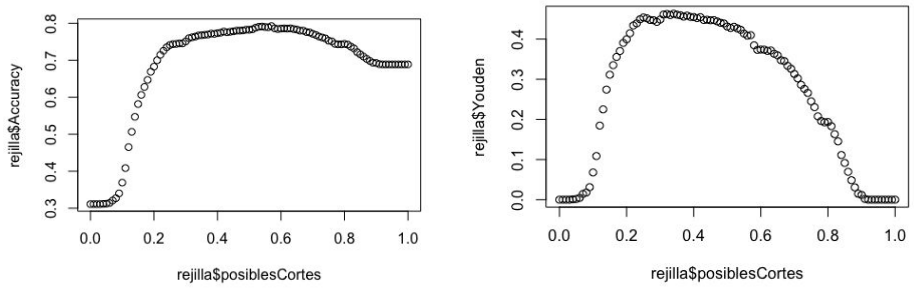



Por lo que veo en el histograma está entre 0.25 y 0.50. Parece ser más o menos 0.40. Probaré algunos.

| Código | Accuracy | Sensitivity | Specificity | Pos. Pred Value | N. Pred Value |
|---|---|---|---|---|---|
| `sensEspCorte(modelogStepBIC,data_test,"EleccionesObjBin",0.35,"1")` | 0.7677141 | 0.6316832 | 0.8291592 | 0.6254902 | 0.8328841 |
| `sensEspCorte(modelogStepBIC,data_test,"EleccionesObjBin",0.40,"1")` | 0.7732594 | 0.6059406 | 0.8488372 | 0.6442105 | 0.8266551 |

A continuación genero[37] una rejilla de posibles valores para el punto de corte:

[37] Ver Anexo D.10 para ver el código




Parece que el punto de corte que maximiza el índice de Youden está en torno al 0.3 y el que maximiza la tasa e aciertos está entre 0.4 y 0.65.

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

## [1] 0.34
```

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

## [1] 0.57
```

Vemos finalmente que estos puntos de corte son 0.34 y 0.57 respectivamente. Voy a comparar los resultados con estos dos puntos de corte y la estabilidad mediante las diferencias entre train y test.[38]

Ambos son bastante estables. De todas formas, no estoy muy satisfecha con el nivel de sensibilidad alcanzado en ninguno de los puntos de corte aunque en este caso me quedaría con el punto de corte 0.4 porque prefiero sacrificar algo de especificidad para aumentar la sensibilidad: en este caso prefiero predecir bien qué municipios van a tener abstención alta para poder actuar en consecuencia.
En conclusión, he elegido este modelo de todos los posibles porque el sesgo y la varianza se parecían bastante y este era el que menos parámetros tenía y mayor facilidad para interpretar

## 6.7 Interpretación del modelo ganador[39]

Interpretación parámetros:

- CCAABin grupo Asturias: La probabilidad de que en un municipio ubicado en Asturias, Baleares, Canarias, Ceuta o Melilla haya abstención alta es 4.44 veces mayor que en un municipio ubicado en Andalucía, Galicia, Navarra o País Vasco.
- Age_over_65_pct: por cada aumento de un punto porcentual de habitantes mayores de 65 años la probabilidad de que haya abstención alta en el municipio aumenta 1.05 veces.

[38] Ver Anexo D 11 para ver los resultados del código
[39] Ver anexo D.12 para ver los coeficientes y OR del modelo